Generative Modeling of NAND Flash Memory Voltage Level

Ziwei Liu, Yi Liu, and **Paul H. Siegel**
Center for Memory and Recording Research, University of California, San Diego, La Jolla, CA 92093 U.S.A
iriscloudlau@gmail.com, {yil333, psiegel}@eng.ucsd.edu

***Abstract*—Program and erase cycling (P/E cycling) data is used to characterize flash memory channels and support realistic performance simulation of error-correcting codes (ECCs). However, these applications require a massive amount of data, and collecting the data takes a lot of time. To generate a large amount of NAND flash memory voltage levels using a relatively small amount of measured data, we propose a *voltage level generator* based on Time-Dependent Generative Moments Matching Network (TD-GMMN). This model can generate authentic voltage level distributions over a range of possible P/E cycles for a specified program level based on known voltage level distributions. Experimental results based on data generated by a mathematical MLC NAND flash memory voltage level generator demonstrate the model's effectiveness.**

I. Introduction

When a cell level read operation is performed on a page in a MLC NAND flash memory chip, there is the option of performing a hard read or a soft read. In hard read mode, the device will compare the cell's voltage level with three specified thresholds V_A, V_B, and V_C. If a cell's level is below V_A, it is designated as level E (or 0); if above V_A but below V_B, it is designated as level A (or 1); if above V_B but below V_C, is designated as level B (or 2); and finally, if above V_C, it is designated as level C (or 3). Usually, a cell's read level is equal to its program level 0,1, 2, or 3; otherwise, there is a read error. In soft read mode, the device will compare the cell's voltage level with a more fine-grained set of levels, and the returned value can be treated as the cell's actual voltage level.

Both flash memory hardware characterization and error-correcting code (ECC) design rely heavily on such read voltage level data. Yet, the measurement of such data is very time-consuming. To generate a large amount of NAND flash memory voltage levels using a relatively small amount of measured such data, we propose a *voltage level generator*. This software can generate authentic voltage level distributions over a range of possible P/E cycles and each specified program level after learning from measured target distributions.

II. Training Dataset

The dataset we used in this demonstration is obtained from a Normal-Laplace based model of voltage level distributions in MLC NAND flash memories [1]. The Normal-Laplace model could generate realistic MLC NAND flash memory voltage level distributions for P/E cycles from 10 to 1000, and for each program level respectively, as shown in Fig. 1.

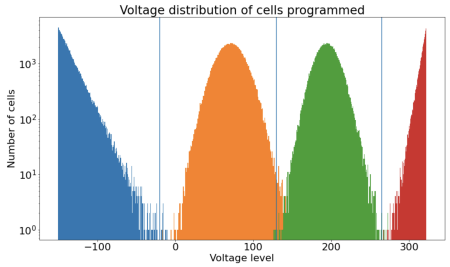

Fig. 1: Distribution of voltage levels at P/E cycle 10 by mathematical voltage level generator

III. Time-dependent Generative Moments Matching Network

In this section, we discuss the structure of the Time-dependent Generative Moments Matching Network (TD-GMMN). Since existing deep learning generative models' performance in learning multi-model Gaussian distributions is sub-optimal [2], we proposed to generate read voltage levels for each program level separately using Generative Moments Matching Network (GMMN) [3]. Motivated by the observation that voltage level distributions at different P/E cycles are different because of the program disturb effects and the wear-out of the flash memory cells, we further designed our neural network to be a time-dependent neural network, whose weights and biases are estimated using degree-3 polynomial functions $f(t) = at^3 + bt^2 + ct + d$, where t represents the P/E cycle count and a, b, c, d are parameters to be learned in the training process [4]. Such a time-dependent neural network was shown to be effective in predicting page bit error counts over a range of P/E cycles in [5].

The TD-GMMN consists of two kinds of layers: time-dependent dense layer (TDDL) $\mathcal{D}(\mathbf{x}, t)$ and arctan layer $\mathcal{A}(\mathbf{x}, t)$. Given an input vector $\mathbf{x} \in \mathbb{R}^m$, each TDDL layer defines the following operation:

$$\mathbf{y} = \mathcal{D}(\mathbf{x}, t) = \mathbf{W}(t)\mathbf{x} + \mathbf{B}(t) \tag{1}$$

where $\mathbf{y} \in \mathbb{R}^n$ is the output vector and $\mathbf{W} \in \mathbb{R}^{n \times m}$, $B \in \mathbb{R}^n$ are network parameters to be learned in the training.

The TD-GMMN contains three TDDLs. The input to the first TDDL is noise samples drawn from a normal distribution of size 1. The output vector of size 8 is then put into the second TDDL, resulting in an output of size 8. The third TDDL has the input size 8 and output size 1. The size of both the input and output of the complete network structure

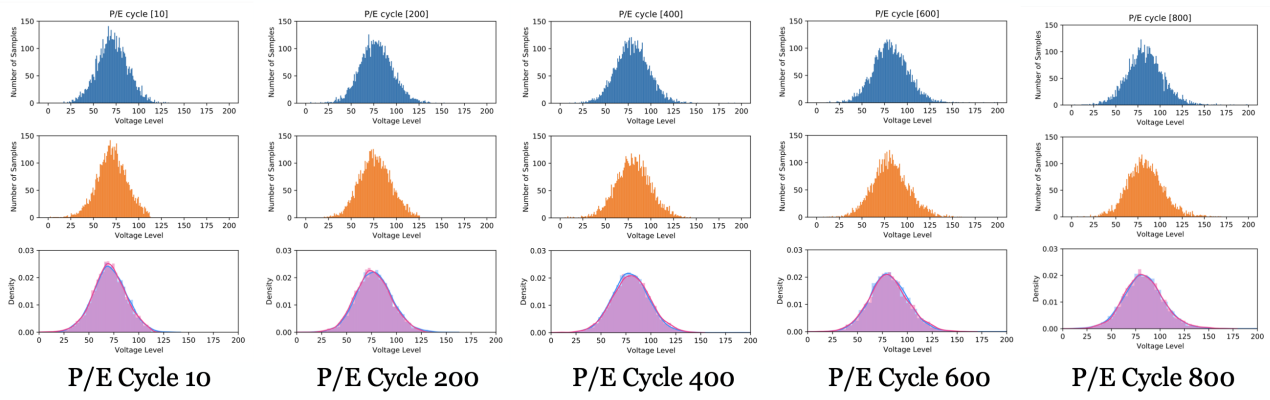

Fig. 3: The histogram of samples drawn from the Normal-Laplace model (top), generated voltage levels (middle), and the estimated probability density function for both (bottom) for MLC program level 1 at P/E cycles 10, 200, 400, 600 and 800, separately. Number of samples: 5,000.

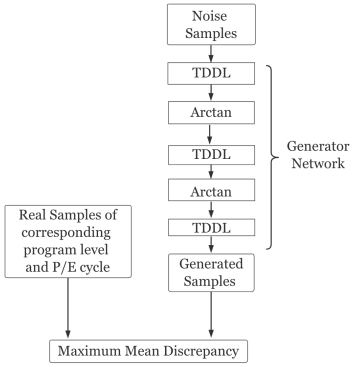

Fig. 2: The workflow for voltage level distribution generation.

is 1, guaranteeing that generated samples are independently and identically distributed. The network can be expressed as below:

$$\mathbf{y} = \Phi(\mathbf{x}, t, e) = \mathcal{D}_2(\mathcal{A}(\mathcal{D}_1(\mathcal{A}(\mathcal{D}_0(\mathbf{x}, t))))) \tag{2}$$

Output $\mathbf{y}$ is a size 1 vector representing the voltage level of one cell at P/E cycle count t and programmed level e.

The loss function is based on Maximum Mean Discrepancy, which measures the mean squared difference between the statistics of two sets of samples and, in this way, judges the possibility of them coming from the same distribution [6], as shown in (3). Then the Gaussian kernel trick (5) was used to cover all orders of means, which leads to (4).

$$\begin{aligned}\mathcal{L}_{MMD^2} &= ||\frac{1}{N}\sum_{i=1}^{N}\phi(x_i) - \frac{1}{M}\sum_{j=1}^{M}\phi(y_i)||^2 \\ &= \frac{1}{N^2}\sum_{i=1}^{N}\sum_{i'=1}^{N}\phi(x_i)^T\phi(x_{i'}) - \frac{2}{NM}\sum_{i=1}^{N}\sum_{j=1}^{M}\phi(x_i)^T\phi(y_j) \\ &+ \frac{1}{M^2}\sum_{j=1}^{M}\sum_{j'=1}^{M}\phi(y_j)^T\phi(y_{j'})\end{aligned} \tag{3}$$

$$\begin{aligned}\mathcal{L}_{MMD^2} &= \frac{1}{N^2}\sum_{i=1}^{N}\sum_{i'=1}^{N}k(x_i, x_{i'}) - \frac{2}{NM}\sum_{i=1}^{N}\sum_{j=1}^{M}k(x_i, y_j) \\ &+ \frac{1}{M^2}\sum_{j=1}^{M}\sum_{j'=1}^{M}k(y_j, y_{j'})\end{aligned} \tag{4}$$

$$k(x, x') = \exp(-\frac{1}{2\sigma}|x - x'|^2) \tag{5}$$

Here σ is the bandwidth parameter. Fig. 2 shows the workflow of training the TD-GMMN voltage level generator.

IV. Experimental Results

We draw samples from the Normal-Laplace voltage level generator described in [1] as the real samples to guide the training. The number of the real samples are the same as the generated samples when calculating the maximum mean discrepancy. Fig. 3 shows the experimental results of using the deep learning-based voltage level generator to generate voltage levels for program level 1 at P/E cycles ranging from 10, 200, 400, 600, and 800.

References

[1] T. Parnell, N. Papandreou, T. Mittelholzer, and H. Pozidis, "Modelling of the threshold voltage distributions of sub-20nm NAND flash memory," in *2014 IEEE Global Communications Conference*, 2014, pp. 2351–2356.

[2] F. Farnia, W. Wang, S. Das, and A. Jadbabaie, "GAT-GMM: Generative adversarial training for gaussian mixture models," *arXiv preprint arXiv:2006.10293*, 2020.

[3] Y. Li, K. Swersky, and R. Zemel, "Generative moment matching networks," in *International Conference on Machine Learning*, 2015, pp. 1718–1727.

[4] D. Holden, T. Komura, and J. Saito, "Phase-functioned neural networks for character control," *ACM Trans. Graph.*, vol. 36, no. 4, Jul. 2017.

[5] Y. Liu, S. Wu, and P. Siegel, "Bad page detector for NAND flash memory," in *11th Annual Non-Volatile Memories Workshop (NVMW)*, Mar. 2020.

[6] A. Gretton, K. M. Borgwardt, M. Rasch, B. Schölkopf, and A. J. Smola, "A kernel method for the two-sample-problem," ser. NIPS'06. Cambridge, MA, USA: MIT Press, 2006, p. 513–520.