PATCH: Process-Variation-Resilient Space Allocation for Open-Channel SSD with 3D Flash

Jing Chen¹, Yi Wang², Amelie Chi Zhou³, Rui Mao¹, Tao Li²
1. College of Computer Science and Software Engineering, Shenzhen University, Shenzhen, China
2. Department of Electrical and Computer Engineering, University of Florida, Gainesville, FL, USA.
{yiwang, chi.zhou, mao}@szu.edu.cn, taoli@ece.ufl.edu

Abstract—Advanced three-dimensional (3D) flash memory adopts charge-trap technology that can effectively improve the bit density and reduce the coupling effect. Despite these advantages, 3D charge-trap flash brings a number of new challenges. First, current etching process is unable to manufacture perfect channels with identical feature size. Second, the cell current in 3D charge-trap flash is only 20% compared to planar flash memory, making it difficult to give a reliable sensing margin. These issues are affected by process variation, and they pose threats to the integrity of data stored in 3D charge-trap flash.

This paper presents PATCH, a process-variation-resilient space allocation scheme for open-channel SSD with 3D charge-trap flash memory. PATCH is a novel hardware and file system interface that can transparently allocate physical space in the presence of process variation. PATCH utilizes the rich functionalities provided by the system infrastructure of open-channel SSD to reduce the uncorrectable bit errors. We demonstrate the viability of the proposed technique using a set of extensive experiments. Experimental results show that PATCH can effectively enhance the reliability with negligible extra erase operations in comparison with representative schemes.

Index Terms—Three-dimensional flash memory, process variation, open-channel SSD, charge-trap flash, space allocation

I. INTRODUCTION

Advanced 3D flash memory adopts novel charge-trap technology that stacks storage cells vertically through a cylindrical channel. This technology enables the higher integration capacity and prevents data corruption caused by coupling effect. Despite these advantages, several recent studies found that charge-trap flash is vulnerable to process variation. Both charge loss and retention properties are dependent on the hole etching and gate patterning processes [1]. This poses challenges to the data integrity of 3D flash memory unless counteracted by architectural and design innovations.

Figure 1(a) illustrates a typical flash memory array with 3D charge-trap technology. In this architecture, the NAND strings are drilled in the gate layer to form the 3D infrastructure. With a large number of gate layers, current etching technology is unable to manufacture identical NAND strings. The fluctuation of channel’s size leads to the variations in access latency and retention properties. Another critical problem for 3D charge-trap flash is the small cell current. As shown in Figure 1(b), conventional planar floating-gate (FG) flash normally requires 200 nA/cell saturation current to guarantee a reliable sensing margin. However, with a stack of over 24 layers, the cell current is less than 20% of the floating-gate cell [2].

In order to provide a reliable and stable storage device with 3D charge-trap flash, existing approaches focus primarily on the system infrastructure and circuit-level hardware optimization [3]–[6]. Although the hardware-based approaches offer one part of the solution, we believe that system software and device driver can provide scalable yet efficient design options. Open-channel SSD (Solid State Drive) is an emerging system architecture for flash memory, and it can provide rich functionalities to access flash storage media [7]–[9]. It is possible to utilize the unique features of open-channel SSD to cope with the process variation issue in 3D charge-trap flash.

This paper presents PATCH, a process-variation-resilient space allocation scheme for open-channel SSD with 3D charge-trap flash memory. PATCH adopts two reliability enhancement strategies, fault-cube creation and live migration, to allocate write requests to physical blocks considering process variation. It aims to prevent the usage of unreliable physical blocks with very low overhead. We implemented PATCH in the Linux kernel with the system support provided by open-channel SSD. PATCH is compared with a representative 3D flash memory data allocation scheme [10] and the default scheme in open-channel SSD [11]. Experimental results show that, PATCH can significantly reduce the uncorrectable errors with the negligible RAM space and timing overhead.

The main contributions of this paper are:

- The proposed strategy utilizes system software in the host
system provided by open-channel SSD to enhance the data integrity of 3D charge-trapping flash.

- A space allocation strategy is proposed to enable the transparent mapping in the presence of process variation.
- As a proof of concept, we compare the proposed technique with representative schemes using a set of real workloads and standard benchmarks.

The rest of this paper is organized as follows. Section II overviews background and discusses the motivation of this paper. Section III presents our proposed scheme PATCH in detail. Section IV presents experimental results. Section V concludes this paper and discusses future work.

II. BACKGROUND AND MOTIVATION

A. 3D Flash Memory

The 3D NAND flash memory becomes an innovative process architecture to continue the current trend of increasing bit density. There are mainly two types of 3D NAND flash memory technologies [12], charge-trap flash and floating-gate flash. Charge-trap flash is considered as one of the most promising technologies for 3D integration due to the better scalability than floating-gate flash [1].

The 3D charge-trap flash causes reliability issues due to its ring shape stack of the memory cell. The most critical reliability issue is charge loss [13], [14]. Charge loss represents leakage paths for charge, from each cell’s active area towards other cells on the same NAND string. Figure 2 illustrates the charge loss due to vertical charge loss and lateral charge migration. Among them, vertical charge loss represents the cases that the charges go through blocking oxide or tunnel oxide. Lateral charge migration towards the spacer represents the case that the charges go through the region between different control gates.

B. Open-Channel SSD

Different from the traditional SSD that provides the fixed functionality and unpredictable I/O latencies, open-channel SSD exposes the internal parallelism of the SSD to the host. In this way, some key functionalities of conventional flash translation layer (FTL) can be managed by the host system with LightNVM [11]. By migrating address translation modules from embedded processor in SSD device to multicore CPU on the host, it can effectively handle I/O scheduling and boost the access performance.

In open-channel SSD, the host system with LightNVM issues requests with physical pages as the input. This can fully utilize the parallelism provided by SSD, and it can maximize overall read and write bandwidth with physical page addressing. This can also simplify the design of SSD controller hardware and provide energy efficient storage device. Applications can directly access host software in the kernel to enable a flexible and application-specific storage device. We capture the property of open-channel SSD and manipulate the address translation of physical space, resulting in an efficient storage interface to address the process variation issue of 3D flash memory.

C. Motivation

The reliability issues of 3D charge-trap flash are normally process-based and they reflect the increasing difficulty in patterning smaller and smaller feature sizes with reliable lithography and etching technologies. For 3D charge-trap flash, the process status of each individual NAND cell is jointly affected by fabrication of bit line and the NAND string it belongs to [3]. By programming or reading a physical page, the process status of this page can be known. This information can help predict the process status of its geographically adjacent pages.

To fully explore the unique aspects of 3D charge-trap flash and take advantage of system software to handle the process variation issue, several factors have to be considered. First, as the emerging open-channel SSD moves the module of address translation to the host system, it is possible to manipulate data allocation and enhance data integrity when deploying open-channel SSD. Second, the process status of charge-trap flash has the location-dependent property. The prediction of the process status has to consider this property, and it has to be efficient with low overhead. These observations motivate us to propose a novel process-variation-resilient space allocation scheme for open-channel SSD with 3D flash.

III. PATCH: PROCESS VARIATION RESILIENT RELIABILITY ENHANCEMENT SCHEME

A. Overview

Figure 3 illustrates the system architecture for PATCH with open-channel SSD. PATCH adopts two reliability enhancement strategies, fault-cube creation, and live migration. It integrates a set of new features into the 3D flash memory storage system. First, PATCH addresses the process variation issue and presents a system software based solution to transparently allocate data in the presence of process variation. Second, PATCH does not have to maintain the process variation status of each physical block, thereby reducing address translation overhead. Third, PATCH presents a general block allocation strategy without further modifications in the circuit-level or chip-level design.

B. Modeling of PV Status

The process variation causes nonideal scaling of threshold voltage. To model the process variation of NAND strings,
Fig. 3. The system architecture of PATCH with open-channel SSD.

gate sensing and channel sensing can be used to derive the charge position and charge density [15]. The injected charge density $Q_h$ and mean vertical location of the hole $\hat{x}_h$ can be obtained from Equations 1 and 2, where $T_{T_{ox}}$ and $T_{B_{ox}}$ are thickness of top oxide and bottom oxide, respectively; $T_N$ is the thickness of the nitride; $EOT$ denotes oxide thickness of oxide-nitride-oxide (ONO); $\varepsilon_{N}$ and $\varepsilon_{ox}$ are oxide and nitride dielectric constants, respectively; $\Delta V_{FB, ch}$ and $\Delta V_{FB, pl}$ are flat-band voltage shifts with respect to the initial uncharged flat-band voltage; and $\hat{x}_h$ is the mean vertical location.

$$Q_h = \varepsilon_0 \varepsilon_{ox} \cdot \frac{\Delta V_{FB, ch} + \Delta V_{FB, pl}}{EOT}$$

$$\hat{x}_h = \frac{\Delta V_{FB, pl} (T_{T_{ox}} \varepsilon_{N} + T_{N} \varepsilon_{ox}) - \Delta V_{FB, ch} T_{B_{ox}} \varepsilon_{N}}{\varepsilon_{ox} (\Delta V_{FB, ch} + \Delta V_{FB, pl})}$$

For 3D charge-trap flash, $Q_h$ and $\hat{x}_h$ can form multiple curves under different process variations. Based on the empirical data, the distribution of charges can be obtained. As long as the process status and location (layer or control gate) of a NAND cell are known, it can obtain the process status of each NAND cell within the same NAND string.

C. PV Classification with Fault-Cube Creation

PATCH aims to handle the data allocation of critical data, especially file system metadata, page mapping tables, and frequently updated data. PATCH adopts the bloom-filter based strategies to identify critical data. Note that the identification of critical data is not the major contribution of this work. Our scheme is transparent to different identification strategies and can be combined with these strategies to improve the efficiency of identification. PATCH classifies status of process variation and tries to avoid the use of unreliable physical blocks. This is handled by the reliability enhancement strategy fault-cube creation. The fact is that, the main source of the faults is generated by unreliable physical blocks.

Fig. 4. (a) The creation of a fault-cube with an unreliable block in the center. (b) The fault-cube uses three physical blocks (a center block and two corner blocks) to record the location.

Fault-cube creation aims to generate several virtual cubes, called fault-cubes. The reliability grade is based on the granularity of a physical block. Once a physical block is evaluated as an unreliable physical block, our strategy will create a fault-cube with its surrounding blocks in all three dimensions. This block will become the center block of the fault-cube. Each fault-cube needs three physical blocks to record its location. Figure 4 shows an example.

For the basic steps to generate or shrink a fault-cube, PATCH first checks whether the physical block $B_i$ belongs to an existing fault-cube. If the physical block $B_i$ is not a faulty block, this case shows that the range of the fault-cube $C_x$ is too big. It has to remove blocks between the current boundary and block $B_i$ in the direction from the center block $B_0$ to $B_i$. For the case that the physical block $B_i$ is a faulty block and it is a boundary block in $C_x$, this case shows that the current range of fault-cube $C_x$ could not cover all faulty blocks. The fault-cube $C_x$ should increase its range to incorporate more potentially faulty blocks. The creation of the fault-cube is triggered when the physical block $B_i$ is a faulty block and it does not belong to any of existing fault-cubes. PATCH will create a new fault-cube $C_y$ and set block $B_i$ as the center block of fault-cube $C_y$. PATCH will finally update the grade for block $B_i$, as this block can obtain its actual process status after page programming.

D. Reducing Faults with Live Migration

This section presents the proposed live migration strategy. This strategy aims to reduce the faults if a critical page

Fig. 5. (a) Shrinking an existing fault-cube, if the boundary block is a reliable block. (b) Creating a new fault-cube, if the boundary block is an unreliable block.
Algorithm III.1 Live migration operation.

Require: The reliability grade $G_n \in \{G_1, ..., G_m\}$ of a block;
current $N_f$ fault-cubes; and issuing a critical page write request
to physical block $B_i$, $i \in [0, N_{b,k}]$.
Ensure: Perform live migration operations.
1: if physical block $B_i$ is a faulty block then
2: if $B_i$ does not belong to a fault-cube $C_k$ then
3: Create a new fault-cube $C_k$ and set $B_i$ as the center of the
fault-cube $C_k$.
4: end if
5: Find another block $B_j$.
6: if physical block $B_j$ is a faulty block then
7: Trigger garbage collection for reliable blocks and find
block $B_k$.
8: Invalidate the first page of $B_j$.
9: Issue writes to $B_k$.
10: else
11: Issue writes to $B_j$.
12: end if
13: Invalidate the first page of $B_j$.
14: end if
15: Update the block grade.
16: Update the address mappings.

is issued to an unreliable block. PATCH manipulates the
space allocation and performs live migration to reallocate the
incoming write requests to other physical blocks. For critical
data, the actual reliable block will be served first, and the
unknown status block will be selected, which may trigger the
fault-cube creation operation. If current actual physical
block is with low grade, PATCH will select the potentially
poor quality block in the existing fault-cubes. Therefore, the
unreliable blocks will have very low chance to be selected to
serve critical data.

For normal data, the selection procedure first chooses the
unknown status block, which will help figure out the process
status of other blocks and refine the range of each fault-cube.
The potentially poor quality block will be the second choice,
and these blocks are located inside fault-cubes. If the statuses
of all physical blocks are known, the normal data will choose
the reliable block to cater the write request. The last choice
would be the unreliable block, which will eventually lead to
the high error rate.

Algorithm III.1 presents the basic steps for the live migration
strategy. It handles the case that a critical page is issued to an
unknown status block. The live migration strategy will
invalidate the first page of the faulty block, leaving it for
further usage as log blocks or replacement blocks. If the block
is faulty, live migration strategy will trigger the fault-cube
creation. Therefore, it can work with the fault-cube creation
strategy to enhance the reliability.

E. Overhead analysis

PATCH maintains fault-cube to track the faulty blocks. The
timing overhead is the selection of free blocks. PATCH has to
first check whether the candidate free block belongs to fault-
cubes. In the worst-case, all blocks in the flash are faulty,
and it takes $O(\log n)$ time, where $n$ is the number of physical
blocks. Using hash table can further reduce the processing
time to $O(1)$. Therefore, this overhead is negligible compared to
flash operations.

PATCH has to maintain the information for fault-cubes, and
this will incur memory space overhead or storage space
overhead. Each fault-cube needs to store three numbers (i.e.,
the physical block number for two diagonal blocks and one
center block in the cube). For current 3D charge-trap flash,
each flash consists of about 2K physical blocks. It requires
11 bits $\times 3 = 33$ bits. By default, the fault-cube creates a virtual
cube with 125 $(5 \times 5 \times 5)$ physical blocks. Therefore, for each
chip, at most $2K/((5+1)\times 125) = 85$ fault-cubes are currently
maintained by PATCH. For an open-channel SSD with eight
flash chips, if this information is maintained in memory, it will
cost at most $33 \times 85 \times 8 = 21.91$ KBytes memory space
overhead. PATCH performs space re-allocation to avoid the
use of faulty blocks. The mapping table for the open-channel
SSD is $22 \times 2K \times 8 = 352$ KBytes. Therefore, the memory
space overhead is about 373.91 KBytes.

IV. Evaluation

A. Experimental Setup

The proposed PATCH was implemented in Linux kernel
4.14.4. The system integrates open-channel SSD in the Linux
kernel. Open-channel SSD provides LightNMV subsystem
architecture that consists of a software module called Physical
Block Device (pblk). The address translation of PATCH
was implemented in the pblk. In the experiments, we tested
sampling 3D NAND flash memory chips from SK Hynix
Corporation. The capacity of the sampling chip is 64 GBytes.
It is a 72-layer Triple-Level-Cell (TLC) NAND flash. The page
size is 8,192 bytes. Each block contains 1,024 physical pages.
Each chip contains 911 physical blocks. The typical time to
read a page is 90 $\mu$s, and the time to program a page is
1,100 $\mu$s. The erase time for a physical block is 10 ms. In
the evaluation, an 8-chip 512 GBytes SSD is configured.

We used standard benchmarks from the SNIA IOTTA
trace repository [16]. The basic characteristics of traces are
listed in Table I. In the experiments, the error correction
capability is 512-bit per physical page. The distribution and
the probability of errors are obtained from [17], [18]. A
representative process-variation-aware reliability management

<table>
<thead>
<tr>
<th>Trace</th>
<th># of write operations</th>
<th># of read operations</th>
<th>% of write</th>
<th>% of read</th>
</tr>
</thead>
<tbody>
<tr>
<td>BuildServer-1</td>
<td>942,177</td>
<td>8,022,275</td>
<td>10.51</td>
<td>89.49</td>
</tr>
<tr>
<td>BuildServer-2</td>
<td>2,289,583</td>
<td>4,639,405</td>
<td>33.04</td>
<td>66.96</td>
</tr>
<tr>
<td>BuildServer-3</td>
<td>4,687,821</td>
<td>2,901,366</td>
<td>61.77</td>
<td>38.23</td>
</tr>
<tr>
<td>mail-01</td>
<td>6,303,408</td>
<td>1,267,017</td>
<td>83.25</td>
<td>16.74</td>
</tr>
<tr>
<td>mail-02</td>
<td>7,440,684</td>
<td>129,741</td>
<td>98.27</td>
<td>1.73</td>
</tr>
<tr>
<td>mail-03</td>
<td>7,018,771</td>
<td>551,656</td>
<td>92.71</td>
<td>7.29</td>
</tr>
<tr>
<td>Nexus</td>
<td>8,651,664</td>
<td>4,667,748</td>
<td>64.96</td>
<td>35.04</td>
</tr>
<tr>
<td>TPCC Trace1</td>
<td>150,153</td>
<td>2,992,497</td>
<td>4.78</td>
<td>95.22</td>
</tr>
</tbody>
</table>
scheme called P-Alloc [10] is selected for comparison. The default LightNVM [11] is selected as the baseline scheme.

B. Results and Discussion

1) Uncorrectable Bit Errors: We first present the total number of uncorrectable bit errors, that are beyond the error correction capability of ECC. In Figure 6, the X-axis denotes the percentages of completion of the trace. From the experimental results, the proposed PATCH can effectively reduce the uncorrectable bit errors compared to the baseline scheme and P-Alloc. PATCH aims to capture the location-dependent property of 3D charge-trap flash. The error rate of PATCH is relatively stable. Although P-Alloc also tries to avoid using the physical block with bad performance, the selection of free block is based on prediction. It may not be effective, especially when the process statuses for most of the blocks are unknown.

2) Allocation of Critical Data: According to the program speed, we divide the physical blocks into five grades: \{G_1, G_2, G_3, G_4, G_5\}. Among them, G_1 represents the most reliable physical blocks, while G_5 represents the blocks with the lowest level of reliability. The grades are five predefined and evenly distributed ranges based on the programming speed. Figure 7 presents the allocation of critical data. As expected, the proposed PATCH can effectively utilize the reliable physical blocks. Most of the critical data are allocated to the physical blocks with grades G_1 and G_2. The limitation of P-Alloc is the scale of prediction. It cannot globally locate the group of unreliable physical blocks. Therefore, our proposed PATCH can achieve better results. The baseline scheme does not enforce priority on physical blocks, so the block usage actually illustrates the process variation information for the 3D charge-trap flash.

3) System Response Time: P-Alloc needs to update the predicted status of a physical block. Since P-Alloc has to maintain the process status of each physical block, the update of this information incurs both timing and storage overhead. Our proposed PATCH postpones the use of unreliable physical blocks. This leads to the reduction of the number of available physical blocks. The write pressure will be served by less number of physical blocks, resulting in more garbage collection operations and valid page copy operations. Therefore, PATCH could increase the average response time. PATCH also has to record the faulty cube in order to prevent using unreliable physical blocks. Based on the analysis, this overhead is negligible. From the experimental results in Figure 8, PATCH incurs only 0.16% extra timing overhead. PATCH can also reduce system response time by 12.98% compared to P-Alloc.

4) Block Erase Counts: This section presents the experimental results for the worst block erase count. As shown in Figure 9, P-Alloc gets the largest number of the worst block erase count. The reliable blocks, especially the block after actual programming, will be prioritized to handle the request
for critical data. Therefore, these reliable blocks will have more erase operations compared to unreliable ones. PATCH looks for the reliable blocks globally, and the search area is much larger than P-Alloc. Then PATCH can find more reliable blocks in the initial stage of the search. This results in the reduction of the worst erase count compared to P-Alloc.

V. CONCLUSION

This paper presents PATCH, the first system-level data allocation strategy to optimize the process variation issue for open-channel SSD with 3D charge-trap flash. Two optimization strategies, i.e., fault-cube creation and live migration, have been presented to prevent using unreliable physical blocks. Experimental results show that our PATCH can significantly reduce the number of uncorrectable bit errors, improve the allocation of critical data, while suffering only negligible extra block erase counts. In the future, we plan to investigate the simplified flash memory access model of open-channel SSD, as it can potentially eliminate redundant data accesses to further improve the performance.

ACKNOWLEDGMENT

This work was supported in part by the National Natural Science Foundation of China (61502309 and 61802260), in part by the Guangdong Natural Science Foundation (2016A030313045, 2018A030310440, and 2017B030314073), in part by the Shenzhen Science and Technology Foundation (JCYJ20170817100300603, JCYS2017081609341397, and JCYJ20170302153955969), in part by the Guangdong Provincial Project (2014GKM054), in part by the Natural Science Foundation of SZU (803/000027060147), and in part by the State Key Laboratory of Computer Architecture, Institute of Computing Technology, Chinese Academy of Science (CARCH201608). Yi Wang and Rui Mao are corresponding authors.

REFERENCES