Solid State Drives: Memory Driven Design Methodologies for optimal Performance

Lorenzo Zuolo, Cristian Zambelli, *Member, IEEE,* Rino Micheloni, *Senior Member, IEEE,* and Piero Olivo

***Abstract*—Solid State Drives (SSDs) faced an astonishing development in the last few years, becoming the cornerstone to new paradigms and markets of the Information Technology, such as cloud computing and big data centers. So far, the SSD design approach was focused on the optimization of the Flash Translation Layer, the firmware devoted to fulfill the compatibility with traditional Hard Disk Drives. For hyperscaled SSDs this strategy is no longer valid since their performance and reliability are strictly linked to that of the NAND Flash memories that constitute the storage medium, in particular when the multilevel cell paradigm is considered. For this reason the design flow must follow a bottom-up approach that, starting from an accurate knowledge of the time and use dependent reliability of the NAND Flash memories, selects the most appropriate error correction strategy to extend the SSD lifetime while reducing its performance degradation. Then the design flow moves to that of the SSD controller and of the interface towards the host where the application is running.**

This paper will thoroughly discuss this bottom-up approach and finally it will show how it is possible to leverage new approaches, such as the *software defined storage system* that, by exploiting a hardware/software co-design of the SSD controller architecture and of the host application will be able to revolutionize the traditional computer/memory interaction.

***Index Terms*—Solid State Drive, SSD, NAND Flash memories, Memory reliability, SSD design, SSD performance, Software defined flash**

I. Introduction

Solid State Drives (SSDs) are one of the electronic systems with the higher development rate in the last decade: they are widely used in hyperscale systems such as cloud computing and big data servers where performance is a constraint, as well as in consumer electronics by replacing traditional hard-disk drives (HDDs) [1].

SSDs' design, in the last 5 years, faced an extraordinary evolution caused by the continuous development of NAND Flash memories representing their storage medium [2]. With this respect, as shown in Fig. 1, NAND Flash memories have completely transformed the way information is processed and stored. Starting as film and tape replacement for cameras and voice recorders, NAND Flash memories rapidly surpassed traditional magnetic storage supports and now they represent an obliged choice for high-performance storage solutions. The availability of NAND Flash-based SSDs also materialized as an astonishing proliferation of global-scaled corporations whose commercial strength is tightly coupled to the availability of SSDs engineered for big data centers and cloud computing. The previous developing strategy of SSDs, in fact, was based on a full compatibility with HDDs and therefore the SSDs' performance optimization was focused on that of the Flash Translation Layer (FTL), the firmware managing the basic memory operations [3], [4], [5]. FTL is responsible for a plug-and-play connection between the host system where the application is running and the SSD. To this respect it must be considered that in the last 4 decades user applications have been designed to work with traditional magnetic HDDs, which are conceptually different from SSDs. Therefore, rather than redesign the whole architecture of the application, it is more convenient to leverage a command translation layer.

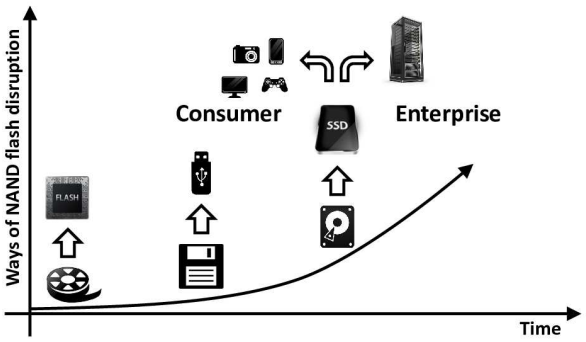


Fig. 1. Evolution of NAND Flash-based systems: from tape, film and floppy disk replacement to the explosive SSDs applications for cloud computing and big data centers.

The development of SSDs was made possible by the use of sufficiently reliable Single Level Cells (SLC) NAND Flash memories [6], storing a single bit per cell in the traditional 0/1 digital paradigm with a low read error probability, thus requiring the design of simple engines for Error Correction Codes (ECC) [7]. The SATA protocol [8] interfacing the memory system and the host was sufficient to guarantee the requested Quality of Service (QoS), that is the ability of keeping a sustained performance over time within a defined threshold [9], [10]. As a whole, the SSD architecture optimization and the development of dedicated CAD tools for the exploration of the SSD design space were FTL-oriented, in a top-down approach.

In the last few years, the need for SSDs with higher storage capacities and performance joined to the availability of high density NAND Flash memories able to store 2, 3 or even 4 bits in a single cell [11], moved the design paradigm from a Top-Down to a Bottom-Up approach where the performance

This work has been partially supported by the Università degli Studi di Ferrara through the initiative Bando per il finanziamento della ricerca scientifica "Fondo per l'Incentivazione alla Ricerca" (FIR) - 2016.

Lorenzo Zuolo and Rino Micheloni are with Microsemi, Via Torri Bianche 1, 20871 Vimercate (Italy). Work done at the Università di Ferrara (Italy).

Cristian Zambelli and Piero Olivo are with the Dipartimento di Ingegneria, Università di Ferrara, via G. Saragat,1 - 44122 Ferrara (Italy).

and the reliability of the storage medium dictate the design constraints [12]. NAND Flash memories with scaled technologies, in fact, suffer from several physical mechanisms able to impact their reliability figures such as: *i*) *Endurance*, that is the maximum number of Program/Erase (P/E) operations that the memory can withstand before leading to a failure; *ii*) *Data Retention*, denoting the ability of a memory to keep a stored information over time with no biases applied ; *iii*) the immunity from *Read Disturbs*, representing the stress suffered by a memory cell when reading neighbor cells [13], [14], [15].

In NAND Flash memories information is associated to the amount of charge present in a storage layer. P/E operations rely on charge transport through a thin oxide via Fowler-Nordheim (FN) tunneling into/from the storage layer [16]. Electron tunneling is responsible for a slow, but continuous, oxide wear out causing undesired charge flowing into/from the storage layer. As the number of P/E cycles increases, this effect strongly impacts the writing operations. To deal with endurance effects, sophisticated (but slow and power hungry) algorithms are adopted to tightly control the amount of charge transferred into/from the storage layer [17]. However, the relentless oxide degradation strongly affects the ability of keeping unaltered the charge content into the storage layer for extremely long times, a mandatory requirement to fulfill the nonvolatile paradigm. These reliability issues become more and more significant in Multi-Level Cells (MLC) [18], Triple-Level Cells (TLC) [19] and Quadruple-Level Cells (QLC) [20] storing 2, 3, and 4 bits per cell, respectively, where the undesired transfer of few electrons into/from the storage layer may alter significantly the memory information content. Hereafter MLC, TLC, and QLC architectures will be generically denoted as *multilevel cells*.

The basic parameter characterizing the NAND Flash memory reliability is the Raw Bit Error Rate (RBER), representing the fraction of erroneous bits retrieved during a read operation [15]. The knowledge of this parameter whose value increases with: technology scaling, the number of bits that a cell can store, the number of P/E operations, the time elapsed between two successive read operations, the number of repeated read operations on the same memory location, is now the driver for architectural and software design of present SSDs [21].

Multilevel NAND Flash memories require the availability of an ECC scheme able to correct the errors detected when reading the memory. The choice of the ECC code and the design of the correction engine represent the key point for present SSDs design since they must be carefully calibrated with respect to the figures of merit of the selected nonvolatile memories. A too simple ECC scheme may not be able to guarantee a suitable reliability, whereas a too complex one may reduce severely the read bandwidth because of the time required for error correction, with a consequent impact also on the system power consumption [22]. On the basis of the selected ECC code and of the designed ECC engine, an optimal error reduction algorithm for the memory read operation can be identified. The selection of the appropriate NAND Flash memories and the identification of the adequate ECC scheme represent the key point to guarantee a high QoS for the SSD to be designed.

Once the ECC scheme has been designed, the Bottom-Up design flow rises to the memory controller, representing the interface towards the ECC engine and the memory storage system. The bandwidth provided by the ECC block must be guaranteed by the controller, to avoid that the design efforts devoted to optimize the ECC scheme vanish. With this respect, the SSD controller must be designed in order to manage a sufficient amount of commands to fully exploit the bandwidth of the underlying storage system. Similarly, also the interface towards the host must be able to guarantee the expected bandwidth. For this reason, SATA protocol is no longer able to deal with the performance made available by the other blocks in the SSD architecture [23] so that SAS [24] and PCI-Express [25] are adopted for enterprise environments.

On the basis of this bottom-up SSDs design flow, from an accurate knowledge of the performance and limits of the selected NAND memories to the design of a suitable ECC engine and, successively to that of the controller and of the host interface, also CAD tools for SSD design must follow this Bottom-Up vision, while relaxing the efforts previously devoted to the FTL design [26].

In this paper, starting from a review of the basic reliability issues in multilevel NAND Flash memories, several aspects related to the design of an SSD architecture will be presented. Emphasis will be given to the choice of the appropriate ECC code, the design constraints of the ECC engine able to guarantee the optimal trade-off between performance and reliability [27], the controller design and the selection of the host interface protocol able to sustain the bandwidth provided by the storage system. All the elements will be provided to understand why the SSD performance rapidly decreases with use and time and why a different design approach allows fully exploiting the NAND Flash features while extending the SSD lifetime.

The paper is organized as follows: in Section II multilevel NAND Flash operations and reliability will be analyzed with emphasis on how oxide ageing impacts on endurance, data retention and read disturbs. In Section III, the need of ECC in NAND Flash memories will be discussed focusing on the choice of the proper ECC scheme, on the impact of the decoding time on data read throughput, on the design of read algorithms improving data correction, on the design of ECC engines trading off between architecture efficiency and read bandwidth. In Section IV, a high abstraction-level internal architecture will be described, with emphasis on design constraints for optimal performance exploitation, on the advantages introduced by dedicated command queueing strategies, and on the adoption of DRAM-caching [28] . In Section V, the criteria for the optimal host interface selection will be discussed, focusing on the trade-off between cost and perfomances, on the relationship between queue depth and bandwidth, and on the host payload co-design for optimal performance exploitation. Finally, in Section VI, the paper will speculate on future research opportunities made possible by high-performance SSDs leveraging multi-core controllers, able to revolutionize the traditional computer-memory interaction paradigm by introducing new concepts such as *software defined storage systems* [29].

II. NAND Flash memory cells: operations and reliability

A. Fundamental principles

The traditional Flash memory cell is a metal-oxide-semiconductor device with an electrically isolated floating gate (FG). The insulation is achieved by a tunnel oxide and an interpoly oxide (see Fig. 2) [30]. The former oxide plays a basic role for the control of the device threshold voltage V_T whose value represents, from a physical point of view, the stored information. In quiescent conditions, thanks to the two oxides, the charge stored into the FG does not leak away, thus granting the nonvolatile paradigm fulfillment.

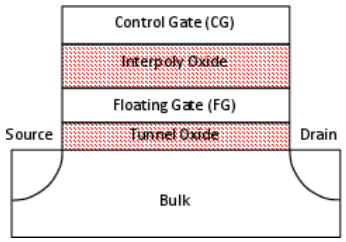


Fig. 2. Standard floating gate memory cell used in NAND architectures

By referring for sake of simplicity to a SLC architecture, storing just one bit per cell, programming is performed by injecting electrons within the FG, whereas erasing is performed by removing that charge from the FG [31]. The charge within the FG modifies substantially the cell's V_T and, consequently, the voltage to be applied to the Control Gate (CG) to switch on the cell as well as the current flowing through the device when a fixed voltage V_{CG} is applied to the CG [32]. Cell writing occurs thanks to the FN tunneling [16]: by applying high electric field to the tunnel oxide, it is possible to transfer charge to/from the FG. This operation requires an accurate control of both V_{CG} and the pulse duration t_p, since V_T must be placed in a well defined interval $[V_{TPmin}, V_{TPmax}]$ (see Fig. 3, where the V_T distributions of a cell array are shown). Using $V_T < V_{TPmin}$ would reduce the threshold window, that is the read margin guaranteeing a read operation immune from errors, whereas $V_T > V_{TPmax}$ could provoke read errors in other cells of the array due to the over-programming phenomenon [33], [34].

During a cell programming, the charge injected within the FG reduces the electric field applied to the oxide. Therefore, to avoid a reduction of the program efficiency, this operation is accomplished by applying to the CG a sequence of pulses with duration t_p and increasing amplitude. Each pulse is followed by a verify operation [35] that ends the program operation when the target V_T interval has been reached, thus realizing the so-called Incremental Step Pulse Programming (ISPP) algorithm [17], [36]. It can be demonstrated that the amplitude increment ΔV_{CG} almost coincides with the threshold shift ΔV_T produced by the pulse itself [37]. The choice of the two parameters ΔV_{CG} and t_p allows controlling the overall programming time and the accuracy of the placement of the cell V_T within the target interval. Long pulses and/or high ΔV_{CG} reduce the programming time with a difficult control of the cell final V_T, whereas short pulses and/or reduced ΔV_{CG} increase the programming time but allow a tight control of the electron transfer to the FG [37], [38].

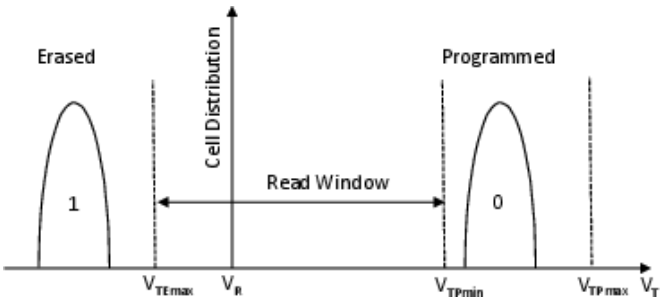


Fig. 3. Threshold voltage distributions in SLC cells. V_{TPminx} and V_{TPmax} represent the minimum and the maximum target V_T for a programmed cell, respectively. V_{TEmax} represents the maximum V_T for an erased cell while V_R denotes the read voltage.

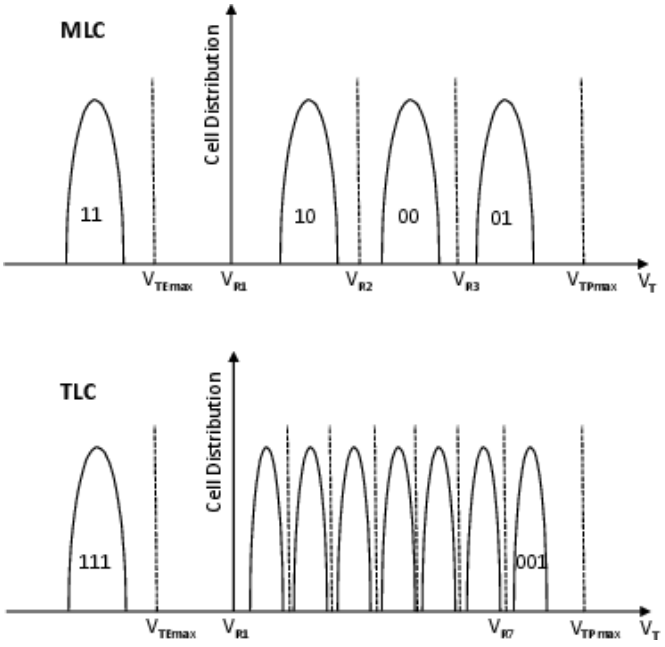


Fig. 4. Threshold voltage distributions in MLC and TLC cells. For the MLC case the 3 reference voltages V_{R1}, V_{R2}, and V_{R3} are shown, whereas for the TLC case only 2 out of 7 reference voltages are shown.

Read operation is performed by evaluating the current flowing through the cell when a fixed reference voltage V_R is applied to CG (see Fig. 3) [30], [39]. In a programmed cell (high V_T) the current is limited and the read circuitry produces a bit equal to 0, whereas in an erased cell (negative V_T) the high measured current is interpreted as a 1.

With the introduction of multilevel architectures (MLC, TLC, QLC) able to store 2, 3 and even 4 bits in a single cell, the programming and the reading operations become much more complex [18], [19], [20]. Since V_{TPmax} cannot be increased because of architectural and operating constrains [40], 3, 7 or even 15 different threshold intervals must be allocated within the same voltage range, each one corresponding to a different set of 2, 3 or 4 bits stored within the cell (see Fig. 4). The amplitude reduction of each interval calls for a very tight control of the charge injected within the FG. Since

the relationship $\Delta V_{CG} \simeq \Delta V_T$ is still valid [37], the ΔV_T reduction forces the overall program time to increase with the number of bits stored in a cell. The read operation too, requires longer times since successive read procedures with different threshold voltage references must be considered [19], [18]. In addition, the reduced separation between adjacent intervals may trigger read errors.

Erase operation, bringing back the cells to the logical "all-1" state is performed simultaneously on all cells belonging to the same block of cells sharing the same Source line [41], [42].

The operations of Flash memory cells described so far refer to an ideal case. In the real world, tunnel oxides face a continuous wear-out reducing the FN efficiency and triggering long-term reliability effects; the charge stored in the FG is not stable but leaks away producing read errors; cell dimensions are so scaled that cell-to-cell variability must be taken into account [43]; the number of electrons injected in the FG is so small that statistical effects during programming may produce errors. Finally, even an ideal cell is embedded in a complex array architecture so that write and read operations performed on neighbor cells may alter its stored content.

B. Reliability effects

Tunnel oxide degradation represents the fundamental cause affecting Flash memories reliability. Because of the continuous charge transport through the insulator, traps can be created at the SiO_2 interfaces or within the oxide, which can modify the FN tunneling dynamics [13], [40], [44]. The ability of controlling tight threshold distributions decreases with the number of Program/Erase (P/E) operations, thus affecting the memory endurance [14], [15]. Fig. 5a sketches the effects of a reduced ability in producing tight distributions as the number of P/E cycles increases. The *Program & Verify* approach stops the program operation of a cell when the target threshold interval has been reached [35]. However, because of the tunnel oxide wear-out, some cells can be slightly over-programmed and their thresholds could end in an adjacent interval [33], [40]. As a consequence of this distribution broadening, read errors are produced. Fig. 6 shows the RBER measured in a TLC NAND Flash manufactured in the 1x-nm planar technology node as a function of the number of P/E cycles, evidencing a reliability reduction induced by successive write operations.

Oxide ageing and traps creation also reduce the data retention feature, that is the ability of keeping unaltered the charge within the FG when the cell is in a quiescent state. Electrons may escape from the FG because of trap-assisted tunneling or Stress-Induced-Leakage-Current (SILC) effects [45], [46], [47], [48], [49], [50]. Fig. 5b outlines the threshold distribution shifts that may produce read errors that become more probable with the time elapsed since the last program operation. The risk that the threshold of a cell programmed in a given interval shifts to an adjacent interval increases significantly with the number of bits stored in a single cell. It is worth to point out that in a MLC or TLC architecture the number of electrons differentiating two adjacent intervals is in order of few tens, whereas in QLC cells it is sufficient that one or two electrons escape from the FG to produce a read error [51].

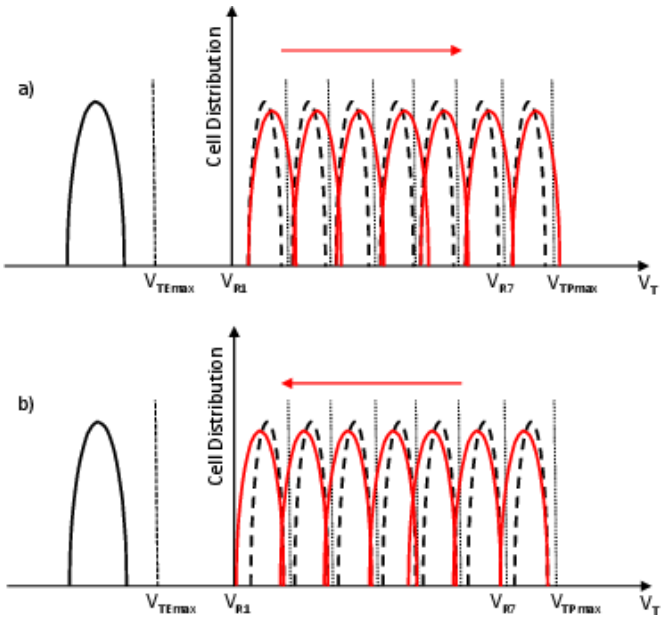


Fig. 5. Shifts of the threshold voltage distributions in TLC cells caused by oxide ageing (dashed line: virgin samples; full line: ageing effects). Shifts towards higher intervals are caused by endurance effects (a), since the correct placement of the threshold voltage in a given interval becomes more difficult, whereas shifts towards lower intervals are due to electrons escaping from the FG causing a reduced data retention (b)

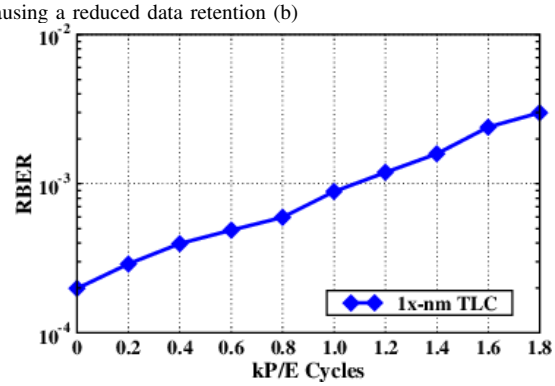


Fig. 6. RBER measured in a 128 Gb TLC NAND Flash die manufactured in the 1x-nm planar technology node as a function of the number of P/E cycles, up to twice the rated endurance (900 P/E cycles).

Besides the degradation mechanisms related to oxide wear-out described so far, other effects may worsen the ability of controlling the correct number of electrons to be transferred in the FG during a single programming pulse. Among them, the Random Telegraph Noise (RTN) related to filling/empting of tunnel oxide traps affects the V_T distributions stability few microseconds after the application of the programming pulse, creating distribution tails below the target verification level [52], [53], [54], [55]. Additionally, positive trapped charge in the tunnel oxide during cycling results in a modified FN tunnel dynamics that may trigger erratic effects [33], [56], [57], [58]. These sporadic mechanisms, that may potentially affect any cell in the array, have a random and transient nature; they can occur during any programming pulse and they may produce threshold shifts larger than expected, with the risk of programming some cells with a threshold voltage larger than the desired one. The limited number of electrons discriminating

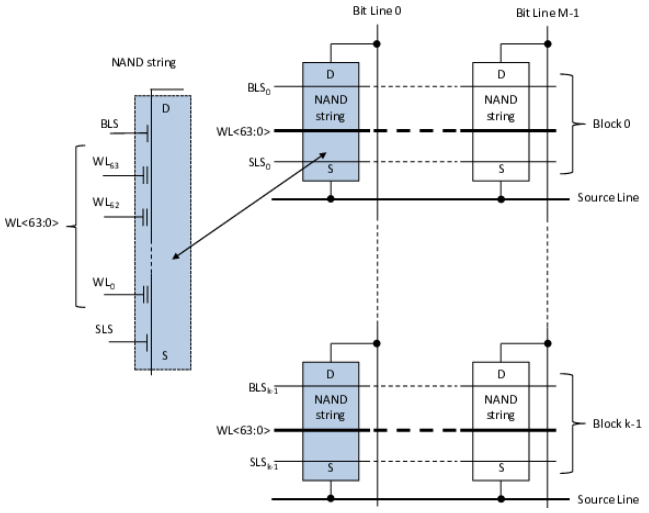


Fig. 7. Schematic organization of a NAND flash array. Each cell string is connects to a Bit line and a Source line through two select transistors (BLS and SLS, respectively).

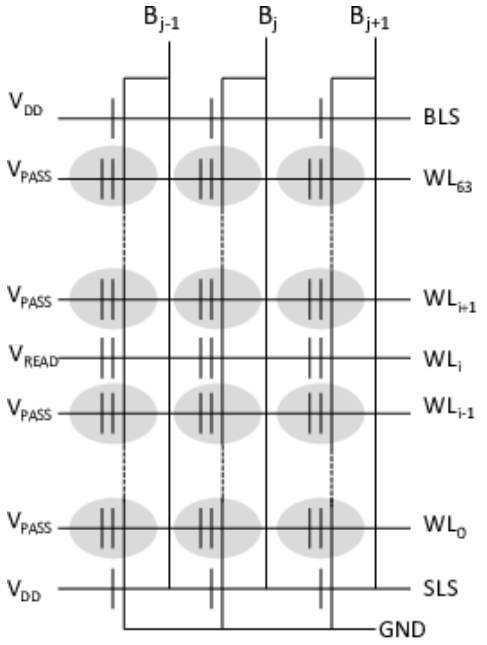


Fig. 8. Representation of read disturb in a NAND Flash array when reading cells in the WL_i word line. All cells sharing the same strings (marked in gray) are potentially affected by the read disturb.

between adjacent intervals makes the programming operation discrete [38], [59], [60].

Fig. 7 shows the schematic of a typical memory array. Cells are organized in strings, which are the minimum read unit. Read and program operations are performed *page-wise*, by reading/programming simultaneously 2 kB or 4 kB cells belonging to the same word line [32].

Architectural solutions for memory operations may also affect the overall reliability, by producing errors and even cell failures. The most common effects are the so called disturbs, that can be interpreted as the influence of an operation performed on a cell (Read or Write) on the charge content of a different cell. Read disturbs are the most frequent source of disturbs in NAND architectures [32], [61], [62]. This kind of disturb may occur when reading many times the same cells without any erase operation of the entire block they belong to. All the cells belonging to the same string of the cell to be read must be driven in an ON state, independently of their stored charge (see Fig. 8). The relatively high $V_{PASS} > V_{TPmax}$ applied to the CG of the unselected cells to turn on their conduction and the sequence of pulses applied during successive read operations may induce a charge gain due to SILC effects [61] or hot carrier effects [62]. These cells suffer a threshold voltage shift that may lead to read errors, when addressed. The probability of suffering from read disturb increases with the P/E number (i.e., towards the end of the memory useful lifetime) and it is higher in damaged cells. Read disturbs do not provoke permanent oxide damages: if erased and then reprogrammed, the correct charge content will be present within the FG.

The NAND Flash technology scaling has introduced additional disturbance mechanisms affecting the array reliability: the cell-to-cell interference [63], [64], [65], [66] and the Gate Induced Drain Leakage (GIDL) [67], [68]. The former issue is mainly caused by the FG coupling due to parasitic capacitances between cells, thus it is greatly affected by cell scaling, and is well known to widen the V_T distributions by producing read errors. The latter effect is due to the usage of the self-boosting technique to inhibit unselected cells during programming [69]. An electron-hole pair generation mechanism triggered by high electric fields during the program operation leads to the generation of charge in the region between the Source Line Selector (SLS) and the WL_0 that can be injected as hot electrons in the floating gate of cells belonging to WL_0 [67]. To avoid this effect, dummy word-lines need to be integrated in the array.

III. The impact of ECC on SSD performance

As summarized in the previous section, because of endurance problems, poor data retention or read disturbs, the actual threshold voltage read in a cell may be different from the programmed one [15]. Therefore, when a page is read, some cells may return a wrong value, thus producing read errors. To overcome these problems, data encoding guaranteeing a reconstruction of the correct read page data is mandatory in electronic systems using NAND Flash memories.

The correction capability of the code to be adopted is strictly related to the error probability. For a given technology node, since physical degrading mechanisms are the same independently of the different storage paradigms (SLC, $\cdots$, QLC), the error probability increases with the number of bits stored in a single cell since the smaller the number of electrons associated to each data pattern, the higher the probability of having a V_T different from the expected one.

In the first SLC memories, thanks to the large V_T gap between the two threshold voltage distributions, the error probability was very low, so that Bose-Chaudhuri-Hocquengham (BCH) codes able to correct few tens of bits in a 1 kB or 2kB page were sufficient. With limited number of errors to be corrected, the correction time was not an issue and the read

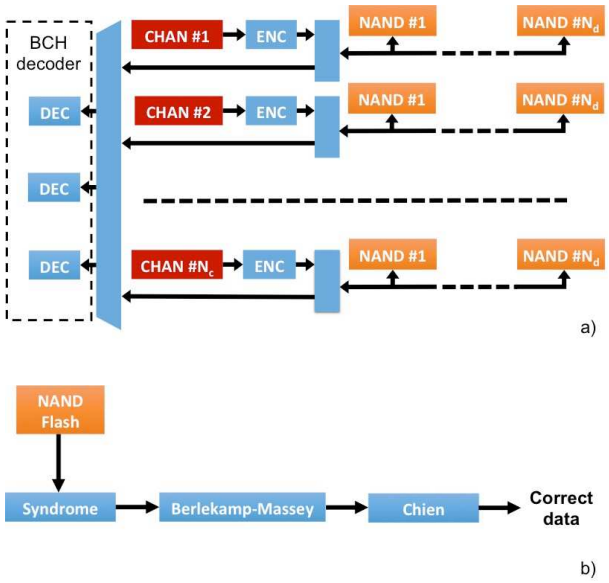


Fig. 9. a): schematic representation of an ECC architecture based on BCH codes. A high-speed encoder is connected to each SSD channel whereas a a reconfigurable parallel decoder is shared among the N_c channels. b): schematic representation of the BCH decoder.

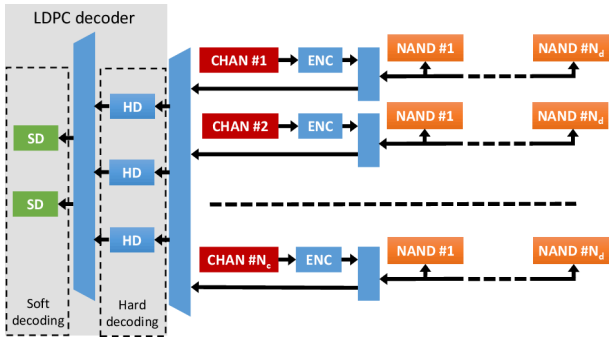


Fig. 10. Schematic representation of an ECC architecture based on LDPC codes. The decoding path is composed by two main blocks: the hard decoding, whose architecture is similar to that designed for BCH engines and the soft-level decoding.

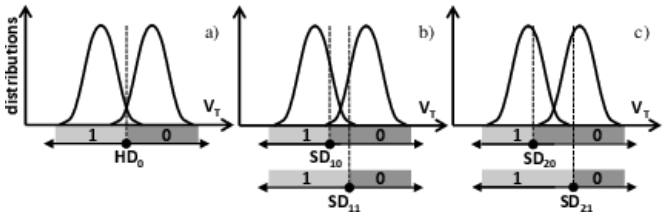


Fig. 11. NAND Flash read references used in the two levels LDPC sensing scheme to discriminate between two adjacent threshold voltage distributions. A memory page is read by setting the read voltage at HD_0 and determining, for each bit, whether $V_T < HD_0$ or $V_T > HD_0$ (a). If the ECC engine is not able to correct possible read errors, the soft decision algorithm starts and the page is read twice by moving the read references around HD_0, to SD_{10} and SD_{11} (b). If the page is still marked as uncorrectable, the page is read again with the SD_{20} and SD_{21} references (c). Reprinted with permission from [76].

bandwidth and latency were marginally affected by the use of ECCs [70]. Read bandwidth is the number of read operations sustained in a given time, whereas latency is the time elapsed between a read command submission and its completion. Fig. 9a shows the typical blocks for ECC engines based on BCH codes: a high-speed encoder is connected to each one of the N_c SSD channels (that is a bus used to communicate with an array of N_d memory dies), whereas a reconfigurable parallel decoder (i.e. a multi-engine decoder) is shared among the channels [71]. The structure of the decoder is represented in Fig. 9b, where the *Syndrome block* determines whether an error is present, the *Berlekamp-Massey block* calculates the coefficients of the error locator polynomial, and the *Chien machine* locate the errors [70].

In multilevel architectures the number of errors to be corrected increases by an order of magnitude for any further bit stored in a single cell. Although ECC engines based on BCH codes are still used thanks to their simple hardware implementation, high numbers of bits to be corrected may impact significantly on the overall read time. As a consequence, the correction time may become the bottleneck of the entire read procedure [21]. In addition, because of the high number of errors, the probability of having uncorrectable pages (that are pages read with a number of wrong bits higher than the ECC correction capabilities) increases [72]. When a page is marked as uncorrectable, the read operation fails and the page content is irremediably lost. The adoption of parallel decoding architectures can reduce the bandwidth and latency degradation (at the expenses, however, of both area occupation and power consumption) but it cannot solve the problems caused by uncorrectable pages.

To deal with the presence of uncorrectable pages, two alternatives exist: *i)* keep BCH codes and their ease of implementation while defining sophisticated read algorithms in order to reduce the number of errors [73], [74]; *ii)* develop ECC solutions based on different coding concepts, like Low Density Parity Check (LDPC) codes [75]. In the former case, the basic idea in the presence of uncorrectable pages consists in re-read the page with different read reference voltages, in the attempt of tracking the shift of the threshold voltage distributions. Such a solution led to the development of different read algorithms, generally defined as *read retry* [73]: they are automatically managed by the ECC engine and they call for (at least) a page re-reading with the unavoidable degradation of the read bandwidth. The latter solution adopts LDPC codes that, differently from BCH codes, present a much higher correction capability [75]. Fig. 10 shows the typical blocks for ECC engines based on LDPC codes: the decoding engine is composed by two main blocks: the Hard Decoding (HD) and the Soft Decoding (SD).

From an operative point of view, LDPC decoding works as follows. As shown in Fig. 4, multilvel NAND Flash memories are read page-wise by using a set of read reference voltages, hereafter denoted as HD_0 (see Fig. 11a showing the read reference discriminating between two adjacent threshold voltage distributions). Cells are read as 1 or 0 depending on their threshold voltage V_T with respect to HD_0. If during the ECC decoding phase the page is evaluated as uncorrectable, the LDPC decoding algorithm can be retried with the SD. To accomplish this second step, more information about the actual position of the NAND Flash threshold voltage distributions

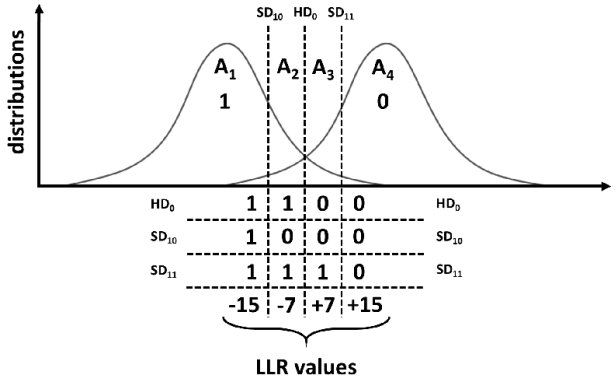


Fig. 12. Voltage threshold partitioning performed during a one Soft Decoding (SD). Four regions (A1, A2, A3, and A4) are marked by HD_0, SD_{10}, and SD_{11}.

must be collected. Basically, the algorithm moves sequentially the internal read references to SD_{10} and SD_{11} (Fig. 11b) thus reading the page twice. Data are transferred to the LDPC decoder and then they are bit-wise combined with those previously read with HD_0. This step is possible because during the whole SD process the data read with the HD_0 reference are stored in a dedicated buffer inside the SSD controller and used as a reference.

Thanks to this multiple read operation it is possible to calculate the information needed by the SD: the Log-Likelihood Ratios (LLRs) [7]. The calculated numbers are used as input for the soft decoder and are defined as follows:

$$LLR(y_i) = ln\frac{P(x=0|y_i)}{P(x=1|y_i)} = ln\frac{P(y_i|x=0)}{P(y_i|x=1)} \quad (1)$$

where P is the probability, whereas x and y_i represent the transmitted (i.e., the programmed value) and the received (i.e., the read bit) symbols, respectively [77]. As a matter of fact, when a set of read reference voltages is used (SD_{10} and SD_{11}), Eq. (1) defines that the LLRs can be viewed as the probability of reading a 0 or a 1 given the value of a specific programmed bit [7]. In other words, the higher the absolute value of the LLR is, the higher the confidence that the read bit is correct. [78].

An example of the result of the SD process is sketched in Fig. 12. As it can be seen, the bit-wise combination of the data read from the NAND flash memory defines four different regions of the threshold voltage distributions. These represent *de facto* a probability density function of the programmed cells: the probability that a bit belongs to one of the areas (A_i with $i = 1, ..., 4$ in the example of Fig. 12) identified by the hard and the soft references is defined as follows:

$$P(X \in A_i) = \int_{A_i} p_X(x)dx \quad (2)$$

where X represents the programmed bit, and $p_X(x)$ is the actual threshold voltage distribution. At this point, it is clear that to extrapolate the LLRs expressed in Eq. (1) it is sufficient to calculate a bounded logarithmic ratio between the number of cells read as 0 and those read as 1.

Once the LLRs are calculated for all the regions, instead of using the raw bits coming from the NAND flash memory

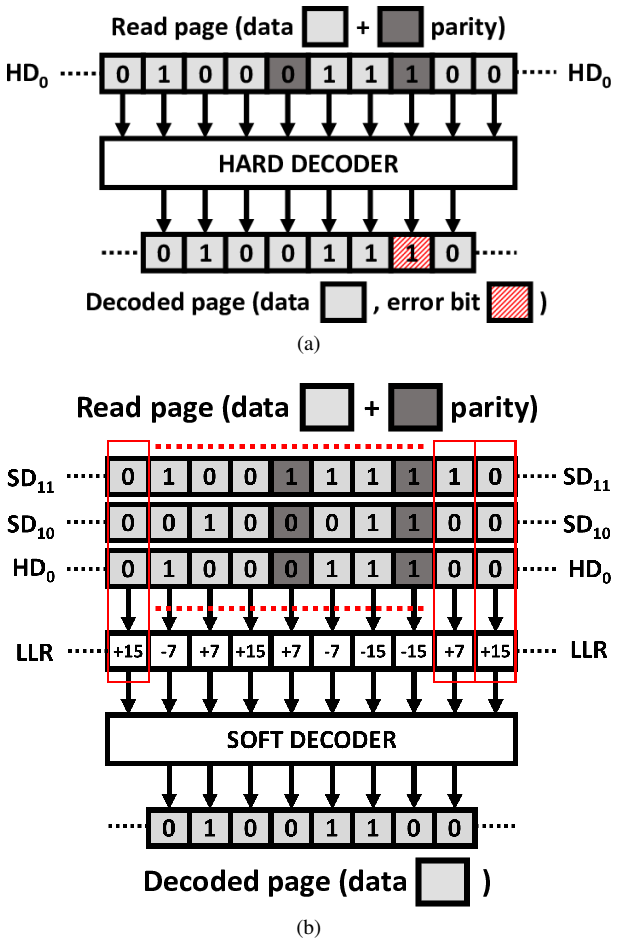


Fig. 13. Difference between the HD and the SD decoding phase. When HD is considered, raw bits coming from the NAND flash memory are used as input of the decoder. In this example after the decoding step one bit is still in error, therefore SD is required. When SD is considered, LLRs computed by the bit-wise combination of data read with HD_0, SD_{10}, and SD_{11} are used as input (see Fig. 12).

(HD decoding sketched in Fig. 13a), the SD decoder translates the bit-wise combination of the data read with HD_0, SD_{10}, and SD_{11} with the corresponding LLR values, and it starts the decoding procedure (see Fig. 13b). At this point, a purely probabilistic decoding process is triggered.

If the decoding process still fails, a second iteration is performed by moving the read references to SD_{20} and SD_{21} (Fig. 11c) and comparing the new read data with those previously analyzed and stored in the dedicated buffer. In this case the number of regions defined by the read voltage references switch from 4 to 6, therefore the LLRs values must be computed again by the decoder. The algorithm continues this process until the page is correctly read or the maximum number of soft-levels is reached and the page is marked as uncorrectable [22]. Finally, since LDPC codes provide a probabilistic correction, they are not immune from errors like false-decoding that occurs when the ECC performs erroneous correction while declaring successful decoding [79]. The presence of false-decoding errors is strictly related to the LDPCs mathematical characteristics and, therefore, it is essential to identify *a priori* the algorithm minimizing these errors [70].

LDPC codes, although presenting much higher correction

TABLE I
LDPC AND BCH FEATURES BENCHMARK (DATA FROM [82]).

	BCH	LDPC
Decoding Algorithm	Algebraic-based	Probability-based
Guaranteed correction	Yes	No
Soft Bit Decoding	Hard (Read Retry)	Easy
Hard Decoding Performance	Code dependent	Similar to BCH
Soft Decoding Performance	-	2X-3X
Decoding complexity	Low	High
Power consumption	Medium	High
Cost	Low	High

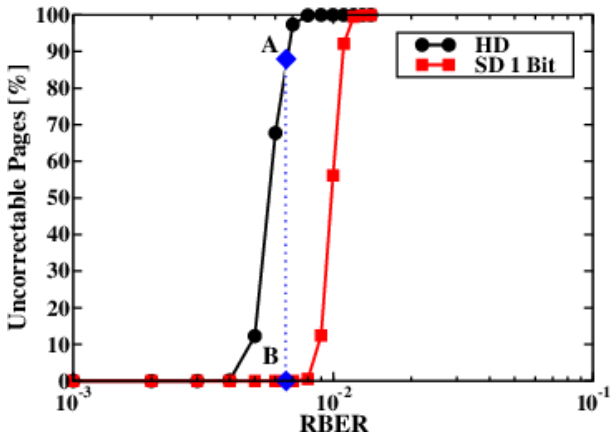


Fig. 14. Correction strength of both HD and SD when a LDPC able to correct up to 100 Bits in a 4320 Bytes codeword is considered for a 128-Gbits TLC NAND Flash memory manufactured in a planar 1x-nm technology node. Points A and B represent the maximum measured percentage of uncorrectable pages at the end of the memory lifetime, when HD and SD are used, respectively.

capabilities with respect to BCH, can still fail the correction process in presence of pages with large numbers of errors. Also in these cases there exist re-reading algorithms (for instance the *multiple soft decision*) that can correct pages initially marked as uncorrectable at the expense of the overall reading time [80], [81], [76]. Table I summarizes the features of LDPC and BCH described in this section.

To evaluate the optimal ECC engine design in terms of HD and SD implementation, the knowledge of the actual memory RBER is mandatory. With this respect, it is usual to leverage a *worst-case* design methodology where the correction strength figure of the HD is compared with the maximum percentage of uncorrectable pages measured at the end of the memory's lifetime. Fig. 14 shows this process when a LDPC able to correct up to 100 bits in a 4320 Bytes codeword is considered for a TLC NAND Flash memory manufactured in a planar 1x-nm technology node. Point A marks the maximum percentage of uncorrectable pages measured at the end of the memory's lifetime. As it can be seen, in this case switching from the HD to a one bit SD is sufficient to correct all the errors (point B). Other correction strategies like a two bits SD, become an over-design.

The above considerations are mandatory when it is required to design the optimum LDPC architecture (both in terms of correction strength and correction bandwidth) for the target

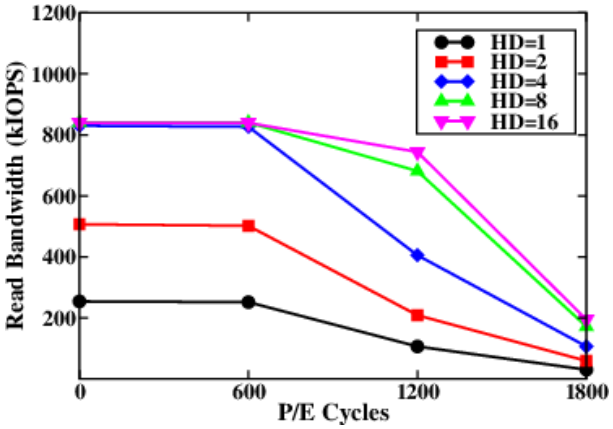


Fig. 15. Read bandwidth evolution as a function of the number of P/E cycles in a 2 TB SSD featuring a PCI-Express GEN3x4 host interface and 16 channels each connected to eight 128-Gb TLC NAND Flash dies manufactured in a planar 1x-nm technology node with a rated endurance of 900 P/E cycles. The ECC engine is composed by a variable pool of HD decoders and a single SD decoder. Each hard decoder has a decoding bandwidth of about 1.2 GB/s.

SSD. In fact, since the SD directly impacts the drive's bandwidth, once the correction strategy is defined (a one bit SD rather then a two bits SD) and the decoder's bandwidth is fixed, it is important to find the right balance between the number of HD and SD decoders. Fig. 15 shows the read bandwidth obtained, for different HD implementations, in a 2 TB SSD featuring 16 channels each one connected to eight 128-Gbits TLC NAND Flash dies manufactured in a planar 1x-nm technology node, as a function of the number of P/E cycles. The correction strategy used in this example is the same sketched in Fig. 14, therefore, a 1 Bit SD has been used. All results have been obtained by using the SSDExplorer simulator [26]. Since each hard decoder has a bandwidth of 1.2 GB/s and the SSD host interface is a PCI-Express GEN3x4 [25] with a maximum bandwidth of 4 GB/s, it is clear that a coarse design choice (that neglects the actual RBER evolution) requires 4 HD decoders and any higher number would result in a cost ineffective overdesign.

However, since RBER increases with the number of P/E cycles (see Fig. 6), the percentage of uncorrectable pages detected by the HD increases as well. As a consequence SD is triggered and the read bandwidth rapidly decreases when the memory rated endurance (P/E = 900) is approached. To guarantee the expected performance and to extend the SSD working window, it is necessary to increase the number of HD decoders (see Fig. 15) as well as that of SD decoders. Fig. 16a shows the calculated read bandwidth degradation with respect to the beginning of life at P/E = 1200 and P/E = 1800 (i.e., at twice the rated endurance) by implementing 8 HD decoders and different numbers of SD decoders. As it can be seen, to reduce the read bandwidth degradation at twice the rated endurance, 2 SD decoders can be used, while any larger number of decoders would result in an overdesign. Fig. 16b shows the results obtained by using 16 HD decoders and different numbers of SD decoders, showing a significant performance improvement thanks to a much higher hardware cost. From a designer point of view, an accurate

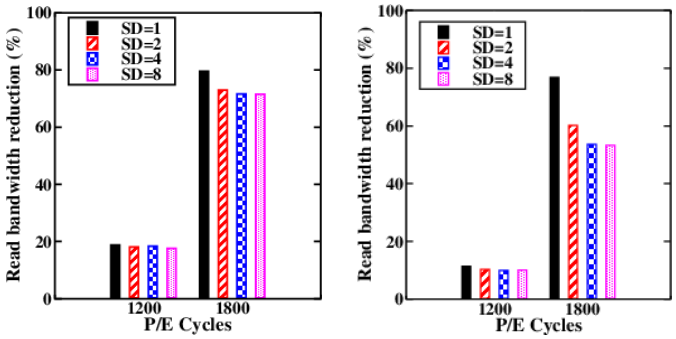


Fig. 16. Read bandwidth degradation with respect to the beginning of life at P/E = 1200 and P/E = 1800 (*i.e.* at twice the rated endurance) considering different SD levels. 8 and 16 HD decoders have been considered in Fig. a) and b), respectively.

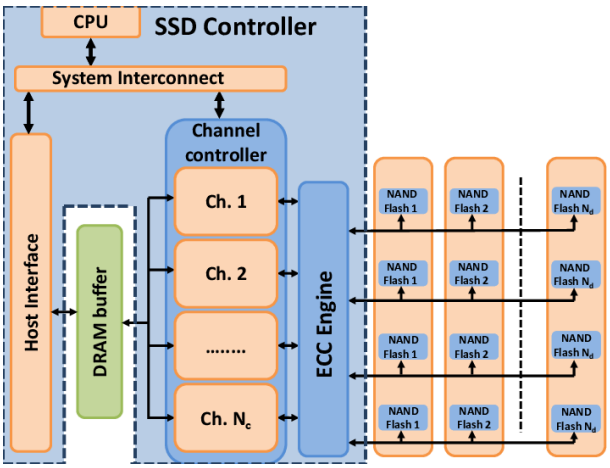


Fig. 17. Schematic representation of the SSD controller, considering N_c channels and N_d memory dies connected to each channel.

trade-off evaluation between performance (*i.e.* read bandwidth reduction) and hardware cost must be based on the actual knowledge of the memory RBER evolution.

By summarizing the previous reasonings, in multilevel Flash memories the use of sophisticated ECC architecture is mandatory in order to efficiently correct a number of errors that increases with the memory endurance and with the time elapsed between two successive read operations of the same page. These ECC engines, however, strongly impact on the read bandwidth and latency. This holds true, in particular, when uncorrectable pages are detected, since advanced read algorithms are required. Therefore, the choice of the ECC code to be implemented and of its correction capability, the design of the ECC engine architecture, and the identification of the most effective re-reading algorithm depend on the memory reliability and, in particular, on the BER whose value grows with the memory wear-out.

The optimal design of the reading path for a *delay insensitive* SSD must be based on the accurate knowledge of the performance and reliability of the selected memories and, therefore, on a careful pre-characterization of the memories themselves in order to estimate their BER [83].

IV. SSD CONTROLLER DESIGN

The main block diagram of an SSD controller is shown in Fig. 17. Once the SSD's specifications have been fixed, and hence the maximum device bandwidth has been defined, the SSD controller design follows a simple rule of thumb to calculate N_c and N_d needed to meet the requirements. Basically, to calculate the actual controller bandwidth B_{cont}, it is sufficient to sum the bandwidth contributions B_{ch} of each channel:

$$B_{cont} = \sum_{i=1}^{N_c} B_{ch_i} \ . \tag{3}$$

The maximum channel bandwitdth $B_{ch_i}^{max}$ is obtained under the assumption that all the memory dies connected to channel i are addressed at the same time. By defining B_d as the bandwidth of each memory die, the theoretical controller bandwidth B_{cont}^{th} is given by:

$$B_{cont}^{th} = \sum_{i=1}^{N_c} B_{ch_i}^{max} = \sum_{i=1}^{N_c} N_{d_i} B_d \ . \tag{4}$$

Eq. (4) represents, however, the theoretical condition under the hypothesis that all single dies can communicate simultaneously with the controller and, therefore, it represents the maximum achievable value. Unfortunately, for several reasons (e.g., access request to the same die, die's response time slowed down by a read retry operation, die busy for a program operation whose latency is much higher with respect to read latency, etc.), the probability that all dies can communicate simultaneously with the controller is generally < 1. Taking into account that a number n of dies in a channel cannot serve new requests since they are processing other commands, the actual controller bandwidth is given by:

$$B_{cont} = \sum_{i=1}^{N_c} (N_{d_i} - n_i) B_d \leq B_{cont}^{th} \ . \tag{5}$$

The above equation calculates the controller bandwidth in a fresh condition (i.e., at the beginning of the drive's lifetime). However, as previously shown in Section III, the actual performance of the SSD is strongly affected by the reliability phenomena associated with the storage layer. As a consequence, to take into account these effects, Eq. 5 can be modified as follows:

$$\begin{aligned} &B_{cont}(P/E, T, RD, WAF) = \\ &= \sum_{i=1}^{N_c} (N_{d_i} - n_i(P/E, T, RD, WAF)) B_d \leq B_{cont}^{th} \end{aligned} \tag{6}$$

where P/E, T, RD and WAF are the current Program/Erase cycle number of the drive, the working Temperature, the Read Disturb level of the memories, and the Write Amplification Factor, respectively. The WAF factor is defined as

$$WAF = \frac{data\ written\ to\ the\ NAND\ flash}{data\ written\ by\ the\ host} \geq 1 \ ; \tag{7}$$

it has been accurately described in [84] and it depends on several factors ascribed to the FTL implementation including Wear Leveling, Garbage Collection, and Bad Block management algorithms. Along with WAF, P/E, T, and RD introduce hard-to-model effects that complicate the description of the controller's bandwidth in a *closed form*. Therefore, to help SSD designers to calculate the actual performance and latency of a target SSD over time and use, the adoption of sophisticated simulation tools like SSDExplorer is mandatory [26].

Overall, what ultimately stands out from both Eq. (5) and Eq. (6) is that, to approach as much as possible the ideal controller bandwidth, it is necessary to:

- reduce the probability that a command addresses a busy die (i.e., a die already scheduled by another operation);
- maximize the number of dies that can process a new command.

This can be accomplished: i) by increasing the number N_d of dies connected to each channel, which however impacts on the SSD cost; ii) with an effective command management performed by the FTL; iii) by using a DRAM as a data buffer.

A. *Efficient command management*

In nowadays SSDs, to efficiently manage the commands issued by the host, it is possible to leverage the *Command Queue* (CQ) concept [85]. This resource is usually implemented as a software routine shared between the host interface, which pushes host commands inside the CQ, and the SSD controller which manages the requested operations and pulls out the commands from the CQ.

Fig. 18 shows the queuing hierarchy usually implemented in traditional SSD controllers [86]. Besides the external host CQ, it is common to have a dedicated small command queue for each NAND Flash memory die: the *Target Command Queue* (TCQ). Basically, thanks to the TCQ, the host can continue to issue commands even when it tries to read or program a die which is in the busy state. In fact, when this condition is verified, the command is simply queued in the TCQ and the SSD controller can continue to fetch other commands from the host CQ. This technique allows maximizing B_{cont} since TCQs keep always busy all the NAND Flash dies. It is thus clear that the main parameters controlling B_{cont} are the parallelism (i.e., N_c and N_d) and the queue depth (QD), that is the number of commands that the host interface can store.

The attempt of approaching the ideal performance in terms of bandwidth by increasing QD presents an unavoidable disadvantage: the increase of the service time (*i.e.* the time elapsed between the issue and the execution of a command) and, consequently, of the SSD latency. Therefore QD has a severe impact on QoS, that basically defines the maximum acceptable latency of the drive and it is calculated as the 99.99-th percentile of the SSD latencies cumulative distribution. To this extent, QoS is used to quantify how the SSD behaves in the worst-case conditions [9]. By using this metric it is possible to understand if the target SSD architecture is suitable for a specific application, such as real-time and safety-critical systems [87]. Fig. 19 shows an example of how B_{cont} and

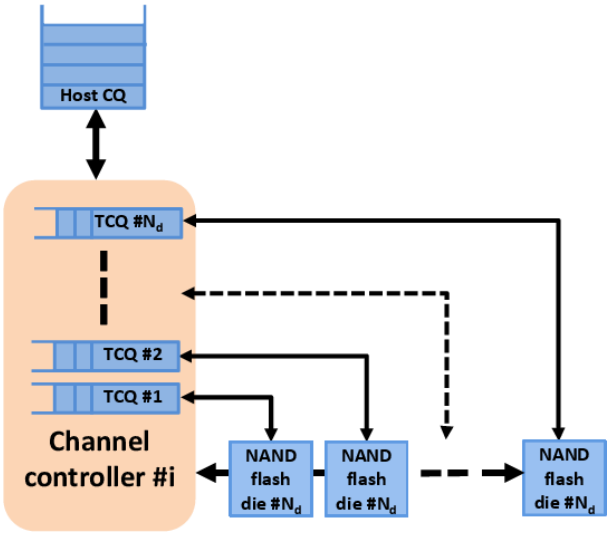


Fig. 18. Queueing hierarchy implemented inside the SSD controller for a generic channel

QoS scale with the host QD. As expected, both B_{cont} and QoS increase with QD. This behavior, however, is in contrast with the requirements of high performance SSDs, which ask for achieving the target bandwidth with the lowest QoS. In fact, state-of-the-art user applications such as financial transactions or cloud platforms [88], [89] are designed to work with storage devices which have to serve an I/O operation within a specific time-frame which is usually upper-bounded by the QoS requirement.

To deal with this requirement it is possible to use the *Head-of-Line* (HoL) blocking concept, whose effect is to limit the number of outstanding commands inside the SSD, thus partially solving the latency issue [90]. The HoL blocking is managed by the controller firmware implementing a FIFO stack whose dimensions can be dynamically defined. When the number of commands queued in a TCQ exceeds a predefined threshold, it is possible to trigger a blocking state inside the SSD controller which stops the submission of a new command from the host CQ. In such a way, depending on the HoL threshold value, it is possible to avoid long command queues inside the TCQs and, hence, the device QoS can be limited within a defined window.

Figure 20 shows the effectiveness of the HoL blocking for the case analyzed in Fig. 19. As soon as the target performance of 300 kIOPS is reached (QD = 64), the HoL blocking effect starts keeping the QoS below the target requirements even when long QDs, such as QD = 128 and QD = 256, are considered.

The fine-grained QoS calibration made available by the HoL blocking, however, does not come for free. If, besides B_{cont} and QoS, the average SSD latency is taken into account, it is clear that the HoL blocking effect has to be wisely used (see Fig. 21). When the HoL blocking is triggered, it trades the QoS reduction with an increase of the average latency. Moreover, this behavior becomes more pronounced when high QDs are used, *i.e.* when a higher QoS reduction is required.

Summing up, it becomes clear that the performance opti-

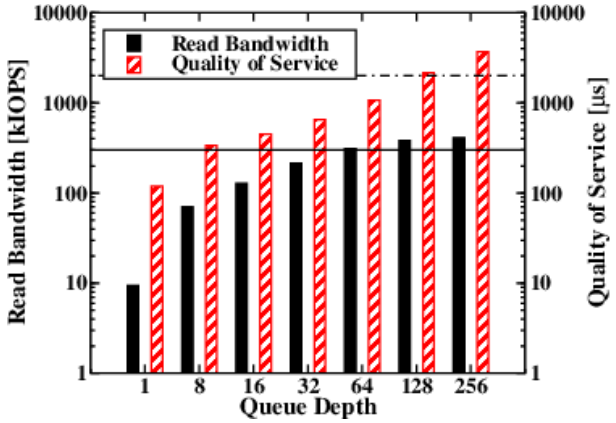


Fig. 19. B_{cont} and QoS as a function of the host Queue Depth. The full line and the dashed-dotted line represent the target B_{cont} and the target QoS, respectively. Simulations refer to an SSD featuring N_c = 8 and N_d = 8 TLC NAND Flash manufactured in a planar 1x technology node. Average read time is 86 μs and workload is 100% 4 kB random read.

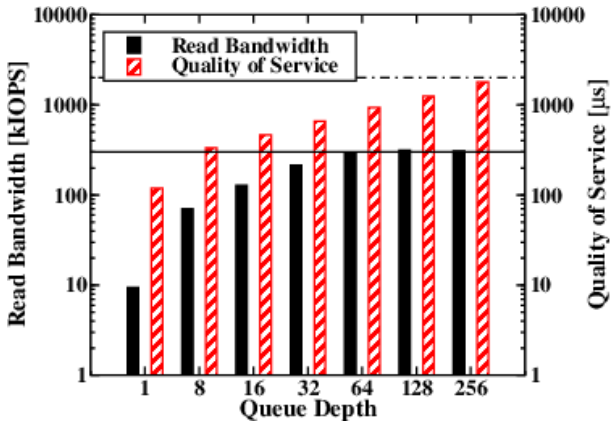


Fig. 20. SSD bandwidth and QoS for the same case of Fig 19 as a function of the host queue depth when HoL blocking is used. The full line and the dashed-dotted line represent the target B_{cont} and the target QoS, respectively.

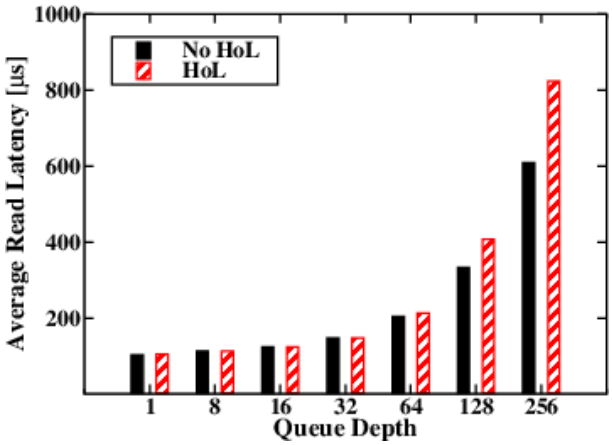


Fig. 21. Average SSD latency evaluated as a function of the host queue depth, for the same case of Fig 19, with and without the HoL blocking.

mization process that has to be followed by SSD designers must involve the optimization of the bandwidth, the average and the maximum latency, the length of the command queue, the command management policy, the head of line blocking, all considered at the same time.

TABLE II
NAND/DRAM SIZE RATIO AND SSD PERFORMANCE FOR THE SAME CONFIGURATION OF FIG. 19 CONSIDERING AN UNIFORMLY DISTRIBUTED LBA SPACE.

NAND/DRAM size ratio	No cache	256	50	15
Cache hit probability [%]	0	0.6	2.7	8.2
Read Bandwidth [kIOPS]	301	312	318	337
Average latency [μs]	206	204	200	189
QoS [ms]	1.07	1.19	1.13	1.03

B. DRAM data caching

To increase the controller bandwidth and to approach as much as possible the theoretical bandwidth B^{th}_{cont}, it is possible to use a DRAM as data cache buffer [28]. As shown in Fig. 17, this block is located between the host interface and the channel controller. Standard data caching algorithms can be adopted, such as Least Recently Used (LRU) or Least Frequently Used (LFU) [91], to decrease the number of accesses to the Flash memories. Since data are addressed in a much faster memory, the access time can be reduced with respect to a standard NAND Flash read/program operation. In addition, since part of the data to be read/written are stored in the DRAM buffer, the number of accesses to the NAND Flash dies are reduced, thus limiting the number of busy dies.

These effects positively impact the SSD bandwidth and the average latency. Moreover, the reduction of the number of accesses to the NAND Flash dies increases their reliability. This point is strictly related to the smaller number of write operations, thus limiting endurance effects and, possibly, leading to a reduced read disturb issue (see Section II-B).

Table II shows the cache hit probability, the read bandwidth, the average latency, and the QoS calculated for the "no cache" case (i.e., a case where the DRAM data cache buffer is not present, assumed as reference) and for different ratios between the total NAND and the DRAM sizes. The number of cache hits (*i.e.* the percentage of memory accesses to the DRAM buffer with respect to the total number of data accesses) depends on the probability of addressing any single nonvolatile memory page. All data have been collected considering a uniformly distributed Logical Block Address (LBA) space of the SSD and a LRU eviction policy is used as caching algorithm.

As it can be seen, the perfomance metrics of the simulated drive are not significantly influenced by the DRAM size. This is due to the fact that the LBA space is uniformly distributed across all the SSD pages, therefore all data locations have the same probability to be addressed.

An uniformly distributed LBA space, however, represents the worst-case condition for the assessment of the benefits materialized by a caching algorithm. In general real user workloads tend to follow different LBA distributions which are more similar to a Gaussian or a Log-Normal with a mode around a specific address. As a consequence, if the I/O address profile of the target application is known, it is possible to optimize the DRAM cache size depending on the statistical parameters presented by the LBA profile itself.

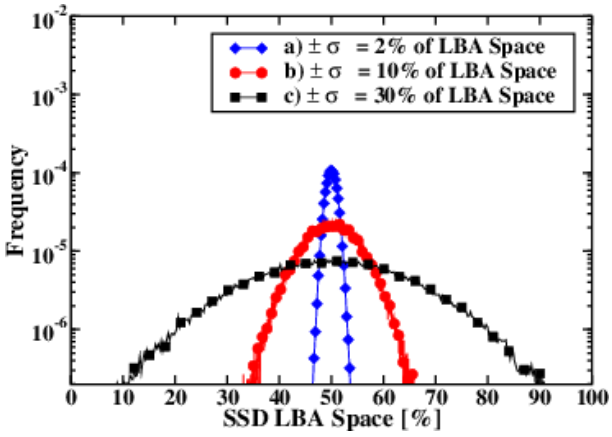


Fig. 22. Examples of three gaussian distributions of the I/O addressing space. The median of the distributions is placed at the the 50% of the SSD LBA addressing space in all cases.

Fig. 22 shows three examples of Gaussian workloads spanning across the whole LBA space of the drive. By considering a $\pm\sigma$ deviation around the average of the total SSD LBA address space, it is possible to design the proper DRAM size ratio in two different ways:

- reducing the DRAM capacity while keeping the same cache hit probability and drive performance;
- increasing the DRAM capacity maximizing the number of cache hits and, therefore, boosting the drive performance.

Table III shows, for the three cases of Fig. 22, the NAND/DRAM size ratio, the cache hit probability, the read bandwidth, the average latency, and the QoS of the target SSD architecture. As it can be seen, the performance metrics are almost similar with a significant reduction of the DRAM size for the tightest workload distribution of Fig. 22.

Table IV shows, for the b) case shown in Fig. 22, the NAND/DRAM size ratio, the cache hit probability, and the performance metrics of the target SSD architecture. With respect to the b) case of Table III the NAND/DRAM size ratio has been reduced from 50 to 15. As it can be seen, it is possible to almost triplicate the cache hit probability thus increasing the read bandwidth while reducing the average latency. It is worth to highlight that this performance improvement marginally impacts the QoS, since it is related to the worst case (usually a read operation performed on a NAND Flash die.)

Summing up, the use of a DRAM cache offers advantages in terms of bandwidth, latency, and reliability. The design of an application specific SSD, in addition, can be optimized if the LBA space distribution is known, in order to reduce the DRAM size. Therefore, the drive design must be done concurrently with the application for which it represents the storage element. This concept, leading to the development of Software Defined Flash (SDF, [29]), will be extensively treated in Section VI.

TABLE III
NAND/DRAM SIZE RATIO AND SSD PERFORMANCE FOR THE SAME CONFIGURATION OF FIG. 19 AS A FUNCTION OF THE LBA SPACE DISTRIBUTIONS OF FIG. 22.

Case	a)	b)	c)
NAND/DRAM size ratio	256	50	15
Cache hit probability [%]	15.3	15.3	15.3
Read Bandwidth [kIOPS]	367	364	365
Average latency [μs]	173	175	175
QoS [ms]	0.98	1.27	1.29

TABLE IV
NAND/DRAM SIZE RATIO AND SSD PERFORMANCE FOR THE SAME CONFIGURATION OF FIG. 19 AS A FUNCTION OF THE LBA SPACE DISTRIBUTION OF THE B) CASE OF FIG. 22.

NAND/DRAM size ratio	50	15
Cache hit probability [%]	15.3	42.1
Read Bandwidth [kIOPS]	364	536
Average latency [μs]	175	118
QoS [ms]	1.27	1.19

V. CRITERIA FOR OPTIMAL HOST INTERFACE SELECTION

The host interface represents the link between the SSD controller and the host where the application is running. Differently from the SSD controller that is fully customized, the physical structure of the communication interface follows consolidated standards. At the moment, the used interfaces are SATA [8] (mainly for consumer applications), SAS [24], and PCIe [25] (for enterprise environments).

The correct choice of the host interface represents a crucial aspect along the drive design phase since it allows guaranteeing that the SSD controller is used in optimal conditions. In a traditional design approach for general purpose SSDs, where both controller and host interface are chosen separately without any knowledge of the final application, the constraint of selecting a host interface able to guarantee a bandwidth $B_{hi} \geq B_{cont}$ (where B_{hi} is the maximum bandwidth of the host interface) at the lowest cost represents the standard approach, whereas a host interface whose $B_{hi} < B_{cont}$ would act as a bottleneck limiting the SSD performance. A detailed analysis of the impact of the host interface on the SSD's performance has been presented in [26].

If the application to be run on the host is known, a different approach can be adopted. It must be taken into account that the design of a fully customized SSD controller is much more expensive with respect to that of the host interface, which follows well defined standards [86]. By considering this economic aspect, it is convenient to design an SSD controller with top performance (rather than a family of controllers with different quality metrics) and to operate at the host interface level to satisfy the application requirements. As an example, if the controller has been designed to sustain a certain B_{cont}^{th} and the application requires a lower bandwidth B_{app}, an interface satisfying the condition

$$B_{app} \leq B_{hi} \leq B_{cont}^{th} \tag{8}$$

can be selected, confirming that the ideal host interface must be chosen on the basis of the application and, therefore, on the drive use. In such a way, with a single SSD controller design, different application requirements can be satisfied by using different host interfaces. Such methodology allows reducing the controller bandwidth to match that of the application and lowering the design cost of the SSD controller. In addition,

it allows also reducing the drive power consumption since, operating at a lower throughput, a lower number of NAND Flash dies are activated simultaneously.

An evolution of this design methodology, envisaging a single controller associated to different interfaces as a function of the application, considers an unique combination of SSD controller and host interface. In this case, each block is able to provide the maximum theoretical performance. The effective performance, however, can be tuned dynamically at software level by acting on the SSD's firmware and especially on the command queue depths, which can be modified during the normal execution. An example of this methodology can be found in [92], [93] where the SSD controller is able to automatically limit the performance of the drive depending on the allowed power consumption or on the thermal dissipation level. Such an approach, that calls for the design of a single block embedding the SSD controller and the host interface, however, implies a higher design cost for the development of a controller whose hardware resources can be programmed by the user.

VI. Research scenario opened by hardware-software co-design for high-performance SSDs

In the last 40 years all software applications and Operating Systems (OS) which make use of persistent storage architectures have been designed to work with HDDs [1]. However, SSDs are physically and architecturally different from HDDs so that they need to execute the FTL algorithm to translate host commands [3], [4], [5]. Basically, the main role of FTL is to mimic the behavior of a traditional HDD and to enable the usage of SSDs in any electronic system without acting on the software stack. Besides this translation operation, SSD controllers have to run garbage collection, command scheduling algorithms, data placement schemes, wear-leveling, and errors correction. All these routines, even if on the one hand allow a "plug and play" connection of the SSD with traditional hardware and software, on the other hand they limit actual SSD performance. The main drawback of FTL is the *Garbage Collection* (GC), that is performed when valid pages belonging to a block to be erased are read and written in a different block. Such an operation, that is time and power consuming, reduces both drive bandwidth and NAND Flash reliability [84]. In the enterprise market and hyperscale data centers, performance and reliability losses induced by GC are not tolerable.

To deal with the above mentioned challenges, software developers of hyperscale data centers have shown, in the past few years, a growing interest for *Software-Defined Flash* (SDF) [29]. In this kind of environments the driving forces in the design of computational nodes are reliability and high performance: therefore, even the I/O management has to be re-architected. SDF leverages a new SSD design approach called *Host-Based* FTL (HB-FTL) which allows the host system to:

- optimize the host payload, i.e., the amount of data read/written with a single command and hence relieve the SSD from any host command translation or manipulation;
- remove the GC related to FTL execution;
- execute the FTL directly on top of its computational node (*Open-Channel* architecture [94]).

A. HB-FTL operations

HB-FTL considers the migration of all FTL routines from the SSD to a more powerful processor located outside the SSD. To this purpose, the processor must be able to issue commands to be interpreted directly by the NAND Flash dies, such as read, program and, especially, erase [95]. In this context, a new protocol called *Light NVME* (LNVME) [96] allows a native communication between NAND memories and the external processor. Thanks to this protocol, the FTL can be implemented and executed by the external processor such as the host where the application is running.

A first advantage provided by this approach concerns the optimization of the host payload. With this respect, since ECC coding/decoding operate on an entire memory page, read/write operations on a NAND Flash page must follow the constrains imposed by the ECC itself. As an example, consider a NAND Flash memory whose page size is 4 kB and a host reading/writing data on a 512 B basis.

Write operations are performed on the NAND memories only when eight 512 B data chunks have been transferred by the host. However, the host considers as accomplished a write operation when the SSD has acknowledged the data acquisition. If a power fail occurs between the data load and the effective storage in the nonvolatile layer, data are considered as lost. To avoid this occurrence, dedicated solutions such as supercapacitors [97] or the introduction of emerging nonvolatile technologies, such as MRAM [98], replacing DRAM buffers can be adopted [99], [100]. On the contrary, a NAND memory page is read every time the host requires even a single chunk. Therefore, even if only 512 B are requested by the host, the entire 4 kB page is read and decoded by the ECC. It is clear that, in this case, the SSD is operating at 1/8 of its theoretical read bandwidth.

To improve the SSD performance and to better exploit its internal resources, it is convenient to co-design the application payload with the ECC engine. The optimal solution is achieved by data chuncks that are an integer multiple of the actual ECC codeword.

A more powerful approach takes into account that in HB-FTL-based SDF both the application and the FTL are processed in the same software environment [101]. Therefore, they can be co-designed in order to optimize the access pattern to the nonvolatile memory. As an example, the application can be designed to perform only sequential accesses to the storage medium, respecting the physical *in-order-program* of NAND Flash memories [102]. By following this approach, the actual access to the NAND Flash dies is *block-based* rather than *page-based* which is typical of random write accesses. By moving the write granularity from pages to blocks, GC is no longer necessary. In addition, by serializing the write traffic to the NAND Flash memories, the write bandwidth is maximized.

TABLE V
HGST SN150 ULTRASTAR CONFIGURATION.

Parameter	Configuration
Channels	16
Dies per channel	16
SSD Capacity	3.2 TB
NAND Flash die	128 Gb Toshiba A19 eMLC
Host interface	PCI-Express GEN3x4

B. The Open-Channel architecture

The Open-Channel architecture [94], [101] allows implementing the management of HB-FTL-based SDF.

Fig. 23 sketches a template architecture that can be modeled by Open-Channel. Basically, thanks to the PCI-Express interconnection and the LNVME protocol, a bunch of NAND Flash cards can establish a peer-to-peer communication with the host processor without requesting any specific management to the SSD controller [103]. In this architecture "NAND Flash cards" are not standard SSDs because, besides a simple I/O processor, a channel controller for NAND addressing and an ECC engine, they do not embody any complex processor, DRAM or even FTL (see Fig. 24). As a consequence, data read/write from/to these cards have to be considered as the raw output/input of NAND memories without any further manipulation.

Fig. 25 shows the effectiveness of HB-FTL with respect to a standard FTL in increasing the SSD performance. To this purpose the HGST SN150 Ultrastar SSD [104], whose configuration is reported in Table V, has been compared with a simulated drive feauturing a HB-FTL approach and the same SSD configuration.

The comparison has been performed for different mixed workloads, from a 100% 4 kB random read, 0 % random write to a 0 % random read, 100 % 4kB random write.

All results show that in a standard FTL-based SSD performance decreases with the write percentage, whereas in a HB-FTL-based SSD performance is mostly independent from the write percentage. This result is due to the absence of the GC algorithm that strongly affects standard FTL-based SSDs.

Another architecture that can fully exploit the Open-

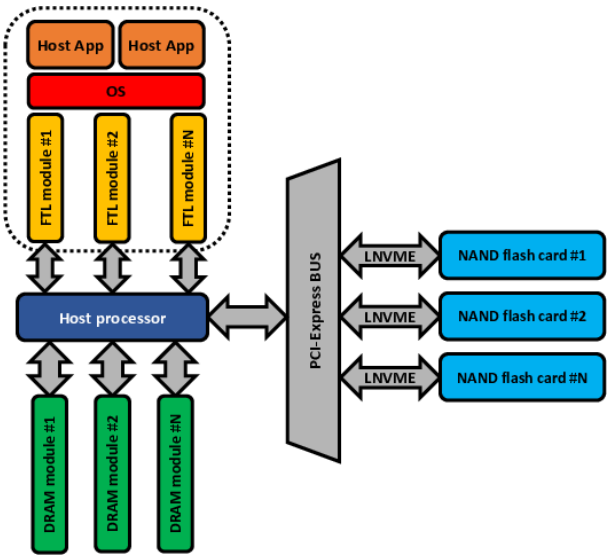


Fig. 23. Reference architecture modeled by the Open-Channel storage layer when the host processor is used for HB-FTL execution. More than one NAND Flash card are connected to the PCI-Express bus. Different FTL modules are executed by the host processor.

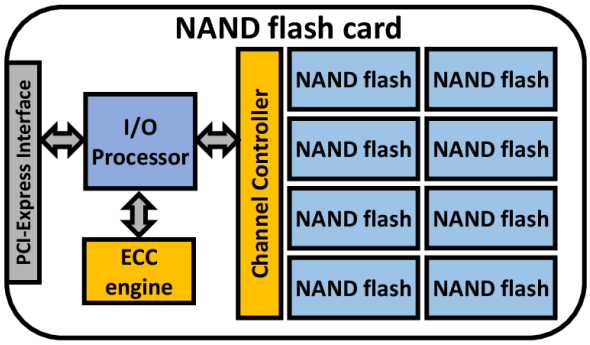
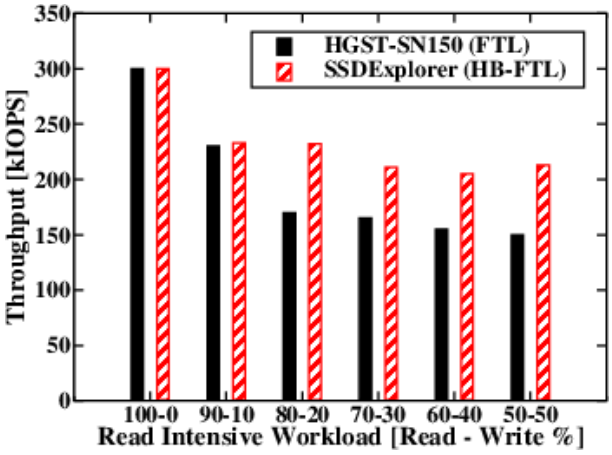
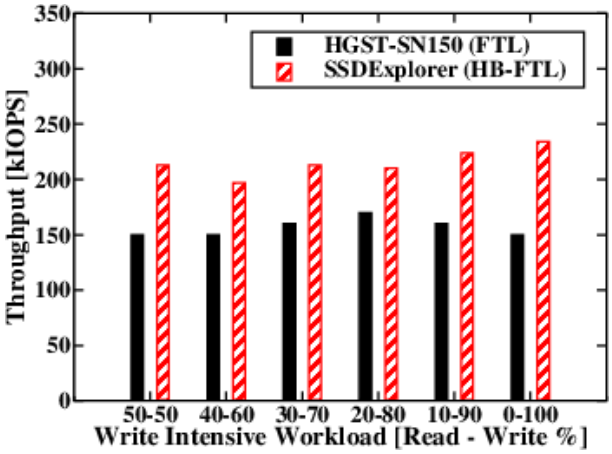


Fig. 24. Schematic of a NAND Flash card used in the Open-Channel storage system.



(a)



(b)

Fig. 25. Throughput (expressed in kIOPS) of HGST SN150 Ultrastar SSD architecture compared to that of a simulated HB-FTL-based drive with the same configuration: (a) read intensive and (b) write intensive workloads. A queue depth of 32 commands is used. Simulations have been performed with SSDExplorer [26].

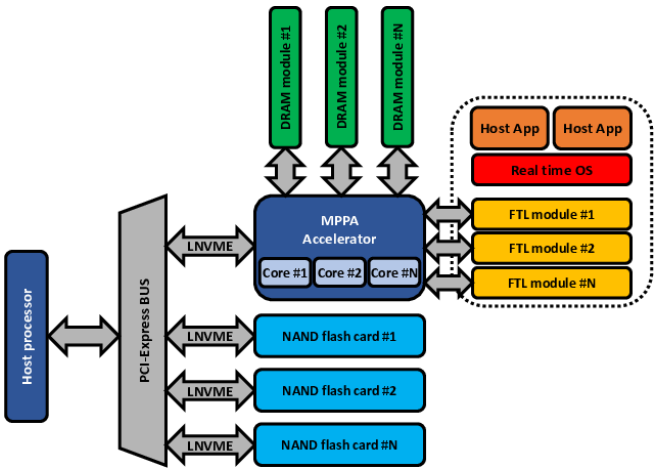


Fig. 26. Reference architecture modeled by the Open-Channel storage layer when a MPPA is used for HB-FTL execution. Besides the NAND Flash cards, the PCI-Express bus is connected to a MPPA accelerator executing different FTL modules.

Channel concept and the LNVME protocol relies on the usage of a dedicated accelerator in the form of a *Multi-Purpose Processing Array* (MPPA) [105], [106], as shown in Fig. 26. This solution allows the reduction of the host I/O command submission/completion timings.

These delays are strictly related to the host's processing capabilities and they represent the time spent by the host to execute the LNVME driver and the OS file system for each submitted/completed I/O. It has been demonstrated that the performance of nowadays SSDs is heavily affected by the I/O submission/completions timings [107]. Moreover, in most recent architectures like the one based on the 3D Xpoint technology [108], these delays can even represent the actual bottleneck of the whole storage layer, whose IOPS are limited by the host system itself. As a consequence, reducing these timings is the key for designing ultra-high performance storage systems.

A possible solution to this problem is to switch the LNVME protocol from an interrupt-driven I/O completion mechanism to a polling-driven approach. Basically, in standard SSDs, when an I/O is completed, the Flash controller sends an interrupt to the host notifying that the transaction is ready to be transferred/processed. After that, the host can submit another command to the drive because the submission of an I/O is driven by a completion event. In theory this approach requires that the host takes action only when I/Os are submitted/completed, but in practice it introduces long processing delays because of the OS interrupt service routines [107]. Polling the I/O completion events, on the contrary, can minimize the above mentioned processing timings. It requires, however, that the host system monitors continuously the I/Os, thus wasting part of its processing capabilities. In light of all these considerations, moving the whole submission/completion process to a dedicated MPPA represents a good solution which can offload the host system and, at the same time, exploit the full performance of the NAND Flash cards.

Fig. 27 shows the bandwidth comparison among the HGST SN150 Ultrastar SSD [104] and two simulated drives with the same architectural configuration, the former executing the FTL on the host (HB-FTL), the latter on a dedicated MPPA (HB-FTL-MPPA). Five different MPPA acceleration levels have been considered, ranging from a 0% speed-up of the host up to the 95%. The maximum I/O acceleration was imposed by the hardware limitations introduced by the PCI-Express bus.

As it can be seen the HB-FTL-MPPA is able to heavily improve performance in all the tested conditions, but it is extremely effective when write intensive workloads are considered. This phenomenon is related to the fact that program operations on NAND flash cards still follow a *Write-Through* (WT) [109] caching policy; therefore, once the data payload is transferred to the target card, a completion packet goes immediately back to the MPPA. At this point it is clear that, since the access time of WT buffers is in the order of a few μs, the reduction of the I/O submission/completion timings impacts the overall transfer time of the payload. This is also true for read operations, but because of the pipelining and queuing effects of the NAND flash cards, the overall improvement is not so evident.

These considerations push towards a new SSD design methodology: a complete virtualization of the storage backbone. In fact, both HB-FTL and Open-Channel allow to virtually separating the internal resources of the SSD (like channels and targets), providing a clear and straight path to OS data partitioning.

VII. Conclusions

An SSD design aimed at optimizing its performance must follow a Bottom-Up approach, since most of the design constraints are strongly related to the performance and reliability of the nonvolatile storage medium. A detailed knowledge of the memory performance degradation caused by oxide wear-out and related to endurance phenomena, data retention and read disturbs is mandatory to efficiently design the whole SSD architecture. RBER represents the main figure of merit driving the designers choices. The knowledge of RBER and, in particular, of its evolution over stress and time, allows selecting the most effective methodology and architecture in order to extend as much as possible the memory working window. Since RBER increases with aggressive technology scaling, the use of LDPC codes now represents the preferable solution for ECC engines. Once the NAND Flash memories (together with the knowledge of their RBER) and the most appropriate ECC algorithm (together with either read retry techniques or LDPC soft decisions) have been selected, the design of the SSD controller must be based on several aspects:

- the ECC architecture, trading off between performance (bandwidth, latency, power consumption) and area occupation;
- the number of memory channels, trading off again between performance and area occupation;
- the number of memory dies per channel, that is generally a power of 2;
- the appropriate command management, maximizing the number of active dies and hence the SSD bandwidth,

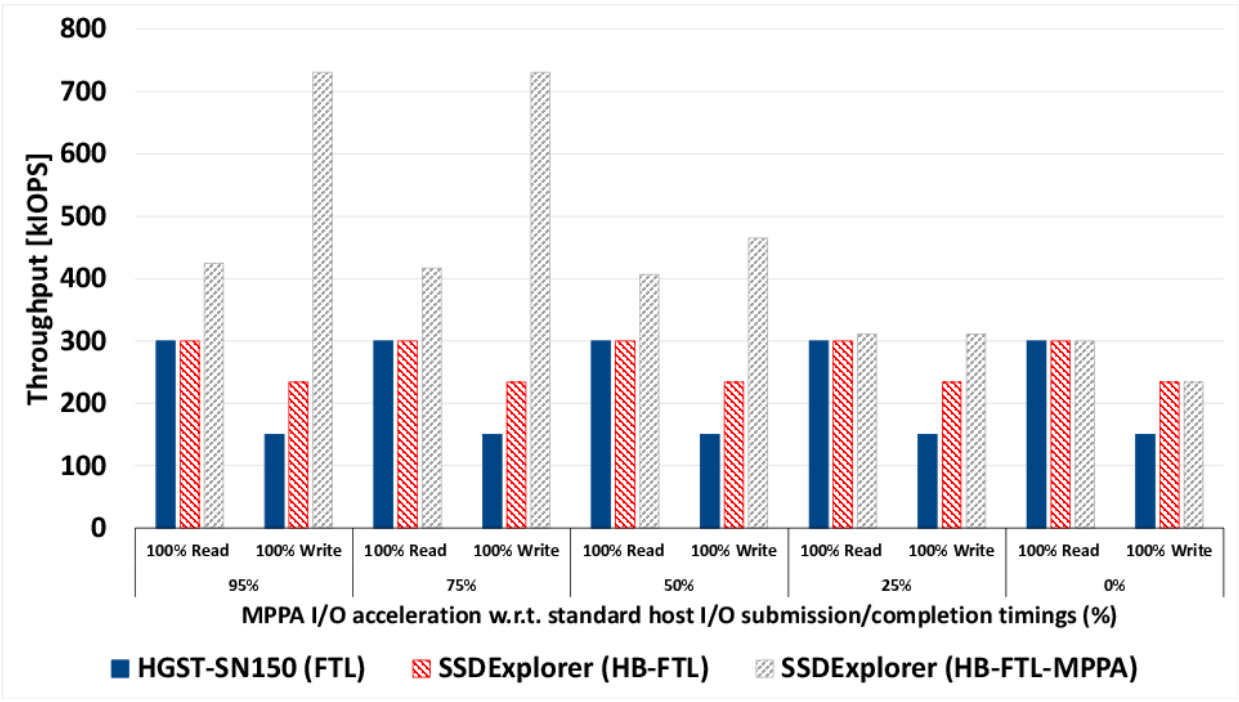


Fig. 27. Throughput (expressed in kIOPS) of HGST SN150 Ultrastar SSD compared with the simulated HB-FTL or HB-FTL-MPPA for different percentages of host I/O submission/completion timings accelerations. Command queue depth is 32. Simulations have been performed with SSDExplorer [26].

whereas limiting as much as possible the maximum latency (*i.e.* the QoS) by leveraging the head of line blocking concept;

- the introduction of a DRAM data cache buffer able to reduce the number of access operations to the NAND Flash memories, thus increasing SSD bandwidth while reducing NAND Flash degradation effects;
- the choice of the most suitable host interface able to guarantee the performance requested by the host applications.

To further improve the performance of next generation SSDs to be used in hyperscaled environments it is possible to leverage new approaches, like SDF, exploiting hardware/software co-design of the SSD controller architecture and of the host applications.

REFERENCES

[1] G. Wong, "SSD Market Overview," in *Inside Solid State Drives (SSDs)*, R. Micheloni, A. Marelli, and K. Eshghi, Ed. Springer, 2012, pp. 1–17.

[2] Semiconductor Industry Association, "International Technology Roadmap for Semiconductors," [Online]. Available: {http://www.semiconductors.org/main/2015_international_technology_roadmap_for_semiconductors_itrs/}, 2015.

[3] D. Liu, Y. Wang, Z. Qin, Z. Shao, and Y. Guan, "A Space Reuse Strategy for Flash Translation Layers in SLC NAND Flash Memory Storage Systems," *IEEE Trans. on VLSI systems*, vol. 20, no. 6, pp. 1094–1107, June 2012.

[4] T. Wang, D. Liu, Y. Wang, and Z. Shao, "FTL2: A Hybrid Flash Translation Layer with Logging for Write Reduction in Flash Memory," *ACM SIGPLAN Not.*, vol. 48, no. 5, pp. 91–100, 2013.

[5] Y. H. Chang, P. C. Huang, P. H. Hsu, L. J. Lee, T. W. Kuo, and D. Du, "Reliability Enhancement of Flash-Memory Storage Systems: An Efficient Version-Based Design," *IEEE Trans. on Computers*, vol. 62, no. 12, pp. 2503–2515, 2013.

[6] JEDEC Org., "JESD 22- A 117 document," Oct. 2011.

[7] R. Micheloni, A. Marelli and R. Ravasio, "Basic Coding Theory," in *Error Correction Codes for Non-Volatile Memories*, R. Micheloni, A. Marelli, and R. Ravasio, Ed. Springer, 2008, pp. 1–33.

[8] Serial ATA International Organization, "SATA Revision 3.0 Specifications," [Online]. Available: www.sata-io.org.

[9] Intel Inc., "Intel Solid-State Drive DC S3500 Series Quality of Service," [Online]. Available: {http://www.intel.com/content/www/us/en/solid-state-drives/ssd-dc-s3500-spec.html}, p. 9, 2013.

[10] A. Grossi, L. Zuolo, F. Restuccia, C. Zambelli, and P. Olivo, "Quality-of-Service Implications of Enhanced Program Algorithms for Charge-Trapping NAND in Future Solid-State Drives," *IEEE Trans. on Devices and Materials Reliability*, vol. 15, no. 3, pp. 363–369, 2015.

[11] S. Aritome, *NAND Flash Memory Technologies*. Wiley-IEEE Press, 2016.

[12] C. Zambelli and P. Olivo, "SSD Reliability," in *Inside Solid State Drives (SSDs)*, R. Micheloni, A. Marelli, and K. Eshghi, Ed. Springer, 2013, pp. 203–231.

[13] J. D. Lee, J. H. Choi, D. Park, and K. Kim, "Degradation of Tunnel Oxide by FN Current Stress and its Effects on Data Retention Characteristics of 90 nm NAND Flash Memory Cells," in *Proc. Int. Rel. Phys. Symp.*, Mar. 2003, pp. 497–501.

[14] N. Mielke, H. Belgal, I. Kalastirsky, P. Kalavade, A. Kurtz, Q. Meng, N. Righos, and J. Wu, "Flash EEPROM Threshold Instabilities due to Charge Trapping during Program/Erase Cycling," *IEEE Trans. on Devices and Materials Reliability*, vol. 4, no. 3, pp. 335–344, 2004.

[15] N. Mielke, T. Marquart, N. Wu, J. Kessenich, H. Belgal, E. Schares, F. Trivedi, E. Goodness, and L. R. Nevill, "Bit Error Rate in NAND Flash Memories," in *Proc. Int. Rel. Phys. Symp.*, Apr. 2008, pp. 9–19.

[16] M. Lenzlinger and E. H. Snow, "FowlerNordheim Tunneling into Thermally Grown SiO_2," *J. Appl. Phys.*, vol. 40, pp. 278 – 283, 1969.

[17] K. D. Suh, B. H. Suh, Y. H. Um, J. K. Kim, Y. J. Choi, Y. N. Koh, S. S. Lee, S. C. Kwon, B. S. Choi, J. S. Yum, J. H. Choi, J. R. Kim, and H. K. Lim, "A 3.3 V 32 Mb NAND Flash Memory with Incremental Step Pulse Programming Scheme," in *IEEE Int. Solid-State Circuits Conf.*, Feb. 1995, pp. 128–129.

[18] K. Fukuda, Y. Watanabe, E. Makino, K. Kawakami, J. Sato, T. Takagiwa, N. Kanagawa, H. Shiga, N. Tokiwa, Y. Shindo, T. Ogawa, T. Edahiro, M. Iwai, O. Nagao, J. Musha, T. Minamoto, Y. Furuta, K. Yanagidaira, Y. Suzuki, D. Nakamura, Y. Hosomura, R. Tanaka, H. Komai, M. Muramoto, G. Shikata, A. Yuminaka, K. Sakurai, M. Sakai, H. Ding, M. Watanabe, Y. Kato, T. Miwa, A. Mak, M. Nakamichi, G. Hemink, D. Lee, M. Higashitani, B. Murphy, B. Lei, Y. Matsunaga, K. Naruke, and T. Hara, "A 151-mm^2 64-Gb 2 Bit/Cell NAND Flash Memory in 24-nm CMOS Technology," *IEEE J. of Solid State Circuit*, vol. 47, no. 1, pp. 75–84, 2012.

[19] K. T. Park, O. Kwon, S. Yoon, M. H. Choi, I. M. Kim, B. G. Kim, M. S. Kim, Y. H. Choi, S. H. Shin, Y. Song, J. Y. Park, J. E. Lee, C. G. Eun, H. C. Lee, H. J. Kim, J. H. Lee, J. Y. Kim, T. M. Kweon, H. J. Yoon, T. Kim, D. K. Shim, J. Sel, J. Y. Shin, P. Kwak, J. M. Han, K. S. Kim, S. Lee, Y. H. Lim, and T. S. Jung, "A 7MB/s 64Gb 3-Bit/Cell DDR NAND Flash Memory in 20nm-Node Technology," in *IEEE Int. Solid-State Circuits Conf.*, Feb. 2011, pp. 212–213.

[20] C. Trinh, N. Shibata, T. Nakano, M. Ogawa, J. Sato, Y. Takeyama, K. Isobe, B. Le, F. Moogat, N. Mokhlesi, K. Kozakai, P. Hong, T. Kamei, K. Iwasa, J. Nakai, T. Shimizu, M. Honma, S. Sakai, T. Kawaai, S. Hoshi, J. Yuh, C. Hsu, T. Tseng, J. Li, J. Hu, M. Liu, S. Khalid, J. Chen, M. Watanabe, H. Lin, J. Yang, K. McKay, K. Nguyen, T. Pham, Y. Matsuda, K. Nakamura, K. Kanebako, S. Yoshikawa, W. Igarashi, A. Inoue, T. Takahashi, Y. Komatsu, C. Suzuki, K. Kanazawa, M. Higashitani, S. Lee, T. Murai, K. Nguyen, J. Lan, S. Huynh, M. Murin, M. Shlick, M. Lasser, R. Cernea, M. Mofidi, K. Schuegraf, and K. Quader, "A 5.6MB/s 64Gb 4b/Cell NAND Flash Memory in 43nm CMOS," in *IEEE Int. Solid-State Circuits Conf.*, Feb. 2009, pp. 246–247.

[21] L. Zuolo, C. Zambelli, R. Micheloni, D. Bertozzi, and P. Olivo, "Analysis of Reliability/Performance Trade-off in Solid State Drives," in *Proc. Int. Rel. Phys. Symp.*, June 2014, pp. 4B.3.1–4B.3.5.

[22] K. Zhao, W. Zhao, H. Sun, X. Zhang, N. Zheng, and T. Zhang, "LDPC-in-SSD: Making Advanced Error Correction Codes Work Effectively in Solid State Drives," in *USENIX Conf. on File and Storage Technologies*, 2013, pp. 243–256.

[23] L. Zuolo, C. Zambelli, R. Micheloni, S. Galfano, M. Indaco, S. D. Carlo, P. Prinetto, P. Olivo, and D. Bertozzi, "SSDExplorer: a Virtual Platform for Fine-Grained Design Space Exploration of Solid State Drives," in *Proc. of IEEE Eur. Design Automation and Test Conf.*, Mar. 2014, pp. 1–6.

[24] Seagate Technology LLC, "Serial Attached SCSI (SAS)," [Online]. Available: http://www.seagate.com/staticfiles/support/disc/manuals/Interface%20manuals/100293071c.pdf, 2009.

[25] PCI-SIG Ass., "PCI Express Base 3.0 Specification," [Online]. Available: {http://www.pcisig.com/specifications/pciexpress/base3/}, 2013.

[26] L. Zuolo, C. Zambelli, R. Micheloni, M. Indaco, S. Di Carlo, P. Prinetto, D. Bertozzi, and P. Olivo, "SSDExplorer: A Virtual Platform for Performance/Reliability-Oriented Fine-Grained Design Space Exploration of Solid State Drives," *IEEE Trans. Computer-Aided Design*, vol. 34, no. 10, pp. 1627–1638, 2015.

[27] C. Zambelli, M. Indaco, M. Fabiano, S. D. Carlo, P. Prinetto, P. Olivo, and D. Bertozzi, "A Cross-Layer Approach for new Reliability-Performance Trade-offs in MLC NAND Flash Memories," in *Proc. of IEEE Eur. Design Automation and Test Conf.*, Mar. 2012, pp. 881–886.

[28] INTEL inc., "Intel X18-M X25-M SATA Solid State Drive. Enterprise Server/Storage Applications," [Online]. Available: {http://cache-www.intel.com/cd/00/00/42/52/425265_425265.pdf}.

[29] J. Ouyang, S. Lin, S. Jiang, Z. Hou, Y. Wang, and Y. Wang, "SDF: Software-defined Flash for Web-scale Internet Storage Systems," in *Proc. ACM Int. Conf. on Architectural Support for Programming Languages and Operating Systems*, Mar. 2014, pp. 471–484.

[30] F. Masuoka, M. Momodomi, Y. Iwata, and R. Shirota, "New Ultra High Density EPROM and Flash EEPROM with NAND Structure Cell," in *IEDM Tech. Dig.*, Dec. 1987, pp. 552–555.

[31] D. Kahng and S. Sze, "A Floating Gate and its Application to Memory Devices," *Bell Syst. Tech. J.*, vol. 46, p. 1288, 1967.

[32] C. Zambelli, A. Chimenton, and P. Olivo, "Reliability Issues of NAND Flash Memories," in *Inside NAND Flash Memories*, R. Micheloni, L. Crippa, and A. Marelli, Ed. Springer, 2010, pp. 89–113.

[33] A. Chimenton, C. Zambelli, and P. Olivo, "A Statistical Model of Erratic Behaviors in Flash Memory Arrays," *IEEE Trans. Electron Devices*, vol. 58, no. 11, pp. 3707–3711, 2011.

[34] A. Torsi, Y. Zhao, H. Liu, T. Tanzawa, A. Goda, P. Kalavade, and K. Parat, "A Program Disturb Model and Channel Leakage Current Study for sub-20 nm NAND Flash Cells," *IEEE Trans. Electron Devices*, vol. 58, no. 1, pp. 11–16, 2011.

[35] M. Momodomi, T. Tanaka, Y. Iwata, Y. Tanaka, H. Oodaira, Y. Itoh, R. Shirota, K. Ohuchi, and F. Masuoka, "A 4 Mb NAND EEPROM with Tight Programmed Vt Distribution," *IEEE J. of Solid State Circuit*, vol. 26, no. 4, pp. 492–496, 1991.

[36] G. J. Hemink, T. Tanaka, T. Endoh, S. Aritome, and R. Shirota, "Fast and Accurate Programming Method for Multi-Level NAND EEPROMs," in *VLSI Symp. on Tech.*, Jun. 1995, pp. 129–130.

[37] C. Monzio Compagnoni, A. S. Spinelli, R. Gusmeroli, S. Beltrami, A. Ghetti, and A. Visconti, "Ultimate Accuracy for the NAND Flash Program Algorithm Due to the Electron Injection Statistics," *IEEE Trans. Electron Devices*, vol. 55, no. 10, pp. 2695–2702, 2008.

[38] C. Monzio Compagnoni, R. Gusmeroli, A. S. Spinelli, and A. Visconti, "Analytical Model for the Electron-Injection Statistics During Programming of Nanoscale NAND Flash Memories," *IEEE Trans. Electron Devices*, vol. 55, no. 11, pp. 3192–3199, 2008.

[39] M. Momodomi, Y. Itoh, R. Shirota, Y. Iwata, R. Nakayama, R. Kirisawa, T. Tanaka, S. Aritome, T. Endoh, K. Ohuchi, and F. Masuoka, "An Experimental 4-Mbit CMOS EEPROM with a NAND-Structured Cell," *IEEE J. of Solid State Circuit*, vol. 24, no. 5, pp. 1238–1243, 1989.

[40] A. Chimenton, P. Pellati, and P. Olivo, "Analysis of Erratic Bits in Flash Memories," *IEEE Trans. on Devices and Materials Reliability*, vol. 1, no. 4, pp. 179–184, 2001.

[41] S. Aritome, R. Kirisawa, T. Endoh, R. Nakayama, R. Shirota, K. Sakui, K. Ohuchi, and F. Masuoka, "Extended Data Retention Characteristics after more than 10^4 Write and Erase Cycles in EEPROMs," in *Proc. Int. Rel. Phys. Symp.*, Mar. 1990, pp. 259–264.

[42] R. Shirota, R. Nakayama, R. Kirisawa, M. Momodomi, K. Sakui, Y. Itoh, S. Aritome, T. Endoh, F. Hatori, and F. Masuoka, "A 2.3 μm^2 Memory Cell Structure for 16 Mb NAND EEPROMs," in *IEDM Tech. Dig.*, Dec. 1990, pp. 103–106.

[43] E. I. Vatajelu, H. Aziza, and C. Zambelli, "Nonvolatile Memories: Present and Future Challenges," in *Int. Design and Test Symposium*, Dec. 2014, pp. 61–66.

[44] Y. Park and D. K. Schroder, "Degradation of Thin Tunnel Gate Oxide under Constant Fowler-Nordheim Current Stress for a Flash EEPROM," *IEEE Trans. Electron Devices*, vol. 45, no. 6, pp. 1361–1368, 1998.

[45] T. N. Nguyen, P. Olivo, and B. Riccò, "A New Failure Mode of Very Thin ($< 50\mathring{A}$) Thermal SiO_2 Films," in *Proc. Int. Rel. Phys. Symp.*, Apr. 1987, pp. 66 – 71.

[46] P. Olivo, T. N. Nguyen, and B. Riccò, "High-Field-Induced Degradation in Ultra Thin SiO_2 Films," *IEEE Trans. Electron Devices*, vol. 351, no. 12, pp. 2259–2267, 1988.

[47] G. J. Hemink, K. Shimizu, S. Aritome, and R. Shirota, "Trapped Hole Enhanced Stress Induced Leakage Currents in NAND EEPROM Tunnel Oxides," in *Proc. Int. Rel. Phys. Symp.*, Apr. 1996, pp. 117–121.

[48] R. Yamada, Y. Mori, Y. Okuyama, J. Yugami, T. Nishimoto, and H. Kume, "Analysis of Detrap Current due to Oxide Traps to Improve Flash Memory Retention," in *Proc. Int. Rel. Phys. Symp.*, Apr. 2000, pp. 200–204.

[49] J. Lee, J. Choi, D. Park, and K. Kim, "Effects of Interface Trap Generation and Annihilation on the Data Retention Characteristics of Flash Memory Cells," *IEEE Trans. on Devices and Materials Reliability*, vol. 4, no. 1, pp. 110–117, 2004.

[50] K. Lee, M. Kang, S. Seo, D. Kang, D. H. Li, Y. Hwang, and H. Shin, "Separation of Corner Component in TAT Mechanism in Retention Characteristics of Sub 20-nm NAND Flash Memory," *IEEE Electron Device Lett.*, vol. 35, no. 1, pp. 51–53, 2014.

[51] G. Molas, D. Deleruyelle, B. De Salvo, G. Ghibaudo, M. Gely, S. Jacob, D. Lafond, and S. Deleonibus, "Impact of Few Electron Phenomena on Floating-Gate Memory Reliability," in *IEDM Tech. Dig.*, Dec. 2004, pp. 877–880.

[52] R. Gusmeroli, C. Monzio Compagnoni, A. Riva, A. S. Spinelli, A. L. Lacaita, M. Bonanomi, and A. Visconti, "Defects Spectroscopy in SiO_2 by Statistical Random Telegraph Noise Analysis," in *IEDM Tech. Dig.*, Dec. 2006, pp. 1–4.

[53] K. Fukuda, Y. Shimizu, K. Amemiya, M. Kamoshida, and C. Hu, "Random Telegraph Noise in Flash Memories - Model and Technology Scaling," in *IEDM Tech. Dig.*, Dec. 2007, pp. 169–172.

[54] C. Monzio Compagnoni, A. Spinelli, S. Beltrami, M. Bonanomi, and A. Visconti, "Cycling Effect on the Random Telegraph Noise Instabilities of NOR and NAND Flash Arrays," *IEEE Electron Device Lett.*, vol. 29, pp. 941–943, 2008.

[55] A. Chimenton, C. Zambelli, and P. Olivo, "A New Methodology for Two-Level Random-Telegraph-Noise Identification and Statistical Analysis," *IEEE Electron Device Lett.*, vol. 31, no. 6, pp. 612–614, 2010.

[56] ——, "A Statistical Model of Erratic Erase Based on an Automated Random Telegraph Signal Characterization Technique," in *Proc. Int. Rel. Phys. Symp.*, Apr. 2009, pp. 896–901.

[57] C. Zambelli, G. Koebernik, R. Ullmann, M. Bauer, G. Tempel, F. D. Tano, M. Atti, F. P. Pistone, A. Siviero, and P. Olivo, "Exposing Reliability/Performance Tradeoff in Non-Volatile Memories Through Erratic Bits Signature Classification," *IEEE Trans. on Devices and Materials Reliability*, vol. 14, no. 1, pp. 66–73, 2014.

[58] C. Zambelli, T. Vincenzi, and P. Olivo, "A Compact Model for Erratic Event Simulation in Flash Memory Arrays," *IEEE Trans. Electron Devices*, vol. 61, no. 11, pp. 3716–3722, 2014.

[59] C. Monzio Compagnoni, A. S. Spinelli, R. Gusmeroli, A. L. Lacaita, S. Beltrami, A. Ghetti, and A. Visconti, "First Evidence for Injection Statistics Accuracy Limitations in NAND Flash Constant-Current Fowler-Nordheim Programming," in *IEDM Tech. Dig.*, Dec. 2007, pp. 165–168.

[60] C. Friederich, J. Hayek, A. Kux, T. Muller, N. Chan, G. Kobernik, M. Specht, D. Richter, and D. Schmitt-Landsiedel, "Novel Model for Cell - System Interaction (MCSI) in NAND Flash," in *IEDM Tech. Dig.*, Dec. 2008, pp. 1–4.

[61] S. Satoh, G. Hemink, K. Hatakeyama, and S. Aritome, "Stress-Induced Leakage Current of Tunnel Oxide derived from Flash Memory Read-Disturb Characteristics," *IEEE Trans. Electron Devices*, vol. 45, no. 2, pp. 482–486, 1998.

[62] H. H. Wang, P. S. Shieh, C. T. Huang, K. Tokami, R. Kuo, S. H. Chen, H. C. Wei, S. Pittikoun, and S. Aritome, "A New Read-Disturb Failure Mechanism Caused by Boosting Hot-Carrier Injection Effect in MLC NAND Flash Memory," in *IEEE Int. Memory Workshop*, May 2009, pp. 1–2.

[63] J. Lee, S. Hur, and J. Choi, "Effects of Floating-Gate Interference on NAND Flash Memory Cell Operation," *IEEE Electron Device Lett.*, vol. 23, no. 5, pp. 264–266, 2002.

[64] K. T. Park, M. Kang, D. Kim, S. W. Hwang, B. Y. Choi, Y. T. Lee, C. Kim, and K. Kim, "A Zeroing Cell-to-Cell Interference Page Architecture With Temporary LSB Storing and Parallel MSB Program Scheme for MLC NAND Flash Memories," *IEEE J. of Solid State Circuit*, vol. 43, no. 4, pp. 919–928, 2008.

[65] S. Seo, H. Kim, S. Park, S. Lee, S. Aritome, and S. Hong, "Novel Negative Vt Shift Program Disturb Phenomena in 2X-3X nm NAND Flash Memory Cells," in *Proc. Int. Rel. Phys. Symp.*, 2011, pp. 6B.2.1–6B.2.4.

[66] C. Zambelli, F. Andrian, S. Aritome, and P. Olivo, "Compact Modeling of Negative Shift Disturb in NAND Flash Memories," *IEEE Trans. Electron Devices*, vol. 63, no. 4, pp. 1516–1523, 2016.

[67] I. Park, W.-G. Hahn, K.-W. Song, K. Choi, H.-K. Choi, S. Lee, C.-S. Lee, J. Song, J. Han, K. Kyoung, and Y.-H. Jun, "A New GIDL Phenomenon by Feld Effect of Neighboring Cell Transistors and its Control Solutions in sub-30 nm NAND Flash Devices," in *VLSI Symp. on Tech.*, June 2012, pp. 23–24.

[68] J. Lee, C. Lee, M. Lee, H. Kim, K. Park, and W. Lee, "A New Programming Disturbance Phenomenon in NAND Flash Memory By Source/Drain Hot-Electrons Generated By GIDL Current," in *Non-Volatile Semiconductor Memory Workshop*, Feb. 2006, pp. 31–33.

[69] S. Satoh, H. Hagiwara, T. Tanzawa, K. Takeuchi, and R. Shirota, "A Novel Isolation-Scaling Technology for NAND EEPROMs with the Minimized Program Disturbance," in *IEDM Tech. Dig.*, Dec. 1997, pp. 291–294.

[70] R. Micheloni, A. Marelli and R. Ravasio, "Cyclic Codes for Non Volatile Storage," in *Error Correction Codes for Non-Volatile Memories*, R. Micheloni, A. Marelli, and R. Ravasio, Ed. Springer, 2008, pp. 167–198.

[71] Y. Lee, H. Yoo, I. Yoo, and I.-C. Park, "6.4Gb/s Multi-Threaded BCH Encoder and Decoder for Multi-Channel SSD Controllers," in *IEEE Int. Solid-State Circuits Conf.*, Feb 2012, pp. 426–428.

[72] R. Micheloni, A. Marelli and R. Ravasio, "BCH Hardware Implementation in NAND Flash Memories," in *Error Correction Codes for Non-Volatile Memories*, R. Micheloni, A. Marelli, and R. Ravasio, Ed. Springer, 2008, pp. 199–247.

[73] S. M. Jeff Yang, "High-Efficiency SSD for Reliable Data Storage Systems," in *Flash Memory Summit*, 2012.

[74] A. Cometti, L. Huang, and A. Melik-Martirosian, "Apparatus and Method for Determining a Read Level of a Flash Memory after an Inactive Period of Time," Feb. 4 2014, US Patent 8,644,099.

[75] X. Wang, G. Dong, L. Pan, and R. Zhou, "Error Correction Codes and Signal Processing in Flash Memory," in *Flash Memories*, I. Stievano, Ed., 2011, pp. 57–82.

[76] L. Zuolo, C. Zambelli, A. Marelli, R. Micheloni, and P. Olivo, "LDPC Soft Decoding with Improved Performance in 1X-2X MLC and TLC NAND Flash-Based Solid State Drives," *IEEE Trans. on Emerging Topics in Computing*, in press, 2017.

[77] S. N. R. Motwani, Z. Kwok, "Low Density Parity Check (LDPC) Codes and the Need for Stronger ECC ," in *Flash Memory Summit*, Aug. 2011.

[78] M. Ivkovic, S. K. Chilappagari, and B. Vasic, "Eliminating trapping sets in low-density parity-check codes by using Tanner graph covers," *IEEE Trans. on Information Theory*, vol. 54, no. 8, pp. 3763–3768, 2008.

[79] A. Marelli, "False Decoding Probability (Detection) of BCH and LDPC Codes," in *Flash Memory Summit*, 2016.

[80] Y. Yamaga, C. Matsui, S. Hachiya, and K. Takeuchi, "Application Optimized Adaptive ECC with Advanced LDPCs to Resolve Trade-Off among Reliability, Performance, and Cost of Solid-State Drives," in *IEEE Int. Memory Workshop*, May 2016, pp. 1–4.

[81] L. Zuolo, C. Zambelli, P. Olivo, R. Micheloni, and A. Marelli, "LDPC Soft Decoding with Reduced Power and Latency in 1X-2X NAND Flash-Based Solid State Drives," in *IEEE Int. Memory Workshop*, May 2015, pp. 1–4.

[82] W. Lin, "Advanced Controller Technology for 3D NAND Flash," in *Flash Memory Summit*, Aug. 2016.

[83] R. Micheloni, A. Marelli, L. Crippa, A. Aldarese, L. Zuolo, C. Zambelli, and P. Olivo, "Fully Integrated SSD-NAND Characterization Flow," in *Flash Memory Summit*, 2015.

[84] X. Hu, E. Eleftheriou, R. Haas, I. Iliadis, and R. Pletka, "Write Amplification Analysis in Flash-based Solid State Drives," in *Proc. ACM Int. Systems and Storage Conf.*, May 2009, pp. 10:1–10:9.

[85] D. Rollins, "Best Practices for SSD Performance Measurement," in *Micron Technology, Inc., Technical Marketing Brief*, 2011. [Online]. Available: https://www.micron.com/~/media/documents/products/technical-marketing-brief/brief_ssd_performance_measure.pdf

[86] K. Eshghi and R. Micheloni, "SSD Architecture and PCI Express Interface," in *Inside Solid State Drives (SSDs)*, R. Micheloni, A. Marelli, and K. Eshghi, Ed. Springer, 2012, pp. 19–45.

[87] L. M. Grupp, J. D. Davis, and S. Swanson, "The Bleak Future of NAND Flash Memory," in *Proc. Usenix Int. Conference on File and Storage Technologies*, 2012, pp. 1–8.

[88] Avago Tech., "Accelerating Financial Applications Using Solid State Storage," [Online]. Available: http://docs.avagotech.com/docs/12353095, 2011.

[89] Amazon Inc., "New SSD-Backed Elastic Block Storage," [Online]. Available: https://aws.amazon.com/it/blogs/aws/new-ssd-backed-elastic-block-storage/, 2014.

[90] M. Karol, M. Hluchyj, and S. Morgan, "Input Versus Output Queueing on a Space-Division Packet Switch," *IEEE Trans. on Communications*, vol. 35, no. 12, pp. 1347–1356, 1987.

[91] E. G. Coffman Jr. and P. J. Denning, *Operating Systems Theory*. Prentice Hall Professional Technical Reference, 1973.

[92] S. Lee, T. Kim, K. Kim, and J. Kim, "Lifetime Management of Flash-based SSDs Using Recovery-aware Dynamic Throttling," in *Proc. Usenix Int. Conference on File and Storage Technologies*, 2012.

[93] R.-S. Liu, C.-L. Yang, and W. Wu, "Optimizing NAND Flash-Based SSDs via Retention Relaxation," in *Proc. Usenix Int. Conference on File and Storage Technologies*, 2012.

[94] "Open-Channel Solid State Drives," [Online]. Available: http://openchannelssd.readthedocs.org/en/latest/, 2016.

[95] A. Batwara, "Leveraging Host based Flash Translation Layer for Application Acceleration," in *Flash Memory Summit*, Aug. 2012.

[96] "Open Channel Solid State Drives NVMe Specification," [Online]. Available: http://bit.ly/2gfidpQ, 2016.

[97] Samsung Electronics Co., "Power loss Protection (PLP) - Protect your Data Against Sudden Power Loss," [Online]. Available: http://www.samsung.com/semiconductor/minisite/ssd/downloads/document/Samsung_SSD_845DC_05_Power_loss_protection_PLP.pdf, 2014.

[98] D. Apalkov, B. Dieny, and J. M. Slaughter, "Magnetoresistive Random Access Memory," *Proc. IEEE*, vol. 104, no. 10, pp. 1796–1830, 2016.

[99] Xilinx Inc., "Everspins NVMe Storage Accelerator mixes MRAM, UltraScale FPGA, delivers 1.5M IOPS," [Online]. Available: https://forums.xilinx.com/t5/Xcell-Daily-Blog/Everspin-s-NVMe-Storage-Accelerator-mixes-MRAM-UltraScale-FPGA/ba-p/733781, 2016.

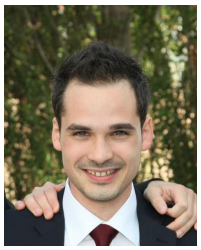
[100] G. Sun, Y. Joo, Y. Chen, D. Niu, Y. Xie, Y. Chen, and H. Li, "A Hybrid Solid-State Storage Architecture for the Performance, Energy Consumption, and Lifetime Improvement," in *IEEE Int. Symposium on High-Performance Computer Architecture*, 2010, pp. 1–12.

[101] J. Gonzalez, M. Bjrling, S. Lee, C. Dong, and Y. R. Huang, "Application-Driven Flash Translation Layers on Open-Channel SSDs," in *Non Volatile Memory Workshop*, Mar. 2016, pp. 1–2.

[102] C. Friederich, "Program and Erase of NAND Memory Arrays," in *Inside NAND Flash Memories*, R. Micheloni, L. Crippa, and A. Marelli, Ed. Springer, 2010, pp. 55–88.

[103] S. Bates, "Accelerating Data Centers Using NVMe and CUDA," in *Flash Memory Summit*, Aug. 2014.

[104] HGST Inc., "Ultrastar SN150 Series NVMe PCIe x4 Lane Half-Height Half-Length CardSolid-State Drive Product Manual," [Online]. Available: https://www.hgst.com/sites/default/files/resources/US_SN150_ProdManual.pdf.
[105] Karlay Inc., "The KalRay Multi-Purpose-Processing-Array (MPPA)," [Online]. Available: http://www.kalrayinc.com/kalray/products/#processors, 2016.
[106] P. Couvert, "High Speed IO Processor for NVMe over Fabric (NVMeoF)," in *Flash Memory Summit*, Aug. 2016.
[107] J. Yang, D. B. Minturn, and F. Hady, "When Polling is Better than Interrupt," in *USENIX Conf. on File and Storage Technologies*, Feb. 2012.
[108] F. Hady, "Wicked Fast Storage and Beyond," in *Non Volatile Memory Workshop*, Mar. 2016.
[109] J. L. Hennessy, D. A. Patterson, "B.2 Cache Performance," in *Computer Architecture: A Quantitative Approach*, Morgan Kaufmann, Ed. Elsevier, 2011.



Lorenzo Zuolo received the Laurea Magistrale degree (M.Sc.) and Ph.D. in electronic engineering from the University of Ferrara, Ferrara, Italy, in 2012 and 2016, respectively. Since June 2017 he is with Microsemi Corporation.

Cristian Zambelli received the M.Sc. and Ph.D. degrees in electronic engineering from the University of Ferrara, Ferrara, Italy, in 2008 and 2012, respectively. Since 2015 he has been an Assistant Professor with the Department of Engineering, University of Ferrara. His current research interests include the characterization, physics, and modeling of nonvolatile memories reliability and solid state drives reliability.

Rino Micheloni is Fellow at Microsemi Corporation where he currently runs the Non-Volatile Memory Lab. Early in his career, he led Flash design teams at STMicroelectronics, Hynix, and Infineon; during this time, he developed the industrys first MLC NOR device with embedded ECC technology and the industrys first MLC NAND with embedded BCH. He holds 278 patents worldwide (including 125 US patents) and he has published the following books with Springer: *Solid-State-Drives (SSDs) Modeling* (2017), *3D Flash Memories* (2016), *Inside Solid State Drives* (2013), *Inside NAND Flash Memories* (2010), *Error Correction Codes for Non-Volatile Memories* (2008), *Memories in Wireless Systems* (2008), and *VLSI-Design of Non-Volatile Memories* (2005).

Piero Olivo received the Ph.D. in electronic engineering from the University of Bologna, Bologna, Italy, in 1987. He is a Full Professor of Electronics with the University of Ferrara, Ferrara, Italy, since 1994. His research interests include the physics, the reliability and the experimental characterization of innovative non-volatile memorys and solid state drive reliability.