A Versatile Emulator for the Aging Effect of Non-Volatile Memories: The case of NAND Flash

Antonios Prodromakis, Stelios Korkotsides and Theodore Antonakopoulos
Department of Electrical and Computer Engineering
University of Patras
Patras, 26504, Greece
e-mails: aprodromakis@upatras.gr, stelkork@ece.upatras.gr and antonako@upatras.gr

***Abstract*—This work presents a versatile and flexible FPGA-based platform, designed for accurate emulation of the aging effect of non-volatile memories. The emulator is based on a reconfigurable hardware-software architecture which enables the accurate representation of various non-volatile memory technologies, like NAND Flash and PCM. The proposed architecture can be used for emulating memories at the cell, chip and system level, while the proposed hardware platform can be used as a valuable tool for the development and evaluation of memory-related algorithms and techniques. In this paper, we analyze the architecture of the non-volatile memory emulator, focusing mainly on the NAND Flash case, we present details about its internal functionality and, using experimental results, we demonstrate the high accuracy achieved when it is used to emulate a specific NAND Flash chip.**

***Index Terms*—Non-volatile memories, NAND Flash, Phase Change Memory, Memory Aging, FPGA emulator.**

I. Introduction

Over the last few years, non-volatile memory (NVM) has shown a great potential in replacing volatile memory, like dynamic random access memory (DRAM), and magnetic hard disk drives (HDDs) in caching and storage applications. NAND Flash-based solid state drives (SSDs) have already emerged as a low-cost, high-performance and reliable storage medium for both commercial and enterprise storage systems. Additionally, the properties of phase-change materials and the recent scaling of Phase-Change Memory (PCM) has made it a perfect candidate for developing phase-change random access memories (PCRAMs).

The rapid scaling of NVMs, with process nodes below 19nm, and the use of multi-level cell (MLC) technologies has increased their storage density and reduced the storage cost per bit dramatically. However, their lifetime capacity has not remained unaffected. Different noise sources and interferences along with aging effects have now a great impact on the reliability and endurance of these memory technologies, and hence, on the storage systems where these memories are used (SSDs, PCRAMs). Numerous methods and techniques, such as wear-leveling, specialized error correcting codes (ECC) and precoding techniques have been employed to compensate these effects [1], [2], [3], [4], while others, more complex but also more efficient, like dynamic adaptation of read reference thresholds, are at an experimental level [5].

The development of these techniques are based on experimental characterization of NVM cells and chips. Characterization is related with measuring bit error ratio (BER) and response time (read and write time) during the whole lifetime of a device, for various loading data patterns and timing scenarios. This process is performed using real NVM ICs, usually the engineering, pre-production parts, while more thorough testing at the system level is performed when production parts are available. This approach has two major drawbacks. On one hand it is a very time-consuming process, since the aging of an NVM may require a large number of program/erase (P/E) cycles to be performed for each experiment, ranging from tens of thousands (NAND Flash) to millions (PCM) program cycles. On the other hand, the aging characteristics of an NVM are proportionally dependent on the number of the performed P/E cycles, thus making it impossible to conduct different or successive experiments at the same aging state of a memory chip.

In [6] we presented a model that accurately represents the aging process of a MLC NAND Flash cell, while in [7] the analysis of a MLC NAND Flash memory as a time-variant communications channel, based on the asymmetric 4-PAM model, was presented. In this paper, we expand our work and present the architecture of a flexible FPGA-based platform, designed for accurate emulations of NVM technologies, focusing mainly on MLC NAND Flash technologies. Accuracy is measured in reference to experimentally specified bit error probabilities for various aging conditions (ie. the number of P/E cycles applied to a NAND Flash chip), usually for random data patterns.

The hardware platform presented in this work is based on a reconfigurable hardware-software architecture which enables the accurate emulation of new and emerging technologies and models of NVMs. The developed platform can be a valuable tool for the evaluation of memory-related algorithms, signal processing and coding techniques.

The remainder of this paper is organized as follows. Section II analyzes the most common architectures and I/O interfaces of NAND Flash and PCM. Section III analyzes the level distributions of these two NVM technologies and how they are affected by the aging conditions. In Section IV we discuss the main architectural components of the presented NVM emulator, while in Section V we present two cases of using

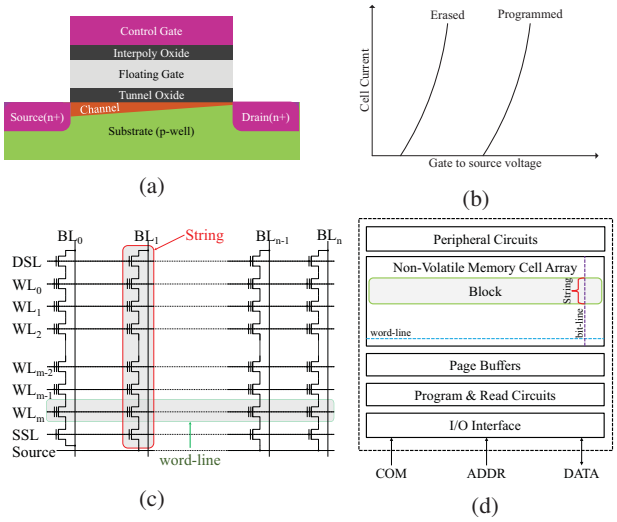

Fig. 1: (a) Floating Gate transistor, (b) I-V Characteristic, (c) NAND Flash block, (d) Simplified chip architecture

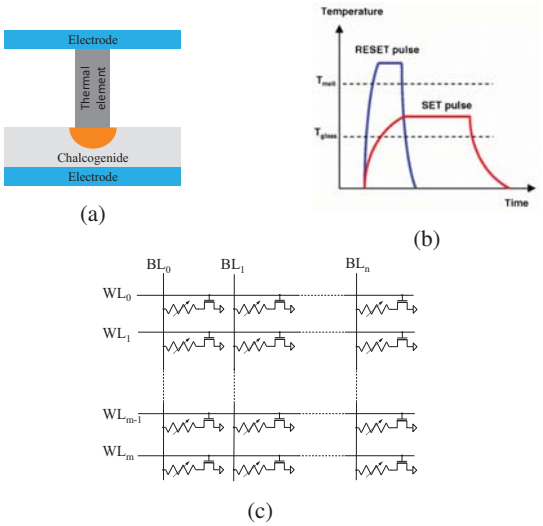

Fig. 2: (a) Phase Change Element, (b) Programming Pulses, (c) PCM circuit schematic

the proposed system. Finally, in Section VI we present experimental results of a NAND Flash chip and demonstrate how the designed emulator accurately emulates its BER behavior.

II. NAND Flash and PCM Technologies

NAND Flash memories are based on Flash cells which are implemented using floating gate transistors (FGTs), that is, field effect transistors (FETs) with an additional floating gate between the substrate and the control gate, as depicted in Fig. 1a. The effective threshold voltage of FGT, and thus its I-V characteristic, depends on the charge stored in its floating gate (Fig. 1b) [8]. NAND Flash acquires non-volatile properties as the floating gate is surrounded by dielectrics which ensure the reliable isolation of the trapped charge for long periods of time [9]. Fig. 1c illustrates the interconnection of FGTs in a NAND Flash block. Groups of FGTs sharing the same bit-line are connected in series, forming strings, while logical pages are formed by cells sharing the same word-line. All strings of cells sharing the same group of word-lines form a NAND Flash block.

Programming of NAND Flash cells is achieved by applying bias voltages to the control gate and the drain of the FGTs, which causes the Fowler-Nordheim (FN) tunnelling phenomenon and traps the charge into the floating gate. This operation is only allowed if the cell was previously in the erased state (no charge stored in the floating gate). Information stored in the cells is read by applying a small voltage at the drain of the FGT and sensing the current that flows through it. Program and read commands are performed on a page basis, while erase is performed on a block basis.

Fig. 1d illustrates a simplified block diagram of a NAND Flash chip, which consists of a 2D memory cell array, page buffers, program and read circuits and the I/O interface. The interface between a NAND Flash chip and its controller consists of three sets of I/O signals. The COM set determines which command shall be executed, (i.e. program, erase, read), the ADD I/O set specifies the row and/or column address that the command refers to, while the DATA I/O set contains the data to be read or those to be programmed. The most common I/O interfaces of NAND Flash chips are the Toggle [10] and the ONFI [11], with data rates up to 400 MBps.

PCM memories, also known as storage class memories, are based on PCM cells which consist of an active phase-change element (PCE), which lies between two electrodes and a heating element, as shown in Fig.2a. In order to store information, PCM cells take advantage of the ability of reversible, thermally induced phase transformation of chalcogenide alloys, such as $Ge_2Sb_2Te_5$. This phase transformation can set the structure of chalcogenide alloy in multiple states, which have different electrical resistance. The poly-crystalline state results in a high resistance, while in the amorphous state the chalcogenide alloy has low resistance. The transition between these states can be defined by a RESET and SET operation respectively.

The RESET operation is performed by slowly heating the memory element above its melting point and then rapidly cooling it down in order to maintain its amorphous molecule structure. On the other hand, in the SET operation, the cell is heated above its glass temperature for a longer period of time, and thus the material is crystallized. The temperature variations during programming are shown in Fig.2b. Using advanced writing techniques, setting the cell's resistance in states between SET and RESET has become feasible and multi-level cell programming in PCM is a reality nowadays [5].

In SLC memories, the programming of a cell is performed with the use of SET and RESET operations. In MLC memories, a more complicated scheme is used in order to place the chalcogenide alloy in discrete intermediate states. Reading is performed by precharging a bit-line and measuring its discharge time, which is directly connected to the resistance

of the selected cells. Fig.2c illustrates the interconnection of PCEs in a 2-D array. The selection of a PCM cell in the memory array is performed by a FET. In contrast to NAND Flash, PCM access can be performed at byte level, although small blocks of data are usually used. Since PCM was initially developed for DRAM replacement, PCM chips use DRAM interfaces, like DDR3, but PCM can also be integrated in chips that use ONFI/Toggle-like interfaces, when storage applications are targeted [12].

Both NV memory technologies described in this section are affected by the number of programm cycles that have been applied since the first use of these devices. In NAND Flash, programming can be performed only once after an erase cycle, while in PCM phase alternation determines a write cycle. Therefore, aging is determined differently in these technologies. Generally speaking, for NAND Flash aging is related with the number of program/erase cycles applied, while for PCM, aging is determined by the number of set/reset cycles applied. Although the following sections are mainly related with the emulation of NAND Flash memories, the main ideas implemented in the presented emulator can also be applied to PCM with the appropriate modifications to the noise model.

III. The Impact of Aging on the Reliability of NAND Flash

A. Modeling Level Distributions

Storing data to an MLC NVM is achieved by accurately programming its memory cells into intermediate voltage (Flash based) or resistance (PCM) levels. More specifically, for an n-bit/cell NVM, each cell can be programmed into 2^n different levels. Each level corresponds to a symbol, represented as an n-bit binary vector, which can be mapped using different schemes (i.e. Gray mapping, direct mapping) [1]. As the number of states increases, the margin separating them diminishes, thus MLC NVMs are more vulnerable to noise sources than the SLCs. The voltage/level distributions are affected by different noise sources, such as cell-wearing in NAND Flash memories or resistance drifting in PCM, while cell to cell interference (CCI) plays a major role when process nodes below 30nm are used [13]. It has been shown in [14] and [15] that each NAND Flash cell can be modeled as a level-dependent additive white noise channel (AWGN) with high accuracy. Although the relative characterization of PCMs is an on-going process, the same assumption can also be made for PCMs [16], [12].

The noise characteristics depend on the aging state of the cell and the input symbol s, the noise is data-dependent. Let L_s denote the ideal level of the input symbol s, then the read-out signal S will be a random variable with the likelihood function (1), where μ_s and σ_s are the mean and standard deviation of level-dependent Gaussian noise:

$$p(S) = \frac{1}{\sqrt{2\sigma_s^2\pi}} e^{-(S-(\mu_s+L_s))^2/2\sigma_s^2} \quad (1)$$

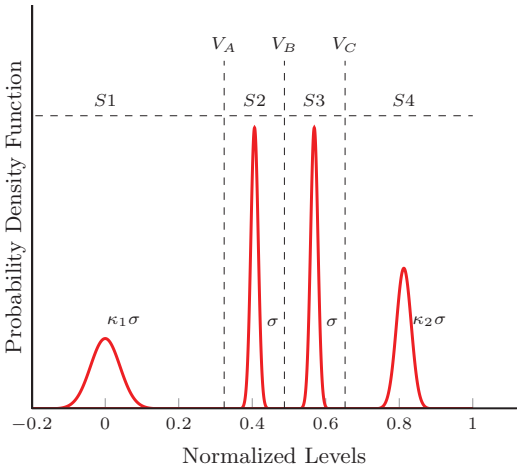

Fig. 3: Level distributions of a 2-bits/cell NAND Flash

Cycling a memory cell alters the parameters μ and σ of the Gaussian probability density function. The mean noise levels are usually shifted to higher values and level distributions become wider. Consequently, level distributions of different symbols may increase their overlap and an erroneous read of the stored information is more likely to happen, therefore leading to higher raw bit error ratio (RBER).

In this paper, we use the results of [17] to determine the normalized error-free level L of each symbol. Furthermore, we assume that if σ denotes the AWGN standard deviation of an intermediate level then the outer levels (erased and fully programmed states) will have a standard deviation of $k_1\sigma$ and $k_2\sigma$, respectively. Fig. 3 presents the level distributions for each model. This approach can cover all different NAND Flash technologies presented in the existing literature. For example, based on [18], $k_1 = 2$ and $k_2 = 1$, while based on [19], $k_1 = 1.5$ and $k_2 = 1.2$. Additionally, $k_1 = 4$ and $k_2 = 2$ as in [1] and [2]. Table I illustrates the normalized means and standard deviations for each state, as well as two coding schemes used to map the binary vectors into symbols in 2-bits/cell NAND Flash.

TABLE I: Normalized mean and standard deviation of level distributions of a 2-bits/cell NAND Flash

State	$S1$	$S2$	$S3$	$S4$
Direct Mapping	11	10	01	00
Gray Mapping	11	01	00	10
Normalized Mean	0.0	0.40625	0.56875	0.8125
Standard Deviation	$k_1\sigma$	σ	σ	$k_2\sigma$

IV. Emulating NVMs as a Time-Variant Communications Channel

Emulating NVM for achieving accurate bit error characteristics for a given technology requires that the relationship

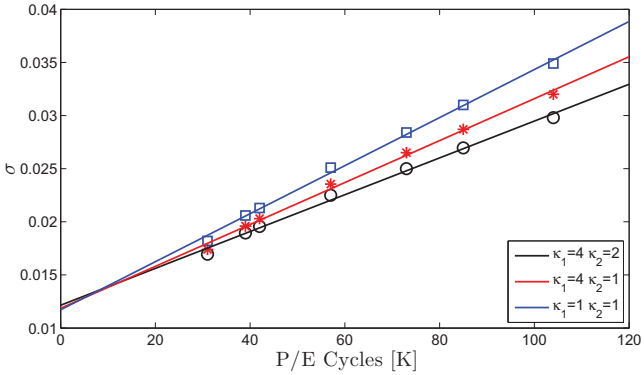

Fig. 4: Standard deviation as a function of P/E cycles for different models of NAND Flash.

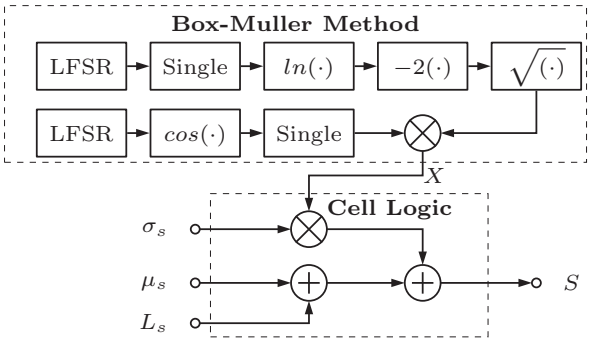

Fig. 5: Block diagram of the Cell Core architecture.

between P/E cycling and the noise characteristics to be determined. By treating the memory device as an asymmetric n-PAM communication channel with time-variant (aging) characteristics (the case of 4-PAM has been analyzed in [7]), one can express the relation of RBER with μ and σ of the Gaussian probability density function in a closed form. Furthermore, this analysis can be extended for non-equiprobable data and for different noise models determined by k_1 and k_2.

Although the value of σ is a statistical metric which is not directly measurable in a memory device, the qualitative similarity between RBER as a function of σ and RBER as a function of P/E cycles indicates that there is a relationship between σ and the number of P/E cycles. In [14], it is stated, with a justification based on measurements, that this relationship is linear for the nominal lifetime of a memory device. We have verified this by studying measurements from several MLC memories and different statistical characteristics of the Gaussian distributions. Moreover, as shown in Fig. 4, the linear relationship is preserved in all three different MLC NAND Flash models.

The importance of this observation lies on the fact that the bit error emulation of an NVM can be accomplished with high precision by observing its aging behavior, without any knowledge of its internal architecture or the electrical specifications of its cells. However, if we are interested in emulating the internal electrical characteristics of memory cells (e.g. the threshold voltages in a NAND flash memory cell), then the mean values of the distributions must be provided, since they cannot be acquired by mere observation.

V. The NVM Emulator

The aim of the proposed design is to develop a high performance emulation platform, able to interface with existing NVM controllers and emulate the bit error characteristics of different NVM technologies with high accuracy. Emulations can be performed at any user-specified state of the aging process, thus eliminating the need of cycling on a real memory device. Furthermore, it provides the capability to conduct different experiments at the same aging conditions, which is not possible when real chips are used. The proposed NVM emulator is capable of conducting experiments at the cell, page or even chip level.

A. Memory Cell Emulation

As analyzed in section III, a NVM cell can be modeled as a level dependent AWGN channel. The implementation of a Gaussian noise generator is based on the Box-Muller method [20]. According to this method, if a and φ are independent random variables from the same uniform density function on the intervals $(0, 1)$ and $(0, 2\pi)$ respectively, and

$$X = \sqrt{-2 \cdot \ln(a)} \cdot \cos(\varphi) \tag{2}$$

then X will be a variable from the normal distribution with unit variance, and zero mean ($X \sim \mathcal{N}(0, 1)$). Finally, if

$$S = (X \cdot \sigma_s) + (\mu_s + L_s), \tag{3}$$

then $S \sim \mathcal{N}(\mu_s + L_s, \sigma_s^2)$. The implementation of each of the Cell Cores, shown in Fig. 5, is based on the architecture described in [6]. Each Cell Core has three inputs, the mean (μ_s) and standard deviation (σ_s) of the noise, as well as the ideal level (L_s) of the input symbol. The result of the cell emulation is a single precision read-back signal S.

Apart from the Cell Cores, we implemented two additional modules, the Aging Logic and the Hard-decision Logic. The former maps the user-specified aging condition (P/E cycles) to the equivalent noise characteristics (μ_s, σ_s) based on the provided distribution model and the analysis of section IV. The latter implements the decoding of the read-back signals to n-bit symbols. The hard-decision is taken based on the provided read reference thresholds or by applying a dynamic adaptation of a read reference thresholds algorithm.

B. Page and Chip Emulation

There are several architectures which determine how pages are formed within word-lines. For example in the odd/even bit-line architecture of a NAND Flash, odd pages are formed by cells belonging to the odd bit-lines, while even pages are formed by cells belonging to the even ones, respectively. However, in the all bit-line (ABL) architecture there is no such

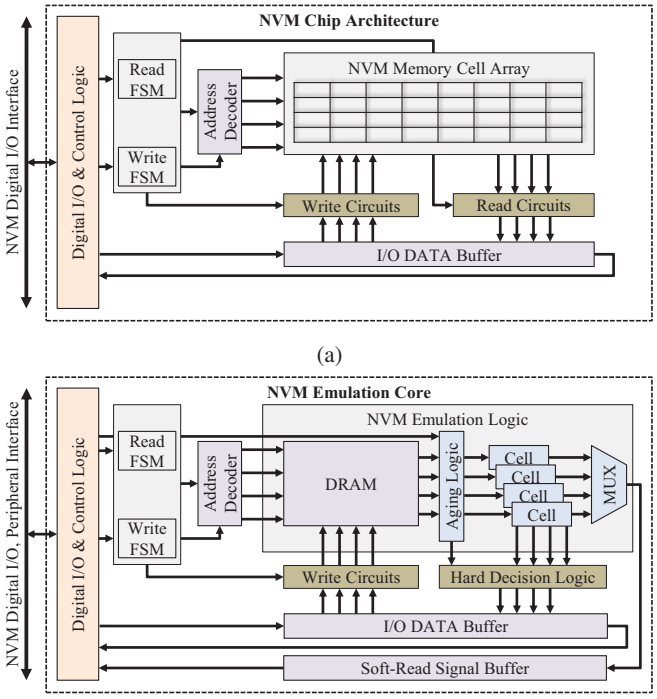

Fig. 6: (a) NVM chip architecture, (b) NVM emulator's architecture.

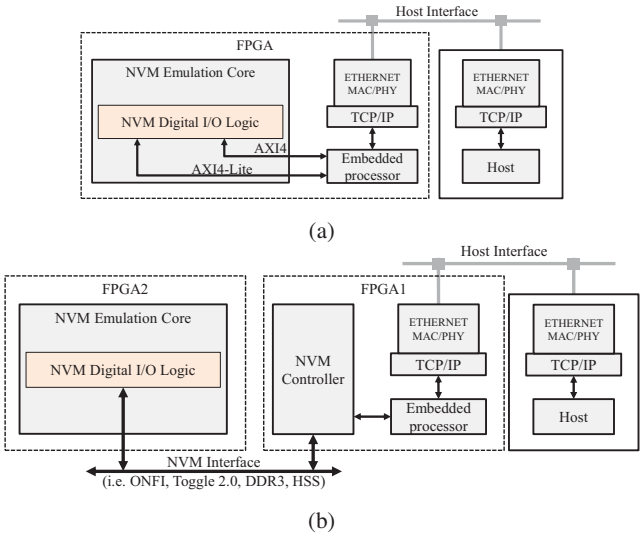

Fig. 7: Implementation of the NVM Emulator (a) as an AXI Peripheral, (b) as an NVM device

separation [10]. Without loss of generality, in this work we assume that each word-line of cells contains only one logical page.

Fig. 6a demonstrates the internal architecture of an NVM chip, while Fig. 6b highlights the architecture of the presented NVM emulator. The emulator's block diagram has been configured at a module-level abstraction in order to demonstrate its structural similarity with a real NAND Flash chip. The main parts of the NVM emulator consist of the Digital I/O & Control Logic, the Read and Write FSMs, the bi-directional Page Buffer, as well as the modules to emulate the cell array and the Hard-Decision circuits.

In a real NVM device, the Digital I/O & Control Logic decodes the commands sent by the controller through the NVM Interface and activates the respective FSMs. When a *page program* command is executed, data are stored in the internal page buffer (I/O DATA Buffer) and then they are stored to the equivalent cells of the cell array using the write circuits. Respectively, when a *page read* command is executed, data are read from the cell array with the use of the read circuits and are stored to the page buffer, by exploiting the hard-decision circuits. The Address Decoder translates the column and row address of the command to be executed and activates the write and read circuits to perform operations on the specified cells. When a *block erase* command is executed, a set of pages is programmed to erase states.

The operations of the NVM emulator are based on the functionality of real memory devices. In the case of a *page program* command, data are stored to the internal I/O DATA Buffer, from which they are written to the respective address of a dynamic random access memory (DRAM). When a page read command is executed, data are read from the DRAM and are supplied to the Emulation Logic module, which consists of up to eight parallel Cell Cores. The read-out of 16-bits signals are then stored to the Soft-Read Signal Buffer, in parallel are hard-decoded by the Hard-Decision module and stored back to the I/O DATA Buffer as n-bit symbols.

C. The Emulator's Architecture

The NVM Emulation Core presented in the previous subsection can be used in various system configurations, depending on the specific application. Fig. 7a presents a version of the NVM Emulator in which the NVM Digital I/O Logic has been designed to interface with a microprocessor, such as an AMBA AXI4 peripheral for embedded systems. The advantages of this architecture are two-fold. First, it provides the ability of testing and debugging the NVM emulator without the use of a specific NVM controller. Secondly, emulation of the bit error characteristics of a NVM is performed independently to the I/O interface. In this version we have used an FPGA board which contains the NVM Emulator, an embedded microprocessor and an Ethernet connection to interface with a host machine. The host machine contains a user-friendly environment, with the use of which we can conduct experiments by employing a set of high-level commands. The communication between the host machine and the embedded microprocessor has been designed using the TCP/IP protocol stack and a custom data transfer protocol [16]. The microprocessor processes the commands sent by the host machine and initiates the respective operations at the NVM Emulator.

Fig. 7b presents a second architecture that uses the NVM Emulator on another system-abstraction level. The advantage

of this approach is that it gives the ability to replace a real NAND Flash chip with the NVM Emulator and to conduct experiments by exploiting an existing NVM controller. Hence, the development of various signal processing algorithms and techniques can be applied directly to the data acquired by the NVM Emulator in the same way as if a real chip was used. Using this approach, two FPGA boards are necessary to implement the overall system. One FPGA board contains the NVM controller and the embedded microprocessor, while the second FPGA board contains the NVM Emulation Core. In this case, the microprocessor processes the commands sent by the host machine and initiates the operations at the NVM Emulator by using the NVM Controller. The NVM Controller is designed as an AMBA AXI4 peripheral and implements the data acquisition and the logic to interface with the NVM Emulation Core. Data symbols and commands, such as page read and page program, are exchanged between the controller and the emulator via this module. Additional commands to determine the aging conditions and other properties of the system must also be provided as additional commands by the controller. The NVM Controller Interface can be implemented using typical NVM interfaces such as ONFI and Toggle.

The above described NVM Emulator can be used to emulate not only single chip NVM chips, but also more complex configurations. For example, multiple NVM chips can also be emulated using multiple instantiations of the presented NVM Emulator, forming a single NVM channel with multiple NVM chips sharing the same data lines and operating on a pipeline fashion. Depending on the available glue logic, multiple instantiation of the aforementioned NVM channel can also be used to emulate the whole storage area of a Solid-State Device with multiple NVM channels.

VI. Experimental Results

This section presents the experimental results of the NVM Emulator, when implemented as an AMBA AXI4 peripheral. The design has been developed using the Xilinx Vivado 2013.4 suite and implemented on a Zynq-7000 zc706 Xilinx evaluation board. Experimental loading scenarios and analysis have been performed using the MATLAB environment at the host side.

We used a 2-bits/cell commercial MLC NAND Flash as the emulation target. The internal characteristics of the memory, such as voltage threshold distributions and their variation as a function of its aging state were unknown. We assumed that the effect of the mean voltage drift was negligible compared to the effect of the standard deviation and therefore during the emulation process we kept the mean noise value equal to zero. This fact was compensated by adjusting the noise's standard deviations as the number of P/E cycles was increased. For the voltage threshold distribution model we used the values $k_1 = 4$ and $k_2 = 2$. Fig. 8 illustrates the relationship between BER as a function of noise's standard deviation according to the asymmetric 4-PAM model of [7].

The next step was to determine the relationship between BER and P/E cycles of the real NAND Flash memory. This process was performed by erasing, programming with random data and reading the pages of various memory blocks, while the raw BER was computed at each P/E Cycle. Due to the fact that the outcome curve had significant fluctuations, the BER curve was approximated by a fitting curve. In Fig. 9 we present

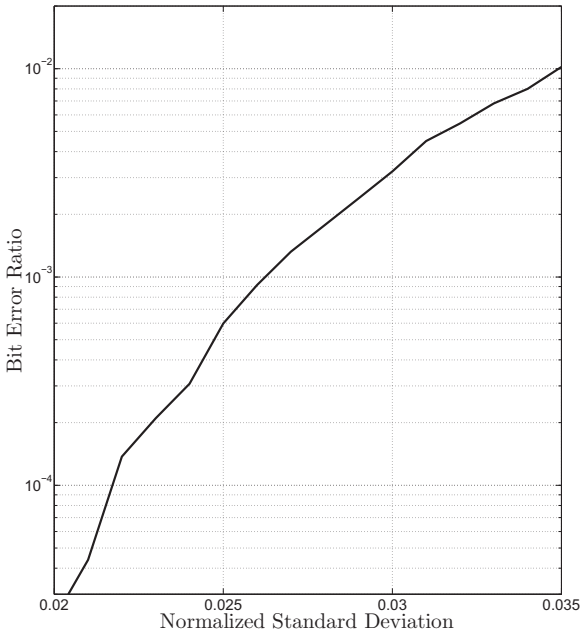

Fig. 8: BER as a function of standard deviation.

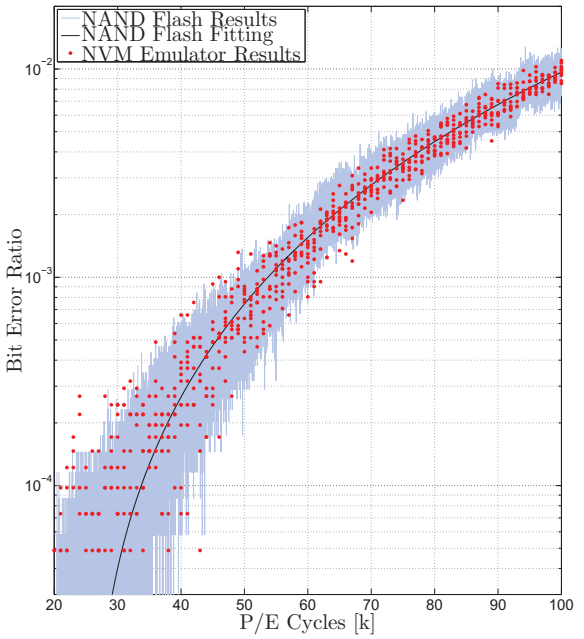

Fig. 9: BER as a function of P/E cycles, using a NAND Flash device and the NVM Emulator.

the measurements of BER as a function of P/E cycles of the NAND Flash chip. The experimental BER results are indicated with gray-blue, while the solid dark-blue curve represents the fitting curve.

Using the two curves of Figs. 8 and 9, we determined the relation between standard deviation and P/E cycles for the whole lifetime of the device. For a given BER value in Fig. 8 we determine the standard deviation and for the same BER value in Fig. 9 we find out the respective P/E cycles value. Therefore, the Aging Logic of the NVM Emulator was configured with the outcome parameters of the above mentioned process. Then for various aging conditions we collected measurements using the NVM Emulator using the same procedure as the real NAND Flash device. A set of commands was applied (block erase, programming all pages of the block with random data and reading them back) and BER statistics were collected. The emulator's BER measurements are also indicated in Fig. 9, where they are marked as red dots. Comparing the experimental results of the real NAND Flash device with the results generated by the proposed NVM Emulator, when it is configured with the appropriate parameters, it becomes obvious that the presented emulator can represent accurately the behavior of a real NVM device, a MLC NAND Flash chip in our case.

VII. Conclusions

In this paper, we presented the architecture and functionality of a real-time emulator for non-volatile memories. The emulator can accurately represent the bit error characteristics of a real memory device during its whole lifetime, by associating the device's aging conditions with emulator's internal parameters. Experimental results from real NAND Flash memories have been used to validate the emulator's performance.

The presented NVM Emulator provides a valuable tool for the development and evaluation of memory-related algorithms, interface circuits and even whole storage systems, since it offers real-time and high precision emulation under user-defined aging conditions and adjustability to the characteristics of the emulated NVM technology. The emulator supports single-level and multi-level cells and can be used for repetitive experiments under the same aging conditions, a procedure that cannot be performed for a large number of experiments on a real memory device due to the programming-related aging process.

References

[1] B. Chen, X. Zhang, and Z. Wang, "Error correction for multi-level nand flash memory using reed-solomon codes," in *Signal Processing Systems, 2008. SiPS 2008. IEEE Workshop on*, 2008, pp. 94–99.

[2] F. Sun, K. Rose, and T. Zhang, "On the use of strong bch codes for improving multilevel nand flash memory storage capacity," in *IEEE Workshop on Signal Processing Systems (SiPS): Design and Implementation*, 2006.

[3] Z. Wang, M. Karpovsky, and A. Joshi, "Reliable mlc nand flash memories based on nonlinear t-error-correcting codes," in *Dependable Systems and Networks (DSN), 2010 IEEE/IFIP International Conference on*, 2010, pp. 41–50.

[4] W. Xu and T. Zhang, "A time-aware fault tolerance scheme to improve reliability of multilevel phase-change memory in the presence of significant resistance drift," *Very Large Scale Integration (VLSI) Systems, IEEE Transactions on*, vol. 19, no. 8, pp. 1357–1367, 2011.

[5] H. Pozidis, N. Papandreou, A. Sebastian, T. Mittelholzer, M. BrightSky, C. Lam, and E. Eleftheriou, "Reliable mlc data storage and retention in phase-change memory after endurance cycling," in *Memory Workshop (IMW), 2013 5th IEEE International*, 2013, pp. 100–103.

[6] A. Prodromakis, G. Sklias, and T. Antonakopoulos, "Emulating the aging of nand flash memories as a time-variant communications channel," in *The 6th International Symposium on Communications, Control, and Signal Processing (ISCCSP 2014)*, 2014.

[7] S. Korkotsides, G. Bikas, E. Eftaxiadis, and T. Antonakopoulos, "Ber analysis of mlc nand flash memories based on an asymmetric pam model," in *The 6th International Symposium on Communications, Control, and Signal Processing (ISCCSP 2014)*, 2014.

[8] J. Brewer and M. Gill, *Nonvolatile Memory Technologies with Emphasis on Flash: A Comprehensive Guide to Understanding and Using Flash Memory Devices*. Wiley. com, 2011, vol. 8.

[9] R. Bez, E. Camerlenghi, A. Modelli, and A. Visconti, "Introduction to flash memory," *Proceedings of the IEEE*, vol. 91, no. 4, pp. 489–502, 2003.

[10] K. Kanda, N. Shibata, T. Hisada, K. Isobe, M. Sato, Y. Shimizu, T. Shimizu, T. Sugimoto, T. Kobayashi, N. Kanagawa, Y. Kajitani, T. Ogawa, K. Iwasa, M. Kojima, T. Suzuki, Y. Suzuki, S. Sakai, T. Fujimura, Y. Utsunomiya, T. Hashimoto, N. Kobayashi, Y. Matsumoto, S. Inoue, Y. Suzuki, Y. Honda, Y. Kato, S. Zaitsu, H. Chibvongodze, M. Watanabe, H. Ding, N. Ookuma, and R. Yamashita, "A 19 nm 112.8 mm^2 64 gb multi-level flash memory with 400 mbit/sec/pin 1.8 v toggle mode interface," *Solid-State Circuits, IEEE Journal of*, vol. 48, no. 1, pp. 159–167, Jan 2013.

[11] "Open nand flash interface specification, revision 2.0," *ONFI Workgroup*, 2008.

[12] N. Papandreou, A. Pantazi, A. Sebastian, M. Breitwisch, C. Lam, H. Pozidis, and E. Eleftheriou, "Multilevel phase-change memory," in *Electronics, Circuits, and Systems (ICECS), 2010 17th IEEE International Conference on*, 2010, pp. 1017–1020.

[13] K. Prall, "Scaling non-volatile memory below 30nm," in *Non-Volatile Semiconductor Memory Workshop, 2007 22nd IEEE*, Aug 2007, pp. 5–10.

[14] Y. Cai, E. F. Haratsch, O. Mutlu, and K. Mai, "Threshold voltage distribution in mlc nand flash memory: Characterization, analysis, and modeling," in *Design, Automation Test in Europe Conference Exhibition (DATE), 2013*, 2013, pp. 1285–1290.

[15] D. hwan Lee and W. Sung, "Estimation of nand flash memory threshold voltage distribution for optimum soft-decision error correction," *Signal Processing, IEEE Transactions on*, vol. 61, no. 2, pp. 440–449, 2013.

[16] N. Papandreou, T. Antonakopoulos, U. Egger, A. Palli, H. Pozidis, and E. Eleftheriou, "A versatile platform for characterization of solid-state memory channels," in *Digital Signal Processing (DSP), 2013 18th International Conference on*, 2013, pp. 1–5.

[17] G. Atwood, A. Fazio, D. Mills, and B. Reaves, "Intel strataflash memory technology overview," *Intel Technology Journal*, 1997.

[18] S. Li and T. Zhang, "Improving multi-level nand flash memory storage reliability using concatenated bch-tcm coding," *Very Large Scale Integration (VLSI) Systems, IEEE Transactions on*, vol. 18, no. 10, pp. 1412–1420, 2010.

[19] Y. Maeda and H. Kaneko, "Error control coding for multilevel cell flash memories using nonbinary low-density parity-check codes," in *Defect and Fault Tolerance in VLSI Systems, 2009. DFT '09. 24th IEEE International Symposium on*, 2009, pp. 367–375.

[20] G. E. P. Box and M. E. Muller, "A note on the generation of random normal deviates," *Ann. Math. Statist.*, vol. 29, no. 2, pp. 610–611, 1958.