Coset Coding to Extend the Lifetime of Non-Volatile Memory

Adam N. Jacobvitz, Robert Calderbank, and Daniel J. Sorin
Department of Electrical and Computer Engineering
Duke University

Abstract*—*In this paper, we show how to apply coset coding to extend the lifetime of non-volatile memories, including phase change memory and Flash.

1. Introduction

Non-volatile memory (NVM), such as phase change memory (PCM) and Flash memory, is attractive for a variety of purposes and it has become widely used in a variety of platforms. One challenge with using several types of NVM is that the NVM cells can be degraded by being written or erased. Eventually, the accumulated wear on an NVM cell can render that cell unusable.

In this project, we explore how error-correcting codes can be used to record data in a way that reduces wear on NVM cells, thereby extending the life of the memory. Specifically, we use endurance coding based on *coset coding* [1][2] to minimize the number of writes and/or erases, depending on the characteristics of the particular NVM.

2. Coset Coding

The key enabling technology that we use is coset coding. Coset coding differs from traditionally used codes in how it maps from a dataword to a codeword. With standard error correction, such as a Hamming error correcting code (ECC), there exists a one-to-one mapping from a k-bit dataword to an n-bit codeword, where $n \geq k$. The n-k extra bits in the codeword provide features such as error correction.

With coset coding, there is a one-to-one mapping from a k-bit dataword to a coset of n-bit codewords. The set of all possible 2^n n-bit strings is partitioned into equal-size, non-overlapping cosets. In a coset coding, every n-bit string in a particular coset can be used to represent the single dataword that maps to that coset.

The appeal of coset coding is that it provides multiple options for mapping a dataword to a codeword. That flexibility enables us to choose the codeword so as to optimize an objective. We show later in this paper how utilize this flexibility for both PCM and Flash.

There are many possible coset codes that can be used to partition a space into cosets. These codes offer differing trade-offs between cost (the n-k bits of storage overhead required to store alternative coset representatives) and benefit (how many options are in each coset). Coset codes are also compatible with error correcting codes; we can embed a coset code inside of an error correcting code such that we get the benefits of both coset coding and error correction.

Many coset codes lend themselves to efficient encoding (from dataword to coset) and decoding (from codeword in a coset to dataword). The potentially challenging aspect of coset coding is dynamically searching a coset for the element in that coset that optimizes the desired objective, but we have shown that there are efficient schemes for this search.

3. Application to Phase Change Memory

Our first use of coset coding was to extend the lifetime of PCM [4]. PCM cells suffer wear when their values change, and thus our goal is to minimize bit flips. Because we cannot change the number of writes to the PCM, we are constrained to minimize the number of bits that flip per write.

Our approach, which we call FlipMin, chooses the codeword to write based on its Hamming distance from the existing codeword in the memory location. By minimizing the Hamming distance between the previous memory contents and what we then write to that location, we minimize the number of bit flips. One cost of FlipMin is that each write now requires a preceding read to discover the previous codeword in the memory location; this cost is minimal, however, because reads are so much faster than writes.

We have experimentally evaluated FlipMin, and our results show that FlipMin can greatly extend the lifetime of PCM. We compare FlipMin to ECC and other previous approaches, and we equalize the comparisons by starting each simulation at time zero with the same number of PCM cells. At time zero, FlipMin has fewer usable memory locations than prior work, because of its coding overhead. For example, with a rate ½ coset code, FlipMin's coding overhead is 100% and thus it has only half the usable memory locations of a memory with no protection. However, despite its time-zero disadvantage, FlipMin provides far greater lifetimes. In Figure 1, we provide an illustration that is representative of the actual results but omits the details (e.g., which ECC we use and which coset code we use) and the exact numbers.

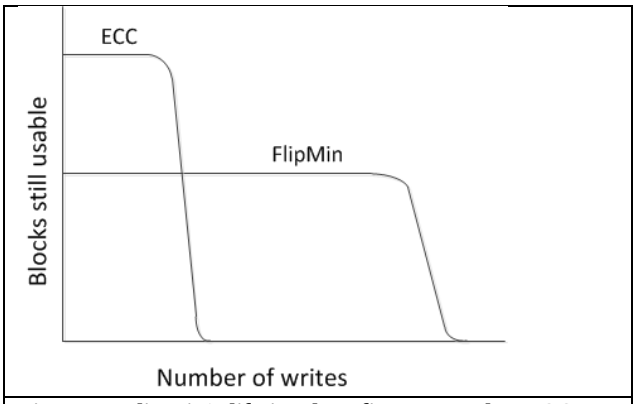

Figure 1. FlipMin's lifetime benefit compared to ECC

4. Application to Flash

Our second use of coset coding was to extend the lifetime of Flash [3]. Flash is similar to PCM in that it wears out, but its wearout phenomenon is different. Flash does not degrade as a result of writes but rather as a result of erases. Thus our goal is to use coset coding to minimize the number of erases per write.

Flash is organized into blocks, each of which has many pages. A page is the granularity at which Flash can be read/written, and a block is the granularity at which it can be erased. A block is erased to make room for new page writes.

Typically, each Flash page is written once before needing to be erased, when its block is erased, so that the page can be re-written. The underlying reason that each page can be written only once is that Flash only permits charge to be added to a cell, but not removed (until erase). Thus, re-writing an already written page would require that the write would only increase the amount of charge required in each cell. With pages containing on the order of 1024 cells, the likelihood of every cell being compatible with a re-write (i.e., the write only requires charge to be added to every cell) is virtually zero.

With coset coding, we can write each page multiple times before needing to erase it. The key to re-writing is that coset coding provides us with options for what to write. We can choose to write an *n*-bit string that is compatible with the previous contents of the page and that will maximize the number of additional writes that are possible. We developed a set of heuristics for choosing codewords. The heuristics prioritize compatibility first and then, if we have multi-level cells (MLCs), choose the codeword that minimizes how many MLCs are saturated by writing that codeword.

We have simulated a Flash SSD with coset coding, and the experimental results show that coset coding can greatly increase the lifetime of the SSD. The lifetime benefits are particularly pronounced for Flash cells with at least four levels. We have also implemented coset coding in a Flash SSD hardware prototype [5], and we are currently using this prototype for further experimentation.

5. Conclusions and Future Directions

Our work thus far demonstrates that we can use coset coding to extend the lifetime of PCM and Flash. The work is promising, yet we believe coset coding's potential is actually greater than what we have demonstrated thus far. First, we have used coset codes that are sub-optimal, in particular the short block codes used in the PCM work. We have since shown that, in theory, there is benefit to being able to optimize over longer blocks. Second, our heuristics for choosing codewords within cosets are ad hoc; they work well, but we believe there are opportunities to improve them. Third, our composition of coset coding with error coding has assumed equal likelihood of any error, both in terms of error location (i.e., errors are not spatially correlated) and type of error (i.e., a 0-to1 bit flip is as likely as a 1-to-0 bit flip). Other researchers have shown that real NVM exhibits error types and error locations that are not equally likely. By leveraging *a priori* knowledge of error likelihoods, we believe we can design codes that are more effective.

6. References

[1] G. D. Forney, "Coset Codes. I. Introduction and Geometrical Classification," *Information Theory, IEEE Transactions on*, vol. 34, no. 5, pp. 1123–1151, Sep. 1988.

[2] G. D. Forney, "Coset Codes. II. Binary Lattices and Related Codes," *Information Theory, IEEE Transactions on*, vol. 34, no. 5, pp. 1152–1187, Sep. 1988.

[3] A. N. Jacobvitz, A. R. Calderbank, and D. J. Sorin, "Writing Cosets of a Convolutional Code to Increase the Lifetime of Flash Memory," in *Proceedings of the 50th Annual Allerton Conference on Communication, Control, and Computing*, 2012.

[4] A. N. Jacobvitz, A. R. Calderbank, and D. J. Sorin, "Coset Coding to Improve the Lifetime of Memory," in *Proceedings of the 19th International Symposium on High Performance Computer Architecture*, 2013.

[5] "Jasmine OpenSSD Platform." [Online]. Available: http://www.openssd-project.org/wiki/Jasmine_OpenSSD_Platform.