Transactions Briefs

ICS: Interrupt-Based Channel Sneaking for Maximally Exploiting Die-Level Parallelism of NAND Flash-Based Storage Devices

Juhyung Hong, Sangwoo Han, Young Min Park, and Eui-Young Chung

***Abstract*— The performance bottleneck of NAND flash-based storage devices (NFSDs) is mainly due to the slow NAND flash memories (NFMs). One of the well-known techniques for overcoming the bottleneck is an interleaving technique. This technique aims to maximize the utilization of high-speed channels by allowing slow NFMs to operate in parallel. Typically, NFSDs hierarchically apply the multilevel interleaving technique—channel, way, and die-level. While channel/way-level interleaving has already matured, die-level interleaving faces practical difficulties. The most critical issue is that it is not easy to identify individual die status because multiple dies in a package share a status pin, because of the form factor and cost issues of NFSDs. For this reason, NFSD has to issue a read status (RS) command to check the die status, which requires nonmarginal performance overhead. Moreover, the RS overhead attenuates the advantages of channel sneaking (CS) introduced in this brief to accelerate interleaving. To tackle this issue, we propose interrupt-based CS (ICS) that maximizes the impact of die-level interleaving while paying marginal overhead for identifying the die status. ICS performs on-demand die-status monitoring with minor modification of the NFM interface. We prove its effectiveness by conducting experiments in which ICS improves the performance by 19.8% over the typical scheme.**

***Index Terms*— Memory interface, NAND flash memory, storage device.**

I. Introduction

NAND flash memory (NFM) has emerged as one of the most dominant storage media because of advantages such as its shock resistance, low power, and rapid response time. In addition, multiple level cell (MLC) and vertical NFM technologies steeply decline cost-per-bit. These advantages have allowed NAND flash-based storage device (NFSD) applications to scale from mobile systems to high-end server systems. As most of these systems require large storage space capacity, most NFSDs are equipped with multiple NFMs. Under this circumstance, NFSD has resolved the latency and bandwidth issues by employing channel, way, and die-level interleaving techniques. These multilevel interleaving techniques effectively exploit multiple NFMs by overlapping their operations.

As shown in Fig. 1(a), NFSDs typically employ multiple shared buses called channels. Each channel is shared by multiple NFMs. The channels are independent so that NFMs connected to different channels can be operated in parallel, a process called channel-level interleaving. Therefore, the overall bandwidth is proportional to the number of channels. For each channel, way-level interleaving overlaps the operations of multiple NFMs to minimize the idleness of the channel. This is very effective for hiding the long latency of NFMs.[1] Way-level interleaving can be further extended to die-level, as a single way consists of multiple dies. Fig. 1(b) shows the interleaving of read or write requests for one way, and slashed boxes represent command or address sets. While a die is under the preparation of the response for a request, the channel can be utilized for servicing requests to another die. Multiple dies in a way share the data buses as well as a chip enable (CE) pin, while each way has a separate CE pin. For instance, the commercial dual-die NFM package offers only one IO port and a CE pin. Though both way-level and die-level interleaving can overlap the operations of multiple dies, die-level interleaving that operates dies within a package in parallel has some restriction in open NAND flash interface (ONFI) specification [1]. Such multilevel interleaving techniques have dramatically improved the performance of NFSDs by handling IO requests in parallel.

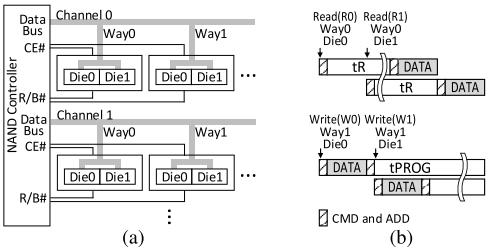

Fig. 1. (a) NFM array with multiple dies. (b) Die-level interleaving.

To maximize the effect of the interleaving techniques, NFSDs are required to efficiently schedule an enormous number of channels, ways, and dies. For this purpose, many studies have proposed page allocation schemes [2], [3], queueing mechanisms [4]–[8], and reordering schemes [9], [10]. These techniques mainly focus on how to effectively allocate or schedule IO requests to the multilevel NFM architecture. Unfortunately, they have paid little attention to the command overhead for identifying the idleness of individual dies for die-level interleaving. Such overhead comes from the property of the typical NFM interface, as each way supports only a single status pin common to all dies within it. Moreover, state-of-the-art interleaving techniques cannot fully hide the long latency of NFMs, as the busy waiting time for processing a single read or write request increases owing to the growth of the page size.[2] Though the NFM interface speed has increased, the host interface speed has also increased.[3] Therefore, the response of a single request can still be delayed.

To cope with this issue, we extend the data burst pause operation specified in ONFI. The NFM controller (NFMC) supporting this operation can arbitrarily suspend data transfer and issue read status (RS) commands to each individual die to check its status. During this data suspension, once a die is identified in ready status, the NFMC can immediately drive cell operations such as tR or tPROG by issuing commands without waiting for the entire data transfer. Therefore, the time for a die to stay in ready status can be shortened. We call a series of these operations as channel sneaking (CS).

Manuscript received September 26, 2017; revised December 31, 2017 and February 24, 2018; accepted March 28, 2018. Date of publication April 20, 2018; date of current version August 23, 2018. This work was supported in part by the National Research Foundation of Korea under Grant 2016R1A2B4011799, in part by the ICT Research and Development program of MSIP/IITP under Grant 2016(R7177-16-0233), and in part by Samsung Electronics Company, Ltd., Suwon, South Korea. *(Corresponding author: Eui-Young Chung.)*

The authors are with the Department of Electrical and Electronic Engineering, Yonsei University, Seoul, South Korea (e-mail: eychung@yonsei.ac.kr).

Digital Object Identifier 10.1109/TVLSI.2018.2824818

[1]Page read time (tR), page program time (tPROG), and block erase time (tBER) range from tens to thousands of micro seconds.

[2]The page size of NFM has increased from 2 to 16 KB.

[3]The NFM interface speed has increase 200 to 800 Mb/s and the host interface speed has increased from 600 MB/s of SATA III to 3.91 GB/s of peripheral component interconnect express (PCIe) GEN3 4lane.

The suspended data transfer is resumed after CS. A simple and intuitive implementation of CS is polling-based CS (PCS), which periodically checks the die status by issuing RS commands. PCS is fully compatible with the typical NFM interface. However, the benefit of PCS is lessened as the overhead of issuing RS increase. The RS overhead is inevitable, as all dies in a single way share two pins that are important for die interleaving, i.e., CE and ready/busy (R/B), because of the form factor, pin count limit, and cost issue. In other words, pin sharing implies that a certain command (in this case, RS) should be asserted for targeting a specific die. To overcome this issue, we propose another type of CS called interrupt-based CS (ICS) to reduce the overhead of RS with a minor modification of the conventional NFM interface. The impact of ICS will increase, as the timing parameters of RS are maintained as constants [1], while data transfer time is shortened owing to the advance of process technology.

The remainder of this brief is organized as follows. First, we review the motivation and related studies in Section II. In Section III, we discuss the limitations of CS and the details of ICS to overcome them. Finally, we show the effectiveness of ICS through experiments in Section IV, followed by a conclusion in Section V.

II. Motivation and Related Works

A. Motivating Example

It is obvious that multilevel interleaving improves the throughput of NFSDs. However, a cost overhead is experienced at some levels. Channel-level interleaving requires additional parallel IO pins (16 pins per channel for 8-bit data in DDR2/3 interface), and way-level interleaving requires multiple CE and R/B signals. On the other hand, die-level interleaving operates with the shared IO bus and common CE and R/B pins. Therefore, maximizing die-level interleaving is one of the most appropriate solutions from both performance and implementation cost perspectives. This observation motivates us to further investigate optimization of die-level interleaving.

Suppose an NFSD that has one channel with three ways with each way consisting of two dies. Fig. 2(a) shows how IO requests are interleaved with ways and dies over time when read requests are issued after write requests. The T period of Fig. 2(a) is enlarged to the transaction-level behavior of the NFM channel in Fig. 2(c). This channel behavior is the same with a typical NFM interface without CS (WOCS), and Fig. 2(b) shows the legend for first row of Fig. 2(c) that shows the timing diagram of channel, and the numbers of way and die that are the monopolizing channel. The timing diagram of channel is subdivided into the die behavior of Way0 in the second row. Die1 of Way0 first is in tPROG for W2, while Die1 of Way1 transmits data through for R1. After these transactions are committed to NFM interface, the read transaction of Way0–Die0 for R3 is enqueued and performs the operation for tR, while Die1 of Way2 transmits data for R2. Before starting a transaction, the NFMC issues a set of commands to the NFMs to prepare for each transaction. The command set includes not only primary commands, such as erase, read, and program, but also auxiliary commands, such as set feature, get feature, and RS. Among these commands, RS is an inevitable command to verify the pass/fail of the program and erase, as well as monitor the ready or busy status of the dies. The NFMC periodically sends RS to dies in a polling fashion when the channel is idle. One RS monopolizes the channel for at least 130 ns in the DDR2/3.

Since multiple dies in a way share an R/B pin, the R/B signal value of a way is "0" (busy) if at least one die is in cell operation. Therefore, to identify which die is busy, ONFI provides an enhance version of RS called RS enhanced (RSE). More specifically, RSE is issued with row addresses to specify the target die. This modification lengthens the command issuing time by 190 ns in DDR2/3. In Fig. 2(c), the slashed and back-slashed boxes represent the

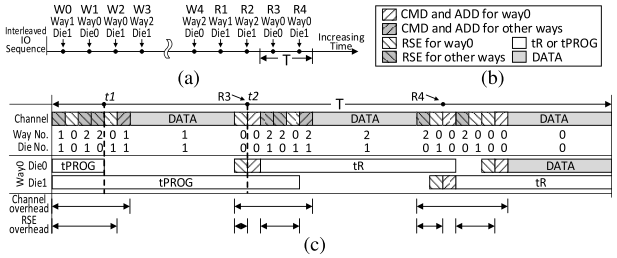

Fig. 2. Timing diagram of IO requests and NFM channel. (a) Interleaved IO sequence. (b) Legend for channel behavior. (c) Typical NFM channel WOCS.

occupancy of command sets, while the gray boxes represent the occupancy of data on the channel. It means that slashed and back-slashed boxes waste channel bandwidth. For this reason, it is obvious that reducing the command sets is one of the efficient ways of increasing channel utilization. Interestingly, the RSE commands (back-slashed boxes) occupies 70% of the full occupancy of the command sets because the NFMC has to verify all active dies from Way0 to Way2. Using RSE, the modern NFMC can detect individual die status without connecting to R/B pins, which has the advantage of reducing the pin count. However, as the active dies increase owing to advanced scheduling or queuing techniques, the issuing count of RSE increases, which negatively affects performance. This example motivates us to reduce the RSE overhead through a tradeoff between performance improvement and pin counts.

To realistically understand the overhead of RSE, we have tested two commercial solid state drives (SSDs) that support nonvolitile memory express (NVMe) over PCIe. The sequential read performance under the deep queue depth (QD) depends on the NFM interface speed that is a major bottleneck in NFSDs [11]. Although we cannot uncover the internal architecture of commercial SSDs, it can be simply conjectured by the performance and IO specification of each SSD. To hide 4-KB tR latency of the triple level cell (TLC) [12] and achieve a throughput of 400 Mb/s per 8-bit channel, the NFSD should have at least six dies of busy status on 400-MB/s 8-channel of which bandwidth is theoretically 3.2 GB/s. However, the measured sequential read performances of the two SSDs are 2.73 and 3.02 GB/s under QD 64, respectively, which is mainly owing to the commands sets for driving or checking NFMs. More specifically, RSE commands for at least six active dies consecutively monopolize channel between every transaction. It takes at least 1.14 μs (190 ns $\times$ 6 dies) at a time, while read or write commands normally take 150 ns. Therefore, RSE is the major overhead as mentioned earlier.

B. Related Works

Multilevel interleaving has contributed significantly to the performance of NFSDs. However, more effort needs to be directed toward enhancing performance as the size of the NFM array continuously grows. To address this issue, many interesting techniques have been introduced, and we categorize them into three.

First, some approaches focus on page allocation to maximize IO parallelism. The performance variation according to page allocation schemes is covered in [2] and [3] and the enhancement of plane-level parallelism owing to advanced commands is exploited in [2] and [4]. Second, several techniques have focused on scheduling issue. They queue IOs in the internal buffer of the NFSD [4], [5], [7] or the buffer of the host system [6], [8], where the IO kernel of operating system resides. The enqueued IOs are reordered by predicting the response time of IOs [8], [9] or the intrinsic property of the NFM [5], [10].

Even if the queueing and reordering of IOs provide higher parallelism, the bandwidth of the NFM interface is not fully utilized because of the long cell operation time or auxiliary commands

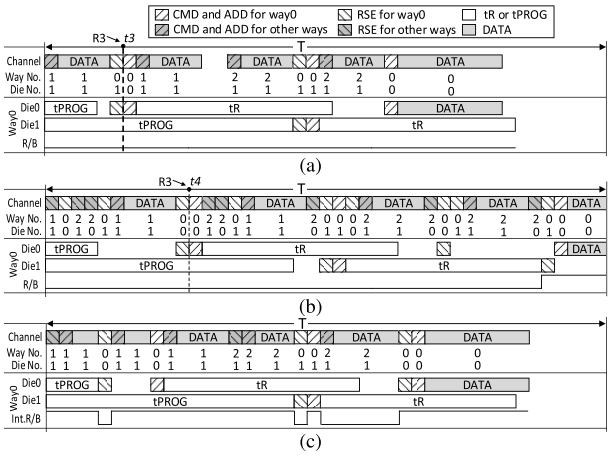

Fig. 3. Timing diagram according to the RS and CS methods. (a) Dedicated R/B signal-based CS. (b) PCS. (c) ICS.

such as set feature. To address this limitation, the approaches in the third category modify the NFM architectures or command sets. Wu *et al.* [13] propose the suspension scheme of program/erase to eliminate contention of read requests because of long tPROG or tBER. Kim *et al.* [14] propose the multipage sized NFM to reduce channel inefficiency owing to invalid sectors. Jeong *et al.* [15] present the new read command combined with read level cycles to reduce the issuing overhead of set feature commands. Although few studies have attempted to mitigate the channel inefficiency owing to commands, to the best of our knowledge, no study has investigated the reduction of the RSE overhead, the critical factor of channel inefficiency. The goal of this brief is, therefore, to resolve this issue.

III. Implementation Methods of Channel Sneaking

A. Typical NFM Interface Without Channel Sneaking

In Fig. 2(c), though the page program of Way0–Die0 for W1 is completed at *t1*, the NFMC cannot convey the next tR command for R3 on the channel until *t2* because of the exclusive channel possession by RSE, command sets, or the data transfer of Way1-Die1 for R1. To formulate the RSE overhead described as examples in Figs. 2 and 3, we define (1)–(4) generalized to the typical NFM hierarchy with notations as summarized in Table I. These equations are derived based on command operations and timing parameters of ONFI specification. $d_i(t)$ is "1" if data is being transmitted on i channel at cycle time t. In WOCS and PCS, $rb_j(t)$ indicates the status of j way. When $rb_j(t)$ is low, it indicates that one or more dies within j way operation are in the busy status. In contrast, when high, j way is in the ready status. When lth transaction is completed, the NFMC knows how many RSE commands will be issued on each channel, and the RSE issuing count can be defined as follows in WOCS:

$$\mathrm{NR}_i^{\mathrm{WOCS}}\left(t_l^c\right)=\sum_{j=0}^{W}\sum_{k=0}^{D}\left(1-d_i\left(t_l^c\right)\right)\cdot\left(1-rb_j\left(t_l^c\right)\right)\cdot b_{jk}\left(t_l^c\right) \quad (1)$$

where the number of ways and dies is W and D, respectively. For example, assuming that the data transfer of Way1-Die1 for R1 is lth transaction in Fig. 2(c), $d_i(t_l^c)$ becomes low at t_l^c and there are total four active dies. Therefore, $\mathrm{NR}_i^{\mathrm{WOCS}}(t_l^c)$ is 4 and the start-event cycle time of the (l+1)th transaction, t_{l+1}^s, should be delayed by at least $t_l^c + \mathrm{DR}(l+1^s)$. $\mathrm{DR}(l+1^s)$, the delay cycle caused by RSE, is 304 when tRSE and tCLK are 190 and 2.5 ns, respectively. It is generalized as follows on i channel in WOCS:

$$\mathrm{DR}_i^{\mathrm{WOCS}}(l+1^s)=\mathrm{NR}_i^{\mathrm{WOCS}}\left(t_l^c\right)\cdot\mathrm{tRSE/tCLK} \quad (2)$$

TABLE I
Frequently Used Notations

t_l^s	start-event cycle time of l-th transaction
t_l^c	completion-event cycle time of l-th transaction
$d_i(t)$	data transfer function on i channel (CH) at cycle time t
$rb_j(t)$	value of R/B signal j way at cycle time t
$b_{jk}(t)$	busy ("1") or ready ("0") status of j way and k die
$cs_i(t)$	channel sneaking function on i-CH at t
$NR_i(t)$	number of RSE commands to be issued forward on i-CH at t before next start-event
$DR_i(l^s)$	delay time of l-th transaction start by RSE overhead on i-CH

where tRSE is the time for one RSE and tCLK is a clock period. The high utilization of dies increases $\mathrm{NR}_i(t_l^c)$ and $\mathrm{DR}_i(l+1^s)$ of WOCS as a result.

B. Dedicated Signal-Based Channel Sneaking

CS extends the data burst pause operation of ONFI to accelerate interleaving and reduce latency. Fig. 3(a) shows the timing diagram for entering the CS operation during half of the data transfer, within the T period of Fig. 2(a). During CS, the NFMC issues tR of R3 to the Die0 of Way0 that is in ready status at *t3*. Although CS requires additional commands to resume data transfer, it reduces the total operation time compared to the WOCS shown in Fig. 2(c).

However, the procedure illustrated in Fig. 3(a) assumes that the way consists of only one die or multiple dies with the dedicated CE and R/B pins, which is called dedicated signal-based CS (DCS). Therefore, the RSE overhead of (2) does not exist in the channel. However, as most commercial NFM packages for large capacity are organized with shared R/B signals because of cost issue, RSE is an inevitable overhead of the NFM interface.

C. Polling-Based Channel Sneaking

In the PCS that is intuitive implementation of CS, the NFMC should issue RSE to check the status of dies in the way of which a R/B signal is low, as shown in Fig 3(b). The RSE issuing count of PCS is given as

$$\mathrm{NR}_i^{\mathrm{PCS}}\left(t_l^c\right)=\mathrm{NR}_i^{\mathrm{WOCS}}\left(t_l^c\right)+\sum_{j=0}^{W}\sum_{k=0}^{D}cs_i\left(t_l^c\right)\cdot d_i\left(t_l^c\right)\cdot\left(1-rb_j\left(t_l^c\right)\right)\cdot b_{jk}\left(t_l^c\right). \quad (3)$$

When $cs_i(t)$ is high, it means that CS is operating in i channel. Because CS operates during the data transfer, it should be coupled $d_i(t)$. When $cs_i(t_l^c)\cdot d_i(t_l^c)$ is high, it denotes the CS operation during the data transfer of the i channel at time t_l^c. Even if the NFMC issues the tR of R3 to Die0 of Way0 earlier than *t2* of Fig. 2(c) at *t4* of Fig. 3(b), the data transfer following tR is delayed as compared to that in Fig. 3(a) because the RSE issuing count increases during CS. This increment is formulated as the second term of (3). For example, assuming that the data transfer of Way1–Die1 is lth transaction in Fig. 3(b), $\mathrm{NR}_i^{\mathrm{WOCS}}(t_l^c)$ is 3 because there are total three active dies between lth and (l+1)th transactions. And total four RSEs are issued during CS of lth transaction. Therefore, $\mathrm{NR}_i^{\mathrm{PCS}}(t_l^c)$ becomes 7.

PCS has another critical problem in controlling CS. It is difficult to determine the CS interval that can derive performance benefits from CS. If the CS interval is excessively short, the RSE overhead will increase, and if it is excessively long, CS is not allowed during data transfer. To remedy the drawbacks of PCS, we propose ICS based on an interrupt protocol.

D. Interrupt-Based Channel Sneaking

NFSDs connect R/B pins of NFMs with an open-drain circuit in ONFI, and a pull-up resistor is used for termination as shown in Fig. 4(a). The R/B output is only high if all dies are in ready

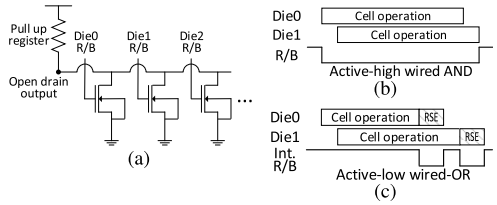

Fig. 4. Implementation and timing diagrams of R/B signal. (a) Open drain circuit of R/B connection. (b) Conventional R/B. (c) Interrupt-based R/B.

status as depicted in Fig. 4(b). The NFMC should continuously issue RSE to confirm which die turns into ready status while the active-high wired-AND bus is low.

In contrast, ICS reduces the number of RSE issues without changing the open-drain circuit because the NFMC checks only dies in the way that triggers the interrupt. Each R/B signal of the dies connects to the active-low wired-OR bus and it works as an interrupt source. Each die enables the R/B signal when the cell operation is completed as shown in Fig. 4(c). At the same time, this active-low event forces the NFMC to enter an interrupt handler. The NFMC issues RSE to dies sharing the R/B signal until the R/B signal is high. Each die that triggered the interrupt releases the interrupt when the status value is read by RSE. Moreover, ICS can have one R/B signal per channel to further reduce pin count. As ICS decreases the RSE overhead, as shown in Fig. 3(c), it can improve channel utilization when the channel is idle as well as during CS. In the ICS scheme, the RSE count is represented as follows:

$$\mathrm{NR}_i^{\mathrm{ICS}}(t_l^c) = \sum_{j=0}^{W}\sum_{k=0}^{D}\left(1-rb_j(t_l^c)\right)\cdot b_{jk}(t_l^c) \tag{4}$$

where $rb_j(t_l^c)$ indicates whether j way has triggered an active-low interrupt. During the data transfer, ICS verifies active dies within ways that triggered the interrupt through CS operation. If a channel is idle, ICS immediately verifies active dies within ways that triggered the interrupt. Therefore, $\mathrm{NR}_i^{\mathrm{ICS}}(t_l^c)$ is the sum of the cycles required to verify only active dies within ways that trigger the interrupt until t_l^c regardless $d_i(t_l^c)$. In contrast, $\mathrm{NR}_i^{\mathrm{WOCS}}(t_l^c)$ and $\mathrm{NR}_i^{\mathrm{PCS}}(t_l^c)$ are the sum of the cycles to verify all active dies at t_l^c. If the data transfer of Way1–Die1 is lth transaction, $\mathrm{NR}_i^{\mathrm{ICS}}(t_l^c)$ is 2 because there are two RSE commands requested by two interrupts between the lth and (l+1)th transaction. Although ICS should modify the peripheral circuits of NFM to set the R/B signal according to the RSE command and the die status, this modification can be implemented with simple AND or OR gates. ICS also requires additional time to handle interrupt; however, it is tens of nanoseconds, which is negligible compared to the RSE overhead.

IV. Experiments

A. Experimental Setup

To evaluate the performance improvement of the proposed schemes, we added (1)–(4) into the trace-driven simulator employed in [2] and [7] by applying the timing parameters of ONFI. We also modified the scheduler and the performance calculator of the existing simulator to implement the RSE overhead. In addition, we modified the simulator to apply TLC NFMs that are widely adopted for NFSDs. We also implemented the high-speed program algorithm [15] with TLC timing parameters [12] as summarized in Table II. Each scheme used the same queuing and allocation method for fair comparison. Moreover, to evaluate our approaches under various environments, we collected workloads from real applications as summarized in Table III. Various traces for mobile, server, and database systems were from [16]. The online transaction processing and search-engine IO trace were obtained from [17]. We also collected synthetic traces and typical PC traces using benchmark programs under a win7 system.

TABLE II
NFM and Timing Parameters in Simulation

Parameters	Values	Parameters	Values (us)
Number of Channel-Way-Die	4/8-8-2/4/8	Page read time (tR) LSB-CSB-MSB-4K	80-105-80-45
Number of Plane-Block-Page	2-2048-63	HSP program time (tPROG)	700
Page size	16KB	Block erase time (tBER)	3500
Data rate	400/666/800Mbps	RSE command time	0.19

TABLE III
Experimental Workloads Information

Performance Metrics	Applications	Workloads
IOPS	Server [16]	Home, Online, Webuser, WebVM, Build
	Database [16], [17]	Fin, Search, ADdata, MSNFS, TPCC, TPCE
	Synthetic	Rand0 (ASbench), Rand1 (IOmeter)
Bandwidth	Database [16].	Livemap, SQL
	PC	Game, Adobe (PCmark)
	Mobile [16].	Boot, Copy (Nexus)
	Synthetic	Seq0 (ASbench), Seq1 (IOmeter)

B. Performance Results

To quantify performance, we classified workloads with two performance metrics, IOs per second (IOPS) and throughput (GB/s), according to the variance of the IO size. Typically, as IOPS are measured with a constant transfer size, we evaluated the average throughput in workloads whose IO sizes have large variance. In addition, as the NFM interface speed and the number of dies have gradually increased for high-performance and large-capacity NFSDs, we explored performance trends according to these.

1) Average IOPS and Throughput: DCS provided the best performance because this scheme fully utilizes the benefit of CS without the channel overhead caused by RSE. Therefore, the IOPS and the throughput of each scheme were normalized to DCS. Fig. 5 shows the performance comparison on the NFM interface with a transfer rate of 666 Mb/s in the 8-channel, 8-way, and 4-die configurations. WOCS and PCS reduced average IOPS compared to DCS by 12% and 20%, respectively, for all workloads shown in Fig. 5(a). In contrast, performance degradation of ICS was 1.2% compared to DCS, which was negligible. These trends were also observed in workloads whose throughput was measured, as shown in Fig. 5(b). Interestingly, PCS had slightly higher IOPS than WOCS in Fin. This was because the RSE overhead during CS was marginal in read-intensive workloads with small size requests, and the early issue of cell operations reduced the response time. However, as the die array extended, the RSE during CS became the critical overhead even in Fin.

In addition, quality of service (QoS) of ICS was better than WOCS and PCS. We exploited confidence plot widely used as a QoS metric. In Rand1, the response time of ICS at 99.99% of total host IOs is 2.28 ms, which is close to 2.26 ms of DCS. In contrast, those of WOCS and PCS are 2.39 and 2.40 ms, respectively.

2) Sensitivity Analysis With Respect to Die Expansion: The growth of dies can increase the absolute capacity and performance of NFSDs, as described in Section II-A. However, this expansion of dies widens the performance gap from the ideal method, as shown in Fig. 6. Each scheme is normalized to DCS according to the die expansion under the same NFM hierarchy and the same data rate. Line graphs indicate the average values of IOPS or throughput in all workloads. In Fig. 6(a), as dies increase, WOCS and PCS show significant performance degradation (up to 18.7% and 26.9%, respectively). In contrast, the die expansion had a marginal impact (up to 1.7%) on performance degradation in ICS. The IOPS drop was larger than the bandwidth drop, as workloads whose IOPS was measured had relatively small size requests and these workloads had higher chip utilization than the workloads whose bandwidth was measured. In Fig. 6(b), the performance variation was slightly

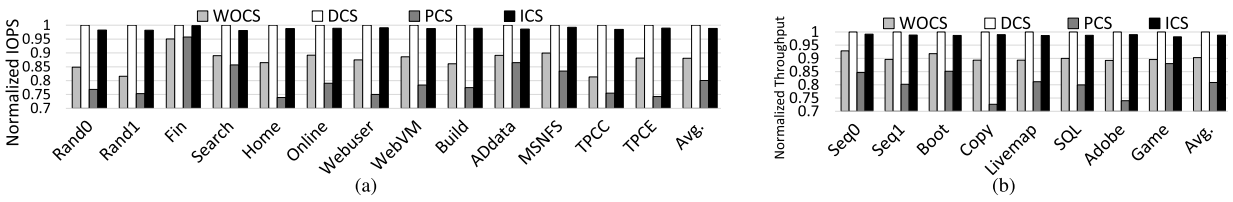

Fig. 5. Performance comparison for workloads at 666-Mb/s data transfer rate on 8-channel, 8-way, and 4-die. (a) IOPS comparison. (b) Throughput comparison.

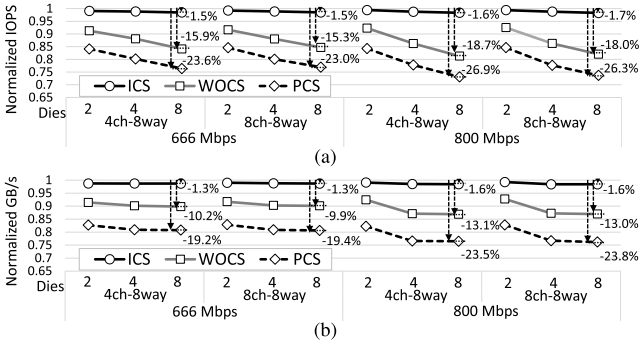

Fig. 6. Performance sensitivity with respect to the expansion of dies. (a) IOPS comparison. (b) Throughput comparison.

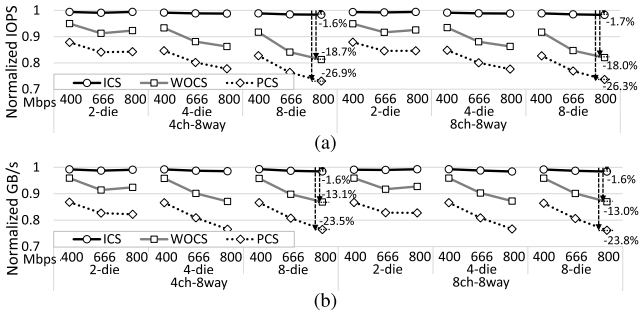

Fig. 7. Performance sensitivity with respect to the data transfer rate. (a) IOPS comparison. (b) Throughput comparison.

between four dies and eight dies. This is because 4-die interleaving is the maximum limit that can hide cell operations in corresponding workloads.

3) Sensitivity Analysis With Respect to Data Transfer Rate: A data transfer rate on channels is a critical factor in the NFSD performance. DDR2/3 standard supports IO speeds of up to 800 Mb/s. In contrast, at 100 Mb/s, the transfer rate of command or address is constant, as command and address are unprotected, and therefore, more vulnerable to errors. Therefore, as the IO speed raises, RSE increases its share of the channel, i.e., the number of RSE per certain time increases on channels. Fig. 7 shows performance variation according to the data transfer rate. As expected, the IOPS and throughput gradually decreased as the transfer rate increased, except for the 2-die configuration. In the 2-die configuration, die interleaving was not accelerated even when the transfer was raised from 666 to 800 Mb/s. In Fig. 7(a), WOCS and PCS reduced the IOPS compared to DCS by up to 18.7% and 26.9%, respectively, while ICS decreased the IOPS by only a maximum value of 1.7%. In the throughput results shown in Fig. 7(b), performance degradation of ICS, WOCS, and PCS were up to 1.6%, 13.1%, and 23.8%, respectively.

V. Conclusion

In this brief, we revisited the data burst pause of an NFM specification, and extended it to support CS that enables the early issue of NFM cell operations. Moreover, to reduce the RSE overhead incurred during CS, we have proposed ICS through a simple protocol change without modifying the NFM interface. In the experimental results, conventional WOCS and PCS reduced IOPS compared to the ideal method by up to 18.7% and 26.9%, respectively. On the other hand, the performance degradation of ICS was less than 1.7% in all NFM configurations, which was close to the ideal performance. In addition, the performance drop of ICS according to the increase in die and NFM IO speed-up was marginal (up to 1.7%), while those of WOCS and PCS significantly increased. Consequently, these results show that the proposed scheme is an effective solution for the latest technological trends of NFSD with increasing capacity and throughput.

References

[1] (2017). *Specifications—ONFi*. [Online]. Available: http://www.onfi.org
[2] Y. Hu, H. Jiang, D. Feng, L. Tian, H. Luo, and C. Ren, "Exploring and exploiting the multilevel parallelism inside SSDs for improved performance and endurance," *IEEE Trans. Comput.*, vol. 62, no. 6, pp. 1141–1155, Jun. 2013.
[3] M. Jung, "Exploring parallel data access methods in emerging non-volatile memory systems," *IEEE Trans. Parallel Distrib. Syst.*, vol. 28, no. 3, pp. 746–759, Mar. 2017.
[4] M. Jung, E. H. Wilson, III, and M. Kandemir, "Physically addressed queueing (PAQ): Improving parallelism in solid state disks," in *Proc. IEEE 39th Annu. Int. Symp. Comput. Archit. (ISCA)*, Washington, DC, USA, 2012, pp. 404–415.
[5] S. S. Hahn, S. Lee, and J. Kim, "SOS: Software-based out-of-order scheduling for high-performance NAND flash-based SSDs," in *Proc. IEEE 29th Symp. Mass Storage Syst. Technol. (MSST)*, May 2013, pp. 1–5.
[6] Q. Zhang, D. Feng, F. Wang, and Y. Xie, "An efficient, QoS-aware I/O scheduler for solid state drive," in *Proc. IEEE 10th Int. Conf. Embedded Ubiquitous Comput. Int. Conf. High Perform. Comput. Commun.*, Nov. 2013, pp. 1408–1415.
[7] C. Gao, L. Shi, M. Zhao, C. J. Xue, K. Wu, E. H.-M. Sha, "Exploiting parallelism in I/O scheduling for access conflict minimization in flash-based solid state drives," in *Proc. 30th Symp. Mass Storage Syst. Technol.*, Jun. 2014, pp. 1–11.
[8] M. H. Jo and W. W. Ro, "Dynamic load balancing of dispatch scheduling for solid state disks," *IEEE Trans. Comput.*, vol. 66, no. 6, pp. 1034–1047, Jun. 2017.
[9] N. Elyasi, M. Arjomand, A. Sivasubramaniam, M. T. Kandemir, C. R. Das, and M. Jung, "Exploiting intra-request slack to improve SSD performance," in *Proc. 22nd Int. Conf. Archit. Support Programm. Lang. Oper. Syst. (ASPLOS)*, New York, NY, USA, 2017, pp. 375–388.
[10] E. H. Nam, B. S. J. Kim, H. Eom, and S. L. Min, "Ozone (O3): An out-of-order flash memory controller architecture," *IEEE Trans. Comput.*, vol. 60, no. 5, pp. 653–666, May 2011.
[11] M. Abraham, "Architectural considerations for optimizing SSDs," in *Proc. Flash Memory SUMMIT*, Aug. 2014, pp. 1–22.
[12] D. Sharma, "System design for mainstream TLC SSD meeting the performance challenge," in *Proc. Flash Memory SUMMIT*, Aug. 2014, pp. 1–20.
[13] G. Wu, P. Huang, and X. He, "Reducing SSD access latency via NAND flash program and erase suspension," *J. Syst. Archit.*, vol. 60, no. 4, pp. 345–356, 2014.
[14] J.-Y. Kim, S.-H. Park, H. Seo, K.-W. Song, S. Yoon, and E.-Y. Chung, "NAND flash memory with multiple page sizes for high-performance storage devices," *IEEE Trans. Very Large Scale Integr. Syst.*, vol. 24, no. 2, pp. 764–768, Feb. 2016.
[15] W. Jeong *et al.*, "A 128 Gb 3b/cell V-NAND flash memory with 1 Gb/s I/O rate," *IEEE J. Solid-State Circuits*, vol. 51, no. 1, pp. 204–212, Jan. 2016.
[16] (2017). *Storage Networking Industry Association*. [Online]. Available: http://iotta.snia.org/
[17] (2017). *UMass Trace Repository*. [Online]. Available: http://traces.cs.umass.edu/