A New Data Allocation Method for Parallel Probe-Based Storage Devices

Maria Varsamou[1] and Theodore Antonakopoulos[2]

[1]IBM Research, Zurich Research Laboratory, Rueschlikon 8803, Switzerland
[2]Department of Electrical and Computers Engineering, University of Patras, Rio-Patras 26500, Greece

We present a new data allocation method for probe-based storage devices that use multiple, simultaneously accessed parallel data fields. Our method uses blocks of data of unequal length for allocating a sector in the various storage fields. The amount of data stored in each field depends on the sector's offset from the beginning of the allocation round and on the storage field used. Numerical results demonstrate the storage efficiency improvement that is achieved by the proposed method. We show that this method can be applied to atomic force microscopy-based probe storage devices.

Index Terms—**Data allocation, interleaving, probe storage, Reed–Solomon codes.**

I. Introduction

THE storage requirements of portable devices have grown steadily over the past few years, and it is expected that they will continue to grow in the near future because of the greater use of audio and video in various consumer applications [1]. Flash memory is the main technology used today in consumer handheld devices and it will dominate this market for a few more years. As the demand for storage solutions with higher density and capacity grows, research into novel technologies, such as holographic storage and nanostorage, has intensified [2], [3].

Probe-based data-storage devices that use nanometer-sharp tips, similar to those used in atomic force microscopy (AFM) and scanning tunneling microscopy (STM) [4], [5], for imaging and investigating the structure of materials down to the atomic scale, are being considered for use in future ultrahigh-density devices [6]. A single tip can be used for probing and modifying on the nanoscale, various surface properties, i.e., electrical, magnetic, mechanical, or even optical properties. There is a lot of ongoing research on storing information by altering the electric properties of ferroelectric materials [7], changing the state of phase-change materials [8], or using the traditional magnetic techniques with probes [9]. One of the most promising approaches is the thermomechanical formation of indentations in thin polymer films [10]. With probe-based techniques, storage densities of over 1 Tb/in^2 have been demonstrated. However, the data rates that can be achieved by a single AFM probe are not competitive with current magnetic-recording technologies. One solution to achieve a higher total data rate is the use of arrays of probes operating in parallel [11]. In this case, each probe performs read/write/erase operations on a dedicated area, named a storage or data field, while the storage medium is placed in the x–y plane. Regarding the thermomechanical approach, a large 2-D array consisting of up to 4096 (64×64) cantilevers with integrated tips and sensors has been successfully fabricated using silicon micromachining techniques [12]. The allocation of data in the various storage fields of these devices, which are mainly known as "probe storage devices," is addressed in this paper.

Like conventional storage devices, AFM probe-based devices also experience random and burst errors. A common technique used for correcting these errors is based on Reed–Solomon (RS) codes along with proper interleaving [13]. As a result, the error bursts are spread across multiple codewords, and, consequently, the number of errors that occur within one codeword may be smaller than the error-correction capability of the RS code. To store the data in the various fields, the encoded data are split into smaller blocks, with each block being stored in a single storage area, as will be explained in the next section.

Section II analyzes the storage efficiency achieved by using the conventional data allocation technique in multiple storage fields, and demonstrates the need for a new, more efficient data allocation method. Section III describes such an enhanced data allocation method and analyzes its efficiency. This section also estimates the number of symbols stored in each field for a given sector and determines the starting address of the corresponding block of interleaved symbols. Finally, Section IV demonstrates the application of the proposed method to AFM-based probe-storage devices.

II. Conventional Data Allocation Method

The dataflow used in a probe-storage device is shown in Fig. 1 [14]. We consider the case where a sector of L symbols has to be stored in a device that consists of N storage fields. Usually, L also includes some cyclic redundancy code (CRC) symbols.

The following definitions apply:

N	number of storage fields in the device;
L	sector size + CRC;
n	$\mathrm{RS}(n,k)$ codeword size;
k	$\mathrm{RS}(n,k)$ dataword size;
r	$\mathrm{RS}(n,k)$ parity symbols;
M	number of codewords in a sector;
L_t	symbols stored in each field using the conventional method;
L_{p1}	padding symbols added in the initial sector data;
L_{p2}	padding symbols added in each field by the conventional method;

Digital Object Identifier 10.1109/TMAG.2007.915059

Color versions of one or more of the figures in this paper are available online at http://ieeexplore.ieee.org.

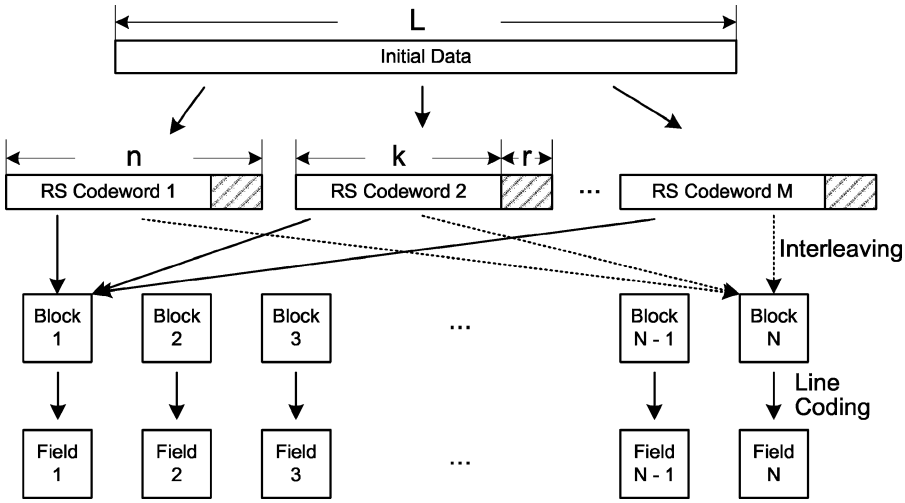

Fig. 1. Data allocation in multiple storage fields.

- k_1 number of fields in the first group of the method proposed;
- k_2 number of fields in the second group of the method proposed;
- L_1 symbols per codeword in a field of the first allocation group;
- L_2 symbols per codeword in a field of the second allocation group;
- $S_{j,i}$ physical position where sector j ends in field i;
- S_j maximum physical position where sector j ends;
- S_{total} symbols allocated in each field for a complete allocation round;
- a_{j-1} last field where S_{j-1} symbols were stored;
- s_e storage efficiency.

With previous definitions, the data are partitioned into M datawords. If k is the size of each dataword, then the number of datawords is given by

$$M = \left\lceil \frac{L}{k} \right\rceil \tag{1}$$

and $L_{p1} = k \cdot M - L$ padding symbols are added to the initial sector data.

The M datawords are then transformed into codewords using an (n, k) RS code with r parity symbols. The length of each codeword is $n = k + r$ symbols, and the codewords can be decoded correctly if up to $\lfloor r/2 \rfloor$ symbol errors have occurred in each codeword. Note that for simplicity only 8-bit symbols (bytes) are considered in this paper, but the presented analysis and method can also be applied to different symbol sizes. To deal with correlated errors and spread them over several codewords, an interleaver of depth M is used, so that M codewords are block-interleaved on a symbol basis. For a given error pattern, the device's error-correction capability depends on r, k, and M. The interleaved codewords are partitioned further and form N blocks, each of which is stored in a single storage field. Usually an additional inner line code is used before the final bits are stored in the medium. In this work, we do not consider the inner code, because it does not affect the overhead introduced by the aforementioned data allocation process. The number of symbols that are finally stored in each storage field is given by

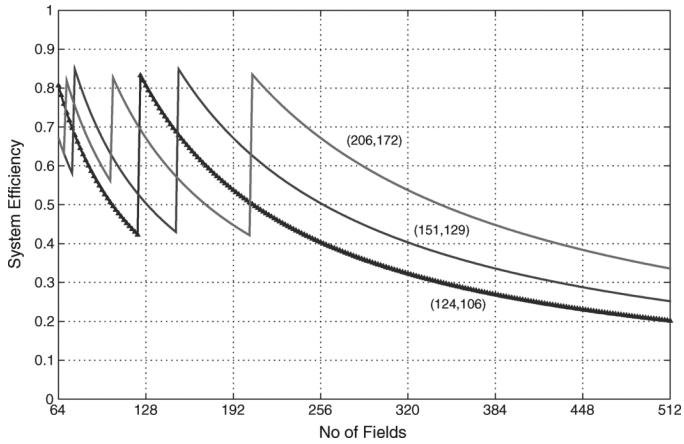

Fig. 2. Storage efficiency versus number of storage fields for various RS codes with 0.854 efficiency.

$$L_t = \left\lceil \frac{L}{k} \right\rceil \cdot \left\lceil \frac{n}{N} \right\rceil \tag{2}$$

where $L_{p2} = \lceil L/k \rceil \cdot (N \cdot \lceil n/N \rceil - n)$ padding symbols are added at the last step. Therefore, the device's overall storage efficiency is given by

$$s_e = \frac{L}{N \cdot L_t} = \frac{L}{N \cdot \left\lceil \frac{L}{k} \right\rceil \cdot \left\lceil \frac{k+r}{N} \right\rceil} \tag{3}$$

whereas the maximum storage efficiency that can be achieved for a specific device configuration (in terms of interleaver depth and RS code) is given by

$$\max s_e = \frac{L}{L + \frac{L}{k} \cdot r}. \tag{4}$$

The maximum storage efficiency is achieved only when the sector size is a multiple of the size of the dataword (zero padding symbols at the initial step), and when the codeword size is a multiple of the number of storage fields (zero padding symbols

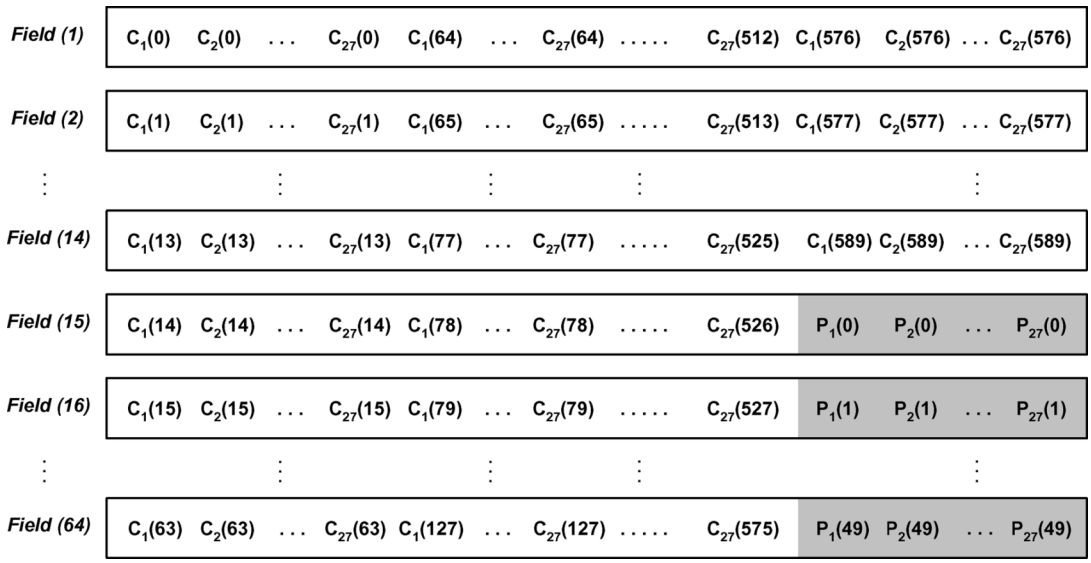

Fig. 3. Data allocation of a 16-kB sector in 64 storage fields when the RS(590,504) code is used.

at the last step). Usually, the padding needed for allocating the codewords to the various storage fields is the predominant factor that determines how close to the maximum storage efficiency a given device configuration is. As the number of storage fields increases, the use of larger RS codes is necessary for approaching the maximum storage efficiency, but the hardware complexity increases significantly [15].

As shown in Fig. 2, for a specific device and a given sector size, the device's storage efficiency is mainly affected by the RS codeword length and the number of storage fields. The maximum storage efficiency is achieved only when $\lceil n/N \rceil = n/N$, i.e., when the symbols of each codeword are spread uniformly to all storage fields, otherwise the storage efficiency may decrease dramatically. The sector size used for the results shown in Fig. 2 is 2048 bytes of data plus 4 bytes of cyclic redundancy code (CRC), while the rate of all RS codes is 0.854.

As most device interfaces use fixed sector sizes and the RS encoding/decoding is performed in specific hardware modules, the optimum device configuration is rarely met in the case of the conventional data allocation approach. This is also true for variable sector sizes and more powerful RS codes. For example, we consider RS(590, 504), i.e., a very powerful 0.854 rate RS code over $\mathrm{GF}(2^{10})$, and a 16-kB sector size that results in 27 codewords. If the storage device uses 8×8 storage fields ($L = 16384$ bytes $= 13108$ symbols (10 bits/symbol), $N = 64$ and $M = 27$), then 10 bytes from each codeword are stored in each field, and the storage efficiency becomes 0.758, because a large number of padding bytes is used. The data allocation for this specific example is depicted in Fig. 3, where $C_i(j)$ ($1 \leq i \leq 27$, $0 \leq j \leq 589$) corresponds to the jth symbol of the ith codeword and the data are stored row by row in all storage fields. The last symbol of each codeword is stored in the 14th field. To store the same number of symbols in all storage fields, a padding symbol has to be added for every codeword in each one of the remaining 50 storage fields. In Fig. 3, $P_i(z)$ ($1 \leq i \leq 27$, $0 \leq z \leq 49$) corresponds to the zth padding symbol added in the ith codeword. A total of 1350 padding symbols have to be added.

In the following section, we present a new data allocation method that can be used in devices based on multiple storage fields, operating in parallel. This method achieves almost the maximum storage efficiency for a given reliability, without imposing any restrictions on the error-correction code used, the interleaver depth, and the number of storage fields.

III. The New Data Allocation Method

Analyzing the process above described for generating the data blocks that are stored in the N storage fields, we observe that, in the final symbol-allocation round, one symbol is used per codeword in each storage field, and a large number of padding symbols may be required to complete the interleaving process in all storage fields. Consequently, as the number of padding symbols per field is M, a maximum of $(N-1) \cdot M$ padding symbols per device may be required, which is the case when the last symbol of each codeword is allocated in the first storage field.

In the new data allocation method, we still allocate the same number of symbols per codeword in each field, but we organize the data fields into two different groups, where we store a slightly different number of symbols, i.e., each field of the first group contains M more symbols than the fields of the second group. As will be explained later, the number of symbols stored in a specific field depends on the *sector number*, i.e., the sector offset from the beginning of the device's storage area. In any case, the new method always preserves the interleaver depth determined by the number of codewords specified in (1).

Let n be the size of each codeword. As the number of storage fields is independent of the codeword size, two cases are in order: $n \geq N$ and $n < N$. The following analysis is based on the first case, but the second case can also be analyzed using the same methodology.

Let L_1 be the number of symbols of each codeword that is allocated in each storage field of the first group and L_2 be the corresponding number of the second group. If k_1 and k_2 are the

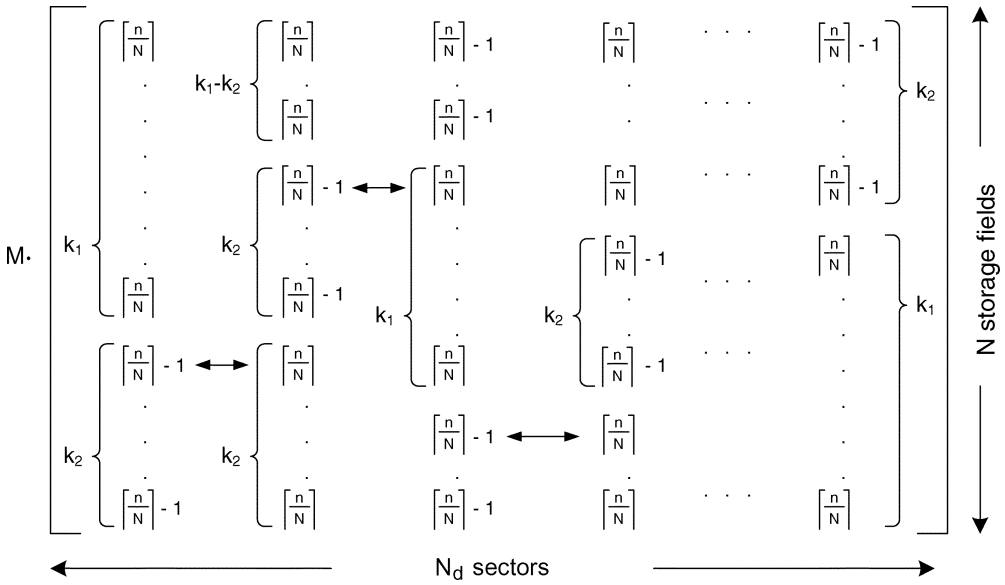

Fig. 4. The symbol-allocation matrix per codeword for a single period of allocation.

number of storage fields belonging to the first and the second group, respectively, it follows that:

$$n = k_1 \cdot L_1 + k_2 \cdot L_2 \quad \text{and} \quad N = k_1 + k_2 \tag{5}$$

where L_1, L_2, k_1, and k_2 are integers.

To avoid the use of padding, which is introduced in the final symbol-allocation round in the conventional method, we set $L_2 = L_1 - 1$. Therefore, there are k_1 storage fields, where we store $M \cdot L_1$ symbols per field per sector, whereas $M \cdot (L_1 - 1)$ symbols per sector are stored in each of the remaining k_2 fields. The integer values that satisfy (5) are the following:

$$\begin{aligned} L_1 &= \left\lceil \frac{n}{N} \right\rceil \qquad k_1 = n - N \cdot \left(\left\lceil \frac{n}{N} \right\rceil - 1 \right) \\ L_2 &= \left\lceil \frac{n}{N} \right\rceil - 1 \quad k_2 = N \cdot \left\lceil \frac{n}{N} \right\rceil - n. \end{aligned} \tag{6}$$

In probe-storage devices, as well as in conventional devices, the sectors are stored in successive positions in the medium. The starting address of each sector is determined by means of its *sector number*, s_j. The first sector, s_1, is stored at the beginning of the data storage area. In the following subsections, we determine the number of symbols per codeword that are stored in a specific field for a given *sector number* and the starting address of the corresponding block of interleaved symbols.

A. Number of Symbols per Storage Field

It is clear that storing $M \cdot \lceil n/N \rceil$ symbols in the first k_1 fields and $M \cdot (\lceil n/N \rceil - 1)$ symbols in the next k_2 fields of every sector is not an efficient approach. Therefore, to exploit the storage capabilities of all storage fields, we developed a method that determines the first field in which the $M \cdot \lceil n/N \rceil$ symbols are allocated. The first field filled with $M \cdot \lceil n/N \rceil$ symbols is the field that contains the smallest number of symbols, starting from the beginning of the storage area. If multiple fields satisfy the above criterion, it is the field that has the smallest index among them. Therefore, the number of symbols stored in a specific field depends on the respective *sector number*. More specifically, we specify that for the first sector, s_1, $M \cdot \lceil n/N \rceil$ symbols are stored in the first k_1 fields and $M \cdot (\lceil n/N \rceil - 1)$ in the remaining k_2 fields, as shown in Fig. 4. For the second sector, s_2, the first field to store $M \cdot \lceil n/N \rceil$ symbols is field $k_1 + 1$, whereas $(2 \cdot k_1 + 1) \bmod N$ is the first field where $M \cdot (\lceil n/N \rceil - 1)$ symbols are stored. For every subsequent sector, the first field to store $M \cdot \lceil n/N \rceil$ symbols is the first field in which $M \cdot (\lceil n/N \rceil - 1)$ symbols of the preceding sector were allocated. This procedure continues until all fields have been filled with the same number of symbols and a data allocation round has been completed. In this case, the last sector is stored in such a way that $M \cdot (\lceil n/N \rceil - 1)$ symbols are allocated in each of the first k_2 fields and $M \cdot \lceil n/N \rceil$ symbols in each of the remaining k_1 fields. Therefore, all N fields contain the same amount of data, and the data allocation of the next sector is performed by using $M \cdot \lceil n/N \rceil$ symbols in each of the first k_1 fields and $M \cdot (\lceil n/N \rceil - 1)$ symbols in each of the remaining k_2 fields, i.e., it uses the same allocation pattern as the first sector.

This data allocation method can be described using the symbol-allocation matrix, H, shown in Fig. 4. Each row of H corresponds to one of the N storage fields, while each column indicates how many symbols are stored in each field per *sector number*. The matrix dimensions are $N \times N_d$, where

$$\begin{aligned} N_d &= \begin{cases} N, & \text{if } N \bmod N_{\min} \neq 0 \\ N/N_{\min}, & \text{otherwise} \end{cases} \quad \text{and} \\ N_{\min} &= \min(k_1, k_2) \end{aligned} \tag{7}$$

and N_d expresses the periodicity of the data allocation process. In a complete data allocation round, the number of symbols per codeword allocated to each storage field is given by[1]

$$S = \begin{cases} k_1 \cdot \left\lceil \frac{n}{N} \right\rceil + k_2 \cdot \left(\left\lceil \frac{n}{N} \right\rceil - 1 \right), & \text{if } N \bmod N_{\min} \neq 0 \\ \frac{k_1}{N_{\min}} \cdot \left\lceil \frac{n}{N} \right\rceil + \frac{k_2}{N_{\min}} \cdot \left(\left\lceil \frac{n}{N} \right\rceil - 1 \right), & \text{otherwise.} \end{cases} \tag{8}$$

[1]If $N \bmod N_{\min} = 0$, then $N = x \cdot N_{\min}$, where x is an integer. Assuming $N_{\min} = k_1$, we can write $N = k_1 + (x-1) \cdot k_1$. But as $N = k_1 + k_2$, it follows that $k_2 = (x-1) \cdot k_1 \Longrightarrow k_2/N_{\min} = x - 1$, thus, $k_2/N_{\min}$ is an integer. Using the same approach, it can be proved that $k_1/N_{\min}$ is also an integer when $N_{\min} = k_2$.

Therefore, the number of symbols allocated to each storage field in a complete data allocation round is given by

$$S_{\text{total}} = M \cdot n \cdot \frac{N_d}{N} = \begin{cases} M \cdot n, & \text{if } N \bmod N_{\min} \neq 0 \\ M \cdot n \cdot \frac{1}{N_{\min}}, & \text{otherwise.} \end{cases} \tag{9}$$

For a given *sector number*, we determine how the symbols of the respective sector are allocated in the various fields by using the column of H that corresponds to this sector. If j is the *sector number*, the sector is allocated in data allocation round $\lceil j/N_d \rceil$ and in position c_j of the corresponding column of H, which is given by

$$c_j = ((j-1) \bmod N_d) + 1. \tag{10}$$

The first field to store $\lceil n/N \rceil$ symbols of each codeword of sector j is

$$f_{1,j} = (((c_j - 1) \cdot k_1) \bmod N) + 1 \tag{11}$$

whereas the first field to store $(\lceil n/N \rceil - 1)$ symbols of each codeword of sector j is given by

$$f_{2,j} = (f_{1,j} + k_1) \bmod N. \tag{12}$$

B. Sector-Starting Address per Storage Field

If $S_{j,i}$ is the physical position, counted in number of symbols, where sector j ends in field i, then it holds:

$$S_{j,i} = \begin{cases} S_j, & \text{if } i \leq ((j \cdot k_1 - 1) \bmod N) + 1 \\ S_j - M, & \text{otherwise} \end{cases}$$

where

$$S_j = M \cdot \left(j \cdot \left(\left\lceil \frac{n}{N} \right\rceil - 1 \right) + \left\lceil \frac{j \cdot k_1}{N} \right\rceil \right). \tag{13}$$

We check the validity of (13) initially for $j = 1$. In the first sector case, $M \cdot \lceil n/N \rceil$ symbols are stored in the first k_1 fields, while $M \cdot (\lceil n/N \rceil - 1)$ symbols are stored in the k_2 remaining fields. Obviously, (13) holds for $j = 1$. We assume that (13) holds for sector $(j-1)$, so that

$$S_{j-1,i} = \begin{cases} S_{j-1}, & \text{if } i \leq (((j-1) \cdot k_1 - 1) \bmod N) + 1 \\ S_{j-1} - M, & \text{otherwise.} \end{cases} \tag{14}$$

If a_{j-1} is the last field where $S_{j-1,i} = S_{j-1}$ symbols have been allocated, then it holds that

$$a_{j-1} = (((j-1) \cdot k_1 - 1) \bmod N) + 1. \tag{15}$$

To prove that given (14), (13) also holds for sector j, we consider the following three cases.

1) $k_1 < N - a_{j-1}$.
   An example of this case is presented in Fig. 5(a). As the first field to store $M \cdot \lceil n/N \rceil$ symbols is the first field with

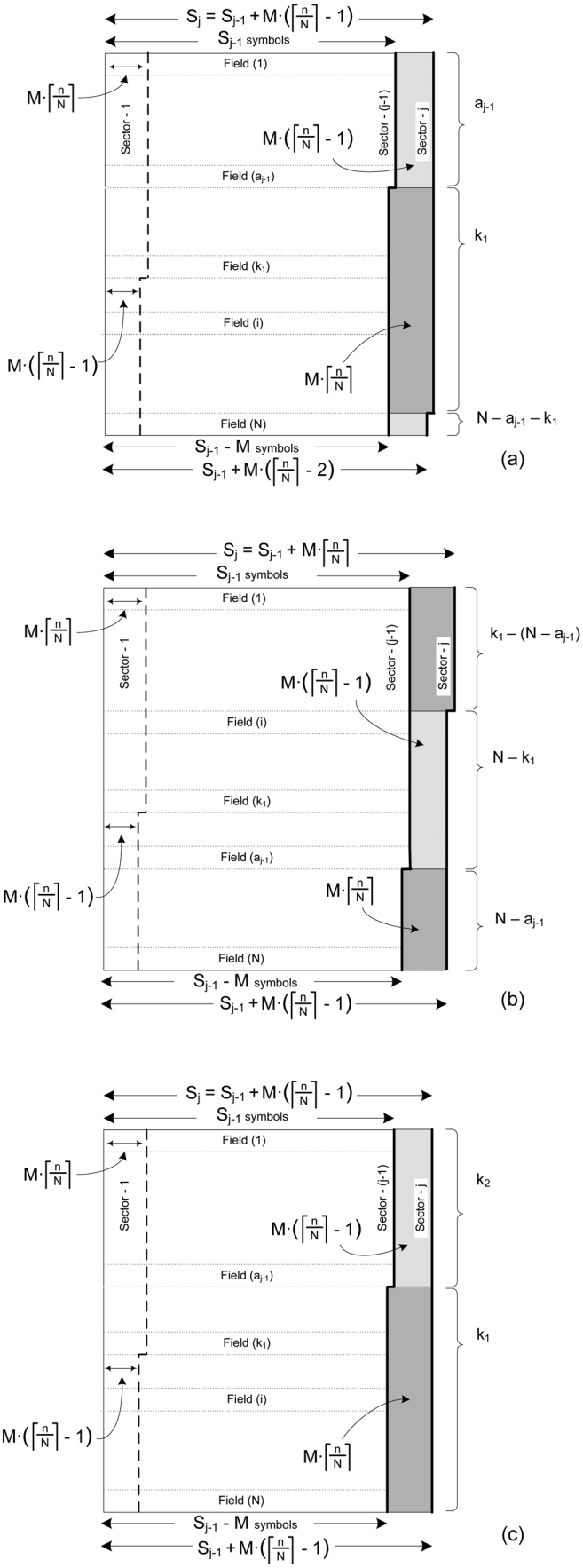

Fig. 5. Data allocation of sector j.

the fewest symbols already stored, namely, field $(a_{j-1}+1)$, then $S_{j,i}$ is given by

$$S_{j,i}=\begin{cases} S_{j-1}+M\cdot\left(\left\lceil\frac{n}{N}\right\rceil-1\right), \\ \qquad \text{if } i\le a_{j-1} \\ S_{j-1}-M+M\cdot\left\lceil\frac{n}{N}\right\rceil, \\ \qquad \text{if } a_{j-1}<i\le a_{j-1}+k_1 \\ S_{j-1}-M+M\cdot\left(\left\lceil\frac{n}{N}\right\rceil-1\right), \\ \qquad \text{if } a_{j-1}+k_1<i\le N \end{cases}$$

which can be written as

$$S_{j,i}=\begin{cases} S_{j-1}+M\cdot\left(\left\lceil\frac{n}{N}\right\rceil-1\right), & \text{if } i\le a_{j-1}+k_1 \\ S_{j-1}+M\cdot\left(\left\lceil\frac{n}{N}\right\rceil-2\right), & \text{otherwise.} \end{cases} \quad (16)$$

If $i\le a_{j-1}+k_1$ then $i\le((j\cdot k_1-1)\bmod N)+1$, and in this case it holds that $\lceil(j\cdot k_1)/N\rceil=\lceil((j-1)\cdot k_1)/N\rceil$. It can be easily proved that $S_j=S_{j-1}+M\cdot(\lceil n/N\rceil-1)=M\cdot(j\cdot(\lceil n/N\rceil-1)+\lceil(j\cdot k_1)/N\rceil)$; therefore the data of sector j that are allocated in field i end at position $S_{j,i}$:

$$S_{j,i}=\begin{cases} M\cdot\left(j\cdot\left(\left\lceil\frac{n}{N}\right\rceil-1\right)+\left\lceil\frac{j\cdot k_1}{N}\right\rceil\right), \\ \qquad \text{if } i\le((j\cdot k_1-1)\bmod N)+1 \\ M\cdot\left(j\cdot\left(\left\lceil\frac{n}{N}\right\rceil-1\right)+\left\lceil\frac{j\cdot k_1}{N}\right\rceil\right)-M, \\ \qquad \text{otherwise.} \end{cases}$$

2) $k_1>N-a_{j-1}$.
A typical example of this case is depicted in Fig. 5(b). As the first field to store $M\cdot\lceil n/N\rceil$ symbols is the first field with the fewest symbols already stored, namely, field $(a_{j-1}+1)$, $S_{j,i}$ is given by

$$S_{j,i}=\begin{cases} S_{j-1}+M\cdot\left\lceil\frac{n}{N}\right\rceil, \\ \qquad \text{if } i\le k_1-(N-a_{j-1}) \\ S_{j-1}+M\cdot\left(\left\lceil\frac{n}{N}\right\rceil-1\right), \\ \qquad \text{if } k_1-(N-a_{j-1})<i\le a_{j-1} \\ S_{j-1}-M+M\cdot\left\lceil\frac{n}{N}\right\rceil, \\ \qquad \text{if } a_{j-1}<i\le N \end{cases}$$

which can be written as

$$S_{j,i}=\begin{cases} S_{j-1}+M\cdot\left\lceil\frac{n}{N}\right\rceil, & \text{if } i\le k_1-(N-a_{j-1}) \\ S_{j-1}+M\cdot\left(\left\lceil\frac{n}{N}\right\rceil-1\right), & \text{otherwise.} \end{cases} \quad (17)$$

If $i\le k_1-(N-a_{j-1})$ then $i\le((j\cdot k_1-1)\bmod N)+1$, and it also holds that $\lceil(j\cdot k_1)/N\rceil=\lceil((j-1)\cdot k_1)/N\rceil+1$. It can be easily proved that $S_j=S_{j-1}+M\cdot\lceil n/N\rceil=M\cdot(j\cdot(\lceil n/N\rceil-1)+\lceil(j\cdot k_1)/N\rceil)$; therefore, the data of sector j that are allocated in field i end at position $S_{j,i}$:

$$S_{j,i}=\begin{cases} M\cdot\left(j\cdot\left(\left\lceil\frac{n}{N}\right\rceil-1\right)+\left\lceil\frac{j\cdot k_1}{N}\right\rceil\right), \\ \qquad \text{if } i\le((j\cdot k_1-1)\bmod N)+1 \\ M\cdot\left(j\cdot\left(\left\lceil\frac{n}{N}\right\rceil-1\right)+\left\lceil\frac{j\cdot k_1}{N}\right\rceil\right)-M, \\ \qquad \text{otherwise.} \end{cases}$$

3) Finally, if $k_1=N-a_{j-1}$, as is shown in Fig. 5(c), we deal with the special case where sector j is the last sector of a data allocation round, and it holds that $S_{j,i}=S_{j-1}+M\cdot(\lceil n/N\rceil-1)=M\cdot(j\cdot(\lceil n/N\rceil-1)+\lceil(j\cdot k_1)/N\rceil)=S_j$ in all fields. Therefore, sector j ends at the same position in all fields, that is

$$S_{j,i}=M\cdot\left(j\cdot\left(\left\lceil\frac{n}{N}\right\rceil-1\right)+\left\lceil\frac{j\cdot k_1}{N}\right\rceil\right)\quad\forall i=1,2,\ldots,N. \quad (18)$$

This equation is also the expression of (13) for the last sector of a data allocation round.

C. Efficiency of the Method Proposed

For a complete data allocation round of the presented method, the overall storage efficiency of the device is given by

$$\max ns_e=\frac{L}{n\cdot M} \quad (19)$$

which is independent of the number of storage fields and is equal to or slightly smaller than the maximum storage efficiency defined in (4). By defining the method's efficiency as the ratio

$$\frac{\max ns_e}{\max s_e}=\frac{L+\lceil\frac{L}{k}\rceil\cdot r}{(k+r)\cdot\lceil\frac{L}{k}\rceil} \quad (20)$$

it is clear that if the total sector (data plus CRC) is divisible by the size of the dataword (i.e., $\lceil L/k\rceil=L/k$), the efficiency of the presented method approaches the efficiency of the error-correcting code used.

Figs. 6 and 7 show how the storage efficiency and the device capacity are affected by the number of storage fields and the RS code used. In these figures, we consider a storage area of 100×100 μm^2 per data field, 18 nm symbol and line distance, 2048-byte sector size, 4-byte CRC, and three different error correcting codes, namely, RS(124,106), RS(151,129), and RS(206,172), which have almost the same coding efficiency. As shown in these figures, the proposed method outperforms the conventional method in all system configurations, whereas the two methods achieve the same efficiency and capacity in only a very small number of configurations. The small variations in the performance of the new method are because at the end of a line, a few symbols remain unused as there is not enough space to store a complete sector.

IV. Data Allocation in AFM-Based Probe Storage Devices

In thermomechanical probe-based storage, information is stored as sequences of indentations formed on thin polymer films using a 2-D array of AFM cantilevers/tips [10]. Each tip performs read/write/erase operations over an individual storage field. The presence or the absence of indentations corresponds to logical "1"s or "0"s, respectively. The tip-medium spacing is controlled globally, and write/read operations depend on mechanical x–y-scanning of the storage medium [16], [17]. To provide information about the motion of the micro-scanner, two pairs of thermal position sensors are used and medium-derived positioning techniques are also employed [18], [19].

Thermomechanical writing is performed by applying an electrostatic force through the tip to the polymer layer and simultaneously softening the polymer layer by local heating. The tip is heated by applying a current pulse to a micro-heater integrated

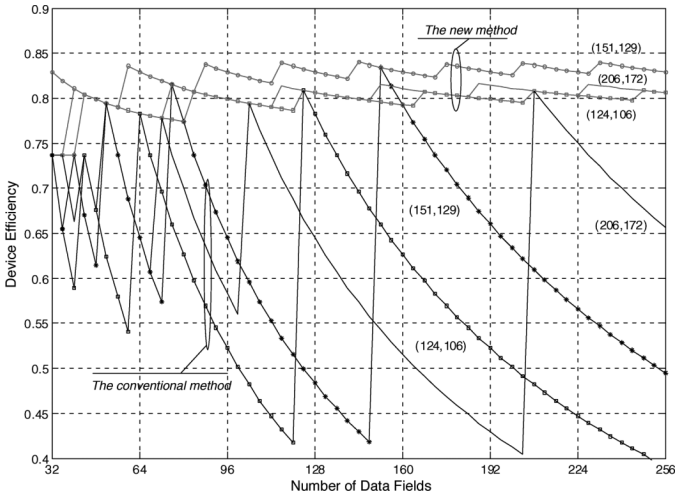

Fig. 6. Storage efficiency versus the number of storage fields and the RS code used.

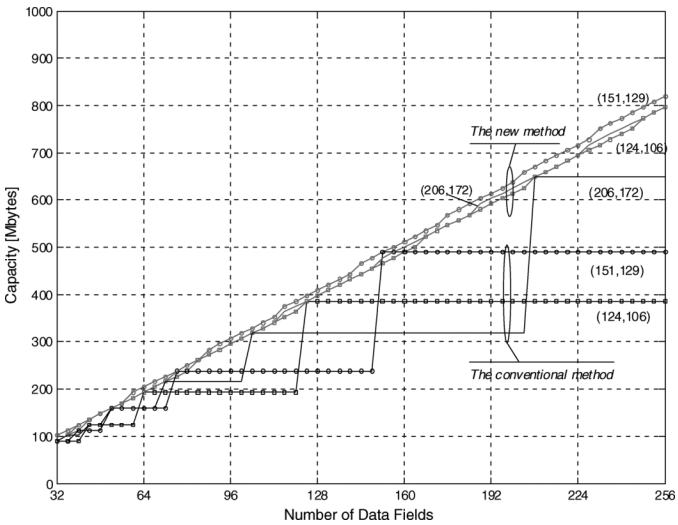

Fig. 7. Device capacity versus the number of storage fields and the RS code used.

into the cantilever directly above the tip. Reading is done using a thermomechanical sensing concept that exploits the fact that the thermal conductance between the heater platform and the storage substrate changes as a function of the distance between them. Using a second heater placed beside the tip, heat is transferred from the cantilever to the substrate through the air between them. As the tip moves into an indentation, the distance between the heater and the substrate decreases, and the heat transfer through the air becomes more efficient. As a result, the temperature of the heater changes faster and, as the electrical resistance depends on the temperature, the value of the heater resistance decreases faster when the tip moves into an indentation. Experimental results using single cantilevers have shown that data can be recorded at a density of 641 Gb/in^2 and read back with raw error rates better than 10^{-4} [20]. Furthermore, a feasibility study has shown that densities of 4 Tb/in^2 can be achieved using an advanced polymer medium [21].

Data erasing and overwriting are achieved by decreasing the pitch of writing so that the previously stored information is erased as new data is being written. It has to be emphasized

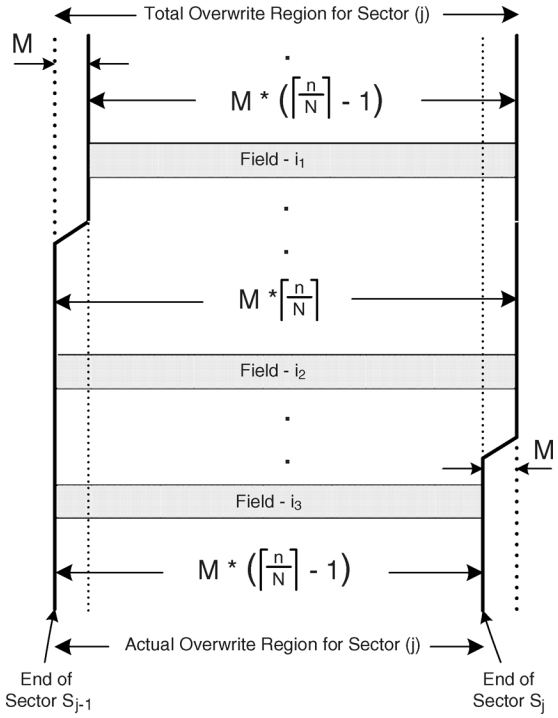

Fig. 8. Example of the data overwriting process on different storage fields.

that in order to overwrite a specific data pattern in the medium, the pattern has to be coded properly prior to applying the data-writing process. In addition, writing a block of "0"s directly to the medium will not affect the previously stored information because in this case, writing is performed by moving the tip over the data field without touching the medium and without driving the "write" circuits [22].

The new data allocation method exploits the above-described characteristics of the overwriting mechanism to perform writing/overwriting of data blocks with unequal lengths using parallel operating probes. As previously explained, the number of data of a sector allocated to the various storage fields is either $M \cdot \lceil n/N \rceil$ or $M \cdot (\lceil n/N \rceil - 1)$. The starting addresses of these data blocks are either $S_{j-1} - M$ or S_{j-1}, whereas the ending addresses are either S_j or $S_j - M$. S_j is the distance, in number of symbols, from the beginning of each storage field, and S_0 is the starting address of the first sector. As illustrated in Fig. 8, the length of the area (overwrite region) in each storage field that corresponds to a given sector takes one of two different values, as does also the starting address. The total overwrite region corresponds to the overwrite process over all storage fields for a given sector.

As the probes operate simultaneously over all storage fields, the overwriting of unequal data blocks can be achieved by using the attribute of the overwriting mechanism, i.e., that writing a sequence of zeros does not affect the previously stored information. Therefore, instead of generating overwriting data blocks having three different combinations of lengths and starting addresses, equal-length data blocks are generated by appending a block of zeros to the coded data (either at the beginning or at the end), and the overwrite process always starts at position $(S_{j-1} - M)$ and ends at position S_j. As shown in Fig. 8, a block of zeros is prefixed at the coded data of field i_1, and another block of zeros is affixed at the coded data of field i_3. As the length of the coded data in field i_2 has the maximum possible

value, no block of zeros is added to the coded data of that field. If multiple consecutive sectors have to be updated in a single write operation, the affixing of blocks of zeros is performed only at the beginning of the first sector and at the end of the last sector.

In the following example, we consider an AFM-based probe storage device that consists of 64 data fields. The length of a storage line is 100 μm, the symbol distance is 18 nm, and it uses 2048-byte long sectors. This is the maximum sector size supported by interfaces such as the Secure Digital [23] and the IEEE1394/Firewire [24]. Each sector is appended with 4 CRC bytes, and the (151,129) RS code is used for error protection. The rate of the (151,129) RS code is 0.854. The above system configuration results in 16 codewords and an efficiency of 0.6452 when using the conventional data allocation method. The partition of the initial data into 16 datawords required 12 padding bytes and the partition of the 16 interleaved codewords to the data blocks stored in the 64 storage fields required 656 additional padding bytes (16×3 coded and padding bytes are stored in each data field).

When the new data allocation method is used, the initial 12 padding bytes are still required for creating the 16 datawords, but the partition of the 16 interleaved codewords to the data blocks stored in the 64 storage fields does not require any additional padding bytes, as 16×3 bytes are stored in 23 data fields and 16×2 bytes are stored in each of the remaining 41 data fields. The data allocation round is restarted every 64 sectors. The new system configuration results in an efficiency of 0.8295, which is almost the rate of the RS code used.

V. Conclusion

The presented data allocation method can be exploited in storage devices that consist of multiple, simultaneously operating data fields. As the analysis showed, the conventional data allocation method results in a poor utilization of the device's resources as the number of data fields increases. The new method presented here uses variable-length data blocks to store a sector in the various storage fields and is independent of both the number of storage fields and the error control coding scheme used. Numerical results demonstrated the performance improvement that is achieved by the presented method. The applicability of the proposed method to AFM-based probe storage devices has been described, and improved utilization of device's resources have been demonstrated by means of an example.

References

[1] R. M. Sherwin, "Memory on the move," *IEEE Spectr.*, vol. 38, no. 5, pp. 55–59, May 2001.

[2] S. Lee and H. Bahn, "Data allocation in MEMS-based mobile storage devices," *IEEE Trans. Consum. Electron.*, vol. 52, no. 2, pp. 472–476, May 2006.

[3] G. Lawton, "Working today on tomorrow's storage technology," *Computer*, vol. 39, no. 12, pp. 19–22, Dec. 2006.

[4] G. Binnig, C. F. Quate, and C. Gerber, "Atomic force microscope," *Phys. Rev. Lett.*, vol. 56, no. 9, pp. 930–933, Mar. 3, 1986.

[5] H. J. Mamin, R. P. Ried, B. D. Terris, and D. Rugar, "High-density data storage based on the atomic force microscope," *Proc. IEEE*, vol. 87, no. 6, pp. 1014–1027, Jun. 1999.

[6] B. Bhushan, *Handbook of Nanotechnology*. Berlin, Germany: Springer-Verlag, Jan. 2004.

[7] T. Hidaka, T. Maruyama, M. Saitoh, M. Saitoh, N. Mikoshiba, M. Shimizu, T. Shiosaki, L. A. Wills, R. Hiskes, S. A. Dicarolis, and J. Amano, "Formation and observation of 50 nm polarized domains in $PbZr_{1-x}Ti_xO_3$ thin film using scanning probe microscope," *Appl. Phys. Lett.*, vol. 68, no. 17, pp. 2358–2359, Apr. 1996.

[8] H. Kado and T. Tohda, "Nanometer-scale recording on chalcogenide films with an atomic force microscope," *Appl. Phys. Lett.*, vol. 66, no. 22, pp. 2961–2962, May 29, 1995.

[9] L. R. Carley, G. Ganger, D. Guillou, and D. Nagle, "System design considerations for MEMS-actuated magnetic probe-based mass storage," *IEEE Trans. Magn.*, vol. 37, no. 3, pp. 657–662, May 2001.

[10] E. Eleftheriou, T. Antonakopoulos, G. K. Binnig, G. Cherubini, M. Despont, A. Dholakia, U. Dürig, M. A. Lantz, H. Pozidis, H. E. Rothuizen, and P. Vettiger, "Millipede—a MEMS-based scanning-probe data-storage system," *IEEE Trans. Magn.*, vol. 39, no. 2, pp. 938–945, Mar. 2003.

[11] P. Vettiger, G. Cross, M. Despont, U. Drechsler, U. Dürig, B. Gotsmann, W. Häberle, M. Lantz, H. Rothuizen, R. Stutz, and G. Binnig, "The "millipede"—nanotechnology entering data storage," *IEEE Trans. Nanotechnol.*, vol. 1, no. 1, pp. 39–55, Mar. 2002.

[12] M. Despont, U. Drechsler, R. R. Yu, B. H. Pogge, and P. Vettiger, "Wafer-scale microdevice transfer/interconnect: From a new integration method to its application in an AFM-based data-storage system," *J. Microelectromech. Syst.*, vol. 13, no. 6, pp. 895–901, Dec. 2004.

[13] J. K. Wolf, "ECC performance of interleaved RS codes with burst errors," *IEEE Trans. Magn.*, vol. 34, no. 1, pp. 75–79, Jan. 1998.

[14] T. Albrecht, T. Antonakopoulos, G. Cherubini, A. Dholakia, E. Eleftheriou, and H. Pozidis, "Writing and reading of data in probe-based data storage devices," U.S. Patent 7 257 691, 2007.

[15] M. A. Hasan, V. K. Bhargava, and T. Le-Ngoc, "Algorithms and architectures for the design of a VLSI Reed-Solomon codec," in *Reed-Solomon Codes and Their Applications*, S. B. Wicker and V. K. Bhargava, Eds. Piscataway, NJ: IEEE Press, 1994, pp. 60–107.

[16] M. A. Lantz, H. E. Rothuizen, U. Drechsler, W. Häberle, and M. Despont, "A vibration resistant nanopositioner for mobile parallel-probe storage applications," *J. Microelectromech. Syst.*, vol. 16, no. 1, pp. 130–139, Feb. 2007.

[17] A. Pantazi, M. Lantz, G. Cherubini, H. Pozidis, and E. Eleftheriou, "A servomechanism for a micro-electro-mechanical-system-based scanning-probe data storage device," *J. Inst. Phys. Nanotechnol.*, vol. 15, pp. S612–S621, Aug. 2004.

[18] M. A. Lantz, G. K. Binnig, M. Despont, and U. Drechsler, "A micromechanical thermal displacement sensor with nanometer resolution," *J. Inst. Phys. Nanotechnol.*, vol. 16, pp. 1089–1094, May 2005.

[19] A. Pantazi, A. Sebastian, G. Cherubini, M. Lantz, H. Pozidis, H. Rothuizen, and E. Eleftheriou, "Control of MEMS-based scanning-probe data-storage devices," *IEEE Trans. Control Syst. Technol.*, vol. 15, no. 5, pp. 824–841, Sep. 2007.

[20] H. Pozidis, W. Häberle, D. Wiesmann, U. Drechsler, M. Despont, T. Albrecht, and E. Eleftheriou, "Demonstration of thermomechanical recording at 641 Gbit/in^2," *IEEE Trans. Magn.*, vol. 40, no. 4, pp. 2531–2536, Jul. 2004.

[21] D. Wiesmann, U. Dürig, B. Gotsmann, A. Knoll, H. Pozidis, F. Porro, and R. Vecchione, "Ultra-high storage densities with thermo-mechanical probes and polymer media," in *Proc. Innovative Mass Storage Technologies, "IMST 2007"*, Enschede, The Netherlands, Jun. 2007.

[22] T. Antonakopoulos, E. Eleftheriou, and H. Pozidis, "Data overwriting in probe-based data storage devices," U.S. Patent 7 245 575, 2007.

[23] *SD Memory Card Specifications, Part 1, Physical Layer Specification, Version 1.0*, SD Group Std., Mar. 2000.

[24] *P1394 Standard for a High Performance Serial Bus*, IEEE Working Draft Proposed Standard, Rev. 8.0v4, Nov. 21, 1995.

Manuscript received July 5, 2007; revised December 19, 2007. Corresponding author: T. A. Antonakopoulos (e-mail: antonako@upatras.gr).