CSIT (March 2014) 2(1):65–76
DOI 10.1007/s40012-014-0047-3

ORIGINAL RESEARCH

A simple wear leveling algorithm for NOR type solid storage device

Sanjat Kumar Panigrahi · Chandan Maity · Ashutosh Gupta

Received: 31 October 2012 / Accepted: 1 July 2014 / Published online: 29 July 2014

Abstract The role and importance of solid storage devices (SSD) is rapidly increasing for the purpose of data storage. The SSDs are rapidly replacing the old fashioned and traditional magnetic storage medium. Few factors responsible for this metamorphic turnaround are due to the better performance and low power requirement of SSDs. But, as with every pros there are some associated cons. One limiting factor of SSDs in replacing traditional magnetic storage media is its low endurance. These solid state devices can be re-programmed for a limited number of times. Unlike, magnetic storage media, the SSDs can be re-programmed for a limited number of times. Internally, SSDs are organized either in bytes or blocks. Generally, for memory access operations, either the byte or blocks are used. With each write operation, the byte gets worn out and its lifetime decreases. To guard this limiting factor of SSD, wear leveling mechanism is used. Wear leveling mechanism is provided as a feature in flash type SSDs, which evenly distributes the write operation throughout the flash memory and prevents the early wear out of memory. The existing wear leveling mechanism is limited only to flash type SSDs. In this paper, we propose a wear leveling algorithm for SSD and precisely NOR type SSD. NAND types SSDs are well equipped with algorithms to enhance their life and reliability. The algorithm proposed will provide a mechanism that will enhance the life, endurance and reliability of SSD of NOR type.

Keywords Wear leveling · SSD · NOR · NAND · Endurance · Reliability · Flash

S. K. Panigrahi · C. Maity · A. Gupta (✉)
Embedded Systems Group, Centre for Development of Advanced Computing (C-DAC), Noida, India
e-mail: ashutoshgupta@cdac.in

S. K. Panigrahi
e-mail: skpanigrahi@cdac.in

C. Maity
e-mail: chandanmaity@cdac.in

1 Introduction

With growing technological advancement, it is evident that flash memory, like NAND and NOR types, has magnum opus potential to overcome the limitations with the traditional magnetic storage devices. Flash memories, like NAND and NOR, has already got a place in many a devices with the requirement of high speed, low power consumption and compact size. Most common devices with a requirement of solid storage devices (SSD) are mobile handsets, digital devices with storage requirement and to some extent in computers. One of the primary reasons of SSDs finding a place in embedded devices with the requirement of large storage space is due to their lack of any mechanical devices. As there are no mechanical devices in SSD, it is less prone to failures. SSDs have two variants, namely, NAND and NOR, both having their own different individual properties. Both the types of memory are inherently same i.e. they are composed of the same material but with different architecture and arrangement of cells. Cells are locations where a bit in the form of an electron is saved. Presence of an electron in a cell represents a bit 1 and absence of it represents a bit 0. NAND memory types are usually used where large amount of storage space is required, as NAND type memories are highly compact and have less cost per bit as compared to the NOR type memory devices. The lithography of NAND memory devices are of basically of two types i.e. Single

level cell (SLC) and Multi level cell (MLC). In SLC, only a single bit can be stored in a single cell, whereas in MLC multiple numbers of electrons can be stored. The NOR type SSD have several features that distinguishes it from its counterpart NAND type SSD. Some of them are listed below.

(1) Density: The term density means the overall storage capacity of a memory device. Generally, the NAND type SSD have a higher density as compared with the NOR type memory device. NAND memory device comes with a density of 512 Mbits to 4 Gbits where as NOR devices comes with a density of 16 Mbits to 4,096 Kbits. Hence, NAND type memory devices has better and larger density as compared to NOR type memory and therefore they have a better cost per bit i.e. cheaper than NOR.
(2) Read speed: Read speed of a memory device indicates the number of bits that can be read in a unit of time i.e. in a second. Usually, NAND memory devices have speed of 18.6 Mbytes/s whereas, NOR devices have a speed of 103 Mbytes/s. NOR type SSDs has a very high speed of reading.
(3) Write speed: Write speed of memory devices refer to the number of bits that can be written or programmed into the memory device. A NAND memory device has a write/program speed of 7 Mbytes/s whereas, NOR memory devices have a write/program speed of 0.47 Mbytes/s. In contrast to read speed of NAND and NOR memory devices, the write/programming speed of NOR devices are quite low in comparison to NAND memory devices.
(4) Erase speed: In order to write or program SSDs, erasure of the location is mandatory. Erasure of SSD means, writing/programming the location to bit 1. If prior to a write/program operation, the memory location is not erased, then the location gets erased first and then the data is being written. NAND memory devices has an erase speed of 64 Mbytes/s and NOR has 0.032 Mbytes/s.
(5) Re-write speed: Re-write speed refers to overwriting. Overwriting operation involves both the erasure and write/program operation. Typically, NAND memory device has re-write speed of 61 Mbytes/s whereas NOR memory devices have 0.026 Mbytes/s.
(6) Interface: Interface refers to the means of data access to and from the memory device. Usually, data is either accessed sequentially or randomly from a memory device. Sequential memory operation leads to a time consuming operation as compared to a random memory operation. NAND memory devices have a indirect access whereas NOR memory devices support random memory access.
(7) Asymmetric read-write access: Memory of NAND memory devices are organized in blocks and pages. The size of block and pages varies from one manufacturer to another. Most common page size is of 256 bytes and block size is of 4 pages i.e. 1024 bytes. Whereas, NOR memory devices are organized in bytes and pages, where the most common page size is of 256 bytes. Memory access in both the type of SSD is significantly different. In NAND type memory devices, memory is often accessed either in pages or blocks whereas, in NOR type memory devices is accessed either in bytes or pages. This difference in the ways of memory access operation leads to a significant time difference in the memory access operation.
(8) Wear out of blocks: NAND type memory devices are well equipped with algorithms to prolong their life cycle whereas, NOR type memory devices are not equipped with mechanisms to increase their life cycle.
(9) Application: NAND memory devices are used where the memory devices are used as data storage device with a significant amount of memory whereas, NOR memory devices are used for in program memory or execute in program (XIP). In program memory means, the program written into the NOR memory device can be run directly from the memory device in use.

In this paper our main emphasis is on the wear out problem that is encountered in SSDs and specifically in NOR type SSDs, such as EEPROMs. A wear leveling algorithm's intended purpose is to even out the distribution of writes/programming locations, so that each byte or block of the memory device is entertained equally. This even distribution of entertainment during any write/program cycle increases the life of any byte or block and hence the overall life of the memory device increases. In this paper, we define an algorithm that will evenly distribute the write/programming cycle so that the overall life cycle of the memory device will increase substantially. The primary task of the wear leveling algorithm is to increase the time required for any page or byte to reach its worn out phase. Typically, any byte or page of a NOR type memory device is at around 100 K to 1 million cycles, after this the byte or page in action will get worn out and will be out of use i.e. will be less reliable. Out of use means, the byte or page will not be able to store the data that is needed to be saved reliably. Hence, the reliability of the memory is decreased. In order to increase the reliability and life expectancy of

NOR type memory devices, an efficient wear leveling algorithm is needed.

Basically, there are two types of wear leveling that are used. These wear leveling algorithm are specifically intended for flash type memory devices i.e. NAND SSDs. Namely, these wear leveling algorithms are the following

(1) Static wear leveling: in static wear leveling algorithm, the cold data i.e. the data that are least updated are moved to a page which has been erased more number of times but hasn't reached its worn out phase. Hence, there is always an overhead of keeping track of cold data, hot data and the erased number of pages.
(2) Dynamic wear leveling: in dynamic wear leveling algorithm, pages/blocks with least erase counts are used repeatedly. The cold data are untouched and hence pages/blocks that contain the cold data have a very optimal wear out phase. This algorithm causes a great degree of unevenness in the distribution of wear in blocks or pages.

The wear leveling algorithm that we have proposed in this paper for a NOR type memory device is more inclined towards the dynamic wear levelling algorithm with some exceptions. Our goal in the paper is to give equal importance and opportunity to each and every byte/page in the entire memory of the NOR type memory device, to serve for the purpose of data storage with increasing the wear out time of the byte/page.

Most of the existing wear leveling algorithms is designed to be used with NAND type SSDs, which are abundantly used in embedded systems. However, NOR type SSDs are also used quite frequently for the purpose of limited data storage. Applications that need few amount of memory space and regular data updating, find NOR type data storage devices more suitable as compared to NAND type SSDs. Reducing the wear out time for NOR type SSDs is also quite necessary and important as like in NAND type SSDs. With this motivation we have tried to devise an algorithm that has the capability to increase the life expectancy and reliability of NOR type SSD substantially and efficiently.

By carefully reading the existing wear leveling algorithms [1, 2, 7] we have made some observations. First, one of the most important aspects of wearing leveling algorithm is in taking care of hot and cold data [8]. If the hot data are update on few limited numbers of pages then it is quite obvious that the particular pages will get worn out quickly as compared to the pages containing cold data. Ultimately, this process will increase the unevenness of wearing out of pages in the memory. In order to achieve a balanced wearing of pages the placement of hot and cold data is very much required and needed. Second, an easy approach is to keep the hot data in less worn out page and cold data in more worn out page. Third, most of the wear leveling algorithm primarily targets to reduce the wearing difference of blocks/pages throughout the lifetime of the memory device. These wear leveling algorithm in doing their intended task leads to an overhead of keeping track of a number of parameters. These parameters are the number of writes on each page/block, location of hot/cold data, number of access to each page/block, maintaining a flash translation layer (FTL) [7] and the location of wear out page/block. These overheads can lead to degradation of the overall system performance for small embedded systems where code memory and power is very limited.

2 Related work

As aforementioned, there are many existing algorithms that are used for wear leveling to enhance the life expectancy of SSDs, they fall broadly under two categories- static and dynamic wear leveling. Due to the simplicity and easy implementation of dynamic wear leveling, it is used widely. The simplicity of dynamic wear leveling lies in its way of storing data in pages/blocks that are least worn out. Hence, there is not very much of overhead incurred. One approach used by Atmel Corporation [3] is having two O buffers for storing a pointer that point to the last written location. This increases the overload as the buffer has to be saved either in the RAM of the (microcontroller) MCU or in the flash itself. Preferably, the buffer is stored in the flash as a reset will erase all the information required about the pointer to the last page will be lost. The buffer information that has been stored also needs to be wear levelled in order to increase the life expectancy of the overall NOR storage media.

Chang proposed a static wear leveling technique in which a bit erase table (BET) is maintained. A BET is a buffer which contains the number of erase cycle encounter by a page/block. Whenever a block is erased its corresponding bit is set in the BET. This BET buffer is used to determine how the wear leveling algorithm is to be used. In Chang's [9] proposed idea the movement of cold data is very crucial and important, as cold data are least supposed to get modified and this results an overhead.

Agrwal [10] proposed another novel idea of balancing tradeoffs between cleaning of blocks and wear leveling. In his proposed idea the blocks with higher wear out parameters are least recycled. In order to justify which block has a higher wear out value, a buffer has to be maintained, which again creates an overhead.

3 Wear leveling algorithm for NOR SSD

In this section we will describe the wear leveling algorithm that can be used with any type of SSD. Specifically with SSDs that are not facilitated with wear leveling algorithm to enhance their life expectancy. The design objective of this wear leveling algorithm is to facilitate any SSD with the feature of wear leveling with minimum overhead, high performance and efficiency.

The wear leveling algorithm for any NOR type SSD is explained using a system and mechanism consisting of a master device such as a microcontroller, a NOR type SSD such as an Electrically erasable read only memory (EEPROM) and data fetching sensor such as a temperature sensor or humidity sensor or vibration sensor etc. The master microcontroller controls the slave devices i.e. the data storing and sensing parts. The sensing part gathers the information which it is supposed to gather and passes it to the microcontroller. As, data gathered is required to be saved, the microcontroller transfers the data to the storage device.

The procedure of data storage depends on how the internal memory is organized within the storage device. The storage device in our system is a NOR type SSD where the memory is organized into bytes and pages of 256 bytes. Data can be written sequentially or randomly either one byte at a time or one page i.e. 256 bytes.

When the data is stored sequentially with one page at a time i.e. 256 bytes, the write process starts from the page number specified. If further data are required to be written, then the next write starts from the next adjacent page of the pervious page. In this way the data is written up to its maximum limit.

The aforementioned scenario is described for a single iteration. If further operation are needed to be done by the system and the data are required to be stored in the SSD, then the write process will again start form the first page, if no start page number is specified.

The proposed technique can be implemented in two ways: wear levelling during byte wise sequential write operation and wear levelling during page wise sequential write operation.

(A) Wear levelling during byte wise sequential write operation: In byte wise operation, data is written one byte at a time at the provided address location A of address size of N bits. The NOR SSD has a page size of P bytes with a total memory of M bits. Data is fetched from the data fetching module such as a temperature sensor or humidity sensor or vibration sensor etc. The fetched data is stored at a temporary location of the RAM in the microcontroller. The stored data is then written on to the NOR SSD using any data communication protocol such as Inter Integrated Communication (I2C) or Serial Peripheral Interface (SPI) or Universal Synchronous Asynchronous Receive Transmit (USART) etc. Data is written at the location provided. This location is very important.

In conventional write process, whenever a new write process is started it is always started from the first byte of the memory. As, whenever a new start of write is started, the master i.e. the microcontroller gets re-started and write process is started from the beginning.

The process proposed in this paper is look and start writing. If the device is performing its intended task for the first time in its life then the write starts from the first byte of the memory i.e. the 0th location i.e. $A = 0$. Further byte wise write operation are performed by incrementing the byte address by one. The byte wise write operation continues up to a predefined memory location, i.e.

$$\left(\left(2^N/8\right)-P\right)bytes \tag{1}$$

where $2^N/8$ is the total number of bytes contained in the NOR type SSD in use.

The last remaining bytes are used as a reference table that the wear levelling algorithm uses for write process in next iteration. The last look up page is also used for reading the data from the SSD.

For implementing wear levelling the number of bytes written to the SSD and the byte address from where start of read has to be performed is required to be stored in a reserved page. The reserved page is the last page of the SSD. Whenever one complete operation is performed the last page is updated with the start address START_OF_READ of 3 bytes, from where the data is read, START_OF_WRITE of 3 bytes i.e. the memory address from where the write process got initiated and the total number of bytes NO_OF_BYTES written on to the SSD which is needed to read the number of bytes. To write the START_OF_READ and START_OF_WRITE, we need N bits, where N is the number of bits containing in the maximum address value of the SSD. To write the NO_OF_BYTES, the maximum number of bits required is

$$\log_2(M\,bits/8\,bits). \tag{2}$$

The above three information i.e. START_OF_READ, START_OF_WRITE and NO_OF_BYTES are updated every time the device is operated for storage.

During byte wise write operation the byte address where data is being updated, is always checked

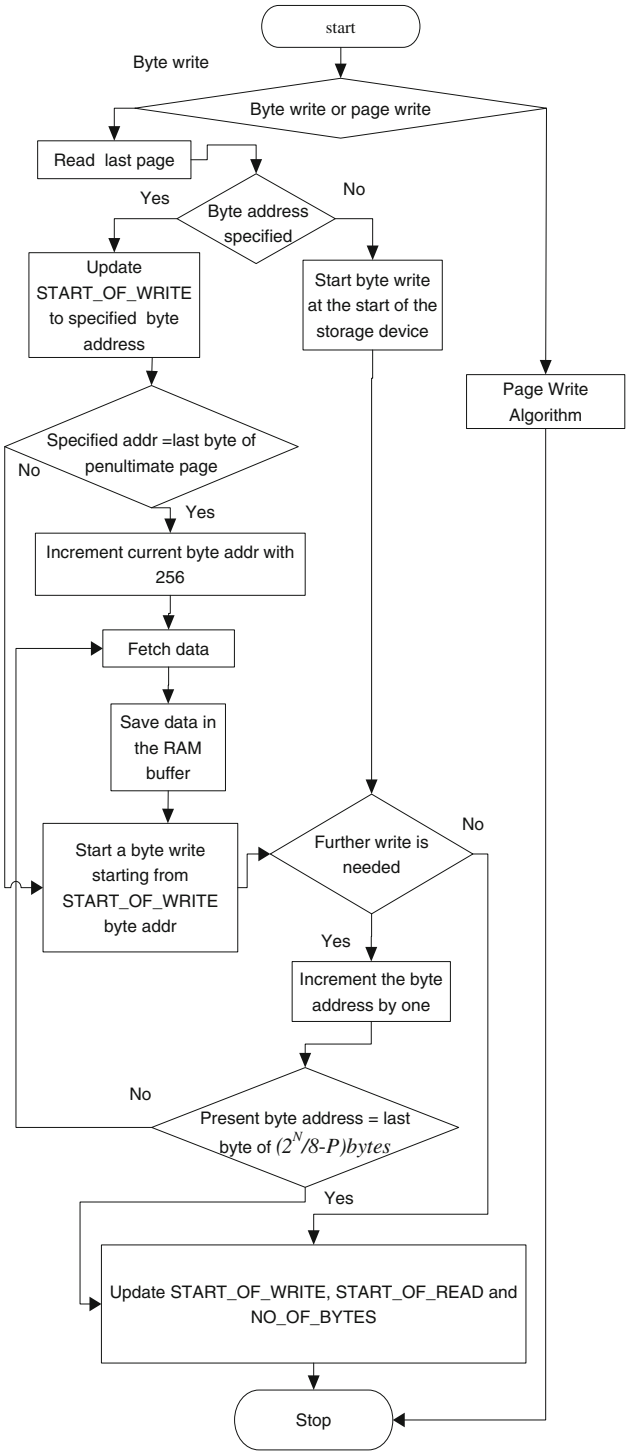


Fig. 1 Byte wise write operation

whether it has reached its limit or not. If the maximum byte write limit is reached i.e.

$$\left(2^N/8\right) - P \text{ bytes} \tag{3}$$

then, the write process is stopped. The termination of write process enables the last reserved page to maintain its integrity as well as prevents the rollover of byte address of the SSD. The roll over will create a situation which will garble the data stored in the SSD. While writing data byte wise, the byte address is always prevented to get incremented to the $2^N/8$-P location. While writing data on to the SSD if the $2^N/8$-P location is reached then the byte address is added with 256. This prevents the last reserved page not to be updated with the value fetched from the data sensing module (Fig. 1).

The wear levelling algorithm updates the START_OF_READ during the start of a new data logging process. The START_OF_READ is updated with the following value:

$$START\_OF\_READ = START\_OF\_WRITE \tag{4}$$

After the completion of an iteration of data logging, the START_OF_WRITE and NO_OF_BYTES of 3 bytes values is updated with the following values

$$START\_OF\_WRITE = last\ byte\ address + 1 \tag{5}$$

(B) Wear levelling during page wise sequential write operation: the way data is written byte wise on to the NOR type SSD, aforementioned, data can also be written a single page at a time. The process involved in writing data page wise sequentially is

$$NO\_OF\_BYTES = the\ total\ no.\ of\ bytes\ written. \tag{6}$$

similar to the above mentioned process with a difference. The difference is, in the number of data bytes that can be written with a single write command and the start of the write address.

During the start of writing data page wise sequentially on to the NOR type SSD, the last page is read in order to fetch the value of START_OF_PAGE_READ, START_OF_PAGE_WRITE and NO_OF_PAGES.

If the write operation is done for the first time, the start of data write takes from the first page of the NOR type SSD memory. Page size can vary from vendor to vendor. Usually, page sizes are of 64 or 128 or 256 bytes. The most common page size is of 256 bytes. In the paper proposed, we are using a page size of 256 bytes.

The NOR type SSD has maximum memory of M *bits* and the maximum page size is of P *bytes,* then the total number of pages *TOTAL_NO_PAGES* are

$$TOTAL\_NO\_PAGES = (M/8)/P \tag{7}$$

A page write can't be performed until and unless there are data of a full page size. Hence, in order to perform a page write, a page full of data must be collected. A page write is performed by providing the start of page address

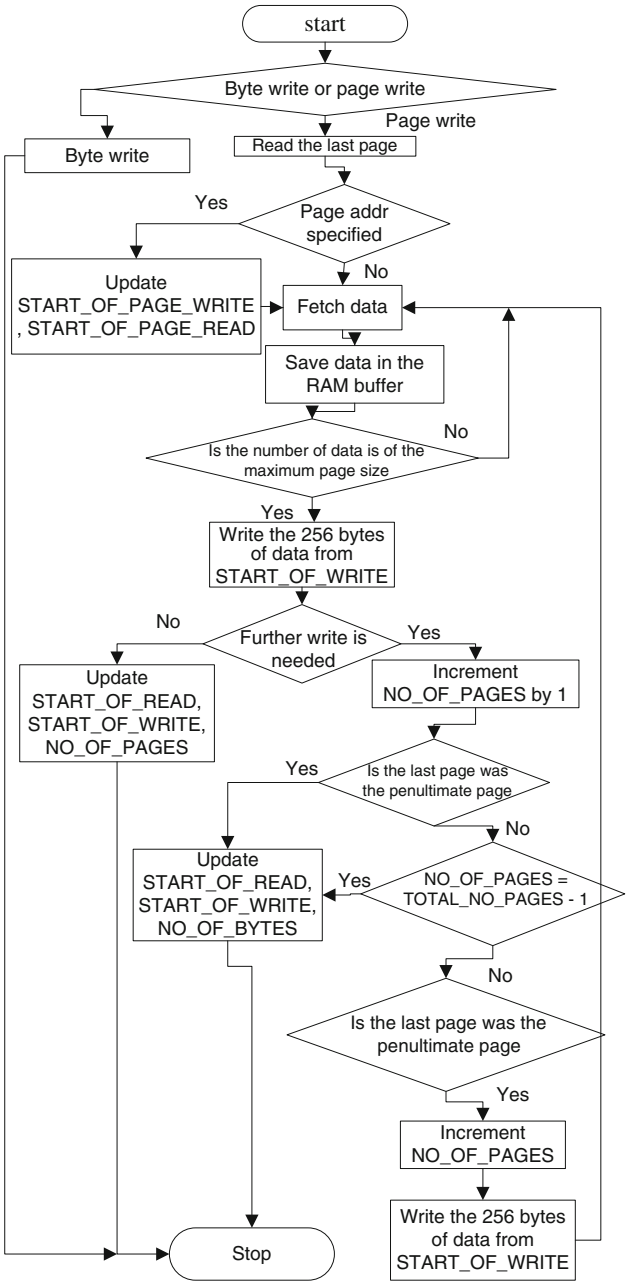


Fig. 2 Page wise write operation

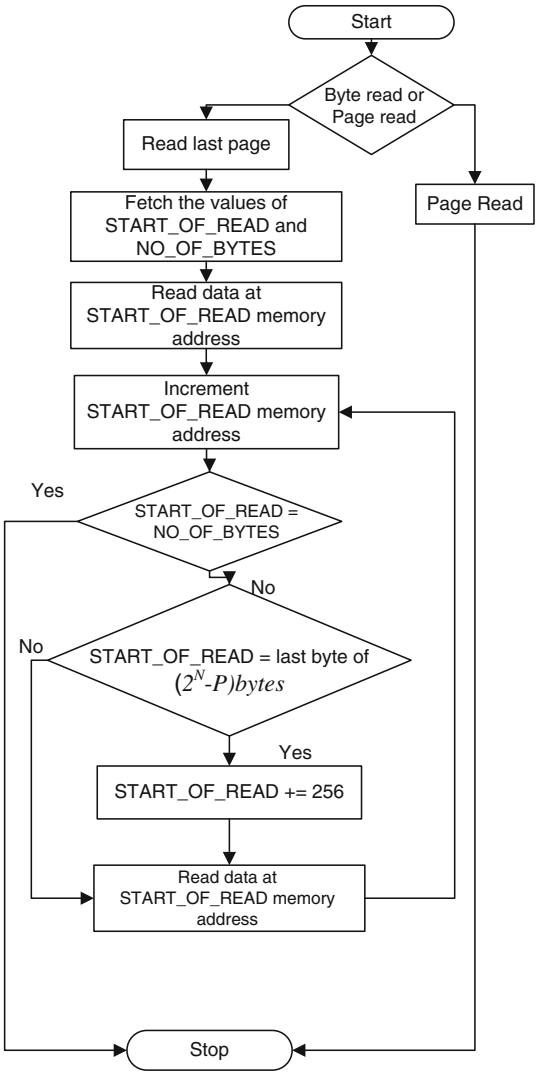


Fig. 3 Byte wise read operation

and data of a page size. Whenever, more than a page of data is to be written, collect a page full of data and provide the next page address. The start of a page address is always a multiple of its size i.e.

$$Start\ of\ Page\ address\ =\ P\ bytes \times n \tag{8}$$

Where, $0 \leq n \leq$ Maximum number of pages.

During multi-page write process, it is always checked whether the penultimate page has been reached or not. If the penultimate page is reached than the write process is stopped, else the write process continues till the logging operation is required.

Prior, to the end of an operation, the START_OF_PAGE_READ is updated with the previous START_OF_-PAGE_WRITE i.e.

$$START\_OF\_PAGE\_READ = START\_OF\_PAGE\_WRITE$$

START_OF_WRITE is updated with the next page address i.e.

$$START\_OF\_PAGE\_WRITE = NO\_OF\_PAGES + 1$$

The NO_OF_PAGES is also updated. These three values are then written byte wise on the last page form the first byte.

Whenever a new write is initiated, the last page is read. The address of the last page is *LAST_PAGE_ADDR*

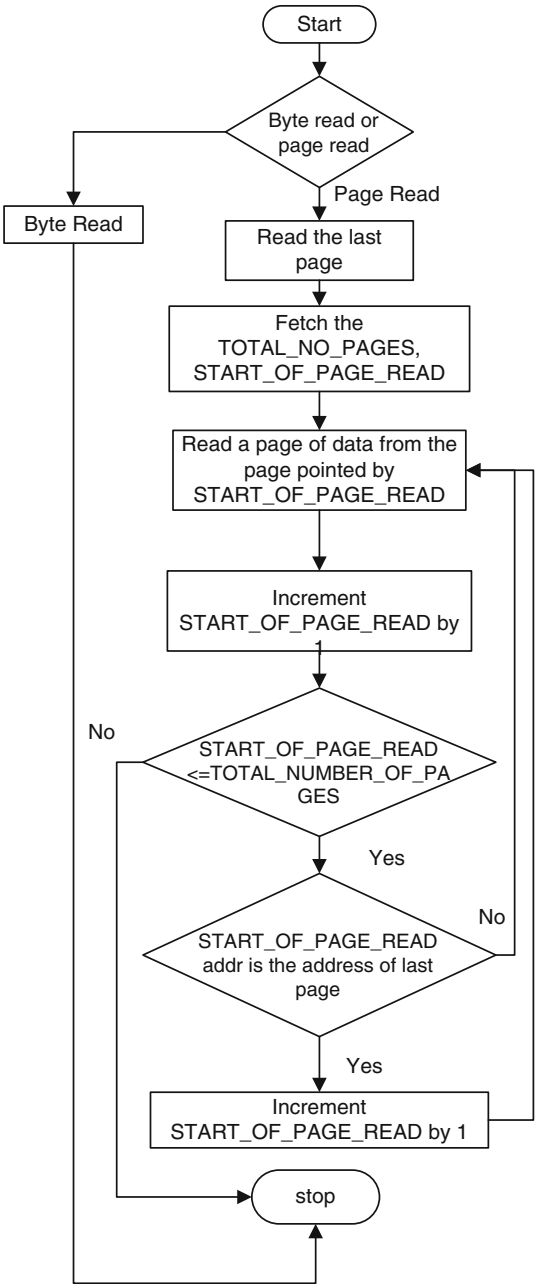


Fig. 4 Page wise read operation

$$LAST\_PAGE\_ADDR = M/8 - P \quad (9)$$

From the last page, the START_OF_PAGE_WRITE value is fetched and from this address the write process is started.

If, the LAST_PAGE_ADDR is the page address of the last page, which is treated as a reserved page, for storing the last write information then, the page address is changed to the first page of the NOR type SSD memory. Figure 2 explains the whole process.

The proposed wear levelling algorithm has to read the stored data as efficiently as it has written the data on to the NOR type SSD. The wear levelling algorithm requires two different mechanism for writing data on to the SSD i.e. byte write and page write, similarly it requires two different ways for reading data from the SSD, i.e. byte wise sequential read and page wise sequential read.

(A) Byte wise sequential read: In order to read the stored data from the NOR type SSD, the last page has to be read. From the last page, the START_OF_READ and NO_OF_BYTES have to be read. These two values are required and used to read the data from the NOR type SSD. START_OF_READ value is the address from where the read process will start and ends when the total number of data read is equal to the NO_OF_BYTES (Fig. 3).
Fetching of data starts from the START_OF_READ address and continues till the number of bytes read is not equal to NO_OF_BYTES. While reading the data, if the START_OF_READ address increments to the last byte address of the penultimate page and the number of bytes read is not equal to the NO_OF_BYTES, then the START_OF_READ is incremented by 256 i.e. the last page is skipped and the START_OF_READ points to the starting address of the first page. This is done in order to prevent the last page, treated as a reserved page, is skipped from being read.

(B) Page wise sequential read: While reading the data from the SSD page wise, the last page has to be read. From the last page, the START_OF_PAGE_READ and TOTAL_NO_PAGES are fetched, which are used to read the data from the NOR type SSD (Fig. 4).

Fetching of data starts from the START_OF_PAGE_READ address and continues till the total number of pages read is not equal to TOTAL_NO_PAGES. If, while reading data from the SSD, the START_OF_PAGE_READ address reaches the last page but the total number of pages read is not equal to TOTAL_NO_PAGES, then the START_OF_PAGE_READ is incremented by one. This is done in order to read all the written pages except the last page, which is treated as a reserved page.

4 Results

The wear levelling algorithm evenly distributes the wearing out of bytes, pages and block of SSD. Due to the even distribution of wearing of the memory, the endurance and reliability of the SSD increases significantly. In order to implement such enhancement, certain overhead is incurred. This overhead injects a little bit of inefficiency on the overall performance of the algorithm. Wear levelling

algorithm is broadly classified into dynamic and static. Both types of the wear levelling algorithm treat the data as either cold data or hot data. All the present wear levelling algorithms move the cold and hot data in a judicious way to even out the wearing of memory cells. The movement of hot and cold data requires a look up table. The look up table contains address of pages or blocks which are least and most worn out. Hot data are moved to the least worn out pages and cold data are moved to most worn out pages or blocks. The wear levelling algorithm traverses the look up table and identifies the page or block address from where data is required to be moved in our out. This movement of hot and cold data, traversing the look up table and regular updating of the look up table creates a lot of overhead and performance degradation of the overall wear levelling algorithm.

The above mentioned overhead and performance degradation motivated us to propose a new wear levelling algorithm. The algorithm proposed in this paper is very simple and efficient algorithm, with less incurred overhead and no hindrance in efficiency. Prior to any write or read process, the last page of the SSD is read to fetch data for further operation. The concept of cold and hot data is not being entertained in the proposed algorithm. Hence, the requirement of data movement is eradicated. The only task to be done is to update the start point from where data is to be written and read from and the total number of bytes or pages written on to the SSD.

In a conventional write on to a NOR type SSD, the start is always from the first byte, if a byte write is needed, or from the first page, if a page write is done. This conventional process, lead to the early wearing out of the first few pages.

Two common cases for conventional write process

(A) Byte write: There are total N bytes in a NOR type SSD. The data to be written is N′ number of bytes. The N′ number of bytes are written on to the first N′ number of byte locations, where $N' \ll N$. Hence, the conventional write process decreases the endurance and reliability of the NOR type SSD. The defect occur in the SSD is difficult to find, because, for each wrote byte, a write and read to that location is needed and verified.

(B) Page write: In conventional page write process, page write process starts from the first page and sequentially increases to the last page. Due to this, the probability of wearing of first few pages is very high.

By using the proposed wear levelling algorithm, described in the paper for NOR type SSD, the wearing out of initial bytes and pages is reduced significantly. If the N′ number of bytes/pages to be written on to the N number of total bytes/pages is evenly spread out then the life expectancy of the bytes increases by a percentage of (Eq. 10)

$$(((N/N') \times L - L)/L) \times 100\% \quad (10)$$

Where L (1 million) is the normal endurance of the NOR flash memory.

Three exemplary cases would be described to show the relation between number of bytes/pages to be written and total number of bytes/pages that are contained in the memory.

Case I. If the number of bytes/pages i.e. N′ to be written is very less as compared to the total number of bytes/pages space available in the memory i.e. N. $N' \ll N$.

When the number of bytes to be written is very less as compared to the total number of bytes available, then the life expectancy and reliability of the SSD increases significantly.

For example, if $N' = 10$ and $N = 500$, the life expectancy of the first 10 bytes increases by

$$(((500/10) \times 1000000 - 1000000)/1000000) \times 100\% = 4900\%$$

Thus, the life expectancy increases by 4,900 %, which increases the write cycle up to 49 million cycles.

Case II. If the number of bytes/pages i.e. N′ to be written is half as compared to the total number of bytes/pages space available in the memory i.e. N. $N' = N/2$.

When the number of bytes to be written is half of the total available space, then the life expectancy of the number of bytes to be written is increased by 100 %.

$$(((N/(N/2) \times 1000000) - 1000000)/1000000) \times 100\% = 100\%$$

Case III. If the number of bytes/pages i.e. N′ to be written is near to the total number of bytes/pages space available in the memory i.e. N. N′–N.

When the number of bytes to be written is approximately same as to the total number of available byte space, the wearing out time is not increased. In such case, it's always near to the original life expectancy.

For example, if $N' = 450$ and $N = 500$, the life expectancy of the first 450 bytes increases by

$$(((500/450) \times 1000000) - 1000000)/1000000) \times 100 = 11\%$$

This aforementioned increase in life expectancy is very nominal and is the worst case under consideration.

Thus, by using the proposed wear levelling algorithm in the paper, it is evident that the life expectancy, endurance and reliability of the SSD can be substantially increased.

5 Comparison study

The below Table 1 shows a comparative study between the different wear levelling algorithm that exists and are used.

Table 1 Comparison among wear levelling algorithms

Algorithm	Data migrations	Best efficiency condition
TrueFFS [5]	Periodical	Large data block, typically 1 K and large memory available
Dual-poll [1]	Threshold triggered	Large data block and large memory available
BET [11]	Threshold triggered	Large data block and large memory available
Hot–cold swapping algorithm [12]	Periodical	Large data block and large memory available
Rejuvenator [13]	Minimal	Large data block [8 K, 15 K, 20 K]and large memory available
Algorithm described in this paper	Independent	Very efficient when small burst data are frequently needed to be saved in external NOR type flash memories

Although, most of the comparison studies available are on NAND type memory, the paper has tried to justify the relative superiority of the algorithm in discussion with the existing wear levelling algorithms.

The table below shows the summary of some of the existing wear levelling algorithm and the wear levelling technique aforementioned.

It is evident from the above table, how the algorithm described here is different from all the other existing wear levelling algorithms.

The papers referenced [1, 2, 4, 6–14] in this paper talks about the implementation of wear levelling algorithm in NAND type memory devices. The NAND type memory devices are inherently used where the huge data is to be stored, as NAND type memory devices have large memory area. In contrast to NAND, NOR type memory devices have usually less memory space. Applications where the small burst data are frequently needed to be updated and power is limited, the NOR are type memories are usually preferred. Another promising factor which impulse to the use of NOR type memories for aforementioned application areas is their low power requirement for read operations and very low standby power as compared to NAND type memories [15]. Hence, NOR type memories can be used in application areas such as wireless sensor network, temperature and humidity data collection in cold chain management, blood bag temperature data collection etc.

The papers [1, 2, 4, 6–14] referenced here have proposed a very novel idea of wear levelling to increase the lifetime of NAND memory devices. In contrast to the referred papers, this paper describes a wear levelling algorithm which can be used in NOR type data memories, which eventually increases the life time of the memory.

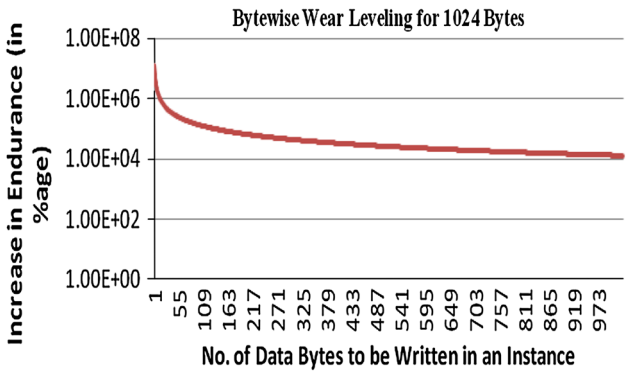


Fig. 5 Increase in endurance %age w.r.t total no. of bytes that are to be updated

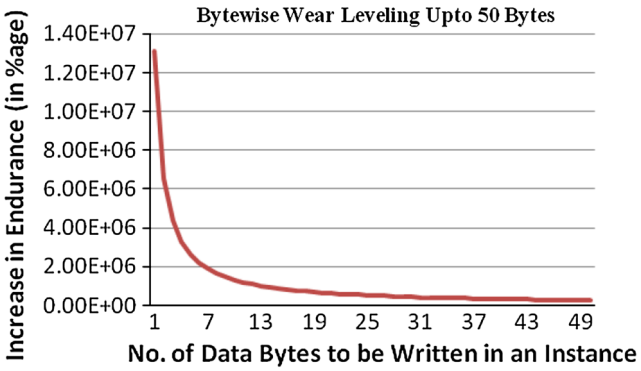


Fig. 6 Increase in endurance %age w.r.t total no. of bytes that are to be updated

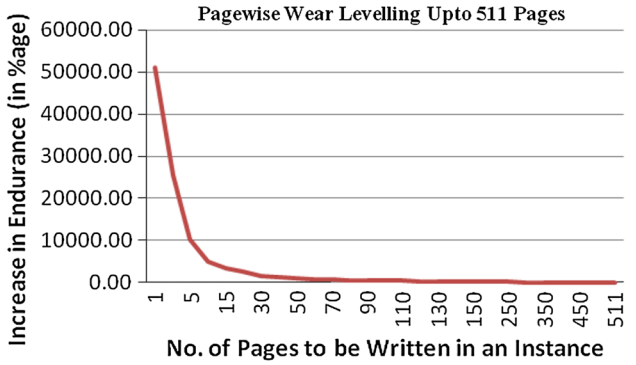


Fig. 7 Increase in endurance %age w.r.t total no. of bytes that are to be updated

The below Figs. 5, 6, 7 shows the graph for the increase percentage of endurance of the SSD with correspondence to the number of bytes that are to be updated in an instance. The total number of available bytes or pages is 130,826 or 511 respectively.

The above graph in Fig. 5 shows the increase in the endurance of the SSD considered in the paper with respect to the number of bytes that are to be updated in the SSD in an instance. The total number of bytes considered in the

Table 2 Performance issues

Algorithm	Overheads
TrueFFS	Swapping.
Dual-pool	Swapping, queue insertion
Rejuvenator	Swapping, list search for clean blocks, list insertion
Algorithm described in this paper	Storing the number of bytes written, storing the start address of data written to the NOR at the completion of one session of usage which is less than 0.2 % of the total available memory. Reading from the last page of NOR memory, whenever a new session for data logging is started

paper for the comparision study is 130,286 bytes. The Fig. 5 shows a logarithmic graph. It is quite evident from the graph that, the increase in endurance percentage is inverse exponential to the number of bytes that are to be updated in an instance.

The efficiency of the proposed wear leveling algorithm in this paper can be more clearly visualize in the Fig. 6, which considers a total number of 50 bytes to be updated in an instance on a SSD memory of 130,826 bytes of memory.

As aforementioned, the proposed wear levelling algorithm can also be used when the preferable write operation is page wise write operation. The efficiency of the proposed wear levelling algorithm reduces a bit when data is written page wise. This would be evident from Fig. 7.

The above Fig. 7 shows the increase in endurance of the SSD when data is written one page at a time rather than one byte at a time.

To conclude from the above explanations, this paper proposes a novel way to implement wear levelling algorithm where the amount of memory is less and is of the SSD type as well as the data to be saved is less but frequent.

6 Implementation issues and overheads

As with any algorithm there exist some implementation and overhead issues.

The Table 2 shows the effect on performance in using the flash memory while using the four different wear levelling techniques.

In the algorithms like TrueFFS, dual pool and rejuvenator, they use the concept of hot data and cold data. In order to evenly distribute the whole write process so that the life of the flash storage device increases, the data (in specified block size) are constantly moved from location to another. This causes a huge overhead in terms of code size, monitoring and maintenance. These factors might not truly exploit the overall efficiency of the wear levelling algorithm. The algorithm that is described in this paper doesn't have such type of overheads. Though it has some overheads but very negligible as compared to the other wear levelling algorithms in discussion.

The algorithm proposed here doesn't require any further metadata about the data for its working. The extra overhead that are required, are the number of bytes written and the start address from where the data was written on to the NOR memory. These two data are saved in the last page of the NOR type memory. The only requirement is to read the last page of NOR, prior to any write on the NOR and update after the completion of data write process. The information about the last page is saved in the flash memory of the microcontroller. Hence, it is quite evident that a minimum overhead is required.

The information about the start of read or writes operation and the number of bytes to be read or written is stored in the last page of the storage device consisting of few bytes. The size of the page used for this purpose is 256 bytes and the total number of available pages in the system in consideration has 512 pages, 511 pages for saving the sensed data and the last page for wear levelling information. Hence, the overhead incurred in storing the information is only 0.2 %, from Eq. 11. The system taken into consideration is having 131,072 numbers of bytes and each page is of 256 bytes.

$$(1/P) \times 100\,\% \tag{11}$$

where, P is the total number of available pages in the NOT type SSD.

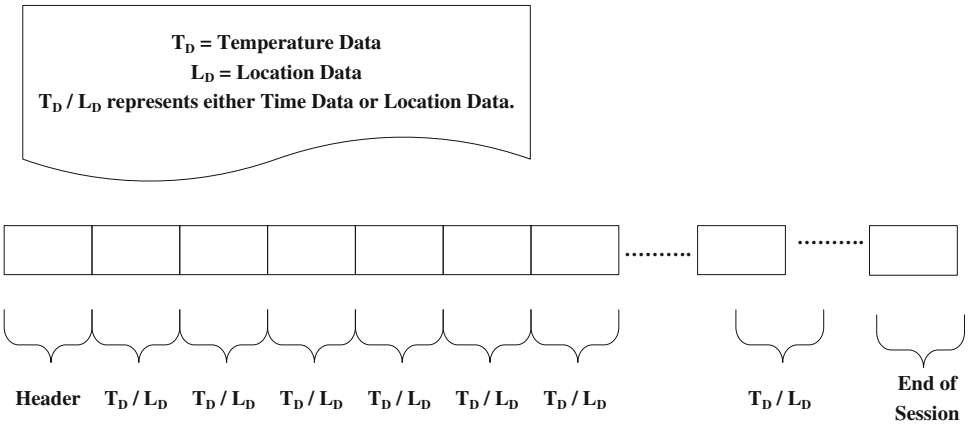
There are many data structures [14] available that can enhance the storage mechanism for the aforementioned algorithm. But, the drawback of using such available data structures would lead to increase in overhaul overhead of the system.

7 Evaluation

This section explains in detail of our experiment and implementation of the devised wear levelling algorithm. The system in which the wear levelling algorithm, in discussion, is being used is an ultra low power data logger. The primary data to be logged is the environmental temperature data. The data logger uses a low power microcontroller for governing the tasks and a 1 Mbit serial EEPROM for data storage. All the data fetched are stored in the EEPROM. The format of the data storage is described below.

The Fig. 8 shows the way the data in the external EEPROM is organized into. The first block of the data format

Fig. 8 Data format



is the header part. The header part contains the wakeup time interval in the first three bytes (B0–B2), the logging start time in the next four bytes (B3–B6) and the actual temperature in the next two bytes (B7–B8). The preceding sections contain either the temperature difference data along with the time data or a valid location data along with the relevant time data. The temperature difference and corresponding time data is stored in two byte. The location data and the corresponding time data are stored in seven bytes.

The write operation is done byte/page wise. If power is a crucial factor then page wise write operation is preferred over byte wise as less power is incurred in write operation.

In each data logging session, at least a minimum of 12 bytes are written.

Consider a case when these bytes are only written. In such a case, the first 12 bytes will wear out after 1 million write cycles. Thus if each second the data is updated, then the device ran only for about 11.6 days!

$$1000000 / (3600 \times 24) = 11.57 \text{ days}$$

In contrast to this, if the wear levelling algorithm described here is implemented, then the life expectancy would increase by 1,090,000 % as per equation 10. Hence, now the device can run for a time duration of 126,440 days, which is a substantial increase in endurance when wear level algorithm was not used.

If page wise write operation is selected, then the life of the device increases by

$$((((511/1) \times 1000000) - 1000000)/1000000) \times 100 = 51000\%$$

Hence, the device can run for 5,901 days, if in each second a page is written to the serial external EEPROM.

In the present wear levelling algorithm that we have proposed, the efficiency gets enhanced when lesser amount of data is supposed to be updated in the serial flash based EEPROM. The reason is that, lesser data provides more space for its data update process within the serial EEPROM.

8 Conclusion and future work

This research paper provides a new and better performance technique with lesser overheads for data storing on SSD, which can result to an increased life expectancy, endurance and reliability of the SSD such as EEPROM and flash memories. The proposed wear leveling algorithm has definitely increased the life expectancy of the SSD used for storing the data. The authors have tested and used the optimized algorithm in the e-safeT data logger developed by C-DAC. In the next few months, we hope to gain significant experience with the algorithm overall performance, by deploying the e-safeT data logger in cold supply chain.

In the future, the wear leveling algorithm can be enhanced for the last reserved page also. As, the last reserved page's life expectancy, endurance and reliability should be in sync with rest of the remaining NOR type SSD memory used for storing the fetched data.

Acknowledgments This work was done as a part of project titled "Design and Development of object tracking system for environmental sensitive object in transit" funded by Department of Electronics and Information Technology (DeitY), Ministry of Communications and Information Technology, Government of India. Authors are thankful to Dr. G.V Ramaraju, (Scientist G & Group Coordinator) and Smt. Geeta Kathpalia (Scientist G & Head of Division) for the support. The authors are indebted to Dr. B K Murthy, Executive Director, C-DAC & Shri V. B. Taneja to give enough space and freedom to cultivate and nurture the research areas in embedded systems.

References

1. Chang LP (2007) On efficient wear leveling for large scale flash memory storage system. In: SAC'07: proceedings the 2007 ACM symposium on applied computing, ACM, New York, pp 1126–1130
2. Sanvido MAA, Chu FR, Kulkarni A, Selinger R (2008) NAND flash memory and its role in storage architecture. Proc IEEE 96(11):1864–1874
3. High Endurance EEPROM Storage, AVR101, ATMEL Corporation

4. Chen CH, Hsiu PC, Kuo TW, Yang CL, Wang CY (2012) Age-Based PCM wear leveling with nearly zero search cost. In: Design automation conference (DAC), pp 453–458
5. Shmidt D (2002) Technical not: TrueFFS wear leveling mechanism. Technical report Msystems
6. Microelectronics ST (2006) Wear leveling in single cell NAND flash memories, Application Note (AN1822). Geneva, Switzerland
7. Understanding Flash Translation Layer (FTL) Specification, Intel Application Layer, AP-684
8. Park D, Du DHC (2011) Hot and cold identification for flash memory using multiple bloom filters
9. Chang LP, Kuo TW (2002) An adaptive striping architecture for flash memory storage systems of embedded systems. In: Proceedings of the eigth ieee real-time and embedded technology and applications symposium (RTAS'02)
10. Agrawal N, Prabhakaran V, Wobber T, Davis JD, Manasse M, Panigrahy R (2008) Design tradeoffs for SSD performance. In: ATC'08: USENIX 2008 annual technical conference on annual technical conference. USENIX Association, Berkeley, pp 57–70
11. Chang YH, Hsieh JW, Kuo TW (2007) Endurance enhancement of flash-memory storage systems: an efficient static wear leveling design. In: Procedings of the design automation conference (DAC)
12. Chang L-P, Kuo TW (2005) Efficient management for large-scale flash-memory storage systems with resource conservation. Trans Storage 1:4
13. Murugan M, Du D (2011) Rejuvenator: a staticwear leveling algorithm for flash memory. University of Minnesota, Minneapolis
14. Gal E, Toledo S (2005) Algorithms and data structures for flash memories. ACM Comput Surv 37:2
15. Toshiba NAND versus NOR Flash Memory Technology overview.