Write Improvement Strategies for Serial NOR Dataflash Memory

Scott Fazackerley
Department of Computer Science
University of British Columbia
Email: scott.fazackerley@alumni.ubc.ca

Wade Penson
Department of Computer Science
University of British Columbia
Email: wpenson@alumni.ubc.ca

Ramon Lawrence
Department of Computer Science
University of British Columbia
Email: ramon.lawrence@ubc.ca

***Abstract*—Embedded systems are ubiquitous and perform tasks such as data logging and monitoring. For these devices, lifetime, power use, and data consistency are critical. Systems require robust and energy efficient storage strategies. Serial NOR Dataflash is commonly used, but suffers from high write and erase times as well as limited lifetime. This work proposes write strategies for serial NOR Dataflash that improves efficiency and power use, and decreases write times. Experimental results demonstrate that using *masked overwriting* strategies can improve write times by an order of magnitude and reduce the number of required page erases, reduce energy consumed by writing and reduce data transfers by up to 90% for specific applications.**

I. Introduction

This work examines how write strategies optimized for serial NOR Dataflash can significantly improve device performance including fewer page erases, faster write operations, and less energy consumed. The technique applies to many applications and devices. In this work, the target devices are 8-bit processors which are commonly used due to low cost and complexity [1] and are well-suited for data collection and logging applications.

Embedded devices have limited SRAM and use either NAND or serial NOR flash for persistent storage. NAND flash has faster performance and larger capacity than NOR flash but requires a higher pin count and more complex data management strategies. NOR flash has simpler management requirements and would be a more useful technology if its write performance was more comparable to NAND flash.

This work presents serial NOR Dataflash optimized writing strategies that greatly improve its performance and usability for embedded devices. The contributions of this work are:

- A technique for reducing the time and energy required for writing to serial NOR Dataflash using *masked overwriting* to an existing page.
- A performance analysis and experimental evaluation of how overwriting and write masking techniques improve write performance.
- A use case analysis demonstrating performance advantages in common scenarios.

Corresponding Author: scott.fazackerley@alumni.ubc.ca. The authors gratefully acknowledge the support of the Natural Sciences and Engineering Research Council of Canada.

The organization of this paper is as follows. Section II presents background information on flash memory technologies. Section III provides an overview and analysis of write strategies for serial NOR Dataflash with experimental results discussed in Section IV. Section V presents use cases demonstrating the real-world impact of the performance improvements. The paper closes with future work and conclusions.

II. Background

Embedded devices need to be able to store and process data. The Internet of Things (IOT) involves devices such as wireless sensor networks and mobile computing platforms interacting with each other [2]. It is anticipated that by the end of this decade there will be between 30 and 50 billion devices participating in the IOT [3].

IOT vendors such as Cisco anticipate the direct sharing of data between devices driving the need for local storage and processing [4]. Devices such as the Telos, Btnode, MicaZ [5] platforms have been previously used as research and development platforms. Recently, the Arduino [6] family of devices has driven low cost development and exploration. These devices are typically small 8-bit devices [7] with power and persistent storage constraints as well as minimal memory (often less than 4KB SRAM) [8].

With the increased availability of low cost flash ($0.003 per Kbyte) [9], devices can now store large quantities of data. Considering the computation vs. communication trade-off [10], edge devices now maintain the ability to aggregate and analyze data locally and only transmit data on demand. This results in significant energy savings for devices [11], [12].

Flash is used for persistent storage and is available in two distinct types: *NOR* and *NAND*. While both share common attributes, the physical implementation of each type offers different performance characteristics.

NAND flash is the most commonly found format in general purpose devices [13], and is characterized by fast access times for sequential byte access, high density and low cost. Data is only accessible in a page format [12], [14] which limits the types of read and write operations. NAND flash is typically accessed in a parallel fashion, requiring a high pin count commitment from the host processor which makes it unsuitable for small, low pin count devices.

NOR flash was first commercialized by Intel in 1988 [15] and initially offered read and write units of one byte. The

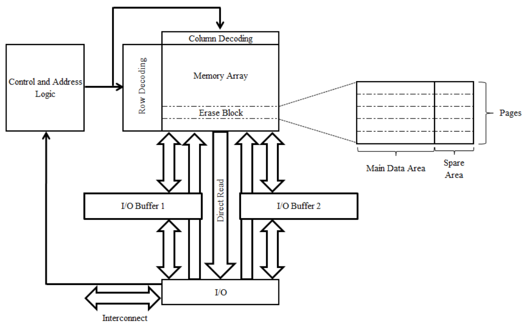


Fig. 1: Organization of Serial NOR Dataflash

cell size for NOR flash is 2.5x larger than NAND [16] due to additional circuitry required for reading the state of cells.While NOR flash is typically less dense and less energy efficient than NAND flash, it is available in both parallel or serial access formats [12], which increases its desirability for resource constrained systems.

While NOR flash has previously been discounted as a suitable candidate for read/write storage for general purpose computing [17], current trends have seen the evolution of NOR flash as a suitable storage candidate for embedded systems due to increased robustness and usabilty compared to NAND flash. NOR flash is the most popular format for embedded devices [18], being found in millions of embedded devices. It is available with a serial interface that makes it desirable for pin limited systems. Unlike NAND flash, NOR memory devices are also available with internal SRAM page buffers, which can be used to buffer page data, limiting the amount of data that is required to be transferred between a host processor and memory device. Figure 1 shows the organization of an Adesto serial NOR Dataflash device with two SRAM I/O buffers [19].

A challenge with flash memory is its asymmetric read and write costs in both time and energy. Devices have bytes organized into read, write and erase units. Read and write units are organized as pages that must generally be written or read as an atomic unit. Flash memory is also divided into erase blocks which may contain a single or multiple pages. With flash memory, a single memory location cannot be erased; all pages in the erase block must be erased together. Compared to other memory technologies, flash memory has a limited lifetime which is directly linked to erase and write operations. Depending on the type of flash, page lifetime is limited to between 10,000 to 100,000 erase/write cycles. This presents challenges in data management using flash memory.

Flash memory has the limitation that a page must be erased before data can be written or rewritten to the same physical location. In order to change the contents of a single page, the page must be first copied out along with all other data in the same block to a buffer. Once all pages within the erase block have been moved, the erase can proceed at which point data can be written back.

Two approaches had been suggested to deal with these challenges [15], [20]. The first approach is to use a flash-aware file system that is specifically designed to accommodate flash devices, but such systems are not suitable for resource constrained devices. The second approach is to use a translation layer [15] to act as an intermediary between the physical device and the application requiring the storage. This address translation is referred to as a *Flash Translation Layer* (FTL) [20], and is responsible for swizzling data addresses as it is physically relocated. Regardless of the strategy used, flash-based memory technologies require an address translation scheme and write normalisation strategy to be considered functionally useful [21], but both are generally not available for small embedded systems.

For the simplest of devices, direct mapping can be used but this leads to accelerated wear as data is copied out of the page to be mutated, the page erased and then the changed data written back to the same physical location. Due to architecture limitations of flash memory, a block of pages may be required to be moved to SRAM before the erase operation can proceed which further complicated operations for resource constrained systems. In the most constrained devices, this is not feasible.

None of these approaches truly solve the fundamental erase constraint; they just try to minimize the erases performed. This work analyzes the fundamental engineering of NOR flash and demonstrates write techniques that use page overwrites to minimize the need for costly page erases.

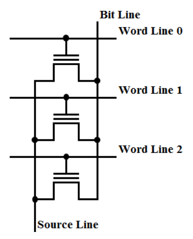
III. Write Strategies for Improved Performance with Serial NOR Dataflash

Understanding NOR flash memory technology allows for write optimizations. Flash memory technology is based on the floating gate MOSFET which has a similar architecture to the MOSFET used in SRAM. The floating gate MOSFET [22] can encode a persistent state for a long period of time without the requirement that it be continually powered.

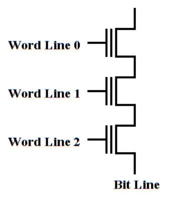
The floating gate is located between the control gate and the channel substrate in the MOSFET. It is electrically isolated from all other parts of the circuit, acts as a barrier between the control gate and the channel substrate, and can hold charge for long periods of time (years to 10's of years) [23]. It will overtime dissipate charge and loose the data encoded in the device [16].

In the electronic design of flash memory, an erased cell will have the state of a logical "1". When programmed, the cell value will be set to a logical "0" [24, p.29]. The cells are connected together in a matrix using the word line for addressing and the bit line for sensing. The most significant difference between the architecture of NOR and NAND flash is how the state of a cell is determined [23] and physical interconnect as shown in Figure 2.

When a charge is present on the floating gate, it will block the electric field from the control gate which is normally capacitively coupled to the channel substrate preventing the formation of the conduction channel [23]. It is not a binary effect; the charge level present on the floating gate modifies the threshold voltage needed to be induced on the control gate in order to form a conduction channel. If charge is present in the floating gate when the word line is activated to read from the cell, the bit line will not see a current flow which is interpreted as a logical "0". The electric field from the control gate to the channel substrate is shielded by the floating gate. If



(a) NOR Flash



(b) NAND Flash

Fig. 2: Memory Cell alignment for NOR and NAND Flash

no charge is present when a voltage is induced at the control gate, a conduction channel will be formed allowing for current to flow which is interpreted as a logical “1”.

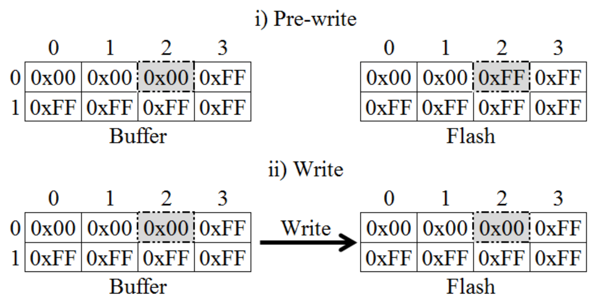
With the NOR architecture (Figure 2a) each element in the matrix has its control gate connected to a word line and bit line connected to the drain [23]. This allows the matrix to address a single element in the memory array without disturbing any other element. NAND flash shares a similar configuration in terms of the word line which is used to activate the control gate of the element or elements being read. The single largest difference in the architecture difference between NAND and NOR is how the bit line is connected. Unlike in NOR memory, the source and drains of the memory cells in NAND flash are linked together (Figure 2b) in a daisy chain fashion [23] which can lead to disruption of neighboring cells.

NAND flash using *Fowler Nordheim* (FN) tunneling for both erase and write operations whereas NOR only using FN tunneling for erase operations. For write operations, *channel hot election* (CHE) injection is used to inject electronics into the floating gate. This operation is self limiting in such that the injection operation will stop when sufficient charge has built up proportional to the strength of the electric field being applied.

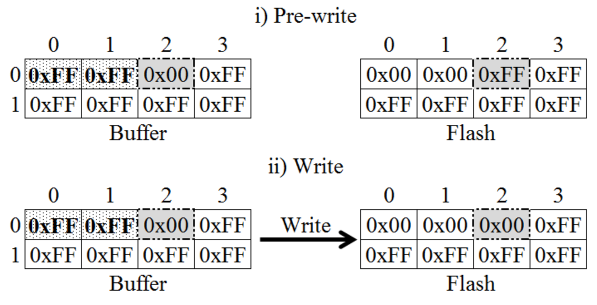
The floating gate of an erased NOR memory cell has a deficit of electrons. When being read, a voltage is applied to the control gate via the word line. A conduction channel will be formed between the source and drain, allowing for the flow of current. When the device is programmed, electrons are injected into the floating gate which will prevent the formation on an electric field when being read. This leads to a lack of conduction channel being formed which prevents current flow.

This observation leads to a unique opportunity with NOR flash that is not possible with NAND flash. When a page is erased, each cell will encode a logical 1 (lack of charge). When the cell is written (set to logical 0), charge is allowed to accumulated in the floating gate via CHE injection. If an attempt is made to write a logical 0 to the cell again, the existing build of charge will prevent any additional charge entering the floating gate due to the self limiting properties of CHE injection. It is hypothesized that this unique observation allows for the re-writing of memory cells under specific conditions which can be exploited to increase device performance.

It is conjectured that a NOR cell may be re-written as long as the transition is from logical 1 to logical 0 without disturbing neighboring cells due to the internal structure of NOR flash (Figure 2a) for page accessible serial NOR Dataflash.



(a) Overwritting



(b) Masked Overwritting

Fig. 3: Overwritting strategies for data movement from SRAM buffer to flash page for Serial NOR Dataflash

Under normal operation, data is loaded in to the SRAM buffer and then written to a previously erased page. In the case of append type operations, data is held in the SRAM buffer, new data appended, the target page erased and then rewritten. In this type of operation where new data is being written to previously erased locations with existing data being unchanged, it is hypothesized that the buffer can be rewritten back to the same pages without having to occur an addition erase operation. Figure 3a demonstrates this *overwriting* operation for a simplified example where the page and buffer size are 8 bytes. In this example, locations 0 and 1 have previously been written by the buffer to flash. In overwriting, new data is appended to location 2 (indicated by dashed outline) and then the buffer is written back to the same flash page. As the contents of the previously written locations are unchanged, only new locations will be modified.

A limiting factor in the overwriting operation is that a write to a previously written location will still induce CHE oxide degradation. It is hypothesized that this can be minimized by changing write patterns to previously written cells. When a logical 1 is transitioned to logical 0, charge is injected into the floating gate. In the case where the cell is already a logical 0 and a logical 1 write is attempted, no CHE injection will occur leaving the cell in its previous state. This transition can only physically occur with FN tunneling that is developed during an erase cycle. Figure 3b demonstrates this *masked overwriting* operation for a simplified example where the page and buffer size are 8 bytes. In this example, locations 0 and 1 have previously been written by the buffer to flash. In masked overwriting, new data is appended to location 2 (indicated by dashed outline) and previous written location in the buffer are masked to 0xFF (indicated in bold). The buffer is written back to the same flash page. The contents of the previously written locations are unchanged due to the forbidden 0 to 1 transition described, modifying only the unmasked, unwritten area.

For data storage operations this is a desirable operation as it allows new data to be appended to a page without having to occur additional erase/write cycles as the page can be rewritten in place. This presents significant improvements for specific classes of operations. It is estimated that the saving in terms of page erases and energy consumption is significant compared to write operations that are written to fresh pages for every commit. This will offer record level consistency without having to occur high levels of erase operations.

Given this understanding of the fundamental NOR flash architecture, the following hypotheses are tested:

Hypothesis 1: A serial NOR Dataflash page can be overwritten in place with no data loss as long as the only bit transitions are from 1 to 0 or 0 to 0.

Hypothesis 2: The page write time in serial NOR Dataflash is proportional to the number of bit transitions from 1 to 0 written.

Hypothesis 3: When overwriting a serial NOR Dataflash page, utilizing a bit mask (masked overwriting) of ones for all bits applied to previously written data in a page will improve performance while maintaining data correctness.

IV. Experimental Results

The experiments were run on a serial NOR Dataflash memory device (AT45DB161E from Adesto Technologies [19]).

To validate Hypothesis 1, memory pages were continually re-written with page data that contained an increasing number of zero bit values. The first write had zero zeros, the second had one (in the first bit), the third write had zeros in the first two bits, and so on. Each page consists of 512 bytes (4096 bits). Each page on the device was written 4096 consecutive times with each operation increasing the number of zero bits actively written. This was repeated for each device page for a total memory device writes of 16 777 216 times without a single bit error.

These results validate Hypothesis 1, supporting that it is possible to overwrite a NOR flash page without data loss with the constraint that all bit transitions are from 1 to 0 or 0 to 0.

The second hypothesis examines the correlation between write patterns and write times. It suggests that there is a correlation between the number of bit transitions (1 to 0) in a write operation and the amount of time to complete the buffer write to the main flash page. In this experiment, the number of 0's in the SRAM was increased after each write to the same page in memory. Two test conditions were used to examine if specific patterns impacted write times: A baseline where the target page was erased before each write and a test using overwriting (Figure 3a). The test conditions were evaluated with an increasing number of zeros starting with byte zero incrementally written to a flash page. Two SRAM buffers were used on the serial NOR Dataflash. One buffer was used for the masking test condition and the second buffer was used to maintain a mirrored state of data in the flash page for validation of contents. The test was repeated across

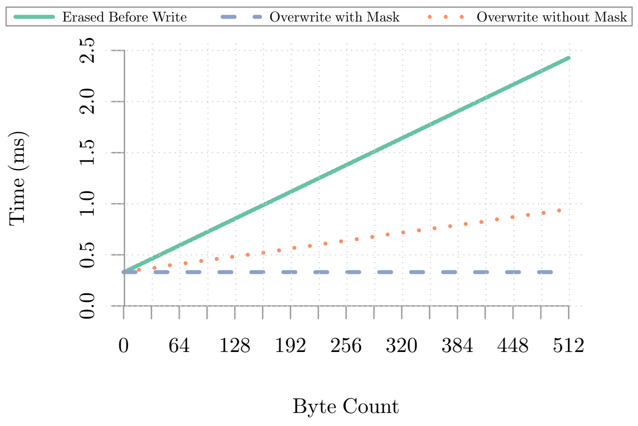


Fig. 4: A timing comparison of overwriting techniques for serial NOR Dataflash.

multiple device pages. Least squares analysis was performed on each test condition. The results and number of observations are presented in Table I.

Figure 4 shows the time is milliseconds to complete writing the n^{th} byte under a given test. For the baseline test (Figure 4: Erase Before Write), the flash page was erased before each write. The graph shows that the time of the device to complete the write of the buffer to the target page in flash memory is directly related to the number of 0 values being written. This is supported by the high degree of correlation (Table I) between 1 to 0 transitions per page and write times.

For the overwriting test (Figure 4: Overwrite without Mask), the flash page was only erased before the initial write. Each subsequent write to the page in flash used the overwriting technique (Figure 3a) where the SRAM buffer maintains consistency with the values written to flash and updates a single byte in a previously unwritten area of the target flash page. The results demonstrate a similar linear relationship and high degree of correlation (Table I), but faster write times were observed. This is due to faster equalization times for CHE injection when writing to previously written location as cells already contain excess electrons.

From the high degree of correlation observed, it is the act of attempting to modify the floating gate memory cell that is the dominate factor in writing, but the initial state of the cell in memory and buffer will impact write times.

Hypothesis 3 was that single byte write times could be increased through strategic management of previously written data in the buffer (Figure 4: Overwrite with Mask). Data within a page that is not actively being written is masked with a high logic state (Figure 3b). From Hypothesis 2, the write time is linearly related to the total number of 0 bits in the buffer being written. Thus, when writing a page to the device, it may be valuable to have all bits in high logic state (1) except for the bits being written (Figure 3b). This not only decreases page write times but will not degrade existing cells through additional CHE injection.

TABLE I: Least Squares Coefficients for Different Writing Methods

Write Strategies	Erase Before Write	Overwrite	Mask Overwrite
Slope	$4.099\times10^{-3***}$ (3.793×10^{-7})	$1.201\times10^{-3***}$ (5.647×10^{-7})	$2.016\times10^{-7***}$ (3.459e-8)
Intercept	$3.287\times10^{-1***}$ (1.122×10^{-4})	$3.311\times10^{-1***}$ (1.670×10^{-4})	$3.308\times10^{-1***}$ ($1.023e\times10^{-5}$)
Observations	2 097 152	2 097 152	2 097 152

Standard errors in parentheses.
(***) indicates significance at the p=0.01 level.
Each column contains regression coefficients for the linear model for a given write strategy.

As in the previous test, one serial NOR Dataflash SRAM buffer was used for consistency and validation checking of the flash page being written. It maintained the true state of what should appear in the flash page after each write to check the correctness of the masked overwrite. The second buffer implemented the overwriting technique (Figure 3b). The same test operations were used as in the previous test, except during each write, the location of the byte being written was unmasked and updated in the second buffer, written to the flash page and then re-masked. Least squares analysis was performed on each test condition. The results of each test and number of observations are presented in Table I.

In examining the results (Figure 4: Overwrite with Mask), it was found that the write times are not proportional to the actual number of 0's in the main flash page but to the number of 0's in the buffer being written. Unlike the baseline condition or overwriting condition, the page write times using masking were constant for each append operation as it prevented additional 0 to 0 writes due to previously written values. This suggests that strategic use of write patterns by way of masked overwriting when modifying a previously unwritten field in a page can lead to significant write time improvements for small data writes over other techniques.

V. Use Cases

For embedded applications, utilizing the ability to append or modify existing data without having to incur erases significantly extends the life of the device as well as simplifies data management for common applications. The following use cases show how masked overwriting can offer significant performance improvements in both time and lifetime.

A. *Data Logging*

Data logging applications [25] often use serial NOR Dataflash, and energy consumption and lifetime is paramount with the goal of minimizing service and maximizing field life. With data logging, the system is configured to take a series of defined measurements at regular periodic intervals. Devices such as the Arduino have found significant inroads in these applications [25] due to the low cost and ease of use. Common applications use a 12-bit analog-to-digital converter to record values every 1 minute and store the data to persistent storage.

An application logging a 2-byte record every minute writes to the same logical page 256 times (512 byte page size). Without overwriting, each record stored causes a full page write to a newly erased flash page. In comparison, using masked overwriting allows the application to write to the same page 256 times (appending a new record after each write). Using masked overwriting is significantly faster for page writes and reduces page erases by 256 times. For each write operation, 2 bytes of data and 2 bytes of mask are transferred between the host and memory for a total of 4 bytes per write. Overwriting improves write performance, extends lifetime, and reduces energy requirements by reducing the ratio of writes to erases while being able to maintain record level consistency.

	Memory Device	Record Size (bytes)	Data Transfer Size (bytes)	Data Overhead per write	Time (ms)
Data Logging	SD Card	2	512	510	2.61
	SNDF with masked overwriting	2	4	2	0.40
Bit Vector	SD Card	1	1024	1023	4.57
	SNDF with masked overwriting	1	2	1	0.33

TABLE II: Comparison between raw SD storage and serial NOR Dataflash (SNDF) using masked overwrite strategy.

Utilizing overwriting with serial NOR Dataflash makes it competitive with storing data on NAND flash technologies such as SD cards. Although NAND flash has inherent speed advantages over NOR flash, NAND flash must always read and write complete pages. SD cards do not maintain internal buffers, so a complete page must be transferred to the host, modified and then written back to the SD card. With masked overwriting and serial NOR Dataflash, only the new record is transferred to the device as it is able to buffer pages internally using the SRAM buffers and a new page is not needed for each write.

Table II presents actual device measurements using these two strategies. For a single record write, serial NOR Dataflash with masked overwriting significantly outperforms the SD card. In the course of one day, this translates to 1440 records. For the SD card this results in 737 280 bytes transferred versus 5 760 bytes resulting in a savings of approximately 99%. Additionally, the erase operations have been reduced by a factor of 256 which directly translates to significant energy savings and increased device lifetime without having to use a complex page remapping strategy.

B. *Bit Vectors*

Bit vectors [26] allow for compact storage of data. Bit vectors are not as efficient in persistent flash memory as data must be read and written at the page level. Consequently, updating a single bit in a bit vector in flash requires a whole page to be written.

Overwriting with serial NOR Dataflash offers a significant advantage when using bit vectors. Not only can the amount of data be reduced, but using the masked overwrite strategy, extra page copies and erases can be minimized. To accomplish this, a write mask is brought into one of the SRAM buffers on the memory device. As demonstrated, the mask can be written over live data without impacting the state of the data. Taking advantage of the proposed strategy, the host can then update the location of the bit vector and overwrite the data in memory assuming it is an allowed transition (1 to 0). Since only one field has changed in the buffer, it can be written back to its original page leaving the original data intact. Compared to utilizing a bit vector strategy for record management with SD cards, this strategy significantly reduces data transfers and writes as well as minimizing data movement and erases on the serial NOR Dataflash. Table II presents actual device measurements using these two strategies with masked overwriting offering a significant performance improvement.

VI. Conclusions

This work presents a low energy write optimization strategy called *masked overwriting* to increase serial NOR Dataflash lifetime, decrease energy consumption per operation and reduce average time per write operation. The result of this work supports the ability to do in-place append operations or bit manipulations for serial NOR Dataflash and provides an analysis of device performance based on write patterns.

This strategy reduces the complexity of data management as well as reducing energy costs and extending serial NOR Dataflash field life through lower write cost and a lower ratio of erase to writes on device. Use cases are presented, demonstrating the significant advantages of this strategy.

Future work will examine the use of this strategy for devices using serial NOR Dataflash to improve the operation of data structures and storage of data.

References

[1] C. J. Murray, "Why 8-bit MCUs Refuse to Go Away: New Peripherals are Paving the Way for the Continued Success of the 8-bit Microcontroller," *Design News*, vol. 70, no. 9, p. 30, 2015. [Online]. Available: http://www.designnews.com/author.asp?doc_id=278431

[2] D. Giusto, A. Iera, G. Morabito, and L. Atzori, Eds., *The Internet of Things: 20th Tyrrhenian Workshop on Digital Communications*. Springer, 2010.

[3] C. Witchalls, "The Internet of Things Business Index: A Quiet Revolution Gathers Pace," The Economist Intelligence Unit Limited, Tech. Rep., 2013.

[4] D. Evans, "The Internet of Things: How the Next Evolution of the Internet Is Changing Everything," Cisco Internet Business Solutions Group, Tech. Rep., 2001.

[5] P. Baronti, P. Pillai, V. W. C. Chook, S. Chessa, A. Gotta, and Y. F. Hu, "Wireless Sensor Networks: A Survey on the State of the Art and the 802.15.4 and ZigBee Standards," *Computer Communications*, vol. 30, no. 7, pp. 1655–1695, 2007. [Online]. Available: http://www.sciencedirect.com/science/article/B6TYP-4MP569D-2/2/2cb7b0fa0bd9d0dec76e4702a4d76937

[6] C. Severance, "Massimo Banzi: Building Arduino," *Computer*, vol. 47, no. 1, pp. 11–12, Jan. 2014.

[7] I. F. Akyildiz, W. Su, Y. Sankarasubramaniam, and E. Cayirci, "Wireless Sensor Networks: A Survey," *Computer Networks*, vol. 38, no. 4, pp. 393–422, 2002. [Online]. Available: http://www.sciencedirect.com/science/article/B6VRG-44W46D4-1/2/f18cba34a1b0407e24e97fa7918cdfdc

[8] H. Dai, M. Neufeld, and R. Han, "ELF: An Efficient Log-structured Flash File System for Micro Sensor Nodes," in *Proceedings of the 2nd International Conference on Embedded Networked Sensor Systems*. New York, NY, USA: ACM, 2004, pp. 176–187. [Online]. Available: http://doi.acm.org/10.1145/1031495.1031516

[9] S. Fazackerley and R. Lawrence, "A Flash Resident File System for Embedded Sensor Networks," in *Electrical and Computer Engineering (CCECE), 2011 24th Canadian Conference on*, May 2011, pp. 001 400–001 405.

[10] G. J. Pottie and W. J. Kaiser, "Wireless Integrated Network Sensors," *Commun. ACM*, vol. 43, no. 5, pp. 51–58, May 2000. [Online]. Available: http://doi.acm.org/10.1145/332833.332838

[11] J. Hill, R. Szewczyk, A. Woo, S. Hollar, D. Culler, and K. Pister, "System Architecture Directions for Networked Sensors," *SIGPLAN Not.*, vol. 35, pp. 93–104, Nov. 2000. [Online]. Available: http://doi.acm.org/10.1145/356989.356998

[12] G. Mathur, P. Desnoyers, D. Ganesan, and P. Shenoy, "Ultra-low Power Data Storage for Sensor Networks," in *Proceedings of the 5th International Conference on Information Processing in Sensor Networks*, ser. IPSN '06. New York, NY, USA: ACM, 2006, pp. 374–381. [Online]. Available: http://doi.acm.org/10.1145/1127777.1127833

[13] A. Zuck, O. Barzilay, and S. Toledo, "NANDFS: A Flexible Flash File System for RAM-constrained Systems," in *Proceedings of the Seventh ACM International Conference on Embedded Software*, ser. EMSOFT '09. New York, NY, USA: ACM, 2009, pp. 285–294. [Online]. Available: http://doi.acm.org/10.1145/1629335.1629374

[14] D. Zeinalipour-Yazti, S. Lin, V. Kalogeraki, D. Gunopulos, and W. A. Najjar, "Microhash: An Efficient Index Structure for Flash-based Sensor Devices," in *Proceedings of the 4th Conference on USENIX Conference on File and Storage Technologies - Volume 4*. Berkeley, CA, USA: USENIX Association, 2005, pp. 3–3. [Online]. Available: http://portal.acm.org/citation.cfm?id=1251028.1251031

[15] S. J. Kwon, A. Ranjitkar, Y.-B. Ko, and T.-S. Chung, *FTL Algorithms for NAND-Type Flash Memories*. Springer Berlin - Heidelberg, Mar. 2011, vol. 15. [Online]. Available: http://www.springerlink.com/index/10.1007/s10617-011-9071-9

[16] M. Sanvido, F. R. Chu, A. Kulkarni, and R. Selinger, "NAND Flash Memory and Its Role in Storage Architectures," in *Proceedings of the IEEE*, vol. 96-11. IEEE, 2008, p. 18641874. [Online]. Available: http://ieeexplore.ieee.org/xpls/abs_all.jsp?arnumber=4694025

[17] Y. Deng and J. Zhou, "Architectures and Optimization Methods of Flash Memory Based Storage Systems," *J. Syst. Archit.*, vol. 57, no. 2, pp. 214–227, Feb. 2011. [Online]. Available: http://dx.doi.org/10.1016/j.sysarc.2010.12.003

[18] C. S. I. Zitlaw, "The Future of NOR Flash Memory," 2011. [Online]. Available: http://eetimes.com/design/memory-design/4215634/The-Future-of-NOR-flash-memory

[19] Adesto Technologies. (2015, july) 16-mbit dataflash (with extra 512-kbits), 2.3v or 2.5v minimum spi serial flash memory. [Online]. Available: http://www.adestotech.com/wp-content/uploads/doc8782.pdf

[20] T.-S. Chung, D.-J. Park, S. Park, D.-H. Lee, S.-W. Lee, and H.-J. Song, "A Survey of Flash Translation Layer," *J. Syst. Archit.*, vol. 55, no. 5-6, pp. 332–343, May 2009. [Online]. Available: http://dx.doi.org/10.1016/j.sysarc.2009.03.005

[21] D. Ma, J. Feng, and G. Li, "LazyFTL: A Page-Level Flash Translation Layer Optimized for NAND Flash Memory," in *Proceedings of the 2011 International Conference on Management of Data*, ser. SIGMOD '11. New York, NY, USA: ACM, 2011, pp. 1–12. [Online]. Available: http://doi.acm.org/10.1145/1989323.1989325

[22] K. Kahng and S. M. Sze, "A Floating Gate and its Application to Memory Devices," *Electron Devices, IEEE Transactions on*, vol. 14, no. 9, pp. 629–629, 1967.

[23] R. Bez, E. Camerlenghi, A. Modelli, and A. Visconti, "Introduction to Flash Memory," *Proceedings of the IEEE*, vol. 91, pp. 489–502, Apr. 2003.

[24] R. Micheloni, G. Campardo, and P. Olivo, *Memories in Wireless Systems*, 1st ed. Springer Publishing Company, Incorporated, 2008.

[25] D. K. Fisher and P. J. Gould, "Open-Source Hardware Is a Low-Cost Alternative for Scientific Instrumentation and Research," *Modern Instrumentation*, vol. 1, no. 2, pp. 8–20, 2012.

[26] D. Rotem, K. Stockinger, and K. Wu, "Optimizing I/O Costs of Multi-dimensional Queries Using Bitmap Indices," in *Database and Expert Systems Applications*, ser. Lecture Notes in Computer Science, K. Andersen, J. Debenham, and R. Wagner, Eds. Springer Berlin Heidelberg, 2005, vol. 3588, pp. 220–229.