Lattices for Error Correction and Rewriting in Flash Memories

Brian M. Kurkoski

***Abstract*—This paper gives an overview of the author's recent results on using lattices for error-correction and rewriting in flash memories. A construction using the E8 lattice and Reed-Solomon codes for error-correction in flash memory has a performance advantage of 1.7 to 2.0 dB. A rewriting code construction for flash memories based upon lattices has an minimum number of writes linear in one code parameter.**

I. Background

Flash memory is being used in an increasing larger variety of applications, expanding from digital cameras and MP3 players, to solid-state drives for not only laptop computers, but also in high-performance drives for "cloud computing" data centers. As such, improving the reliability, data density, and lifetime of the memory have become significant goals. Coding theory has two roles to play in achieving these goals. The first is familiar — to correct errors induced by the reading and writing process. The second is more novel — allowing rewriting while avoiding the erase operation that shortens the life of the flash memory.

For error-correction, numerous approaches have been considered, although BCH codes are predominant in practice [3]. In commercial flash memory products, the memory and error-correction functions are separated. The flash chip makes hard decisions internally; these hard decisions are passed to an external chip for implementation of error-correction. Single-level flash stores just two levels (or one bit), but multi-level flash stores q levels (or $\log_2 q$ bits) [4] [5].

Rewriting codes are a coding-theoretic approach to allow rewriting to memories flash memories, where values stored in memory may only be increased. While codes for binary media were proposed in the 1980s [6], [7], within the past few years, a large number of rewriting codes directed at flash memory have been described [8], [9], [10], [11], [12]. As with error-correcting codes, most of these these floating codes or flash codes are designed for flash memory cells that can store one of q discrete levels.

However, charge is stored in a physical flash cell during write operations. Charge, read as a voltage, is an inherently continuous quantity. Commercial flash memory makes hard decisions, and any coding, for error-correction and rewriting, must operate on these discrete values. It is reasonable that future flash-memory chips may incorporate on-chip error-correction or provide soft information, to improve the error-correction performance. In such a case, the state of the flash cell is expressed as a voltage, which is a continuous quantity, rather than discrete. The written values are read back, with added noise. With this assumption, flash memory strongly resembles conventional AWGN communication systems, which transmit and receive continuous signals. While AWGN systems have a power constraint averaged over time, for the flash memory system, the power constraint is that there is a maximum and minimum value that can be written in each cell.

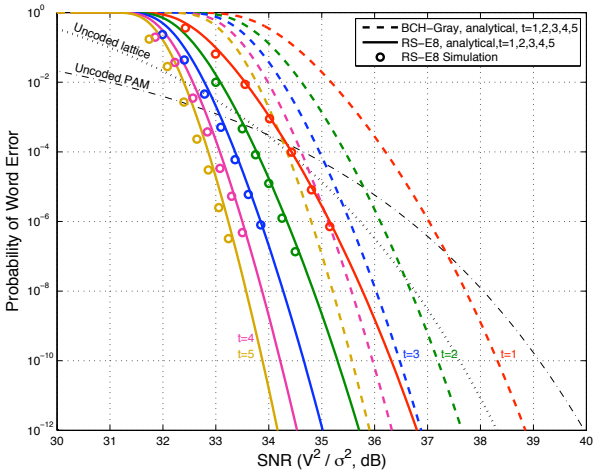

Fig. 1. Evaluation of analytical expressions for $q = 8$ (uncoded 3 bits/cell), and probability of word error from simulation. The proposed construction of RS codes with E8 lattices has 1.7 to 2 dB better performance than BCH codes with PAM.

Because the flash cell values are continuous quantities, this paper takes the signal-space viewpoint that has long been used for the AWGN channel. Among other results, it is now known that lattices can achieve the capacity of the AWGN channel [13] [14], and lattices appear to be a promising practical approach for bandwidth-constrained channels [15]. In fact, a related technique, trellis-coded modulation, has already been considered for error-correction in flash memories [3].

II. Lattices for Flash Memory

This paper gives a brief overview of the author's recent results on using lattices for rewriting and error-correction

kurkoski@ice.uec.ac.jp — Univ. of Electro-Communications, Tokyo, Japan. This research was supported in part by the Ministry of Education, Science, Sports and Culture; Grant-in-Aid for Scientific Research (C) number 21560388.

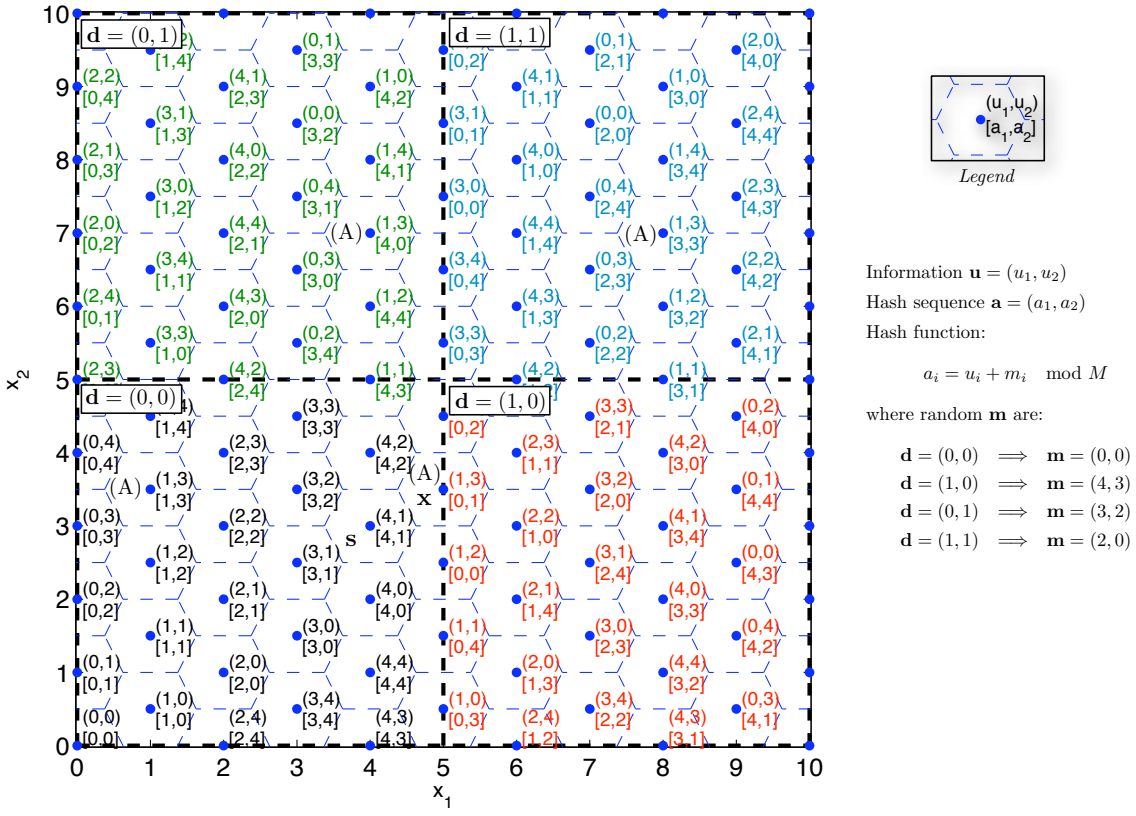

Fig. 2. Illustration of the proposed code for two dimensions, $n = 2, G = [1\ 0; \frac{1}{2}\ 1], M = 5, D = 2$.

in flash memories. Two submitted conference papers are available on arXiv, see References [1] and [2].

It is assumed that the values stored in flash cells correspond to lattice points. From a lattice perspective, conventional flash, using PAM-like encoding, stores data at the points $\{0, \ldots, q-1\}^n$ in a rectangular lattice. However, rectangular lattices are inefficient, and there exist lattices that have many desirable properties such as better packing efficiency. The E8 lattice has a number of desirable properties. Besides being the best-known lattice in eight dimensions, it also has an efficient decoding algorithm. The lattice generator matrix is triangular, which makes it suitable for encoding. In addition, the E8 lattice points are either integers or half-integers; for implementations, this may be more suitable than writing arbitrary values to memory.

A. Lattices for Error-Correction in Flash

For error-correction, the merit of lattices is clear. With integer spacing, the minimum Euclidean distance of PAM is 1. For the E8 lattices, the minimum Euclidean distance is as much as $\sqrt{2} \approx 1.4$. However, an outer error-correcting code is still needed to guarantee data reliability. Because E8 decoding induces burst-like errors, Reed-Solomon (RS) codes constructed over GF(2^8) are used for error correction. Only the modulo-2 value of the lattice points are protected by the RS codes; the Euclidean separation of the lattice points is also important. This system might be regarded as a type of trellis-coded modulation.

Because flash memory operates at high SNR region, analytical expressions of word error performance, based on the union bound, can be developed. Fig. 1 shows the probability of word error using these analytical expressions for $q = 8$ (3 bits per flash cell), and various code rates. The code parameters are in Table I.

At a probability of word error rate of 10^{-12}, the uncoded E8 lattice has approximately 1.7 dB better performance than uncoded PAM. For each code comparison, the error-correction capability of the RS and BCH code is essentially the same. This benefit is preserved, and after coding, gains of 1.7 to 2.0 dB are observed. In addition, simulation of the proposed system shows that the analytical expressions are tight at high SNR.

For more details, see Reference [1].

B. Lattice for Rewriting in Flash

Similarly for a rewriting code based on lattices, the cell values are points of an n-dimensional lattice inside the cube $(0, q-1)^n$. To allow rewriting, there is a one-to-many mapping between from the information to the codebook. To encode an information sequence, the encoder searches over the candidate codewords and selects one. To aid this encoding and search, the codebook is partitioned into subcodebooks, most with a one-to-one mapping. Adding

RS over GF(2^8)			BCH over GF(2^{13})		
(n_c, k_c, t)	cells N	bits k	(n_c, k_c, t)	cells N	bits k
(172,170,1)	1376	4112	(4109,4096,1)	1370	4096
(172,168,2)	1376	4096	(4122,4096,2)	1374	4096
(173,167,3)	1384	4104	(4135,4096,3)	1379	4096
(174,166,4)	1392	4112	(4148,4096,4)	1383	4096
(174,164,5)	1392	4096	(4161,4096,5)	1387	4096

TABLE I
RS AND BCH CODES CONSIDERED IN FIG. 1

a random component to the encoding will improve the average number of writes.

The rewriting code construction is best illustrated using a two-dimensional example, shown in Fig. 2. Since the dimension is $n = 2$, this corresponds to writing data into 2 flash cells. The lattice has a generator matrix $G = [1\ 0; \frac{1}{2}\ 1]$. The code consists of all lattice points inside the square of volume $(MD)^n$, where M and D are two parameters, $M = 5$ and $D = 2$ in this example. This code can write $\log_2 M = \log_2 5$ bits per cell into 2 flash cells. The lattice is then partitioned into $D^n = 4$ blocks. Each block has a one-to-one mapping between information and lattice points, so the overall code has a one-to-four (in general, one-to-D^n) mapping.

Here an encoding example is given. Note that each block has a unique pseudorandom mapping. The original information is denoted $\mathbf{u}$, and this is mapped to a "hashed" sequence $\mathbf{a}$. Assume that the current state of the memory is $\mathbf{s} = (4, 3)$ (indicated in the figure), and the information to be written is $\mathbf{u} = (1, 3)$.

Each block is indexed by a vector $\mathbf{d} \in \{(0,0), (0,1), (1,0), (1,1)\}$. For each $\mathbf{d}$, the hashed sequence is computed, and the candidate vector is found:

$$\begin{aligned}
\mathbf{d} = (0,0) : \mathbf{u} = (1,3) \rightarrow \mathbf{a}[\mathbf{d}] = (1,3) \rightarrow \mathbf{x}[\mathbf{d}] = (1,3.5)\\
\mathbf{d} = (0,1) : \mathbf{u} = (1,3) \rightarrow \mathbf{a}[\mathbf{d}] = (0,1) \rightarrow \mathbf{x}[\mathbf{d}] = (5,3.5)\\
\mathbf{d} = (1,0) : \mathbf{u} = (1,3) \rightarrow \mathbf{a}[\mathbf{d}] = (4,1) \rightarrow \mathbf{x}[\mathbf{d}] = (4,7)\\
\mathbf{d} = (1,1) : \mathbf{u} = (1,3) \rightarrow \mathbf{a}[\mathbf{d}] = (3,3) \rightarrow \mathbf{x}[\mathbf{d}] = (8,7).
\end{aligned}$$

These $\mathbf{x}[\mathbf{d}]$ are indicated in Fig. 2 by "(A)". For the first candidate $(1, 3.5)$, the difference $\mathbf{x}[(0,0)] - \mathbf{s}$ is negative in the first component, and so this point cannot be written, because values are only allowed to increase. For each of the remaining three, the point $\mathbf{x}[(0,1)]$ is the most suitable, and so it is selected as the point to be written, $\mathbf{x}$ (indicated in the figure).

It is fairly clear that with the proposed construction, the *minimum* number of word writes is D. However, it is not so easy to determine the *average* number of word writes. Naturally, there is a tradeoff between code rate and the average number of writes, and this is demonstrated in Fig. 3, obtained by computer simulation. Values of q were fixed, with $q = DM + 1$. The code rate $R = \log_2 M$, and D was allowed to be a non-integer. The most striking feature is that the number of writes depends strongly upon q.

Also shown in Fig. 3 is the average number of writes

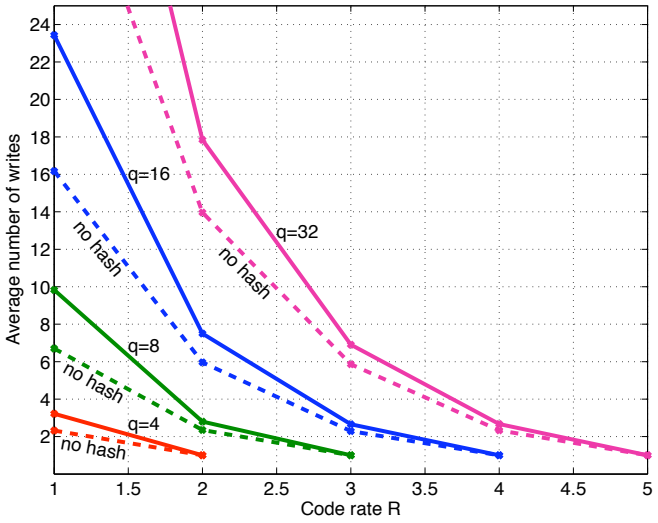

Fig. 3. Average number of word writes using the E8 lattice, with $q-1 = DM$ and code rate $R = \log_2 M$. Code rate R is in bits/cell.

if hashing is not used, that is, the hash vector $\mathbf{m}$ is all-zeros. At low rates, the random hash increases the average number of writes. But as the rate increases, this advantage diminishes. Note that the hash has no influence on the minimum number of writes.

For more details, see Reference [2].

REFERENCES

[1] B. Kurkoski, "The E8 lattice and error correction in multi-level flash memory," submitted to International Conference on Communications, IEEE, 2011. Available http://arxiv.org/abs/1009.5764.

[2] B. Kurkoski, "Rewriting codes for flash memories based upon lattices, and an example using the E8 lattice," in *Proceedings IEEE Global Telecommunications Conference*, (Miami, Florida, USA), IEEE, December 2010. To appear. Available http://arxiv.org/abs/1007.1819.

[3] F. Sun, S. Devarajan, K. Rose, and T. Zhang, "Design of on-chip error correction systems for multilevel nor and nand flash memories," *Circuits, Devices Systems, IET*, vol. 1, pp. 241–249, June 2007.

[4] H. Nobukata and et al., "A 144-mb, eight-level NAND flash memory with optimized pulsewidth programming," *IEEE Journal of Solid-State Circuits*, vol. 35, pp. 682–690, May 2000.

[5] M. Grossi, M. Lanzoni, and B. Ricco, "A novel algorithm for high-throughput programming of multilevel flash memories," *IEEE Transactions on Electron Devices*, vol. 50, pp. 1290–1296, May 2003.

[6] R. L. Rivest and A. Shamir, "How to reuse a "write-once" memory," *Information and Control*, vol. 55, pp. 1–19, December 1982.

[7] G. D. Cohen, P. Godlewski, and F. Merkx, "Linear binary code for write-once memories," *IEEE Transactions on Information Theory*, vol. 32, pp. 697–700, September 1986.

[8] A. Jiang, V. Bohossian, and J. Bruck, "Floating codes for joint information storage in write asymmetric memories," in *Proceedings of IEEE International Symposium on Information Theory*, pp. 1166–1170, June 2007.

[9] V. Bohossian, A. Jiang, and J. Bruck, "Buffer coding for asymmetric multi-level memory," in *Proceedings of IEEE International Symposium on Information Theory*, pp. 1186–1190, June 2007.

[10] E. Yaakobi, A. Vardy, P. H. Siegel, and J. K. Wolf, "Multidimensional flash codes," in *Proceedings 46th Annual Allerton Conference on Communication, Control, and Computing*, (Monticello, IL, USA), pp. 392–399, September 2008.

[11] H. Finucane, Z. Liu, and M. Mitzenmacher, "Designing floating codes for expected performance," in *Proceedings 46th Annual Allerton Conference on Communication, Control, and Computing*, (Monticello, IL, USA), September 2008.

[12] A. Jiang and J. Bruck, "Information representation and coding for flash memories," in *Communications, Computers and Signal Processing, 2009. PacRim 2009. IEEE Pacific Rim Conference on*, pp. 920–925, August 2009.

[13] H.-A. Loeliger, "Averaging bounds for lattices and linear codes," *IEEE Transactions on Information Theory*, vol. 43, pp. 1767–1773, November 1997.

[14] U. Erez and R. Zamir, "Achieving $\frac{1}{2}\log(1+\mathrm{SNR})$ on the AWGN channel with lattice encoding and decoding," *IEEE Transactions on Information Theory*, vol. 50, pp. 2293–2314, October 2004.

[15] N. Sommer, M. Feder, and O. Shalvi, "Low-density lattice codes," *IEEE Transactions on Information Theory*, vol. 54, pp. 1561–1585, April 2008.