Using Multilevel Phase Change Memory to Build Data Storage: A Time-Aware System Design Perspective

Qi Wu, *Senior Member*, *IEEE*, Fei Sun, *Senior Member*, *IEEE*, Wei Xu, *Senior Member*, *IEEE*, and Tong Zhang, *Senior Member*, *IEEE*

Abstract—This paper advocates a time-aware design methodology for using multilevel per cell (MLC) phase-change memory (PCM) in data storage systems such as solid-state disk and disk cache. It is well known that phase-change material resistance drift gradually reduces memory device noise margin and degrades the raw storage reliability. Intuitively, due to the time-dependent nature of resistance drift, if we can dynamically adjust storage system operations adaptive to the time and, hence, memory cell resistance drift, we may improve various PCM-based data storage system performance metrics. Under such an intuitive time-aware system design concept, we propose three specific design techniques, including time-aware variable-strength error correction code (ECC) decoding, time-aware partial rewrite, and time-aware read-&-refresh. Since PCM-based data storage systems have to use powerful ECC whose decoding can be energy-hungry, the first technique aims to minimize the ECC decoding energy consumption. The second technique improves the data retention limit when using partial rewrite in MLC PCM, and the third technique can further improve the efficiency of time-aware variable-strength ECC decoding. Using hypothetical 2-bit/cell PCM with device parameters from recent device research as a test vehicle, we carry out mathematical analysis and trace-based simulations, which show that these techniques can improve the data retention limit by few orders of magnitude, and enable up to 97 and 79 percent energy savings for PCM-based solid-state disk and PCM-based disk cache.

Index Terms—Phase change memory, MLC, ECC, resistance drift

1 Introduction

Due to well-recognized scalability limits of current mainstream memory technologies such as flash memory and DRAM, there has been recently a resurgence of interest in search of highly scalable next-generation memory technology. Phase-change memory (PCM) is one of the most promising candidates that have attracted a lot of attentions (e.g., see [1], [2], [3], [4], [5], [6], [7], [8]). Data storage in PCM is realized by configuring the chalcogenide material, usually $Ge_2Sb_2Te_5$ (GST), into either a crystalline or an amorphous phase with resistances differing by several orders of magnitude [9]. Due to the large resistance margin between crystalline and amorphous phases, PCM can realize multilevel per cell (MLC) storage based upon incomplete phase transition [4], [5], which is conceptually similar to MLC NAND flash memory. As pointed out by a recent survey paper from IBM [10], whether PCM will survive or completely fail as a mainstream technology essentially depends on its cost effectiveness. Therefore, economic and reliable realization of MLC PCM can be almost indispensable for PCM to successfully enter mainstream market [10]. As demonstrated in recent studies [11], [12], [13], [14], resistance of phase-change material tends to drift over the time with significant variability in terms of drift speed. This could severely degrade the noise margin and, hence, raw storage reliability of MLC PCM.

Recent significant progress of PCM technology motivated the computing system research community to investigate the potential applications of PCM in future computing systems. Most prior work focused on using PCM to replace or complement with DRAM as main memory, for example, see [15], [16], [17], [18], [19], [20], [21]. However, due to its relatively long access latency, limited cycling endurance, and much worse raw storage reliability, MLC PCM may hardly compete with DRAM as main memory, especially considering the fact that DRAM technology is entering sub-30-nm region. In this work, we are interested in using MLC PCM in low-level data storage, i.e., solid-state disk and disk cache, which can much more naturally embrace the long access latency and much worse raw storage reliability of MLC PCM. Nevertheless, resistance drift becomes a much more critical issue when using MLC PCM as data storage that tends to have long data retention requirement [10]. Intuitively, because of the time-dependent nature of PCM cell resistance drift, system deign techniques aware of data lifetime (i.e., how long the data has resided in PCM) could better embrace the resistance drift issue and, hence, improve various system performance metrics. This very straightforward intuition motivates us to investigate the PCM-based data storage system design from a time-aware design perspective.

- *Q. Wu is with Skyera, 1704 Automation Parkway, San Jose, CA 95131. E-mail: seaflywu@gmail.com.*
- *F. Sun and W. Xu are with Marvell, Santa Clara, CA 95054. E-mail: {fei.sun, wei.xu.rpi}@gmail.com.*
- *T. Zhang is with the RPI, 15 Windrose Way, Watervliet, NY 12189. E-mail: tong.zhang@ieee.org.*

Manuscript received 13 Apr. 2011; revised 23 Jan. 2012; accepted 10 May 2012; published online 4 June 2012.
Recommended for acceptance by L. John.
For information on obtaining reprints of this article, please send e-mail to: tc@computer.org, and reference IEEECS Log Number TC-2011-04-0241.
Digital Object Identifier no. 10.1109/TC.2012.128.

First, we propose a *time-aware variable-strength error correction code (ECC) decoding* design strategy to minimize the ECC decoding energy consumption in PCM-based data storage. MLC PCM inevitably demands very powerful ECC, especially as we push the technology scaling to its limit. Due to the resistance drift, the raw storage reliability of MLC PCM cells gradually degrades with the time, and the error correction capability of ECC should be strong enough to tolerate the worst-case raw storage reliability when data lifetime reaches the retention limit (e.g., 10 or 20 years). Assume we use an ECC, which can correct up to t errors, to protect all the data stored in MLC PCM to ensure a data retention time of 10 years. If we always carry out the full-strength ECC decoding (i.e., trying to correct up to t errors) whenever we read data from the memory, most of time such a full-strength ECC decoding is more-than-enough because the data lifetime most likely is (much) shorter than 10 years. Because the ECC decoding strength can largely affect the decoding computational complexity and energy consumption, we propose to dynamically adjust the ECC decoding strength adaptive to the data lifetime to reduce the decoding energy consumption: If we know the data lifetime during the memory read, we can estimate the resistance drift significance and correspondingly estimate the raw read bit error rate (BER), based on which we can calculate the necessary error correction capability. Suppose the calculation shows that correction up to $t_s \leq t$ errors is sufficient, we can adjust the ECC decoding procedure so that it can only correct up to t_s errors. Being widely used in various data storage systems, BCH code is used to further demonstrate this proposed design method in this paper.

It is straightforward that MLC PCM memory sensing should be carried out in a time-aware manner as proposed in [22], i.e., if we know the data lifetime and, hence, correspondingly estimate the resistance drift significance, we should accordingly adjust the cell resistance sensing quantization thresholds to minimize the sensing BER. Such time-aware memory sensing apparently demands that all the memory cells in the data block experience the same degree of resistance drift. A technique called partial rewrite has been widely studied in the open literature as an effective way to mitigate the long PCM write latency and high PCM write energy [15], [16], [17], [21], [23], [24], [25]. However, prior work ignored the resistance drift issue, and time-independent partial rewrite apparently makes different cells experience different degree of resistance drift and have different resistance distribution. This will degrade the effectiveness of time-aware memory sensing and results in more memory sensing errors. To address this issue, we propose a *time-aware partial rewrite* strategy to improve the feasibility of partial rewrite in PCM. The basic idea is to adaptively adjust the rewrite operations so that the rewritten cells tend to have the same resistance distribution as those unrewritten cells. It can minimize sensing BER and, hence, improve the achievable data retention time. Finally, we propose a *time-aware read-&-refresh* scheme to further improve the energy savings of time-aware variable-strength ECC decoding. When a data block is read, if its lifetime is longer than a prespecified lifetime threshold, then we refresh this data block by completely rewriting the entire data block into the MLC PCM. When a data block is written, if its lifetime is longer than a prespecified life threshold, we simply perform a whole page write instead of partial rewrite. This will reset the resistance drift and, hence, largely reduce the time-aware ECC decoding energy consumption for subsequent read of this data block.

We carry out mathematical analysis and trace-based simulations to demonstrate the proposed time-aware design techniques, where we use MLC PCM as either solid-state disk or disk cache. Assuming Gaussian distributions of PCM cell resistance drift parameters as suggested by recent experimental measurements [6], [26], we mathematically derive how the memory sensing error rate can be estimated with and without using time-aware memory sensing and time-aware partial rewrite. Using hypothetical 2-bit/cell PCM with memory cell resistance drift statistical parameters from recent device studies [13], [14] and using BCH code as ECC, we quantitatively demonstrate that the use of time-aware partial rewrite can enable up to four orders of magnitude improvement on the achievable data retention limit. Based upon a variety of disk access traces and ASIC (application-specific integrated circuit) design results of BCH decoders at 65-nm node, we carry out extensive simulations that show the time-aware variable-strength BCH decoding combined with time-aware read-&-refresh can reduce the overall system energy consumption up to 97 percent for PCM-based solid-state disk and 79 percent for PCM-based disk cache.

2 Background

Each PCM cell consists of an access control device (e.g., an nMOS transistor) and a storage element, i.e., a chalcogenide layer ($Ge_2Sb_2Te_5$, or GST). Being heated the GST can switch between the amorphous (RESET) state and crystalline (SET) state. When the chalcogenide material is amorphous (or crystalline), the GST shows high (or low) resistance. The resistance may differ by several orders of magnitude, leading to excellent nonvolatile storage capability. The amorphous state is obtained by a short current pulse with high peak value, referred to reset pulse, which first rises the temperature of GST above the melting temperature (about $620\,^{\circ}\mathrm{C}$) and then quickly cools it down [9]. To obtain the crystalline state, the GST is heated by a longer current pulse with lower peak value, referred to set pulse. It heats the GST up to around $550\,^{\circ}\mathrm{C}$, which is lower than the melting temperature but high enough to realize the crystalline transition [9]. Due to the large resistance margin between crystalline and amorphous states, PCM can support MLC storage by realizing incomplete phase transition, for example, researchers have recently demonstrated 4-level per cell (i.e., 2-bit per cell) PCM cells with four distinguishable resistance states [4]: very amorphous, amorphous, semicrystalline and crystalline.

PCM, particularly MLC PCM, is subject to a severe storage reliability issue: while the crystalline phase of phase-change chalcogenide material is relatively stable over the time, the amorphous phase (and other phases obtained by incomplete phase transition in MLC PCM) is metastable and experiences structural relaxation [11], which results in resistance drift over the time. Experimental results [11], [12], [13] suggest that the resistance tends to drift over the time with t^{ν}, i.e.,

$$R(t) = R_0\left(\frac{t}{t_0}\right)^{\nu}, \tag{1}$$

where R_0 is the initial resistance and t_0 is a normalized time constant, and ν is the drift exponent. Experimental results show that ν strongly depends on the initial resistance and exhibits a large degree of variability. Since R_0 and ν depend on various factors [5], [13], [14], [27], it is very difficult to accurately model their variability through theoretical analysis. Based upon test chip measurements [6], [26], R_0 and ν appear to approximately follow Gaussian distributions. Meanwhile, drift exponent ν tends to increase as the initial resistance increase due to a larger structural relaxation effect. As a result, memory cells in MLC PCM experience different resistance drift over the time, leading to significant degradation of memory operational noise margin.

3 Proposed Time-Aware PCM Data Storage System Design Techniques

Due to the resistance drift of memory cells, MLC PCM exhibits strong time-dependent dynamics of raw storage reliability. Intuitively, if a PCM system can dynamically adjust its operations adaptive to the time, it may potentially improve various system performance metrics such as energy consumption and endurance. Such an intuition motivates us to develop several time-aware PCM system design techniques as described in this section. Before presenting these design techniques, we note that, to enable any time-aware operations, each data block must be appended with the present time when it is being written into the PCM (i.e., its time stamp). To ensure PCM storage integrity, we use SLC PCM to store time stamp associated with each PCM page. Time stamp only needs few tens of bits (e.g., a 4-byte time stamp with a unit of second can cover an over 10-year period). Compared with the page size (e.g., 512-byte or 4k-byte), it induces negligible storage density degradation.

3.1 Time-Aware Variable-Strength ECC Decoding

ECC will inevitably play an important role in real-life PCM products, especially MLC PCM [10]. Clearly, the error correction capability of ECC must ensure the specified decoding failure rate (e.g., 10^{-15} and below) under the PCM data retention requirement (e.g., 10 years). Due to the gradual raw storage reliability degradation induced by resistance drift, the maximum error correction capability of ECC is inherently *more than enough* (or *underutilized*) before the data lifetime reaches the PCM data retention limit. In general, ECC decoding energy consumption depends on the error correction decoding strength, i.e., as the ECC decoding intends to correct more bit errors, the necessary decoding computational complexity and, hence, energy consumption will accordingly increase.

Straightforwardly, the above discussion motives us to propose time-aware variable-strength ECC decoding to opportunistically reduce ECC decoding energy consumption. Once we know the lifetime of one page stored in PCM (i.e., how long it has resided in PCM), based on which we can estimate the resistance drift significance and, hence, raw storage reliability, we can accordingly adjust the ECC decoding strength to provide *just-enough* error correction capability. Therefore, the ECC decoder can always carries out *just-enough* computation and, hence, consumes *just-enough* energy. In the following, we use the binary BCH code as an example to further elaborate this concept.

A binary BCH code is typically denoted as (n, k, t) BCH code (i.e., it protects k-bit user data using $(n-k)$-bit redundancy and can correct up to t-bit errors). Given an (n, k, t) BCH code, to correct up to t_s errors (where $t_s \leq t$), its decoding mainly carries out three computational tasks [28], including 1) syndrome computation with a computational complexity proportional to $n \cdot t_s$, 2) error locator calculation with a computational complexity proportional to t_s^2, and 3) Chien search with a computational complexity proportional to n. When t_s is relatively small (e.g., 10 or 20), syndrome computation and Chien search tend to largely dominate the overall computational complexity and decoding energy consumption, while the energy consumption of error locator calculation can be noticeable or even comparable as t_s increases. In conventional design practice, an (n, k, t) BCH code decoder always aims to correct up to t errors, i.e., spend the highest computational complexity and energy to maximize the error correction capability. This is referred to as fixed-maximum-strength BCH decoding.

Under the proposed time-aware variable-strength ECC decoding framework, we should dynamically adjust the value of t_s in the BCH decoding based upon the data lifetime. Compared with conventional fixed-maximal-strength decoding, it can dynamically reduce the energy consumption of syndrome computation and error locator calculation. We note that the inputs to BCH decoder are always the same regardless to decoding strength, and decoding energy savings are gained through simple hardware shutdown and/or shorter execution of syndrome computation and error locator calculation. This is illustrated in Fig. 1. All the data blocks stored in PCM are protected by the same (n, k, t) BCH code, which can ensure data storage integrity as the data block lifetime reaches the data retention limit. When a data block is being read, its lifetime information is also sent to the decoder, which calculates the appropriate decoding strength t_s using the lifetime information and PCM device parameters. Since the calculation of t_s could be very complex as we will describe in Section 4.1, it can be realized through lookup table (LUT) in practice.

3.2 Time-Aware Memory Operations

Besides the external ECC decoding, internal PCM circuit operations such as read and write should also be carried out in a time-aware manner. As recently pointed out in [22], in the presence of significant resistance drift, PCM memory cell sensing should be done in a time-aware manner, i.e., we should keep track of memory content lifetime and accordingly adjust how to quantize and interpret memory cell resistance to minimize memory sensing BER. This can be illustrated in Fig. 2. It shows the resistance drift of 4-level per cell PCM cells. Since a larger initial resistance tends to have a larger drift exponent, higher storage levels with larger initial resistance are subject to bigger resistance drift, as illustrated in Fig. 2b. In this example we assume that memory cells are written at $t = 0$ and read at $t = T$. As

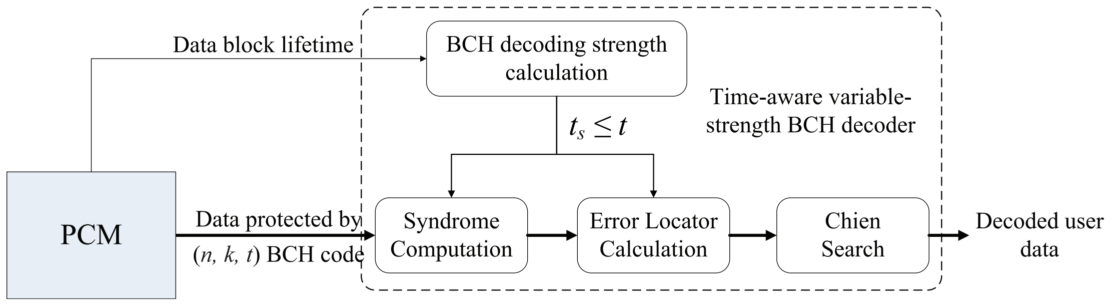

Data block lifetime
BCH decoding strength calculation
$t_s \leq t$
Time-aware variable-strength BCH decoder
PCM
Data protected by (n, k, t) BCH code
Syndrome Computation
Error Locator Calculation
Chien Search
Decoded user data

Fig. 1. Illustration of time-aware variable-strength BCH decoding that dynamically adjust its decoding complexity and energy consumption based on time-dependent memory cell resistance drift significance.

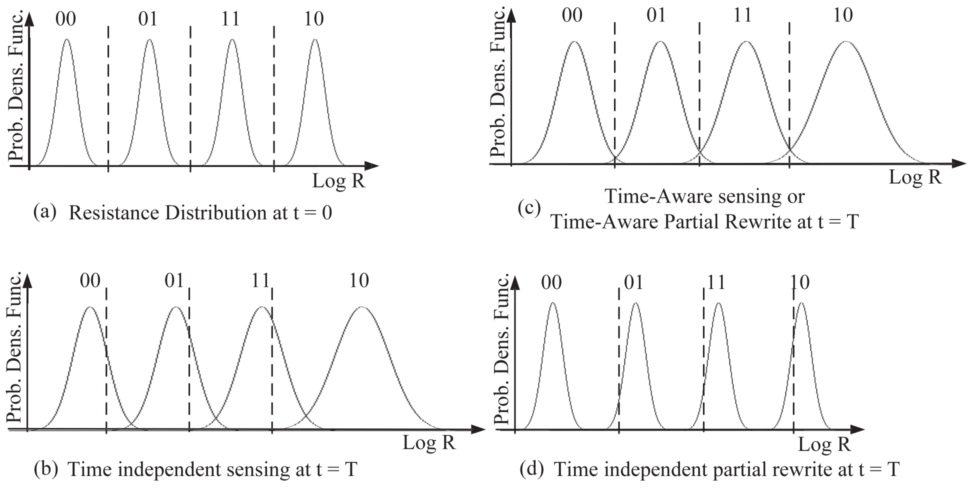


Fig. 2. An example to illustrate the basic concept of time-aware PCM read and time-aware PCM partial rewrite.

illustrated in Fig. 2 b, if we use a static memory cell resistance quantization, i.e., the memory cell resistance quantization thresholds are fixed, the sensing BER will rise dramatically over the time. On the other hand, if we keep track of the data lifetime, we can estimate the resistance drift parameters, and hence, dynamically adjust the memory cell resistance quantization thresholds, as shown in Fig. 2c. Beyond the above time-aware memory cell sensing, we propose two time-aware design techniques for PCM circuit operations:

1. *Time-aware partial rewrite*. Because PCM write is very time and energy consuming, the potential effectiveness of partial rewrite has been well studied and demonstrated [15], [16], [17], [21], [23], [24], [25]. However, resistance drift has been largely ignored in prior work. If partial rewrite is carried out in a time-independent manner, the rewritten cells and unrewritten memory cells will experience different resistance drift and, hence, have different resistance distributions. Since we cannot keep track which cells are rewritten or unrewritten, we will not be able to use the optimal memory sensing quantization schemes for all the memory cells when the data block is being read, which apparently leads to higher read BERs. This problem can be solved if we use time-aware partial rewrite as shown in Fig. 3. The basic idea is simple: When we intend to partially rewrite a data block after a period of T because the data block has been written into PCM, we can accordingly configure the memory write operation so that the rewritten memory cells may have same or similar distribution as those unrewritten cells. This can be realized by adjusting the memory cell resistance target, for example, suppose the resistance of unrewritten memory cells have the distribution as shown in Fig. 2c with the optimal memory cell sensing threshold denoted as the three dashed lines, we can rewrite those memory cells in such a way that their resistance exactly fall into the same distribution regions. With this time-aware memory partial rewrite, we can always use the optimal time-aware memory cell sensing quantization configuration for all the memory cells in one data block, leading to minimized read BERs.

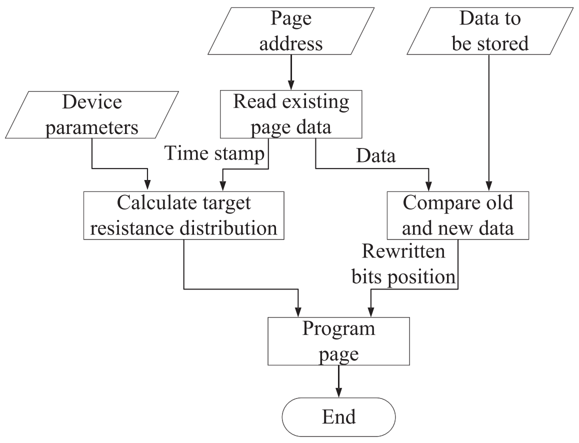


Fig. 3. Operational flow diagram of time-aware partial rewrite.

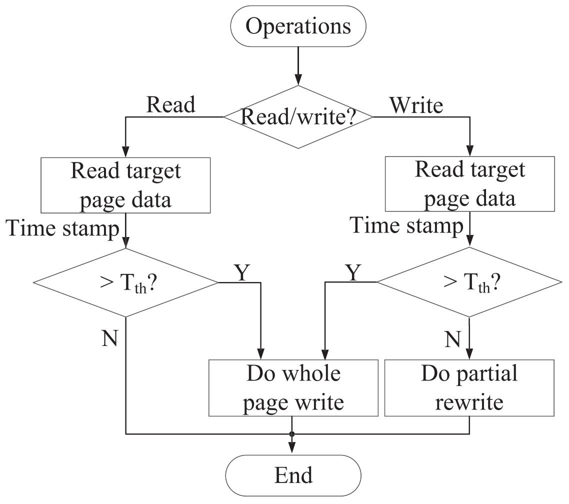


Fig. 4. Operational flow diagram of time-aware read-&-refresh.

2. *Time-aware read-&-refresh*. If a data block has resided in PCM for a relatively long time and, hence, experienced significant resistance drift, reading this data block will suffer a high BER, and hence, the corresponding ECC decoding will consume more energy. In this context, we propose a time-aware read-&-refresh scheme, as shown in Fig. 4, which can complement with the above proposed time-aware variable-strength ECC decoding to reduce the overall system energy consumption. The basic idea is described as follows: When a data block is read, if its lifetime is longer than a prespecified threshold T_{th}, then we refresh this data block by completely rewriting the entire data block into the PCM. When a data block is written, if its lifetime is longer than a prespecified life threshold T_{th}, we simply perform a whole page write instead of partial rewrite. This will reset the resistance drift and, hence, largely reduce the time-aware ECC decoding energy consumption for subsequent read of this data block. Notice that we do not aim to exhaustively refresh all the data as in DRAM, instead we perform refresh only when a data block is being accessed by the host. Therefore, with a sufficiently large lifetime threshold T_{th}, such read-&-refresh operations will not noticeably degrade the PCM data storage system availability

Finally, we discuss the comparison between the above proposed time-aware design techniques and simple periodic refresh. At the first glance, one may argue that periodic refresh as in DRAM can effectively handle PCM resistance drift and make those time-aware design techniques less effective or even unnecessary. Since storage disks must guarantee a stringent data retention time (e.g., at least a few years) even without power supply, the ECC strength is solely determined by the worst-case resistance drift induced by the worst-case data retention time, regardless to whether periodic refresh is being used or not. Therefore, periodic refresh cannot be used to reduce ECC redundancy. Therefore, we still need apply the time-aware variable-strength ECC decoding to reduce energy consumption, while the energy saving can be more significant if periodic refresh is being used. Being completely orthogonal to refresh (either periodic refresh or time-aware read-&-refresh), time-aware partial rewrite aims to reduce BER when partial rewrite strategy is being employed to reduce PCM write energy consumption. Compared with brute-force periodic refresh, the proposed time-aware read-&-refresh can be considered as *on-demand* selective refresh. It will trigger full page write only when a page is accessed and its age is bigger than a preset threshold value. This time-aware read-&-refresh can better exploit the data access temporal locality, i.e., if a page has not been accessed recently, then likely it will not be accessed in the near future either; hence, it is not very useful to refresh this page frequently. Nevertheless, these two types of refresh operations can be possibly combined together to further reduce energy consumption. Section 5.1 shows simulations results to quantitatively demonstrate the comparison and combination of the proposed design techniques and periodic refresh.

4 Experiment Methodology

This section describes the experiment methodology being used to quantitatively evaluate the above proposed time-aware design techniques. First, by assuming cell initial resistance and drift exponent follow Gaussian distributions [6], [26], we discuss the mathematical formulation on estimating error rates. Then, we describe the system simulations platform and present ASIC implementation results of BCH decoders.

4.1 PCM Error Rate Mathematical Formulations

We first derive the optimal memory sensing resistance quantization thresholds when time-aware sensing is being used. Given resistance drift exponent ν, according to (1) we can write cell resistance in logarithmic domain as

$$\lg R(t) = \lg R(1) + \nu \lg t, \tag{2}$$

where $\lg R(1)$ is the initial resistance value at time $t = 1$ s. In the logarithmic domain, $\lg R(t)$ increases linearly with $\lg t$. For m-level per cell PCM, let $\lg R_i$ denote the resistance value in logarithmic domain of the ith storage level ($i = 1, \ldots, m$). We assume that the initial resistance $\lg R_i$ and the drift exponent ν_i follow Gaussian distributions $\mathcal{N}(\mu_{R_i}, \sigma^2_{R_i})$ and $\mathcal{N}(\mu_{\nu_i}, \sigma^2_{\nu_i})$, respectively, due to process variation. According to (2), the resistance value $\lg R_i(t_d)$ at the lifetime t_d will also follow a Gaussian distribution $\mathcal{N}(\mu_i(t_d), \sigma_i^2(t_d))$, where

$$\begin{cases} \mu_i(t_d) = \mu_{R_i} + \mu_{\nu_i} \cdot \lg t_d, \\ \sigma_i^2(t_d) = \sigma^2_{R_i} + \sigma^2_{\nu_i} \cdot (\lg t_d)^2. \end{cases} \tag{3}$$

Let $T_i(t_d)$ denote the quantization threshold between the memory cell resistance level i and $i+1$ at time t_d, we can calculate the error probability for each resistance level i, denoted as P_{raw_i}, as

$$\begin{cases} Q\Big(\frac{T_i(t_d)-\mu_i(t_d)}{\sigma_i(t_d)}\Big), & i=1,\\ 1-Q\Big(\frac{T_{i-1}(t_d)-\mu_i(t_d)}{\sigma_i(t_d)}\Big), & i=m,\\ 1-Q\Big(\frac{T_{i-1}(t_d)-\mu_i(t_d)}{\sigma_i(t_d)}\Big)+Q\Big(\frac{T_i(t_d)-\mu_i(t_d)}{\sigma_i(t_d)}\Big), & \text{otherwise,}\end{cases} \tag{4}$$

where $Q(x)$ is the Q-function for the standard normal distribution, defined as $\int_x^\infty \frac{1}{\sqrt{2\pi}}e^{-t^2/2}dt$. Assuming all the m memory cell resistance levels are equally possible with a probability of $\frac{1}{m}$, the total memory read BER, P_{raw}, can be represented as

$$P_{raw}=\frac{1}{m}\sum_{i=1}^{m-1}\Bigg(1-Q\Big(\frac{T_i(t_d)-\mu_{i+1}(t_d)}{\sigma_{i+1}(t_d)}\Big) + Q\Big(\frac{T_i(t_d)-\mu_i(t_d)}{\sigma_i(t_d)}\Big)\Bigg). \tag{5}$$

The optimal sensing thresholds at time t_d are those that can minimize the memory cell read BER (P_{raw}), i.e.,

$$\underset{T_1(t_d),\ldots,T_{m-1}(t_d)}{\operatorname{argmin}}\big(P_{raw}\big).$$

Since all the items in (5) for $i=1,\ldots,m-1$ are independent to each other, minimization of P_{raw} is equivalent to the minimization of all the items separately. Therefore, the selection of $T_i(t_d)$ only depends on the resistance distributions of the resistance level i and $i+1$, i.e.,

$$\underset{T_i(t_d)}{\operatorname{argmin}}\Bigg(1-Q\Big(\frac{T_i(t_d)-\mu_{i+1}(t_d)}{\sigma_{i+1}(t_d)}\Big)+Q\Big(\frac{T_i(t_d)-\mu_i(t_d)}{\sigma_i(t_d)}\Big)\Bigg).$$

Hence, each optimal memory cell resistance quantization threshold $T_i(t_d)$ should satisfy

$$\frac{\partial\Big(1-Q\Big(\frac{T_i(t_d)-\mu_{i+1}(t_d)}{\sigma_{i+1}(t_d)}\Big)+Q\Big(\frac{T_i(t_d)-\mu_i(t_d)}{\sigma_i(t_d)}\Big)\Big)}{\partial T_i(t_d)}=0.$$

Recall that $Q(x)=\int_x^\infty \frac{1}{\sqrt{2\pi}}e^{-t^2/2}dt$, the solution of the above equation can be derived as

$$T_i(t_d)=\frac{\mu_{i+1}(t_d)\sigma_i(t_d)+\mu_i(t_d)\sigma_{i+1}(t_d)}{\sigma_{i+1}(t_d)+\sigma_i(t_d)}. \tag{6}$$

Therefore, given the time-aware resistance level statistical distribution parameters, we can use (6) to calculate the $m-1$ optimal memory cell resistance quantization thresholds. Subsequently, we can get the optimal raw BER P_{raw}. Assuming t-error-correcting (n,k,t) binary BCH code is used, the BCH code decoding failure rate, or page error rate (PER), can be calculated as

$$PER=\sum_{i=t+1}^{n}\Bigg(\binom{n}{i}P_{raw}^i(1-P_{raw})^{(n-i)}\Bigg). \tag{7}$$

The above BER and PER calculation are also applicable when time-aware partial rewrite is being used. When conventional time-independent partial rewrite is being used, the raw BER of rewritten cells and unrewritten cells will be different, and the overall PER becomes

TABLE 1
Configurations of Hypothetical Four-Level per Cell PCM

storage level	data	$\lg R(1)$ mean	$\lg R(1)$ deviation	ν mean	ν SDMR
1	00	3.0	0.17	0.001	40%
2	01	4.0		0.02	
3	11	5.0		0.06	
4	10	6.0		0.10	

$$PER=\begin{cases}\sum_{i=t+1}^{n}\sum_{j=0}^{n\_pw}\Bigg(\binom{n\_pw}{j}P_{raw\_pw}^j\\ \cdot(1-P_{raw\_pw})^{(n\_pw-j)}\binom{n-n\_pw}{i-j}\\ \cdot P_{raw}^{i-j}(1-P_{raw})^{(n-n\_pw-i+j)}\Bigg), n\_pw\le i\\ \sum_{i=t+1}^{n}\sum_{j=0}^{i}\Bigg(\binom{n\_pw}{j}P_{raw\_pw}^j\\ \cdot(1-P_{raw\_pw})^{(n\_pw-j)}\binom{n-n\_pw}{i-j}\\ \cdot P_{raw}^{i-j}(1-P_{raw})^{(n-n\_pw-i+j)}\Bigg), n\_pw> i,\end{cases} \tag{8}$$

where n_pw is the number of rewritten PCM cells and P_{raw_pw} is the raw BER of the rewritten PCM cell. P_{raw_pw} can be represented as

$$P_{raw\_pw}=\frac{1}{m}\sum_{i=1}^{m-1}\Bigg(1-Q\Big(\frac{T_i(t_d)-\mu_{R_{i+1}}}{\sigma_{R_{i+1}}}\Big) + Q\Big(\frac{T_i(t_d)-\mu_{R_i}}{\sigma_{R_i}}\Big)\Bigg). \tag{9}$$

This work uses 2-bit/cell PCM as a test vehicle. We assume that the initial resistance $lgR(1)$ of all the storage levels are equally spaced and follow the same Gaussian distribution. The resistance drift exponent ν also follows Gaussian distribution $\mathcal{N}(\mu_\nu,\sigma_\nu^2)$. Table 1 lists all the configurations. As demonstrated in recent experimental studies [13], a larger initial resistance tends to have a larger μ_ν and σ_ν. Hence, as shown in Table 1, we assign monotonically increasing μ_ν for all the storage levels and assume that all the storage levels have the same standard deviation to mean ratio (SDMR) so that the absolute value of σ_ν increases as μ_ν increases.

4.2 Evaluation Platform

This work considers the use of PCM as either solid-state disk or disk cache: 1) *PCM-based solid-state disk*: In our simulation, we set the data retention limit as 20 years. Assuming the page length is 4k-byte and the target PER is below 10^{-14}, based upon the PCM cell device parameters as listed in Table 1, we have that a binary (38112, 32768, 334) BCH code should be used when time-aware sensing is being used. 2) *PCM-based disk cache*: Using PCM as disk cache can be much more complex than using PCM as solid-state disk. To reduce the disk transactions and leverage the nonvolatility of PCM, write-back caching policy should be used. Under normal disk operational condition, the dirty cache lines are written back to disk periodically due to cache line replacement. However, if the system is shut down abnormally, for example, unexpected power down, the dirty cache lines may need to stay in the PCM-based

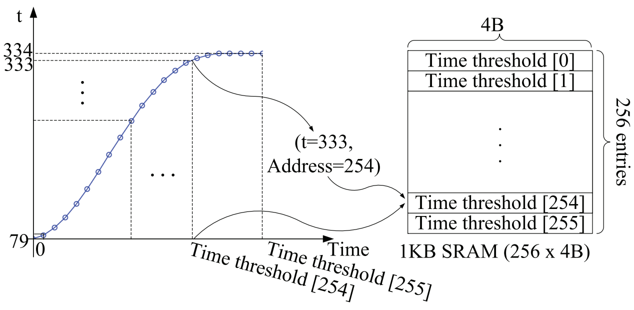


Fig. 5. LUT implementation for dynamically determining BCH decoding strength t.

disk cache for a relatively long time. Therefore, the BCH code should be strong enough to ensure the data storage integrity during the period of system shutdown. On the other hand, if disk cache does not contain any dirty data, we can largely reduce the BCH code strength to reduce latency, energy consumption, and improve disk cache effective storage capacity. This motivates us to separate disk cache to disk read cache and disk write cache, and we do not keep dirty cache lines in read cache. To keep data consistent, when a write operation hit on read cache, we simply write the data into write cache and mark the old data in read cache invalid. Assuming that dirty cache lines stay in PCM-based disk write cache no longer than 1 year, we have that the PCM-based disk write cache should be protected by a (36640, 32768, 242) BCH code; assuming that we only need to ensure the integrity of cache lines in read cache by 1 day, we have that the PCM-based disk read cache should be protected by a (34688, 32768, 120) BCH code. If we find the lifetime of a page in read cache is longer than 1 day, we simply mark this page invalid and reload the data from disk. The size of read and write cache are both 128 MB in our simulation. The cache replacement policy is least recently used (LRU).

We assume there is an external resource to feed the PCM storage system with accurate 32-bit time information including 6-bit for year, 4-bit for month, 5-bit for day, 5-bit for hour, 6-bit for minute, and 6-bit for second. Clearly, compared with the 4k-byte page size, such time information incurs completely negligible storage capacity degradation. Our disk trace files are composed of Postmark from [29], Finance1, Finance2, WebSearch1 from [30], and Trace1, Trace2, Trace4, Trace7 from [31].

As we mentioned in the above, dynamically calculating BCH decoding strength t is quite complicated. Therefore, we use a straightforward LUT-based implementation. Fig. 5 illustrates the basic concept. In our case study, the LUT is implemented as a 1-KB ROM with 256 entries and each entry is 4 bytes. The content stored in each entry is a 32-bit time threshold value. For instance, as illustrated in Fig. 5, if the time stamp from one PCM page is bigger than the time threshold value in the 253rd entry and smaller than the time threshold value in the 254th entry, the BCH decoding strength should be set to 333. Since the time threshold values in this LUT are ordered, we can simply use binary search to determine the BCH decoding strength. The searching process takes eight (i.e., $\log_2 256$) clock cycles. Using Synopsys tool set with 65-nm library, we estimate that the search only consumes 8 nJ. Compared with the BCH code decoding latency and energy consumption (as described below), both delay and energy consumption of the LUT-based implementation can be safely ignored.

To evaluate silicon and energy cost of BCH decoder, we carried out ASIC design using 65-nm CMOS standard cell and SRAM libraries, where Synopsys tools are used throughout the design hierarchy down to place and route. We set the number of metal layers as 4 in the place and route. Postlayout results verify that the decoders can operate at 400 MHz with the power supply of 1.08 V. We designed the decoder with the following configurations: The syndrome computation and Chien search blocks have a parallelism factor of 4; the error locator calculation block is fully serial and takes $t_s(t_s+3)/2$ clock cycles. The inversion-free Berlekamp-Massey algorithm [32] is used to realize error location calculation. The silicon area, decoding latency, and energy consumption are listed in Table 2. BCH encoding is very simple (i.e., only collections of shift registers), and consumes less than 5 percent of decoding power. Hence, we do not explicitly take it into account. Finally, according to the results presented in [6], [23], [33], [34], [35], [36], we assume that the average PCM read latency and current are 50 ns and 40 μA, respectively. The average PCM write latency and current are 150 ns and 140 μA, respectively. Therefore, the average PCM read energy is 2 pJ per cell and average PCM write energy is 21 pJ per cell.

5 Simulation Results

5.1 Time-Aware Sensing and Time-Aware Partial Rewrite

We first demonstrate how the time-aware memory sensing and time-aware partial rewrite can improve the ECC decoding performance and, hence, increase memory data retention limit. This study is done based on the above mathematical formulation, independently from specific disk traces. Fig. 6 shows the simulated PER results when different PCM sensing and partial rewrite schemes are used. When using conventional memory sensing with time-independent resistance quantization thresholds, the PER increases quickly over the time even without partial rewrite, leading to very short allowable memory content lifetime

TABLE 2
BCH Decoder ASIC Design Postlayout Results

(n,k,t) BCH Codes	Area (mm^2)	Latency (μs)	Throughput ($Gbps$)	Energy* (μJ)
$(38112, 32768, 334)$	2.03	180.7	0.23	19.643
$(36640, 32768, 242)$	1.49	114.1	0.44	5.835
$(34688, 32768, 120)$	0.77	58.5	1.78	0.448

* *Full-strength BCH decoding energy consumption (i.e., $t_s = t$ in syndrome computation and error locator calculation).*

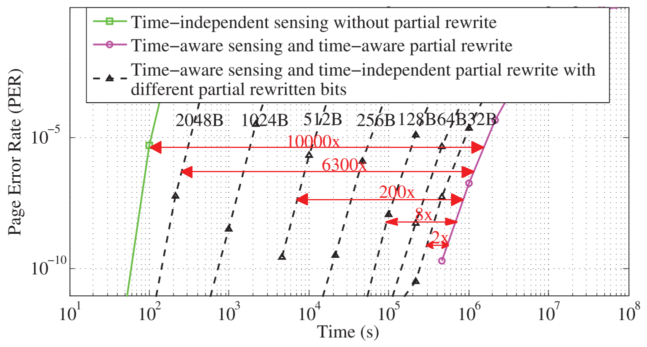


Fig. 6. Simulated BCH decoding PER under different sensing and partial rewrite schemes.

and, hence, memory data retention limit. As shown in Fig. 6, the time-aware memory sensing scheme can improve the memory data retention limit by four orders of magnitude. Regarding the use of partial rewrite in PCM, we have

- In the context of time-independent memory sensing, we can only use time-independent partial rewrite. The use of time-independent partial rewrite can slightly improve the PER performance, because the resistance of those rewritten cells have been reset to their initial value and the time-independent quantization thresholds are optimal for those rewritten cells. But all the other un-rewritten cells still suffer from inaccurate quantization threshold value as illustrated in Fig. 2b.
- In the context of time-aware memory sensing, we can use either time-aware or time-independent partial rewrite. If the proposed time-aware partial rewrite is used, it will not affect the memory sensing error rate, and hence, the PER will remain the same. However, conventional time-independent partial rewrite will increase the memory sensing error rate and, hence, degrade the PER performance. First, we assume only one partial rewrite occurs within each 4k-byte page, and we calculate the PER when we read the partially rewritten page right after the partial rewrite occurs. We consider the amount of partial rewritten bits to be 2^i-byte, where $i = 5, 6, \ldots, 11$, and Fig. 6 shows the simulated PER performance. Clearly, the more memory cells are rewritten, the PER degradation will be higher. Furthermore, we study the scenario when multiple partial rewrites occurs over the data lifetime. Fig. 7 shows the PER when multiple consecutive time-independent partial rewrites occur in one page with a fixed time interval in log domain (i.e., partial rewrite occurs at $10s, 10^2$s$, \ldots, 10^9$s, respectively), and each time 256 bytes are rewritten. As shown in Fig. 7, the effects of time-independent partial rewrite will accumulate over time and gradually degrade the PER performance compared with time-aware partial rewrite.

The above simulation results clearly demonstrate that time-aware memory sensing is very effective to mitigate the resistance drift, and the well studied partial rewrite scheme must be carried out in the time-aware manner.

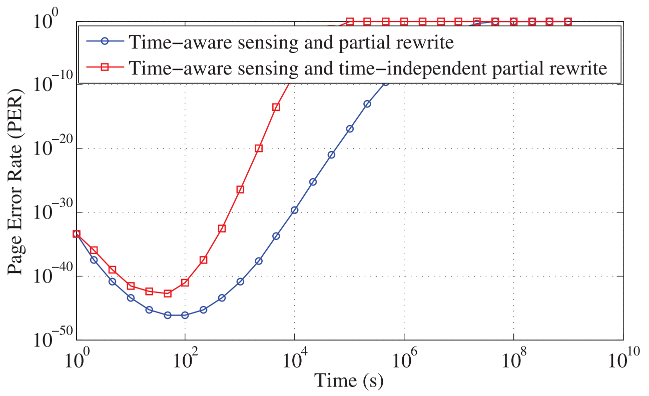


Fig. 7. Simulated PER when using time-independent or time-aware partial rewrite with multiple 256-bye partial rewrite on the same page accumulated over the time.

5.2 Time-Aware Variable-Strength ECC Decoding

Since the effectiveness of the proposed design techniques depends on the data lifetime (i.e., how long the data resided in PCM), we considered different scenarios with different data (used in the trace file) lifetime ranging from days to years. As pointed out earlier, for PCM-based solid-state disk, we set the data retention limit as 20 years; hence, we assume the disk trace files are collected when all the data have resided in the disk for 20 years, 10 years, 1 year, 1 month, 1 day, and 1 hour, respectively. PCM-based disk cache contains both write cache and read cache, where write cache data retention limit is 1 year and read cache data retention limit is 1 day. Therefore, for PCM-based disk cache, we assume the disk trace files are collected when all the data have resided in write cache and read cache for 1 year and 1 day, 1 month and 1 day, 1 day and 1 day, and 1 hour and 1 hour, respectively. In this work, we consider three energy consumption sources, including memory write energy, memory read energy, and BCH decoding energy. Regardless to the data lifetime, memory write and read energy consumption will remain the same for each disk trace; while BCH decoding energy consumption can vary significantly at different data lifetime due to the use of time-aware variable-strength decoding.

For PCM-based solid-state disk, since the BCH decoding consumes the most energy when data lifetime is 20 years, we normalize the decoding energy consumption of all the other scenarios against the case when data lifetime is 20 years. In addition, we normalize the memory write and read energy consumption with the worst-case BCH decoding energy consumption. Fig. 8 shows the normalized BCH code decoding energy, PCM write energy, and PCM read energy for various disk traces. As shown in Fig. 8, compared with the case when data lifetime reaches 20 years and, hence, BCH decoding consumes the most energy, the average disk energy consumption saving over all the eight disk trace files is 97, 95, 88, 71, and 27 percent when the disk data lifetime is 1 hour, 1 day, 1 month, 1 year, and 10 years, respectively. The results clearly demonstrate the effectiveness of the proposed time-aware variable-strength ECC decoding. As we can intuitively expect, as the data lifetime increases and, hence, resistance drift becomes more significant, the energy saving of time-aware variable-strength ECC decoding will reduce.

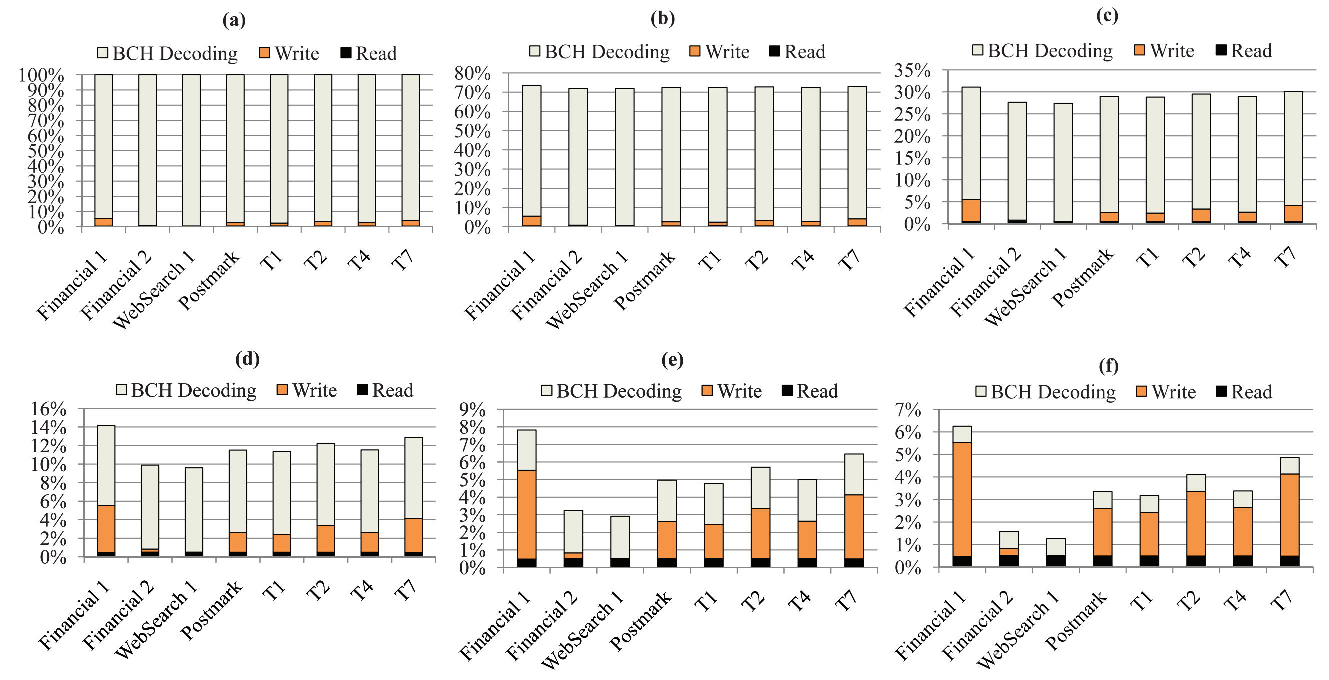


Fig. 8. Normalized energy consumptions when time-aware variable-strength BCH decoding is used for PCM-based solid-state disk, where the disk trace files are collected when all the data have resided in the disk for (a) 20 years, (b) 10 years, (c) 1 year, (d) 1 month, (e) 1 day, and (f) 1 hour, respectively.

Regarding PCM-based disk cache, as pointed out earlier, both write cache and read cache are 128 MB and we assume the use of the LRU cache replacement policy. For each disk trace file, we simulate the data access activities of both the write cache and read cache, and accordingly estimate the PCM-based cache write/read energy consumption and BCH decoding energy consumption. Similarly, for PCM based disk cache, since the BCH decoding consumes the most energy when the lifetime of data in write cache is 1 year and the lifetime of data in read cache is 1 day, we normalize the decoding energy consumption of all the other scenarios against with such a worst-case scenario. Fig. 9 shows the simulation results considering the energy of both the write cache and read cache. Compared with the worst-case energy consumption when the lifetime of data in write cache and read cache is 1 year and 1 day, the average disk cache energy consumption saving over all the eight disk trace files is 79, 66, and 48 percent for the other three data lifetime scenarios. We note that, for the benchmark WebSearch1, there is almost no energy savings when the write cache data lifetime is 1 year and 1 month, as shown in Figs. 9b and 9c, which can be explained as follows: For these two scenarios, since we assume the disk trace files are collected when all data have resided in read cache for 1 day, the BCH decoding energy savings are only from the read operations that hit in write cache. However, the benchmark WebSearch1 has extremely small amount of read operations (almost zero) that hit in write cache, leading to almost no energy savings in these two scenarios.

5.3 Time-Aware Read-&-Refresh

As demonstrated above, the effectiveness of time-aware variable-strength ECC decoding largely degrades as the data lifetime increases. To mitigate this issue to a certain extent, we can use the time-aware read-&-refresh as proposed in Section 3.2. In this study, we set the read-&-refresh data lifetime threshold T_{th} as 1 day, i.e., whenever we read a page that has resided in PCM longer than 1 day, we always rewrite the entire page. We consider both the

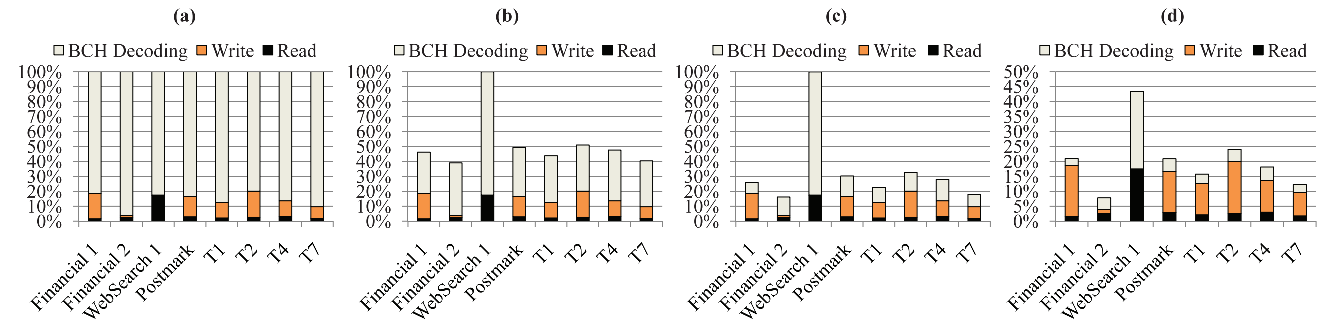


Fig. 9. Normalized energy consumptions when time-aware variable-strength BCH decoding is used for PCM-based disk cache, where the disk trace files are collected when all data have resided in write and read cache for (a) 1 year and 1 day, (b) 1 month and 1 day, (c) 1 day and 1 day, and (d) 1 hour and 1 hour, respectively.

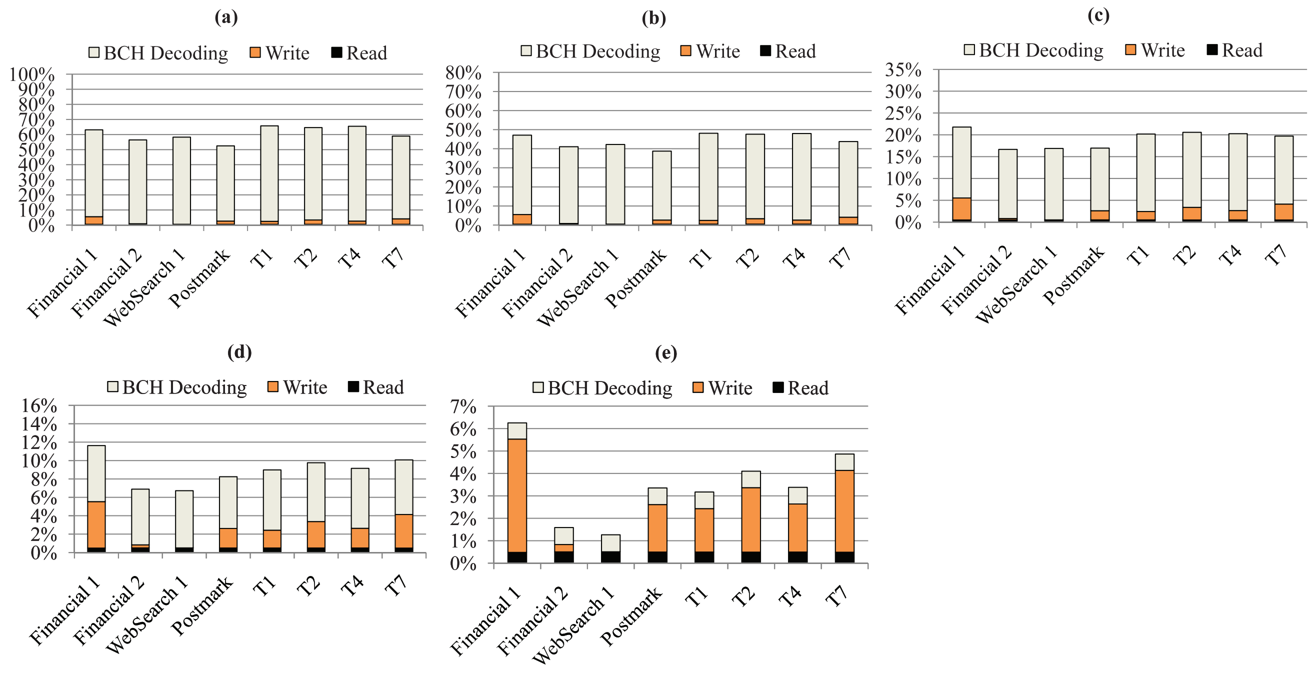


Fig. 10. Normalized energy consumptions when complementing time-aware variable-strength BCH decoding with time-aware read-&-refresh for PCM-based solid-state disk. The results in (a) ~ (d) correspond to the worst-case scenario when all the data have resided in the disk for (a) 20 years, (b) 10 years, (c) 1 year, and (d) 1 month, respectively. The results in (e) correspond to the best-case scenario where all the data stored in disk have just been refreshed (or rewritten) right before the traces are collected.

best-case and worst-case scenarios when carrying out trace-based simulations: 1) Under the best-case scenario, we assume that all the pages accessed in the traces have just been refreshed (or rewritten) right before the traces are collected. This can be considered that the traces occur when the lifetime of data stored in PCM is zero. 2) Under the worst-case scenario, we assume that all the pages accessed in the traces have never been refreshed. Fig. 10 shows the simulated energy consumption results when using PCM as solid-state disk.

All the energy consumptions are normalized against the case when all the data on the disk have 20-year lifetime and time-aware read-&-refresh is not used. Comparing the results shown in Figs. 8 and 10, we can see that, when time-aware read-&-refresh is being used together with the time-aware variable-strength ECC decoding, the average energy consumption saving over all the eight disk trace files under the worst-case scenario can increase from 0 to 40 percent, from 28 to 56 percent, from 71 to 81 percent, and from 89 to 91 percent, in the cases of 20-year, 10-year, 1-year, and 1-month data lifetime, respectively. Under the best-case scenario, the energy saving is independent on the previous data lifetime because all the data have just been refreshed, and the average energy consumption saving is 97 percent. In practice, since read-&-refresh may occur anytime on different pages, the average energy savings should be between the above presented best-case and worst-case values.

Fig. 11 shows the simulation results for PCM-based disk cache. As pointed out in Section 4.2, we set the target data retention limit for PCM-based read cache as 1 day. Therefore, the use of time-aware read-&-refresh with the data lifetime threshold of 1 day has no effects on the read cache. However, when a read operation hits in the write cache whose data retention limit is 1 year, the time-aware read-&-refresh can improve the effectiveness of the variable-strength ECC decoding as we see in the above

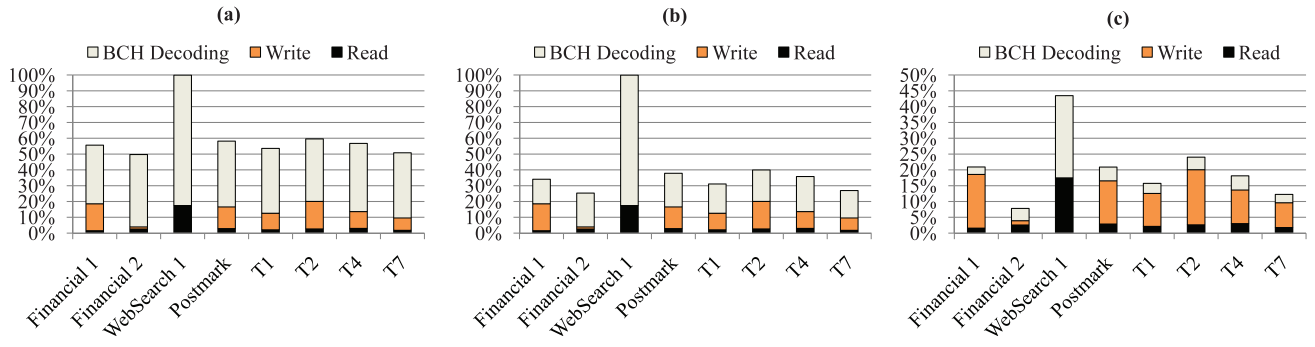


Fig. 11. Normalized energy consumptions when complementing time-aware variable-strength BCH decoding with time-aware read-&-refresh for PCM-based disk cache. The results in (a) and (b) correspond to the worst-case scenario when all the data have resided in the write and read cache for (a) 1 year and 1 day, and (b) 1 month and 1 day, respectively. The results in (c) correspond to the best-case scenario where all the data stored in disk have just been refreshed (or rewritten) right before the traces are collected.

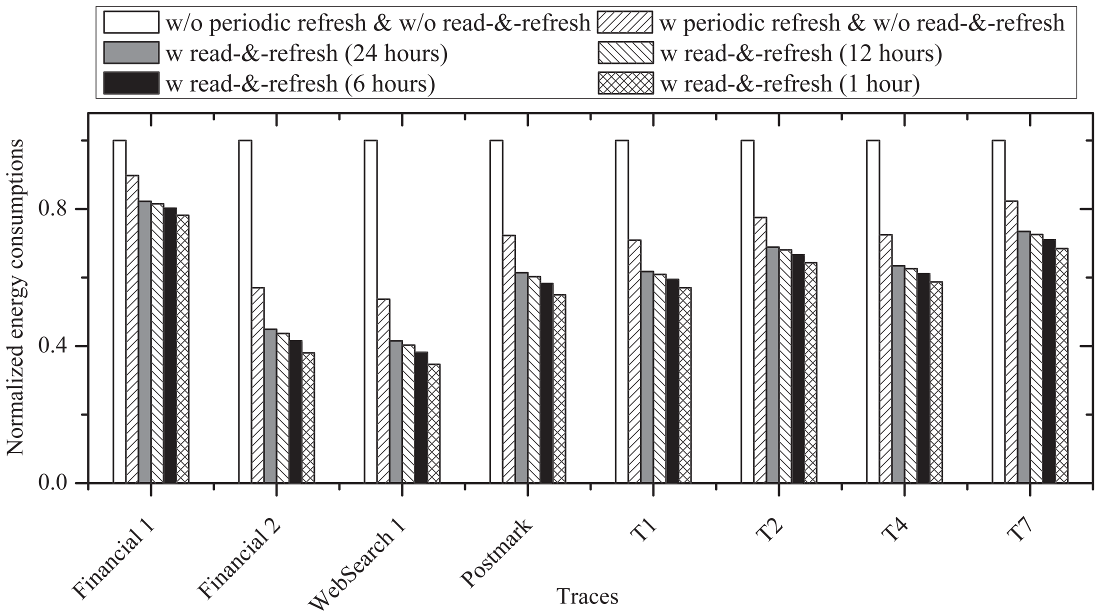


Fig. 12. Normalized energy consumptions when complementing time-aware variable-strength BCH decoding with both periodic disk scrubbing and time-aware read-&-refresh for PCM-based disk. The results are normalized to the energy consumption with time-aware variable-strength BCH decoding but without periodic disk scrubbing and time-aware read-&-refresh. Assume the disk trace files are collected when all the data have resided in the disk for 1 month.

simulations for PCM-based disk. As shown in Fig. 11, after complementing the time-aware variable-strength ECC decoding with time-aware read-&-refresh for PCM-based disk cache, the average energy consumption saving over all the eight disk trace files under the worst scenario increases from 0 to 40 percent and from 48 to 60 percent in the cases of 1-year and 1-month data lifetime, respectively. The average energy consumption saving under the best-case scenario is 79 percent. Again, due to the extremely low read hit write cache rate of the benchmark WebSearch1, there is almost no energy savings for the worst-case scenarios as shown in Figs. 11a and 11b.

5.4 Combination with Periodic Refresh

As we discussed in Section 3.2, the proposed time-aware design techniques can be combined with periodic refresh to further reduce energy consumption. The latest Intel enterprise flash-based 710 series SSD [37] can achieve maximum 270 and 210-MB/s sequential read and write bandwidth, and we assume future PCM-based disk can achieve $4\times$ higher maximum read and write bandwidth. Assume refresh operations can consistently occupy 3 percent of maximum read and write bandwidth without causing noticeable degradation of normal I/O request response time, we can estimate that it takes over two days to refresh a 2-TB solid-state disk once. We calculate average refresh energy (J/s) according to our configurations (refresh period is two days and disk capacity is 2 TB). Then, we calculate the refresh energy consumption during the trace files was executed as (execution time $\times$ average refresh energy). The simulation results in Fig. 12 show that periodic refresh can further reduce the energy consumption by 28 percent on average compared to the system with only time-aware variable-strength BCH decoding applied. Meanwhile, our proposed read-&-refresh is still able to further reduce the energy consumption by 15 percent on average based on periodic refresh.

6 Related Work

Most prior work focused on using PCM as main memory in computing systems. Zhou et al. [15] proposed a suite of hierarchical techniques, including redundant bit-write removal, row shifting, and segment swapping, for PCM-based main memory. Lee et al. [16] investigated PCM buffer organization and proposed partial rewrites to mitigate PCM's long latencies, high energy, and limited endurance. Qureshi et al. [17] proposed a hybrid PCM/DRAM main memory design strategy to combine the high capacity of PCM and high speed of DRAM. They also proposed three techniques to reduce the write traffic to PCM and two new wear-leveling techniques for PCM [18]. Qureshi et al. [19] further proposed write cancelation and write pausing techniques to improve the read performance of PCM. Zhang and Li [21] proposed cross-layer design techniques to embrace the PCM process variations when using PCM as main memory. The authors of [24], [25] proposed techniques to further improve the effectiveness of partial rewrite for PCM by using conditional bit flipping or symbol masking. None of these prior work considered the impact of PCM cell resistance drift, especially for MLC PCM. Only a recent work [22] proposed a time-aware memory sensing scheme for MLC PCM.

In contrast to these prior work, we are primarily interested in using MLC PCM as data storage such as solid-state disk and disk cache. As pointed out in the recent IBM survey paper on PCM [10], whether PCM will succeed or completely fail in the mainstream market depends on how much we can bring down the cost of PCM systems and the development of MLC PCM becomes almost inevitable. We choose disk and disk cache as our target applications, which have much relaxed requirements on speed, cycling endurance and raw device reliability. In addition, the use of large page size (e.g., 4k-byte) in these data storage applications makes it possible to employ very powerful ECC to improve the PCM technology scalability.

7 Conclusions

Recent studies show that the resistance of phase-change material tends to drift over the time with significant variability, which will gradually reduce memory storage noise margin and degrades the raw storage reliability of MLC PCM. In this work, under a time-aware design methodology, we proposed three specific time-aware design techniques to mitigate the impacts of PCM cell resistance drift when using MLC PCM to implement data storage systems. First, we proposed a time-aware variable-strength ECC decoding method, which can dynamically change the decoding strength adaptive to the data lifetime to reduce ECC decoding power consumption. Simulations show that it can reduce up to 97 percent energy savings for PCM-based solid-state disk and 79 percent energy savings for PCM-based disk cache. However, time-aware variable-strength ECC decoding suffers a dramatic effectiveness degradation when the data lifetime is approaching its retention limit. Hence, we proposed a time-aware read-&-fresh technique to further complement with it to improve the energy consumption savings. We also proposed a time-aware partial rewrite method to enable the practical use of partial rewrite, and simulation results show that, compared with time-independent partial rewrite, it can improve the achievable data retention limit by up to four orders of magnitude.

Acknowledgments

This work was supported by US National Science Foundation (NSF) grant CCF-0937794.

References

[1] F. Bedeschi et al., "A Bipolar-Selected Phase Change Memory Featuring Multi-Level Cell Storage," *IEEE J. Solid-State Circuits,* vol. 44, no. 1, pp. 217-227, Jan. 2009.

[2] R. Bez, "Chalcogenide PCM: A Memory Technology for Next Decade," *Proc. IEEE Int'l Electron Devices Meeting (IEDM),* Dec. 2009.

[3] S. Raoux et al., "Phase-Change Random Access Memory: A Scalable Technology," *IBM J. Research and Development*, vol. 52, pp. 465-479, July 2008.

[4] F. Bedeschi et al., "A Multi-Level-Cell Bipolar-Selected Phase-Change Memory," *Proc. Int'l Solid-State Circuits Conf.*, pp. 428-625. Feb. 2008.

[5] T. Nirschl et al., "Write Strategies for 2 and 4-Bit Multi-Level Phase-Change Memory," *Proc. Int'l Electron Devices Meeting (IEDM),* pp. 461-464, Dec. 2007.

[6] S. Kang et al., "A 0.1-μm 1.8-V 256-Mb Phase-Change Random Access Memory (Pram) with 66-MHz Synchronous Burst-Read Operation," *J. Solid-State Circuits,* vol. 42, pp. 210-218, Jan. 2007.

[7] H. Oh et al., "Enhanced Write Performance of a 64-Mb Phase-Change Random Access Memory," *J. Solid-State Circuits,* vol. 41, pp. 122-126, Jan. 2006.

[8] K. Osada et al., "Phase Change RAM Operated with 1.5-V CMOS as Low Cost Embedded Memory," *Proc. Custom Integrated Circuits Conf.,* pp. 431-434, Sept. 2005.

[9] A.L. Lacaita, "Phase Change Memories: State-of-the-Art, Challenges and Perspectives," *Solid-State Electronics,* vol. 50, pp. 24-31, Jan. 2006.

[10] G.W. Burr et al., "Phase Change Memory Technology," *J. Vacuum Science and Technology B,* vol. 28, pp. 223-262, Mar./Apr. 2010.

[11] A. Pirovano et al., "Low-Field Amorphous State Resistance and Threshold Voltage Drift in Chalcogenide Materials," *Trans. Electron Devices,* vol. 51, pp. 714-719, May 2004.

[12] D. Ielmini et al., "Recovery and Drift Dynamics of Resistance and Threshold Voltages in Phase-Change Memories," *Trans. Electron Devices,* vol. 54, pp. 308-315, Feb. 2007.

[13] D. Ielmini et al., "Reliability Impact of Chalcogenide-Structure Relaxation in Phase-Change Memory (PCM) Cells - Part I: Experimental Study," *Trans. Electron Devices,* vol. 56, pp. 1070-1077, May 2009.

[14] S. Braga et al., "Dependence of Resistance Drift on the Amorphous Cap Size in Phase Change Memory Arrays," *Applied Physics Letters,* vol. 94, pp. 092112-1-092112-3, 2009.

[15] P. Zhou et al., "A Durable and Energy Efficient Main Memory Using Phase Change Memory Technology," *Proc. Int'l Symp. Computer Architecture,* 2009.

[16] B. Lee et al., "Architecting Phase Change Memory as a Scalable DRAM Alternative," *Proc. Int'l Symp. Computer Architecture,* 2009.

[17] M.K. Qureshi et al., "Scalable High Performance Main Memory System Using Phase-Change Memory Technology," *Proc. Int'l Symp. Computer Architecture,* 2009.

[18] M.K. Qureshi et al., "Enhancing Lifetime and Security of PCM-Based Main Memory with Start-Gap Wear Leveling," *Proc. Int'l Symp. Microarchitecture,* 2009.

[19] M.K. Qureshi et al., "Improving Read Performance of Phase Change Memories via Write Cancellation and Write Pausing," *Proc. Int'l Symp. High-Performance Computer Architecture,* 2010.

[20] M.K. Qureshi et al., "Morphable Memory System: A Robust Architecture for Exploring Multi-Level Phase Change Memories," *Proc. Int'l Symp. Computer Architecture,* 2009.

[21] W. Zhang and T. Li, "Characterizing and Mitigating the Impact of Process Variations on Phase Change Based Memory Systems," *Proc. Int'l Symp. Microarchitecture,* 2009.

[22] W. Xu and T. Zhang, "Using Time-Aware Memory Sensing to Address Resistance Drift Issue in Multi-Level Phase Change Memory," *Proc. IEEE Int'l Symp. Quality Electronic Design (ISQED)* pp. 356-361, Mar. 2010.

[23] B.D. Yang et al., "A Low Power Phase-Change Random Access Memory Using a Data-Comparison Write Scheme," *Proc. IEEE Int'l Symp. Circuits and Systems,* 2007.

[24] W. Xu et al., "Data Manipulation Techniques to Reduce Phase Change Memory Write Energy," *Proc. Int'l Symp. Low Power Electronics and Design,* 2009.

[25] S. Cho and H. Lee, "Flip-N-Write: A Simple Deterministic Technique to Improve PRAM Write Performance, Energy and Endurance," *Proc. Int'l Symp. Microarchitecture,* 2009.

[26] M. Boniardi et al., "Statistical and Scaling Behavior of Structural Relaxation Effects in Phase-Change Memory (PCM) Devices," *Proc. IEEE Int'l Reliability Physics Symp.,* Apr. 2009.

[27] S. Kostylev, "Drift of Programmed Resistance in Electrical Phase Change Memory Devices," *Proc. European Phase Change and Ovonics Symp. (EPCOS),* Sept. 2008.

[28] R.E. Blahut, *Theory and Practice of Error Control Codes.* Addison Wesley, 1984.

[29] N. Agrawal et al., "Design Tradeoffs for SSD Performance," *Proc. USENIX Ann. Technical Conf. Ann. Technical Conf. (ATC '08),* pp. 57-70. May 2008.

[30] Storage Performance Council, http://traces.cs.umass.edu/, technical report, 2013.

[31] C. Dirik and B. Jacob, "The Performance of PC Solid-State Disks (SSDs) as a Function of Bandwidth, Concurrency, Device Architecture, and System Organization," *SIGARCH Computer Architecture News,* vol. 37, pp. 279-289, 2009.

[32] H.O. Burton, "Inversionless Decoding of Binary BCH Codes," *IEEE Trans. Information Theory,* vol. IT-17, no. 4, pp. 464-466, July 1971.

[33] K.J. Lee et al., "A 90nm 1.8V 512Mb Diode-Switch PRAM with 266 MB/s Read Throughput," *IEEE J. Solid-State Circuits,* vol. 43, no. 1, pp. 150-162, Jan. 2008.

[34] H. Oh et al., "Enhanced Write Performance of a 64Mb Phase-Change Random Access Memory," *Proc. Int'l Solid-State Circuits Conf.,* 2005.

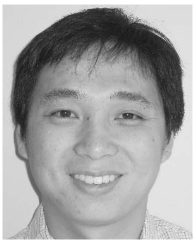
[35] S. Ahn et al., "Highly Manufacturable High Density Phase Change Memory of 64Mb and Beyond," *Proc. Int'l Electron Devices Meeting,* 2004.

[36] F. Bedeschi et al., "An 8Mb Demonstrator for High-Density 1.8V Phase-Change Memories," *Proc. Symp. VLSI Circuits,* July 2004.

[37] http://www.intel.com/content/www/us/en/solid-state-drives/solid-state-drives, 2013.

Qi Wu (M'06-SM'11) received the BS and MS degrees in electrical engineering from the Xian Jiaotong University, China, in 2003 and 2006, respectively, and the PhD degree in electrical engineering from Rensselaer Polytechnic Institute, Troy, New York in 2007. Currently, he is a principle engineer in StorCloud Inc. He is in charge of defining the architecture of next generation enterprise solid-state drive products. His research interests include circuit and architecture design for memory and data storage systems, including NAND Flash and Phase-Change Memory-based SSDs. He is a senior member of the IEEE.

Fei Sun (M'03-SM'07) received the BS and MS degrees from the Xian Jiaotong University, Xian, China, in 2000 and 2003, respectively, and the PhD degree from Rensselaer Polytechnic Institute, Troy, New York in 2007, all in electrical engineering. Currently, he is an engineering manager in storage division at Marvell Semiconductor Inc., Santa Clara, California. He is in charge of VLSI architectures and implementations for advanced NAND Flash controller IP development. He is a senior member of the IEEE.



Wei Xu (M'06-SM'09) received the BS and MS degrees from Fudan University, Shanghai, China, in 2003 and 2006, respectively, and the PhD degree in electrical engineering from Rensselaer Polytechnic Institute, Troy, New York, in 2009. He joined the Marvell Technology, Santa Clara, CA, in 2009 as a senior design engineer. His research interests include circuit and architecture design for memory and data storage systems, including phase-change memory, spin-transfer torque magnetoresistive memory, solid-state drive, and hard disk drive. He is a senior member of the IEEE.

Tong Zhang (M'02-SM'08) received the BS and MS degrees in electrical engineering from the Xian Jiaotong University, Xian, China, in 1995 and 1998, respectively, and the PhD degree in electrical engineering from the University of Minnesota, Minneapolis, in 2002. Currently, he is an associate professor in Electrical, Computer and Systems Engineering Department at Rensselaer Polytechnic Institute, Troy, New York. His current research interests include circuits and systems for data storage, signal processing, and computing. Currently, he serves as an associate editor for the *IEEE Transactions on Circuits and Systems - II*, and the *IEEE Transactions on Signal Processing*. He is a senior member of the IEEE.