Table C.1 shows that each (valid) codeword of a code is far *at least* d_{min} from all the other (valid) codewords.

C.1.1 Error Detection

Fig. C.2 shows how a single-bit error can modify a 0000 codeword.

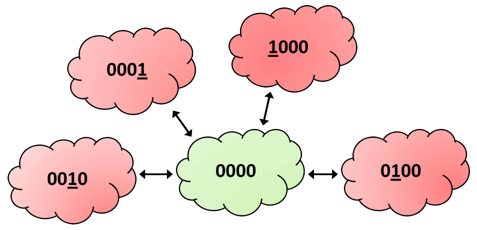

Figure C.2: A "0000" codeword after a single-bit error

E.g., if we read the 0001 codeword from the memory, it is not a valid codeword. In fact, 0001 does not belong to the code of Table C.1. Therefore, the error can be detected.

Fig. C.3 provides a generic example of the encoding/decoding process.

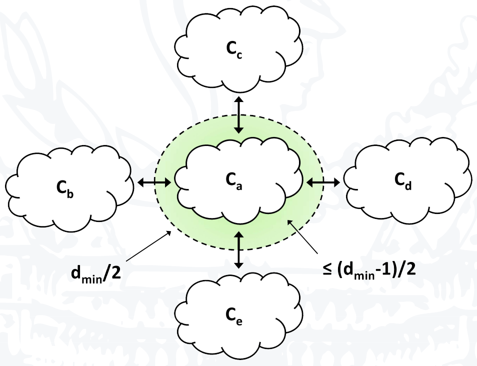

Figure C.3: Generic case Codeword

Fig. C.3 shows that each (valid) codeword is far from the other (valid) codeword at least d_{min}. At least d_{min} single-bit errors have to occur in order to produce another valid codeword. As a consequence, all d_{min}-1 single-bit errors can be detected.

"*A code with d_{min}=d+1 is able to detect d single-bit errors*"

E.g., the code of Table C.1 has $d_{min} = 2$. Therefore, it is able to detect all 1-single-bit error. In fact, a single-bit error on a valid codeword never provides a valid codeword.

C.1.2 Error Correction

Let us discuss the correction. Supposing a single bit error, Fig. C.4 shows how the wrong 0001 codeword can be corrected.

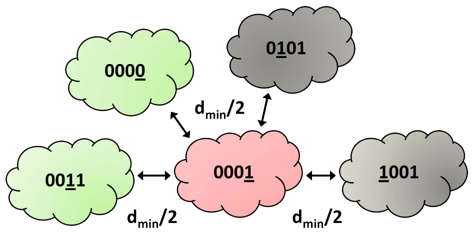

Figure C.4: The wrong "0001" read codeword

0001 is "halfway" between any pair of these codewords[1]. Therefore, it is not possible to understand which codeword 0001 originally was. In other words, this code can only detect 1-single-bit errors and is not able to correct any error.

If the codeword C_a of Fig. C.3 is affected by less than $d_{\min}/2$ single bit errors, then the closest codeword to the faulty one is C_a itself.

"*Any codeword affected by* $\#errors \leq (d_{min} - 1)/2$ *is correctable. Therefore, the correcting power of the code is* $t = \lfloor (d_{min} - 1)/2 \rfloor$"

In order to correct t errors, we need a code with:

$$d_{min} \geq 2t + 1 \tag{C.1}$$

C.1.3 Hamming bound

Let us assume to have a n-bit codeword, a k-bit data, q symbols[2], minimum Hamming distance $d_{\min}$ and a correction capability $t = \lfloor (d_{min} - 1)/2 \rfloor$.

Eq. C.2 has to be satisfied in order to proof the validity of Eq. C.1.

$$n - k \geq log_q \left\{ \sum_{i=0}^{t} \left[\binom{n}{i} (q-1)^i \right] \right\} \tag{C.2}$$

We usually refer to Eq. C.2 as *Hamming bound* [56].

[1] "0101" and "1001" are not valid codewords and will be not valid options
[2] if q = 2, symbols are called bits

C.2 Bose-Chaudhuri-Hocquenhem Codes Design Flow

Fig. C.5 resumes the BCH codes design flow.

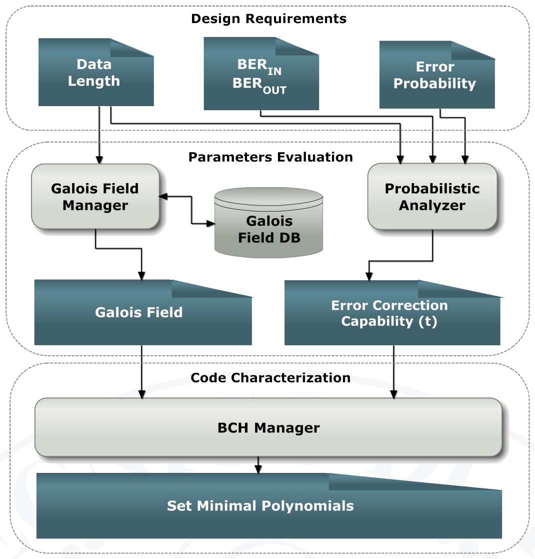

Figure C.5: BCH Code Design Flow

Three main functional steps compose the BCH design flow: (i) *Design Requirements*, (ii) *Parameters Evaluation*, and *Code Characterization*. After the last step, the BCH code is completely defined.

C.2.1 Design Requirements

The first step of each BCH code design flow is to define the mission-critical *requirements*. ECC algorithm works on data of fixed length (i.e., *Data Length*). The correction capability is determined w.r.t. probabilistic studies. The Bit Error Rate (BER) of the page [90], i.e., the fraction of its erroneous bits, is mainly composed by two values: (i) Raw BER (RBER) and (ii) UBER.

The former is the Raw BER (RBER), i.e., the BER before applying the error correction. RBER is technology/environment dependent and is not constant; it increases with aging of the page [13, 90].

The latter is the Uncorrected BER (UBER), i.e., the BER after the application of the ECC, which is application dependent. It can be computed as the probability of having more than t errors in the codeword (calculated as a binomial distribution of randomly occurred bit errors) divided by the length of the codeword [34]:

$$UBER = \frac{P(E>t)}{n} = \frac{1}{n}\sum_{i=t+1}^{n}\binom{n}{i}\cdot RBER^{i}\cdot(1-RBER)^{n-i} \tag{C.3}$$

if $n\cdot RBER \ll 1$, [56] rewrites Eq. C.3 as:

$$UBER \approx \frac{1}{n}\cdot\binom{n}{t+1}\cdot RBER^{t+1}\cdot(1-RBER)^{n-t-1} \tag{C.4}$$

C.2.2 Parameters Evaluation

The Bit Error Rate (BER) of the page [90], i.e., RBER and UBER, is the key factor used to select the correction capability. Fig. C.6 shows the resulting UBER for $k = 2^{14} = 16,384 bits$ $= 2Kbytes$ and $t = \{0, 1, 5, 10, 15\}$.

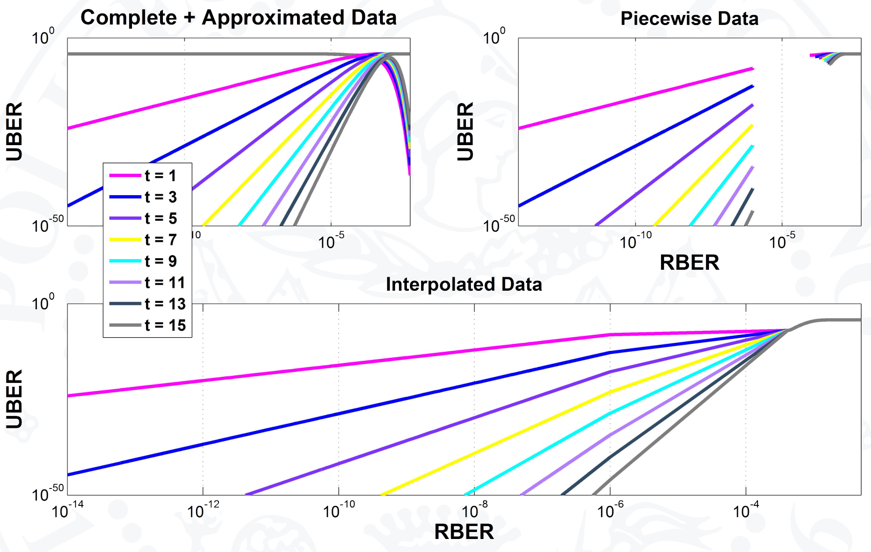

Figure C.6: Examples of Raw BER and Uncorrected BER

The second parameter is the Galois Field (GF). Many codes are based on the abstract algebra and, in particular, on GF [2]. A GF is a finite field with order q, i.e., it has a finite number of elements represented with q symbols). The set of m-tuples of elements from GF is the GF($\mathrm{q^m}$) vector space. Linear q-ary are a set of m-tuples over GF(q) or, in other words, are subspaces of GF($\mathrm{q^m}$) [56]. A GF($\mathrm{q^m}$):

- contains q^m elements, defined as $p_m(x=\alpha)=0 \Longleftrightarrow \alpha^m = b_{m-1}a^{m-1}+b_{m-2}a^{m-2}+ \ldots + b_0$;
- all elements can be expressed as α^i with $i\epsilon\left(0,\ldots,q^m-2\right)$;
- always $\alpha^{q^m-1}=1=\alpha^0$;
- is closed with respect to addition and multiplication (i.e., the sum or the product of two codewords is a codeword);

Different GFs matches different codes. In particular, two main parameters set the GF: (i) the data length k and (ii) the correction capability t.

E.g., if q = 2, Eq. C.5 set the minimum GF(2^m) required for the data length k [81].

$$k + m \times t \leq 2^m - 1 \tag{C.5}$$

E.g., replacing k = 2^{14} = 16,384 bits = 2KBytes into Eq. C.5, we need at least a Galois Field with $2^m = 2^{15}$ = 32,767 elements.

Spare area and parity bits Eq. C.5 set the minimum m to generate the related GF^m. The number of parity bits is denoted as $r = m \times t$. Such r parity bits are usually stored in the spare area of the flash memory. Therefore, a proper trade-off is needed when designing the ECC in terms of resources overhead.

C.2.3 Code Characterization

Finally, we exploit the correcting power t and the Galois Field to generate the *Minimal Polynomials* $\psi_1(x)$, $\psi_2(x)$, ..., $\psi_{2t}(x)$ [2, 81]. They fully characterize the BCH code.

The set of *Minimal Polynomials* defines the *Polynomial Generator g(x)* of the BCH code [2] as:

$$g\left(x\right) = LCM\left[\psi_1\left(x\right), \psi_2\left(x\right) \ldots, \psi_{2t}\left(x\right)\right] \tag{C.6}$$

LCM is the Least Common Multiple operator among the *2t* minimal polynomials defined above.

Table C.2 summarize the main BCH code properties.

Table C.2: BCH code properties

Specified by	zeroes $\alpha, \alpha^2, \alpha^3, ..., \alpha^{2t}$ of all the codewords $w(x)$
Codewords Length	$n = 2^m - 1$
Information Symbols	$k = n - degree$ of the generator polynomial $g(x)$
Minimum Distance	$d \geq 2t - 1$
Error Control Capability	Corrects t errors

C.2.4 Shortened Codes

In system design, a code of suitable natural length or suitable number of information digits usually cannot be found. Therefore, it may be desirable to *shorten* a code to meet the requirements. Whenever $n = k + r < 2^m - 1$, the BCH code is called *shortened* or *polynomial*. In a shortened BCH code the codeword includes less binary symbols than the ones the selected Galois field would allow. The missing information symbols are imagined to be at the beginning of the codeword and are considered to be 0. A shortened code has at least the same error-correcting capability as the code from which it is derived [74].

E.g., protecting k = 2^{14} = 16,384 bits data length implies to adopt a GF with 32,767 elements (refer to Eq. C.5). Assuming to correct $t = 5$ errors, we have a resulting codeword n = k+m×t = 16,384 + 15×5 = 16,459 bits < 32,767 = 2^{15}-1. Therefore, we may adopt a code which is shortened of 32,767 - 16,459 = 16,308 bits. A complete BCH[n, k, t] = [32,768, 16,384, 5] becomes a shortened BCH[16,459, 16,384, 5] BCH code.

C.3 Error Detecting and Correcting Codes: The actual trend

ECCs are moving toward two main directions [42]: (i) stronger ECCs and (ii) larger data block.

A stronger ECC has higher correcting power t. However, bigger t implies a higher number $r = m \times t$ of check bits. An higher complexity is also required to detect/correct higher number of errors.

On the other hand, the current trend is to adopt k = 512 Byte. A bigger data length size k may better handle higher concentration of errors. However, bigger k implies bigger symbol size (see Eq. C.5).

Fig. C.7 shows an example of moving toward bigger data length.

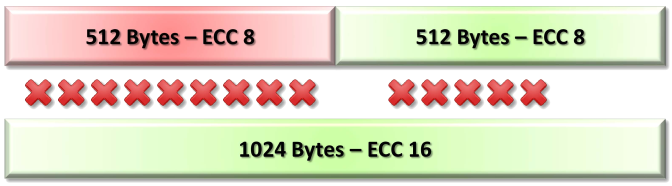

Figure C.7: ECC Example for point "Large Block..."

The first part of Fig. C.7 has two data blocks with k = 512 Bytes. Each block is protected with an ECC with t = 8. This is usually denoted as ECC-8. The second part of Fig. C.7 has one block with k = 1,024 Bytes with ECC-16.

Although the situation looks similar, having 9 and 5 errors in the two k = 512 Bytes block implies a critical failure. Having 9 + 5 = 16 errors are correctable within the k = 1,024 Bytes data blocks.

C.3.1 Examples

Fig. C.8 shows the UBER for several ECCs.

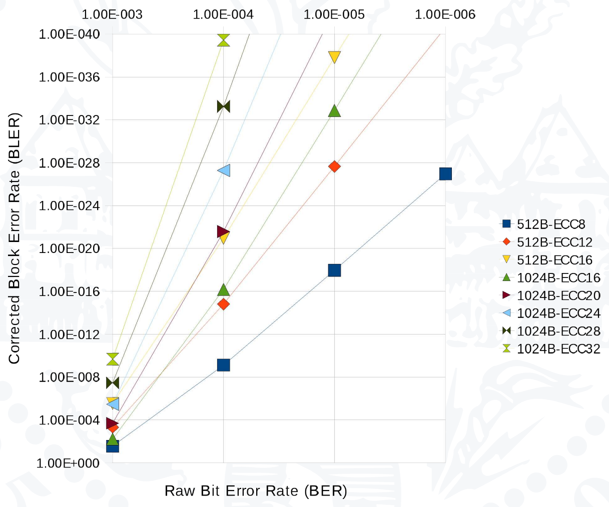

Figure C.8: Uncorrected BER for different ECCs

Fig. C.8 shows that moving toward bigger data blocks improves the UBER. Furthermore, a 512B-ECC16 and a 1024B-ECC16 are equivalent from a UBER standpoint. We provide some simple examples to understand the trade-off to tackle during ECC design.

Example 1 Fig. C.9 shows a first possible example.

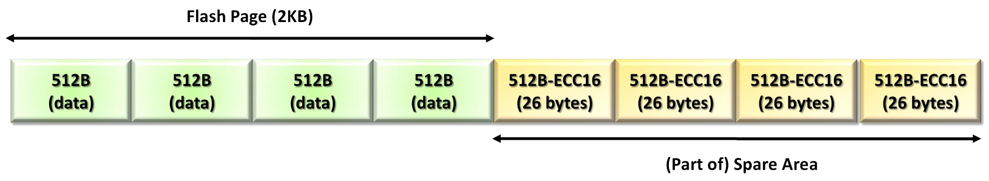

Figure C.9: 512B-ECC16 protecting a 2KB page

Let us assume k = 512 Bytes protected by ECC16 (i.e., 16 errors can be corrected). This is usually denoted as 512B-ECC16. We need:

- **Parity Symbol Size** (m): Eq. C.5 set $m = 13$, i.e., 13-bit parity symbols;
- **Correcting Power**(t): $t = 16$, which implies 13 bit × 16 parity symbols/block = 26 Bytes/block;
- **Complexity**: a 512B-ECC16 requires 4×26 Bytes = 104Byte;

Example 2 Fig. C.10 shows another example.

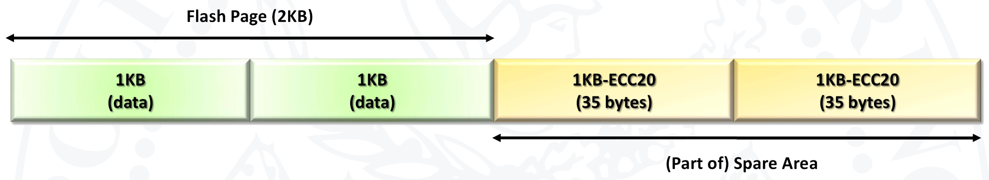

Figure C.10: 1KB-ECC16 protecting a 2KB page

Let us assume k = 1 KBytes protected by ECC20 (i.e., 20 errors can be corrected). This is usually denoted as 1KB-ECC20. We need:

- **Parity Symbol Size** (m): Eq. C.5 set $m = 14$, i.e., 14-bit parity symbols;
- **Correcting Power**(t): $t = 20$, which implies 14 bit × 20 parity symbols/block = 35 Bytes/block;
- **Complexity**: a 1KB-ECC20 protecting a 2KB page requires 2×35 Bytes = 70Byte/-page;

As well as Fig. C.8 shows, the 1KB-ECC20 (Fig. C.9) provides a better UBER than 512B-ECC16 (Fig. C.10), but at lower resource overhead in terms of occupied spare area.

C.4 Error correcting techniques for future NAND flash memory

Thanks to their lower RBER, a 512B-ECC1 (i.e., single-bit correction) may be sufficient for Single Level Cell (SLC) NAND flash. Multi Level Cell (MLC) NAND flashes have higher RBER. Therefore, they require higher correction capability (e.g., at least 512B-ECC4) [48].

20nm NAND flash The continuous scaling-down and the related increasing density of NAND flash implies to adopt proper ECC controllers and algorithms. The first 20nm NAND flash devices are currently available [85]. Such a quick scaling-down implies fewer electrons to enter the Floating Gate (FG). Therefore, there is a higher uncertainty about the charge in the FG.

More bits per cell Nowadays, MLC-based NAND flash can store up to 4 or 8 bit per cell. Although the density of the memory is dramatically increased, also the possible disturbances are much worse. As a consequence, ECCs have to increase their correcting power.

Larger page size The current trend is to increase the page size. 4KB or also 8KB is the most common page size, especially for Solid State Drive.

APPENDIX D

LIST OF SYMBOLS AND ACRONYMS

Due to the large number of symbols used in this thesis to support the description of covered material, we provide the following list of symbols and abbreviations. This list is intended to help the reader identify the meaning of a given symbol or acronym in a fast and easy way.

ADAGE	ADaptive ECC Automatic GEnerator
ARM	Advanced RISC Machine
B	Bulk
BC	BL Coupling
BCH	Bose-Chaudhuri-Hocquenhem
BED	Bit-line Erase Disturbance
BF	Bridging Fault
BL	Bit-Line
BED	Bit-line Erase Disturbance
BER	Bit Error Rate
BPD	Bit-line Program Disturbance

CC	Capacitive Coupling
CFAC	Coupling Fault between Adjacent Cells
CFFS	Core Flash File System
CG	Control Gate
D	Drain
DC	Direct Coupling or Direct field effects
DC – E	DC-Erase
DC – P	DC-Programming
DD	Drain Disturbance
DED	Double Error Detection
DRAM	Dynamic RAM
ECC	Error Correcting Code
EEPROM	Electrically Erasable-programmable read-only memory
EOL	End-Of-Life
exFAT	The ExtendedFAT
ext2	Second Extended File System
FARM	Fault Activation Readout Measure
FAT	File Allocation Table
FFS	Flash File System
FG	Floating Gate
FIFO	First In First Out
FIT	Failure In Time
FLARE	FLash ARchitecture Evaluator

FlexFS	Flexible FFS
FTL	Flash Translation Layer
GF	Galois Field
GNU	GNU is Not Unix
HD	Hard Disk
LSB	Least Significant Byte
JTAG	Joint Test Action Group
JFFS	Journaling Flash File System
KLE	Kernel Level Emulation
MLC	Multi Level Cell
MMFU	Mass Memory Formatting Unit
MP3	Moving Picture Expert Group-1/2 Audio Layer 3
MSB	Most Significant Byte
MTBF	Mean Time Between Failures
MTD	Memory Technology Device
MTTF	Mean Time To Failure
MTTR	Mean Time To Repair
NOP	Number Of PPP
NTFS	New Technology File System
NVRAM	Non Volatile RAM
OED	Over-Erase Disturbance
OEP	Over-Erase Program
ONFi	Open NAND Flash interface

OPD	Over-Program Disturbance
OS	Operating System
P/E	Program/Erase
PD	Program Disturbance
PPP	Partial Page Programming
RAM	Random Access Memory
RD	Read Disturbance
RBER	Raw BER
RDA(E)	RD Addressed Erase
RDA(P)	RD Addressed Program
RDI	Remote Debug Interface
RDU(E)	RD Unaddressed Erase
RDU(P)	RD Unaddressed Program
RISC	Reduced Instruction Set Computer
RS	Reed-Solomon
S	Source
SAF	Stuck-At Fault
SDRAM	Synchronous DRAM
SEC	Single Error Correction
SG	Select Gate
SLC	Single Level Cell
SRAM	Static RAM
SSD	Solid State Drive

TrueFFS	True FFS
UBER	Uncorrected BER
ULE	User Level Emulation
USB	Universal Serial Bus
WED	Word-line Erase Disturbance
WL	Word-Line
WPD	Word-line Program Disturbance
YAFFS	Yet Another Flash File System
YDI	YAFFS Direct Interface

BIBLIOGRAPHY

[1] MAGMA computational algebra system. Retrieved February 24, 2013 from the World Wide Web `http://magma.maths.usyd.edu.au/magma/`.

[2] Jiri Adamek. *Foundations of Coding: Theory and Applications of Error-Correcting Codes, with an Introduction to Cryptography and Informat.* John Wiley & Sons, Inc., New York, NY, USA, 1st edition, 1991.

[3] Advantest. Advantest announces NAND flash test solutions. Retrieved February 24, 2013 from the World Wide Web `http://www.advantest.co.jp/news/press-2011/20110711/en-index.shtml`, July 2011.

[4] Aeroflex. Aeroflex hiRel memories. Retrieved February 24, 2013 from the World Wide Web `http://www.aeroflex.com/ams/pagesproduct/prods-hirel-mems.cfm`.

[5] Aivosto. Visustin. Retrieved February 24, 2013 from the World Wide Web `http://www.aivosto.com/visustin.html`, 2010.

[6] Aleph One Ltd. Yaffs direct interface (YDI). Retrieved February 24, 2013 from the World Wide Web `http://users.actrix.co.nz/manningc/yaffs_direct2doc.pdf`, 2010.

[7] Aleph One Ltd. Yet another flash file system 2 (YAFFS2). Retrieved February 24, 2013 from the World Wide Web `http://www.yaffs.net/`, 2011.

[8] A. Ban. Flash file system, U.S. patent 5404485, apr. 4. Retrieved February 24, 2013 from the World Wide Web `http://www.freepatentsonline.com/5404485.pdf`, 1995.

[9] A. Benso and P. Prinetto. *Fault Injection Techniques and Tools for Embedded Systems Reliability Evaluation*, volume 1-4020-7589-8. Kluver Academic, 2003.

[10] E. Berlekamp. Goppa codes. *IEEE Transactions on Information Theory*, 19(5):590 – 592, sep 1973.

[11] Elwyn R. Berlekamp. *Algebraic coding theory*. McGraw-Hill, 1968.

[12] R. C. Bose and D. K. Ray-Chaudhuri. On a class of error correcting binary group codes. *Information and Control*, 3(1):68–79, 1960.

[13] Joe Brewer and Manzur Gill. *Nonvolatile Memory Technologies with Emphasis on Flash: A Comprehensive Guide to Understanding and Using Flash Memory Devices*. Wiley-IEEE Press, January 2008.

[14] M. Brüggemann, H. Schmidt, D. Walter, F. Gliem, R. Harboe-Sørensen, P. Roos, and M. Stähle. SEE tests of NAND flash-memory devices for use in a safeguard data recorder. *Radiation Effects on Components and Systems (RADECS)*, A-3, 2006.

[15] M. Brüggemann, H. Schmidt, D. Walter, F. Gliem, and H. Michalik. Further heavy ion and proton SEE evaluation of high capacity NAND-flash-memory devices for safeguard data recorder. *8th ESA/ESTEC D/TEC-QCA Final Presentation Day*, February 2007.

[16] M. Caramia, S. Di Carlo, M. Fabiano, and P. Prinetto. Exploring design dimensions in flash-based mass-memory devices. *Proceedings of IWSSPS 2009 : 4th International Workshop on Software Support for Portable Storage, Grenoble (France), Oct 2009, 15*, pages 43–48, 2009.

[17] M. Caramia, S. Di Carlo, M. Fabiano, and P. Prinetto. FLARE: A design environment for flash-based space applications. In *Proceedings of IEEE International High Level Design Validation and Test Workshop*, HLDVT '09, pages 14–19, San Francisco, CA, USA, 4-6 Nov. 2009.

[18] M. Caramia, S. Di Carlo, M. Fabiano, and P. Prinetto. Flash-memories in space applications: Trends and challenges. In *Proceedings of the 7th IEEE East-West Design & Test Symposium*, EWDTS '09, pages 429–432, Moscow, Russian Federation, 18-21 Sept. 2009.

[19] M. Caramia, M. Fabiano, A. Miele, R. Piazza, and P. Prinetto. Automated synthesis of EDACs for FLASH memories with user-selectable correction capability. *Proceedings of IEEE International High Level Design Validation and Test Workshop (HLDVT)*, pages 113 –120, june 2010.

[20] M. Cassel, D. Walter, H. Schmidt, F. Gliem, H. Michalik, M. Stähle, K. Vögele, and P. Casel Roos. NAND-flash-memory technology in mass memory systems for space applications. *Proceedings Data Systems In Aerospace (DASIA) 2008*, 2008. Palma de Mallorca, Spain.

[21] Li-Pin Chang. On efficient wear leveling for large-scale flash-memory storage systems. In *SAC '07: Proceedings of the 2007 ACM symposium on Applied computing*, pages 1126–1130, New York, NY, USA, 2007. ACM.

[22] Li-Pin Chang and Tei-Wei Kuo. An efficient management scheme for large-scale flash-memory storage systems. In *SAC '04: Proceedings of the 2004 ACM symposium on Applied computing*, pages 862–868, New York, NY, USA, 2004. ACM.

[23] Yuan-Hao Chang, Jen-Wei Hsieh, and Tei-Wei Kuo. Endurance enhancement of flash-memory storage systems: an efficient static wear leveling design. In *Proceedings of the 44th annual Design Automation Conference*, DAC '07, pages 212–217, San Diego, California, 4-8 June 2007. ACM.

[24] E. Chen and T. Yen. Comparing SLC and MLC flash technologies and structure. Retrieved February 24, 2013 from the World Wide Web `http://www.advantech.com.tw/epc/newsletter/Whitepaper/WhitePaper_Comparing_SLC_and_MLC_Flash_Technologies_and_Structure_200909.pdf`, Advantech, 2009.

[25] Te-Hsuan Chen, Yu-Ying Hsiao, Yu-Tsao Hsing, and Cheng-Wen Wu. An adaptive-rate error correction scheme for NAND flash memory. *Proceedings of 27th IEEE VLSI Test Symposium (VTS)*, pages 53–58, 2009.

[26] Yanni Chen and Keshab K. Parhi. Small area parallel Chien search architectures for long BCH codes. *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, 12:545–549, 2004.

[27] Kuo-Liang Cheng, Jen-Chieh Yeh, Chih-Wea Wang, Chih-Tsun Huang, and Cheng-Wen Wu. RAMSES-FT: a fault simulator for flash memory testing and diagnostics. pages 281 – 286, 2002.

[28] Mei-Ling Chiang, Paul C. H. Lee, and Ruei-Chuan Chang. Using data clustering to improve cleaning performance for flash memory. *Software - Practice and Experience*, 29:267–290, 1999.

[29] R. Chien. Cyclic decoding procedures for Bose-Chaudhuri-Hocquenghem codes. *IEEE Transactions on Information Theory*, 10(4):357–363, oct 1964.

[30] Sau-Kwo Chiu, Jen-Chieh Yeh, Chih-Tsun Huang, and Cheng-Wen Wu. Diagonal test and diagnostic schemes for flash memories. In *Proc. International Test Conference*, pages 37–46, 7–10 Oct. 2002.

[31] Junho Cho and Wonyong Sung. Software implementation of Chien search process for strong BCH codes. In *IEEE International Symposium on Circuits and Systems (ISCAS)*, pages 1842–1845, may 2008.

[32] Junho Cho and Wonyong Sung. Efficient software-based encoding and decoding of BCH codes. *IEEE Transactions on Computers*, 58(7):878–889, july 2009.

[33] H. Choi, W. Liu, and W. Sung. VLSI implementation of BCH error correction for multilevel cell NAND flash memory. *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, 18:843–847, 2010.

[34] Jim Cooke. The inconvenient truths of NAND flash memory. `http://download.micron.com/pdf/presentations/events/flash_mem_summit_jcooke_inconvenient_truths_nand.pdf`, 2007.

[35] SanDisk Corporation. Sandisk flash-memory cards wear leveling. Technical Report 80-36-00278, October 2003.

[36] Corsair. USB flash wear-leveling and life span. Retrieved February 24, 2013 from the World Wide Web `http://docs.aboutnetapp.ru/FAQ_flash_drive_wear_leveling.pdf`, June 2007.

[37] CSIE. Csie. Retrieved February 24, 2013 from the World Wide Web `http://newslab.csie.ntu.edu.tw/~flash/index.php?SelectedItem=Traces`, 2005.

[38] Cypress. Cypress SONOS technology. Retrieved February 24, 2013 from the World Wide Web `http://www.cypress.com/?docID=30113`.

[39] R. Dan and R. Singer. Implementing MLC NAND flash for cost-effective, high-capacity memory. `http://support.gateway.com/s/Manuals/Desktops/5502664/Implementing_MLC_NAND_Flashwhite%20paper.pdf`, 2011.

[40] Datalight. XCfiles file system for next generation removable storage. Retrieved February 24, 2013 from the World Wide Web `http://www.datalight.com/products/embedded-file-systems/xcfiles`, 2010.

[41] B. De Salvo, G. Ghibaudo, G. Pananakakis, B. Guillaumot, P. Candelier, and G. Reimbold. A new extrapolation law for data-retention time-to-failure of non-volatile memories. *Electron Device Letters, IEEE*, 20(5):197 –199, may 1999.

[42] Eric Deal. Trends in NAND flash memory error correction. `http://cyclicdesign.com/whitepapers/Cyclic_Design_NAND_ECC.pdf`, 2011.

[43] Memory Technology Devices. NAND simulator. Retrieved February 24, 2013 from the World Wide Web `http://www.linux-mtd.infradead.org/faq/nand.html`, 2011.

[44] S. Di Carlo, M. Fabiano, M. Indaco, and P. Prinetto. ADAGE: An Automated Synthesis tool for Adaptive BCH-based ECC IP-Cores. *Proceedings of IEEE International Test Conference (ITC)*, page 15, 2012.

[45] S. Di Carlo, M. Fabiano, M. Indaco, and P. Prinetto. Design and Optimization of Adaptable BCH Codecs for NAND Flash Memories. *Elsevier Microprocessors and Microsystems (MICPRO), revisions being processed*, 2013.

[46] S. Di Carlo, M. Fabiano, R. Piazza, and P. Prinetto. Exploring modeling and testing of NAND flash memories. *Design Test Symposium EWDTS 2010 East West*, pages 47–50, 2010.

[47] S. Di Carlo, M. Fabiano, P. Prinetto, and M. Caramia. *Design Issues and Challenges of File Systems for Flash Memories*, chapter 1, pages 28 (pp.3–30). InTech, 2011, ISBN 9789533072722.

[48] N. Duann. Error correcting techniques for future NAND flash memory in SSD applications. Retrieved February 24, 2013 from the World Wide Web `http://www.bswd.com/FMS09/FMS09-201-Duann.pdf`, 2009.

[49] M. Fabiano and G. Furano. Nand flash storage technology for mission-critical space applications. *accepted for publication on IEEE Aerospace and Electronic Systems Magazine (AESS)*, -, 2013.

[50] M. Fabiano and P. Prinetto. FLARE: Technology roadmap. Retrieved February 24, 2013 from the World Wide Web `http://www.testgroup.polito.it/FLARE/FLARE-Roadmap.pdf`, 2010.

[51] Free Software Foundation. GDB: The GNU project debugger. Retrieved February 24, 2013 from the World Wide Web `http://www.gnu.org/s/gdb/`, 2011.

[52] Eran Gal and Sivan Toledo. Algorithms and data structures for flash memories. *ACM Comput. Surv.*, 37:138–163, June 2005.

[53] GNU. GNU octave. Retrieved February 24, 2013 from the World Wide Web `www.gnu.org/software/octave/`, 2010.

[54] M. J. E. Golay. Notes on digital coding. *Proceedings of The IEEE*, 37:657, 1949.

[55] J. Gray and C. van Ingen. Empirical measurements of disk failure rates and error rates. Retrieved February 24, 2013 from the World Wide Web `http://arxiv.org/ftp/cs/papers/0701/0701166.pdf`, 2011.

[56] S. Gregori, A. Cabrini, O. Khouri, and G. Torelli. On-chip error correcting techniques for new-generation flash memories. *Proceedings of the IEEE*, 91:602–616, 2003.

[57] Rw Hamming. Error Detecting and Error Correcting Codes. *Bell System Technical Journal*, 26:147–160, 1950.

[58] Yea-Ling Horng, Jing-Reng Huang, and Tsin-Yuan Chang. A realistic fault model for flash memories. *ATS '00: Proceedings of the 9th Asian Test Symposium*, page 274, 2000.

[59] IEEE Standards Department. IEEE standard definitions and characterization of floating gate semiconductor arrays. *IEEE Std 1005-1998*, 1998.

[60] D. Ielmini. Reliability issues and modeling of flash and post-flash memory. *Microelectronic Engineering*, 86:1870–1875, 2009.

[61] Intel. Understanding the Flash Translation Layer (FTL) specification, AP-684 (order 297816). Retrieved February 24, 2013 from the World Wide Web `http://staff.ustc.edu.cn/~jpq/paper/flash/2006-Intel%`

20TR-Understanding%20the%20flash%20translation%20layer%20(FTL)%20specification.pdf, Dec. 1998.

[62] F. Irom, D. N. Nguyen, M. L. Underwood, and A. Virtanen. Effects of scaling in SEE and TID response of high density NAND flash memories. 57(6):3329–3335, 2010.

[63] Isilon. OneFS. Retrieved February 24, 2013 from the World Wide Web http://www.isilon.com/onefs-operating-system, 2011.

[64] Lee Jae-Duk, Hur Sung-Hoi, and Choi Jung-Dal. Effects of floating-gate interference on NAND flash memory cell operation. *IEEE Electron Device Letters*, 23:264–266, 2002.

[65] Yeh Jen-Chieh, Wu Chi-Feng, Cheng Kuo-Liang, Chou Yung-Fa, Huang Chih-Tsun, and Wu Cheng-Wen. Flash memory built-in self-test using march-like algorithms. In *Proceedings of the First IEEE International Workshop on Electronic Design, Test and Applications*, pages 137–141, Christchurch , New Zealand, 29-31 Jan. 2002.

[66] Hsieh Jen-Wei, Tsai Yi-Lin, Kuo Tei-Wei, and Lee Tzao-Lin. Configurable flash-memory management: Performance versus overheads. *IEEE Trans. on Computers*, 57(11):1571–1583, Nov. 2008.

[67] Cho Junho and Sung Wonyong. Efficient software-based encoding and decoding of BCH codes. *IEEE Transactions on Computers*, 58(7):878–889, July 2009.

[68] J. Katcher. Postmark: A new file system benchmark. Retrieved February 24, 2013 from the World Wide Web https://koala.cs.pub.ro/redmine/attachments/download/605/Katcher97-postmark-netapp-tr3022.pdf, 2011.

[69] Atsuo Kawaguchi, Shingo Nishioka, and Hiroshi Motoda. A flash-memory based file system. In *Proceedings of the USENIX Annual Technical Conference*, TCON'95, pages 13–13, New Orleans, Louisiana, 16-20 Jan. 1995. USENIX Association.

[70] A. Keshk. Flash memory testing for realistic fault modeling. In *International Conference on Electrical, Electronic and Computer Engineering, 2004. ICEEC '04*, Sept. 2004.

[71] Kijun Lee, Sejin Lim, and Jaehong Kim. Low-cost, low-power and high-throughput BCH decoder for NAND Flash Memory. *IEEE International Symposium on Circuits and Systems (ISCAS)*, pages 413–415, may 2012.

[72] Sungjin Lee, Keonsoo Ha, Kangwon Zhang, Jihong Kim, and Junghwan Kim. FlexFS: a flexible flash file system for MLC NAND flash memory. In *Proceedings of the USENIX Annual Technical Conference*, USENIX'09, pages 9–9, San Diego, California, 14-19 June 2009. USENIX Association.

[73] S. Lin and D. Costello. *Error Control Coding: Fundamentals and Applications*. 2004.

[74] Wei Liu, Junrye Rho, and Wonyong Sung. Low-power high-throughput BCH error correction VLSI design for multi-level cell NAND flash memories. *Signal Processing Systems Design and Implementation, 2006. SIPS '06. IEEE Workshop on*, pages 303 –308, oct. 2006.

[75] M-Systems. Flash-memory Translation Layer for NAND flash (NFTL). Retrieved February 24, 2013 from the World Wide Web `http://www.freepatentsonline.com/5404485.pdf`, 1998.

[76] C. Manning. How YAFFS works. Retrieved February 24, 2013 from the World Wide Web `http://www.dubeiko.com/development/FileSystems/YAFFS/HowYaffsWorks.pdf`, 2010.

[77] C. Manning. How YAFFS handles NAND errors. Retrieved February 24, 2013 from the World Wide Web `http://yaffs.net/gitweb?p=yaffs-docs;a=blob;f=NANDFailureMitigation.odt`, 2011.

[78] MathWorks. Simulink HDL coder. Retrieved February 24, 2013 from the World Wide Web `http://www.mathworks.com/products/slhdlcoder/`, 2011.

[79] The Mathworks. Communication System Toolbox. `http://www.mathworks.nl/help/comm/ref/primpoly.html`, 2012.

[80] MemoryTechnologyDevice. Memory technology device (MTD). Retrieved February 24, 2013 from the World Wide Web `http://www.linux-mtd.infradead.org/`, 2010.

[81] R. Micheloni, A. Marelli, and R. Ravasio. *Error Correction Codes for Non-Volatile Memories*. Springer Publishing Company, 2008.

[82] R. Micheloni, R. Ravasio, A. Marelli, E. Alice, V. Altieri, A. Bovino, L. Crippa, E. Di Martino, L. D'Onofrio, A. Gambardella, E. Grillea, G. Guerra, D. Kim, C. Missiroli, I. Motta, A. Prisco, G. Ragone, M. Romano, M. Sangalli, P. Sauro, M. Scotti, and S. Won. A 4Gb 2b/cell NAND flash memory with embedded 5b BCH ECC for 36MB/s system read throughput. *Proceedings of IEEE Solid-State Circuits Conference (ISSCC)*, pages 497–506, 2006.

[83] Micron. NAND flash memory (MT29F4G08AAA, MT29F8G08BAA, MT29F8G08DAA, MT29F16G08FAA). Retrieved February 24, 2013 from the World Wide Web `http://www.datasheets.org.uk/dl/SFDatasheet-4/sf-00095776.pdf`, Feb. 2007.

[84] Micron. Hamming codes for NAND flash-memory devices overview. `http://download.micron.com/pdf/technotes/nand/tn2908.pdf`, 2011.

[85] Micron. A trillion bits on a fingertip. Retrieved February 24, 2013 from the World Wide Web `http://www.micron.com/about/blogs/2011/december/a-trillion-bits-on-a-fingertip`, 2011.

[86] Microsoft. Microsoft report. Retrieved February 24, 2013 from the World Wide Web `http://technet.microsoft.com/en-us/sysinternals/bb896646.aspx`, 2006.

[87] Microsoft. Description of the exFAT file system driver update package. Retrieved February 24, 2013 from the World Wide Web `http://support.microsoft.com/kb/955704/en-us`, 2009.

[88] Microsoft. exFAT file system. Retrieved February 24, 2013 from the World Wide Web `http://www.microsoft.com/about/legal/en/us/IntellectualProperty/IPLicensing/Programs/exFATFileSystem.aspx`, 2011.

[89] Microsoft. File system functionality comparison. Retrieved February 24, 2013 from the World Wide Web `http://msdn.microsoft.com/en-us/library/ee681827(v=vs.85).aspx`, 2011.

[90] N. Mielke, T. Marquart, Ning Wu, J. Kessenich, H. Belgal, E. Schares, F. Trivedi, E. Goodness, and L.R. Nevill. Bit error rate in NAND flash memories. *Proceedings of the IEEE International Reliability Physics Symposium*, pages 9–19, 2008.

[91] Park Mincheol, Kim Keonsoo, Park Jong-Ho, and Choi Jeong-Hyuck. Direct field effect of neighboring cell transistor on cell-to-cell interference of NAND flash cell arrays. *IEEE Electron Device Letters*, 30:174–177, 2009.

[92] M.G. Mohammad, K. K. Saluja, and Alex S. Yap. Fault models and test procedures for flash memory disturbances. *J. Electron. Test.*, 17(6):495–508, 2001.

[93] M.G. Mohammad and K.K. Saluja. Flash memory disturbances: modeling and test. *VLSI Test Symposium, 19th IEEE Proceedings on. VTS 2001*, pages 218 –224, 2001.

[94] M.G. Mohammad and K.K. Saluja. Optimizing program disturb fault tests using defect-based testing. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, 24(6):905 – 915, june 2005.

[95] M.G. Mohammad and K.K. Saluja. Testing flash memories for tunnel oxide defects. In *Proc. 21st International Conference on VLSI Design VLSID 2008*, pages 157–162, 4–8 Jan. 2008.

[96] M.G. Mohammad, K.K. Saluja, and A. Yap. Testing flash memories. In *Proceeding of the Thirteenth International Conference on VLSI Design*, pages 406–411, Calcutta, India, 4-7 Jan. 2000. IEEE Computer Society.

[97] M.G. Mohammad and Laila Terkawi. Fault collapsing for flash memory disturb faults. In *ETS '05: Proceedings of the 10th IEEE European Symposium on Test*, pages 142–147, Washington, DC, USA, 2005. IEEE Computer Society.

[98] Todd K. Moon. *Error Correction Coding: Mathematical Methods and Algorithms*. Wiley-Interscience, 2005.

[99] NetApp. Postmark. Retrieved February 24, 2013 from the World Wide Web `https://launchpad.net/ubuntu/+source/postmark`, 2010.

[100] D. N. Nguyen, S. M. Guertin, and J. D. Patterson. Radiation tests on 2Gb NAND flash memories. In *Proc. IEEE Radiation Effects Data Workshop*, pages 121–125, 2006.

[101] D. N. Nguyen and L. Z. Scheick. TID, SEE and radiation induced failures in advanced flash memories. In *Proc. IEEE Radiation Effects Data Workshop*, pages 18–23, 2003.

[102] T. R. Oldham, M. Friendlich, J. W. Howard, M. D. Berg, H. S. Kim, T. L. Irwin, and K. A. LaBel. TID and SEE response of an advanced samsung 4gb NAND flash memory. In *Proc. IEEE Radiation Effects Data Workshop*, volume 0, pages 221–225, 2007.

[103] Timothy R. Oldham et al. Correlation of Pulsed Laser and Milli-Beam TM Heavy Ion Results for NAND Flash Memory. *To be presented at the 1st NASA Electronic Parts and Packaging (NEPP) Program Electronic Technology Workshop June 22-24, 2010, NASA GSFC, Greenbelt, MD.*, 2012.

[104] ONFI. Open NAND flash interface (ONFi) specification. Retrieved February 24, 2013 from the World Wide Web `http://www.onfi.org/~/media/ONFI/specs/onfi_3_1_spec.pdf`, 2010.

[105] Perisoft. Bushound. Retrieved February 24, 2013 from the World Wide Web `http://www.perisoft.net/bushound/`, 1998.

[106] J.A. Perschy. Space systems general-purpose processor. *Aerospace and Electronic Systems Magazine, IEEE*, 15(11):15 – 19, nov 2000.

[107] K. Rajesh Shetty, U. Sripati, H. Prashantha Kumar, and B. Shankarananda. Synthesis of BCH codes for enhancing data integrity in flash memories. *International Conference on Industrial and Information Systems (ICIIS)*, pages 119–124, 29 2010-aug. 1 2010.

[108] I. S. Reed and G. Solomon. Polynomial Codes Over Certain Finite Fields. *Journal of the Society for Industrial and Applied Mathematics*, 8:300–304, 1960.

[109] UMass Trace Repository. Storage traces. Retrieved February 24, 2013 from the World Wide Web `http://traces.cs.umass.edu/index.php/Storage/Storage`, 2011.

[110] Mendel Rosenblum and John K. Ousterhout. The design and implementation of a log-structured file system. *ACM Trans. Comput. Syst.*, 10:26–52, February 1992.

[111] Samsung. XSR1.5 bad block management. Retrieved February 24, 2013 from the World Wide Web http://www.findthatpdf.com/download.php?i=4450573&t=hPDF, May 2007.

[112] Samsung. XSR1.5 wear leveling. Retrieved February 24, 2013 from the World Wide Web http://en.pudn.com/dl.asp?id=965593, May 2007.

[113] Samsung. K9XXG08UXM NAND flash memory. http://www.arm9board.net/download/FL6410/datasheet/K9G8G08.pdf, 2011.

[114] Samsung. OneNAND. Retrieved February 24, 2013 from the World Wide Web https://u-boot-all-in-one.googlecode.com/files/onenand_brochure_200609.pdf, 2011.

[115] Ltd Samsung Electronics Co. NAND flash ECC algorithm (error checking & correction). Retrieved February 24, 2013 from the World Wide Web http://www.elnec.com/sw/samsung_ecc_algorithm_for_256b.pdf, June 2004.

[116] SanDisk. TrueFFS. http://www.texim-europe.com/promotion/119/trueffs

[117] SanDisk. Sandisk's know-how strengthens the SSD industry. Retrieved February 24, 2013 from the World Wide Web http://www.sandisk.com/business-solutions/ssd/technical-expertise-metrics, 2011.

[118] H. Schmidt, D. Walter, M. Brüggemann, F. Gliem, R. Harboe-Sørensen, and A. Virtanen. Heavy ion SEE studies on 4-gbit NAND-flash memories. *Radiation Effects on Components and Systems (RADECS)*, September 2007.

[119] H. Schmidt, D. Walter, F. Gliem, B. Nickson, R. Harboe-Sorensen, and A. Virtanen. TID and SEE tests of an advanced 8 Gbit NAND-flash memory. In *Proceedings of IEEE Radiation Effects Data Workshop, 2008*, pages 38 –41, July 2008.

[120] Bianca Schroeder, Eduardo Pinheiro, and Wolf-Dietrich Weber. Dram errors in the wild: A large-scale field study. In *SIGMETRICS*, 2009.

[121] SD Association. SDXC. Retrieved February 24, 2013 from the World Wide Web https://www.sdcard.org/consumers/sdxc_capabilities/, 2011.

[122] Segger. J-link flash breakpoints. Retrieved February 24, 2013 from the World Wide Web http://www.segger.com/jlink-unlimited-flash-breakpoints.html, 2005.

[123] Segger. emfile file system. Retrieved February 24, 2013 from the World Wide Web http://www.segger.com/cms/emfile.html, 2010.

[124] Lim Seung-Ho and Park Kyu-Ho. An efficient NAND flash file system for flash memory storage. *IEEE Transactions on Computers*, 55(7):906–912, July 2006.

[125] Abraham Silberschatz, Peter Baer Galvin, and Greg Gagne. *Operating System Concepts*. Wiley Publishing, 2008.

[126] SourcenavNGDevelopmentGroup. Source navigator. Retrieved February 24, 2013 from the World Wide Web http://sourcenav.sourceforge.net/online-docs/index.html, 2010.

[127] Spansion. What types of ECC should be used on flash-memory? Retrieved February 24, 2013 from the World Wide Web http://www.spansion.com/Support/Application%20Notes/Types_of_ECC_Used_on_Flash_AN.pdf, 2011.

[128] Scott Speaks. Reliability and MTBF overview. Retrieved February 24, 2013 from the World Wide Web http://www.vicorpower.com/documents/quality/Rel_MTBF.pdf, 2009.

[129] M. Staehle et al. Sentinel-2 MMFU: The first European Mass Memory System based on NAND-Flash Storage Technology. *Proceedings of the DASIA (DAta Systems In Aerospace) 2011 Conference, San Anton, Malta, May 17-20, 2011, ESA SP-694, August 2011.*

[130] Sheng-Jie Syu and Jing Chen. An active space recycling mechanism for flash storage systems in real-time application environment. In *Proc. 11th IEEE International Conference on Embedded and Real-Time Computing Systems and Applications*, pages 53–59, 17–19 Aug. 2005.

[131] Richard Tervo. EE4253 Digital Communications. http://www.ee.unb.ca/cgi-bin/tervo/bch.pl, 2010.

[132] D. Woodhouse. JFFS : The journalling flash file system. In *Proceedings of the Ottawa Linux Symposium*, Ottawa, Ontario Canada, 26-29 July 2001.

[133] D. Woodhouse. JFFS2: The journalling flash file system, version 2. Retrieved February 24, 2013 from the World Wide Web `http://sourceware.org/jffs2/`, 2009.

[134] Michael Wu. The architecture of eNVy, a non-volatile, main memory storage system. Master's thesis, Rice University, 1994.

[135] Michael Wu and Willy Zwaenepoel. envy: a non-volatile, main memory storage system. *SIGOPS Oper. Syst. Rev.*, 28:86–97, Nov. 1994.

[136] www.rfic.co.uk. Component reliability tutorial. Technical report, 2009.

[137] Yu Xin, Rong Chun-ming, and Huang Ben-xiong. A flexible garbage collect algorithm for flash storage management. In *Proc. Second International Conference on Future Generation Communication and Networking FGCN '08*, volume 1, pages 354–357, 13–15 Dec. 2008.

[138] E. Yaakobi, J. Ma, A. Caulfield, L. Grupp, S. Swanson, P.H. Siegel, and Wolf J.K. Error correction coding for flash memories. `http://cmrr.ucsd.edu/research/documents/Number31Winter2009_000.pdf`, 2009.

[139] J. C. Yeh, Kuo-Liang Cheng, Yung-Fa Chou, and Cheng-Wen Wu. Flash memory testing and built-in self-diagnosis with march-like test algorithms. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, 26(6):1101–1113, June 2007.

[140] Xu Youzhi. Implementation of Berlekamp-Massey algorithm without inversion. *IEEE Proceedings of Communications, Speech and Vision*, 138:138–140, 1991.

[141] Chen Yuan. Flash memory reliability NEPP 2008 task final report. Retrieved February 24, 2013 from the World Wide Web `http://trs-new.jpl.nasa.gov/dspace/bitstream/2014/41262/1/09-9.pdf`, 2008.

[142] C. Zambelli, M. Indaco, M. Fabiano, S. Di Carlo, P. Prinetto, P. Olivo, and D. Bertozzi. A cross-layer approach for new reliability-performance trade-offs in MLC NAND flash memories. *Proceedings of Design, Automation & Test in Europe (DATE)*, pages 881–886, 2012.

[143] W. Zhang and S. H. Tan. Oxide particle induced leakage in flash memory endurance test. In *Proc. 44th Annual. IEEE International Reliability Physics Symposium*, pages 608–610, 26–30 March 2006.