# Fast performance analysis of NAND flash-based storage device

S.K. Won, S.H. Ha and E.Y. Chung

A fast performance analysis method for NAND flash-based storage devices (NFSDs) is proposed. The method first profiles the operational statistics of an NFSD and then estimates its throughput based on the proposed model. Experimental results show that its accuracy reaches up to 98.7% of the cycle-accurate simulation, while the analysis speedup is more than three orders of magnitude. Also, the generality of the method is shown by applying it to NFSDs with an arbitrary number of channels.

*Introduction:* A NAND flash-based storage device (NFSD) has received much attention as a next generation storage device. One of its advantages is its read/write throughput compared to hard disk drives (HDDs). However, its performance is still insufficient to amortise its bit cost which is higher than that of HDDs. For this reason, efforts to improve the performance of NFSDs are continuing.

A typical architecture of an NFSD is shown in Fig. 1. It consists of two parts: a controller and NAND flash memories (NFMs). The NFMs store the data and the controller manages the data as well as the hardware resources. These two parts are communicating with each other through channels. Furthermore, several NFMs are connected to a single channel in a time-multiplexed (interleaved) manner. The number of channels determines the overall bandwidth, while the interleaving depth (the ways) affects the channel utilisation. Another performance critical component is a software layer running on the processor called the flash translation layer (FTL). It translates the logical addresses from the host machine to the physical addresses of NFMs. Also, it performs garbage collections to hide the bad characteristics of NFMs such as the limited cell lifetime and erase-before-write. Depending on its algorithm, the NFM access pattern largely varies, which eventually affects the channel utilisation and the overall performance of an NFSD. For this reason, we need to consider the effect of FTLs in conjunction with the effect of the channels and ways for analysing the overall performance. Unfortunately, most of the previous works focusing on FTLs ignored the hardware (channels and ways) effect [1], while the hardware oriented works ignored the effect of the FTL [2]. In these works, the hardware modelling effort and the simulation speed are the major obstacles to take both effects into consideration. We tackle this issue in the work reported in this Letter.

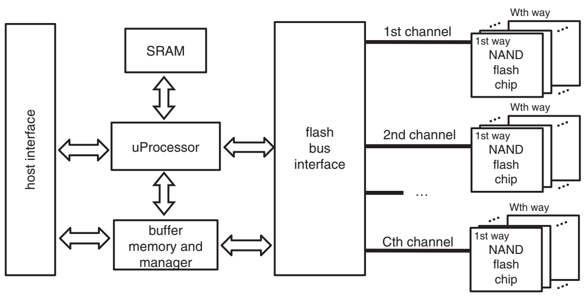

**Fig. 1** *Architecture of NAND flash-based storage system*

*Preliminaries:* An NFM consists of multiple blocks. Each block is organised with multiple pages. A block and a page are the base units for erase operations and read/program (write) operations, respectively. The read operation consists of three phases. In command phase ($t_{CMD}$), a command is asserted. In busy phase ($t_{BUSY}$), a page is read from the cell array of an NFM to its page register. Finally, it is transferred to the controller during the transfer phase ($t_{TRANS}$). The program operation is similar to the read operation except that the transfer phase precedes the busy phase. We denote $t_{BUSY}$ as $t_R$ in read and as $t_{PROG}$ in write, since $t_{PROG}$ is about 10 times larger than $t_R$. Also, the erase operation consists of command phase and erase phase ($t_{BER}$).

To update a page already written, the page should be erased before being overwritten. Unfortunately, the erase operation is performed for all the pages in a block rather than a specific page. For this reason, an FTL copies the valid pages in the block to an empty block and writes the new data to the new block. The old block is erased for future use. The extra operations incur a large performance overhead. When no more empty block is left, a merge operation occurs to prepare an empty block by exploiting unused pages of used blocks. It also incurs a large amount of erase and copy operations. The number of extra operations critically depends on the algorithm of an FTL. Hence, the performance analysis of an NFSD should reflect the FTL's behaviour.

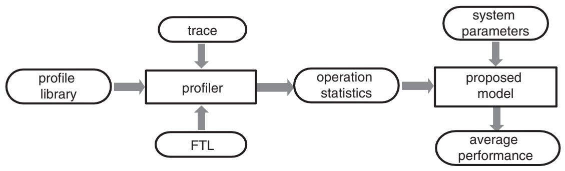

**Fig. 2** *Overall flow for fast architecture exploration*

*Proposed method:* Fig. 2 shows the overall flow of the proposed method. The profiler links an FTL with the profile library for gathering the operational statistics for the given trace. The statistics include the number of erase operations, the number of copying operations, and the number of merge operations. In the next step, the analytical model takes the statistics as an input and estimates the throughput for the given hardware parameters. The profiler is similar to those in [3], hence we omit its details and focus on the analytical performance model. We describe the analytical model in an incremental way. First, we show a performance model for a single-channel/single-way NFSD. Then, we expand the model for an NFSD with multiple channels. Finally, we provide a performance model for an NFSD with an arbitrary number of channels and ways.

In general, P, the throughput of an NFSD is expressed as follows:

$$P = \frac{1}{N}\sum_{n=1}^{N}\frac{Q_n}{T_n} \qquad (1)$$

where N is total number of the requests and $Q_n$ is the data size of *n*th requests. If the page size is S, it is required to transfer $\alpha$ pages given by $Q_n/S$. $T_n$ is the response time of an NFSD for $Q_n$ and defined as follows:

$$T_n = T_{BUF} + T_{OP} + T_M \qquad (2)$$

$T_{BUF}$ and $T_{OP}$ represent the latencies for transferring $Q_n$ to/from the buffer (shown in Fig. 1) and transferring $Q_n$ to/from the NFMs, respectively. Also, $T_M$ is the time for merge operations. It is zero in read mode, since the merge operation only occurs when a page is updated. $T_{BUF}$ is defined as $\rho \times Q_n \times t_{bclk}$, where $t_{bclk}$ is the clock frequency of the buffer and $\rho$ is a scaling factor to adjust the clock frequencies of the buffer and the controller. On the other hand, $T_{OP}$ is the read/write time scaled by $\alpha$ as follows:

$$T_{OP} = \begin{cases} \lceil\alpha\rceil\, t_{CMD} + \alpha\, t_{TRANS} + \lceil\alpha\rceil\, t_{PROG} & \text{in write mode} \\ \lceil\alpha\rceil\, t_{CMD} + \lceil\alpha\rceil\, t_R + \alpha\, t_{TRANS} & \text{in read mode} \end{cases} \qquad (3)$$

Note that $\alpha$ for $t_{TRANS}$ is to consider when $Q_n$ is not the multiple of S. Finally, $T_M$ for merge operations is defined as follows:

$$T_M = m\,(2\,t_{CMD} + t_R + t_{PROG}) + q\,t_{BER} \qquad (4)$$

where m is the number of copying pages and q is the number of erased blocks. Both are obtained from the profiling.

Next, we consider the NFSDs with multiple channels (a single way for each channel) and denote the number of channels as C. We expand the model by simply redefining $\alpha$ such that $\alpha = Q_n/(C \cdot S)$. It means that higher performance gain is possible by processing the data in parallel. Even if $Q_n < S$, $Q_n$ is divided into smaller pieces to maximise the parallelism.

Finally, we generalise our model for multichannel/multi-way NFSDs. We denote the number of ways per channel as W. Since $\alpha$ is the total work on a single channel, the work for each way is $\alpha/W$. The complete write/read of $Q_n$ needs to iterate W-way interleaving $\alpha/W$ times. We show the interleaving behaviour in write operation in Fig. 3. The interleaving starts from the first way. After the completion of its data transfer phase, the second way immediately starts. Hence, total time for the command phase and data transfer phase of all the ways is $W \cdot (t_{CMD} + t_{TRANS})$ in the first iteration. The first iteration takes additional $t_{PROG}$ from the last way, since it cannot be hidden by other ways. The same behaviour occurs in other iterations. On the other hand, the second iteration may start earlier than the completion of the first iteration if the following two conditions are met. First, the busy phase of the first way is completed. Secondly, the data transfer phase of the last way is completed. Then, the second iteration starts as early as K which is max($t_{PROG}$-(W-1) • ($t_{CMD}$ + $t_{TRANS}$), 0) with respect to the end of the first iteration. Similar analysis is possible for read operation but omitted due to limited space. Finally, we revise $T_{OP}$ for multichannel/multi-way NFSDs as follows:

$$T_{OP} = \begin{cases} \lceil\alpha\rceil t_{CMD} + \alpha t_{TRANS} + t_{PROG} + K(\lceil\alpha/W\rceil - 1) & \text{in write mode} \\ \lceil\alpha\rceil t_{CMD} + \lceil\alpha\rceil t_{TRANS} + \lceil\alpha/W\rceil\{t_R - (W-1)t_{CMD}\} & \text{in read mode} \end{cases} \quad (5)$$

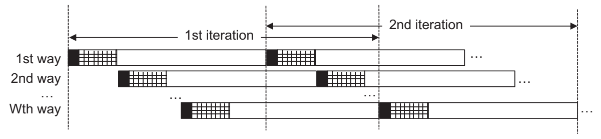

**Fig. 3** *Timing diagram for write operation with multi-way interleaving*

■ command phase ($t_{CMD}$)
▦ data transfer phase ($t_{TRANS}$)
☐ busy phase ($t_{PROG}$)

*Results:* We compared our method with the method proposed in [4], where the authors modelled an NFSD in cycle-accurate level and simulated it using a cycle-accurate simulator [5]. We tested the method for 10 different architectures, while varying the number of channels and ways. We adopted the system parameters from the commercial product [6] and implemented a profiler for the log-based FTL in [1]. Two synthetic traces (one for read and one for write) were created for this experiment. Each trace included 50 000 requests and the size and address of each request was generated with a uniform distribution. Fig. 4 shows good agreement of the two methods for both read and write modes. The average accuracy of our method was 98.7% over the cycle-accurate simulation and the worst-case accuracy was bounded by 92.7%. On the other hand, we achieved the analysis speedup by three orders of magnitude.

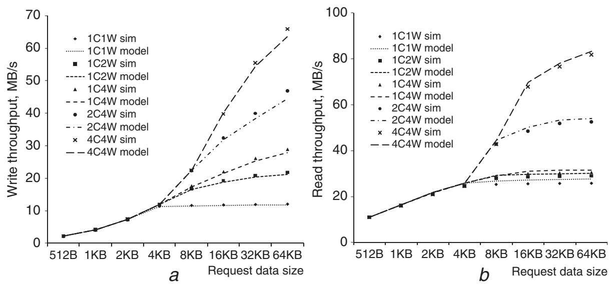

**Fig. 4** *Performance against request data size for various architectures*

*a* Write mode
*b* Read mode
nCmW sim: cycle-accurate simulation for n-channel/m-way NFSD
nCmW model: proposed method for n-channel/m-way NFSD

*Conclusion:* We propose a fast performance analysis method for NFSDs. The method is based on the profiling and an analytical model. The proposed method was validated against the cycle-accurate simulation for various architectures. It was proved that our method drastically improved analysis time while maintaining accuracy.

*Acknowledgments:* This work was supported in part by the IT R&D programme of MKE/IITA 2009-S005-01 (Development of Configurable Devices & S/W environment), by the Korea Research Foundation Grant funded by the Korean Government (MEST) (KRF-2007-313-D00578) and by Hynix Semiconductor Inc.

© The Institution of Engineering and Technology 2009
*28 July 2009*
doi: 10.1049/el.2009.2166

S.K. Won, S.H. Ha and E.Y. Chung (*School of Electrical and Electronic Engineering, Yonsei University, 134 Sinchon-dong, Seodaemun-gu, Seoul 120-749, Korea*)

E-mail: eychung@yonsei.ac.kr

## References

1 Kim, J., Kim, J.M., Noh, S., Min, S.L., and Cho, Y.: 'A space-efficient flash translation layer for compact flash systems', *IEEE Trans. Consum. Electron.*, 2002, **48**, (2), pp. 366–375
2 Kang, J.-U., Kim, J.-S., Park, C., Park, H., and Lee, J.: 'A multi-channel architecture for high-performance NAND flash-based storage system', *J. Syst. Archit.*, 2007, **53**, (9), pp. 644–658
3 Kang, J.-U., Jo, H., Kim, J.-S., and Lee, J.: 'A superblock-based flash translation layer for flash memory'. EMSOFT'06, Seoul, Korea, October 2006, pp. 161–170
4 Kim, S., Park, C., and Ha, S.: 'Architecture exploration of NAND flash-based multimedia card'. DATE'08, Munich, Germany, March 2008, pp. 218–223
5 Carbon SoC Designer, Carbon Design Systems Inc.: http://www.carbonsystems.com/downloads/datasheets/CMS-SoCD.pdf
6 Hynix Semiconductor: 'HY27UH08AG5B 2G x 8bit NAND flash memory data sheet Rev.0.2', January 2008