Reliable Architecture for Flash Memory

Amit Berman and Uri C. Weiser
Department of Electrical Engineering, Technion – Israel Institute of Technology
Technion City, Haifa 32000, Israel
{bermanam@tx,uri.weiser@ee}.technion.ac.il

Abstract— The mounting demand for high density non-volatile flash memories consequences new challenges of flash memory reliability issues [1,2]. Common reliability-improving techniques approaches are addressing device and process manufacturing improvements [3] or error correction codes (ECC) techniques [4]. This paper proposes a novel memory architecture and method of increasing reliability through reducing the program-erase (P/E) cycles. Reduction of P/E cycles is achieved through a memory architecture which enables rewrite operation in a single flash memory cell, instead erase-and-program which is currently applied. Furthermore, a potential improvement of the write operation time is also achieved when using sufficiently large write buffer, regarding modern write techniques [5,6].

Keywords: Flash Memory, Reliability, Architecture, Write Performance.

I. INTRODUCTION

Flash memory is a solid state non-volatile memory which is based on electron storing mechanism. The information is stored in a varying voltage threshold transistor, similar to MOSFET transistor with an extra layer to store the electrons (which make the threshold voltage variable) and some more extra layers to form a stable memory device, i.e. prevent electrons leakage, and to minimize physical disturbs. Due to its properties, flash memory has become one of the fastest growing segments in the global semiconductor industry. It is developing rapidly with annual sharp decrease of its prices. The largest demand among the types of flash memory is the demand for storage applications.

Technology scaling and the demand for high density products results in extensive effort to develop multi-level memory cells (MLC), i.e., storing more bits per memory cell. Such a device results in a major reliability challenges. Flash memory reliability defined as the major product challenge for current lithography's. Reliability improvement efforts are mainly made through manufacturing process and by implementing error correction codes (such as Reed-Solomon codes or BCH codes) to recover from bit errors.

Flash memory data array management is not done on the memory chip level but on the system level, commonly with external controller who decides where to write the data and the time and priority for writing, as well as command sequence and data management. Potential improvement of flash reliability and write time can be achieved if data management mechanism would consider the past data programmed pattern versus the current data pattern transitions and will adjust the memory array to the new pattern instead of erasing it and re-programming it with the new pattern. In such case, flash memory chip architecture would need to support the controller algorithms.

Flash memory structure implies that erase operation is necessary for a program of a new data pattern. Erase operation is relatively slow and therefore, in current architectures it is applied to large blocks (such as 64KB). The current architecture is likely to perform many unnecessary program and erase transitions. For example, assuming binary symmetric source, we get that in average, 50% of the data pattern is similar to the past one. In this case there is an obvious redundant latency and power in programming the memory cell, erasing it, and then programming it again to its original location, as shown in figure 1.

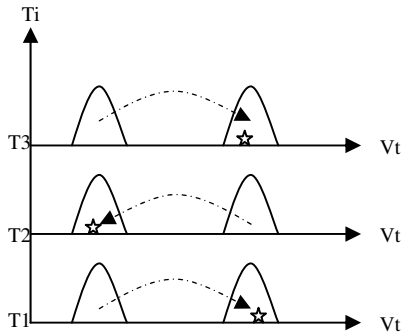

Figure 1: unnecessary erase operation

We propose flash memory architecture to enhance memory cell reliability. The architecture consists of virtual erase and rewrite mechanisms. Conventional erase operation turn to virtual erase, in which erase is applied to an alternate valid-mark memory cells array and not to the actual memory cells it was applied in the information array. Rewrite mechanism is applied for adjustment of the difference between past and current data patterns. The rewrite method consists of adaptive erase pulses for the memory cells with erase transitions and soft program pulses for the memory cells with program transitions.

II. RELIABLE ARCHITECTURE

The proposed architecture consists of virtual erase and re-write mechanisms. The main concept is that each program of data pattern will be based of transitions from the past data pattern (which were already programmed to the flash). The conventional erase operation is turn to "virtual erase" as will be described in the next subsection.

Virtual Erase

When the user will apply erase operation to flash memory, the operation is not performed on the actual array cells it was assigned, but to a non-volatile spare array that will indicate the valid sign of each non-volatile page (as described on figure 2). While programming a new program pattern, the new program method (consists of soft erase pulses and soft program pulses will adjust the current sector to current data pattern).

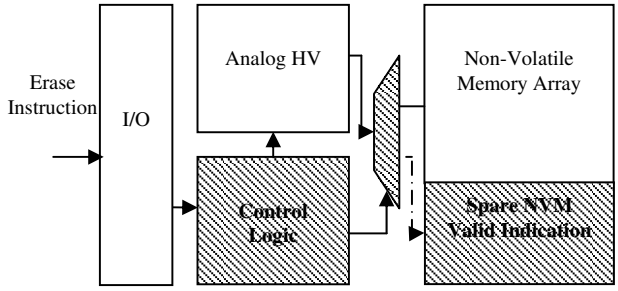

Figure 2: virtual erase
additional control logic and spare NVM is marked with dashed lines

Rewrite Mechanism

In our analysis, we will assume binary symmetric data source. We will use the following definitions in statistical description of the difference between the data patterns:

Let D_{IN1} data pattern of size n, to be programmed to flash memory array at time T_1.

Let D_{IN2} next data pattern of size n, to be programmed at time T_2 where $T_1 < T_2$

$D_{IN1} = (x_1, x_2, ..., x_n)$

$D_{IN2} = (y_1, y_2, ..., y_n)$

$x_i, y_i \in F, 1 \le i \le n, F = \{0,1\}$

Let assume binary symmetric data source:

$$\Pr\{x_i = 0\} = \Pr\{x_i = 1\} = \Pr\{y_i = 0\} = \Pr\{y_i = 1\} = \frac{1}{2}$$

Let $M(D_{IN1}, D_{IN2})$ be the measurement of difference between data patterns D_{IN1}, D_{IN2}

$$M(D_{IN1}, D_{IN2}) = D_{IN1} \oplus D_{IN2} = \sum_{i=1}^{n} x_i \oplus y_i$$

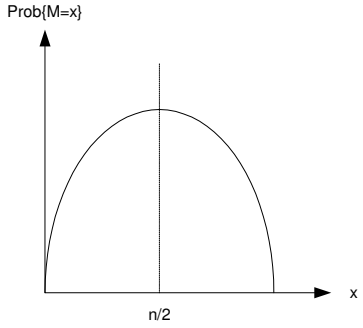

Figure 3: Probability of difference between data patterns assuming binary symmetric data source (maximum point is at half size of the data pattern)

When program command applied, we will get the data pattern D_{IN2} from the user. The next algorithm step is to read the array pattern D_{IN1} that was programmed in that address. Given data patterns D_{IN1}, D_{IN2} we need to adjust memory array from D_{IN1} to D_{IN2}.

(We have a specific difference $M(D_{IN1}, D_{IN2})$ that we would like to minimize it: perform a transition scrambling function that minimizes $M(D_{IN1}, D_{IN2})$, or if minimization is not sufficient, group similar erased cells to squares, to accelerate flash erase. The transition scrambling function is a different optimization subject and will be discussed later).

The first adjustment step is to perform erase pulses for the cells that need erase transition (i.e. lower threshold voltage). Statistically, it is dependent on the number of flash memory levels:

Let n be the data pattern size [measured in symbols],

$|D_{IN1}| = |D_{IN2}| = n$

Each symbol of the data pattern consists of m bits. Each flash cell can store a symbol.

Let $l = 2^m$ be the number of distinguished threshold levels on flash memory technology, then:

$|F| = 2^m = l$

For example, in the case of flash cell stores 2 bits of information per cell:

$|F| = 2^2 = 4$

$F = \{00, 01, 10, 11\}$

(Therefore, if we would like to store l bits per cell, we would need 2^l distinguished threshold levels. Increasing the threshold levels would increase the overlap errors probability and would cause the data to be unreliable).

Let define the number of cells which have no transition (i.e. same data at the past data pattern and the current data pattern) to be $NT(l(m),n)$. Those cells will be left as they are in the new architecture. Assuming binary symmetric source, the number of cells without transition is distributed with Gaussian distribution with the mean of:

$$NT(l,n)=\frac{l}{2^l}\cdot n$$

For example, in flash memory with 4 distinguished levels:

$$NT(l=4,n)=\frac{1}{4}\cdot n$$

(Hence, in average, 25% of the cells need to stay in the same level).

Let define the number of cells with program transition (i.e. the data at the past data pattern is in lower threshold level then the current data pattern) to be $PT(l(m),n)$, then the number of cells with program transition is a Gaussian distribution with mean of:

$$PT(l,n)=\left(\frac{1-\frac{l}{2^l}}{2}\right)\cdot n=\left(\frac{1}{2}-\frac{l}{2^{l+1}}\right)\cdot n=\left(\frac{2^l-l}{2^{l+1}}\right)\cdot n$$

For example, in flash memory with 4 distinguished levels:

$$PT(l=4,n)=\left(\frac{2^4-4}{2^5}\right)\cdot n=0.375\cdot n$$

(Hence, in average, 37.5% of the cells need a program transition).

Let define the number of cells with erase transition (i.e. the data at the past data pattern is in higher threshold level then the current data pattern) to be $ET(l(m),n)$ be, then, in the same way:

$$ET(l,n)=PT(l,n)$$

Erase transition (ET) and program transition (PT) can be classified according to the transition distribution distance. Let c be the specific charge level we want to find its number of cells that need program transition. Concluding symmetric calculations, the number of cells with program transition with given charge level c, are distributed in Gaussian distribution with mean of:

$$PTL(l,n,c)=\left(\frac{\frac{2^l-l}{2^{l+1}}}{\left(\sum_{i=1}^{l}i\right)-l}\cdot(l-c)\right)\cdot n=\left(\frac{\left(2^l-l\right)\cdot(l-c)}{2^{l+1}\cdot\left(\left(\sum_{i=1}^{l}i\right)-l\right)}\right)\cdot n$$

For example, in flash memory with l=4 distinguished levels, the number of program transitions from level number c=2,

$$PTL(l=4,n,c=2)=\left(\frac{\left(2^4-4\right)\cdot(4-2)}{2^5\cdot\left((1+2+3+4)-4\right)}\right)\cdot n=0.125\cdot n$$

(Hence, in average, 12.5% of the total cells need a program transition from level 2 to a higher level).

In the same way, let $ETL(l,n,c)$ be the number of cells with erase transition (i.e. the data at the past data pattern is in higher threshold level then the current data pattern), from a specific level c, then:

$$ETL(l,n,c)=PTL(l,n,c)$$

Memory write and erase operations flow are described on figure 4.

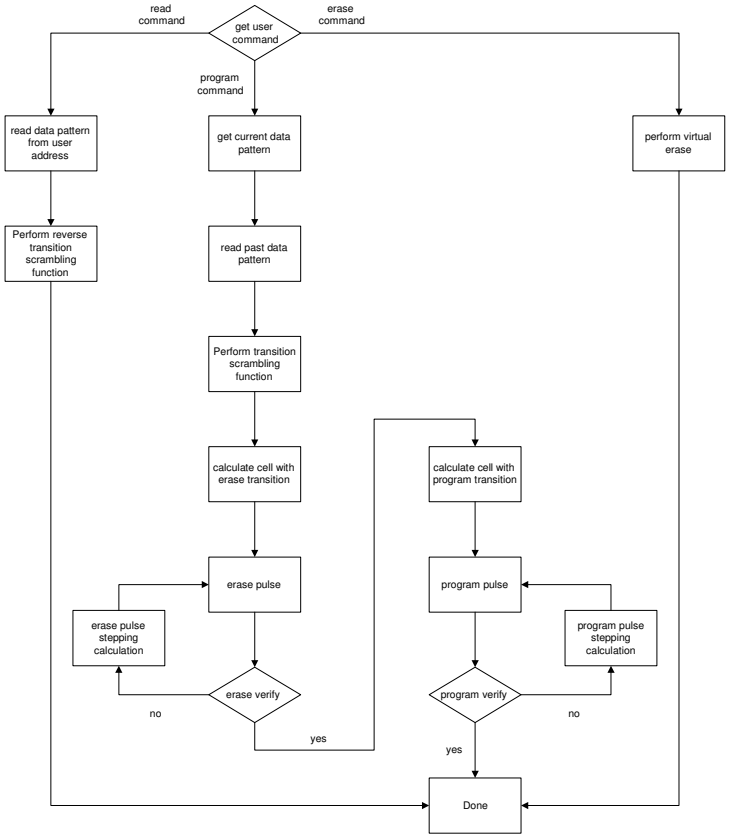

Figure 4: Write and Erase Operations Flow

III. RESULTS

The proposed architecture was modeled using MATLAB software and tested assuming that program pulse time is 300 nano-seconds and erase pulse time is 200 micro-seconds. For the conventional architecture reference, the average number program pulses needed is 15 and the average number of erase pulses is 5. Industry-standard performance had been aligned, providing ~1mili-second for page program of ~4Kbytes. Erase operation is expected to be longer than total erase pulses time due to the fact that common erase algorithms uses pre-program and soft post-erase pulses to handle unified pattern set and to avoid over-erase disturbs. Program operation is always done on a page basis which is a single word-line. Therefore, erase operations (voltages via the bit-lines) can be applied to a single flash memory cell.

We can now estimate the memory reliability improvement since it does depend only on the property of binary symmetric source and the number of distinguishable levels in the flash memory. Reliability is measured by P/E (program-erase) cycles, since $ET(l,n)$ is the number of cells that are to be on erase transition, we will measure the new P/E cycling by it.

Let define Reliability Improvement Factor to be $RIF(l)$, then:

$$RIF(l) = \frac{n}{ET(l,n)} = \left(\frac{2^{l+1}}{2^l - l} \right)$$

For example, in flash memory with 2 distinguished levels (1 bit per cell) results reliability improvement by x4:

$$RIF(l=2) = 4$$

Flash memory with 4 distinguished levels (2 bit per cell) resulting that the improvement is x2.66

$$RIF(l=4) = 2.66$$

Figure 5 shows the dependent of RIF in the number of levels of the flash memory.

Note that the minimum improvement is x2.

$$RIF(l) \xrightarrow{l \to \infty} 2$$

The implication on program time performance is described on figure 6.

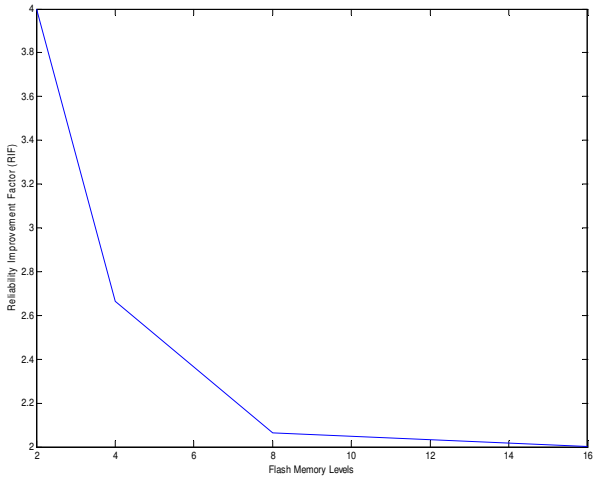

Figure 5: Reliability Improvement Factor (RIF) vs. Flash Memory Levels

IV. Discussion

The implications of the new architecture and algorithm on program time depend on the free variables: erase and program pulses time and distribution properties, memory page size. Simulation checks the implication of the new algorithm on memory write time comparing to the regular algorithm.

Flash memory program pulses consumes relatively large current, so program pulse is commonly limited to a small number of memory cell, therefore memory write speed is growing in a linear factor with memory sector size.

Erase pulses consumes much less current then program pulses, commonly be a factor of 1000+ less current [8], hence we expect the algorithm to be more efficient on large sector sizes. Program pulses are regularly shorter then erase pulses by a factor of ~50-1000, depend on the manufacturing process. As erase pulse is shorter, the algorithm can improve results even on a small page sizes and increase the potential improvement of the memory program time.

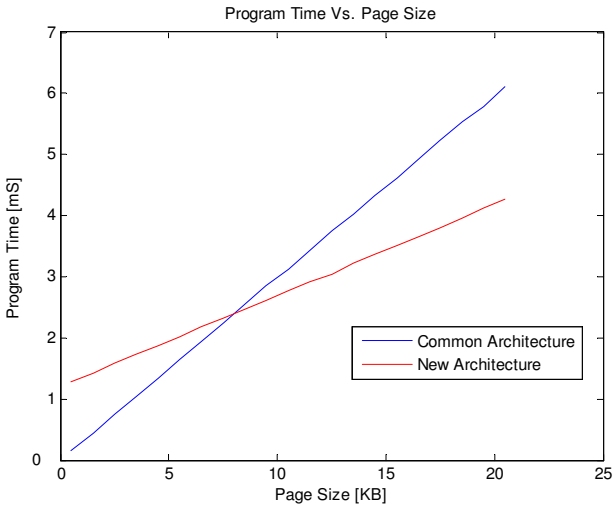

Figure 6: Program Time Performance vs. Page Size

From the simulation results we notice a page size~8KB when the program time penalty becomes a program time gain. This is happening in a relatively high page size, which are regularly massive storage devices like storage application and solid-state-disks (SSD).

The optimization point at 8KB can be moved to a smaller page sizes if process enables to make the erase pulses shorter, with proper manufacturing process changes on device physics.

Given a specific product properties we can decide upon the trade-off of the new reliability architecture implementation:

Area: To implement the new architecture, there is a requirement of 1 spare NVM bit to each page. For example if page size is 4KB, 1 bit is 0.003% spare area and it becomes smaller as the page size grows. Logic and analog blocks design to support the following architecture can be estimated to be 2%-5% of total die size.

Performance: Program time is dependent on the page size, as described on the simulation graph in the previous section. Below 8KB there is a program time penalty which grows as the page size becomes smaller. Above 8KB there

is a program time gain which grows as the page size becomes bigger.

Erase time is expected to be almost eliminated, as we have to erase now 1 bit in comparison to 32 pages in regular erase. Read time is with no change.

Reliability: Gain of x4-x2, depends on the number of distinguishable flash analog levels. It decreases to x2 with the increasing of the Flash Analog Levels.

Yield: Removal of erase operation minimizes the negative effects of over erase phenomena. Yield is expected to grow with dependence in the manufacturing process.

The New Architecture is more effective on mass storage devices (large page sizes). It can be effective on a small memory sizes if process enables to make the erase pulses shorter, those moving the optimization point on figure 6, to a smaller page sizes. In this case, small memory sizes are commonly SLC and will gain more reliability improvement of ~x4.

Future Development

Transition Scrambling Function

Let transition scrambling function

$$f : D_{IN2} \mapsto D_{SB2}$$

$$f(D_{IN2}) = \min M(D_{IN1}, D_{IN2})$$

Transition scrambling function objective is to minimize the transitions between different data patterns. One possible implementation is a time-varying vector which does XOR operation on different sizes of symbols from the data pattern and measures the minimum transitions.

Process Research

Process parameters such as erase pulse width and height may improve the architecture if it were optimized. Future development can focus on making the architecture suitable for small page sizes by optimizing the erase operation.

IV. CONCLUSIONS

Flash memory reliability is a leading issue for future memory design with high densities and smaller lithographs. A reliability improvement and potential write performance improvement opportunities can be achieved if erase pulse can be applied to each memory cell, combined with a data management mechanism to minimize data patterns transitions on the memory cell. A reliable architecture for flash memories was presented utilizing data-management control logic, where erase command is turn to virtual erase hence erasing only a valid bit indication of the addressed page. A rewrite algorithm is applied which combines soft erase pulses for memory cells with erase transitions and soft program pulses for memory cells with program transition. P/E cycles reliability issue is shown to be improved by more then x2 with 2-5% area penalty overhead. A potential program speed can we achieved while using large write buffer.

ACKNOLEDEMENT

The authors would like to thank Idit Keidar for her useful comments and inputs.

REFERENCES

[1] B. Eitan, R. Kazerounian, A. Roy, "Multilevel Flash cells and their Trade-offs", IEEE IEDM, pp.169-172, 1996.

[2] C. Calligaro, A. Manstretta, A. Modelli, G. Torelli, "Technological and Design constraints for multilevel Flash Memories", IEEE ICECS, pp.1006-1008, 1996.

[3] A. Modelli, A. Visconti, R. Bez, "Advanced Flash Memory Reliability", IEEE ICDT, pp. 211-218, 2004.

[4] Y. Cassuto, M. Schwartz, V. Bohossian, J. Bruck, "Codes for Multi-Level Flash Memories: Correcting Asymmetric Limited-Magnitude Errors", IEEE ISIT, pp. 1176-1180, June 2007.

[5] R. Sahar, A. Lavan, A. Berman, B. Eitan et al., "A 4b/Cell 8Gb NROM Data-Storage Memory with Enhanced Write Performance", IEEE ISSCC Dig. Tech. Papers, pp. 6-9, Feb. 2008.

[6] T. Kobayashi, Y. Sasago, H. Kurata, S. Saeki, Y. Goto, T. Arigane, Y. Okuyama, H.Kume, K. Kimura, "A Giga-Scale Assist-Gate (AG)-AND-Type Flash Memory Cell with 20-MB/s Programming Throughput for Content-Downloading Applications", IEEE IEDM, pp.(01)29-(01)-32, 2001.

[7] P. Pavan, R. Bez, P Olivo, E. Zanoni, "Flash Memory Cells—An Overview", Proceeding of the IEEE, Vol. 85, No. 8, Aug. 1997.

[8] P. Cappelletii, C. Golla, P. Olivo, E. Zanoni, "Flash Meomories", Kluwer Academic Publishers-KAP, 2000.

[9] R. Schuetz, H. Oh, J. Kim, H. Pyeon, S. Przybylski, P. Gillingham, "HyperLink NAND Flash Architecture for Mass Storage Applications", Non-Volatile Semiconductor Memory 22nd Workshop, IEEE, Aug. 2007.

[10] Web-Feet Research, Electronic Talks, 2005 http://www.electronicstalk.com/news/wbf/wbf101.html

[11] D. Ielmini, A. Spinelli, A. Lacaita1, "Recent developments on Flash memory reliability", Proceedings of the 14th biennial conference on insulating films on semiconductors 2005, pp. 321-328.

[12] R. Verma, "Flash Memory Quality and Reliability Issues", Workshop on Memory Technology, Design and Testing, 1996. pp. 32-36.

[13] S. Aritome, R. Shirota, G. Hemink, T. Endoh, F. Masuoka, "Reliability issues of flash memory cells", Proceedings of the IEEE, May 1993, Vol 81, pp.: 776-788.

[14] K. Adkins, R. Frizzell, J. Lyle, "The NM29A040 / NM29A080 Serial Flash Memory Architecture and its Application", 1996 Int. NonVolatile Memory

Technology Conference, pp.76-79.
[15] K. Cheng, J. Yeh, C. Wang, C. Huang, C. Wu, "RAMSES-FT: a fault simulator for flash memory testing and diagnostics", VLSI Test Symposium, 2002, pp. 281- 286.
[16] P.Cappelletti, R.Bez, D.Cantarelli, L.Fratin, "Failure Mechanisms of Flash Cell in Program/Erase Cycling", International Electron Devices Meeting, 1994. IEDM pp. 291-294.
[17] C. Park, J. Seo, S. Kim, "Cost-Efficient Memory Architecture Design of NAND Flash Memory Embedded Systems", 21st International Conference on Computer Design, 2003., pp. 474-480.
[18] L. Chang, T. Kuo, "An Adaptive Striping Architecture for Flash Memory for Storage Systems of Embedded Systems", Real-Time and Embedded Technology Applications Symposium, 2002 pp. 187-196.
[19] A. Roy *et al*, "New Flash Architecture with A 5.83L2 Scalable AMG flash cell", Symposium on VLSI Technology 1997, pp. 67-68.