Multi-token based Power Management for NAND Flash Storage Devices

Taehee You, Sangwoo Han, Young Min Park, Hyuk-Jun Lee, and Eui-Young Chung

***Abstract*—NAND Flash based Storage Devices (NFSDs) have been widely employed in various systems including cloud servers as well as mobile devices. The core component of NFSDs is NAND Flash Memory (NFM) which has several advantages over the conventional Hard Disk Drives (HDDs). An NFSD typically adopts a bunch of NFMs which are operated in parallel for maximizing the I/O throughput. However, optimizing for performance may not be desirable from the power budget perspective. In other words, concurrent operations of NFMs often drain inordinate current, which leads to the violation of the power budget allocated for a storage device. In this paper, we propose a novel power management scheme which maximizes concurrent operations of NFMs under the given power constraint. The proposed method quantizes the given power constraint of an NFSD. A quantum also called token is the basic unit of power management. The proposed power management scheme allocates tokens to NFMs and only the NFMs having enough tokens can perform their operations. We call this method Multi-Token based Power Management (MTPM). The critical issue of MTPM is a deadlock which is resolved with the key allocation scheme. Furthermore, we enhance MTPM to improve performance. The extended method called Keyless MTPM (KMTPM) improves the overall performance by relaxing the key acquisition requirement and allowing sub-atomic operations. In the experimental results, we confirm that the proposed methods always meet the given power constraint. The proposed KMTPM improves throughput by 22.85% compared to state of the art technique. In addition, KMTPM only incurs 3.8% of performance overhead and 0.015% of area overhead.**

***Index Terms*—NAND flash memory, power management, peak current, multi-token, power budget, power budget violation**

I. Introduction

RECENTLY, the demand of NAND Flash based Storage Devices (NFSDs) has been growing in various portable devices, data centers, server machines, and laptop PCs due to its non-volatility, high performance, small form factor, shock resistance and so on [1], [2]. To realize a high performance NFSD, a single NFM chip should support high I/O frequency and throughput. However, the program throughput of a single NFM chip has been saturated in recent years, while the requirement of host systems is continuously increasing [3].

This work was supported in part by the National Research Foundation of Korea (NRF) grant funded by the Korea government (MSIT) (2016R1A2B4011799); in part by the MOTIE (Ministry of Trade, Industry & Energy) (10080722, 10080590) and KSRC (Korea Semiconductor Research Consortium) support program for the development of the future semiconductor device; in part by SK hynix. The EDA tool was also supported by the IC Design Education Center (IDEC), Korea.

T. You, S. Han, Y. M. Park, and E.-Y. Chung are with the School of Electrical and Electronic Engineering, Yonsei University, Seoul 03722, South Korea (e-mail: eychung@yonsei.ac.kr).

H.-J. Lee is with the School of Computer Science and Engineering, Sogang University, Seoul 04107, South Korea.

In order to overcome such circumstance, a conventional NFSD is equipped with multiple NFM chips which are connected to the NFM controller through the interconnection network called channels and ways. This architecture enables NFM chips to be operated in parallel for achieving high throughput. It is obvious that interleaving operations for multi-channel/multi-way NFM chips are an attractive method from the performance perspective. However, this often leads to inordinate power consumed by NFM chips, which contributes to the system failure.

In a typical system design, each system component is constrained by its power consumption to meet the overall system power requirement. An NFSD is also one of the system components and constrained by its allowable peak current budget (i.e. power budget [4]), even though the power budget of NFSDs varies from the servers to portable devices [5]. Therefore, increasing channels and ways for higher throughput may inadvertently cause an NFSD to violate the given power budget. Once an NFSD violates its power budget, the behavior of NFM chips become unstable due to the voltage source drop, ground bounce, signal noise, black-out, unstable NAND operation, shutdown, and so on [3]–[7]. Therefore, an efficient power management is essential for realizing reliable high-performance NFSDs.

Several methods were proposed for this purpose [3], [4], [6], [7]. The key idea of [3], [6] is to limit the parallelism of power critical operations. In [3], [6], they focused on the program operation since it is one of the most power critical operations. A program operation can be divided into two sub-operations - program execution and verify [1]. They observed that an NFM chip drains much higher current in the pre-charge phase of the program execution compared to the verify. Based upon the observation, their methods prevent a program operation for other NFM chip if at least one of other NFM chips is in the pre-charge phase. However, these methods still vulnerable to power budget violation as the combination of other operations often incurs higher current than the program operation. In [8], researches show that the read or erase operation drains 70% of current for the pre-charge phase. In other words, it is necessary to consider all possible combinations of NFM operations rather than focusing on a few power-critical NFM operations. Authors in [4] proposed a channel/way architecture so that the number of concurrent program operations can be constrained for the given power budget. This method prevents power violation, but performance degradation occurs due to limited parallel operations. In [7], they consider the current of parallel operations and shift some of them to meet the given power budget. This allows the NFSD to operate within the given power budget, but performance reduction occurs due to the delayed operations. In addition, their method is a firmware-based method and requires space to store the sum of current for each operation. This requires more computation and storage space as the number of chips increases, which leads to the scalability issue.

To resolve the issues of prior schemes, we propose a multi-token based power management (MTPM) scheme which considers all possible combinations of NFM operations and minimizes performance reduction. MTPM uses the given power budget as a hard constraint. In MTPM, we quantize the given power budget and a quantum is called token which is the base unit of power management. Tokens are managed in a distributed manner by the MTPM manager which is implemented in each NFM chip. The MTPM managers are connected as a ring network for token transfer. In this configuration, only the NFM chips which have sufficient tokens for the requested operations are allowed to perform their operations.

This token-based method is simple and intuitive but has several issues to be resolved. The first issue is a deadlock. It occurs when all NFM chips do not have enough tokens for the requested operations. We resolve this issue by proposing a key allocation method. The second issue is to minimize the performance degradation due to the power management. MTPM is a conservative method to strictly meet the given power budget, hence it naturally suppresses the parallelism of NFM operations. For this purpose, we model the current consumption of each operation by tokens. Then, to resolve the performance degradation issue, we identify the mismatch between the current and the modeled tokens of requested operations and provide means for accurate modeling.

The contributions of this paper are as follows. First, we propose a token-based power management called MTPM that strictly satisfies the power budget of NFSDs. It is compatible with contemporary NFM controllers. We also resolve the deadlock issue of MTPM by a key allocation method ('key' is described in detail in Section III-B). Second, we enhance MTPM to reduce the overhead of key allocation. This extended method called Keyless MTPM (KMTPM) allows more parallel operations to improve performance. Third, we further improve the performance of MTPM by managing the tokens in sub-operation level. KMTPM in sub-operation level improves average throughput by 22.85% compared to related work.

The remainder of this paper is organized as follows. In Section II, we present the related work and motivation. In Section III, we describe the proposed methods including MTPM and KMPTM. Finally, we present the experimental results in Section IV followed by the conclusions in Section V.

II. Related Work and motivation

A. Token based power management

The multi-token based power management for main memory based on Single-Level Cell (SLC) and Multi-Level Cell (MLC) Phase Change Memory (PCM) are proposed in [9] and [10], respectively. These methods prevent the power budget violation as program operations are constrained by limited tokens. In [9] and [10], one token represents the power consumed

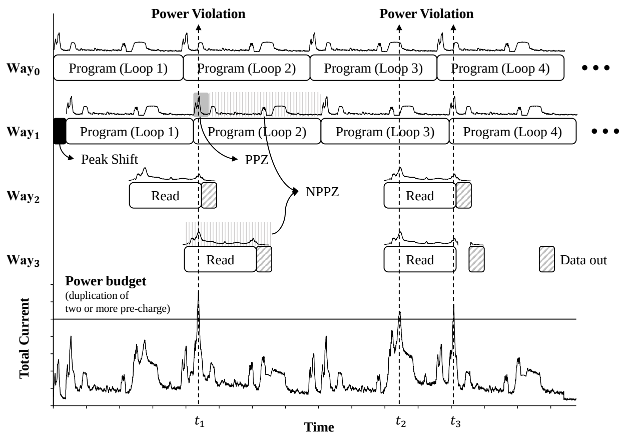

Fig. 1: Power violation caused by overlapping PPZs and NPPZs under a PPZ-only management scheme

by programming one bit and the same number of tokens are allocated to each chip. The controller monitors and stores the number of tokens held in each chip and determines whether transfers an operation by comparing the number of bits to program and tokens.

However, applying these techniques to NFSDs can decrease performance because the centralized controller continuously monitors the holding tokens of the chips. In the case of NFSDs, chips are connected to a channel via a shared-bus. As the data size is increased up to 32KB [11], [12], the channel occupancy increases. When the channel is occupied by the large data transmission, transmitting the power management related commands by the controller is delayed. One way to solve this problem is to add a separate path between the memory and the controller. However, this method increases the complexity of the memory controller by changing the protocol and interface [9].

In this paper, we apply MTPM to each chip, and each chip itself determines whether an operation can be performed within the given power budget in a distributed manner. This method does not change protocol and interface between the chips and the controller, thus maintaining compatibility with conventional controllers.

B. Prior Power management schemes for NFSDs

A peak power consumption occurs at the pre-charge phase of a program operation [3], [6]. If two or more pre-charge phases overlap, the power supply of NFSDs drop by more than 0.3V [6]. Therefore, prior works proposed methods to prevent overlapping pre-charge phases.

Prior works for power management can be classified into two. The first is to prevent overlapping pre-charge phases, which incurs the largest peak current in NFSDs. In [6], they added a circuit to detect pre-charge, and the NFM controller does not issue a program operation in the case that prior pre-charge is detected. In [3], they added a new command to start a program operation, which prevents pre-charge duplication. In order to perform a program operation in each chip, the new command must be received from the NFM controller. The power management techniques in [3] and [6] consider only the

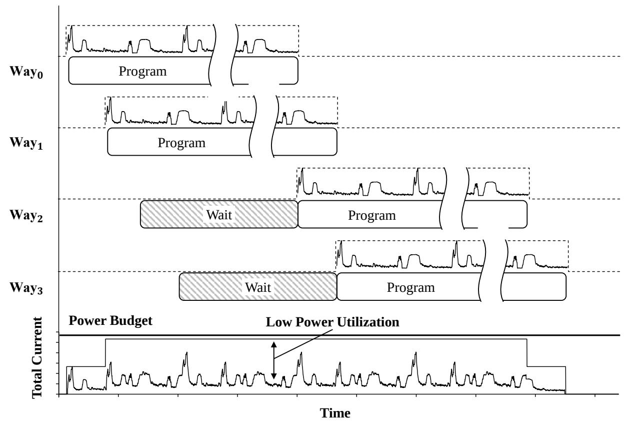

Fig. 2: Power utilization when limiting the number of parallel operations

peak power zone (PPZ) of NFM operations (i.e., pre-charge of program). However, according to [8], the peak current of read and erase operations are 85.43% and 72.29% of the program operation, respectively. Thus, overlapping PPZ and non-peak power zone (NPPZ) could cause the current to exceed the power budget of the NFSDs.

To justify the power management considering NPPZ as well as PPZ of NFM operations, we use a simple example shown in Fig. 1. The example only employs peak power zone management [called PPM]. In Fig. 1, we assume that NFSDs consist of 1-channel and 4-ways. Program operations are performed on way 0 (w_0) and way 1 (w_1), and read operations are performed on w_2 and w_3. As we predict, Fig. 1 shows power budget violations. Power budget violations of t_1 and t_3 are produced by overlapping PPZ and NPPZ, and power budget violation of t_2 is only by NPPZs. This justifies that we need to manage the current of all types of operations, not just program operations.

The second approach is to consider PPZs and NPPZs [4], [7]. In [4], the maximum number of program operations executed in parallel is determined. Therefore, subsequent program operations beyond its maximum number have to wait. In [7], Dynamic Current Capping (DCC) is proposed for power management of NFSDs, which includes current models for all operations based on a series of linear functions. DCC calculates the current for an operation before starting the operation using the current model in the controller and determines the start time of the operation constrained by the power budget. As the performance required by applications is increasing, NFSDs increase the channel/way and parallel processing of operations. However, the above methods limit the parallelism of NFSDs, causing performance degradation.

Fig. 2 shows performance degradation in prior works considering both PPZ and NPPZ when limiting the number of parallel operations. In Fig. 2, we assume that the power budget is twice the peak current. If program operations are to be performed in w_0 and w_1, the program for w_2 and w_3 are delayed due to the power budget limit of the NFSD. The delayed operations will be processed after completing in w_0 and w_1. In Fig. 2, the power budget of the NFSD is not exceeded, but the current is much lower than the power budget of the NFSD. In other words, power management incurs low power utilization, which causes performance degradation. To overcome this, we propose the power management scheme using atomic operations which compose the NFM operation (e.g., program, read, erase). The atomic operations are explained in the next section.

TABLE I: Atomic operations of NFSDs

Operation	**Atomic operation**		
Program (1 loop)	Program Execution	Verify	-
Read	Init. Setup	Read Execution	Data Out
Erase	Erase Execution	Verify	-

C. Atomic operations of NFSDs

The NFM cell contains a floating gate. The operations of NFSDs are program, read, erase that perform injecting, sensing and removing electrons in a floating gate, respectively [1]. As shown in Table I, each operation consists of several atomic operations which can be suspended and resumed.

A program operation consists of program execution and verify. Program execution includes bit-line pre-charge and applies a high voltage to the NFM cell to fill the floating gate with electrons. Verify is a step to ensure that the program is successful. A loop containing program execution and verify is repeated several times until the program is successful [1]. A read operation determines the value by sensing the number of electrons in the floating gate. During the initial setup, bit-lines are discharged and the page buffer, which temporarily stores data before programming or reading NFM cells, is initialized to 0. Read execution is the process of sensing the number of electrons in NFM cells. Data-out is the process of transmitting data in the page buffer to the NFM controller [13]. An erase operation consists of erase execution and verify. Erase execution removes electrons from the floating gate and verify is to ensure that the erase is successful [1].

Since each operation proceeds independently, suspension and resumption are possible. In [14], [15], suspension and resumption of operations (i.e. program and erase) are used to reduce read latency. We also use atomic operations to improve performance. To avoid power budget violation, power consumption should be defined for all types of NFSDs' operations. If the power management is performed per operation, power consumption for program is defined by the program execution as pre-charge consumes the largest power. This over-constrains the power budget for the verify step of a program operation. Therefore, we propose the power management per atomic operation. Also, we define sub-atomic operations which compose atomic operations to increase the power utilization in Section III-D2.

III. Proposed Methods

A. Overview

In this paper, we propose a power management scheme that prevents power budget violation using multi-token. In

TABLE II: Description of terms used in the proposed methods

Parameter type	Parameter	Description
Power management	Key	A Chip with a key can accumulate tokens upon receiving a request
	RT (Required Token)	The number of tokens required to perform operations
	HT (Holding Token)	The number of tokens a chip owns
	UT (Used Token)	The number of tokens used performing an operation
Device module	OI (Operation Information)	A path from a chip to MTPM for transmitting the type of operations
	OE (Operation Enable)	A path from MTPM to a chip for indicating whether an operation can be performed
	OTC (Operation-Token Convertor)	Notify the required tokens of an operation and the end of an operation
	ORD (Operation Requirement Decision)	Determine whether there is an operation to perform or an operation is finished
	SP (State Pointer)	Map the state of operation to tokens
Token modeling	TT (Total Token)	The total number of tokens in the system
	PB (Power Budget)	Allowable peak current in the device
	TG (Token Granularity)	The number of bits to represent the max current
	P_{Ai},R_{Ai}, E_{Ai}	The i_{th} atomic operation of the program, read, and erase operation
	P_{SAi},R_{SAi}, E_{SAi}	The i_{th} sub-atomic operation of the program, read, and erase operation

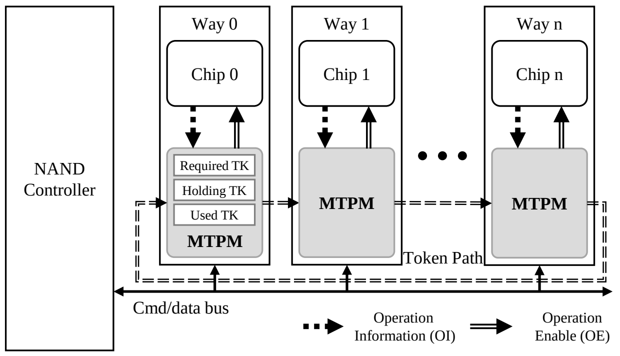

Fig. 3: Overview of multi-token based power management

short, each chip transmits an operation request to MTPM, and MTPM checks the number of tokens it has. If the MTPM has enough tokens to perform the operation, it sends a signal to start the operation on the chip. In Table II, we include the description for the terms used in the proposed methods.

Fig. 3 shows an architectural overview of NFSDs with MTPM. The MTPM is instantiated per chip and is serially connected to its neighbor to transfer the tokens. The MTPM and chip are connected with two paths, Operation Information (OI) and Operation Enable (OE). The OI is a path from a chip to MTPM for transmitting the type of operations to perform. OE is a path from the MTPM to the chip for indicating whether an operation can be performed. MTPM contains three registers for storing the tokens according to the state of the chip. The required token (RT) represents the number of tokens required to perform operations on the chip, and the holding token (HT) indicates the amount of tokens it owns. When there is a request to perform an operation on the chip, the MTPM compares RT with HT. If the HT is larger, the operation is enabled via OE. When the operation is performed on the chip, using token (UT) is incremented by RT, and the unused tokens are stored in the HT.

The proposed method performs power management for atomic operations to improve performance. For this, the MTPM includes the token information required by each atomic operation. The token modeling formula is as follows.

$$RT_i = \left\lceil \frac{I_{i,max}}{I_{token}} \right\rceil, \quad I_{token} = \frac{PB}{TT} \tag{1}$$

RT_i is the RT to perform the i_{th} atomic operation of each operation, which is the largest current consumed in the i_{th} atomic operation ($I_{i,max}$) divided by the current per token (I_{token}). We use the current for each operation measured every 40 ns. I_{token} is the power budget (PB) of the NFSDs divided by the total tokens (TT).

$$PB = \alpha \times I_{max}, \quad TT = \alpha \times (2^{TG} - 1) \tag{2}$$

PB may differ depending on NFSDs. We assume that the maximum current operations can be performed in half of the total chips (i.e. α = number of chips / 2). TT may differ depending on token granularity (TG), which is the number of bits allocated to represent tokens for I_{max}.

B. Deadlock Free MTPM with Key

This section presents a deadlock-free multi-token based power management. The use of multi-token can cause a deadlock. Fig. 4 indicates the deadlock problem. In this example, NFSD has 1-channel and 4-ways. The number of TT is 10. Each chip may be in one of four states. Each state is described below.

- Operable State: A state that has enough tokens to perform a given operation.
- Waiting State: A state that waits for tokens to perform a given operation.
- Operating State: A state that is performing a given operation.
- Ready State: No operations to perform.

In Fig. 4(a)-i, w_0 and w_2 require 3 and 8 tokens, respectively, to perform each operation. w_0 is in the operable state because it has 10 tokens, but w_2 is in the waiting state because it does not have tokens. Other ways are in the ready state because there is no operation to process. In Fig. 4(a)-ii, the state of w_0 transits from the operable state to the operating state and processes its operation using 3 tokens. w_0 sends 7 tokens, excluding 3 tokens being used, to the next way. Since w_1 has no operation to process, it passes 7 tokens to w_2. w_2 receives 7 tokens, but still has insufficient tokens to process its operation and the waiting continues. In Fig. 4(a)-iii, w_1 receives a command that requires 4 tokens. w_1 is in the waiting state because it does not have tokens. After completing the operation of w_0, w_0 transfers 3 tokens to w_1. However, w_1 requires 4 tokens to process the operation, so the waiting state persists. As a result, both w_1 and w_2 have fewer tokens than their RTs to process the given operations and a deadlock occurs.

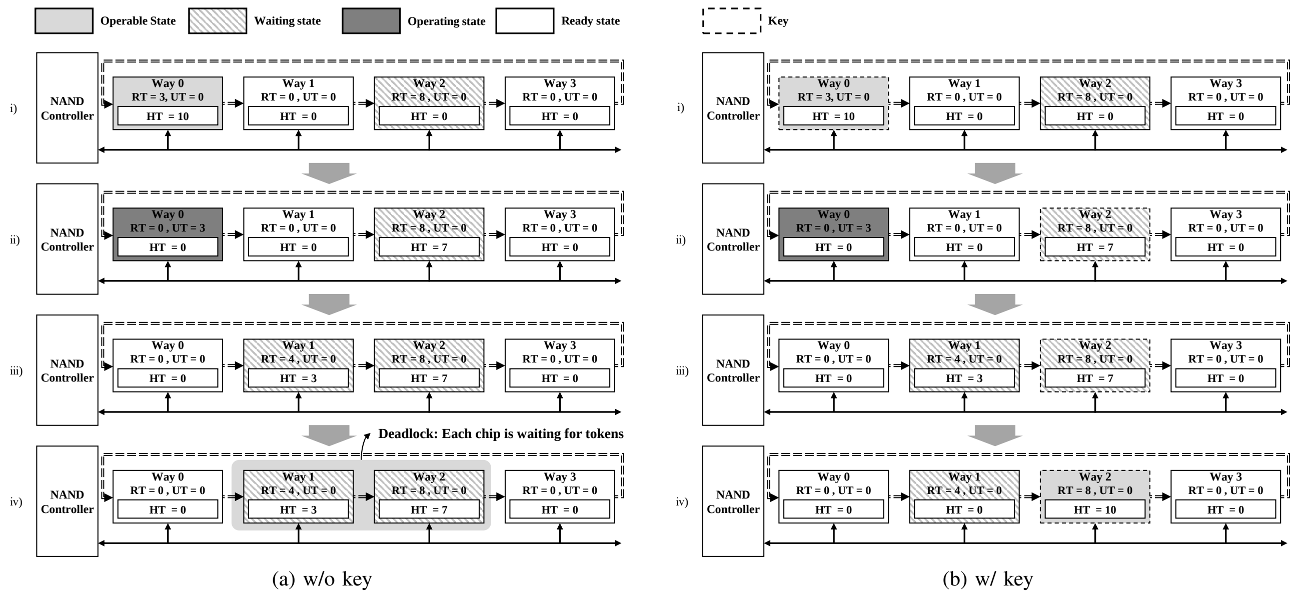

Fig. 4: Two MTPM designs showing (a) deadlock and (b) no deadlock

Multi-token based MTPM may cause a deadlock as shown in Fig. 4(a). To solve the deadlock problem of MTPM, we add the concept of 'key'. In order to process an operation, MTPM must require a key as well as tokens. In Fig. 4(b), we propose a deadlock-free key-based MTPM. Each chip must hold a key as well as tokens to process the operation. In the case of holding no key, tokens are transmitted to the next way. The deadlock can be resolved since only the chip holding a key is able to hold tokens. In Fig. 4(b)-i, w_0 is in the operable state because it owns 10 tokens that are larger than RT and a key, and w_2 is in the waiting state. In Fig. 4(b)-ii, w_0 goes into the operating state and sends the remaining 7 tokens and a key to the next way. w_1 is in the ready state and passes the key and tokens received from w_0 to w_2. Since w_2 in the waiting state holds the key, it holds the tokens until enough tokens for the operation are gathered. In Fig. 4(b)-iii, w_1 receives the operation which requires 4 tokens. It also receives 3 tokens that were used for the operation in w_0. However, since w_1 does not own the key, w_1 passes the received tokens to w_2. As a result, w_2 has enough tokens and a key to process the operation. As shown in the example, by adding a key, we solve the deadlock problem caused by multiple tokens.

The flow chart of the proposed MTPM is shown in Fig. 5. For every clock cycle, MTPM takes three different actions depending on the state of an operation request. First, when the chip is requested to perform an operation ①, MTPM checks the possession of a key. In the case that we have non-zero HT without a key, the HT is transferred to the next way and MTPM is terminated. The key and token information are transferred as a single packet. When holding a key, MTPM compares HT and RT to ensure that it has enough tokens to perform its operation. If HT is larger than RT, MTPM set properly registers and allows performing the operation. Then, the remaining tokens and a key are passed to the next way. If HT is smaller than RT, MTPM is terminated. Second, when

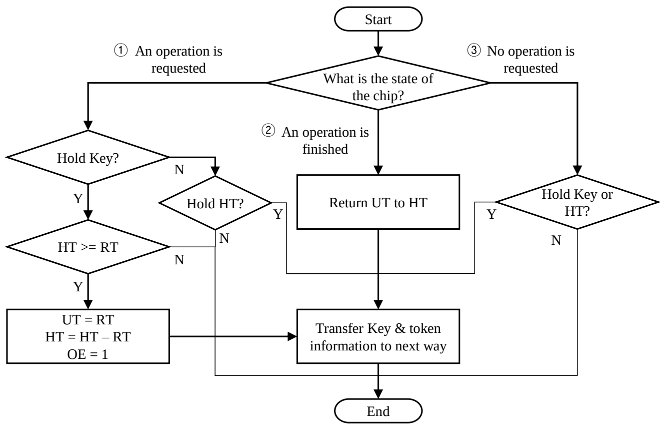
Start
① An operation is requested
③ No operation is requested
What is the state of the chip?
② An operation is finished
Hold Key?
Hold HT?
Return UT to HT
Hold Key or HT?
HT >= RT
UT = RT
HT = HT – RT
OE = 1
Transfer Key & token information to next way
End
Y
N

Fig. 5: Overall operation flow of the MTPM

an operation is finished in the chip ②, MTPM returns UT to HT and passes the key and token information to the next way. Third, when no operation is requested ③, if the chip holds a key or tokens, they are transmitted to the next way. The power budget of NFSDs is equal to the total amount of tokens which prevents power budget violation.

C. Architecture and implementation of MTPM

This section describes the architecture design of MTPM. As shown in Fig. 6, MTPM consists of operation-token converter (OTC), comparator, token receiver and sender.

The comparator can determine whether an operation can be performed in the chip. It contains four registers to store tokens and a key. The HT and key values of the comparator are received from in_PM_inform, and the RT value is received from the required_token. The comparator changes the register value according to Fig. 5 when opReq is asserted and permits the start of an operation in the chip via OE. The end of

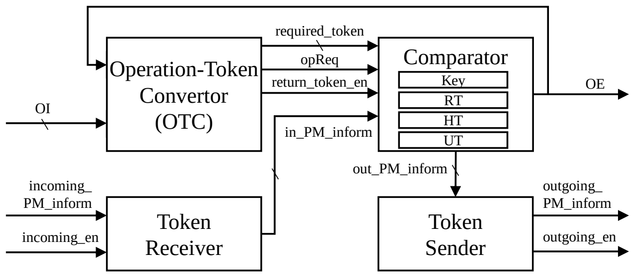

Fig. 6: Architecture of MTPM

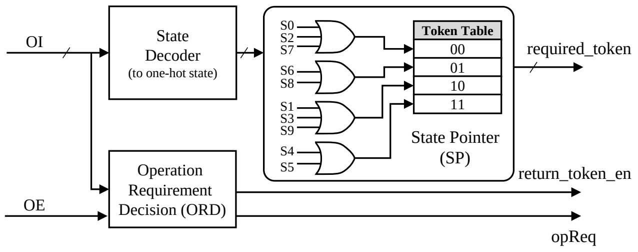

Fig. 7: Architecture of OTC

the operation can be determined via return_token_en, and the value of UT is returned to HT.

The token receiver and sender are interface blocks to move the key and tokens between the ways. They perform serial-to-parallel and parallel-to-serial conversion for key and token information. The key and token information are validated by incoming_en and outgoing_en. We serially transmit key and token information for compatibility and low hardware overhead. The size of key and token information is determined by TG. An optimal TG is different depending on the power budget and the current used for operations, which vary for different NFSDs. That is, parallel transmission of key and token information between chips requires changing the interface width according to NFSDs. Serial transmission is more flexible for varying key and token information.

The OTC indicates that a chip is waiting to perform an operation through opReq, and notifies the finish of an operation through the return_token_en. If there are operations to perform in the chip, the OI is converted into tokens and transmitted to the comparator through required_token. Fig. 7 shows the structure of OTC, which includes state decoder, operation requirement decision (ORD), and state pointer (SP).

SP sends the RT corresponding to OI input to comparator via required_token output. SP contains RTs for operations. To keep the size of the token table small, it groups operations with the same amount of tokens. Operations with the same RTs are ORed to address the token table. For this, we also add a state decoder because the input types of SP (one-hot) and OI (binary) are different. This method reduces the storage overhead required for MTPM. The number of entries in the token table is 2^{TG} and one entry size is TG bits. For TG of 3 or 4, the maximum token table size is 3 or 8 bytes. In MTPM, as all entries of the token table are not used, we can reduce the token table size. We use 27 bits and 44 bits for 3 and 4 TG, respectively.

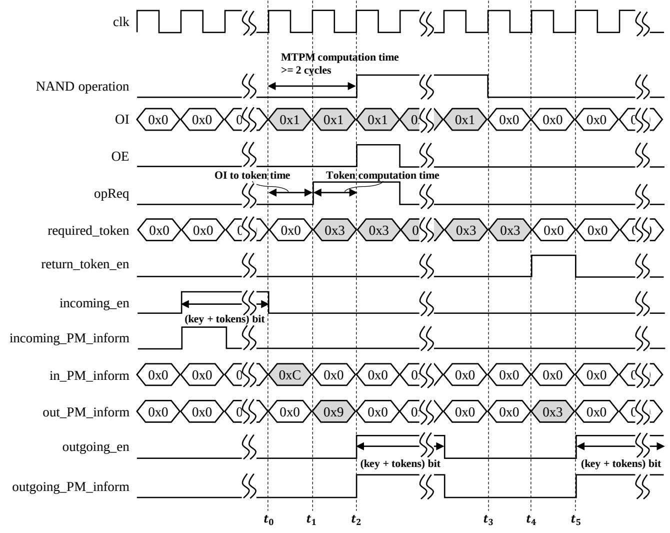

Fig. 8: Timing diagram for MTPM

ORD determines whether there is an operation to perform in the chip or an operation is finished. ORD asserts opReq to the comparator when receiving non-zero OI values which indicate there is an operation to perform in the chip. ORD deasserts opReq when OE is asserted, which indicates the start of an operation in the chip. When ORD receives zero OI which means the finish of an operation, ORD asserts return_token_en for token return.

Fig. 8 shows an example that processes an operation requiring 3 tokens in MTPM. Once MTPM has a key and enough tokens, it takes 2 cycles to calculate tokens. At time t_0, MTPM receives OI (0x1), a key, and token information via in_PM_inform (0xC == 1100_2, MSB indicates a key and the LSBs represent tokens). Then, OTC converts the OI to tokens. At time t_1, OTC asserts required_token (0x3) and opReq (0x1) to the comparator. Consequently, the comparator holds a key, 4 tokens for HT and 3 tokens for RT. It compares RT and HT and asserts OE to the chip at time t_2 because HT is larger than RT. In addition, the comparator transfers one token and a key to the token sender via out_PM_inform. At time t_2, the chip performs the operation triggered by OE, and the token sender transfers the remaining one token and the key to the next way. OI is unchanged until the operation is finished (t_3), and OTC informs the comparator that the operation is finished via return_token_en at time t_4. At time t_5, 3 tokens used to perform the operation are transmitted to the next way through the token sender.

D. Enhanced MTPM

In this section, we introduce two methods to improve the performance of MTPM. First, the operation is performed only when a key exists in MTPM. Waiting for the key incurs performance degradation. We enhance MTPM by allowing operations in the case that a chip has sufficient tokens although it does not hold a key. This method improves the performance of NFSDs by reducing waiting time.

Secondly, we further subdivide atomic operations into sub-atomic operations to overcome inefficient power utilization due

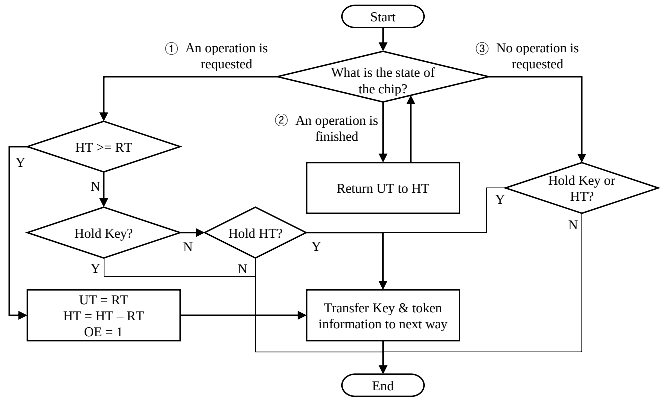
Start
① An operation is requested
What is the state of the chip?
③ No operation is requested
② An operation is finished
HT >= RT
Return UT to HT
Hold Key or HT?
Hold Key?
Hold HT?
UT = RT
HT = HT – RT
OE = 1
Transfer Key & token information to next way
End
Y
N

Fig. 9: Overall operation flow of KMTPM

to the gap between transient operation current and modeled token value based on the worst-case current. The proposed method improves the performance of NFSDs by allowing execution of more sub-atomic operations in parallel.

1) Keyless MTPM (KMTPM): Using a key in MTPM resolves the deadlock problem. In order to perform an operation, however, MTPM waits until a key is received, which decreases the performance of the NFSDs. The KMTPM allows operations to be performed without a key but still does not cause a deadlock problem.

Fig. 9 shows the overall operation flow of KMTPM. The difference from MTPM is that it can perform operations regardless of possessing a key as long as it has sufficient tokens. Also, a subsequent operation can be continuously processed after an earlier operation is finished as long as HT is greater than RT. In case of ①, KMTPM compares HT and RT. If HT is larger than RT, the operation is performed regardless of a key. In the case HT is smaller than RT, KMTPM terminates or transfers key and token information depending on whether the chip has a key and HT. In case of ②, KMTPM returns the UT to HT and goes back to the start to determine whether a subsequent operation is waiting. In case of ③, if the chip has a key or tokens, they are transmitted to the next way. The proposed MTPM performs power management based on atomic operations or sub-atomic operations (described in detail in the next section). As there are many atomic operations waiting for tokens and a key, the KMTPM effectively reduces the waiting time.

2) Sub-atomic operation based token modeling: MTPM shows a large difference in performance depending on operation modeling. Fig. 10 shows two different modeling schemes based on atomic operations and sub-atomic operations. MTPM models the current in each operation using tokens. Each modeling are assigned based on the largest current generated in each operation. In the modeling based on atomic operations, the unit of modeling interval is atomic operation. A program operation is performed by executing the loop of program execution and verify repeatedly. Fig. 10 (a) shows one loop, and we represent program execution as P$_{A1}$ and verify as P$_{A2}$ according to the operation processing sequence. Read and erase operations are also the same.

An atomic operation is divided into sub-atomic operations

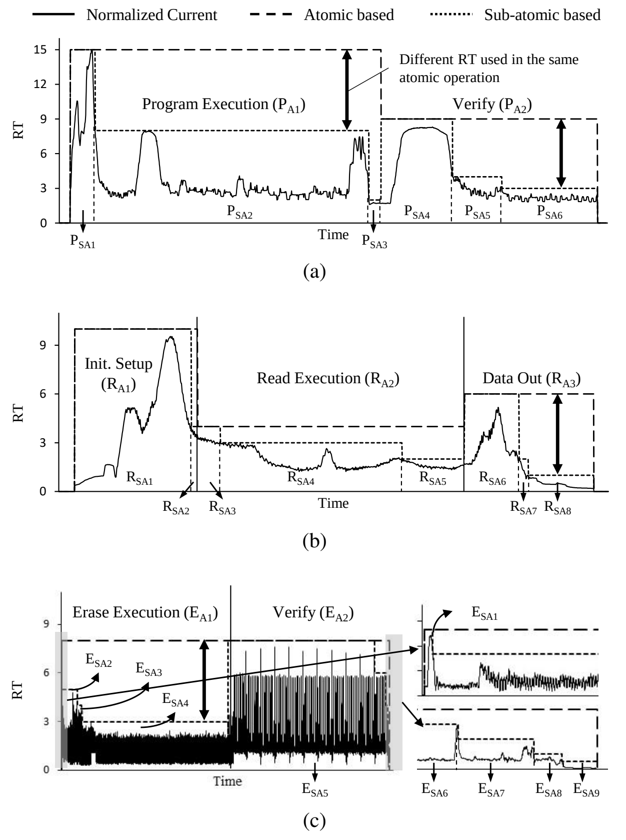

Fig. 10: Results of atomic and sub-atomic modeling schemes (TG = 4, (a) program (1 loop), (b) read, and (c) erase).

that satisfy the condition: the power budget for subsequent sub-atomic operations within an atomic operation is monotonically decreasing. Thus, tokens for subsequent sub-atomic operations decrease accordingly. Upon completion of a sub-atomic operation, surplus tokens are transmitted to the next way without suspending the subsequent sub-atomic operation.

For this, we need to define the interval for sub-atomic based modeling. We sample the maximum current usage for an atomic operation every sampling interval (500 ns). When the maximum current for the subsequent interval is equal or greater than one for the previous interval, the two intervals are combined and formed into a sub-atomic operation. We repeat this process until we find an interval with the maximum current being smaller than the currently formed sub-atomic operation. This new interval becomes the beginning segment of a new sub-atomic operation.

If a sub-atomic operation is defined with smaller intervals, the number of sub-atomic operations increases and the memory space for storing them also increases. That is, the smaller the interval, the higher the accuracy but the higher the hardware overhead. We can obtain an appropriate number of sub-atomic operations using 500 ns of modeling interval.

In the program operation, six sub-atomic operations are defined, and they are represented P$_{SA1}$ to P$_{SA6}$ according to the operation processing sequence. Similarly, read and erase operations contain eight and nine sub-atomic operations, respectively. Modeling schemes use equation (1) and (2) of

Section III-A.

When the power management is performed on atomic operations, the gap between tokens and the normalized current is huge as shown in Fig. 10. As this gap grows, performance degradation also increases as the gap reduces the number of operations executed in parallel. For instance, in Fig. 10, sub-atomic operation based modeling uses fewer tokens to process the same operation than atomic operation based modeling. In Fig. 10 (a), the modeling based on atomic operations uses 15 tokens during the P$_{A1}$. However, P$_{A1}$ is divided into 3 sub-atomic operations in the modeling based on sub-atomic operations. Each sub-atomic operation use 15, 8, 2 tokens, respectively. That is, P$_{SA1}$ uses 15 tokens, and then P$_{SA2}$ uses 8 tokens and returns 7 tokens. The returned tokens are passed to the next way and can be used for other operation. To increase power utilization, we break an atomic operation into sub-atomic operations that require smaller RTs. A key idea is that a sub-atomic operation is defined as the interval in which spare tokens from the prior sub-atomic operation may be transferred to the next way without suspending the current atomic operation until its finish. In other words, current for sub-atomic operations of an atomic operation monotonically decreases.

As the gap between the modeled token value and the normalized current decreases, we achieve the ideal performance. For this, we define sub-atomic operations to make their token values closer to the normalized current. We can achieve it by increasing TG, but it increases the token transition time between chips. Therefore, we analyze an experiment to find the optimal value of TG in Section IV-C1.

E. Overhead analysis

In this section, we analyze the overhead of the proposed method in terms of area and performance. First, we analyze the area overhead. We implement the proposed method in RTL (Register Transfer Level). To analyze overhead, we use Synopsys Design Compiler [16] for a TSMC 180 nm technology. The cell areas of MTPM, KMTPM (atomic), and KMTPM (sub-atomic) are 17124.48 um^2, 17965.44 um^2, 18950.4 um^2, respectively. Since the KMTPM (sub-atomic) stores more state information than other methods, it requires the largest area. In [17], [18], the NFM chip size is 130 mm^2 and 162.4 mm^2 respectively. The area of the proposed method KMTPM (sub-atomic) is less than 0.015 $\%$ of the NFM chip.

For TG = 4 (optimal value in Section IV-C1) and α = 4, KMTPM (sub-atomic) requires total 77 bits for the register (token table = 44 bits, HT and UT = 7 bits, RT = 4 bits, key = 1 bit, token receiver and sender = 7 bits). On the other hand, DCC in Section IV requires 144 bytes (operation information = 82 bytes, list for power management = 64 bytes).

Second, we analyze the overhead in terms of performance (latency). The proposed methods require a key and enough tokens to perform an operation. Therefore, two factors affect performance: the key and token transmission and computation time. Upon performing each operation, the following latency overhead ($L_{overhead}$) occurs.

TABLE III: Latency overhead of each operation

Operation	**Program (LSB)**		**Read**		**Erase**	
N_{atomic}	20		3		2	
$L_{overhead}$ (cycles)	Min. (n=0)	Max. (n=7)	Min. (n=0)	Max. (n=7)	Min. (n=0)	Max. (n=7)
	40	1020	6	153	4	102
$Operation\ Latency$	770 us		52 us		7.4 ms	
$L_{overhead,max}$ (%)	0.66 %		1.47 %		0.01 %	

$$L_{overhead}(cycles) = N_{atomic} \times (T_{trans} \times n + T_{comp}) \quad (0 \leq n < N_{chip\_max}) \tag{3}$$

- N_{atomic}: The number of atomic operations of each operation.
- Key and token transition time (T_{trans}): Total bits for the key and tokens (cycles). In TG = 4 and α = 4, total bits for key and tokens are 7 (1 bit for key and 6 bits for tokens). Transferring 7 bits takes 7 cycles.
- n: The maximum distance between the chip waiting to perform the operation and the chip holding the tokens or key. In this paper, the NFSD consists of 8 chips.
- MTPM computation time (T_{comp}): 2 cycles
- N_{chip_max}: The maximum number of chips in NFSD.

Sub-atomic operations within an atomic operation do not have latency overhead because they do not have to wait for the key and token (e.g. P$_{SA2}$, P$_{SA3}$, etc. in Fig. 10(a)).

Table III shows the latency overhead of the best and worst cases for each operation in proposed methods. We assume that the system clock is 200 Mhz. The proposed method has low overhead in terms of latency as reported in Table III. In MTPM, latency overhead increases as the number of NFM chips increases and the difference between current and token modeled gets larger. For this, we minimize overhead by proposing KMTPM and sub-atomic based token modeling.

F. Discussion on variations and scalability

1) Temporal and spatial variations in NFSDs: In this section, we discuss how the proposed method deals with power and latency variations due to temporal (aging) and spatial (process variation or variation in 3D technology) variations. Latency is affected by these variations, but the peak current is not affected. In [7], latency and power are measured with respect to program/erase (P/E) cycles, and they show that power variation due to P/E cycles is not significant. Therefore, we only consider the effect on latency variation.

Temporal variation is caused by P/E cycles performed on the NFM chip. As the P/E cycle increases, the tunnel oxide of the NFM cell is damaged, allowing electrons to pass through the tunnel oxide easily. Also, it is more easily programmed due to tunnel oxide trapped electrons [8], [19]. That is, the accumulation of P/E cycles reduces the latency of the program operation. In contrast, the erase operation takes longer due to trapped electrons. The read latency is not affected by temporal variation [7].

MTPM is performed using only OI of an NFM chip. MTPM reacts to the transition of atomic operations performed on the NFM chip. That is, MTPM is not sensitive to the latency change due to temporal variation. Meanwhile, KMTPM performs power management based on either atomic or sub-atomic operations. The transition of atomic operations is maintained internally by the NFM chip and transmitted to KMTPM (atomic). In this case, temporal variation is also handled by the conventional NFM chip. However, in KMTPM (sub-atomic), an NFM chip transmits OI to KMTPM after a fixed delay and should consider temporal variation. The program or read operation does not pose problems because the latency for sub-atomic operations is reduced or does not change over time even with temporal variation (aging). However, in the erase operation with increased sub-atomic operation latency, KMTPM may return tokens too early and cause power budget violations. Thus, token modeling should consider the worst-case latency for the lifetime of NFM chips. An alternative way is to apply atomic operation based token modeling for the erase operation.

Spatial variation is caused by the process variation and the structure of NFSDs (2D or 3D NFSD). We assume that the program operation repeats the loop 10 times and each loop includes program execution and verify. However, the number of loop iterations can vary depending on the characteristics of the NFM cells (i.e. the process variation). In addition, in 3D NFSDs, the page location can affect the latency of each operation. This spatial process variation does not affect MTPM or KMTPM (atomic) because they only respond to the transition of each atomic operation and do not depend on the latency of atomic operations. However, in KMTPM (sub-atomic), the timing for the sub-atomic operations should be modeled using the worst-case latency.

2) NFSDs with multiple channels: In this section, we discuss how we apply the proposed method to NFSDs with multiple channels. We can think of three potential scenarios. First, all chips in multiple channels are connected via a single ring network. Second, the proposed method is independently applied to each channel. Third, tokens between channels are transmitted via the NFM controller.

First, all chips in multiple channels are connected via a ring network. In this case, we can apply proposed methods to NFSD without modifying the protocol and interface of the NFM controller. However, the overhead of MTPM will increase as the number of chips increases because of the elongated wait time for a key and tokens to perform operations. KMTPM with sub-atomic based token modeling can reduce the overhead.

Second, the proposed method is independently applied to each channel. The proposed method can be applied to NFSD without modifying the protocol and interface of the NFM controller. The FTL (Flash Translation Layer) distributes operations evenly through channel interleaving [20], [21] and bursty operations are not likely sent to one channel. However, sophisticated scheduling would help prevent uneven command distribution.

Third, tokens between channels are transmitted via the NFM controller. In this case, the centralized NFM controller continuously monitors the state of each chip and transmits tokens via a shared-bus (channel). However, as the data size increases, the channel occupancy increases, which and disturbing monitoring and transmitting of tokens by the NFM controller. One way to solve this problem is to add a separate path between the NFM chips and the controller. However, this method increases the complexity of the NFM controller by changing the protocol and interface.

TABLE IV: Configuration of NFSDs

Parameter	Value
Number of channel/way	1 / 8
tPROG (LSB / MSB)	770us / 1.022ms
tR	52us
tBER	7.4ms
System clock [22]	200Mhz

IV. Experimental results

A. Experimental Setup

We have designed a Synopsys Platform Architect based trace-driven simulator to evaluate the proposed MTPM, which includes an NFM controller, NFMs, and MTPM [23]. The NFM controller and NFMs are implemented by systemC TLM (Transaction Level Modeling), and MTPM is implemented in RTL.

The NFM controller receives the address, command, and length included in the trace and delivers commands to each chip according to the address. Each chip performs an operation according to the command and transfers the state of the operation to the MTPM through OI before performing an operation. After that, each chip performs an operation when an enable signal is received from the MTPM through the OE. Each chip notifies the NFM controller that the operation has been completed after the operation ends.

Table IV indicates the timing parameter and configuration of NFM. The tPROG, tR, and tBER are the time of a program, read, and erase, respectively. We model both LSB (Least Significant Bit) and MSB (Most Significant Bit) program operation. A program operation iteratively executes a loop, consisting of program execution and verify. In this paper, we assume a loop is iterated 10 times for both LSB and MSB programs. In [1], [24], the loop is iterated from 6 to 16 times depending on NFSDs or the cell characteristics. Therefore, assuming 10 for the number of loop iterations is reasonable. In additional, the difference between the LSB and MSB programs is that the number of verify in each iteration. A loop of the LSB program includes one program execution and one verify. Meanwhile, a loop of the MSB program includes one program execution and several verify operations (up to 3 times) [1]. We measure the current for each operation every 40 ns, and the simulator uses the measurement data. We use a synthetic trace because power violation occurs when chips process the burst operations. A trace consists of 5000 operations and one erase is inserted every 500 program operations. We vary the ratio of reads and programs. The ratio of reads is increased with the interval of 25% from 0%, which generates 5 traces (R0, R25, R50, R75, R100). We also use real traces collected from various systems. The read-program ratio of these traces vary. The read ratio of Financial1/2, Web1, MSR1, and TPC-C are 23%, 82%, 99%, 12%, and 64%, respectively [25], [26].

We evaluate performance for six methods described below.

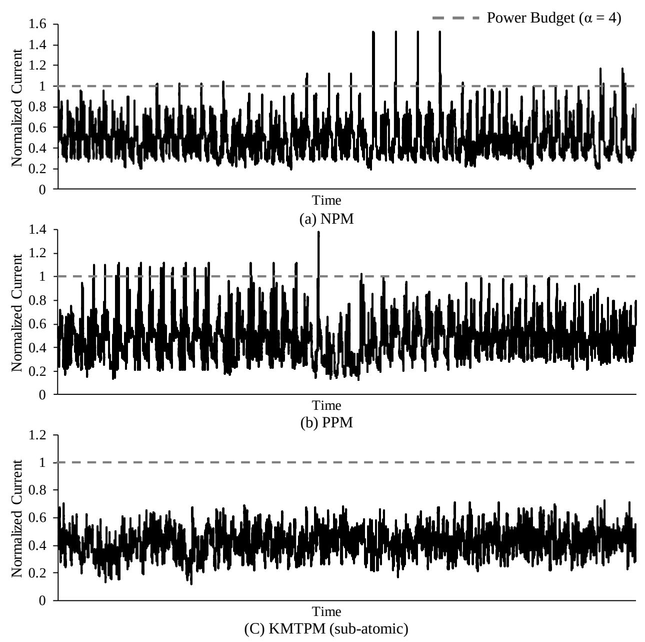

Fig. 11: Normalized current of (a) NPM, (b) PPM, and (c) KMTPM (sub-atomic) for trace R50

- No-power management (NPM): Applying no power management scheme.
- Peak Power management (PPM) [3], [6]: Applying power management considering only the pre-charge phase of the program operation. Only one chip performs the precharge phase at the same time.
- Dynamic Current Capping (DCC) [7]: Calculating the current of the operations in NFM controller and determining the start time of each operation.
- MTPM: Requiring enough tokens and a key to start operations. Power management is performed on an atomic operation basis.
- KMTPM (atomic): Relaxing MTPM by starting an operation provided with enough tokens. Power management is performed on an atomic operation basis.
- KMTPM (sub-atomic): Same as KMTPM (atomic) except for power management being performed on a sub-atomic operation basis.

B. Experimental Results for power budget violation

Fig. 11 indicates the normalized current for NPM, PPM, KMTPM (sub-atomic). Y-axis is the current normalized to power budget of α = 4. As the simulator includes 8 chips, the power budget of α = 4 means that the maximum current operations can be performed in the half of the total chips. Fig. 11(a) and (b) are the results for the R50 trace.

Experimental results confirm that power violation occurs frequently. The maximum current of Fig. 11(a) is larger by 53.18% compared to the power budget and the maximum of Fig. 11(b) is also larger by 38.01%. We confirm that NPM and PPM incur power violations on all traces as well as R50 trace. Especially, in Fig. 11(b), despite power management over precharge phases, power budget violations in NFSDs occur due to the overlapping of PPZ and NPPZ as mentioned in Section II. The results show that previous works cannot prevent power violation completely. However, Fig. 11(c), we confirm that KMTPM (sub-atomic) performs operations within the power budget.

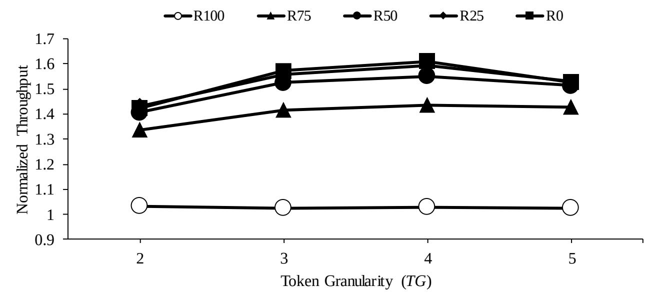

Fig. 12: Throughput of KMTPM (sub-atomic) normalized to that of MTPM (TG = 2)

C. Experimental Results for performance

Since proposed MTPM performs operations constrained by tokens, it can prevent power budget violations in NFSDs. However, it may cause performance degradation. We propose the methods to improve performance and compare results with the best prior works.

1) Throughput with respect to modeling granularity: Increasing TG improves the performance by reducing the difference between the amount of required tokens and maximum current for a given operation. However, the increment of TG increases TT, resulting in more token storage space and more cycles to transfer tokens between chips. To find an optimal TG, we measure the throughput of KMTPM (subatomic) under various TG configurations while varying read and program ratios, as shown Fig. 12. The x-axis and the y-axis represent the modeling token granularity and the throughput normalized to MTPM (TG = 2), respectively.

Fig. 12 shows a similar trend for all traces except the R100 trace. R100 consists of only read operations consuming fewer tokens than other operations. Therefore, it rarely waits for tokens, and the performance shows not much difference for different TG values. For traces other than R100, the throughput are maximum around TG = 4. In TG = 5, the throughput decreases because of the large overhead due to token transfer with respect to the performance improvement due to TG increase. Therefore, we use TG of 4 for further analysis.

2) Average throughput: Fig. 13 shows the improvement in throughput by the proposed methods when TG and α are 4 and we use synthetic traces. Y-axis is normalized to the throughput of the NPM method. In Fig. 13, NFSD cannot perform operations in parallel on all chips due to the given power beudget (α = 4). That is, performance degradation occurs to meet the power budget.

On average, the throughput of KMTPM (sub-atomic) is decreased by 6.21% compared to that of NPM. KMTPM (sub-atomic) has less throughput degradation compared to other methods guaranteeing power budget violation in most

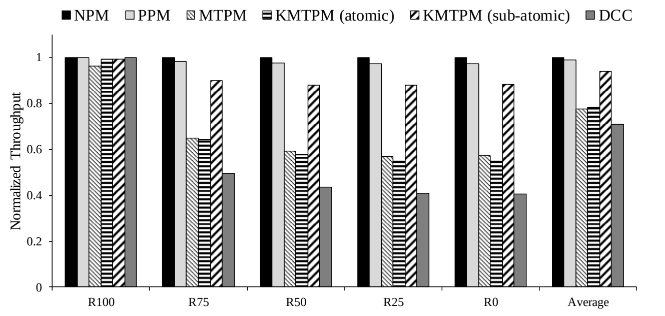

Fig. 13: Throughput normalized to NPM in synthetic traces (α=4, TG=4)

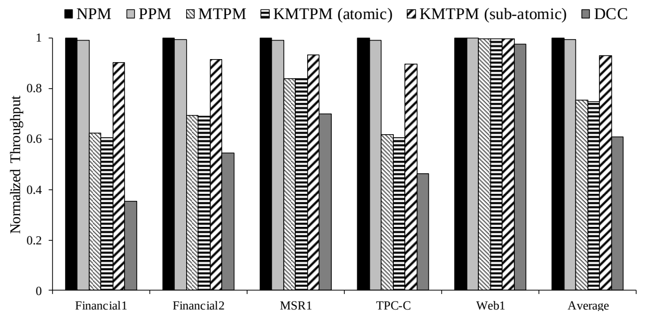

Fig. 14: Throughput normalized to NPM in real traces (α=4, TG=4)

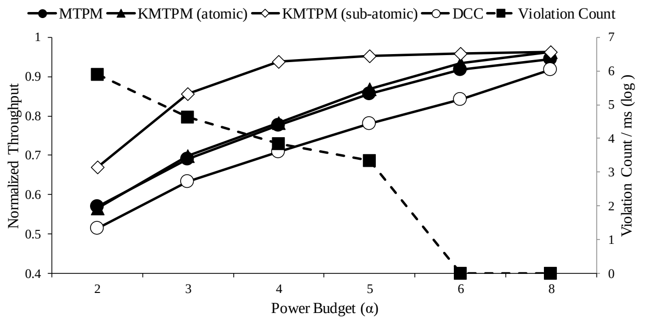

Fig. 15: Average throughput normalized to that of NPM and violation count of NPM with respect to power budget

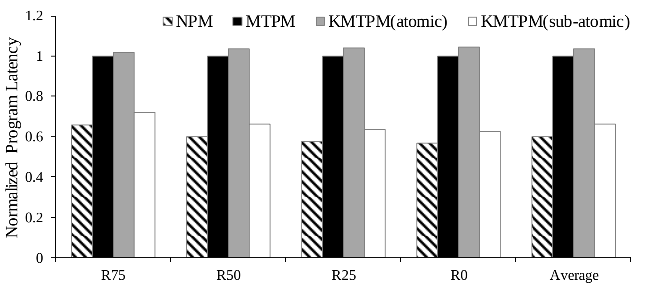

Fig. 16: Program latency normalized to MTPM (TG=4)

traces except for R100. In R100 that consists of only read operations, performance degradation is small. Since a read operation consumes a small number of tokens, waiting time for tokens and a key is short, performance degradation is not significant.

Throughput for MTPM and KMTPM (atomic) are decreased by 22.44% and 21.65% compared to that of NPM. But both methods show less throughput degradation than DCC. The throughput of KMTPM (atomic) is lower by 0.79% than MTPM. This is caused by the increased program latency of KMTPM (atomic), which we describe in detail in Section IV-C4. KMTPM (sub-atomic) improves the average throughput by 22.85% compared to that of NPM's normalized DCC. KMTPM (sub-atomic) reduces waiting time for the key by performing operations as long as there are enough tokens in the chip. Also, a small difference between current and modeled tokens due to fine token modeling leads to good performance by allowing more parallel execution of operations.

DCC precomputes the current of an operation to perform and determines the start time of the operation to prevent power budget violation of NFSDs. If the scheduling result of the operation causes power budget violation at time t_x, the start time of the operation is delayed after t_x. DCC has low throughput on the trace that contains read operations. As read operation time is shorter than program operation time, it is greatly affected by delayed time. PPM shows similar throughput on overall traces compared to NPM. However, as shown in Fig. 11, PPM frequently shows the power budget violations which cause serious problems in NFSDs.

Fig. 14 shows the improvement in throughput by the proposed methods when TG and α are 4 and we use real traces. The trend of throughput for real traces is similar to synthetic traces. In the read-intensive trace (e.g. Web1), the performance degradation of all methods is small compared to NPM. KMTPM (sub-atomic) shows good performance for all traces and, on average, the throughput is decreased only by 7.1% compared to NPM which has power budget violations.

3) Relationship between performance and power budget: Fig. 15 shows the average throughput of each method with respect to the power budget in synthetic traces. The x-axis represents the power budget of NFSD depending on the α value of the equation (2), whereas the y-axes on the left and the right represent the average throughput and violation count per milliseconds (log scale), respectively. We measure violation count every 40ns.

In Fig. 15, as the power budget increases, the count of power budget violations for NPM decreases and the throughputs increase as more operations are performed in parallel. The power budget of α = 8 is the case where the maximum current operations are simultaneously performed on all chips. That is, each method performs operations without the constraint of the power budget. Therefore, for α = 8, throughput difference from NPM can be regarded as a performance overhead. KMTPM (sub-atomic) and DCC have performance overhead of 3.8% and 8.21%, respectively.

For α = 4 and 5, the NPM suffers from power budget violations. In KMTPM (sub-atomic), throughput is saturated beyond α = 4. This means that KMTPM (sub-atomic) has the optimal performance beyond α = 4.

4) Average latency: In this section, we analyze the proposed methods in terms of latency. The latency is the time until an operation is completed after the NFM controller transmits the operation to the NFM chip. KMTPM (atomic) relaxes

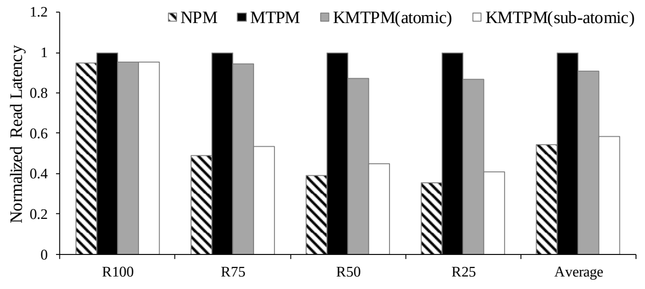

Fig. 17: Read latency normalized to MTPM (TG = 4)

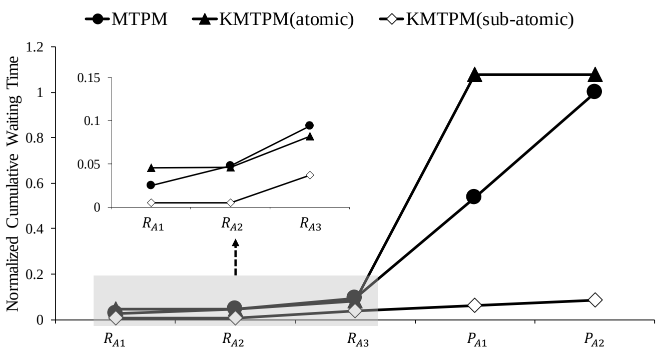

Fig. 18: Cumulative waiting time for the key and tokens in atomic operations (normalized to that of MTPM)

the wait time for a key compared to MTPM, but has similar throughput in Section IV-C2. This is due to the increase in program latency and the decrease in read latency. KMTPM (atomic) increases the program latency by 3.6% compared to MTPM. However, KMTPM (atomic) decreases read latency by 9.12%, which has the greater effect on overall system performance [14], [27].

Fig. 16 shows the improvement in program latency of the proposed method for TG of 4. Y-axis is normalized to the program latency of the MTPM method.

On average, program latency increases by 3.6% in KMTPM (atomic) and decreases by 33.9% in KMTPM (sub-atomic). KMTPM (sub-atomic) decreases by as much as 37.27% compared to the MTPM in R0. It decreases program latency for all traces. In addition, the program latency of KMTPM (sub-atomic) increases by only 5.97% compared to the NPM which suffers from power budget violations.

However, in KMTPM (atomic), program latency slightly increases compared to the MTPM. The program operation consists of two atomic operations - P_{A1} and P_{A2} (in Section III-D2). P_{A1} includes the pre-charge phase which requires the most tokens. P_{A2} requires fewer tokens than the first one. MTPM transmits the used tokens for P_{A1} to the next way after completing P_{A1}. As a result, the execution of P_{A2} for competing program operations are serialized. After the first states (P_{A1}) are completed, the second states (P_{A2}) are processed in parallel as they need smaller tokens. However, a program operation in KMTPM (atomic) processes the P_{A2} immediately after the P_{A1}, transmitting remaining tokens to the next way. However, tokens are insufficient to initiate P_{A1} in the next way, which increases the program latency. More details on this are discussed in the next section.

Fig. 17 presents the average read latency. As shown in Fig. 17, read latency of KMTPM (sub-atomic) is improved in all traces. On average, KMTPM (atomic) and KMTPM (sub-atomic) are improved by 9.12% and 41.45% and by up to 13.39% and 59.28% in R25. KMTPM (atomic) significantly improves read latency compared to program latency reduction by reducing wait time of a key. Similarly, read latency of KMTPM (sub-atomic) increases by only 3.98% compared to the NPM.

5) Token & key wait time: Fig. 18 shows the result of cumulative waiting time for tokens and the key at each chip when TG is 4. Waiting time occurs when there is no key or insufficient tokens to perform an operation. Fig. 18 is the waiting time of atomic operations as waiting only occurs at the start of an atomic state that can be suspended or resumed. Y-axis is normalized to MTPM.

KMTPM (sub-atomic) has a short waiting time for all atomic operations. KMTPM (sub-atomic) processes the same operation with fewer tokens and it can use the remaining tokens to reduce the waiting time of other operations.

As in Section IV-C4, KMTPM (atomic) shows good read performance because the waiting time of read operation compared to MTPM is short. However, the program waiting time of KMTPM (atomic) is larger than MTPM. In KMTPM (atomic), waiting occurs in P_{A1} which performs a pre-charge phase that requires the maximum current. Let's illustrate the difference between MTPM and KMTPM (atomic) with an example. Chip 0 is processing P_{A1} and chip 1 is waiting to process P_{A1}. In case of MTPM, used tokens are transmitted to chip 1 after completing P_{A1} so that chip 1 is able to process P_{A1}. However, in case of KMTPM(atomic), chip 0 continues to process P_{A2} right after processing P_{A1} and transmits only the remaining tokens to chip 1. Since the transmitted tokens are not enough to process P_{A1}, the chip 1 must wait. For this reason, program latency increases for KMTPM (atomic).

Let's take another example. Assume that chip 0 is processing P_{A1} and chip 1 is waiting to process R_{A1}. In MTPM, chip 1 processes R_{A1} after chip 0 completes P_{A1}, and chip 0 waits for P_{A2} because of insufficient tokens. However, in KMTPM (atomic), chip 0 processes P_{A2} and chip 1 processes R_{A1} at the same time after processing P_{A1}. Although KMTPM (atomic) improves parallelism by performing operations even if there is no key, waiting time may occur when it processes operations like P_{A1} requiring a lot of tokens. We solve this problem with sub-atomic based token modeling.

V. Conclusion

In this paper, we propose a novel power management scheme using tokens and a key, which guarantees no power budget violation. This method improves performance of throughput by maximizing parallel execution of operations. Especially, KMTPM (sub-atomic) maximizes token utilization by fine-grained token modeling of sub-atomic operations.

KMTPM (sub-atomic) only incurs performance overhead of 3.8% versus NPM and area overhead of 0.015% versus conventional NFM chip. The results of KMTPM (sub-atomic) method show that throughput decreases by 6.21% compared to NPM, but it outperforms the best existing scheme (DCC) by 22.85% in throughput.

In this paper, we evaluate the power management schemes using one channel. Transmitting tokens between channels increases token utilization, which can increase performance. Therefore, we will study power management for multiple channels in future work. Also, we will enhance the proposed method considering the effects of temporal (aging) and spatial (process variation or variation in 3D technology) variations in the future work.

REFERENCES

[1] Rino Micheloni, Alessia Marelli, and Kam Eshghi. *Inside solid state drives (SSDs)*, volume 37. Springer Science & Business Media, 2012.

[2] George Lawton. Improved flash memory grows in popularity. *Computer*, 39(1):16–18, 2006.

[3] Mario Sako, Yoshihisa Watanabe, Takao Nakajima, Jumpei Sato, Kazuyoshi Muraoka, Masaki Fujiu, Fumihiro Kono, Michio Nakagawa, Masami Masuda, Koji Kato, et al. A low power 64 gb mlc nand-flash memory in 15 nm cmos technology. *IEEE Journal of Solid-State Circuits*, 51(1):196–203, 2016.

[4] Balgeun Yoo, Youjip Won, Seokhei Cho, Sooyong Kang, Jongmoo Choi, and Sungroh Yoon. Ssd characterization: From energy consumption's perspective. In *HotStorage*, 2011.

[5] Jinha Park, Sungjoo Yoo, Sunggu Lee, and Chanik Park. Power modeling of solid state disk for dynamic power management policy design in embedded systems. In *IFIP International Workshop on Software Technolgies for Embedded and Ubiquitous Systems*, pages 24–35. Springer, 2009.

[6] Ken Takeuchi. Novel co-design of nand flash memory and nand flash controller circuits for sub-30 nm low-power high-speed solid-state drives (ssd). *IEEE Journal of Solid-State Circuits*, 44(2):1227–1234, 2009.

[7] Li-Pin Chang, Chia-Hsiang Cheng, Shu-Ting Chang, and Po-Han Chou. Current-aware flash scheduling for current capping in solid state disks. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, 2018.

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

[9] Andrew Hay, Karin Strauss, Timothy Sherwood, Gabriel H Loh, and Doug Burger. Preventing pcm banks from seizing too much power. In *Proceedings of the 44th Annual IEEE/ACM International Symposium on Microarchitecture*, pages 186–195. ACM, 2011.

[10] Lei Jiang, Youtao Zhang, Bruce R Childers, and Jun Yang. Fpb: Fine-grained power budgeting to improve write throughput of multi-level cell phase change memory. In *Proceedings of the 2012 45th Annual IEEE/ACM International Symposium on Microarchitecture*, pages 1–12. IEEE Computer Society, 2012.

[11] Jin-Young Kim, Sang-Hoon Park, Hyeokjun Seo, Ki-Whan Song, Sungroh Yoon, and Eui-Young Chung. Nand flash memory with multiple page sizes for high-performance storage devices. *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, 24(2):764–768, 2015.

[12] Yejia Di, Liang Shi, Congming Gao, Qiao Li, Kaijie Wu, and Chun Jason Xue. Loss is gain: Shortening data for lifetime improvement on low-cost ecc enabled consumer-level flash memory. In *ACM Great Lakes Symposium on VLSI*, pages 225–230, 2018.

[13] Jin-Ki Kim, Koji Sakui, Sung-Soo Lee, Yasuo Itoh, Suk-Chon Kwon, Kazuhisa Kanazawa, Ki-Jun Lee, Hiroshi Nakamura, Kang-Young Kim, Toshihiko Himeno, et al. A 120-mm/sup 2/64-mb nand flash memory achieving 180 ns/byte effective program speed. *IEEE Journal of Solid-State Circuits*, 32(5):670–680, 1997.

[14] Guanying Wu and Xubin He. Reducing ssd read latency via nand flash program and erase suspension. In *FAST*, volume 12, pages 10–10, 2012.

[15] Hiroshi Maejima, Kazushige Kanda, Susumu Fujimura, Teruo Takagiwa, Susumu Ozawa, Jumpei Sato, Yoshihiko Shindo, Manabu Sato, Naoaki Kanagawa, Junji Musha, et al. A 512gb 3b/cell 3d flash memory on a 96-word-line-layer technology. In *2018 IEEE International Solid-State Circuits Conference-(ISSCC)*, pages 336–338. IEEE, 2018.

[16] Design compiler. Synopsys Inc.

[17] Seungjae Lee, Jin-yub Lee, Il-han Park, Jongyeol Park, Sung-won Yun, Min-su Kim, Jong-hoon Lee, Minseok Kim, Kangbin Lee, Taeeun Kim, et al. 7.5 a 128gb 2b/cell nand flash memory in 14nm technology with tprog= 640μs and 800mb/s i/o rate. In *Solid-State Circuits Conference (ISSCC), 2016 IEEE International*, pages 138–139. IEEE, 2016.

[18] Chulbum Kim, Jinho Ryu, Taesung Lee, Hyunggon Kim, Jaewoo Lim, Jaeyong Jeong, Seonghwan Seo, Hongsoo Jeon, Bokeun Kim, Inyoul Lee, et al. 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*, 47(4):981–989, 2012.

[19] Ellis H Wilson, Myoungsoo Jung, and Mahmut T Kandemir. Zombienand: Resurrecting dead nand flash for improved ssd longevity. In *2014 IEEE 22nd International Symposium on Modelling, Analysis & Simulation of Computer and Telecommunication Systems*, pages 229–238. IEEE, 2014.

[20] Yang Hu, Hong Jiang, Dan Feng, Lei Tian, Hao Luo, and Shuping Zhang. Performance impact and interplay of ssd parallelism through advanced commands, allocation strategy and data granularity. In *Proceedings of the international conference on Supercomputing*, pages 96–107. ACM, 2011.

[21] Ji-Yong Shin, Zeng-Lin Xia, Ning-Yi Xu, Rui Gao, Xiong-Fei Cai, Seungryoul Maeng, and Feng-Hsiung Hsu. Ftl design exploration in reconfigurable high-performance ssd for server applications. In *Proceedings of the 23rd international conference on Supercomputing*, pages 338–349. ACM, 2009.

[22] Hynix Semiconductor et al. Open nand flash interface specification. *Technical Report ONFI*, 2006.

[23] Synopsys virtual prototyping solution. http://www.synopsys.com /systems/virtualprototyping/pages/default.aspx, 2013. Synopsys Inc.

[24] Yong Sung Cho, Il Han Park, Sang Yong Yoon, Nam Hee Lee, Sang Hyun Joo, Ki-Whan Song, Kihwan Choi, Jin-Man Han, Kye Hyun Kyung, and Young-Hyun Jun. Adaptive multi-pulse program scheme based on tunneling speed classification for next generation multi-bit/cell nand flash. *IEEE Journal of Solid-State Circuits*, 48(4):948–959, 2013.

[25] Umass trace repository.0000 http://traces.cs.umass.edu/.

[26] Storage networking industry association, 2011. http://iotta.snia.org.

[27] Minkyeong Lee, Dong Hyun Kang, Minho Lee, and Young Ik Eom. Improving read performance by isolating multiple queues in nvme ssds. In *Proceedings of the 11th International Conference on Ubiquitous Information Management and Communication*, page 36. ACM, 2017.

Taehee You received the B.S. degree in electrical and electronic engineering from Yonsei University in Seoul, Korea, in 2009. He is currently a Ph.D. candidate in Yonsei University. His research interests include high performance system architecture and VLSI design with the special emphasis on NVM memory applications.

Sangwoo Han received the B.S. degree in electrical and electronic engineering from Yonsei University in Seoul, Korea, in 2014. He is currently a Ph.D candidate in Yonsei University. His research interests include System-level architecture and design, Advanced storage systems and applications.

Young Min Park received the BS degree in electrical and electronic engineering from Yonsei University, Seoul, Korea, in 2015, where he is currently working toward the PhD degree in electrical and electronic engineering. His research interests include solid-state disk system architecture and CAD flow on near threshold voltage.

Hyuk-Jun Lee received the BS degree in computer engineering from University of Southern California, Los Angeles, CA, in 1993, and the MS and PhD degrees in electrical engineering from from Stanford University, Stanford, CA, in 1995 and 2001, respectively. From 2001 to 2011, he served as a senior engineer in routing technology group at Cisco System, San Jose, CA, where he participated in developing CRS-1 and CRS-3. Currently, he is a professor with the Department of Computer Science and Engineering, Sogang University, Seoul, Korea. His research interests include embedded systems, low-power design, and memory/storage architectures.

Eui-Young Chung received the BS and MS degrees in electronics and computer engineering from Korea University, Seoul, Korea, in 1988 and 1990, respectively, and the PhD degree in electrical engineering from Stan-ford University, Stanford, California, in 2002. From 1990 to 2005, he was a principal engineer with SoC R&D Center, Samsung Electronics, Yongin, Korea. Currently, he is a professor in the School of Electrical and Electronics Engineering, Yonsei University, Seoul, Korea. His research interests include system architecture, bio-computing, and VLSI design, including all aspects of computer-aided design with the special emphasis on low power applications, and flash memory applications. He is a member of the IEEE.