Non-Volatile Memory Emulator

Antonios Prodromakis and Theodore Antonakopoulos

e-mails: aprodromakis@upatras.gr and antonako@upatras.gr

Department of Electrical and Computer Engineering, University of Patras, Patras 26504, Greece

Abstract

This work presents a versatile and flexible FPGA-based platform, capable to emulate various NVM technologies, either at the chip or channel level, along with the effect of aging on their reliability and on their read/write response times. The proposed NVM emulator (NVM-E) is based on a reconfigurable hardware-software architecture, which enables the accurate representation of various NVM technologies, focusing especially at the MLC NAND Flash case. The NVM-E 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. In this paper, we focus on the architecture of a NVM channel emulator and we present details about its internal functionality.

Keywords: Non-volatile memories, NAND Flash, memory aging, FPGA emulator.

1. Introduction

NAND Flash memories are used extensively for replacing magnetic hard disk drives and caches based on volatile memories in enterprise storage systems. The rapid scaling of NAND Flash memories and the use of multi-level cell (MLC) technologies have increased their storage density and reduced the storage cost per bit. However, different noise sources and interferences along with aging effects have reduced their expected lifetime. Numerous methods and techniques, such as error correcting codes (ECC), have been employed to compensate these effects, while others, more complex but also more efficient techniques are at an experimental stage. The development of these techniques is based on experimental characterization of non-volatile memory (NVM) cells and chips.

Memory characterization is performed by measuring bit error ratio (BER) and response times during the whole lifetime of a device, for various loading data patterns and timing scenarios. Since this process is performed using real NVM ICs, it has two major drawbacks. On one hand, it is a very time-consuming process, since it requires a large number of program/erase (P/E) cycles to be performed at each experiment. On the other hand, the aging characteristics of a NVM are proportional to the number of the applied P/E cycles, thus making it impossible to conduct different or successive experiments at the same aging state of a memory chip.

The aim of our work is to develop a flexible Field Programmable Gate Array (FPGA)-based platform, capable of emulating different NVM technologies, either at a chip or at a channel level, along with the effect of aging on their reliability (BER) and on their read/write (R/W) response times. The presented NVM emulator is based on a reconfigurable hardware-software architecture, which enables the accurate representation of various NVM technologies, focusing especially at the MLC NAND Flash case. The proposed architecture is capable of emulating any user-specific aging state, thus eliminating the need of cycling on a real memory device. Furthermore, it 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.

The NVM-E 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. In this paper, we focus on the architecture of NVM channel emulation and we present details about its functionality.

2. Previous work

Storing data to an MLC NAND Flash is achieved by accurately programming its cells into various intermediate voltages. More specifically, for an *n*-bit/cell NAND Flash, each cell can be programmed in up to 2^n different voltage 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 becomes shorter, therefore MLC memories are more vulnerable to noise sources than single-level cell (SLC) ones. The threshold voltage distributions are affected by different noise sources, such as cell-wearing, while cell to cell interference (CCI) plays a major role as technology scales. It has been shown in [2] and [3] that each NAND Flash cell can be modelled as a level dependent - additive white noise channel (LD-AWGN).

Cycling a memory cell alters the parameters of the Gaussian probability density function. The mean noise levels are usually shifted to higher values and the level distributions become wider. Consequently, distributions of adjacent symbols may overlap further and an erroneous read of the stored information is more likely to happen, therefore leading to higher raw BER.

In [3], we presented a hardware circuit that emulates the BER of a single NAND Flash cell as a function of the aging process, more specifically, its noise characteristics. The accuracy of the proposed circuit was validated with experimental results and comparisons with theoretical values. A model based on asymmetric Pulse Amplitude

Modulation (aPAM) with level-dependent additive noise was introduced in [4]. The proposed model can be used for all MLC NAND Flash technologies presented in the literature and moreover, it can be used as the basis to model different cell technologies.

In [5], a methodology was introduced in order to determine the relationship between cycling and noise characteristics of the cells. The proposed methodology offers the capability of emulating the BER characteristics of different NVM technologies with high precision, by observing their aging behaviour, without any knowledge of their internal architecture or the electrical characteristics of their cells. These results indicate that the NVM-E can accurately represent the bit error characteristics of a real memory device during its whole lifetime, by associating the target device's aging conditions with emulator's internal parameters.

3. Emulating NVMs as a Time-Variant Communications Channel

The digital logic used for emulating a NVM cell is summarized in the block diagrams of Fig. 1. More specifically, Fig. 1a demonstrates the process - which has been analytically described in [5] - of associating the aging state of the memory with the P/E cycles. Fig. 1b highlights the emulation of a NVM cell as a time-variant communications channel. When the user specifies the aging of the emulated device, the noise parameters for this aging state are stored in the Noise Mapping module. Then, depending on the input symbol *s*, the aging parameters, along with the ideal (error-free) voltage level of *s*, are provided to the LD-AWGN module. Fig. 1c illustrates the block diagram of the LD-AWGN module, which generates a soft read-back signal, *S*, representing the actual voltage, sensed from the memory when the symbol *s* is read at the specified aging conditions.

Finally, the Hard Decision module implements the decoding of the read-back signal to the *n*-bit symbol *s'*. The hard-decision is based on the provided read reference thresholds. The proposed architecture can support emulation of cells with different storage capabilities (SLC, MLC, TLC, QLC), by adjusting 'on-the-fly' the value of *k* that will be used. The implementation of the LD-AWGN module is based on the architecture described in [3]. The LD-AWGN module operates at 200 MHz, providing a read-back signal *S* per clock, that is, a processing rate of 50MBps for MLC.

4. Emulating an ONFI3.2 NV-DDR2 MLC NAND Flash

An architecture of the NVM-E, designed to interface with an embedded microprocessor, has been proposed in [5]. The proposed architecture is based on the AMBA AXI4 specs for transferring data and commands between a microprocessor and the NVM-E core. The main advantage of this approach is that it enables the execution of BER experiments of different memory technologies from the same hardware system, even if they differ on their I/O interfaces. Furthermore, since the microprocessor and the NVM-E are embedded onto the same FPGA, high transfer rates can be reached, even higher than those of the real memory device. Finally, the processing performance of the system can be increased dramatically by implementing more LD-AWGN modules in parallel.

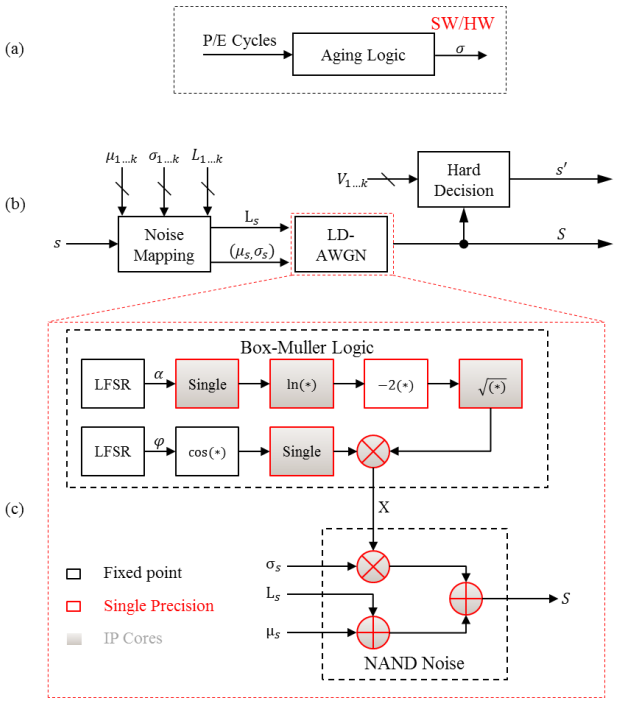

Fig. 1 Emulation logic block diagrams: a. Aging Logic, b. Communications Channel, c. LD-AWGN

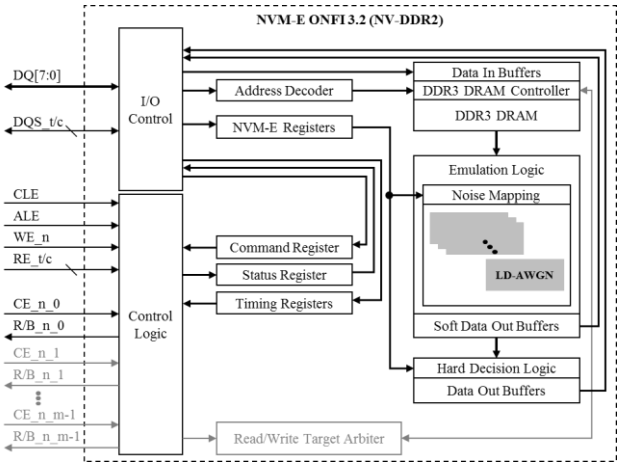

Fig. 2 Architecture of the ONFI3.2 NV-DDR2 NVM-E

However, the aforementioned architecture does not cover the case where the I/O interface and the effect of aging on the R/W performance has to be emulated. Fig. 2 highlights an alternative architecture of the NVM-E core which is compatible with the ONFI3.2 NV-DDR2 interface. The advantage of this configuration is that the real memory chip can be replaced by the NVM-E. Hence, memory-related signal processing algorithms can be applied directly to the NVM-E in the same manner as if the real memory device were used.

Apart from the basic NAND Flash commands of ONFI3.2 specification, e.g. page program, page read and block erase, all the additional operations of the NVM-E are based on the same functionality. For instance, the access of the additional NVM-E registers is performed by applying Set/Get features commands. The NVM-E registers of Fig. 2 refer to the registers and internal block RAM used to store the parameters of the specified aging state, while the timing registers are used to store the parameters of R/W latency for this state.

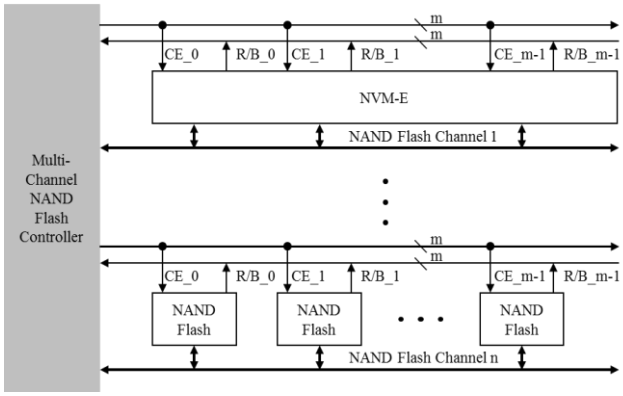

Fig. 3 Emulation of a NVM channel

The proposed design has been implemented using a 16GB DDR3 DRAM. However, depending on the available glue logic and the capacity of the installed DRAM on the FPGA board, the whole memory space of a NAND Flash device can be emulated. This functionality is implemented by the address decoder, which associates the column and row addresses of the NAND Flash memory to the flat address space of the DRAM. Furthermore, the address space of the DRAM can be partitioned to serve multiple NAND Flash targets simultaneously, enabling the emulation of more complex configurations.

For instance, a NAND Flash channel with multiple memory devices, which share the same data lines on a pipeline fashion, can be emulated by the same FPGA platform. In this case, a set of data buffers are implemented for each emulation target, while the Read/Write Target Arbiter schedules the R/W accesses on the DRAM. Fig. 3 illustrates an architecture, where a multi-channel NAND Flash controller is used to interface with *n* channels, where each channel contains *m* chips. As shown in Fig. 3, the NVM-E has replaced a whole NAND Flash channel and hence, experiments can be performed either using the NVM-E or the real NAND Flash devices.

In a typical NAND Flash device, the write response time t_{prog} is usually an order of magnitude larger than the read latency t_{read}. Therefore, the number of targets which share a NVM channel is determined by the t_{prog}, the page size (PS) and the data transfer time (t_{DT}). However, due to other limitations, such as the parasitic capacitances of the data lines, a typical NAND Flash channel consists of 4 NAND Flash targets. The NVM-E has been implemented to support up to 8 emulation targets per channel. Fig. 4 depicts the program operation times when 4 stages of pipeline are utilized.

The most common NAND Flash I/O interfaces are based on the ONFI and Toggle specifications. The data transfer rate (DTR) ranges from 40 MBps in ONFI1.0 up to 400 MBps in ONFI3.2. Table 1 illustrates the I/O characteristics of eight NAND Flash devices, with different I/O interfaces, cell technologies, R/W response times and page sizes. The maximum read pipeline depth (RPD) and the maximum program pipeline depth (PPD), which can be utilized under optimum loading conditions, have been estimated using equations (1) and (2).

$$RPD = \left\lfloor 1 + \frac{t_{read}}{t_{DT}} \right\rfloor \quad (1)$$

$$PPD = \left\lfloor 1 + \frac{t_{prog}}{t_{DT}} \right\rfloor \quad (2)$$

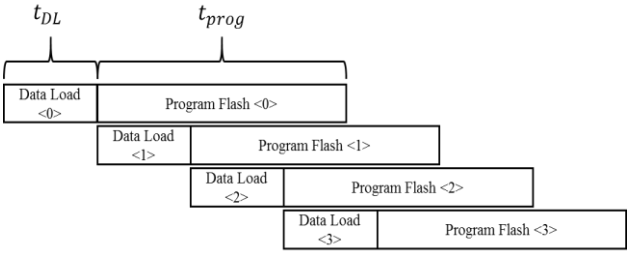

Fig. 4 Page program operations in a NAND Flash channel with 4 stages of pipelining

	ONFI	Cell	t_{read} (us)	t_{prog} (us)	PS (B)	DTR (MBps)	t_{DT} (us)	mRPD	mPPD
1	ONFI 1.0	SLC	60	800	2112	40	52.8	2	16
2	ONFI 1.0	SLC	60	800	4224	40	105.6	2	9
3	ONFI 2.0	MLC	50	900	4320	166	26.0	3	36
4	ONFI 2.0	MLC	25	200	4320	166	26.0	2	9
5	ONFI 2.0	MLC	50	1300	8640	166	52.0	2	26
6	ONFI 2.0	TLC	90	2400	9640	166	58.1	3	42
7	ONFI 2.2	MLC	35	300	8640	200	43.2	2	8
8	ONFI 3.0	MLC	50	1400	16384	400	41.0	2	35

Tab. 1. I/O characteristics of different NAND Flash technologies

	SRTR1 (MBps)	SRTR4 (MBps)	SRTR8 (MBps)	SPTR1 (MBps)	SPTR4 (MBps)	SPTR8 (MBps)
1	18.7	40.0	40.0	2.5	9.9	19.8
2	25.5	40.0	40.0	4.7	18.7	37.3
3	56.8	166.0	166.0	4.7	18.7	37.3
4	84.7	166.0	166.0	19.1	76.5	152.9
5	84.7	166.0	166.0	6.4	25.6	51.1
6	65.1	166.0	166.0	3.9	15.7	31.4
7	110.5	200.0	200.0	25.2	100.7	200.0
8	180.1	400.0	400.0	11.4	45.5	91.0

Tab. 2. Sustainable read and program transfer rates without pipelining, with 4 stages of pipelining and with 8 stages of pipelining, respectively.

From equations (3) and (4) we calculate the sustained read transfer rate (SRTR) and the sustained program transfer rate (SPTR), when the channel is configured without pipelining, with 4 stages of pipelining (typical NVM channel) and with 8 stages of pipelining (NVM-E channel), respectively. The SRTR and SPTR results for each memory device of Table 1 and for each channel configuration are illustrated in Table 2.

$$SRTR = \min\left(\frac{RPD \cdot PS}{t_{DT} + t_{read}}, DTR\right) \quad (3)$$

$$SPTR = \min\left(\frac{PPD \cdot PS}{t_{DT} + t_{prog}}, DTR\right) \quad (4)$$

5. Emulating complex NVM-based systems

A more complex configuration is presented in Fig. 5. The proposed system communicates with the host machine via PCIe Gen. 3 interface with 8 lanes, providing a throughput of approximately 5 GBps.

The FPGA logic consists of the PCIe core, two NVM-E cores, AXI4 and AXI4-Lite interconnects, as well as two DDR3 DRAM controllers with embedded ECC logic. Each NVM-E core has access to a DDR3-1333 ECC RAM with 32GB capacity. The proposed architecture can be used as a basis to emulate the whole memory space of a PCIe based solid state drive (SSD).

Core	LUT	*FF*	*Memory LUT*	BRAM	*DSP*
AXI4_0	6209	8323	163	0	0
AXI4_C0	1117	1609	76	0	0
AXI4_C1	1115	1609	76	0	0
AXI4_Lite	2943	4029	301	0	0
DDR3 C.	28760	20928	4220	1	0
PCIe	17885	26515	4172	67	32
NVM-E 1	38499	65775	2129	98	192
NVM-E 2	38502	65775	2135	98	192
Total	135053	194649	13275	264	416
Available	432368	864736	173992	1470	3600
Utilization (%)	31	23	8	18	12

Tab. 3. Resources used and utilization factor.

The system has been implemented on a Xilinx's Virtex-7 xc7vx690tffg1157-2 FPGA device and the host machine is a Linux-based workstation. Tab. 3 shows in details the resources used for each of the main cores, as well as the total utilization factor of the implemented system. Each AXI4 interconnect has a data bus of 256 bits and operates at 250MHz, providing an internal transfer rate of 8GBps.

6. Conclusions

In this paper, we presented the architecture of the NVM-E, configured to emulate complex NVM-based systems. The proposed system can accurately represent the BER characteristics and R/W response times of different NVM technologies at a user-specified aging condition and it can interface with existing NVM controllers. The NVM-E supports emulation at the chip, channel and system level, providing a valuable tool for developing and evaluating memory related algorithms and techniques. Performance studies indicate that the NVM-E has a great potential in reducing the execution time of experiments related with SSD development and their performance evaluation.

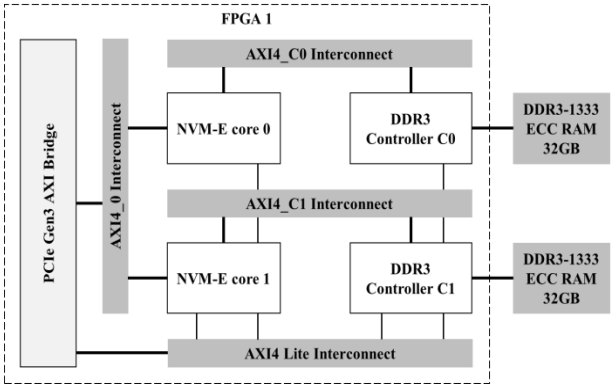

Fig. 5 Emulation of a complex NVM-based system

References

1. Bainan Chen, Xinmiao Zhang and Zhongfeng Wang, "Error correction for multi-level NAND flash memory using Reed-Solomon codes" IEEE Workshop on Signal Processing Systems, SiPS 2008, pp.94,99, 8-10 Oct. 2008 doi: 10.1109/SIPS.2008.4671744.
2. Yu Cai, Erich F. Haratsch, Onur Mutlu and Ken Mai, "Threshold voltage distribution in MLC NAND flash memory: Characterization, analysis, and modeling," Design, Automation & Test in Europe Conference & Exhibition, 2013 , vol., no., pp.1285,1290, 18-22 March 2013 doi: 10.7873/DATE.2013.266.
3. Dong-hwan Lee and Wonyong Sung "Estimation of NAND Flash Memory Threshold Voltage Distribution for Optimum Soft-Decision Error Correction," IEEE Transactions on Signal Processing, vol.61, no.2, pp.440,449, Jan.15, 2013 doi: 10.1109/TSP.2012.2222399 .
4. A. Prodromakis, G. Sklias and T. Antonakopoulos, "Emulating the aging of NAND Flash memories as a time-variant communications channel," Communications, 2014 6th International Symposium on Control and Signal Processing (ISCCSP), vol., no., pp.278,281, 21-23 May 2014 doi: 10.1109/ISCCSP.2014.6877868.
5. S. Korkotsides,G. Bikas, E. Eftaxiadis and T. Antonakopoulos, "BER analysis of MLC NAND Flash memories based on an asymmetric PAM model," 2014 6th International Symposium on Communications, Control and Signal Processing (ISCCSP), , vol., no., pp.558,561, 21-23 May 2014 doi: 10.1109/ISCCSP.2014.6877936.
6. A. Prodromakis, S. Korkotsides and T. Antonakopoulos, "A Versatile Emulator for the Aging Effect of Non-volatile Memories: The Case of NAND Flash," 2014 17th Euromicro Conference on Digital System Design (DSD), vol., no., pp.9,15, 27-29 Aug. 2014 doi: 10.1109/DSD.2014.56.