LDPC-in-SSD: Making Advanced Error Correction Codes Work Effectively in Solid State Drives

Kai Zhao[*], *Wenzhe Zhao*[†], *Hongbin Sun*[†], *Tong Zhang*[*], *Xiaodong Zhang*[‡], *and Nanning Zheng*[†]
[*] *ECSE Department, Rensselaer Polytechnic Institute, USA*
[†] *Xi'an Jiaotong University, P.R.China*
[‡] *Department of Computer Science and Engineering, The Ohio State University, USA*

Abstract

Conventional error correction codes (ECCs), such as the commonly used BCH code, have become increasingly inadequate for solid state drives (SSDs) as the capacity of NAND flash memory continues to increase and its reliability continues to degrade. It is highly desirable to deploy a much more powerful ECC, such as low-density parity-check (LDPC) code, to significantly improve the reliability of SSDs. Although LDPC code has had its success in commercial hard disk drives, to fully exploit its error correction capability in SSDs demands unconventional fine-grained flash memory sensing, leading to an increased memory read latency. To address this important but largely unexplored issue, this paper presents three techniques to mitigate the LDPC-induced response time delay so that SSDs can benefit its strong error correction capability to the full extent. We quantitatively evaluate these techniques by carrying out trace-based SSD simulations with runtime characterization of NAND flash memory reliability and LDPC code decoding. Our study based on intensive experiments shows that these techniques used in an integrated way in SSDs can reduce the worst-case system read response time delay from over 100% down to below 20%. With our proposed techniques, a strong ECC alternative can be used in NAND flash memory to retain its reliability to respond the continuous cost reduction, and its relatively small increase of response time delay is acceptable to mainstream application users, considering a huge gain in SSD capacity, its reliability, and the price reduction.

1 Introduction

It is well known that technology scaling makes flash memory cells subject to increasingly severe noise and distortion, which can largely degrade the NAND flash memory storage reliability and performance [1]. Therefore, increasingly powerful and sophisticated error correction and signal processing capabilities become indispensable for future solid-state drive (SSD) [2, 3]. In particular, it has been well recognized that conventional BCH codes [4, 5], which are being used in all the commercial SSDs today, become inadequate to handle continued technology scaling and must be replaced by a stronger alternative error correction code (ECC). Due to their superior error correction capability and recent success in commercial hard disk drives, low-density parity-check (LDPC) codes [6, 7] have attracted much attention [8–12] and are seriously considered as the choice of ECC for future SSDs.

LDPC code decoding is essentially an iterative belief-propagation process [13] and the decoding input are certain probability information associated with each bit in the codeword. As a result, LDPC code error correction strength heavily depends on the accuracy of the input probability information that is directly determined by the high precision NAND flash memory and read operations. Nonvolatile data storage of n-bit-per-cell NAND flash memory is realized by configuring the threshold voltage of each memory cell (i.e., floating-gate transistor) into 2^n non-overlapping storage states. Flash memory read operations aim to sense and digitally quantize the threshold voltage of each memory cell. If memory sensing uses only one quantization level between two adjacent storage states, it is called *hard-decision* memory sensing; if more than one quantization levels are used between two adjacent storage states, it is called *soft-decision* memory sensing. As to be presented in the paper, although LDPC code decoding with hard-decision memory sensing can achieve a noticeable coding gain over conventional BCH code[*], soft-decision memory sensing can significantly improve the LDPC code decoding error correction strength and hence increase the coding gain over

[*]This material is based upon work supported by the National Science Foundation under Grants No. 1162152 and 1162165

[*]BCH code decoding only demands hard-decision memory sensing.

the BCH code. Latest commercial NAND flash memory chips already support soft-decision memory sensing, e.g., Samsung 21nm 2bits/cell NAND flash memory chips can support seven quantization levels between two adjacent storage states [14]. Unfortunately, since NAND flash memory read latency is proportional to the memory sensing precision, soft-decision memory sensing directly causes an increased NAND flash memory read latency, leading to storage system read response time degradation. To the best of our knowledge based on open research and technical literature, the impact of the increased latency on overall SSD performance and techniques to minimize it have not been systematically studied. We believe the above mentioned issue is a major roadblock for us to continue increasing the capacity and decreasing the price of SSDs subject to a high reliability.

This paper presents the first study on quantitatively investigating how the use of LDPC codes may impact the storage system response time performance and, more importantly, developing techniques to minimize the system response time latency. Since flash memory cell storage reliability gradually degrades with program/erase (P/E) cycling, a hard-decision LDPC code decoding is sufficient to ensure a very low decoding failure probability (e.g., 10^{-15} and below) at the early lifetime of flash memory and can even have a reasonable decoding failure probability (e.g., 10^{-2}) at the late lifetime of flash memory. Therefore, our basic structure for LDPC-based SSDs is formed by a two-step trial-and-error procedure: upon a read request, SSDs always first execute hard-decision memory sensing and decoding, and only when hard-decision LDPC code decoding fails, execute soft-decision memory sensing and decoding. Therefore, the LDPC-induced system response time delay becomes noticeable only after a certain number of P/E cycles. We carry out extensive characterization of 25nm MLC NAND flash memory reliability and LDPC code decoding, based on which we accordingly configure an SSD simulation software tool (i.e., the SSD model [15] in DiskSim [16]) to evaluate the LDPC-induced system response time delay. Results show that, once NAND flash memory chips are heavily cycled (10,000 P/E cycling in this study), soft-decision memory sensing and LDPC code decoding will be frequently invoked and the system read response time degradation can be even over 100% for read-intensive workloads.

The results motivate us to investigate possible solutions to minimize such a significant system latency, and accordingly we propose three techniques in this paper. The first technique, called look-ahead soft-decision memory sensing, aims to execute soft-decision memory sensing immediately after hard-decision memory sensing, instead of waiting for the finish of hard-decision LDPC code decoding. By concurrently carrying out hard-decision LDPC code decoding and soft-decision memory sensing, this can reduce the latency overhead in case of a hard-decision LDPC code decoding failure. Second, motivated by the fact that LDPC code error correction strength gradually improves as we progressively increase the memory sensing precision, we propose a progressive memory sensing and LDPC code decoding design strategy. The basic idea is as follows: in case of hard-decision LDPC code decoding failure, instead of immediately invoking full-precision soft-decision memory sensing, we only progressively increase the memory sensing precision level-by-level and accordingly retry LDPC decoding. Such a fine-grained progressive sensing and decoding strategy can track the *just-enough* sensing precision at runtime and hence obviate unnecessary extra sensing latency. Motivated by the noticeable chip-to-chip reliability variation observed in our experiments with 25nm flash memory chips, we further propose a data placement interleaving technique that can reduce hard-decision LDPC code decoding failure probability and hence reduce the overall sensing and decoding latency overhead. Using representative workload traces and the SSD model [15] in DiskSim [16] and based upon extensive characterization of NAND flash memory chip reliability and LDPC code decoding, we quantitatively evaluate the effectiveness of these techniques, and results show that they can reduce the latency overhead from over 100% down to below 20%. With our proposed techniques, a strong ECC alternative can be used in NAND flash memory to retain its reliability under continuous cost deduction, and its relatively small increase of response time is acceptable to mainstream application users, considering a huge gain in SSD capacity, reliability and price.

The remainder of this paper is organized as follows. Section 2 reviews the basics of NAND flash memory and LDPC code, and presents a simple study to demonstrate the impact of LDPC codes on SSD response time. Section 3 describes the proposed three techniques for reducing LDPC-induced system response time delay. Section 4 presents SSD simulation setup and characterization results of 25nm NAND flash memory reliability and LDPC code decoding, based upon which simulations are carried out to evaluate the proposed techniques in Section 5. Section6 surveys related work and Section 7 draws the conclusions and discusses the future research directions.

2 Background and Motivations

This section first presents the basics of NAND flash memory and LDPC code, and demonstrates the significant coding benefits gained by using soft-decision LDPC

code decoding. By trace-driven simulations supported by a widely used SSD simulator, we show that a straightforward use of LDPC code with soft-decision decoding could cause a significant and unacceptable response time increase of SSD. This preliminary work motivates us to minimize this running time delay in order to truly gain the merits of LDPC in SSDs.

2.1 Basics of NAND Flash Memory

Each NAND flash memory cell is a floating gate transistor whose threshold voltage can be programmed by injecting certain amount of charges into the floating gate. Before a flash memory cell can be programmed, it must be erased (i.e., remove all the charges from the floating gate, which sets its threshold voltage to the lowest voltage window). The flash memory program/erase (P/E) cycling causes damage to the tunnel oxide of floating gate transistors in the form of charge trapping in the oxide and interface states [17–19], which directly results in threshold voltage shift and fluctuation and hence gradually degrades memory device noise margin. Since the significance of these noises grows with the trap density and trap density grows with P/E cycling, NAND flash memory cell noise margin monotonically degrades with P/E cycling. In practice, this sets a NAND flash memory P/E cycling endurance limit, beyond which memory cell noise margin degradation can no longer be accommodated by the memory system fault tolerance capability. As technology continues to scale, floating gate transistor becomes smaller and hence can only hold less amount of electrons for nonvolatile data storage. As a result, NAND flash memory cells are more sensitive to P/E cycling and other distortion sources [20], leading to continuous degradation of almost all the important reliability and performance metrics [1].

NAND flash memory accesses are performed in the unit of page, and all the memory cells within one page are driven by a single word-line. Aiming to digitally quantize the threshold voltage of all the memory cells in one page, memory sensing is carried out recursively in a level-by-level manner. It can be illustrated in Fig. 1: Assume we want to carry out a 3-level soft-decision sensing at the quantization voltage levels of V_1, V_2, and V_3 (i.e., to determine the threshold voltage of each memory cell fall into which region out of the four regions I, II, III, and IV). The word-line voltage is first raised to V_1 and check whether the threshold voltage of each memory cell is above or below V_1 through a bit-line charge-discharge cycle. If the threshold voltage of one memory cell is below V_1, then we know that it falls into the region I, otherwise further sensing is needed to determine its region. Then the word-line voltage is further raised to V_2 and check whether the threshold voltage of each memory cell is above or below V_2 through another bit-line charge-discharge cycle. If the threshold voltage of a memory cell is above V_1 but below V_2, we know it falls into the region II. Finally the word-line voltage is raised to V_3 and we can distinguish between the region III and IV through the 3rd bit-line charging-discharging cycle. Clearly, this process shows that NAND flash memory sensing latency is linearly proportional to the number of sensing quantization levels.

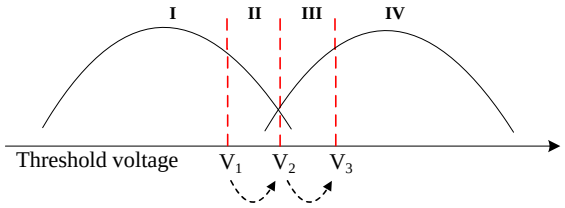

Figure 1: Illustration of 3-level soft-decision sensing: three bit-line charge-discharge cycles at word-line voltages of V_1, V_2, and V_3.

2.2 Basics of LDPC Codes

Invented by Gallager [6] in 1962, LDPC codes have been largely neglected by the scientific community for several decades until the remarkable success of Turbo codes in 3G wireless communication that led to the re-discovery of LDPC codes in 1996 by MacKay and Neal [21] and Wiberg [22]. LDPC codes have attracted tremendous research attention in both academia and industry over the past 15 years, and have been widely adopted in various data communication and storage systems, e.g., DVB (digital video broadcasting), WiFi, WiMAX, 10G Ethernet, and more recently hard disk drives.

An LDPC code is defined as the null space of an $M \times N$ sparse parity check matrix. Its sparse parity check matrix can be represented by a bipartite graph, between M check (or constraint) nodes in one set and N variable (or message) nodes in the other set. An LDPC code can be decoded by iterative belief-propagation (or message-passing) decoding algorithms [7, 23] that directly matches the code bipartite graph as illustrated in Fig. 2: After each variable node is initialized with the the input channel message (i.e., the probability information associated with each bit), the decoding messages are iteratively computed by all the variable nodes and check nodes and exchanged through the edges between the neighboring nodes.

Error correction strength of LDPC code decoding strongly depends on the accuracy of the input probability information of each bit, which further depends on memory sensing precision in the context of NAND flash memory. Let us use the following example to illustrate the im-

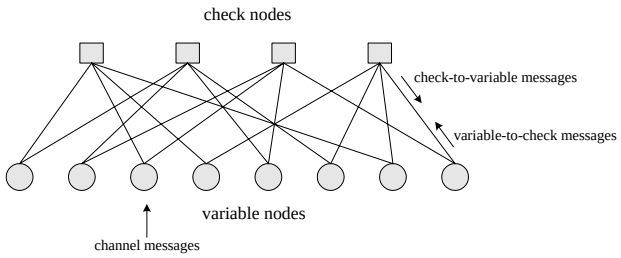

Figure 2: Iterative belief-propagation decoding process based on LDPC code bipartite graph.

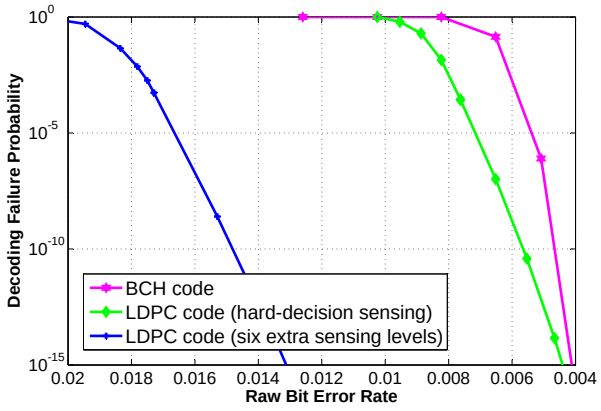

Figure 3: Simulated decoding failure probability vs. NAND flash memory raw bit error rate performance of hard-decision and soft-decision LDPC decoding and BCH code decoding. Both LDPC code and BCH code protect each 4kB user data with 512B coding redundancy.

pact of sensing precision on the error correction strength. Assume each chunk of 4kB user data is protected by a rate-8/9 LDPC code (i.e., the coding redundancy is 512B per 4kB user data). Based upon [14], we set that the NAND flash memory chips can support seven-level soft-decision sensing (i.e., between two adjacent threshold voltage states there are one hard-decision sensing level and six extra soft-decision sensing levels). Fig. 3 shows the simulated LDPC code decoding failure probability vs. NAND flash memory raw bit error rate when using either hard-decision or soft-decision LDPC code decoding. For the purpose of comparison, we also show the case of BCH code being used in today's commercial SSDs. Assuming the storage system target decoding failure probability is 10^{-15}, the results show that, although the hard-decision LDPC code decoding can only tolerate slightly higher raw bit error rate than BCH code, soft-decision decoding with six extra sensing levels can tolerate about $3\times$ worse raw bit error rate. This directly translates into higher P/E cycling endurance and/or longer data retention time.

2.3 Using LDPC Codes in SSDs

Although soft-decision LDPC code decoding can gain a strong benefit for SSDs as shown in Fig. 3, soft-decision memory sensing can increase the read latency. Based upon our measurements with 25nm MLC NAND flash memory chips, it could take about 125μs to carry out seven-level soft-decision memory sensing in order to maximize the LDPC code error correction strength. Meanwhile, with 200MB/s NAND flash chip I/O bandwidth as specified in ONFI 2.1, it takes about 80μs to transfer the seven-level soft-decision sensing results of 4kB data to the controller. Assuming the LDPC code decoder can operate at only 4Gbps (it could achieve much higher decoding throughput, e.g., beyond 10Gbps [24]), it only takes 8μs to decode one 4kB LDPC codeword. The above example shows that soft-decision memory sensing and corresponding data transfer completely dominate the overall read latency.

Fig. 4 illustrates the two-step read strategy to reduce the latency overhead induced by soft-decision memory sensing. Upon a read request, the SSD controller always starts with hard-decision memory sensing and hard-decision LDPC code decoding, only if the hard-decision LDPC code decoding fails, it invokes the soft-decision memory sensing and executes soft-decision LDPC code decoding. This two-step read strategy can be justified in two aspects: (i) NAND flash memory raw storage reliability gradually degrades with the P/E cycling, hence high-precision soft-decision sensing may only be necessary as flash memory chips have been heavily cycled, and fast hard-decision sensing could be sufficient during the early lifetime of NAND flash memory. (ii) Since LDPC code must ensure an extremely low page error rate (e.g., 10^{-12} and below), hard-decision LDPC code decoding may still achieve a reasonably low failure rate (e.g., 10^{-1} or 10^{-2}) even when flash memory chips have been heavily cycled, which can largely reduce the occurrence of soft-decision sensing.

However, even with this two-step read strategy, LDPC-based SSDs may still be subject to unacceptably increased latency. To quantitatively evaluate the degradation, we carried out trace-driven simulations using the SSD module [15] in DiskSim [16]. Besides widely used traces including Finance1, Finance2, Postmark, and WebSearch, we also use two synthetic workloads in which 66% and 99% of requests are read requests. We set the hard-decision memory sensing latency as 55μs (upper page) and 41μs (lower page), data transfer latency as 20μs, and LDPC code decoding latency as 8μs. The SSD has 8 channels and each contains 8 flash

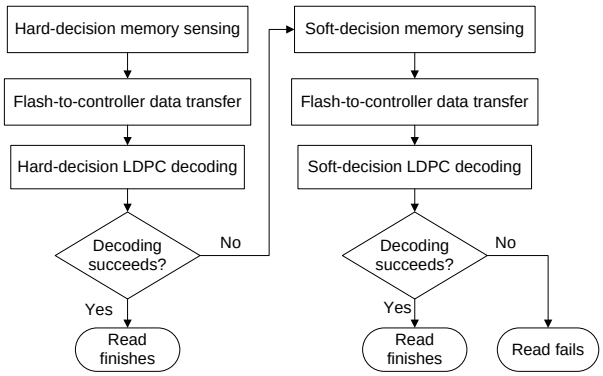

Figure 4: Illustration of operational flow of a straightforward two-step read strategy.

chips. Each flash chip contains 2 dies that share an 8-bit I/O bus and a number of common control signals, and each die contains 4 planes and each plane contains 2048 blocks. Each block contains 64 4kB pages. In our evaluation, we increase the hard-decision LDPC code decoding failure probability from 1% to 3%, and to 30% (i.e., 1%, 10% and 30% of all the read operations invoke the 2nd-step six-extra-level soft-decision memory sensing). Fig. 5 shows the read response time with the scenario without any hard-decision decoding failure, which proportionally increases as the failure rate increases.

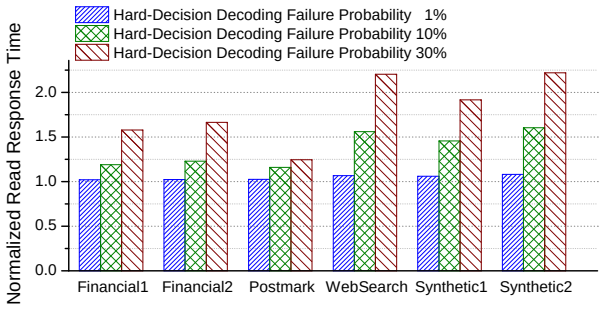

Figure 5: Simulated SSD read response time normalized with the scenario without any hard-decision decoding failure.

Besides read response time, the use of LDPC code will also increase write response time. We collected the write response time results of the above trace-driven simulations. Under the 30% of hard-decision decoding failure probability, the average write response time of all the traces increases no more than 2%. Since read response time has a much higher priority than write response time, the results suggest that LDPC-induced write response time is a much less concern. Therefore, we focus our efforts on reducing LDPC-induced read response time.

3 Proposed Techniques

Further reducing the LDPC-induced SSD read response time delay can be pursued from two orthogonal aspects: (i) reduce the latency overhead of soft-decision memory sensing and data transfer, and (ii) reduce the probability of low-precision LDPC code decoding failure and hence reduce the probability of high-precision soft-decision memory sensing being invoked. This section presents three techniques to explore the design space from these two orthogonal aspects.

3.1 Look-Ahead Memory Sensing

In the basic two-step sensing and decoding strategy described in Section 2.3, soft-decision sensing is only initiated after the LDPC code decoder on the controller reports a decoding failure. As a result, let $P_{\text{hd-fail}}$ denote the hard-decision LDPC code decoding failure probability, the average total sensing and decoding latency experienced by the SSD controller can be expressed as

$$\begin{aligned}\tau_{\text{total}} &= \tau_{\text{hd-sen}} + \tau_{\text{hd-xfer}} + \tau_{\text{dec}} \\ &\quad + P_{\text{hd-fail}} \cdot (\tau_{\text{sd-sen}} + \tau_{\text{sd-xfer}} + \tau_{\text{dec}}), \quad (1)\end{aligned}$$

where $\tau_{\text{hd-sen}}$ and $\tau_{\text{hd-xfer}}$ are the latency of hard-decision memory sensing and data transfer, $\tau_{\text{sd-sen}}$ and $\tau_{\text{sd-xfer}}$ are the latency of soft-decision memory sensing and data transfer, and τ_{dec} is the LDPC code decoding latency.

Exploiting overlapping opportunities between hard- and software-decisions, we propose a method called *look-ahead soft-decision memory sensing* to reduce the overall latency overhead. The key is to initiate soft-decision memory sensing before the hard-decision LDPC code decoding finishes. As illustrated in Fig. 6, right after the memory chip finishes the hard-decision memory sensing and starts to transfer the sensing result to the controller, if there is no outstanding data access request for this memory plane, the memory chip immediately starts the soft-decision memory sensing. Since the soft-decision memory sensing and data transfer operation can be immediately terminated once the hard-decision decoding succeeds, this look-ahead design does not increase the latency in case of a hard-decision decoding success. Hence, we can estimate the average sensing and decoding latency experienced by the SSD controller as

$$\begin{aligned}\tau_{\text{total}}^{\text{look-ahead}} &= \tau_{\text{hd-sen}} + (1 - P_{\text{hd-fail}}) \cdot (\tau_{\text{hd-xfer}} + \tau_{\text{dec}}) \\ &\quad + P_{\text{hd-fail}} \cdot (\tau_{\text{sd-sen}} + \tau_{\text{sd-xfer}} + \tau_{\text{dec}}). \quad (2)\end{aligned}$$

As P/E cycling continues to wear out NAND flash memory cells, the hard-decision LDPC code decoding failure probability $P_{\text{hd-fail}}$ continues to increase. Comparing the average latency estimation in (1) and (2), we show

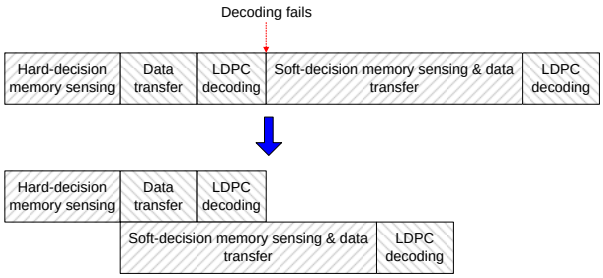

Figure 6: Illustration of the proposed look-ahead memory sensing. Note: the soft-decision memory sensing and data transfer can be immediately terminated in case of a hard-decision decoding success.

that the look-ahead memory sensing becomes more effective as the hard-decision decoding failure probability $P_{\text{hd-fail}}$ increases. As a cost of average latency reduction, this look-ahead memory sensing technique incurs a large number of memory sensing operations, which directly results in a higher on-chip memory sensing power consumption. Nevertheless, since latency is generally a much more important metric than power consumption in SSDs, we expect that such a trade-off is practically justifiable. We will show quantitative power consumption results in Section 5.5.

3.2 Fine-Grained Progressive Sensing and Decoding

Motivated by the fact that the error correction strength of LDPC code may gracefully improve as the accuracy of decoding input information gradually increases, we propose to adopt a fine-grained progressive sensing and decoding design strategy as illustrated in Fig. 7. In the basic two-step sensing and decoding approach, once the hard-decision LDPC code decoding fails, the controller invokes the full-strength memory sensing (e.g., six extra sensing levels between adjacent storage states) provided by the NAND flash memory chip and hence full-strength LDPC code decoding. As illustrated in Fig. 3, it may leave a large latency vs. error correction capability trade-off space unexplored. Our simulations reveal that, even when we only increase the number of memory sensing quantization levels by one, it could noticeably improve LDPC code decoding strength. Meanwhile, as we discussed in Section 2.1, soft-decision memory sensing latency is proportional to the number of memory sensing quantization levels, the fine-grained progressive sensing and decoding strategy shown in Fig. 7 may effectively reduce the overall latency overhead.

Let $\tau_{\text{sen}}^{(1)}$ and $\tau_{\text{xfer}}^{(1)}$ denote the latency of sensing and transferring one extra level, and $P_{\text{sd-fail}}^{(j)}$ denote the probability that LDPC code decoding fails with j extra sensing levels between two adjacent storage states. Let m denote the maximum number of extra sensing levels, we can estimate the average sensing and decoding latency experienced by the controller as

$$\begin{aligned}\tau_{\text{total}}^{\text{prog}} &= \tau_{\text{hd-sen}} + \tau_{\text{hd-xfer}} + \tau_{\text{dec}} \\ &\quad + P_{\text{ave}} \cdot (\tau_{\text{sen}}^{(1)} + \tau_{\text{xfer}}^{(1)} + \tau_{\text{dec}}),\end{aligned} \tag{3}$$

where

$$P_{\text{ave}} = P_{\text{hd-fail}} \cdot \left(1 + \sum_{i=2}^{m} \left(\prod_{j=1}^{i-1} P_{\text{sd-fail}}^{(j)}\right)\right). \tag{4}$$

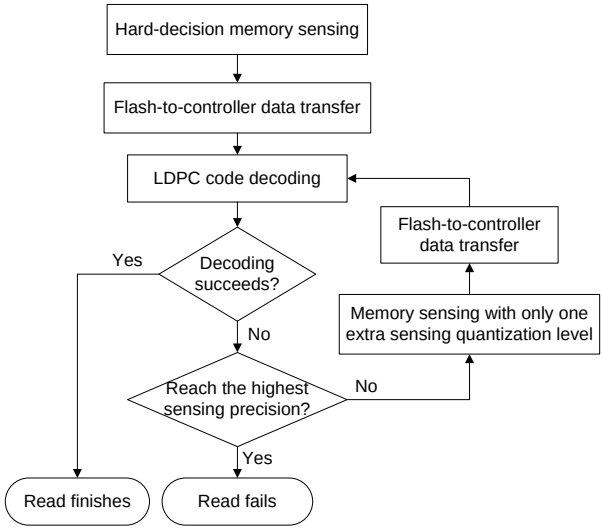

Figure 7: Illustration of operational flow of the proposed progressive sensing and decoding design strategy.

As to be presented later in the paper, the progressive sensing and decoding scheme can be directly combined with look-ahead memory sensing to further reduce the average latency overhead. We note that one major overhead of this progressive sensing and decoding technique is the LDPC code decoding power consumption. As illustrated in Fig. 7, SSD controller has to repeatedly execute LDPC decoding as it progressively increases flash memory on-chip sensing precision. This directly leads to a higher power consumption of LDPC code decoding. Section 5.5 will present quantitative power consumption results in this regard.

3.3 Data Placement Interleaving

The above two techniques aim to reduce the LDPC-induced SSD read response time delay by directly reducing the soft-decision memory sensing and data transfer latency. As discussed above, if we can reduce the LDPC

code decoding failure probability with low-precision memory sensing, we can reduce the occurrence of costly high-precision soft-decision memory sensing, which can further reduce the SSD read response time delay. For this purpose, we propose a technique called *data placement interleaving* to reduce the LDPC code decoding failure probability with low-precision memory sensing. It is directly motivated by the noticeable reliability variation among different NAND flash memory chips. As quantitatively demonstrated later in Section 4.1, memory cell reliability exhibits a noticeable variability across different chips due to die-to-die and wafer-to-wafer fabrication process variation. As a result, pages on one chip could suffer from relatively worse raw storage reliability and hence a high LDPC code decoding failure probability, which leads to a large SSD response time delay. Meanwhile, pages on another chip could be only subject to very low LDPC code decoding failure probability with little effect to system read response time.

Ideally, if we could place the hot data that are frequently accessed onto these flash chips with relatively high reliability, the LDPC-induced system read response time delay can be directly reduced. Nevertheless, this requires sufficiently accurate estimation and prediction on the user data hotness. Although how to identify data hotness in flash memory storage systems has been studied (e.g., see [25–27]), all the on-line data characteristics estimation and prediction methods are subject to implementation overhead in different ways and may not work well for all the possible real-life workloads. As a simple and orthogonal alternative, we propose a workload-independent data placement interleaving strategy to exploit the chip-to-chip reliability for reducing read response time delay. In current design practice, each ECC codeword is entirely stored within one physical page in flash memory. Since modern NAND flash memory chips have large physical page size (e.g., 8kB and 16kB) and ECC codeword size is typically 2kB or 4kB, each physical page contains several logic sub-pages and each logic sub-page is a ECC codeword. When using data placement interleaving, we group and re-organize n ECC codewords to generate n interleaved logic sub-pages, where each logic sub-page contains data from all the n ECC codewords. This is referred to as n-way data placement interleaving. All the n new logic sub-pages are stored on different flash memory chips in SSD. Since different chips have different memory cell storage reliability, different portions in each ECC codeword will experience different bit error rate statistics, and the worst-case bit error rate statistics within each ECC codeword can improve. In case of the two-step sensing and decoding, this can lead to a lower probability of hard-decision LDPC code decoding failure, and in case of fine-grained progressive sensing and decoding, this can lead to lower overall average decoding failure probability P_{ave} as in (3), both of which could reduce system response time delay.

When using n-way data placement interleaving, we should group the ECC codewords that associate with the same write request so that they may be read and updated together. If only one ECC codeword is being read at runtime, all the n memory chips need to carry out sensing and transfer the corresponding data portion. Hence, in practice we should keep n relatively small such as 2 or 4, and may even selectively enable/disable interleaving (e.g., we enable interleaving only when some memory chips show significantly severe reliability issues). In addition, we should mention that this method would need an additional effort in on-chip buffer implementation in the SSD controller.

Since the proposed technique distributes one ECC codeword among several different physical pages, each physical page will contain more logic sub-pages. As a result, changes with any user data will affect more physical pages, leading to a more significant write amplification effect. Moreover, we note that such an interleaving strategy may have different formats for different purposes (e.g., increasing operational concurrency or improving flash memory endurance [28]). The proposed interleaving method in this paper is specifically designed for reducing LDPC-induced latency overhead in SSDs.

4 Experiment Setup and Preparation

To quantitatively evaluate the three techniques through trace-driven simulations, we use the SSD module [15] in DiskSim [16] with different workload traces. Similar to the study presented in Section 2.3, besides widely used traces including Finance1, Finance2, Postmark, and WebSearch, we also use two synthetic workloads in which 66% and 99% of requests are read requests. The simulator can support the use of several parallel packages that can work in parallel. The system works in the synchronous-share-control ganging mode as presented in [15], and has 8 channels and each channel contains 8 flash chips. Each flash chip contains 2 dies that share an 8-bit I/O bus and a number of common control signals, and each die contains 4 planes and each plane contains 2048 blocks. Following the version 2.1 of the Open NAND Flash Interface (ONFI) [29], we set the NAND flash chip interface bus frequency as 200MB/s. Based upon our measurement results on 25nm MLC NAND flash memory chips, we set the block erase latency as 3.24ms, programming latency as 1.45ms (upper page) and 121μs (lower page), and hard-decision read latency as 55μs (upper page) and 41μs (lower page). Since upper-page read involves one extra sensing level than lower-page read, we estimate that sensing one extra level in soft-decision sensing takes 14μs.

Moreover, a quantitative evaluation needs the estimation of various LDPC code decoding failure probabilities (e.g., $P_{\text{hd-fail}}$ and $P_{\text{sd-fail}}$) as described in Section 3. This further demands the availability of NAND flash memory bit error rate statistics and LDPC code decoding failure probabilities under different bit error rates. The remainder of this section discusses how we collect these necessary data points in this study.

4.1 Measurement of Flash Memory Bit Error Rate Statistics

Using commercial 25nm MLC NAND flash memory chips, we carry out memory bit error rate statistics measurements, based on which the proposed techniques can be quantitatively evaluated. As discussed earlier, memory bit error rate gradually increases with P/E cycling. In addition, memory bit error also heavily depends on the data retention time (i.e., how long the data have resided in the flash memory). Based upon prior work in semiconductor device research community [19, 30–32], we use the experiment flow as shown in Fig. 8 to measure memory chip bit error rate patterns under target P/E cycling of N_{cyc} and data retention time of t_{rt}.

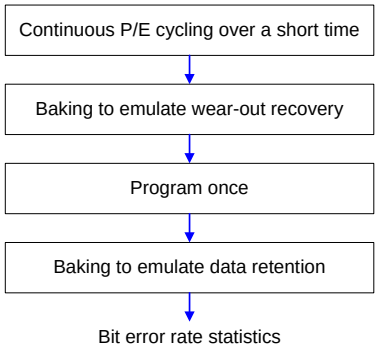

Figure 8: Experiment flow to measure bit error rate statistics of NAND flash memory considering wear-out recovery and data retention in realistic usage scenario.

For each randomly chosen block in each memory chip, as the first step shown in Fig. 8, we continuously carry out N_{cyc} P/E cycles during a short period (e.g., few hours). Nevertheless, in practical SSDs these N_{cyc} P/E cycles on each block are distributed over a much longer period (e.g., several weeks or months). Since the memory cell wear-out caused by P/E cycling can partially recover over the time [31,32], reliability of memory blocks cycled during a short time in the lab cannot faithfully represent the reliability characteristics experienced by real-life SSDs. Therefore, as suggested in [30], we should bake the cycled memory chip to emulate the wear-out recovery over a long period (i.e., the second step in Fig. 8). Then, we write new data into the chosen blocks once and bake the memory chip to emulate the data retention time (i.e., the last step in Fig. 8), after which more realistic reliability characteristics can be measured.

Given target backing temperature, the Arrhenius law [33] can be used to determine the baking time for both wear-out recovery emulation and data retention emulation. Let T_{cyc} denote the temperature under which the flash memory chips are continuously cycled by N_{cyc} times in the lab, t_{cyc} denote the period over which the total N_{cyc} P/E cycles occur in realistic SSD operations, t_{rt} denote the target data retention time in real-life operations, T_{opt} denote normal operating temperature of SSDs, T_{rc}^{B} and T_{rt}^{B} denote the baking temperature for emulating wear-out recovery and data retention, and t_{rc}^{B} and t_{rt}^{B} denote the baking time for emulating wear-out recovery and data retention, we have

$$t_{\text{rc}}^{\text{B}} = A \cdot t_{\text{cyc}} \cdot e^{E_A\left(\frac{1}{k \cdot T_{\text{rc}}^{\text{B}}} - \frac{1}{k \cdot T_{\text{cyc}}}\right)}, \quad (5)$$

$$t_{\text{rt}}^{\text{B}} = t_{\text{rt}} \cdot e^{E_A\left(\frac{1}{k \cdot T_{\text{rt}}^{\text{B}}} - \frac{1}{k \cdot T_{\text{opt}}}\right)}, \quad (6)$$

where k is the Boltzmann constant, the activation energy E_A is 1.1eV, the parameter A is a constant that depends on fabrication technologies and is set as 0.022 according to [30].

In our experiments, we randomly chose 128 blocks from 4 different 25nm MLC NAND flash memory chips, and continuously cycled these blocks by 10,000 times under room temperature of 25°C. We set the SSD normal operating temperature T_{opt} as 40°C, target data retention time t_{rt} as one month, baking temperature T_{rc}^{B} and T_{rt}^{B} as 105°C, and t_{cyc} as 400 days (i.e., each block experiences 25 P/E cycles per day in real-life SSD operations). Accordingly, the baking time for emulating wear-out recovery and data retention is 11 minutes and 39 minutes, respectively. Fig. 9 shows the bit error rate statistics from all the blocks of the four different chips. The results clearly show noticeable chip-to-chip reliability variation, which essentially motivates the proposed data placement interleaving technique.

4.2 Decoding Failure Probability

In the experiments, we use the same rate-8/9 length-4kB LDPC code in Section 2.2. Following the specification in Samsung MLC NAND flash memory chips [14], we set that soft-decision sensing can support up to six extra sensing levels between two adjacent storage states. The proposed progressive memory sensing and decoding strategy aims to reduce the latency overhead by sensing only one extra level between two adjacent storage states at a time. Hence, we should simulate LDPC code decoding performance under various sensing configuration. Fig. 10 shows the simulation results, which clearly

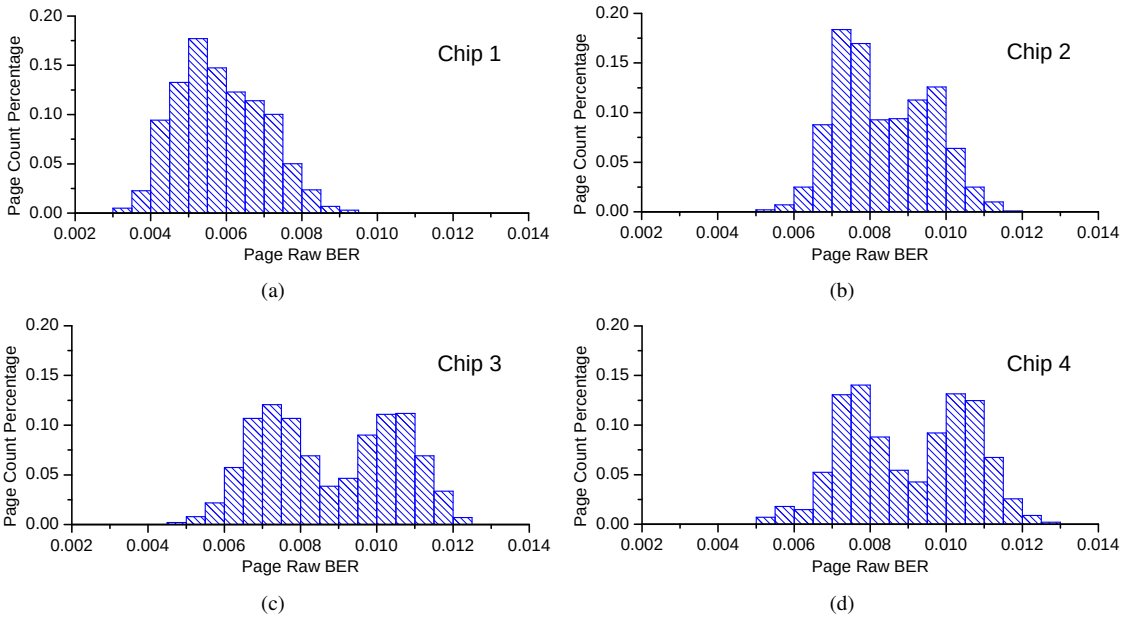

Figure 9: Measured memory bit error rate histogram of four 25nm MLC NAND flash memory chips following the experiment flow described above.

demonstrates the graceful trade-off between decoding failure probability and sensing precision. The results further justifies the underlying rationale of the progressive sensing and decoding strategy.

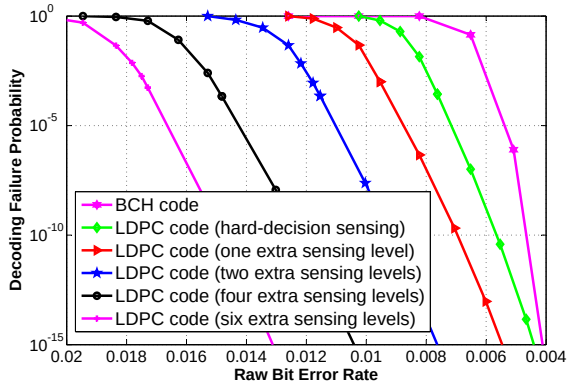

Figure 10: Simulated decoding failure probability vs. NAND flash memory raw bit error rate of BCH code decoding and LDPC code decoding with various sensing precision.

5 Experiment Results

This section presents experiment results to demonstrate the effectiveness of the proposed three techniques for reducing LDPC-induced SSD read response time delay. For the purpose of comparison, we set the baseline scenario as the basic two-step sensing and decoding strategy presented in Section 2.3, where the 2nd-step full-strength soft-decision memory sensing has six extra sensing levels between two adjacent storage states [14]. Based upon the measured 25nm MLC NAND flash raw bit error rate statistics at 10,000 P/E cycling presented in Section 4.1 and the LDPC code decoding failure probability vs. raw bit error rate curves presented in Fig. 10, we estimate that the hard-decision LDPC code decoding failure probability is 28.8%. These results are further fed into the SSD simulator and the corresponding simulation results are shown in Fig. 11. For each trace, the read response time is normalized to the scenario without any soft-decision memory sensing. The response time delay increases range from 25% (for Postmark) to 120% (for Synthtic2).

5.1 Look-Ahead Sensing

We first study the effectiveness of look-ahead memory sensing. We modify the SSD simulator so that six-level soft-decision sensing immediately starts once hard-decision memory sensing finishes if there is no outstanding read request for the memory plane. In case of a hard-decision LDPC code decoding failure, the six-level soft-decision sensing starts about 28μs earlier (including 20μs of hard-decision sensing result data transfer and 8μs LDPC code decoding). In case of a hard-decision LDPC code decoding success, the look-ahead soft-decision sensing will be immediately terminated

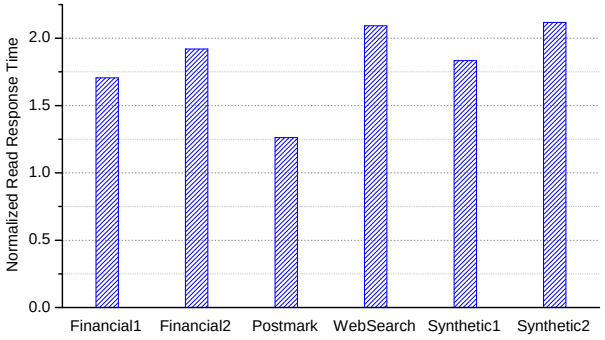

Figure 11: Simulated system read response time when using the two-step sensing and decoding strategy at 10,000 P/E cycling, which will be used as the baseline for the study in this work.

through the RESET command supported by commercial NAND flash memory chips. Fig. 12 shows the simulated SSD read response time that is normalized against the baseline scenario. Results show that the look-ahead sensing can modestly reduce the system read response time delay.

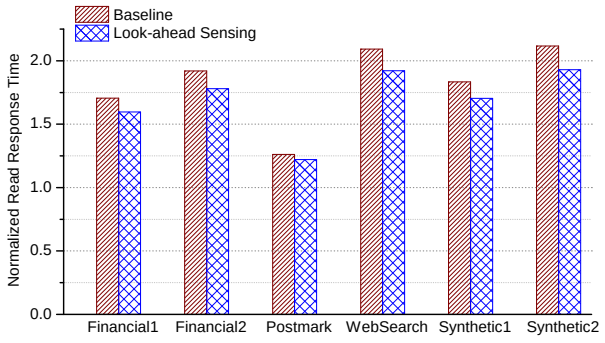

Figure 12: Simulated SSD read response time when using look-ahead memory sensing in comparison with the baseline scenario.

5.2 Progressive Sensing and Decoding

Next, we evaluate the fine-grained progressive sensing and decoding design strategy. As described in Section3.2, throughout the progressive sensing and decoding procedure, in case of LDPC code decoding failure, we only introduce one more extra sensing level between two adjacent storage states. Based upon the measured flash raw bit error rate statistics and LDPC code decoding performance characteristics as shown in Section 4, we accordingly configure the SSD simulator and carry out simulations over various traces. Fig. 13 shows the simulated read response time normalized to the baseline scenario. Results show that such a design strategy can very effectively reduce the response time, and its effectiveness is more significant for those read-intensive traces such as WebSearch.

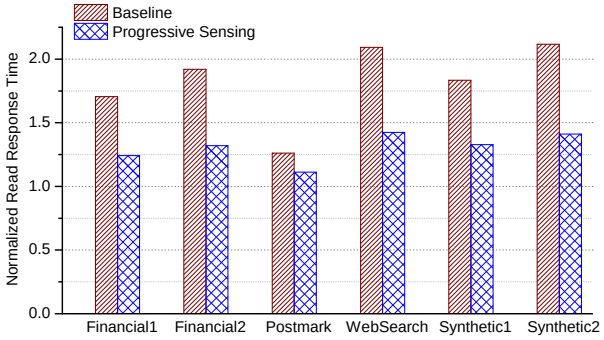

Figure 13: Simulated SSD read response time when using fine-grained progressive memory sensing and decoding in comparison with the baseline scenario.

This progressive sensing and decoding strategy and the above look-ahead memory sensing are orthogonal and can be directly combined to further improve the efficiency, as shown in Fig. 14.

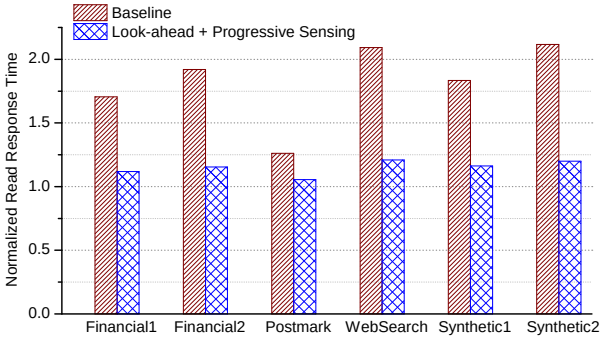

Figure 14: Simulated SSD read response time when combining look-ahead memory sensing and fine-grained progressive memory sensing and decoding in comparison with the baseline scenario.

5.3 Data Placement Interleaving

We further study the data placement interleaving scheme. As described in Section 3.3, by exploiting the chip-to-chip reliability variation, data placement interleaving aims to average out the memory raw bit errors experienced by each ECC codewrod and hence reduce the occurrence of high-precision soft-decision memory sensing. In this study, we collected memory raw bit error rate patterns of four different chips as shown in Fig. 9. Hence, we consider both 2-way interleaving and 4-way interleaving. In 2-way interleaving, we assume the data placement interleaving occurs between chip 1 and 3, and

between chip 2 and 4, respectively. In 4-way interleaving, we assume the data placement interleaving occurs among all the four chips. Intuitively, as we increase from 2-way to 4-way interleaving, we can better average out the bit error rate and hence improve the latency reduction, but meanwhile the implementation could become more complicated. Fig. 15 shows the simulated read response time when using 2-way and 4-way data placement interleaving.

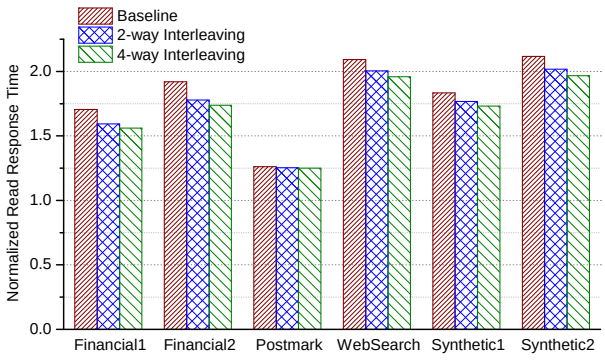

Figure 15: Simulated SSD read response time when using 2-way and 4-way data placement interleaving in comparison with the baseline scenario.

Fig. 16 further shows the aggregated effectiveness when we combine all the three techniques together. The results show that the proposed techniques working in an integrated way can can significantly reduce the LDPC-induced SSD read response time degradation.

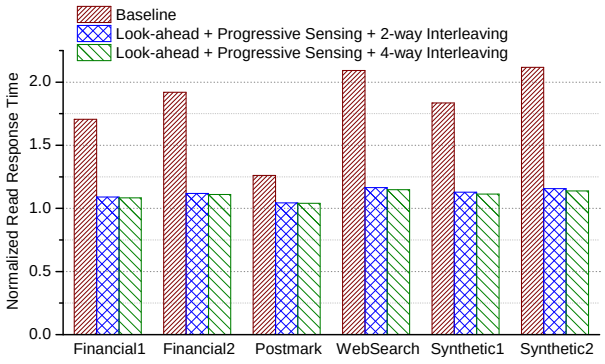

Figure 16: Simulated SSD read response time when combining all the three techniques together in comparison with the baseline scenario.

5.4 Impact of SSD Parallelism

All our simulations are carried out under a highly parallel SSD with 8 channels. To study the impact of SSD parallelism on the effectiveness of the proposed techniques, we reduce the SSD architectural parallelism to 2 (i.e., the SSD only contains 2 channels) while keeping all the other configurations the same. Accordingly, we carry out simulations and the results are shown in Fig. 17. Results suggest that the proposed techniques are insensitive to the SSD parallelism.

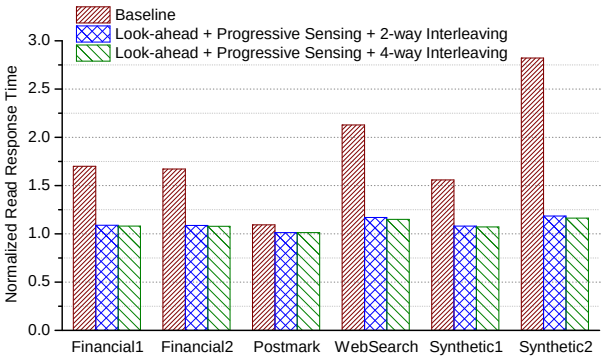

Figure 17: Simulated SSD read response time when reducing the SSD channel parallelism to 2. All the three techniques are used.

5.5 Quantitative Evaluations of Overheads

As pointed out earlier, as the cost of reducing read response time delay, the techniques come with their own overheads. The look-ahead memory sensing can cause higher flash memory on-chip sensing power consumption. The progressive sensing and decoding can result in more frequent execution of LDPC code decoding, leading to a higher LDPC code decoding power consumption. The data placement interleaving could cause a more significant write amplification. In this work, we quantitatively studied the power consumption overhead of the first two techniques. The study is based upon power measurement of 25nm NAND flash memory chips. Each page in the flash memory chips stores 4kB user data, and our measurement results show that on average programming one page consumes about 52μJ, hard-decision memory sensing per page consumes about 2.4μJ, and transferring one page of hard-decision sensing results consumes about 3.7μJ, based upon which we further estimate sensing one extra level consumes about 0.7μJ. Based upon the trace-driven simulation results presented above, we estimate that the use of look-ahead memory sensing increases flash memory access power consumption by 102.9% on average. Regarding energy consumption overhead of LDPC code decoding, we set that decoding one 4kB-length LDPC codeword consumes 0.8μJ according to the results shown in [34]. Again, based upon the trace-driven simulation results presented above, we estimate that the use of progressive sensing and de-

coding increases LDPC code decoding power consumption by 35.2% on average, compared with the ideal scenario where LDPC code decoding is only executed once for each read operation. When considering the memory sensing and LDPC code decoding energy consumption together, the total energy consumption overhead caused by the use of look-ahead memory sensing and progressive sensing/decoding is 42.3%. Because of the higher priority of response time than power consumption in SSDs, we expect that such power consumption overhead are justifiable in most practical applications.

6 Related Work

Many prior work has been done on improving flash memory storage system speed performance. Most prior work focused on how to exploit internal parallelism in SSD to improve system speed performance. Chen *et al.* [35] carried out comprehensive study on the importance and potential by exploiting various internal parallelism for speed performance optimization. Agrawal *et al.* [15] analyzed the effect of page size, striping and interleaving policy on the memory system performance, and proposed a concept of *gang* as a higher-level "superblock" to facilitate SSD system-level parallelism configurations. Min and Nam [36] developed several NAND flash memory performance enhancement techniques such as write request interleaving. Seong *et al.* [37] applied bus-level and chip-level interleaving to exploit the inherent parallelism in multiple flash memory chips to improve the SSD speed performance. In [38, 39], the impact of page allocation on system speed performance has been thoroughly analyzed. The trace-driven simulation approach for SSD designs have also be conducted in multiple studies, e.g. [40]. All these existing solutions are completely orthogonal to this work, and can be used concurrently with the proposed techniques to optimize LDPC-based SSD system speed performance.

ECC for flash memory storage systems has been widely studied as well. As the most promising next-generation ECC, LDPC receives the most attentions from the industry (e.g., see several industrial presentations at Flash Summit this year [8–10]). LDPC decoding strategy optimized for flash memory was proposed in [11]. Optimization of soft-decision memory sensing was analyzed from information theoretical perspective to facilitate the use of LDPC codes in flash memory [41]. Tanakamaru *et al.* [12] have shown that LDPC codes can improve SSD lifetime by 10×. Nevertheless, to the best of our knowledge, no prior work has been carried out to address the system-level response time issue.

7 Conclusion

Making efforts to replace the existing BCH code by powerful LDPC codes in future flash-based storage systems, we have studied the performance implication of the advanced error advanced codes to the system response time. Since unconventional long-latency soft-decision memory sensing will be used in order to fully materialize the error correction capability of LDPC codes, LDPC-based SSDs are inevitably subject to read response time degradation. This paper presents three techniques that aim to reduce the LDPC-induced system speed degradation from different aspects. The key is to appropriately exploit the characteristics of NAND flash memory chips and LDPC code decoding at the SSD system level. In order to quantitatively demonstrate LDPC-induced system read response time degradation and the effectiveness of the proposed techniques, we carried out extensive experimental characterization of NAND flash memory reliability and LDPC code decoding. Accordingly we carried out trace-based SSD simulations and the results show that using these techniques in an integrated way can reduce the worst-case system read response time delay from over 100% down to below 20%. Finally, we note that this work by no means exploits all the possible directions for reducing LDPC-induced speed degradation and much more research from a variety of aspects are necessary to fully address this challenge. One promising future research direction is to investigate how one can exploit data read access locality/intensity variations of different application workloads to reduce overhead of soft-decision memory sensing and LDPC code decoding. Another possible research direction is to study whether or how we should revisit flash translation layer functions in the presence of LDPC codes in future SSDs.

Acknowledgments

We thank the anonymous reviewers and our shepherd Cheng Huang for their feedback and comments that help us to improve the quality and presentation of this paper.

References

[1] L. M. Grupp, J. D. Davis, and S. Swanson, "The bleak future of NAND flash memory," in *Proc. of USENIX conference on file and storage technologies (FAST)*, 2012.

[2] B. Pierce, "Five key steps to high-speed NAND flash performance and reliability," in *Proc. of Flash Memory Summit*, Aug. 2010.

[3] V. Agrawal, R. Ravasio, and G. Wong, "ECC and signal processing technology for solid state

drives and multi-bit per cell NAND flash memories," Tech. Rep. FI-NFL-FSP-0110, FORWARD INSIGHTS, Jan 2010.

[4] R. C. Bose and D. K. Ray-Chaudhuri, "On a class of error correcting binary group codes," *Journal of Information and Control*, vol. 3, no. 1, pp. 68–79, March 1960.

[5] S. Lin and D. J. Costello, *Error Control Coding: Fundamentals and Applications (2nd Ed.)*, Prentice Hall, 2004.

[6] R. G. Gallager, "Low-density parity-check codes," *IRE Transactions on Information Theory*, vol. IT-8, pp. 21–28, Jan. 1962.

[7] D. J. C. MacKay, "Good error-correcting codes based on very sparse matrices," *IEEE Transactions on Information Theory*, vol. 45, pp. 399–431, Mar. 1999.

[8] J. Yang, "Novel ECC architecture enhances storage system reliability," in *Proc. of Flash Memory Summit*, Aug. 2012.

[9] X. Hu, "LDPC codes for flash channel," in *Proc. of Flash Memory Summit*, Aug. 2012.

[10] E. Yeo, "An LDPC-enabled flash controller in 40nm CMOS," in *Proc. of Flash Memory Summit*, Aug. 2012.

[11] R. Motwani and C. Ong, "Robust decoder architecture for multi-level flash memory storage channels," in *International Conference on Computing, Networking and Communications (ICNC)*, Feb. 2012, pp. 492–496.

[12] S. Tanakamaru, Y. Yanagihara, and K. Takeuchi, "Over-10x-extended-lifetime 76%-reduced-error solid-state drives (SSDs) with error-prediction LDPC architecture and error-recovery scheme," in *IEEE International Solid-State Circuits Conference (ISSCC)*, Feb. 2012, pp. 424–426.

[13] J. Chen and M. P. C. Fossorier, "Near optimum universal belief propagation based decoding of low-density parity-check codes," *IEEE Transactions on Communications*, vol. 50, pp. 406–414, March 2002.

[14] C. Kim, J. Ryu, T. Lee, H. Kim, J. Lim, J. Jeong, S. Seo, H. Jeon, B. Kim, I. Lee, D. Lee, P. Kwak, S. Cho, Y. Yim, C. Cho, W. Jeong, K. Park, J.-M. Han, D. Song, K. Kyung, Y.-H. Lim, and Y.-H. Jun, "A 21 nm high performance 64 Gb MLC NAND flash memory with 400 MB/s asynchronous Toggle DDR interface," *IEEE Journal of Solid-State Circuits*, vol. 47, no. 4, pp. 981–989, April 2012.

[15] N. Agrawal, V. Prabhakaran, T. Wobber, J. D. Davis, M. Manasse, and R. Panigrahy, "Design tradeoffs for SSD performance," in *Proc. of USENIX Annual Technical Conference*, 2008, pp. 57–70.

[16] J. S. Bucy, J. Schindler, S. W. Schlosser, and G. R. Ganger, "The DiskSim Simulation Environment Version 4.0 Reference Manual," Tech. Rep. CMU-PDL-08-101, Carnegie Mellon University Parallel Data Lab, May 2008.

[17] P. Olivo, B. Ricco, and E. Sangiorgi, "High Field Induced Voltage Dependent Oxide Charge," *Applied Physics Letter*, vol. 48, pp. 1135–1137, 1986.

[18] P. Cappelletti, R. Bez, D. Cantarelli, and L. Fratin, "Failure Mechanisms of Flash Cell in Program/erase Cycling," in *Proc. of International Electron Devices Meeting (IEDM)*, 1994, pp. 291–294.

[19] 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 Transactions on Device and Materials Reliability*, vol. 4, no. 3, pp. 335–344, 2004.

[20] K. Prall, "Scaling Non-Volatile Memory Below 30nm," in *Proc. of IEEE Non-Volatile Semiconductor Memory Workshop*, Aug. 2007, pp. 5–10.

[21] D. J. C. MacKay and R. M. Neal, "Near Shannon limit performance of low density parity check codes," *Electronics Letters*, vol. 32, pp. 1645–1646, Aug. 1996.

[22] N. Wiberg, "Codes and decoding on general graphs," Ph.D. Dissertation, Linkoping University, Sweden, 1996. available at http://www.essrl.wustl.edu/˜jao/itrg2000/.

[23] F. R. Kschischang, B. J. Frey, and H.-A. Loeliger, "Factor graphs and the sum-product algorithm," *IEEE Transactions on Information Theory*, vol. 47, pp. 498–519, Feb. 2001.

[24] C.-L. Chen, K.-S. Lin, H.-C. Chang, W.-C. Fang, and C.-Y. Lee, "A 11.5-Gbps LDPC decoder based on CP-PEG code construction," in *Proceedings of European Solid-State Circuits Conference (ESSCIRC)*, Sept. 2009, pp. 412–415.

[25] J.-W. Hsieh, T.-W. Kuo, and L.-P. Chang, "Efficient identification of hot data for flash memory storage systems," *ACM Transactions on Storage*, vol. 2, no. 1, pp. 22–40, Feb. 2006.

[26] D. Park and D. Du, "Hot and cold data identification for flash memory using multiple Bloom filters," in *USENIX Conference on File and Storage Technologies (FAST)*, 2011.

[27] T. Luo, R. Lee, M. Mesnier, F. Chen, and X. Zhang, "hStorage-DB: heterogeneity-aware data management to exploit full capacity of hybrid storage systems," in *Proc. of International Conference on Very Large Databases (VLDB)*, 2012.

[28] J. Peterson, J. Strasser, and J. Hyun, "Bit error reduction through varied data positioning," *United States Patent Application 20120304039*, Nov. 2012.

[29] "Open NAND Flash Interface Specification," Tech. Rep., Hynix Semiconductor and Intel Corporation and Micron Technology, Inc. and Numonyx and Phison Electronics Corp. and Sony Corporation and Spansion, 2009.

[30] C.M. Compagnoni, C. Miccoli, R. Mottadelli, S. Beltrami, M. Ghidotti, A.L. Lacaita, A.S. Spinelli, and A. Visconti, "Investigation of the threshold voltage instability after distributed cycling in nanoscale NAND Flash memory arrays," in *Proc. of IEEE International Reliability Physics Symposium (IRPS)*, May 2010, pp. 604–610.

[31] N. Mielke, H.P. Belgal, A. Fazio, Q. Meng, and N. Righos, "Recovery Effects in the Distributed Cycling of Flash Memories," in *Proc. of IEEE International Reliability Physics Symposium*, 2006, pp. 29–35.

[32] H. Yang, H. Kim, S.-I. Park, J. Kim, S.-H. Lee, J.-K. Choi, D. Hwang, C. Kim, M. Park, K.-H. Lee, Y.-K. Park, J. K. Shin, and J.-T. Kong, "Reliability Issues and Models of sub-90nm NAND Flash Memory Cells," in *Proc. of International Conference on Solid-State and Integrated Circuit Technology*, 2006, pp. 760–762.

[33] K. J. Laidler, *Chemical Kinetics (3rd Ed.)*, Prentice Hall, 1987.

[34] T. Mohsenin, H. Shirani-mehr, and B. Baas, "Low power LDPC decoder with efficient stopping scheme for undecodable blocks," in *Proc. of IEEE International Symposium on Circuits and Systems*, 2011, pp. 1780–1783.

[35] F. Chen, R. Lee, and X. Zhang, "Essential roles of exploiting internal parallelism of flash memory based solid state drives in high-speed data processing," in *Proc. of International Symposium on High Performance Computer Architecture (HPCA)*, 2011.

[36] S. L. Min and E. H. Nam, "Current Trends in Flash Memory Technology," *Proc. of Asia and South Pacific Conference on Design Automation.*, p. 2., Jan. 2006.

[37] Y. J. Seong, E. H. Nam, J. H. Yoon, H. Kim, J. Choi, S. Lee, Y. H. Bae, J. Lee, Y. Cho, and S. L. Min, "Hydra: A Block-Mapped Parallel Flash Memory Solid-State Disk Architecture," *IEEE Transactions on Computers*, vol. 59, no. 7, pp. 905 –921, Jul. 2010.

[38] Y. Hu, H. Jiang, D. Feng, L. Tian, H. Luo, and S. Zhang, "Performance impact and interplay of ssd parallelism through advanced commands, allocation strategy and data granularity," in *Proceedings of the International Conference on Supercomputing*, 2011, pp. 96–107.

[39] M. Jung and M. Kandemir, "An evaluation of different page allocation strategies on high-speed SSDs," in *Proc. of HotStorage*, 2012.

[40] F. Chen, T. Luo, and X. Zhang, "CAFTL: a content-aware flash translation layer enhancing the lifespan of flash memory based solid state devices," in *Proc. of USENIX conference on file and storage technologies (FAST)*, 2011.

[41] J. Wang, T. Courtade, H. Shankar, and R.D. Wesel, "Soft information for LDPC decoding in flash: Mutual-information optimized quantization," in *Proc. of IEEE Global Telecommunications Conference (GLOBECOM)*, Dec. 2011, pp. 1–6.