EURASIP Journal on Advances in Signal Processing
a SpringerOpen Journal

RESEARCH **Open Access**

# Reducing latency overhead caused by using LDPC codes in NAND flash memory

Wenzhe Zhao[1*], Guiqiang Dong[2], Hongbin Sun[1], Nanning Zheng[1] and Tong Zhang[3]

**Abstract**

Semiconductor technology scaling makes NAND flash memory subject to continuous raw storage reliability degradation, leading to the demand for more and more powerful error correction codes. This inevitable trend makes conventional BCH code increasingly inadequate, and iterative coding solutions such as low-density parity-check (LDPC) codes become very natural alternative options. However, fine-grained soft-decision memory sensing must be used in order to fully leverage the strong error correction capability of LDPC codes, which results in significant data access latency overhead. This article presents a simple design technique that can reduce such latency overhead. The key is to cohesively exploit the NAND flash memory wear-out dynamics and impact of LDPC code structure on decoding performance. Based upon detailed memory device modeling and ASIC design, we carried out simulations to demonstrate the potential effectiveness of this design method and evaluate the involved trade-offs.

**Keywords:** NAND flash memory, LDPC code, Hard-decision decoding

## Introduction

Solid-state storage systems based upon NAND flash memory technology must use error correction code (ECC) to ensure the system-level data storage integrity. In current design practice, BCH codes with classical hard-decision decoding algorithms [1] are being widely used. As the semiconductor industry continues to push the technology scaling envelope and pursue aggressive use of multi-level per cell storage, raw storage reliability of NAND flash memory continues to degrade, which quickly makes current design practice inadequate and hence naturally demands more powerful ECCs. Because of their well-proven error correction capability with reasonably low decoding complexity and recent success in hard disk drives, low-density parity-check (LDPC) codes [2,3] have attracted many attentions because of their applications in NAND flash memory. To maximize their error correction capability, LDPC codes demand NAND flash memory carry out finer-grained (i.e., soft-decision) sensing. As a result, straightforward use of LDPC codes tends to significantly degrade the overall data storage system read response latency. In particular, since NAND flash memory on-chip sensing latency is linearly proportional to the sensing quantization granularity, soft-decision memory sensing will largely increase on-chip memory sensing latency compared with current practice. Meanwhile, in the presence of soft-decision memory sensing, the flash-to-controller data transfer latency will accordingly increase. Since the read response latency is a very critical metric in many data storage systems, it is highly desirable to reduce the read latency overhead caused by the use of LDPC codes in NAND flash memory.

In this article, we present a simple design technique to reduce the latency overhead caused by the use of LDPC codes. First, we note that NAND flash memory cells gradually wear out with the program/erase (P/E) cycling [4], which is reflected as gradually diminishing memory cell storage noise margin (or increasing raw storage bit error rate). To meet a specified P/E cycling endurance limit, the LDPC code decoding with the maximum allowable soft-decision memory sensing precision should be able to tolerate the worst-case raw storage reliability at the end of memory lifetime. Clearly, the memory cell wear-out dynamics makes the maximally achievable error correction capability of LDPC codes largely *more-than-enough* over the entire lifetime of memory, especially at

*Correspondence: venturezhao@gmail.com
[1]The Institute of Artificial Intelligence and Robotics, Xi'an Jiaotong University, Shaanxi, China
Full list of author information is available at the end of the article

Springer
© 2012 Zhao et al.; licensee Springer. This is an Open Access article distributed under the terms of the Creative Commons Attribution License (http://creativecommons.org/licenses/by/2.0), which permits unrestricted use, distribution, and reproduction in any medium, provided the original work is properly cited.

its early lifetime when P/E cycling number is relatively small. Meanwhile, the error correction capability of LDPC codes strongly depends on the soft-decision input precision. Therefore, it is very straightforward to apply a progressive-precision LDPC decoding strategy to reduce the average latency, i.e., we always start with the hard-decision memory sensing (and hence hard-decision LDPC code decoding), and only if LDPC decoding fails, we progressively increase the sensing precision and retry the decoding until LDPC decoding succeeds.

Under such a straightforward progressive-precision decoding design framework, it is very critical to minimize the hard-decision LDPC decoding failure rate in order to minimize the overall latency overhead. Hard-decision LDPC decoding employs the bit-flipping decoding algorithm [5], which flips the hard-decision of those bits that participate in the most number of unsatisfied parity checks during each iteration. In contrast, soft-decision LDPC decoding employs either sum–product decoding algorithm [3], min–sum decoding algorithm [6], or their many variants. These soft-decision decoding algorithms iteratively update the likelihood probability estimation of each bit. In the context of soft-decision decoding, it has been well known that the decoding performance heavily depends on the column weight of LDPC code parity check matrices, and low-weight LDPC codes tend to have stronger error correction capability than high-weight LDPC codes. However, in the context of hard-decision decoding based upon bit-flipping decoding algorithm, high-weight LDPC codes tend to outperform their low-weight counterparts, which is completely opposite to the scenario of soft-decision decoding. Under the straightforward progressive-precision decoding framework, such conflicting impact of code parity check matrix column weight on error correction capability directly leads to a design dilemma: If we use low-weight LDPC codes to maximize the achievable error correction capability and hence maximize the tolerable worst-case raw storage reliability, the corresponding hard-decision decoding failure rate will relatively be higher, leading to a larger latency overhead. To address this design dilemma, instead of using the same LDPC code throughout the entire NAND flash memory lifetime, we propose to adaptively adjust the LDPC code parity check matrix column weight based upon the wear-out progress of the NAND flash memory. In particular, given the raw storage reliability of NAND flash memory, we always use the LDPC code that can meet the target soft-decision decoding failure rate and meanwhile has the highest column weight. As a result, we can always ensure that the hard-decision decoding failure rate is minimized throughout the entire lifetime of NAND flash memory, leading to the minimal overall latency overhead.

Based upon NAND flash memory erase and programming characteristics, we derive mathematical formulations to approximately model threshold voltage distribution of memory cells in the presence of various major NAND flash memory device noise and distortion sources. Using a hypothetical 2 bits/cell NAND flash memory and rate-8/9 LDPC codes with different column weights, we carry out extensive computer simulations to evaluate the effectiveness of the proposed design technique. In addition, we carried out LDPC decoder ASIC design at 65-nm technology node to evaluate and compare the hard-decision and soft-decision LDPC decoder silicon cost.

## Basics and background

### Basics of NAND flash memory physics

Each NAND flash memory cell is a floating gate transistor whose threshold voltage can be programmed by injecting certain amount of charges into the floating gate. Before a flash memory cell can be programmed, it must be erased (i.e., remove all the charges from the floating gate, which sets its threshold voltage to the lowest voltage window). For $n$ bits/cell flash memory, the goal of memory programming is to move the memory cell threshold voltage into one of $2^n$ non-overlapping storage levels that are apart from each other with certain noise margin. However, the memory cell storage noise margin can seriously be degraded in practice, mainly due to P/E cycling effects and cell-to-cell interference, which will be discussed below.

#### *Effects of P/E cycling*

Flash memory P/E cycling causes damage to the tunnel oxide of floating gate transistors in the form of charge traps in the oxide and interface states [7-9], which directly results in memory cell threshold voltage shift and fluctuation and hence degrades memory device noise margin. Let $N$ denote the number of P/E cycles that memory cells have gone through and $\Delta N_{\text{trap}}$ denote the density growth of either interface or oxide traps. We can approximately quantify the relation between interface/oxide traps generation and the number of P/E cycles as

$$\Delta N_{\text{trap}} = A \cdot N^{a}, \tag{1}$$

where $A$ is a constant factor fitted from measurements. Such a power–law relationship can be explained by the widely accepted reaction–diffusion model in negative bias temperature instability [10,11] and the scattering-induced diffusion model [12]. Those gradually accumulated traps result in two major types of noises:

1. Electrons capture and emission events at charge trap sites near the interface developed over P/E cycling directly result in memory cell threshold voltage random fluctuation, which is referred to as random telegraph noise (RTN) [13,14];

2. Interface state trap recovery and electron detrapping [12,15] gradually reduce memory cell threshold voltage, leading to the data retention limitation. This is referred to as data retention noise.

Since the significance of these noises grows with the trap density and trap density grows with P/E cycling, NAND flash memory cell noise margin monotonically degrades with P/E cycling. This leads to the NAND flash memory P/E cycling endurance limit, beyond which memory cell noise margin degradation can no longer be accommodated by the memory system fault tolerance capability.

#### *Cell-to-cell interference*

In NAND flash memory, the threshold voltage shift of one floating gate transistor can influence the threshold voltage of its neighboring floating gate transistors through parasitic capacitance-coupling effect [16]. This is referred to as cell-to-cell interference, which has been well recognized as the one of major noise sources in NAND flash memory [17-19]. Threshold voltage shift of a victim cell caused by cell-to-cell interference can be estimated as [16]

$$F = \sum_k (\Delta V_t^{(k)} \cdot \gamma^{(k)}), \tag{2}$$

where $\Delta V_t^{(k)}$ represents the threshold voltage shift of one interfering cell which is programmed after the victim cell, and the coupling ratio $\gamma^{(k)}$ is defined as

$$\gamma^{(k)} = \frac{C^{(k)}}{C_{\text{total}}}, \tag{3}$$

where $C^{(k)}$ is the parasitic capacitance between the interfering cell and the victim cell, and $C_{\text{total}}$ is the total capacitance of the victim cell.

### Use of soft-decision ECC in NAND flash memory

As technology continues to scale down, NAND flash memory cell storage distortion and noise sources become increasingly significant, leading to continuous degradation of memory raw storage reliability. As a result, the industry has very actively been pursuing the transition of ECC from conventional BCH codes to more powerful soft-decision iterative coding solutions, in particular LDPC codes. Nevertheless, since NAND flash memory sensing latency is linearly proportional to the number of sensing quantization levels and the sensing results must be transferred to the memory controller through standard chip-to-chip links, a straightforward use of soft-decision ECC in NAND flash memory can result in significant memory read latency overhead.

The linear dependency of memory sensing latency on the number of sensing quantization levels is caused by the underlying NAND flash memory structure. NAND flash memory cells are organized in an array→block→page hierarchy, where one NAND flash memory array is partitioned into blocks, and each block contains a number of pages. Within one block, each memory cell string typically contains 32 to 128 memory cells, and all the memory cells driven by the same word-line are programmed and sensed at the same time. All the memory cells within the same block must be erased at the same time. Data are programmed and fetched in the unit of page, where the page size ranges from 512 Bytes to 8 kBytes user data. As illustrated in Figure 1, all the memory cell blocks share the same set of bit-lines and an on-chip page buffer that contain sensing circuitry and hold the data being programmed or fetched. As illustrated in Figure 1, when we read one memory page with $m$-level sensing quantization, the word-line voltage consecutively sweeps through the $m$ different sensing quantization levels, and all the bit-lines are charged and discharged once for every sensing quantization level. This clearly shows the linear dependency of memory sensing latency on the number of sensing quantization levels.

## Reducing LDPC-induced latency overhead

### Progressive-precision LDPC decoding

From the discussions in "Basics of NAND flash memory physics" section, it is clear that NAND flash memory cell raw storage reliability gradually degrades with the P/E cycling: During the early lifetime of memory cells (i.e., the P/E cycling number $N$ is relatively small), the aggregated P/E cycling effects are relatively less significant, which leads to a relatively large memory cell storage noise margin and hence good raw storage reliability (i.e., low raw storage bit error rate); since the aggregated P/E cycling effects scale with $N$ in approximate power–law fashions, the memory cell storage noise margin and hence raw storage reliability gradually degrade as the P/E cycling number $N$ increases. As a result, it is sufficient for ECC to provide gradually stronger error correction capability throughout the entire NAND flash memory lifetime.

Meanwhile, the error correction capability of LDPC code decoding gradually improves as we increase the memory sensing precision. If NAND flash memory uses conventional hard-decision memory sensing (i.e., there are only $l-1$ sensing quantization levels for $l$-level per cell NAND flash memory), LDPC code decoder can only carry out hard-decision decoding (e.g., using the hard-decision bit-flipping decoding algorithm) and achieve relatively poor error correction capability. As NAND flash memory uses soft-decision memory sensing with higher and higher sensing quantization granularity, LDPC code

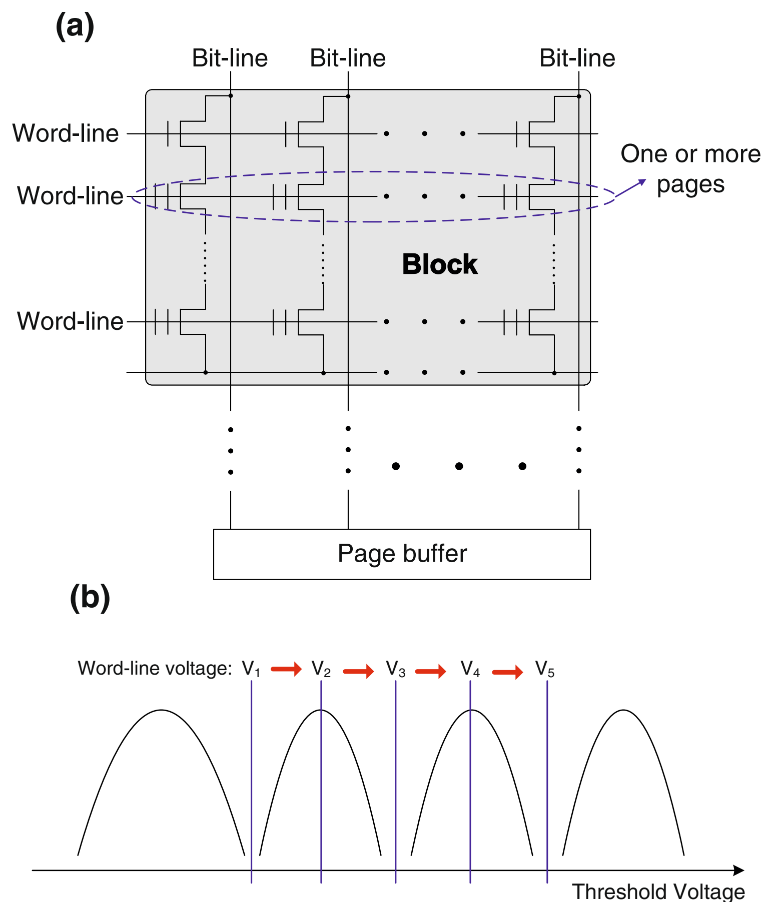


**Figure 1 Illustration of NAND flash memory sensing, (a) Illustration of NAND flash memory structure, and (b) illustration of word-line voltage sweeping for memory sensing, all the bit-lines are charged and discharged once for each word-line voltage point.**

decoder can carry out soft-decision decoding (e.g., using the sum–product or min–sum decoding algorithm) and achieve stronger and stronger error correction capability.

Very naively, the above discussion suggests that we can use a simple progressive-precision LDPC decoding strategy to reduce the memory sensing latency and flash-to-controller data transfer latency caused by the use of LDPC codes. As illustrated in Figure 2, this straightforward design strategy aims to use just-enough sensing precision for LDPC code decoding through a trial-and-error manner. As discussed above, NAND flash memory raw storage reliability gradually degrades with the P/E cycling, hence fine-grained sensing may only be necessary as flash memory approaches its end of lifetime, and low-overhead coarse-grained sensing (and even hard-decision sensing) could be sufficient during memory early lifetime. In addition, since ECC must ensure an extremely low page error rate (e.g., $10^{-12}$ and below) for data storage in NAND flash memory, low-overhead coarse-grained sensing (and even hard-decision sensing) may achieve a reasonably low error rate (e.g., $10^{-4}$), which clearly makes the progressive-precision sensing and decoding strategy well justified, especially for applications that are not very sensitive to read latency variability.

### Reducing hard-decision LDPC decoding failure probability

Under the above design framework of the progressive-precision LDPC code decoding, it is highly desirable to maximize the utilization efficiency of hard-decision LDPC code decoding (i.e., to minimize the hard-decision LDPC code decoding failure probability) in order to reduce the overall latency overhead. In this study, we propose a design method that can reduce hard-decision LDPC code decoding failure probability by adaptively configuring LDPC code parity check matrix construction throughout the entire flash memory P/E cycling lifetime. Because their inherent structural regularity can greatly facilitate efficient decoder silicon implementation, quasi-cyclic LDPC (QC-LDPC) codes have widely been studied and adopted in real-life applications. Therefore, we are only interested in the use of QC-LDPC codes in NAND flash memory. The parity check matrix **H** of a QC-LDPC code can be

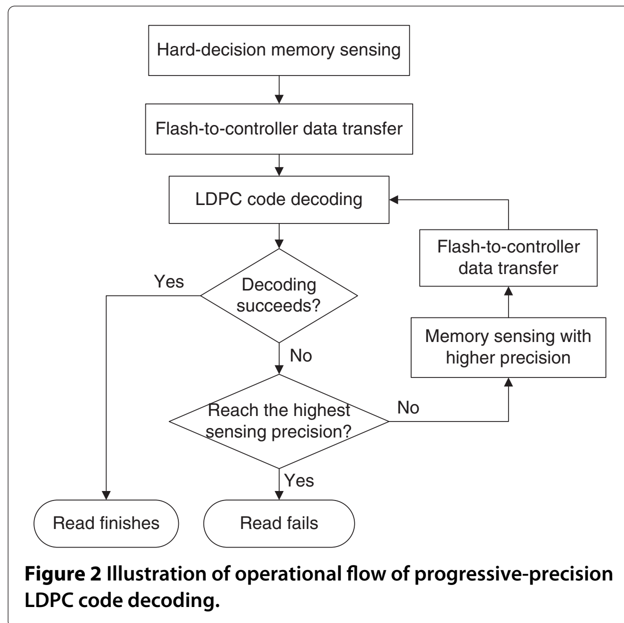


**Figure 2 Illustration of operational flow of progressive-precision LDPC code decoding.**

written as

$$\mathbf{H} = \begin{bmatrix} \mathbf{H}_{1,1} & \mathbf{H}_{1,2} & \dots & \mathbf{H}_{1,n} \\ \mathbf{H}_{2,1} & \mathbf{H}_{2,2} & \dots & \mathbf{H}_{2,n} \\ \dots & \dots & \ddots & \dots \\ \mathbf{H}_{m,1} & \mathbf{H}_{m,2} & \dots & \mathbf{H}_{m,n} \end{bmatrix},$$

where each sub-matrix $\mathbf{H}_{i,j}$ is a circulant matrix in which each row is a cyclic shift of the row above. The column weight (or row weight) of each circulant $\mathbf{H}_{i,j}$ can be 0, 1, or 2.

It is well known that LDPC code decoding performance spectrum contains two regions (i.e., water-fall region and error-floor region [20]): Starting from the worst raw storage reliability with very high raw bit error rate, as we increase the raw storage reliability, the LDPC code decoding failure rate will continue to rapidly reduce like a water-fall; however, as the raw storage reliability improves over a certain threshold, the reduction slope of LDPC code decoding failure rate will noticeably degrade, entering the so-called error-floor region. It is well known that LDPC code decoding performance spectrum is fundamentally subject to a trade-off between water-fall region performance and error-floor threshold: As illustrated in Figure 3, an LDPC code with relatively low column weight (e.g., 3 or 4) can achieve good soft-decision decoding performance within water-fall region but tends to have a relatively worse error-floor threshold (i.e., enter the error-floor region at relatively high soft-decision decoding failure probability); on the other hand, an LDPC code with relatively high column weight (e.g., 5 or 6) has a relatively better error-floor threshold but achieves worse soft-decision decoding performance within waterfall region. Straightforwardly, designers tend to choose the LDPC codes that can satisfy the target page error rate with the least parity check matrix column weight. This will accordingly maximize the soft-decision decoding performance and hence tolerate worse raw storage reliability. In addition, since decoding computational complexity is proportional to the number of 1's in the code parity check matrix, the use of low-weight LDPC codes can also reduce the decoder implementation silicon cost.

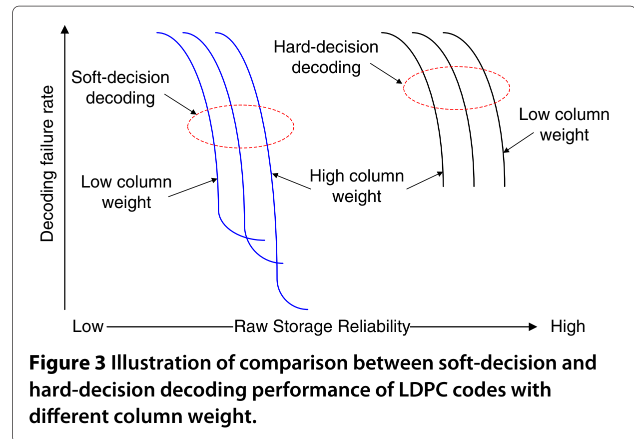


**Figure 3 Illustration of comparison between soft-decision and hard-decision decoding performance of LDPC codes with different column weight.**

On the contrary to the scenario of soft-decision decoding, a low-weight LDPC code tends to have worse hard-decision decoding performance than a high-column-weight code. This can be illustrated in Figure 3 and will be further demonstrated in "Case studies" section. This observation directly motivates us to adaptively change the LDPC code parity check matrix column weight throughout the entire NAND flash memory lifetime. Its basic idea is to use the LDPC code that has the largest possible column weight and meanwhile can meet the target page error rate through soft-decision decoding under present flash memory P/E cycling. It can be further described as follows: assume the NAND flash memory controller can support $s$ different LDPC codes, $\mathcal{C}_1, \mathcal{C}_2, \dots, \mathcal{C}_s$. Let $w_i$ represent the column weight of the code $\mathcal{C}_i$, and we have $w_s > w_{s-1} > \dots > w_1$. Let $N_i$ denote the threshold P/E cycling number beyond which the soft-decision decoding of the code $\mathcal{C}_i$ cannot satisfy the target page error rate. Based on the above discussions, we have $N_1 < N_2 < \dots < N_s$. In order to reduce the hard-decision decoding failure rate, we should use the weight-$w_i$ LDPC code $\mathcal{C}_i$ when the NAND flash memory cycling number $N \in [N_{i-1}, N_i)$, where we set $N_0$ as 0 and $N_{s+1}$ as $\infty$. Figure 3 illustrates the scenario when we have three different LDPC codes.

Compared with using a single fixed low-weight LDPC code throughout the entire lifetime of NAND flash memory, this proposed adaptive design method can achieve better hard-decision decoding performance (hence lower

hard-decision decoding failure rate) throughout the entire NAND flash memory lifetime. This can directly reduce the average latency of on-chip memory sensing and flash-to-controller data transfer caused by the use of LDPC codes in NAND flash memory. Meanwhile, we note that such an adaptive design method will lead to higher silicon implementation cost of flash memory controller since the soft-decision and hard-decision decoders must be able to support different codes with different column weights. Since we only consider the use of QC-LDPC codes, it will be sufficient for the decoder to support the maximum allowable number of column weight and run-time configuration in terms of circulant size and cyclic shift value of each circulant. Most QC-LDPC decoder architectures ever reported in the open literature (e.g., see) [21-24] can readily support such configurability.

## Case studies

For the purpose of quantitative evaluation, we develop a quantitative NAND flash memory device model that can capture the major threshold voltage distortion sources. Based upon this model, we carry out simulations to demonstrate the proposed design method.

### NAND flash device model

#### *Erase and programming operation modeling*

The threshold voltage of erased memory cells tends to have a wide Gaussian-like distribution [25], and we approximately model the threshold voltage distribution of erased state as

$$p_e(x) = \frac{1}{\sigma_e\sqrt{2\pi}} e^{-\frac{(x-\mu_e)^2}{2\sigma_e^2}}, \quad (4)$$

where $\mu_e$ and $\sigma_e$ are the mean and standard deviation of the erased state. Regarding memory programming, a tight threshold voltage control is typically realized by using incremental step pulse program [4,26], i.e., memory cells on the same word-line are recursively programmed. At older technology nodes (e.g., 90-nm node), the threshold voltage of programmed states tends to have a uniform distribution [14]. Nevertheless, in highly scaled technology nodes (e.g., 65-nm and below), the electron injection statistical spread [27] has become significant and tends to make the threshold voltage of programmed states more like Gaussian distribution. Hence, in this study we approximately model the distribution of programmed state as

$$p_p(x) = \frac{1}{\sigma_p\sqrt{2\pi}} e^{-\frac{(x-\mu_p)^2}{2\sigma_p^2}}, \quad (5)$$

where $\mu_p$ and $\sigma_p$ are the mean and standard deviation of the programmed state right after programming.

#### *RTN*

The fluctuation magnitude of RTN is subject to exponential decay. The probability density function $p_r(x)$ of RTN-induced threshold voltage fluctuation is modeled as a symmetric exponential function [14]:

$$p_r(x) = \frac{1}{2\lambda_r} e^{-\frac{|x|}{\lambda_r}}. \quad (6)$$

Since the significance of RTN is proportional to the interface trap density, we model the mean of RTN, i.e., $\mu_{\mathrm{RTN}} = \frac{1}{\lambda_r}$, approximately follows

$$\mu_{\mathrm{RTN}} = A_{\mathrm{RTN}} \cdot N^{a_{\mathrm{IT}}}. \quad (7)$$

#### *Retention process*

Since interface trap recovery and electron detrapping processes tend to follow Poisson statistics [9], we approximately model the induced threshold voltage reduction as a Gaussian distribution, i.e., $p_t(x) = \mathcal{N}(\mu_d, \sigma_d^2)$. As demonstrated in relevant device studies (e.g., see) [9,28], mean value of threshold voltage shift scales approximately with $\ln(1+t)$ over the time. The mean value of retention shift is set to follow the mean of sum of interface traps and oxide traps:

$$\mu_d = (A_t \cdot N^{a_{\mathrm{IT}}} + B_t \cdot N^{a_{\mathrm{OT}}}) \cdot \ln(1+t). \quad (8)$$

Moreover, the significance of threshold voltage reduction induced by interface trap recovery and electron detrapping is also proportional to the initial threshold voltage magnitude [29], i.e., the higher the initial threshold voltage is, the faster the interface trap recovery and electron detrapping occur and hence the larger threshold voltage reduction will be. Hence, we set the generated retention noise approximately scale $K_s(x - x_0)$, where $x$ is the initial threshold voltage before retention, and $x_0$ and $K_s$ are constants.

#### *Cell-to-cell interference*

To capture inevitable process variability in cell-to-cell interference model, we set both the vertical coupling ratio $\gamma_y$ and diagonal coupling ratio $\gamma_{xy}$ as random variables with truncated Gaussian distribution:

$$p_c(x) = \begin{cases} \frac{c_c}{\sigma_c\sqrt{2\pi}} \cdot e^{-\frac{(x-\mu_c)^2}{2\sigma_c^2}}, & \text{if } |x-\mu_c| \le w_c \\ 0, & \text{else} \end{cases}, \quad (9)$$

where $\mu_c$ and $\sigma_c$ are the mean and standard deviation, and $c_c$ is chosen to ensure the integration of this bounded

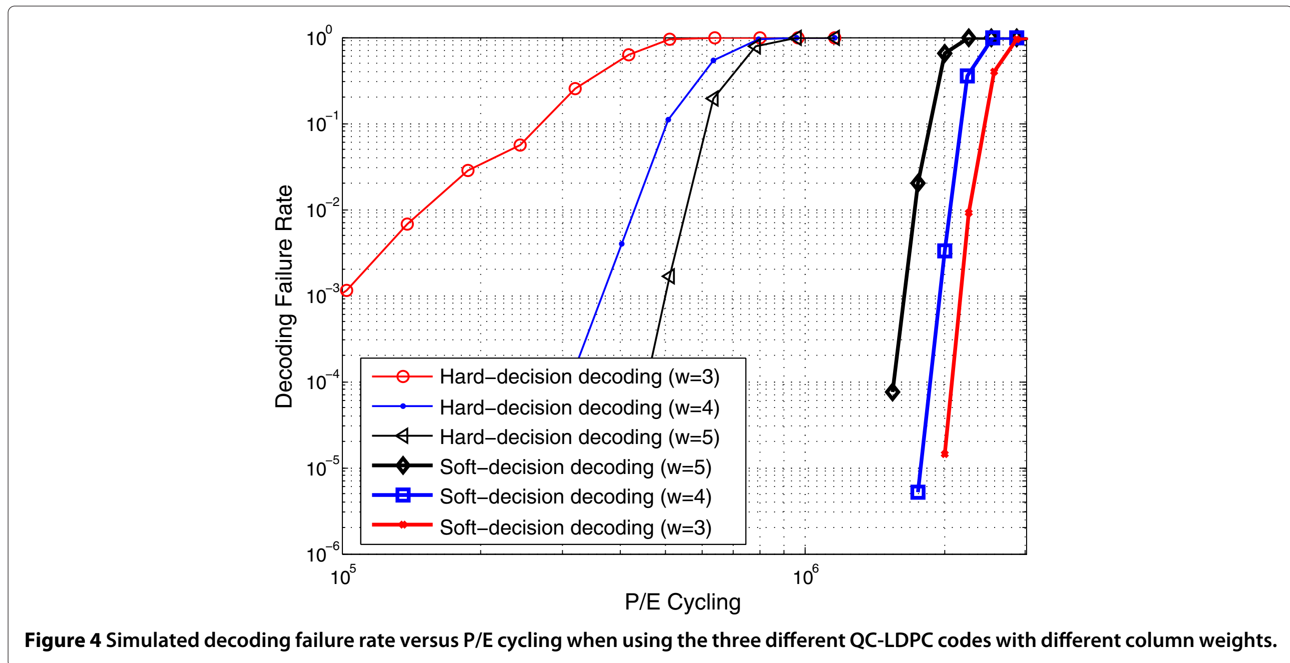


**Figure 4 Simulated decoding failure rate versus P/E cycling when using the three different QC-LDPC codes with different column weights.**

Gaussian distribution equals to 1. According to [18], we set the ratio between the means of $\gamma_x$, $\gamma_y$, and $\gamma_{xy}$ as 0.1:0.08:0.006.

#### *Overall device model*

Based upon (4) and (5), we can obtain the threshold voltage distribution function $p_p(x)$ right after programming operation. Then we have the threshold voltage distribution after incorporating RTN $p_{\text{ar}}(x)$ as

$$p_{\text{ar}}(x) = p_p(x) \bigotimes p_r(x). \tag{10}$$

After we incorporate the cell-to-cell interference, we have the threshold voltage distribution $p_{\text{ac}}$. Let $p_t(x)$ denote the distribution of retention noise caused by interface state trap recovery and electron detrapping. The final threshold voltage distribution $p_f$ is obtained as

$$p_f(x) = p_{\text{ac}}(x) \bigotimes p_t(x). \tag{11}$$

### Experiments

Based upon the above NAND flash memory device model, we carried out simulations to compare the error-correction performance between soft-decision and hard-decision LDPC code decoding and demonstrate the effectiveness of the proposed adaptive design method. In this study, we set that each LDPC codeword protects 2 kB user data, and construct three rate-8/9 QC-LDPC codes with the column weight of 3, 4, and 5, respectively. The code parity check matrices of these three codes contain $3 \times 27$, $4 \times 36$, and $5 \times 45$ circulants, respectively, where all the circulants have a column weight of 1 and are constructed randomly subject to the 4-cycle-free constraint. LDPC code soft-decision decoding employs the min–sum decoding algorithm [6], and hard-decision decoding employs the bit-flipping decoding algorithm [5].

Based upon the NAND flash memory device model described in "NAND flash device model" section, we use 2 bits/cell NAND flash memory with the following device parameters as a test vehicle. We set the normalized $\sigma_e$ and $\mu_e$ of the erased state as 0.35 and 1.4, respectively. For programmed state, we set the normalized program step voltage $\Delta V_{\text{pp}}$ as 0.2, and its deviation as 0.05. According to [12], the exponents for interface and oxide traps

**Table 1 LDPC decoder design results at 65-nm technology node**

| | Silicon area (mm$^2$) | | | | | Throughput |
|---|---|---|---|---|---|---|
| | Computation | Register array | SRAM | Others | Total | |
| Soft-decision decoder | 0.40 | 0.26 | 1.96 | 0.19 | 2.81 | 2.1 Gbps |
| Hard-decision decoder | 0 | 0.06 | 0.38 | 0.26 | 0.70 | @ 300 MHz |

generation are estimated as $a_{\mathrm{IT}} = 0.62$ and $a_{\mathrm{OT}} = 0.3$, respectively. For RTN, we set $A_{\mathrm{RTN}} = 2.72 \times 10^{-4}$. The coupling strength factor is set as 1. As for retention shift, we set $\sigma_d = 0.3|\mu_d|$, and $A_t = 3.5 \times 10^{-5}$, $B_t = 2.35 \times 10^{-4}$, which are chosen to match the 70%:30% ratio of interface trap recovery and electron detrapping presented in [12]. Regarding the influence of the initial threshold voltage, we set $x_0 = 1.4$ and $K_s = 0.333$. We set $w_c = 0.1\mu_c$ and $\sigma_c = 0.4\mu_c$. Accordingly, we carried out Monte Carlo simulations to evaluate the decoding failure rate statistics when using different LDPC codes with both hard-decision decoding and soft-decision decoding. The simulation results are shown in Figure 4. The simulation results clearly show that high-weight LDPC codes perform better than their low-weight counterparts under hard-decision decoding. This is completely opposite to the scenario of using soft-decision decoding. Therefore, it is highly desirable to employ the high-weight LDPC codes in the early lifetime of NAND flash memory in order to reduce the latency overhead.

We further carried out ASIC design to evaluate the silicon overhead of implementing both soft-decision and hard-decision LDPC decoders. With the RTL-level design entry using Verilog, we use Synopsys tool set and 65-nm CMOS standard cell library. The target decoding throughput is 2 Gbps. Both decoders carry out the decoding in a partially parallel manner, and can be on-the-fly configured in terms of cyclic shift value of each circulant, circulant size, and column weight. The soft-decision decoder architecture directly follows the one presented in [22], and the hard-decision decoder employs the similar architecture. All the decoding messages in the soft-decision decoder have 4-bit precision. Table 1 summarizes the ASIC design results, which clearly show that the addition of a hard-decision decoder only induces a relatively small silicon overhead.

## Conclusion

This article concerns the potentially significant latency overhead caused by the use of powerful soft-decision ECC, in particular LDPC codes, in future NAND flash memory. Although LDPC codes can achieve excellent error-correction capability, their soft-decision decoding nature directly results in significant latency overhead in terms of on-chip memory sensing and flash-to-controller data transfer. We propose a simple yet effective design technique that can reduce such latency overhead. Based upon an approximate NAND flash memory device model, we carried out simulations and the results clearly demonstrate the potential effectiveness of the proposed design solution.

#### Competing interests

The authors declare that they have no competing interests.

#### Acknowledgement

This research was funded in part by grants from the National Natural Science Foundation of China (No. 61274028), the National High-tech R&D Program of China (No. 2011AA010405), NSF grants CNS-1162152 and NSF grants CCF-0937794. The authors are also grateful to the anonymous reviewers for their valuable and constructive comments.

#### Author details

[1]The Institute of Artificial Intelligence and Robotics, Xi'an Jiaotong University, Shaanxi, China. [2]Skyera Inc.1704, Automation pkwy, San Jose, CA 95131, USA. [3]Electrical, Computer and Systems Engineering Department, Rensselaer Polytechnic Institute, Troy, NY, USA.

Received: 12 April 2012 Accepted: 21 August 2012
Published: 19 September 2012

#### References

1. R. E. Blahut, *Algebraic Codes For Data Transmission*. (Cambridge, MA: Cambridge University Press, 2003)
2. RG Gallager, Low-density parity-check codes. IRE Trans. Inf. Theory. **IT-8**, 21–28 (1962)
3. DJC MacKay, Good error-correcting codes based on very sparse matrices. IEEE Trans. Inf. Theory. **45**, 399–431 (1999)
4. R Bez, E Camerlenghi, A Modelli, A Visconti, Introduction to Flash memory. Proc. IEEE. **91**, 489–502 (2003)
5. Y Kou, S Lin, MPC Fossorier, Low-density parity-check codes based on finite geometries: a rediscovery and new results. IEEE Trans. Inf. Theory. **47**, 2711–2736 (2001)
6. J Chen, A Dholakia, E Eleftheriou, M Fossorier, X-Y Hu, Reduced-complexity decoding of LDPC codes. IEEE Trans. Commun. **53**, 1288–1299 (2005)
7. P Olivo, B Ricco, E Sangiorgi, High-field-induced voltage-dependent oxide charge. Appl. Phys. Lett. **48**, 1135 (1986)
8. P Cappelletti, R Bez, D Cantarelli, L Fratin, Failure mechanisms of Flash cell in program/erase cycling. in *International Electron Devices Meeting*. (San Francisco, 291–294, 1994)
9. N Mielke, H Belgal, I Kalastirsky, P Kalavade, A Kurtz, Q Meng, N Righos, J Wu, Flash EEPROM threshold instabilities due to charge trapping during program/erase cycling. IEEE Trans. Dev. Mater. Reliab. **4**(3), 335–344 (2004)
10. J. B. Yang, T. P. Chen, S. S. Tan, L Chan, Analytical reaction-diffusion model the modeling of nitrogen-enhanced negative bias temperature instability. Appl. Phys. Lett. **88**, 172109–172109-3 (2006)
11. S Ogawa, N Shiono, Generalized diffusion-reaction model for the low-field charge-buildup instability at the Si-Sio2 interface. Phys. Rev. B. **51**, 4218–4230 (1995)
12. Hong Yang, H Kim, J Park, S Kim, S Lee, J Choi, D Hwang, C Kim, M Park, K Lee, Y Park, J Shin, J Kong, Reliability issues and models of sub-90nm NAND Flash memory cells. in *International conference on Solid-State and Integrated Circuit Technology*. (Shanghai, 760–762, 2006)
13. K Fukuda, Y Shimizu, K Amemiya, M Kamoshida, C Hu, Random telegraph noise in flash memories model and technology scaling. in *IEEE International Electron Devices Meeting*. (Washington, 169–172, 2007)
14. C Compagnoni, M Ghidotti, A Lacaita, A Spinelli, A Visconti, Random telegraph noise effect on the programmed threshold-voltage distribution of flash memories. IEEE Electron. Dev. Lett. **30**(9), 984–986 (2009)
15. N Mielke, H Belgal, A Fazio, Q Meng, N Righos, Recovery effects in the distributed cycling of flash memories. in *Proc. of IEEE International Reliability Physics Symposium*. (San Jose, 29–35, 2006)
16. J-D Lee, S-H Hur, J-D Choi, Effects of floating-gate interference on NAND flash memory cell operation. IEEE Electron. Dev. Lett. **23**(5), 264–266 (2002)
17. K Kim, Future memory technology: challenges and opportunities. in *Proc. of International Symposium on VLSI Technology, Systems and Applications*. (Hsinchu, 5–9, 2008)
18. K Prall, Scaling non-volatile memory below 30nm. in *IEEE 2nd Non-Volatile Semiconductor Memory Workshop*. (Monterey, 5–10, 2007)
19. H Liu, S Groothuis, C Mouli, J Li, K Parat, T Krishnamohan, 3D simulation study of cell-cell interference in advanced NAND flash memory. in *Proc. IEEE Workshop on Microelectronics and Electron Devices*. (Boise, 1–3, 2009)
20. T Richardson, Error floors of LDPC codes. in *Proc. 41st Allerton Conference on Communications, Control and Computing*, **41**(3) , 1426–1435, (2003)
21. M Mansour, NR Shanbhag, A 640-mb/s 2048-bit programmable ldpc decoder chip. IEEE J. Solid State Circuits. **41**(3), 684–698 (2006)

22. H Zhong, W Xu, N Xie, T Zhang, Area-efficient min-sum decoder design for high-rate quasi-cyclic low-density parity-check codes in magnetic recording. IEEE Trans. Magnet. **43**, 4117–4122 (2007)
23. Y Dai, Z Yan, N Chen, Memory-efficient and high-throughput decoding of quasi-cyclic ldpc codes. IEEE Trans. Commun. **57**(4), 879–883 (2009)
24. K Zhang, X Huang, Z Wang, A high-throughput ldpc decoder architecture with rate compatibility. IEEE Trans. Circuits Syst. I: Regular Papers. **58**(4), 839–847 (2011)
25. K Takeuchi, T Tanaka, H Nakamura, A double-level-Vth select gate array architecture for multilevel NAND flash memories. IEEE J. Solid-State Circuits. **31**(4), 602–609 (1996)
26. K-D Suh, B Su, Y Lim, J Kim, Y Choi, Y Koh, S Lee, S Kwon, B Choi, J Yum, J Choi, J Kim, H Lim, A 3.3 V 32 Mb NAND flash memory with incremental step pulse programming scheme. IEEE J. Solid State Circuits. **30**(11), 1149–1156 (1995)
27. C Compagnoni, A Spinelli, R Gusmeroli, A Lacaita, S Beltrami, A Ghetti, A Visconti, First evidence for injection statistics accuracy limitations in NAND Flash constant-current Fowler-Nordheim programming. in *IEEE International Electron Devices Meeting*, 165–168, (2007)
28. CM Compagnoni, C Miccoli, R Mottadelli, S Beltrami, M Ghidotti, AL Lacaita, AS Spinelli, A Visconti, Investigation of the threshold voltage instability after distributed cycling in nanoscale NAND flash memory arrays. in *IEEE International Reliability Physics Symposium (IRPS)*. (Anaheim, 604–610, 2010)
29. J Lee, J Choi, D Park, K Kim, R Center, S Co, S Gyunggi-Do, Effects of interface trap generation and annihilation on the data retention characteristics of flash memory cells. IEEE Trans. Dev. Mater. Reliab. **4**(1), 110–117 (2004)

doi:10.1186/1687-6180-2012-203
**Cite this article as:** Zhao *et al.*: Reducing latency overhead caused by using LDPC codes in NAND flash memory. *EURASIP Journal on Advances in Signal Processing* 2012 **2012**:203.

**Submit your manuscript to a SpringerOpen journal and benefit from:**

- ► Convenient online submission
- ► Rigorous peer review
- ► Immediate publication on acceptance
- ► Open access: articles freely available online
- ► High visibility within the field
- ► Retaining the copyright to your article

**Submit your next manuscript at ► springeropen.com**