Combating Bit Errors From Stuck Cells in Flash Memory Using Novel Information Theory Techniques

Ravi H. Motwani[†], **Zion S. Kwok**[‡], **and Poovaiah M. Palangappa**[†]
Non-Volatile Memory Solutions Group
Intel Corporation
[†]Santa Clara, California, USA, and [‡]Vancouver, Canada
Email: *ravi.h.motwani@intel.com, zion.s.kwok@intel.com, poovaiah.m.palangappa@intel.com*

***Abstract*—Low-density parity-check (LDPC) codes have been successfully deployed in NAND Flash memory based Solid State Drives (SSDs). As Flash memory scales, and has now advanced from planar architectures to three-dimensional ones, defects in the form of stuck cells have increased. Stuck cells are more difficult to correct using LDPC codes because they typically masquerade as reliable bits, but their persistence also offers opportunities to identify and fix them.**

First, we propose the use of LDPC errors and erasures decoding to erase the bits read from known stuck cells. Second, we use sectionalized Flip-N-Write (FNW) to preprocess the codeword during writes to NAND to minimize bit errors due to stuck cells together with an LDPC inner code. Both proposals have been validated using simulation of a one kilobyte information block encoded in an LDPC code of rate 0.9. LDPC errors and erasures decoding and sectionalized FNW with LDPC result in 1.92× and 1.94× raw bit error rate (RBER) gains, respectively, for soft-decision decoding (SDD).

I. Introduction

Three-dimensional (3D) NAND Flash memory is replacing planar NAND in Solid State Drives (SSDs) because of 3D NAND's higher data storage density. But in comparison to planar NAND, 3D NAND has more stuck-cell defects due to short and open circuits in word lines and bit lines. These defects contribute to the raw bit error rate (RBER) in 3D NAND.

Prior work on handling stuck cells consists of the work of Heegard [1] and work in [2], [3], [4], which mostly use code construction for resilience against stuck cells. However, these schemes result in codes with low rates or a practical stuck-cell probability. Recent work on using error correction pointers (ECP) [5] and a more refined implementation of ECP using pay-as-you-go (PAYG) [6] are very interesting but they need extra memory to store the data and locations of the stuck cells.

In this paper, we propose two methods to counter the impact of stuck cells and improve the performance of low-density parity-check (LDPC) codes. First, we propose to flag the stuck cells as erasures and perform errors and erasures decoding of the LDPC codes. Second, we apply sectionalized Flip-N-Write (FNW) with an LDPC inner code. A patent application [8] has also been filed for this work. These methods assume that the locations of stuck cells are available. [7] describes how these stuck cell locations may be identified.

In the next section, we formulate the problems faced by LDPC codes due to stuck cells. Sections III and IV provide more details about LDPC errors and erasures decoding, and sectionalized FNW with LDPC, respectively. Section V shows the results and we draw conclusions in Section VI.

II. Impact of High Confidence Errors from Stuck Cell Reads on LDPC Coding Gains

In Figure 1, we show a single-level cell (SLC) Flash memory threshold voltage (V_t) distributions for the erase (L0) and the program (L1) states. For a 5-strobe read, a read will give either a low, medium, or a high confidence logical bit '0' or '1'. A major source of bit errors is the overlap between two adjacent V_t distributions, which are mostly read as low confidence bits. However, if the cell is open circuited, the cell would appear to have a high V_t, which is outside the V_t overlap region, so the cell would read as a high confidence '0'. Since bits read from stuck cells result in high confidence bit errors with probability 0.5, a large number of stuck cells can derail the LDPC decoding, which make use of confidence information.

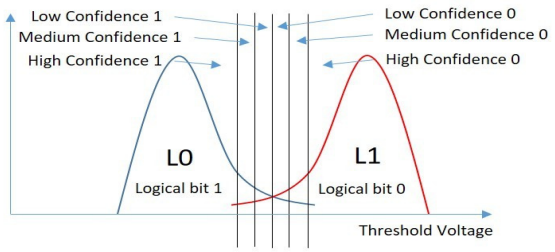

Fig. 1: Single-Level Cell (SLC) Flash Memory threshold voltage (V_t) distributions for the erase and program states.

III. LDPC Errors and Erasures Decoding Where Bits Read from Stuck Cells Are Erasures

LDPC errors and erasures decoding can be performed by setting the log-likelihood ratios (LLRs) to zero for the bits corresponding to stuck cells. We need to obtain the locations of the stuck cell before we can use this method to decode a codeword that is read from NAND.

IV. Sectionalized Flip-N-Write to Shape the Data in Accordance with Bits Read from Stuck Cells

Alternatively, we could eliminate some of the bit errors due to stuck cells by preprocessing the data using sectionalized FNW, and then constructing an LDPC codeword to protect

against any additional bit errors. The stuck cell locations are needed when writing to the NAND, but not when reading from the NAND.

A. Flip-N-Write (FNW)

The Flip-N-Write (FNW) [9] data-shaping procedure is the building block of sectionalized FNW. FNW writes data in either its original form or its flipped form, in which all of the data bits are complemented. A one-bit flag indicates whether or not the written data is flipped.

For a given stuck cell, we can choose to write a logical bit '0' or '1' to it. By keeping the data the same or flipping all of the data bits, we can force the bit we write to match the defective value of the stuck cell. The subsequent read will see the expected bit value at the stuck cell and a bit error is prevented. After the codeword is corrected by the LDPC decoder, the data bits may need to be flipped back based on the value of the one-bit flag.

When there are more than one stuck cells, the maximum number of errors in data written to the NAND can be reduced to at most $\lfloor \frac{s}{2} \rfloor$, where s is the number of stuck cells in the FNW-encoded data.

B. Sectionalized FNW

In sectionalized FNW, the data is divided into multiple sections, where each section of data can be separately flipped or not flipped to minimize the mismatches between the written data and the stuck cells' defective values. Each section costs one additional flag bit, but increasing the number of sections results in a lower average RBER due to stuck cells.

We experimented with an LDPC codeword of data size one kilobyte and a code rate of 0.9. We divided the data bits into 10 sections, hence the length of each section is around $n = (\lceil \frac{1024\times 8}{10} \rceil) = 820$, and we performed FNW for each section. We punctured 10 LDPC parity bits to make up for the 10 additional FNW flag bits. Given a probability q=2e-3 that a bit corresponds to a stuck cell, and a stuck cell is an error half of the time, we would normally expect the stuck cells to contribute $\frac{q}{2}$=1e-3 to the RBER. In contrast, the RBER of the data after sectionalized FNW is 4.33e-4: a reduction by 2.3× or a difference of 6.67e-4.

The post sectionalized FNW data and FNW flag bits are encoded in an LDPC code. FNW is not applied to the LDPC parity bits, so they are vulnerable to stuck cells. When we include the LDPC parity bits, the RBER from stuck cells post sectionalized FNW increases to 4.89e-4.

V. Results

Figure 2 shows the codeword failure probability vs. RBER plots for LDPC hard-decision decoding (HDD) and soft-decision decoding (SDD) where q=2e-3 of the bits being stuck cells. LDPC errors and erasures decoding and sectionalized FNW with LDPC result in 1.18× higher RBERs at the same codeword failure probabilities for HDD, and 1.92× and 1.94× higher RBERs, respectively, for SDD at codeword failure probability of 1E-7. Comparing with LDPC SDD without stuck cells, these methods completely eliminate the degradation in RBER caused by stuck cells, while Sectionalized FNW with LDPC performs even better with stuck cells.

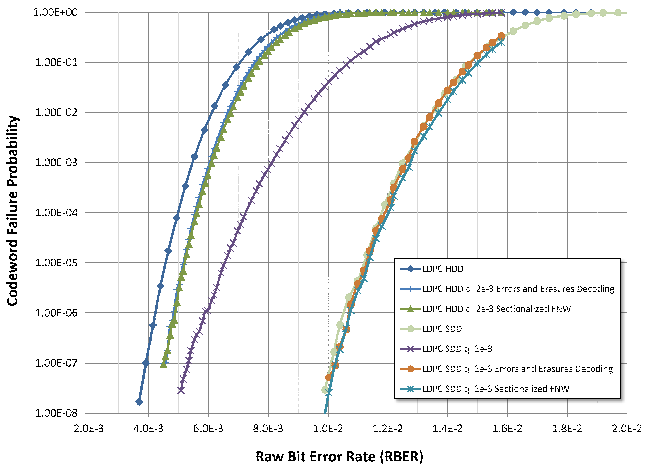

Fig. 2: The codeword failure probability simulated over a range of RBERs with stuck cells.

VI. Conclusions

In this paper, we proposed methods to improve the coding gain of the LDPC code, which is impaired by stuck cell defects and validated them with simulation results. First, we proposed LDPC errors and erasures decoding by treating the bits read from stuck cell as erasures. Assuming no information is better than wrong high confidence information when it comes to stuck cells. Second, we proposed a divide and conquer strategy where sectionalized FNW reduces the RBER due to stuck cells and LDPC corrects the residual bit errors. Rather than using LDPC to correct both bit errors due to stuck cells and bit errors due to V_t overlaps, it is better to use two different techniques to combat the two types of errors.

References

[1] C. Heegard, " Partitioned Linear Block Codes for Computer Memory with Stuck at defect," *IEEE Trans. Information Theory, Vol. IT - 29, no. 6, Nov. 1983,* , pp. 831–842.

[2] A. Wachter-Zeh and E. Yaakobi, " Codes for Partially Stuck-at Memory Cells," *Int. ITG Conf. on Systems, Communications and Coding (SCC)*, Feb. 2015.

[3] J. Borden and A. Vinck, " On coding for stuck-at defects," *IEEE Tran. Information Theory*, vol. 33, no. 5, pp. 729–735, Sep. 1987.

[4] R. Gabrys, R. Sala, and L. Dolecek, Coding for unreliable flash memory cells," *IEEE Comm. Letters, Vol. 18, no. 9, July 2014*, pp. 1491–1494.

[5] Schechter Stuart, Loh Gabriel H. , Strauss Karin , Burger Doug, " Use ECP, Not ECC, for Hard Failures in Resistive Memories," *Proceedings of the 37th Annual International Symposium on Computer Architecture*, pp. 141–152, 2010

[6] M.K. Qureshi, " Pay-As-You-Go: low-overhead hard-error correction for phase change memories," *Proceedings of the 44th Annual IEEE/ACM International Symposium on Microarchitecture*, pp. 318–328, 2011

[7] R. Motwani and P. Kalavade, "Apparatus and Method for detecting and mitigating bit-line opens in Flash Memory," *Patent application no. 20160283320*, filed March 27, 2015.

[8] R. Motwani, Z. Kwok, and P. Palangappa, "Memory Circuit Defect Correction," *US Patent application*, filed Dec. 23, 2015.

[9] S. Cho and H. Lee, "Flip-N-Write: A simple deterministic technique to improve PRAM write performance, energy and endurance," *Proceedings of 42nd Annual IEEE/ACM International Symposium on Microarchitecture*, pp. 347–357, 2009