Springer Series in Advanced Microelectronics 37

Rino Micheloni
Alessia Marelli
Kam Eshghi

Inside Solid State Drives (SSDs)

Foreword by Andrew Viterbi

Springer

Springer Series in
ADVANCED MICROELECTRONICS 37

Rino Micheloni • Alessia Marelli • Kam Eshghi

Inside Solid State Drives (SSDs)

Springer

Rino Micheloni
Integrated Device Technology
Enterprise Computing Division
via Cardano 2
20864 Agrate Brianza, Italy

Alessia Marelli
Integrated Device Technology
Enterprise Computing Division
Via Cardano 2
20864 Agrate Brianza, MB, Italy

Kam Eshghi
Integrated Device Technology, Inc.
Enterprise Computing Division
6024 Silver Creek Valley Rd
95138 San Jose California, USA

ISSN 1437-0387
ISBN 978-94-007-5145-3 ISBN 978-94-007-5146-0 (eBook)
DOI 10.1007/978-94-007-5146-0
Springer Dordrecht Heidelberg New York London

Library of Congress Control Number: 2012944752

Printed on acid-free paper

Springer is part of Springer Science+Business Media (www.springer.com)

To my wife Sabrina, and my daughters Laura and Greta

Rino

To my parents Angelo and Margherita and to my sister Silvia for their continuous support and unconditional love

Alessia

To my parents, Alireza and Tara, who have blessed me with their selfless love

Kam

Foreword

Error Correcting Coding for Solid State Disk Data Storage

Wireless communication had existed for half a century when Information Theory was expounded by Claude Shannon in the Bell System Technical Journal in 1948. Error correcting coding followed in primitive formulations which brought early digital communication systems only a short way toward the Shannon capacity limit. Various generations of algebraic codes: Hamming, BCH and Reed Solomon made gradual progress. With the advent of digital satellite transmission and soft-decision decoding of convolutional codes, the gap between uncoded performance and the Shannon limit was cut in half. Similar technology was used in second and third generation (2 G and 3 G) mobile phone voice modems. Finally turbo codes and low-density parity check (LDPC) codes, which arrived about two decades ago, gradually were shown to greatly decrease the distance to the capacity limit. These technologies have entered predominant use for data transmission in 3 G and 4 G mobile modems.

High density data storage technology has followed a similar trajectory though with a more contracted time span. BCH and Reed Solomon codes were the norm until recently for hard disk drives (HDD). Recently though LDPC has taken root here too with major improvements in data density and reading and writing controller speeds. With the advent of the "smart phones" and tablets, solid state drives (SSD) became ever more important for their low latency and low power operation. For this use LDPC is becoming the norm as well. This book which covers all aspects of SSD technology also provides coverage of the important topic of ECC.

Andrew Viterbi
President, Viterbi Group, LLC
La Jolla, California, USA

Preface

Solid State Drives (SSDs) are gaining momentum in enterprise and client applications, replacing Hard Disk Drives (HDDs) by offering higher performance and lower power. In the enterprise, developers of data center server and storage systems have seen CPU performance growing exponentially for the past two decades, while HDD performance has improved linearly for the same period. Additionally, multi-core CPU designs and virtualization have increased randomness of storage I/Os. These trends have shifted performance bottlenecks to enterprise storage systems. Business critical applications such as online transaction processing, financial data processing and database mining are increasingly limited by storage performance.

In client applications, small mobile platforms are leaving little room for batteries while demanding long life out of them. Therefore, reducing both idle and active power consumption has become critical. Additionally, client storage systems are in need of significant performance improvement as well as supporting small robust form factors. Ultimately, client systems are optimizing for best performance/power ratio as well as performance/cost ratio.

SSDs promise to address both enterprise and client storage requirements by drastically improving performance while at the same time reducing power.

Inside Solid State Drives walks the reader through all the main topics related to SSDs. Chapter 1 provides an overview of the SSD market and applications, including a review of client PC, tablet, and enterprise computing usage models.

A Solid State Drive is a very complex system: Chap. 2 contains an overview of the main blocks, including hardware and software.

Chapters 2 and 3 cover different SSD implementations with host interfaces ranging from SAS/SATA to PCI Express (PCIe). SAS/SATA offer compatibility with legacy storage infrastructure. However, for many applications, NAND Flash read and write speeds are exceeding the capabilities of these legacy interconnects. PCIe SSDs overcome this bottleneck and deliver unparalleled performance while, at the same time, reducing latency, power and cost by eliminating the traditional storage infrastructure and attaching directly to a platform's PCIe I/O interconnect.

SSDs and HDDs can also be combined together in various forms, as explained in Chap. 4 where "hybrid" storage is analyzed.

At the end of the day, a SSD is made up of NAND memories and a controller. Therefore, to understand SSDs it is important to understand all the basics of NAND Flash technology (Chap. 5) as well as design (Chap. 6).

To realize a low-power high-speed SSD, the overall performance of the NAND Flash memory and the NAND controller should be optimized by co-designing both NAND and controller circuits. Chapter 7 describes the most advanced circuits in this field. Furthermore, 3D-integration in the SSD system becomes a key topic and an example of low power 3D-integrated SSD is shown.

When aiming to replace HDDs, particularly in enterprise applications, another key consideration is reliability. SSDs are complex electronic systems prone to wear-out and failure mechanisms mainly related to NAND. SSD reliability is analyzed at different levels in Chap. 8. The basic physical mechanisms affecting the traditional floating-gate cells and the possibility of anomalous erratic behavior is discussed, as well as disturbs arising because several cells share the same control lines. Solutions adopted to improve system reliability are presented, such as the use of RAID and protection against power loss during write operations. Test methods for endurance and retention verification are also described.

The physical constraints of Flash memory pose a lifetime limitation on these storage devices. Multilevel Flash technologies (MLC) further degrade endurance, as 2 bits are stored in the same physical cell. As a result, NAND devices may experience an unexpectedly short lifespan, especially when accessing these devices at high frequencies. In order to enhance the endurance, wear leveling algorithms are used to evenly erase blocks. Chapter 9 describes some existing wear leveling algorithms, highlighting their pros and cons.

Despite all the possible Flash management algorithms run by the memory controller, the residual BER needs to be properly managed in order to achieve a reliable system. That is why *Error Correction Codes* (ECCs) are so important in SSD design. Two main issues arise when an ECC is used inside an SSD. First, the ECC engine should not limit the performance of the drive. This requirement is addressed with a hardware ECC implementation that supports multiple devices (channels) in parallel. Second, ECC must avoid erroneous corrections when the error correction capability of the code is overcome; that is, it must have a high detection property.

Nowadays, the most popular ECC approach in commercial SSDs is BCH, which is covered in Chap. 10. As the NAND technology scales down, NAND raw BER becomes worse and a more powerful ECC is needed. Chapter 11 covers LDPC codes which are capable to get closer to the Shannon limit; in other words, they can handle higher BER at the expense of a higher complexity.

SSD security is another key requirement because sensitive data must be protected against external attacks. Unfortunately, existing methods in the HDD world cannot be applied to SSDs. These days encryption is the most popular method to secure SSDs. Chapter 12 covers encryption basics and their application to solid state drives.

Finally, Chap. 13 covers Flash signal processing techniques. When NAND raw BER overcomes the Shannon limit, NAND-controller interaction jumps to a deeper level. In other words, parameters like retention time, floating gate coupling, number of erase cycles, etc., need to be considered during runtime.

We are in the midst of an exciting storage market transition, where Flash is expanding its reach to replace HDDs with dramatically faster and more efficient SSDs. After reading this book, the reader will get a comprehensive look at SSD applications and technologies. As you'll see, a Solid State Drive is a complex mix of digital and analog circuits working in concert with firmware and I/O software protocols. We hope you enjoy this tour inside Solid State Drives.

Rino, Alessia and Kam
The Editors

About the Editors

Rino Micheloni (rino.micheloni@ieee.org) is Lead Flash Technologist at IDT (Integrated Device Technology). He has 18 years experience in NAND/NOR Flash memory design, architecture and algorithms as well as the related intellectual property. Before IDT, he was Senior Principal for Flash and Director of Qimonda's design center in Italy, developing 36 and 48 nm NAND memories. From 2001 to 2006 he managed the Napoli design center of STMicroelectronics focusing on the development of 90 nm and 60 nm MLC NAND Flash. Before that, he led the development of MLC NOR Flash. He is co-author of 103 U.S. patents and five Springer books on NOR/NAND/ECC/SSD. He is IEEE Senior Member and received the STMicroelectronics Exceptional Patent in 2003 and 2004, and the Qimonda IP Award in 2007.

Alessia Marelli was born in Bergamo, Italy in 1980. She received her degree in Mathematical Science from "Università degli Studi di Milano – Bicocca", Italy in 2003 with a thesis about ECC applied to Flash Memories. In 2003 she joined STMicroelectronics, Agrate B., Italy. She was involved in digital design of Multilevel NAND Memories, especially redundancy, ECC and algorithms. In 2007, she joined Qimonda as senior digital designer. In 2009 she joined Integrated Device Technology (IDT) as senior digital designer, where she takes care of ECC applied to SSD. She is co-author of some patents regarding Redundancy and ECC applied to Flash Memories. She is co-author of *Memories in Wireless Systems* (Springer, 2008), *Error Correction Codes for Non-Volatile Memories* (Springer, *2008), Inside NAND Flash Memories* (Springer, 2010).

Kam Eshghi is Sr. Director of Marketing in Enterprise Computing Division of IDT. Kam leads IDT's business strategy, marketing and business development for flash controllers and PCIe switches. Kam drove the creation of IDT's PCIe Enterprise Flash Controller product line and established IDT as a leader in this market.

Kam has more than 18 years of industry experience. Prior to joining IDT, Kam built product lines in server and networking markets at HP, Intel, Crosslayer Networks, and Synopsys. During his time at Intel, Kam led a team to define strategy and product roadmaps for server chipsets. Previous to that as Vice President of Marketing and Business Development at Crosslayer Networks, Kam defined a family of Ethernet-IP switch processors and led all customer engagements, ultimately leading to LSI's acquisition.

Kam holds a M.S. in Electrical Engineering & Computer Science from Massachusetts Institute of Technology, and a Master's in Business Administration and B.S. in Electrical Engineering & Computer Science both from University of California at Berkeley.

Acknowledgements

After completing a book on a complex system like a Solid State Drive, we wish to thank all the authors of the contributed chapters.

We are especially grateful to Luca Crippa for his tremendous dedication to this project.

We also want to thank Mark de Jongh (Springer) for giving us the opportunity to publish this work and Cindy Zitter (Springer) for her continuous support.

Last but not least, we express our gratitude to all the people who reviewed the chapters.

Rino Micheloni, Alessia Marelli, and Kam Eshghi

Contents

Chapter 1
SSD Market Overview

G. Wong

Abstract The unparalleled cost and form factor advantages of NAND flash memory has driven 35 mm photographic film, floppy disks and one-inch hard drives to extinction. Due to its compelling price/performance characteristics, NAND Flash memory is now expanding its reach into the once-exclusive domain of hard disk drives and DRAM in the form of Solid State Drives (SSDs). Driven by the proliferation of thin and light mobile devices and the need for near-instantaneous accessing and sharing of content through the cloud, SSDs are expected to become a permanent fixture in the computing infrastructure.

1.1 Computing Market and Applications

In the January 2006 issue of IEEE Spectrum, *Solid State Drives* (SSDs) developed by Samsung were declared a "Loser: Too Little, Too Soon."[1] The article argued that SSDs were too expensive at $45/GB and that *Hard Disk Drives* (HDDs) outclassed SSDs in performance. Today SSDs for PCs are now priced twenty times less than the price quoted in the IEEE Spectrum article, sequential read and write performance are close to saturating SATA 6 Gbps interface speeds and random performance is at least an order of magnitude better than HDDs.

During this time, NAND Flash memory manufacturers, Flash memory card suppliers, controller makers and startups have piled into the market. Even HDD vendors are being forced to react to the stampede by developing their own SSD offerings. Altogether, there may be 200 companies offering SSD products.

[1]Loser: Too Little Too Late, Harry Goldstein, IEEE Spectrum, January 2006.

G. Wong (✉)
Forward Insights, Toronto, Canada
e-mail: greg@forward-insights.com

Prior to the current interest in NAND Flash-based SSDs, SSDs have a long and uncolorful history, starting out with RAM-based SSDs pioneered by StorageTek in 1978. It wasn't until late 1980's and early 1990's when the first Flash memory-based SSDs were developed. Western Digital demoed a 2.5″ NAND Flash SSD in 1989, however the main promoters of Flash-based SSDs, SanDisk and Intel, based their SSDs on NOR Flash technology in the early 1990's. Due to the higher cost of NOR Flash versus DRAM, Flash-based SSDs were relegated to niche markets.

M-Systems pioneered Flash memory-based SSDs in 1995 for use in industrial and military applications. However, it wasn't until NAND Flash memory surpassed DRAM in process technology in 2004 that the economics of NAND Flash memory opened up new opportunities for SSDs in enterprise and client computing applications just a few years later.

The following provides an overview of the computing market and applications for SSDs.

1.2 SSD Overview

1.2.1 What Is a Solid State Drive?

A solid state drive (SSD) is a storage device that incorporates solid-state memory and emulates a hard disk drive to store data. Because a SSD emulates a hard disk drive, it usually uses hard disk drive interfaces and protocols such as Parallel ATA, Serial ATA, Serial Attached SCSI and Fibre Channel and can readily replace it in most applications. However, SSDs based on non-HDD interfaces and protocols such as USB and PCI Express are also available.

As CPU performance has scaled through faster cores and multi-core architectures, hard disk drive (HDD) performance has stalled due to limitations in increasing the rotational speed of the magnetic media. Since 1996, CPU computing and processing performance of mobile and desktop systems has grown a remarkable 30 times whereas the performance of hard disk drives has managed to grow around 30% over the same period (Fig. 1.1).

Solid state storage employing NAND Flash memories offer the promise of improving the performance and reducing the access latency of storage as a replacement or complement to HDDs in both client and enterprise computing environments.

1.2.2 The Memory Hierarchy

The memory hierarchy along with accompanying attributes are summarized in Fig. 1.2. As it is shown, there is an inverse relationship between cost and performance. The temporary storage of information and the random accessing of data become more important as storage technologies interface more directly with the processor of an electronic system.

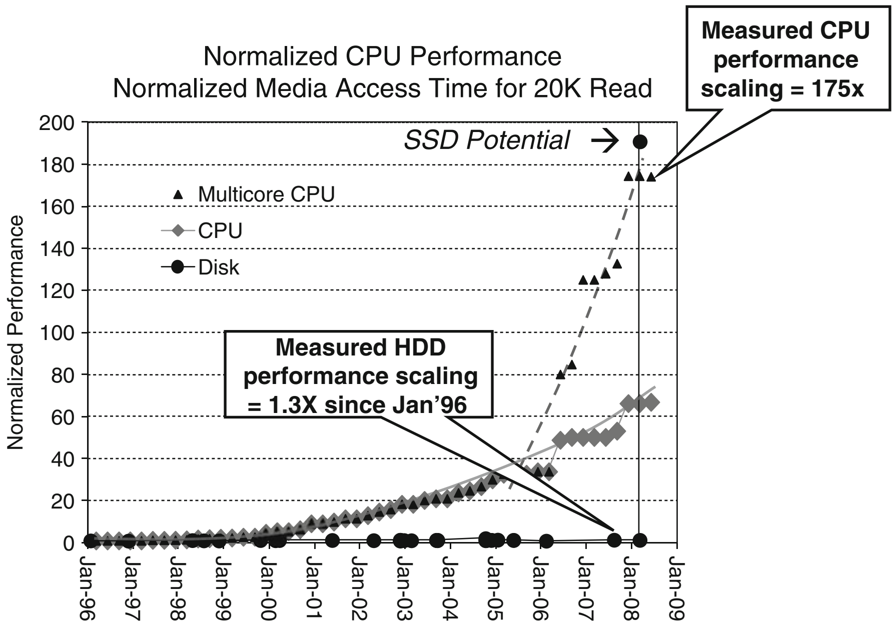


Fig. 1.1 Normalized CPU and HDD performance (Source: Intel measurements)

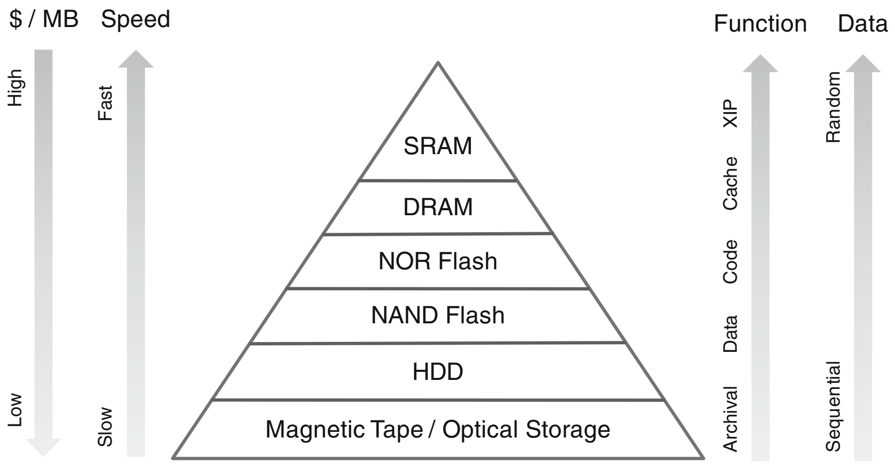


Fig. 1.2 Memory hierarchy (Created by Forward Insights)

At the bottom of the pyramid is magnetic tape or optical storage. Magnetic tape/optical storage has the lowest cost per MB of all the technologies and stores data sequentially. It is primarily used for archiving.

The next level of storage technology, hard disk drives is more expensive but provides better performance attributes and is used for data storage as well as archival storage. HDD technology performs best when the data is sequential.

The realm of semiconductor storage technologies is next. NAND Flash has the lowest cost of all the semiconductor storage technologies. Its serial architecture means it is more well suited to handling sequential data and its retention characteristics of 10 years makes it suitable as a data storage medium and less so for archival storage. NAND Flash has also been used to store code in store-and-download configurations in cellular phones as well as a cache in PC and enterprise systems. However, caching involves the reading and writing of small random files which is better suited to semiconductor memories with a parallel architecture.

In contrast to NAND Flash, NOR Flash has a parallel architecture making it suitable for storing and executing code. NOR Flash has much faster read and program times than NAND Flash but higher cost. Both NAND and NOR Flash are non-volatile memories which means they continue to retain information after power is switched off.

DRAM and SRAM are volatile semiconductor memories requiring power to maintain the information. Both memories are mainly used as a cache and code can be executed directly from the devices. Due to its fast speeds and compatibility with the CMOS logic process, SRAM is used extensively as on-chip cache for CPU designs.

Historically, SSDs have employed SRAM and DRAM, particularly in enterprise storage applications to improve overall system performance; however, as the cost of NAND Flash has fallen dramatically since 2004, NAND Flash SSDs have begun to encroach on the enterprise space as well as in the consumer space.

The crossover point between DRAM and NAND Flash in terms of process technology occurred in 2005 at the 90 nm generation. Ever since, NAND Flash has become and remained the main process technology driver.

1.2.3 SSD vs. HDD

Figure 1.3 compares the major components of a SSD and HDD. The HDD, being based on storage in a spinning magnetic platter, requires an actuator and actuator arm to move the head to the appropriate sector to be read or written. The HDD controller, particularly if it is a system-on-chip, incorporates the processor, servo control logic, interface, error correction code, disk sequencer and buffer controller. HDDs typically contain a DRAM buffer to hide the seek time.

In contrast, the SSD contains no mechanical parts and consists of a few major components: Flash memory, SSD controller, connector, DRAM buffer, PCB and passives.

Because of the simple structure of the SSD, the main cost driver for SSDs is the Flash memory. Also due to the lack of mechanical parts, failure rates can be significantly lower than for HDDs.

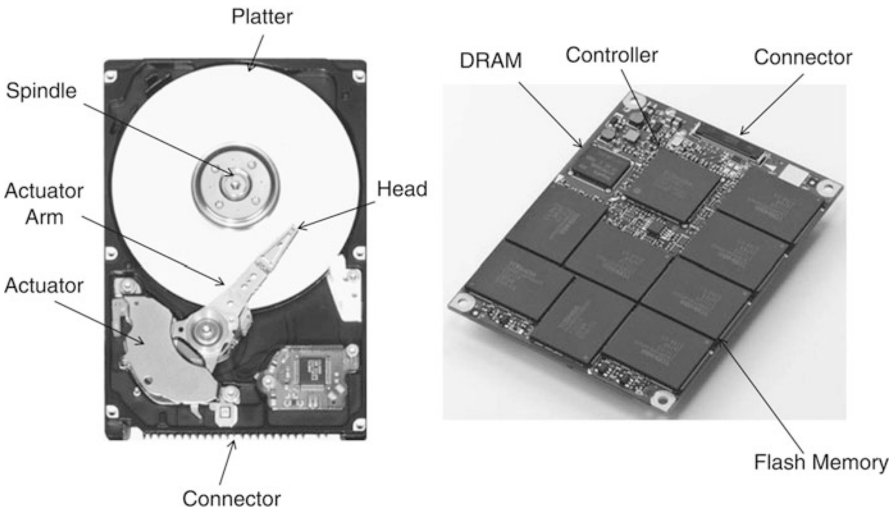


Fig. 1.3 Major components of SSD and HDD (Source: Toshiba Corp., Forward Insights)

Data from Intel Corp. has shown lower annual failure rates (AFR) of 90% or better[2] for SSDs compared to HDDs which reduces user downtime as well as IT costs for troubleshooting, rebuilding laptops and providing loaner PCs during troubleshooting or repair.

1.3 Client PC Applications

SSDs are being offered as a replacement for HDDs in client PC systems. To facilitate this replacement, SSDs have been designed in standard 2.5″ and 3.5″ HDD form factors. As illustrated in Fig. 1.4, SSDs provide an advantage over HDDs in a host of performance and power metrics. However, SSDs suffer from high costs – roughly ten times the $/GB of a HDD – which limits their capacity footprint in systems with a storage budget.

The fast responsiveness and better performance of a SSD over a HDD results in productivity benefits. Table 1.1 summarizes the time savings for a notebook PC with Intel’s 160 GB series 320 SSD running Windows 7 for various tasks such as boot up, standby, e-mail usage, etc. versus a HDD-based system. Over a 36 month PC lifecycle, 80.4 h can be saved translating into cost savings of $4,821 assuming an employee salary of $60/h.

[2]http://www.intel.com/content/dam/doc/white-paper/enterprise-reliability-accelerating-deployment-of-intel-ssds-paper.pdf

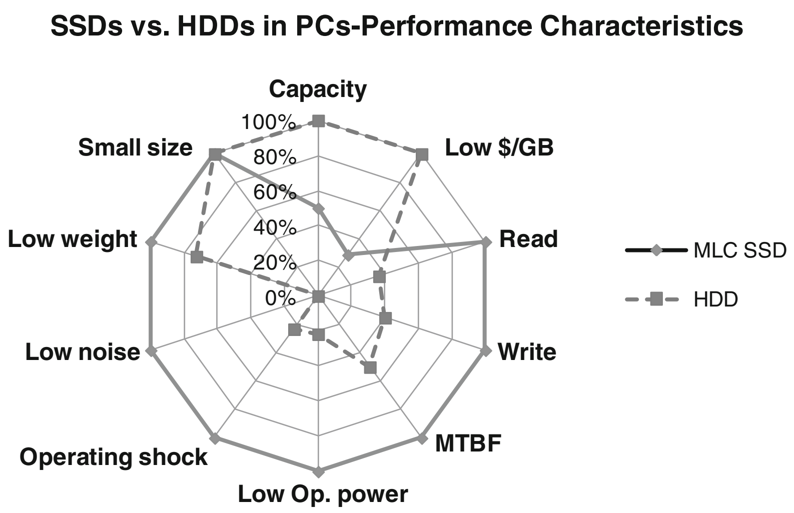
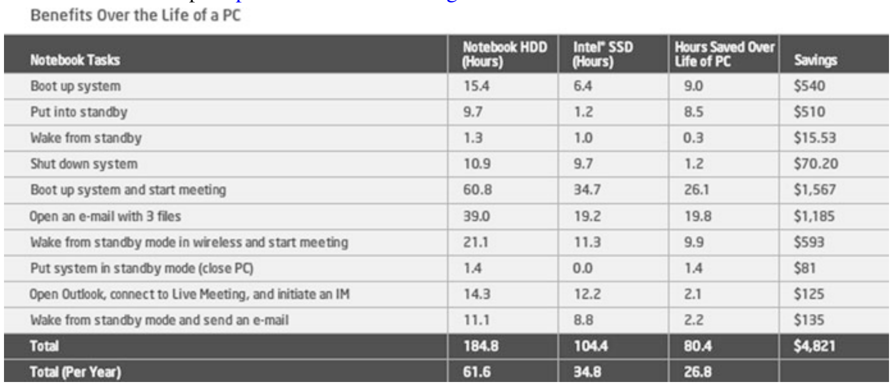


Fig. 1.4 2.5″ SSDs vs. HDDs in PCs – performance characteristics (Created by Forward Insights)

Table 1.1 Intel Corp. http://www.intel.com/design/flash/NAND/tco_tool/demo.htm



Benefits Over the Life of a PC

Notebook Tasks	Notebook HDD (Hours)	Intel® SSD (Hours)	Hours Saved Over Life of PC	Savings
Boot up system	15.4	6.4	9.0	$540
Put into standby	9.7	1.2	8.5	$510
Wake from standby	1.3	1.0	0.3	$15.53
Shut down system	10.9	9.7	1.2	$70.20
Boot up system and start meeting	60.8	34.7	26.1	$1,567
Open an e-mail with 3 files	39.0	19.2	19.8	$1,185
Wake from standby mode in wireless and start meeting	21.1	11.3	9.9	$593
Put system in standby mode (close PC)	1.4	0.0	1.4	$81
Open Outlook, connect to Live Meeting, and initiate an IM	14.3	12.2	2.1	$125
Wake from standby mode and send an e-mail	11.1	8.8	2.2	$135
Total	**184.8**	**104.4**	**80.4**	**$4,821**
Total (Per Year)	**61.6**	**34.8**	**26.8**	

Intel Corp. has retrofitted their employee notebooks with SSDs instead of HDDs. Based on empirical data and estimates from empirical data, SSDs have a 90% lower failure rate than HDDs over a 3-year refresh period. This translates into 90% less employee time lost due to drive failures and a 96% reduction in IT support time for PC rebuilds. These are direct cost savings and do not include the cost of loaner PCs which would tide the employee over while his/her PC is being re-built and should also be included in the calculation.

Time savings due to faster booting and loading of software is estimated at 44%. In addition, SSD-based notebooks showed a lower thermal footprint as they ran 12% cooler than HDD-based notebooks (Fig. 1.5).

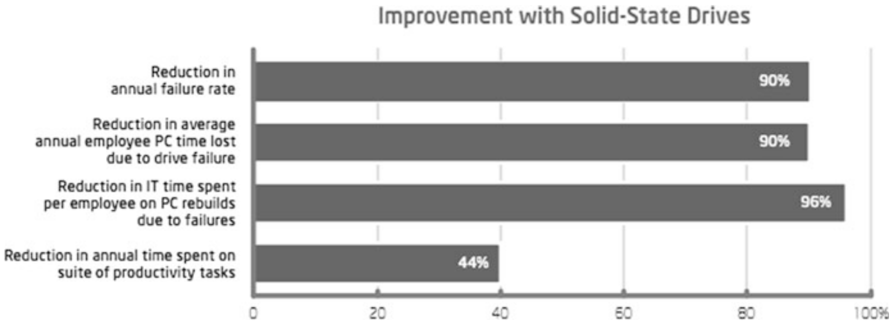


Fig. 1.5 SSD Benefits compared to HDDs (Source: Intel)

Based on current SSD prices, the reduced failure rates and productivity benefits of a SSD are already more than sufficient to justify the adoption of SSD-based notebooks for businesses.

In addition, large capacity storage is not a prerequisite in the corporate environment where files can be stored and pulled from the company intranet. Local storage of 128 GB is more than sufficient to store the operating system, work applications and files.

1.4 Notebook PCs and Tablets

A SSD actually does not have to come packaged in a HDD form factor. The NAND Flash memory components and SSD controller can be soldered onto a printed circuit board or the NAND Flash memory die and controller die can be packaged in a BGA package.

These small form factors enable thin, lightweight portable devices such as ultra-thin laptops and tablets. Ultra-thin laptops which include Apple Corp.'s MacBook Air and ultrabooks are less than 21 mm or 0.8″ thick making it difficult to incorporate a thin 7 mm high 2.5″ HDD. A modular SSD such as the Toshiba Blade X-Gale SSD (Fig. 1.6) is only half the z-height of a thin HDD.

To make a slim, sturdy outline, ultra-thin laptops will require aluminum unibody or glass fiber casing. Lithium-polymer batteries, such as those used in the MacBook which are roughly one-third the thickness of conventional cylindrical lithium-ion batteries, may also be required. These design modifications along with others including SSDs raises the bill of material of ultra-thin laptops. However, costs are expected to fall with economies of scale due to increased production and price declines for NAND Flash memory.

Tablets primarily incorporate NAND Flash on the motherboard or eMMC for storage device. Some Intel-based tablets incorporate mSATA SSDs. However as

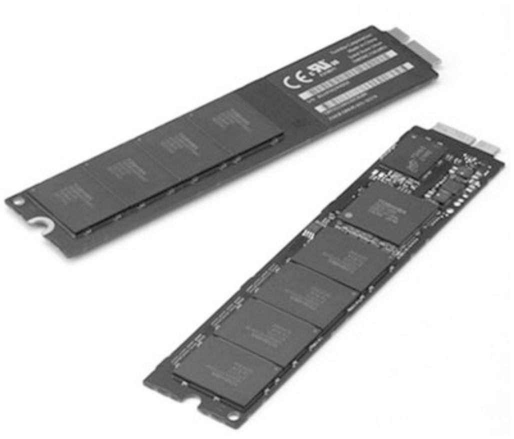


Fig. 1.6 Toshiba Blade X-Gale SSD

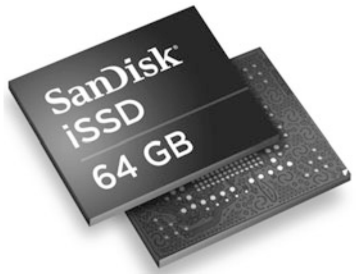


Fig. 1.7 SanDisk iSSD

tablet performance requirements increase and devices become sleeker, μSSD or SSDs in a BGA package such as SanDisk's iSSD (Fig. 1.7) are expected to become more widely adopted.

1.5 Desktop PCs

The benefits of SSDs in desktop systems are less apparent than for notebook PCs. The small form factor, lightweight and low power advantages of SSDs compared to HDDs are of less value here. The primary selling point is the faster response times and better performance and reliability compared to HDD-based systems.

In the corporate environment, SSDs may be incorporated in thin clients. Since thin clients are used to access information and data through corporate networks, minimal local storage is sufficient.

However for the consumer segment, users are used to large storage capacities. A desktop PC offers extra drive bays to add storage. By adding a small capacity SSD to store the operating system and applications and HDD for data, a mix of high capacity storage and high performance can be attained. Such hybrid dual drive systems will appeal to gamers who value performance and are willing to pay a premium for that performance.

1.6 Client PC and Tablet SSD Forecast

SSDs in client PCs consist of product sold directly to OEMs, i.e. already installed in systems sold to the end user or sold to the channel. The channel includes retail, e-tail and value-added resellers. In the nascent stages of the SSD market, the channel comprised the majority of shipments, however, as SSDs become more prevalent in PCs, the channel portion is expected to shrink.

Tablets and PC Flash cache/hybrid dual drive SSDs are expected to be future growth products. The market for PC and tablet SSDs is forecast to grow from $600 million in 2009 to almost $7 billion in 2015 (Fig. 1.8).

1.7 Enterprise Computing Applications

In the enterprise space, high throughput and low access latencies are key. Enterprise HDDs employ several techniques to improve system throughput and decrease the access latency. High performance 15 k rpm HDDs may be organized in a RAID

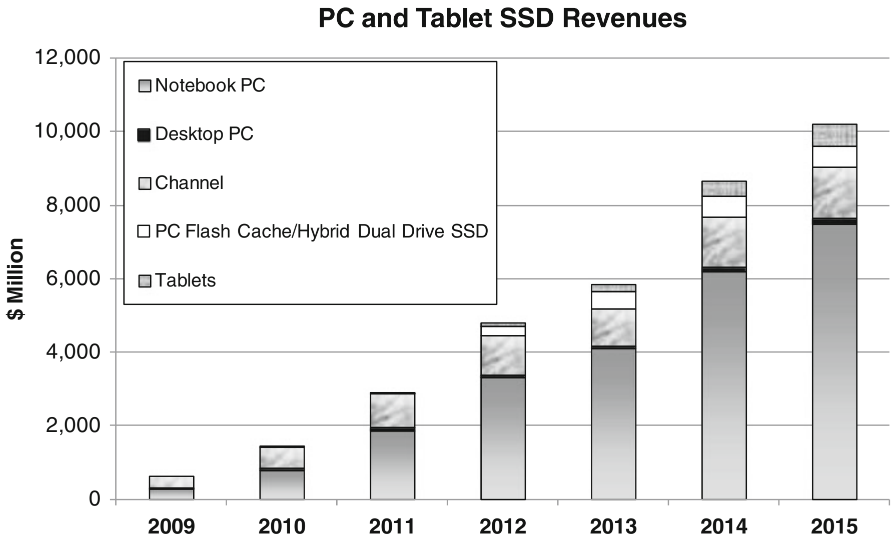


Fig. 1.8 PC and tablet SSD revenue forecast [1]

Table 1.2 Enterprise SSD vs. HDD key metrics [2, 3]

SPECIFICATION	Enterprise Class SSD[1]	Enterprise Class HDD[2]
Form Factor	2.5"	2.5"
Capacity	100/200/400GB	147/300GB
Media	SLC NAND	1-2 disks/2-4 heads
Interface	SAS 6Gbps	SAS 6Gbps
Spindle Speed	-	15k rpm
Performance		
Average Access/Seek Time	0.1ms	3.0ms (read)/2.7ms (write)
Random		
Read	90,000 IOPS	385 IOPS (est)
Write	16,000 IOPS	325 IOPS (est)
Sequential		
Read	500MB/s (64k)	200MB/s
Write	250MB/s (64k)	200MB/s
Reliability		
MTBF	2.0 m hrs	1.6 m hrs
Non-recoverable read errors	1 per 10^{17}	1 per 10^{16}
Power		
Voltage	5/12V	5/12V
Sleep/Idle mode	<1W	4.5W
Operating mode	6.5W	8.7W
Environmental		
Operating shock	1,000G (0.5ms)	100G (1ms)
Non-operating shock	1,000G (0.5ms)	400G (1ms)
Operation temperature	0-55°C	5-55°C
Noise	0dB	3.3 bels
Dimensions (mm)	69.85(W) x 100(D) x 15.4(H)	69.85(W) x 100.45(D) x 15.4(H)
Weight	152g	220g
Warranty	5 years	5 years

[1]Toshiba MKxxx1GRZB
[2]Toshiba MKxxx1GRRB-R

configuration with the data being striped across multiple disks, thereby increasing the bandwidth of the total system. Adding HDDs increases system bandwidth but results in increased space, power and cooling requirements.

"Short-stroking" the HDDs whereby data is placed on the outer tracks of the disk to reduce the seek time of the mechanical heads reduces system access latencies. However, this entails that only part of the HDD is being used to achieve the required performance resulting in excess storage capacity than would otherwise be necessary. Since performance in a SSD is relatively consistent regardless of where the data sits in the drive, a SSD can replace several striped, short-stroked HDDs.

Table 1.2 compares an enterprise class SSD with an enterprise class HDD. As can be seen, access times for a SSD are in the microseconds versus milliseconds for a HDD and the random IOPS performance is two times greater. Moreover, on all other counts, including sequential performance, power consumption, shock resistance and weight, the SSD outclasses the HDD.

The only area where the SSD lags is in price per gigabyte where a SLC SAS SSD is currently about ten times more expensive than a SAS HDD. The price differential

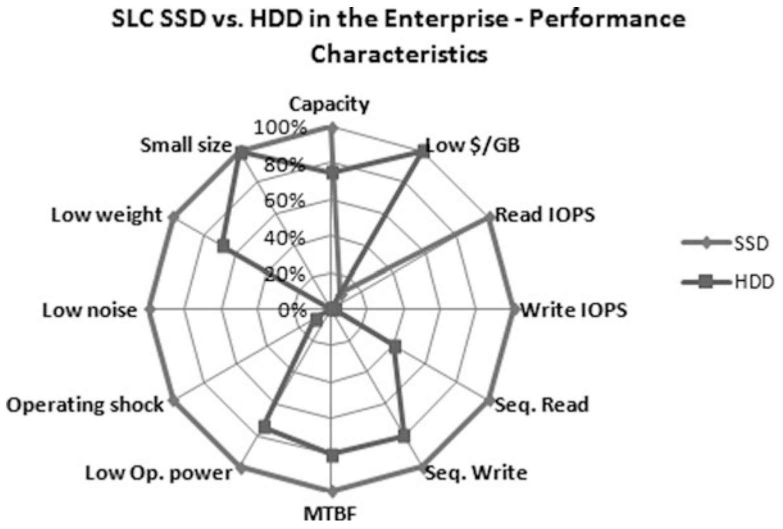


Fig. 1.9 SSD vs. HDD in the enterprise – performance characteristics (Created by Forward Insights)

can be reduced to about five times by employing MLC NAND Flash memory instead of SLC NAND. This comes at the expense of slightly lower performance for a MLC SSD. Nevertheless, the higher cost versus a HDD is more than made up by the much better performance and power characteristics. SSDs are far more superior if metrics in terms of $/IOPS, IOPS/GB or $/W are considered.

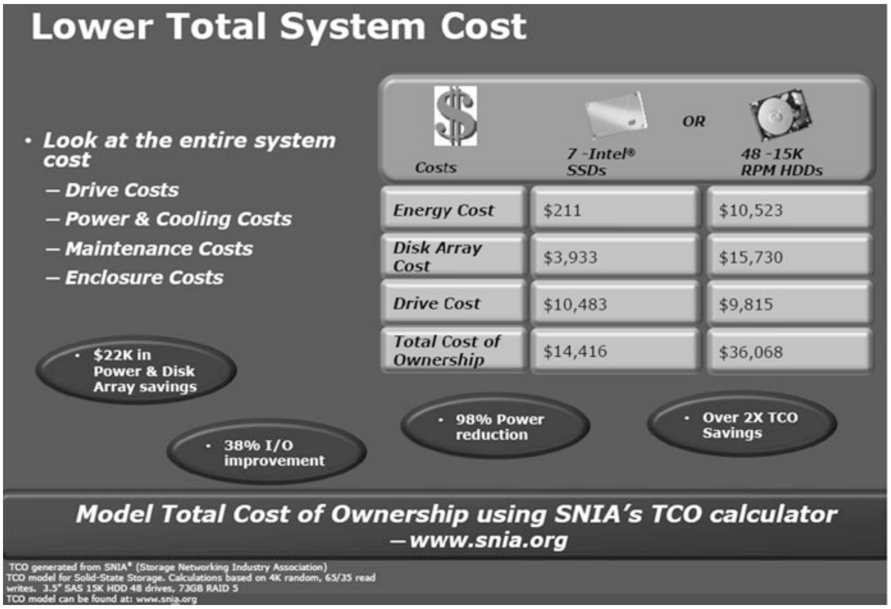
A graphical comparison of the performance characteristics of an enterprise class SSD with an enterprise class HDD is shown in Fig. 1.9. This is just the storage comparison. The superior performance of the SSD translates into significant savings in the total storage system.

Figure 1.10 shows a total cost of ownership calculation for an array employing HDDs versus SSDs. Due to the much higher performance of a SSD, an array of forty-eight high performance 15 k rpm HDDs can be replaced by seven SSDs. The lower number of SSDs drives savings in other areas as well, such as lower energy costs, smaller footprint and enclosure costs, lower power and cooling costs and reduced maintenance costs.

When the price of the SSDs and these ancillary costs are considered, the total cost of ownership is 60% less than the array of forty-eight 15 k rpm HDDs.

In summary, SSDs offer a clear benefit under the following conditions:

- System performance is I/O limited not CPU limited. If the system is CPU limited, improving the I/O performance will not help;
- I/O profile is skewed towards small random files;
- low power consumption is required.



Lower Total System Cost

- *Look at the entire system cost*
  - *Drive Costs*
  - *Power & Cooling Costs*
  - *Maintenance Costs*
  - *Enclosure Costs*

Costs	7 -Intel® SSDs	OR 48 -15K RPM HDDs
Energy Cost	$211	$10,523
Disk Array Cost	$3,933	$15,730
Drive Cost	$10,483	$9,815
Total Cost of Ownership	$14,416	$36,068

- $22K in Power & Disk Array savings
- 38% I/O improvement
- 98% Power reduction
- Over 2X TCO Savings

Model Total Cost of Ownership using SNIA's TCO calculator —www.snia.org

TCO generated from SNIA* (Storage Networking Industry Association)
TCO model for Solid-State Storage. Calculations based on 4K random, 65/35 read writes. 3.5" SAS 15K HDD 48 drives, 73GB RAID 5
TCO model can be found at: www.snia.org

Fig. 1.10 Total system cost (Intel Corp.)

The enterprise market can be segmented into the following main categories: transaction processing, video server, high performance computing and internet server.

1.7.1 Transaction Processing

Fast, real-time processing and accessing of data is a characteristic of transaction processing. Relatively small data files of 2 kB or less are frequently written and retrieved and, due to the real-time nature, system reliability and high IOPS are a prerequisite. The databases are typically large and must be scalable. Transaction processing systems are typically running 24 h a day, 7 days a week.

Applications include stock trading, electronic banking, financial currency exchange, credit card processing, reservation systems and e-commerce.

Although Flash-based SSDs are limited by write endurance of 100 k cycles per block, the write endurance can be extended if the specifications for retention are made less stringent. In transaction-oriented applications, the SSD is acting as temporary store for data, meaning that data retention of weeks could be acceptable. In such a case, the block write endurance can be extended to one million cycles per block.

1.7.2 Video Server

Video-on-demand (VOD) is becoming more prevalent as the bandwidth of home broadband connections improves. Ever larger storage capacities will be required as video transitions from standard definition to high definition. Time-shifted playback of video and network digital video recording will further increase in storage capacity. In addition to storage capacity, the VOD system must be able to rapidly upload large amounts of content quickly.

Fast and reliable play-out of video streams or broadcasts is necessary to provide a satisfactory customer experience. HDDs are not capable of switching in real-time between video selections on the same storage device. In contrast, SSDs have performance levels that allow real-time low-latency switching between random video streams stored in the same storage device necessary for satisfying the demands of multiple users. SSDs also provide benefits in terms of low power consumption, small footprint and system robustness.

VOD systems employ tiered storage systems where high demand titles can be accessed via RAM; medium demand content accessed via Flash and HDD used as a library for content.

1.7.3 High Performance Computing

High Performance Computing (HPC) includes scientific and engineering modelling, simulation and problem solving and includes heavy computational fluid dynamics, seismic tomography, aerospace design and aerodynamic modelling, climate and biological modelling and simulation. This heavy computational number crunching is achieved through clusters of high performance computers. In some cases, the footprint of the cooling systems is as large as the space occupied by the computers: therefore, power savings is as much a consideration as performance.

SSDs augment HDDs in network storage as the Tier 0 layer. SSDs can also serve as boot drives for metadata and paging.

1.7.4 Internet/Network Server

Internet/network servers service many users. As a consequence, response times increase exponentially as the number of operations per second or number of user requests increases. In addition, these user requests are not usually constant or predictable and the servers have to be able to manage spikes in demand. The low latency and high IOPS characteristics of SSDs is especially important for fulfilling these requirements.

Complex server systems also require more time to power up hardware and load the Operating System. Low capacity SSDs may be used in general purpose servers or blade servers as a boot-Flash.

1.7.5 Server-Attached vs. Storage-Attached SSDs

SSDs can either be attached directly to the host server or externally in the storage arrays in NAS/SAN configurations. Within the network storage, there also resides a storage server whereby a SSD may be attached directly to the storage server. Host- or server-attached SSDs are primarily used as a cache to improve system throughput. In the storage-attached configuration, a SSD resides between each storage cluster and the host and acts to improve retrieval time of data stored in the storage array. Server-attached SSDs are dedicated to the server whereas SSDs are shared in the network-attached configuration.

The SSD in the server-attached configuration generally stores a smaller amount of index files than in the storage-attached configuration and is less expensive to implement.

Currently, there is a debate between server vendors and storage vendors as to the merits of attaching the SSD on the server side or storage side. The configuration will largely depend on the system architecture, the application and application workload.

General purpose servers including blade servers with Itanium or x86 CPU cores contain a capacity-optimized 5.4 k rpm HDD for booting. These HDDs are low capacity drives in the 80 GB range directly attached to the server. A low capacity 16 GB or 32 GB SSD can replace these drives directly.

High end servers use network storage, so the SSD is likely to be storage-attached. Mid-range servers may use a combination of network storage plus a server-attached boot SSD.

Although heavy computational computing performed by high end servers have storage-attached architectures, SSDs could be attached close to the CPU via the PCIe slot to speed up processing. Applications which require intense searching or mining of databases on indexes such as internet/networking and video applications may benefit from a storage-attached configuration. Depending on the application, either server- or storage- attached may be suitable for transactional operations, however, to date all are storage-attached.

1.8 Enterprise SSD Forecast

The total enterprise market for SSDs consists of SSDs either attached to the host server or residing in the storage array. In the storage array, SSDs are replacing HDDs with standard interfaces such as SAS and Fibre Channel. The majority of the SSDs

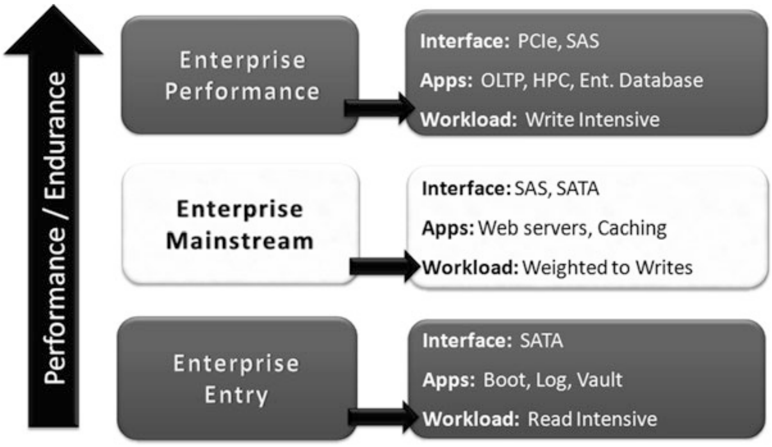


Fig. 1.11 Enterprise SSD segmentation [4]

deployed in servers use a SATA interface. These are the lowest cost SSDs available and may be used either for booting or to improve performance. SSDs employing a PCIe interface are used as a cache or accelerator.

Figure 1.11 provides an applications segmentation of the enterprise SSD market into entry level, mainstream and performance categories. High speed interfaces such as PCIe and SAS will be required to satisfy the fast access and high IOPS characteristics of online transaction process, high performance computing and database applications.

At the other end of the spectrum, the entry level will be serviced by low cost SATA SSDs whereas webserver applications which require a combination of performance and low cost will be supported by SATA and SAS SSDs.

The following summarizes some of the attributes of the different SSD interfaces.

1.8.1 PCIe SSD

PCIe SSDs are slotted near the CPU in servers or in the storage (SAN/NAS) controller head. Since there are only a few PCIe slots available in a server, the PCIe SSD tends to be higher capacity than, for example, a SATA SSD. Lower capacity PCIe SSDs, i.e. less than 1 TB, is typically used for caching or acceleration whereas higher capacity SSDs, greater than 1 TB, can be used for storage.

The disadvantage of a PCIe SSD is that the device is slotted into a server which makes it difficult for the information stored to be shared amongst several servers. In addition, since PCIe SSDs are relatively new to the memory and storage hierarchy, vendors need to provide custom drivers to enable PCIe-based solutions; however, this will change with the implementation of the NVM Express standard (http://www.nvmexpress.org) which defines an optimized register interface, command set and feature set for PCIe SSDs. Also, new PCIe SSDs are already in the market that support standard 2.5″ from factor. These new SSDs are hot-pluggable, front loadable and are designed for mainstream applications.

In many use cases, the PCIe SSD acts as a read cache. Since retention requirements are less stringent, MLC technology is sufficient for this type of usage model.

1.8.2 SATA Boot Drives

Boot drives are low capacity SSDs residing in servers. Traditionally, low capacity 5.4 k rpm HDDs have been used for booting the server after it is down due to, for example, a power outage. The booting function is primarily a sequential reading operation.

SSDs are well suited to booting because of the fast read access times and low capacities required – in the 40 GB or so range. Boot drive SSDs have less demanding performance requirements and require no overprovisioning due to the sequential read nature of the application.

1.8.3 SATA Drives

SATA drives are slotted into servers and also in disk arrays for accelerating performance. They are typically greater than 100 GB in capacity and in some cases, multiple SATA drives are RAIDed and connected through an HBA and slotted into a PCIe slot. This type of implementation has the disadvantage of longer latency due to the additional HBA connection.

1.8.4 Fiber Channel/SAS

Fiber channel (FC) drives are used in SANs. SAS drives reside in servers and SANs/NAS. In a SAN/NAS, multiple drives are connected in a disk array with the SSD acting as a Tier 0 storage layer, storing data which needs to be accessed quickly. The performance benefits are so compelling compared to a HDD that multiple high performance 15 k rpm HDDs can be replaced by one SSD. In a SAN/NAS configuration, the data that resides in the SSDs can be accessed and shared amongst multiple servers. Due to ability to implement multiple drives in a network of tiered storage layers, capacities for FC/SAS SSDs will on average be smaller than in server-attached PCIe implementations. The trend is for SAS to replace FC in NAS implementations.

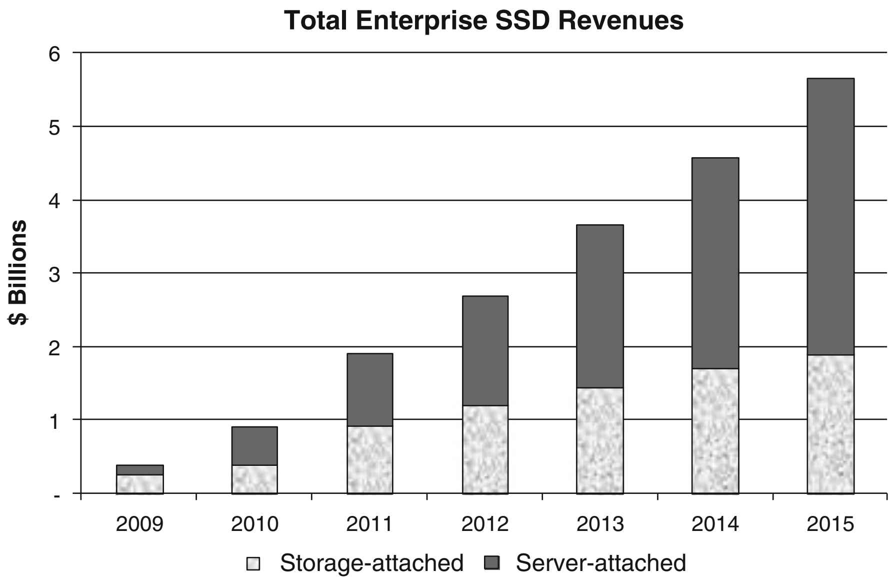


Fig. 1.12 Enterprise SSD forecast [1]

SATA SSDs currently dominate shipments in servers, however as the price gap between SAS and SATA SSDs close, the dual port performance benefits of SAS is likely to expand its presence in servers.

The initial enterprise class SSDs were implemented in SANs/NAS, however the potential market on the server-side is actually much larger than the storage-side. The total enterprise SSD market is forecast to increase from $390 million in 2009 to $5.7 billion in 2015 with server-attached SSDs accounting for two-thirds of the market in 2015 (Fig. 1.12).

References

1. G. Wong, Forward Insights, SSD Insights Q4/11: Opportunity knocks. http://www.forward-insights.com/report36.html. Nov 2011
2. http://storage.toshiba.com/docs/product-datasheets/mkxxx1grzb.pdf?sfvrsn=0
3. http://storage.toshiba.com/techdocs/MK01GRRB-R.pdf
4. K. Dibelius, J. Ellefson, *Micron Technology; Micron's Growing Enterprise Portfolio—A Speedy PCIe System*, May 2011

Chapter 2
SSD Architecture and PCI Express Interface

K. Eshghi and R. Micheloni

Abstract Flash-memory-based solid-state disks (SSDs) provide faster random access and data transfer rates than electromechanical drives and today can often serve as rotating-disk replacements, but the host interface to SSDs remains a performance bottleneck. PCI Express (PCIe)-based SSDs together with an emerging standard called NVMe (Non-Volatile Memory express) promises to solve the interface bottleneck.

This chapter walks the reader through the SSD block diagram, from the NAND memory to the Flash controller (including wear leveling, bad block management, and garbage collection). PCIe basics and different PCIe SSD architectures are reviewed. Finally, an overview on the standardization effort around PCI Express is presented.

2.1 Introduction

Creativity is just connecting things. When you ask creative people how they did something, they feel a little guilty because they didn't really do it, they just saw something. It seemed obvious to them after a while.

– Steve Jobs

Solid-state drives promise to greatly enhance enterprise storage performance. While electromechanical disk drives have continuously ramped in capacity, the rotating-storage technology doesn't provide the access-time or transfer-rate performance required in demanding enterprise applications, including on-line transaction processing, data mining, and cloud computing. Client applications are also in need of

K. Eshghi (✉) • R. Micheloni
Enterprise Computing Division, Integrated Device Technology, Inc.,
San Jose, CA, USA
e-mail: kamyar.eshghi@alum.mit.edu; rino.micheloni@ieee.org

an alternative to electromechanical disk drives that can deliver faster response times, use less power, and fit in smaller mobile form factors.

Flash-memory-based *Solid-State Disks* (SSDs) can offer much faster random access to data and faster transfer rates. Moreover, SSD capacity is now at the point that the drives can serve as rotating-disk replacements. But for many applications the host interface to SSDs remains a bottleneck to performance. PCI Express (PCIe)-based SSDs together with emerging host control interface standards address this interface bottleneck. SSDs with legacy storage interfaces are proving useful, and PCIe SSDs will further increase performance and improve responsiveness by connecting directly to the host processor.

2.2 SSD Architecture

Flash cards, USB keys and Solid State Disks are definitely the most known examples of electronic systems based on non-volatile memories, especially of NAND type (Sect. 2.4).

Several types of memory cards (CF, SD, MMC, …) are available in the market [1–3], with different user interfaces and form factors, depending on the needs of the target application: e.g. mobile phones need very small-sized removable media like μSD.

SSDs are the emerging application for NAND. A SSD is a complete, small system where every component is soldered on a PCB and is independently packaged: NANDs are usually available in TSOP packages.

A basic block diagram of solid state disk is shown in Fig. 2.1. In addition to memories and a controller, there are usually other components. For instance, an

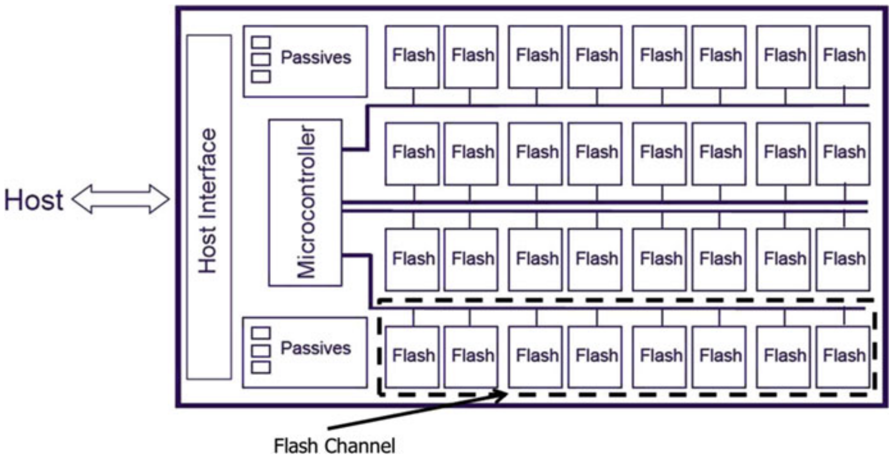


Fig. 2.1 Block diagram of a SSD

external DC-DC converter can be added in order to derive the internal power supply, or a quartz can be used for a better clock precision. Of course, reasonable filter capacitors are inserted for stabilizing the power supply. It is also very common to have a temp sensor for power management reasons. For data caching, a fast DDR memory is frequently used: during a write access, the cache is used for storing data before transfer to the Flash. The benefit is that data updating, e.g. in routing tables, is faster and does not wear out the Flash.

In order to improve performances, NANDs are organized in different Flash channels, as shown in Fig. 2.1.

2.3 Non-volatile Memories

Semiconductor memories can be divided into two major categories: RAM (*Random Access Memories*) and ROM (*Read Only Memories*): RAMs lose their content when power supply is switched off, while ROMs virtually hold it forever. A third category lies in between, i.e. NVM (*Non-Volatile Memories*), whose content can be electrically altered but it is also preserved when the power supply is switched off. NVMs are more flexible than the original ROM, whose content is defined during manufacturing and cannot be changed by the user anymore.

NVM's history began in the 1970s, with the introduction of the first EPROM memory (*Erasable Programmable Read Only Memory*). In the early 1990s, Flash memories came into the game and they started being used in portable products, like mobile phones, USB keys, camcorders, and digital cameras. Solid State Disk (SSD) is the latest killer application for Flash memories. It is worth mentioning that, depending on how the memory cells are organized in the memory array, it is possible to distinguish between NAND and NOR Flash memories. In this book we focus on NAND memories as they are one of the basic elements of SSDs. NOR architecture is described in great details in [4].

NAND Flash cell is based on the Floating Gate (FG) technology, whose cross section is shown in Fig. 2.2. A MOS transistor is built with two overlapping gates

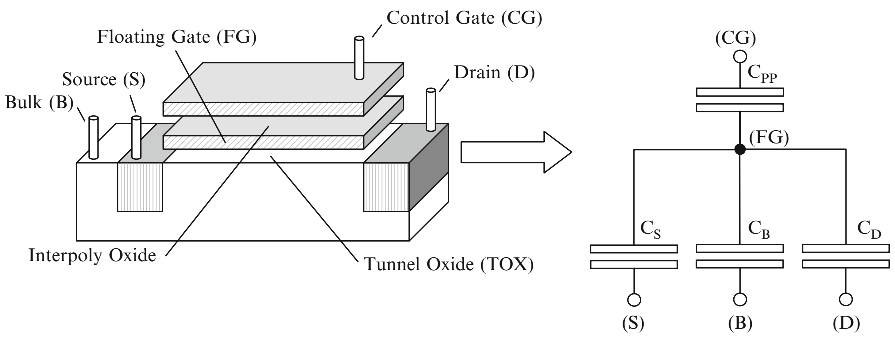


Fig. 2.2 Schematic representation of a floating gate memory cell (*left*) and the corresponding capacitive model (*right*)

rather than a single one: the first one is completely surrounded by oxide, while the second one is contacted to form the gate terminal. The isolated gate constitutes an excellent "trap" for electrons, which guarantees charge retention for years. The operations performed to inject and remove electrons from the isolated gate are called program and erase, respectively. These operations modify the threshold voltage V_{TH} of the memory cell, which is a special type of MOS transistor. Applying a fixed voltage to cell's terminals, it is then possible to discriminate two storage levels: when the gate voltage is higher than the cell's V_{TH}, the cell is on ("1"), otherwise it is off ("0").

It is worth mentioning that, due to floating gate scalability reasons, charge trap memories are gaining more and more attention and they are described in Chap. 5, together with their 3D evolution.

2.4 NAND Flash

2.4.1 NAND Array

A Flash device contains an array of floating-gate transistors: each of them acts as memory cell. In Single Level Cell (SLC) devices, each memory cell stores one bit of information; Multi-Level Cell (MLC) devices store 2 bits per cell.

The basic element of a NAND Flash memory is the NAND string, as shown in Fig. 2.3a. Usually, a string is made up by 32 (M_{C0} to M_{C31}), 64 or 128 cells connected in series. Two selection transistors are placed at the edges of the string: M_{SSL} ensures the connection to the source line. M_{DSL} connects the string to the bitline BL. The cell's control gates are connected through the wordlines (WLs). Figure 2.3b shows how the matrix array is built starting from the basic string. In the WL direction, adjacent NAND strings share the same WL, DSL, BSL and SL. In the BL direction, two consecutive strings share the bitline contact. Figure 2.4 shows a section of the NAND array along the bitline direction.

All the NAND strings sharing the same group of WL's form a Block. In Fig. 2.3b there are three blocks:

– BLOCK0 is made up by $WL_0 < 31{:}0>$;
– BLOCK1 is made up by $WL_1 < 31{:}0>$;
– BLOCK2 is made up by $WL_2 < 31{:}0 >$.

Logical pages are made up of cells belonging to the same WL. The number of pages per WL is related to the storage capabilities of the memory cell. Depending on the number of storage levels, Flash memories are referred to in different ways:

- SLC memories stores 1 bit per cell;
- MLC memories stores 2 bits per cell;
- 8LC memories stores 3 bits per cell;
- 16LC memories stores 4 bits per cell.

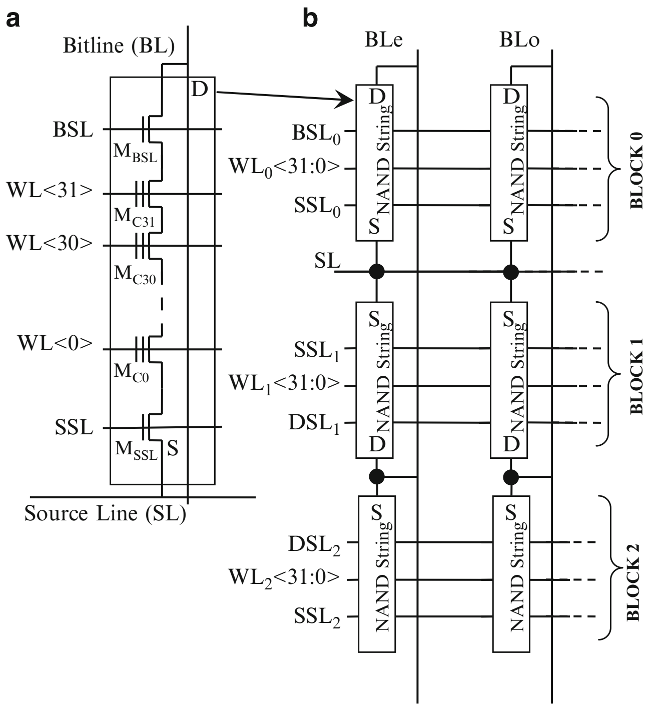


Fig. 2.3 NAND String (**a**) and NAND array (**b**)

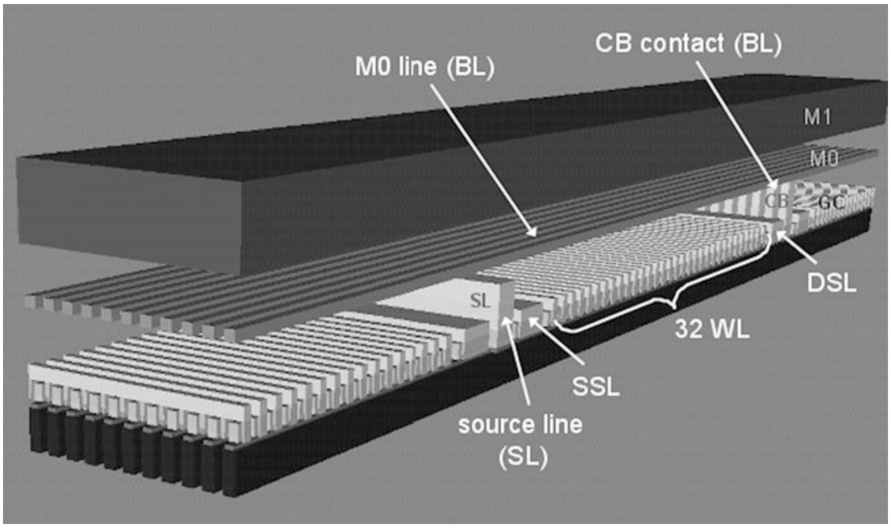


Fig. 2.4 NAND array section along the bitline direction

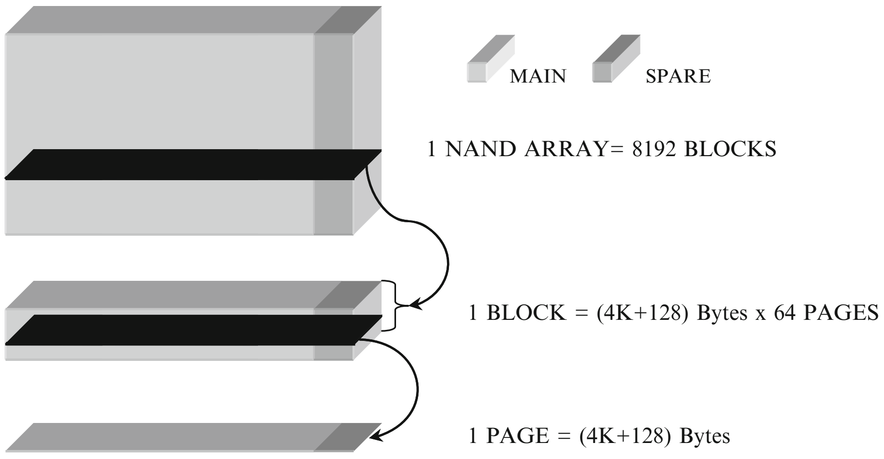

MAIN
SPARE
1 NAND ARRAY= 8192 BLOCKS
1 BLOCK = (4K+128) Bytes x 64 PAGES
1 PAGE = (4K+128) Bytes

Fig. 2.5 32 Gbit memory logic organization

If we consider the SLC case with interleaved architecture (Chap. 6), even cells belong to the "even" page (BLe), while odd pages belong to the "odd" page (BL0). For example, a SLC device with 4 kB page has a WL of $32{,}768 + 32{,}768 = 65{,}536$ cells. Of course, in the MLC case there are four pages as each cell stores one Least Significant Bit (LSB) and one Most Significant Bit (MSB). Therefore, we have MSB and LSB pages on even BL, and MSB and LSB pages on odd BL.

In NAND Flash memories, a logical page is the smallest addressable unit for reading and writing; a logical block is the smallest erasable unit (Fig. 2.5).

Each page is made up by main area (data) and spare area as shown in Fig. 2.5. Main area can be 4 kB, 8 kB or 16 kB. Spare area can be used for ECC and is in the order of hundred of bytes every 4 kB of main area.

Figure 2.5 shows the logic organization of a SLC device with a string of 32 cells, interleaving architecture, 4 kB page, and 128 bytes of spare.

NAND basic operations, i.e. read, program, and erase are described in Chap. 5 and Chap. 6 of this book.

2.4.2 *NAND Interface*

For many years, the asynchronous interface (Fig. 2.6) has been the only available option for NAND devices.

Asynchronous interface is described below.

- CE# : it is the Chip Enable signal. This input signal is "1" when the device is in stand-by mode, otherwise it is always "0".
- R/B# : it is the Ready/Busy signal. This output signal is used to indicate the target status. When low, the target has an operation in progress.

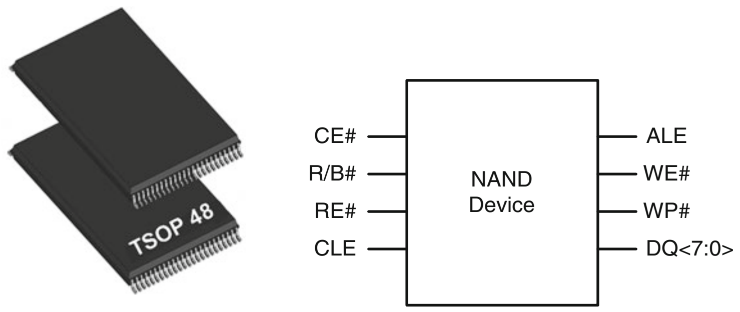


Fig. 2.6 TSOP package (*left*) and related pinout (*right*)

- RE# : it is the Read Enable signal. This input signal is used to enable serial data output.
- CLE : it is the Command Latch Enable. This input is used by the host to indicate that the bus cycle is used to input the command.
- ALE : it is the Address Latch Enable. This input is used by the host to indicate that the bus cycle is used to input the addresses.
- WE# : it is the Write Enable. This input signal controls the latching of input data. Data, command and address are latched on the rising edge of WE#.
- WP# : it is the Write Protect. This input signal is used to disable Flash array program and erase operations.
- DQ < 7:0> : these input/output signals represent the data bus.

As a matter of fact, this interface is a real bottleneck, especially looking at high performance systems like SSDs.

NAND read throughput is determined by array access time and data transfer across the DQ bus. The data transfer is limited to 40 MB/s by the asynchronous interface. As technology shrinks, page size increases and data transfer takes longer; as a consequence, NAND read throughput decreases, totally unbalancing the ratio between array access time and data transfer on the DQ bus. A DDR-like interface (Chap. 6) has been introduced to balance this ratio.

Nowadays two possible solutions are available on the market. ONFI (Open NAND Flash Interface) organization published the first standard at the end of 2006 [5]; other NAND vendors like Toshiba and Samsung use the Toggle-Mode interface. JEDEC [6] is now trying to combine these two approaches together.

Figure 2.7 shows ONFI pinout. Compared to the Asynchronous Interface, there are three main differences:

- RE# becomes W/R# which is the Write/Read direction pin;
- WE# becomes CLK which is the clock signal;
- DQS is an additional pin acting as the data strobe, i.e. it indicates the data valid window.

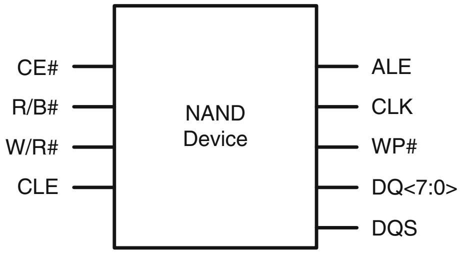


Fig. 2.7 Pinout of a NAND flash supporting ONFI interface

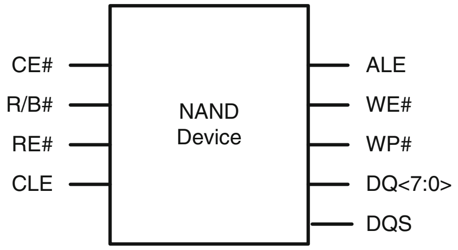


Fig. 2.8 Pinout of a NAND Flash supporting Toggle-Mode Interface

Hence, the clock (CLK) is used to indicate where command and addresses should be latched, while a data strobe signal (DQS) is used to indicate where data should be latched. DQS is a bi-directional bus and is driven with the same frequency as the clock. Toggle-Mode DDR interface uses the pinout shown in Fig. 2.8.

It can be noted that only the DQS pin has been added to the asynchronous interface. In this case, higher speeds are achieved increasing the toggling frequency of RE#.

2.5 Memory Controller

A memory controller has two fundamental tasks:

1. to provide the most suitable interface and protocol towards both the host and the Flash memories;
2. to efficiently handle data, maximizing transfer speed, data integrity and information retention.

In order to carry out such tasks, an application specific device is designed, embedding a standard processor – usually 8–16 bits – together with dedicated hardware to handle timing-critical tasks.

Generally speaking, the memory controller can be divided into four parts, which are implemented either in hardware or in firmware (Fig. 2.9).

Proceeding from the host to the Flash, the first part is the host interface, which implements the required industry-standard protocol (PCIe, SAS, SATA, etc.), thus ensuring both logical and electrical interoperability between SSDs and hosts. This

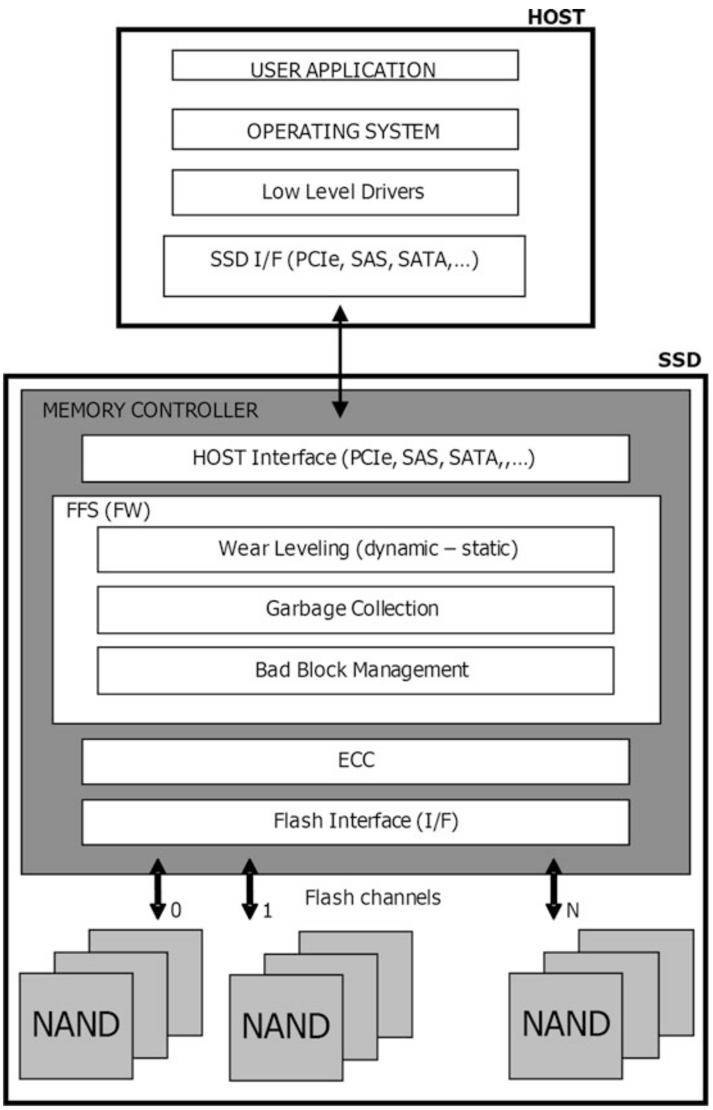


Fig. 2.9 High level view of a Flash controller

block is a mix of hardware – buffers, drivers, etc. – and firmware – command decoding performed by the embedded processor – which decodes the command sequence invoked by the host and handles the data flow to/from the Flash memories.

The second part is the Flash File System (FFS) [7]: that is, the file system which enables the use of SSDs like magnetic disks. For instance, sequential memory access on a multitude of sub-sectors which constitute a file is organized by linked lists (stored on the SSD itself) which are used by the host to build the File Allocation

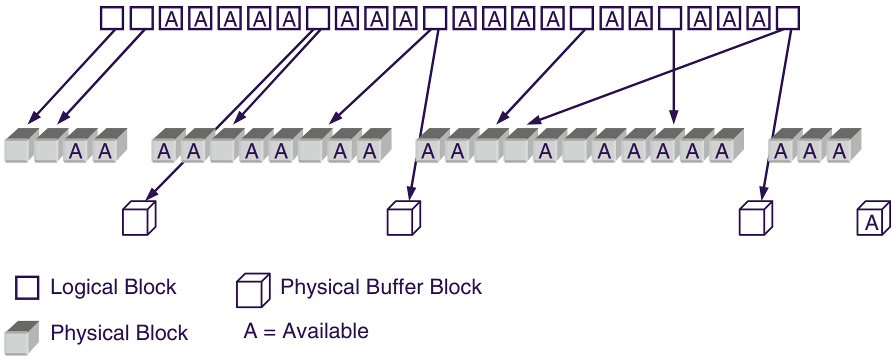


Fig. 2.10 Logical to physical block management

Table (FAT). The FFS is usually implemented in form of firmware inside the controller, each sub-layer performing a specific function. The main functions are: *Wear leveling Management*, *Garbage Collection* and *Bad Block Management*. For all these functions, tables are widely used in order to map sectors and pages from logical to physical (Flash Translation Layer or FTL) [8, 9], as shown in Fig. 2.10. The upper block row is the logical view of the memory, while the lower row is the physical one. From the host perspective, data are transparently written and overwritten inside a given logical sector: due to Flash limitations, overwrite on the same page is not possible, therefore a new page (sector) must be allocated in the physical block and the previous one is marked as invalid. It is clear that, at some point in time, the current physical block becomes full and therefore a second one (Buffer) is assigned to the same logical block.

The required translation tables are always stored on the SSD itself, thus reducing the overall storage capacity.

2.5.1 Wear Leveling

Usually, not all the information stored within the same memory location change with the same frequency: some data are often updated while others remain always the same for a very long time – in the extreme case, for the whole life of the device. It's clear that the blocks containing frequently-updated information are stressed with a large number of write/erase cycles, while the blocks containing information updated very rarely are much less stressed.

In order to mitigate disturbs, it is important to keep the aging of each page/block as minimum and as uniform as possible: that is, the number of both read and program cycles applied to each page must be monitored. Furthermore, the maximum number of allowed program/erase cycles for a block (i.e. its endurance) should be considered: in case SLC NAND memories are used, this number is in the order of 100 k cycles, which is reduced to 10 k when MLC NAND memories are used.

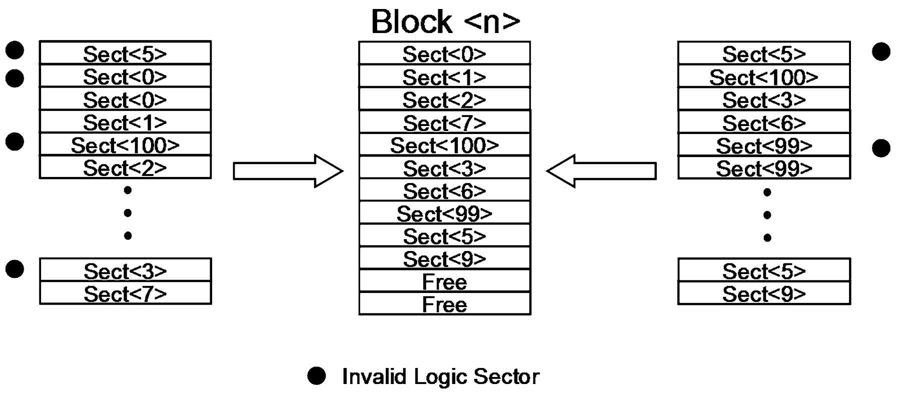


Fig. 2.11 Garbage collection

Wear Leveling techniques rely on the concept of logical to physical translation: that is, each time the host application requires updates to the same (logical) sector, the memory controller dynamically maps the sector onto a different (physical) sector, keeping track of the mapping either in a specific table or with pointers. The out-of-date copy of the sector is tagged as both invalid and eligible for erase. In this way, all the physical blocks are evenly used, thus keeping the aging under a reasonable value.

Two kinds of approaches are possible: Dynamic Wear Leveling is normally used to follow up a user's request of update, writing to the first available erased block with the lowest erase count; Static Wear Leveling can also be implemented, where every block, even the least modified, is eligible for re-mapping as soon as its aging deviates from the average value.

2.5.2 *Garbage Collection*

Both wear leveling techniques rely on the availability of free sectors that can be filled up with the updates: as soon as the number of free sectors falls below a given threshold, sectors are "compacted" and multiple, obsolete copies are deleted. This operation is performed by the Garbage Collection module, which selects the blocks containing the invalid sectors, copies the latest valid copy into free sectors and erases such blocks (Fig. 2.11).

In order to minimize the impact on performance, garbage collection can be performed in background. The equilibrium generated by the wear leveling distributes wear out stress over the array rather than on single hot spots. Hence, the bigger the memory density, the lower the wear out per cell is.

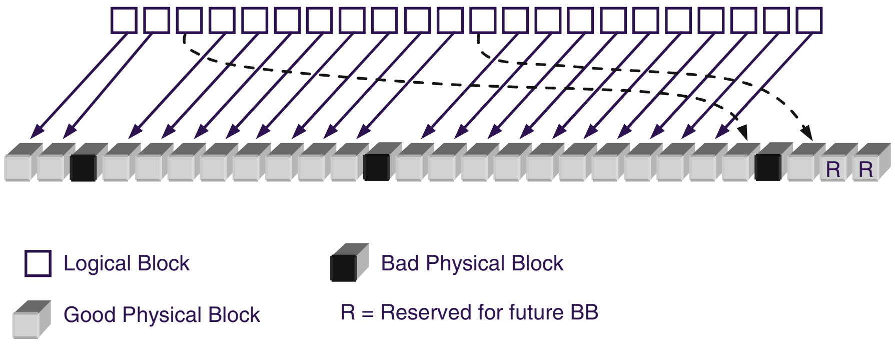


Fig. 2.12 Bad Block Management (BBM)

2.5.3 *Bad Block Management*

No matter how smart the Wear Leveling algorithm is, an intrinsic limitation of NAND Flash memories is represented by the presence of so-called Bad Blocks (BB), i.e. blocks which contain one or more locations whose reliability is not guaranteed.

The Bad Block Management (BBM) module creates and maintains a map of bad blocks, as shown in Fig. 2.12: this map is created during factory initialization of the memory card, thus containing the list of the bad blocks already present during the factory testing of the NAND Flash memory modules. Then it is updated during device lifetime whenever a block becomes bad.

2.5.4 *Error Correction Code (ECC)*

This task is typically executed by a specific hardware inside the memory controller. Examples of memories with embedded ECC are also reported [10–12]. Most popular ECC codes, correcting more than one error, are Reed-Solomon and BCH [13]. Chapter 10 gives an overview of how BCH is used in the NAND world, including an analysis of its detection properties, which are essential for concatenated architectures. The last section of Chap. 10 covers the usage of BCH in high-end SSDs, where the ECC has to be shared among multiple Flash channels.

With the technology shrink, NAND raw BER gets worse, approaching the Shannon limit. As a consequence, correction techniques based on soft information processing are required: LDPC (Low Density Parity Check) codes are an example of this soft information approach and they are analyzed in Chap. 11.

2.6 Multi-channel Architecture

A typical memory system is composed by several NAND memories. Typically, an 8-bit bus, usually called channel, is used to connect different memories to the controller (Fig. 2.1). It is important to underline that multiple Flash memories in a system are both a means for increasing storage density and read/write performance [14].

Operations on a channel can be interleaved, which means that a second chip can be addressed while the first one is still busy. For instance, a sequence of multiple write operations can be directed to a channel, addressing different NANDs, as shown in Fig. 2.13: in this way, the channel utilization is maximized by pipelining the data load phase; in fact, while the program operation takes place within a memory chip, the corresponding Flash channel is free. The total number of Flash channel is a function of the target applications, but tens of channels are becoming quite common. Figure 2.14 shows the impact of interleaving. As the reader can notice, given the same Flash programming time, SSD's throughput greatly improves.

The memory controller is responsible for scheduling the distributed accesses at the memory channels. The controller uses dedicated engines for the low level communication protocol with the Flash.

Moreover, it is clear that the data load phase is not negligible compared to the program operation (the same comment is valid for data output): therefore, increasing I/O interface speed is another smart way to improve performances: DDR-like interfaces are discussed in more details in Chap. 6. Impact of DDR frequency on program throughput is reported in Fig. 2.15. As the speed increases, more NAND can be operated in parallel before saturating the channel. For instance, assuming a target of 30 MB/s, 2 NANDs are needed with a minimum DDR frequency of about 50 MHz. Given a page program time of 200 μs, at 50 MHz four NANDs can operate in interleaved mode, doubling the write throughput. Of course, power consumption has then to be considered.

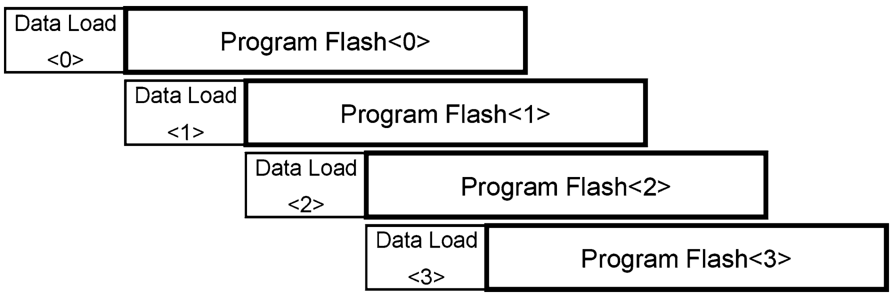


Fig. 2.13 Interleaved operations on one Flash channel

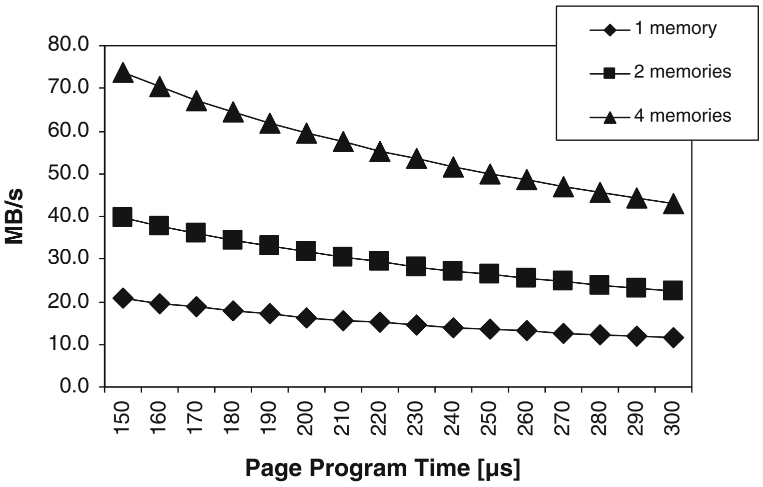


Fig. 2.14 Program throughput with an interleaved architecture as a function of the NAND page program time

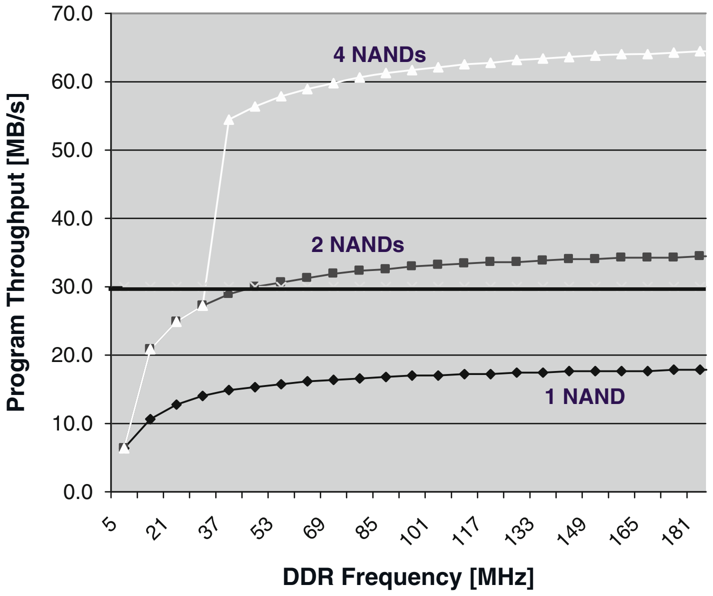


Fig. 2.15 Program throughput with an interleaved architecture as a function of the channel DDR frequency. 4 kB page program time is 200 μs

After this high level overview of the SSD architecture, let's move to the interface towards the host. PCI Express (PCIe) is the emerging interface for high performance SSDs.

2.7 What Is PCIe?

PCIe (Peripheral Component Interconnect Express) is a bus standard that replaced PCI and PCI-X. PCI-SIG (PCI Special Interest Group) creates and maintains the PCIe specification [15].

PCIe is used in all computer applications including enterprise servers, consumer personal computers (PC), communication systems, and industrial applications. Unlike older PCI bus topology, which uses shared parallel bus architecture, PCIe is based on point-to-point topology, with separate serial links connecting every device to the root complex (host). Additionally, a PCIe link supports full-duplex communication between two endpoints. Data can flow upstream (UP) and downstream (DP) simultaneously. Each pair of these dedicated unidirectional serial point-to-point connections is called a *lane*, as depicted in Fig. 2.16. The PCIe standard is constantly under improvement, with PCIe 3.0 being the latest version of the standard (Table 2.1).

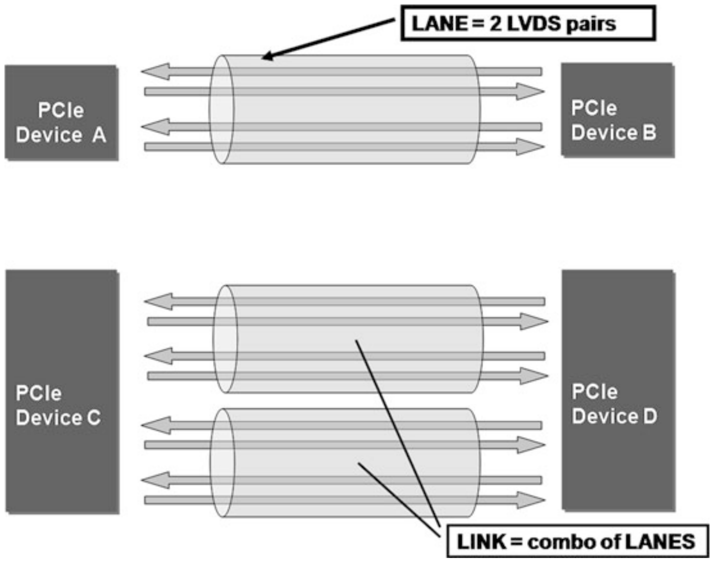


Fig. 2.16 PCI Express lane and link. In Gen2, 1 lane runs at 5 Gbps/direction; a 2-lane link runs at 10 Gbps/direction

Table 2.1 Throughput of different PCIe generations

PCIe version	Year introduced	Throughput per lane
PCIe 1.0 (Gen1)	2003	250 MB/s
PCIe 2.0 (Gen2)	2007	500 MB/s
PCIe 3.0 (Gen3)	2010	1 GB/s

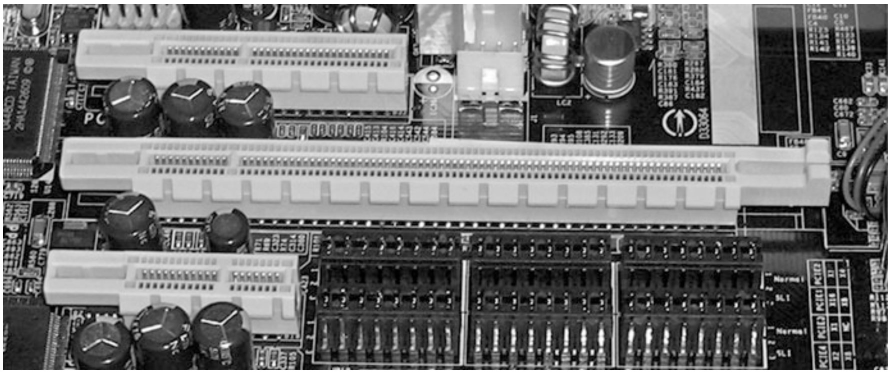


Fig. 2.17 Various PCIe slots. From top to bottom: PCIe × 4, PCIe × 16, PCIe × 1

Other important features of PCIe include power management, hot-swappable devices, and the ability to handle peer-to-peer data transfers (sending data between two end points without routing through the host) [16]. Additionally, PCIe simplifies board design by utilizing serial technology, which eliminates wire count of parallel bus architectures.

The PCIe link between two devices can consist of 1–32 lanes. The packet data is striped across lanes, and the lane count is automatically negotiated during device initialization.

The PCIe standard defines slots and connectors for multiple widths: ×1, ×4, ×8, ×16, ×32 (Fig. 2.17). This allows PCIe to serve lower throughput, cost-sensitive applications as well as performance-critical applications.

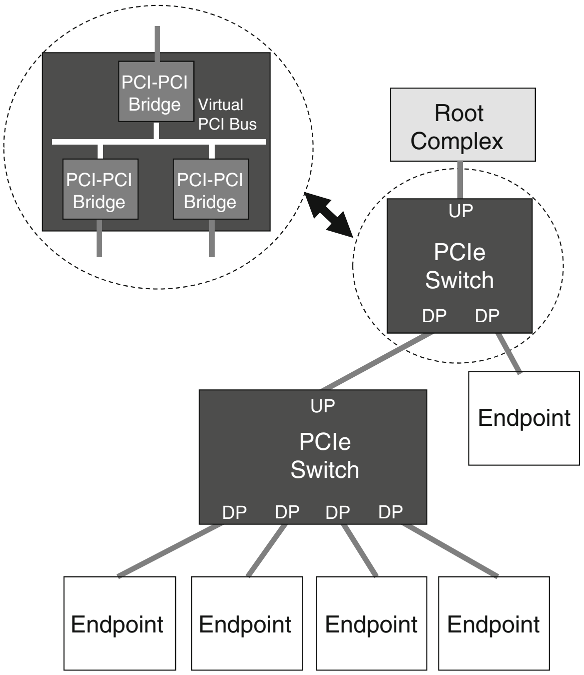
There are basically three different types of devices in a native PCIe system as shown in Fig. 2.18 [17]: *Root Complexes* (RCs), PCIe switches, and *EndPoints* (EPs). A Root Complex should be thought of as a single processor sub-system with a single PCIe port, even though it consists of one or more CPUs, plus their associated RAM and memory controller. PCIe routes data based on memory address or ID, depending on the transaction type. Therefore, every device must be uniquely identified within the PCI Express tree. This requires a process called enumeration. During system initialization, the Root Complex performs the enumeration process to determine the various buses that exist and the devices that reside on each bus, as well as the required address space. The Root Complex allocates bus numbers to all the PCIe buses and configures the bus numbers to be used by the PCIe switches.

A PCIe switch behaves as if it were multiple PCI-PCI Bridges, as shown in the inset of Fig. 2.18. Basically, a switch decouples every UP and DP ports so that each link can work as a point-to-point connection.

Within a PCIe tree, all devices share the same memory space. RC is in charge of setting the Base Address Register (BAR) of each device.

In multi-RC systems, more than one processor sub-system exists within a PCIe tree. For example, a second Root Complex may be added to the system via the DP

Fig. 2.18 PCIe tree topology



of a PCIe switch, possibly to act as a warm stand-by to the primary RC. However, an issue arises when the second RC also attempts the enumeration process: it sends out Configuration Read Messages to discover other PCIe devices on the system. Unfortunately, configuration transactions can only move from UP to DP. A PCIe switch does not forward configuration messages that are received on its DP. Thus, the second RC is isolated from the rest of the PCIe tree and will not detect any PCIe devices in the system. So, simply adding processors to a DP of a PCIe switch will not provide a multi-Root Complex solution.

One method of supporting multiple RCs is to use a *Non-Transparent Bridging* (NTB) function to isolate the address domains of each of the Root Complexes [18]. NTB allows two Root Complexes or PCIe trees to be interconnected with one or more shared address windows between them.

In other words, NTB works like an address translator between two address domains. Of course, multiple NTBs can be used to develop multi-RC applications. An example of PCIe switch with embedded NTB functions is shown in Fig. 2.19: an additional bus, called NT Interconnect, is used for exchanging Transaction Layer Packets among RCs.

PCIe uses a packet-based layered protocol, consisting of a transaction layer, a data link layer, and a physical layer, as shown in Fig. 2.20.

The transaction layer handles packetizing and de-packetizing of data and status-message traffic. The data link layer sequences these *Transaction Layer Packets* (TLPs) and ensures they are reliably delivered between two endpoints (devices A

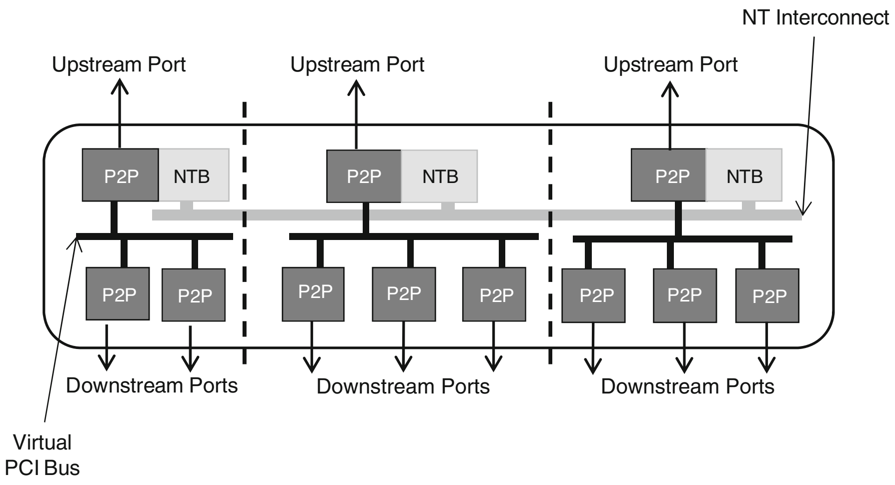


Fig. 2.19 PCIe switch with multiple NTB functions

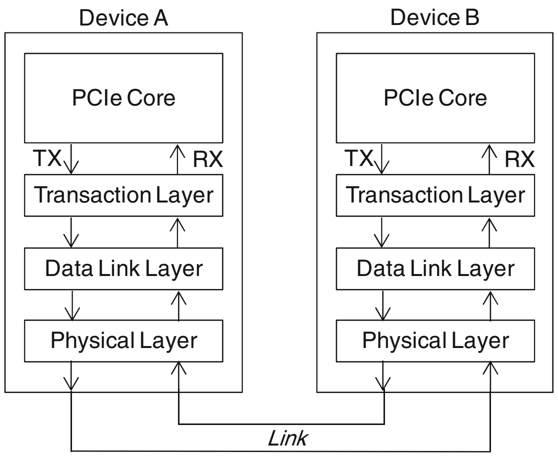


Fig. 2.20 PCIe Layered architecture

and B in Fig. 2.5). If a transmitter device sends a TLP to a remote receiver device and a CRC error is detected, the transmitter device gets a notification back. The transmitter device automatically replays the TLP. With error checking and automatic replay of failed packets, PCIe ensures very low *Bit Error Rate* (BER).

The Physical Layer is split in two parts: the Logical Physical Layer and the Electrical Physical Layer. The Logical Physical Layer contains logic gates for processing packets before transmission on the Link, and processing packets from the Link to the Data Link Layer. The Electrical Physical Layer is the analog interface of the Physical Layer: it consists of differential drivers and receivers for each lane.

TLP assembly is shown in Fig. 2.21. Header and Data Payload are TLP's core information: Transaction Layer assembles this section based on the data received

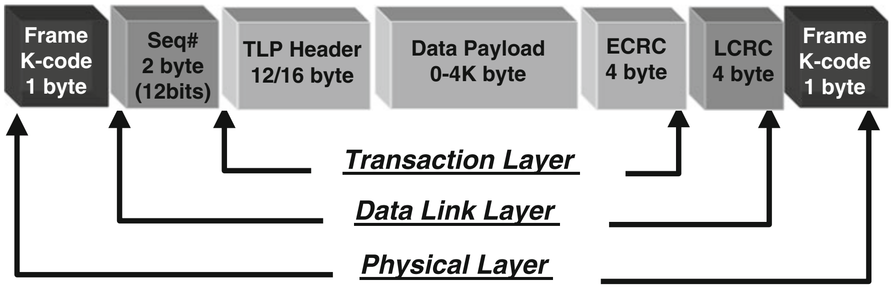


Fig. 2.21 Transaction Layer Packet (TLP) assembly

from the application software layer. An optional End-to-End CRC (ECRC) field is can be appended to the packet. ECRC is used by the ultimate targeted device of this packet to check for CRC errors inside Header and Data Payload. At this point, the Data Link Layer appends a sequence ID and local CRC (LCRC) field in order to protect the ID. The resultant TLP is forwarded to the Physical Layer which concatenates a Start and End framing character of 1 byte each to the packet. Finally, the packet is encoded and differentially transmitted on the Link using the available number of Lanes.

Today, PCIe is a high volume commodity interconnect used in virtually all computers, from consumer laptops to enterprise servers, as the primary motherboard technology that interconnects the host CPU with on-board ICs and add-on peripheral expansion cards.

2.8 The Need for Storage Speed

The real issue at hand is the need for storage technology that can match the exponential ramp in processor performance over the past two decades. Processor vendors have continued to ramp the performance of individual processor cores, to combine multiple cores on one IC, and to develop technologies that can closely-couple multiple ICs in multi-processor systems. Ultimately, all of the cores in such a scenario need access to the same storage subsystem.

Enterprise IT managers are eager to utilize the multiprocessor systems because they have the potential of boosting the number of I/O operations per second (IOPS) that a system can process and also the number of IOPS per watt (IOPS/W) in power consumption. The ramping multi-processing computing capability offers better IOPS relative to cost and power consumption – assuming the processing elements can get access to the data in a timely fashion. Active processors waiting on data waste time and money.

There are of course multiple levels of storage technology in a system that ultimately feeds code and data to each processor core. Generally, each core includes local cache memory that operates at core speed. Multiple cores in a chip share a second-level and sometimes a third-level cache. And DRAM feeds the caches. The DRAM and cache access-time and data-transfer performance has scaled to match the processor performance.

The disconnect has come in the performance gap that exist between DRAM and rotating storage in terms of access time and data rate. Disk-drive vendors have done a great job of designing and manufacturing higher-capacity, lower-cost-per-Gbyte disk drives. But the drives inherently have limitations in terms of how fast they can access data and then how fast they can transfer that data into DRAM.

Access time depends on how fast a hard drive can move the read head over the required data track on a disk, and the rotational latency for the sector where the data is located to move under the head. The maximum transfer rate is dictated by the rotational speed of the disk and the data encoding scheme that together determine the number of bytes per second read from the disk.

Hard drives perform relatively well in reading and transferring sequential data. But random seek operations add latency. And even sequential read operations can’t match the data appetite of the latest processors.

Meanwhile, enterprise systems that perform on-line transaction processing such as financial transactions and that mine data in applications such as customer relationship management require highly random access to data. Cloud computing also has a random element and the random issue in general is escalating with technologies such a virtualization expanding the scope of different applications that a single system has active at any one time. Every microsecond of latency relates directly to money lost and less efficient use of the processors and the power dissipated by the system.

Fortunately Flash memory offers the potential to plug the performance gap between DRAM and rotating storage. Flash is slower than DRAM but offers a lower cost per Gbyte of storage. That cost is more expensive than disk storage, but enterprises will gladly pay the premium because Flash also offers much better throughput in terms of Mbytes/s and faster access to random data, resulting in better cost-per-IOPS compared to rotating storage.

Ramping Flash capacity and reasonable cost has led to a growing trend of SSDs that package Flash in disk-drive-like form factors. Moreover, the SSDs have most often utilized disk-drive interfaces such as SATA (serial ATA) or SAS (serial attached SCSI).

2.9 Why PCIe for SSD Interface?

The disk-drive form factor and interface allows IT vendors to substitute an SSD for a magnetic disk drive seamlessly. There is no change required in system hardware or driver software. An IT manager can simply swap to an SSD and realize significantly better access times and somewhat faster data-transfer rates.

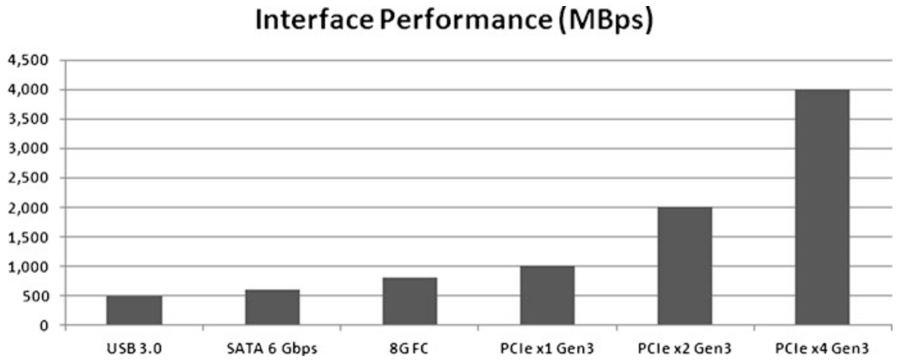


Fig. 2.22 Interface performance. PCIe improves overall system performance by reducing latency and increasing throughput

Neither the legacy disk-drive form factor nor the interface is ideal for Flash-based storage. SSD manufacturers can pack enough Flash devices in a 2.5-in. form factor to easily exceed the power profile developed for disk drives. And Flash can support higher data transfer rates than even the latest generation of disk interfaces.

Let's examine the disk interfaces more closely (Fig. 2.22). Most mainstream systems today are migrating to third-generation SATA and SAS that support 600 Mbytes/s throughput, and drives based on those interfaces have already found usage in enterprise systems. While those data rates support the fastest electromechanical drives, new NAND Flash architectures and multi-die Flash packaging deliver aggregate Flash bandwidth that exceeds the throughput capabilities of SATA and SAS interconnects. In short, the SSD performance bottleneck has shifted from the Flash devices to the host interface. Therefore, many applications need a faster host interconnect to take full advantage of Flash storage.

The PCIe host interface can overcome this storage performance bottleneck and deliver unparalleled performance by attaching the SSD directly to the PCIe host bus. For example, a 4-lane (x4) PCIe Generation 3 (Gen3) link, shipping in volume in 2012, can deliver 4 GByte/s data rates. Simply put, PCIe affords the needed storage bandwidth. Moreover, the direct PCIe connection can reduce system power and slash the latency that's attributable to the legacy storage infrastructure.

Clearly an interface such as PCIe could handle the bandwidth of a multi-channel Flash storage subsystem and can offer additional performance advantages. SSDs that use a disk interface also suffer latency added by a storage-controller IC that handles disk I/O. PCIe devices connect directly to the host bus eliminating the architectural layer associated with the legacy storage infrastructure. The compelling performance of PCIe SSDs has resulted in system manufacturers placing PCIe SSDs in servers as well as in storage arrays to build tiered storage systems (Fig. 2.23) that accelerate applications while improving cost-per-IOPS (*Input/Output Operations per Second*).

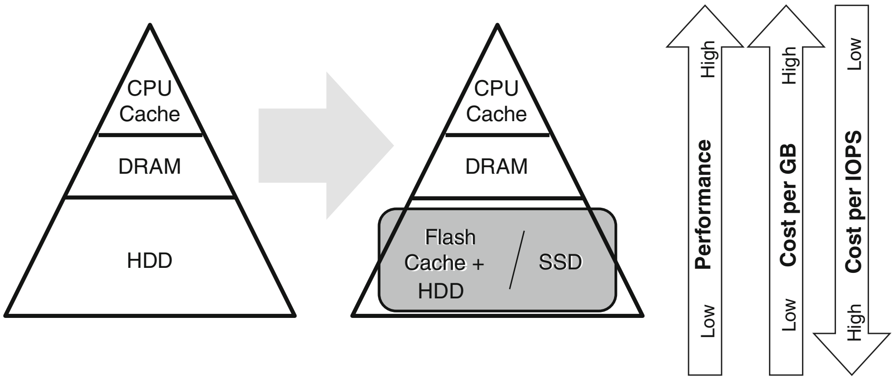


Fig. 2.23 Enterprise memory/storage hierarchy paradigm shift

Moving storage to a PCIe link brings additional challenges to the system designer. As mentioned earlier, the SATA- and SAS-based SSD products have maintained software compatibility and some system designers are reluctant to give up that advantage. Any PCIe storage implementation will create the need for some new driver software.

Despite the software issue, the move to PCIe storage in enterprises is already happening. Performance demands in the enterprise are mandating this transition. There is no other apparent way to deliver improving IOPS, IOPS/W, and IOPS per dollar characteristics that IT managers are demanding.

The benefits of using PCIe as a storage interconnect are clear. You can achieve over 6x the data throughput relative to SATA or SAS. You can eliminate components such as host bus adapters and SERDES ICs on the SATA and SAS interfaces – saving money and power at the system level. And PCIe moves the storage closer to the host CPU reducing latency, as shown in Fig. 2.24.

So the question the industry faces isn't really whether to use PCIe to connect with Flash storage, but how to do so. There are a number of options with some early products already in the market.

Let's now take a deeper look at PCIe-based SSD architectures.

2.10 PCIe SSD Implementations

The simplest PCIe SSD implementations can utilize legacy Flash memory controller ICs that while capable of controlling memory read and write operations, have no support for the notion of system I/O. Such Flash controllers would typically work behind a disk interface IC in existing SATA- or SAS-based SSD products (Fig. 2.25).

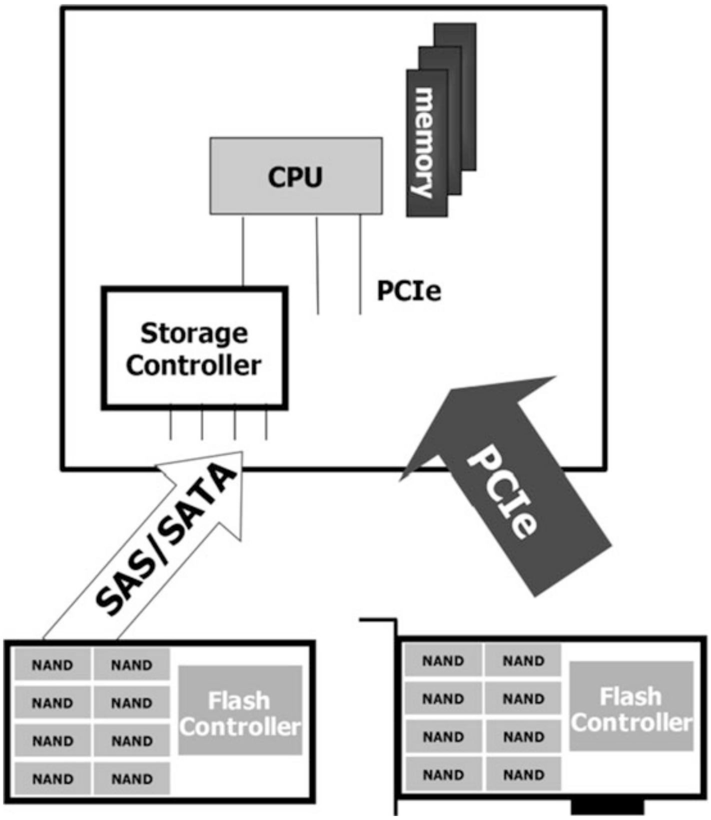


Fig. 2.24 PCIe SSD vs. SAS/SATA SSD

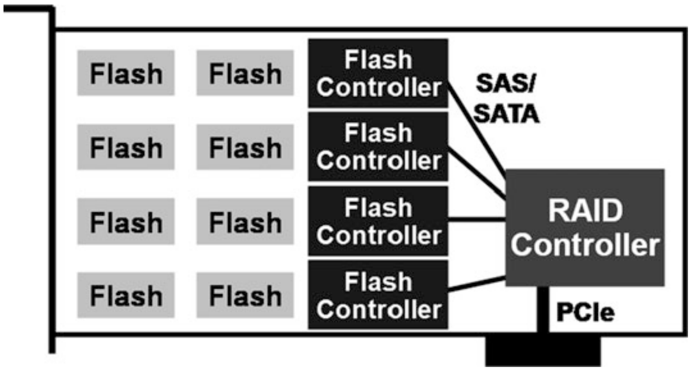


Fig. 2.25 RAID-based PCIe SSDs not optimized for performance/power

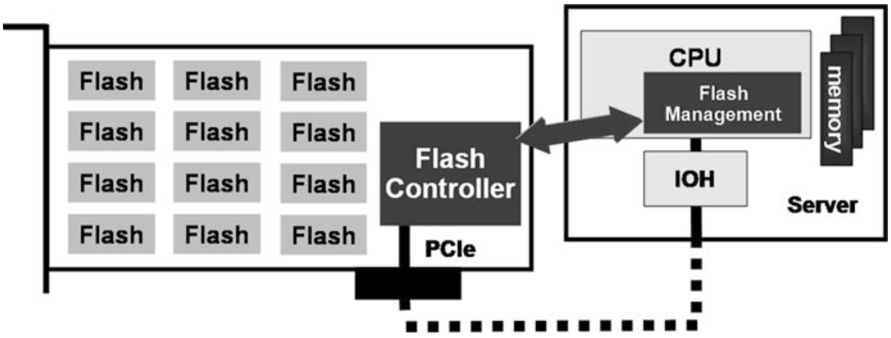


Fig. 2.26 Running flash management algorithms on the host drains the host CPU/RAM resources

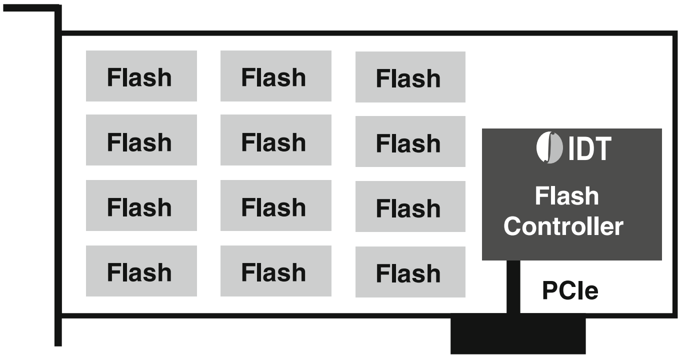


Fig. 2.27 Native PCIe Flash Controller improves performance, while reducing cost & complexity

Alternatively, it is possible to run Flash-management software on the host processor to enable a simple Flash controller to function across a PCIe interconnect (Fig. 2.26).

That approach has several drawbacks. First it consumes host processing and memory resources that ideally would be handling more IOPS. Second it requires proprietary drivers and raises OEM qualification issues. And third it doesn't deliver a bootable drive because the system must be booted for the Flash-management software to execute and enable the storage scheme.

Clearly, these designs have found niche success. These products are used by early adopters as caches for hard disk drives rather than mainstream replacements of high-performance disk drives.

Longer term, more robust and efficient PCIe SSD designs are relying on a complex SoC that natively supports PCIe, integrates Flash controller functionality, and that completely implements the storage-device concept (Fig. 2.27). Such a product offloads the host CPU and memory of handling Flash management, and ultimately enables standard OS drivers that support plug-and-play operations just as we enjoy with SATA and SAS today.

2.11 Standards Driving Broader Adoption of PCIe SSDs

New standards will ultimately deliver plug-and-play functionality for PCIe-connected SSDs. Table 2.2 is a summary of the industry effort in this direction.

The NVM Express (NVMe) 1.0 specification, developed cooperatively by more than 80 companies from across the industry, was released in March, 2011, by the NVMHCI Work Group – now more commonly known as the NVMe Work Group. The specification defines an optimized register interface, command set, and feature set for PCIe SSDs. The goal of the standard is to help enable the broad adoption of PCIe-based SSDs, and to provide a scalable interface that realizes the performance potential of SSD technology now and into the future. By maximizing parallelism and eliminating complexity of legacy storage architectures, NVMe supports future memory developments that will drive latency overhead below one microsecond and SSD IOPS to over one million. The NVMe 1.0 specification may be downloaded from www.nvmexpress.org.

The NVMe specification is specifically optimized for multi-core system designs that run many threads concurrently with each thread capable of instigating I/O operations. Indeed it's optimized for just the scenario that IT managers are hoping to leverage to boost IOPS. NVMe specification can support up to 64 k I/O queues with up to 64 k commands per queue. Each processor core can implement its own queue.

In June, 2011, the NVMe Promoter Group was formed to enable the broad adoption of the NVMe Standard for PCIe SSDs. Seven companies hold permanent seats on the board: Cisco, Dell, EMC, IDT, Intel, NetApp, and Oracle. NVMe supporters include IC manufactures, Flash-memory manufacturers, operating-system vendors, server manufacturers, storage-subsystem manufacturers, and network-equipment manufacturers.

SCSI Express is another industry initiative that is planning to address the host control interface of PCIe SSDs, with support for legacy enterprise storage command set. SCSI Express uses SCSI over PCIe (SOP) and PCIe architecture queuing interface (PQI) model being defined within the T10 Technical Committee.

Table 2.2 Industry standards for PCIe SSDs

Standard	Status	Benefits
NVMe	Spec 1.0 released March 2011 Standard OS driver implementations available (Windows and Linux) 80+ members of NVMe working group	Designed for servers and clients
SCSI over PCIe (SOP)	Under development in T10	Utilizes SCSI software infrastructure
SSD Form Factor Working Group [19] (2.5" with new connector)	Spec 1.0 released December 2011	Serviceability Hot-plugability

The NVMe and SOP standards do not address the subject of form factors for SSDs and that's another issue that has been addressed through another working group.

In enterprise-class storage, devices such as disk drives and SSDs are typically externally accessible and support hot-plug capabilities. In part the hot-plug capability was required due to the fact that disk drives are mechanical in nature and generally fail sooner than ICs. The hot-plug feature allows easy replacement of failed drives.

With SSDs, IT managers and storage vendors will want to stay with an externally-accessible modular approach. Such an approach supports easy addition of storage capacity either by adding SSDs, or replacing existing SSDs with more capacious ones.

Indeed another standards body was formed to address the form factor issue. The SSD Form Factor Working Group is focused on promoting PCIe as an SSD interconnect. The working group is driven by five Promoter Members including Dell, EMC, Fujitsu, IBM, and Intel.

Enterprise SSD Form Factor Version 1.0 specification was released in December 2011, focusing on three areas:

- a connector specification that will support PCIe as well as SAS/SATA;
- a form factor that builds upon the current 2.5-in. standard while supporting the new connector definition and expanding the power envelope in support of higher performance;
- the support for hot plug capability.

The building blocks are all falling into place for broader usage of PCIe-connected SSDs and deliverance of the performance improvements that the technology will bring to enterprise applications. And while our focus has been more on the enterprise, the NVMe standard will also trickle down to client systems, offering a performance boost even in notebook PCs while reducing cost and system power. The standard will drive far more widespread use of PCIe SSD technology as compatible ICs and drivers come to fruition.

References

1. www.mmca.org
2. www.compactflash.org
3. www.sdcard.com
4. G. Campardo, R. Micheloni, D. Novosel, *VLSI-Design of Non-Volatile Memories* (Springer, Berlin, 2005)
5. www.onfi.org
6. www.jedec.org
7. A. Kawaguchi, S. Nishioka, H. Motoda, A flash-memory based file system, in *Proceedings of the USENIX Winter Technical Conference*, 1995, pp. 155–164
8. J. Kim, J.M. Kim, S. Noh, S.L. Min, Y. Cho, A space-efficient flash translation layer for compactflash systems. IEEE Trans. Consum. Electron. **48**(2), 366–375 (May 2002)

9. S.-W. Lee, D.-J. Park, T.-S. Chung, D.-H. Lee, S.-W. Park, H.-J. Songe, FAST: A log-buffer based FTL scheme with fully associative sector translation, in *2005 US-Korea Conference on Science, Technology, & Entrepreneurship*, Seoul, Aug 2005
10. T. Tanzawa, T. Tanaka, K. Takekuchi, R. Shirota, S. Aritome, H. Watanabe, G. Hemink, K. Shimizu, S. Sato, Y. Takekuchi, K. Ohuchi, A compact on-chip ECC for low cost Flash memories. IEEE J. Solid-State Circuits **32**(May), 662–669 (May 1997)
11. G. Campardo, R. Micheloni et al., 40-mm^2 3-V-only 50-MHz 64-Mb 2-b/cell CHE NOR Flash memory. IEEE J Solid-State Circuits. **35**(11), 1655–1667 (Nov 2000)
12. R. Micheloni et al., A 4Gb 2b/cell NAND flash memory with embedded 5b BCH ECC for 36 MB/s system read throughput, in *IEEE International Solid-State Circuits Conference Dig. Tech. Papers*, Feb 2006, pp. 142–143
13. R. Micheloni, A. Marelli, R. Ravasio, *Error Correction Codes for Non-Volatile Memories* (Springer, Dordrecht, 2008)
14. C. Park et al., A high performance controller for NAND flash-based Solid State Disk (NSSD), in *IEEE Non-Volatile Semiconductor Memory Workshop NVSMW*, Feb 2006, pp. 17–20
15. www.pcisig.com
16. R. Budruk, D. Anderson, T. Shanley, Mindshare, *PCI Express System Architecture* (Addison-Wesley, Boston, 2003)
17. K. Kong, *Enabling Multi-peer Support with a Standard-Based PCI Express Multi-ported Switch*, White Paper, www.idt.com Jan 2006
18. K. Kong, Non-Transparent Bridging with IDT 89HPES32NT24G2 PCI Express NTB Switch, AN-724, www.idt.com (Sept 2009)
19. www.ssdformfactor.org

Chapter 3
SAS and SATA SSDs

S. Yasarapu

Abstract This chapter focuses on the different types of solid state drives. The chapter details the differences between consumer and enterprise solid state drives and also details the differences between SAS and SATA solid state drive and what lies ahead for SATA and SAS protocols for SSDs.

3.1 Introduction

Data centers today require fast and reliable storage to provide end-users with high quality of service. Data centers operators are continuously challenged to improve performance to keep up with the demands of high throughput applications. Space, power and cooling limitations require data centers to find the most cost-, space-, and energy efficient products. Solid state drives increase the performance and reliability of the enterprise while reducing the overall space, power, energy footprint of the data centers. However, not all data center and enterprise environments are created equal. Depending on the size, number of users, serviceability requirements and applications running in the data center, the need for performance and storage capacity varies and so do the solid state devices used within these environments.

In fact, not all SSDs are created the same. Some are designed for the enterprise and some are designed for consumer applications. Even in the enterprise segment, some are intended for direct attach to servers and some are designed for shared storage enclosures. Understanding the differences between the various solid state drives helps consumers, as well as, enterprises to select the right solution for their intended applications.

S. Yasarapu (✉)
Director, SSD Product Marketing, STEC, Santa Ana, CA, USA
e-mail: syasarapu@stec-inc.com

3.2 Enterprise vs. Consumer SSDs

Let's first start by understanding the difference between enterprise and consumer solid state devices [1]. To really understand the differences between consumer and enterprise solid state drives, let's start by first observing where these devices are used. This will highlight the fundamental assumptions made by designers of consumer and enterprise solid state drives.

Consumer solid state devices are used in laptops, desktops, and mobile devices where conserving power is the most important criteria to ensure long battery life of the device. Now, let's think about the typical usage pattern of a laptop user. Laptop user, either a business or a home user, generally turns on the laptop at the beginning of the day. Typical applications running on the laptop are email, internet explorer, Microsoft Word, Excel, PowerPoint. For the majority of time, user reads information – reading emails, browsing the web etc. The laptop is perhaps left idle during meetings; is left idle during lunch time. Laptop is turned off at the end of the day. Let's take another example – a desktop user's typical day. In addition to everything the laptop user does, desktop users may also play video games, listen to music and access other digital content. Again, this involves fetching of relatively large amounts of data from the storage -> fast reads. So, what makes the laptop and desktop users happy? Laptops should turn on as soon as they are powered on to minimize the wait for system boot up -> fast boot times; user would like to use the laptop on battery for as long as possible -> low power footprint and email and browser applications should load up fast -> fast reads. Now let's contrast this with typical usage patterns in an enterprise data center.

Enterprise Solid State Drives are used in corporate data centers where uninterrupted operations and high reliability are the most important criteria. Data center of an enterprise is the information technology hub that holds the most important intellectual property of any business/enterprise – DATA. The data stored in the data center is made available via different applications such as Oracle databases, email applications, customer relation management systems and is used by multiple users – R&D, finance, sales, operations, customer service etc. Data is accessed from different locations, at different times. Loss of data is not an acceptable event because of its disastrous consequences to the business. Let's take the example of a financial institution where customers make deposits/withdrawals of money from their accounts. If a withdrawal transaction is lost due to loss of data in the institution's data center then the financial institution loses money. Now this may not seem like a big deal but if it happens systematically, then this could add up to millions of dollars in losses. Or worse, if a deposit amount is not posted to a customer's account, then customer loses money which could be even more disastrous because the bank loses its credibility and hence customers -> loss of revenues. So what makes the *Chief Information Officer* (CIO) happy? All systems in the data center should run uninterrupted -> 24 h/day – 7 days/week – 365 days/year operations with minimal maintenance; there should never be a case leading to data loss – > high reliability; ability to service multiple users at any time -> high performance.

Table 3.1 Application level usage pattern

Criteria	Consumer SSD	Enterprise solid state drive
Hours of operation	Interrupted Fast boot time for frequent power up	24/7/365 Uninterrupted operation
Performance	Fast large block reads only	Fast small block random reads and writes
Access pattern	Single threaded accesses	Multi user accesses
Power consumption	Low power to improve battery life	Reduce total data center power and energy footprint
High availability	Not required	High availability
Reliability	Ease of replacement Loss of data is managed	High reliability Loss of data is catastrophic

As summarized in Table 3.1, we can conclude that the usage pattern of a consumer solid state drive is dramatically different from that of an enterprise solid state drive. This primarily drives completely different design criteria.

Let's see how the consumer and enterprise solid state drives differ in their construction. This will highlight the fundamental assumptions made by the testers and integrators of consumer Flash and enterprise solid state drives. To do this, let's first understand the composition of consumer and enterprise solid state drive. The basic composition of an SSD is a controller and a Flash as shown in Fig. 3.1. But that is where the similarity ends.

What really separate enterprise solid state drive from consumer SSD is the design of the controller hardware and more importantly the controller firmware features and the rigors of testing and qualification process the enterprise solid state drive is put through before it makes it to the market in a product form.

Controller hardware and firmware running on the Enterprise SSDs are the brains of the device. Their primary functions are to respond to host commands, to transfer data between the host and Flash media and to manage the Flash media to achieve high reliability and endurance throughout the operational lifetime of the drive. How well a controller handles Flash management and host data transfers simultaneously is what differentiates it from a consumer SSD. In addition, enterprise SSDs have additional built-in features to improve the reliability and endurance of the Flash and hence the enterprise SSD. Enterprise solutions require 24/7/365 uninterrupted operation. Therefore, controllers in enterprise SSDs are designed to maintain consistent performance behavior while transferring data irrespective of the amount of Flash capacity in use and also the traffic generated to the drive. Wear leveling operations and background media error correction algorithms are designed such that data transfer performance to the host is unchanged while these operations run in the background to the Flash.

Enterprise solutions are required to support a large number of users, i.e., multiple initiators running different types of traffic patterns independent of one another resulting in random traffic. Therefore, the controller hardware and firmware is designed to support multi-threaded access where up to hundreds of streams of

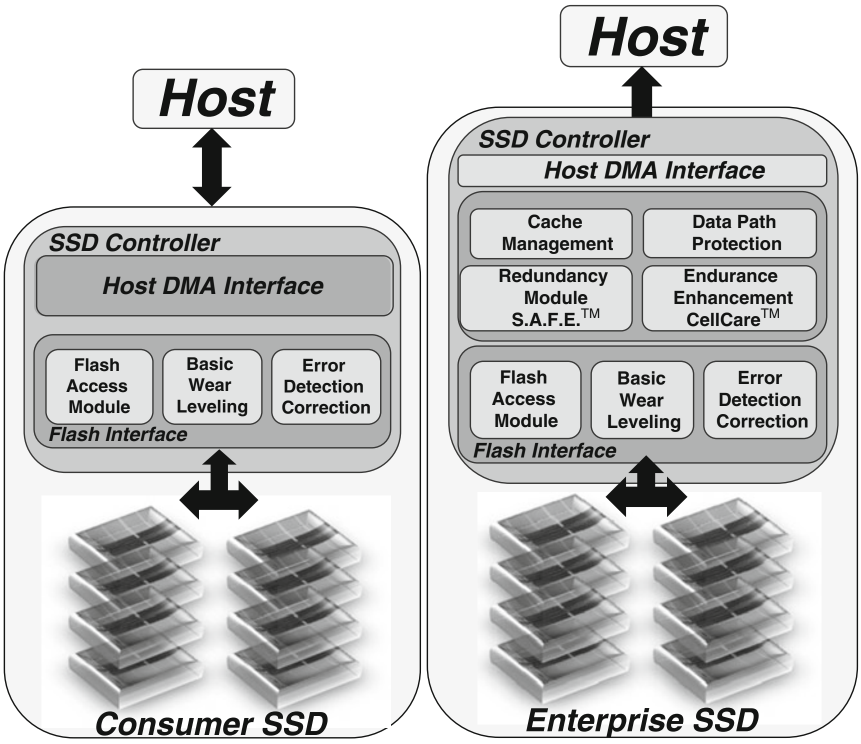


Fig. 3.1 Consumer (*left*) and enterprise (*right*) solid state drives

data per drive can be pushed between host and the device while maintaining the performance as well as integrity of data. Therefore, enterprise SSDs are designed to perform extremely well even for small transfers of varying sizes and for simultaneous reads and writes.

Data integrity and availability is of the highest importance in enterprise solutions. Therefore, enterprise SSDs are designed to provide full data path protection with ECC and CRC coverage and power fail protection against unscheduled power loss.

Reliability and high endurance are extremely important for enterprise application because solutions deployed into enterprise have a long working life. Unlike consumer deployments, enterprise deployments have a long service life. Therefore, enterprise SSDs are designed to survive in mission critical storage area networks under 24/7/365 workloads for over 10 years. To this effect, enterprise SSDs have built-in redundancy to ensure that even if Flash die fails, the SSD can successfully recover data by using the redundancy built into the data stored on the Flash.

Enterprise SSDs are also built with features to improve the endurance of the Flash – one such feature is *CellCare*^{TM}. Though this capability is not yet available on all enterprise SSDs, it is absolutely required to counter the deterioration in Flash endurance as technology nodes shrink.

Enterprise SSDs have the characteristics of drives designed for use in all environments (like the ones on Mars): this allows for drive to operate in environments that do not require human presence and can handle unknown conditions as they arise.

The above mentioned design capabilities are driven by the application use cases where enterprise solid state drives are used. Consumer SSD, unlike enterprise SSD is not designed with these assumptions and is therefore unsuitable for enterprise applications.

Consumer SSDs are designed for cost, which may or may not include robust controller/Flash management technology. Consumer SSD doesn't have power fail protection and do not have ECC and CRC protection that ensures full data-path protection. Consumer SSDs are not designed to endure under enterprise workloads; they are designed for laptops and desktops not expected to work beyond a few years.

Since consumer SSD is focused on providing faster boot time, and application load time, they are optimized to provide fast large block read transfers. Given that consumer SSD is left idle for long durations, consumer SSD depends on host side to manage the SSD media. This in turn leads to short lifetime of the consumer SSD. In addition, consumer SSD is designed for single operation management, for data loading, installing, saving, etc.

Consumer SSDs have been designed for single user usage and are only designed for less than a year of operation because of higher rate of Flash wear out due to sub-standard wear leveling, where only a small amount of data is written and up time is average of 4 h with many hours of idle time.

Therefore, consumer SSDs though suitable for low end applications where the devices are not challenged to work at high performance levels, are not suitable for high performance, high reliability enterprise deployments. Typically consumer SSD uses SATA interface to connect to host systems.

Enterprise SSDs come in different form factors with different interfaces. There are 3 main interface protocols used to connect SSDs into server and/or storage infrastructure: *Serial Attached SCSI* (SAS), *Serial ATA* (SATA) and PCIe. PCIe based SSDs delivering the highest performance are mainly used in server based deployments as a plug in card inside a server. SAS SSDs deliver high levels of performance and are used in both high end server and midrange – high end storage enclosures. SATA based SSDs are used mainly in client applications and in entry and midrange server and storage enclosures as shown in Fig. 3.2.

Per a recent IDC study [2], SAS and SATA SSDs combined continue to hold the lion share of the enterprise SSD market, $>70\%$, as shown in the Fig. 3.3 below.

In this chapter, let's focus on the SAS and SATA SSD – protocol differences, key feature highlights, similarities and differences and where they are used.

3.3 SAS vs. SATA Protocol

Serial Attached SCSI (SAS) is a communication protocol traditionally used to move data between storage devices and host. SAS is a point to point connection using a serial physical connection. It uses a standard SCSI command set to drive device

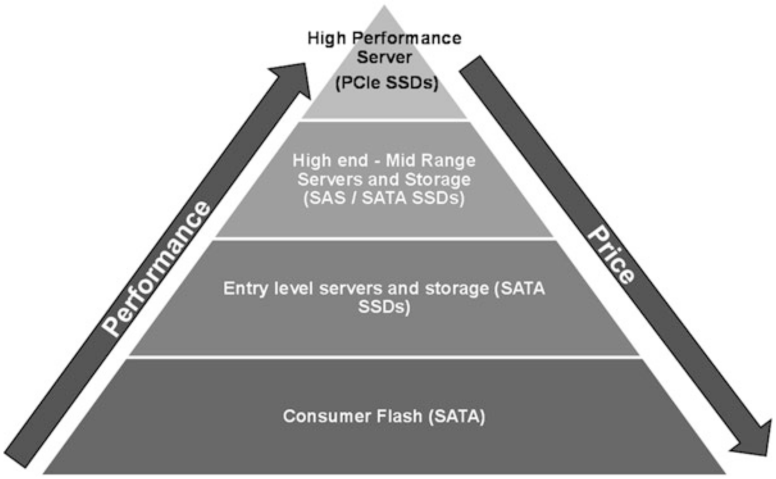

High Performance Server (PCIe SSDs)
High end - Mid Range Servers and Storage (SAS / SATA SSDs)
Entry level servers and storage (SATA SSDs)
Consumer Flash (SATA)
Performance
Price

Fig. 3.2 Different types of SSDs

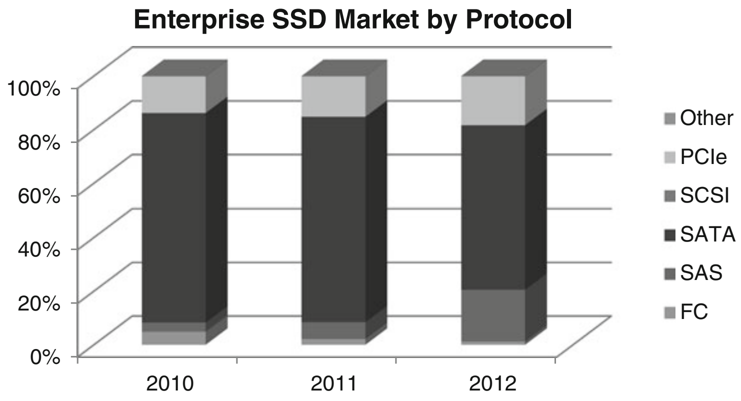


Fig. 3.3 Enterprise SSD market share by protocol [2]

communications. Today, SAS based devices most commonly run at 6 Gbps. There is ongoing development of a faster 12 Gbps SAS based devices that will be brought to market in the near future. On the other side, SAS interface can also be run at slower speeds – 1.5 Gbps and/or 3 Gbps to support legacy systems.

SAS also offers backwards-compatibility with second-generation SATA drives. The T10 technical committee of the International Committee for Information Technology Standards (INCITS) develops and maintains the SAS protocol; the SCSI Trade Association (SCSITA) promotes the technology.

Serial ATA (SATA or Serial Advanced Technology Attachment) is another interface protocol used for connecting host bus adapters to mass storage devices such as hard disk drives and solid state drives. Serial ATA was designed to replace

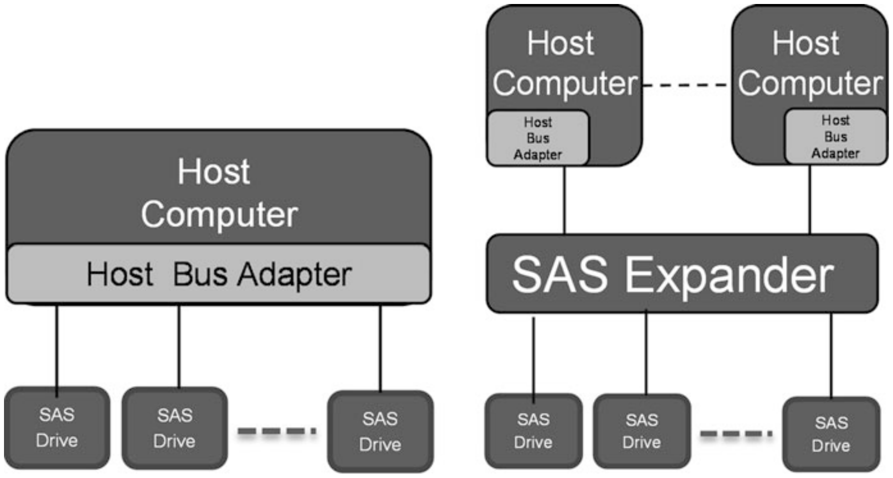


Fig. 3.4 SAS connectivity

the older parallel ATA/IDE protocol. SATA is also a point to point connection using a serial physical connection. It uses ATA and ATAPI command set to drive device communications. Today, SATA based devices most commonly run either at 3 Gbps and/or 6 Gbps.

Serial ATA industry compatibility specifications originate from The Serial ATA International Organization [3] (aka. SATA-IO).

3.3.1 Connectivity and High Availability

A typical SAS eco-system consists of SAS SSDs plugged into a SAS backplane or a host bus adapter via a point to point connection, which in turn is connected to the host microprocessor either via an expander or directly, as shown in Fig. 3.4.

Each expander can support 255 connections to enable a total of 65535 (64 K) SAS connections. Therefore, SAS based deployments enable use of a large number of SAS SSDs in a shared storage environment.

SAS SSDs are built with two ports. This dual port functionality allows host systems to have redundant connections to SAS SSDs. In case one of the connections to the SSD is either broken or malfunctions, host systems still have the second port that can be used to maintain continuous access to the SAS SSD. In enterprise applications where high availability is an absolute requirement, this feature, unique to SAS SSDs, makes it the SSD of choice for enterprise applications. Figure 3.5 below shows the dual port connector used with SAS SSDs.

SAS SSDs also support hot plug. Hot plug feature enables SAS SSDs to be dynamically removed or inserted while the system is running. This feature allows for

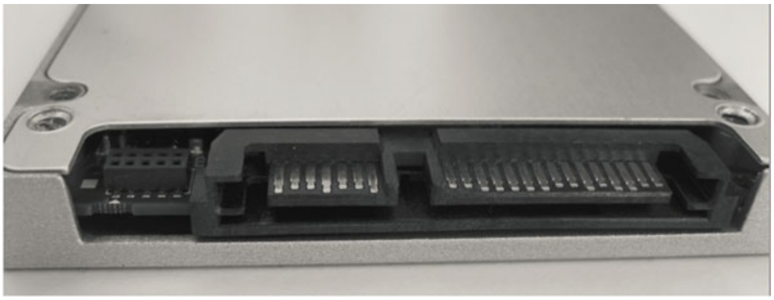


Fig. 3.5 Dual port SAS connector

automatic detection of newly inserted SAS SSDs. While a server or storage system is running, newly inserted SAS SSDs can be dynamically configured and put to use. Even more importantly, even if SAS SSDs are pulled out of a running system, all the in-flight data that is committed by the host system is properly stored inside a SAS SSD and can be accessed once the SSD is powered back on.

As opposed to SAS, a typical SATA eco-system consists of SATA SSDs connected to host bus adapter via a point to point connection, which in turn is connected to the host microprocessor. In addition, SATA SSDs are built with one port unlike SAS SSDs. These two main differences make SATA based SSDs more suited for entry or mid-range deployments and consumer applications.

SATA SSDs also support hot plug which enables SSDs to be dynamically removed or inserted while the system is running. While a server or storage system is running, newly inserted SATA SSDs can be dynamically configured and put to use. However, not all SATA SSDs are designed to withstand hot plug functionality and to ensure that if pulled out of a running system, all the inflight data that is committed by the host system is properly stored inside a SATA SSD. This capability, also commonly known as power failure protection, is an extremely important feature and is generally only supported by selected enterprise grade SATA SSD vendors. Most SATA SSDs are not built with this feature.

SATA drives may be connected to SAS backplanes, but SAS drives may not be connected to SATA backplanes.

This is an important feature, in that physically SAS infrastructure is designed to accommodate SATA SSDs. Connector on SATA SSDs is designed such that they can be plugged into SAS receptacles though the reverse is not true. This enables SATA SSDs to be plugged into SAS based storage system making the SATA SSD more ubiquitous for use.

In addition, even though SAS uses SCSI as the primary communication protocol, SAS also supports STP (*Serial ATA Tunneled Protocol*) that allows SAS infrastructure is built to ensure communication with SATA SSDs hence enabling interoperability. Again, reverse is not true, in that SAS SSDs cannot be plugged into SATA based deployments.

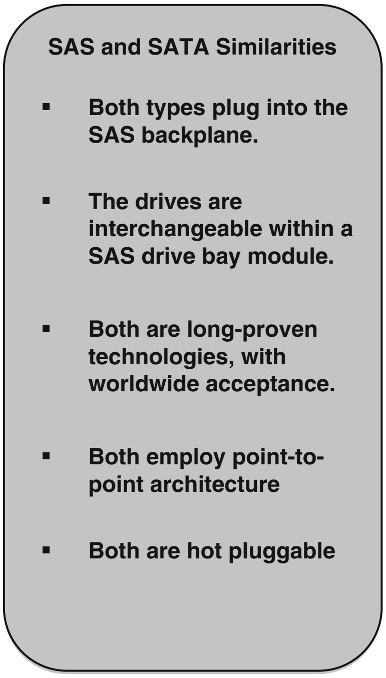


Fig. 3.6 Similarities between SAS and SATA technologies

SAS and SATA Similarities

- **Both types plug into the SAS backplane.**
- **The drives are interchangeable within a SAS drive bay module.**
- **Both are long-proven technologies, with worldwide acceptance.**
- **Both employ point-to-point architecture**
- **Both are hot pluggable**

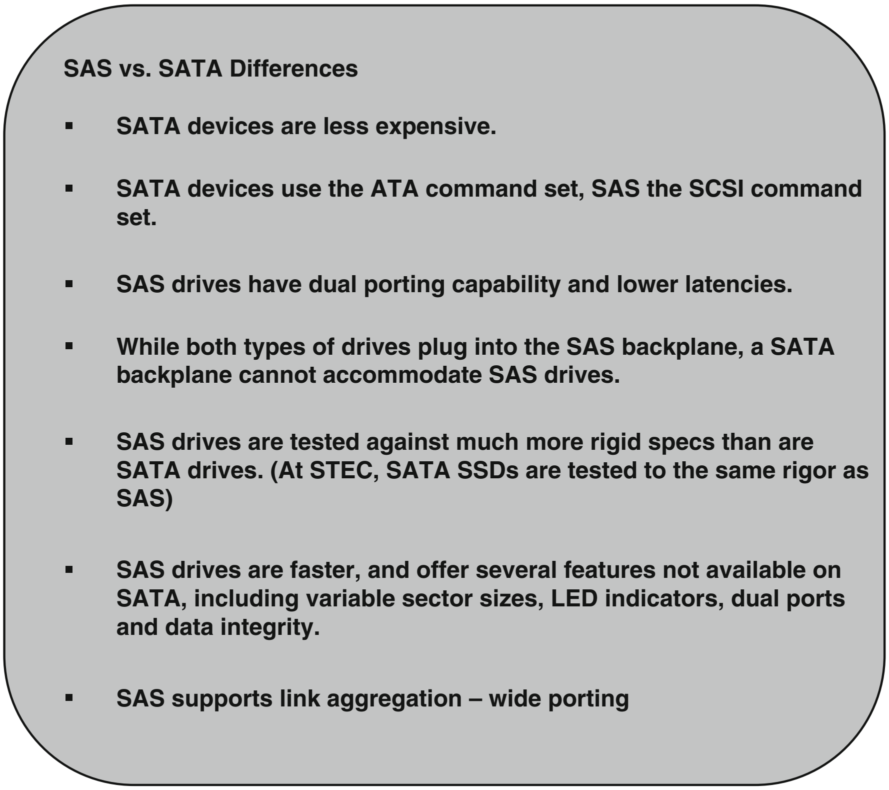
Similarities between SAS and SATA technologies are summarized in Fig. 3.6; differences between the two are in Fig. 3.7.

3.3.2 Form Factor and Capacity

SAS and SATA SSDs come in a variety of capacities and form factors.

SAS SSDs are designed in two popular form factors – 2.5″ and 3.5″ drive form factors. This form factor is primarily defined and driven by the small form factor working group and the T-10 organization. Since SAS SSDs are designed primarily for the enterprise, the capacity of SAS SSDs varies from 100 GB up to 2 TB in capacity.

SATA SSDs are designed in a variety of form factors – 2.5″, 1.8″ as well as smaller form factors – MO-297, SlimSATA, mSATA (Fig. 3.8). Typical enterprise applications use either 1.8″ or 2.5″ SATA SSDs. For example, 1.8″ SATA SSDs are popularly used in blade servers as boot devices. The smaller form factors enable SATA to be used in space constrained embedded applications. SATA SSDs in capacity vary anywhere between 25 GB–400 GB and are generally used either in consumer or entry and mid-range data center applications.



SAS vs. SATA Differences

- **SATA devices are less expensive.**
- **SATA devices use the ATA command set, SAS the SCSI command set.**
- **SAS drives have dual porting capability and lower latencies.**
- **While both types of drives plug into the SAS backplane, a SATA backplane cannot accommodate SAS drives.**
- **SAS drives are tested against much more rigid specs than are SATA drives. (At STEC, SATA SSDs are tested to the same rigor as SAS)**
- **SAS drives are faster, and offer several features not available on SATA, including variable sector sizes, LED indicators, dual ports and data integrity.**
- **SAS supports link aggregation – wide porting**

Fig. 3.7 Main differences between SAS and SATA

3.3.3 Performance

SAS uses SCSI command set to transfer data. SCSI is a more efficient command set with features such as command queuing that enable higher performance of SAS SSDs. Therefore, SAS SSDs are used where extremely high performance is required. Unlike SAS, SATA SSDs using the ATA protocol have lower performance compared to SAS and therefore are more widely for mid-range and entry level system.

However, a point to note is that both SATA and SAS SSDs are orders of magnitude faster than hard disk drives (HDDs). To better understand the performance characteristics of SSDs first, it is important to know what is inside an SSD compared to HDD.

Hard disk drives are electro-mechanical devices which inherently is limited by the mechanical element utilized to build them, i.e., rotating magnetic disk. In order to retrieve data that is stored on the magnetic disk, one must rotate the disk to place it under the media head (rotational latency), moving the head to the right track (seek

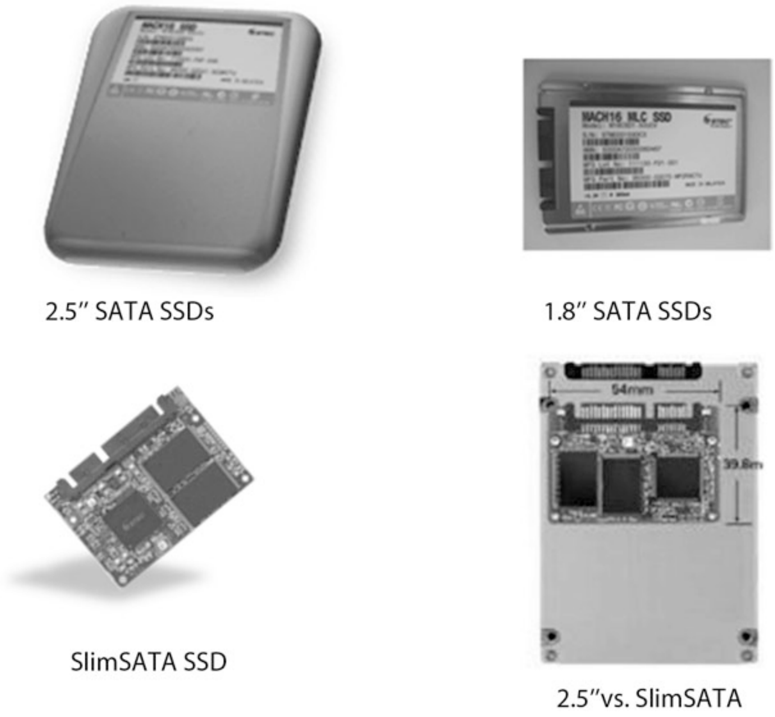


Fig. 3.8 SATA form factors

latency) and then using a combination of electronics and mechanics to transfer the data to/from the host devices (transfer time). The only way to hide rotational and seek latencies is by transferring large sequential data from the disk once the right track on the disk is located. Therefore, hard drives are inherently sequential devices and limited in random performance. Sequential performance is generally measured in MBps or GBps, whereas, random performance is measured in IO per seconds (IOPs). The fastest hard drives on the market today provide at best 350 IOPS under random workloads. However, real world applications are random by nature.

In contrast to hard drives, solid state drives are electronic devices. There are no mechanical elements on a solid state drive. Data is stored in NAND Flash devices, and is retrieved from the NAND Flash by on board controller. All blocks of data on the NAND Flash are equally accessible by the controller, i.e., there are no rotational and/or seek latencies to get to the right block of data.

Performance, reliability, and endurance of SSDs are highly dependent on the design of the SSD controllers as discussed in earlier sections.

How efficiently HW (Hardware) and FW (Firmware) of the SSD controller handle data streaming while also performing Flash management determines the performance of the SSD. Controllers in enterprise SAS and SATA SSDs are

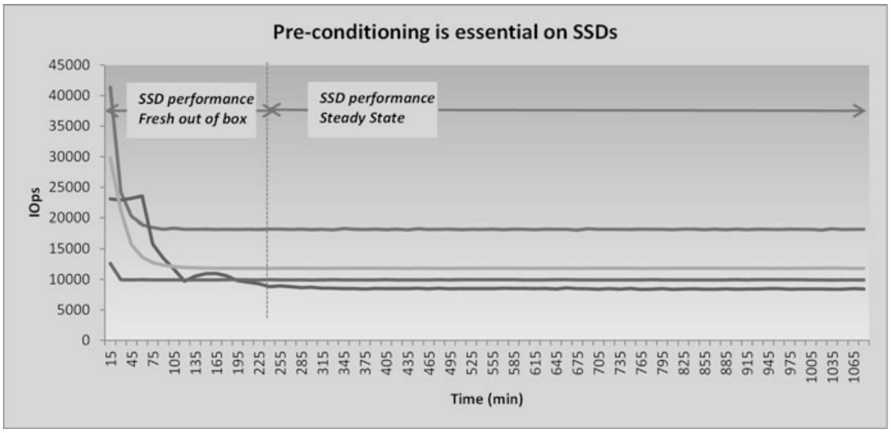


Fig. 3.9 Effect of preconditioning on performances

designed to maintain consistent performance behavior while transferring data, regardless of the amount of Flash capacity in use, and irrespective of the volume of traffic being generated to the drive at any point in time. Wear-leveling operations and background media error correction algorithms are designed so that data transfer performance to the host is unchanged while these operations run in the background. An enterprise-class SSD is designed to handle these heavy workloads 24/7/365 for 5 years or more.

To be of real value, SSD performance needs to be measured after the SSD reaches steady. Performance measured on a fresh out of the box SSD – SAS or SATA will not truly represent the performance of the drive in a real deployment. Therefore, before measuring SSD performance, one must precondition the SSD under test. This is accomplished by writing random data patterns to completely fill all NAND blocks and engage the drive's wear-leveling and Flash management routines. Properly managing data flow and internal NAND will make the measurement a more useful gauge of SSD performance under real-world conditions. Figure 3.9 illustrates the higher performance of fresh out of box SSDs that reach steady state after pre-conditioning the SSD.

To understand the real world benefits of SAS and SATA SSDs, performance is usually measured for large block 128 KB or larger sequential and small block 4 KB or 8 KB random read, write and mixed workloads.

Figures 3.10 and 3.11 show a real world comparison between SAS and SATA SSDs. As seen in these charts, SAS SSDs deliver almost 2x higher performance compared to SATA SSDs.

As seen from the charts above, both type of enterprise SSDs – SAS or SATA, have place in the data center. SAS SSDs are used for high end performance critical enterprise systems and SATA SSDs are used with mid-range or entry level systems.

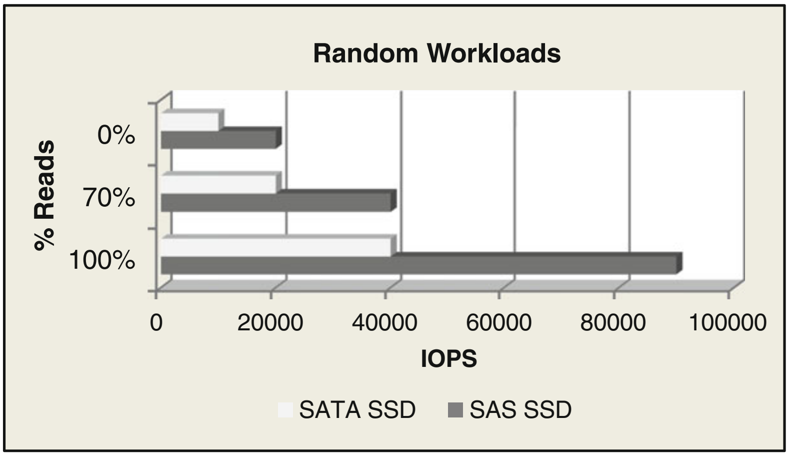


Fig. 3.10 SAS vs. SATA under random workloads

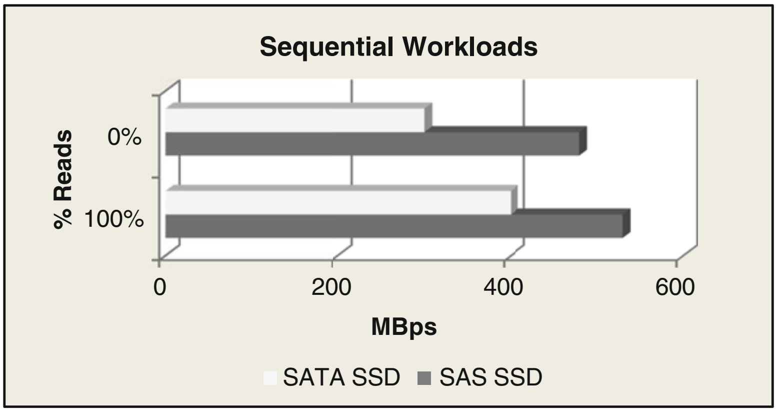


Fig. 3.11 SAS vs. SATA under sequential workloads

3.4 What's Ahead

SATA and SAS based SSDs are increasingly being adopted in consumer and enterprise applications [4]. This adoption is expected to continue and expand in the coming years. Data center and enterprise applications are using increasingly large amounts of SSDs and also SSDs of higher capacities to deliver on the need for ever increasing demands for data storage. As the NAND Flash geometries shrink, the capacity of SATA and SAS SSDs is expected to increase to address this need for higher capacity SSDs.

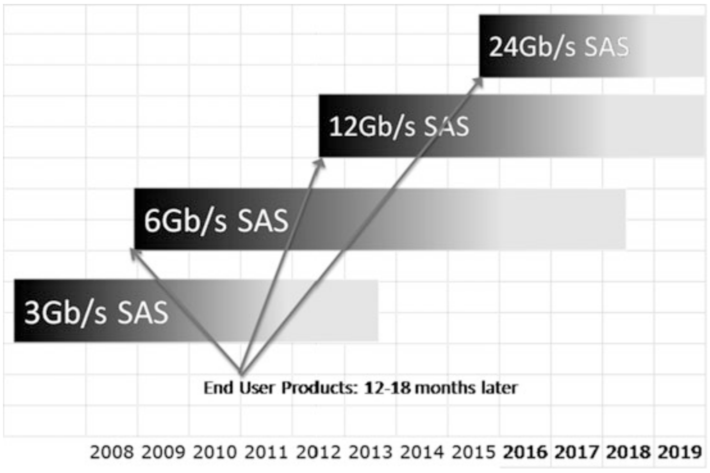


Fig. 3.12 SAS speed evolution

On the SATA front, the SATA protocol is expected to continue to deliver 6 Gbps interface speeds for the near future. There is ongoing discussion in the industry to determine the next generation technology after SATA 6 Gbps.

In addition to the explosive growth in data, the definite shift in consumer and data center demands for instant data delivery is driving the need for even higher performing SSDs in the enterprise. To enable this, SAS interface is expected to double its speed to support 12 Gbps and then 24 Gbps interface speeds as shown in Fig. 3.12.

As discussed above, enterprise solid state drives increase the performance and reliability of the enterprise while reducing the overall space, power, and energy footprint of the data centers. Key features for the enterprise are long service life, high endurance, consistent and high performance, high reliability which are delivered by SAS and/or SATA SSDs.

Choosing the right SSD – SATA or SAS, depends on the end user application. Use of SSDs leads to improved performance, higher reliability and reduced power space and energy consumption which reduces CapEx and OpEx of next generation data centers. This is what makes SSDs a great product to enable highest levels of performance and fast access to data in consumer as well as enterprise applications.

References

1. www.stec-inc.com
2. IDC, Worldwide Solid State Drive 2011–2015 Forecast and Analysis, Doc # 228894, Jun 2011
3. http://www.-t10.org
4. http://www.sataio.org

Chapter 4
Hybrid Storage

R. Micheloni, L. Crippa, and M. Picca

Abstract In recent years, both industry and academia have increased their research effort in the hybrid memory management space, developing a wide variety of systems. It is worth mentioning that "hybrid" is a generic term and it can have different meanings depending on the context. For instance, a storage system can be hybrid because it combines HDD and SSD; an SSD can be hybrid because it combines SLC and MLC Flash memories, or it combines different non-volatile memories like NAND and ReRAM. In this chapter we look at all these different meanings.

The last section covers over-provisioning and the *Write Amplification Factor* (WAF): these parameters have a great impact on SSD performances and reliability, as well as on the available storage capacity.

4.1 NAND Flash Memory and HDD

If we look at the DRAM history [1], DRAM data access speeds have increased at a faster rate than *Hard Disk Drives* (HDDs), leaving a gap in the memory hierarchy as shown in Fig. 4.1. The gap in read and write performances between DRAM and HDD has widened in the last years, leaving an opportunity for a new intermediate memory/storage technology between HDDs and DRAM: NAND Flash memory can fill this performance gap.

R. Micheloni (✉) • L. Crippa
Integrated Device Technology, Enterprise Computing Division, Agrate Brianza, Italy
e-mail: rino.micheloni@ieee.org; luca.crippa@ieee.org

M. Picca
Thales Alenia Space, Vimodrone, Italy
e-mail: massimiliano.picca@thalesaleniaspace.com

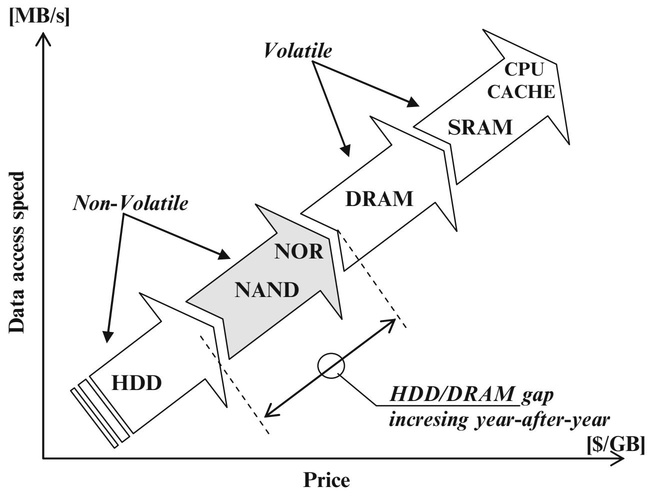


Fig. 4.1 Memory hierarchy

While HDDs are the most common secondary storage devices, their high power consumption and low shock resistance limit them as an ideal mobile storage solution [2]. On the other hand, Flash memories (especially of NAND type) overcome the main problems of HDDs, but they are still more expensive and can only support a limited number of erase cycles [3].

Researchers generally agree that disk-storage performance is subject to the handling of small files and filesystem metadata. Unlike traditional disk storage, flash memory has no seek penalty, but is subject to garbage collection and wear leveling.

To avoid excessive wear-out of Flash memories, and to mitigate their low write throughput, it is a good approach to migrate frequently-read data to the Flash and frequently-written data to HDD, as sketched in Fig. 4.2. In other words, there should be a caching software that dynamically manages the use of the entire drive capacity for superior overall storage performance, where the most frequently/recently used "hot" data are cached for ultra-fast access, while the "cold" data remains on the primary storage partition.

The trade-offs associated with HDDs and Flash memories motivate lots of storage system designs [4–8]. Many applications use Flash memory as a non-volatile cache storing data blocks which are likely to be accessed in the near future, and thus allowing the disk to spin down for longer periods.

However, these schemes treat flash memory as complement of DRAM buffer cache, and only a subset of data blocks are cached in flash memory; as a result, the disk is used quite frequently due to cache misses or flushing. As flash memory's capacity increases, a real hybrid secondary storage solution is expected to be more effective [9]. Different from data block level cache, Flash memory stores files and can be accessed independently in hybrid secondary storage system.

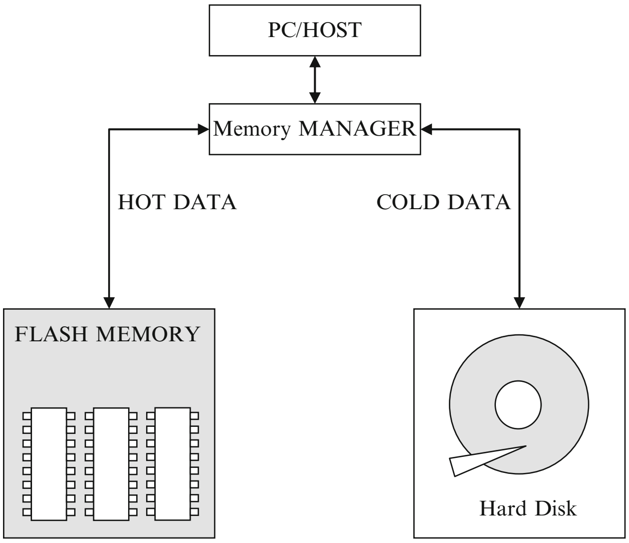


Fig. 4.2 The hybrid storage system

In recent years, both industry and academia have increased their research effort in the hybrid memory management space, developing a wide variety of systems [10–12]. At this point it is worth mentioning that "hybrid" is a generic term and it can have different meanings depending on the context. Figure 4.3 is a summary of what a hybrid storage could be.

We will look at each of these ways to combine Flash memory and HDDs in the following sections. The reader can refer to Chap. 13 for an example of an SSD integrating different non-volatile technologies (NAND/ReRAM).

4.2 External NAND + HDD

One of the first examples of NAND used as an external memory was ReadyBoost [13–15]. It works by using flash memory, a USB flash drive, SD card, CompactFlash or any kind of portable flash mass storage system as a cache, as shown in Fig. 4.4.

The core idea of ReadyBoost is that a flash drive has a much faster seek time than HDD, allowing it to satisfy requests faster than reading files from the hard disk.

When an *EXternal Memory* (EXM) is plugged into the computing device, the system populates EXM with disk sectors and/or memory sectors. The system routes I/O read requests directed to the sector to the EXM cache instead of the actual sector. The use of EXMs increases performance and productivity on the computing device systems for a fraction of the cost of adding memory to the computing device.

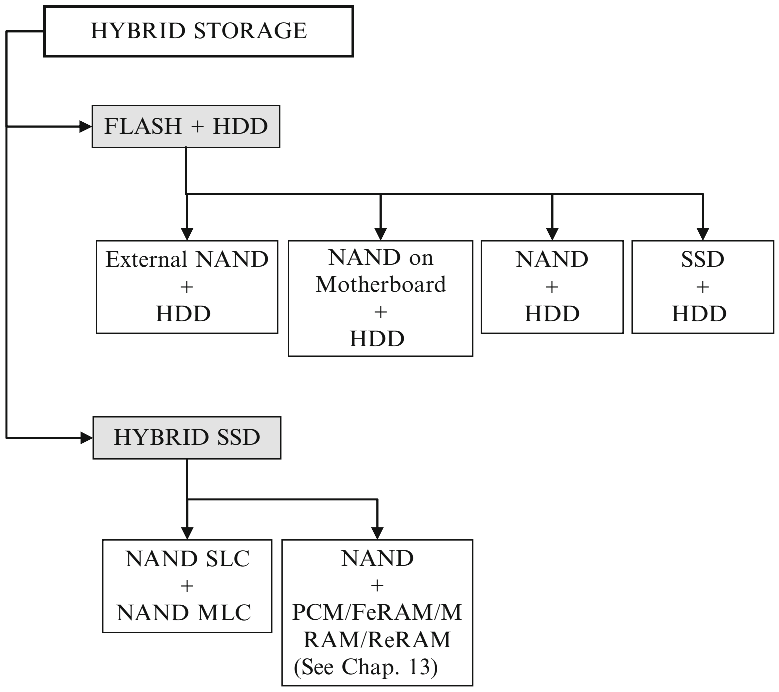


Fig. 4.3 Hybrid storage overview

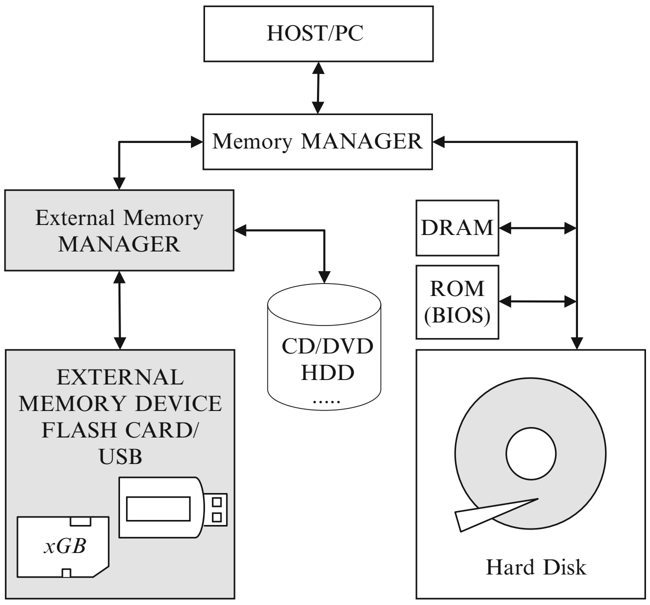


Fig. 4.4 Flash memory as external memory device

The system detects when an EXM is used for the first time. Once the type of EXM is discovered, a driver is installed and it is used to cache disk sectors on the external memory. Sectors from any disk and/or slower memory device on the system can be cached to EXM. Without a prior knowledge of which sectors are valuable in terms of frequent access, the system may use data on the computing machine to determine which sectors are used to populate the EXM cache. Alternatively, the system populates the EXM cache with a particular sector when that particular sector is accessed during operation. The next time that particular sector is to be accessed for a read operation, the system directs the read operation to access the copy from the EXM. The system may track usage patterns and determine which disk sectors are most frequently accessed. On subsequent uses of the EXM, the system caches those sectors that are most frequently accessed onto the EXM. If the EXM is present when the computing device is powered up, the EXM can be pre-populated with data during start-up of the operating system [13].

4.3 NAND on Motherboard + HDD

Computer motherboards contain the processor chip and some high performance SRAM and DRAM memories. In the last few years there have been proposals to add Flash memory to the computer motherboard for a non-volatile memory layer to the motherboard memory/storage architecture. The motherboard Flash memory could be inserted into the motherboard with an ONFI module or DIMMs similar to those currently used for DRAM, allowing memory replacement when faster or larger memory becomes available.

Intel introduced a motherboard Flash memory technology in 2007, known as “Robson Technology” or “Turbo Memory” [16, 17]. This early implementation ran into issues due to lack of support for management of Flash/HDD partition in main operating systems. In fact, central to the operation of any hybrid storage computer architecture is management to determine which data is to be kept on the HDD and which data will be kept on the Flash memory.

Figure 4.5 shows a storage management controller that determines what data should be stored on each memory device. This storage management function must balance the needs of data access, power savings opportunities, and data security.

As with any NAND based memory product solution, the NAND flash memory controller is also key in executing the NAND wear leveling algorithm, managing the reads, writes, erases, and performing the ECC (Error Correction Code) as needed [16].

With NAND moving into the demanding computing environment, the wear leveling algorithm must comprehend not only the usage statistics of the NAND flash but also track the key reliability statistics. In other words, the controller must track all the failure mechanisms known in the NAND Flash industry (Chap. 8): program disturb, read disturb, program/erase cycles, data retention, etc.

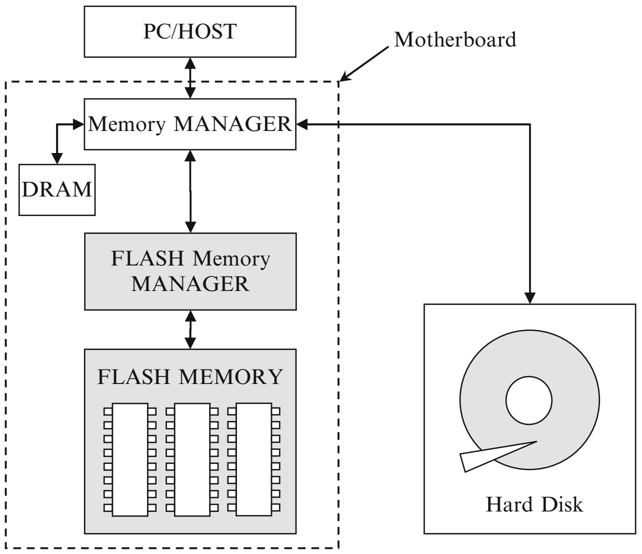


Fig. 4.5 Flash on computer motherboard

In the next section, HDD is combined with another drive, a *Solid State Drive* (SSD).

4.4 NAND/SSD + HDD

A block diagram of the monolithic HDD + SSD solution, usually referred to as hybrid drive, is shown in Fig. 4.6 [18–25]. A Solid State Drive is made up of several NAND chips plus a controller: therefore, all the considerations of this section also apply to a storage system composed by HDD and a single NAND device. Unlike standard HDDs, the hybrid drive in its normal state has its platters at rest, without consuming power or generating heat. When reading data from the platters, extra data are read and stored in buffer memory in the hope of anticipating future requirements as in any disk cache. For example, data required for the next boot-up can be stored in the non-volatile buffer before shutting down the computer.

In 2010 Seagate released the Momentus XT [20, 21], which uses so-called "adaptive memory" for its SSD portion, which does not rely on driver support from the operating system. This removes the need for a special operating system, and the speed benefits can be used by any OS.

The Flash memory is used to store frequently accessed content using an adaptive memory algorithm. This algorithm monitors data access transactions and maintains frequently accessed data on the Flash memory. The drive includes software that tracks a person's use trends and then uses the SSD component of the drive to

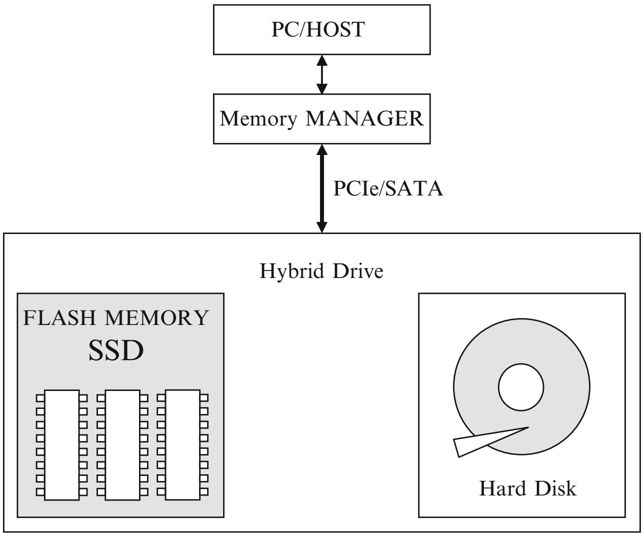


Fig. 4.6 Monolithic hybrid drive

optimize performance, and it can adjust that performance over time with changes in user behavior. Up to 50% performance improvement is seen between the first and second iteration of data access [18].

Manufacturers claim several benefits of the hybrid drive over standard hard drives, especially for use in notebook computers: among them, speed of data access and consequent faster computer boot process, decreased power consumption, and improved reliability.

There are some drawbacks too, especially when accessing non-cached data. In fact, if the data being accessed is not in the cache and the drive has spun down, access time will be greatly increased since the platters will need to spin up.

Another concern is the lower performance for small disk writes. NAND is significantly slower when writing small data; an effect that is amplified when the file system is using journaling techniques.

Anyhow, hybrid drives have a great potential and the industry is actively working in this field. As a matter of fact, Windows Vista and Windows 7 natively support the use of hybrid drives (ReadyDrive) [22].

As mentioned, a NAND device can experience a limited number of program/erase cycles. With the hybrid drive, a simple solution to mitigate this wear-out effect would be to place all the data that is accessed by read operations on the Flash memory device, and the remaining data on the HDD. This placement would save a substantial amount of the energy consumption while a longer lifetime for the Flash memory device is expected [12].

However, in practice, we cannot know in advance whether data should be placed on the Flash memory device or the hard disk.

We now review an existing method of skewing frequently accessed data, called *Popular Data Concentration* (PDC): it was proposed by Pinheiro et al. [23] to deal with the highly skewed file access frequencies exhibited by the workloads of network servers. The idea of PDC is to concentrate the most popular (i.e. most frequently accessed) disk data by migrating it to a subset of the disks, so that the other disks can be sent to a low-power mode to conserve energy. PDC redistributes data across the disk array according to its popularity, so that the first disk stores the most popular data, the second disk stores the next most popular data, and so on.

However, if the frequency of file access varies significantly with time, PDC may cause a lot of file migrations, which will increase energy use, in particular by disturbing idle disks. This also happens when new files are created, because they will be stored on the disk with the least popular data, which has to be woken up.

PDC concentrates on popular data without considering whether I/O accesses are reads or writes. If we split I/O transactions into reads and writes and move only the data corresponding to one sort of access, we can reduce the amount of migrations. For instance, if the total amount of data associated with reads is less than that associated with writes, then transferring the data that is being read will be more profitable. This scheme is called PB-PDC (pattern-based PDC): it improves the PDC technique by moving frequently-accessed read and write data to separate sets of disks [9].

Thus, while the disks containing data which are accessed in one way (read or write) are being accessed frequently, the disks storing data accessed in the other way can be sent to a low power mode to conserve energy.

We can apply PB-PDC to a hybrid drive. Because a Flash memory device has low write throughput and limited erasure cycles, PB-PDC moves the popular write data to the hard disk and the popular read data to the Flash memory device.

Another possible approach when looking at data partitioning within a hybrid drive is to employ cache device organization where a subset of disks are treated in the storage system as cache disks to absorb I/O traffic [24].

Summarizing, PDC does not ask for file duplication while, in the caching approach, files in Flash memory are a copy of that on disk.

The cached file selection algorithm decides files to be cached in Flash. Usually, both static and dynamic types of selections can be used. The static approach is more suitable for files frequently accessed by users: for example, the operating system, compiler and some C libraries.

When the remaining capacity of Flash memory cache device reaches a threshold value, replacement is needed. The main guideline for replacement algorithm is that files accessed less frequently and files that will not be accessed in near future should be removed from Flash memory cache. The oldest and yet still widely used algorithm in cache management is LRU [12].

The above mentioned algorithms are just a small part of what is available in the open literature: it is clear that in order to really exploit all the benefits of hybrid storage, it is fundamental to decide where it is the right place to store data, depending

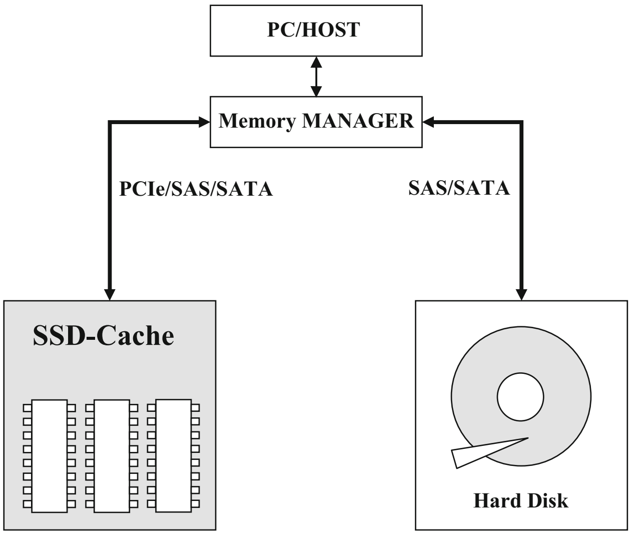


Fig. 4.7 SSD-Cache

on their characteristics. Of course, workloads are application and user specific: therefore, the storage management algorithm should be able to adapt to different needs.

At the end of this section it is worth mentioning that another term is becoming very popular in the hybrid storage world: SSD-Cache [26].

SSD-Cache is a discrete, separate memory component, as sketched in Fig. 4.7: in other words, HDD and SSD are housed separately. While all the hot/cold topics mentioned above remain valid, discrete cache SSDs and HDDs are easier to scale, with a broad selection of drive manufacturers [27–32].

4.5 Hybrid SSD

NAND Flash memories fall into different categories, depending on the number of bits stored in the same physical cell [33], as shown in Fig. 4.8. SLC and MLC store 1 and 2 bits per cell, respectively. *Triple-Level Cell* (TLC) stores 3 bits within a memory cell; 4 bit/cell is still a research topic.

Downsides of storing more bits per cell are slower speeds, higher error rates and lower endurance/retention [34, 35]. The advantage is clearly the reduced silicon area, and therefore cost. Of course, NAND with equal number of bits per cell can have different performances: this can be related to either process or design aspects. For example, when the process technology moved from 5X nm to 3X nm, MLC endurance changed from 10 k program/erase (P/E) cycles to 5 k. 2X nm is now in the range of 3 k [30, 36, 37].

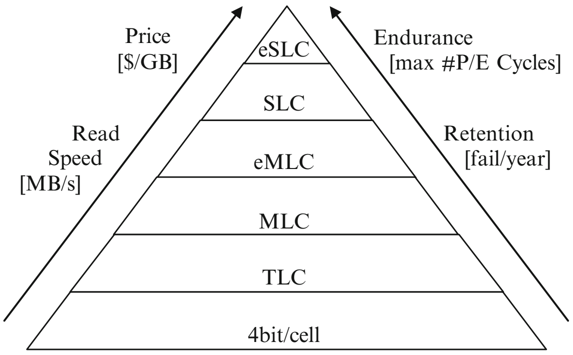


Fig. 4.8 NAND flash families

Table 4.1 SLC and MLC specifications

NAND type	SLC	MLC
Page read	25 μs	50–60 μs
Page write	200 μs	800–1,200 μs
Block erase	1–2 ms	1–2 ms
Endurance	100 K	5–10 K
Operating (read, write, erase voltage/current)	3.3 V/15 mA	3.3 V/15 mA

eMLC and eSLC ("e" stands for enterprise) offer a higher number of erase/program cycles. For instance, if standard MLC and SLC run for 10 k and 100 k program/erase, respectively, eMLC can sustain 30 k and eSLC 300 k [38].

Table 4.1 compares typical SLC and MLC specifications [38]: SLC is much faster than MLC during both read and write.

Performances of an individual Flash device are still insufficient to meet the bandwidth requirements of the interface (SAS/SATA/PCIe) and, therefore, interleaving is very common in most high-performance SSDs. The interleaving technique is also useful to extend the endurance because write operations can be distributed over multiple devices [39].

Because of the cost benefit, there have been many attempts to address performance and endurance problems in MLC-based storage systems. One possible approach is to combine SLC and MLC Flash memories inside a single SSD, which is called "hybrid" [40–46]. A basic block diagram is shown in Fig. 4.9. The goal of this hybrid-SSD design is to achieve the response time of SLC, while having the cost structure of MLC. In other words, SLC capacity must be small.

The basic idea is to use SLC for storing small random (hot) data and MLC for large sequential (cold) data [47–54]. In fact, SLC has better endurance and small random data tend to be updated more frequently. However, MLC is still the limiting factor when long sequential data writes frequently occur to the storage.

From a design perspective, it is quite easy to create a hybrid SSD starting from a conventional SSD, as many typical SLC and MLC chips share the same pin definition and package dimensions.

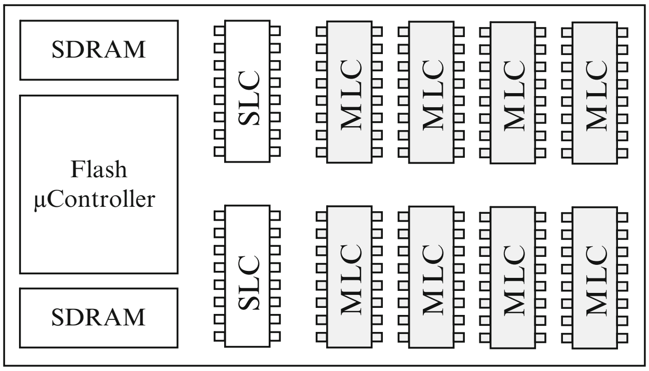


Fig. 4.9 SLC + MLC hybrid SSD

Figure 4.10 shows a possible data flow during write. Every write request enters in the "Data Sensor": cold data directly go to MLC. Hot data move to another block called "Utilization Limiter". If the SLC NAND wears out too fast, this limiter has the task to reduce the write traffic to the SLC Flash. In other words, a second level of data classification is adopted: hot-data go to SLC and quasi-hot-data are switched to MLC.

As mentioned, SLC capacity has to be small; therefore, when data become cold, they should be removed from SLC in order to maximize the space for hot data.

At this point it is clear that the foundation of this approach is the ability of classifying data. A lot of methods to identify hot data have proposed, including LRU, LRU-k [55], hash-table-based approaches [56], and [48, 49]. The reader can refer to this extensive literature for more details.

Li-Pin et al. [49] showed that, by adding a 256 MB SLC Flash to a 20 GB MLC-Flash array, the hybrid SSD improves over a conventional SSD by 4.85 times in terms of average response. The average throughput and energy consumption are improved by 17% and 14%, respectively. The hybrid SSD is only 2% more expensive than a purely MLC-Flash-based SSD.

Of course, the hybrid concept can be extended to a Solid State Drive made up by different types of NAND memories, as shown in Fig. 4.11.

4.6 Over-Provisioning

When looking at the overall capacity of a solid state drive, over-provisioning must be taken into account. Over-provisioning is the difference between the physical capacity of the Flash memory and the logical capacity available for the user. Of course, this is also true for hybrid SSDs [57].

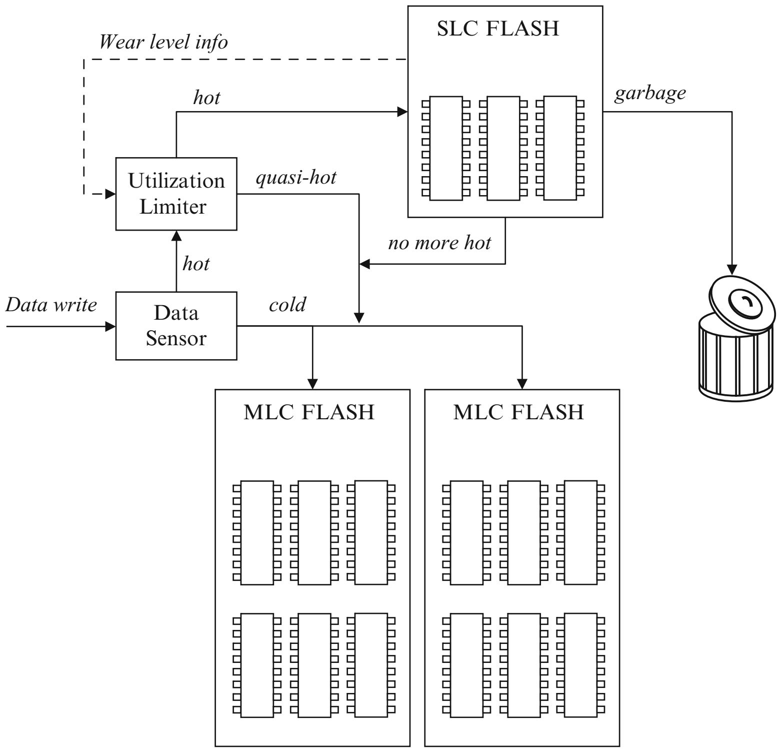


Fig. 4.10 Write flow

The idea behind over-provisioning is to have a "reserve" of spare blocks that can be used by the controller

Let's assume an application that wants to randomly write data to the SSD drive. The drive controller writes these data to some erased pages in a particular block. After a while, the application decides to update the content: given the nature of Flash memories, this would imply erasing the block. In order to improve performances, the drive controller just marks those pages as unavailable and writes the new content to different physical pages: actually, no electrical erase takes place. When the entire block has been used and another write comes in, a real erase operation is needed. At this point, the controller needs to go through the following process:

- copy the entire content of the block to a temporary location (likely cache);
- remove the unused data from the cache;
- add the new data to the block in cache;
- erase the addressed block on the SSD drive;
- copy the entire block from the cache;
- empty the cache.

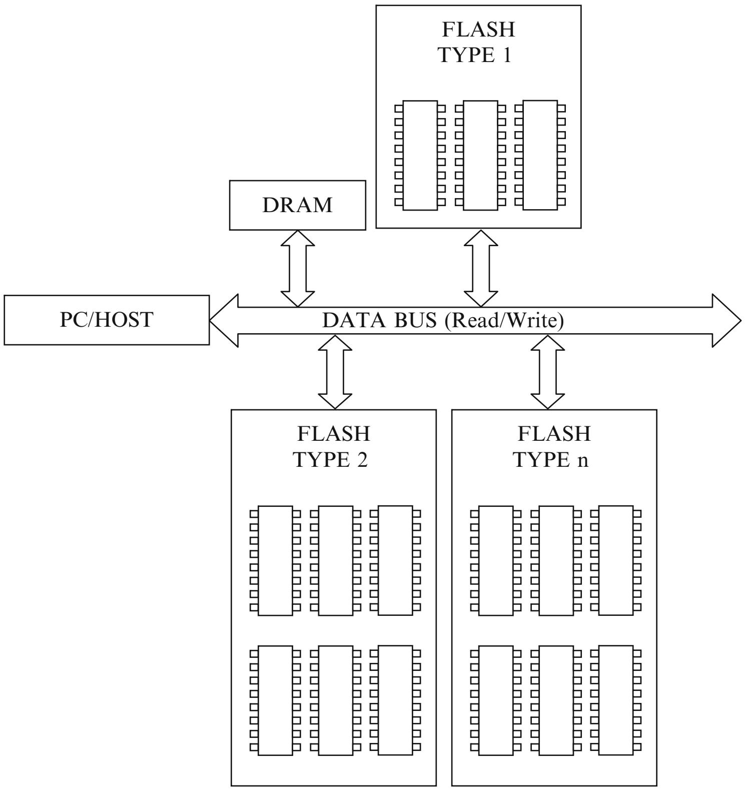


Fig. 4.11 Hybrid SSD including different types of NAND Flash memories

This sequence is very time consuming and kills write throughput performances [57, 58]. When over-provisioning is used, the flow can be different. Instead of having to erase the unavailable portion of the block to accommodate new data, the controller can use some of the spare space instead. This means that the sequence of reading the entire block, merging the new data, erasing the block, and writing the entire new block back, can be avoided. The controller just maps spare space to be part of the drive capacity (so it is seen by the OS) and moves the unused pages to the spare capacity portion of the drive.

Anyhow, at some point the unavailable pages will have to be erased forcing the erase/write sequence mentioned above. In real world applications, 100% random writes are unlikely and the Flash controller does the erase/write sequence in background or when the drive is not in use. To get to the worst case, the host has to randomly write across all the drive's capacity without stopping to read.

Some controllers may not actively defragment the space to save costs, so the worst case performance becomes typical after the drive has been written few times.

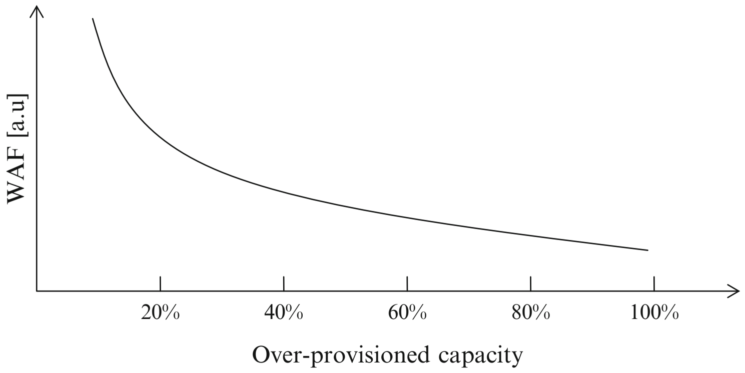


Fig. 4.12 Write Amplification Factor (*WAF*) vs. over-provisioned capacity

Spare capacity can also be used when "bad" areas develop in the drive. For example, if a certain set of pages/blocks has much fewer remaining erase/write cycles than most of the drive, then the controller can remap them to spare pages/blocks. Moreover, the controller can watch for bad writes and use the spare capacity as a "backup" (similar to extra blocks on hard drives). The controller can check for bad writes by doing read-after-write (reads are much faster than writes).

During the garbage collection, wear-leveling, and bad block mapping operations inside the SSD, the additional space from over-provisioning helps lowering the *Write Amplification Factor* (WAF) [58–61]; this factor corresponds to the additional writes caused by garbage collection (see flow above) and wear leveling (Chap. 9). Jedec defines WAF as the data written to the Flash divided by data written by the host to the SSD [62].

Figure 4.12 sketches a typical behavior of WAF vs. over-provisioned capacity. In commercial products over-provisioned capacity is usually around 30%. On one side, with a very small over-provisioning percent, the amount of data "moves" that have to take place can be very high, lowering the achievable write IOPS. On the other side, still looking at Fig. 4.12, 30% looks a good trade-off between performances and area (cost): in fact, beyond 30% WAF reduces at a lower rate [58].

In summary, reducing the amount of over-provisioned capacity can lower the cost per GigaByte, but then WAF can become a real problem. Please bear in mind that the over-provisioned space shrinks over time as it is also intended to countermeasure wear out of Flash blocks.

References

1. The DRAM story, with articles by Dennard, Itoh, Koyanagi, Sunami, Foss and Isaac. IEEE SSCS. News. **13**(1) (Winter 2008) www.ieee.org/sscs-news
2. Debasis Baral *Life Cycle Power Consumption HDD Vs. SSD*, Flash Memory Summit, Session 101 (Storage Labs Samsung Information Systems America, San Jose, 2009)

3. V. Kasavajhala, *Solid State Drive vs. Hard Disk Drive Price and Performance Study* (Dell Technical White Paper, Dell Power Vault Storage Systems, May 2011), http://www.dell.com/downloads/global/products/pvaul/en/ssd_vs_hdd_price_and_performance_study.pdf
4. B. Marsh, F. Douglis, P. Krishnan, Flash memory file caching for mobile computers, in *Proceedings of the Twenty-Seventh Hawaii International Conference on System Sciences*, Wailea, HI, 1994, pp. 451–460
5. T. Bisson, S.A. Brandt, D.D.E. Long, NVCache: Increasing the effectiveness of disk spin-down algorithms with caching, in *MASCOTS 2006*, Monterey, 2006, pp. 422–432
6. T. Bission, S. Brandt, Reducing energy consumption with a non-volatile storage cache, in *Proceedings of International Workshop on Software Support for Portable Storage*, San Francisco, CA, 2005
7. F. Chen, S. Jiang, X. Zhang, SmartSaver: Turning flash drive into a disk energy saver for mobile computers, in *Proceedings of the 2006 International Symposium on Low Power Electronics and Design*, Tegernsee, Germany, 2006, pp. 412–417
8. R. Panabaker, Hybrid Hard Disk And ReadyDrive™ Technology, Improving performance and power for windows vista mobile PCs, in *Proceedings of MicrosoftWinHEC*, Los Angeles, CA, 2006
9. Y.-J. Kim, K.-T. Kwon, J. Kim, Energy-efficient file placement techniques for heterogeneous mobile storage systems, in *Proceedings of the 6th ACM & IEEE International Conference on Embedded software*, Seoul, Korea, 2006, pp. 171–177
10. T. Kgil, T. Mudge, FlashCache: a NAND flash memoryfile cache for low power web servers, in *Proceedings of the International Conference on Compilers, Architecture and Synthesis for Embedded Systems*, Seoul, Korea, 2006
11. T. Kgil, D. Roberts, T. Mudge, Improving NAND flash based disk caches, in *ISCA'08 Proceedings of the 35th Annual International Symposium on Computer Architecture*, Beijing, China
12. S. Liu, X. Cheng, X. Guan, D. Tong, *Energy Efficient Management Scheme for Heterogeneous Secondary Storage System in Mobile Computers SAC'10*, Sierre, Switzerland, 22–26 Mar 2010
13. A. Kirshenbaum et al., Using external memory devices to improve system performance, U.S. Patent No. 7,805,571 and U.S Patent application No. 20100217929, Assignee: Microsoft Corporation
14. Microsoft Windows, Windows 7 features – ReadyBoost – Microsoft Windows, http://windows.microsoft.com/en-US/windows7/products/features/readyboost
15. W.R. Stanek, *Windows 7: The Definitive Guide* (O'Reilly Media, 2010), Sebastopol, CA 95472, pp. 105–109
16. White Paper Intel® Flash Memory Intel® NAND Flash Memory for Intel® Turbo Memory (2007), http://download.intel.com/design/flash/nand/turbomemory/whitepaper.pdf
17. Intel® Turbo Memory – Overview and Support, http://www.intel.com/cd/channel/reseller/apac/eng/products/mobile/mprod/turbo_memory/396715.htm
18. T. Coughlin, J. Handy, *Two May Be Better Than One: Why Hard Disk Drives and Flash Belong Together* (White Paper SNIA, Feb 2011), http://www.snia.org/sites/default/files/Storage%20Pairing%20WP%20FEB%202011.pdf
19. T. Coughlin, J. Handy, *HDDs and Flash Memory: A Marriage of Convenience* (SNIA, Feb 2011), http://www.snia.org/sites/default/files2/SDC2011/presentations/Monday/TomCoughlin_and_Handy_HDDS_Flash_Memory.pdf
20. Seagate MomentusXT Datasheet, http://www.seagate.com/files/staticfiles/docs/pdf/datasheet/disc/momentus-xt-data-sheet-ds1704-4-1205-us.pdf
21. Seagate MomentusXT: Overview features and specs, http://www.seagate.com/internal-hard-drives/laptop-hard-drives/momentus-xt-hybrid/
22. R. Panabaker, Hybrid Hard Disk and ReadyDrive™ Technology: improving performance and power for windows vista mobile PCs, in *Proceedings of Microsoft WinHEC* (2006)
23. E. Pinheiro, R. Bianchini, Energy conservation techniques for disk array-based servers, in *Proceedings of the 18th International Conference on Supercomputing (ICS'04)*, June 2004
24. D. Colarelli, D. Grunwald, Massive arrays of idle disks for storage archives, in *Proceedings of the 2002 ACM/IEEE Conference on Supercomputing*, Baltimore, MD, 2002, pp.1–11

25. G. Symons, *Hybrid SSD/HDD Storage: A New Tier?*, Flash Memory Summit (Xiotech Corporation, Colorado Springs, 2011)
26. Intel® RAID SSD Cache 2.0, http://www.intelraid.com/uploads/Intel_RAID_SSD_Cache2_PB_080911.pdf
27. Intel® Solid-State Drive 313 Series, http://www.intel.com/content/www/us/en/solid-state-drives/solid-state-drives-313-series.html
28. Adaptec maxCache 2.0 Series, http://www.adaptec.com/en-us/_common/maxcache/
29. OCZ Synapse Cache SATA III 2.5″ SSD http://www.ocztechnology.com/ocz-synapse-cache-sata-iii-2-5-ssd.html
30. Corsair Accelerator Series SSD Cache, http://www.corsair.com/ssd/accelerator-series-ssd-cache-drives.html
31. LSI Nytro MegaRAID Application, http://www.lsi.com/products/storagecomponents/Pages/NytroMegaRaid.aspx.
32. Crucial Adrenaline Solid State Cache (Windows 7 PCs), http://www.crucial.com/store/ssc.aspx
33. R. Micheloni, L. Crippa, A. Marelli, *Inside NAND Flash Memories* (Springer, New York, 2010)
34. N. Duann, *SLC & MLC Hybrid*, Flash Memory Summit (Silicon Motion, Inc., 2011)
35. B. Chang, *SSD with Hybrid NAND* Novachips, Flash Memory Summit, 2011
36. Y. Koh, *NAND Flash Scaling beyond 20 nm,* Memory Workshop, in *IMW '09,* IEEE International, 2009
37. White paper, *Engineering MLC Flash-Based SSDs to Reduce Total Cost of Ownership in Enterprise SSD Deployments,* STEC's CellCare™ Technology, http://www.stec-inc.com/downloads/MLC_flash_based_SSDs_Reduce_TCO.pdf
38. C.C. Wu, Quality comparison of SLC, MLC and eMLC., in *InnoDisk International Memory Workshop IMW*, San Diego, CA, 2011
39. E. Bek, A. Klein, *The Future of SSD Architectures*, *International Memory Workshop IMW*, SanDisk, 2011
40. W.H. Radke et al., Hybrid memory management, U.S. Patent No. 8,060,719, Assigned: Micron Technology, Inc., 28 May 2008
41. C. Lee et al., Hybrid SSD using a combination of SLC and MLC flash memory arrays, U.S. Patent No. 8078794, Assignee: Super Talent Electronics, Inc., San Jose, 29 Oct 2007
42. Y.S. Kim, Semiconductor memory device, and multi-chip package and method of operating the same, U.S. Patent No. 8085569, Assignee: Hynix Semiconductor Inc., 14 Dec 2010
43. H. Tan et al., Portable data storage using SLC and MLC flash memory, U.S. Patent App. No. 20080215801, Assignee: Trek 2000 International Ltd., 28 Sept 2005
44. M. Moshayedi, Enhanced MLC solid state device, U.S. Patent App. No. 20090327590, Assignee: STEC, Inc., 24 June 2009
45. M. Moshayedi, SLC-MLC combination flash storage device, U.S. Patent App. No. 20090327591, Assignee: STEC, INC., 24 June 2009
46. L.E. Aszmann et al., Solid state drive data storage system and method, U.S. Patent App. No. 20110010488 (12 Jul 2009)
47. T.-W. Kuo et al., Configurability of performance and overheads in Flash Management, in *11th Asia and South Pacific Design Automation Conference (ASP-DAC)* (2006)
48. L.-P. Chang, Hybrid solid-state disks: Combining heterogeneous NAND flash in large SSDs, in *13th IEEE/ACM Asia and South Pacific Design Automation Conference (ASP-DAC)*, (2008)
49. L.-P. Chang, A hybrid approach to NAND-flash-based solid-state disks. IEEE Trans. Comput. **59**(10), 1337–1349 (Oct 2010)
50. L.-P. Chang, Y.-C. Su, Plugging versus logging: A new approach to write buffer management for solid-state disks, in *The 48-th Design Automation Conference (DAC)*, Monterey, CA, 2011
51. S. Hong, D. Shin, NAND flash-based disk cache using SLC/MLC combined flash memory, in *2010 International Workshop on Storage Network Architecture and Parallel I/Os*.
52. S. Jung, Y.H. Song, Hierarchical use of heterogeneous flash memories for high performance and durability. IEEE Trans. Consum. Electron. **55**(3), 1383–1391 (Aug 2009)

53. M. Murugan and D.H.C. Du, Hybrot: Towards Improved performance in hybrid SLC-MLC devices, *20th IEEE International Symposium on Modeling, Analysis and Simulation of Computer and Telecommunication Systems* (MASCOTS) (Short Paper), (Aug 2012)
54. B.-W. Nam, A hybrid flash memory SSD Scheme for Enterprise Database applications, in *12th International Asia-Pacific Web Conference*, Busan, Korea, 2010
55. E.J. O'Neil, P.E. O'Neil, G. Weikum, The LRU-k page replacement algorithm for database disk buffering. ACM SIGMOD Rec. **22**(2), 297–306 (1993)
56. J.W. Hsieh, T.W. Kuo, L.P. Chang, Efficient identification of hot data for flash memory storage systems. ACM Trans. Storage **2**(1), 22–40 (2006)
57. D.A. Heger, *SSD Write Performance – IOPS Confusion Due to Poor Benchmarking Techniques* (Aug 2011), http://www.cmg.org/measureit/issues/mit82/m_82_4.pdf
58. X.-Y. Hu, Write amplification analysis in Flash-based solid state drives, in *SYSTOR'09*, IBM Zurich Research Laboratory, Haifa, Israel
59. K. Smith, *Benchmarking SSDs: The Devil is in the Preconditioning Details*, Flash Memory Summit (2009)
60. White Paper, Intel *High-Performance SATA Solid-State Drive: Over-Provisioning an Intel SSD*, http://www.matrix44.net/cms/wp-content/uploads/2011/07/intel_over_provisioning.pdf
61. T. Frankie, *SSD Trim Commands Considerably Improve Overprovisioning*, Flash Memory Summit (2011)
62. JEDEC STANDARD, Solid-State Drive (SSD) Requirements and Endurance Test Method, JESD218, (Sept 2010), http://www.jedec.org/sites/default/files/docs/JESD218A.pdf

Chapter 5
NAND Flash Technology

M.F. Beug

Abstract This chapter describes the basic operating principle and presents the major reliability and scaling limitations of floating gate NAND non-volatile memory as used in SSD applications. It further discusses charge trapping memory cells as a potential replacement for floating gate cells in the NAND array and evaluates the potential of both memory cell principles in future 3D memory approaches.

5.1 Flash for SSD Application

Flash memory for non-volatile data storage was introduced commercially in the mid-1980s. Since then, common ground NOR and NAND architecture have become the most common memory array architectures. Traditionally, NOR Flash is used for code storage due to faster memory cell access. NAND Flash is used for mass data storage as a result of its higher memory density, enabling higher storage capacities.

The memory cell area difference can already be seen from the schematic NOR and NAND array images in Fig. 5.1. In the NOR array, two memory cells each share one contact to ground and one contact to the bit line (see Fig. 5.1a). This results in an effective memory cell area of about 10 F^2 (where F is the minimum feature size). The effective memory cell area of NAND cells is only slightly more than 4 F^2. Figure 5.1b shows the so-called NAND string with up to 64 memory cells connected in a row. To operate the NAND string two additional select transistor devices (GSL: "Ground Select Line" and SSL: "String Select Line") and contacts to ground (SL: "Source Line") and the bit line (BL) need to be added. These additional structures cause the effective cell area consumption to be slightly higher than 4 F^2

M.F. Beug (✉)
Physikalisch-Technische Bundesanstalt (PTB), Division 2 "Electricity", Bundesallee 100, 38116 Braunschweig, Germany
e-mail: Florian.Beug@ptb.de

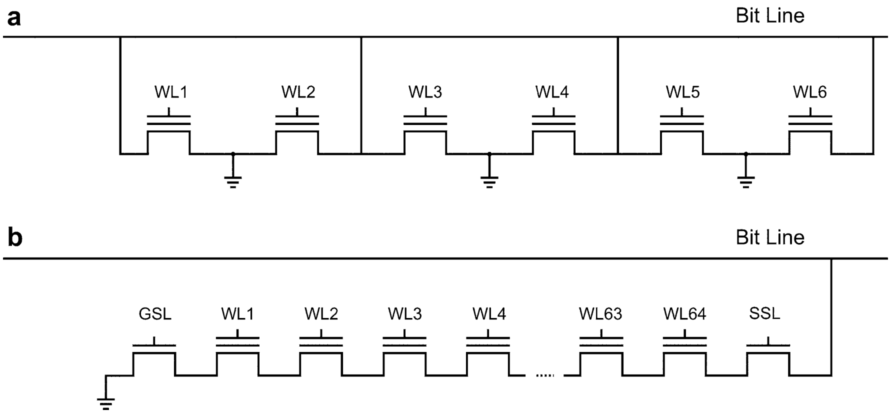


Fig. 5.1 Schematic memory cell organization of the NOR array (**a**) and the NAND array (**b**). The word lines (*WL*) run perpendicular to the bit lines (*BL*)

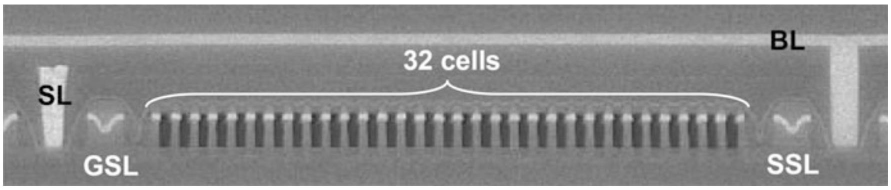


Fig. 5.2 SEM picture of a NAND string with 32 cells per string in a 48 nm floating gate NAND technology [2]

- the theoretically smallest effective cell size. The cross section of a 48 nm NAND technology with 32 cells per string is shown in Fig. 5.2.

For SSD application, only NAND Flash is a viable option due to the required high memory capacity and bit cost structure. Therefore, the following sections will focus on operation, reliability, and scaling topics of NAND Flash.

5.2 Introduction to Floating Gate NAND Operation

A floating gate memory cell stores information in terms of charge in an isolated gate electrode (floating gate: FG). The FG is located between the memory transistor channel and the active gate electrode (control gate: CG). This data storage principle was proposed by Kang and Sze in 1967 [1] and enables data to be stored without the connection of a supply voltage over time periods of several years.

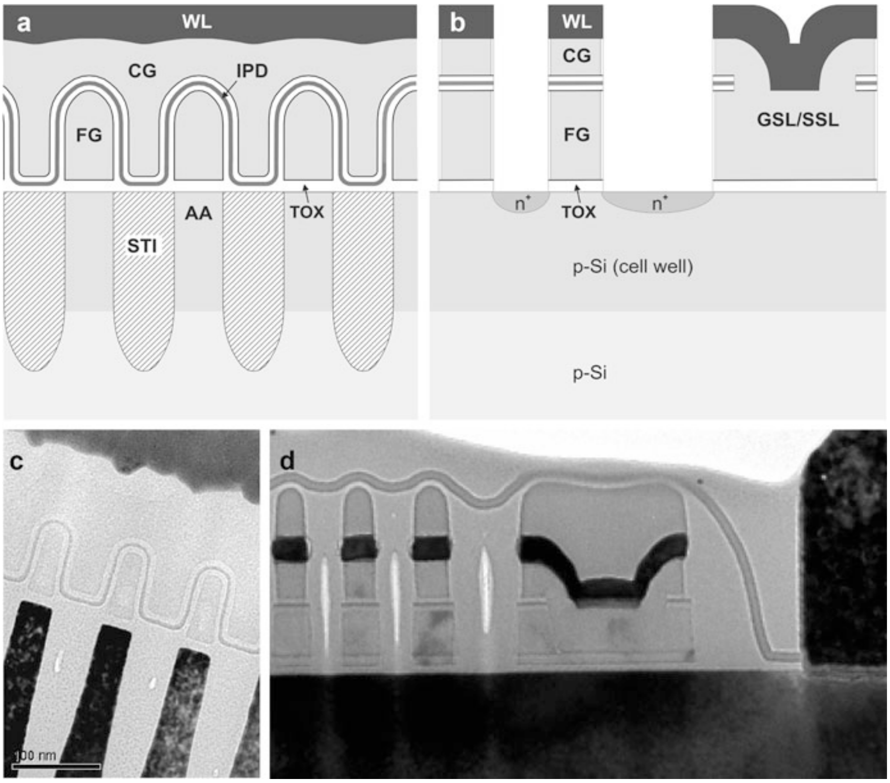


Fig. 5.3 Schematic structure of a floating gate NAND array in word line (*WL*) (**a**) and bit line (*BL*) direction (**b**). Corresponding TEM pictures of a 48 nm floating gate NAND technology [2] in WL direction (**c**) and BL direction (**d**)

5.2.1 The Floating Gate NAND Memory Structure

The schematic structure of floating gate NAND cells is shown in Fig. 5.3a, b. Figure 5.3c, d shows the cross sections of a 48 nm floating gate NAND technology [2]. The FG and the CG are typically made of polysilicon. For all operations of the floating gate cell, the active control gate electrode capacitive couples to the floating gate. The dielectric between the FG and the CG is referred to as inter-poly dielectric (IPD) and is typically made of a silicon oxide/silicon nitride/silicon oxide triple layer (ONO). The alterable threshold voltage of a floating gate cell, which represents the bit information, consequently depends on the coupling strength between the FG and the CG, and the amount of charge on the FG.

The FG NAND structure in word line direction is shown in Fig. 5.3a, c.

The CG is wrapped around the FG to improve the capacitive coupling from the CG to the FG. This reduces the operating voltages of the floating gate cells and ensures a reliable operation as will be described in the next section. The active areas

(AA) of two neighboring NAND strings are separated by shallow trench insulation (STI) and are about 200 nm deep in current generations. The memory cell transistor gate oxide is denoted as tunnel oxide (TOX) because the charge for bit information storage is transferred through this SiO_2 dielectric by quantum mechanical tunneling.

Generally, it is a very crucial point for reliable floating gate cell operation that charge during program and erase operations is only transferred through the TOX. Every charge transfer through the IPD (between FG and CG) needs to be urgently avoided to prevent severe reliability issues.

In BL direction, the cell strings run as shown in Fig. 5.1a, c and d. The floating gate cells are patterned by a vertical WL etch step. In the etched spaces between the floating gate cells, shallow n^+ junctions are implanted in order to define the memory cell transistors and reduce the string resistance. To improve the charge retention of the memory cells, the side wall of the floating gate is passivated by a thermal oxidation process.

The generated high quality thermal side wall oxide (SWOX) forms an effective tunnel barrier against charge loss from the FG. Subsequently, the space between the FG cells is filled with a deposited silicon oxide (inter-word line dielectric: IWD) which generally has a reduced electrical quality. The select devices (GSL and SSL) are processed together with the floating gate cells and consequently use the TOX as the gate dielectric. The select transistor gate length is typically in the range of 150–200 nm. To obtain a real transistor for the select devices, the word line layer is connected to the floating gate layer. This contact is made by removing the ONO IPD in the middle of the select transistors prior to the CG poly-Si deposition (see Fig. 5.3d).

The complete process of a floating gate NAND technology is typically based on 30–40 lithographic mask steps and includes 2 poly-Si and 3 metal levels. To obtain the highest memory density in each technology generation, typically 3 levels are structured in the most advanced technology node. The levels of advanced feature size are active area/STI, word line and bit line. The bit line is either done in the first or second metal layer. There are some more process steps with stringent lithographic requirements, such as the contacts to the bit line, but also the source contacts, the CG to FG contacts in the select devices, and others.

5.2.2 *The Floating Gate Cell Capacitive Coupling Model*

It was described that floating gate NAND cells are arranged in strings with up to 64 memory cells in actual NAND technologies. However for the basic understanding of the floating gate cell functionality it is necessary to look at a single FG cell first.

Since the floating gate is isolated from the active control gate, all voltages for operation of the memory cell need to be capacitively coupled to the floating gate. In principle, the floating gate cell forms a capacitive voltage divider which is typically described with the aid of the FG cell capacitive coupling model [3] as shown in Fig. 5.4.

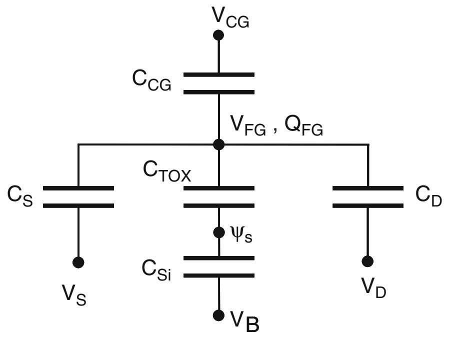


Fig. 5.4 Capacitance model of a floating gate memory device

It describes the voltage of the floating gate as a function of the other terminals of a FG cell. These terminals are typically source (V_S), drain (V_D), the bulk terminal (V_B), the control gate (V_{CG}), and a number of other (parasitic) terminals. All these terminal voltages are capacitive coupled to the floating gate. The floating gate voltage can be written as

$$V_{FG} = \alpha_G \cdot V_{CG} + \alpha_S \cdot V_S + \alpha_D \cdot V_D \cdot \frac{C_{TOX}}{C_T} \cdot \psi_S + \frac{Q_{FG}}{C_T} + \sum \alpha_{other} \cdot V_{other}. \tag{5.1}$$

The gate coupling ratio α_G in Eq. (5.1) is an important factor and is defined as

$$\alpha_G = \frac{C_{CG}}{C_T}. \tag{5.2}$$

C_T is the total capacitance and is given by

$$C_T = C_{CG} + C_{TOX} + C_S + C_D + \sum C_{other}. \tag{5.3}$$

The sum of C_{other} contains all other terminals which couple to a specific floating gate and represent neighboring bit and word lines or neighboring floating gates. The capacitive components in the sum are traditionally small compared to the other terms, but gain significantly in importance when floating gate cells are scaled to feature sizes below 50 nm [4].

The gate coupling ratio α_G describes the portion of the voltage applied between the CG and the channel that drops across the TOX. For grounded source, drain, bulk, and other terminals during program operation, the floating gate voltage is given by

$$V_{FG} = \alpha_G \cdot V_{CG}. \tag{5.4}$$

A control gate voltage $V_{CG} = 20$ V in combination with a gate coupling ratio of $\alpha_G = 0.6$ results in a voltage drop of $V_{FG} = 12$ V across the tunnel oxide. Consequently, the CG voltage is concentrated on the tunnel oxide, when a high C_{CG} to C_T ratio and therefore a high α_G can be realized.

Under such coupling conditions, the requested floating gate cell operation can be obtained, where charge is only transferred between the channel region and the floating gate.

The FG voltage formulation Eq. (5.1) and α_G formula in Eq. (5.2) were described in [5] and only take into account the voltage drop across the tunnel dielectric (across C_{OX}). It does not consider the voltage drop in the Si substrate (across C_{Si}) [6].

The source and drain coupling ratios have the same form as the α_G expression Eq. (5.2) and are given by $\alpha_S = C_S/C_T$ and $\alpha_D = C_D/C_T$.

The capacitive coupling model and Eq. (5.1) also yield the formula for the floating gate cell threshold voltage shift ΔV_{th} caused by charge stored on the floating gate. The threshold voltage shift is in principle the voltage increase which is necessary at the control gate to compensate the floating gate charge induced field effect. Therefore, it is the additional CG voltage for resuming the floating gate voltage that would be present without the FG charge and results in a defined TOX field which is necessary to invert the memory cell channel. For constant potentials at source and drain during the read operation, Eq. (5.1) can be rearranged to

$$\Delta V_{th} = \Delta V_{CG}|_{\Delta V_{FG}=0} = -\frac{\Delta Q_{FG}}{\alpha_G \cdot C_T} = -\frac{\Delta Q_{FG}}{C_{CG}}. \tag{5.5}$$

This means that for an optimized high gate coupling ratio value and a given threshold voltage shift, the number of stored electrons is increased (which is beneficial for charge retention). The required high C_{CG} value can be either obtained by a large coupling area between the CG and the FG, (the previously described CG wrapped around the FG), or a reduction in the electrical IPD thickness.

The effect of the latter option on the ability to program and erase floating gate cells will be discussed in the following section.

5.2.3 Program and Erase of a Single Floating Gate Cell

Floating gate cells in NAND applications are programmed and erased by the Fowler-Nordheim (FN) tunneling mechanism [7]. This quantum mechanical tunneling mechanism is based on a strong electric field across the tunneling barrier of the TOX. The electric field across the typically 8 nm thick tunnel oxide causes a band distortion. The induced FN tunneling current has a strong electric tunnel oxide field (E_{TOX}) dependency. The FN current density changes over several orders of magnitude and is the result of a significant reduction in the effective tunneling distance x_t, as shown in Fig. 5.5 and its inset.

The Fowler-Nordheim tunneling current density is given by

$$J_{FN} = A_t \cdot E_{ox}^2 \cdot \exp\left(-\frac{B_t}{E_{ox}}\right), \tag{5.6}$$

with the two tunneling constants A_t and B_t which are given by

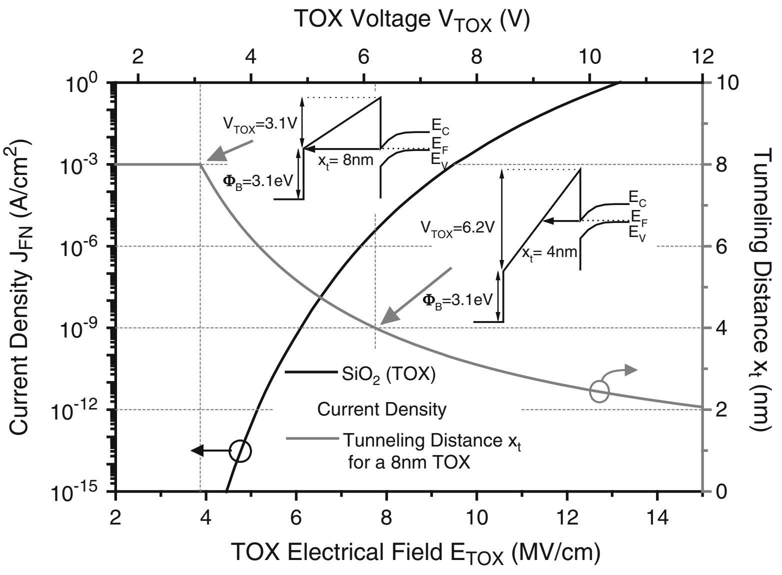


Fig. 5.5 Fowler-Nordheim tunneling current density and effective tunneling distance x_t for a 8 nm tunnel oxide (*TOX*)

$$A_t = \frac{q^3 m_e}{8\pi h m^* \Phi_B}; \quad B_t = \frac{8\pi\sqrt{2m^*\Phi_B^3}}{3qh}. \tag{5.7}$$

In Eq. (5.7), q is the electron charge, m_e and m* the mass of the electron and the effective electron mass in the SiO_2, h is Planck's quantum and Φ_B the tunnel barrier height between Si and SiO_2. The Fowler-Nordheim tunneling current density for a 8 nm thick $Si0_2$ tunnel dielectric with an exponential dependence on the electric oxide field E_{ox} is shown in Fig. 5.5.

Significant amounts of charge are transferred during a program pulse typically shorter than 1 ms, where the TOX electric field is in the strong Fowler-Nordheim tunneling regime above 10 MV/cm. Such strong oxide fields reduce the effective tunnel distance x_t of the triangular barrier to values below 3 nm as shown in Fig. 5.5.

When a floating gate cell is intended to be programmed to a certain V_{th} state, this is typically accomplished by the so-called "incremental step pulse programming" (ISPP) scheme [8]. To reach a targeted cell threshold voltage, programming pulses with durations in the range of $t_{pp} = 100$ μs are applied with increasing pulse amplitude. Each programming step is followed by a sense operation to evaluate whether the target V_{th} has already been reached. The increment of program pulse voltage steps depends on the required accuracy of the programmed V_{th} value. Therefore, the program step voltage directly affects the cell V_{th} distribution width in a memory array with large numbers of cells [9].

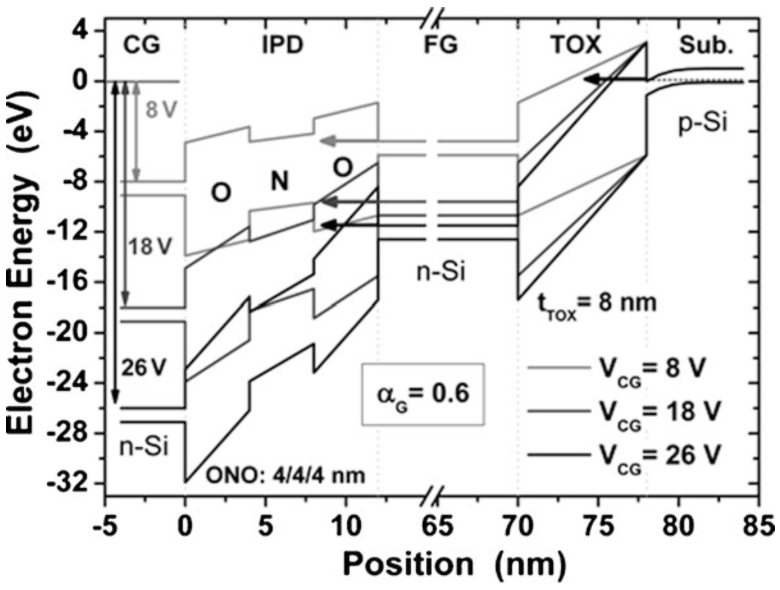


Fig. 5.6 Band diagram of a floating gate cell with $t_{TOX} = 8$ nm, an ONO IPD of 4/4/4 nm and a gate coupling ratio $\alpha_G = 0.6$ for the program voltages $V_{CG} = 8$ V, $V_{CG} = 18$ V, and $V_{CG} = 26$ V after the program charge transfer, if applicable (compare Fig. 5.7). For $V_{CG} = 8$ V, the tunnel oxide field E_{TOX} is too low for electron injection through the TOX. For $V_{CG} = 18$ V, charge is injected into the FG until E_{TOX} is reduced to 12 MV/cm (shown here), the threshold program field. For $V_{CG} = 26$ V in the assumed simplified model, the FG charge increases until the electric fields in the TOX and the IPD suboxide equal each other. The FG charge remains constant in principle, but a strong tunneling current continuously passes through the hole FG stack and would in reality cause significant damage

For a relatively low programming voltage of only $V_{CG} = 8$ V at the beginning of the ISPP sequence, this voltage is divided between the tunnel oxide and the IPD according to the gate coupling ratio α_G. The band diagram of a floating gate cell for such a small voltage is shown in Fig. 5.6. However, for the assumed values $\alpha_G = 0.6$ and IPD layer thicknesses of O/N/O = 4 nm/4 nm/4 nm, no significant amount of charge is transferred to the floating gate, since the TOX field is only 6 MV/cm (see Fig. 5.7). The assumed ONO layer thicknesses of 4 nm for each layer are already very small values as similarly used in state-of-the-art floating gate NAND Flash technologies in the range of 25 nm [10, 11]. Due to the exponential field dependency of Fowler-Nordheim tunneling, programming starts at a certain program threshold voltage which is equivalent to a fixed threshold electric TOX field. For the threshold field conditions a significant amount of charge can be injected into the FG within the short program pulse time of typically $t_{pp} = 100$ μs. A typical value for the program start or threshold field is in the range of 12–13 MV/cm and depends on the process of the tunnel oxide formation which can influence the oxide barrier height. In addition, factors like the TOX thickness profile and the STI edge shape can affect

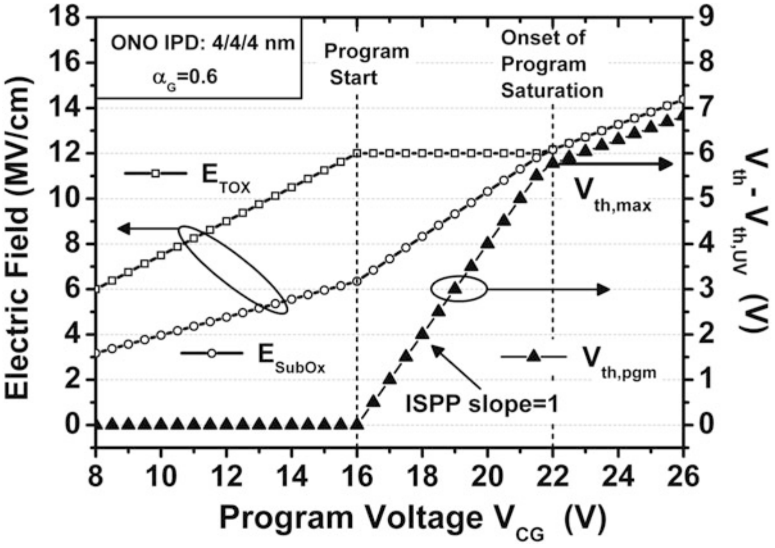


Fig. 5.7 Electric field condition in the tunnel oxide (E_{TOX}) and the IPD suboxide (E_{SubOx}) during ISPP programming of a floating gate cell with $t_{TOX} = 8$ nm, $\alpha G = 0.6$, and ONO IPD layer thicknesses of 4 nm each for the suboxide, the silicon nitride and the top oxide. Programming with an ideal ISPP slope = 1 takes place until E_{SubOx} at the end of programming equals the TOX electric threshold field of 12 MV/cm

this value. Due to this programming threshold field (which will be assumed to be 12 MV/cm in the following), it can be assumed that the same field strength will be present at the end of programming. This assumption is realistic because at a constant programming voltage, negative charge (electrons) is transferred to the floating gate as long as the additional charge has reduced the electric TOX field (Eq. (5.1)) to such an extent, that no more significant charge transfer can take place.

For the described exemplary FG cell configuration used for Fig. 5.6 and Fig. 5.7, programming with no significant IPD current takes place in the CG voltage range between $V_{CG} = 16$ V and $V_{CG} = 22$ V. The ISPP slope in this V_{CG} range is essentially at unity [2]. At around $V_{CG} = 22$ V and beyond this CG voltage value it can be observed that the TOX and the IPD suboxide electric fields equal each other. This results in an electron tunneling to the FG and at the same time an electron tunneling out of the FG towards the CG. For an IPD purely consisting of SiO_2, the same fields in TOX and IPD would result in the same currents tunneling into and out of the floating gate, which results in program saturation.

For an ONO IPD with additional SiN layer, charge can be injected into the SiN layer and will be stored in this layer as in a charge trapping memory cell storage layer. The charge injected and trapped in the ONO increases the effective barrier height [12] (compare Fig. 5.16b) and is therefore able to block weak and leaky spots of the ONO IPD by this means. This is one reason why an ONO IPD is generally used.

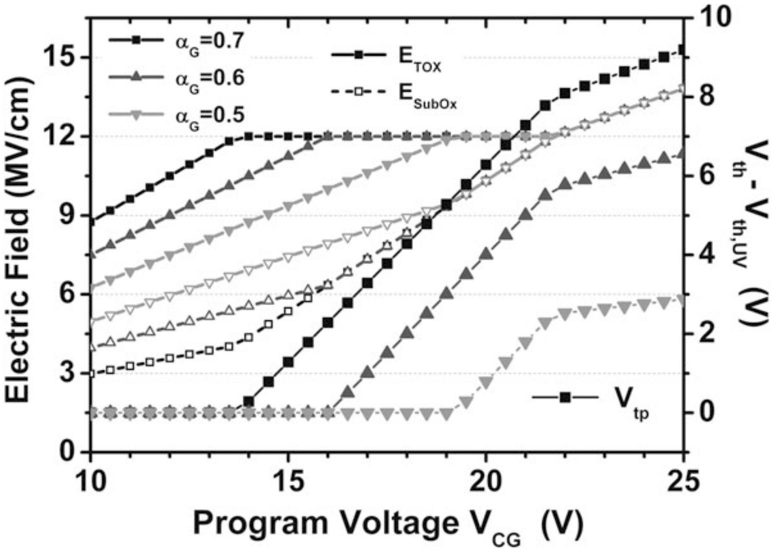


Fig. 5.8 Effect of FG cell geometrically increased gate coupling ratio α_G on program saturation. The TOX thickness $t_{TOX} = 8$ nm and the ONO layer thicknesses ($t_{SubOx/tSiN/tTopOx} = 4/4/4$ nm) are unchanged

However, the electrons injected and finally stored in the ONO IPD beyond the program saturation starting point cause a permanent FG memory cell threshold voltage shift [10]. In addition to the stored charges, a large current is transferred through the whole FG cell stack from the channel towards the control gate which will substantially damage the memory cell. These large permanent currents become clear when looking at the strongly reduced TOX and IPD suboxide x_t for $V_{CG} = 26$ V in Fig. 5.6.

By equating the electric fields in the TOX and the IPD suboxide, a simple model for the onset of program saturation can be derived [13].

Finally, an expression for the maximum reachable programmed threshold voltage (program saturation point) can be obtained, which is given by

$$V_{th,max} = 12\ \frac{\mathrm{MV}}{\mathrm{cm}} \cdot \left(t_{TOX} + t_{IPD-EOT} - \frac{t_{TOX}}{\alpha_G} \right) \tag{5.8}$$

It can be seen from Eq. (5.8) that in principle a thick tunnel oxide and a large equivalent oxide thickness of the IPD ($t_{IPD\text{-}EOT}$) are beneficial for good programmability of floating gate cells. Also a large gate coupling ratio improves $V_{th,max}$. However, due to the middle term in Eq. (5.7) the increase of the control gate to floating gate area is preferred over a reduction of $t_{IPD\text{-}EOT}$ to obtain a large α_G.

Figure 5.8 examines the effect of an increased α_G due to cell geometry means while keeping the TOX and IPD thicknesses unchanged.

It can be observed that for increasing the gate coupling ratio the initial (uncharged FG) field difference between the TOX and IPD electric fields increases. Consequently FG cells with a higher gate coupling ratio can be programmed to higher V_{th} levels before program saturation occurs. The program saturation point ($V_{th,max}$) can be found in the V_{th} ISPP curves in Fig. 5.8, where the ISPP slope changes from unity to a value significantly lower than one. ISPP slopes lower than unity [14] generally show that the combination of cell geometry and IPD current blocking ability is not sufficient to avoid an IPD electron tunneling current during program operation.

The floating gate memory cell erase works principally in the same way, but with control gate voltages negative with respect to the cell channel region. Consequently, the electric field direction is reversed and the erase is mainly due to electron tunneling from the floating gate towards the channel. Again, as described for program saturation, the TOX erase field is reduced for decreasing erase cell V_{th} values while the IPD field increases. In practice, erase saturation can in principle also become a problem, e.g. for bi-layer high-k dielectric containing IPD options. However, for NAND FG Flash only one single erase V_{th} distribution needs to be placed in the negative V_{th} range which generally does not require erasing the cells to large negative threshold voltages. For the positive V_{th} range the situation is different, because for a multi-level cell (MLC), four, and for a triple-level cell (TLC), eight different V_{th} distributions need to be placed, which requires at least that a $V_{th} = +4$ V can be programmed.

Consequently, program saturation is usually a more severe issue than erase saturation.

5.2.4 Program, Erase, and Read of FG Cells in the NAND String

When a large number of a floating gate cells need to be operated in the NAND array it has to be taken into account that one floating gate cell is located at every crossing point of bit lines and word lines. Therefore, the memory cells in the NAND array cannot be operated independently of each other anymore. In the word line direction (depending on the page size), a couple of thousand FG cells are controlled by the same word line. In bit line direction, the string size (64–66 cells in latest NAND generations) defines the number of cells that cannot be operated independently. Consequently, it is very important to bear in mind what is happening with all neighboring cells when one cell is treated. This is even more important since the threshold voltage of each memory cell needs to be carefully adjusted as shown for SLC and MLC cells in Fig. 5.9.

The erased V_{th} cell distribution is placed at negative V_{th} values. In an ISPP-like sequence the erase voltage is increased until all cells are erased below the erase verify (EV) level. The programmed V_{th} distributions are placed in the positive V_{th} range. For a single level cell (SLC) the ISPP programming is continued until all cells

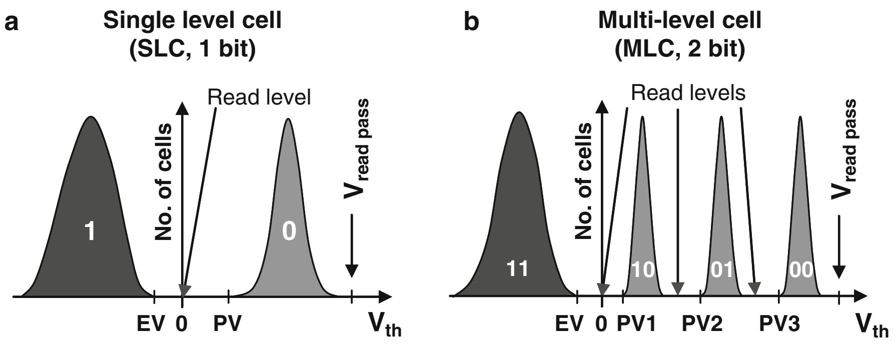


Fig. 5.9 Memory cell threshold voltage distributions for one bit per cell (*SLC*) data storage (**a**) and two bit per cell (*MLC*) data storage (**b**) in a NAND flash array

designated for programming are above the program verify (PV) level. In the case of multi-level cells (MLC), there are consequently three program verify levels (PV1, PV2, and PV3). In addition, it has to be guaranteed that the margins between the different programmed V_{th} distributions are large enough to place the read levels and have sufficient margin for charge/retention loss-caused V_{th} reductions (see Sect. 5.3). To obtain these kinds of narrow cell V_{th} distributions it is necessary to apply a specific distribution shaping algorithm with a small program step increase in certain stages of ISPP programming [9].

5.2.4.1 NAND Cell Programming and Self-Boosted Program Inhibit (SBPI)

Figure 5.10 shows the voltage condition in the NAND array when the FG cell at WL3 in BL2 is programmed. For this purpose, a program pulse with the pulse amplitude of $V_{pp} = 20$ V is applied to WL3. To conduct a successful program it is also required to transfer 0 V to the channel region of the programmed cell as shown in Fig. 5.10i. Consequently, the 0 V potential is applied to BL2 and then needs to be transferred to the whole string including the programmed cell at WL3. This is done by applying the pass voltage (e.g. $V_{pass} = 10$ V) to all other word lines.

In principle, all cells addressed by WL3 could be programmed by this means at the same time. However, the programming of arbitrary information requires that specific memory cells at WL3 are excluded from programming. The cell at the crossing point of BL1 and WL3 represents, in this example, the cells which should be prevented from programming (program-inhibited cell in Fig. 5.10ii). In former FG NAND generations, programming in certain NAND strings was avoided by actively applying a positive voltage to the corresponding bit lines. As a result, the voltage difference between the channel and the control gate was not high enough for programming in these strings. This procedure was complicated and the voltage pumps used for this purpose required additional power and chip area. Therefore, in

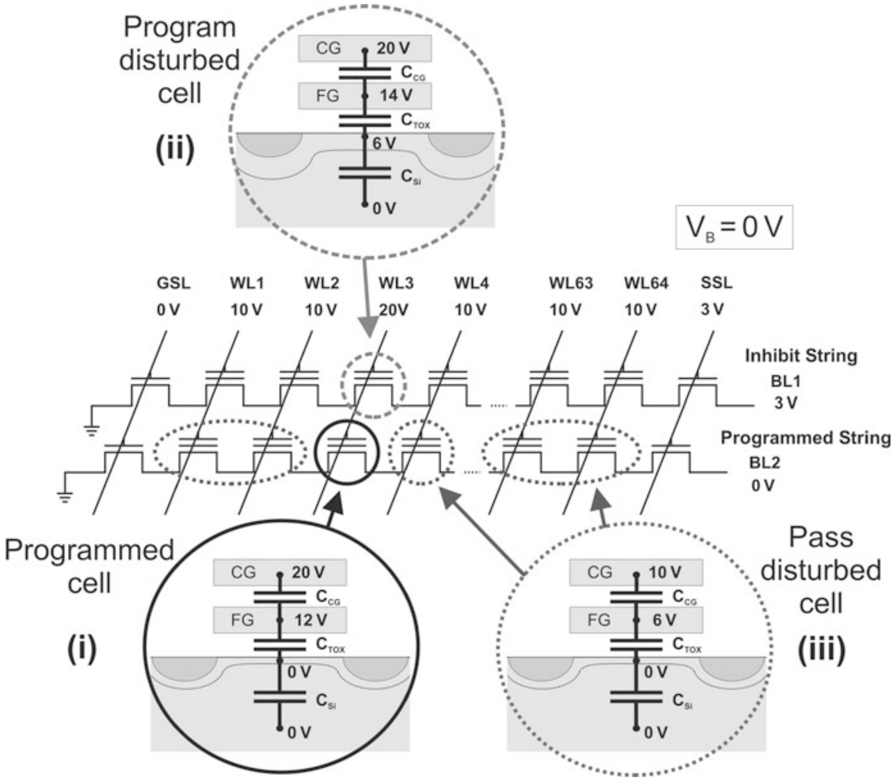


Fig. 5.10 Voltage conditions during program operation in the NAND array. The memory cell at the crossing point of WL3 and BL2 is programmed; several other cells are disturbed by either program disturb or pass disturb

later generations the so-called "Self-Boosted Program Inhibit" (SBPI) scheme was introduced [8]. The principle of the SBPI scheme is that the channel potential in the inhibited strings is not actively raised by applying a voltage, but capacitively raised, as will be seen in the following.

The voltages applied to different word lines, bit lines and select devices in the SBPI sequence are shown in Fig. 5.10. The corresponding detailed timing of the signals at different signal lines is shown in Fig. 5.11. For a successful program inhibit at the programmed word line an inhibit channel potential in the range of typically 6–8 V is required. The exactly required channel potential further depends on the maximal used programming voltages.

In the first step (t_1), V_{CC} (e.g. 3 V) is connected to the SSL and the inhibit strings at the same time (Fig. 5.11a). This results in a pre-charge of the inhibit string to a channel potential of $V_{pre\text{-}ch} = V_{CC} - V_{th,SSL}$ as shown in Fig. 5.11d. During this pre-charge of the string the channel side of the select transistor acts as the source. Accordingly, a charging current flows until the gate-to-source voltage equals the

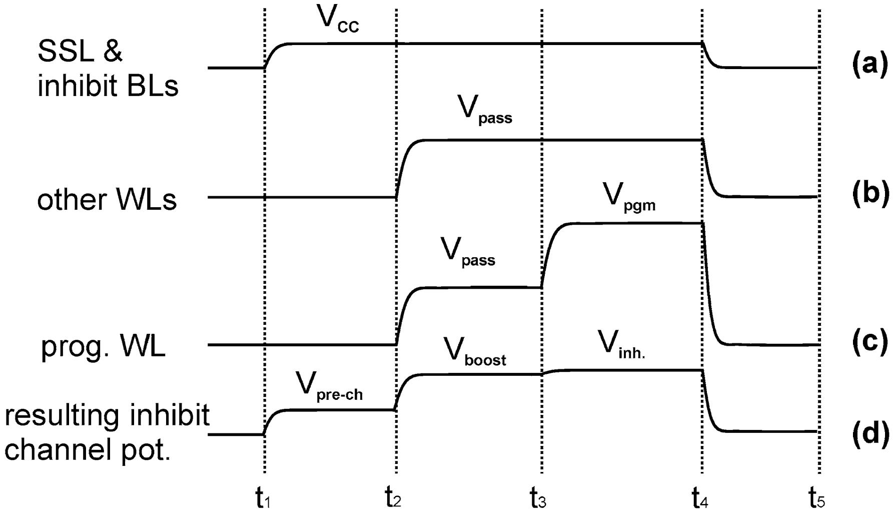


Fig. 5.11 Signal timing for the self-boosted program inhibit (SBPI) scheme

threshold voltage of the select transistor. In the second time step t_2, all word lines are raised to the program pass voltage V_{pass} (Fig. 5.11b, c) and the channel inhibit potential is increased by capacitive coupling. This can be done because the select transistor is closed since the pre-charge was finished. At time t_3 the word line selected for programming (WL3 in Fig. 5.10) is raised to the full program voltage in the ISPP sequence which further increases the channel potential to its full inhibit voltage V_{inh}.

In this last step, only a small channel voltage increase is achieved which results from the CG to channel capacitance ratio of one cell in relation to the whole cell string. Therefore, a larger channel voltage increase can be obtained when not the whole string is boosted, but only a few cells in the vicinity of the programmed word line. Such an approach is called the "local self-boosted program inhibit" (local SBPI) scheme [15, 16].

It is clear that a major part of the inhibit channel potential depends on the pass voltage, since V_{inh} is partly generated by the capacitive channel boosting.

On the one hand, the ability to prevent programming at the "program disturbed cell" (WL3 of BL1 in Fig. 5.10ii) improves with increasing pass voltage V_{pass} as shown in Fig. 5.12. On the other hand, the pass cells located in a string with a memory cell dedicated for programming (BL2) experience a soft programming when the pass voltage is increased beyond a certain limit (pass disturbed cell in Fig. 5.10iii).

The general effect of a pass voltage variation on a program disturbed and a pass disturbed cell in a 48 nm FG NAND technology is shown in Fig. 5.12. Since both effects, program and pass disturb, result in a threshold voltage increase and are more severe on erased cells, the memory cells in Fig. 5.12 were first erased to a

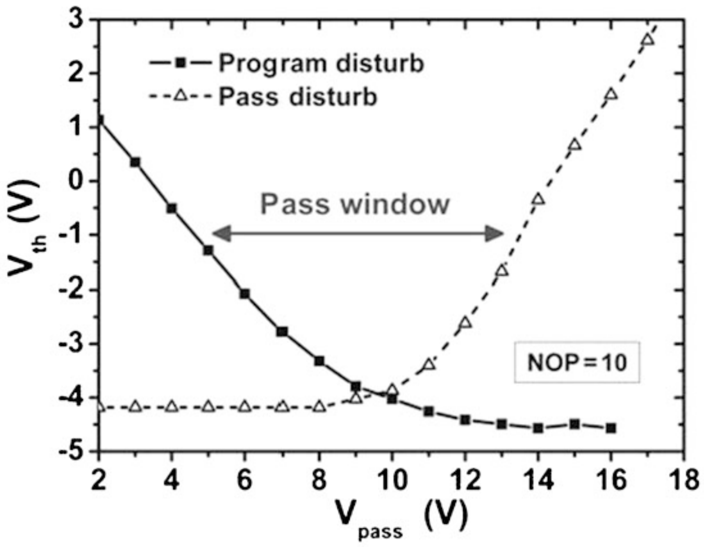


Fig. 5.12 Program and pass disturb characteristic and the resulting "pass voltage window" of a 48 nm floating gate cell in the NAND array

threshold voltage below $V_{th} = -4$ V before the program and pass disturbs could be measured. In addition to the pass voltage pulse amplitude value, the number of disturbing pulses is very important for the disturb strength. The determining factor here is the number of program operations (NOP) carried out at each word line [18]. In the example given in Fig. 5.12, the operation of a FG memory cell used in MLC mode was chosen which results, e.g., in NOP = 10. This is because every word line is logically divided into different pages which need to be separately programmed. Finally, a NOP = 10 results in approximately 100 program pulses with the highest program voltage assumed for the slowest cell in programming and about 5000 pass voltage pulses, because each of the 64 cells in the string needs to be programmed.

It can be observed that the selection of the pass voltage results in a trade-off between program and pass disturb. Generally it needs to be guaranteed that the V_{th} of all erased cells remains (with a certain margin) below $V_{th} = 0$ V.

Therefore, a "pass window" with suitable pass voltages could be determined at the level $V_{th} = -1$ V. The optimum for the trade-off between program and pass disturb can be found in Fig. 5.12 slightly below $V_{pass} = 10$ V.

5.2.4.2 Erase and Read of FG Cells in the NAND String

The advantage of the NAND Flash erase operation is that a whole erase block is erased at once. The voltage conditions during erase are shown in Fig. 5.13. All word lines are at ground potential ($V_{CG} = 0$ V) and the erase voltage is applied to

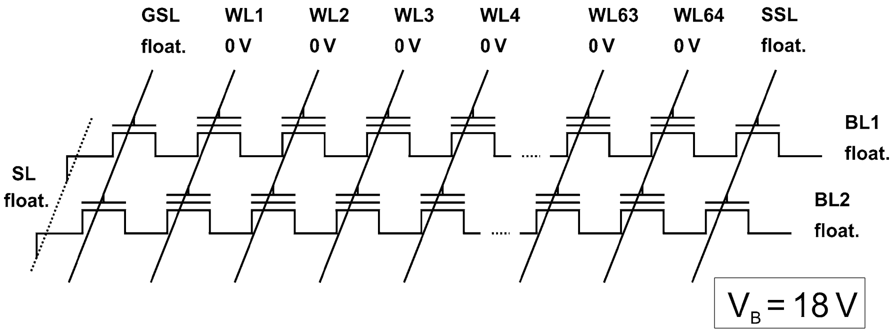


Fig. 5.13 The erase of floating gate cells in the NAND array is carried out in electrically separated erase sectors. By applying a positive voltage (e.g. $V_B = 18$ V) to the well of the erase sector, all cells are erased at the same time

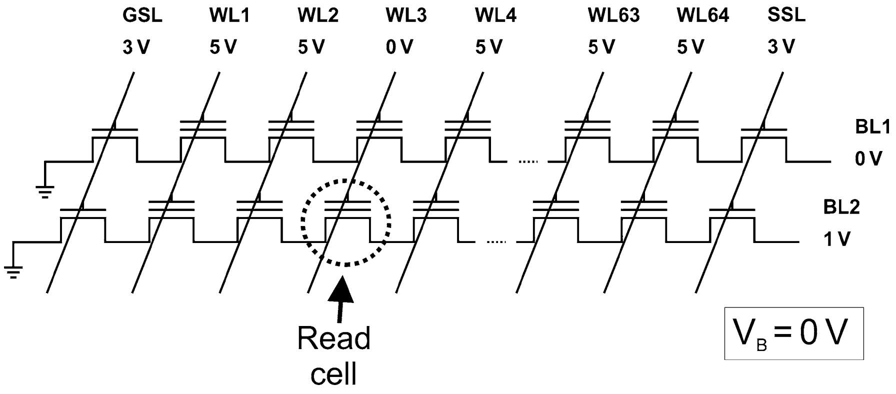


Fig. 5.14 Read operation in the NAND flash array

the well of the erase block. Very important during erase is that the select transistors as well as the bit line and the source line are left floating. For this purpose, the usually grounded source line needs to be disconnected from the ground potential. By this means, the source line and the bit line, and to a certain extend the select transistors, can follow the bulk potential, and large currents into the source line and the bit line are avoided. Due to the improved coupling when the same voltage is applied to all cells, the voltage difference between the control gate and the channel required for erase (e.g. $V_B = 18$ V) is lower than the programming voltage. The erase operation is successful when all cells in the erase block are erased below the EV level as described above.

The read operation in the NAND array is carried out word line by word line. For a current sensing read scheme the bit line which is selected for read operation (BL2 in Fig. 5.14) can be set to the read voltage (e.g. $V_{BL2} = 1$ V). For a SLC read operation the word line at the read cell is set to 0 V, while typically 5 V are applied as read pass voltage for all other word lines.

By this means it can be detected if the cell at WL3 in the string of BL2 is in the programmed or erased cell. It is clear that for reading one cell, the read current needs to flow through all cells in the 64 cell string and that only one cell in the string can be read at a time.

It needs to be mentioned that also the read pass voltage of only $V_{rpass} = 5$ V can result in a change of the threshold voltage (read disturb [19]) when only the number of read operations is high enough. For SLC FG NAND cells it is assumed that 10^6 read operations with 15 μs durations need to be guaranteed without read fails. This results in a total disturb time of about 15 s. Again, erased cells are most susceptible to read disturb as described before for program and pass disturb.

5.3 Reliability of Floating Gate NAND Memory Cells

The reliability of FG NAND Flash memory is one of the most important criteria, since typically 10 years of charge retention and 1 k to 100 k program/erase cycles need to be guaranteed for a NAND Flash product chip.

In Fig. 5.15, a typical charge retention requirement is shown. It needs to be guaranteed for a successful read-out of the stored information that the programmed V_{th} (above the PV level) is not decreased more than 10% over the product relevant time period of 10 years.

In principle, there are multiple leakage paths which can lead to a loss of the programmed floating gate electron charges as shown in Fig. 5.16a. The electrons can

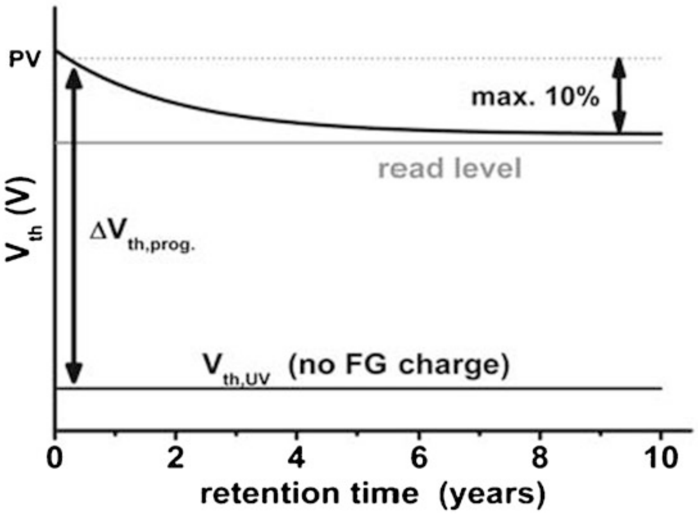


Fig. 5.15 Charge retention of an FG cell. A certain amount of charge loss needs to be tolerated (e.g. 10% V_{th} loss over the time period of 10 years)

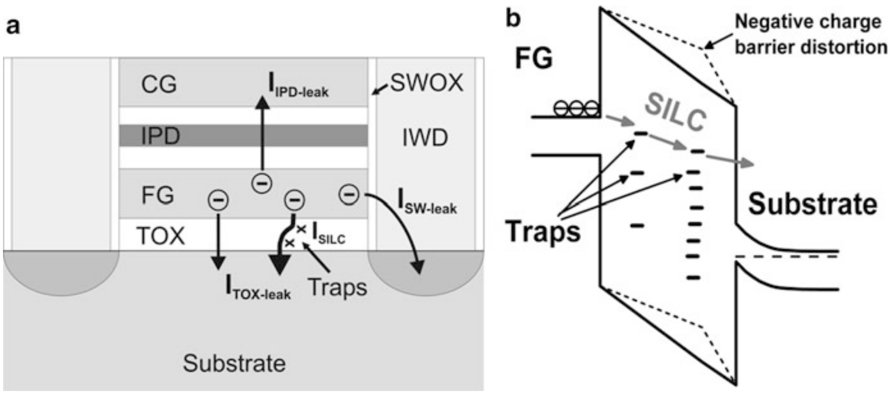


Fig. 5.16 Possible leakage path for charge loss from the floating gate (**a**). Tunnel oxide damage due to program/erase cycling and the resulting stress-induced leakage current (*SILC*) are usually the main reasons for retention loss (**a**), (**b**). Negative trap charge built up over cycling additionally induces a barrier distortion which results in an increased tunnel barrier (**b**) [12]

be lost through the IPD towards the control gate ($I_{IPD\text{-leak}}$) or leak through the cell side wall oxide (SWOX $\rightarrow I_{SW\text{-leak}}$) and the inter-word line oxide (IWD $\rightarrow I_{IWD\text{-leak}}$) to the cell junction area.

However, the most severe charge loss component of an optimized floating gate cell process is the leakage through the TOX ($I_{TOX\text{-leak}}$). This is not only because the TOX is physically the thinnest dielectric layer which holds the electrons on the floating gate, but there are additional processes which cause wear of the FG cells. As shown in Fig. 5.16a, b, the charge transfer during program and erase generates electric states in the TOX (and the TOX should be the only dielectric where charge is transferred, as previously discussed) which are called oxide traps. These traps are broken bonds of the atoms in the oxide matrix due to the electron tunneling processes [20]. The density of traps in the tunnel oxide consequently increases with the number of program/erase cycles which cause so-called oxide stress. The traps in the TOX barrier can act as stepping stones when floating gate electrons leak via a trap-assisted tunneling process towards the cell channel region. The probability of this trap-to-trap tunneling (called stress-induced leakage current, SILC) [21] is much higher than a direct tunneling process through the whole TOX thickness. The reason is that the effective tunnel distance of each tunneling step is significantly reduced for the SILC.

The TOX trap generation during the product lifetime and the corresponding SILC is the reason for a general TOX thickness scaling limitation in floating gate cells [22]. Therefore the TOX cannot be scaled below 8.0–7.5 nm. To understand this TOX thickness limitation in more detail we need to determine the oxide electric field, or alternatively, the oxide voltage during retention conditions, which is given by

$$V_{\mathrm{FG,\,Re}t.} = \alpha_{\mathrm{G}} \cdot \Delta V_{th,prog}, \tag{5.9}$$

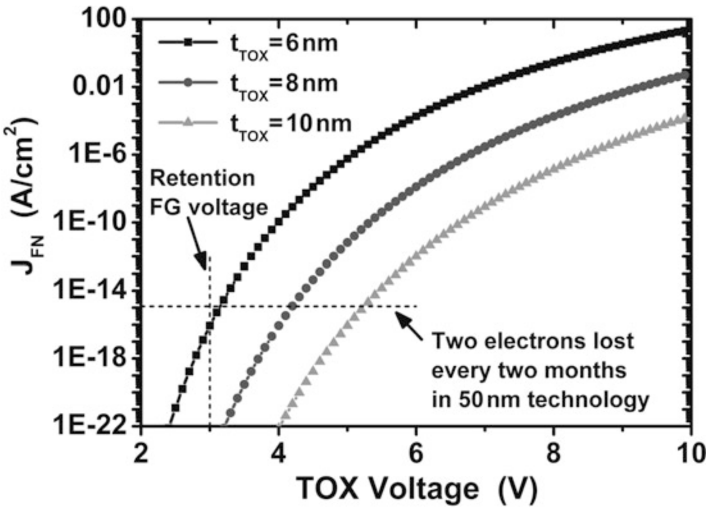


Fig. 5.17 Leakage current density through the tunnel oxide of an FG cell under retention conditions for different TOX thicknesses [22]

where α_g is again the gate coupling ratio and $\Delta V_{th,prog.}$ is the programmed threshold voltage shift as shown in Fig. 5.15. For assumed values of $\Delta V_{th,prog.} = 4–5$ V and $\alpha_g = 0.6$, the TOX voltage under retention conditions is about 3 V. The second criterion of interest is the acceptable leakage current for the 10-year charge retention.

The number of stored floating gate electrons in a 50 nm FG NAND technology for a threshold voltage shift of $\Delta V_{th} = 4$ V is about 600 (the exact number will be discussed in Sect. 5.4.4). The 10% loss criterion over the time period of 10 years results in a tolerable loss of one electron every two months (or a leakage current of 3E-26 A). Converted to a current density this is equivalent to 1E-15 A/cm².

Figure 5.17 shows the Fowler-Nordheim leakage current densities for TOX thicknesses of 6 nm, 8 nm, and 10 nm as a function of the TOX voltage. It can be seen that for an unstressed TOX and the estimated TOX retention voltage $V_{FG,Ret} = 3$ V and current criterion, a tunnel oxide thickness of 6 nm would be sufficient. However, 2 nm additional TOX thickness is required to fulfill the retention criterion for a damaged TOX with trap-to-trap SILC leakage as discussed above.

Figure 5.18 shows the endurance of FG cells in a 48 nm NAND technology. All program and erase cycles were carried out with unchanged program and erase cycle voltages of $V_{CG,prog} = 23$ V and $V_{erase} = -19$ V for the indicated pulse times. For low cycle numbers the V_{th} window is slightly increases, whereas for higher cycle number above 300 cycles the V_{th} window closes. Furthermore, a general V_{th} upward shift is visible.

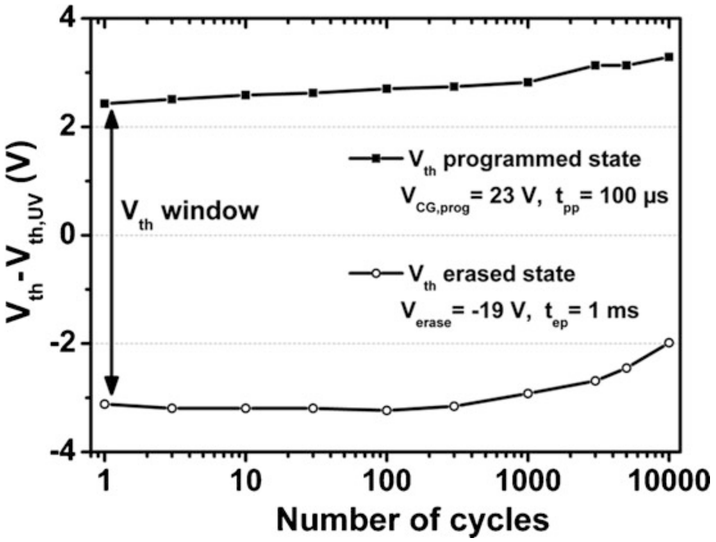


Fig. 5.18 Program/erase cycling endurance of a FG cell in a 48 nm NAND technology

This behavior can be explained with positive charge trapping at low cycle counts which leads to a reduced TOX barrier and negative charge trapping which results in an increased barrier height (see Fig. 5.16b) at higher cycle numbers.

For a reduced tunneling barrier, more electrons can be transferred through the TOX for unchanged program and erase voltages, whereas for an increased barrier this number of transferred electrons is reduced. Additionally, the fixed negative charges which are generated in the TOX for higher cycle counts generally increase the cell V_{th}. In the case shown in Fig. 5.18, the erased cell V_{th} is shifted by one volt after 10 k program/erase cycles. Besides the increased retention problem for higher cycle numbers due to trap generation, the window closing and the general V_{th} upward shift will result in increased pulse voltages, especially for erase.

5.4 Scaling of Floating Gate NAND Memory Cells

The NAND Flash memory scaling of the last 15 years was accomplished by reducing the cell dimensions, whereas the cell construction principle was unchanged. The effective cell size of NAND Flash in 1995 was in the range of 1 μm^2 which resulted in a product chip memory capacity of 32 Mb [8]. In 2010, the cell size was reduced to 0.0028 μm^2 [10] with a chip capacity of 64 Gb. This strong reduction of the cell geometry leads to scaling issues which are discussed in the following.

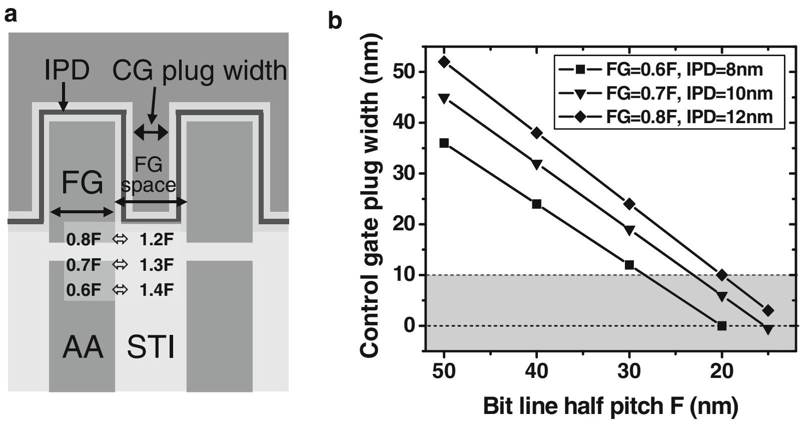


Fig. 5.19 Bit line pitch scaling limitation for the typical control gate to floating gate enhanced coupling area FG NAND cell. To fit two times the IPD thickness plus the poly plug (**a**) with an assumed minimum width of 10 nm, the active area (*AA*) width can be reduced below the half pitch F to clear a space for the CG plug (**b**)

5.4.1 Scaling of the Floating Gate Cell Geometry

As described in Sect. 5.2.3, it is very important for a programmability of floating gate cells to have an enhanced control gate to floating gate area by a control gate which is wrapped around the floating gate. However, this requires a certain space between adjacent floating gates, since this space needs to fit two times the IPD thickness plus the poly plug. Depending on the FG NAND ground rule (or half pitch F), this has some implications for the remaining control gate plug width as shown in Fig. 5.19a.

Figure 5.19b shows the remaining control gate plug width as a function of the bit line half pitch F. To obtain more space for the control gate plug, the width of the floating gate can be reduced with respect to the space between the floating gates as done in the latest FG NAND generations [23], [24]. The space between adjacent floating gates consequently becomes wider, as indicated in Fig. 5.19a. Additionally, the physical IPD thickness can be reduced. These two options are combined in Fig. 5.19b with the result that for an FG width of 0.6 F and a physical IPD thickness of only 8 nm a control gate plug width of 10 nm can be realized down to a bit line half pitch of 20 nm. Due to this bit line pitch scaling limitation it can be observed in the latest FG NAND technology generations that the bit line pitch is less aggressively scaled than the word line pitch [10, 11, 23].

In case of very narrow control gate plugs, it may be that the poly-Si doping level in the CG plug cannot be maintained sufficiently high. This would result in poly-Si depletion and consequently in an electrically inactive CG plug. An alternative could be a metal control gate material as presented in [2].

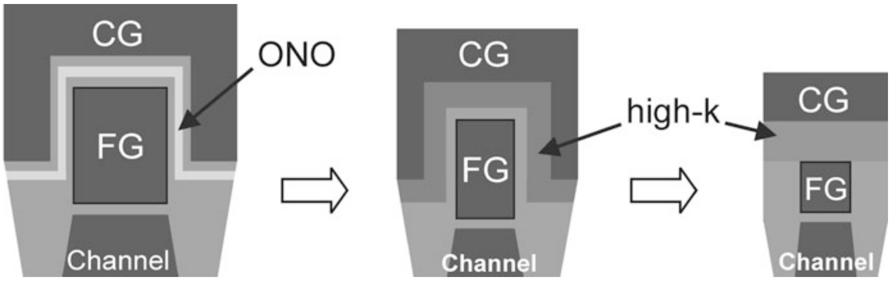


Fig. 5.20 Floating gate NAND cell scaling: The requirement for a continued reduction in the floating gate cell dimensions in combination with a high gate coupling ratio leads from the typical ONO IPD cell with a control gate wrapped around the floating gate to a high-k containing IPD, and finally due to the lack of space for the control gate plug to a planar floating gate cell

Continued scaling of floating gate NAND cells (see Fig. 5.20) in combination with a sufficiently high gate coupling ratio requires efforts to reduce the electrical IPD thickness (EOT). One option to do so is the introduction of high-k dielectrics in the IPD stack. However, at a certain floating gate NAND technology node there won't be sufficient space for the control gate plug, which automatically leads to a planar floating gate cell as shown in Fig. 5.20.

It was discussed in Sect. 5.2.3 that for insufficiently high gate coupling ratios together with an electrically thin IPD, tunnel currents can flow through the IPD during the program and erase conditions. The IPD leakage results in degraded program and erase behavior, visible in reduced ISPP and erase slopes [25]. Consequently, a fully planar floating gate cell with ONO IPD cannot be programmed and erased in the traditional manner where charge is transferred through the tunnel oxide only. Even an IPD layer combination of SiO_2 and high-k or a pure high-k IPD layer is problematic with respect to program/erase saturation [13].

A possible way out of the planar floating gate cell scaling dilemma may be a dual layer floating gate as proposed in [26]. Figure 5.21 illustrates the advantages of a dual layer floating gate with an n-doped poly-Si bottom part (adjacent to the tunnel oxide) and a high work function metal layer on top (adjacent to the high-k IPD) with respect to program and erase saturation.

Figures 5.21a, b shows the conditions during program operation. The n-poly-Si floating gate in Fig. 5.21a has the problem of the insufficient effective IPD barrier which does not provide sufficient current blocking margin to program the cells to high V_{th} levels. The situation is improved by the introduction of the high work function metal gate layer, as shown in Fig. 5.21b, where the barrier height and the effective electron tunneling barrier (shadowed area) is significantly larger. The advantage of the dual layer floating gate under erase conditions and why simply a single layer high work function metal FG cannot replace the poly FG are illustrated in Fig. 5.21c, d, respectively. The single layer metal floating gate has a larger barrier between the FG and TOX which would hinder the erase when electrons are tunneling out of the FG towards the channel region (Fig. 5.21c). Consequently, a

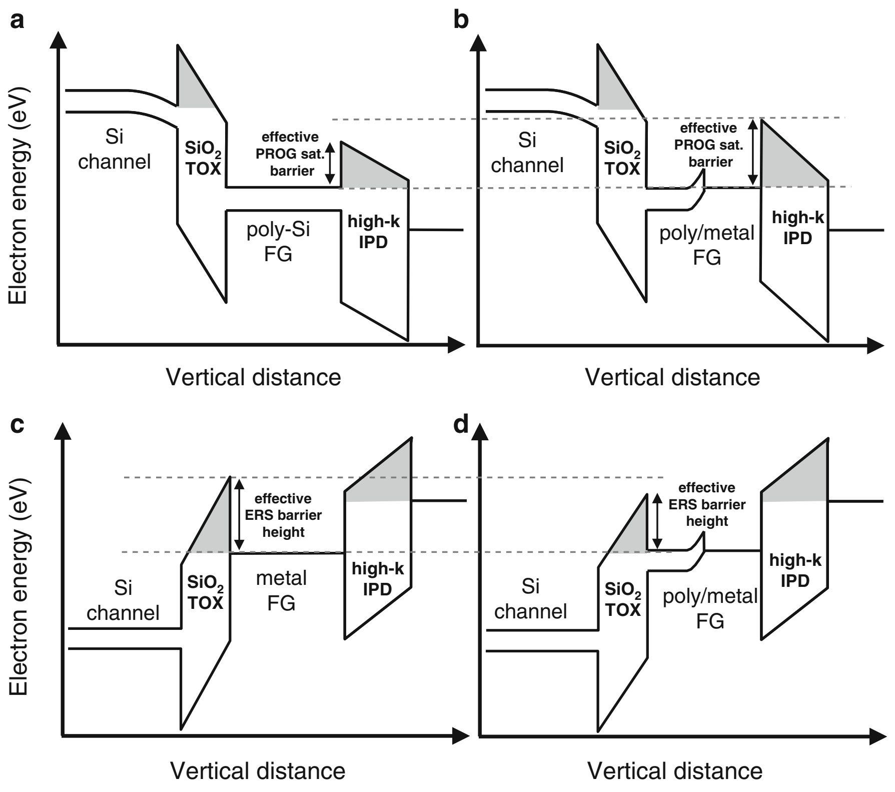


Fig. 5.21 Field improvement in planar floating gate cells and how program and erase saturation can be avoided by the usage of a dual layer FG structure [26]

higher erase voltage would be necessary with the even more problematic effect that at the same time electrons tunnel from the control gate to the floating gate (electron back tunneling) and cause erase saturation. This electron back tunneling will be seen in Sect. 5.6 to be one of the major issues of charge trapping memory cells, but is less problematic for the dual layer FG as seen in Fig. 5.21d.

5.4.2 *Floating Gate Cell Cross-Coupling*

Another general problem for floating gate NAND cells in technology generations below 50 nm is the cell-to-cell cross-coupling. This effect is the direct coupling from one floating gate to the nearest neighboring floating gates as shown in Fig. 5.22. It is clear that this direct coupling increases for reduced dimensions since the cells move closer together and therefore the relative coupling capacitance increases. Most significant is the FG to FG coupling in the direction along the bit lines (y-direction

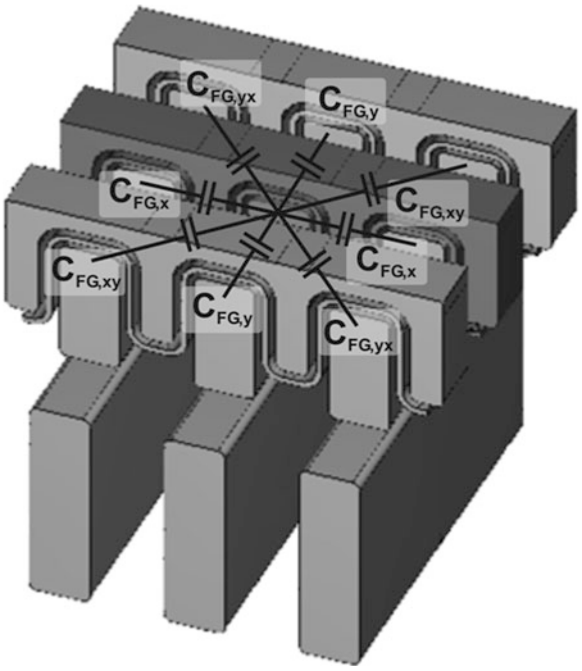


Fig. 5.22 Floating gate cross-coupling in scaled NAND Flash technologies [27]

in Fig. 5.22). This is because the floating gates are directly face each other with the full FG height and full FG width in this direction. Consequently, $C_{FG,y}$ is the largest of the FG to FG coupling capacitance terms. In the direction along the word lines (x-direction), parts of the FG to FG coupling are screened by the control gate plug and therefore $C_{FG,x}$ is typically smaller than $C_{FG,y}$. To minimize the coupling capacitance in x-direction it would be beneficial to have a very deep position of the CG plug, ideally down to the STI level, which would mean a complete screening in x-direction. However, the full programming voltage drop between the control gate plug and the channel limits the minimum CG plug to channel distance. The diagonal coupling components $C_{FG,xy}$ and $C_{FG,yx}$ are typically the smallest ones.

In cell programming schemes, where even and odd bit lines are programmed separately (because they belong to different logical pages), the programming of a cell can change the threshold voltage of a directly neighboring cell which was already programmed. This effect is called floating gate cross-coupling or floating gate interference [27].

The cell-to-cell coupling potentially leads to a decreased gate coupling ratio since all increased capacitance terms from FG cross-coupling are added in the denominator of the gate coupling ratio Eq. (5.3). Therefore, the gate coupling ratio decreases at least in the case where the floating gate cell dimensions are scaled proportional to the technology node, while TOX and IPD thicknesses are kept constant.

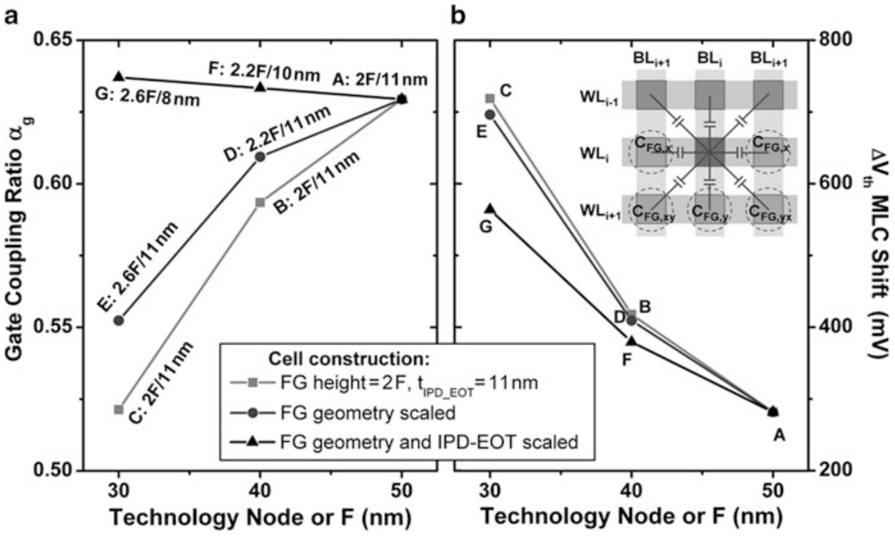


Fig. 5.23 Gate coupling ratio (**a**) and threshold voltage shift (MLC shift) due to the programming of five directly neighboring cells (**b**) by a $\Delta V_{th,prog} = 5$ V as a function of cell technology generation. For each point, the floating gate height and the IPD EOT value (e.g. 2 F/11 nm) are given. The floating gate width is 1 F for each technology node and the TOX thickness is always 8.5 nm

This behavior can be seen in the lowermost curve of Fig. 5.23a obtained from 3D simulations with a commercial field solver [28]. It can be seen that for a constant IPD EOT of 11 nm in combination with a floating gate whose height is two times the width (width = F, height = 2 F in points A, B, and C), the gate coupling ratio decreases from 0.63 in the 50 nm technology node to only 0.52 in the 30 nm technology. A slight gate coupling ratio improvement can be seen for an increased floating gate height to width ratio with decreasing half pitch in the middle curve (points A, D, and E) of Fig. 5.23a. A slightly increasing α_g for smaller dimensions is only obtained here for an increased FG height to width ratio in combination with a decreased IPD effective thickness (points A, F, and G).

However, in Fig. 5.23b, it is apparent that all efforts to keep the gate coupling ratio value high do not significantly improve the V_{th} shift due to neighboring cell programming in conventional cell programming schemes. For the simulation of the depicted ΔV_{th} MLC shift it is assumed that five neighboring cells influence the V_{th} of each ready programmed cell in worst case, as indicated in the inset of Fig. 5.23b. In detail, this five cell consist of two neighboring cells in word line direction, two diagonal cells, and one directly neighboring cell in bit line direction, resulting from an assumed conventional word line by word line programming scheme for serial even and odd bit line addressing. In the NAND chip layout belonging to the serial WL programming of even and odd bit lines the serial treatment is necessarily performed, since two neighboring bit lines share one single sense amplifier for reading the V_{th} state during ISPP programming. The cross-coupling capacitance terms were again taken from the 3D field simulations, and for the depicted MLC shift it is assumed that all five cells are programmed by a $\Delta V_{th} = 5$ V. This would be the threshold voltage shift for erased FG cells which are programmed to $V_{th} = 4$ V.

The fact that for conventional programming schemes the simulated MLC shift at 30 nm cannot be reduced below 500 mV leads to the conclusion that at a certain point in shrinking the FG NAND Flash dimensions the program algorithm needs to take care of the floating gate cross-coupling issue. The strategy is simply to reduce the number of neighboring cells that are programmed after reaching the final programming target V_{th} of each cell, in combination with a reduction of the amount these neighboring cells increase their V_{th}.

One component for reducing the unwanted FG cross-coupling is the all bit line (ABL) architecture, where each bit line has a separate sense amplifier and therefore all bit lines can be programmed at the same time.

Together with the improved program algorithm with respect to the order in which the cells are programmed, it was possible to master FG cross-coupling even for three bit per cell (TLC) and four bit per cell (XLC) technologies [11, 29, 30].

5.4.3 *Word Line to Word Line Leakage Current*

The reduced cell–to-cell distances with scaled dimensions also cause strongly increased electric fields between neighboring word lines during program operation.

The WL-to-WL voltages during erase are uncritical because all cells are erased at the same time and therefore all word lines are at the same potential.

High WL voltage differences during program operation are even more critical since the programming voltage does not scale or rather increase slightly, as described above. As a result of the strong electric fields between word lines, electrons can tunnel from a programmed floating gate to the control gate that is on the high program voltage V_{pgm} [31] or generally introduce WL-to-WL leakage currents as shown in Fig. 5.24. The electric field strength in an assumed SiO_2 IWD is shown for different WL-to-WL distances as a function of the WL difference voltage in Table 5.1.

Generally speaking, electric fields up to 4 MV/cm can be handled with deposited oxides as the IWD with sufficient reliability. The field range above 4 MV/cm becomes critical, but the range of 8 MV/cm and above is already in the Fowler-Nordheim tunneling regime for a thermally grown oxide which would not allow a reliable operation anymore.

Options to reduce WL-to-WL leakage by use of a special program algorithm would include limiting the difference voltage between adjacent word lines. This could be accomplished with a specific handling of the word lines close to the program word, similar to the individual word line treatment in local program inhibit schemes [16]. However, effectively increasing the pass voltage at the cells adjacent to the programmed cell will adversely affect the pass disturb.

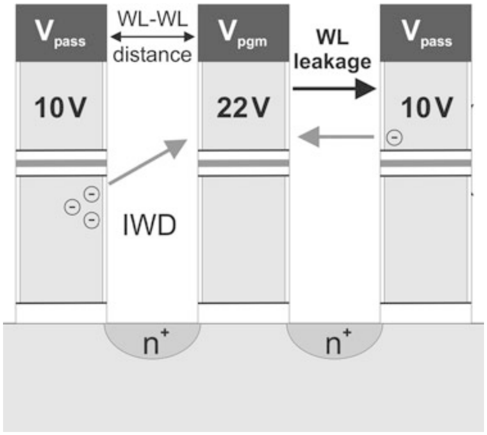


Fig. 5.24 The voltage conditions during the program operation can cause a leakage current between neighboring word lines or from an already programmed FG to the actually programmed WL

Table 5.1 WL-to-WL IWD (SiO_2) electric field in MV/cm as a function of the voltage and the distance between different word lines. The light grey shaded WL-WL distance and voltage combinations represent electric IWD fields above the usual 4 MV/cm operation conditions. The dark grey shaded electric IWD field range above 8 MV/cm represent very high values in the Fowler-Nordheim tunnelling regime (see Fig. 5.5)

WL-to-WL Distance

WL-to-WL Difference Voltage	10 nm	15 nm	20 nm	25 nm	30 nm	35 nm	40 nm	45 nm	50 nm
20 V	20.0	13.3	10.0	8.0	6.7	5.7	5.0	4.4	4.0
19 V	19.0	12.7	9.5	7.6	6.3	5.4	4.8	4.2	3.8
18 V	18.0	12.0	9.0	7.2	6.0	5.1	4.5	4.0	3.6
17 V	17.0	11.3	8.5	6.8	5.7	4.9	4.3	3.8	3.4
16 V	16.0	10.7	8.0	6.4	5.3	4.6	4.0	3.6	3.2
15 V	15.0	10.0	7.5	6.0	5.0	4.3	3.8	3.3	3.0
14 V	14.0	9.3	7.0	5.6	4.7	4.0	3.5	3.1	2.8
13 V	13.0	8.7	6.5	5.2	4.3	3.7	3.3	2.9	2.6
12 V	12.0	8.0	6.0	4.8	4.0	3.4	3.0	2.7	2.4
11 V	11.0	7.3	5.5	4.4	3.7	3.1	2.8	2.4	2.2
10 V	10.0	6.7	5.0	4.0	3.3	2.9	2.5	2.2	2.0
9 V	9.0	6.0	4.5	3.6	3.0	2.6	2.3	2.0	1.8
8 V	8.0	5.3	4.0	3.2	2.7	2.3	2.0	1.8	1.6
7 V	7.0	4.7	3.5	2.8	2.3	2.0	1.8	1.6	1.4
6 V	6.0	4.0	3.0	2.4	2.0	1.7	1.5	1.3	1.2
5 V	5.0	3.3	2.5	2.0	1.7	1.4	1.3	1.1	1.0

IWD Electric Field in MV/cm

5.4.4 *Number of Stored Floating Gate Electrons*

When the dimensions of floating gate cells are scaled down, also the number of floating gate electrons needed for a certain threshold voltage shift ΔV_{th} is reduced. On the one hand, this reduced number of stored floating gate electrons is critical for reliability and charge retention because the loss of one electron has increasing impact on the cell V_{th} loss. On the other hand, the charge granularity of single

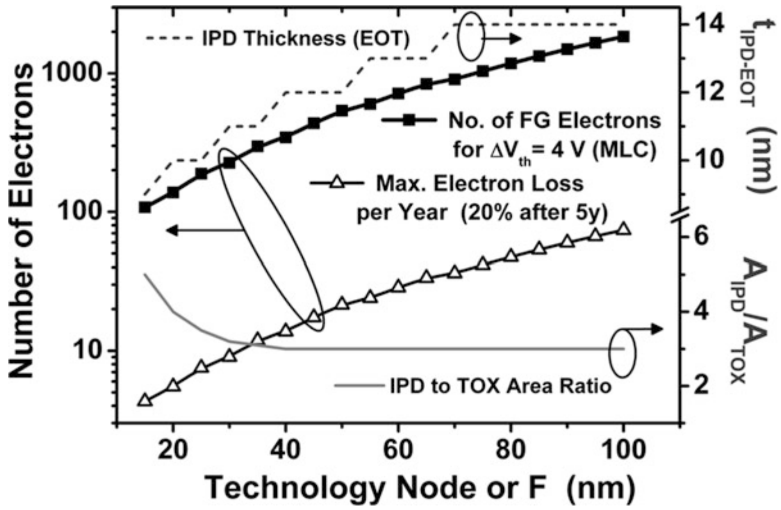


Fig. 5.25 Number of electrons as a function of the technology node F. To havethe programming voltages remain similar over different technology generations, the gate coupling ratio is optimized by means of an IPD EOT reduction and an increase of the CG-FG to TOX area ratio

electrons affects, at a certain stage, the ability to program narrow V_{th} distributions. The effect is most critical in TLC or XLC NAND technologies with very narrow V_{th} distributions in case one electron causes a significant threshold voltage shift.

The approximated number of floating gate electron can be derived from Eq. (5.5) and is given as a function of the feature size F for different NAND technology nodes by

$$N = \frac{C_{CG}}{e} \cdot \Delta V_{th} = \frac{\varepsilon_0 \varepsilon_r}{e} \frac{A_{CG-FG}}{t_{IPD-EOT}} \cdot \Delta V_{th} = \frac{\varepsilon_0 \varepsilon_r}{e} \frac{A_{IPD}/A_{TOX}}{t_{IPD-EOT}} \cdot F^2 \cdot \Delta V_{th} \quad (5.10)$$

where e is the electron charge and A_{IPD}/A_{TOX} is the CG-FG area to TOX area ratio.

As shown in Fig. 5.25 and discussed beforehand, this area ratio needs to be increased in combination with a reduction of the IPD EOT value to have the programming voltages remain the same. The shown values for the A_{IPD}/A_{TOX} ratio and IPD EOT are similar to the values used by major NAND Flash manufacturers in recent generation.

The simple planar plate capacitor approximation of Eq. (5.10) results in the estimate of about 200 stored electrons, in case a 25 nm FG NAND cell is programmed to $\Delta V_{th} = 4$ V above the UV level, as depicted in Fig. 5.25. The tolerable electron loss per year for this technology node is already less than ten, if a relaxed retention criterion compared to Sect. 5.3 with 20% tolerable V_{th} loss after 5 years is assumed.

However, the general trend of the number of stored electrons as a function of the FG cell technology node in Fig. 5.25 shows a strong reduction with reduced dimensions.

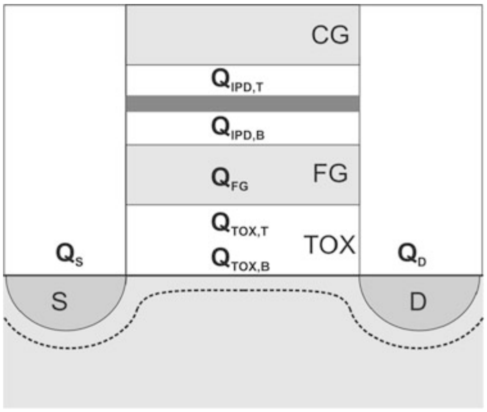


Fig. 5.26 Locations of trapped charges in an FG NAND memory cell which cause a threshold voltage shift

Table 5.2 Electron sensitivity of different FG NAND Flash technology generations

Technology	50 nm	35 nm	25 nm
$Q_{TOX,B}/e$	4	2	1
$Q_{TOX,T}/e$	9	7	4
Q_{FG}/e	18	12	10
$Q_{IPD,B}/e$	22	17	11
$Q_{IPD,T}/e$	149	103	100
Q_S/e	33	9	5
Q_D/e	61	16	10

The table indicates the number of electrons required at different locations in an FG cell for a 100 mV threshold voltage shift as determined by TCAT simulations in [10]

A similar consideration based on TCAD simulations was carried out and presented in [10]. The result of the number of electrons stored in different FG cell locations (see Fig. 5.26) that cause a threshold voltage shift of $\Delta V_{th} = 100$ mV is shown in Table 5.2 for 50 nm, 35 nm, and 25 nm technology generations. The number of electrons required for a $\Delta V_{th} = 4$ V shift in a 25 nm technology taken from these values is 400 and therefore two times higher than the estimate of Eq. (5.10), but the trend over different technology generations is the same.

Table 5.2 indicates that especially electrons stored in tunnel oxide traps which are generated during program and erase operations cause higher V_{th} shifts per electron than electrons in the FG. Therefore, uncontrolled electron storage in the TOX can be a significant issue as discussed in the following section.

5.4.5 Random Telegraph Noise

Random telegraph noise (RTN) can be observed in different types of field effect devices and can be explained by electron capture and emission processes in oxide

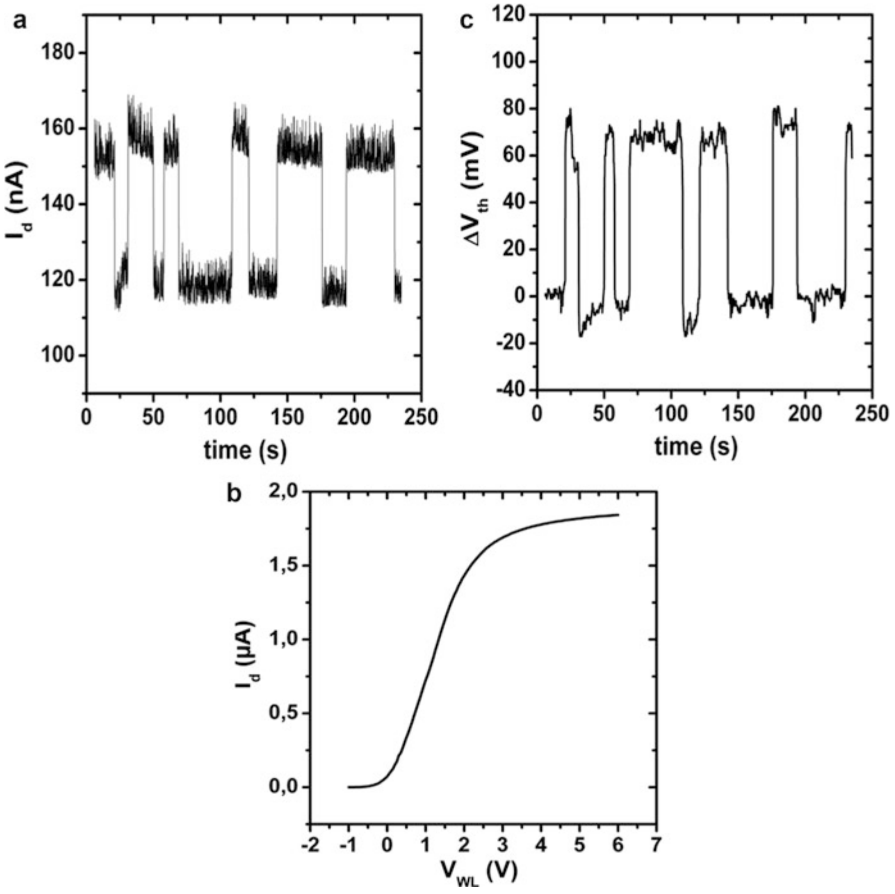


Fig. 5.27 Random telegraph noise of 48 nm FG cells in a NAND string configuration. The variation in the string current (**a**) due to charging and discharging of one oxide trap in the channel region can be converted by the string transfer curve (**b**) into a V_{th} variation (**c**)

traps close to the channel of a MOSFET device [32]. As mentioned previously, the same process can take place in the TOX of a floating gate NAND cell [33, 34].

Figure 5.27 shows RTN measurements in a 32 cell string of a 48 nm FG NAND technology. Operated in the sub-threshold region, the drain current of the investigated cell (or the string current) shows a characteristic two level I_d signature as shown in Fig. 5.27a. The two level signature and the time constants for capture and emission in the second range indicate that a single tunnel oxide trap about 1–2 nm from the channel/TOX interface [35] is charged and discharged by direct tunneling.

With the aid of the string $I_d - V_{WL}$ transfer curve in Fig. 5.27b, the current signal can be converted into a threshold voltage shift ΔV_{th} as depicted in Fig. 5.27c. The resulting RTN amplitude is about 70 mV and in this case higher than expected from the TCAD simulations [10] in Table 5.2.

However, for scaled dimensions the RTN threshold voltage shifts can cause read fails, which is even more significant for MLC and TLC functionality with small distances between V_{th} distributions.

5.5 Shrinking the Floating Gate NAND Technology Beyond the Direct Optical Lithography Limitation

The effects of scaled dimension on the functionality of floating gate NAND cells as described in the last section are one aspect of the shrinking issues. Another aspect is the generation of the extremely small structures in NAND Flash memory cells which currently arrived in the sub-20 nm range [23].

This development of the feature size or critical dimension (CD) is even more impressive, because the size of actual cell structures is one order of magnitude smaller than that of the 193 nm wavelength of the ArF laser which is used for illumination.

To understand the challenge to generate such small structures, Fig. 5.28 shows the CD development of the NAND Flash technology half pitch and the used lithography wavelength since 1996.

At the end of the 1990s, the NAND Flash CD in the cell array was close to the lithography wavelength. However, since the 193 nm was the last reduction of the

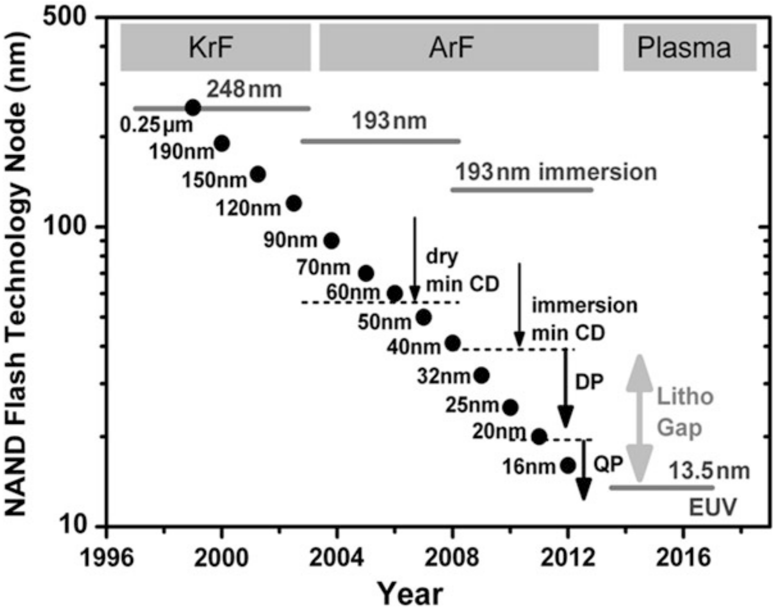


Fig. 5.28 NAND Flash technology generations and lithographic resolutions

wavelength used as a light source for lithography, the gap between the NAND Flash technology node and the lithographic wavelength has been increasing since then.

The ability of a lithographic system to generate a minimum CD is described by

$$\mathrm{CD} = \mathrm{k}_1 \frac{\lambda}{\mathrm{NA}} \tag{5.11}$$

where k_1 is a constant, λ is the wavelength, and NA is the numerical aperture of the optical illumination system. For a single exposure, dry 193 nm lithography with optimized illumination conditions with, e.g., $\mathrm{k}_1 = 0.28$ in combination with a numerical aperture in the range of $\mathrm{NA} = 0.93$, the minimum CD is limited to values slightly below 60 nm [36].

With the introduction of immersion lithography with a liquid on top of the wafer during illumination, the NA could be improved to 1.35, which is also the reason why the 193 nm immersion lithography wavelength is shown in Fig. 5.28 "virtually" reduced by this factor. The smallest achievable half pitch for single exposure 193 nm immersion lithography is therefore about 38 nm [37].

To bridge the gap to extreme UV (EUV) lithography (see litho gap in Fig. 5.28), which is currently not expected to be available for industrial volume production before 2013, the semiconductor industry introduced special process sequences to generate small structures that cannot be obtained by single exposure direct printing.

For logic circuits, such as microprocessors, it is usually sufficient to generate the required small gate length by a trimming of larger lithographically generated structures. The required short gate length in logic circuits can therefore be obtained by tapered trim etch processes.

In memory products such as DRAM or NAND Flash it is not the small memory cell structure itself that is important, but the high memory cell density. Besides, the memory cell arrays have the great advantage that the basic structure consists of a very regular line and space pattern, which can be printed more easily than complex state-of-the-art SRAM structures.

Consequently, it is necessary to generate additional features that cannot be directly printed by lithography.

Most common for NAND Flash memory are process sequences which generate two smaller lines with a corresponding space out of one larger line that can be printed lithographically. This kinds of process sequences which basically make two lines out of one are known as self-aligned double patterning (SADP) [38, 39], or sometimes pitch fragmentation [28]. The typical SADP approach is schematically shown in Fig. 5.29.

The starting point is a multiple layer stack of CVD-deposited materials like a-Si, Si_3N_4, SiO_2, and carbon hard masks which can be selectively etched to each other. Double patterning starts with a directly printed equal line and space pattern which has two times the half pitch of the final structures (Fig. 5.29a). For a 20 nm target half pitch, the initial line and space half pitch consequently would be 40 nm. With the aid of the tapered trim etch process, this pattern is transferred to the underlying layer with a line width half of the initial line. Subsequently, a conformal liner is

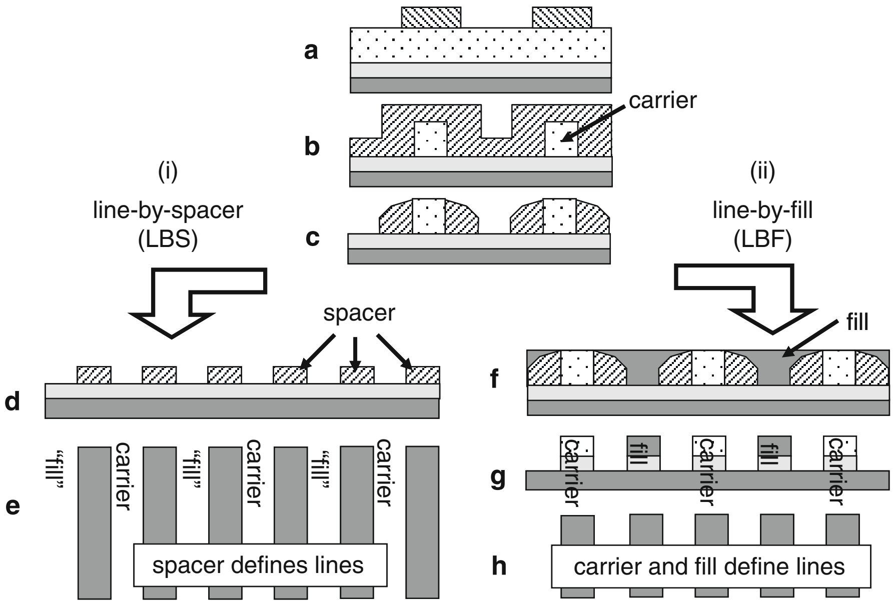


Fig. 5.29 Schematic double patterning sequences line-by-spacer (*LBS*) (i) and line-by-fill (*LBF*) (ii) [28]. The line width of an equal line and space pattern (**a**) is reduced by a trim etch process and a conformal liner is deposited (**b**) in order to generate spacer (**c**) of the same width as the trimmed lines. In the line-by-spacer sequence the spacers are used after line removal (**d**) to generate the final pattern (**e**), in contrast to the line-by-fill sequence where additional "fill" lines are generated in between the spacers (**f**) and the carrier and fill lines are used after spacer removal (**g**) to generate the target pattern (**h**)

deposited (Fig. 5.29b) to generate a spacer with the width of the target half pitch as shown in (Fig. 5.29c). Proceeding from this processing stage, two different SADP final sequences can be principally chosen. Option (i) is the so-called line-by-spacer (LBS) sequence because it uses the generated spacer (Fig. 5.29d) to transfer the obtained pattern into the underlying hard mask. Prior to this, the carrier needs to be removed. The resulting hard mask structure is the equivalent of a single exposure lithographically generated pattern at larger half pitches, which is, in turn, used for patterning of the active chip structure as shown in Fig. 5.29e). Processing images of a LBS SADP sequence is shown in Fig. 5.30.

Figure 5.30a shows the situation after the trim etch step with a line width one quarter of the initial pitch. Fig. 5.30b, c illustrates the process after the spacer etch and the carrier recess etch, where the trimmed initial line is removed. In Fig. 5.30d the spacer pattern is transferred into the hard mask and the spacer is removed in (e). When the small SADP-generated structures in the memory array are generated, the close connection of every two neighboring lines needs to be etched away. This cut etch process can be carried out together with the patterning of periphery structures or, e.g., the select transistors as shown in Fig. 5.30f.

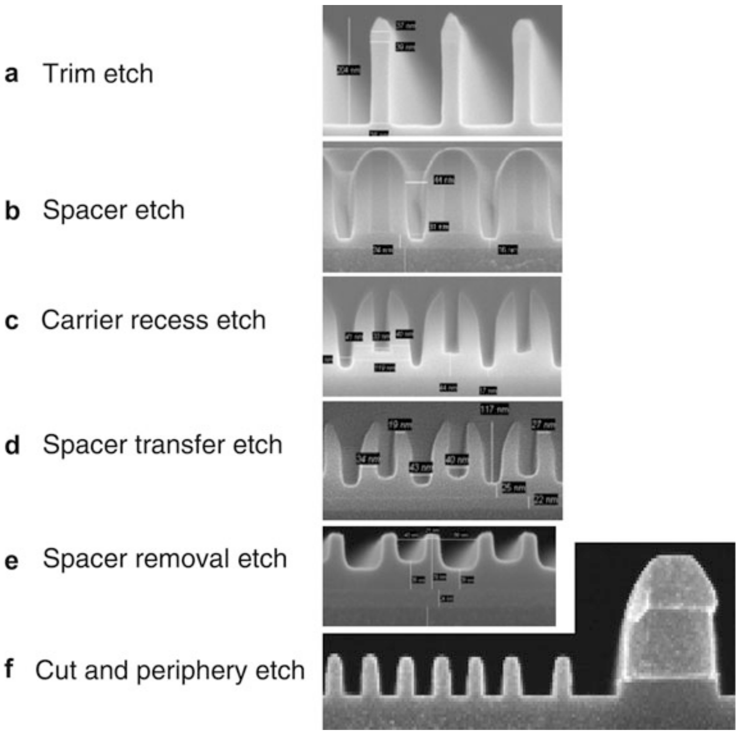


Fig. 5.30 Exemplary line-by-spacer process sequence [28], [39]

The second SADP processing option (ii) in Fig. 5.29 is the line-by-fill (LBF) sequence. Subsequent to the spacer formation in Fig. 5.29c, a material that can be as selectively etched to the spacer (e.g. the same material as the carrier) is filled in between the spacers. Therefore, the material is called "fill" as shown in Fig. 5.29f. Before the spacer material in between the carrier and fill lines can be removed as depicted in Fig. 5.29g, a chemical-mechanical planarization (CMP) process step is needed to have a better exposure of the spacer material to the etch chemistry. In the final step, the pattern can be transferred into the hard mask which is shown in Fig. 5.29h.

With respect to CD variations, it should be mentioned that generally the spacer width in SADP schemes can be better controlled than the carrier and fill width. The spacer width variations mostly depend on thickness conformity of the deposited spacer liner. In contrast, the carrier and fill line widths essentially depend on two critical processes, which are the carrier trim etch and the spacer formation.

The knowledge of this different CD control can be used to guarantee a reliable operation of FG NAND cells. It was described that the control gate plug is essential for the gate coupling ratio and consequently for the FG cell performance.

Based on this, it is beneficial to use the LBF sequence for the one-step patterning of the active area and floating gate width in a self-aligned STI (SA-STI) cell approach [40] as shown in Fig. 5.31a.

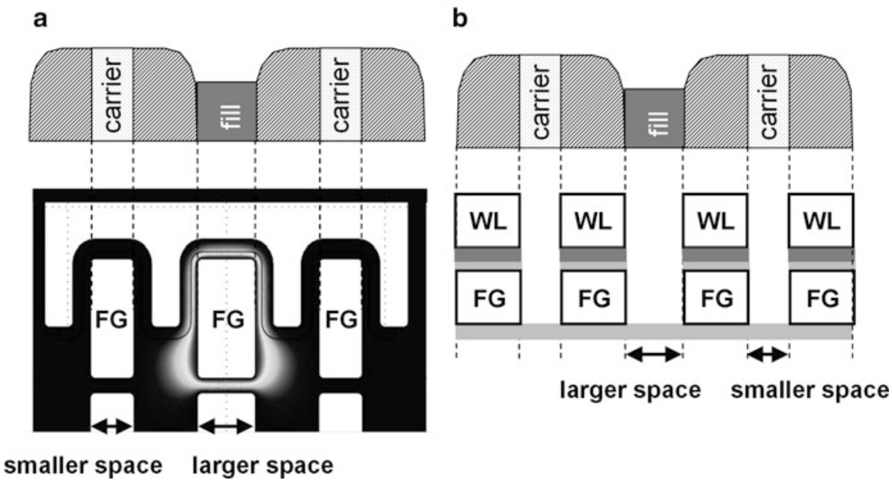


Fig. 5.31 Major variations in LBF (**a**) and LBS (**b**) pitch fragmentation sequences [28]

This choice has the major advantage that the space for the critical control gate plug has a good controllability [28]. For the patterning of the word line level which defines the length of the FG cells it could be beneficial to use the LBS sequence. The consequential spacer-defined good control of the FG cell length can help to reduce cell-to-cell V_{th} variations since the latest NAND cell generations are definitely in the short channel regime which increases cell length effects.

As shown in Fig. 5.28, it is required for FG NAND technologies beyond 20 nm half pitch to use quadruple patterning (QP) techniques [23, 37] to generate such small structures. Quadruple patterning is essentially two times the consecutive usage SADP with its logical consequences for the CD control of lines and spaces.

5.6 Charge Trapping NAND Memory Cells as Floating Gate Cell Replacement

The construction of charge trapping (CT) memory cells for NAND application is at first glance not very different from the floating gate NAND cell construction. The major difference is that charge is stored in a non-conducting dielectric layer with high trap density instead of the conducting floating gate. This non-conducting charge storage layer has two major consequences:

1. The surface of the dielectric charge storage layer is not an equipotential surface as the floating gate. The stored charge can be inhomogeneously distributed when the injection is locally enhanced.
2. In a planar cell structure, no capacitive voltage divider can be formed to concentrate the voltage drop and, therefore, the electric field to the tunnel oxide as in floating gate cells (with optimized gate coupling ratio α_g).

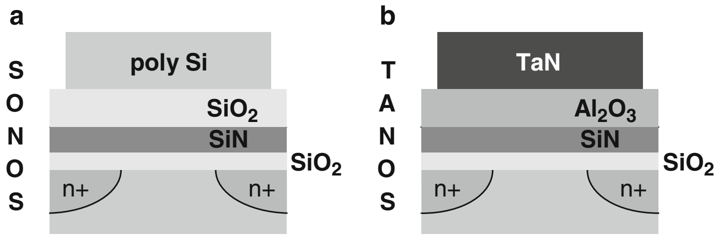


Fig. 5.32 Charge trapping stacks in SONOS (**a**) and TANOS [48] (**b**) memory cells

The typical layout of CT memory cells is shown in Fig. 5.32. The traditional SONOS (poly-Si/SiO_2/Si_3N_4/SiO_2/Si) cell, as shown in Fig. 5.32a, stores the charge in a Si_3N_4 (SiN) layer. SiN is widely used as the charge trapping layer (CTL) due to its high trap density of a few times 10^{19} cm^{-3} and its good process compatibility with Si and SiO_2. Sometimes other dielectrics are used for charge storage, such as Al_2O_3 [41].

CT memory cells typically have a planar cell layout and therefore resemble planar FG cells, layout-wise. Due to the lack of an increased gate coupling ratio it cannot be realized that charge is only transferred through the tunnel oxide during program and erase operation. Under the Fowler-Nordheim program condition in the CT cell the injected electron current tunnels through the whole CT stack. Only a certain part of this tunneling current is trapped in trap states and cause a V_{th} increase. The rest of the injected electron current leaves the charge trapping layer towards the gate electrode. Consequently, the ISPP slope for CT memory cells is not at unity, but rather in the range between 0.6 and 0.8 [42]. This tunneling current passing the whole memory cell stack resembles FG cells in the program saturation regime as described in Sect. 5.2.3. However, the program operation is generally not the problem of CT cells, since usually high V_{th} levels (even suitable for MLC) can be reached.

One of the major issues of SONOS memory cells is the erase. It can be observed that the erasability of SONOS cells significantly deteriorates when the tunnel oxide thickness is increased above 2 nm [43]. In the TOX thickness range up to 2 nm the erase mechanism is based on direct tunneling of holes from the channel region to the SiN CTL. For thicker tunnel dielectric layers the direct tunneling probability is significantly reduced and for an efficient erase operation the electric field strength needs to be increased up to the Fowler-Nordheim tunneling regime. The problem that occurs in SONOS cells with thick tunnel oxide under FN erase conditions is the so-called erase saturation which is illustrated in Fig. 5.33a. Under FN tunneling conditions for holes from the cell channel, the electric field in the top SiO_2 (blocking oxide: BLOX) layer is already high enough to inject electrons from the gate towards the storage SiN (back tunneling). These injected gate electrons compensate the positive charge of the injected holes and stop the V_{th} decrease (erase saturation). Other erase mechanisms which do not suffer from erase saturation, such as hot hole

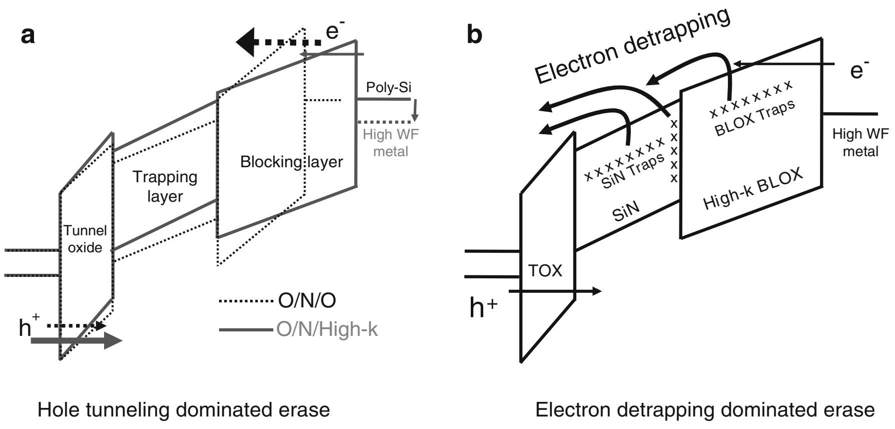


Fig. 5.33 TANOS erase due to reduced electron back tunneling [48]. (**a**) Hole tunneling dominated erase, (**b**) Electron detrapping dominated erase

injection (HHI) [44], are limited to the NOR array structure where NROM-like cells [45] are commercially available, but cannot be implemented in the NAND array.

Erase saturation in planar CT cells can be improved when a gate material with high work function and/or a high-k blocking oxide is used, as shown in Fig. 5.33a. A higher work function can be obtained by a p-doped poly-Si layer instead of the n-doped poly-Si gate [46], or by the use of a high work function metal gate [47]. The combination of both program saturation improvement approaches was the reason for the introduction of so-called TANOS (TaN/Al_2O_3/Si_3N_4/Si) CT memory cells [48]. In the ideal TANOS image, the erase mechanism is solely due to hole tunneling from the channel, the charge is only stored in the SiN CTL, and the Al_2O_3 blocking oxide is assumed to be trap free.

However, there are several indications that the ideal TANOS image is not fully true. Other investigations of the TANOS erase even describe that electron detrapping from SiN traps is the predominant effect [49], as illustrated in Fig. 5.33b.

It was additionally found that the Al_2O_3 BLOX of the TANOS stack is not trap-free and acts as a charge trapping layer as well [41, 42]. Consequently, detrapping from Al_2O_3 traps could be another contribution to the improved erase performance of TANOS memory cells.

The major reason why CT Flash memory cell containing NAND product chips are to date not commercially available is the observation of a general trade-off between erasability and retention of CT memory cells.

Assuming that detrapping is an important component for CT cell erase, this could be principally understood since energetically deep trap levels would be beneficial for a good retention, but hinder the erase, and vice versa.

Compared to FG NAND cells, the retention of TANOS memory cells is generally not sufficient for MLC application. This can be seen for TANOS cells in a 48 nm NAND Flash technology in Fig. 5.34. The TANOS cell (without sealing oxide)

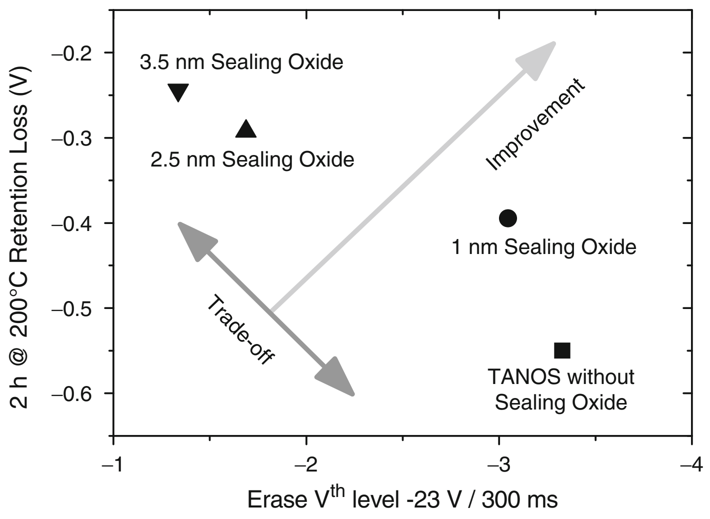


Fig. 5.34 Trade-off between erasability and retention performance for 48 nm TANOS NAND cells with an additional SiO_2 layer at the interface between the SiN charge trapping layer and the Al_2O_3 blocking layer [42]

shows a good erase level for $V_{ers} = -23$ V with a long $t_{ers} = 300$ ms erase pulse, but the retention loss of nearly 550 mV after a 2 h retention bake at 200°C is not suitable for MLC. This high retention loss is most likely due to a combination of electrons lost from the storage SiN due to hopping conduction over Al_2O_3 traps and a direct charge loss of electrons stored in Al_2O_3 BLOX traps. Figure 5.34 shows the retention improvements at the expense of erase performance when parts of the Al_2O_3 BLOX adjacent to the SiN charge trapping layer are replaced by an SiO_2 layer (sealing oxide) with identical electrical thickness (EOT). The reduction of the retention loss to 250 mV for the 3.5 nm sealing oxide results in CT TAONOS ($TaN/Al_2O_3/SiO_2/Si_3N_4/Si$) cells that can hardly be erased below $V_{th} = -1$ V (both values are critical for MLC).

A similar trade-off between erase performance and retention was obtained from large area CT memory cells in the μm range, where the SiN CT composition was varied with respect to the Si content [50] (see Fig. 5.35a), or with an additional high-k BLOX layer, introduced on top of the Al_2O_3 to reduce gate back tunneling during erase [51] (see Fig. 5.35b). In all cases shown in Fig. 5.35a, b, the standard TANOS cell behavior is among the best performing CT cells, or only the described trade-off between retention and erase performance is seen.

The endurance behavior of TANOS or similar CT cells is also generally worse than that of floating gate cells. This might be correlated to the inevitable tunnel currents through the hole CT stack as mentioned before.

Besides, the charge storage in a non-conducting layer can lead to inhomogeneously distributed charges which adversely affect the erase performance of CT

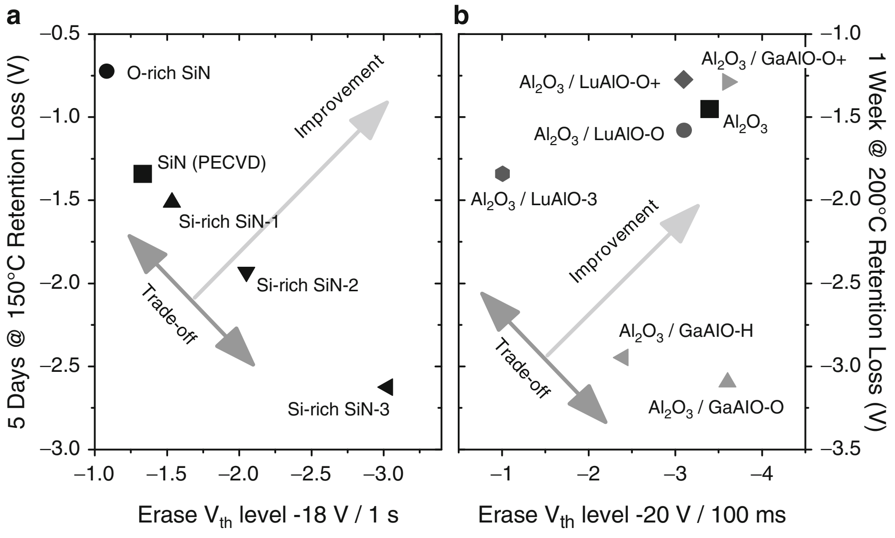


Fig. 5.35 TANOS trade-off between erasability and retention performance on large memory cells (μm range) with variation of the Si content in the SiN CTL (**a**) [50], and for different high-k layers on top of the Al_2O_3 blocking oxide (**b**) [51]

cells [52, 53] and can also be responsible for the worse retention performance of small ground rule CT cells compared to large CT cells [42, 54].

All described reliability issues (erase performance, retention, and endurance) of CT memory cells are responsible for the fact that TANOS cells have so far not been able to replace floating gate cells in NAND Flash applications.

The traditional SONOS CT memory cell with thick bottom oxide, which is able to fulfill the required retention criteria, may experience a revival in combination with the FN erase mechanism when realized in a cylindrical cell geometry, as will be discussed in the following section.

5.7 3D Memory Cell Integration for Future Mass Storage Applications

In the last few sections it was discussed that planar memory cells (either FG or CT) constitute one option for extending the shrink roadmap of conventional floating gate NAND memory cells with a control gate wrapped around the floating gate. However, the planar shrinking will end somewhere around the 10 nm technology node. At these dimensions the number of atom layers is already countable. Further scaling is limited due to the need for conducting and isolating layers to build memory cells.

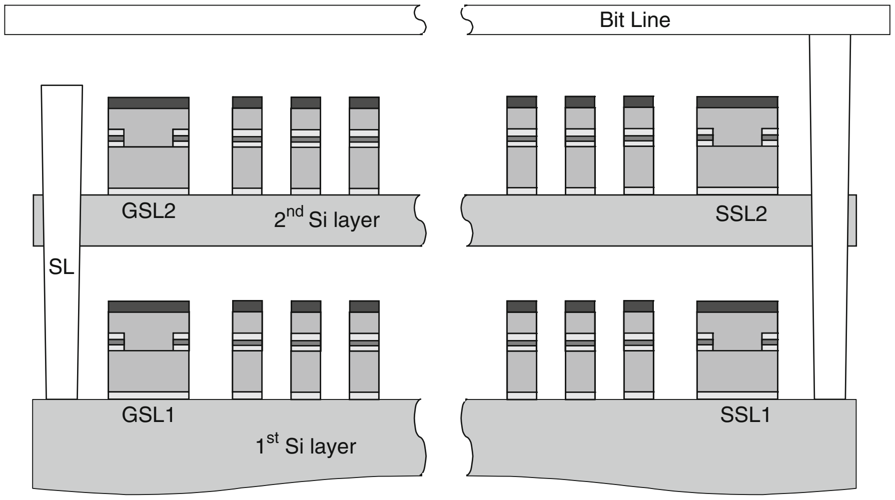


Fig. 5.36 Stacking of multiple NAND layers one on top of the other as presented for TANOS CT cells [55] and FG gate memory cells [56]

Each layer dimension in horizontal direction is already in the direct tunneling regime at such small feature sizes which is contradictory to non-volatile charge storage.

The only option for a continued memory density increase is the introduction of 3D memory cell arrangements.

Examples of 3D stacked NAND memory cell arrangements were presented with an integration of two TANOS layers [55] and two floating gate layers [56]. The schematic two layer stacked NAND structure is shown in Fig. 5.36. Both memory layers use a common source line and a common bit line. The process sequence for generating these two NAND Flash layers on top of each other is the consecutive processing of two times almost the whole NAND process. Therefore this approach is very complex and nearly doubles the number of process steps. It additionally requires the use of epi-Si growth in order to form the channel region of the top memory cell level.

The problem with this kind of 3D stacking approach is that each doubling of the number of stacked layers nearly doubles the processing cost and eventually requires additional space for the WL decoders of the different levels [56]. It also becomes more critical to activate the cell transistor junction implants of higher cell levels since the acceptable thermal budget of the lower levels is very limited once completed.

For these reasons it appears more attractive to think of 3D memory cell integration schemes, where a larger number of layers can be processed at once.

The schematic structure of a 3D memory cell array [57] that follows this approach is shown in Fig. 5.37. Examples of this kind of array structure are the p-BICS (pipe-shaped Bit Cost Scalable) Flash approach [58] and the TCAT (Terabit Cell Array Transistor) technology [59].

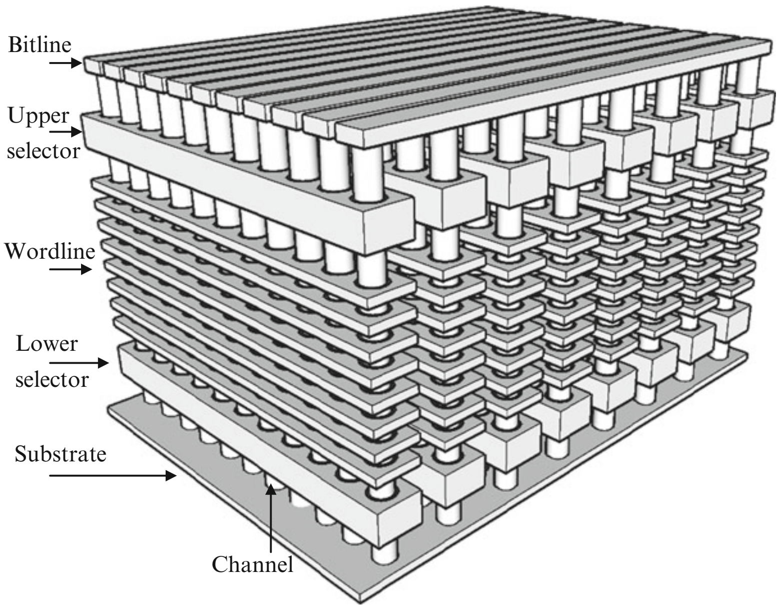


Fig. 5.37 Schematic structure of a vertical 3D memory array [57] with CT cells as used in p-BICS [58] and TCAT [59]

The starting point for both concepts is a multiple stack of conducting and isolating, or two different conducting layers. The channel regions are generated in the vertical direction subsequent to a vertical punch etch process step through all layers.

In the p-BICS the memory cell transistors are generated in holes vertically etched into the word line planes. In order to be able to separately operate all memory cells it is necessary to separate the word lines into slices by vertical slits. One of the major challenges is to generate the CT memory cells into the etched vertical holes.

Consequently, the CT cell formation process is carried out in a reversed order since the deposition of the charge trapping stack is performed before the poly-Si channel formation. The principle challenge of this reversed processing order is the quality of the channel Si and the tunnel oxide of the CT cell, which in a planar cell integration is a high quality bulk Si channel and a high quality thermal SiO_2.

The TCAT technology is in principle similar to p-BICS. Due to a special process integration sequence, it can be achieved in TCAT that an active CTL is only present in the memory transistor region. This measure avoids lateral charge movement and therefore principally improves the retention [60]. The cylindrical shape of the CT cells in the p-BICS and TCAT 3D array approaches have one major advantage over planar CT cells, namely the electric field enhancement in the TOX and the field reduction in the BLOX [61]. The band diagram and the electric fields under erase conditions ($V_{CG,ers} = -20$ V) for a planar MONOS cell vs. a cylindrical

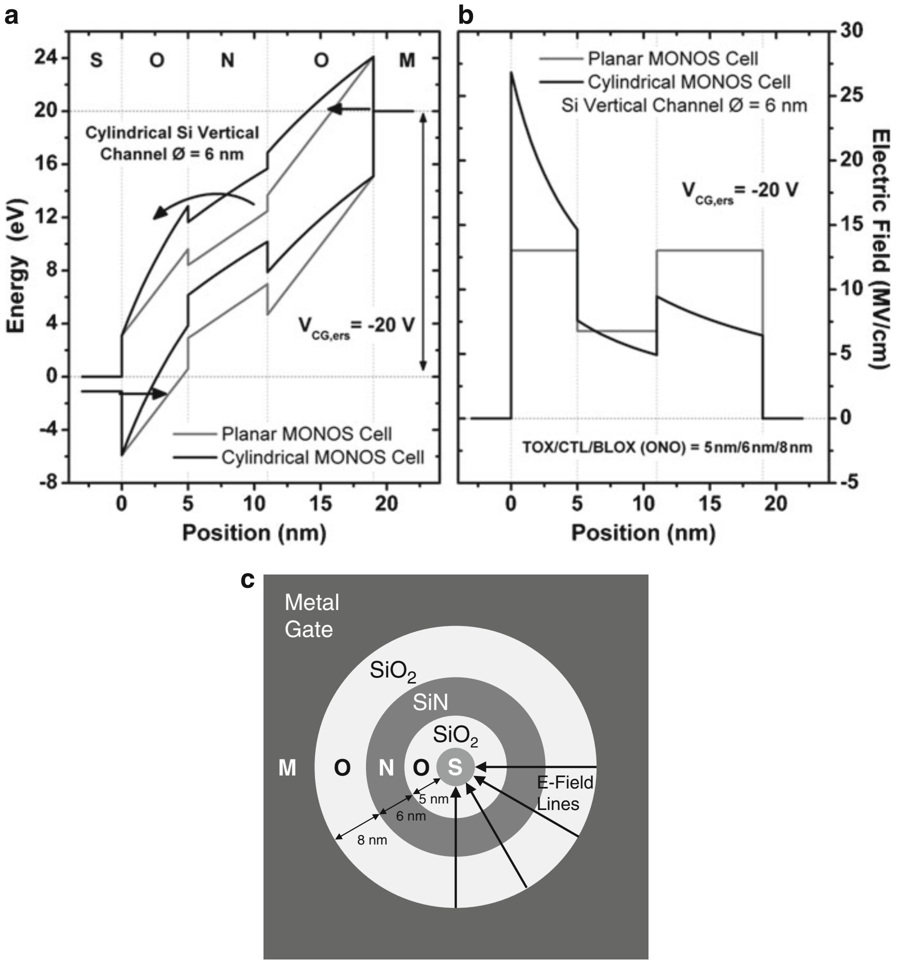


Fig. 5.38 Comparison of uncharged (no electrons stored in the SiN CTL) planar MONOS cells and cylindrical MONOS cells with an inner Si channel diameter of 6 nm. The band diagram (**a**) and the electric field conditions (**b**) show strongly increased fields in the tunnel oxide and significantly reduced fields in the BLOX of the cylindrical SONOS cell. The SONOS CT stack dimension (ONO) used in the simulations was TOX/CTL/BLOX = 5 nm/6 nm/8 nm (**c**)

MONOS with a 6 nm inner-channel diameter are shown in Fig. 5.38a, b. The ONO stack dimensions used in the field calculations were $t_{TOX} = 5$ nm, $t_{SiN} = 6$ nm, and $t_{BLOX} = 8$ nm. It is clearly visible that the cylindrical cell geometry with an inner cell channel position strongly increases the TOX field in relation to the BLOX field. Therefore, the cylindrical geometry effectively acts as an increased gate coupling ratio of a floating gate cell. The TOX electric field enhancement can also be seen in the form of denser E-field lines in Fig. 5.38c. Consequently, it could be possible to

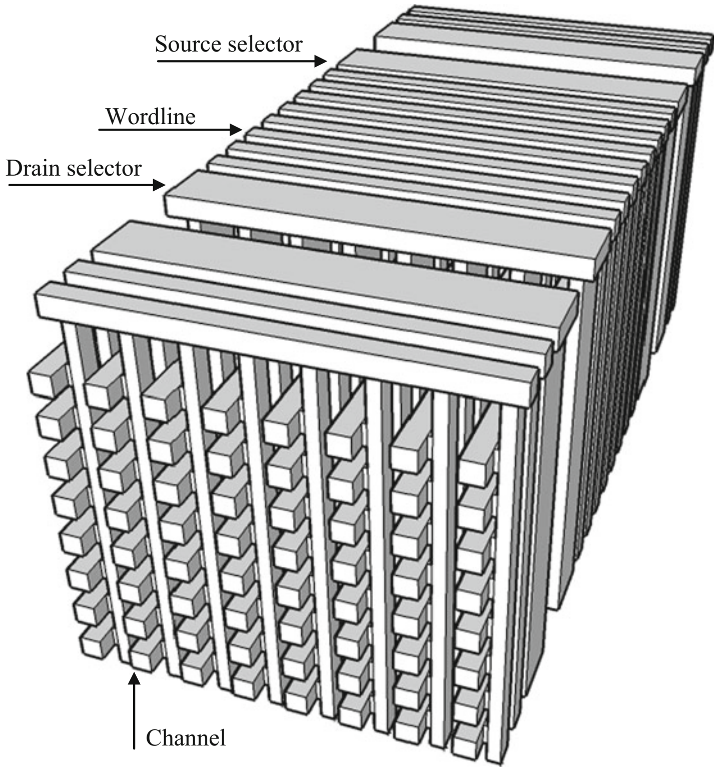


Fig. 5.39 Schematic structure of vertical gate 3D CT memory arrays [57] as used in VG TFT NAND [62] and VG-NAND [63]

use high-quality deposited SiO_2 as BLOX material in cylindrical CT cells instead of Al_2O_3 and by this means improve the CT charge retention without deterioration of the erase performance.

Other 3D array integration approaches use vertical gates (Vertical Gate Thin Film Transistor (VG TFT) NAND [62] and VG NAND [63]) or another option to generate vertical memory transistor channels (Vertical-Stacked-Array-Transistor (VSAT) [64]). Both concepts use a multiple layer of deposited conducting (poly-Si, or metal) and isolating layers patterned in stripes. The CT layer is deposited at the side walls of this multiple line structure, and perpendicular to these lamellar structures, an equal line and space conducting structure which is used as side gates in VG NAND (see Fig. 5.39) and as a side wall channel in VSAT, is generated.

Generally speaking, the VG TFT NAND [62], VG NAND [63], and VSAT [64] concepts are 3D arrays based on planar CT cells. Therefore, the memory cells in these concepts suffer in principle from the same erasability and retention problems as the one layer planar CT NAND cells discussed in Sect. 5.6.

Another important criterion is the minimal final footprint of one layer of the 3D integrated memory arrays. Compared to p-BiCS, TCAT, and VSAT with vertical channel direction, the 3D array option with planar devices (channel direction parallel to the substrate plane) and with vertical gates (VG NAND [62, 63]) have smaller footprints and seem to scale to smaller half pitches (F) [65].

However, the 3D cell array options presented so far are all based on CT cells which have never yet been in production in planar NAND strings.

For this reason it can be easily understood that NAND memory producers would principally aim to continue the use of the approved floating gate cell approach in 3D arrays. One option for integrating FG cells into 3D vertical strings was presented in [66]. The proposed 3D memory cell integration concept is very complex as evident in previous CT 3D memories. It additionally generates the poly-Si channel after the IPD, FG and TOX formation. This reversed processing order again yields the risk of low TOX quality and reduced retention and endurance performance.

Whether one of the numerous proposed 3D cell integration concepts will finally be able to extend the memory density scaling of planar floating gate NAND Flash will strongly depend on the memory array functionality, reliability and manufacturability. Especially the strong difference between MOSFET-based 3D memory cell integration and planar cell integration with decades of process experience will make it challenging to establish a yielding 3D NAND chip production.

Acknowledgment Part of the work for this chapter was financially supported by the European Commission in the FP7 project GOSSAMER (project contract No. 214431) and by the Federal Ministry of Education and Research of the Federal Republic of Germany in the HELEKTRON project (project No. 01 M316). The author would like to acknowledge the whole Flash development team of the former Qimonda Company. Special thanks are addressed to Torsten Müller, Nigel Chan, and Stefano Parascandola for discussions, provision of a 3D FG cell field simulator script, RTN measurements, program saturation evaluations, and pitch fragmentation process images.

References

1. D. Kahng, S.M. Sze, A floating gate and its application to memory devices. Bell. Syst. Techn. J. **46**(6), 1288–1295 (1967)
2. N. Chan, M.F. Beug, R. Knoefler, T. Mueller, T. Melde, M. Ackermann, S. Riedel, M. Specht, C. Ludwig, A.T. Tilke, Metal control gate for sub-30 nm floating gate NAND memory, in *Proceedings of the 9th NVMTS*, Pacific Grove, Nov 2008, pp. 82–85
3. A. Kolodny, S.T.K. Nieh, B. Eitan, J. Shappir, Analysis and modelling of floating gate EEPROM cells. IEEE Trans. Electron Devices **33**(6, June), 835–844 (1986)
4. K. Kim, J. Choi, Future outlook of NAND flash technology for 40 nm node and beyond, in *Non-Volatile Semiconductor Memory Workshop, Monterey, USA. IEEE NVSMW 2006. 21st*, 2006, pp. 9–11
5. M. Wong, D.K.-Y. Liu, S.S.-W. Huang, Analysis of the subthreshold slope and the linear transconductance techniques for the extraction of the capacitance coupling coefficients of floating gate devices. IEEE Electron Device Lett. **13**(11, November), 566–568 (1992)
6. M.F. Beug, Q. Rafhay, M.J. van Duuren, R. Duane, Investigation of back-bias capacitance coupling coefficient measurement methodology for floating gate non-volatile memory cells. IEEE Trans. Electron Devices **57**(6, June), 1253–1260 (2010)

7. R.H. Fowler, L. Nordheim, Electron emission in intense electric films. Proc. R. Soc. Lond. **119**, 173–181 (1928)
8. K.-D. Suh, B.-H. Suh, Y.-H. Um, J.-K. Kim, Y.-J. Choi, Y.-N. Koh, S.-S. Lee, S.-C. Kwon, B.-S. Choi, J.-S. Yum, J.-H. Choi, J.-R. Kim, H.-K. Lim, A 3.3 V 32 Mb NAND flash memory with incremental step pulse programming scheme, in *IEEE International Solid-State Circuits Conference*, San Francisco, USA, Feb 1995, pp. 128–129
9. C. Friederich, J. Hayek, A. Kux, T. Muller, N. Chan, G. Kobernik, M. Specht, D. Richter, D. Schmitt-Landsiedel, Novel model for cell – system interaction (MCSI) in NAND flash, in *IEEE International Electron Devices Meeting (IEDM)*, San Francisco, USA, Dec 2008, pp. 831–834
10. K. Prall, K. Parat, 25 nm 64Gb MLC NAND technology and scaling challenges, in *IEEE International Electron Devices Meeting (IEDM)*, San Francisco, USA, Dec 2010, pp. 102–105
11. C.-H. Lee, S.-K. Sung, D. Jang, S. Lee, S. Choi, J. Kim, S. Park, M. Song, H.-C. Baek, E. Ahn, J. Shin, K. Shin, K. Min, S.-S. Cho, C.-J. Kang, J. Choi, K. Kim, J.-H. Choi, K.-D. Suh, T.-S. Jung, A highly manufacturable integration technology for 27 nm 2 and 3bit/cell NAND flash memory, in *IEEE International Electron Devices Meeting (IEDM)*, San Francisco, USA, Dec 2010, pp. 98–101
12. D.J. DiMaria, E. Cartier, Mechanism for stress-induced leakage current in thin silicon dioxide films. J. Appl. Phys. **78**(6, September), 3883–3894 (1995)
13. M.F. Beug, N. Chan, T. Hoehr, L. Mueller-Meskamp, M. Specht, Investigation of program saturation in scaled interpoly dielectric floating gate memory devices. IEEE Trans. Electron Devices **56**(8, August), 1698–1704 (2009)
14. D. Wellekens, J. De Vos, J. Van Houdt, K. van der Zanden, Optimization of Al_2O_2 interpoly dielectric for embedded flash memory applications, in *Proceedings of the Joint NVSMW/ICMTD*, Opio, France, May 2008, pp. 12–15
15. T.-S. Jung, Y.-J. Choi, K.-D. Suh, B.-H. Suh, J.-K. Kim, Y.-H. Lim, Y.-N. Koh, J.-W. Park, K.-J. Lee, J.-H. Park, K.-T. Park, J.-R. Kim, J.-H. Yi, H.-K. Lim, A 117-mm2 3.3-V only 128-Mb multilevel NAND flash memory for mass storage applications. IEEE J. Solid-State Circuits **31**(11, November), 1575–1583 (1996)
16. T. Cho, Y.-T. Lee, E.-C. Kim, J.-W. Lee, S. Choi, S. Lee, D.-H. Kim, W.-G. Han, Y.-H. Lim, J.-D. Lee, J.-D. Choi, K.-D. Suh, A dual-mode NAND flash memory: 1-Gb multilevel and high-performance 512-Mb single-level modes. IEEE J. Solid-State Circuits **36**(11, November), 1700–1706 (2001)
17. R. Cernea, D.J. Lee, M. Mofidi, E.Y. Chang, Wy-Yi Chien, L. Goh, Y. Fong, J.H. Yuan, G. Samachisa, D.C. Guterman, S. Mehrotra, K. Sato, H. Onishi, K. Ueda, F. Noro, K. Mijamoto, M. Morita, K. Umeda, and K. Kubo, "A 34 Mb 3.3 V serial flash EEPROM for solid-state disk applications," IEEE International Solid-State Circuits Conference (ISSCC), San Francisco, Feb 1995, pp. 126–127
18. R. Micheloni, L. Crippa, A. Marelli, *Inside NAND Flash Memories* (Springer, New York, Berlin, 2010)
19. A. Chimenton, P. Olivo, Fast identification of critical electrical disturbs in nonvolatile memories. IEEE Trans. Electron Devices **54**(9), 2438–2444 (Sept 2007)
20. J.H. Stathis, Reliability limits for the gate insulator in CMOS technology. IBM J. Res. Dev. **46**(2/3), 265–286 (March/May 2002)
21. P. Olivo, T.N. Nguyen, B. Ricco, High-field-induced degradation in ultrathin SiO2 films. IEEE Trans. Electron Devices **35**(12, December), 2259–2267 (1988)
22. S. Lai, Electrical properties of nitrided-oxide systems for use in gate dielectrics and EEPROM, in *Proceedings of the International Non-Volatile Memory Technology Conference*, Albuquerque, 1998, pp. 6–7
23. J. Hwang, J. Seo, Y. Lee, S. Park, J. Leem, J. Kim, T. Hong, S. Jeong, K. Lee, H. Heo, H. Lee, P. Jang, K. Park, M. Lee, S. Baik, J. Kim, H. Kkang, M. Jang, J. Lee, G. Cho, J. Lee, B. Lee, H. Jang, S. Park, J. Kim, S. Lee, S. Aritome, S. Hong, S. Park, A Middle-1X nm NAND flash memory cell (M1X-NAND) with highly manufacturable integration technologies, in *IEEE International Electron Devices Meeting (IEDM)*, Washington, USA, Dec 2011, pp. 199–202

24. U. Ganguly, Y. Yokota, T. Jing, S. Shiyu, M. Rogers, J. Miao, K. Thadani, H. Hamana, L. Garlen, B. Chandrasekaran, S. Thirupapuliyur, C. Olsen, V. Nguyen, S. Srinivasan, Scalability enhancement of FG NAND by FG shape modification, in *IEEE International Memory Workshop (IMW)*, Seoul, South Korea, May 2010
25. D. Wellekens, J. De Vos, J. Van Houdt, K. van der Zanden, Optimization of Al_2O_3 interpoly dielectric for embedded flash memory applications, in *Proceedings of the Joint NVSMW/ICMTD*, Opio, France, May 2008, pp. 12–15
26. P. Blomme, M. Rosmeulen, A. Cacciato, M. Kostermans, C. Vrancken, S. Van Aerde, T. Schram, I. Debusschere, M. Jurczak, J. Van Houdt, Novel dual layer floating gate structure as enabler of fully planar flash memory, in *Symposium on VLSI Technology (VLSIT)*, Honolulu, Hawaii, June 2010, pp. 129–130
27. J.-D. Lee, S.-H. Hur, J.-D. Choi, Effects of floating-gate interference on NAND flash memory cell operation. IEEE Electron Device Lett. **23**(5, May), 264–266 (2002)
28. M.F. Beug, S. Parascandola, T. Hoehr, T. Muller, R. Reichelt, L. Muller-Meskamp, P. Geiser, T. Geppert, L. Bach, U. Bewersdorff-Sarlette, O. Kenny, S. Brandl, T. Marschner, S. Meyer, S. Riedel, M. Specht, D. Manger, R. Knofler, K. Knobloch, P. Kratzert, C. Ludwig, K.-H. Kusters, Pitch fragmentation induced odd/even effects in a 36 nm floating gate NAND technology, in *Proceedings of the NVMTS*, Pacific Grove, USA, Nov 2008, pp. 77–81
29. N. Shibata, H. Maejima, K. Isobe, K. Iwasa, M. Nakagawa, M. Fujiu, T. Shimizu, M. Honma, S. Hoshi, T. Kawaai, K. Kanebako, S Yoshikawa, H. Tabata, A. Inoue, T. Takahashi, T. Shano, Y. Komatsu, K. Nagaba, M. Kosakai, N. Motohashi, K. Kanazawa, K. Imamiya, H. Nakai, A 70 nm 16Gb 16-level-cell NAND flash memory, in *IEEE Symposium on VLSI Circuits*, Kyoto, Japan, 14–16 June 2007, pp. 190–191
30. R. Cernea, L. Pham, F. Moogat, S. Chan, B. Le, Y. Li, S. Tsao, T.-Y. Tseng, K. Nguyen, J. Li, J. Hu, J. Park, C. Hsu, F. Zhang, T. Kamei, H. Nasu, P. Kliza, K. Htoo, J. Lutze, Y. Dong, M. Higashitani, J. Yang, H.-S. Lin, V. Sakhamuri, A. Li, F. Pan, S. Yadala, S. Taigor, K. Pradhan, J. Lan, J. Chan, T. Abe, Y. Fukuda, H. Mukai, K. Kawakamr, C. Liang, T. Ip, S.-F. Chang, J. Lakshmipathi, S. Huynh, D. Pantelakis, M. Mofidi, K. Quader, A 34 MB/s-Program-Throughput 16Gb MLC NAND with All-Bitline Architecture in 56 nm, in *IEEE International Solid-State Circuits Conference (ISSCC)*, San Francisco, USA, Feb 2008, pp. 420–624
31. Y.S. Kim, D.J. Lee, C.K. Lee, H.K. Choi, S.S. Kim, J.H. Song, D.H. Song, J.-H. Choi, K.-D. Suh, C. Chung, New scaling limitation of the floating gate cell in NAND flash memory, in *IEEE International Reliability Physics Symposium (IRPS)*, Anaheim, USA, May 2010, pp. 599–603
32. H.H. Mueller, D. Wörle, M. Schulz, Evaluation of the coulomb energy for single-electron interface trapping in sub-μm metal-oxide-semiconductor field effect transistors. J. Appl. Phys. **75**(6, March), 2970–2979 (1994)
33. H. Miki, T. Osabe, N. Tega, A. Kotabe, H. Kurata, K. Tokami, Y. Ikeda, S. Kamohara, R. Yamada, Quantitative analysis of random telegraph signals as fluctuations of threshold voltages in scaled flash memory cells, in *IEEE International Reliability Physics Symposium (IRPS)*, Phoenix, USA, 2007, pp. 29–35
34. K. Seidel, R. Hoffmann, D. A. Löhr, T. Melde, M. Czernohorsky, J. Paul, M. F. Beug, V. Beyer, Comparison and analysis of trap mechanisms responsible for random telegraph noise and erratic programming on sub-50 nm floating gate flash memories, in *Non-Volatile Memory Technology Symposium (NVMTS)*, Portland, USA, Oct 2009 pp. 67–71
35. M.F. Beug, R. Ferretti, K.R. Hofmann, Analysis and modeling of the transient local tunneling in gate oxides. IEEE Trans. Device Mater. Reliab. **4**(1, March), 73–79 (2004)
36. M.C. Chiu, B. Szu-M. Lin, M.F. Tsai, Y.S. Chang, M.H. Yeh, T.H. Ying, C. Ngai, J. Jin, S. Yuen, S. Huang, Y. Chen, L. Miao, K. Tai, A. Conley, I. Liu, Challenges of 29 nm half-pitch NAND Flash STI patterning with 193 nm dry lithography and self-aligned double patterning. Proc. SPIE **7140**, Taipei, Taiwan, November 2008, 714021 (2008) doi:10.1117/12.804685
37. P. Xu, Y. Chen, Y. Chen, L. Miao, S. Sun, S.-W. Kim, A. Berger, D. Mao, C. Bencher, R. Hung, C. Ngai, Sidewall spacer quadruple patterning for 15 nm half-pitch. Proc. SPIE **7973**, San Jose, USA, Feb. 2011, 79731Q (2011) doi:10.1117/12.881547

38. C. Bencher, Y. Chen, H. Dai, W. Montgomery, L. Huli, 22 nm half-pitch patterning by CVD spacer self alignment double patterning (SADP). Proc. SPIE **6924**, San Jose, USA, Feb. 2008, 69244E (2008), doi:10.1117/12.772953
39. C. Ludwig, S. Meyer, Double patterning for memory ICs, in Recent Advances in Nanofabrication Techniques and Applications, ed. B. Cui, ISBN: 978-953-307-602-7, InTech, pp. 417–432 (2011), Available from http://www.intechopen.com/articles/show/title/double-patterning-for-memory-ics
40. S. Aritome, S. Satoh, T. Maruyama, H. Watanabe, S. Shuto, G.J. Hemink, R. Shirota, S. Watanabe, F. Masuoka, A 0.67 μm2 self-aligned shallow trench isolation cell (SA-STI cell) for 3 V-only 256 Mbit NAND EEPROMs, in *IEEE International Electron Devices Meeting(IEDM)*, San Francisco, USA, Dec 1994, pp. 61–64
41. M. Specht, H. Reisinger, F. Hofmann, T. Schulz, E. Landgraf, R.J. Luyken, W. Rösner, M. Grieb, L. Risch, Charge trapping memory structures with Al_2O_3 trapping dielectric for high-temperature applications. Solid-State Electron. **49**(5, May), 716–720 (2005)
42. M.F. Beug, T. Melde, M. Czernohorsky, R. Hoffmann, J. Paul, R. Knoefler, A.T. Tilke, Analysis of TANOS memory cells with sealing oxide containing blocking dielectric. IEEE Trans. Electron Devices **57**(7, July), 1590–1596 (2010)
43. R. van Schaijk, M. van Duuren, W.Y. Mei, K. van der Jeugd, A. Rothschild, M. Demand, Oxide–nitride–oxide layer optimisation for reliable embedded SONOS memories. Microelectron. Engineering **72**(1–4, April), 395–398 (2004)
44. T.Y. Chan, K.K. Young, C. Hu, A true single-transistor oxide-nitride-oxide EEPROM device. IEEE Electron Device Lett. **8**(3, March), 93–95 (1987)
45. B. Eitan, P. Pavan, I. Bloom, E. Aloni, A. Frommer, D. Finzi, NROM: A novel localized trapping, 2-bit nonvolatile memory cell. IEEE Electron Device Lett. **21**(11, November), 543–545 (2000)
46. H. Bachhofer, H. Reisinger, E. Bertagnolli, Transient conduction in multidielectric silicon–oxide–nitride–oxide semiconductor structures. J. Appl. Phys. **89**(5, March), 2791–2800 (2001)
47. A. Goda, M. Noguchi, Improvement of erase saturation for a highly reliable MONOS memory cell, in *IEEE Non-Volatile Semiconductor Memory Workshop (NVSMW)*, Monterey, USA, Feb 2003, pp. 65–68
48. C.H. Lee, K.I. Choi, M.K. Cho, Y.H. Song, K.C. Park, K. Kim, A novel SONOS structure of SiO2-SiN-Al_2O_3 with TaN metal gate for multi-giga bit flash memories, in *IEEE International Electron Devices Meeting (IEDM)*, Washington, USA, Dec 2003, pp. 613–616
49. S.-C. Lai, H.-T. Lue, J.-Y. Hsieh, M.-J. Yang, Y.-K. Chiou, C.-W. Wu, T.-B. Wu, G.-L. Luo, C.-H. Chien, E.-K. Lai, K.-Y. Hsieh, R. Liu, C.-Y. Lu, Study of the erase mechanism of MANOS (metal/Al_2O_3/SiN/SiO_2/Si) device. IEEE Electron Device Lett. **28**(7, July), 643–645 (2007)
50. G. Van den bosch, A. Furnemont, M.B. Zahid, R. Degraeve, L. Breuil, A. Cacciato, A. Rothschild, C. Olsen, U. Ganguly, J. Van Houdt, Nitride engineering for improved erase performance and retention of TANOS NAND flash memory, in *Proceedings of the Joint NVSMW/ICMTD 2008*, Opio, France, 18–22 May 2008, Joint, pp. 128–129
51. L. Breuil, C. Adelmann, G. Van Den Bosch, A. Cacciato, M.B. Zahid, M. Toledano-Luque, A. Suhane, A. Arreghini, R. Degraeve, S. Van Elshocht, I. Debusschere, J. Kittl, M. Jurczak, J. Van Houdt, Optimization of the crystallization phase of Rare-Earth aluminates For blocking dielectric application in TANOS type flash memories, in *Proceedings of Solid-State Device Research Conference (ESSDERC)*, Sevilla, Spain, 14–16 Sept 2010 pp. 440–443
52. M.F. Beug, T. Melde, M. Isler, L. Bach, M. Ackermann, S. Riedel, K. Knobloch, C. Ludwig, Anomalous erase behavior in charge trapping memory cells, in *Proceedings of the Joint NVSMW/ICMTD*, Opio, France, May 2008, pp. 121–123
53. Y.-J. Chen, L. H. Chong, S.-W. Lin, T.-H. Yeh, K.-F. Chen, J.-S. Huang, C.-H. Cheng, S.-H. Ku, N.-K. Zous, I-J. Huang, T.-T. Han, T.-H. Hsu, H.-T. Lue, M.-S. Chen, W.-P. Lu, K.-C. Chen, C.-Y. Lu, Source/Drain dopant concentration induced reliability issues in charge trapping NAND flash cells, in *IEEE International Reliability Physics Symposium (IRPS)*, Anaheim, USA, May 2010, pp. 634–638

54. M.F. Beug, T. Melde, J. Paul, R. Knoefler, TaN and Al_2O_3 side wall gate-etch damage influence on program, erase, and retention of sub-50 nm TANOS NAND flash memory cells. IEEE Trans. Electron Devices **58**(6, June), 1728–1734 (2011)
55. S.-M. Jung, J. Jang, W. Cho, H. Cho, J. Jeong, Y. Chang, J. Kim, Y. Rah, Y. Son, J. Park, M.-S. Song, K.-H. Kim, J.-S. Lim, K. Kim, Three dimensionally stacked NAND flash memory technology using stacking single crystal Si layers on ILD and TANOS structure for beyond 30 nm node, in *International Electron Devices Meeting (IEDM)*, San Francisco, USA, Dec 2006
56. K.-T. Park, M. Kang, S. Hwang, D. Kim, H. Cho, Y. Jeong, Y.-I. Seo, J. Jang, H.-S. Kim, Y.-T. Lee, S.-M. Jung, C. Kim, A fully performance compatible 45 nm 4-gigabit three dimensional double-stacked multi-level NAND flash memory with shared bit-line structure. IEEE J. Solid-State Circuits **44**(1, January), 208–216 (2009)
57. R. Micheloni, L. Crippa, A. Grossi, P. Tessariol, Chapter 6, in *Memory Mass Storage* (Springer, 2011)
58. R. Katsumata, M. Kito, Y. Fukuzumi, M. Kido, H. Tanaka, Y. Komori, M. Ishiduki, J. Matsunami, T. Fujiwara, Y. Nagata, L. Zhang, Y. Iwata, R. Kirisawa, H. Aochi, Pipe-shaped BiCS flash memory with 16 stacked layers and multi-level-cell operation for ultra high density storage devices, in *Symposium on VLSI Technology*, Kyoto, Japan, June 2009, pp. 136–137
59. J. Jang, H.-S. Kim, W. Cho, H. Cho, J. Kim, S.I. Shim, Y. Jang, J.-H. Jeong, B.-K. Son, D.W. Kim, K. Kim, J.-J. Shim, J.S. Lim, K.-H. Kim, S.Y. Yi, J.-Y. Lim, D. Chung, H.-C. Moon, S. Hwang, J.-W. Lee, Y.-H. Son, U-In Chung, W.-S. Lee, Vertical cell array using TCAT (Terabit Cell Array Transistor) technology for ultra high density NAND flash memory, in *Symposium on VLSI Technology*, Kyoto, Japan, June 2009, pp. 192–193
60. J.S. Sim, J. Park, C. Kang, W. Jung, Y. Shin, J. Kim, J. Sel, C. Lee, S. Jeon, Y. Jeong, Y. Park, J. Choi, W.-S. Lee, Self aligned trap-shallow trench isolation scheme for the reliability of TANOS (TaN/AlO/SiN/Oxide/Si) NAND flash memory, in *IEEE Non-Volatile Semiconductor Memory Workshop (NVSMW)*, Monterey, USA, Aug 2007, pp. 110–111
61. E. Nowak, A. Hubert, L. Perniola, T. Ernst, G. Ghibaudo, G. Reimbold, B. De Salvo, F. Boulanger, In-depth analysis of 3D silicon nanowire SONOS memory characteristics by TCAD simulations, in *IEEE International Memory Workshop*, (IMW), Seoul, South Korea, May 2010
62. H.-T. Lue, T.-H. Hsu, Y.-H. Hsiao, S.P. Hong, M.T. Wu, F.H. Hsu, N.Z. Lien, S.-Y. Wang, J.-Y. Hsieh, L.-W. Yang, T. Yang, K.-C. Chen, K.-Y. Hsieh, C.-Y. Lu, A highly scalable 8-layer 3D vertical-gate (VG) TFT NAND flash using junction-free buried channel BE-SONOS device, in *Symposium on VLSI Technology*, Honolulu, Hawaii, USA, June 2010, pp. 131–132
63. W. Kim, S. Choi, J. Sung, T. Lee, C. Park, H. Ko, J. Jung, I. Yoo, Y. Park, Multi-layered vertical gate NAND flash overcoming stacking limit for terabit density storage, in *Symposium on VLSI Technology*, Kyoto, Japan, June 2009 pp. 188–189
64. J. Kim, A.J. Hong, S.M. Kim, E.B. Song, J.H. Park, J. Han, S. Choi, D. Jang, J.-T. Moon, K.L. Wang, Novel Vertical-Stacked-Array-Transistor (VSAT) for ultra-high-density and cost-effective NAND flash memory devices and SSD (Solid State Drive), in *Symposium on VLSI Technology*, Kyoto, Japan, June 2009, pp. 186–187
65. Y.-H. Hsiao, H.-T. Lue, T.-H. Hsu. K.-Y. Hsieh, C.-Y. Lu, A critical examination of 3D stackable NAND flash memory architectures by simulation study of the scaling capability, in *IEEE International Memory Workshop (IMW)*, Seoul, South Korea, May 2010
66. S.J. Whang, K.H. Lee, D.G. Shin, B.Y. Kim, M.S. Kim, J.H. Bin, J.H. Han, S.J. Kim, B. M. Lee, Y.K. Jung, S.Y. Cho, C.H. Shin, H.S. Yoo, S.M. Choi, K. Hong, S. Aritome, S.K. Park, S.J. Hong, Novel 3-dimensional Dual Control-gate with Surrounding Floating-gate (DC-SF) NAND flash cell for 1 Tb file storage application, in *IEEE International Electron Devices Meeting (IEDM)*, San Francisco, USA, Dec 2010, pp. 668–671

Chapter 6
NAND Flash Design

L. Crippa and R. Micheloni

Abstract A Solid-State-Disk is made up by a Flash controller plus a bunch of NAND Flash devices. This chapter focuses on design aspects of NAND chips. The information stored in each memory cell is fully analog because it is related to the number of electrons stored in the floating gate. When we program, erase or read, electrons must be injected, extracted and counted, respectively. All these operations require a mix of analog and digital circuits that need to be properly and timely driven.

Starting from a generic floorplan of a NAND memory, we guide the reader through the main building blocks. First of all, we describe the logic part of the chip, from the embedded microcontroller, who is in charge of running all the internal algorithms, to the fast DDR interface.

Counting the number of electrons in the floating gate is definitely one of the most challenging task, considering that has to be performed with few transistors: sensing techniques are described in Sect. 6.5.

Programming and erasing floating gate cells require voltages higher than the chip power supply. Therefore, charge pumps are used to generate all the needed voltages within the chip. In multilevel storage, cell's gate biasing voltages need to be very accurate and voltage regulators become a must. All these circuits are described in the High Voltage Management section.

Last but not least, the row decoder is introduced. This circuit has the task of properly biasing each single wordline in the NAND array, transferring the regulated high voltages to the gate of the memory cell.

L. Crippa (✉) • R. Micheloni
Integrated Device Technology, Enterprise Computing Division, Agrate Brianza, Italy
e-mail: luca.crippa@ieee.org; rino.micheloni@ieee.org

6.1 NAND Flash Memories

A NAND chip contains a lot of different circuits, both digital and analog. Figure 6.1 sketches a floorplan of a Flash device. The basic architecture of the NAND array has already been presented in Chap. 2. With reference to Fig. 6.1, the memory array has been split in two independent planes. On the horizontal direction a wordline (WL) is highlighted, while a bitline (BL) is shown in the vertical direction. The Row Decoder is the block in charge of addressing and biasing each single wordline and it is located between the planes. BLs are connected to a sensing circuit (Sense Amp). The purpose of sense amplifiers is to read the analog information stored in the memory cell. In the periphery, we find charge pumps, voltage regulators, reference circuits, digital circuits, and redundancy structures. This chapter gives an overview of all the above mentioned circuits.

6.2 Logic Device View

Let's start our analysis from the peripheral circuits. First of all, we have the "Logic", a set of digital gates which enables the communication to the external host and manages data inside the device. In other words, it is the real brain of the memory.

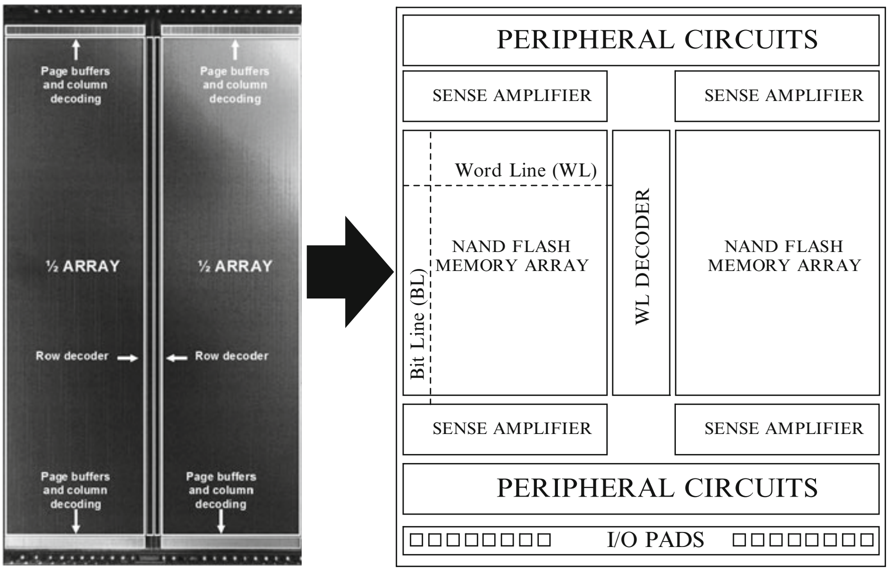


Fig. 6.1 A typical NAND Flash floorplan [1]

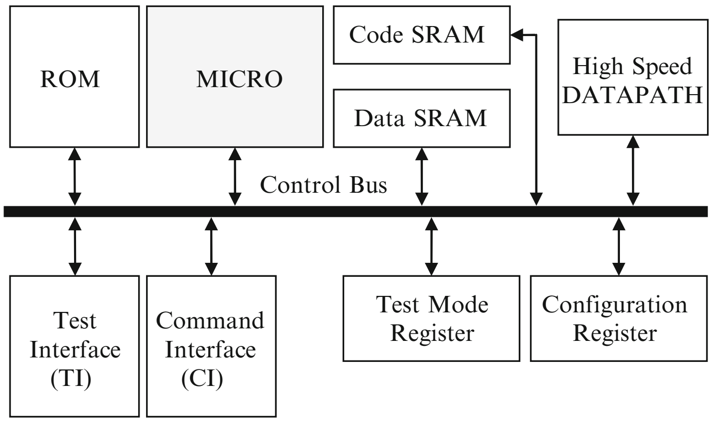


Fig. 6.2 Logic view of a NAND device

We can identify some basic logic blocks, as shown in Fig. 6.2.

1. Control Interface (CI) [2–4]. It is the command interface between the NAND Flash and the external user;
2. Microcontroller. It stores and executes all the internal algorithms, such as read, program, erase and testmode operations.
3. Error Correction Code (ECC) [5] could be embedded in the memory device. ECC improves the reliability of the read operation.
4. Memory testing is a fundamental functionality. For this reason, there is a Test Interface (TI) block, i.e. the interface to the user when device is in test mode.
5. Datapath. Basically, it is the fast link between I/Os and read circuits.
6. There are also a lot of registers, mainly for storing the configurations of the analog circuitry.
7. Redundancy: it can be managed by the microcontroller or it can be implemented as a finite state machine (FSM). This logic is used to increase the wafer yield.

6.2.1 Command Interface

In order to talk with the external user, Flash memory has to understand commands, take data and output data.

The logic block implementing this functionality is basically a finite state machine and is represented by the Command Interface (CI) when the device is in user mode and by the Test Interface (TI) when the device is in test mode.

CI understands legal or illegal command sequences, defined in the device specifications and interacts with other logic blocks as datapath and microcontroller. Control signals have been already described in Chap. 2. CI is composed by a huge

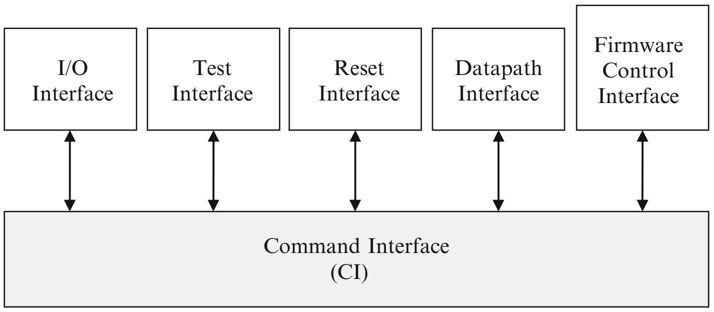


Fig. 6.3 Command interface and its interaction blocks

finite state machine clocked by WE# and driven by all I/O signals such as ALE or CLE. Figure 6.3 represents CI and its interaction blocks.

1. I/Os are all control signals: R/B#, CLE, ALE, WP#, WE#, RE#, CE#, DQ[7:0].
2. Reset Interface exchanges reset information with logic global reset.
3. Datapath interface controls input and output datapaths.
4. Test interface toggles between user mode and test mode.
5. Firmware Control Interface enables microcontroller to execute internal algorithms.

CI is made up by multiple finite state machines, one for each basic function. The Command Interface Controller disables a specific FSM if that specific command is not allowed. During power up, CI Controller disables every commands, so that all the FSMs are disabled too. There is also a FSM that recognizes if a specific command is a read, a program or an erase and enables the correct sub-FSM. Every time the Controller receives an illegal sequence, the device goes into an IDLE state.

When the internal microcontroller executes a specific algorithm, the device is busy. In this situation, the only commands that the CI can accept are a reset and a testmode entry command.

6.2.2 Test Interface

Test Interface (TI) is used when we want to test some particular features, usually not accessible during normal operations (usermode). Test Interface is enabled by a specific command sequence, called testmode entry. Generally speaking, a NAND device can have these modes:

- Usermode that represents the standard functionality, where commands described in the device specification are available;
- Usertestmode that represents the standard functionality plus some particular commands;
- Testmode that is the test operational mode.

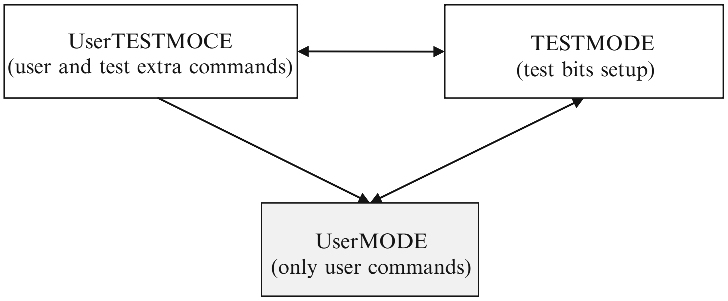


Fig. 6.4 Flow diagram used to change operational modes among usermode, usertestmode and testmode

Fig. 6.5 Testmode registers

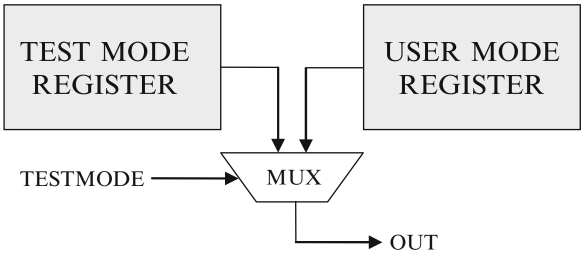


Figure 6.4 represents how it is possible to change the operational modes with proper command sequences recognized by the CI Controller.

Once TI is enabled, it substitutes CI: TI recognizes the command set and drives input and output data/address on the logic bus. Test Interface is allowed to access the different registers and different memory circuits without the aid of the microcontroller.

TI is built as a finite state machine in a similar way to the Command Interface.

Let's now explain what testmode registers are. All the circuits added for test purposes can't influence the standard user mode functionality and can't worsen performances. The adopted solution is sketched in Fig. 6.5. A TM register is associated with a UM register: when the signal TESTMODE is high, the output takes the value contained in the register TM, influencing the behavior of the circuitry downstream. When the signal TESTMODE goes low, the standard usermode functionality is enabled.

6.2.3 Datapath

Till few years ago, NAND memories had an asynchronous interface and it was very difficult to run frequencies higher that 40 MHz for data download/upload [6]. NAND chips have linear dimensions easily higher than 10 mm so that data have to flow through a long path with an unavoidable impact on the transmission time through

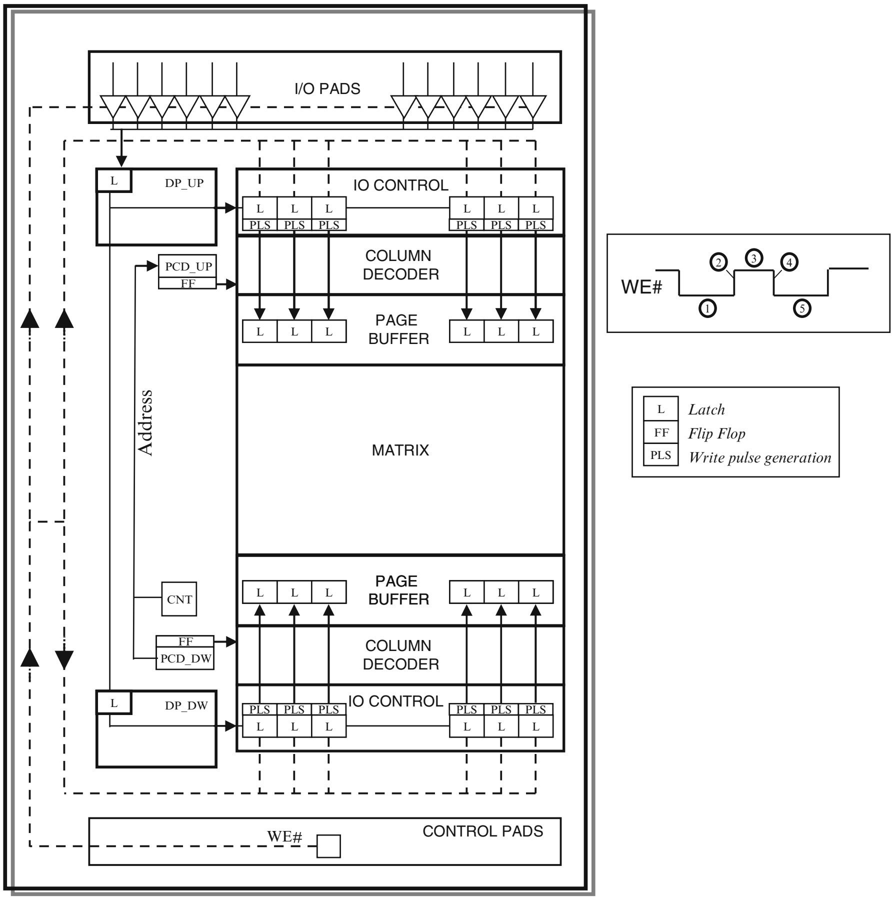


Fig. 6.6 Input datapath

the chip. One of the most adopted solutions to overcame this problem is the use of a pipeline on the datapath [7].

In the following we will describe datapath structure for a NAND memory with double side architecture and with control pads on the opposite side with respect to data pads.

With reference to Fig. 6.6 the data input sequence is here described.

1. During the low-phase of WE#, input buffers on I/O PADS block and latches on DP_UP and DP_DW blocks are enabled. In this way, input data flow to the latches placed in DP_UP and DP_DW blocks.
2. On the rising edge of WE#, I/O PADS input buffers are disabled. Data are latched in DP_UP latches till the next falling edge of WE#. The counter addresses the appropriate page buffers for the following write operation.
3. On the high-phase of WE#, IO CONTROL latches are open and the COLUMN DECODER is addressing the right page buffers.

4. On the falling edge of WE#, data are latched in the IO CONTROL latches.
5. On the next low-phase of WE#, while I/O PADS input buffers and DP_UP latches receive new data from the user (as in phase 1), IO CONTROL generates write pulses for loading the latched data into the page buffer latches.

A similar approach is adopted for data output.

Performance driven applications like Solid-State-Disks (SSDs) are now forcing the NAND towards the adoption of a DDR interface, as described in Sect. 6.3.

6.2.4 Microcontroller

As already said, the microcontroller inside the memory is the "brain" of the device. Microcontroller implements the needed algorithms for a Flash memory. In order to be able to perform the necessary operations, these conditions must hold true:

- each sequence of operations that must be executed for a specific algorithm (read, program, erase etc.) has to be non-volatile;
- the microcontroller needs to perform arithmetical, logical and output operations.

Usually, microcode (FW) is stored in a ROM memory (Fig. 6.7). There could also be a Code RAM memory containing the specific firmware for testing and debugging.

The microcontroller contains a number of different blocks. First of all there is the Program Counter. It stores the address of the memory location containing the instruction that must be executed. It is also able to handle the address increment, the absolute or relative jumps and the calls to subroutines with different stack levels. The levels of stack indicate how a subroutine is far away from the main program.

Another important block is composed by the Internal Registers: they are necessary for the execution of an operation or a sequence of operations. A register can be either loaded with a constant value or with a value read from the ROM, and it can also be the result of an operation.

The microcontroller computational center is the Arithmetic Logic Unit or ALU. The ALU executes an operation associated with a specific opcode and implemented in the microcontroller. The operations can be with one or two operands. The operands can be internal registers, flags or constants read from the ROM. The result of the operation is stored in the internal registers, with the exception of test and compare operation.

Finally, the last block of the microcontroller is constituted by the Output Registers. Each register is made up by a number of latches. The most advantageous structure for the output registers is based on the dual ports concept.

With this structure, the registers are handled by two independent ports called port A and port B. For instance, port A operates over all the outputs, while port B operates only over some output registers.

The dual ports structure allows the use of two different bank registers at the same time, so that it is possible to move more control signals at each clock cycle.

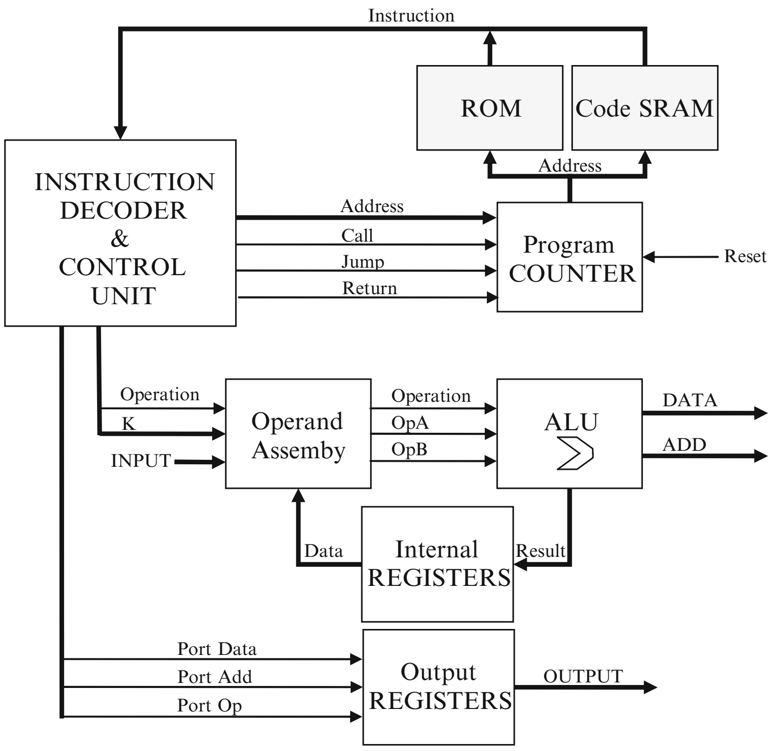


Fig. 6.7 Microcontroller structure with ROM and RAM memories

Apart from the internal structure, the characterizing feature of a microcontroller is what it is able to do, that is its Instruction Set. Before designing a microcontroller, we need to understand the must-have operations. In fact, general purpose microcontrollers are not useful in the NAND memory environment, because they are generally bigger and slower, in order to guarantee a full flexibility not needed in the device. In other words, it is useless to implement operations not used, but it is better to optimize the used ones.

6.3 NAND DDR Interface

Flash based systems are made up by several NAND memory devices and one controller. The controller has the primary function to communicate with NANDs and conveys data from/towards the external interface. Especially, SSDs call for a higher Read/Write throughputs; in other words, SSDs need to manage more NAND dies in parallel. Basically, there are a couple of options.

The first one is to increase the number of dies per channel as shown in Fig. 6.8a. This solution encounters limitations from channel parasitic loading. It has the

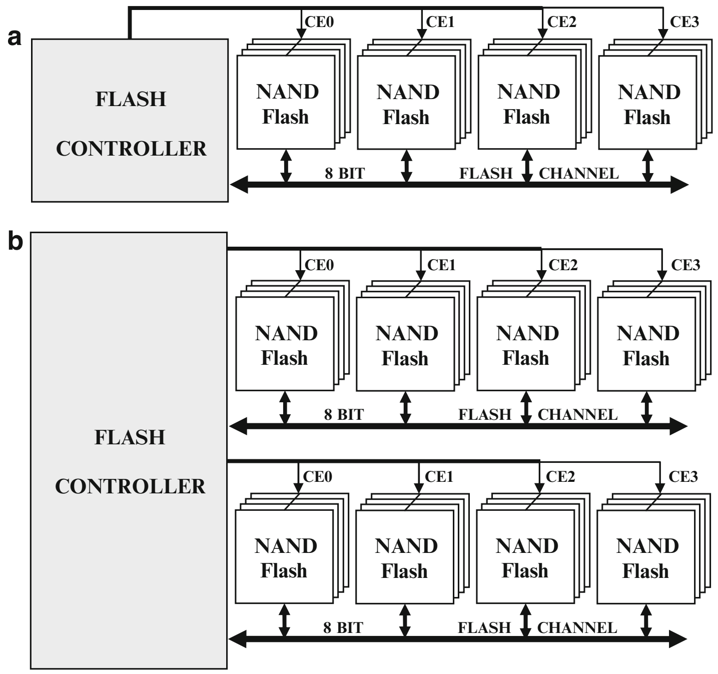


Fig. 6.8 SSD system enhancement: (**a**) increased number of dies per channel (**b**) increased number of channels

advantage of lower pin count and lower hardware cost, especially for the controller, but it might not satisfy the requirements of Write throughput.

The second option is to increase the number of channels (Fig 6.8b). This solution shifts all the problems inside the memory controller which has to manage the parallel data flow coming from all the memory channels. The drawback is that the controller has to manage the ECC for each channel and have the need of dedicated SRAM. On the positive side, this solution is scalable and flexible and allows to reach very high Read/Write throughput. Nowadays, multiple channel architectures are quite common in SSD design.

In every case, power and signal integrity must be addressed with careful interface design considerations. In this section, we mainly deal with the I/O bottleneck problem which must first be solved by a proper interface roadmap.

6.3.1 DDR Interface

High speed NAND introduced a Double Data Rate (DDR) interface in year 2008. As a matter of fact, NAND memories are now following the same path that DRAMs experienced from year 2000.

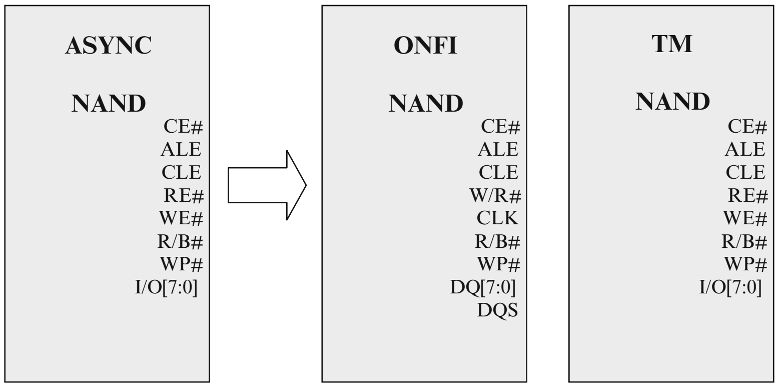


Fig. 6.9 Legacy NAND vs. ONFI 2 and Toggle-mode synchronous NAND interface

The challenge in the coming years will be the standardization of the interface among vendors. Two solutions are available in the market, as draft in Fig. 6.9. On one side, ONFI organization [8] introduced an interface with a clock and data strobe, ready for a DRAM-like evolutionary path. Pinout differences between legacy and ONFI 2.0 interfaces are:

- WE# becomes a fast CLK;
- RE# handles data direction by becoming W/R# (Write/Read#);
- I/O[7:0] renamed to DQ[7:0] (name change only, functionally identical);
- DQS, a new bi-directional signal, is enabled.

On the other side, Samsung decided for a different approach named "Toggle" [9] where only data strobe has been added to the legacy NAND pinout; Toggle mode adds DQS data strobe signals; RE# is used to trigger the read cycle as done in asynchronous interface; DQS is used to strobe the data on both edges.

As usual, JEDEC is now working on combining the above interfaces in a single standard.

ONFI has already released the third generation of specifications where they target 400 MB/s throughput, and Toggle is targeting the same speed. The interface roadmap stays with LVTTL bus driving style as long as possible in order to ease integration, but some design tricks have to be introduced in order to sustain higher bandwidths. This will include the proper scaling of interface voltage, the use of a specific termination type, On-Die Terminations, differential strobes and, going beyond, synchronization circuit. Finally, it will include DLL/PLL and the change to a SSTL class of terminated bus.

The DDR protocol diagrams are sketched in Fig. 6.10. A Synchronous clock must be provided to the memory chips (not needed in Toggle-mode interface). Bidirectional Data bus DQ is driven at every clock edge. Therefore, data throughput is doubled compared to a Single data rate system, assuming the same clock frequency.

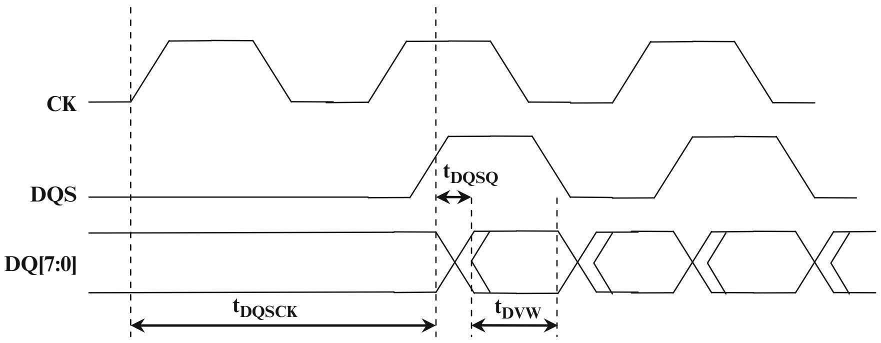


Fig. 6.10 DDR timing diagram

Data strobe signal DQS behaves like all other DQs and it is used as data capture signal on the receiver side. Systems scalability benefits from this approach since DQS load always matches that of DQ lines, ensuring same timings: this is very important in SSD design because the parasitic load of a Flash channel changes when more dies are used.

6.3.2 *Power*

Let's consider an SSD where multiple Flash channels are used. Due to the channel parasitic capacitance, each time a single NAND die is written or read, the entire capacitance of data lines needs to be driven.

I/O power consumption in a DDR system can be written as [10]:

$$P = 9 \cdot \eta \cdot f \cdot C \cdot V^2 \tag{6.1}$$

where η is the bit activity ratio, f is the DDR frequency, C is the capacitance of a single line and V is the supply voltage of the interface. Figure 6.11 shows the impact of I/O power supply. Therefore, scaling the I/O interface voltage becomes a must, especially looking at higher clock frequencies.

6.3.3 *Capacity*

SSD storage capacity can be increased in two ways:

- by increasing the number of Flash channels;
- by increasing the number of NAND dies connected to a single channel.

As already mentioned, the first solution has been widely adopted, even if it increases the hardware complexity of the SSD controller

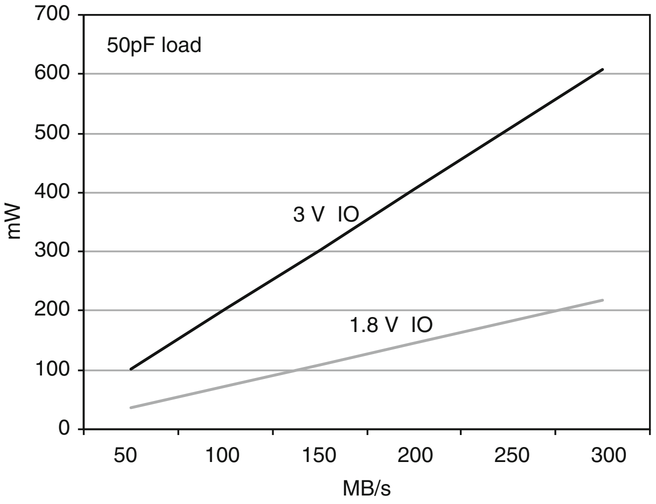


Fig. 6.11 I/O power as a function of channel throughput

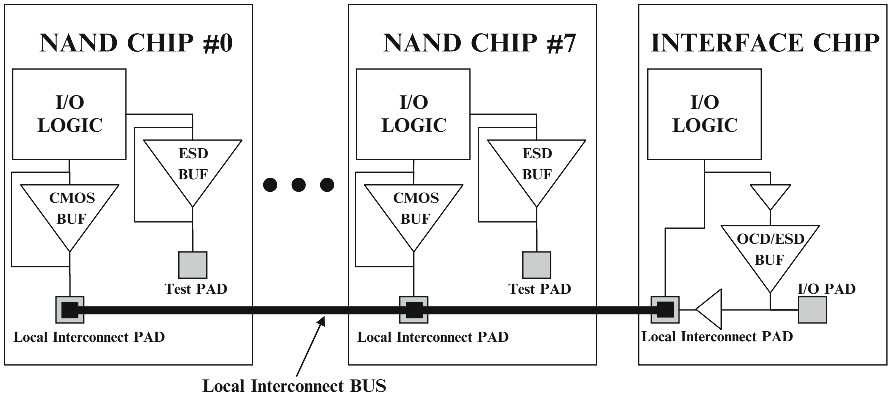


Fig. 6.12 Local interconnect bus architecture

The adoption of the second solution is mainly limited by the resulting I/O parasitic capacitance of the Flash channel. To partially overcome this limitation, it is possible to use advanced System in Package technologies such as *Through Silicon Vias* (TSV) [11]. TSV creates interesting opportunities for stacking, thanks to its low parasitic capacitance.

Figure 6.12 depicts a system in which memory chips are stacked and connected using a Local Interconnect Bus. The Interface Chip provides data translation from local interconnect bus to the external bus (i.e. Flash channel) by means of a standard off chip driver (OCD). It is worth mentioning that the local bus can be driven by standard CMOS buffers instead of OCD ESD-compliant structures. Furthermore, by using simplified ESD structures, the bus parasitic capacitance can become even lower.

6.4 I/O Design

This section starts with an overview of I/O design problems in legacy asynchronous NAND products available in the market. Design of high-speed I/O is then reviewed.

6.4.1 Basic CMOS Output Buffer Design

Usually, NAND output buffers need to drive large capacitive loads, in the range of 50–100 pF. In this situation the output capacitance transition is very long compared to the buffer switching time. The buffer conductance is usually made very large to reduce the charge/discharge time and match the specifications.

The memory data bus can be 8/16 bits: the current sunk by the parasitic capacitor of a single output buffer has to be multiplied by the number of switching data bits. Moreover, the inductance of the bonding wire (5÷10 nH in TSOP packages) might generate bounces on internal power supply lines that could affect the functionalities of analog circuits [12]. This effect is called *Simultaneous Switching Noise* (SSN) and will be treated in more details later.

A basic output buffer with push-pull architecture is shown in Fig. 6.13.

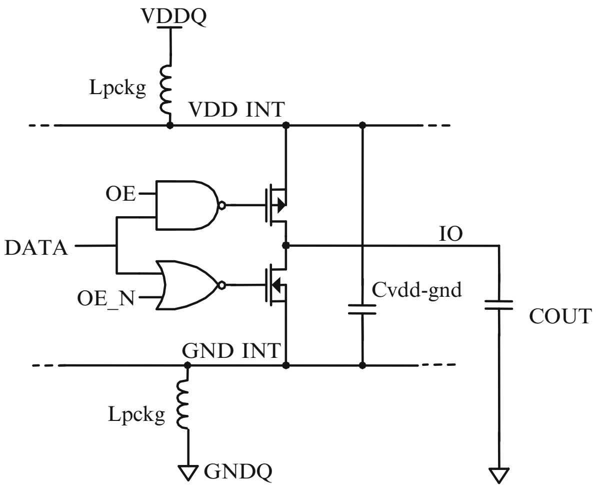


Fig. 6.13 Output buffer model

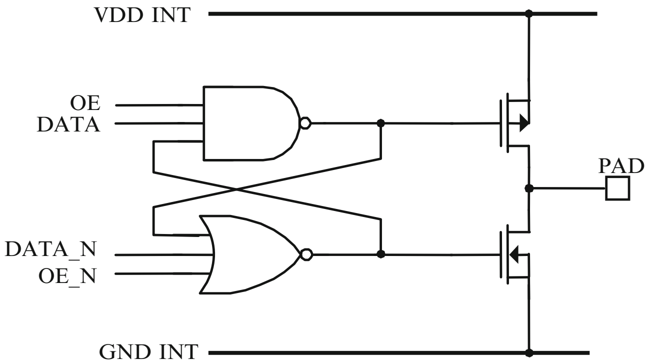


Fig. 6.14 Pre-driver to avoid crowbar in push-pull stage

In order to reduce the current peak, switching time of push-pull drivers have to be carefully controlled. As a consequence, if gates of PMOS and NMOS are driven at a lower speed, crowbar current becomes an issue. Crowbar occurs when both PMOS and NMOS are ON at the same time. To avoid this situation, the buffer structure of Fig. 6.14 can be adopted [3, 13]. In this configuration the pull-up is switched-off before the pull down is turned on (and vice versa).

NAND and NOR gates can be tuned to obtain a fast switching-off and a proper switching-on time. In the figure it is also shown the output enable signal OE that is used to turn the output stage in high impedance: in this way, data bus can be driven by somebody else.

Another important design constraint is the slew rate of the output driver. In asynchronous devices, the slew rate is generally controlled by acting on the pre-driver, so that the pull-up and pull-down transistors are gradually switched on/off [13, 14].

Generally, this is optimized in the slow corner and the result is a big variation with Process/Voltage/Temperature (PVT). The pre-driver *RC* output constant must be much smaller than the data window, otherwise there is a risk to have a data dependent jitter. If a wide data bus is used, it could be beneficial to consider skewing the output enable by a proper small delay and consequently spreading in time the current requests.

6.4.2 Simultaneous Switching Noise (SSN)

One of the main responsible for data window margins degradation is the simultaneous switching noise [13, 15–18]. SSN is an inductive noise caused by several outputs switching at the same time. One single buffer could have a good transient behavior, but, when all the data buffers are switching at the same time, the data AC behavior could be corrupted. The problem is serious in output buffer memory design because of two effects:

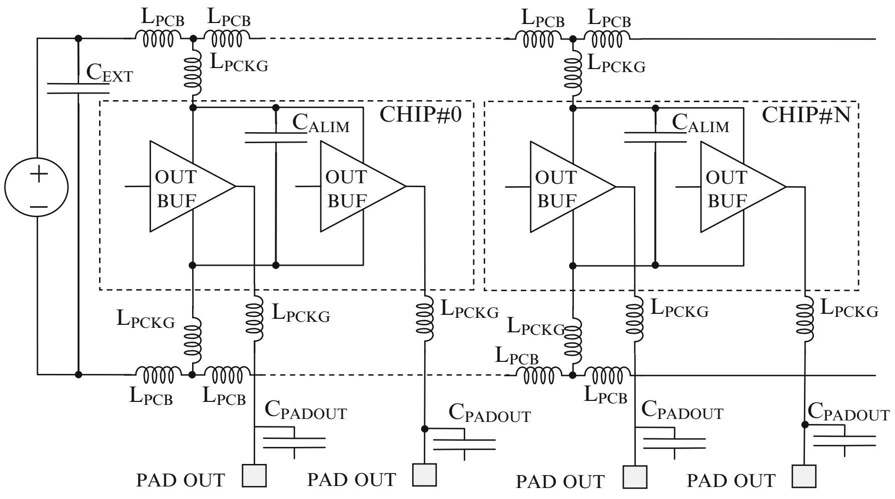


Fig. 6.15 Model example used to evaluate SSN

- jitter and signal bounces are increased and data window margin is reduced;
- the generated noise could affect other circuits, especially analog circuits and memory sense amplifiers, reducing operating margin or creating systematic non-working windows.

With a large capacitive load, a large current is requested to charge the load and the power network must supply that current. The current flows in inductances, typically in the bonding wires or leads of the package, and the resulting noise is injected into power and ground supplies. This noise is transferred to the output and the output AC characteristics are affected.

The simultaneous switching noise is determined, in principle, by the following equation:

$$V_{SSN} = N \cdot L \cdot \frac{\partial I}{\partial t} \tag{6.2}$$

where N is the number of switching outputs, L the equivalent inductance in which current must flow, and I the current per driver.

Since this mechanism is dependent on the number of output switching N, this makes the noise dependent also on the data sequence.

To deal correctly with SSN it is necessary to understand the complete signal current paths in the memory. In Fig. 6.15 a complete path is shown. Local metal resistances are omitted but they should be evaluated as possible sources of interference. It is straightforward to understand that the problem is really connected with the package. When TSOP packages are used, very long bonding wires can be present leading to high inductance values. Moving to higher data rates requires to leave such packages for more controllable Ball Grid Arrays.

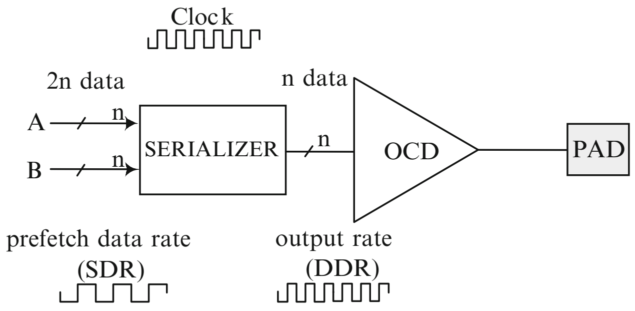


Fig. 6.16 OCD schematic block diagram

6.4.3 High Speed NAND I/O Design

Output buffer in high speed signal transmission is often named *Off-Chip Driver* (OCD). In addition to the task of being the interface circuit between inside and outside, OCD in high speed memories has to accomplish several additional tasks.

- Translate data flow between single data rate (SDR) and DDR domains.
- Voltage domain change. The core of the memory could operate at a different voltage level than the I/O interface and the data signals have the need to be shifted from the core level to the interface voltage.
- Provide the AC/DC requirements such as V_{OL}/V_{OH}, slew rate or impedance matching.
- Provide the On-Die-Termination (ODT).
- ESD protection.

Various types of OCD are used in memory design depending on the interface type and speed. In order to introduce all the basic concepts, we focus here on the single ended CMOS buffer, which is widely used in DDR designs.

6.4.4 Double Data Rate OCD

A DDR OCD is a synchronous output buffer. In synchronous systems, OCD includes a register stage used to synchronize the output with the internal data bus. In DDR design a block named *serializer* is included in the buffer design as shown in Fig. 6.16. Serializer block performs the Single Data Rate (SDR) to DDR conversion: it receives 2n data at a given rate R (SDR) and multiplexes these data onto an internal line at a higher rate 2R (DDR).

We should highlight that the OCD is operating at a frequency higher than the one used by other blocks in the memory chip. Therefore, since we have to deal with smaller delays inside the OCD, it is necessary to take more countermeasures in designing the block to avoid jitter eating almost all margins.

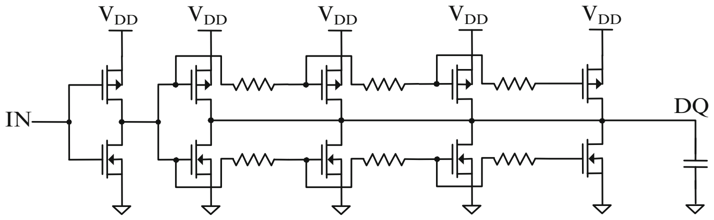


Fig. 6.17 Slew rate control by output driver time-distributed activation

6.4.4.1 OCD Linearity: Push-Pull and Open-Drain Configurations

It is of primary importance to offer a linear behavior of the output characteristics because of the system signal integrity. In other words, OCD linearity is key for impedance matching with the external line.

6.4.4.2 Slew Rate Control and Bandwidth

Drivers should be designed in order to avoid driving frequencies greater than the signaling rate. Simple and sophisticated methods can be used, such as passive delays after the pre-driver or current control technique for the pre-driver stage. A time-split method is widely used. The basic principle is to split output pull-up and pull-down devices into branches and activate them serially with proper sequential delays. This time-distributed driver can be implemented in a simple analog form suitable for relative low operating frequency or digital form [13, 19, 20]. Figure 6.17 shows a basic implementation of the analog form where the pull-up/down branches are driven by a resistive line which contributes to define the *RC* delay element for each branch. Each branch can be "weighted" to obtain the best slew rate conditions.

6.4.4.3 Voltage Domain Change: Level Shifting

I/O voltage usually differs from the power supply of the NAND core. For example, the memory could internally operate at 1.5 V by means of a DC-DC down-converter, whereas the data interface needs a 3 V or 1.8 V driving. Voltage domain change occurs also when the memory has different power pins for core supply voltage and I/Os. This situation allows the use of independent supply generators to separate the noise coming from data bus and from the core region. In a simpler system design it is still possible to connect the pins to the same supply on the PCB. The OCD structure implements the level shifting function which consists in shifting the levels of the digital signals from the core voltage GND/VDD to the interface voltage GNDQ/VDDQ. Figure 6.18 shows a modified structure where NMOS transistors M5 and M6 are added in order to speed up the transition of nodes from low to high.

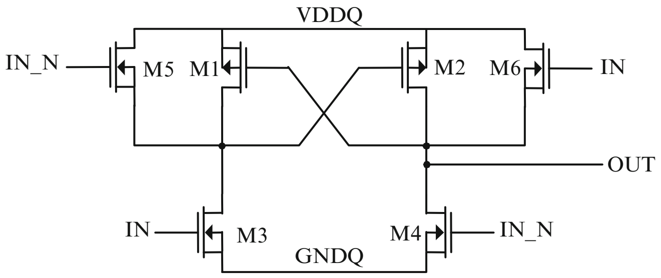


Fig. 6.18 Level shifter modified

The level shifting circuit or, more generally, the point where the data change voltage domain, is critical in jitter generation. The two domains provide two different references for the signal detection; therefore, any disturbs on the power supply lines lead to the introduction of additional distortion.

6.4.4.4 Jitter Sources and Duty Cycle Distortion

Off-Chip Driver complexity implies that data is travelling along many gates before reaching the output stage. The design of the chain of inversions is fundamental in the control of duty cycle distortion. Duty cycle distortion occurs when:

- positive and negative slopes are different;
- number of inversion is odd;
- ground or power shifts.

To reduce the jitter in a chain of inverters it is necessary to keep the same slope in the chain, i.e. using the same ratio between the driver strength and the load, instead of trying to minimize the number of inverters in the chain. Another source of jitter is hidden in level shifters and voltage domain change. Level shifter sketched in Fig. 6.18 introduces asymmetric positive/negative slopes detected by a receiver gate with different time delay.

In conclusion, high-speed NANDs require a very sophisticated I/O design because of its impact on SSD's power, performances and signal integrity.

6.5 Read Operation: The Sense Amplifier

Let's now move in the core region. The reading operation is designed to address specific memory cells within the array and measure their information content. As in other types of Flash memories, the stored information is associated with the cell's threshold voltage V_{TH}: in Fig. 6.19 the threshold voltage distributions of cells containing one logic bit are shown. If the cell has a V_{TH} belonging to the

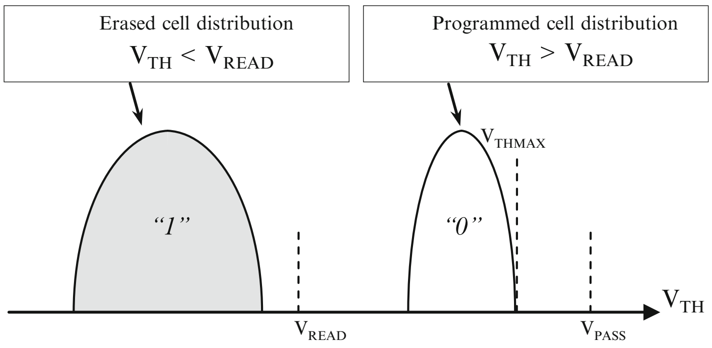


Fig. 6.19 Threshold voltage distributions of erased ("1") and programmed ("0") cells

erased distribution, it contains a logic "1", otherwise it contains a logic "0". Cells containing *n* bit of information have 2^n different levels of V_{TH}.

Flash cells act like usual MOS transistors. Given a fixed gate voltage, the cell current is a function of its threshold voltage. Therefore, through a current measure, it is possible to understand which V_{TH} distribution the memory cell belongs to.

The fact that a memory cell belongs to a string made up by other cells has some drawbacks. First of all, the unselected memory cells must be biased in a way that their threshold voltages do not affect the current of the addressed cell. In other words, the unselected cells must behave as pass-transistors. As a result, their gate must be driven to a voltage (commonly known as V_{PASS}) higher than the maximum possible V_{TH}. In Fig. 6.19 V_{PASS} has to be higher than V_{THMAX}.

However, the presence of 2^n-1 transistors in series has a limiting effect (saturation) on the current's maximum value; this maximum current is, therefore, much lower than the one available in NOR-type Flash memories.

Figure 6.20 shows the I–V (current-voltage) characteristic of a NAND cell (string): V_{READ} is applied to the selected gate while V_{PASS} bias the unselected gates. V_{PASS} is a fixed voltage. Three main string working-regions can be highlighted.

1. Region A: the addressed cell is not in a conductive state.
2. Region B: V_{READ} makes the addressed cell more and more conductive.
3. Region C: the cell is completely ON, but the series resistance of the pass transistors (unselected cells) limits the current to I_{SSAT}.

The string current in region C can be estimated as:

$$I_{SSAT} = \frac{V_{BL}}{(n-1)R_{ON}} \tag{6.3}$$

where R_{ON} is the series resistance of a single memory cell, V_{BL} is the voltage applied to the bitline and *n* is the number of the cells in the string. R_{ON}, at a first approximation, is the resistance of a transistor working in the ohmic region.

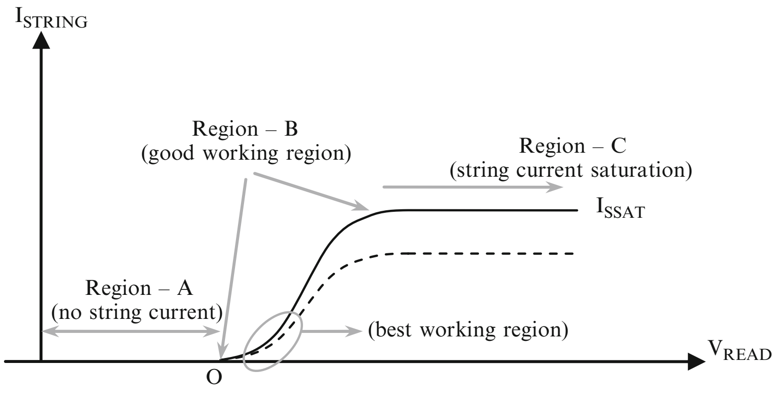


Fig. 6.20 Cell current characteristics versus gate voltage

For a MOS transistor in ohmic region the following equation holds true:

$$I_D = k \cdot \left[(V_{GS} - V_{TH}) \cdot V_{DS} - \frac{{V_{DS}}^2}{2} \right] \tag{6.4}$$

For small V_{DS} values, as in our case, Eq. (6.4) may be simplified as:

$$I_D = k\left[(V_{GS} - V_{TH}) \cdot V_{DS}\right] \tag{6.5}$$

Therefore, R_{ON} is equivalent to

$$R_{ON} = \frac{V_{DS}}{I_D} = \frac{1}{k\,(V_{GS} - V_{TH})} \tag{6.6}$$

Equation (6.6) shows that R_{ON} is a function of V_{TH}. In other words, I_{SSAT} depends on the V_{TH} values of the *n* cells in series. When all the cells are programmed to V_{THMAX}, R_{ON} takes its maximum value (dashed line in Fig. 6.20). R_{ON} influences the I–V characteristic also in region B but in a more negligible way. In order to reduce the dependency from R_{ON}, the cell has to be read in region B as near as possible to point O.

The order of magnitude of the saturation current, in the state-of-the-art NAND technologies, is a few hundreds of nA, that means a reading current of some tens of nA. It is very hard to sense such small currents with the standard techniques used in NOR-type Flash memories, where the reading current is, at least, in the order of some μA. Moreover, in NAND devices, tens of thousands of strings are read in parallel. Therefore, tens of thousands of reading circuits are needed. Due to the multiplicity, a single reading circuit has to guarantee a full functionality with a very low area impact. As a matter of fact, the first memory NAND prototypes used traditional sensing methods, since the said currents were in the order of tens of μA [21].

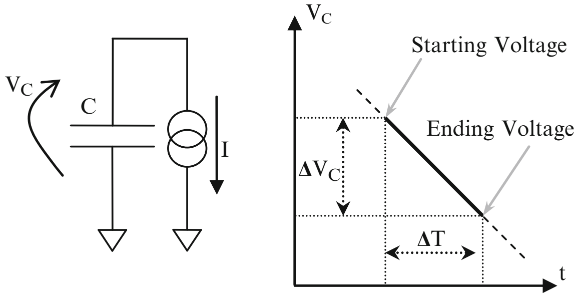


Fig. 6.21 Capacitor discharge through a constant current source

The reading method of the Flash NAND memories consists in integrating the cell current on a capacitor in a fixed time (Fig. 6.21). The voltage ΔV_C across a capacitor C, charged by a constant current I for a time period ΔT, is described by the following equation:

$$\Delta V_C = \frac{I}{C} \Delta T \qquad (6.7)$$

Since the cell current is related to its V_{TH}, the final voltage on the capacitor (ΔV) is a function of V_{TH} too.

There are different reading techniques, starting from the one using the bitline parasitic capacitor, ending with the most recent sensing technique which integrates the current on a little dedicated capacitor. The above mentioned techniques can be used both in SLC and MLC NAND memories. In the MLC case, multiple basic reading operations are performed at different gate voltages.

Historically, the first reading technique used the parasitic capacitor of the bitline as the element of the cell current integration [22–24].

In Fig. 6.22 the basic scheme is shown. V_{PRE} is a constant voltage. At the beginning, C_{BL} is charged up to V_{PRE} and then it is left floating (T_0). At T_1 the string is enabled to sink current (I_{CELL}) from the bitline capacitor. The cell gate is biased at V_{READ}. If the cell is erased, the sunk current is higher than (or equal to) I_{ERAMIN}. A programmed cell sinks a current lower than I_{ERAMIN} (it can also be equal to zero). C_{BL} is connected to a sensing element (comparator) with a trigger voltage V_{THC} equal to V_{SEN}. Since I_{ERAMIN}, C_{BL}, V_{PRE} and V_{SEN} are known, it follows that the shortest time (T_{EVAL}) to discharge the bitline capacitor is equal to:

$$T_{EVAL} = C_{BL} \frac{V_{PRE} - V_{SEN}}{I_{ERAMIN}} \qquad (6.8)$$

If the cell belongs to the written distribution, the bitline capacitor will not discharge below V_{SEN} during T_{EVAL}. As a result, the output node (OUT) of the voltage comparator remains at 0. Otherwise, if the cell is erased, V_{BL} drops below V_{SEN} and the OUT signal is set to 1.

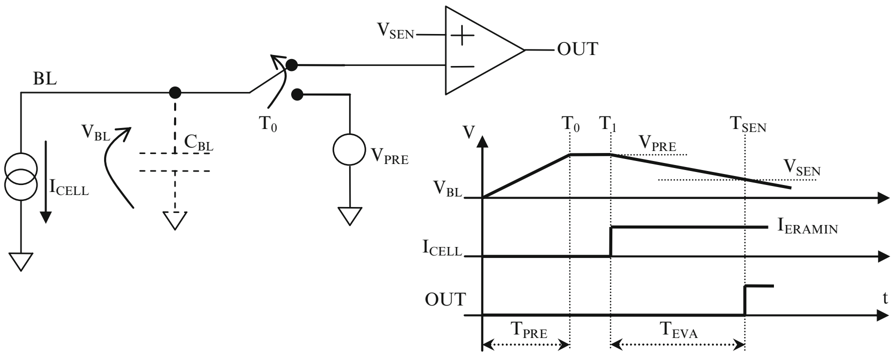


Fig. 6.22 Basic sensing scheme exploiting bitline capacitance and the related timing diagram

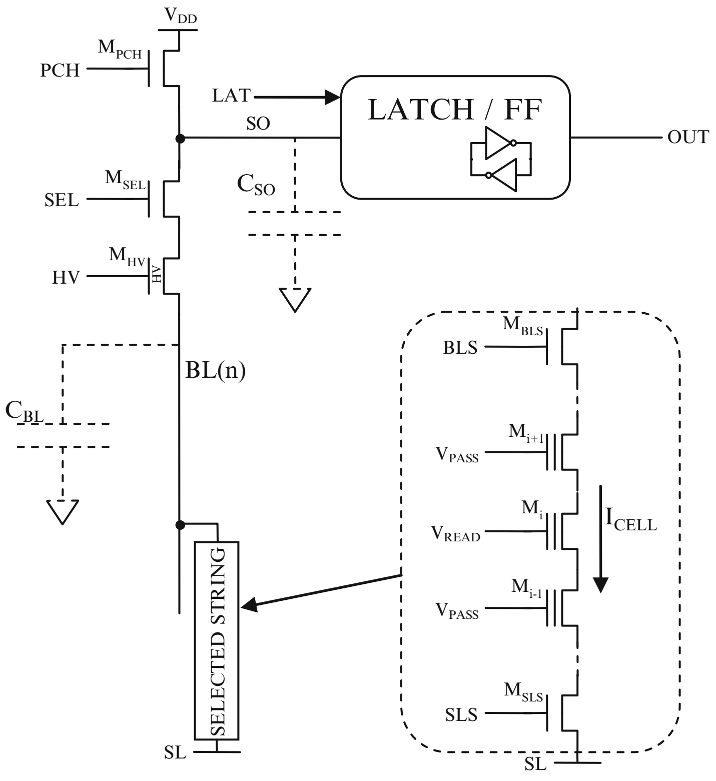


Fig. 6.23 Basic elements of the sense amplifier

The basic sense amplifier structure is sketched in Fig. 6.23. During the precharge phase T_{PRE}, M_{SEL} and M_{PCH} are biased to V_{PRE} and $V_{DD} + V_{THN}$ respectively. V_{THN} is the threshold voltage of a NMOS transistor and V_{DD} is the device's power supply voltage.

As a consequence, C_{BL} is charged to the following value:

$$V_{BL} = V_{PRE} - V_{THN} \tag{6.9}$$

During this phase, the SO node charges up to V_{DD}. Since V_{GS} and V_{DS} can be higher than 20–22 V, M_{HV} has to be a high voltage (HV) transistor. In fact, during the erase phase, the bitlines are at about 20 V and M_{HV} acts as a protection element for the sense amplifier's low voltage components. Instead, during the reading phase, M_{HV} is biased at a voltage that makes it behave as pass-transistor. Moreover, during the precharge phase, the appropriate V_{READ} and V_{PASS} are applied to the string. M_{BLS} is biased to a voltage (generally V_{DD}) that makes it work as pass transistor. Instead, M_{SLS} is turned off in order to avoid cross-current consumption through the string.

Typically, V_{BL} is around 1 V. From Eq. (6.9), V_{PRE} values approximately $1.4 \div 1.9$ V, depending on the V_{THN} (NMOS threshold voltage). The bitline precharge phase usually lasts $5 \div 10$ μs, and depends on many factors, above all the value of the distributed bitline parasitic *RC*.

Sometimes this precharge phase is intentionally slowed down to avoid high current peaks from V_{DD}. In order to achieve this, the M_{PCH} gate could be biased with a voltage ramp from GND to $V_{DD} + V_{THN}$.

At the end of the precharge phase, PCH and SEL are switched to 0. As a consequence, the bitline and the SO node parasitic capacitor are left floating to a voltage of V_{PRE} -V_{THN} and V_{DD} respectively. M_{SL} is then biased in order to behave as pass transistor. In this way the string is enabled to sink (or not) current from the bitline capacitor.

At this point, the evaluation phase starts. If the cell has a V_{TH} higher than V_{READ}, no current flows and the bitline capacitor maintains its precharged value.

Otherwise, if the cell has a V_{TH} lower than V_{READ}, the current flows and the bitline discharges.

6.5.1 Interleaving Architecture

Given the Eq. (6.8), it is clear that the bitline capacitance has a direct influence on the evaluation time. C_{BL} must fulfill the following requirements:

– it must be a known parameter;
– it must be immune to external noise.

Figure 6.24 is a bitline cross-section showing the different contributions to C_{BL}:

– C_{AD} is the parasitic capacitor between the bitline and the lower plane (usually it is the wordline plane);
– C_{AU} is the parasitic capacitor between the bitline and the upper plane (usually it is the source-line plane);

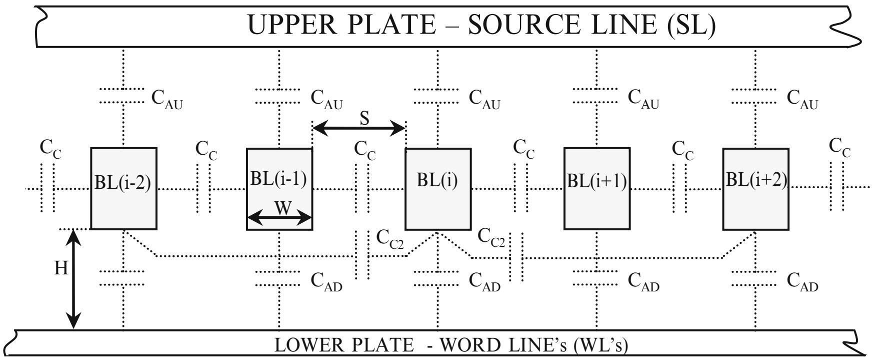


Fig. 6.24 Bitline parasitic capacitors

– C_C is the parasitic capacitor between two adjacent bitlines;
– C_{C2} is the parasitic capacitor between a bitline and its second nearest bitline.

Therefore, C_{BL} can be written as:

$$C_{BL} = C_{AU} + C_{AD} + 2C_C + 2C_{C2} \tag{6.10}$$

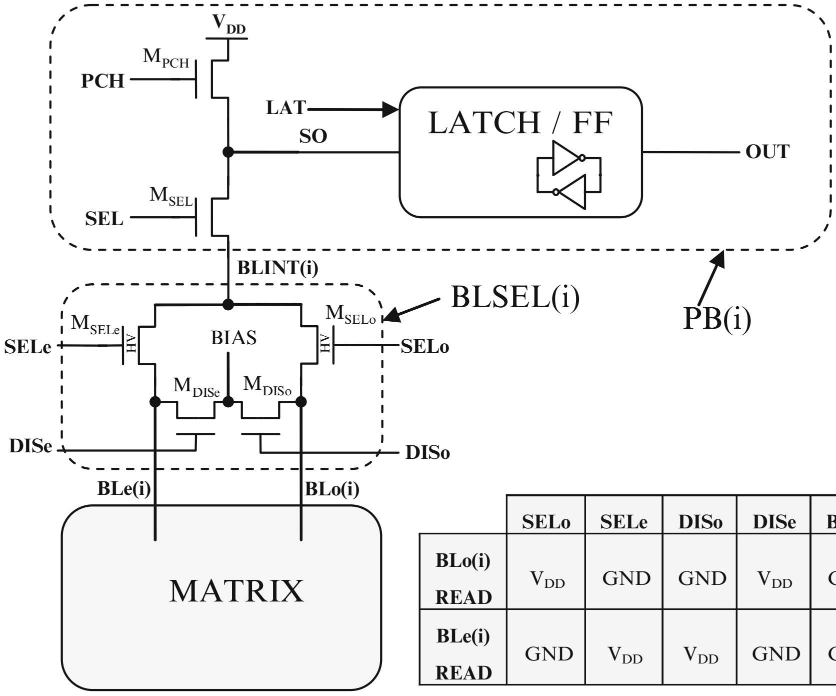
The above mentioned contributions depend on the bitline geometrical values (width W, height H and spacing S in Fig. 6.24), on the distance between upper and lower ground levels and on the oxide thickness. These parameters are not uniform among different wafers, dice and even within the same die. However, a correct reading must be ensured.

In all the explained theory, another important assumption is that the bitline capacitor has one of its terminals fixed to ground. Actually, looking at Fig. 6.24, C_{BL} ground terminal is physically distributed over four nodes:

1. the upper plate, usually the source-line;
2. the lower plate, usually the wordline or the source-line;
3. the left bitline;
4. the right bitline.

During the evaluation time the first two nodes are forced at a fixed voltage. Instead, the adjacent bitlines could be discharged by the strings connected to them.

With the continuous bitline shrinking (W and S in Fig. 6.24), the coupling capacitances play an important role. In sub-40 nm NAND technologies they contribute 80÷90% of the total bitline capacitance. To overcome this issue, the interleaving architecture is introduced. While the even (or odd) bitlines are read, the odd (or even) bitlines are forced to a fixed voltage (generally ground), acting as electrical shield [22–24]. As shown in Fig. 6.25, M_{SLe} and M_{SLo} (bitline selectors) are placed between the bitlines and the page buffer PB(i). If the even bitlines BLe



	SELo	SELe	DISo	DISe	BIAS
BLo(i) READ	V_{DD}	GND	GND	V_{DD}	GND
BLe(i) READ	GND	V_{DD}	V_{DD}	GND	GND

Fig. 6.25 Interleaving bitline architecture

are read, M_{SELe} acts as a pass-transistor. Transistor M_{SELo} is turned off. The DISo signal turns on the M_{DISo} transistor, forcing the odd bitline BLo to the fixed BIAS voltage. M_{DISe} is turned off.

In order to minimize the power consumption, BIAS and the source line (SL) should be biased at the same voltage. In fact, these two nodes are shorted if a cell with $V_{TH} < V_{READ}$ belongs to the unselected bitlines. SL and BIAS are usually grounded during the reading operation.

With this architecture, the noise injection effect through the C_C coupling capacitors is eliminated. However, the coupling through C_{C2} (Fig. 6.24) is still in place. This contribution is not negligible: in the state-of-the-art technologies, C_{C2} contributes 5–10% of the total bitline capacitance. This problem is solved by the architecture described in the next section.

6.5.2 All BitLine (ABL) Architecture

The sensing technique is basically the same used in the interleaving architecture. An intentionally placed capacitor is used instead of the C_{BL} bitline parasitic capacitor [25].

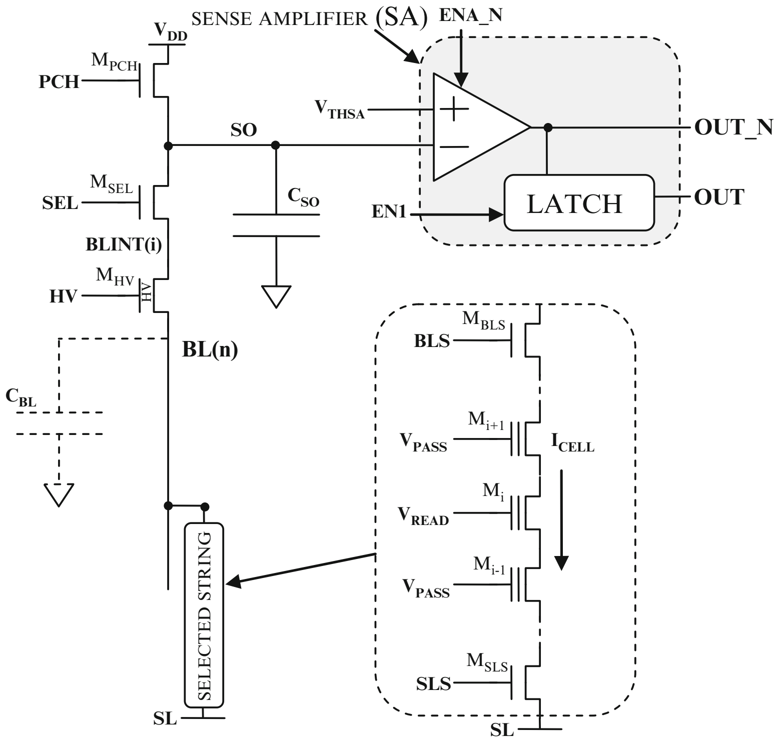


Fig. 6.26 ABL sense amplifier

Figure 6.26 shows the main elements of the ABL sense amplifier. The latch is replaced by a voltage comparator with a V_{THSA} trigger voltage. The other elements are those ones already described in the interleaved architecture, but here used in a different way. The capacitor C_{SO} is involved in the integration of the cell current: it can be done using either MOS gates or poly-poly capacitors.

Figure 6.27 shows the timings used in a single read operation. The precharge phase is similar to that one described for the interleaving architecture, where M_{PCH} and M_{SEL} gates are biased to $V_{DD} + V_{THN}$ and V_{PRE} respectively. M_{HV} HVNMOS has the behavior already described and, during the single read operation phase, works as pass transistor. The signals which drive the string gates (V_{READ}, V_{PASS} and BLS) are activated as usually. Instead SLS signal is immediately activated in order to stabilize the bitlines during the precharge phase. In fact, if the SLS had been activated during the evaluation phase, there would have been a voltage drop on those bitlines with an associated sinking current string.

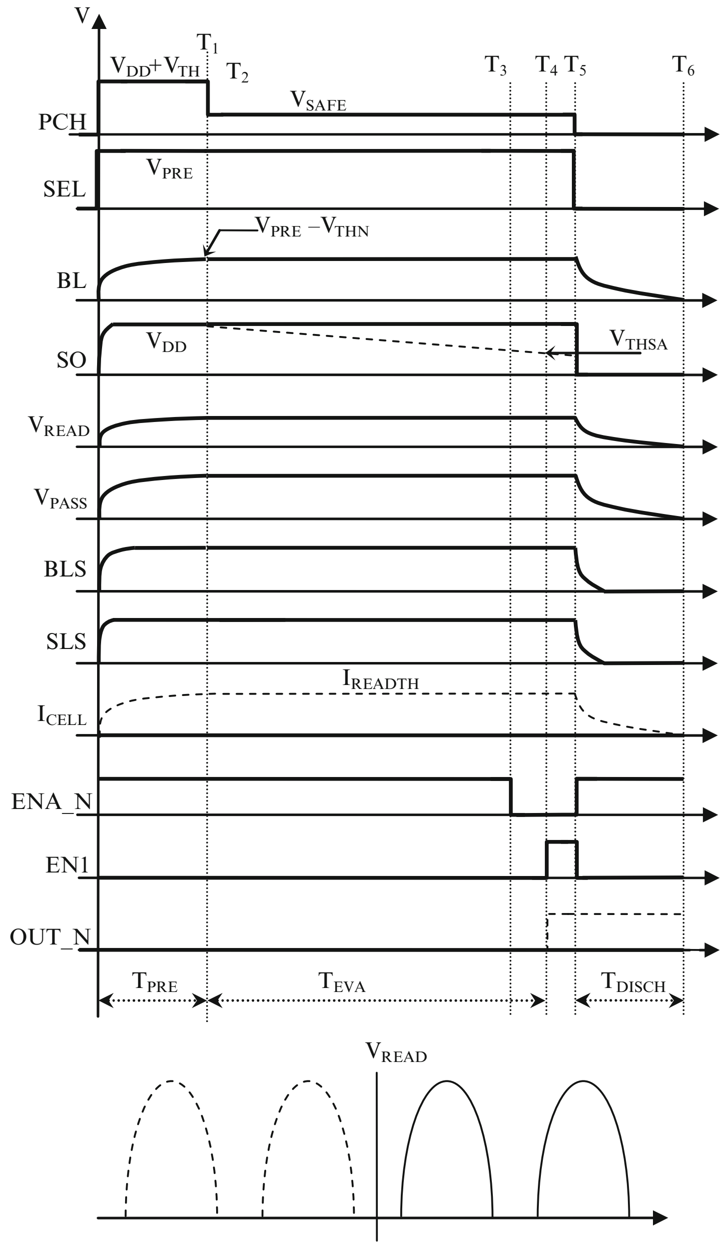


Fig. 6.27 ABL single read operation (SRO) timing diagram

The precharge final condition

$$V_{BL} = V_{PRE} - V_{THN} \tag{6.11}$$

is, therefore, valid only for the bitlines which have an associated string in a non conductive state.

Equation (6.11) should be replaced by:

$$V_{BL} = V_{PRE} - V_{THN} - \Delta \tag{6.12}$$

where Δ is the voltage drop on the bitlines resistance (typical values are in the order of hundreds of kΩ up to one MΩ).

At the end of the precharge phase (T_1), the bitlines are biased to a constant voltage and V_{SO} is equal to V_{DD}. At this point, M_{PCH} is switched off and the evaluation phase starts. Actually, M_{PCH} is biased to a V_{SAFE} voltage value in order to make M_{PCH} behave as a clamp transistor of the SO voltage. The following relation must be valid:

$$V_{SAFE} - V_{THN} \geq V_{PRE} - V_{THN} \Rightarrow V_{SAFE} > V_{PRE} \tag{6.13}$$

This clamp value must not influence the current integration on the SO capacitor, i.e. the clamping function can't take place above the V_{THSA} trigger voltage:

$$V_{SAFE} - V_{THN} \leq V_{THSA} \tag{6.14}$$

Therefore, from Eqs. (6.13) and (6.14), the following conditions must hold true:

$$V_{PRE} - V_{THN} \leq V_{SAFE} - V_{THN} \leq V_{THSA} \tag{6.15}$$

When M_{PCH} is switched off, the cell current (through M_{PRE}) discharges the C_{SO} capacitor. If, during the evaluation time, $V_{SO} < V_{THSA}$ (trigger voltage of Fig. 6.26 comparator), than OUT_N switches (dotted lines in Fig. 6.27). The "threshold current" I_{READTH} is defined as:

$$I_{READTH} = \frac{\Delta V \cdot C_{SO}}{T_{EVAL}} \tag{6.16}$$

where

$$\Delta V = V_{DD} - V_{THSA} \tag{6.17}$$

Observe that, because the bitline is biased to a fixed voltage, a constant current I_{READTH} flows.

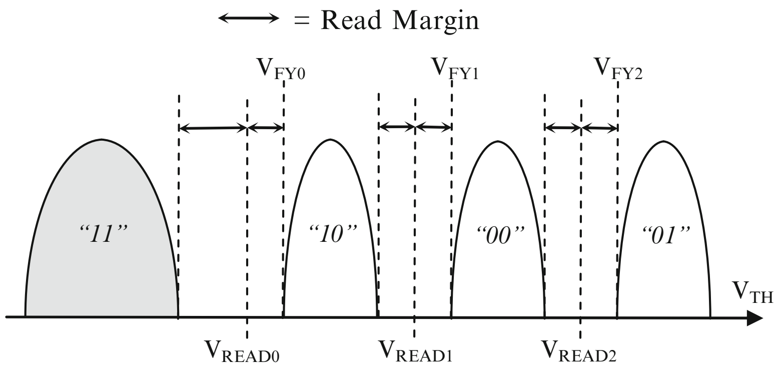


Fig. 6.28 Cell V_{TH} distributions in a 2bit/cell NAND memory

It is possible to extrapolate the evaluation time:

$$T_{EVAL} = \frac{\Delta V \cdot C_{SO}}{I_{READTH}} \tag{6.18}$$

Given the same read currents, it follows that the ratio between Eqs. (6.8) and (6.18) is determined by the ratio between C_{BL} and C_{SO}. C_{BL} is a parasitic element and has a value of 2÷4 pF. Instead, C_{SO} is a design element and has typical values around 20÷40 fF, i.e. two orders of magnitude lower than C_{BL}. The reduction of the evaluation time from 10 μs to hundreds of ns is another advantage of the All Bitline architecture.

In addition, ABL architecture gives further advantages such as energy saving, bitline-coupling reduction and Floating-Gate-coupling reduction during program and read, and program stress reduction [2].

6.5.3 Read Voltage with Thermal Tracking

In a 2bit-per-cell multilevel Flash NAND memory, four different threshold voltage (V_{TH}) distributions exist, as shown in Fig. 6.28. All the cells are in the "11" state after electrical erase. During programming phase, the threshold voltage of the cells is incremented in small steps until the desired value is reached. At the end of each program step, a verify operation is performed, in order to evaluate whether V_{TH} has gone above one of the verify voltages, V_{FY1}, V_{FY2} or V_{FY3}. Of course, verify voltage depends on which bits have to be stored in a given cell. For instance, in order to reach "00" logic value, threshold voltage has to go above V_{FY2}. Once target distribution is reached, further program pulses are not applied to that cell.

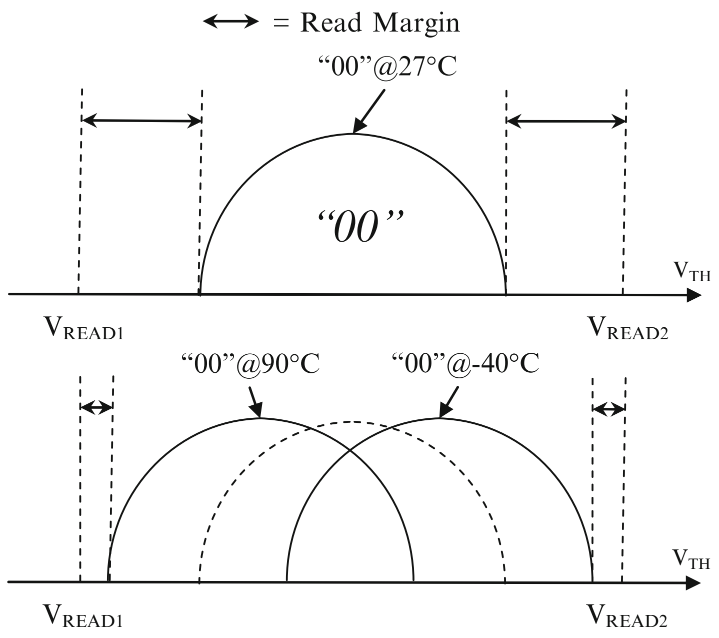


Fig. 6.29 V_{TH} variations with temperature

In order to univocally determine the logic value stored in the selected cell, read operation uses three voltage values, V_{READ0}, V_{READ1}, and V_{READ2} as shown in Fig. 6.28. Each read voltage is centered between two adjacent distributions so that read margins are maximized. For instance, the distance between V_{READ1} and the rightmost side of "10" distribution should be equal to the distance between V_{READ1} and the leftmost side of "00" distribution. With multilevel memories, the typical value for such distances is 300 mV.

In order to achieve the required precision, voltages to be applied to the cells are generated by means of voltage regulators which exploit band-gap techniques to generate a precise reference voltage. In this way, the voltages generated on-chip are independent from temperature, at least to a first approximation. On the other hand, the V_{TH} distributions of the memory cells are highly sensitive to temperature variations: as temperature increases, V_{TH} decreases and vice versa (see Fig. 6.29).

As a result, read margins are reduced when temperature varies, because the tails of the distributions get nearer and nearer to read voltages. For instance, as shown in Fig. 6.29, "00" distribution gets nearer to V_{READ2} at low temperature, while it gets nearer to V_{READ1} at high temperature. The same is true for each distribution. Threshold voltage of the cell typically shifts of −1.5 mV/°C. As a consequence, overall variation is approximately 200 mV if a temperature range of −40°C to 90°C is considered.

Therefore, a specific type of read voltage regulator is needed [26–28]: that is, the thermal coefficient of its output voltage has to be as similar as possible

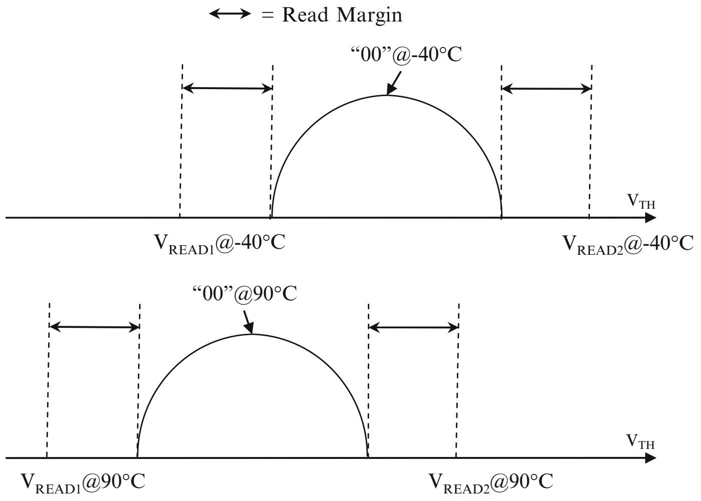


Fig. 6.30 V_{READ} tracking of V_{TH} variations with temperature

to the coefficient of the cell's V_{TH}. In this way, read voltages rigidly shift with distributions, keeping the margins unaltered. (Fig. 6.30). A similar constraint is true for verify voltages.

6.6 Program

As described in Chap. 5, V_{TH} is modified by means of the *Incremental Step Pulse Programming* (ISPP) algorithm (Fig. 6.31): a voltage step (whose amplitude and duration are predefined) is applied to the gate of the cell. Afterwards, a verify operation is performed, in order to check whether V_{THR} has exceeded a predefined voltage value (V_{VFY}). If the verify operation is successful, the cell has reached the desired state and it is excluded from the following program pulses. Otherwise another cycle of ISPP is applied to the cell, where the program voltage is incremented by ΔVpp.

During the program operation, the cells share the high programming voltage on the selected wordline but the program operation has to be bit selective. Therefore, a high channel potential is needed to reduce the voltage drop across the tunneling dielectric and prevents the electrons tunneling from the channel to the floating gate as indicated by Fig. 6.32a. In the first NAND flash devices the channel was charged by applying 8 V to the bitlines of the program inhibited NAND strings. This method suffers from several disadvantages [29], especially power consumption and high stress on the oxide between adjacent bitlines.

The self boost program inhibit scheme is less power consuming. By charging the string select lines and the bitlines connected to inhibited cells to V_{cc}, the

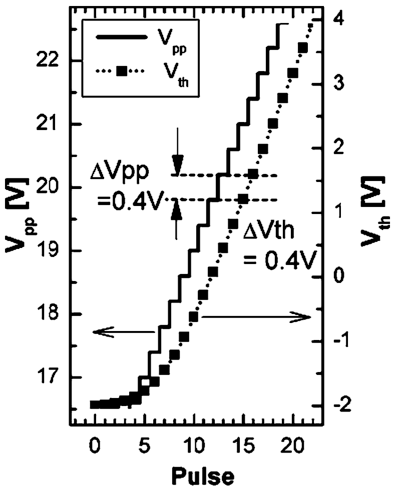


Fig. 6.31 Incremental Step Pulse Programming (ISPP): constant V_{TH} shift

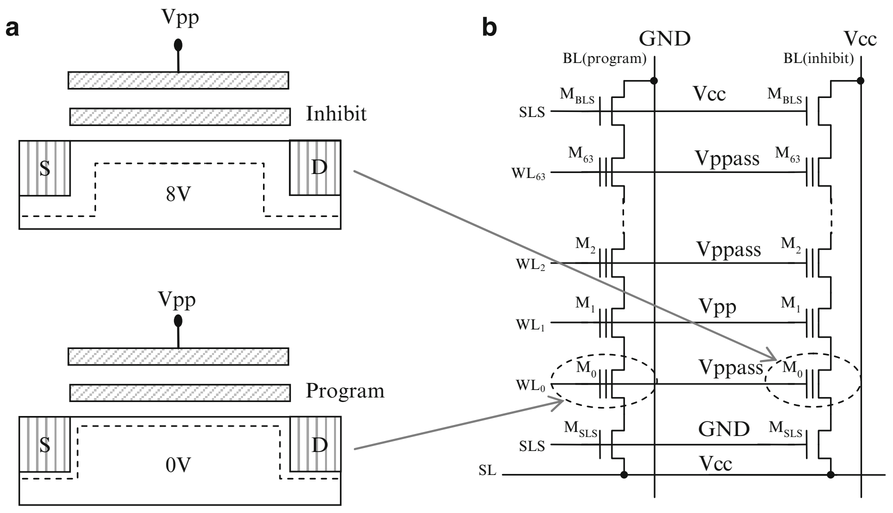


Fig. 6.32 Self boosted program inhibit scheme, (**a**) Cell in program/inhibit state, (**b**) strings biasing in program/inhibit state

select transistors are diode connected (Fig. 6.32b). By raising the wordline potential (selected wordline to V_{pp} and unselected wordlines to V_{ppass}) the channel potential is boosted by the coupled series capacitance through the control gate, floating gate, channel and bulk.

In fact, when the voltage of the channel exceeds $V_{cc} - V_{TH,SSL}$, then SSL transistors are reverse biased and the channel of the NAND string becomes a floating node.

Two important typologies of disturbs are related to the program operation: the *Pass disturb* and the *Program disturb,* as described in Chap. 5.

6.7 Erase

The erase operation resets the information of all the cells belonging to one block simultaneously.

Tables 6.1 and 6.2 summarize the erase voltages. During the erase pulse, all the wordlines belonging to the selected block are kept at ground, the matrix ip-well must rise (through a staircase) to 23 V and all the other nodes are floating. This phase lasts almost a millisecond and it is the phase when the actual electrical erase takes place.

Since the matrix ip-well (as well as the surrounding n-well) is common to all the blocks, it reaches high voltages also for the unselected blocks. In order to prevent an unintentional erase on those blocks, wordlines are left floating; in this way, their voltage can rise thanks to the capacitive coupling between the wordline layer and the underneath matrix layer. Of course, the voltage difference between wordlines and ip-well should be low enough to avoid Fowler-Nordheim tunneling.

After each erase pulse an erase verify (EV) follows. During this phase all the wordlines are kept at ground. The purpose is verifying if there are some cells that have a V_{TH} higher than 0 V, so that another erase pulse can be applied. If EV isn't successful for some columns of the block, there are some columns too programmed.

Table 6.1 Electrical erase pulse voltages for the selected block

	T_0	T_1	T_2	T_3	T_4
BLeven	Float	Float	Float	Float	Float
BLodd	Float	Float	Float	Float	Float
DSL	Float	Float	Float	Float	Float
WLs	0 V	0 V	0 V	0 V	0 V
SSL	Float	Float	Float	Float	Float
SL	Float	Float	Float	Float	Float
ip-well	0 V	V_{ERASE}	V_{ERASE}	0 V	0 V

Table 6.2 Electrical erase pulse voltages for unselected blocks

	T_0	T_1	T_2	T_3	T_4
BLeven	Float	Float	Float	Float	Float
BLodd	Float	Float	Float	Float	Float
DSL	Float	Float	Float	Float	Float
WLs	Float	Float	Float	Float	Float
SSL	Float	Float	Float	Float	Float
SL	Float	Float	Float	Float	Float
ip-well	0 V	V_{ERASE}	V_{ERASE}	0 V	0 V

If the maximum number of erase pulses is reached (typically 4), than the erase exits with a fail. Otherwise, the voltage applied to the matrix ip-well is incremented by ΔV_E and another erase pulse follows.

6.8 MLC and XLC Storage

The obvious advantage of a 2 bit/cell implementation (MLC) with respect to a 1 bit/cell device (SLC) is that the area occupation of the matrix is half as much; on the other hand, the area of the periphery circuits, both analog and digital, increases. This is mainly due to the fact that the multilevel approach requires higher voltages for program (and therefore bigger charge pumps), higher precision and better performance in the generation of both the analog signals and the timings, and an increase in the complexity of the algorithms.

Figure 6.33 shows an example of how 2 bits are associated to the four read threshold distributions stored in the cell, and how the set of programmed distributions is built starting from the erased state "E". In this case the multilevel is achieved in two distinct rounds, one for each bit to be stored [2], [30, 31].

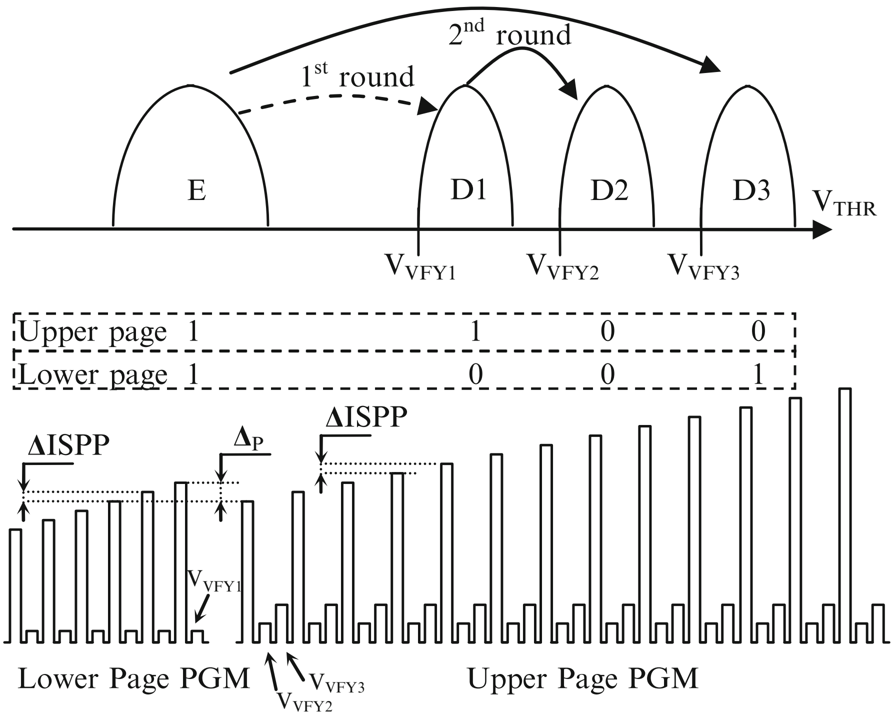


Fig. 6.33 Two rounds MLC program operation

In the first round, the so-called lower-page (associated to the *Least Significant Bit* – LSB) is programmed. If the bit is "1", the read threshold of the cell V_{TH} does not change and, therefore, the cell remains in the erased state, E. If the bit is "0", V_{TH} is increased until it reaches the D1 state.

In the second round, the upper-page (associated to the *Most Significant Bit* – MSB) is programmed. If the bit is "1", V_{TH} does not change and, therefore, the cell remains either in the erased state, E, or in the D1 state, depending on the value of the lower-page.

When MSB is "0", V_{TH} is programmed as follows:

– if, during the first round, the cell remained in E state, then V_{TH} is incremented to D3;
– if, during the first round, the cell was programmed to D1, then, in the second round, V_{TH} reaches D2.

As usual, the program operation makes use of ISPP, and the verify voltages are V_{VFY2} and V_{VFY3}. Lower-page programming only needs the information related to LSB, while for the upper-page it is necessary to know both the starting distribution (LSB) and the MSB.

Because of technological variations, V_{TH} is not perfectly related to the amplitude of the program pulse (during ISPP): there are "fast" cells which reach the desired distribution with few ISPP pulses, while other "slow" cells require more pulses.

The amplitude of the first program pulse ($V_{PGMLSB0}$) of the lower-page should not allow the threshold V_{THR} of the "fastest" cell to exceed V_{VFY1}. If it happens, an undesired widening of distribution D2 occurs or, in the worst case scenario, V_{THR} might reach D2 distribution at once.

Typical $V_{PGMLSB0}$ is around 16 V. In case of program of "slow" cells from E to D1, the last programming step needs values as high as 19 V. Assuming ΔISPP equal to 250 mV, it takes 12 steps to move from 16 to 19 V.

Similarly, the starting pulse of the upper-page $V_{PGMMSB0}$ should have an amplitude such that the "fastest" cell does not go beyond V_{VFY2}.

$$V_{PGMMSB0} = V_{PGMLSB0} + (V_{VFY2} - V_{VFY1}) \tag{6.19}$$

The value of $V_{VFY2} - V_{VFY1}$ is typically around 1 V and, therefore, the initial voltage is about 17 V.

As shown in Fig. 6.33, the upper-page ISPP does not start from the last voltage used for the lower-page programming, but it begins at $V_{PGMLSB0} - \Delta_P$. For example, instead of starting at 19 V, it could start at 17 V, eight steps below.

Driven by cost, Flash manufacturers are now developing 3 bit/cell (8 V_{TH} distributions) and 4 bit/cell (16 V_{TH} distributions) [32–34]. Three and four bits per cell are usually referred to as XLC (8LC and 16LC, respectively). Unfortunately, due to reliability reasons, the V_{TH} window remains the MLC one; in fact, the highest verification level must be low enough to prevent bit failures caused by program disturb and read disturb. The more states a memory cell is made to store, the more finely divided is its V_{TH} window.

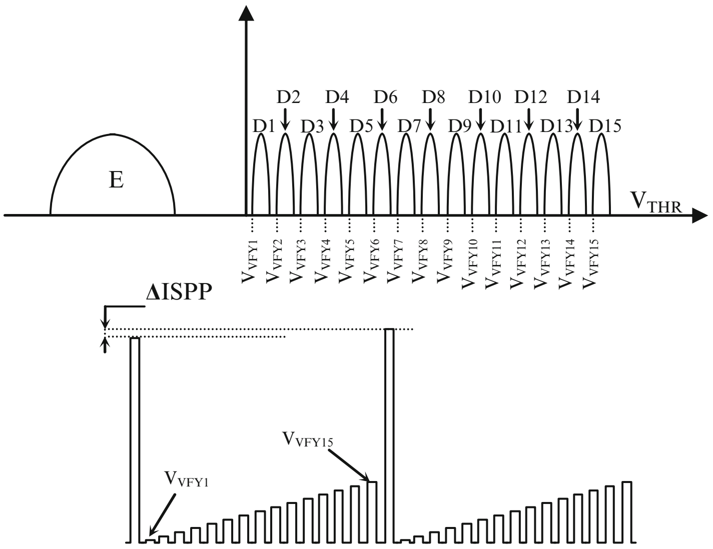


Fig. 6.34 4 bit/cell programming algorithm

Of course, the main drawback is a slow program time. As the distribution width needs to be tighter, ISSP program step is smaller and the number of verify operations increases, as depicted in Fig. 6.34.

6.9 High Voltage Management

Modifying or reading the number of electrons stored into the floating gate requires a big set of voltages. The High Voltage (HV) system has to provide all these voltages with the desired precision, timing and granularity. On top of that, many voltages have a value greater than the NAND power supply VDD, asking for an on-chip charge pump. This section deals with the HV basic building blocks.

6.9.1 Charge Pumps

In the NAND environment, one of the most used type of charge pumps is the Voltage Doubler [3]. The basic stage is shown in Fig. 6.35. It is a feedback system that can duplicate the input voltage and, essentially, it is made up by two n-channel transistors (MN1, MN2), two p-channel transistors (MP1, MP2) and two capacitors (C1, C2) of the same size.

In order to understand the principle of operation of this circuit, it can be assumed that, at the beginning, nodes A and B, as well as CK (pump clock) and

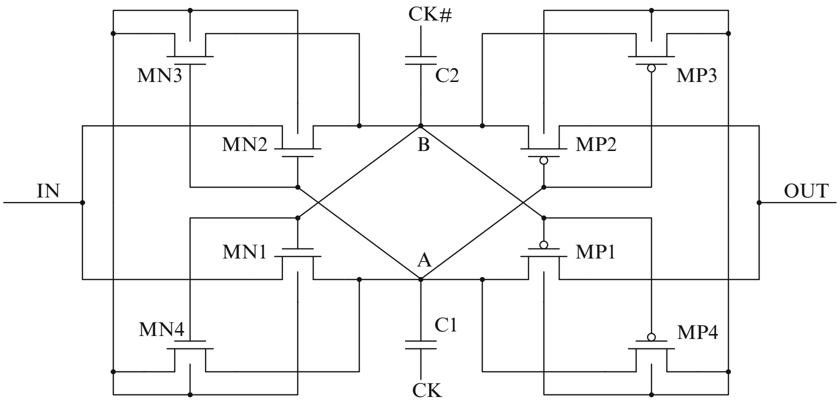


Fig. 6.35 Basic stage of a voltage doubler

its complement (CK#), are at GND. In this way, both transistors MN1 and MN2 are off. Voltage on the node IN (V_{IN}) is set to VDD (i.e. the chip power supply).

As soon as CK toggles from GND to VDD, V_A becomes VDD, activating transistor MN2. Since CK# remains at GND, the charge starts flowing from power supply to capacitor C2 until V_B reaches a value equal to VDD – $V_{TH,MN2}$. When CK goes to GND, transistor MN2 turns off.

At the same time, CK# gets to VDD and, therefore, V_B becomes (VDD – $V_{TH,MN2}$ + VDD), turning on transistor MN1. As a result, C1 is charged up to VDD. Of course, when CK# goes to GND again, V_B is, in principle, equal to VDD – $V_{TH,MN2}$. Since the signal CK is used as a clock, each capacitance is continuously charged and discharged between VDD and 2VDD. In other words, during each period of the clock either V_A or V_B is at 2VDD.

At this point, in order to build a real charge pump, voltages on nodes A and B have to be transferred to the next pump stage. Now MP1 and MP2 come into the game. When CK is at VDD, V_A is 2VDD and V_B is VDD. Transistor MN1 is, therefore, turned off while MP1 is active, transferring the voltage of node A to node OUT. In the meanwhile MP2 is off, MN2 is on and the capacitor C2 is charged up. When CK goes back to GND and CK# becomes VDD, then the circuit behaves in the opposite way: MN1 and MP2 are active (the former charges capacitor C1, the latter transfers the voltage of node B to the output) while MN2 and MP1 are turned off. It is worth to note that no active direct paths between IN and OUT are allowed: these paths would result in a loss of charge and, therefore, in a reduced output voltage.

As usual, when designing a charge pump, one issue to cope with is the biasing of the transistor body terminals. The easiest solution is to connect the body of the n-channel transistor to the power supply and the body of the p-channel transistor to the output node.

The drawback of this solution is that the output voltage is considerably reduced by the body-effect of the transistors itself. In Fig. 6.35a "dynamic biasing" has been chosen: bodies are continuously switched between V_A and V_B. As a result, the body

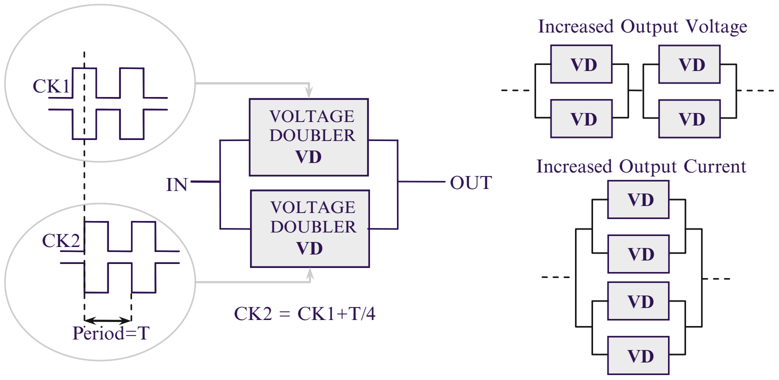


Fig. 6.36 Charge pump as a cascade of basic voltage doubler stages

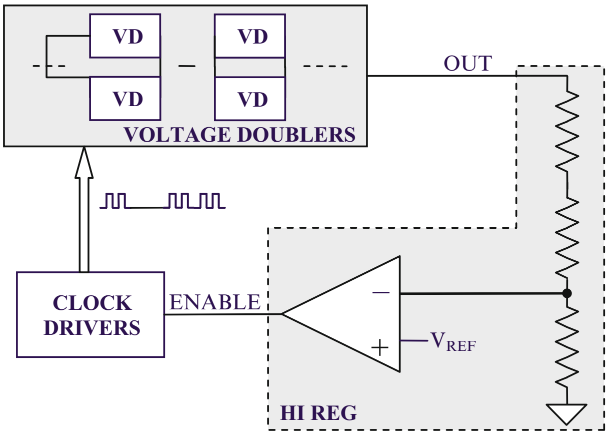


Fig. 6.37 Charge pump architecture

of the NMOS transistors is always kept at the lowest voltage (through MN3 and MN4) while the body of the PMOS transistors is always at the highest voltage (through MP3 and MP4).

The basic stage of Fig. 6.35 can be used to build up more complex structures as depicted in Fig. 6.36. Usually, two stages are used in parallel in order to decrease the ripple of the output voltage.

In fact, due to the internal switching activity of the capacitors, the output of the pump can be more or less noisy. When talking about ripple, we generally refer to the height of the "peaks" that can be found in the output node waveform.

In order to properly control the output voltage, voltage doubler stages are inserted in a feedback loop as described in Fig. 6.37. A block called "Hireg" is used to limit the output voltage. Thanks to a resistive divider (it could also be made by CMOS diodes), the output voltage is compared with V_{REF} (usually a band-gap reference voltage). CK drivers are then enabled/disabled depending on the comparison result.

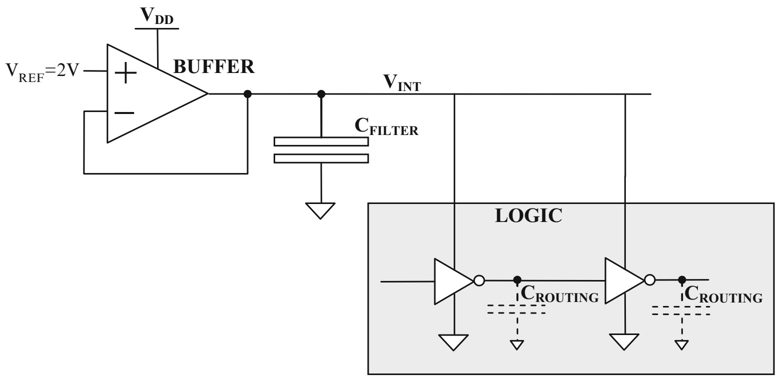


Fig. 6.38 Conceptual scheme of a DC-DC converter

In order to find the best configuration, the output voltage of the charge pump is measured varying the CK period. A faster clock means higher output voltage, but faster clocks means bigger area of the CK drivers. The right trade-off has to be found considering that, in most of the NAND applications, silicon cost is the main driver. Optimum CK period is usually in the range of 60÷80 ns considering an output resistance of around 10 kΩ. The voltage doubler pump can easily achieve voltages above 25 V starting from the chip VDD of 2.5 V. Power efficiency η_P can be as high as 20÷30% if the current load remains in the range of few hundreds microAmpere.

$$\eta_P = \frac{V_{OUT} \cdot I_{OUT}}{V_{IN} \cdot I_{IN}} \tag{6.20}$$

6.9.2 Internal Supply Voltage Regulator

In many NAND devices, external supply voltage VDD is not directly applied to all the circuits [35, 36]. Some of them are powered by an internal supply (V_{INT}) filtered by a proper voltage regulator and this solution brings several advantages. For instance, in case of devices supplied at 3.6 V, a V_{INT} equal to 2 V allows the use of transistors whose oxide thickness is reduced, which are smaller and better performing. In the case of page buffers, by using V_{INT} it is possible to mitigate the dependency of the triggering threshold from VDD (i.e. several tens of milliVolt), which turns into a reduction of the width of the distributions. Of course, inside the NAND memory, there could be more than one V_{INT} regulators, depending on the design constraints (noise, power consumption, precision required by the circuits).

V_{INT} regulator is a DC-DC converter. Its conceptual scheme is shown in Fig. 6.38. For the sake of simplicity, VDD supplies only logic ports. When inverters are switching, voltage drop of V_{INT} is a function of the filtering capacitance C_{FILTER}, of the parasitic capacitance (gates, routing, junctions), and of the cross-conduction current.

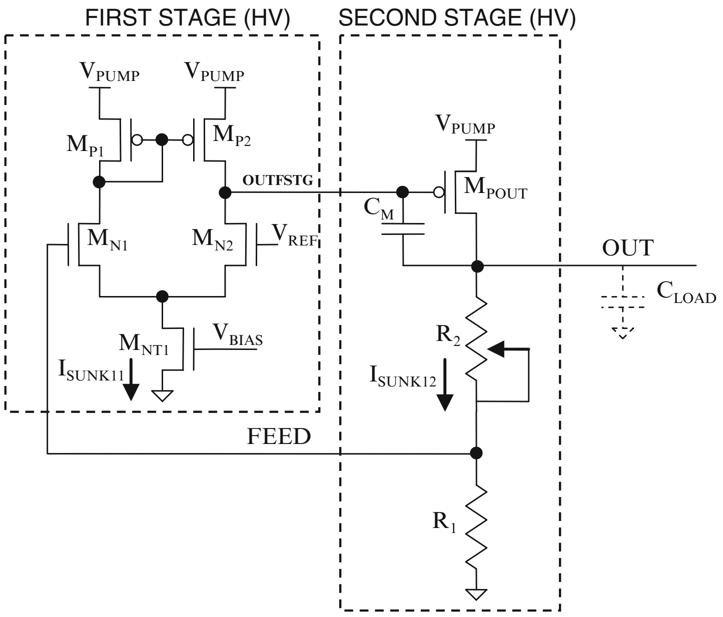


Fig. 6.39 Voltage regulator with high voltage PMOS

Beyond a given maximum switching frequency of the logic, V_{INT} dramatically drops. This frequency is directly related to the cutoff frequency of the regulator. Since the DC-DC converter is designed using the same technology of the inverters, its cutoff frequency cannot be higher than the one of the plain inverter.

6.9.3 Double-Supply Voltage Regulator

Both program and erase operations require voltages higher than VDD. For instance, the programming staircase voltage starts at 14÷15 V and arrives at 25 V and beyond. High voltages are generated by a charge pump and filtered by a proper voltage regulator: in this way it is possible to reduce the ripple and obtain the desired output voltage value.

In 1 bit/cell Flash memories, voltage regulator is omitted and the output voltage of the pump is directly used, regulated by means of an on-off type of control. Typical ripple values are in the order of 1÷2 V. In case of multilevel memories, the target voltage precision cannot be achieved without a voltage regulator.

NAND technology does not usually provide High Voltage (HV) PMOS transistor; therefore; it is not possible to implement traditional voltage regulators like the one shown in Fig. 6.39. In fact, the use of a low-voltage transistor for M_{POUT} would

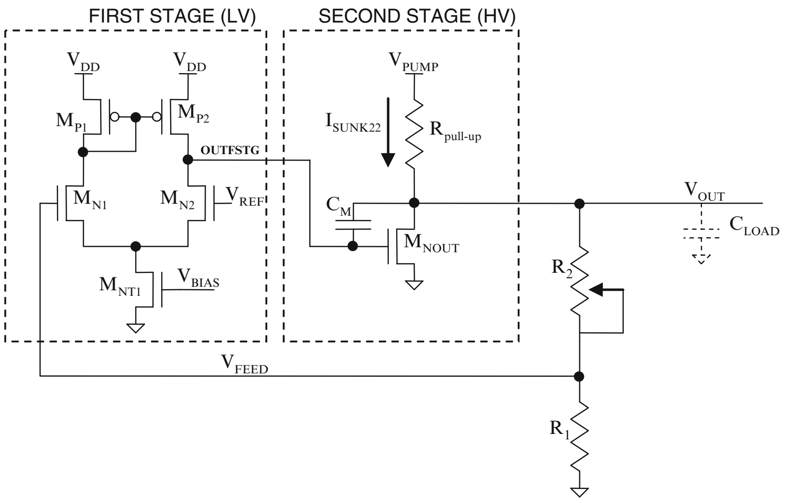


Fig. 6.40 Double-supply voltage regulator

mean that the voltage drop across its terminals must be guaranteed not to exceed 4÷5 V. This must be true both in static and in transient conditions. On top of that, all the required values for the staircase program pulse must be generated out of the pump output voltage (∼30 V), beginning at 15 V: that is, M_{POUT} must be a HV transistor.

In order to solve the issue it is possible to design a voltage regulator [37] whose first differential stage is supplied by VDD, while the second one is supplied by a charge pump so that the HV value can be provided at the output (Fig. 6.40).

By supplying the first stage with VDD, PMOS LV transistors can be used to realize the current mirror (M_{P1} - M_{P2}). The second stage is instead designed using an NMOS HV (M_{NOUT}) together with a resistive pull-up ($R_{pull\text{-}up}$).

6.10 Wordline Decoder

One of the most critical circuits of the *High Voltage* (HV) system is the one used to bias the *WordLine* (WL). Actually, when it comes to NAND memories, a single wordline is not enough: all the wordlines belonging to the same NAND string must be properly biased at the same time. As a result, the Row Decoder, also called Wordline Decoder or Wordline Driver [4], has to provide a set of voltages: these values are defined by the algorithms described in Sects. 6.5 and 6.6.

When NAND technology provides only NMOS-type HV transistors, a possible implementation of the wordline driver is shown in Fig. 6.41. The wordline driver comprises:

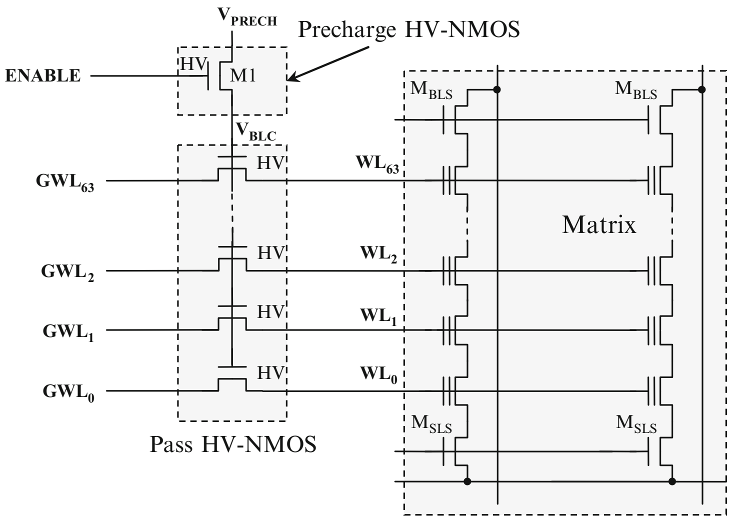


Fig. 6.41 All-NMOS wordline driver

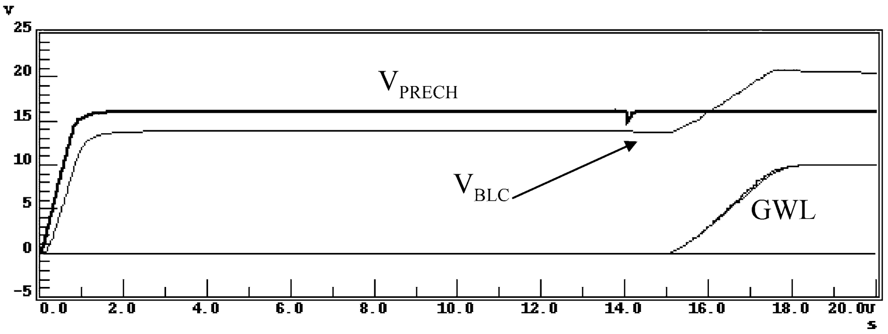


Fig. 6.42 Simulation of the circuit sketched in Fig. 6.39

- a *Pass-Transistor* (PT) for each wordline. These transistors are used to transfer voltages from the *Global WordLines* (GWLs), i.e. electrical signals, to the physical wordlines (WLs);
- a circuit to bias the gates of the above mentioned pass-transistors.

The biasing circuit of the gate of PTs consists of only one high voltage NMOS (M1). At first, all the gates are biased at a high voltage V_{PRECH} through M1. Then, M1 is switched off and, thanks to the gate-drain parasitic capacitance, the rising transient of GWL performs a boost of V_{BLC}, switching PTs on, as shown in Fig. 6.42.

However, there are several critical aspects to consider. First of all, the designer has to deal with a precharge phase of the PT gates: this phase must occur before biasing the global wordlines, otherwise the boost effect would be lost.

The precharge voltage V_{PRECH} has to match V_{MAX}, which is the maximum voltage required during each algorithm. V_{MAX} is not an issue during the read operation, when the voltages are relatively low, but it ends up being close to the breakdown voltage during the program operation. The duration of the precharge phase must be calibrated to allow V_{PRECH} reaching V_{MAX}: this time increases the overall operation time, especially during programming.

With reference to the circuit of Fig. 6.41, precharge is driven by the ENABLE signal. To fully exploit the precharge benefit, ENABLE has to be biased with a voltage greater than V_{PRECH}, in order to recover the threshold voltage $V_{TH,M1}$ of transistor M1.

Particular attention deserves the boost operation. Once the boost has occurred, V_{BLC} has to guarantee that, even varying temperature and technological parameters, each GWL and its corresponding WL are biased with the same voltage. Unfortunately, process and temperature variations mean that the V_{TH} of the pass transistors can vary as much as 100%. Therefore, the risk is to overcome the breakdown voltage of the oxide in some PVT (Process Voltage Temperature) corners allowed by the electrical specification of the NAND Flash memory.

Designers have developed a lot of different solutions for the row decoder, including a hierarchical approach [2, 3, 38]: due to the huge numbers of wordlines contained in a NAND array, the challenge is always to trade off performances with silicon area.

At this point the reader should be reasonably convinced that a NAND Flash memory is not a "pure" digital device: it is a real mix of digital and analog circuits, working at high and low voltages, and designed on a silicon technology developed for floating gate transistors . . . have fun!

References

1. R. Micheloni et al., *A* 4Gb 2b/cell NAND Flash memory with embedded 5b BCH ECC for 36 MB/s system read throughput, in *IEEE International Solid-State Circuits Conference 2006, Digest of Technical Papers, ISSCC 2006*, Feb 2006, pp. 497–506
2. R. Micheloni, L. Crippa, A. Marelli, *Inside NAND Flash Memories* (Springer, New York, 2010)
3. G. Campardo, R. Micheloni, D. Novosel, *VLSI-Design of Non-Volatile Memories* (Springer, New York, 2005)
4. P. Cappelletti, C. Golla, P. Olivo, E. Zanoni (eds.), *Flash Memories* (Kluwer, Boston, 1999), ch. 5
5. R. Micheloni, A. Marelli, R. Ravasio, *Error Correction Codes for Non-Volatile Memories* (Springer, Dordrecht, 2008)
6. G. Campardo et al., An overview of Flash architectural developments. Proc. IEEE **91**(4, April), 523–536 (2003)
7. M. Annaratone, *Digital CMOS Circuit Design* (Kluwer Academic Publishers, Boston, 1986)
8. www.onfi.org
9. https://www.denali.com/en/events/webcasts/2008/togglenand/
10. A. Chandrakasan, R. Brodersen (eds.), *Low Power CMOS Design* (Kluwer Academic Publishers, Boston, 1995)

11. H. Hikeda, A 3D packaging with 4Gb chip-stacked DRAM and 3Gbps high-speed logic, 3D-SIC 2007 (INTERNATIONAL 3D-SYSTEM INTEGRATION CONFERENCE 2007) Tokyo, Japan
12. T. Wada, M.E. Kenji Mami, Simple noise model and low-noise data-output buffer for ultrahigh-speed memories. IEEE J. Solid-State Circuits **25**(6, December), 1586–1588 (1990)
13. S. Dabral, T. Maloney, *Basic ESD and I/O Design* (Wiley, New York, 1998)
14. E. Chioffi, F. Maloberti, High-speed, low-switching noise CMOS memory data output buffer. IEEE J. Solid-State Circuits **29**(11, November), 1359–1365 (1994)
15. S.H. HallGarrett, W. HallJames, A. McCall, *High-Speed Digital System Design-A Handbook of Interconnect Theory and Design Practices* (Wiley, New York, 2000)
16. P. Heydari, M. Pedram, Ground Bounce in digital VLS circuits. IEEE Trans. VLSI Syst. **11**(2, April), 180–193 (2003)
17. R. Senthinathan, J. Prince, Simultaneous switching ground noise calculation for packaged CMOS devices. IEEE J. Solid-State Circuits **26**(November), 1724–1728 (1991)
18. R. Senthinathan, J.L. Prince, *Simultaneous Switching Noise of CMOS Devices and Systems* (Kluwer Academic Publisher, Boston, 1994)
19. S.J. Jou et al., Low switching noise and load-adaptive output buffer design techniques. IEEE JSSC **36**, 1239–1249 (2001)
20. B. Deutschmann, T. Ostermann, CMOS output driver with reduced ground bounce and electromagnetic emission, in *Solid-State Circuits Conference*, *ESSCIRC'03*, (New York, 2003)
21. Y. Itoh, et al., An experimental 4 Mb CMOS EEPROM with a NAND structured cell, in *36th IEEE International Solid-State Circuits Conference 1989, Digest of Technical Papers, ISSCC 1989*, San Francisco, Feb 1989, pp. 134–135
22. T. Tanaka et al., A quick intelligent page-programming architecture and a shielded bitline sensing method for 3 V-only NAND flash memory. IEEE J. Solid-Stare Circuits **29**(11, November), 1366–1373 (1994)
23. T.-S. Jung et al., A 3.3 V 128 Mb multi-level NAND Flash memory for mass storage applications, in *43rd IEEE International Solid-State Circuits Conference 1996, Digest of Technical Papers, ISSCC 1996*, San Francisco, Feb 1996 pp. 32–33, 412
24. K. Imamiya et al., A 130 mm^2 256 Mb NAND Flash with shallow trench isolation technology, in *IEEE International Solid-State Circuits Conference 1999*, *Digest of Technical Papers*, *ISSCC 1999*, Feb 1999, pp. 112–113, 412
25. R.A. Cernea et al., A 34 MB/s MLC write throughput 16 Gb NAND with all bit line architecture on 56 nm technology. IEEE J. Solid-Stare Circuits **44**(1, January), 186–194 (2009)
26. L. Crippa, G. Ragone, M. Sangalli, R. Micheloni, Circuit and method for retrieving data stored in semiconductor memory cells, U.S. Patent No.7474577 Assignee: STMicroelectronics/Hynix Semiconductor
27. T. Tanzawa, T. Tanaka, K. Takeuchi, Nonvolatile semiconductor memory with temperature compensation for read-verify referencing scheme, U.S. Patent No.5864504 Assignee: Kabushiki Kaisha Toshiba (Kawasaki, JP)
28. T.-H. Cho, Y.-T. Lee, Multi-level flash memory with temperature compensation, U.S. Patent No.6870766 Assignee: Samsung Electronics Co., Ltd. (Suwon-si, KR)
29. K.-D. Suh et al., A 3.3 V 32 Mb NAND flash memory with incremental step pulse programming scheme. IEEE J. Solid-State Circuits **30**(11, November), 1149–1156 (1995)
30. S. Lee et al., A 3.3 V 4Gb four-level NAND Flash memory with 90 nm CMOS technology, in *IEEE International Solid-State Circuits Conference*, *ISSCC*, *Digest of Technical Papers*, San Francisco, vol. 1, Feb 2004, pp. 52–53, 513
31. D.-S. Byeon et al., An 8Gb multi-level NAND Flash memory with 63 nm STI CMOS process technology, in *Solid-State Circuits Conference*, *ISSCC*, *Digest of Technical Papers*, San Francisco, vol. 1, Feb 2005, pp. 46–47
32. Y. Li et al., A 16Gb 3b/cell NAND Flash memory in 56 nm with 8 MB/s write rate, in *IEEE International Solid-State Circuits Conference 2008*, *Digest of Technical Papers*, *ISSCC 2008*, San Francisco, Feb 2008, pp. 506–507, 632
33. N. Shibata et al., A 70 nm 16 Gb 16-Level-Cell NAND flash memory. IEEE J. Solid-Stare Circuits **43**(4, April), 929–937 (2008)

34. C. Trinh et al. A 5.6 MB/s 64Gb 4b/Cell NAND Flash Memory in 43 nm, CMOS, in *IEEE International Solid-State Circuits Conference 2009*, *Digest of Technical Papers*, *ISSCC 2009*, San Francisco, Feb 2009, pp. 246–247
35. K. Takeuchi et al., A 56-nm CMOS 99-mm^2 8-Gb multi-level NAND flash memory with 10-MB/s program throughput. IEEE J. Solid-Stare Circuits **42**(1, January), 219–232 (2007)
36. G.A. Rincon-Mora, *Analog IC Design with Low-Dropout Regulators*. Electronic Engineering (McGraw-Hill, New York, 2009)
37. L. Crippa, M. Sangalli, G. Ragone, R. Micheloni, Multistage regulator for charge-pump boosted voltage applications, not requiring integration of dedicated high voltage high side transistors, U.S Patent App. 20070164811, Assignee: STMicroelectronics/Hynix Semiconductor
38. K. Kanda et al., A 120 mm^2 16 Gb 4-MLC NAND with 43 nm CMOS Technology, in *2008 IEEE International Solid-State Circuits Conference (ISSCC)*, *Digest of Technical Papers*, San Francisco, Feb 2008, pp. 430–431

Chapter 7
NAND and Controller Co-design for SSDs[*]

K. Takeuchi

Abstract SSD is made up by NAND Flash memories, DRAMs and a NAND controller. To realize a low-power high-speed SSD, the overall performance of the NAND Flash memory and the NAND controller should be optimized by co-designing both NAND and controller circuits. This chapter describes the most advanced circuits in this field.

Furthermore, 3D-integration in the SSD system becomes a key topic and an example of low power 3D-integrated SSD is shown.

Finally, a couple of techniques, *Asymmetric Coding* and *Stripe Pattern Elimination Algorithm*, for reducing the NAND raw BER are presented.

7.1 Introduction

With highly scaled technologies below 20 nm, the memory capacity increases to several Gbit as shown in Fig. 7.1. By using Gbit-capacity NAND Flash memories, SSDs, *Solid-State Drives*, that uses NAND as a mass storage of personal computers and enterprise servers are the new killer application of NAND Flash memories.

The hardware architecture of SSD is shown in Fig. 7.2. SSD is composed of NAND Flash memories, DRAMs and a NAND controller. To realize a low power high speed SSD, the overall performance of the NAND Flash memory and the

[*]This chapter is an updated version of "Low power 3D-integrated SSD", Chap. 18, *Inside NAND Flash Memories*, Springer, 2010.

K. Takeuchi (✉)
Department of Electrical, Electronic and Communication Engineering, Chuo University, Tokyo, Japan
e-mail: takeuchi@takeuchi-lab.org

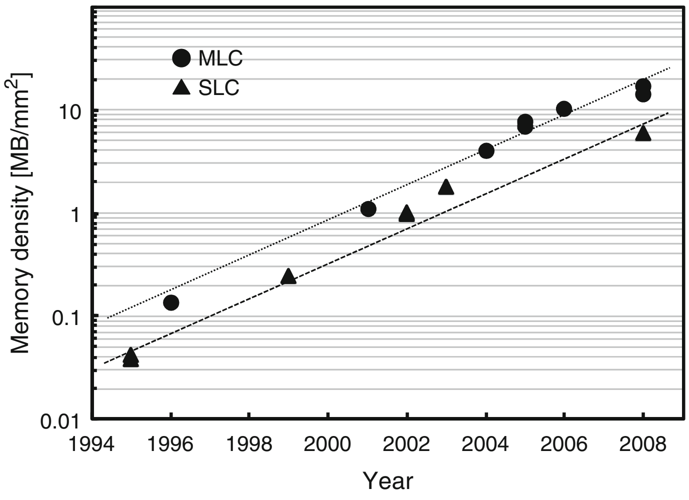


Fig. 7.1 Memory density trend of a NAND Flash memory

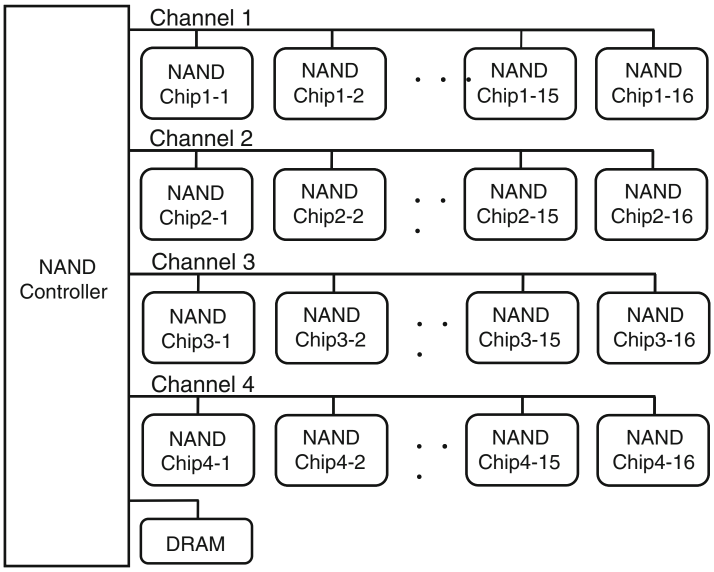


Fig. 7.2 Hardware architecture of SSD

NAND controller should be maximized by co-designing NAND Flash memory and NAND controller circuits [1–3]. Furthermore, an intelligent 3D-integration of various circuits in SSD such as the NAND controller, the NAND Flash memory and voltage generators are essential [4].

7.2 Analysis of SSD Performance

As shown in Fig. 7.2, SSD contains more than 8 NAND chips. For example, in Fig. 7.2, each NAND Flash memory is assumed to have 16 Gbit or 2 GByte capacity. One SSD is composed of 64 NAND chips and the total capacity is 128 GBytes, which is most suitable for the lap top computers. In Fig. 7.2, 4-channel configuration is adopted. The NAND controller can operate four channels independently.

The NAND controller can issue a write command to 4 channels, writing 4 NAND chips at the same time and improving the system-level write performance.

In a channel, 16 NAND Flash memory chips are connected together.

I/O signals and control signals such as the WE (Write Enable)-signal, the RE (Read Enable)-signal, the CLE (Command Latch Enable)-signal, and the ALE (Address Latch Enable)-signal are shared to save area for the bonding and the packaging. To select one NAND chip in a channel, different CE (Chip Enable)-signals and R/B (Ready/Busy)-signals are assigned to the NAND chips in a channel.

Figure 7.3 shows the software architecture of SSD. The NAND controller is composed of the host interface, the *Flash Translation Layer* (FTL) and the NAND interface. FTL is the central part of the NAND controller and perform a bad block management, a logical-physical address translation, a wear-leveling, ECC and an interleaving. The intelligent write algorithm described in Sects. 7.5 and 7.6 is also controlled with FTL.

In a NAND Flash memory, the programming is slower than the read by one order of magnitude [1]. The typical random read time of a 2 bit/cell NAND Flash memory is 50 μs while the typical random program time is 800 μs. Considering the read/write access time of HDD is a few milliseconds, the key design challenge of SSD is to improve the write performance. To accelerate the write speed of SSD, an interleaving is proposed where multiple NAND chips in SSD are programmed at the same time [5]. The write speed of SSD with an interleaving is expressed as follows:

$$Performance\_SSD = N \cdot Performance\_NAND \tag{7.1}$$

N is the number of NAND Flash memories operated simultaneously. *Performance_NAND* is the program speed of one NAND Flash memory. As a memory cell is scaled down or more bits are stored in a memory cell such as 3 bit/cell or 4 bit/cell, a more precise control of the memory cell V_{TH} is required and accelerating the NAND speed, *Performance_NAND*, becomes difficult. Actually, the program time of the 56 nm 3 bits per cell is 1.2 ms [6] and that of the 70 nm 4 bits per cell is 6 ms [7] which are much longer than that of the 2 bits per cell. Thus, the best strategy to improve the SSD speed is to increase the number of NAND chips in parallel, *N*. However, if *N* is maximized and all NAND chips in SSD, for example 64 NAND chips, are programmed simultaneously, 64 times as much current flows and the current consumption increases to an unacceptable high value, two amperes. Therefore, in an actual SSD operation, the maximum number of *N* is restricted by the current consumption constraint.

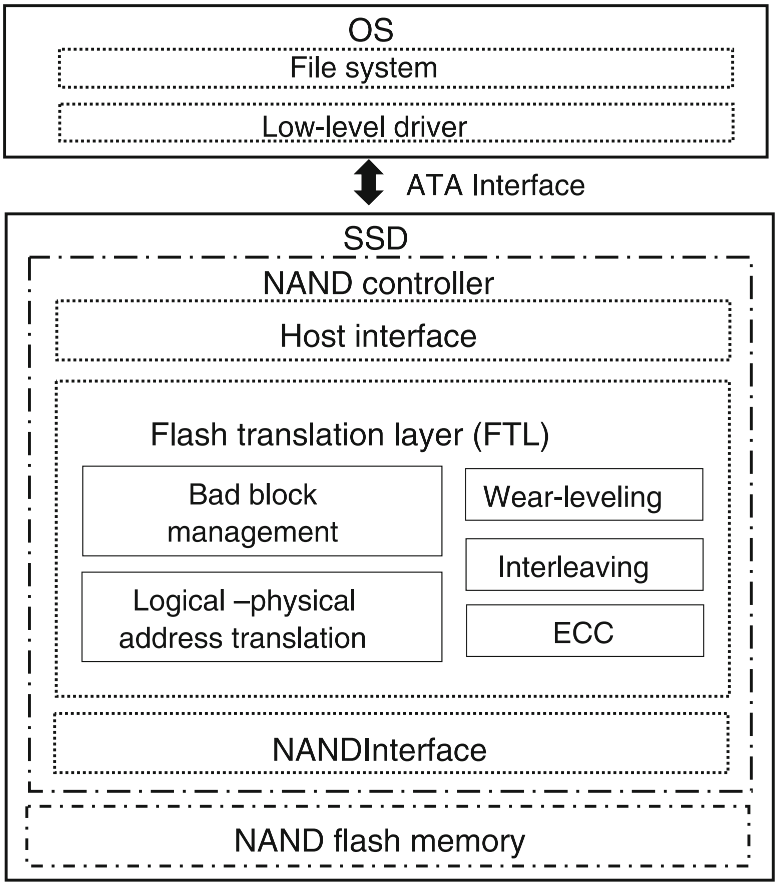


Fig. 7.3 Software architecture of SSD

As the NAND cell is scaled down, the bit-line capacitance drastically increases [8] and consequently the current consumption increases. Figure 7.4 shows the schematic diagram of the NAND Flash memory cell array. The bit-line is arranged with the minimum feature size. Figure 7.5 describes the cross sectional view of the bit-line and the memory cell. As the design rule is decreased, the space between the bit-lines is also decreased. On the other hand, the height of the bit-line is not decreased to keep the low resistance of the bit-line. As a result, the inter bit-line capacitance, $C_{BL\text{-}BL}$ shown in Fig. 7.5a increases and the bit-line capacitance also increases as shown in Fig. 7.6. The total bit-line capacitance in a chip exceeds 200 nF and the current consumption to precharge the huge bit-line capacitance increases as shown in Fig. 7.7.

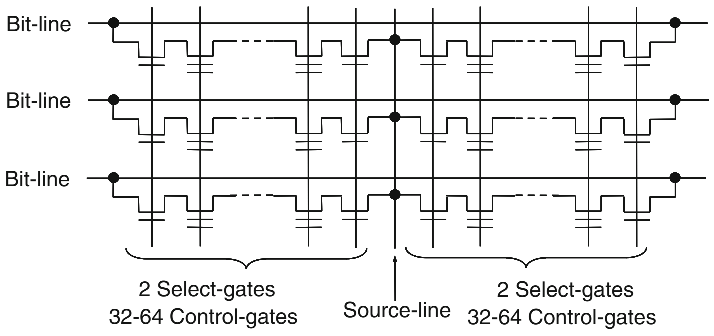


Fig. 7.4 Schematic diagram of a NAND Flash memory cell array

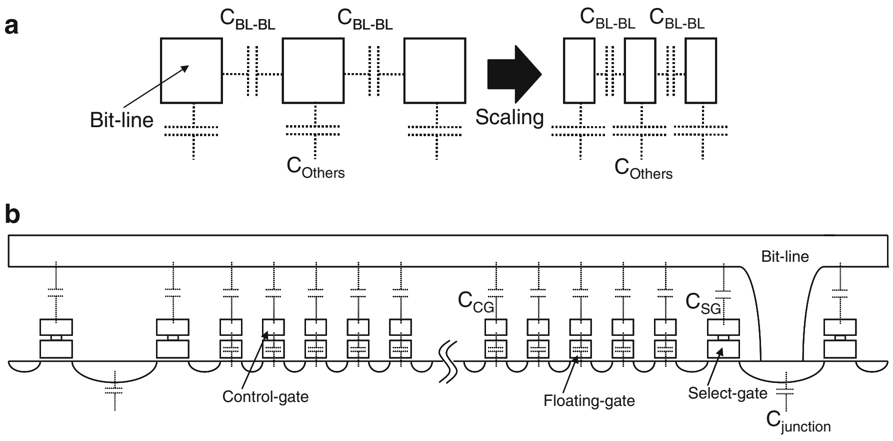


Fig. 7.5 Cross sectional view of (**a**) a bit-line and (**b**) a NAND Flash memory cell

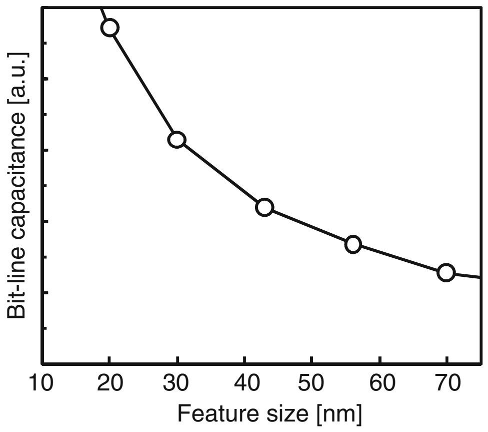


Fig. 7.6 Bit-line capacitance trend of a NAND Flash memory

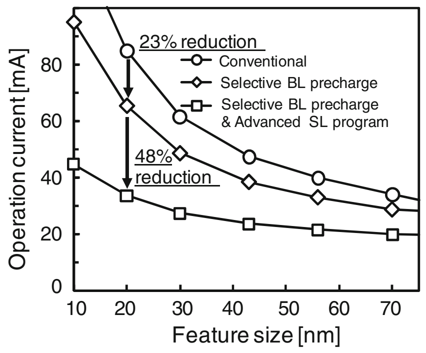


Fig. 7.7 Operation current trend of a NAND Flash memory

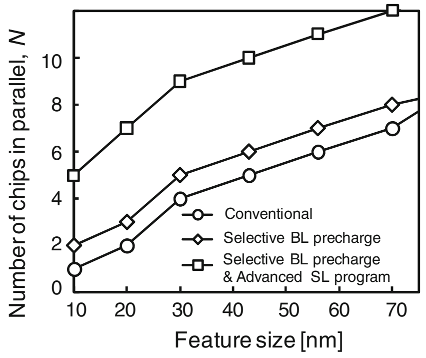


Fig. 7.8 Trend of the number of NAND chips operated in parallel, *N*

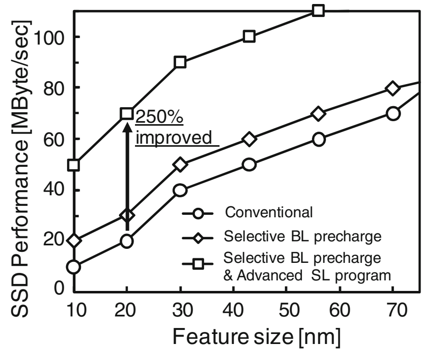


Fig. 7.9 System-level performance trend of SSD

Since the current consumption increases for sub-30 nm generation, the number of chips in parallel, *N* should be smaller as shown in Fig. 7.8 and the SSD speed drastically degrades as shown in Fig. 7.9. To overcome this power consumption problem, new circuit technologies are described in the following sections.

7.3 Selective Bit-Line Precharge Scheme

A bit-by-bit program algorithm of the NAND Flash memory [9] is shown in Fig. 7.10a.

First, the program data is input to the page buffers. Next, the program pulse is applied to the memory cells. Then, the verify-read to check if the memory cells are successfully programmed is performed. The re-program data in the page buffers are modified so that the re-program pulse is applied only to memory cells which are insufficiently programmed. The re-program pulse and the verify-read are repeated until all memory cells are successfully programmed.

In case of the multi-level cell, two bits in a memory cell are assigned to the 1st and 2nd pages and the two bits are programmed at different operations [10]. In the 1st page program, the memory cell is programmed from the erased state to the "A"-state shown in Fig. 7.10b. In the 2nd page program, the memory cell is programmed to the "B"- or "C"-state. The selective bit-line precharge scheme decreases the current consumption during the verify-read.

Figure 7.11 compares the conventional verify-read and the selective bit-line precharge scheme. In the conventional verify-read, all bit-lines are precharged irrespective of the program data and a lot of current flows. In the selective bit-line precharge scheme, bit-lines are selectively precharged based on the program data in the page buffer. By eliminating the unnecessary bit-line precharge, a low current operation is achieved.

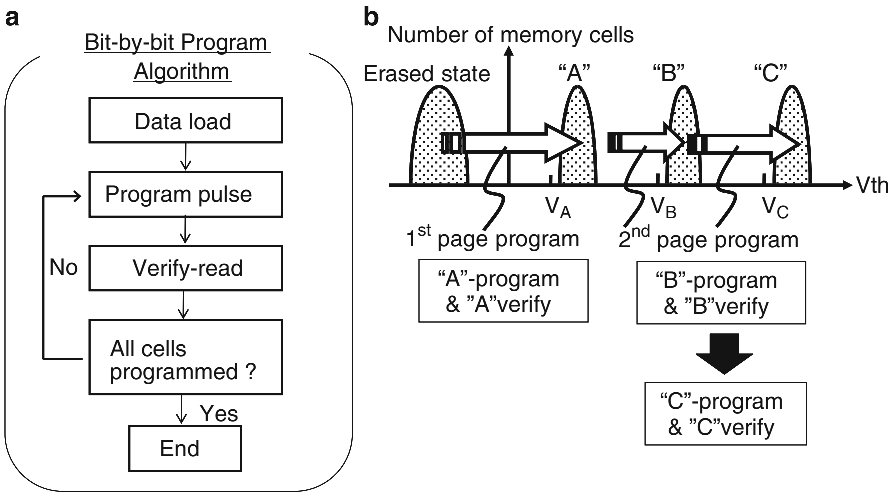


Fig. 7.10 (**a**) Program algorithm of a NAND Flash memory, (**b**) 1st page and 2nd page program of a multi-level cell

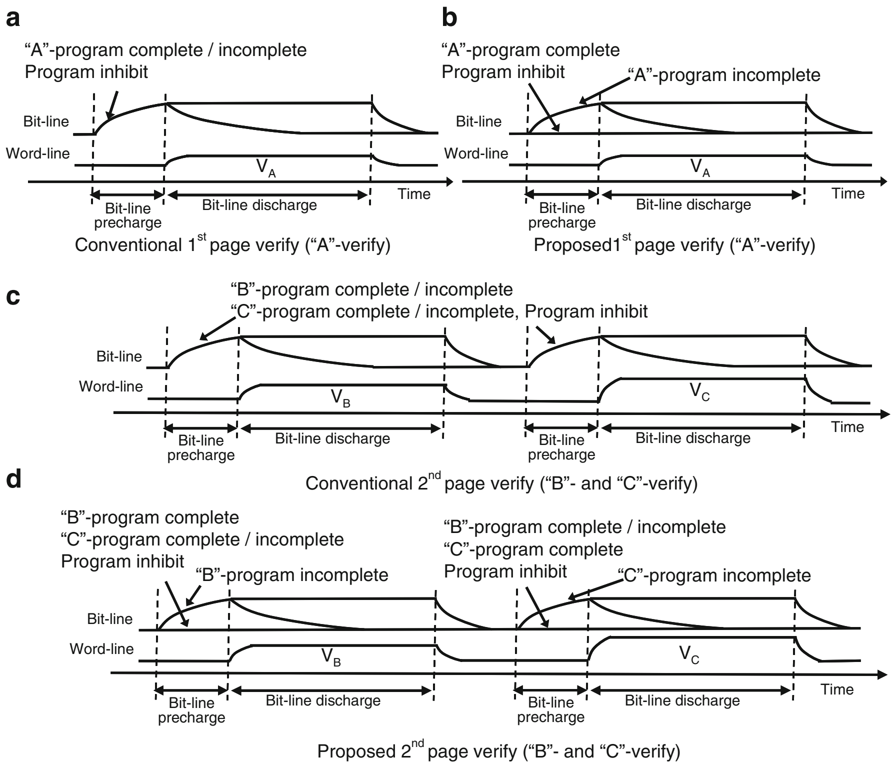


Fig. 7.11 Conventional and selective bit-line precharge scheme: (**a**) Conventional 1st page verify ("A"-verify); (**b**) Proposed 1st page verify ("A"-verify); (**c**) Conventional 2nd page verify ("B"- and "C"-verify); (**d**) Proposed 2nd page verify ("B"- and "C"-verify)

This novel operation is realized by adding five transistors shown in Fig. 7.12 to the conventional page buffer [11]. The die size penalty is less than 1% of the chip size.

In the 1st page verify-read, the data in the page buffer is the program inhibit, the "A"-program complete or the "A"-program incomplete. The program data stored in a page buffer for the "A"-program incomplete is updated so that the program pulse is applied only to the memory cells which are insufficiently "A"-programmed. In the conventional "A"-verify read, all bit-lines are precharged to 1 V as shown in Fig. 7.11a. On the other hand, in the proposed "A"-verify read, in case of the "A"-program complete and the program inhibit, the data in the page buffer does not change and thus the bit-line precharge can be removed. The bias condition of the proposed page buffer is summarized in Table 7.1. By activating PRE1- and PRE2-signals in Fig. 7.12, only bit-lines of the "A"-program incomplete are precharged to 1 V as shown in Fig. 7.11b. Then, the control gate is raised to V_A in Fig. 7.10b and the bit-line is discharged.

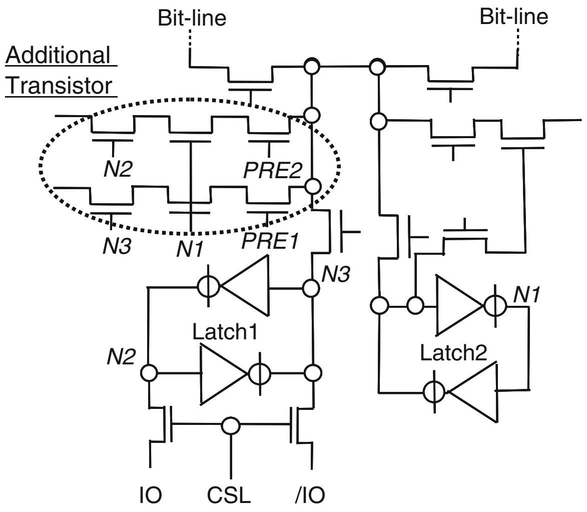


Fig. 7.12 Intelligent page buffer

Table 7.1 Bias condition of the selective bit-line precharge

	PRE1	PRE2
"A" – verify	"High"	"High"
"B" – verify	"High"	"Low"
"C" – verify	"Low"	"High"

In the 2nd page verify-read, "B"- and "C"-verify read are sequentially performed. In the conventional "B"-verify read, all bit-lines are precharged to 1 V, irrespective of the program data as shown in Fig. 7.11c. In the proposed "B"-verify read shown in Fig. 7.11d, the PRE1-signal is turned-on and only bit-lines of the "B"-program incomplete are precharged.

The bit-lines of the "C"-program, the "B"-program complete and the program inhibit are not precharged to save current. Similarly, in the proposed "C"-verify read shown in Fig. 7.11d, the PRE2-signal is activated and only bit-lines of the "C"-program incomplete are precharged.

By removing an unnecessary bit-line precharge, the current consumption decreases by 23% for sub-30 nm NAND as shown in Fig. 7.7.

7.4 Advanced Source-Line Program

The advanced source-line program reduces the current during the program pulse. Figure 7.13 compares the conventional program [12] and the source-line program [13]. In the conventional program shown in Fig. 7.13a, to realize a program inhibit operation a higher program inhibit voltage, 2.5 V, is applied from the high capacitance bit-line to the memory cell, consuming a lot of current. Contrarily, in

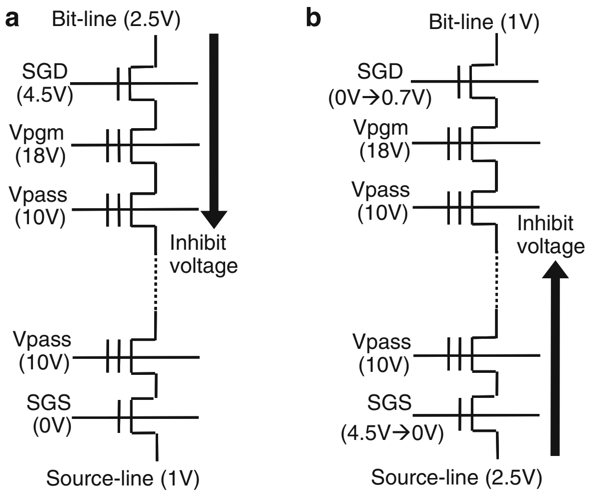


Fig. 7.13 (**a**) Conventional program [12], (**b**) Source-line program [13]

the source-line program shown in Fig. 7.13b, the current consumption is reduced by applying a higher voltage, 2.5 V, from the low capacitance source-line.

By using Fig. 7.5, the capacitance of the bit-line and the source-line is compared. The bit-line capacitance, $C_{Bit\text{-}line}$, and the source-line capacitance, $C_{Source\text{-}line}$, are expressed as follows:

$$C_{Bit-line} = C_{BL-BL} + C_{junction} \tag{7.2}$$

$$C_{Source-line} = C_{Source-line,\,wire} + C_{junction} \tag{7.3}$$

where $C_{BL\text{-}BL}$, $C_{Source\text{-}line,wire}$ and $C_{junction}$ are the inter bit-line wiring capacitance, the source-line wiring capacitance and the junction capacitance, respectively as shown in Fig. 7.5. The junction capacitance of the bit-line is the same as that of the source-line.

In the NAND Flash memory, the inter bit-line wiring capacitance, $C_{BL\text{-}BL}$, is much larger than the source-line wiring capacitance, $C_{Source\text{-}line,wire}$, or the junction capacitance, $C_{junction}$. As shown in Fig. 7.4, the bit-lines cover all memory cells. On the other hand, one source-line is arranged every 64 or 128 control gates and 4 select gates. Thus, the source-line wiring capacitance, $C_{Source\text{-}line,wire}$, is much smaller than the inter bit-line capacitance, $C_{BL\text{-}BL}$. Similarly, the junction capacitance, $C_{junction}$, is much smaller than the inter bit-line wiring capacitance, $C_{BL\text{-}BL}$, because one contact is also shared by 64 or 128 memory cells and 4 select transistors. As a result, the source-line capacitance, $C_{Source\text{-}line}$, is as small as one tenth of the bit-line capacitance, $C_{Bit\text{-}line}$.

The source-line program was demonstrated with a 0.25 μm NAND Flash memory [13]. Yet, in the sub-50 nm gigabit-capacity NAND Flash memory since the

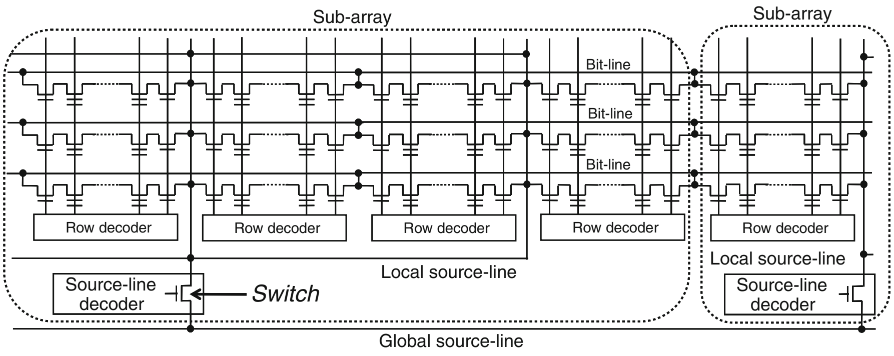


Fig. 7.14 Memory cell array architecture of the advanced source-line program

Table 7.2 Operation principle of the switch in the source-line decoder in the advanced source-line program

	Selected sub-array	Unselected sub-array
Read	ON	ON
Program	ON	OFF
Erase	ON	ON

number of source-lines increases, the total source-line capacitance in a chip exceeds 20 nF and the current consumption to charge a source-line capacitance becomes significant.

To solve this problem, the advanced source-line program is introduced [1]. In this scheme, the source-line is divided to reduce the load capacitance during the program pulse. As shown in Fig. 7.14, one memory cell array is divided to 16 sub-arrays. The source-line has a hierarchy structure where a local source-line in a sub-array connects to the global source-line through the source-line decoders.

Note that only the source-line is divided and that the active region, bit-lines, control gates and select gates are not divided. As the hierarchical source-line structure is realized by changing the layout of the metals and contacts in the memory cell array, there is no cell area overhead with this new architecture. The additional source-line decoders increase the area by less than 1% of the chip size.

During the program pulse, the source-line decoder of the selected sub-array is turned-on and the source-line decoders of the unselected sub-arrays are turned off as shown in Table 7.2.

As a result, only the local source-line of the selected sub-array is biased to 2.5 V and the remaining local source-lines of unselected sub-arrays are not charged. Since the load capacitance of the source-line is reduced by 90%, the total current decreases by 48% as shown in Fig. 7.7.

On the other hand, during the read and the verify-read operations, all source-line decoders are turned on and thus all local source-lines are connected to the global source-line so that the source-line resistance is minimized. Consequently, the circuit noise caused by the source-line bounce [14] is suppressed.

By using the selective bit-line precharge scheme and the advanced source-line program, the current consumption reduces by 60% for sub-30 nm generation SSD. Assuming the same current budget of 250 mA, 2.5 times as many NAND chips operate simultaneously as shown in Fig. 7.8 and the SSD speed improves by 150% as shown in Fig. 7.9.

7.5 Intelligent Interleaving

Figure 7.15 illustrates the current waveform of the NAND Flash memory. A current peak appears during the bit-line precharge and the charge pump ramp-up [8]. In the interleaving operation, if the current peak of two or more NAND chips occurs at the same time, huge current flows in SSD and the power supply drops by more than 0.3 V. To avoid this power supply noise and realize a both highly-reliable and high-speed program, an intelligent interleaving is introduced.

In this scheme, the PD (Power Detect)-signal is added. The PD-signal is connected with wired-or configuration among NAND Flash memories and the NAND controller as shown in Fig. 7.16. In Fig. 7.16, the NAND chips can belong to the same channel or the different channels in Fig. 7.2. If one of the NAND chips starts a bit-line precharge or a charge pump ramp-up that causes a current peak, the NAND chip pulls down the PD-signal. When PD is low, the NAND controller does not issue a write command to avoid the power supply noise.

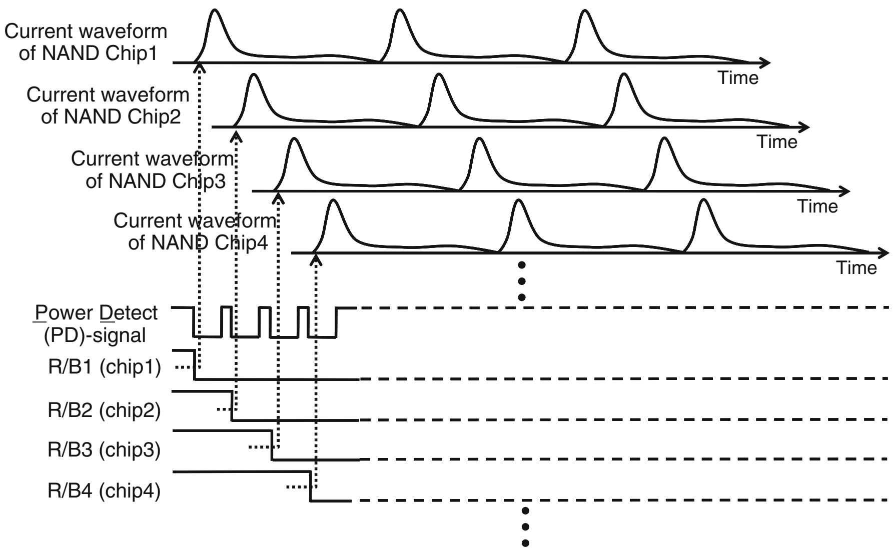

Current waveform of NAND Chip1
Time
Current waveform of NAND Chip2
Time
Current waveform of NAND Chip3
Time
Current waveform of NAND Chip4
Time
Power Detect (PD)-signal
R/B1 (chip1)
R/B2 (chip2)
R/B3 (chip3)
R/B4 (chip4)

Fig. 7.15 Current waveform and operation principle of the intelligent interleaving

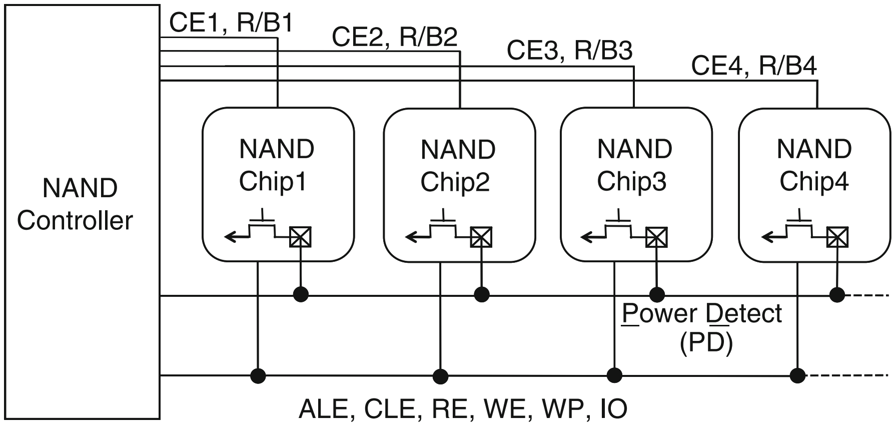


Fig. 7.16 Block diagram of the NAND Flash memory and the NAND controller with the intelligent interleaving

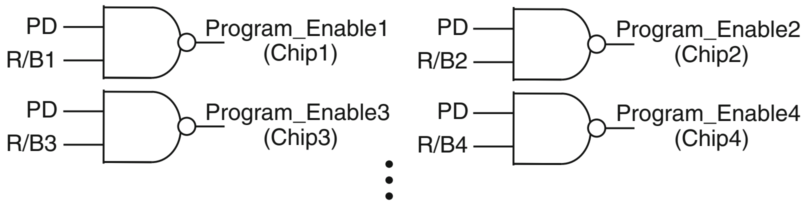


Fig. 7.17 Intelligent interleaving control circuit in the NAND controller

To monitor the status of each NAND chip, the R/B (Ready/Busy)-signal is connected between the NAND controller and each NAND Flash memory chip. The R/B-signal is low if the NAND chip operates a read, program or erase and therefore is in the busy state. When both the PD- and the R/B-signals are high, Program_Enable-signal shown in Fig. 7.17 in the controller becomes low. In that case, there is no current peak and the NAND chip is ready. Then, the NAND controller issues a write command to the NAND chip and the program starts.

By using the intelligent interleaving, multiple NAND chips are programmed at the same time without causing a power supply noise as shown in Fig. 7.15. Therefore, a highly reliable and high speed operation of SSD is achieved.

7.6 Sector Size Optimization

To further improve the write performance of the NAND Flash memory, it is essential to optimize the sector size. The sector size is the minimum file unit for the computer system. The data is transferred from the OS (*Operating System*), to the memory

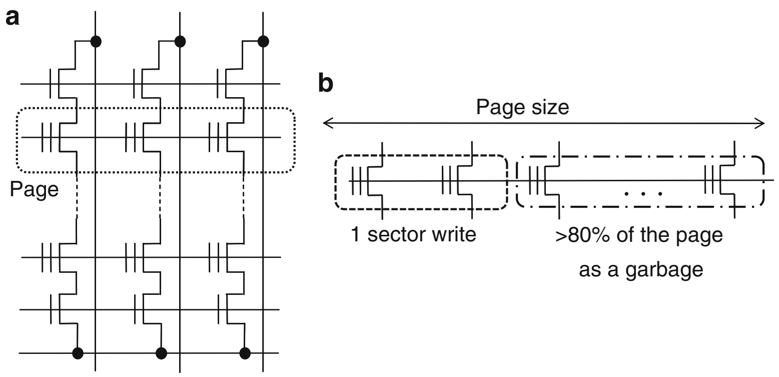


Fig. 7.18 (**a**) Page of a NAND Flash memory; (**b**) One sector write operation of a NAND flash memory

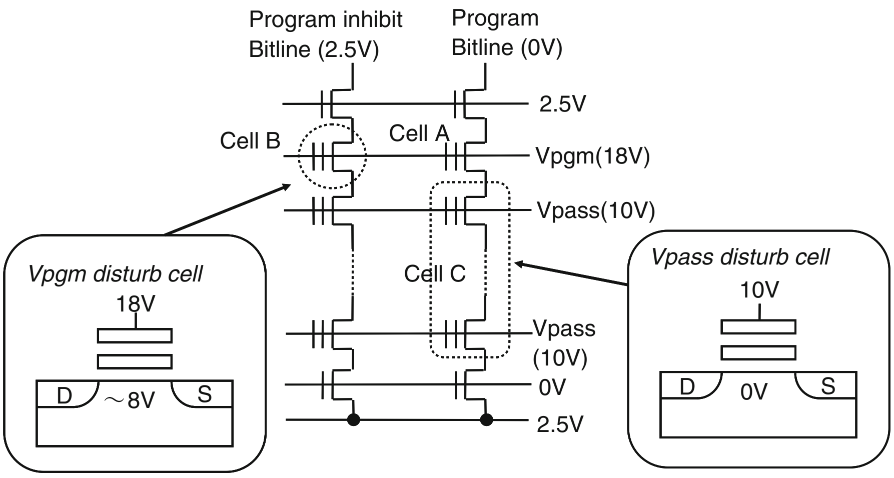


Fig. 7.19 Program disturbance of a NAND Flash memory

storage such as SSD and HDD with the unit of the sector. With current operating systems such as Windows, the sector size is optimized for the magnetic drives and is typically 512 Bytes. The 512 Bytes sector size is much smaller than the page size of the NAND Flash memory, typically 4–8 KBytes. The discrepancy between the sector size and the page size significantly degrades the performance of SSD.

In a NAND Flash memory, all memory cells connected to the same control-gate belong to the same page and are programmed at the same time as shown in Fig. 7.18a.

A page can be written only once to avoid a program disturbance shown in Fig. 7.19. In Fig. 7.19, Cell A is programmed. Cell B and Cell C are unselected and are under a weak program bias condition [12], which is called the program

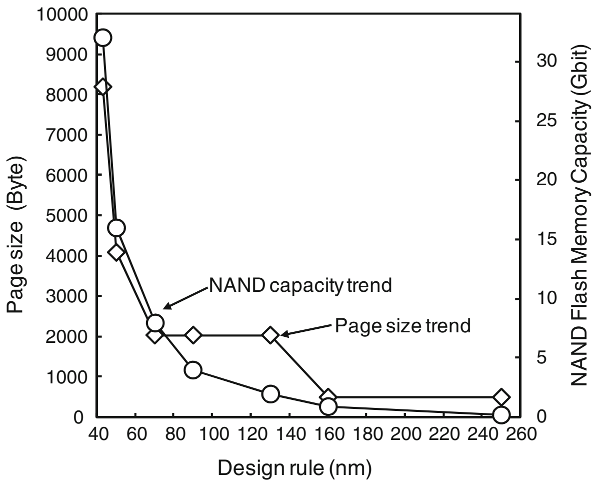


Fig. 7.20 Page size and memory capacity trend of a NAND Flash memory

disturbance. If a page is programmed more than once, the stress time of the program disturbance becomes longer and the unselected Cell B or Cell C are programmed, causing a data failure.

Considering that a page can be written only once, if one sector write is performed, only 512 Bytes of the page is programmed and the remaining more than 80% of the page are wasted as a garbage as shown in Fig. 7.18b. Since the NAND Flash memory in SSD is seriously fragmented, in order to efficiently use SSD, the garbage collection frequently happens. During the garbage collection, the block copy operation of the NAND Flash memory is performed [8]. The block copy is composed of the cell-read, cell-program, data transfer between the NAND Flash memory and the NAND controller, and the ECC calculation. As the block copy takes as long as 100 ms, if the block copy happens frequently, the system-level write performance degrades drastically [1].

To avoid a block copy operation and maximize the system-level write performance, it is crucial to minimize the data fragmentation by optimizing the sector size. The sector size should be the same as or multiples of the page size so that the garbage would not happen in case of the one sector write operation. Figure 7.20 shows the page size trend of the NAND Flash memory. As the memory capacity increases, the number of memory cells connected to the same control gate increases and as a result the page size also increases. Furthermore, in case of the interleaving operation, the effective page size is increased to the number of chips in parallel, *N* times the page size of one NAND Flash memory. Therefore, the optimal sector size should be 128–256 KBytes.

The most straightforward method to increase the sector size is to change the file system of the OS [1]. However, the software size of the OS is very large and it needs huge engineering efforts to change the OS. The OS is usually changed every about 5 years and it is almost impossible to change the OS in accordance with the frequent page size change of the NAND Flash memory. Thus, optimizing the sector size by efficiently using a write buffer in the NAND controller is essential.

In case of the one sector write operation, one sector program data is transferred from the OS to the NAND controller. The NAND controller temporarily stores the program data in the write buffer and the data is not programmed to the NAND Flash memory. Then, multiple program data are transferred from the OS to the NAND controller and are accumulated in the write buffer. When the program data size in the write buffer of the NAND controller becomes comparable with the page size, the write operation to the NAND Flash memory is performed. By efficiently using the write buffer in the NAND controller and optimizing the sector size, the fragmentation of SSD is eliminated and the SSD performance is maximized.

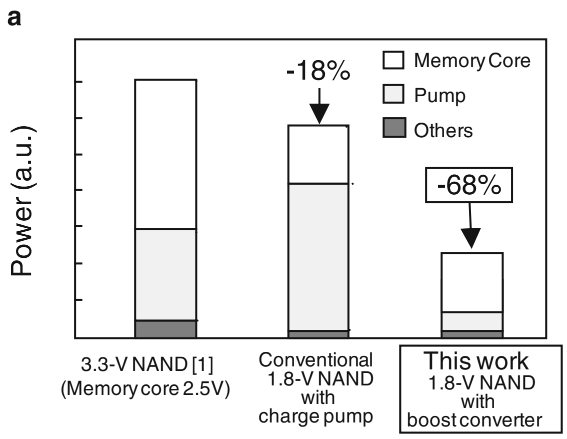
7.7 Adaptive Program-Voltage Generator for 3D-SSD

As mentioned in the previous sections, decreasing power consumption is the key design issue of SSDs. The best strategy to decrease the power is lowering the supply voltage, V_{DD} from 3.3 V to 1.8 V. Yet, at 1.8 V V_{DD} the power consumption of the conventional charge pumps to generate the program voltage, V_{PGM} (20 V), drastically increases and the total power consumption of the NAND only slightly decreases as shown in Fig. 7.21a. What's worse, the area of the charge pump more than doubles, which increases the NAND chip area by 5–10%. To overcome this dilemma, a low power program voltage generator (PVG) using a boost converter with an adaptive frequency and duty cycle (AFD) controller is proposed [4].

The 3D-integrated SSD proposed in [4] is shown in Fig. 7.21b. NAND chips, DRAM, a NAND controller and the proposed PVG are integrated with SiP, System in Package. The PVG consists of an inductor in an interposer, the high voltage MOS (HVMOS) and the AFD controller. In the system, the cost is also minimized. An inductor is available with no area penalty by using wiring in the interposer connecting the NAND controller, NAND Flash memories and DRAMs. The die size of the NAND decreases by 5–10% because no charge pump is needed. The HVMOS is fabricated with a matured NAND process and its area is just 15% of the conventional charge pump. Since the die size of the AFD controller is only 0.188 mm^2 with a 0.18 μm CMOS process, it can be integrated in a NAND controller with a negligible area increase.

A PVG using a boost converter was reported for a NOR Flash memory [15]. A comparison between the PVG for a NOR Flash memory and for a NAND Flash memory is summarized in Fig. 7.22a. In a NOR Flash memory, the load of the PVG is resistive.

The PVG continuously supplies load current of 20 mA at an output voltage, 5 V. In such a resistive load and a low output voltage condition, a conventional PWM is employed [15]. Conversely, in a NAND Flash memory, the load is capacitive and V_{PGM} is 20 V. During the program, V_{PGM} is applied to the word-line and a DC load current of 20 μA flows. Also, a PVG for a NAND Flash memory should operate



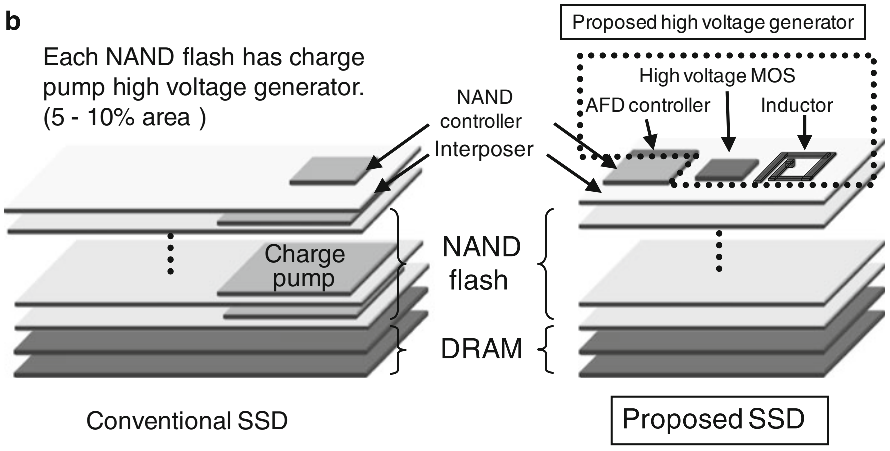


Fig. 7.21 (**a**) Comparison of the power consumption of NAND Flash memories; (**b**) Structure of the 3D-integrated SSD

on-and-off to save power. In this condition both switching frequency and duty cycle must be dynamically optimized and the conventional PWM changing only the duty cycle cannot be used.

To identify the most power efficient frequency and duty cycle, an input supply current, I_{DD} is measured with the PVG. As shown in Fig. 7.22b, each V_{PGM} has different optimal frequency and duty cycle minimizing I_{DD}. In other words, the power efficiency is a function of V_{PGM}, a switching frequency and a duty cycle since the PVG operates in a discontinuous mode with a capacitive load. With a bit-by-bit program verifying scheme, in each program cycle, V_{PGM} is incremented by 0.5 V from 15 V to 25 V [16, 17].

For each V_{PGM}, the AFD controller adaptively manages the switching frequency and the duty cycle simultaneously so that the energy loss is minimized.

a

	PWM [4]	This work
Load	Resistive	Capacitive
V_{PGM}	5V	20V
I_{LOAD}	20mA	20μA
Operation	Continuously on	On-and-off
Control	Duty cycle only	Frequency and duty cycle

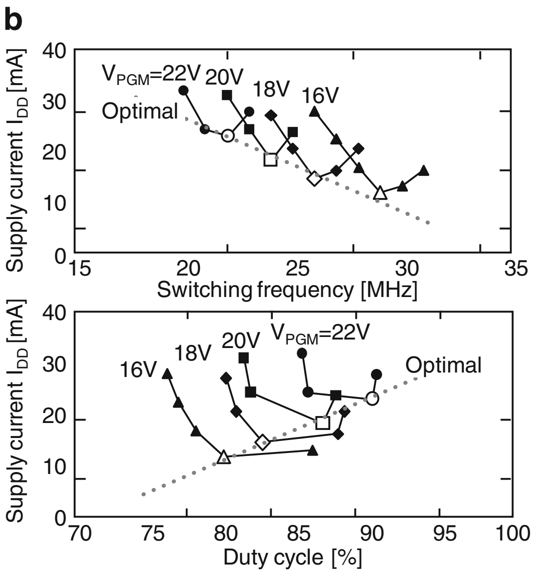


Fig. 7.22 (**a**) Comparison of the conventional boost converter [15] and adaptive program voltage generator [4]; (**b**) Measured optimal switching frequencies and duty cycles for various V_{PGM}

Figure 7.23 shows the schematic diagram of the PVG with the AFD controller. V_{PGM} is monitored with three comparators and the control logic selects the proper switching frequency and duty cycle from the register sets, Reg.$_L$, Reg.$_M$, and Reg.$_H$. These registers store a table of the frequency and the duty cycle which minimize both the power and the output voltage fluctuation. The digital controlled oscillator (DCO) is depicted in Fig. 7.24. The DCO consists of current reference circuits and a couple of capacitor arrays. The DCO enables the clock shape to be only determined by the resistor and the capacitor since the reference current I_{REF} is copied to the node V_{CAPA} and V_{CAPB} with the current mirror [18]. The frequency and the duty cycle are robust against V_{DD} fluctuation, transistor global V_{TH} variation and temperature variation. Switching time of DCO, T_{ON} and T_{OFF} are expressed as $R{\cdot}C_{A1\text{-}An}$ and $R{\cdot}C_{B1\text{-}Bn}$, respectively. Since $C_{A1\text{-}An}$ and $C_{B1\text{-}Bn}$ are independently selected according to the data in the registers, T_{ON} and T_{OFF}, that is, the switching frequency and the duty cycle are independently controlled.

To suppress the V_{PGM} fluctuation, the AFD controller dynamically changes the frequency and the duty cycle in three steps as shown in Fig. 7.25.

In the first step, the power efficient lower frequency is chosen. The AFD controller outputs pulses with the switching frequency and duty cycle set, f_L/D_L determined by Reg.$_L$. V_{PGM} rises coarsely and rapidly until V_{PGM} reaches V_{REFL}. With f_L/D_L, the voltage increment for each pulse is 5 V. The frequency becomes higher in the second and the third steps. In the second step, the AFD controller changes the switching pulse from f_L/D_L to f_M/D_M determined by Reg.$_M$.

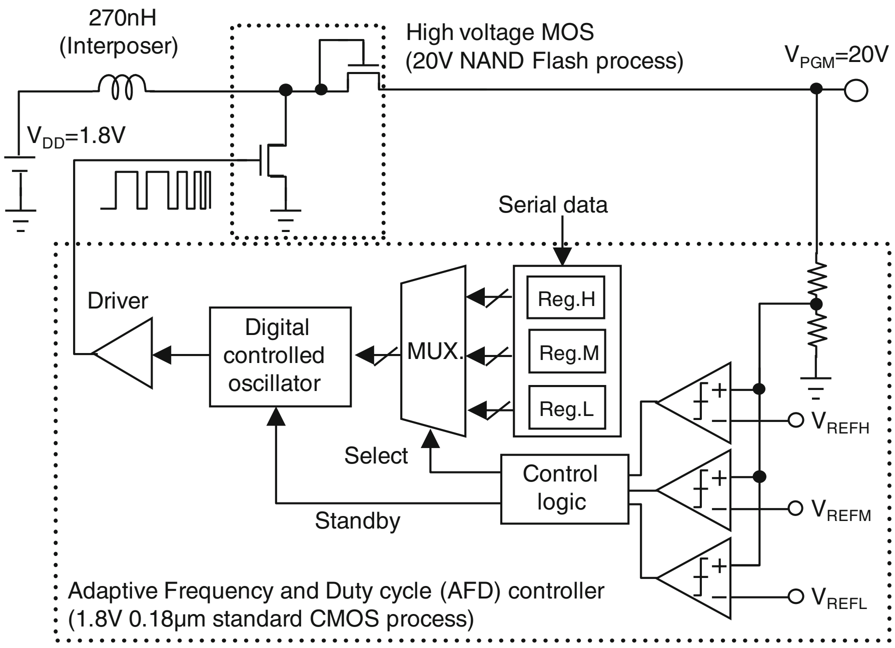
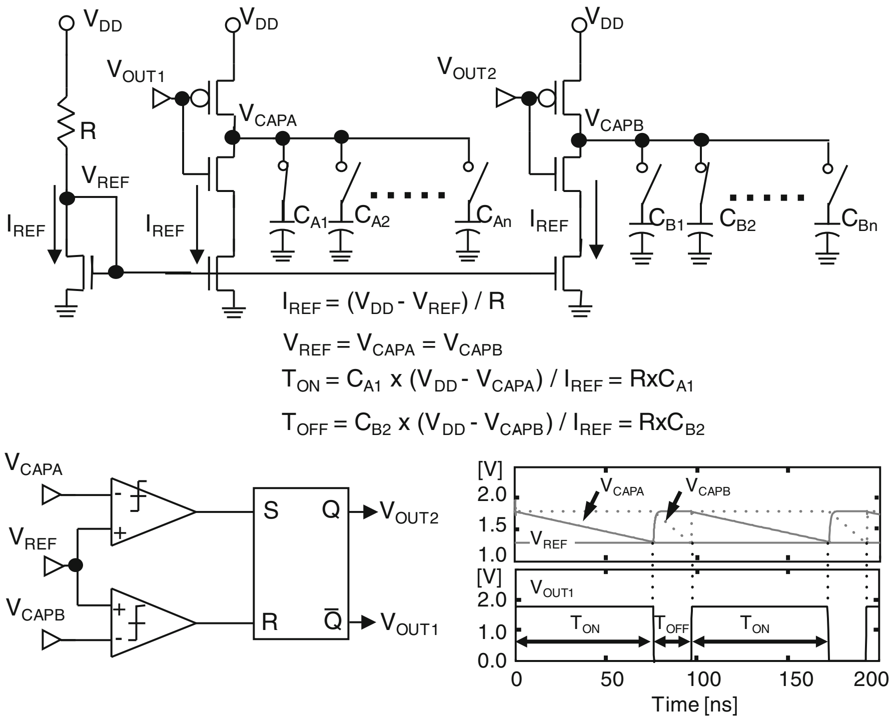


Fig. 7.23 Circuit diagram of the adaptive program voltage generator

Finally, the AFD controller finely raises V_{PGM} with f_H/D_H toward the target voltage. When V_{PGM} reaches the target voltage, the AFD controller stops switching pulses to save power. As a result, the PVG raises V_{PGM} more than three times faster than the conventional charge pump with a minimum power. V_{PGM} is precisely controlled with less than 0.3 V fluctuation, which enables a tight memory cell V_{TH} distribution.

Figure 7.26 shows the microphotograph of the bread board model of SSD consisting of the HVMOS chip (0.35 mm × 0.50 mm), the AFD controller chip (0.67 mm × 0.28 mm), a 270nH, 0.5 Ω inductor in an interposer (5 mm × 5 mm), and a 56 nm 16 Gb NAND Flash memory chip. Key features are summarized in Fig. 7.27. The measured waveforms during the program of a 56 nm 16 Gb NAND Flash memory with the PVG is shown in Fig. 7.27. When a write command inputs to the NAND, the Ready/Busy-signal turns to low and the NAND goes into the busy state. The program voltage is supplied from the PVG and the program pulse is applied to the memory cells. Then, the verify-read detects that all memory cells are successfully programmed and the Ready/Busy-signal returns to high.

The measured power consumption of the PVG is 30 nJ, which is only 12% of the conventional charge pump. Measured rising time of the PVG is 0.92 μs (at V_{DD}, 1.8 V and V_{PGM}, 15 V), while that of the charge pump is 3.45 μs. As the rising time of the V_{PGM} decreases by 2.53 μs, the program pulse width can be shortened by 2.53 μs. As a result, the total program time of a NAND Flash memory, a sum of the program pulse width and the verify-read time is 7.8% shorter than the conventional



V_{DD}
V_{OUT1}
V_{OUT2}
R
V_{REF}
I_{REF}
V_{CAPA}
V_{CAPB}
C_{A1} C_{A2} C_{An}
C_{B1} C_{B2} C_{Bn}
$I_{REF} = (V_{DD} - V_{REF}) / R$
$V_{REF} = V_{CAPA} = V_{CAPB}$
$T_{ON} = C_{A1} \times (V_{DD} - V_{CAPA}) / I_{REF} = R \times C_{A1}$
$T_{OFF} = C_{B2} \times (V_{DD} - V_{CAPB}) / I_{REF} = R \times C_{B2}$
S Q R $\bar{Q}$
[V]
T_{ON} T_{OFF}
Time [ns]

Fig. 7.24 Digital controlled oscillator (DCO)

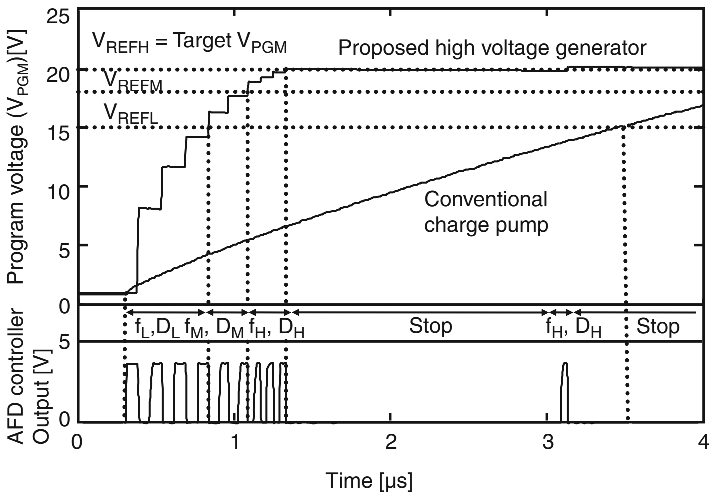


Fig. 7.25 Simulation result of the adaptive program voltage generator

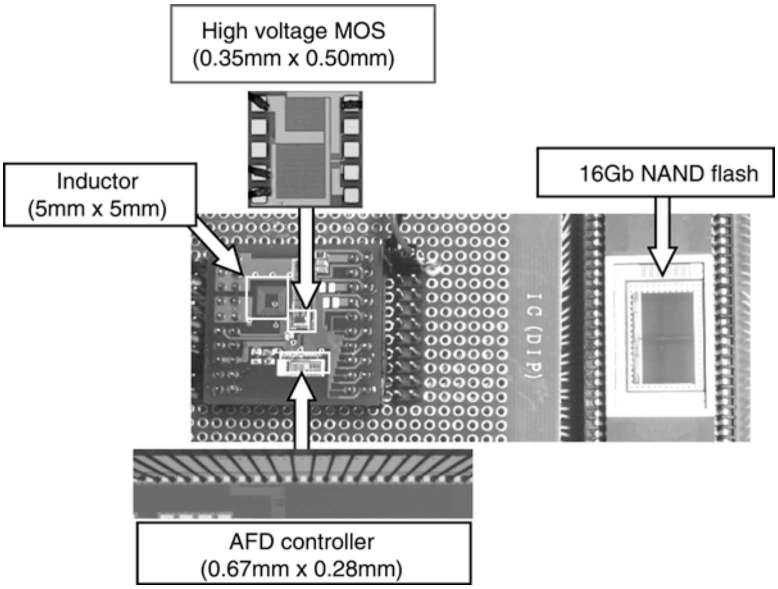


Fig. 7.26 Die microphotograph of 3D-integrated SSD

1.8 V NAND Flash memory. The area of the HVMOS chip is just 15% of the charge pump without a control circuit or an oscillator. By decreasing V_{DD} from 3.3 V to 1.8 V, the total power consumption of a NAND Flash memory decreases by 68% as shown in Fig. 7.21a.

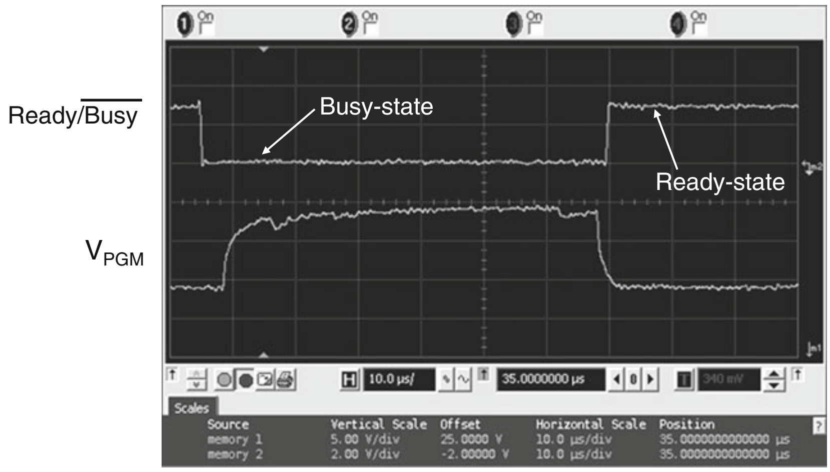
7.8 Asymmetric Coding

With the technology shrink, also the NAND raw BER requires special attention. This section deals with the asymmetric coding which decreases memory errors by 95% [19].

Figure 7.28 shows the measured memory cell error in the data retention and program disturb of 4X, 3X and 2X nm NAND Flash memories.

As the cell size decreases, both data retention and program disturb errors increase due to the interference, random telegraph noise and reduced electrons [20]. In the scaled NAND, the electric field in the channel increases [21] and the program disturb due to GIDL-induced hot electron injection becomes more significant (Fig. 7.28c). In conventional SSDs, 20–40 bit correction per 1 KByte codeword *Error Correcting Code* (ECC) is used to correct errors [22].

As a stronger ECC such as LDPC is developed [23], the capability of ECC is close to the Shannon limit of a few percent error correction. Thus, the additional



```
Source      Vertical Scale  Offset      Horizontal Scale  Position
memory 1    5.00 V/div      25.0000 V   10.0 µs/div       35.0000000000000 µs
memory 2    2.00 V/div      -2.00000 V  10.0 µs/div       35.0000000000000 µs
```

	This work (Measured)	Conventional Charge Pump (Simulated)
Power consumption	30nJ (12%)	253nJ (100%)
Chip Area (high voltage MOS)	0.175mm^2 (15%)	1.19mm^2 (100%)
Chip Area (AFD controller)	0.188mm^2	-----
Rising time	0.92µs (27%)	3.45µs (100%)
Technology (high voltage MOS)	20V NAND flash process	--------
Technology (AFD controller)	1.8V 0.18µm standard CMOS	--------
Supply voltage	1.8V	1.8V

Fig. 7.27 Experimental results and key features of 3D-integrated SSD with the adaptive program voltage generator

high-reliability scheme is required. To improve the reliability in SSDs, Asymmetric Coding is described to improve the memory cell reliability by 95% without access time penalty.

Figure 7.29 shows the measured error analysis. The key observation is that for both data retention and program disturb errors, the number of "0→1"-errors and "1→0"-errors are NOT equal.

The origin of the data retention error is the electron ejection from the floating gate due to the stress-induced leakage current (SILC). During the data retention, the memory cell V_{TH} moves to V_{THi}, the thermally equilibrium V_{TH}, where no electron exists in the floating gate. V_{THi} is around 0 V. Most errors correspond to the V_{TH} decrease of "10→00" or "00→01" due to the higher electric field across the tunnel oxide (Fig. 7.29a). Also, if the memory cell V_{TH} is higher, the program voltage is also higher.

The enhanced voltage stress to memory cells increases the trap density in the tunnel oxide. As a result, SILC increases and the data retention errors also increase.

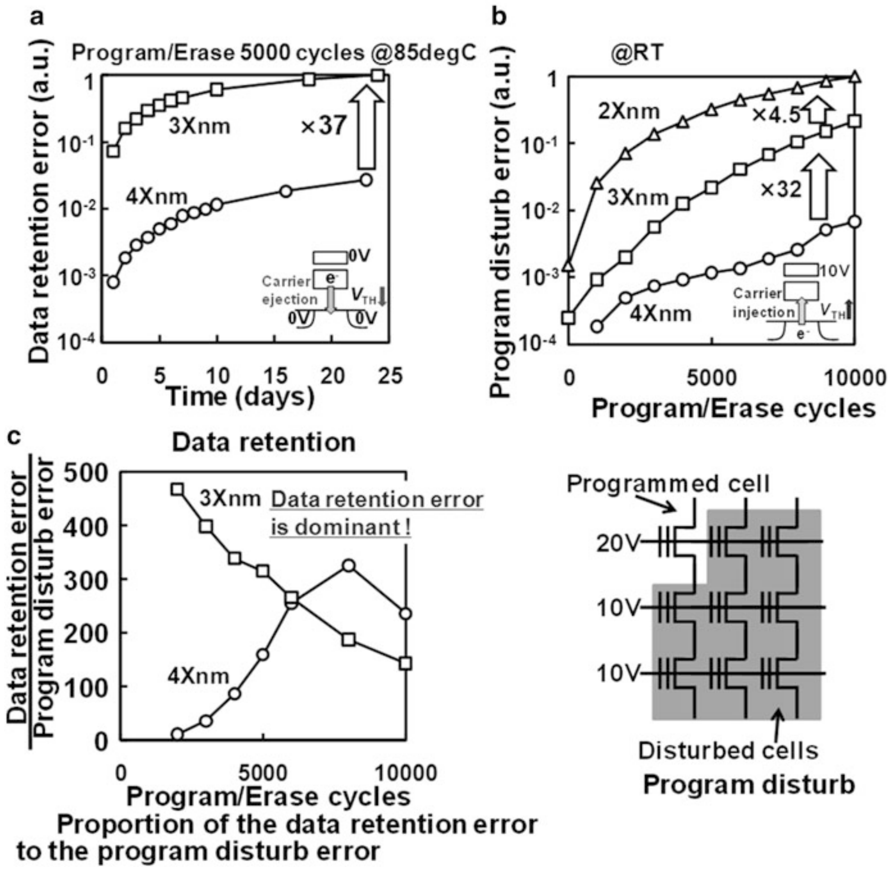


Fig. 7.28 Measured memory cell error rate in SSDs during (**a**) the data retention and (**b**) the program disturb of 4X, 3X and 2X nm NAND Flash memories. The memory cell scaling degrades the cell reliability. (**c**) The data retention errors are 100 times more than the program disturb errors

The program disturb error is caused by the electron injection due to GIDL to the floating gate. Thus, the program disturb error corresponds to the V_{TH} increase of "01→00" or "00→10" (Fig. 7.29b). In the multi-level cell NAND Flash memory, two bits stored in a memory cell are assigned to two different page (row) addresses, upper and lower pages [24]. For the data retention, the major error is "0→1" of the lower page and "1→0" of the upper page. In contrast, the major error of the program disturb is "1→0" of the lower page and "0→1" of the upper page.

Figure 7.30 shows the Randomizing Coding and Asymmetric Coding. Considering the data retention error is over 100-times more than the program disturb error (Fig. 7.28c), the Asymmetric Coding improves the data retention by increasing "1"- and "0"-data of the lower and upper pages, respectively. In the Randomizing Coding, the population of "10" and "00" is about 25% of the total data. In the SSD,

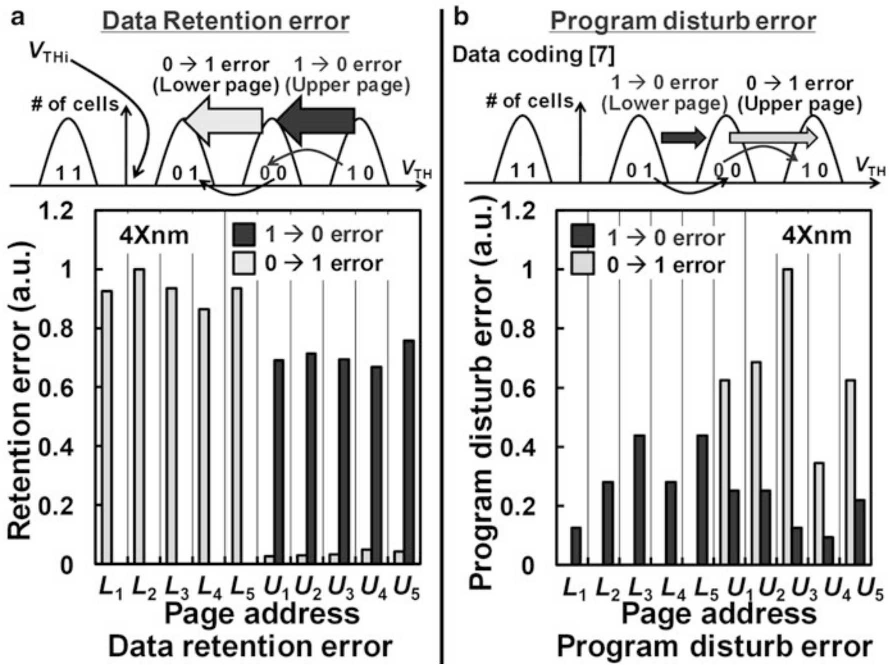


Fig. 7.29 Measured asymmetric memory errors. The observed errors strongly depend on the data pattern. The data retention error is "0→1" of the lower page and "1→0" of the upper page. The program disturb error is "1→0" of the lower page and "0→1" of the upper page

Asymmetric Coding Encoder modifies the data programmed to NAND so that at least 60% of the lower and upper pages are "1" and "0", respectively. As a result, "10" and "00" occupy about 16% and 24% of the total data. By decreasing the population of "10" and "00", the data retention error decreases because (1) the lower program voltage decreases the voltage stress to memory cells and (2) the lower memory cell V_{TH} decreases the electric field across the tunnel oxide and reduces SILC.

The code length, data unit where Asymmetric Coding is applied, is 16. If Data unit1 in Fig. 7.30 contains more than or equal to 8 bit of "0", Data unit1 is flipped to increase the number of "1" and the flag is set to "1". On the other hand, if Data unit2 contains less than 8 bit of "0", the data is not modified. As shown in Fig. 7.31a, the smaller code length realizes a higher population of "1" with a drawback of a larger overhead due to the flag. If the code length is 16, the overhead is 6.3%. The number of "1" of the lower page and that of "0" of the upper page becomes 60% of the total data. Figure 7.31b shows the measured data retention error vs. the program voltage stress. In Asymmetric Coding, the population of high V_{TH} states, "10" and "00", is lower and the program voltage stress to memory cells is reduced. As a result, the data retention error decreases by 91%. Figure 7.31c shows the measured

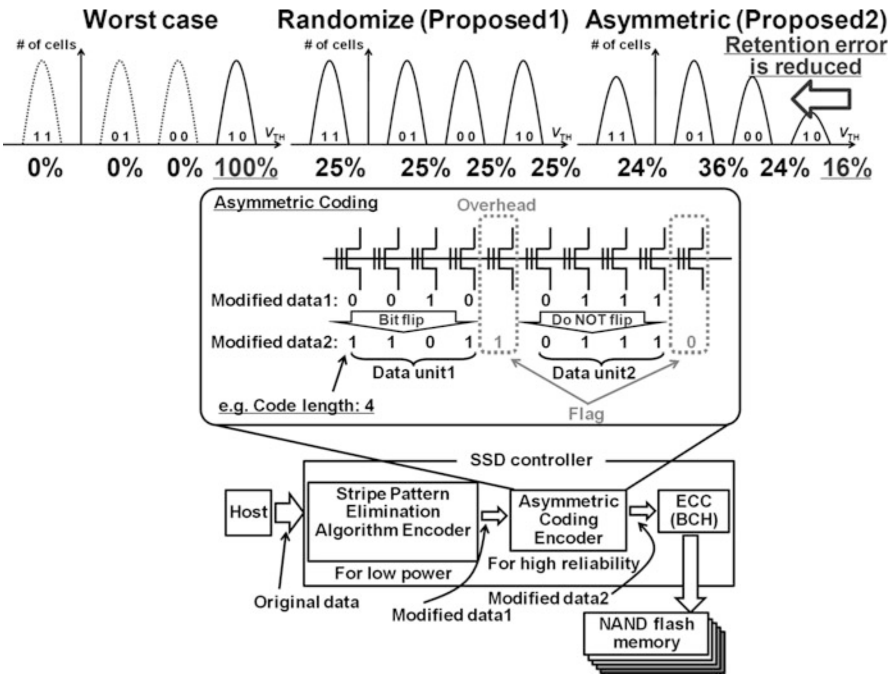


Fig. 7.30 Asymmetric Coding. The program data of NAND is modified to decrease the population of "10" and "00"

errors vs. the data pattern during the data retention. Again, as the asymmetric data pattern decreases the population of the high V_{TH} states, the data retention improves by 40%. Due to two effects mentioned above, the total retention error decreases by 95% (Fig. 7.31d).

7.9 Stripe Pattern Elimination Algorithm

In Sect. 7.2 we have explained how the power consumption is a function of the bit-line capacitance of NAND [2]. Besides the circuit technologies presented in previous sections, *Stripe Pattern Elimination Algorithm* (SPEA) can be used to eliminate the worst program data pattern, thus reducing the overall power consumption during programming [19].

Figure 7.32 shows the Stripe Pattern Elimination Algorithm. During the program of NAND, the selected and unselected bit-lines are biased to V_{CC} and V_{SS}, respectively. When bit-lines are alternately biased to V_{CC} and V_{SS}, which corresponds to the column-stripe data pattern "1010 . . . ", all inter bit-line capacitance is charged and a large current flows. On the other hand, when every bit-line is biased to V_{CC}

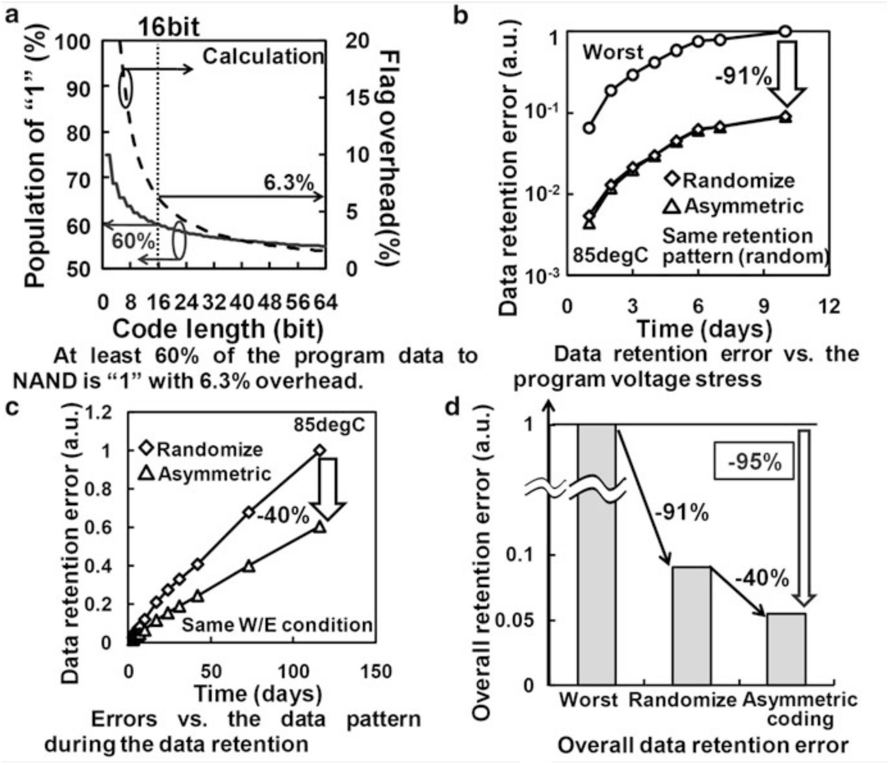


Fig. 7.31 Measured reliability improvement of Asymmetric Coding. (**a**) At least 60% of the program data to NAND is "1" with 6.3% overhead. (**b**) Measured data retention error vs. the program voltage stress. The reduced program voltage stress in Asymmetric Coding decreases the data retention error by 91%. (**c**) Measured errors vs. the data pattern during the data retention. Decreased population of the higher V_{TH} states, "10" and "00", in Asymmetric Coding reduces the data retention errors by 40%. (**d**) Measured overall data retention error

with all "1" data pattern, inter bit-line capacitance diminishes and the program current decreases. SPEA Encoder in Fig. 7.30 modifies the original data from the host to avoid the worst case column-stripe pattern and save the power. SPEA calculates the number of "1" in even and odd columns. Then, SPEA calculates the difference.

If the difference is larger than a threshold value, N_{TH}, the data programmed to NAND is modified so that the odd column data are arranged first and the even column data are arranged next (Fig. 7.32). As a result, the column-stripe pattern which consumes the maximum power is changed to decrease the power. In the SSD (Fig. 7.30), data from the host is first modified by SPEA Encoder to save the power, then modified again with the Asymmetric Coding Encoder to improve the reliability and finally programmed to NAND.

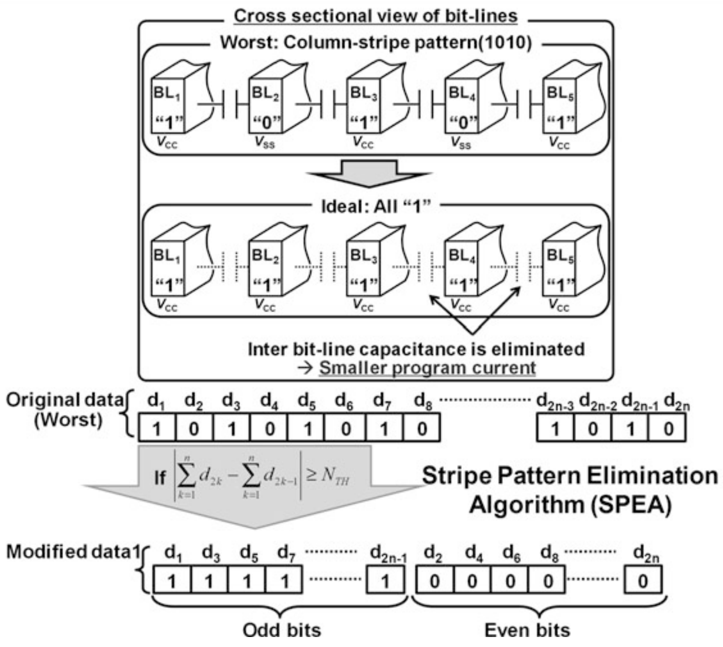

Cross sectional view of bit-lines
Worst: Column-stripe pattern(1010)
BL1 "1" Vcc
BL2 "0" Vss
BL3 "1" Vcc
BL4 "0" Vss
BL5 "1" Vcc
Ideal: All "1"
Inter bit-line capacitance is eliminated
→ Smaller program current
Original data (Worst)
d1 d2 d3 d4 d5 d6 d7 d8 ... d2n-3 d2n-2 d2n-1 d2n
1 0 1 0 1 0 1 0 ... 1 0 1 0
If $\left|\sum_{k=1}^{n} d_{2k} - \sum_{k=1}^{n} d_{2k-1}\right| \geq N_{TH}$
Stripe Pattern Elimination Algorithm (SPEA)
Modified data1
d1 d3 d5 d7 ... d2n-1 d2 d4 d6 d8 ... d2n
1 1 1 1 ... 1 0 0 0 0 ... 0
Odd bits
Even bits

Fig. 7.32 Stripe Pattern Elimination Algorithm (SPEA)

Figure 7.33 shows the measured power consumption. With SPEA, the program current of 4X nm and 3X nm NAND decreases by 35% and 43%, respectively. SPEA is more effective in the scaled NAND because as the memory cell size decreases, the bit-line capacitance as well as the power increase.

7.10 Conclusions

In this chapter, circuit technologies for co-designing NAND and the system controller are described. As a result, a highly reliable and high speed operation of SSD is achieved.

Two low power circuit technologies, the selective bit-line precharge scheme and the advanced source-line program, are discussed. By eliminating the unnecessary bit-line precharge during the verify-read and reducing the load capacitance during the program pulse, the operative current of sub-30 nm NAND Flash memories is reduced by 60%. Moreover, a low noise circuit technology is discussed: by using the intelligent interleaving, multiple NAND chips are simultaneously programmed with a reduced power supply noise. With these new circuit technologies, the performance of sub-30 nm generation SSD improves by 150% without a cost penalty. Furthermore, an intelligent 3D-integration of SSD is described. A PVG using a single-stage boost converter for a NAND Flash memory is introduced.

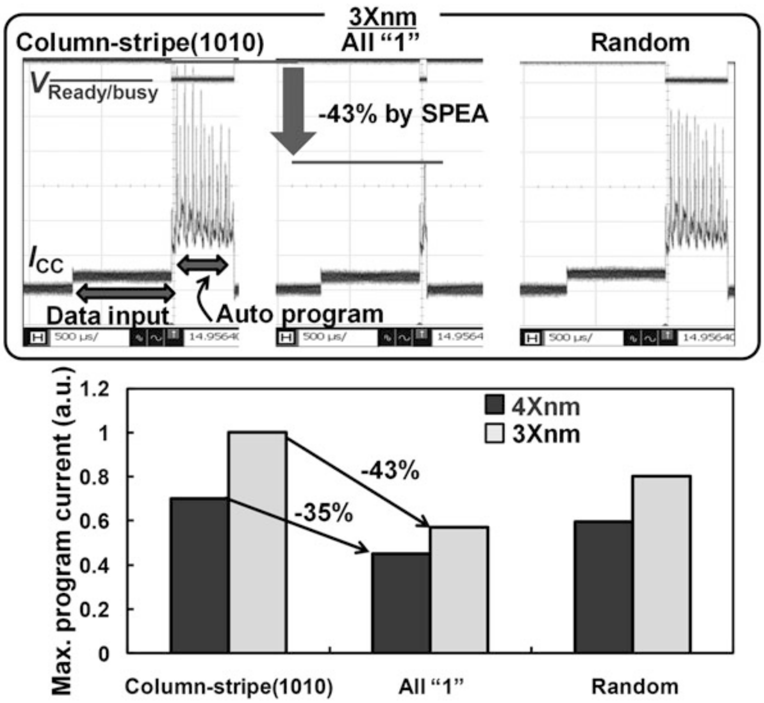


Fig. 7.33 Measured power reduction with SPEA

The PVG with the AFD controller brings a voltage scaling merit for a NAND Flash memory and realizes a drastic power reduction of the 3D-integrated SSD.

Besides circuit technologies, Asymmetric Coding can be used to reduce the number of errors that ECC needs to handle: the population of high V_{TH} states is decreased and the cell error is reduced by 95%. Finally, Stripe Pattern Elimination Algorithm eliminates the worst program data pattern and decreases the overall power during programming.

References

1. K. Takeuchi, NAND successful as a media for SSD. ISSCC, Tutorial T7 (2008)
2. K. Takeuchi, Novel co-design of NAND flash memory and NAND flash controller circuits for sub-30 nm low-power high-speed Solid-State Drives (SSD), in *Symposium on VLSI Circuits Tech. Dig*, 2008, pp. 124–125
3. K. Takeuchi, Novel co-design of NAND flash memory and NAND flash controller circuits for sub-30 nm low-power high-speed solid-state drives (SSD). IEEE J. Solid-State Circuits **44**(4, April), 1227–1234 (2009)
4. K. Ishida et al., A 1.8 V 30nJ adaptive program-voltage (20 V) generator for 3D-integrated NAND flash SSD, in *ISSCC Tech. Dig*, 2009, pp. 238–239

5. C. Park et al., A high performance controller for NAND flash-based solid state disk (NSSD), in *NVSMW Tech. Dig.*, 2006, pp. 17–20
6. Y. Li et al., A 16 Gb 3b/cell NAND flash memory in 56 nm with 8 MB/s write rate, in *ISSCC Tech. Dig.*, 2008, pp. 506–507
7. N. Shibata et al., A 70 nm 16 Gb 16-level-cell NAND flash memory, in *Symposium on VLSI Circuits Tech. Dig.*, 2007, pp. 190–191
8. K. Takeuchi et al., A 56 nm CMOS 99 mm^2 8 Gbit multi-level NAND flash memory with 10 Mbyte/s program throughput. IEEE J. Solid-State Circuits **42**, 219–232 (2007)
9. T. Tanaka et al., A quick intelligent program architecture for 3 V-only NAND EEPROMs, in *Symposium on VLSI Circuits Tech. Dig.*, 1992, pp. 20–21
10. K. Takeuchi et al., A multipage cell architecture for high-speed programming multilevel NAND flash memories, in *Symposium on VLSI Circuits Tech. Dig.*, 1997, pp. 67–68
11. T. Hara et al., A 146 mm^2 8 Gb NAND flash memory with 70 nm COMS technology, in *ISSCC Tech. Dig.*, 2005, pp. 44–45
12. K.D. Suh et al., A 3.3 V 32Mb NAND flash memory with incremental step pulse programming scheme. in *ISSCC Tech. Dig.*, 1995, pp. 128–129
13. K. Takeuchi et al., A source-line programming scheme for low voltage operation NAND flash memories, in *Symposium on VLSI Circuits Tech. Dig.*, 1999, pp. 37–38
14. K. Takeuchi et al., A double-level-Vth select gate array architecture for multilevel NAND flash memories. in *Symposium on VLSI Circuits Tech. Dig.*, 1995, pp. 69–70
15. R. Sundaram et al., A 128 Mb NOR flash memory with 3 MB/s program time and low-power write using an in-package inductor charge-pump, in *ISSCC Dig. Tech. Papers*, 2005, pp. 50–51
16. K. Takeuchi et al., A 56 nm CMOS 99 mm^2 8 Gb multi-level NAND flash memory with 10 MB/s program throughput, in *ISSCC Dig. Tech. Papers*, 2006, pp. 144–145
17. K. Kanda et al., A 120 mm^2 16 Gb 4-MLC NAND flash memory with 43 nm CMOS technology, in *ISSCC Dig. Tech. Papers*, 2008, pp. 430–431
18. T. Tanzawa, T. Tanaka, A stable programming pulse generator for single power supply flash memories. IEEE J. Solid-State Circuits **32**(6), 845–851 (1997)
19. S. Tanakamaru et al., 95%-lower-BER 43%-lower-power intelligent Solid-State Drive (SSD) with asymmetric coding and stripe pattern elimination algorithm, in *ISSCC Dig. Tech. Papers*, 2011, pp. 204–205
20. K. Takeuchi, NAND successful as a media for SSD, in *ISSCC*, Tutorial T-7, 2008
21. J.D. Lee et al., A new programming disturbance phenomenon in NAND flash memory by source/drain hot-electrons generated by GIDL current, in *Non-Volatile Semiconductor Memory Workshop (NVSMW)*, Monterey, CA, USA, 2006, pp. 31–33
22. M. Abraham, NAND Flash Trends for SSD/Enterprise, in *Flash Memory Summit*, 2010
23. Y.Y. Tai, Error control coding for MLC flash memories, in *Flash Memory Summit*, 2010
24. K. Takeuchi et al., A multipage cell architecture for high-speed programming multilevel NAND flash memories, in *Symposium on VLSI Circuits Dig. Tech. Papers*, 1997, pp. 67–68

Chapter 8
SSD Reliability

C. Zambelli and P. Olivo

Abstract SSD are complex electronic systems prone to wear-out and failure mechanisms mainly related to their basic component: the Flash memory. The reliability of a Flash memory depends on many technological and architectural aspects, from the physical concepts on which the store paradigm is achieved to the interaction among cells, from possible new physical mechanisms arising as the technology scales down to the countermeasures adopted within the memory controller to face erroneous behaviors.

The SSD reliability is here analyzed at different levels: from the basic physical mechanisms affecting the traditional floating-gate cells and the possibility of anomalous erratic behavior, up to the disturbs arising because several cells share the same control lines. Solutions adopted to improve system reliability are presented, such as the use of RAID or the protection against power loss during write operations. Finally, test methods for endurance and retention verification are described.

8.1 Introduction

Flash-based Solid State Drives (SSD) envision a complex electronic system featuring different components, each one constituted with a proper wear-out and failure mechanisms' characteristics. However, analyzing in depth the reliability metrics exposed by the SSD to the end-user, it is possible to evidence the bottleneck of the overall system reliability: the non-volatile NAND Flash memories constituting the SSD core.

NAND Flash memories come with unique reliability concerns ascribed both to the physical nature of the storage medium and to the architectural nature of the

C. Zambelli (✉) • P. Olivo
Dipartimento di Ingegneria, Università di Ferrara, Via Saragat 1, Ferrara, Italy
e-mail: cristian.zambelli@unife.it; piero.olivo@unife.it

memory (i.e. how the storage of the information is arranged within the memory). To ensure correct and reliable operations of SSD devices it is mandatory to guarantee correct and reliable operations of the NAND memories. This criterion needs to be pursued along the whole lifetime of the SSD device. In particular, the minimum number of write operations and the ability of keeping unaltered the stored information for years must be guaranteed.

Such a goal is difficult to achieve since new physical mechanisms and architectural issues arise as the technology scales down. To deal with such a target, the co-integration of the NAND memories with external reliability management systems (e.g. ECC, wear leveling, etc.) is required, with the drawback of an increased SSD complexity.

This chapter tackles the SSD reliability issues at different levels. The first part of the chapter will provide an overview of the main reliability mechanisms affecting the traditional floating-gate based NAND Flash showing as these effects increase dramatically their impact in Multi-Level Cells (MLC) architectures. Then, solutions adopted to improve system reliability are described, such as the use of RAID (Redundant Array of Independent Disks) techniques or the protection against power down during programming. Finally, the last part of this chapter will describe the SSD endurance and retention test methods at a system level, considering the NAND memories as a black-box where data are written (and read from) without any knowledge of their technology and architecture.

8.2 Reliability at Physical Level

As stated in the introduction, the core of the SSD is the NAND Flash. Its concept is based on a metal oxide semiconductor device with a floating gate electrically isolated by means of a tunnel oxide and of an interpoly oxide as sketched in Fig. 8.1 [1]. The former oxide plays a basic role for the control of the device threshold voltage whose value represents, from a physical point of view, the stored information. Electrons transferred into the floating gate produce a threshold voltage variation, thus varying the logic data stored within the memory. In quiescent conditions, thanks to the two oxides, the charge stored does not leak away (theoretically), thus granting the nonvolatile paradigm fulfillment.

The cells are rearranged into an array organization [2] whose architecture determines the memory operations in terms of algorithms and applied voltages.

The physical mechanism used for both injecting and extracting electrons to/from the floating gate is the Fowler-Nordheim (FN) tunneling [3]. High electric fields applied to the tunnel oxide allow for electron transfer across the thin insulator to and from the floating gate. In NAND architectures the electron tunneling involves the MOS channel/substrate and requires appropriate biasing of control gate and bulk terminals, whereas drain and source are left floating. The choice of using the tunneling mechanism for writing and erasing the information in the NAND memories is due to the relatively high parallelism of the operation (i.e. thousands of cells belonging to the same page can be written or erased in parallel), although this

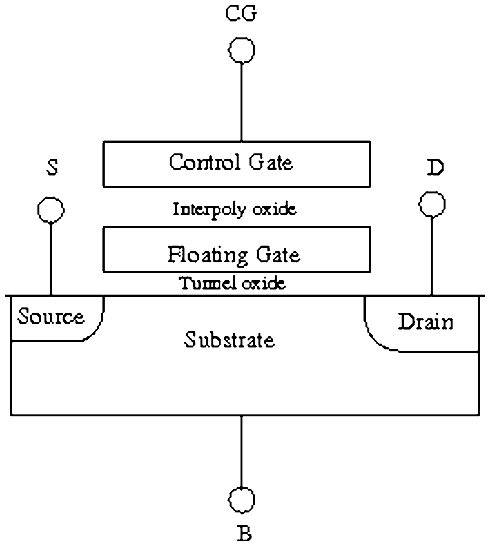


Fig. 8.1 Typical NAND Flash cell structure

significantly impact the reliability of the memory causing progressive degradation of the tunnel oxide. All the NAND Flash modules constituting the SSD undergo a large number of Write/Erase cycles. Every cycle involves very high electric fields applied to the tunnel oxide, eventually mining the reliability of the memory.

In the forthcoming sections we will analyze the basic physical mechanisms related to the tunnel oxide degradation, affecting both memory endurance and data retention. The thin tunnel oxide may be also responsible for other detrimental effects, such as over-programming and erratic bits, possibly causing performance loss in terms of write and read throughput reduction.

8.2.1 NAND Flash Endurance

The endurance of a NAND Flash is defined as the minimum number of Write (i.e. Program/Erase) cycles that the module can withstand before leading to a failure. The erased and programmed statistical threshold distributions along the NAND array must be suitably separated, in order to correctly read the logical state of a cell. The difference between EV (Erase reference Value) and PV (Program reference Value) is defined as the “read window”. However, keeping a correct read window is not sufficient to guarantee a correct read operation: if during its lifetime the threshold voltage of an erased cell exceeds the EV limit and approaches 0 V, the current flowing through the cell may be not high enough to be identified as “erased” by the reading circuitry, thus producing a read error. Similarly, a programmed cell could be read as “erased” if its threshold voltage becomes lower than PV and approaches 0 V. As for the programmed distribution, it is also important that the upper threshold limit does not increase significantly with time, since a too high threshold can block the current flowing through the strings during read operations.

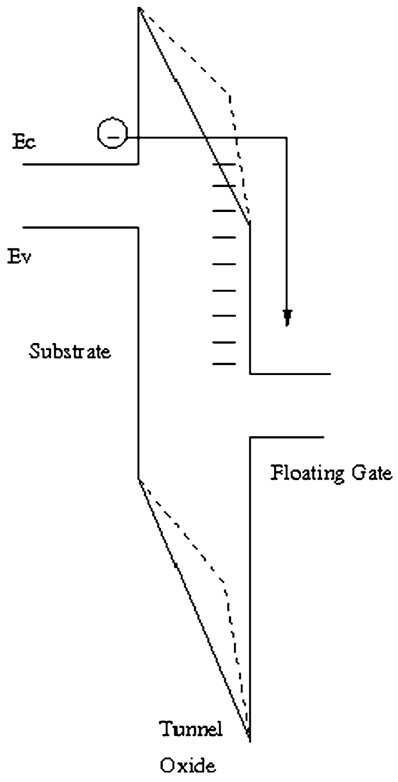


Fig. 8.2 Band diagram during a program operation: without traps (*solid lines* in the oxide region) and with traps (*dashed lines* in the oxide regions)

FN tunneling leads intrinsically to oxide degradation [4]. As a result of consecutive electron tunneling, traps are generated into the oxide [5]. When filled by electrons, charged traps can increase the potential barrier thus reducing the tunneling current, as shown in Fig. 8.2.

Since the programming and erasing pulses feature constant amplitude and duration, less charge is transferred to and from the floating gate causing an efficiency reduction of both the program and erase operations. A narrowing of the read margin window is then expected.

The charge trapped inside the oxide also produces a threshold shift directly proportional to its amount. This shift is symmetrical due to the trap nature and increases the threshold voltage of both erased and programmed cells.

Writing waveform optimization can help in limiting the trapped charge. For example, it has been shown that the window closure can be reduced by using low voltage erasing pulses able to remove the charge accumulated in the oxide [6].

As shown in Fig. 8.3, the threshold voltage shifts increase with the number of program/erase operations until an endurance failure occurs. Threshold voltage shifts could be recovered by applying specific procedures that, however, result inconvenient when comparing their effects with the required architectural overheads and time consumption.

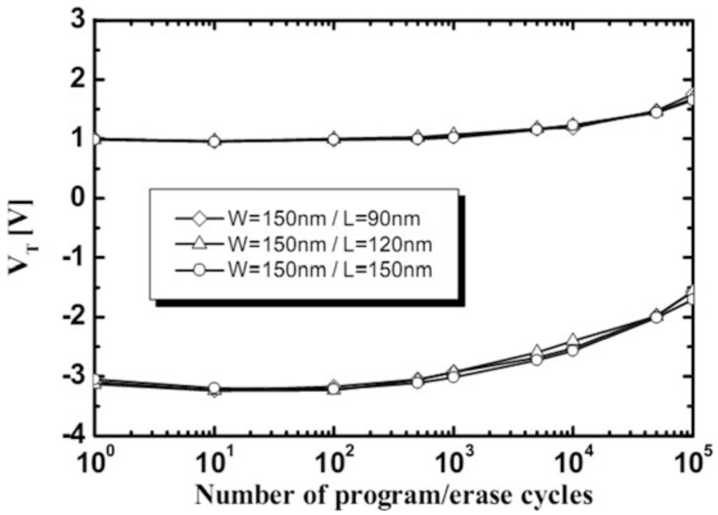


Fig. 8.3 Threshold voltage degradation during cycling of NAND Flash with different geometrical features [7]

As evidenced in Fig. 8.3, the most critical effect is the increase towards 0 V of the erased threshold. To check whether all cells of a sector have been correctly erased (all threshold voltages must be below 0 V), an erase verify procedure is applied after any erase operation. It consists in a particular read operation performed by driving simultaneously all the word lines of the sector at 0 V: if the read current in a bitline is 0, it means that at least a cell blocked the current flow because its threshold voltage was higher than 0 V. The entire block is marked as *bad block* by the memory controller and no longer addressed [2].

8.2.2 NAND Data Retention

The retention concept is the ability of the NAND memory to keep a stored information over time with no biases applied. Electron after electron, charge loss could slowly leading up to a read failure: a programmed cell can be read as erased if its threshold voltage shifts below 0 V.

The intrinsic retention is mainly limited by tunneling (see Fig. 8.4) through the oxide. The minimum limit imposed by present standard is the retention of the data for 10 years at a temperature defined by the application segment where NAND Flash memory is targeted on [8].

The cells retention worsens with memory cycling and this effect is appreciable as a reduction of the threshold voltage levels as sketched in Fig. 8.5. Charge loss from the floating gate moves the threshold voltage distribution towards lower values.

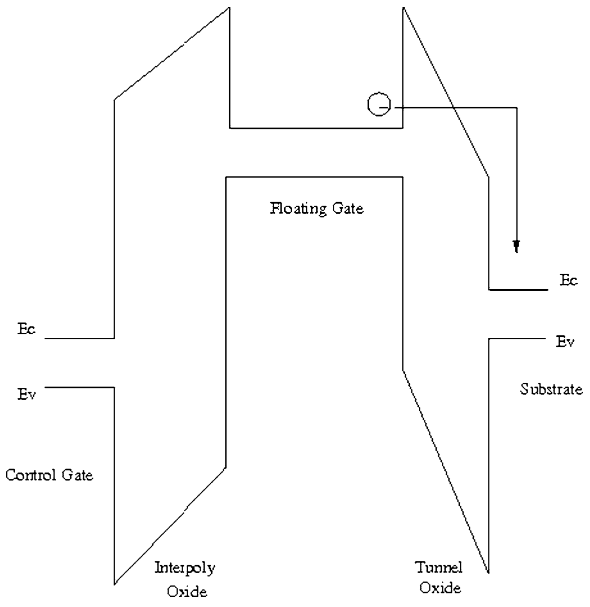


Fig. 8.4 Band diagram of the cell when programmed and not biased. The main mechanism for data loss is tunneling through the tunnel oxide [2]

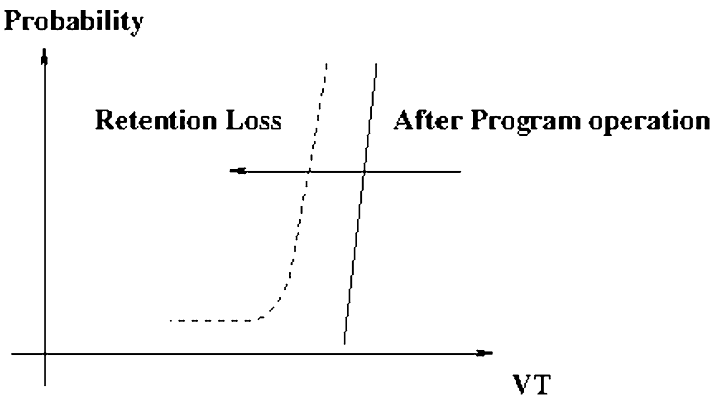


Fig. 8.5 Cumulative distribution of a NAND array in program state. Both de-trapping and SILC effects are appreciable on the time evolution

In additions, a tail in the lower part of the distribution indicates that a small percentage of cells is losing charge faster than average.

The rigid shift of the cumulative V_T distribution can be related to the oxide degradation within the oxide and at the Si – SiO_2 interface. As described in the previous section, successive electron tunneling leads intrinsically to oxide degradation, characterized by traps generation. These traps may be responsible for charge loss from the floating gate towards the silicon substrate. In fact, an

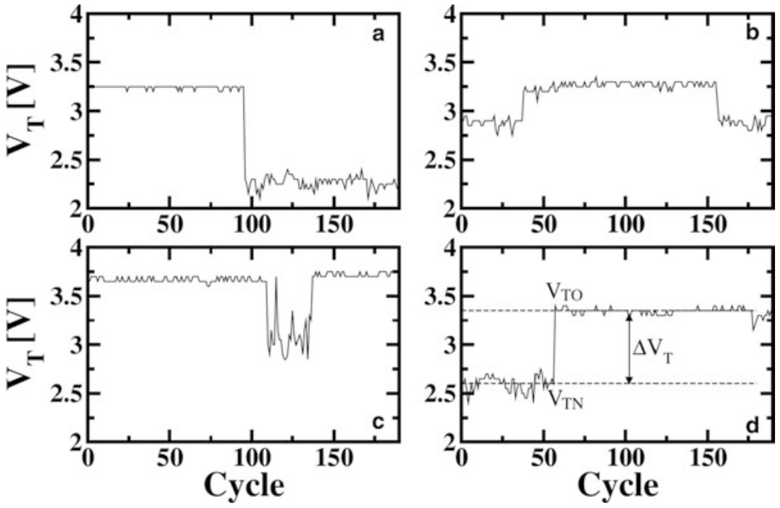


Fig. 8.6 Example of erratic behaviors in four Flash cells. Cells threshold voltage V_T plotted versus the number of cycles exhibits RTN features [9]

empty trap suitably positioned within the oxide can activate *Trap Assisted Tunneling* (TAT) or SILC (*Stress Induced Leakage Current*) mechanisms [8] characterized by a significantly higher tunnel probability. In addition, an electron trapped within the oxide during writing operations and responsible for the threshold voltage increase leading up to endurance failures may be de-trapped when the program pulse is switched off, when the cell is read or even when the cell is not addressed.

It is clear that these mechanisms are strongly related to the oxide degradation and therefore data retention decreases with the number of applied writing pulses.

The position of failure retention cells within the array, however, does not show any clustering that could be related to a technological process.

8.2.3 Erratic Bits and Over-Programming

The FN tunneling mechanism for writing and erasing data in NAND Flash has demonstrated to guarantee a sufficient level of reliability and performance as it has been continuously used throughout various technological generations of Flash up to nowadays.

Nevertheless, it has been found that anomalous FN tunneling currents can occur in random periods of time that can lead to significant variations of the threshold voltage achieved by the cell after the writing operation [9] (see Fig. 8.6). This phenomenon is known as *erratic bits*.

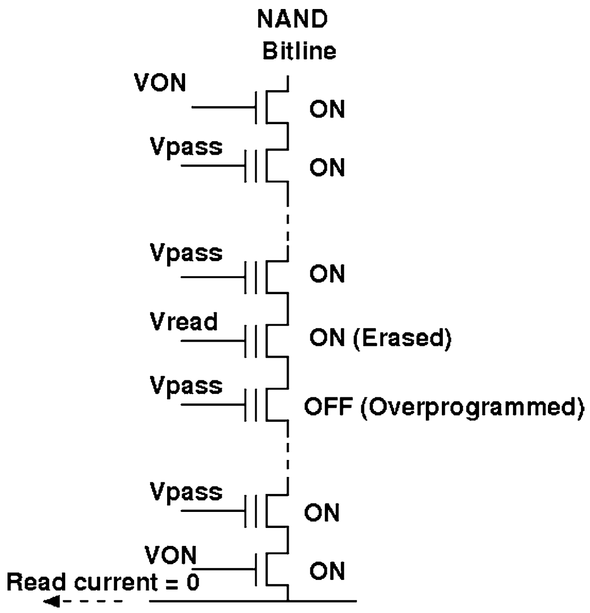


Fig. 8.7 Effect of an over-programmed cell in a NAND Flash string. In normal conditions the status of the cell to be read (supposed to be erased, thus ON) is correctly detected, since all other cells behave as ON pass transistors. In the presence of an over-programmed cell, the current flow through the string is inhibited and the absence of current is attributed to a programmed status of the cell to be read, thus producing a logical error [9]

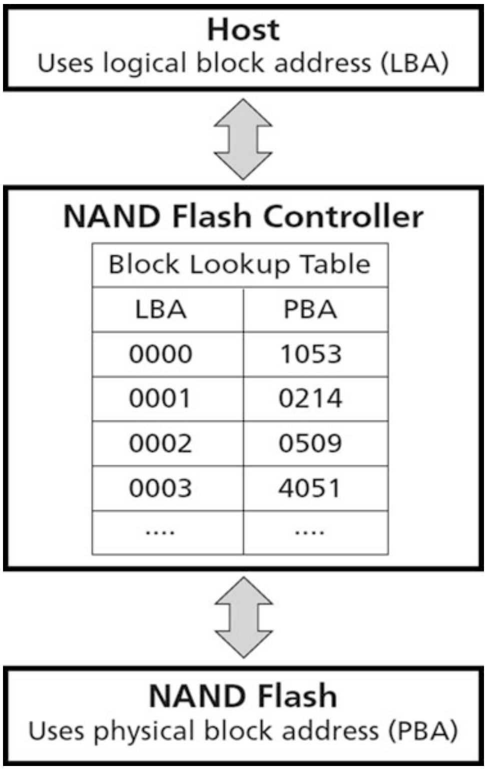
In a NAND array, the presence of this phenomenon is detrimental for the performances of the memory as the sudden major increase of the cells threshold voltage may eventually induce the *over-programming* issue. As shown in Fig. 8.7, erased cells are erroneously read as programmed since over-programmed cells, featuring relatively large threshold voltage, can electrically isolate the NAND string, thus causing read errors and consequent read throughput loss due to the additional work done by the Error Correcting Codes (ECC) trying to repair the failed bits.

Since erratic behaviors are intimately related to the electron tunneling mechanism, they can potentially affect all the cells of an array [9].

Anomalous tunneling has been related to the presence/absence of a cluster of positive charges in the tunnel oxide that strongly affects the result of the FN tunneling operation. In first approximation, erratic behaviors can therefore be described in terms of a two level Random Telegraph Noise (RTN) affecting the threshold voltage during cycling, in which the normal and the over-programmed threshold voltage levels are the result of the presence of a cluster of more than 2, or less than 3, positive charges in the tunnel oxide, respectively.

8.2.4 Reliability Considerations on SLC/MLC NAND Architectures

MLC (Multi Level Cells) architectures are more prone to reliability effects related to oxide degradation with respect to SLC (Single Level Cells) memories. In the MLC approach, the separation between two adjacent threshold levels is a fraction of the read margin typical of a SLC architecture. Therefore, even a slight threshold voltage


Host
Uses logical block address (LBA)
NAND Flash Controller
Block Lookup Table
NAND Flash
Uses physical block address (PBA)

LBA	PBA
0000	1053
0001	0214
0002	0509
0003	4051
....	....

Fig. 8.8 Logic Block Address (*LBA*) to Physical Block Address (*PBA*) translation in wear leveling systems for NAND Flash [11]

variation may lead to a logical error. As a consequence, while a SLC architecture can usually withstand 100 k write-erase cycles, the typical cycle number for a MLC memory is 10 k or even less for advanced scaled technology nodes [2]. Indeed, the aggressive scaling of the NAND cells geometry arise new reliability threats principally ascribed to the discretization of the storage medium. Nowadays, the number of electrons deputed to effectively change the threshold voltage of a cell during the write operation are ruled by discrete statistics, further enhancing the cell-to-cell variability effects.

MLC architectures are anyway preferred to SLC ones when high storage density is required, but at the additional cost of increasing the overhead due to the appropriate management policies adopted for mitigating the issues presented in the previous sections of this chapter.

For instance, it is important to distribute the writing stress over the entire population of cells rather than on a single hot spot, thus avoiding that some blocks are updated continuously while the others keep unaltered their charge content.

It is clear that blocks whose information is updated frequently are stressed with a large number of write-erase cycles. In order to keep the aging effects as uniform as possible, the number of both read and write cycles of each block must be monitored and stored by the memory controller. *Wear Leveling* techniques [2, 10] are based on a logical to physical translation for each sector, as shown in Fig. 8.8.

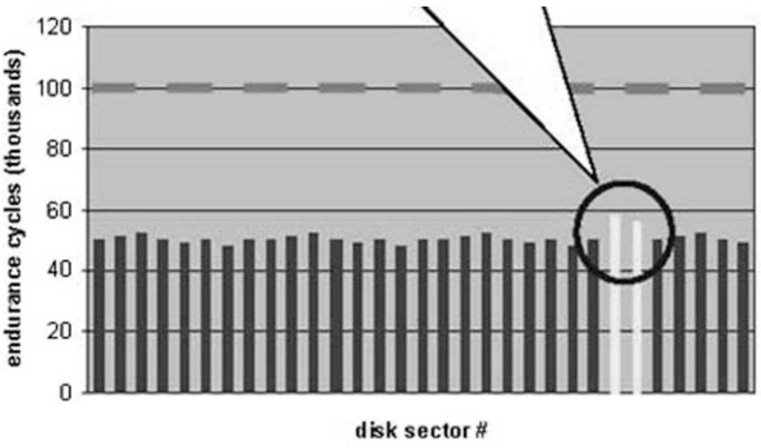


Fig. 8.9 Effects of wear leveling on the NAND endurance feature [12]

Wear leveling is a process that reduces premature wear in NAND Flash devices by equalizing the endurance of a memory on its whole addressable space (see Fig. 8.9). The most common implementation of wear leveling occurs in the SSD Flash controller, which manages access to the memory device and determines how the NAND Flash blocks are used. Depending on the wear-leveling method used, the controller typically either writes to the available erased block with the lowest erase count (dynamic wear leveling); or it selects an available target block with the lowest overall erase count, erases the block if necessary, writes new data to the block, and ensures that blocks of static data are moved when their block erase count is below a certain threshold (static wear leveling).

In this way all the physical sectors are evenly used, thus reducing the overall oxide aging. Wear Leveling techniques are described in more details in Chap. 9.

To reduce possible errors caused by oxide aging and erratic events, Error Correcting Codes (ECC) are widely used in NAND memories and in particular in MLC architectures [10]. BCH and LDPC are the most popular ECC algorithms and they are described in Chaps. 10 and 11, respectively.

8.3 Reliability at Architectural Level

Architectural solutions for memory operations may also affect the overall reliability, by producing errors and even cell failures. The most common effects are the so called “disturbs”, that can be interpreted as the influence of an operation performed on a cell (Read or Write) on the charge content of a different cell.

Read disturbs are the most frequent source of disturbs in NAND architectures. This kind of disturb may occur when reading many times the same cell without any

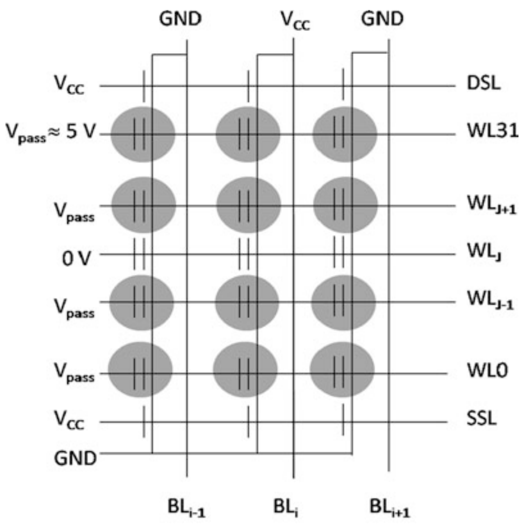


Fig. 8.10 Representation of read disturb in a NAND Flash array. The cells potentially affected by the disturb are marked in *gray*

erase operation. All the cells belonging to the same string of the cell to be read must be driven in an ON state, independently of their stored charge. The relatively high V_{pass} bias applied to the control gate and the sequence of V_{pass} pulses applied during successive read operations may induce a charge increase. These cells suffer a positive shift of their threshold voltage that may lead to read errors, when addressed. Figure 8.10 shows the typical read disturb configuration.

The probability of suffering from read disturb increases with the cycle number (i.e. towards the end of the memory useful lifetime) and it is higher in damaged cells. Read disturbs do not provoke permanent oxide damages: if erased and then reprogrammed, the correct charge content will be present within the floating gate.

Two other important typologies of disturbs are related to the write operation: the *Pass disturb* and the *Program disturb,* which are shown in Fig. 8.11a, b, respectively. The former is similar to the read disturbs and affects cells belonging to the same string of a cell to be programmed. With respect to the read disturb, the Pass one is characterized by the higher V_{pass} voltage applied to cells that are not to be programmed (thus enhancing the electric field applied to the tunnel oxides and the probability of undesired charge transfer). On the other side, the pass disturb may be provoked, in the worst case, by a program operation on all the string cells but the one affected by the disturb (when a string has been fully programmed, an erase operation must be necessarily performed before any other reprogram): therefore the disturb duration is much shorter and the cumulative effect of successive read pulses encountered in read disturbs is not present.

The Program disturb, on the contrary, affects cells that are not to be programmed and belong to the same wordline of those that are to be programmed.

ECC systems in SSD memory controller efficiently manage these reliability threats as most of the time the errors triggered by disturb are easily identifiable and correctable.

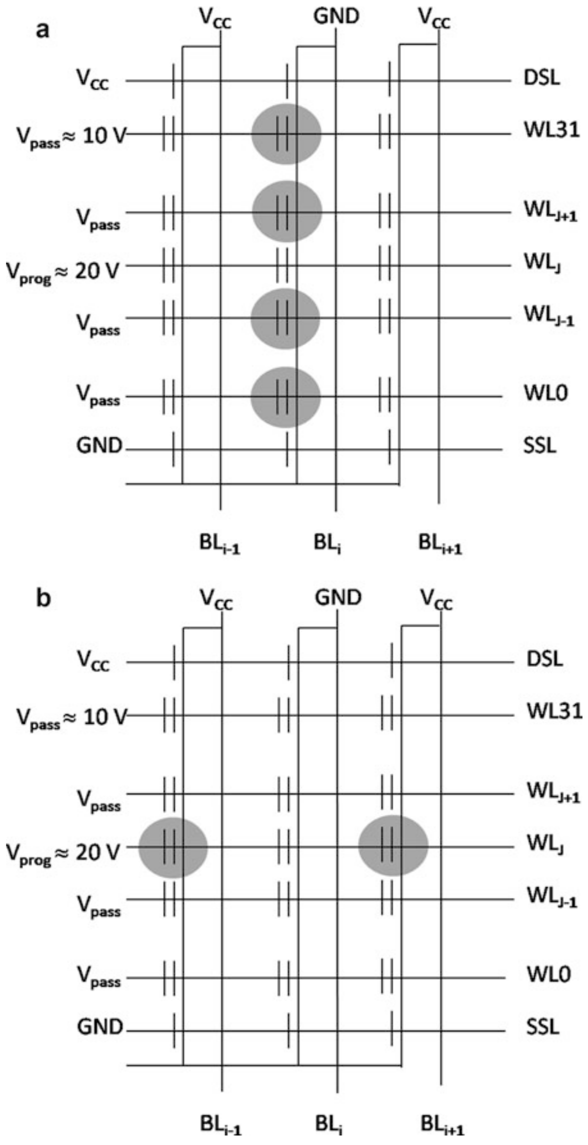


Fig. 8.11 Representation of pass disturb (**a**) and program disturb (**b**) in a NAND Flash array. The cells potentially affected by the disturbs are marked in *gray*

8.4 Reliability at System Level

Since NAND Flash modules are the most critical parts of an SSD, it is obvious that focusing the attention on the Flash reliability is the best way to improve the overall system reliability. From a system point of view, however, it is also possible to implement some specific solutions that may improve the SSD reliability. A practical example in the hard disk drive (HDD) world is the use of RAID (Redundant Array of Independent Disks) protection or the use of cache to buffer HDD from small block random inputs and outputs. These solutions can be applied to Flash-based solid state storage as well [13].

8.4.1 RAID Systems

In a traditional HDD storage array, RAID techniques are based on the storing of the same data in different locations (thus, redundantly) on multiple hard disks. Placing data on multiple disks increases the mean time between failures and fault tolerance, thus increasing reliability. RAID concepts can also be applied to an array of Flash boards (modules or disks), resulting in similar increases in reliability.

While wear leveling and ECC handle the majority of Flash reliability problems at the module level, a few other concerns need to be addressed by the system designer, such as failures in the module responsible for error correction or other system failures that are not related to the memory blocks. It is worth to point out that NAND Flash is constituted not only by the memory array, but also by several sub-systems such as controllers, data interfaces, etc. Thanks to RAID protection, a failure external to the cell array does not result in data loss.

A brief textual summary of the most commonly used RAID levels is here reported [14].

RAID 0 (block-level striping without parity or mirroring) has no (or zero) redundancy. It provides improved performance and additional storage but no fault tolerance. Any SSD drive failure destroys the array, and the likelihood of failure increases with more drives in the array due to the fact that the data are broken into fragments called blocks along different SSD drives. The number of blocks is dictated by the stripe size, which is a configuration parameter of the array. The blocks are written to their respective drives simultaneously on the same sector. This allows smaller sections of the entire chunk of data to be read off each drive in parallel, increasing bandwidth. RAID 0 does not implement error checking, so any error is uncorrectable.

In ***RAID 1*** (mirroring without parity or striping), data is written identically to two drives, thereby producing a “mirrored set”; at least two drives are required to constitute such an array. The array continues to operate as long as at least one drive is functioning. With appropriate operating system support, there can be increased read performance, and only a minimal write performance reduction; implementing RAID 1 with a separate disk controller for each drive in order to perform simultaneous reads (and writes) is sometimes called *multiplexing* (or *duplexing* when there are only two drives).

In ***RAID 2*** (bit-level striping with dedicated Hamming-code parity), all SSD operations are synchronized, and data are striped such that each sequential bit is on a different drive. Hamming-code parity is calculated across corresponding bits and stored on at least one parity drive.

In ***RAID 3*** (byte-level striping with dedicated parity), all SSD operations are synchronized, and data are striped so each sequential byte is on a different drive. Parity is calculated across corresponding bytes and stored on a dedicated parity drive.

RAID 4 (block-level striping with dedicated parity) is identical to RAID 5 (see below), but confines all parity data to a single drive. In this setup, files may be distributed between multiple drives. Each drive operates independently, allowing I/O requests to be performed in parallel. However, the use of a dedicated parity drive could create a performance bottleneck; because the parity data must be written to a single, dedicated parity drive for each block of non-parity data, the overall write performance may depend a great deal on the performance of this parity drive.

RAID 5 (block-level striping with distributed parity) distributes parity along with the data and requires all drives but one to be present to operate; the array is not destroyed by a single drive failure. Upon drive failure, any subsequent reads can be calculated from the distributed parity such that the drive failure is masked from the end user. However, a single drive failure results in reduced performance of the entire array until the failed drive has been replaced and the associated data rebuilt. RAID 5 requires at least three disks.

RAID 6 (block-level striping with double distributed parity) provides fault tolerance of two drive failures; the array continues to operate with up to two failed drives. This makes larger RAID groups more practical, especially for high-availability systems. This becomes increasingly important as large-capacity drives lengthen the time needed to recover from the failure of a single drive. Single-parity RAID levels are as vulnerable to data loss as a RAID 0 array until the failed drive is replaced and its data rebuilt; the larger the drive, the longer the rebuild takes. Double parity gives additional time to rebuild the array without the data being at risk if a single additional drive fails before the rebuild is complete.

8.4.2 Caching

It is somewhat unusual to think of caching as a mechanism to improve system reliability, but the use of a large RAM cache in front of Flash RAID is an excellent way to improve both endurance and performance of the write operation. If several write accesses are to be performed within a Flash module, each one characterized by data dimensions smaller than a page, the number of memory accesses affects the SSD endurance. The large RAM cache aggregates small data blocks into patterns that are Flash friendly, i.e. that make the most efficient use of every Flash cycle by writing optimized data lengths. In this way caching techniques can reduce the number of program/erase cycles thus improving both performances and reliability.

8.5 Reliable Data Management in Power Failure Scenarios

When programming a NAND Flash memory, the program operation must complete to ensure that data are stored reliably within the page. Data are at risk if power is lost when Flash memory cells are in the process of being programmed [15]. SSD have three causes of potential data loss or corruption when system power fails:

Loss of data. This can occur due to the implementation of write caching to achieve peak performance. In this case, the host system is informed that a write operation has completed when, actually, it is still in process. If power fails while the controller is “catching up” with the write operation, the data in the write buffer are not yet hardened and can be lost. When the data are requested later by the host, the controller can either report that data are irrecoverable or (depending on the controller design) it can deliver a previous “stale” version of those sectors to the host. In the latter case, this translates to silent data corruption, since the host system is not informed that the data delivered are incorrect.

Loss of mapping information. Every SSD controller uses mapping information to translate from the host’s Logical Block Address (LBA) to physical Flash memory locations. Mapping information must be created and maintained if the data is to be later retrieved from the drive, and must be updated whenever new data is written to a previously written LBA. If the mapping information is lost when power fails, the drive may show data corruption, deliver stale (corrupted) data or may not be capable of supporting logical I/O on the next power up.

Lower page corruption. MLC NAND Flash uses each physical page to store the data of two logical pages (each memory cell stores two bits). In a tow-rounds algorithm [2] the lower page (the logical page addressed by the lower of the two addresses) is programmed first, followed by the upper page. When programming the upper page, programming voltages are applied to the same cells already storing valid data in the lower page. If power fails while the upper page is being programmed, data in that page are lost, and already-stored data in the lower page are corrupted as well. When these data belonging to the lower page are requested later by the host, the SSD will report the data as irrecoverable.

8.5.1 Power Failure Circuitry in SSDs

Most enterprise and industrial-class SSDs rely on power failure circuitry that monitors the supply voltage and generates an “early warning” signal to the SSD

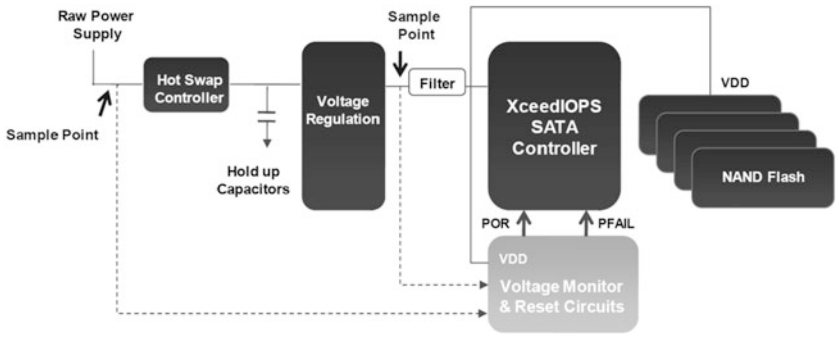


Fig. 8.12 Block diagram of a power failure circuit in a standard SSD [15]

controller if the voltage drops below a predefined threshold (see Fig. 8.12). A secondary voltage hold-up-circuit is implemented to ensure the drive has power for a sufficient time to harden data whenever that warning is received. In addition, writes are not accepted by the drive until the secondary voltage source has been sufficiently charged to protect against loss of data upon power failures.

The secondary voltage source can be a high capacity “supercapacitor” or a bank of discrete capacitors. These solutions are not the same from a performance and reliability standpoint. Descriptions and relevant tradeoffs of a supercapacitor solution and a bank of discrete capacitors in an Enterprise-class SSD design are presented below.

8.5.2 *Supercapacitors*

A supercapacitor is an electrolytic capacitive charge storage device. It is capable of storing a large amount of energy in a relatively small three-dimensional space. A generic supercapacitor-based voltage hold-up circuit is consistent with the block diagram shown in Fig. 8.12. Designing a supercapacitor-based power failure protection circuit is easy to do, and many SSDs employ this approach for this reason. Unfortunately, there are a number of concerns related to long term supercapacitor reliability that makes the use of this component questionable for Enterprise-class SSDs. Supercapacitors are typically Aluminum Electrolytic Capacitors, featuring a high capacitance-to-size ratio and, therefore, they are an attractive choice for applications requiring large bulk capacitance like an SSD. However, like all electrolytic capacitors, supercapacitors suffer from a well known set of deficiencies with regard to long term reliability. In particular, supercapacitors “wear out”, resulting in reduced capacitance over time. They use a wet electrolyte, and the packaging is subject to ongoing losses via leakage and diffusion. The performance of the supercapacitor degrades slowly with electrolyte loss, until the onset of total failure occurs with little or no warning. In addition, loss rate

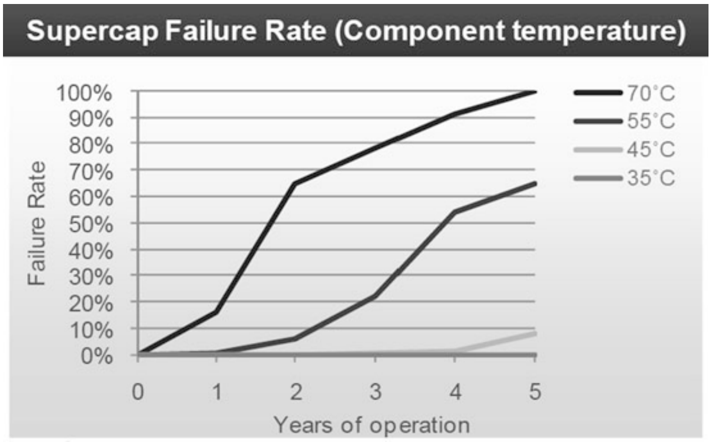


Fig. 8.13 Supercapacitor failure rate with respect to temperature [15]

increases with higher operating voltage, and in higher operating and non-operating temperature environments. For every 10°C of ambient operating temperature rise, the life expectancy of a supercapacitor can be cut approximately in half.

For these reasons, supercapacitors may be not enough reliable to meet the required reliability standards for the enterprise and industrial computing markets served by the SSDs product line. Figure 8.13 shows an example of a supercapacitor reliability projection, based on component life test data. Due to the reliability concerns associated with this capacitor type, it is imperative that an SSD constantly monitors the capacitor's operating capabilities to ensure continued reliable operation as the SSD ages. This is done by periodically measuring the supercapacitor's charge/discharge ("Hold Up") time under a controlled load. The challenges associated with performing this test seamlessly and transparently to the host system are many. Because the secondary power system is under a "live test" during hold up time measurements, the SSD must harden data prior to testing (in case the test fails). This operation and the test time itself almost always result in extended latencies (as much as 100 ms or more) for host commands issued during the test interval.

8.5.3 Discrete Capacitors

This approach requires more design expertise, but overcomes the supercapacitor limitations. A discrete capacitor-based voltage hold-up circuit employs a bank of discrete capacitors connected in parallel, as shown in Fig. 8.14.

Nowadays SSDs utilize either Niobium Oxide or Polymer Tantalum capacitors. These discrete capacitors do not employ a "wet" electrolyte and are not susceptible to the leakage related issues that plague supercapacitor technology. Niobium and Polymer Tantalum capacitors are rated to 85°C, providing a higher temperature

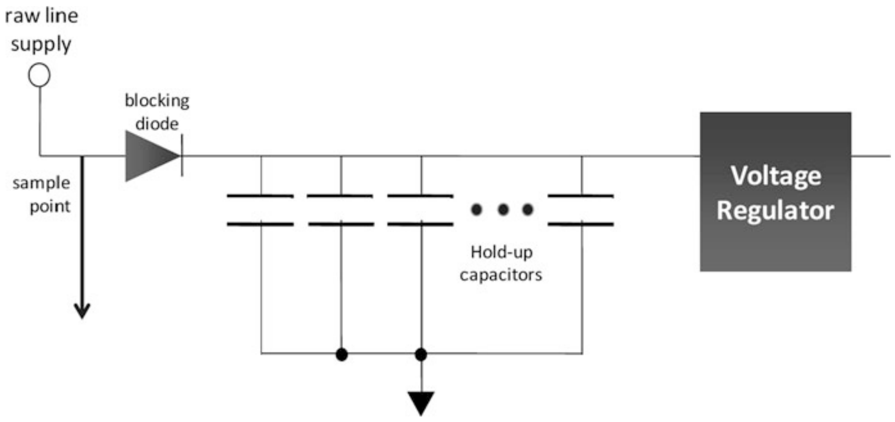


Fig. 8.14 Discrete capacitor rail Hold-up sub-circuit [15]

operating range with respect to supercapacitors (70°C). As a result of these factors, a discrete component based hold up circuit is more able to meet the demands of enterprise and industrial computing environments. Another advantage of discrete capacitors over supercapacitors is that they are highly predictable and reliable. Provisioning can be selected so that it is optimal for the SSD's needs over its lifetime. However, lacking the compactness of supercapacitors, the capacitance-to-size ratio of a discrete solution is less space efficient and its implementation require a more careful design.

8.6 Endurance and Retention Verification in SSDs

As already stated in the previous sections, the overall SSD endurance and retention majorly depend on that of the NAND memories that represent the heart of such devices. The following sections of this Chapter will present the SSDs' reliability requirements in terms of endurance and retention, the criteria adopted to verify these requirements and the possible verification methods.

When testing SSDs (and not single NAND flash chips) it is important to detect or estimate functional failures, errors in reading data, without considering the physical causes that produced such errors or failures. If the amount of errors or functional failures exceeds the acceptable limits, a successive failure analysis will try to investigate on the possible physical causes. Therefore, it is important to remind the basic difference between testing NAND chips and verifying SSDs reliability: the former operation requires to adopt all the possible test procedures to excite physical or architectural weaknesses, the latter consider the SSD as a black box where data are to be written, read and retained at their endurance and retention limits.

Table 8.1 Relationship between User-addressable logical block count (*LBC*) and SSD capacity for different Logical block sizes

Logical block size [bytes]	512	4,096
User-addressable logical block count (LBC)	21,168 + (1,953,504 × SSD capacity)	2,646 + (244,188 × SSD capacity)
SSD capacity [Gbytes]	(LBC – 21,168)/1,953,504	(LBC – 2,646)/244,188

It must be observed, however, that the use of different technologies for NAND memories produces different expectations in terms of both endurance and retention. To deal with different applications, NAND technologies, and producers, standard committees define the conditions of use and the corresponding endurance verification requirements. The following sections will refer to the JEDEC standard JESD218A (Solid-State Drive Requirements and Endurance Test Method) [16], that defines parameters for standardized endurance rating so that the end user may consider the endurance rating as a factor in determining if an SSD is suitable for his particular application.

Since there are different levels of requirements for an SSD based on specific applications and different levels of testing should be applied to verify the SSD suitability for the particular application, it is necessary to group different applications characterized by similar requirements in a limited number of classes: to this purpose, the JESD218A standard considers just two application classes: client and enterprise. These classes, of course, are not all-inclusive and it is clear that variations such as the operating systems and application architectures make a significant impact to the workload of an SSD, that represents the detailed sequence of host writes and reads (including data content and timing) applied during endurance testing. The actual workloads are defined in the JEDEC standard JESD219 [17] for the two considered classes and they are not reported in this text.

8.6.1 SSD Endurance and Retention Rating

A SSD manufacturer shall establish an endurance rating for an SSD that represents the maximum number of terabytes that may be written (*TBW*) by a host to the SSD, such that the following conditions are satisfied:

1. the SSD maintains its capacity, defined as the user-addressable capacity as calculated in Table 8.1 (nonvolatile memory areas reserved for device use are not included in this calculation);
2. the SSD maintains the required Uncorrectable Bit Error Rate (*UBER*) for its application class, where the *UBER* is a metric for the rate of occurrence of data errors, equal to the number of data errors per bit read:

$$UBER = \frac{number\ of\ data\ errors}{number\ of\ bits\ read} \tag{8.1}$$

Table 8.2 SSD class and requirements

Application class	Client	Enterprise
Active use (power on)	8 h/day @ 40°C	24 h/day @ 55°C
Retention use (power off)	1 year @ 30°C	3 months @ 40°C
FFR	$\leq 3\%$	$\leq 3\%$
UBER requirement	$\leq 10^{-15}$	$\leq 10^{-16}$

It is important to state that, in the JESD218A standard, the *UBER* values for SSDs are to be *lifetime* values for the *entire population*. The numerator is the total count of data errors detected over the full *TBW* rating for the population of SSDs in the endurance verification. A sector containing corrupted data is to counted as one data error even if it is read multiple times and each time fails to return correct data. The denominator is the number of bits written at the *TBW* rating limit;

3. the SSD meets the required Functional Failure Requirement (*FFR*) for its application class, that is the allowed cumulative number of failed drives that, over the *TBW* rating, fail to function properly in a way that is more severe than having a data error;
4. the SSD retains data with power off for the required time for its application class.

The requirements for standard classes of SSDs are based on a scenario in which the SSD are actively used for some periods of time during which the SSDs are written to their endurance ratings, followed by a power-down time period in which data must be retained. The requirements for the two SSD classes are reported in Table 8.2.

SSD case temperatures are reported in Table 8.2 and they are intended to represent the relevant temperatures over the respective time periods, for the purpose of endurance and retention estimation, not the maximum and minimum specifications to be found on the SSD datasheets. For the client class, the retention temperature (30°C) is also the temperature for the 16 h/day in which the SSD is off.

8.6.2 Endurance and Retention Stress Methods

There are two approaches for endurance verification: a direct method and a set of extrapolation methods. Both consist of endurance verification followed by retention verification. If the full *TBW* rating can be reached in a 1,000-h stress, the direct method is to be followed. If this is not possible, then an extrapolation method is acceptable. If an SSD product from a qualification family has been qualified using the JESD218A standard, the subsequent products need only data from a 1,000-h direct method evaluation, even if this results in those drives not being fully stressed to their endurance rating limits.

8.6.3 Direct Method

The endurance stress is to be performed both at high and low temperature; then, a retention test shall be performed. Since the retention time requirements are long (see Table 8.2), extrapolation or acceleration is required to validate the retention requirements.

8.6.3.1 Sample Size

For the first product to be qualified in a qualification family, the sample shall consist of SSDs from at least three nonconsecutive production lots and from all the fabrication plants responsible for the manufacture of the NAND memories used in the SSD. For subsequent products from a qualification family, a single production lot is sufficient. The number of SSD in the sample shall be sufficient to establish that both the *FFR* and *UBER* requirements are met at 60% confidence.

The sample size and acceptance criteria are defined by the following equations, which mathematically embody the 60% confidence requirement:

$$UCL(ff) \le FFR \cdot SS \tag{8.2}$$

$$UCL(de) \le \min(TBW,\ TBR) \cdot 8 \cdot 10^{12} \cdot UBER \cdot SS \tag{8.3}$$

where *ff* and *de* are the acceptable numbers of *functional failures* and of *data errors*, respectively; *TBR* represents the number of TBytes Read; *SS* is the sample size in number of drives; *FFR* and *UBER* are expressed as fractions; *UCL*(x) is an upper confidence limit function that depends on the maximum number of accepted errors x.

For instance, for an accept-on-zero plan (no failures/error are accepted), *UCL(0)* = 0.92, while if 1 failure/error is accepted, *UCL(1)* = 2.03 and for 2 failures/errors accepted, *UCL(2)* = 3.11.

As an example, consider an accept-on-zero plan, *FFR* = 0.03 (corresponding to 3%); $UBER = 10^{-16}$, *TWB* = 100, all data read back and verified (therefore *TBR* = 100). Two sample sizes *SS* can be calculated from Eqs. (8.2) and (8.3), respectively:

$$SS \ge UCL(0)\,/FFR\ = 0.92\,/0.03\ = 30.1 \tag{8.4}$$

$$\begin{aligned} SS \ge UCL(0)/\left[\min(TBW,\ TBR) \cdot 8 \cdot 10^{12} \cdot UBER\right] \\ = 0.92/(100 \cdot 8 \cdot 10^{12} \cdot 10^{-16}) = 11.5 \end{aligned} \tag{8.5}$$

The required sample size is the larger of the two results and, therefore, at least 31 SSD must be tested. If the minimum sample size of 31 were chosen, than the verification test would pass if there were no functional failures in 31 drives. However, with $SS = 31$, from Eq. (8.3),

$$UCL(de) \leq 100 \cdot 8 \cdot 10^{12} \cdot 10^{-16} \cdot 31 = 2.48 \tag{8.6}$$

Since $UCL(1) = 2.03 < 2.48 < 3.11 = UCL(2)$, up to one data error would be acceptable. Therefore, the verification would pass if there were no functional failures and no more than one data error.

It is important to notice that *UBER* is defined in terms of bits read, but for the purpose of endurance verification Eq. (8.3) counts the minimum of bits read and bits written. The rationale is twofold.

First, many data errors are transient with respect to rewriting of an SSD, but repeatable with respect to repeated reading. This means that a sector with corrupted data may pass without error if rewritten, however reading non-failing sectors multiple times is unlikely to detect additional errors. This means that if reads are less frequent than writes, then many errors will be missed. All data errors will be detected only if all written data are read before those sectors are rewritten. If the *TBR* is less than the *TBW*, then the *UBER* should be increased because of the likelihood that transient data errors went undetected. Using the *TBR* in place of the TWB accomplishes that goal.

Second, the JEDEC JESD218A standard is aligned to a reference read/write ratio of unity. If the *TBR* is equal to the *TBW*, then the *UBER* may be considered to be an error rate per bit read or per bit written: both are equivalent. If the *TBR* in the endurance stress is greater than the *TBW* the *UBER* must be *TBW* based.

It is important to remind that the previous criterion deals with *endurance* functional failures and *endurance* data errors. Failures that are not related to the act of writing data to its endurance limit, or by the subsequent retention stress, are to be excluded from the endurance verification, even it they must be considered in the drive qualification process. In some cases it is not easy to clearly identify *endurance* and *non-endurance* function failures. Failures that are not in the circuit path of the written data are clearly identified as *non-endurance* failures, while some failures that are in the circuit path of the written data *may* be considered as *non-endurance* failures if the cause of the failures were unrelated to the quantity of data written.

8.6.3.2 Endurance Stress

To verify the endurance capabilities, the drives are stressed to their full endurance specification (in TBW). The stress time depends on the drives performances and on those of the test equipment. If performance variations between test systems or the SSDs themselves cause some SSD to receive more writes than other in a given

Table 8.3 Endurance stress temperatures by drive class

Application class	Client	Enterprise
Low temperature	$\leq 25°C$	$\leq 25°C$
High temperature	$40°C \leq T \leq T_{max}$	$60°C \leq T \leq T_{max}$

stress time, then the endurance specification must be reached by the average amount of data written. All data errors throughout the stress must be recorded, even if those errors are transient in nature. Testing the drive only at the end of the stress cannot be accepted.

Two approaches are acceptable for incorporating both high and low temperatures into the endurance stressing: the ramped-temperature approach and the split-flow approach.

In the ramped-temperature approach the temperature during the stress shall be switched periodically between the low and the high temperatures reported in Table 8.3, so that half of the test is at low and at high temperature, respectively. The ramp timing shall be such that no more than 25% of the stress is performed at intermediate temperatures during the transition between the two limit temperatures. As for the temperature switching frequency, no more than 10% of the endurance stress can be performed within any single half-cycle.

In the split approach, the sample is divided in two groups. The former undergoes endurance testing at a fixed low temperature, the latter at a fixed high temperature. The two temperature ranges are the same as for the ramped approach (see Table 8.3).

The T_{max} values are chosen so that the endurance stress time would be equivalent to 1 year at the active-use temperature and hours/day shown in Table 8.2 assuming an activation energy of 1.1 eV. In fact, although an SSD would be expected to reach its TBW rating over a lifetime of several years, for the specific purpose of calculating T_{max}, the full TBW is assumed to occur within a single year. This is a conservative assumption, since a shorter time allows less relaxation between writes.

In addition, the endurance stress T_{max} values may also account for a realistic amount of delay for relaxation which would occur if the stress temperature were too high. These delays, consisting of the drive being powered down or being powered up but not being written to, combined with the effect of the elevated temperature endurance stressing, must stay within the 1-year equivalent time.

The temperature T_{max} as well the additional delay time and temperatures may be extracted by solving

$$t_D e^{-\frac{E_a}{KT_D}} + t_s \left[FH_S e^{-\frac{E_a}{KT_{SH}}} + (1 - FH_S)\, e^{-\frac{E_a}{KT_{SL}}} \right] = t_U \left[FH_U e^{-\frac{E_a}{KT_{UH}}} + (1 - FH_U)\, e^{-\frac{E_a}{KT_{UL}}} \right] \tag{8.7}$$

where t_D, t_S, t_U are the delay time, the stress time and the use time, respectively; T_D is the temperature applied during the delay; T_{SH} and T_{SL} are the high and the low temperatures during the endurance stress in °K, respectively; T_{UH} and T_{UL} are the high and the low temperatures during the use conditions in °K, respectively; FH_S and FH_U are the fraction of time spent at high temperature during endurance stressing and use condition, respectively; K is the Boltzmann's constant equal to $8.6171 \cdot 10^{-5}$ eV/°K while E_a is the activation energy equal to 1.1 eV.

For example, consider the client application class from Table 8.2, $T_{UH} = 40°C = 313.15°K$; $T_{UL} = 30°C = 303.15$ °K; $FH_U = 1/3$ (8 h/day); a 1,000 h stress time using the ramped approach ($t_s = 1{,}000$ h, $T_{SL} = 25°C = 298.15°K$ and $FH_S = 1/2$) and no additional delays ($t_d = 0$). From Eq. (8.7) it possible to derive the endurance stress high temperature, considering that 1 year of normal use corresponds to $t_U = 8{,}766$ h:

$$\begin{aligned}
&t_s\left[\frac{1}{2}e^{-\frac{E_a}{KT_{SH}}}+\frac{1}{2}e^{-\frac{E_a}{KT_{SL}}}\right]=t_U\left[\frac{1}{3}e^{-\frac{E_a}{KT_{UH}}}+\frac{2}{3}e^{-\frac{E_a}{KT_{UL}}}\right]\\
&e^{-\frac{E_a}{KT_{SH}}}=\frac{2t_U}{t_S}\left[\frac{1}{3}e^{-\frac{E_a}{KT_{UH}}}+\frac{2}{3}e^{-\frac{E_a}{KT_{UL}}}\right]-e^{-\frac{E_a}{KT_{SL}}}\\
T_{SH}&=-\frac{E_a}{K}\frac{1}{ln\left[\frac{2t_U}{t_S}\left(\frac{1}{3}e^{-\frac{E_a}{KT_{UH}}}+\frac{2}{3}e^{-\frac{E_a}{KT_{UL}}}\right)-e^{-\frac{E_a}{KT_{SL}}}\right]}\\
T_{SH}&=-\frac{1.1}{8.6171\cdot 10^{-5}}\frac{1}{ln\left[\frac{2\cdot 8766}{1000}\left(\frac{1}{3}1.978\cdot 10^{-18}+\frac{2}{3}5.156\cdot 10^{-19}\right)-2.544\cdot 10^{-19}\right]}\\
&T_{SH}=330.76^oK=57.61^oC
\end{aligned} \tag{8.8}$$

Hence, the maximum temperature T_{max} for a 1,000-h stress, for the client application class, ramped approach, no delays is 58°C.

If it is chosen to perform the test at 50°C instead of 58°C, it is possible to add an additional delay, whose duration and temperature can also be derived from Eq. (8.7) by imposing $T_{SH} = 50°C = 323.15°K$.

$$\begin{aligned}
t_D e^{-\frac{E_a}{KT_D}} &= t_U\left[FH_U e^{-\frac{E_a}{KT_{UH}}}+(1-FH_U)\,e^{-\frac{E_a}{KT_{UL}}}\right]\\
&\quad -t_s\left[FH_S e^{-\frac{E_a}{KT_{SH}}}+(1-FH_S)\,e^{-\frac{E_a}{KT_{SL}}}\right]
\end{aligned} \tag{8.9}$$

For example, if a 100-h delay is added to the 1,000-h endurance stress, a $T_D = 67°C$ can be directly calculated as in Eq. (8.10):

$$e^{-\frac{E_a}{KT_D}}=\frac{8,766}{100}\left[\frac{1}{3}e^{-\frac{E_a}{313.15k}}+\frac{2}{3}e^{-\frac{E_a}{303.15k}}\right]-\frac{1,000}{100}\left[\frac{1}{2}e^{-\frac{E_a}{323.15k}}+\frac{1}{2}e^{-\frac{E_a}{298.15k}}\right] \tag{8.10}$$

Therefore, the endurance test would consist of 1,000 h of active endurance stress with the temperature ramped between 25°C and 50°C, with an additional 100 h spent in a non-writing mode at a temperature not greater than 67°C.

Table 8.4 Retention stress temperatures and times

Application class	Client	Enterprise
Stress duration and temperature	96 h @ T $\geq$ 66°C or 500 h @ T $\geq$ 52°C	96 h @ T $\geq$ 66°C or 500 h @ T $\geq$ 52°C

8.6.3.3 Retention Stress

After the endurance stress, SSDs are to be powered down and baked at elevated temperatures in order to establish the data retention capability. For the ramped-temperature approach, all drives in the sample are to be baked while, for the split approach, only the drives stressed at high temperatures are to baked. The SSDs are to be fully written with data prior to the bake and fully read after the test with internal error correction bypassed. The number of data errors resulting from the retention stress is to be added to that resulting from the endurance stress.

The temperatures required for the retention verification are reported in Table 8.4.

Two equivalent options are given for the bake temperature and durations and they are chosen to correspond to the required data retention times for the common temperature-accelerated mechanism responsible for data degradation in non-volatile memories, assuming an activation energy of 1.1 eV.

Not all mechanisms responsible for data loss, however, are accelerated by temperature and therefore a second evaluation is required at room temperature. This requirement holds only for the first product in a qualification family to be qualified; subsequent products are exempt. In the ramped-temperature approach the low temperature retention qualification is performed before the high temperature stress: in the split-flow approach, only the drives stressed at low temperatures undergo the low temperature retention test. Since time acceleration via higher temperatures is impossible, the room-temperature retention evaluation requires mathematical extrapolations based on drive-level or component-level bit-error-rate data.

When basing the extrapolation on drive-level bit-error-rate data, low-temperature retention tests require at least 500 h at a temperature between 10°C and 30°C. The bit error rate can be measured at several times (for instance, 48, 168 and 500 h) and then the trend can be extrapolated. The fraction of error bits with respect to the total bit number, called the Raw Bit Error Rate (*RBER*), depends on the program/erase cycle count and the retention time. For $BER < <1$,

$$RBER = RBER_0 + B_0 \cdot t^m \tag{8.11}$$

where $RBER_0$ is the Bit Error Rate at the beginning of the retention period, B_0 is an arbitrary scale factor dependent on materials and processes, t is the retention time and m is a retention power low coefficient (typically 1 or 2).

To verify the useful retention lifetime, the *RBER* can be measured as a function of time and the parameters $RBER_0$, B_0 and m fit Eq. (8.11). The resulting fitted equation may then be used to estimate the *RBER* at the desired retention time of

Table 8.5 Calculated RBER (example)

Retention time (h)	RBER
0	0
48	$5.63 \cdot 10^{-8}$
168	$2.23 \cdot 10^{-7}$
500	$7.41 \cdot 10^{-7}$
1,000	$1.59 \cdot 10^{-6}$

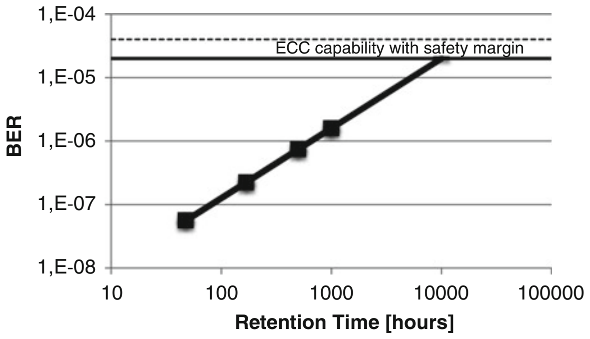


Fig. 8.15 Example of the verification of the retention requirements via extrapolation of the drive-level bit error rate

Table 8.2 and such a value must be below the ECC capability of the SSD controller, also considering a safety margin between the calculated ECC capability and the *RBER*.

Consider for example an SSD with a calculated ECC capability of $4 \cdot 10^{-5}$ and a safety margin equal to 2. Also consider that the *RBER* data of Table 8.5 have been obtained:

The extrapolated *RBER* at $t = 8{,}776$ h (= 1 year) must be below the ECC capability with safety margin whose value is $2 \cdot 10^{-5}$. As it can be seen in Fig. 8.15, the extrapolated *RBER* reaches the ECC capability with safety margin after 10,000 h and, therefore, the retention requirement of Table 8.2 is met.

The mathematical extrapolation can also be performed using raw bit error rate data from nonvolatile memory components, if available: at the end of the endurance stress, the room-retention evaluation can be derived by using the retention data calculated for the nonvolatile memory components inside the SSD for the specific number of program/erase cycles experienced during the extrapolation test.

8.6.4 Extrapolation Method

If the direct method would require more than 1,000 h of endurance stress, an extrapolation method can be used. Some of the proposed methods require special access to SSD internal operation or to nonvolatile memory components information which make these methods possible only for the SSD manufacturer.

Independently of the extrapolation method used for endurance and retention verification, some general requirements are to be ensured:

- the SSD must meet the requirements of Table 8.2 for *FFR* and *UBER,* for the temperatures and times stated in the Table;
- the *FFR* and *UBER* requirements must be met for both low-temperature and high-temperature endurance stressing, with temperature ranges of at least 25°C to 40°C for client SSD and 25°C to 60°C for enterprise SSD;
- data retention is to be verified under the assumption that the endurance stressing in use takes place over no longer than 1 year at the endurance use temperature and hours per day of Table 8.2;
- data retention is to be verified both for a temperature-accelerated mechanism (assuming an activation energy of 1.1 eV) and a non-temperature-accelerated mechanism;
- all requirements are to be established at a 60% statistical confidence level.

8.6.4.1 Accelerated Write Rate Through Modified Workload

With this method, the workload in the endurance test can be modified so that more program/erase cycles are performed on the nonvolatile memories for a given amount of time. In this framework, an SSD is considered to have been stressed to its full endurance rating regardless of the actual TB written if the nonvolatile memory experiences as many program/erase cycles as expected with the standard workload at the actual *TBW* rating.

Example of modified workload are:

- a workload with different ratio of sequential to random writes;
- a workload with a different transfer size;
- a workload which includes proprietary instructions to the SSD to perform internal data transfers from one location to another, which result in writes that bypass the host;
- reduced number of reads, that must be, of course, considered in solving Eq. (8.3).

8.6.4.2 Extrapolation of FFR and Bad-Block Trends

For the endurance evaluation, an SSD may be stressed to only some fraction of the *TBW* rating. During the endurance stress, functional failures may occur, as well as a certain number of blocks marked as "bad". The increase in these two quantities may be plotted as a function of *TB* in a lognormal or Weibull plot and extrapolated to the *TBW* rating to obtain estimates of the final levels of *FFR* and bad blocks.

This extrapolation method is not acceptable for verifying that the *UBER* requirements are met, because *UBER* may have a highly steep dependence on *TBW* that makes extrapolations from low *TBW* data quite unreliable.

8.6.4.3 FFR and UBER Estimation from Reduced-Capacity SSDs

The capacity of an SSD may be artificially reduced so that some nonvolatile memory components or blocks are not written, while the remaining ones are written more extensively than would be the case of the full-capacity SSD. In this context, an SSD will be considered to have reached its endurance rating limit if the stressed fraction of the nonvolatile memory components reaches the target program/erase cycles.

For this approach to be used, the manufacturer must ensure that the method of capacity reduction does not significantly distort the normal internal operation of the SSD. Simply reducing the logical span of written data is generally not sufficient, since the SSD controller and firmware make use of the full nonvolatile memory capacity, if not instructed.

A variation of this method is to extend the nonvolatile memory program/erase cycles beyond the target expected at the *TWB* rating, in order to generate functional failures and data errors. The resulting data can then be plotted and *FFR* and *UBER* can be extrapolated for the expected, lower, *TWB* rating.

8.7 Evaluating SSD Reliability Versus HDD Reliability

In terms of mechanical reliability, conventional HDDs pale when compared to SSDs. The absence of mechanical arms and spinning platters is the reason behind their improved reliability. In demanding environments, SSDs provide the type of ruggedness required for mobile applications. Unlike the HDDs, SSDs can withstand extreme shock and vibration with data integrity and without any danger of data loss. This feature is very important in industrial applications where exposure to highly combustible materials and electromagnetic radiation are typical. Their ability to deliver unnerving performance in extreme conditions also makes SSD play a vital role in military operations, be it in defense, aerospace or aviation applications. Military applications require, in most cases, an operating temperature range of −60°C to +95°C. On the contrary, shock, vibration, and temperature ratings of HDDs hardly comply with military standards.

In addition, SSDs also consume much less power than traditional HDDs. No additional power is required to activate the platters or the mechanical arms present in most HDDs. Their power consumption is practically only a fraction of that of a hard disk drive. In addition, Flash-based SSDs are associated to a significantly lower heat dissipation, because of the absence of the heat generated by the rotating/movable media.

The reduced power requirements and heat dissipation allow reducing the overall power supply in an electronic system and also allow getting rid of large cooling fans, thus contributing to an increased system reliability.

References

1. F. Masuoka, M. Momodomi, Y. Iwata, R. Shirota, New ultra high density EPROM and Flash EPROM cell with NAND structure, in *IEEE Tech. Dig.*, 1987, pp. 552–555
2. R. Micheloni, L. Crippa, A. Marelli, *Inside NAND Flash Memories* (Springer, New York, 2010)
3. M. Lenzlinger, E.H. Snow, Fowler-Nordheim tunneling into thermally grown SiO_2. J. Appl. Phys. **40**, 273–283 (1969)
4. Y.B. Park, D.K. Schroeder, Degradation of thin tunnel gate oxide under constant Fowler-Nordheim current stress for a Flash EEPROM. IEEE Trans. Electron Devices **45**, 1361–1368 (1998)
5. A. Modelli, A. Visconti, R. Bez, Advanced flash memory reliability, in *IEEE International Conference on Integrated Circuit Design and Technology*, Austin, Texas (USA), 2004, pp. 211–218
6. J.H. Lee et al., Using erase self-detrapped effect to eliminate the flash cell program/erase cycling Vth window close, in *Proceedings of the IRPS*, San Diego, California (USA), 1999, pp. 24–29
7. J.D. Lee et al., Degradation of tunnel oxide by FN current stress and its effects on data retention characteristics of 90 nm NAND FLASH memory, in *Proceedings of the IRPS*, Dallas, Texas (USA), 2003, pp. 497–501
8. D. Ielmini, A.S. Spinelli, A.L. Lacaita, Recent developments on Flash memory reliability, in *Microelectronic Engineering*, *14th biennial Conference on Insulating Films on Semiconductors*, Leuven, Belgium, vol. 80, 2005, pp. 321–328
9. A. Chimenton, C. Zambelli, P. Olivo, A statistical model of erratic behaviors in NAND flash memory arrays. IEEE Trans. Electron Devices **58**(November), 3707–3711 (2011)
10. R. Micheloni, A. Marelli, R. Ravasio, *Error Correction Codes for Non-Volatile Memories* (Springer, Dordrecht, 2008)
11. TN-29-42: Wear-Leveling Techniques in NAND Flash Devices, Micron Application Note
12. Datalight FX Pro Wear Leveling application note, http://www.datalight.com
13. Flash Solid-State Disk Reliability – Texas Memory Systems. Ramsan White Paper
14. SNIA Dictionary – white paper of Snia.org, http://www.snia.org/education/dictionary (2010)
15. Power Failure Protection – SMART Systems Application Note AN003
16. Standard JESD218A, *Solid-State Drive (SSD) Requirements and Endurance Test Method* (JEDEC Solid State Technology Association, Arlington, Feb 2011)
17. Standard JESD219, *Solid-State Drive (SSD) Endurance Workloads* (JEDEC Solid State Technology Association, Arlington, Sept 2010)

Chapter 9
Efficient Wear Leveling in NAND Flash Memory

Yuan-Hao Chang and Li-Pin Chang

Abstract In recent years, flash storage devices such as solid-state drives (SSDs) and flash cards have become a popular choice for the replacement of hard disk drives, especially in the applications of mobile computing devices and consumer electronics. However, the physical constraints of flash memory pose a lifetime limitation on these storage devices. New technologies for ultra-high density flash memory such as multilevel-cell (MLC) flash further degrade flash endurance and worsen this lifetime concern. As a result, flash storage devices may experience a unexpectedly short lifespan, especially when accessing these devices with high frequencies. In order to enhance the endurance of flash storage device, various wear leveling algorithms are proposed to evenly erase blocks of the flash memory so as to prevent wearing out any block excessively. In this chapter, various existing wear leveling algorithms are investigated to point out their design issues and potential problems. Based on this investigation, two efficient wear leveling algorithms (i.e., the evenness-aware algorithm and dual-pool algorithm) are presented to solve the problems of the existing algorithms with the considerations of the limited computing power and memory space in flash storage devices. The evenness-aware algorithm maintains a bit array to keep track of the distribution of block erases to prevent any cold data from staying in any block for a long period of time. The dual-pool algorithm maintains one hot pool and one cold pool to maintain the blocks that store hot data and cold data, respectively, and the excessively erased blocks in the hot pool are exchanged with the rarely erased blocks in the cold pool to prevent any block from being erased excessively. In this chapter, a series of explanations

Y.-H. Chang (✉)
Institute of Information Science, Academia Sinica, Taipei, Taiwan
e-mail: johnson@iis.sinica.edu.tw

L.-P. Chang
Department of Computer Science, National Chiao-Tung University, Hsinchu, Taiwan
e-mail: lpchang@cs.nctu.edu.tw

and analyses shows that these two wear leveling algorithms could evenly distribute block erases to the whole flash memory to enhance the endurance of flash memory.

9.1 Introduction

NAND flash memory has been widely adopted in various mobile embedded applications, due to its non-volatility, shock-resistance, low-power consumption, and low cost. It is widely adopted in various storage systems, and its applications have grown much beyond its original designs. The two popular NAND flash memory designs are single-level-cell (SLC) flash memory and multi-level-cell (MLC) flash memory. Each SLC flash-memory cell can accommodate 1-bit information while each MLC$_{\times n}$ flash-memory cell can contain n-bit information. As n increases, the endurance of each block in MLC flash memory decreases substantially.[1] In recent years, Well-known examples are flash-memory cache of hard drives (known as TurboMemory) [13, 40, 48], fast booting devices (for Microsoft Windows Visa), and solid-state disks (SSD) (for the replacement of hard drives).

As the low-cost MLC flash-memory designs are gaining market momentum [11], the endurance of flash memory is an even more challenging problem. For example, the endurance of an MLC$_{\times 2}$ flash-memory block is only 10,000 (or 5,000) erase cycles whereas that of its SLC flash memory counterpart is 100,000 erase cycles [35, 41]. As the number of bits of information per cell would keep increasing for MLC in the near future, the endurance of a block might also get worse, such as few thousand or even hundred erase cycles. This underlines the endurance issue of flash memory. However, improving endurance is problematic because flash-memory designs allow little compromise between system performance and cost, especially for low-cost flash storage devices. Such developments reveal the limitations of flash memory, especially in terms of endurance.

A NAND flash storage device or storage system, e.g., a solid-state disk (SSD) and flash cards, may be associated with multiple chips. Each chip is composed of one or more sub-chips or dies. Each sub-chip might have multiple planes. Each plane is organized in terms of blocks that are the basic unit for erase operations. A block is further divided into a fixed number of pages and can only endure limited erase cycles. A page (that is the unit of read and write operations) consists of a user area and a spare area, where the user area is for data storage, and the spare area stores housekeeping information such as the corresponding logical block addresses (LBAs), status flags, and error correction codes (ECCs). When a page is written with data, it is no longer available unless it is erased. This is called the “write-once property”. As a result, “out-place updates” are adopted so that data are usually updated over free pages. Pages that contain the latest copy of data (i.e., valid data) are considered as live (or valid) pages, and pages with old versions (i.e., invalid data) are dead (or

[1] In this chapter, we consider NAND flash memory, which is the most widely adopted flash memory in storage-system designs.

invalid) pages. Therefore, address translation is needed to map logical addresses of data to their physical addresses, and "garbage collection" is needed to reclaim dead pages. Because each block has a limited number of erase cycles, "wear-leveling" is needed to evenly erase blocks so as to prevent wearing out some blocks excessively.

Engineers and researchers have recently become concerned with how long flash storage devices can withstand daily use when they are adopted in applications with high access frequencies. The host systems, e.g., smart phones and notebooks, access their secondary storages (such as hard drives and SSDs) with temporal localities [33, 32, 46, 6]. Frequently updated data and rarely updated data coexist under such workloads. When reclaiming free space, block erases are always directed to the blocks with few valid data so as to reduce data-copy overheads. Thus, blocks having many static (or immutable) data are rarely chosen for erases, while other blocks are erased many times to circulate frequently updated data. As a result, some blocks are worn out when other blocks remain fresh. The problem of wearing out blocks is a crucial concern for new-generation flash memory, and *wear leveling* is the policy of evenly erasing all flash-memory blocks to keep all the blocks alive as long as possible. Strategies friendly to wear leveling can be adopted in various system layers, including applications, file systems, and firmware. To closely monitor wear in all blocks, the flash management strategies that are usually implemented as firmware implements wear leveling. However, wear leveling is not free, since extra data movement is required. Alleviating wear-leveling overheads is an important task, as wear leveling activities themselves wear flash memory too.

Many excellent wear leveling algorithms have been proposed by academia and industry. Updating data out of place is a simple wear-leveling technique [12, 38, 24, 20, 30]. However, this simple policy is vulnerable in the presence of static data because static data are rarely invalidated and need to be copied out before their residing blocks are erased. In order to reduce live-data-copying overhead, blocks storing a lot of static data rarely participate in the activities of reclaiming free space. Therefore, the key to wear leveling may be to encourage the blocks with static data to participate in block erases. Kim et al. [21] and Chiang et al. [9] proposed value functions for choosing victim blocks. In their approach, a block receives a high score if it currently has few valid data or its number of accumulated erase cycles is low. Another technique is to erase blocks in favor of reclaiming free space most of the time, but periodically, a block is erased in favor of wear leveling [25, 47]. A typical strategy is to occasionally erase a random block. Wear leveling activities can also be completely detached from free-space reclaiming. Hot-cold swapping [28, 6, 17, 10, 21] involves swapping data in a frequently erased block with that in an infrequently erased block whenever the wear of all blocks is unbalanced.

These existing approaches share a common idea: encouraging infrequently erased blocks to contribute to erases cycles. Under the workload of most real access patterns, most block erases are contributed by a small fraction of blocks if wear leveling is not used. According to such observations, static wear leveling algorithms are proposed to move static data away from infrequently erased blocks [7, 2, 43]. However, some existing static wear leveling algorithms don't consider the limited computing power or restricted RAM space, while some don't consider the access patterns and data access frequencies [18, 3, 39, 4, 42]. As a result, these existing static wear leveling algorithms either consume too many hardware resources or introduce too many overheads on extra live page copies and block erases. In order to achieve static wear leveling effectively with limited computing power, limited main memory, and limited overheads, two efficient wear leveling algorithms (i.e., the evenness-aware algorithm and dual-pool algorithm) are proposed and presented in this chapter. The *evenness-aware algorithm* [8] maintains a house-keeping data structure, i.e., a bit array, with a cyclic-queue-based scanning procedure to keep track of the distribution of block erases to prevent any static or cold data staying in any block for a long period of time. The objective is to improve the endurance of flash memory with limited overhead and without excessively modifying popular implementations of flash management designs, such as FTL, NFTL, and BL [1, 16, 45, 14]. The *dual-pool algorithm* [5] maintains one hot pool and one cold pool to maintain the blocks that store hot data and cold data, respectively, and the excessively erased blocks in the hot pool are exchanged with the rarely erased blocks in the cold pool to prevent any block being erased excessively. Whenever a block is excessively erased, it is filled with static data. In this way, such blocks stop participating in free-space reclaiming. This strategy helps conserve data movement because the major contributors of block erases are only a small fraction of all blocks. Second, blocks recently involved in wear leveling should be temporarily isolated from wear leveling activities. For example, after static data are written to a block which has been erased many times, the dual-pool algorithm decides how long this block should wait before it can contribute more erase cycles.

The rest of this chapter is organized as follows: Sect. 9.2 presents the evenness-aware algorithm with the worst-case analysis. In Sect. 9.3, the dual-pool algorithm is presented with a real case study. Section 9.4 concludes this chapter.

9.2 Evenness-Aware Algorithm

9.2.1 Algorithm Design

9.2.1.1 Overview

The motivation of the evenness-aware algorithm is to prevent static data from staying at any block for a long period of time. It minimizes the maximum erase-count difference between any two blocks, so flash memory lifetime is extended. This algorithm could be implemented as a module. In this algorithm, it maintains a *Block Erasing Table (BET)* that identifies the blocks erased during a given period of time (Sect. 9.2.1.2). The BET is associated with the process *SW Leveler* that is activated by some system parameters for the needs of static wear leveling (Sect. 9.2.1.3). When the SW Leveler runs, it either resets the BET or picks up a block that has not been erased so far (based on the BET information), and triggers the garbage

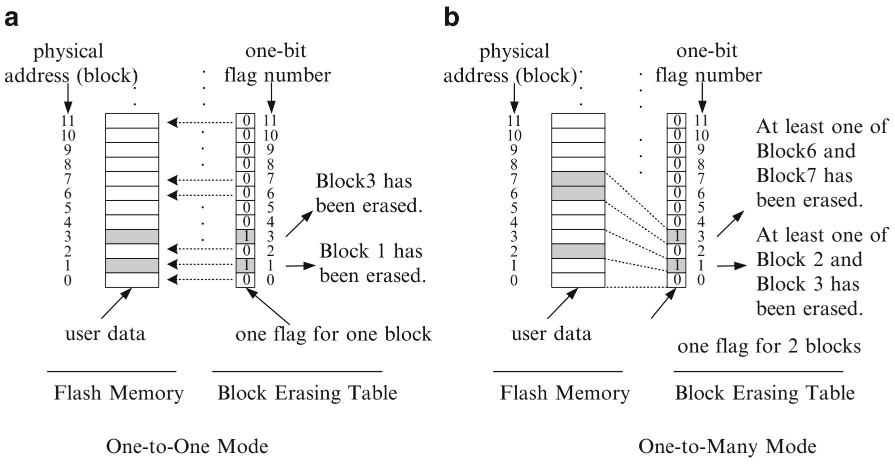


Fig. 9.1 The mapping mechanism between flags and blocks. (**a**) One-to-One Mode. (**b**) One-to-Many Mode

collector to do garbage collection on the block (note that the selection procedure of a block must be performed efficiently and within a limited time). Whenever a block is recycled by the garbage collection, any modification to the address translation is performed as in the original design of a flash management design. The SW Leveler can be implemented as a thread or as a procedure triggered by a timer or the garbage collector based on some preset conditions. Note that, whenever a block is erased, the BET must be updated by a triggering action to the SW Leveler. The design of the BET is scalable to accommodate rapidly increasing flash-memory capacity [34] and the limited RAM space on a controller.

9.2.1.2 Block Erasing Table

The Block Erasing Table (BET) attempts to remember which block has been erased in a pre-determined time frame, referred to as the *resetting interval*, so as to locate blocks of cold data. A BET is a bit array in which each bit corresponds to a set of 2^k contiguous blocks where k is an integer that equals or exceeds 0. Whenever a block is erased by the garbage collector, the SW Leveler is triggered to set the corresponding bit as 1. Initially, the BET is reset to 0 for every bit. As shown in Fig. 9.1, information maintenance is performed in one-to-one and one-to-many modes, and one flag is used to track whether any one of the corresponding 2^k blocks is erased. When $k = 0$, one flag is used for one block (i.e., in the one-to-one mode). The larger the value of k, the greater the chance in the overlooking of blocks of cold data. However, a large value for k could help reduce the RAM space required by a BET controller.

The worst case for a large k value occurs when hot and cold data co-exist in a block set. Fortunately, such a case is eventually resolved when hot data are invalidated. As a result, cold data could be moved to other blocks by the SW Leveler

(See Sect. 9.2.1.3). The technical problem relies on the tradeoff between the time to resolve such a case (bias in favor of a small k) and the available RAM space for the BET (bias in favor of a large k).

Another technical issue is efficiently rebuilding the BET when a flash-memory storage system is attached. One simple but effective solution is to save the BET in the flash-memory storage system when the system shuts down, and then to reload it from the system when needed. Meanwhile, the whole BET is stored in flash memory and loaded to main memory in an on-demand fashion, so that the required main memory could be minimized. If the system is not properly shut down, we propose loading any existing correct version of the BET when the system is attached. Such a solution is reasonable as long as loss of erase count information is not excessive. Note that the crash resistance of the BET information in the storage system could be provided by the popular dual buffer concept. Scanning of the spare areas of pages when collecting related information should also be avoided because of the potentially huge capacity of a flash-memory storage system.

9.2.1.3 SW Leveler

The SW Leveler consists of two procedures in executing wear leveling: SWL-Procedure and SWL-BETUpdate (Please see Algorithms 1 and 2). SWL-BETUpdate is invoked by the garbage collector to update the BET whenever any block is erased by the garbage collector during garbage collection. The SWL-Procedure is invoked whenever static wear leveling is needed. Such a need is tracked by two variables, f_{cnt} and e_{cnt}, which denote the number of 1s in the BET and the total number of block erases performed since the BET was reset, respectively. When the *unevenness level*, i.e., the ratio of e_{cnt} and f_{cnt}, equals or exceeds a given threshold T, SWL-Procedure is invoked to trigger the garbage collector to do garbage collection over selected blocks such that cold data are moved. Note that a high unevenness level reflects the fact that a lot of erases are done on a small portion of the flash memory.

Algorithm 1 shows the algorithm for the SWL-Procedure: the SWL-Procedure simply returns if the BET is just reset (Step 1). When the unevenness level, i.e., e_{cnt}/f_{cnt}, equals or exceeds a given threshold T, the garbage collector is invoked in each iteration to do garbage collection over a selected set of blocks (Steps 2–15). In each iteration, it is checked up if all of the flags in the BET are set as 1 (Step 3). If so, the BET is reset, and the corresponding variables (i.e., e_{cnt}, f_{cnt}, and f_{index}) are reset (Steps 4–7). The f_{index} is the index in the selection of a block set for static wear leveling and is reset to a randomly selected block set or to a predefined block set, e.g. 0. After the BET is reset, SWL-Procedure simply returns to start the next resetting interval (Step 8). Otherwise, the selection index, i.e., f_{index}, moves to the next block set with a zero-valued flag (Steps 10–12). *Note that the sequential scanning of blocks in the selection of block sets for static wear leveling is very effective in the implementation. We surmise that the design approximates that of an*

Algorithm 1: SWL-Procedure

```
   Input: $e_{cnt}$, $f_{cnt}$, $k$, $f_{index}$, $BET$, and $T$
   Output: $null$
1  if $f_{cnt} = 0$ then return;
2  while $e_{cnt}/f_{cnt} \geq T$ do
       /* size(BET) is the number of flags in the BET.                */
3      if $f_{cnt} \geq size(BET)$ then
4          $e_{cnt} \leftarrow 0$ ;
5          $f_{cnt} \leftarrow 0$ ;
6          $f_{index} \leftarrow RANDOM(0, size(BET) - 1)$ or 0;
7          reset all flags in the $BET$;
8          return;
9      end
10     while $BET[f_{index}] = 1$ do
11         $f_{index} \leftarrow (f_{index} + 1) \bmod size(BET)$
12     end
13     EraseBlockSet($f_{index}$, $k$) ;    /* Request the garbage collector to do garbage collection
       over the selected block set. */
14     $f_{index} \leftarrow (f_{index} + 1) \bmod size(BET)$ ;
15 end
```

Algorithm 2: SWL-BETUpdate

```
   Input: $e_{cnt}$, $f_{cnt}$, $k$, $b_{index}$, and $BET$
   Output: $e_{cnt}$, $f_{cnt}$ and $BET$ are updated based on the erased block address $b_{index}$ and $k$ in
           the BET mapping.
1  $e_{cnt} \leftarrow e_{cnt} + 1$ ;                       /* Increase the total erase count. */
   /* Update the $BET$ if needed.                                     */
2  if $BET[\lfloor b_{index}/2^k \rfloor] = 0$ then
3      $BET[\lfloor b_{index}/2^k \rfloor] \leftarrow 1$ ;
4      $f_{cnt} \leftarrow f_{cnt} + 1$ ;
5  end
```

actual random selection policy because cold data can virtually exist in any block in the physical address space of the flash memory. The SWL-Procedure then invokes the garbage collector to do garbage collection over a selected block set (Step 13) and moves to the next block set (Step 14) for the next iteration. We must point out that f_{cnt} and BET are updated by SWL-BETUpdate because SWL-BETUpdate is invoked by the garbage collector during garbage collection. The loop in static wear leveling ends when the unevenness level drops to a satisfactory value.

The SWL-BETUpdate is as shown in Algorithm 2: Given the address b_{index} of the block erased by the garbage collector, SWL-BETUpdate first increases the number of blocks erased in the resetting interval (Step 1). If the corresponding BET entry is not 1, then the entry is set as 1, and the number of 1s in the BET is increased by one (Steps 2–5). The remaining technical question is how to maintain the values of e_{cnt}, f_{cnt}, and f_{index}. To optimize static wear leveling, e_{cnt}, f_{cnt}, and f_{index} should

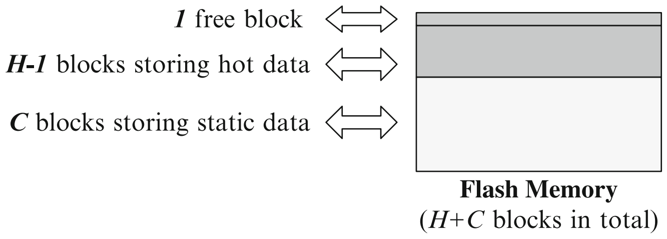

1 free block
H-1 blocks storing hot data
C blocks storing static data
Flash Memory
(*H*+*C* blocks in total)

Fig. 9.2 Flash memory of only static data and hot data

be saved to flash memory as system parameters and retrieved in the attachment of the flash memory. Notably, these values can tolerate some errors with minor modifications to SWL-Procedure in either the condition in Step 3 or the linear traversal of the BET (Steps 10–12). That is, if the system crashes before their values are saved to flash memory, it simply uses the values previously saved to flash memory.

9.2.2 *Worst-Case Analysis*

9.2.2.1 Worst-Case Model for Extra Overheads

Block recycling overhead is indeed increased by the proposed evenness-aware algorithm. A very minor cause of the increase is the execution of SWL-BETUpdate whenever the garbage collector erases a block, i.e., the value updates of e_{cnt} and f_{cnt} as well as the BET flags (compared to the block erase time, which could be about 1.5 ms over a 1 GB MLC$_{\times 2}$ flash memory [29]). As astute readers might point out, the garbage collector might be triggered more often than before because of wear leveling. That might increase the number of block erases and live-page copyings. The increased overheads caused by extra block erases and extra live-page copyings are apparent in the following worst-case scenario: the flash memory contains blocks of hot data, blocks of static data, and exactly one free block in a resetting interval.

Figure 9.2 shows the worst-case model based on a block-level address translation mechanism. In the block-level address translation mechanism, each LBA is divided into a virtual block address (VBA) and a block offset, and a mapping table is adopted for VBAs and their physical block addresses (PBAs). For each write operation, a free block is allocated to save the data of the remaining valid pages of the original mapped block and the new data of the write operation. Assume there are $(H - 1)$ blocks of hot data and C blocks of static data where the number of blocks in the system is $(H + C)$. The worst-case situation occurs when the C blocks are erased, only due to the evenness-aware algorithm. The worst case occurs when hot data are updated with the same frequency and only to the free block or the blocks of hot data, where $k = 0$. Sections 9.2.2.2 and 9.2.2.3 show the analyses for extra block erases and extra live-page copyings in the worst-case model, respectively.

Table 9.1 The increased ratio of block erases of a 1 GB MLC$_{\times 2}$ flash-memory storage system

H	C	H:C	T	Increased ratio (%)
256	3,840	1:15	10	9.46
2048	2,048	1:1	10	5.03
256	3,840	1:15	100	0.95
2,048	2,048	1:1	100	0.50
256	3,840	1:15	1,000	0.09
2,048	2,048	1:1	1,000	0.05

9.2.2.2 Extra Block Erases

When $k = 0$, the BET contains $(H + C)$ bits, i.e., $(H + C)$ 1-bit flags. In each resetting interval, when the updates of hot data result in $(T \times H)$ block erases, SWL-Procedure is activated to recycle one block of cold data for the first time because only H bits of the BET are set, and the unevenness level reaches T (i.e., $(T \times H)/H$). After one block of cold data is recycled by SWL-Procedure, $(H + 1)$ bits of the BET are set, and the number of block erases reaches $(T \times H + 1)$. The unevenness level (i.e., $(T \times H + 1)/(H + 1)$) is then smaller than the threshold T. Thereafter, SWL-Procedure is activated to recycle one block of cold data on all other $(T - 1)$ block erases resulting from hot data updates. Finally, this procedure is repeated C times such that all BET flags are set and the resetting interval ends. Therefore, the resetting interval has $T \times (H + C)$ block erases. For every $T \times (H + C)$ block erases in a resetting interval, SWL-Procedure performs C block erases. Therefore, the increased ratio of block erases (due to static wear leveling) is derived as follows:

$$\frac{C}{T \times (H + C) - C} \approx \frac{C}{T \times (H + C)}, \text{ when } T \times (H + C) \gg C.$$

The increased ratio is even worse when C is the dominant part of $(H + C)$ (an earlier study [18] showed that the amount of non-hot data is often several times that of hot data). Table 9.1 shows different increased ratios in extra block erasing for different configurations of H, C, and T. As shown in the table, the increased overhead ratio in extra block erasing is sensitive to the setting of T. Therefore, to avoid excessive triggering of static wear leveling, T must not be set too small.

9.2.2.3 Extra Live-Page Copyings

The extra overheads in live-page copyings due to the static wear leveling mechanism can be explored by the worst-case model. Let N be the number of pages in a block. Suppose that L is the average number of pages copied by the garbage collector when erasing a block of hot data. Thus, in the worst case, totally $(C \times N)$ live-pages are copied when erasing C blocks of static data (due to the evenness-aware algorithm) in a resetting interval, and $(T \times (H + C) - C) \times L$ live-page copyings are performed in the course of regular garbage collection activities in a resetting

Table 9.2 The increased ratio in live-page copyings of a 1 GB MLC$_{\times 2}$ flash-memory storage system

H	C	H:C	T	L	$\frac{N}{T \times L}$	Increased ratio (%)
256	3,840	1:15	10	16	0.800	75.72
2,048	2,048	1:1	10	16	0.800	40.02
256	3,840	1:15	10	32	0.400	37.86
2,048	2,048	1:1	10	32	0.400	20.00
256	3,840	1:15	100	16	0.0800	7.57
2,048	2,048	1:1	100	16	0.0800	4.00
256	3,840	1:15	100	32	0.0400	3.79
2,048	2,048	1:1	100	32	0.0400	2.00
256	3,840	1:15	1,000	16	0.0080	0.76
2,048	2,048	1:1	1,000	16	0.0080	0.40
256	3,840	1:15	1,000	32	0.0040	0.38
2,048	2,048	1:1	1,000	32	0.0040	0.20

interval. The increased ratio in live-page copyings, due to static wear leveling, can be derived as follows:

$$\frac{C \times N}{(T \times (H + C) - C) \times L} \approx \frac{C \times N}{T \times L \times (H + C)}, \text{ when } T \times (H + C) \gg C.$$

Table 9.2 shows varying increases in the ratios of live-page copyings for different configurations of H, C, T, and L, when $N = 128$. The increased ratio of live-page copyings can be estimated by $\frac{N}{L}$ times the increased ratio of extra block erases. For example, when $T = 100$, $L = 16$, $N = 128$, and $\frac{H}{C} = \frac{1}{15}$, the increased ratio of block erases is 0.95 % (the third row of Table 9.1) and its corresponding increased ratio of live-page copyings is 7.57 %, i.e., 0.95 % $\times \frac{128}{16}$ (the fifth row of Table 9.2). As shown in Tables 9.1 and 9.2, the increased ratios of block erases and live-page copyings would be limited with a proper selection of T and other parameters. The increased ratios could be limited to very small percentages of flash management strategies when the evenness-aware algorithm is supported.

9.3 Dual-Pool Algorithm

9.3.1 Algorithm Design

9.3.1.1 Algorithm Concept

This section introduces the basic concepts of the dual-pool algorithm. Let write requests arriving at the flash storage device be ordered by their arrival times. Let the temperature of a piece of data be inversely proportional to the number of requests between the two most recent writes to that data. A piece of data is *hot* if its temperature is higher than the average temperature of all data. Otherwise, the data is *cold* or *non-hot*. A block is referred to as a *young(/old) block* if its erase-cycle count is smaller(/larger) than the average erase-cycle count of all blocks.

We say that a block contributes or accumulates erase cycles if garbage collection erases this block to reclaim free space. Garbage collection avoids erasing a block having many valid data. If a block has more cold data than other blocks, then it will stop contributing erase cycles. This is because cold data remains valid in the block for a long time. Conversely, if a block has many hot data, then it can accumulate erase cycles faster than other blocks. This is because hot data are invalidated faster than cold data, and the block can become a victim of garbage collection before other blocks. After the block is erased, it can again be written with many hot data, because writes to hot data arrive more frequently than writes to cold data. Thus, this block is again erased and is written with many hot data.

The dual-pool algorithm monitors the erase-cycle count of each block. If an old block's erase-cycle count is larger than that of a young block by a predefined threshold, wear leveling activities are triggered. Cold data are moved to the old block to prevent it from being erased by garbage collection. This strategy is referred to as *cold-data migration*. After this, the old block should stop accumulating erase cycles. Compared to encouraging young blocks to contribute erase cycles, this strategy reduces data-movement overhead. This is because only a small fraction of blocks are worn into old blocks, while the majority are young blocks. Right after cold data are written to an old block, the old block still has a large erase-cycle count. If we are not aware that the old block has been involved in cold-data migration, we may again write some other cold data to the old block. This pointlessly reduces the block's lifetime. Similarly, after a young block is involved in cold-data migration, cold data previously stored in the block are removed. At this point, the young block has no cold data, even though its erase-cycle count is small. So, right after a block is involved in cold-data migration, it should be protected from immediate re-involvement. This strategy is called *block protection*. The protection of an old block is no longer required when other blocks become older than it. The protection of a young block expires when it is worn into an old block.

The access patterns from the host to the flash storage devices can change periodically. For example, a user application in the host may finish using some files and then begin accessing other files. These application-level behaviors can change the frequency with which a piece of data is updated, and thus cold data can change into hot data. Consider an old block written with cold data for cold-data migration. The old block is then protected against cold-data migration. Now suppose that the cold data in the old block happens to become hot. The protected old block will again start participating in garbage collection, and continues to age without interruption from wear leveling because its protection cannot expire. Now consider a young block under protection. The block should accumulate erase cycles. If the young block happens to be written with many cold data, then it stops contributing erase cycles. The young block draws no attention from wear leveling because its protection cannot expire. This dilemma highlights the special cases that must be carefully considered by block protection.

Table 9.3 A summary of symbols used in the dual-pool algorithm

Symbol	Definition
C	The cold pool, a collection of blocks
H	The hot pool, a collection of blocks
U	A collection of all blocks. $C \cap H = \emptyset$ and $C \cup H = U$
Q_P^w	A priority queue that sorts blocks in pool P in terms of information w
$M(Q_P^w)$	The element with the largest priority in Q_P^w
$m(Q_P^w)$	The element with the smallest priority in Q_P^w
$ec(b)$	The erase-cycle count of block b
$rec(b)$	The recent erase-cycle count of block b
TH	The threshold parameter for wear leveling

9.3.1.2 The Dual-Pool Algorithm: A Basic Form

The dual-pool algorithm, as implied by its name, uses a *hot pool* and a *cold pool*. A pool is merely a logical aggregation of blocks. Initially, a block arbitrarily joins one of these two pools. Note that the dual-pool algorithm is not to write cold data to blocks in the cold pool. Instead, it migrates blocks storing cold data to the cold pool.

The dual-pool algorithm uses priority queues to sort blocks in terms of different wearing information. The following section defines some symbols for ease of presentation: Let C and H denote the cold pool and the hot pool, respectively. Each element in C and H is a block. Let U be a collection of all blocks. $C \cap H = \emptyset$ and $C \cup H = U$ are invariants. Let Q_P^w be a priority queue that prioritizes all blocks in pool P in terms of wearing information w. The larger the value of w is, the higher the priority is. Each element in Q_P^w corresponds to a block. For block b, let function $ec(b)$ present its erase-cycle count. In priority queue Q_P^w, $M(Q_P^w)$ is the element with the highest priority and $m(Q_P^w)$ is the element with the lowest priority. $M(Q_P^w)$ and $m(Q_P^w)$ are referred to as the largest queue head and the smallest queue head, respectively. For example, $m(Q_C^{ec})$ denotes the block with the smallest erase-cycle count of all the blocks in the cold pool.

The dual-pool algorithm adopts a user-configurable parameter **TH** to direct how even the wear of blocks is to be pursued. The smaller the value of TH is, the more aggressive the wear-leveling activities would be. Table 9.3 summarizes the symbol definitions, and the following section defines cold-data migration (CDM for short): **Cold-Data Migration (CDM)**: Upon the completion of block erase, check the following condition:

$$ec(M(Q_H^{ec})) - ec(m(Q_C^{ec})) > TH.$$

If this condition is true, then the largest erase-cycle count of the blocks in the hot pool is larger than the smallest count of the blocks in the cold pool by TH. Perform the following procedure:

Step 1. Copy data from $m(Q_C^{ec})$ to $M(Q_H^{ec})$
Step 2. Erase $m(Q_C^{ec})$; $ec(m(Q_C^{ec})) \leftarrow ec(m(Q_C^{ec})) + 1$
Step 3. $C \leftarrow C \cup \{M(Q_H^{ec})\}$; $H \leftarrow H \backslash \{M(Q_H^{ec})\}$
Step 4. $H \leftarrow H \cup \{m(Q_C^{ec})\}$; $C \leftarrow C \backslash \{m(Q_C^{ec})\}$

Because cold-data migration checks the condition immediately after a block is erased, block $ec(M(Q_H^{ec}))$ must be the most-recently erased block if the condition is true. Whenever $ec(M(Q_H^{ec})) - ec(m(Q_C^{ec}))$ is found larger than TH, it is deduced that, on the one hand, block $m(Q_C^{ec})$ has not been erased for a long time because of the storing of many cold data. On the other hand, garbage collection had erased block $M(Q_H^{ec})$ many times, because this block infrequently stores cold data. Next, migrate cold data from block $m(Q_C^{ec})$ to block $M(Q_H^{ec})$. Step 1 moves data from block $m(Q_C^{ec})$ to block $M(Q_H^{ec})$ to complete cold-data migration. After this move, block $M(Q_H^{ec})$ can stop being erased by garbage collection. Step 2 erases block $m(Q_C^{ec})$ and increases the block's erase-cycle count. This erase does not affect the pool membership of block $m(Q_C^{ec})$.

Step 3 moves block $M(Q_H^{ec})$ to the cold pool, and Step 4 moves block $m(Q_C^{ec})$ to the hot pool. These steps swap the two blocks' pool memberships, and enable block protection. When the young block (previously $m(Q_C^{ec})$) joins the hot pool, it may be younger than many blocks in the hot pool. That is because most of the blocks in the hot pool are old. The young block is then protected, because cold-data migration is not interested in a young block in the hot pool. Analogously, when the old block (previously block $M(Q_H^{ec})$) migrates to the cold pool, it may be older than many blocks in the cold pool. The old block in the cold pool is then protected, as cold-data migration is concerned with the youngest block in the cold pool.

The young block in the hot pool (previously $m(Q_C^{ec})$) starts accumulating erase cycles. When the block is worn into the oldest in the hot pool, it will again participate in cold-data migration. On the other hand, the old block in the cold pool (previously $M(Q_H^{ec})$) now stops being erased. When the block becomes the youngest in the cold pool, it is again ready for cold-data migration.

9.3.1.3 Pool Adjustment

The cold pool collects blocks that store cold data. However, the cold pool may also contain blocks that have no cold data. This may be because all the blocks' pool memberships were arbitrarily decided in the very beginning, as all blocks' erase-cycle counts are initially zero. Another possible cause is that applications in the host may change their data-access behaviors. These changes can turn a piece of cold data into hot data.

Garbage collection selects erase victims based on how many invalid data a block has, regardless the block's pool membership. If a block has no cold data, it will continue participating in garbage collection even if it is in the cold pool. In this case, the block's erase-cycle count increases without interruption from wear leveling. This is because cold-data migration always involves the youngest block in the cold pool. Similarly, if a block in the hot pool has many cold data, garbage collection avoids erasing this block. The block cannot be erased into the oldest block in the hot pool, and cannot attract attention from wear leveling.

To deal with this problem, the dual-pool algorithm introduces two operations, cold-pool adjustment (CPA for short) and hot-pool adjustment (HPA for short).

Table 9.4 A summary of the five queue heads used by the dual-pool algorithm

Queue heads	Belongs to	Used in
$M(Q_H^{ec})$	The hot pool	Cold-data migration and hot-pool adjustment
$m(Q_H^{ec})$	The hot pool	Hot-pool adjustment
$m(Q_H^{rec})$	The hot pool	Cold-pool adjustment
$m(Q_C^{ec})$	The cold pool	Cold-data migration
$M(Q_C^{rec})$	The cold pool	Cold-pool adjustment

These two operations identify and correct any improper pool membership in the blocks. Specifically, blocks' pool membership is adjusted according to how frequently they have been erased since their last involvement in cold-data migration. Hot-pool adjustment removes the blocks that do not accumulate erase cycles from the hot pool. Cold-pool adjustment removes the blocks that actively contribute erase cycles from the cold pool. To enable these operations to function, new block-wearing information (i.e., the *recent erase-cycle count*) is introduced. A block's recent erase-cycle count is initially zero. It increases as along with the erase-cycle count, but reset to zero whenever the block is involved in cold-data migration. Thus, cold-data migration includes a new step:

(CDM) Step 5. $rec(M(Q_H^{ec})) \leftarrow 0; rec(m(Q_C^{ec})) \leftarrow 0$

The hot-pool adjustment and cold-pool adjustment operations also require new priority queues and queue heads, which are summarized in Table 9.4. Let function $rec()$ return the recent erase-cycle count of a block. The hot-pool adjustment and cold-pool adjustment are then as follows:

Cold-Pool Adjustment (CPA): Upon completion of block erase, check the following condition:

$$rec(M(Q_C^{rec})) - rec(m(Q_H^{rec})) > TH.$$

If it holds, then the largest recent erase-cycle count of the blocks in the cold pool is larger than the smallest count of the blocks in the hot pool by TH. Perform the following steps:

Step 1. $H \leftarrow H \cup \{M(Q_C^{rec})\}\ ; C \leftarrow C \setminus \{M(Q_C^{rec})\}$

If a block has a large recent erase-cycle count, then the block has contributed many erase cycles since the last time it was involved in cold-data migration. Cold-pool adjustment evicts such a block from the cold pool. This is because the last attempt to stop the block from being erased was not successful, or the block did not have cold data in the very beginning.

Hot-Pool Adjustment (HPA): Upon completion of block erase, check the following condition:

$$ec(M(Q_H^{ec})) - ec(m(Q_H^{ec})) > 2 \times TH.$$

If this condition holds, then in the hot pool the smallest erase-cycle count is smaller than the largest count by $2 \times TH$. Perform the following steps:

Step 1. $C \leftarrow C \cup \{\, m(Q_H^{ec})\}\ ; H \leftarrow H \setminus \{\, m(Q_H^{ec})\}$

Whether or not a block should be written with cold data for wear leveling depends on its erase-cycle count. If a block in the hot pool accumulates erase cycles more slowly than other blocks, then the block contains cold data, and the hot-pool adjustment operations removes this block from the hot pool. *Readers may question that why* $2 \times TH$ *is in this condition.* It is to prevent hot-pool adjustment from conflicting with cold-data migration: when cold-data migration moves a young block from the cold pool to the hot pool, the young block's erase-cycle count is already smaller than the oldest block in the hot pool by TH (see the condition for cold-data migration). To prevent hot-pool adjustment from immediately bouncing the young block back to the cold pool, the condition of hot-pool adjustment allows additional TH cycles ($2 \times TH$ in total).

In the worst case, every time after cold-data migration writes cold data to an old block and moves this block to the cold pool, the cold data become hot. Cold-pool adjustment can identify this old block and move it to the hot pool, after the block contributes TH more cycles of erase operations. Right after this, cold-data migration makes another attempt to write cold data to the block. So in this worst case, the dual-pool algorithm guarantees to involve this old block every other TH erase operations to this block.

9.3.1.4 Algorithm Demonstration

This section presents an example demonstrating how the dual-pool algorithm accomplishes wear leveling.

In Fig. 9.3, there are six flash-memory blocks, labeled from PBA 0 to PBA 5. The threshold parameter TH is 16. In the illustration, each block corresponds to two boxes, which indicate the block's erase-cycle count (ec) and recent erase-cycle count (rec). If a block currently stores cold data, then "C" appears under the block's boxes, and "H" otherwise. The example includes 11 steps. At each step, a block's boxes are shaded in gray if the block has been erased by garbage collection since the last step. A block's boxes are indicated black if it is currently involved in wear leveling. The following discussion refers to a block at PBA x as Block x, where x can be from 0 to 5.

In Step 1, the first three blocks join the hot pool and the rest join the cold pool. Step 2 shows that Blocks 0, 1, and 4 start accumulating erase cycles because they store no cold data. At this point, the largest erase-cycle count in the hot pool and the smallest erase-cycle count in the cold pool are 17 and 0, respectively. As this difference is greater than TH = 16, cold-data migration is triggered. Step 3 shows that the cold data in Block 3 are moved to Block 0, and the pool memberships are switched for both blocks. Notice that a block's wearing information sticks together with that block during cold-data migration. In Step 4, garbage collection erases Blocks 1, 3, and 4 because they had no cold data since Step 3.

Block 0, an old block previously involved in cold-data migration, is written with cold data and stops accumulating erase cycles since Step 3. Even though Block 0 is the oldest among all the blocks in the cold pool, it is now protected against cold-data

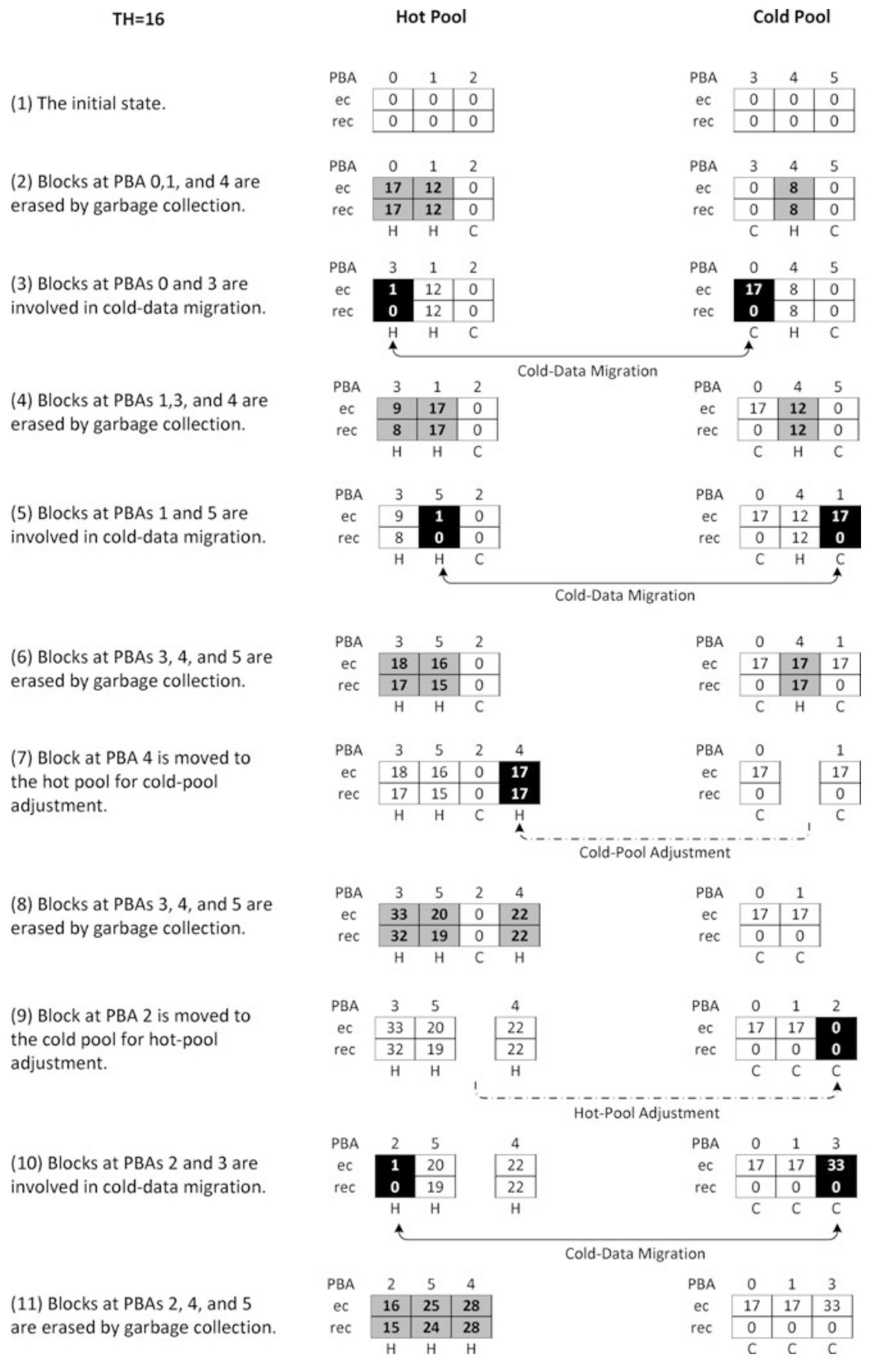

TH=16
Hot Pool
Cold Pool
(1) The initial state.
(2) Blocks at PBA 0,1, and 4 are erased by garbage collection.
(3) Blocks at PBAs 0 and 3 are involved in cold-data migration.
Cold-Data Migration
(4) Blocks at PBAs 1,3, and 4 are erased by garbage collection.
(5) Blocks at PBAs 1 and 5 are involved in cold-data migration.
Cold-Data Migration
(6) Blocks at PBAs 3, 4, and 5 are erased by garbage collection.
(7) Block at PBA 4 is moved to the hot pool for cold-pool adjustment.
Cold-Pool Adjustment
(8) Blocks at PBAs 3, 4, and 5 are erased by garbage collection.
(9) Block at PBA 2 is moved to the cold pool for hot-pool adjustment.
Hot-Pool Adjustment
(10) Blocks at PBAs 2 and 3 are involved in cold-data migration.
Cold-Data Migration
(11) Blocks at PBAs 2, 4, and 5 are erased by garbage collection.
PBA
ec
rec
H
C

Fig. 9.3 A scenario of the dual-pool algorithm. There are six flash-memory blocks, labeled from PBA 0 to PBA 5. Each block is associated with an erase-cycle count (ec), a recent erase-cycle count (rec), and the attribute of its data (hot or cold)

migration because it is not youngest in the cold pool. In Step 5, cold-data migration is triggered by Blocks 1 and 5, and cold data are migrated from Blocks 5 to 1. In Step 6, Blocks 3–5 contribute some more erase cycles since Step 5. Note that after two cold-data migrations, Blocks 0 and 1, which were previously the contributors of erase cycles in Step 2, now store cold data in the cold pool and are no longer being erased.

In Step 6, Block 4 in the cold pool stores no cold data. In Step 7, it is evicted from the cold pool by cold-pool adjustment, because the difference between Block 4s recent erase-cycle count and the smallest recent erase-cycle count in the hot pool (i.e., that of Block 2) is greater than $TH = 16$. In Step 8, Blocks 3–5 keep accumulating erase cycles, and have done so since Step 5. In Step 9, hot-pool adjustment is triggered because the difference between the erase-cycle counts of Blocks 2 and 3 is greater than $2 \times TH = 32$. Hot-pool adjustment moves Block 2 to the cold pool. Right after Step 9, cold-data migration for Blocks 2 and 3 occurs in Step 10. In Step 11, garbage collection erases some more blocks. At this point, the wear of all blocks is considered even, with respect to $TH = 16$.

9.3.2 Case Study: An SSD Implementation of the Dual-Pool Algorithm

9.3.2.1 The Firmware and Disk Emulation

The SSD platform in this study is the FreeScale M68KIT912UF32 development kit [15, 26]. This platform integrates an MC9S12UF32 SoC (referred to as the SSD controller hereafter), various flash-memory interfaces, and a USB interface. The controller contains a 16-bit MCU M68HCS12, 3 KB of RAM, 32 KB of EEPROM, a USB 2.0 interface controller, various flash-memory host controllers, and a DMA engine with an 1.5 KB buffer. The MCU is normally rated at 33 MHz. The NAND flash considered in this study is a 128 MB SmartMedia card (abbreviated as SM card hereafter). SM cards have the same appearance as bare NAND-flash chips in terms of physical characteristics. The block size and the page size of the SM card are 16 KB and 512 bytes, respectively, and it has a block endurance of 100 K erase cycles. Readers may notice that its geometry is finer than that of mainstream NAND flash memory [37]. However, the design and implementation of the proposed algorithm is independent of the block size and the page size.

An SSD presents itself to the host system as a logical disk,[2] so ordinary disk-based file systems (such as FAT and NTFS) are compatible with SSDs. The flash-translation layer (FTL), which is a part of SSD firmware, performs disk emulation [22,27,44,23]. Basically, FTL implements a mapping scheme, an update policy, and a garbage-collection policy. For ease of presentation, this section introduces some necessary terms and assumptions: Let a disk be addressed in terms of disk sectors,

[2]A logical disk is also referred to as a logical unit (i.e., LUN) [31].

each of which is as large as a flash-memory page. A *physical block* refers to a flash-memory block. Let the entire disk space be partitioned in terms of *logical blocks*, each of which is as large as a physical block. LBAs and PBAs are abbreviations of logical-block addresses and physical-block addresses, respectively. Let a *physical segment* be a group of contiguous physical blocks, and a *logical segment* be a group of contiguous logical blocks.

The FTL needs logical-to-physical translation because data in flash memory are updated out of place. However, a solid-state-disk controller cannot afford the space overhead of the RAM-resident data structures for this translation. To save RAM-space requirements, the FTL adopts a two-level mapping scheme. The fist level maps eight logical segments to eight physical segments. This first-level mapping has a one-to-one correspondence. The first level uses a RAM-resident segment translation table ("segment L2P table" for short). This table is indexed by logical-segment numbers, and each table entry represents a physical-segment number. As the first level maps a logical segment to a physical segment, the second level uses a RAM-resident block translation table ("block L2P table" for short) to map the 1,000 logical blocks in the logical block to the 1,024 physical blocks in the physical segment. This table is indexed by logical-block addresses and each table entry represents a physical-block address. Each physical segment has $1,024-1,000 = 24$ unmapped physical blocks, which are spare blocks for garbage collection and bad-block retirement. Thus, the SSD has a total volume of $8 * 1,000 = 8,000$ logical blocks, while the SM card has $8 * 1,024 = 8,192$ physical blocks.

The FTL sequentially writes all sectors of a logical block to the physical block mapped to this logical block, because the smallest granularity for address translation is one block. To translate an LBA into a PBA, first divide the LBA by 1,000. The quotient and the remainder are the logical-segment number and the logical-block offset, respectively. Looking up the segment L2P table and the block L2P table generates a physical segment number and a physical-block offset, respectively. The final PBA is calculated by adding the physical-block offset to the physical-segment number multiplied by 1,024.

For this FTL, there are two types of sector write operations: a write no larger than 4 KB (i.e., eight 512-byte disk sectors) and a write larger than 4 KB. A write larger than 4 KB effectively rewrites a logical block with the necessary copy-back operations: Unchanged sector data are copied from the logical block encompassing the written sectors, and combined with the newly written sector data. A spare block is allocated from the physical segment to which the logical block is mapped, and the combined data are then written to the spare block. The block L2P table is then revised to re-map the logical block to the spare block. The old physical block of the invalidated logical block is erased and converted to a spare block. Spare blocks are allocated in a FIFO fashion for fair use.

Writes no larger than 4 KB are handled in a different way. In this case, a separate spare block collects the newly written data. This spare block is referred to as a log block, as it can be seen as a log of small writes. Whenever the log block is full, the logical blocks modified by the writes recorded in the log block must be rewritten with copy-back operations to apply the changes. In this way, rewriting logical blocks

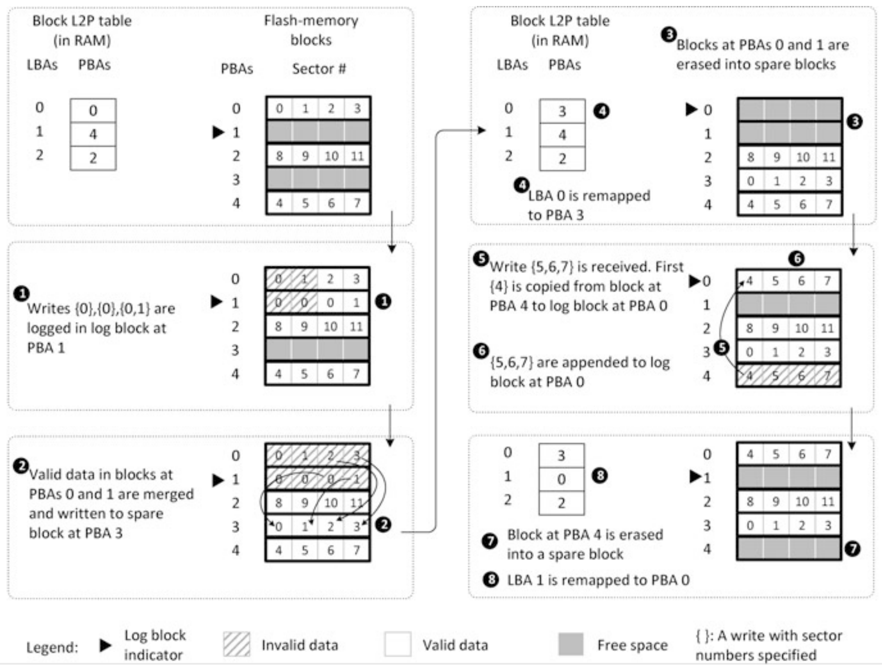

Block L2P table (in RAM)
Flash-memory blocks
LBAs PBAs
PBAs Sector #
Writes {0},{0},{0,1} are logged in log block at PBA 1
Valid data in blocks at PBAs 0 and 1 are merged and written to spare block at PBA 3
Blocks at PBAs 0 and 1 are erased into spare blocks
LBA 0 is remapped to PBA 3
Write {5,6,7} is received. First {4} is copied from block at PBA 4 to log block at PBA 0
{5,6,7} are appended to log block at PBA 0
Block at PBA 4 is erased into a spare block
LBA 1 is remapped to PBA 0
Legend: Log block indicator
Invalid data
Valid data
Free space
{ }: A write with sector numbers specified

Fig. 9.4 A scenario of our disk-emulation algorithm

is delayed until the log block is full. After rewriting all the involved logical blocks, the physical blocks previously mapped to the logical blocks and the spare blocks can be erased and converted to spare blocks. Note that the 4 KB threshold is an empirical setting, and this study provides no further discussion on it. erase and data copy activities for free-space reclaiming are referred to as *garbage collection*.

Figure 9.4 depicts a scenario of the proposed disk-emulation algorithm involving three logical blocks and five physical blocks. Let each physical block have four pages, and let each page be as large as a disk sector. A write is considered large if it is larger than two sectors. The left upper corner shows the initial state. Let a write be denoted by sector numbers enclosed within a pair of braces. Three small writes {0},{0}, and {0,1} arrive in turn. As they are small, they are appended to the free space in the log block at PBA 1 in Step 1. At this point, the log block is full. Step 2 then conducts copy-back operations to gather valid data from blocks at PBAs 0 and 1, and then rewrites the valid data to the block at PBA 3. Step 3 erases the blocks at PBAs 0 and 1. Step 4 revised the block L2P table. In Step 5, the fourth write {5,6,7} arrives. This write is large, and therefore requires that a logical block be rewritten. However, the unchanged data of Sector 4 are first copied from the block at PBA 4 to the log block at PBA 0. Step 6 then appends {5,6,7} to the log block, and Step 7 erases the block of invalid data. Step 8 then revises the block L2P table. Note that disk emulation is traditionally considered to be an issue independent of wear leveling. Refer to [19,23,22,44] for further discussion on disk-emulation algorithms.

The segment L2P table is small enough to be kept in RAM because it has only eight entries. There are eight block L2P tables, one for each pair of a logical segment and a physical segment. As mentioned above, since RAM space is very limited, only two block L2P tables can be cached in RAM. Whenever a block L2P table is needed but is absent from RAM, the least-recently used table in the cache is discarded. The needed table is then constructed by scanning all the physical blocks of the corresponding physical segment. This scanning involves only the spare areas of every physical block's first page, which contain the mapping information.[3]

9.3.2.2 Block-Wearing Information and Priority Queues

The dual-pool algorithm keeps track of every block's wearing information. This includes an erase-cycle count, a recent erase-cycle count, and pool membership. Ideally, this information should be kept in RAM for efficient access. However, this is not feasible because the SSD controller has only about 1 KB of RAM as working space.

One option is to write a block's wearing information in its spare areas [27]. In this approach, a block's wearing information must be committed to one of its spare areas immediately after the block is erased. Later on, when user data are written, error-correcting codes and mapping information are also written to these spare areas. However, this approach can overwrite a spare area multiple times. This is prohibited by many new NAND flash [37, 36]. One alternative is to exclusively write the wearing information to a spare area, but this spoils the existing data layout in spare areas for disk emulation.

Our approach is to reserve one physical block for writing the wearing information. An on-flash block-wearing information table ("**BWI table**" for short) keeps the blocks' wearing information. A new BWI-table can be written to an arbitrarily allocated spare block, which means that the BWI table is subject to wear leveling. Since the entire flash memory is divided into eight physical segments, each segment has its own BWI table. A BWI table contains 1,024 entries, one for each physical block. Each table entry has 4 bytes, including a 18-bit erase-cycle count, a 13-bit recent erase-cycle count, and 1 bit for pool membership. Note that 13 bits are large enough for a recent erase-cycle count because it is reset upon cold-data migration. A BWI table is $1,024 * 4 = 4$ KB large, so one 16-KB physical block can accommodate four revisions of a BWI table. If the block is full, another spare block is allocated for writing the BWI table, and the prior block is discarded for erase.

The on-flash BWI table can be entirely rewritten every time a block's wearing information changes. However, this method considerably increases write traffic to flash memory. Instead, the PBAs of the recently erased blocks are temporarily logged in a RAM buffer. In the current design, this buffer, named the erase-history

[3]The scanning is read-only and does not affect wear leveling. Previous research has developed excellent methods for reducing the time overhead of this scanning. Refer to [22, 19] for details.

table ("**EH table**" for short), has eight entries. If the EH table is full, a new version of the BWI table is written to the block reserved for the BWI table to apply the changes. After this, the in-RAM EH table is emptied.

Blocks are sorted in terms of different wearing information, and the dual-pool algorithm must check queue heads every time it is invoked. To scan the on-flash BWI table to find the queue heads is very slow. To reduce the frequency of BWI-table scanning, a small number of queue-head elements can be fetched for later use. For example, for fast access to $M(Q_H^{ec})$, after the BWI table is scanned, the wearing information of the two blocks with the two largest erase-cycles counts in the hot pool can be stored in RAM. An in-RAM queue-head table ("**QH table**" for short) is created for this purpose. The size of the QH table is fixed, and each of the five types of queue heads (shown in Table 9.4) is allocated to two table entries. A QH-table entry consists of a 2-byte PBA and 4-byte block-wearing information. Cold-data migration, hot-pool adjustment, and cold-pool adjustment check the QH table for queue heads. Wear leveling consumes QH-table entries and modifies the wearing information in the entries. A modified table entry is treated as an EH-table entry. The following section discusses when and how a QH table can be refreshed.

9.3.2.3 Segment Check-In/Check-Out

This section shows how the proposed wear-leveling data structures can be integrated into the segmented management scheme for disk emulation.

Disk emulation uses a two-level mapping scheme, as previously mentioned in Sect. 9.3.2.1. The segment L2P table is indexed by logical-segment numbers, has only eight entries, and is always stored in RAM. Second-level mapping manages the physical segments as if they were small pieces of flash memory. Each segment has an in-RAM L2P table, which maps 1,000 logical blocks to 1,024 physical blocks. Only two segments can have their block L2P tables cached in RAM. A segment is cached if its block L2P table is in RAM.

Each of the two cached segment uses an in-RAM EH table and an in-RAM QH table. Whenever a logical block is accessed, the corresponding physical segment is located by the segment L2P table. The dual-pool algorithm then checks if the segment's block L2P table, the EH table, and the QH table are in RAM. If they are absent, the following procedure, named *segment check-in*, is performed to bring them in: The in-RAM block L2P table is constructed by scanning the spare areas of each block's first page containing the mapping information. During scanning, if a block is found storing the on-flash BWI table, then the most up-to-date BWI table in the block is scanned to create the in-RAM QH table. By the end of this segment check-in procedure, the QH table and the block L2P table are ready. The in-RAM EH table is emptied, and the segment is all set for data access.

As the EH table continues to record the PBAs of erased blocks, sooner or later it will become be full. In this case, a new version of the on-flash BWI table should be created to merge the wearing information in the current on-flash BWI table, the in-RAM EH table, and the in-RAM QH table. The QH table is involved because

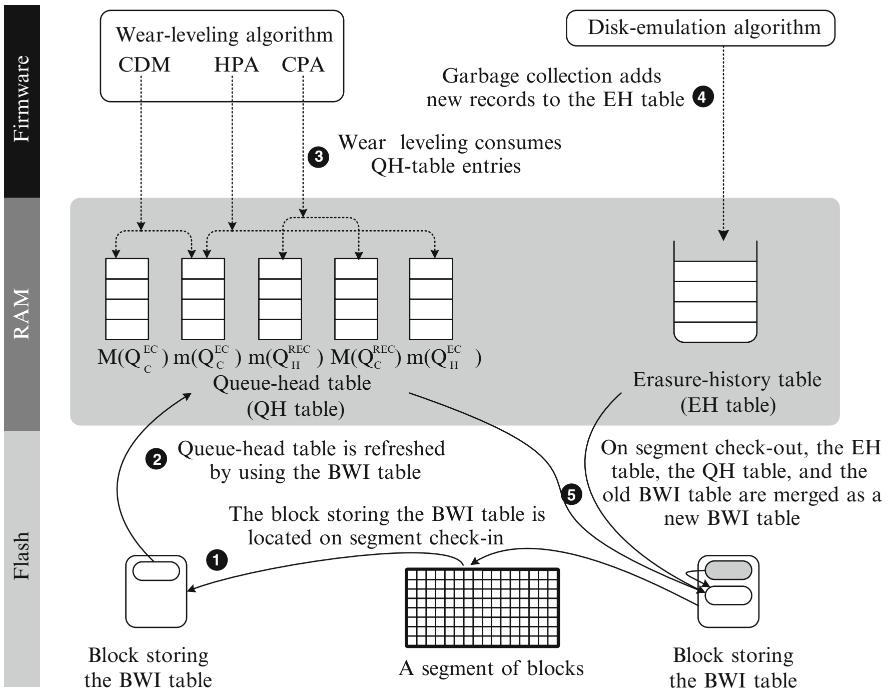

Firmware
RAM
Flash
Wear-leveling algorithm
CDM HPA CPA
Disk-emulation algorithm
Garbage collection adds new records to the EH table 4
3 Wear leveling consumes QH-table entries
$M(Q_C^{EC})$ $m(Q_C^{EC})$ $m(Q_H^{REC})$ $M(Q_C^{REC})$ $m(Q_H^{EC})$
Queue-head table (QH table)
Erasure-history table (EH table)
2 Queue-head table is refreshed by using the BWI table
5
On segment check-out, the EH table, the QH table, and the old BWI table are merged as a new BWI table
The block storing the BWI table is located on segment check-in
1
Block storing the BWI table
A segment of blocks
Block storing the BWI table

Fig. 9.5 Relationship between the in-RAM/on-flash data structures and how they are used by wear leveling, disk emulation, and segment operations

QH-table entries could have been switched to EH-table entries. This merging procedure, called the *BWI-table merge*, is as follows: First the block storing the current BWI table is located. The dual-pool algorithm creates a new BWI table in the same block right after the current BWI table. If there is no free space left, a new spare block is allocated. The four flash-memory pages storing the current BWI table are then copied to the new location. During copying a BWI-table page, the DMA engine first loads one of the four pages from flash memory into the DMA buffer, and then the dual-pool algorithm performs a three-way synchronization that involves the wearing information from the DMA buffer, the QH table, and the EH table. By the end of this merging procedure, the QH table is refreshed to contain new queue-head physical block addresses and their wearing information, and the EH table is emptied.

A segment's in-RAM data structures can also be evicted from RAM to accommodate those of a newly accessed segment. Before a segment vacates RAM space, its EH table and QH table must be merged with the on-flash BWI table. This process is called *segment check-out*. To check out a segment, the BWI-table merge procedure is first performed, and the in-RAM structures of the segment can then be discarded.

Figure 9.5 shows how by wear leveling, disk emulation, and segment operations use the proposed data structures. Step 1 shows that when a segment is checked in, the spare areas of the blocks in that segment are scanned to build the in-RAM block L2P table. This scanning process also locates the block storing the on-flash BWI table. Step 2 refreshes the in-RAM QH table of the segment with information in the on-flash BWI table. Step 3 shows that QH-table entries are consumed by wear leveling. If any block is erased by garbage collection, then a record of the erase is appended to the in-RAM EH table, as shown in Step 4. When the segment is checked out, Step 5 merges the information in the in-RAM QH table, in-RAM EH table, and on-flash old BWI table and writes it to a new BWI table on flash.

9.4 Conclusion

This work addresses a key endurance issue in the deployment of flash memory in various system designs. Unlike the wear leveling algorithms proposed in the previous work, two efficient wear leveling algorithms (i.e., the evenness-aware algorithm and dual-pool algorithm) are presented to solve the problems of the existing algorithms with the considerations of the limited computing power and memory space in flash storage devices. The evenness-aware algorithm proactively moves static or infrequently updated data with an efficient implementation and limited memory-space requirements so as to spread out the wear-leveling actions over the entire physical address space. It proposes an adjustable house-keeping data structure and an efficient wear leveling implementation based on cyclic queue scanning. Its goal is to improve the endurance of flash memory with only limited increases in overhead and without extensive modifications of popular implementation designs. The dual-pool algorithm is to protect a flash-memory block from being worn out if the block is already excessively erased. This goal is accomplished by moving rarely updated data to excessively erased blocks. Because the micro-controllers of flash storage devices are subject to very tight resource budgets, keeping track of wear levels for a large number of blocks is a very challenging task. The dual-pool algorithm keeps only the most frequently accessed data in RAM, while the rest is written to flash memory.

References

1. A. Ban, Flash file system, US Patent 5,404,485, in *M-Systems*, Apr 1995
2. A. Ban, Wear leveling of static areas in flash memory, US Patent 6732221 (2004)
3. A. Ban, R. Hasbaron, Wear leveling of static areas in flash memory, US Patent 6,732,221, in *M-systems*, May 2004
4. A. Ben-Aroya, S. Toledo, Competitive analysis of flash-memory algorithms, in *Proceedings of the 14th Conference on Annual European Symposium*, Zurich, Switzerland (2006)
5. L.-P. Chang, On efficient wear-leveling for large-scale flash-memory storage systems, in *22nd ACM Symposium on Applied Computing (ACM SAC)*, Seoul, Korea, Mar 2007

6. L.-P. Chang, T.-W. Kuo, Efficient management for large-scale flash-memory stroage systems with resource conservation. ACM Trans. Storage **1**(4), 381–418 (2005)
7. L.-P. Chang, T.-W. Kuo, S.-W. Lo, Real-time garbage collection for flash-memory storage systems of real-time embedded systems. ACM Trans. Embed. Comput. Syst. **3**(4), 837–863 (2004)
8. Y.-H. Chang, J.-W. Hsieh, T.-W. Kuo, Endurance enhancement of flash-memory storage systems: an efficient static wear leveling design, in *DAC'07: Proceedings of the 44th Annual Conference on Design Automation*, New York, NY, USA (ACM, 2007), pp. 212–217
9. M.-L. Chiang, P.C.H. Lee, R. chuan Chang, Using data clustering to improve cleaning performance for flash memory. Softw. Pract. Exp. **29**(3), 267–290 (1999)
10. R.J. Defouw, T. Nguyen, Method and system for improving usable life of memory devices using vector processing, US Patent 7139863 (2006)
11. DRAMeXchange: DRAM market-share games shifting from a knockout to a marathon; 4xnm process and multi-bit/cell as fundamental criteria to judge NAND Flash production competitiveness, DRAMeXchange (2008)
12. P. Estakhri, M. Assar, R. Alan Reid, B. Iman, Method of and architecture for controlling system data with automatic wear leveling in a semiconductor non-volatile mass storage memory. US Patent 5835935 (1998)
13. D. Pullen, Flash cache memory puts Robson in the middle. The Inquirer (2006)
14. M-Systems, Flash-memory translation layer for NAND flash (NFTL). M-Systems technical report (1998)
15. Freescale Semiconductor: USB Thumb Drive reference design DRM061, Freescale data sheets (2004)
16. Intel Corporation: FTL Logger Exchanging Data with FTL Systems. Intel technical report (1995)
17. C.J. Gonzalez, K.M. Conley, Automated wear leveling in non-volatile storage systems, US Patent 7120729 (2006)
18. SiliconSystems: Increasing flash solid state disk reliability. Technical report (2005)
19. J.-U. Kang, H. Jo, J.-S. Kim, J. Lee, A superblock-based flash translation layer for NAND flash memory, in *EMSOFT '06: Proceedings of the 6th ACM and IEEE International Conference on Embedded Software*, New York, NY, USA (ACM, 2006), pp. 161–170
20. Perdue, K.: Wear leveling. Spansion Application note AN01 (2008)
21. H.-J. Kim, S.-G. Lee, An effective flash memory manager for reliable flash memory space management. IEICE Trans. Inf. Syst. **85**(6), 950–964 (2002)
22. J. Kim, J.-M. Kim, S. Noh, S.-L. Min, Y. Cho, A Space-Efficient Flash Translation Layer for Compactflash Systems. IEEE Trans. Consum. Electron. **48**(2), 366–375 (2002)
23. S.-W. Lee, D.-J. Park, T.-S. Chung, D.-H. Lee, S. Park, H.-J. Song, A log buffer-based flash translation layer using fully-associative sector translation. Trans. Embed. Comput. Syst. **6**(3), 18 (2007)
24. Micron Technology: Wear-leveling techniques in NAND flash devices, Micron Application Note (2008)
25. Microsoft, Flash-memory abstraction layer (FAL), Windows Embedded CE 6.0 Source Code (2007)
26. Motorola: MC9S12UF32 System on a Chip Guide V01.04, Motorola data sheets (2002)
27. M-Systems: Flash-memory translation layer for NAND flash (NFTL), M-Systems technical note (1998)
28. M-Systems: TrufFFS wear-leveling mechanism, M-Systems Technical Note TN-DOC-017 (2002)
29. STMicroelectronics: NAND08Gx3C2A 8Gbit Multi-level NAND Flash Memory, STMicro-electronics data sheets (2005)
30. Numonyx: Wear Leveling in NAND Flash Memories, Numonyx Application Notes (2008)
31. Open NAND Flash Interface: Open NAND Flash Interface Specification (ONFi) Revision 2.1, ONFi specifications (2009)

32. C. Ruemmler, J. Wilkes, UNIX disk access patterns, in *Usenix Conference*, San Diego, USA (Winter 1993), pp. 405–420
33. D. Roselli, J.R. Lorch, T.E. Anderson, A comparison of file system workloads, in *Proceedings of the USENIX Annual Technical Conference*, San Diego, California, USA (2000), pp. 41–54
34. M. Rosenblum, J.K. Ousterhout, The design and implementation of a log-structured file system. ACM Trans. Comput. Syst. **10**(1), 26–52 (1992)
35. Samsung Electronics: K9F2808U0B 16M * 8 Bit NAND Flash Memory Data Sheet, Samsung Electronics Data sheets (2001)
36. Samsung Electronics Company: K9GAG08U0M 2G * 8 Bit MLC NAND Flash Memory Data Sheet, Samsung Electronics Data sheets (2005)
37. Samsung Electronics Company: K9NBG08U5M4Gb * 8 BitNANDFlashMemory Data Sheet, Samsung Electronics Data sheets (2005)
38. SanDisk Corporation: Sandisk Flash Memory Cards Wear Levelingm, Sandisk white papers (2003)
39. Shmidt, D.: Trueffs wear-leveling mechanism, M-Systems Technical report (tn-doc-017) (2002)
40. Intel Corporation: Software concerns of implementing a resident flash disk. Intel technical report (1998)
41. Spectek: NAND Flash Memory MLC 8Gb, Spectek data sheets (2003)
42. M. Spivak, S. Toledo, Storing a persistent transactional object heap on flash memory, in *LCTES '06: Proceedings of the 2006 ACM SIGPLAN/SIGBED Conference on Language, Compilers, and Tool Support for Embedded Systems*, Ottawa, Canada (2006), pp. 22–33
43. STMicroelectronics: Wear Leveling in Single Level Cell NAND Flash Memories, STMicroelectronics application notes (2006)
44. T.-S. Chung, D.-J. Park, S. Park, D.-H. Lee, S.-W. Lee, H.-J. Song, System software for flash memory: a survey, in *EUC '06: Embedded and Ubiquitous Computing*, Seoul, Korea (2006), pp. 394–404
45. Understanding the Flash Translation Layer (FTL) Specification. Technical report, Intel Corporation (Dec 1998), http://developer.intel.com/
46. W. Vogels, File system usage in windows nt 4.0. SIGOPS Oper. Syst. Rev. **33**(5), 93–109 (1999)
47. D. Woodhouse, Jffs: the journalling flash file system, in Proceedings of Ottawa Linux Symposium, Ottawa, Canada (2001)
48. M. Wu, W. Zwaenepoel, eNVy: a non-volatile main memory storage system, in *Proceedings of the Sixth International Conference on Architectural Support for Programming Languages and Operating Systems*, San Jose, USA (1994), pp. 86–97

Chapter 10
BCH for Solid-State-Drives

A. Marelli and R. Micheloni

Abstract Given that the NAND Flash memory is not a very reliable medium, it follows that a Solid State Disk needs some help to achieve a reliability suitable for computing applications: the *Error Correction Code* (ECC).

As the NAND technology scales down, ECC becomes a critical design topic. This chapter deals with BCH, the most common ECC in solid state disks.

Two main issues arise when an ECC is used inside an SSD. First of all, the ECC should not limit the bandwidth, being the bottleneck of the entire drive: this translates in a hardware implementation that needs to handle multiple devices (channel) in parallel. At the same time, ECC must avoid erroneous corrections when the error correction capability of the code is overcome, i.e. it must have a high detection property.

In this chapter the ECC definitions are reviewed, then the BCH code is presented with its detection property. Finally, the multi-channel topic is addressed.

10.1 Error Correction Codes Basic Definitions

In 1948 Claude Shannon's article "A Mathematical Theory of Communication" gave birth to the two twin disciplines: information theory and coding theory. The article specifies the meaning of efficient and reliable information and, there, the very well known term "bit" has been used for the first time [1]. Anyway, it was only with Richard Hamming in 1950 that a constructive generating method and the basic parameters of Error Correction Codes (ECC) were defined.

Hamming made his discovery at the Bell Telephone's laboratories during a study on communication on long telephone lines corrupted by lightening and crosstalk.

A. Marelli (✉) • R. Micheloni
Integrated Device Technology, Enterprise Computing Division, Agrate Brianza (MB), Italy
e-mail: alessiamarelli@gmail.com; rino.micheloni@ieee.org

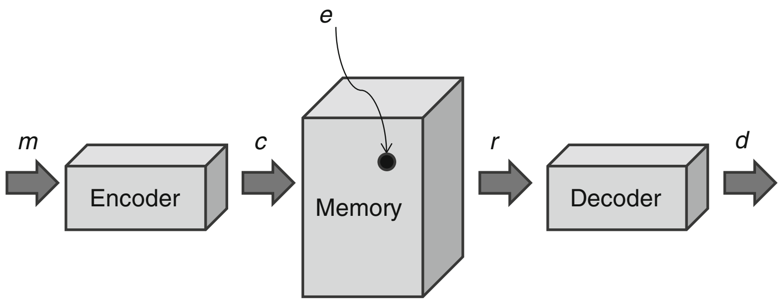


Fig. 10.1 Representation of coding and decoding operations for block codes

The discovery environment shows how the interest in error-correcting codes has taken shape, since the beginning, outside a purely mathematical field.

The codes discovered by Hamming are able to correct only one error, they are simple and widely used in several applications where the probability of error is small and the correction of a single error is considered sufficient.

More powerful codes, such as BCH and Reed-Solomon, were discovered between 1958 and 1960. The first ones were described by Bose and Chaudhuri [2] and through an independent study by Hocquengheim [3]; the second ones were defined by Reed and Solomon a few years later, between 1959 and 1960 [4]. They were immediately used in space missions, and today they are still used in compact discs.

Afterwards, they stopped being of interest for space missions and were replaced by convolutional codes, introduced for the first time by Elias in 1955. Convolutional codes can also be combined with cyclic codes. The study of optimum convolutional codes and the best decoding algorithms continued until 1993 when turbo codes were presented for the first time in the communication environment [5]. In fact, it is in the sector of telecommunications where they have received greater success.

A singular history is that of LDPC (Low Density Parity Checks) codes first discovered in 1962 by Gallager [6], but whose applications are being studied only today [7].

Error correction codes add redundant bits called parity bits to the information data bits so that, on reception, it is possible to detect the errors and to recover the message that has most probably been transmitted.

One of the biggest families in coding theory is block codes [8–10]. Block coding deals with messages of fixed length. Schematically (Fig. 10.1), a block m of k symbols is encoded in a block c of n symbols $(n > k)$ and written in a memory. Inside the memory, different sources may generate errors e, so that the block message r is read. The block r is then decoded in d by using the maximum likelihood decoding strategy, so that d is the message that has most probably been written.

A *Code C* is the set of codewords obtained by associating the q^k messages of length k of the space A to q^k words of length n of the space B in a univocal way.

A code is defined as *linear* if, given two codewords, also their sum is a codeword. When a code is linear, encoding and decoding can be described with matrix operations.

Definition 10.1.1 *G* is called *generator matrix of a code C* when all the codewords are obtainable as a combination of the rows of *G.*

Each code has more than one generator matrix, i.e. all its linear combinations. It follows that each code has infinite equivalent codes, i.e. all those obtained by permutations or linear combinations of the matrix *G*.

Definition 10.1.2 A set of equations that gives parity positions in terms of data positions is called *parity equations set*.

It is possible to express all these equations as a matrix. The matrix *H* is called *parity matrix* for a block code.

Therefore, with reference to Fig. 10.1, encoding a data message *m* consists in multiplying the message *m* by the code generator matrix *G*, according to Eq. (10.1).

$$c = m \cdot G \tag{10.1}$$

Definition 10.1.3 *G* is said in *standard form* or in *systematic form* if $G = (I_k, P)$, where I_k is the identity matrix $k \times k$ and *P* is a matrix $k \times (n-k)$. If *G* is in standard form then the first *k* symbols of a word are called information symbols.

Theorem 10.1.1 If a code $C[n,k]$ has a matrix $G - (I_k,P)$ in standard form, then a parity matrix of *C* is $H = (-P^T, I_{n-k})$ where P^T is the transpose of *P* and it is a matrix $(n-k) \times k$ and I_{n-k} is the identity matrix $(n-k) \times (n-k)$.

Systematic codes have the advantage that the data message is visible in the codeword and can be read before decoding. For codes in non – systematic form the message is no more recognizable in the encoded sequence and it is necessary to have the inverse encoding function to recognize the data sequence.

Definition 10.1.4 The code rate is defined as the ratio between the number of information bits and the codeword length. Given a linear code $[n,k]$ the ratio k/n is defined as *code efficiency*.

Definition 10.1.5 It is called *minimum distance* or *Hamming distance d* of a code, the minimum number of different symbols between any two codewords.

We can see that for a linear code the minimum distance is equivalent to the minimum distance between all the codewords and the codeword 0.

Definition 10.1.6 A code has *detection capability v* if it is able to recognize all the messages, containing *v* errors at the most, as corrupted.

The detection capability is related to the minimum distance as described in Eq. (10.2).

$$v = d - 1 \tag{10.2}$$

Definition 10.1.7 A code has *correction capability t* if it is able to correct each combination of a number of errors equal to *t* at the most. The correction capability is calculated from the minimum distance *d* by the relation:

$$t = \left[\frac{d-1}{2}\right] \tag{10.3}$$

where the square brackets mean the floor function.

Definition 10.1.8 Given a code *C[n,k]* A_i represents the number of codeword with weight *i*. The set $\{A_i\}$ is called weight distribution of the code *C* and $\{A_i\}$ are called the weights of *C*.

The code *C* has a symmetric distribution if Eq. (10.4) holds true.

$$A_i = A_{n-1} \quad 0 \le i \le \left[\frac{n-1}{2}\right] \tag{10.4}$$

Definition 10.1.9 The dual code C^* of a code *C[n,k]* is the set of vectors orthogonal to all the codewords of *C*:

$$C^* = \left\{ v \in \left(F_q\right)^n | v \cdot c = 0 \forall c \in C \right\} \tag{10.5}$$

The weight distribution or the distance of the dual code gives a lot of information on the code itself as the following sections show.

Definition 10.1.10 Given a code *C*, its dual code C^* and $\{A_i\}$ with $i = \{0, \ldots, n\}$ its weight distribution, we define the weight enumerator of the code *C* the polynomial

$$W_C(x, y) = \sum_{i=0}^{n} A_i x^{n-i} y^i \in Z[x, y] \tag{10.6}$$

A fundamental theorem that describe the relationship between $W_C(x,y)$ and $W_{C^*}(x,y)$ is MacWilliams theorem. Here it will be present only the version for linear binary codes.

Theorem 10.1.2 ***MacWilliams equality for binary codes*** Given a linear binary code *C[n,k]* and C^* its dual code the following equation holds true

$$W_{C^*}(x, y) = \frac{1}{|C|} W_C(x + y, x - y) \tag{10.7}$$

where $|C| = 2^k$ is the number of words in *C*. In other words:

$$\sum_{i=0}^{n} A^*{}_i x^{n-i} y^i = \frac{1}{|C|} \sum_{i=0}^{n} A_i (x + y)^{n-i} (x - y)^i \tag{10.8}$$

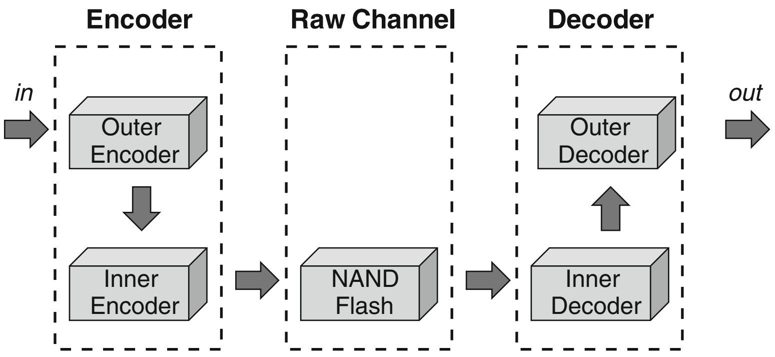


Fig. 10.2 Schematic diagram of code concatenation

This latter equality is called MacWilliams identity.

There is an important operation we can apply to a linear code C called extension. Also in this case, there is a relationship between the original code weights and the weights of its extension.

Definition 10.1.11 Given a code $C[n,k,d]$ we call extension of the code C_E the code obtained from C by adding one more parity bit computed as the logical XOR of all the other bits. The code C_E has only even weighted codeword and

– if d is even C_E is a code $[n+1,k,d]$
– if d is odd C_E is a code $[n+1,k,d+1]$

Given a code C with weight distribution $\{a_i\}$, and $\{A_i\}$ the weight distribution of its extension code C_E, we have

$$A_{2i} = a_{2i} + a_{2i-1} \tag{10.9}$$

with $2 \leq 2i \leq n\text{-}1$.

Given a code $C[n,k,d]$ and C_E its extension: if n is odd and C has a symmetrical weight distribution, then C_E has symmetrical weight distribution.

In a lot of applications there are external factors not subject to error check which determine the length permitted to an error correction code. Non volatile memories, for example, operate on codewords that have a length power of 2.

When the "natural" length of the code is not suitable it is possible to change it with the shortening operation.

Definition 10.1.12 A code $C[n,k]$ is *shortened* into a code $C'[n\text{-}j,k\text{-}j]$ by erasing j columns of the parity matrix.

Codes can be combined together in order to improve their correction capabilities. One way to combine them is with concatenation. In this operation we have an inner code (C_{IN}) and an outer code (C_{OUT}) that work together (Fig. 10.2).

Typically, C_{IN} is decoded with a maximum-likelihood approach and C_{OUT} is a block code of length *n*. In this way the concatenation combines the error probability property of the inner code and the decoding time property of the outer code.

As sketched in Fig. 10.2 the message is firstly encoded with the outer code and the resulting codeword is encoded with the inner code. During the decoding phase the message is decoded with the inner code and the result is then decoded with the outer code. From this description it is clear that a key feature is that the inner code must have a good detection property, since we must be sure that the inner code doesn't perform erroneous correction when the correction capability of the code is overcome.

10.2 BCH Codes

BCH codes belong to the family of cyclic codes. These are, perhaps, the most used codes in applications, since they can be implemented by using high-speed shift-register encoders and decoders [11–13].

Definition 10.2.1 A linear code *C[n,k]* is called *cyclic* if $(x_1,x_2,\ldots,x_n) \in C => (x_n,x_1,\ldots,x_{n-1}) \in C$.

In other words, if we write the vector $a(x) = (a_0,\ldots,a_{n-1})$ as the polynomial $a_0 + a_1x + a_2x^2 + \ldots + a_{n-1}x^{n-1}$, the previous definition states that, if $a(x) \in C$, then also the right shift belongs to *C*.

As seen in the previous section, the distance is a key feature in characterizing a code; in BCH codes the minimum distance can be ensured during construction.

Generally speaking, in order to know the minimum distance for a linear code with generator polynomial *g(x)*, it is necessary to compute the distance between all the possible codewords. BCH codes, by imposing some constraints on the generator polynomial, are able to ensure a "designed distance".

Definition 10.2.2 Let β be an element of $GF(q^m)$. Let *b* be a non-negative integer. A BCH code with "designed" distance *d* is generated by the polynomial *g(x)* of minimal degree that has *d-1* consecutive powers of β: $\beta^b, \beta^{b+1}, \ldots, \beta^{b+d-2}$ as roots. Given Ψ_i the minimal polynomial of β^{b+i} for $0 \leq i < d-1$, *g(x)* is computed as:

$$g(x) = LCM\{\psi_0(x), \psi_1(x), \ldots, \psi_{d-2}(x)\} \tag{10.10}$$

and the data protected by the code is $k = n\text{-}deg(g(x))$.

It is possible to show that the designed distance *d* is at least $2t + 1$, hence the code is able to correct *t* errors. The number of parity bits for a binary BCH code is less than or equal to *mt*. Generally, this number is equal to *mt*; it is less only when the minimum distance is greater than the designed distance the code is constructed with.

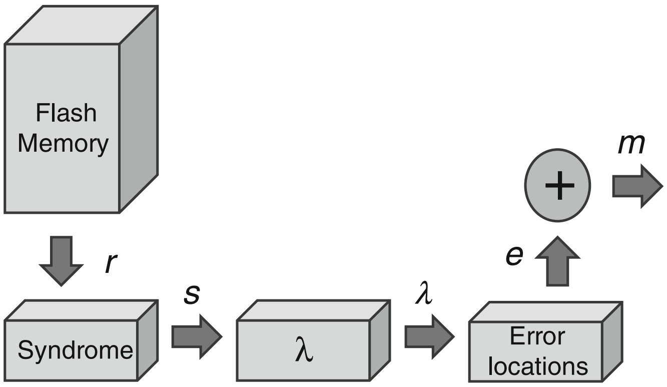


Fig. 10.3 Structure of a binary BCH decoder

If we assume $b = 1$, and β a primitive element of $GF(q^m)$ the code becomes a *narrow-sense* and *primitive* BCH code of length q^m-1 able to correct t errors. We shall now consider primitive BCH codes.

As regards the distance, the important result of Carlitz-Uchiyama inequality is proven.

Theorem 10.2.1 ***Carlitz-Uchiyama inequality*** Given a binary BCH code C of length $n = 2^m - 1$ with designed distance $\delta = 2t + 1$, for the minimum distance of the dual code C^* the following inequality holds true

$$d^* \geq 2^{m-1} - (t-1)\, 2^{\left[\frac{m}{2}\right]} \tag{10.11}$$

The general decoding structure is represented in Fig. 10.3.

In BCH structure there is only one step to encode a message, while there are three steps to decode a message. Generally, we can state that the decoding is ten times more complex than encoding.

The encoding of a systematic BCH code is performed by multiplying the message $m(x)$ by x^{n-k} and calculating the parity bits as the remainder of the division of this multiplication by the generator polynomial, in accordance with Eqs. (10.12) and (10.13).

$$\frac{m(x) \cdot x^{n-k}}{g(x)} = q(x) + \frac{r(x)}{g(x)} \tag{10.12}$$

$$c(x) = m(x) \cdot x^{n-k} + r(x) \tag{10.13}$$

The structure that implements this division is represented in Fig. 10.4.

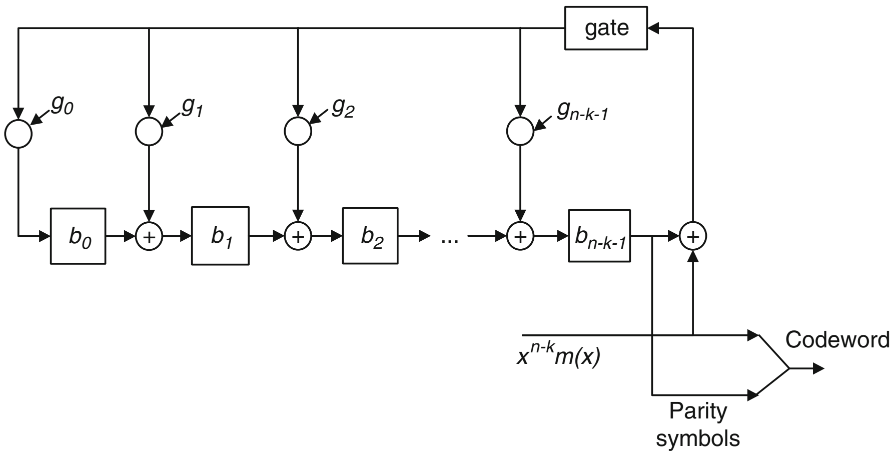


Fig. 10.4 Binary BCH encoder

The decoding operation follows three fundamental steps, as shown in Fig. 10.3.

- calculation of the syndromes;
- calculation of the coefficients of the error locator polynomial;
- calculation of the roots of the error locator polynomial.

Errors in the storage media can be represented by a polynomial that has coefficient 1 in correspondence with every error's position:

$$E(x) = E_0 + E_1 x + \ldots + E_{n-1} x^{n-1} \tag{10.14}$$

Observe that, in order for the code to be corrector of t errors, at most t non-null coefficients are allowed in Eq. (10.14). The read vector *R(x)* is therefore:

$$R(x) = c(x) + E(x) \tag{10.15}$$

The first decoding step consists in calculating the *2t* syndromes for the read message:

$$\frac{R(x)}{\psi_i(x)} = Q_i(x) + \frac{S_i(x)}{\psi_i(x)} \text{ with } 1 \le i \le 2t \tag{10.16}$$

$$S_i(x) = Q_i(x) \cdot \psi_i(x) + R(x) \text{ with } 1 \le i \le 2t \tag{10.17}$$

In accordance with Eqs. (10.16) and (10.17), the received vector is divided by each minimal polynomial Ψ_i forming the generator polynomial, thus getting a quotient $Q_i(x)$ and a remainder $S_i(x)$ called *syndrome*.

At this point the *2 t* syndromes must be evaluated into the elements $\beta, \beta^2, \beta^3, \ldots,$ β^{2t} whose Ψ_i are the minimal polynomials. With reference to Eq. (10.18), this evaluation is computed as the evaluation of the message received in β, β^2, $\beta^3, \ldots,$ β^{2t}, since $\Psi_i(\beta^i) = 0$ (for $1 \leq i \leq 2\,t$) by definition of minimal polynomial.

$$S_i\left(\beta^i\right) = S_i = Q_i\left(\beta^i\right) \cdot \psi_i\left(\beta^i\right) + R\left(\beta^i\right) = R\left(\beta^i\right) \tag{10.18}$$

Consequently, the *i*-th syndrome can be calculated either as the remainder of the division between the received message and the minimal polynomial Ψ_i, then evaluated in β^i, or as the evaluation in β^i of the received message.

Observe that, in case no errors occur, the polynomial received is a codeword: therefore the remainder of the division of Eq. (10.16) is null and all the syndromes are identically null. On the other hand, verifying if the syndromes are identically null is a necessary and sufficient condition to understand if the read message is a codeword or if some errors occurred.

An useful property described in Eq. (10.19) can be exploited to compute only *t* syndromes.

$$S_{2i} = S_i^2 \tag{10.19}$$

The syndromes calculation for a BCH code existing over $GF(2^m)$ involves *t* structures which contemporarily calculate the remainder of the divisions between the received polynomial and the *t* minimal polynomials. These structures are very similar to the one depicted in Fig. 10.4.

Once the syndromes are computed, they are used to search the error locator polynomial.

By indicating the error positions with *X* and the number of errors that occurred with *v* the following equality holds true:

$$S_i = \sum_{l=1}^{v} X_l^i \tag{10.20}$$

Definition 10.2.3 It is defined *error locator polynomial* $\Lambda(x)$ the polynomial whose roots are the inverse of the error positions.

From the definition we have:

$$\Lambda(x) = \prod_{i=1}^{v} (1 - xX_i) \tag{10.21}$$

Please observe that the degree of the error locator polynomial gives the number of errors that occurred. The degree of $\Lambda(x)$ is *t* at most, hence, in the case more than *t* errors occur, the polynomial $\Lambda(x)$ could erroneously indicate *t* or less errors.

The most used algebraic method to perform this step of the decoding is the Berlekamp-Massey algorithm [14]. The complexity of this algorithm grows in a linear way, enabling the construction of efficient decoders able to correct dozens of errors.

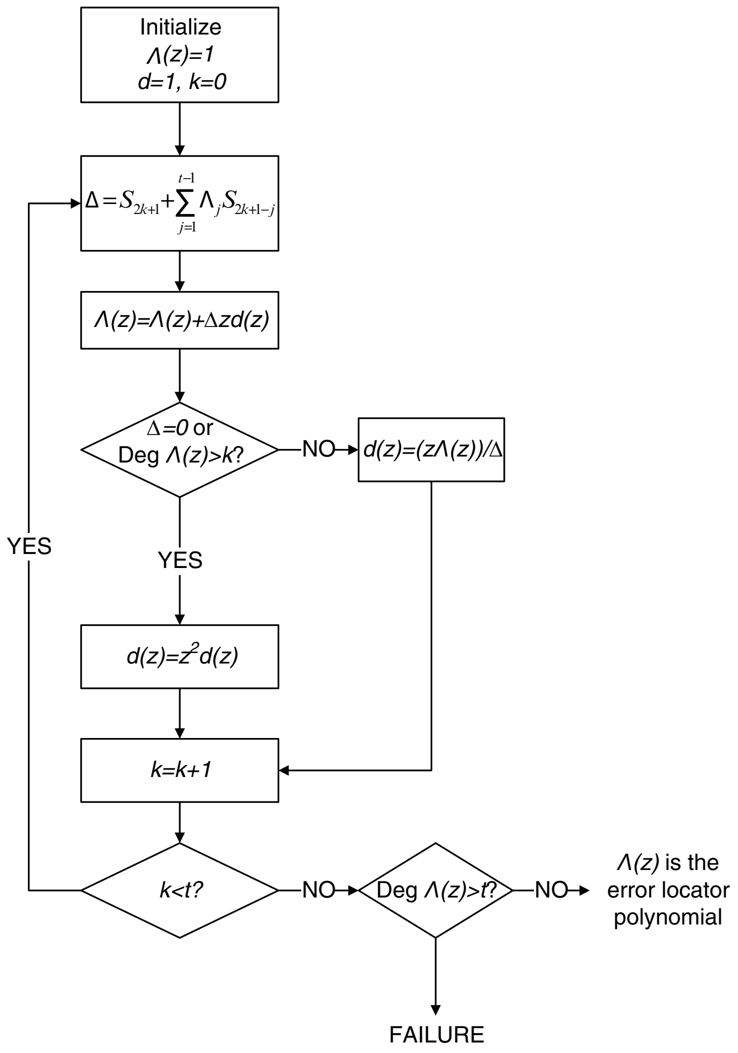

Initialize
$\Lambda(z)=1$
$d=1, k=0$
$\Delta = S_{2k+1} + \sum_{j=1}^{t-1} \Lambda_j S_{2k+1-j}$
$\Lambda(z)=\Lambda(z)+\Delta z d(z)$
$\Delta=0$ or Deg $\Lambda(z)>k$?
NO
$d(z)=(z\Lambda(z))/\Delta$
YES
$d(z)=z^2 d(z)$
$k=k+1$
YES
$k<t$?
NO
Deg $\Lambda(z)>t$?
NO
$\Lambda(z)$ is the error locator polynomial
FAILURE

Fig. 10.5 Flow diagram for the Berlekamp algorithm

Berlekamp algorithm finds the coefficients of the error locator polynomial in an iterative way. At the *i*-th step of the algorithm we find a polynomial $\Lambda(x)$ whose coefficients solve the first *i* equations of Eq. (10.20). Then, we test if $\Lambda(x)$ also solves the equation $i + 1$; if not, we calculate the discrepancy term *d* so that $\Lambda(x) + d$ solves the first $i + 1$ equations. After $2t$ iterations $\Lambda(x)$ is the error locator polynomial.

In the binary case it is possible to perform the Berlekamp algorithm in *t* iterations. There are a number of different implementations of Berlekamp algorithm [15–17], here below we will explain the one following the diagram of Fig. 10.5.

Equation (10.22) shows the syndrome polynomial and the initial conditions for the algorithm:

$$\begin{aligned} 1 + S &= 1 + S_1 z + S_2 z^2 + \ldots + S_{2t-1} z^{2t-1} \\ \Lambda^{(0)}(z) &= 1 \qquad d^{(0)} = 1 \end{aligned} \tag{10.22}$$

At the *i*-th step we proceed as follows:

- if S_{2i+1} is unknown the algorithm is finished;
- otherwise we define $\Delta^{(2i)}$ the coefficient of z^{2i+1} in the product $(1 + S(z))\Lambda^{(2i)}(z)$.

$$\Lambda^{(2i+2)}(z) = \Lambda^{(2i)}(z) + \Delta^{(2i)} \cdot d^{(2i)}(z) \cdot z \tag{10.23}$$

$$d^{(2i+2)}(z) = \begin{cases} z^2 d^{(2i)}(\mathrm{z}) & \text{if } \Delta^{(2i)} = 0 \text{ or if } \deg \Lambda^{(2i)}(z) > i \\ \dfrac{z\Lambda^{(2i)}(z)}{\Delta^{(2\mathrm{i})}} & \text{if } \Delta^{(2i)} \neq 0 \text{ or if } \deg \Lambda^{(2i)}(z) \leq i \end{cases} \tag{10.24}$$

The polynomial $\Lambda^{(2t)}(z)$ is the error locator polynomial.

A number of paper have been published to avoid the inversion or to parallelize the structure. It is not the purpose of this chapter to present these paper but they can be found in [15–17].

The last step of the decoding process consists in searching for the roots of the error locator polynomial. If the roots are separate and they are in the field, then it is enough to calculate their inverse to have the error positions. If they are not separate or they are not in the correct field, it means that the word received has a distance from a codeword greater than *t*. In this case an uncorrectable error pattern occurred and the decoding process fails.

The algorithm used to search the roots, known as Chien algorithm, is a method based on trial and error. Substantially each field element is substituted in the error locator polynomial: if it satisfies the equation it is a root, otherwise the following element is tested. The inverse of the found root indicates an error location.

Recall that the error locator polynomial $\Lambda(x)$ of degree *t* at the most, for a *BCH[n,k]*, is defined as:

$$\Lambda(x) = 1 + \Lambda_1 x + \ldots + \Lambda_t x^t \tag{10.25}$$

Hence, verifying if a field element α^i satisfies the equation means verifying Eq. (10.26):

$$1 + \Lambda_1 \alpha^i + \ldots + \Lambda_t \left(\alpha^i\right)^t = 0 \tag{10.26}$$

If the equation is not satisfied the following element is considered, otherwise α^i is a root. In this case the inverse is an error position, i.e. the position 2^m-*1*-*i* is the erroneous one.

10.3 BCH Decoding Failures

BCH codes are not perfect codes: for this reason it is difficult that a codeword with more than t errors moves in the correction sphere of another codeword. The codewords of BCH codes are well separated one from another and only a number of errors much greater than t could partially overlap their correction spheres.

This is the reason why, when more than t errors occur, most of the time the decoding process fails but erroneous corrections are not performed. It is therefore possible to use an error message showing that more than t errors have occurred.

Suppose we have a message containing more than t errors and see how the decoding proceeds. At the exit of the syndromes calculation block it is not possible to detect if the correction capability has been exceeded; on the contrary, the calculation is completed with success by finding t, apparently valid, syndromes.

The syndromes are transferred to the block that searches for the error locator polynomial. As mentioned, the Berlekamp algorithm is a recursive algorithm that searches for the coefficients of the error locator polynomial using successive approximations, by adding at the i-th iteration a discrepancy term d so that $\Lambda(x) + d$ solves the first $i + 1$ equations. The discrepancy term is a monomial that is added to the error locator polynomial previously found. When the degree of the monomial to be added is greater than t, the correction capability of the code has been exceeded. Recalling that the degree of the error locator polynomial is equal to the number of errors that most likely occurred, we can state that this number is reliable up to t. If a degree higher than t is detected at some point in the algorithm, the decoding terminates with an error message.

Unfortunately it is not granted that, when the correction capability of the code is exceeded, this is what happens. On the contrary, most of the times this does not happen and the error locator polynomial apparently seems a valid one with a degree smaller than or equal to t (most of the times equal to t). Consequently, these coefficients, apparently valid, are loaded into the Chien machine.

When the correction capability of the code is exceeded, the Chien algorithm discloses it, since one of the following cases occurs:

- there are coincident roots;
- a sufficient number of roots is not found. Remember that a number of roots equal to the degree of the error locator polynomial has to be found;
- in case of shortened codes it can also happen that the shortened positions, those ones ideally filled in with 0 s, are recognized as erroneous.

In practical implementations the first condition never happens because, given the implementation of the Chien machine, the same element is never tested more than once.

The second condition is the one that actually occurs in real applications. At the end of the Chien algorithm we verify, through a comparator, if the number of roots found is equal to the degree of the error locator polynomial. If this condition is not satisfied, an error message shows that the correction capability of the code has been exceeded.

Finally, the third condition generally never occurs in shortened codes cases, because the use of an “initialization” constant avoids the testing of shortened positions.

However, remember that if the number of errors is much greater then the correction capability, the received message can be found in a correction sphere of another codeword: in this case the code might not be able to understand if the correction capability has been exceeded and might perform erroneous corrections.

Summarizing, we can state that the BCH decoder can be approximate with an ideal one, since erroneous correction are very unlikely to occur [8], unless the received message really falls in another correction sphere. These cases must be studied based on the algebraic structure of the code and will be presented in the next sections.

10.4 Detection Properties

As explained in the previous section, it is unlikely that the BCH decoding algorithm makes erroneous corrections, i.e. we can approximate it with an ideal decoding. It follows that the erroneous corrections are made only when the received message is located in a correction sphere different from the original codeword.

Definition 10.4.1 Given a binary linear code C able to correct t errors, we call the probability of miscorrection P_{ME} the probability that an ideal bounded distance decoder executes erroneous corrections.

Definition 10.4.2 The weighted probability $P_E(w)$ is the probability of executing erroneous corrections when w errors occurred.

Observe that the probability P_{ME} depends on the code C and on the transmission channel.

Theorem 10.4.1 The weighted probability $P_E(w)$ is computed as:

$$P_E(w) = \frac{D_w}{\binom{n}{w}} \tag{10.27}$$

where D_w is the number of decodable words and w is in the range $[t+1,n]$.

The number of decodable words can be computed as

$$D_w = \sum_{i=0}^{n} a_i \sum_{s=0}^{t} N(i,w;s) \tag{10.28}$$

where *N(i,w;s)* is the number of words with weight w with a distance s from a word of weight i. This is computed by Eq. (10.29)

$$N\left(i,w;s\right)=\begin{cases}\binom{n-i}{\frac{s+w-i}{2}}\binom{i}{\frac{s-w+1}{2}} & \text{if } |w-i|\le s\\ 0 & \text{if } |w-i|>s\end{cases} \tag{10.29}$$

Substituting Eq. (10.28) in Eq. (10.27) we have:

$$P_E(w)=\frac{\sum_{i=0}^{n}a_i\sum_{s=0}^{t}N(i,w;s)}{\binom{n}{w}} \tag{10.30}$$

P_{ME} is computed based on $P_E(w)$ as described in Eq. (10.31)

$$P_{ME}=\sum_{w=t+1}^{n}P_E(w)\phi(w) \tag{10.31}$$

where $\Phi(w)$ is the probability that a word has weight w.

For a binary symmetric channel BSC we have:

$$P_{ME}=\sum_{w=t+1}^{n}D_w p^w(1-p)^{n-w} \tag{10.32}$$

where p is the bit error probability.

It follows that we have to compute the value D_w. This value can be computed according with Eq. (10.28). Unfortunately the weights a_i are unknown for BCH codes and must be estimated.

10.5 BCH Weight Estimation

There are a number of different theorems that helps in estimating the weight of a BCH code. Here below we will see the major ones and how they behave in comparison with real weights.

First of all, we present a result that establishes a relationship between the weight distribution of a BCH code and the weight distribution of its dual code.

Theorem 10.5.1 Given C a *BCH[n,k,d]* code and C_E its extension, C has weight distribution $\{a_i\}$ and C_E has weight distribution $\{A_i\}$. The following equations hold true:

$$\begin{aligned} a_{2i-1} &= \frac{2i}{n} A_{2i} \\ a_{2i} &= \frac{n-2i}{n} A_{2i} \end{aligned} \tag{10.33}$$

Observe that a BCH code has symmetrical weight distribution and the word composed by all 1 is a valid codeword. Moreover, given that BCH has an odd length (i.e. $n = 2^m\text{-}1$), also for the extension BCH$_\text{E}$ the word composed by all 1 is a valid codeword. Finally, observe that the dual code of C_E has only even weight codewords.

One of the most important weight estimation is the Peterson one. It was the first estimation and it is not an upper or a lower bound but an approximation.

Theorem 10.5.2 [18] ***Peterson Estimation*** The weight a_i of a primitive BCH code of length n and error correction capability t can be approximated as

$$a_i \cong \frac{\binom{n}{i}}{(n+1)^t} \tag{10.34}$$

In order to have upper bounds, different correction terms are added to Eq. (10.34). In other words, for the extension code BCH$_\text{E}$ the estimations use the following relationship:

$$A_i = \begin{cases} 0 & i \equiv 1 \bmod 2 \\ \frac{\binom{n}{i}}{2^{mt}} (1 + E_i) & i \equiv 0 \bmod 2 \end{cases} \tag{10.35}$$

In order to compare different estimations a real case is shown. For BCH[255, 207,13] the weights w are known. Figure 10.6 shows the relative errors with respect to the real weights with different estimations for this code. On the x-axis there is the weight w, while on the y-axis we find

$$R(w) = \frac{A(w)_{EST} - A(w)_{REAL}}{A(w)_{REAL}} \tag{10.36}$$

We distinguish three different behaviors. The first estimation set has a very low error on the first weights but a very high error in the middle.

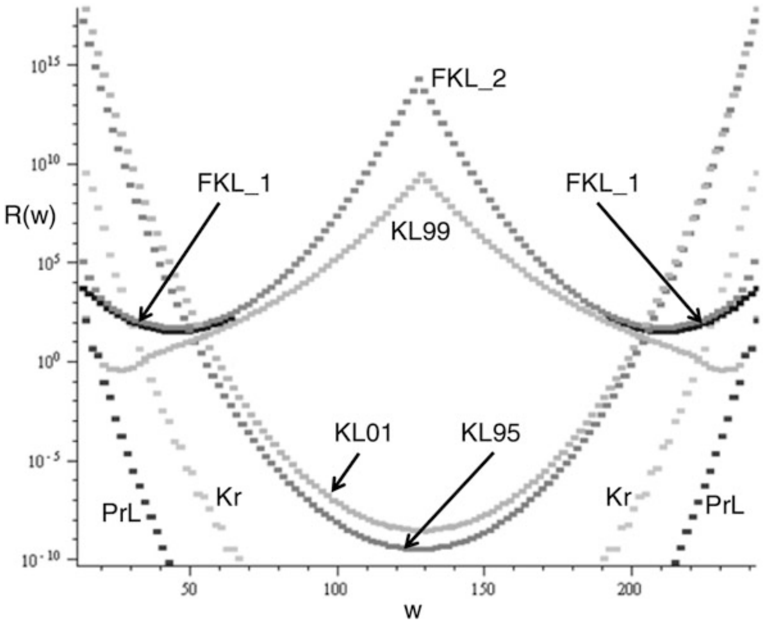


Fig. 10.6 Relative error between real weights and different estimations for BCH[255,207,13]

The following theorems describe the estimations belonging to this set:

Theorem 10.5.3 [19] Given

$$t \geq 3, w = n - 2d^*, t < i \leq \frac{n-w}{4}$$

For cases:

$$\begin{aligned} i &= t+1, & t &= 3, m \geq 5 \\ & & t &= 4, m \geq 9 \\ & & t &= 5, m \geq 15 \\ i &= t+2, & t &= 4, m \geq 7 \\ & & t &= 5, m \geq 9 \\ & & t &= 6, m \geq 11 \\ & & t &= 7, m \geq 15 \end{aligned}$$

Equation (10.37) holds true

$$\frac{E_{2i}}{2} \leq 2^{-m(i-t)} \prod_{h=1}^{2i-1} \left(1 - \frac{h}{n+1}\right)^{-1} \prod_{h=1}^{i} (2h-1) \left(2^{i-1} - 1\right) \tag{10.37}$$

For all other cases Eq. (10.38) holds true

$$\frac{E_{2i}}{2} \le 2^{-m(i-t)} \prod_{h=1}^{2i-1} \left(1 - \frac{h}{n+1}\right)^{-1} \left\{ \prod_{h=1}^{i} (2h-1) \sum_{h=\left\lfloor \frac{i-t}{2} \right\rfloor + 1}^{\left\lfloor \frac{i}{2} \right\rfloor} \binom{i}{2h} + \prod_{h=1}^{i} (2h-1) \frac{(2i)![2(t-1)]^{2i-2q-2t}}{2^q q! (2i-2q)!} \right\}$$

$$q = \left[i + \frac{3}{4} + (t-1)^2 - \sqrt{\frac{1}{16} + \left(2i + \frac{3}{2}\right)(t-1)^2 + (t-1)^4} \right] \tag{10.38}$$

The proof of this theorem is behind the purpose of this chapter. However, note that this is a very complex equation that estimates very well the weights, but the big drawback is that it can be applied only to some cases. For BCH[255,207,13] sketched in Fig. 10.6 this estimation (labeled as FKL_1) is applicable only for weight w in the range [13,64] and in the range [256–64, 256]. However, there is another estimation (labeled as FKL_2 in Fig. 10.6) extended to all weights.

Theorem 10.5.4 [19] For $t < i \le 2^{m-2}$ the following inequality holds true:

$$\left|\frac{E_{2i}}{2}\right| \le 2^{-m(i-t)} \left\{ \sqrt{\frac{8}{\pi}\left(1 - \frac{2i}{n+1}\right)} (2i)^i e^{-ic(i,t-1)} + \frac{2}{\pi}\sqrt{\frac{2i}{2i-1}} \prod_{j=1}^{t} (2j-1) \left|2(t-1)\right|^{-2t} \left(t - 1 + \sqrt{2i + (t-1)^2}\right)^{2i} e^{2ib(2i)} \right\}$$

$$c(u,x) = 1 - (\ln 2) H\left(\frac{x}{u}\right) - \frac{5u}{2(n+1)}$$

$$H(x) = \begin{cases} -x\log_2 x - (1-x)\log_2(1-x) & 0 \le x \le \frac{1}{2} \\ 1 & \frac{1}{2} < x \le 1 \end{cases}$$

$$b(s) = -\frac{1}{2} + \frac{5s}{8(n+1)} + \frac{1}{1 + \sqrt{1 + 4s(n+1)u^{-2}}} \tag{10.39}$$

As shown in Fig. 10.6 this estimation is very similar to the previous one for small weights but it has a huge error in the middle.

Another family of estimations has a big error on the first weights but a very low error in the middle.

Theorem 10.5.5 [20] For a primitive BCH code of length $n = 2^m\text{-}1$, an upper bound for the weight distribution is:

$$a_i = \frac{\binom{n}{i}}{(n+1)^t}\left(1 + E_{i*}\right) \tag{10.40}$$

where $i* = i + 1$ if i is odd and $i* = i$ if i is even and E_i is computed with the following inequality:

$$|E_i| \le \frac{n^t \binom{n+1}{\frac{n+1}{2}} \binom{\frac{n+1}{2}}{\frac{1}{2}}}{\binom{n+1}{i}\binom{n+1}{d*}} \tag{10.41}$$

Another estimation of the correction term E_i is proposed in the following theorem

Theorem 10.5.6 [21] The correction term E_i can be estimated as:

$$|E_i| \le (n+1)^t \sqrt{\frac{(2i\,(n+1-i)+n+1)\,(n+1)^t \binom{i}{\frac{i}{2}}\binom{n+1-i}{\frac{n+1-i}{2}}}{2\,(n-d*)\binom{n+1}{d*}\binom{n+1}{i}}} \tag{10.42}$$

As before, the proof of this theorem is behind the purpose of this chapter. However, it is important to state that it is based on the maximization of a specific class of polynomials called Krawtchouk polynomials.

Definition 10.5.1 For every positive integer n we call Krawtchouk polynomial of degree k $P_k(x,n) = P_k(x)$

$$P_k\,(x,n) = \sum_{j=0}^{k} (-1)^j \binom{x}{j}\binom{n-x}{k-j} \tag{10.43}$$

By using a result of [22] it is possible to prove that:

$$|E_i| \le \frac{2^{mt}}{\binom{n}{i}} \max_{d* \le x \le 2^{m-1}} |P_i\,(x)| \tag{10.44}$$

It follows that an upper bound for max $|P_i(x)|$ is an upper bound for E_i.

It is possible to compute exactly this maximum value, with the drawback of a high computational cost. Hence, it is not always possible for all the length and correction capability. In Fig. 10.6 the relative error obtained with the maximization of Krawtchouk polynomials is labeled as "Kr". Finally the last estimation is presented below.

Theorem 10.5.7 [23] Given *f(x)* and *g(x)* even function with respect *(n + 1)/2* described by Eq. (10.45)

$$f(x) = \sum_{i=0}^{k} f_{2i} P_{2i}(x) \text{ and } g(x) = \sum_{i=0}^{k} g_{2i} P_{2i}(x) \tag{10.45}$$

with the following properties

$$\begin{aligned} &f_{2k} > 0 \\ &g_{2k} > 0 \\ &f(x) \geq 0 \quad \forall x \\ &g(x) \leq 0 \quad d^* \leq x \leq n - d^* \end{aligned} \tag{10.46}$$

It follows that

$$\begin{aligned} &A_{2k}^{f} \leq A_{2k} \leq A_{2k}^{g} \\ &A_{2k}^{f} = \frac{\binom{2^m}{2k}}{2^{mt}}\left(1 - E_{2k}^{f}\right) = \frac{\frac{f(0)}{2^{mt}} - f(0) - \sum\limits_{i=t+1}^{k-1} f_{2i} A_{2i}}{f_{2k}} \\ &A_{2k}^{g} = \frac{\binom{2^m}{2k}}{2^{mt}}\left(1 - E_{2k}^{g}\right) = \frac{\frac{g(0)}{2^{mt}} - g(0) - \sum\limits_{i=t+1}^{k-1} g_{2i} A_{2i}}{g_{2k}} \end{aligned} \tag{10.47}$$

A convenient choice for function *f(x)* and *g(x)* is proposed by the authors and is the following

$$\begin{aligned} f(x) &= P_k^2(x) \\ g(x) &= (P_2(x) + C_t)\, P_{k-1}^2(x) \quad C_t = -P_2\left(d^*\right) \end{aligned} \tag{10.48}$$

Figure 10.7 shows the relative error with respect the real weights (as described in Eq. (10.36)) for the estimation made with the maximization of Krawtchouk polynomials, for the best estimation among all the theoretical estimation and for the estimation with linear programming technique.

As shown in Fig. 10.7, Krawtchouk estimation is better compared with the minimum among all other theoretical estimations. Anyway, we have the special case of linear programming estimation which shows a quasi-null error.

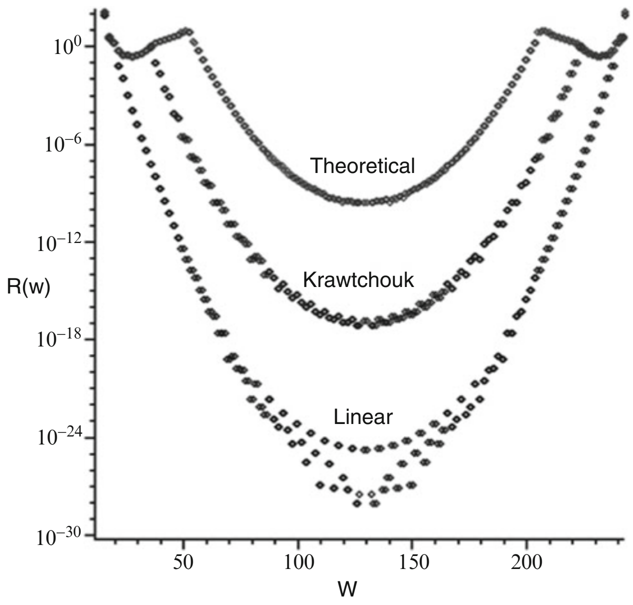


Fig. 10.7 Relative error between real weights and Krawtchouk estimation, linear programming estimation and the minimum among all theoretical estimations for BCH[255,207,13]

A linear programming problem (LP) with N real variables $x_1, \ldots, x_N$ with M constraints like

$$\sum_{j=1}^{N} \alpha_{ij} x_j \le c_i \quad \text{or} \quad \sum_{j=1}^{N} \alpha_{ij} x_j = c_i \tag{10.49}$$

with c_i and α_i positive real variables, can be represented in a matrix form:

$$\begin{pmatrix} \alpha_{11} & \cdots & \alpha_{1N} \\ \vdots & \ddots & \vdots \\ \alpha_{M1} & \cdots & \alpha_{MN} \end{pmatrix} \begin{pmatrix} x_1 \\ \vdots \\ x_N \end{pmatrix} (REL) \begin{pmatrix} c_1 \\ \vdots \\ c_N \end{pmatrix} \tag{10.50}$$

where REL represents the relationship for each components. The purpose is to find a solution x able to maximize or minimize the objective function

$$\sum_{i=1}^{N} o_i x_i \tag{10.51}$$

Linear programming technique is applied to the BCH weight estimation by means of *Fujiwara algorithm* described in [19, 24].

MacWilliams identity is the objective function we need to maximize, where B_j is the weight distribution for the extension of the dual code:

$$\max \sum_{i=d*}^{2^m-d*-1} P_s(j)B_j \;\; s = d*, ..., 2^m - d* - 1 \tag{10.52}$$

The constraints that we need to add on B_j are the Pless power-moment identities [14]

$$\sum_{j=d*}^{2^m-d*-1} \left(\frac{2^m}{2} - j\right)^{2l} B_j = 2^{2^m-k} M_{2l} - (1 + B_n)\left(\frac{2^m}{2}\right)^{2l} 0 \le l \le t$$
$$M_i = 2^{-i}\left(\frac{d^i}{dx^i}\cosh^{2^m} x\right)_{x=0} \tag{10.53}$$

The more constraints we add the easier to find a solution in a fast way. In this case we can add constraints involving the distance of the extension of the dual code as shown in the following example.

Example 1 Let's take the extension of BCH[2048,1992] with distance 12. We can exploit the properties of Reed-Muller codes [9, 10] so that

$$\begin{aligned}
&RM\,(r, m) \subset BCH_E\,(2^m, 2^{m-r} - 1)\\
&RM\,(7, 11) \subset BCH_E\,(2048, 16) \subset BCH_E\,(2048, 12)\\
&RM\,(7, 11) \supset BCH_E{}^*\,(2048, 16) \supset BCH_E{}^*\,(2048, 12)\\
&RM\,(11 - 7 - 1, 11) \subset BCH_E{}^*\,(2048, 12)
\end{aligned} \tag{10.54}$$

The weights of this Reed-Muller code are non-null and multiple of 2^s with $s = 3$. It follows that the weights of the code EBCH*(2048,12) are non-null and multiple of 8 starting from the minimum distance. By using Reed-Muller properties this distance is $d* \ge 2^{m-r} = 256$, while we obtain $d* \ge 2^{m-1} - (t-1)\sqrt{2^m} = 844$ with Carlitz-Uchiyama inequality.

The estimation made with Krawtchouk maximization and Linear Programming technique are the most effective ones even if they are prohibitive for long codes due to computational complexity. Moreover, it's not always possible to find all the constraints and, even if they would be available, it could take a year to find an estimation with today's computers.

10.6 BCH Weight Estimation: Real Cases Analysis

In this part we will analyze different cases to compare different behaviors among estimations.

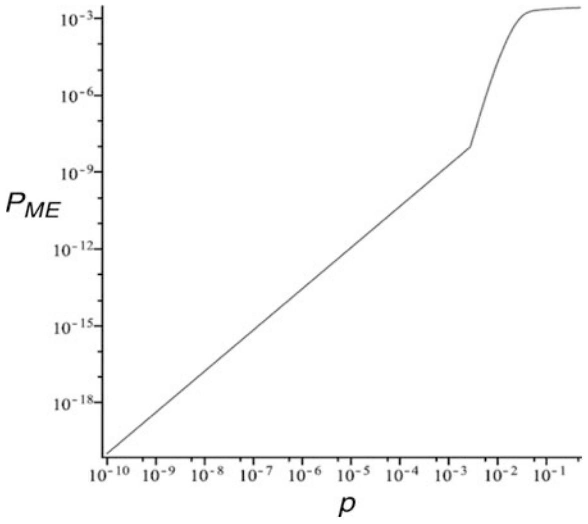


Fig. 10.8 P_{ME} behavior for BCH[255,207,13]

10.6.1 BCH[255,207,13]

In this case the weights of the code are known, since the code is short and the error correction capability of the code is small, i.e. six errors. Figures 10.6 and 10.7 represent the weight estimation comparison for this code. Observe that the weight estimation depends a lot on the estimation on the first weight.

P_{ME} graph is shown on Fig. 10.8. As it is possible to see the behavior is monotonic increasing. The comparison graph for P_E estimation is shown in Fig. 10.9. In the graph, there is the P_E computed by taking the minimum among all the theoretical estimations, the estimation obtained with linear programming technique and the real one (as we know the real weights).

As it is possible to see, the real P_E is a monotonic increasing function, while all the estimations are very good in the middle but have a bump on the first (and last) weight.

10.6.2 BCH[1023, 993,7]

Also in this case the weights are known since the error correction capability of the code is only 3. Figures 10.10 and 10.11 shows the relative error for different weight estimations compared with the real weights and compared with the minimum among all the estimations.

As it possible to see the two figures are quite identical. Figure 10.12 shows P_E behavior for this code using real weights. Also in this case we note that the behavior is monotonic increasing with a very long floor in the middle.

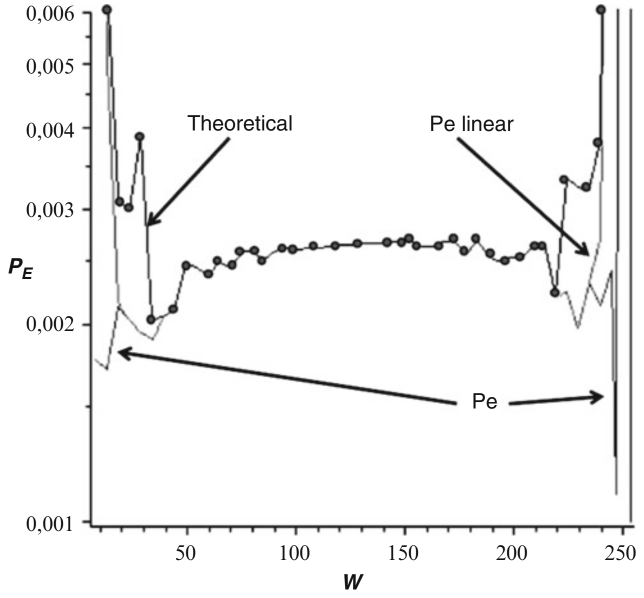


Fig. 10.9 P_E behavior for BCH[255,207,13]. The graph shows the real P_E, the one obtained with linear programming technique and the one obtained by taking the minimum among all the theorical estimations

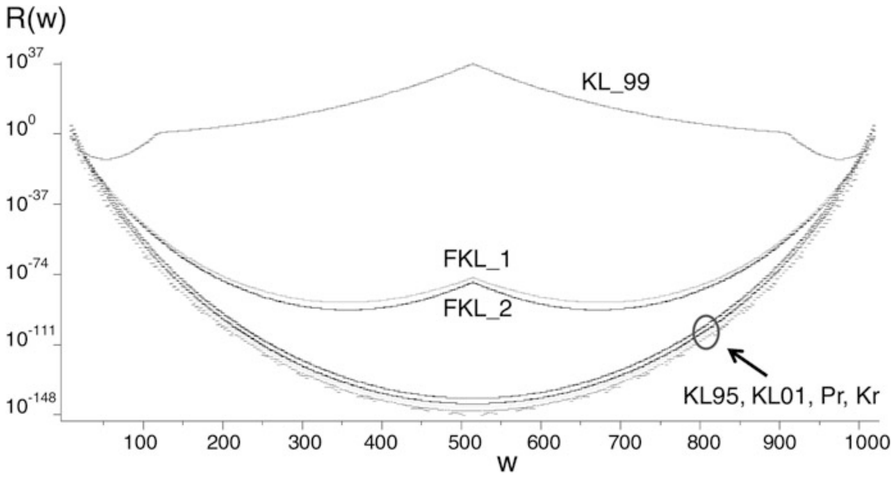


Fig. 10.10 Relative error between real weights and different estimations for BCH[1023,993,7]

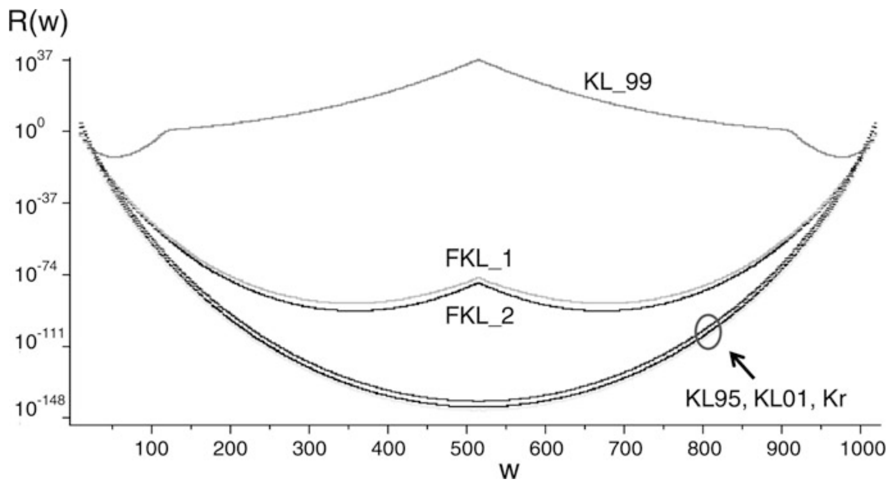


Fig. 10.11 Relative error between different estimations and the minimum among all the different estimations for BCH[1023,993,7]

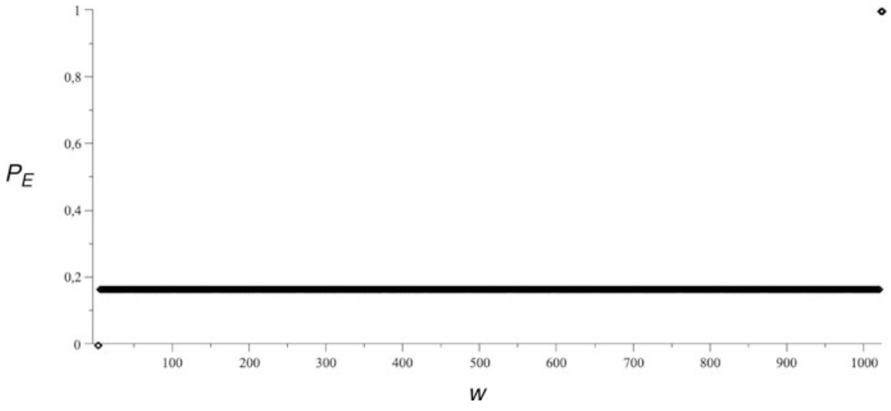


Fig. 10.12 P_E behavior for BCH[1023,993,7]

In Fig. 10.13 P_{ME} is shown. The x-axis represents a probability belonging to the range [0,1/2] and is divided in 500 subsets.

10.6.3 BCH[4095, 3975,21]

This is the first case where we don't know the real weights (Fig. 10.14). The behavior is similar to the previous cases, but a bump on the first weight estimation pops up. This is partially due to the fact that we don't know the minimum distance of the dual code of the extension.

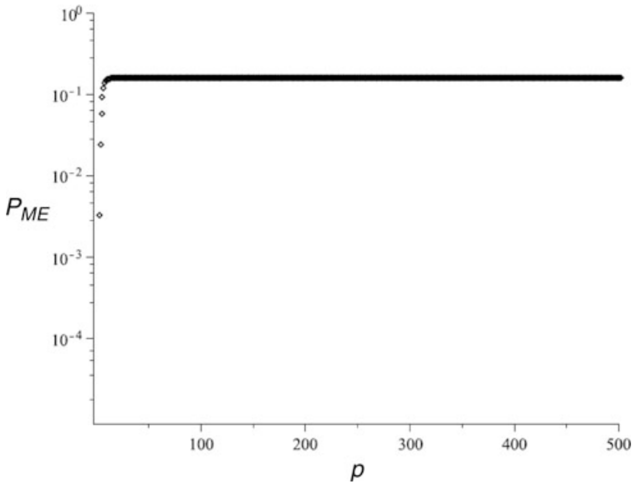


Fig. 10.13 P_{ME} behavior for BCH[1023,993,7]

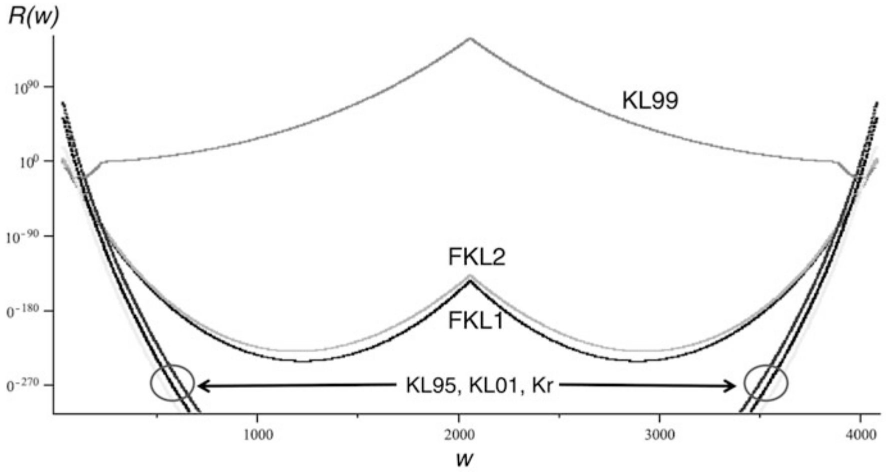


Fig. 10.14 Relative error between different estimations and the minimum among all the different estimations for BCH[4095, 3975,21]

For example, in BCH[255,207,13] case the distance of the code with Carlitz-Uchiyama bound is 48, while the real one is 64. Note that here it is not possible anymore to use Linear Programming technique due to computational complexity.

Figure 10.15 shows P_E behavior: it has the usual long floor, but for the first time we see a bump on the first weights. This behavior is shown also in Fig. 10.16 where we have a zoom on the first weights.

P_{ME} function is not represented here, since it has the same behavior as P_E.

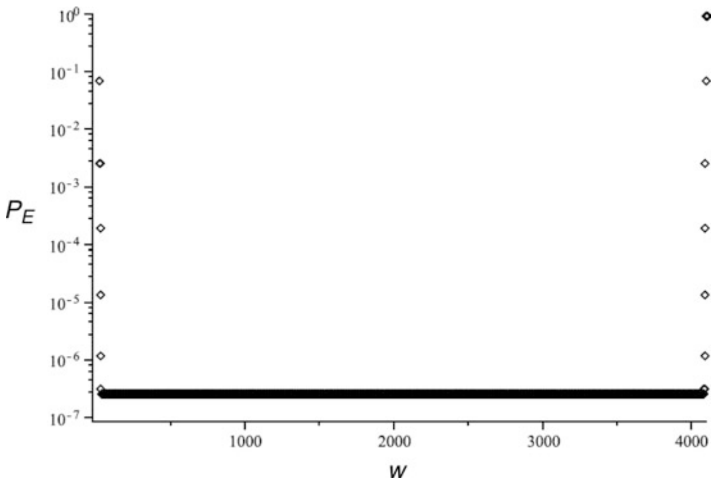
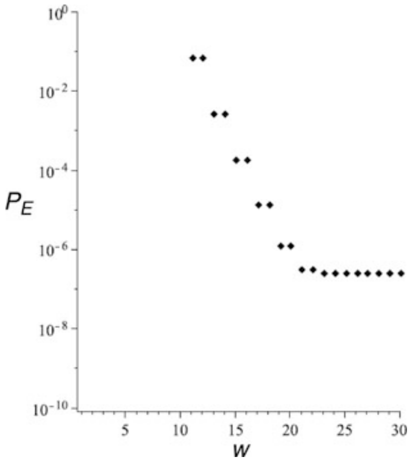


Fig. 10.15 P_E behavior for BCH[4095, 3975,21]

Fig. 10.16 P_E zoom on the first weights for BCH[4095, 3975,21]



10.6.4 BCH[16383, 15851,77]

Also in this case the real weights are unknown. Moreover, it is not possible anymore to use the Krawtchouk estimation and the linear programming estimation due to computational complexity. As shown in Fig. 10.17, we have an error on the first weights of hundreds of order of magnitudes. This is mainly due to the poor estimation on the distance of the dual code that gives also a big error on P_E and P_{ME} estimation (Figs. 10.18 and 10.19).

Please observe that, in this case, P_{ME} is almost 1 for low p values!

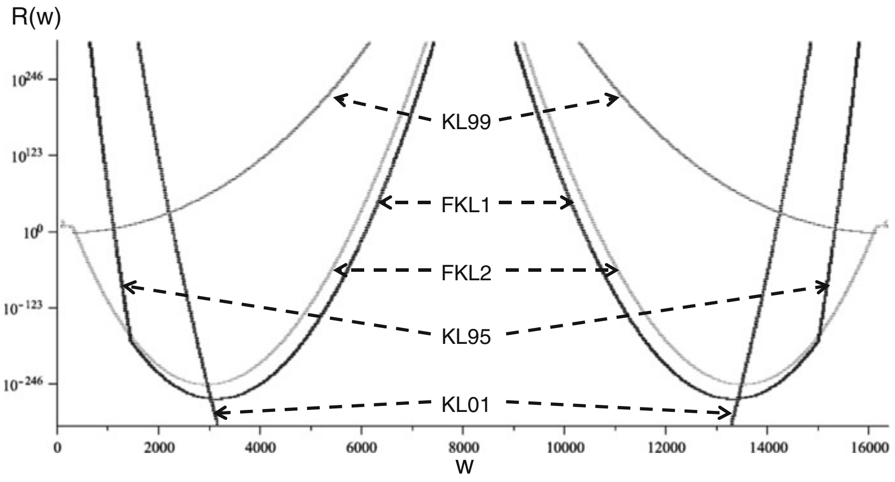
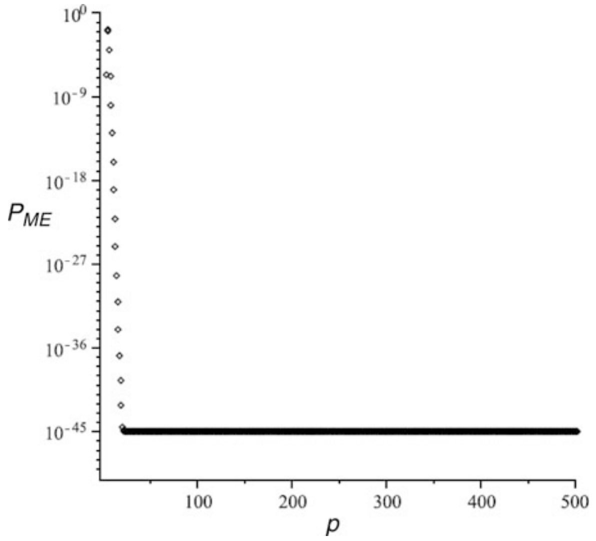


Fig. 10.17 Relative error between different estimations and the minimum among all the different estimations for BCH[16383, 15851,77]

Fig. 10.18 P_{ME} behavior for BCH[16383, 15851,77]



10.7 BCH Detection Conclusion

As shown in the previous section, the error on the estimation on the first weights has a huge effect on P_E and P_{ME}. In particular a poor estimation shows up as a bump on the first weights that become greater as the code length increases.

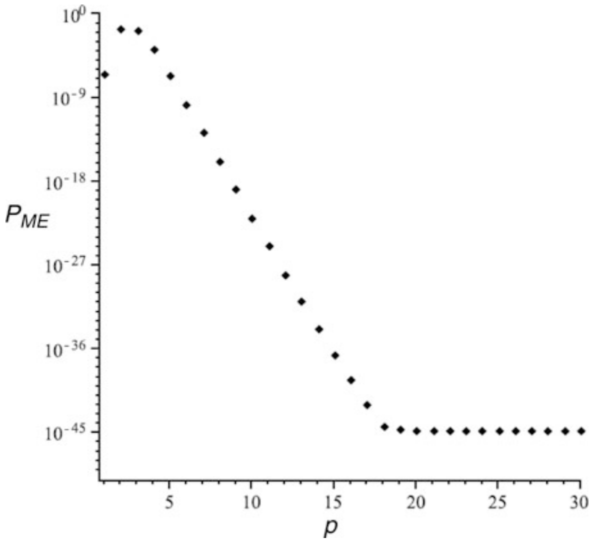


Fig. 10.19 P_{ME} zoom on the first weights for BCH[16383, 15851,77]

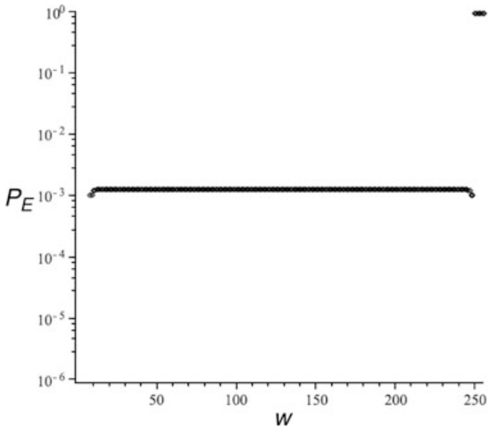


Fig. 10.20 P_E behavior for BCH[255,207,13] using Peterson estimation

One of the best estimation, even if it is not an upper bound, is the Peterson estimation (Theorem 10.5.2). Figures 10.20 and 10.21 shows P_E and P_{ME} behavior for BCH[255,207,13] using Peterson estimation. We can see that we have the monotonic increasing behavior that we expect when real weights are known.

Figure 10.22 shows P_{ME} behavior for BCH[16383, 15851,77] using Peterson estimation. Recall (Sect. 10.6.4) that here the real weights are unknown and we had a bump at the beginning using upper bound estimations. Instead, by using Peterson estimation a monotonic behavior can be seen.

It follows that the real P_E and P_{ME} profile should be monotonic with a wide floor in the middle. When the code length is high and the code rate is high this floor can be approximate with [25].

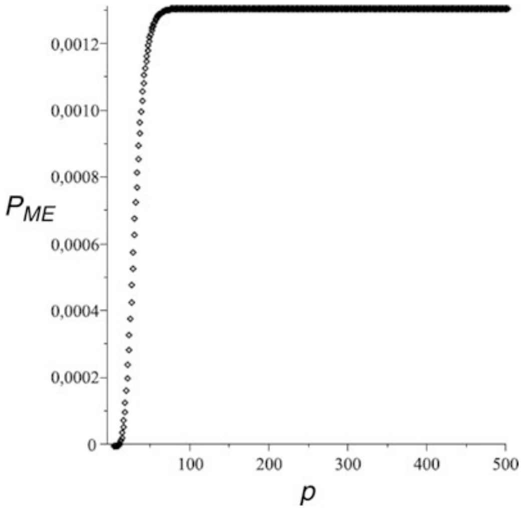


Fig. 10.21 P_{ME} for BCH[255,207,13] for Peterson estimation

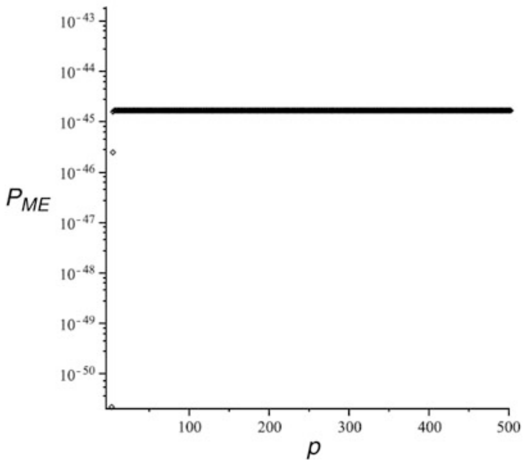


Fig. 10.22 P_{ME} for BCH[16383, 15851,77] for Peterson estimation

$$Q = 2^{-(n-k)} \sum_{s=0}^{t} \binom{n}{s} \tag{10.55}$$

Summarizing, we can state that when the length is high, the BCH code has a very good detection properties that made it suitable for the implementation in SSDs. In fact, when a catastrophic error occurs or when the error correction capability of the code is passed, the BCH code declares a decoding failure without attempting erroneous corrections. This is a key point when using BCH code concatenated with another code.

10.8 Multi-channel BCH

Solid State Disks are built with many Flash channels connected to the host through a high-speed interface such as SATA or PCI Express (Chap. 2). In this scenario the performance of the SSD is determined by the ECC needed to overcome the high error-rate. It follows that binary BCH code must have a structure able to handle a number of channels together, without being the performance bottleneck.

It has already been studied [4] how the native serial structure of BCH can be parallelized to work on one byte or dword at a time. In multi-channel architectures, this is not enough and multiple encoding and decoding machines must be implemented. In particular, given raw bit error rate higher than 10^{-4}, the most likely situation is that almost all the pages read in parallel need correction.

Figure 10.23 shows the probability that *n* chunks require correction given a bit error probability. For example, if the bit error probability is 10^{-5}, we have a probability of around 10^{-2} of having 32 error-free chunks, a probability of 10^{-1} that three chunks over 32 require correction, a probability of 10^{-10} that 24 chunks require correction and so on. The highest curve in the graph is the most likely number of correction required: for example, at BER of 10^{-5} 3-err and 6-err are the highest.

If BER is higher than $2*10^{-4}$ the most likely number of correction over 32 chunks is 32; in other words every chunk requires correction.

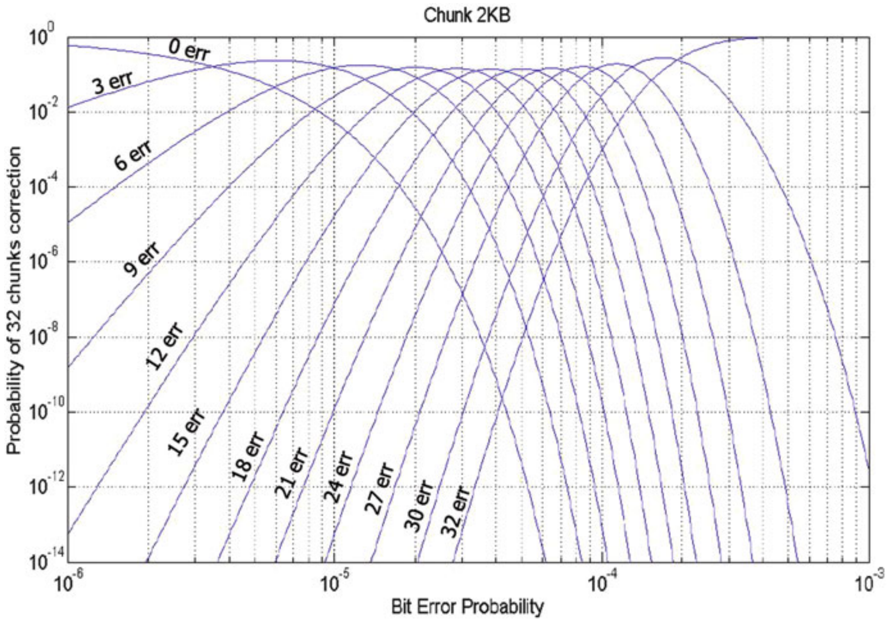


Fig. 10.23 Probability of *n* chunks over 32 requiring correction

In order to keep up with the bandwidth requirements, the most straightforward solution would be to have one encoder and one decoder per channel. However, this approach is extremely area consuming, especially because of the decoder.

As far as the encoding is concerned, it is very important that data coming from the host are dispatched to the various channels without latency. There are three possible approaches, starting from the less area consuming:

- single encoder shared among all Flash channels [26];
- a pool of encoders;
- one encoder per channel.

The right hardware choice comes from the tradeoff between silicon area and latency.

The decoding is trickier than encoding since the algorithm is composed of three steps as shown in Fig. 10.3. Please note that null syndromes mean an error-free message: therefore, decoding doesn't need to go through Berlekamp and Chien. This situation is very common when the solid state drive is fresh.

As the reader can notice, Fig. 10.3 shows a pipelined structure. In order to design each decoding step in the correct way, we need to study its latency:

- the input of the syndrome computation is the read codeword of length n. If the t syndrome machines works with a parallelism of b bits, the latency of the syndrome computation is proportional to n/b;
- Berlekamp algorithm takes the t syndromes as input and finds the error locator polynomial coefficients in t iterations. It follows that it has a latency proportional to t;
- Chien search takes the error locator polynomial computed by Berlekamp as input. It substitutes n elements in the polynomial to see if they are roots. If the machine is able to work on c elements at a time, its latency is proportional to n/c. If the degree of the error locator polynomial is v, the Chien machine stops when it finds v roots, without substituting the remaining elements of the field. Hence n/c is the worst case latency.

In order to exploit the pipelined architecture we must have numbers n/b, t, and n/c as similar as possible. However, n is generally much higher than t and there are design constraints on b and c. Hence, we can achieve a balance by adding more machines either to syndrome or Chien.

Let's assume $n1$ HW machines to perform syndrome computation, $n2$ HW machines to execute Berlekamp algorithm and $n3$ HW machines to perform Chien search. We choose $n1$, $n2$ and $n3$ so that Eq. (10.56) holds true:

$$\frac{n}{(n1 * b)} \approx \frac{t}{n2} \approx \frac{n}{(n3 * c)} \tag{10.56}$$

The resulting decoding structure is sketched in Fig. 10.24.

Finally we can use probabilistic consideration in choosing the number and the size of Chien machines. The size of the Chien block depends on the parallelism and

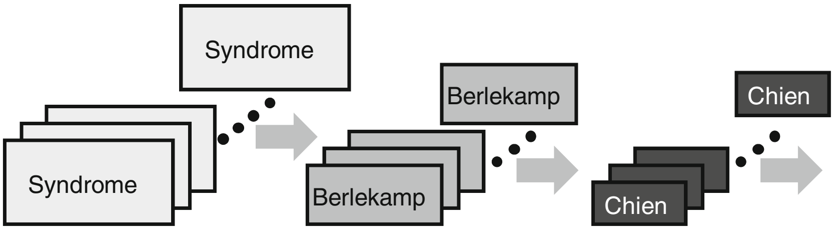


Fig. 10.24 ECC decoding structure for handling multiple channels

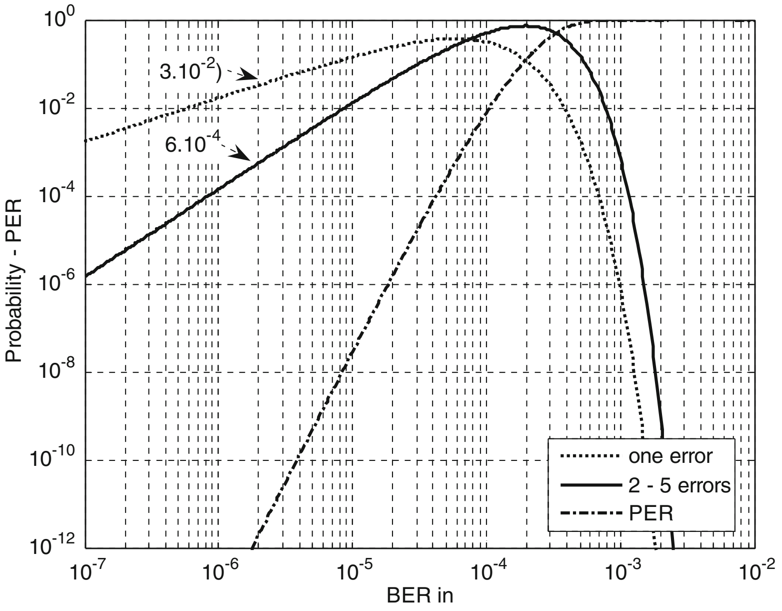


Fig. 10.25 For a 2112-Byte page, representation of single error probability, of two to five error probability and of Page Error Rate using an ECC able to correct five errors

on the error correction capability. In other words, a machine able to correct *x* errors is half in area with respect to a machine able to correct *2x* errors (given the same parallelism). What happens in a real SSD is that the decoding (and so the Chien machine) is always needed but the number of errors that must be corrected is not always *t*.

Figure 10.25 shows, for a 2112-Byte page, the probability of having to correct only one error, the probability of having to correct two to five errors, and the probability of error (PER) after 5 bits correction as a function of the BER_{in}. For a value of BER_{in} around 10^{-6}, we have that the probability of a single error is equal to $3{*}10^{-2}$ and the probability of two to five errors is equal to $6{*}10^{-4}$ respectively. The probability of a single error is definitely more significant and since the Berlekamp algorithm exactly indicates the number of errors to correct, it may be useful to exploit this information [8]. For example, suppose that from Eq. (10.56) we obtain *n3* = *3*. If *t* is equal to 5 and the area of a Chien machine able to correct one error is 1 U, we obtain an area of 5*3 = 15 U for implementing three machines able to correct five errors. However, from Fig. 10.25 we see that most of the time the correction of only one error is required. It follows that we can implement twomachines able to correct one error and only one complete Chien machine able to correct five errors. The area would be 1 + 1 + 5 = 7 U with a gain of 6 U at same performances.

This approach can always be used when the error density function is known. The result is that we can have a pool of Chien machines with different correction capabilities and parallelism. It's Berlekamp machine's task to dispatch the message to the correct Chien machines depending on its degree.

With a good optimization in the number of machines per each step, BCH does not limit the bandwidth between the drive and the host.

As explained in this chapter, multi-channel management and detection properties are the key points to address when developing a BCH engine for Enterprise Class Solid State Disks.

References

1. C.E. Shannon, A mathematical theory of communication. Bell Syst. Tech. J. **27** 379–423, 623–656 (1948)
2. R.C. Bose, D.K. Ray-Chaudhuri, On a class of error-correcting binary group codes. Inform. Contr. **3**(1, March), 68–79 (1960)
3. A. Hocquenghem, Codes correcteurs d'erreurs. Chiffres **2**, Sept (1959)
4. I.S. Reed, G. Solomon, Polynomial codes over certain finite fields. J. SIAM **8**(2, June), 300–304 (1960)
5. C. Berrou, A. Glavieux, P. Thitimajshimima, Near Shannon limit error-correcting coding and decoding: Turbo-codes, in *Proceedings of ICC'93*, Geneva, Switzerland, May 1993, pp. 1064–1070
6. R.G. Gallager, Low-density parity-check codes. IRE Trans. Inf. Theory. **IT-8**, 21–28 (Jan 1962)
7. D.J.C. MacKay, R.M. Neal, Near Shannon limit performance of low density parity check codes. Electron. Lett. **32**(18, August), 1645–1646 (1996)
8. R. Micheloni, A. Marelli, R. Ravasio, *Error Correction Codes for Non-Volatile Memories* (Springer, Berlin, 2008)
9. S. Lin, D.J. Costello, *Error Control Coding* (Prentice Hall, Upper Saddle River, 2004)
10. T.K. Moon, *Error Correcting Coding – Mathematical Methods and Algorithms* (Wiley, Hoboken, 2005)
11. Y. Chen, K. Parthi, Small area parallel Chien search architecture for long BCH codes. IEEE Trans. Very Large Scale Integration (VLSI) Syst. **12**(5, May), 545–549 (2004)
12. R. Micheloni et al., A 4Gb 2b/cell NAND flash memory with embedded 5b BCH ECC for 36MB/s system read throughput, in *ISCC Dig. Tech. Papers*, San Francisco, Feb 2006
13. R. Micheloni, L. Crippa, A. Marelli, *Inside NAND Flash Memories* (Springer, Berlin, 2010)
14. E.R. Berlekamp, *Algebraic Coding Theory* (McGraw-Hill, New York, 1968)
15. H.O. Burton, Inversionless decoding of binary BCH codes. IEEE Trans. Inf. Theory **17**(4, July), 464–466 (1971)

16. I.S. Reed, M.T. Shih, T.K. Truong, VLSI design of inverse-free Berlekamp-Massey algorithm. IEEE Proc. **138**, 295–298 (Sept 1991)
17. S. Mizrachi, D. Stopler, Efficient method for fast decoding of BCH binary codes. US Patent 2003/0159103 A1, Aug 2003
18. W.W. Peterson, E.J. Weldon Jr., *Error-Correcting Codes*, 2nd edn. (MIT Press, Cambridge, 1972)
19. T. Kasami, T. Fujiwara, S. Lin, An approximation to the weight distribution of binary linear codes. IEEE Trans. Inf. Theory **31**(6), 769–780 (1985)
20. I. Krasikov, S. Litsyn, On spectra of BCH codes. IEEE Trans. Inf. Theory **41**, 786–788 (1995)
21. I. Krasikov, S. Litsyn, On the distance distribution of duals BCH codes. IEEE Trans. Inf. Theory **45**, 247–250 (2001)
22. F.J. MacWilliams, N.J.A. Sloane, *The Theory of Error-Correcting Codes*. North-Holland Mathematical Library, vol. 16 (North-Holland Publishing Company, Amsterdam, 1977)
23. O. Keren, S. Litsyn, More on the distance distribution of BCH Codes. IEEE Trans. Inf. Theory **1**, 251–155 (1999)
24. M. Sala, A. Tamponi, A linear programming estimate of the weight distribution of BCH(255, k). IEEE Trans. Inf. Theory **46**(6), 2235–2237 (2000)
25. M.G. Kim, J.H. Lee, Decoder error probability of binary linear block codes and its application to binary primitive BCH codes. IEICE Trans. Fundam. **E79-A**(4), 592–599 (1996)
26. Y. Lee, H. Yoo, I. Yoo, I.C. Park, 6.4Gb/s multi-threaded BCH encoder and decoder for multi-channel SSD controllers, in *ISCC Dig. Tech. Papers*, San Francisco, Feb 2012

Chapter 11
Low-Density Parity-Check (LDPC) Codes

E. Paolini

Abstract In this chapter, low-density parity-check (LDPC) codes, a class of powerful iteratively decodable error correcting codes, are introduced. The chapter first reviews some basic concepts and results in information theory such as Shannon's channel capacity and channel coding theorem. It then overviews the Flash memory channel model. Finally, it addresses LDPC codes describing both their structure and efficient implementation, and their decoding algorithms. Simulation results are also provided.

11.1 Shannon Limit

11.1.1 Entropy and Mutual Information

Let X be a discrete random variable taking its values in a set $\mathcal{X}$, according to some probability mass function (pmf) $p(x) = \Pr\{X = x\}$. The entropy of X is defined as

$$H(X) = -\sum_{x} p(x) \log_2 p(x).$$

Intuitively, the entropy $H(X)$ may be thought as the uncertainty associated with the random variable. For example, a deterministic variable is characterized by a zero entropy while, for a given positive integer M, the random variable with the largest entropy among all discrete random variables whose support set $\mathcal{X}$ has cardinality M is the uniform one, i.e., $p(x) = 1/M$ for all $x \in \mathcal{X}$. In this latter case we obtain $H(X) = \log_2 M$.

E. Paolini (✉)
Department of Electrical, Electronic and Information Engineering G. Marconi (DEI),
University of Bologna, Cesena, Italy
e-mail: e.paolini@unibo.it

Consider now a second discrete random variable $Y \in \mathcal{Y}$ characterized by a pmf $p(y)$. Let $p\,(y|x) = \Pr\{Y = y|X = x\}$ be the pmf of Y conditioned to the event $\{X = x\}$. The entropy of Y given the event $\{X = x\}$ is defined as

$$H(Y|X = x) = -\sum_{y} p(y|x)\log_2 p(y|x).$$

Next, the conditional entropy $H\,(Y|X)$ is defined as

$$\begin{aligned} H(Y|X) &= \sum_{x} p(x) H\,(Y|X = x) \\ &= -\sum_{x}\sum_{y} p\,(y|x)\,p(x)\log_2 p\,(y|x)\,. \end{aligned}$$

Finally, the mutual information $I(X;Y)$ between X and Y is defined as

$$I(X;Y) = \sum_{x}\sum_{y} p\,(y|x)\,p(x)\log_2\frac{p\,(y|x)\,p(x)}{p(x)p(y)}. \tag{11.1}$$

It can be shown that $I(X;Y) = H(Y) - H(Y|X) = H(X) - H(X|Y)$. As such, $I(X;Y)$ intuitively represents the reduction of uncertainty about X due to the fact that we can observe Y (equivalently, reduction of uncertainty about Y due to the fact that we can observe X). The mutual information is well-defined also for continuous random variables. In this case, $p(x)$, $p(y)$, and $p\,(y|x)$ are probability density functions (pdfs), and we have

$$I(X;Y) = \int p\,(y|x)\,p(x)\log_2\frac{p\,(y|x)\,p(x)}{p(x)p(y)}dxdy. \tag{11.2}$$

Moreover, if X is a discrete random variable and Y is a continuous one, $I(X;Y)$ is defined as

$$I(X;Y) = \sum_{x} p(x)\int p\,(y|x)\log_2\frac{p\,(y|x)\,p(x)}{p(x)p(y)}dy. \tag{11.3}$$

11.1.2 System Model and Channel Capacity

The fundamental limit of point-to-point digital communication over a noisy channel was established in 1948 by C. Shannon, who showed that a vanishing error probability can be attained at a finite information rate, provided this rate is smaller than the *capacity* of the noisy channel.

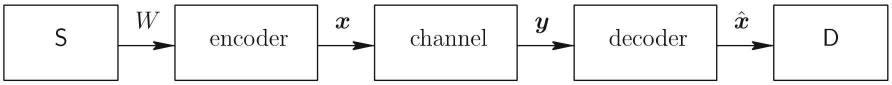


Fig. 11.1 Communication model

With reference to Fig. 11.1, a source S of information generates messages that must be delivered to a destination D through a noisy channel. The generic message, denoted by W, is drawn from a set of M possible messages $\{1, 2, \ldots, M\}$, where all messages are a priori equally likely. Prior to transmission over the channel, the message W is encoded through a *channel encoder,* that maps deterministically (and univocally) each message onto a *codeword* $\boldsymbol{x} = [x_0, x_1, \ldots, x_{n-1}]$, i.e., an n-tuple of symbols belonging to some alphabet $\mathcal{X}$. The ratio

$$R = \frac{\log_2 M}{n}$$

is the code rate of the channel code and the code is named an $(n, 2^{nR})$ code. All n codeword symbols are then transmitted sequentially over the channel, resulting in a sequence $\boldsymbol{y} = [y_0, y_1, \ldots, y_{n-1}]$ whose symbols belong to an alphabet $\mathcal{Y}$. A decoding algorithm is then performed by a *channel decoder* to decide which codeword, out of the set of M candidate codewords, had been transmitted over the channel, given the noisy observation $\boldsymbol{y}$. The codeword $\hat{\boldsymbol{x}}$ returned by the decoder is converted back to the corresponding message $\widehat{W}$ that is finally delivered to the destination. As error occurs whenever $W \neq \widehat{W}$, i.e., a wrong message is delivered.

A probability of error can be defined for each of the M transmitted messages as follows. The probability of error associated with the j-th message, $j \in \{1, 2, \ldots, M\}$, is denoted by $P_{e,j}$ and is defined as

$$P_{e,j} = \Pr\left\{ \widehat{W} \neq W | W = j \right\}.$$

Furthermore, the maximum probability of error is defined as

$$P_{e,\max} = \max_{j \in \{1,2,\ldots,M\}} P_{e,j} \tag{11.4}$$

and the average probability of error as

$$P_e = \frac{1}{M} \sum_{j=1}^{M} P_{e,j}\,. \tag{11.5}$$

The channel code along with its decoding algorithm shall be designed in order to make the maximum probability of error over the given channel as small as possible.

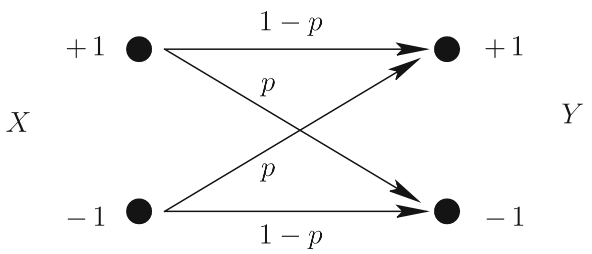


Fig. 11.2 Binary symmetric channel (BSC) model

Assume that both the input alphabet $\mathcal{X}$ and the output alphabet $\mathcal{Y}$ are discrete. Let $X \in \mathcal{X}$ and $Y \in \mathcal{Y}$ be two discrete random variables, representing the input to the channel and the corresponding output. Moreover, assume that the channel is fully defined by the transition probabilities $p(y|x) = \Pr\{Y = y|X = x\}$. In this case, the channel is called a discrete memory-less channel (DMC). The capacity of a DMC is defined as

$$C = \max_{p(x)} I(X;Y) \tag{11.6}$$

i.e., as the maximum amount of uncertainty we can remove from the input symbol (which cannot be observed directly) by observing the output symbol, where the maximum is taken over all possible pmfs for the input symbol. The capacity is an intrinsic parameter of the channel, only depending on the cardinalities of $\mathcal{X}$ and $\mathcal{Y}$ and on the transition probabilities $p(y|x)$. It is expressed in terms of information bits (or Shannon) per channel use.

Example 1. The DMC depicted in Fig. 11.2 is characterized by $\mathcal{X} = \mathcal{Y} = \{-1,+1\}$ and by $\Pr\{Y = +1|X = +1\} = \Pr\{Y = -1|X = -1\} = 1-p$, $\Pr\{Y = +1|X = -1\} = \Pr\{Y = -1|X = +1\} = p$, This channel is known as binary symmetric channel (BSC), and p is called the error (or crossover) probability. Every binary symbol input to the channel is received in error with probability p and is correctly received with probability $1-p$. The capacity of the BSC is achieved for $\Pr\{X = +1\} = \Pr\{X = -1\} = 1/2$ and is given by[1]

$$C = 1 - [-p\log_2 p - (1-p)\log_2(1-p)]\,. \tag{11.7}$$

As we shall see later, the BSC is a possible channel model for SLC Flash memories. Assuming $p \le 1/2$, its capacity is maximum for $p = 0$, where we have $C = 1$ (every binary symbol outcoming from the channel is reliable) and is minimum for $p = 1/2$, where we have $C = 0$ (no uncertainty is removed from X by observing Y).

The concept of capacity, so far introduced for a DMC, can be extended to time-discrete memory-less channels whose input symbol is either a discrete or a

[1]The capacity of the BSC only depends on the crossover probability and not on the values assumed by X and Y.

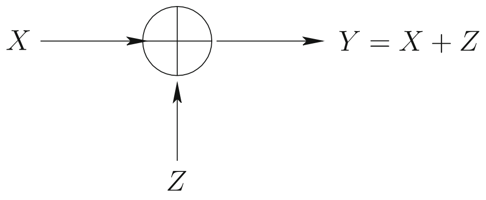


Fig. 11.3 Binary-input additive white Gaussian noise channel model

continuous random variable and whose output symbol is a continuous one. The capacity is still defined by Eq. (11.6), where the mutual information is now given by Eq. (11.2) if X is continuous, and by Eq. (11.3) if X is discrete. As opposed to the DMC case, however, additional constraints to the optimization problem may be introduced (for example, an upper bound on the average transmitted power). The reason is that the solution to the unconstrained optimization problem may correspond to an input variable X for which the channel is essentially noiseless.

Additive noise channels represent an important class of such channels. Here, the output symbol is obtained as $Y = X + Z$, where Z is a continuous random variable, namely, an additive noise. If Z is independent of X and is normally distributed with zero mean and variance σ^2,

$$p(z) = \frac{1}{\sqrt{2\pi\sigma^2}} e^{-\frac{z^2}{2\sigma^2}},$$

then the corresponding channel is called an additive Gaussian channel.

Example 2. Consider the additive Gaussian channel depicted in Fig. 11.3, and assume that $\boldsymbol{X}$ is a Bernoulli (i.e., discrete with a binary alphabet) random variable. Without any further constraint, it is possible to achieve the capacity $C = 1$ (corresponding to a noiseless channel) *regardless* of σ^2 by letting $X \in \{-A, +A\}$, where $A > 0$ is a real, choosing $\Pr\{X = -A\} = \Pr\{X = +A\} = 1/2$, and letting $A \to \infty$. On the other hand, if the maximization problem is constrained to $(1/n) \sum_{i=0}^{n-1} x_i^2 \le E_s$ for any transmitted codeword, then the maximum is attained for $X \in \{-\sqrt{E_s}, +\sqrt{E_s}\}$ and $\Pr\left\{X = -\sqrt{E_s}\right\} = \Pr\left\{X = +\sqrt{E_s}\right\} = 1/2$. In this case Eq. (11.3) yields

$$C = -\int p(y) \log_2 \left(p(y)\sqrt{2\pi e\sigma^2}\right) dy,$$

where

$$p(y) = \frac{1}{\sqrt{8\pi\sigma^2}} \left(e^{-\frac{(y-\sqrt{E_s})^2}{2\sigma^2}} + e^{-\frac{(y+\sqrt{E_s})^2}{2\sigma^2}}\right)$$

and where the capacity, that does not admit a closed-form expression, must be computed via numerical integration. This channel model is known as the binary-input additive white Gaussian noise (Bi-AWGN) channel. It is possible to show that its capacity is a function of parameter E_s/N_0, where $N_0 = 2\sigma^2$. In general, the larger E_s/N_0 the higher C. Moreover, $C \to 1$ as $E_s/N_0 \to \infty$.

Example 3. Consider a channel $X \to Y' \to Y$ composed of the cascade of a Bi-AWGN channel and a one-bit quantizer, returning $Y = +1$ if $Y^{'} > 0$ and $Y = -1$ otherwise (if $Y^{'} = 0$, $+1$ or -1 is returned with equal probability). It is readily shown that this channel is equivalent to a BSC whose crossover probability p is

$$p = \frac{1}{2}\text{erfc}\left(\sqrt{\frac{E_s}{N_0}}\right) \tag{11.8}$$

where

$$\text{erfc}(x) = \frac{2}{\sqrt{\pi}}\int_x^{\infty} e^{-\theta^2} d\theta.$$

Again, the capacity is a monotonically increasing function of parameter E_s/N_0, and again $C \to 1$ as $E_s/N_0 \to \infty$. For the same value of E_s/N_0, the capacity of the output-quantized Bi-AWGN channel is always smaller than the capacity of the corresponding unquantized channel.

With reference to the last two examples, if Y is allowed to assume $q > 2$ different quantized values (which corresponds to adopting $\log_2 q$ quantization bits), the capacity of the obtained channel is upper bounded by that of the unquantized Bi-AWGN channel and is lower bounded by that of the one-bit quantized channel. (Note that the $q - 1$ quantization thresholds shall be properly designed.) In general, the higher q the larger the capacity.

11.1.3 The Channel Coding Theorem

Adopting the formulation in [10], which makes use of the maximum error probability defined in Eq. (11.4), Shannon's channel coding theorem can be stated as follows. "For every rate $R < C$ there exists a sequence of $(n, 2^{nR})$ codes for which $\lim_{n\to\infty} P_{e,max}(n) = 0$. Conversely, if $\lim_{n\to\infty} P_{e,max}(n) = 0$ for a sequence of $(n, 2^{nR})$ codes, then $R \le C$." Note that $\lim_{n\to\infty} P_{e,max}(n) = 0$ implies $\lim_{n\to\infty} P_e(n) = 0$, where $P_e(n)$ is the average error probability defined in Eq. (11.5).

Essentially, Shannon's channel coding theorem states that communication over a noisy channel is possible with an arbitrarily small maximum error rate if and only if the code rate of the employed channel code does not exceed the channel capacity. On the other hand, from the proof of the converse, it is possible to show that, when $R > C$, the average probability of error probability is bounded away from zero. Specifically, we have

$$P_e(n) \ge 1 - \frac{C}{R} - \frac{1}{nR} \tag{11.9}$$

$$\to 1 - \frac{C}{R} \tag{11.10}$$

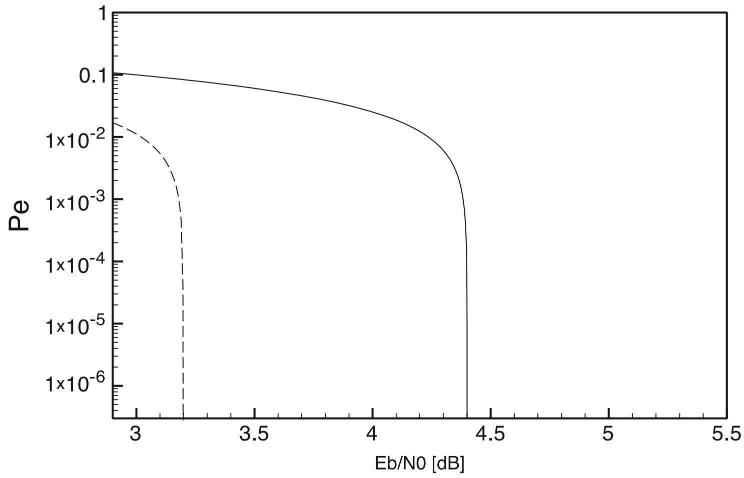


Fig. 11.4 Plot of the Shannon limit for code rate $R = 9/10$, over the Bi-AWGN channel and over the BSC obtained via one-bit quantization of the output of the Bi-AWGN channel

in the limit where $n \to \infty$. Eq. (11.9) defines a *non-achievable region* for the considered communication channel. No channel code of length n exists whose average probability of error over the considered channel is smaller than the right-hand side of Eq. (11.9). For $n \to \infty$, the non-achievable region is identified by Eq. (11.10). For a channel parametrized by some parameter γ (e.g., the crossover probability p for a BSC, or E_s/N_0 for the Bi-AWGN channel or its output-quantized version), the non-achievable region can be reported in the $P_e(n)$ versus γ plane for a specific code rate R, as illustrated in the following example.

Example 4. In Fig. 11.4 the non-achievable region is depicted for both the unquantized Bi-AWGN channel and its one-bit output-quantized version, for code rate $R = 9/10$ and infinite codeword length. Specifically, for fixed $R = 9/10$ the right-hand side of Eq. (11.10) is plotted as a function of E_b/N_0 (in logarithmic scale), where $E_b = RE_s$. If E_s is interpreted as the energy per transmitted binary symbol, E_b can be regarded as the energy per information bit. The dashed curve identifies a non-achievable region over the unquantized Bi-AWGN channel (i.e., no $(E_b/N_0, P_e)$ point inside the corresponding area is achievable), while the solid one a non-achievable region over its one-bit output-quantized version. The fact that the unquantized non-achievable region is contained in the quantized one is coherent with the fact that the capacity of the Bi-AWGN channel is larger than the capacity of its output-quantized version, for the same value of E_s/N_0. In general, if $q > 2$ quantization levels are allowed, the corresponding non-achievable region is identified by a curve falling between the two plotted curves. This serves to illustrate how soft information at the decoder can be exploited to improve the system performance. The smallest value of E_b/N_0 for which communication is possible with a vanishing error probability at the given rate $R = 9/10$ over the Bi-AWGN channel is about 3.198 dB. The corresponding value over the one-bit quantized Bi-AWGN channel is about 4.400 dB.

11.2 Maximum a Posteriori and Maximum Likelihood Decoding of Linear Block Codes

As from Sect. 11.1, decoding is essentially a decision problem. Given the observation $\boldsymbol{y}$ from the communication channel, the decoder has to decide which of the M codewords has been most likely transmitted, in order to minimize the maximum probability of error. Optimum decoding is based on *maximum a posteriori* (MAP) decision criterion, and consists of assuming as the transmitted codeword the one maximizing the a posteriori probability:

$$\hat{\boldsymbol{x}} = \text{argmax}_{\boldsymbol{x}}\, p\,(\boldsymbol{x}\,|\,\boldsymbol{y}).$$

When the codewords are a priori equally likely, then MAP decoding is equivalent to maximum likelihood (ML) decoding, that returns the codeword

$$\hat{\boldsymbol{x}} = \text{argmax}_{\boldsymbol{x}}\, p\,(\boldsymbol{y}\,|\boldsymbol{x})\,.$$

It is readily shown that, over a BSC, ML decoding is equivalent to returning the codeword exhibiting the minimum Hamming distance from the received word $\boldsymbol{y}$. (Recall that the Hamming distance between two sequences is the number of positions at which the corresponding symbols are different.) Moreover, over a Bi-AWGN channel, ML decoding consists of returning the codeword (whose symbols belong to the set $\{-\sqrt{E_s}, +\sqrt{E_s}\}$) exhibiting the minimum Euclidean distance from $\boldsymbol{y}$.

Optimum decoding is unfeasible for most codes (including linear codes), due to the need of computing M metrics, with M prohibitively large. Low-density parity-check codes, introduced in Sect. 11.10, are capable to perform close to the Shannon limit at a manageable complexity.

11.3 NAND Flash Memory Channel Model

In NAND Flash memories, the generic memory cell is a floating gate transistor. Writing the cell consists of exploiting Fowler-Nordheim tunneling effect [14] to inject a certain amount of charges into the floating gate in order to program the threshold voltage V_{th} of the transistor. For an MLC memory with b bits per cell, there are 2^b nominal values for threshold voltage V_{th}, each bijectively associated with a word of b bits. (There are two nominal values for V_{th} in the particular case of an SLC memory.) The whole range of possible values of V_{th} is then partitioned into 2^b intervals, each corresponding to a nominal value of the threshold voltage.

Reading a cell is a decision problem consisting of picking one of the 2^b nominal values of V_{th} and forwarding the corresponding binary b-tuple. The value of V_{th}, however, cannot be observed directly. In order to read the cell, a word-line voltage must be applied and the corresponding transistor drain current measured. In this chapter, we refer to the word-line voltage simply as the "read voltage", denoting it by V_{READ}. If for some V_{READ} a sufficiently high drain current is detected then we conclude that $V_{READ} > V_{th}$, otherwise we conclude that $V_{READ} < V_{th}$. In this sense, the application of a specific read voltage value is capable to provide exactly one bit of information. Therefore, in order to read the full content of a cell in an MLC memory the drain current must be analyzed for a sufficiently large number of read voltage values. A single V_{READ} value is sufficient in the SLC case unless we wish to extract some soft information to improve the performance of the adopted error control coding scheme.

In ideal Flash memories, after a cell is written the corresponding value of V_{th} is exactly equal to one of the 2^b nominal values. In real memories, however, the actual value of V_{th} may differ, even significantly, from its nominal value due to a number of possible physical impairments. For a thorough description of these impairments we refer the reader to Chaps. 6, 8 and, for example, to [21,Ch.4, 22]. As such, the actual value of V_{th} may fall into a voltage interval whose nominal voltage threshold is different from the one we attempted to set during the write operation. When this happens the forwarded binary b-tuple after a read operation differs from the one that was written into the cell. A bit error generated by an erroneous decision about the interval of voltage values V_{th} belongs to is called a *raw bit error*, and the probability of occurrence of raw bit errors is called the raw bit error probability.

The raw bit error probability may be analyzed by modeling the threshold voltage V_{th} of the generic cell as a continuous random variable whose pdf is here denoted by $p(V_{th})$. It must be pointed out that $p(V_{th})$ is not constant during the memory lifetime, as it is modified by subsequent write and read operations, leading to a progressive degradation of the channel in terms of increasing raw bit error probability. The threshold voltages for two different memory cells are typically assumed to be independent and identically distributed (i.i.d.) random variables. In the following two subsections, the channel model for SLC and MLC Flash memories is addressed.

11.3.1 SLC Channel Model

The simplest channel model for an SLC Flash memory consists of modeling the threshold voltage V_{th} of the generic cell as the weighted sum (with the same weights) of two independent Gaussian random variables with the same variance σ^2 neglecting that, in principle, Gaussian random variables assume their values over an infinite range. The mean values of the two Gaussian distributions are the two nominal values of the threshold voltage, namely,$V_{th,1}$ and $V_{th,2}$ where we assume $V_{th,1} < V_{th,2}$. Let $X \in \{0, 1\}$ be a Bernoulli random variable with equiprobable values, representing the bit originally written into the memory cell. Moreover, let Y be the symbol read from the cell. Conditionally to X, the threshold voltage V_{th} is a Gaussian random variable with variance σ^2 and whose mean is $V_{th,1}$ if $X = 1$ (erase state) and $V_{th,2}$ if $X = 0$. This is depicted in Fig. 11.5. Overall, we have

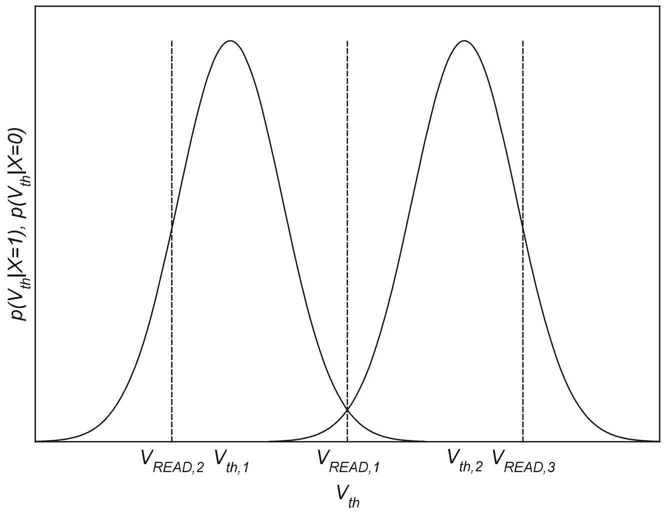


Fig. 11.5 Plot of $p(V_{th}|X = 1)$ and $p(V_{th}|X = 0)$ for an SLC Flash memory where the threshold voltage V_{th} is modeled as the sum of two independent and identically distributed (i.i.d.) Gaussian random variables

$$\begin{aligned} p(V_{th}) &= \frac{1}{2} p\left(V_{th}|X = 1\right) + \frac{1}{2} p\left(V_{th}|X = 0\right) \\ &= \frac{1}{\sqrt{8\pi\sigma^2}} \left(e^{-\frac{(V_{th}-V_{th,1})^2}{2\sigma^2}} + e^{-\frac{(V_{th}-V_{th,2})^2}{2\sigma^2}} \right). \end{aligned}$$

If we apply only one read voltage $V_{th,1} < V_{READ,1} < V_{th,2}$w we get information about the actual value of V_{th} being larger or smaller than the applied read voltage value. Hence, if only one read voltage value is used, Y is a Bernoulli random variable as well as X. In particular, we have $Y = 1$ if $V_{th} < V_{READ}$ is detected, and $Y = 0$ otherwise. A raw bit error occurs any time $Y \neq X$, and the raw bit error probability is trivially minimized by setting $V_{READ,1} = (V_{th,1} + V_{th,2})/2$, as depicted in Fig. 11.5. In this situation, the channel is clearly equivalent to the cascade of a Bi-AWGN channel and a one-bit quantizer described in Example 2 (i.e., to a BSC), and the raw bit error probability is given by Eq. (11.8) where $E_s/N_0 = (V_{th,2} - V_{READ,1})^2/2\sigma^2$. At the beginning of the memory life, σ^2 is very small and the memory is almost ideal. Then, σ^2 increases with the memory use, increasing the raw error probability and degrading the channel. A typical value of the raw bit error probability towards the end of the memory life is 10^{-2}.

If an error correcting code is employed to protect the data stored in the Flash memory, hard-decision decoding must be necessarily performed if only one V_{READ} value is used as no soft information is available at the decoder. As it will be shown in Sect. 11.7, however, the availability of soft information at the decoder

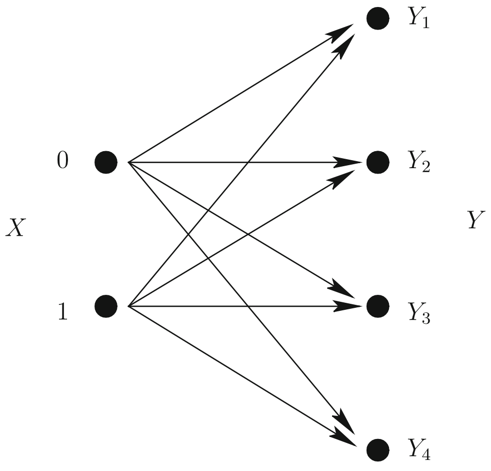


Fig. 11.6 Equivalent channel model for an SLC Flash memory where the threshold voltage is modeled as the sum of two i.i.d. Gaussian random variables and where three read voltage values are employed. Each read operation involves two read voltages

input represents an essential feature to boost the performance of the coding scheme. In order to provide the decoder with soft information, and consequently to increase its coding gain, more read voltages must be applied sequentially. For example, with reference again to Fig. 11.5 we may employ three read voltage values $V_{READ,1}$, $V_{READ,2}$, and $V_{READ,3}$ and apply two of them for each cell read operation. Specifically, $V_{READ,1}$ is applied at first. if $V_{th} < V_{READ,1}$ then $V_{READ,2}$ is applied to discriminate between $V_{th} < V_{READ,2}$ and $V_{READ,2} < V_{th} < V_{READ,1}$. On the contrary, $V_{READ,3}$ is applied to discriminate between $V_{th} > V_{READ,3}$ and $V_{READ,1} < V_{th} < V_{READ,3}$. In this case the output symbol Y is a discrete random variable assuming the four possible values in the set $\{Y_1, Y_2, Y_3, Y_4\}$ and the channel may be represented as the DMC depicted in Fig. 11.6.

Each arrow in the depicted DMC is associated with a transition probability $p\,(y|x)$, where the transition probabilities depend on the choice of the read voltages $V_{READ,2}$ and $V_{READ,3}$. A "natural" approach to choose them consists of maximizing the mutual information between the random variables X and Y under the setting $\Pr(X = 0) = \Pr(X = 1) = 1/2$. This approach, proposed in [32], may be easily extended to any number of read voltages. It may also be easily extended to different choices of the pdf $p(V_{th})$, and therefore to MLC Flash memories.

11.3.2 MLC Channel Model

While the channel model for SLC Flash memories is rather well-established, the development of an MLC channel model is still a subject of research and measurement campaigns, and several models may be found in the literature. These models typically assume the random variable V_{th} to be the weighted sum (with the

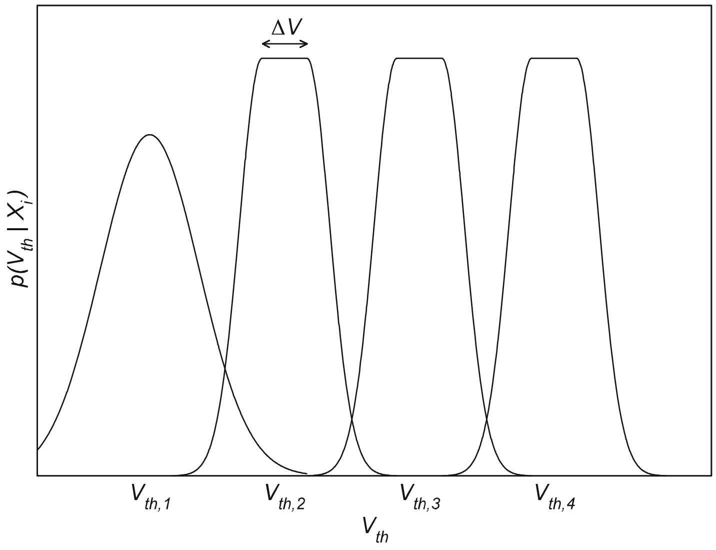


Fig. 11.7 Representation of the four conditional probability density functions $p\,(V_{th}|\boldsymbol{X}_i)$ of the threshold voltage in an MLC Flash memory with $b = 2$ bits per cell

same weights) of 2^b independent random variables, each corresponding to a nominal value of the threshold voltage. Among these models, the one described next has been adopted in several works [17]. Letting $\boldsymbol{X}$ denote the binary b-tuple that was written in the cell, the pdf $p(V_{th}|\boldsymbol{X}_1 = 11\ldots1)$ associated with the lowest nominal threshold voltage value $V_{th,1}$ (erase state) is modeled as Gaussian with mean $V_{th,1}$ and variance σ_0^2, while the pdf $p(V_{th}|\boldsymbol{X}_i)$ associated with any other nominal value $V_{th,i}$ ($\boldsymbol{X}_i \neq 11\ldots1$) is characterized by a uniform central region of size ΔV centered in the mean value $V_{th,i}$ and by two Gaussian tails of variance $\sigma^2 < \sigma_0^2$. Formally, for $i \in \{2, 3, \ldots, 2^b\}$ we have

$$p\,(V_{th}|\boldsymbol{X}_i) = \begin{cases} \frac{1}{\sqrt{2\pi\sigma^2}+\Delta V} e^{-\frac{(V_{th}-V_{th,i}-\Delta V/2)^2}{2\sigma^2}} & V_{th} > V_{th,1} + \frac{\Delta V}{2} \\ \frac{1}{\sqrt{2\pi\sigma^2}+\Delta V} & V_{th,1} - \frac{\Delta V}{2} < V_{th} < V_{th,1} + \frac{\Delta V}{2} \\ \frac{1}{\sqrt{2\pi\sigma^2}+\Delta V} e^{-\frac{(V_{th}-V_{th,i}+\Delta V/2)^2}{2\sigma^2}} & V_{th} < V_{th,1} - \frac{\Delta V}{2} \end{cases}$$

and

$$p(V_{th}) = \frac{1}{2^b} \sum_{i=1}^{2^b} p\,(V_{th}|\boldsymbol{X}_i)\,.$$

A pictorial representation of the four conditional pdfs $p\,(V_{th}|X_i)$, $i \in \{1, 2, 3, 4\}$, for an MLC Flash memory with $b = 2$ bits per cell and equally spaced threshold voltages is shown in Fig. 11.7.

In an analogous way as for the SLC case, a read is performed by applying sequentially a certain number of read voltages V_{READ} in order to identify the interval in which the actual value of the threshold voltage belongs. If $N \geq 2^b - 1$ different read voltages are employed, the equivalent communication channel is a DMC with 2^b equiprobable input symbols $\boldsymbol{X}$ and $N+1$ output symbols $\boldsymbol{Y}$. Again, the larger the number of employed read voltages (i.e., the larger the number of intervals in which the range of possible V_{th} values is partitioned) the more accurate the soft information at the decoder input, the lower the bit error rate after decoding. Again, the values of the N read voltages must be properly designed, for instance, maximizing the mutual information $I(\boldsymbol{X};\boldsymbol{Y})$ under the assumption $\Pr(\boldsymbol{X} = \boldsymbol{X}_i) = 2^{-b}$ for all $i \in \{1, 2, \ldots, 2^b\}$.

11.4 Low-Density Parity-Check Codes

Low-density parity-check (LDPC) codes were introduced by R. Gallager in [15] and have been almost forgotten for about 30 years. They gained a new interest only after the discovery of turbo codes [2], when it was shown that iterative decoding schemes can attain performances very close to the Shannon limit with a manageable complexity [3, 26].

A binary LDPC code is defined as a binary linear block code whose parity-check matrix $\boldsymbol{H}$ is characterized by a relatively small number of 1 entries, i.e., whose parity-check matrix is sparse. LDPC codes are often represented graphically through a bipartite graph $G = (\mathcal{V} \cup \mathcal{C}, \mathcal{E})$ called the Tanner graph [28]. In the Tanner graph there are two different types of nodes, namely, the variable nodes (whose set is $\mathcal{V}$) and the check nodes (whose set is $\mathcal{C}$). The n variable nodes and the m check nodes are associated in a bijective way with the n encoded bits of the generic codeword and with the m parity-check equations, respectively. Each edge $e \in \mathcal{E}$ in the Tanner graph connects a variable node $V \in \mathcal{V}$ with a check node $C \in \mathcal{C}$ if and only if the bit corresponding to V is involved in the parity-check equation corresponding to C. Note that in general not all the m parity-check equations may be linearly independent, so that the actual code rate R of the LDPC code fulfills

$$R \geq \frac{n-m}{n}$$

where equality holds when all m equations are independent. In the Tanner graph of an LDPC code a cycle (or loop) is any closed path starting from a node and ending on the same node. The length of a cycle is the number of edges involved in the cycle. Moreover, the girth g of the Tanner graph is the length of its shortest loop. For reasons that will be clear in the next section, the Tanner graph of an LDPC code should exhibit a large girth. In the Tanner graph, the degree of a variable node or check node is the number of edges incident to it. An LDPC code is said to be regular if all of its variable nodes have the same degree and all of its check nodes have the same degree, and is said to be irregular otherwise.

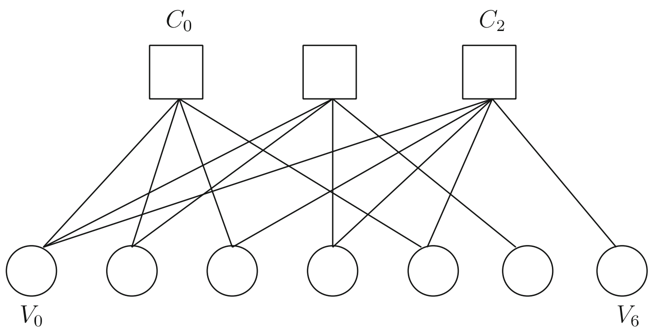


Fig. 11.8 Tanner graph of a (7, 4) Hamming code represented by $H = [1110100, 1101010, 10111001]$. There are seven variable nodes variable nodes $\{V_0, \dots, V_6\}$, one for each encoded bit, and three check nodes, $\{C_0, \dots, C_2\}$, one for each parity-check equation

The representation of LDPC codes in terms of their Tanner graphs is very convenient in order to describe their iterative decoding algorithm, known as belief propagation (BP). In fact, as it will be addressed in Sect. 11.5, BP decoding of LDPC codes may be interpreted as an iterative exchange of messages between the variable nodes and the check nodes along the edges of the Tanner graph. In principle, the Tanner graph can be drawn for any $\boldsymbol{H}$ matrix of any linear block code. As an example, in Fig. 11.8 the Tanner graph is depicted for the (7, 4) Hamming code represented by $\boldsymbol{H} = [1110100, 1101010, 10111001]$.

In this section we provide a few details about binary LDPC code design, while LDPC decoding is discussed in the next section. One of the major issues in LDPC coding is represented by efficient encoding, i.e., the efficient computation of the encoded codeword of n bits from a message W represented by a binary k-tuple. Hence, we focus on the design of quasi-cyclic LDPC (QC-LDPC) codes based on circulant matrices, a class of LDPC codes characterized by low-complexity encoding and good performances [13]. In general, a linear block code is said to be quasi-cyclic when there exists some positive integer q such that a cyclic shift by q positions of any codeword results in another codeword. The encoder of QC-LDPC codes may be implemented very efficiently in hardware using shift register-based circuits [18]. Efficient hardware implementations for the decoder are also available [20].

11.4.1 LDPC Code Ensembles

As opposed to classical algebraic codes, LDPC codes are typically analyzed in terms of average ensemble properties, where an LDPC code ensemble is formed by all LDPC codes having the same codeword length n and nominally the same rate R, and sharing common properties. This approach was introduced by Gallager to analyze his regular LDPC codes [15], and has been successfully adopted to design irregular LDPC codes performing very close to the Shannon limit [24, 9].

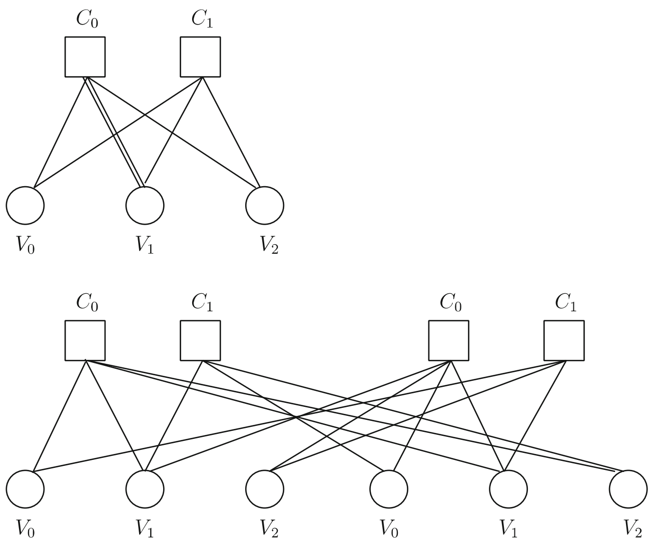


Fig. 11.9 Conceptual example of copy-and-permute protograph procedure

An example of LDPC code ensemble is the *unstructured* irregular one [24]. Let n and m be the numbers of variable and check nodes, respectively. Moreover, let Λ_i and P_i be the fractions of variable nodes and check nodes of degree i, respectively. Hence, in the Tanner graph there are $\Lambda_i n$ variable nodes with i sockets and $\mathrm{P}_i m$ check nodes with i sockets and the number of edges is $E = n \sum_{i=2}^{D} i\Lambda_i = m \sum_{i=2}^{H} i\mathrm{P}_i$ where D is the maximum variable node degree and H the maximum check node degree. For given Λ_i, $i = 2, \ldots, D$ and P_i, $i = 2, \ldots, H$,[2] the unstructured $\mathcal{C}(n, \Lambda, \mathrm{P})$ ensemble includes all LDPC codes corresponding to all possible $E!$ edge permutations between the variable node and the check node sockets, according to a uniform probability distribution.

Another example is the *protograph* ensemble [30] (see also the work [31] on LDPC codes from superposition). A protograph is defined as a small Tanner graph and represents the starting point to derive a larger Tanner graph via a "copy-and-permute" procedure. Specifically, the protograph is first copied Q times. Then, the edges of the individual replicas are permuted among the replicas, leading to a larger graph. The edge permutation is performed in such a way that, if an edge e connects a variable node V to a check node C in the protograph, then in the final graph any of the Q replicas of e may connect only a replica of V to a replica of C. Note that, while parallel edges between nodes are allowed in the protograph, they are avoided in the permutation phase. An example of this copy-and-permute procedure is depicted in Fig. 11.9. For a given protograph and a given Q the ensemble is composed of the

[2]For unstructured ensemble, the minimum variable and check nodes are usually set to 2. The reason for this choice is out of the scope of this chapter.

LDPC codes corresponding to all possible edge permutations fulfilling the described constraints (again, the probability distribution over such permutations is uniform).

11.4.2 QC-LDPC Codes Construction

A very popular technique to design finite length LDPC codes consists of two subsequent steps. An ensemble of LDPC codes with desired properties is first designed and then a code from the ensemble is picked constructing its Tanner graph according to some graph-lifting algorithm. In the first design phase (ensemble optimization) *asymptotic ensembles* are considered, i.e., ensembles of LDPC codes whose codeword length tends to infinity (examples are the unstructured $\mathcal{C}(\infty, \Lambda, \mathrm{P})$ ensemble and the protograph ensemble defined by a specific finite-length protograph in the limit where $Q \to \infty$). The main parameter characterizing an asymptotic ensemble of LDPC codes under iterative decoding is the *asymptotic decoding threshold* [25, 24]. Letting ℓ be the iteration index and assuming that the communication channel is parameterized by some real parameter θ such that $\theta_1 < \theta_2$ means that the channel corresponding to θ_2 is a degraded version of the channel corresponding to θ_1, the asymptotic threshold θ^* is defined as

$$\theta^* = \sup \left\{ \theta \text{ s.t.} P_{e,\ell}^{\infty} \to 0 \text{ as } \ell \to \infty \right\}$$

where $P_{e,\ell}^{\infty}$ is the average error probability under iterative decoding over the asymptotic ensemble (i.e., the expected probability of error for an LDPC code randomly picked in the asymptotic ensemble). For example, over a BSC the parameter θ is the crossover probability p, while over a Bi-AWGN channel it is the noise power σ^2 for given E_s (therefore over the Bi-AWGN channel the threshold may be expressed as $(E_b/N_0)^*$ where $E_b = RE_s$ and R is the nominal ensemble rate). Note that for the same ensemble, the threshold is different for different message passing decoders. For unstructured ensembles the threshold may be calculated exactly via a procedure called *density evolution* [24] or approximately via a tool known as EXIT chart [29]. For protograph ensembles it may be calculated with good approximation via multi-dimensional EXIT analysis [19]. In Sect. 11.6.2 density evolution is reviewed for unstructured regular LDPC ensembles and for a very simple decoder called the Gallager B decoder.

Once a protograph ensemble with a satisfying threshold over the channel of interest has been designed, a QC-LDPC code can be constructed from the protograph. This step is usually performed by first representing the protograph as a *base matrix* $\boldsymbol{B}$. The number of rows and columns in the base matrix equal the number of check and variable nodes in the protograph, respectively. Moreover, the (j, i) −th entry of $\boldsymbol{B}$ is equal to the number of connections between check node C_j and variable node V_i in the protograph. For example, the base matrix corresponding to the protograph depicted in Fig. 11.9 is

$$\boldsymbol{B} = \begin{bmatrix} 1 & 2 & 1 \\ 1 & 1 & 1 \end{bmatrix}.$$

In order to construct the parity-check matrix $\boldsymbol{H}$ of a QC-LDPC code from $\boldsymbol{B}$, each entry in the base matrix is replaced with a $Q \times Q$ circulant matrix, where a circulant matrix is any square matrix such that every row is obtained from the previous row by a cyclic shift to the right by one position. An entry in $\boldsymbol{B}$ equal to t is replaced by a circulant matrix whose rows and columns all have Hamming weight t. (Null entries in B are replaced by zero $Q \times Q$ square matrices.) If the number of variable nodes in the protograph is n_p then the final LDPC code has length Qn_p. Moreover, it is a QC-LDPC code as the cyclic shift of any codeword by n_p positions results in another codeword. The specific circulant matrices used to replace the entries of the base matrix are chosen according to algorithms aimed at increasing the girth g of the graph, making it suitable to iterative message-passing decoding. It is pointed out that sometimes the parity-check matrix $\boldsymbol{H}$ is obtained by lifting the base matrix in several steps. For example, instead of replacing each entry of $\boldsymbol{B}$ by a $Q \times Q$ matrix (for large Q), $\tilde{Q} \times \tilde{Q}$ circulant matrices may be used at first, with Q being a multiple of $\tilde{Q}$, and then circulant permutation matrices of size $Q/\tilde{Q}$ may replace each entry in the "intermediate" matrix.[3]

11.4.3 Error Floor

Finite length LDPC codes are affected by a phenomenon known as the "error floor" [8, 23]. Considering again a communication channel parameterized by a real parameter θ indicating the level of channel noise, the error floor consists of a sudden reduction in the slope of the LDPC code performance curve when θ becomes lower than some value. For example, over the BSC the error floor appears at sufficiently low values of the error probability p, while over the Bi-AWGN channel it appears at sufficiently high values of E_b/N_0. An example performance curve in term of bit error rate (BER) vs. E_b/N_0 exhibiting an error floor is depicted in Fig. 11.10. In NAND Flash memories applications, very pressing requirements are usually imposed on the error floor. More specifically, it is often required that the error floor must not appear above page error rate (i.e., codeword error rate) 10^{-15}.

The error floor of LDPC codes under belief propagation decoding is mainly due to graphical structures in the Tanner graph called *trapping sets* [23]. Given a subset $\mathcal{W}$ of the variable nodes, the subgraph induced by $\mathcal{W}$ is the bipartite graph composed of $\mathcal{W}$, of the subset $\mathcal{U}$ of check nodes connected to $\mathcal{W}$ and of the corresponding edges. By definition, an (a, b) trapping set is any size-a subset $\mathcal{W}$ of the variable nodes, such that there are exactly b check nodes of odd degree (an arbitrary number of check nodes of even degree) in the corresponding induced

[3]The described protograph-based technique is not the only one to construct good QC-LDPC codes. Another possible approach is based on Euclidean and projective finite geometries [5, 27].

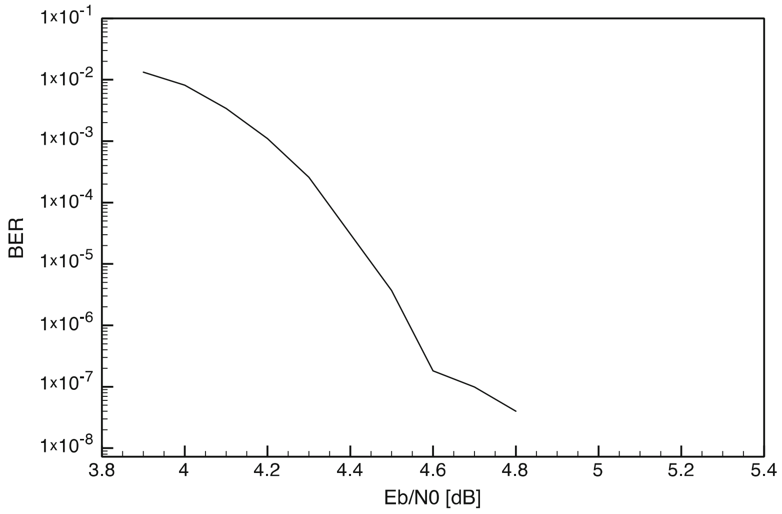


Fig. 11.10 Performance curve (in terms of BER vs. E_b/N_0) exhibiting an error floor at BER $\approx 10^{-7}$ ($E_b/N_0 > 4.6$ dB)

subgraph. The parameter a is called the size of the trapping set. If there are only degree-1 and degree-2 check nodes in the induced subgraph, then the trapping set is said to be *elementary*. Elementary trapping sets of small size are a major cause of error floor for iteratively decoded LDPC codes. We point out that small weight codewords may also contribute to the error floor together with trapping sets.

The need to construct LDPC codes characterized by very low error floors imposes some modifications to the QC-LDPC code design procedure described in the previous subsection, which becomes more involved. The asymptotic decoding threshold is not the only metric to be taken into account during the ensemble optimization phase, as other asymptotic parameters such as the typical relative minimum distance or smallest trapping set size must be considered [1, 12]. We also point out that reliable error floor analysis at very low error rates of LDPC codes for storage applications still represents an open issue. In fact, Monte Carlo software simulation is not feasible at very low error rates because of prohibitively long simulation times. Approaches proposed in the literature are hardware simulation, importance sampling [17, 4], and estimation techniques [11].

11.5 Belief Propagation (BP) Decoding of LDPC Codes

11.5.1 Introduction

As opposed to MAP and ML decoding algorithms (Sect. 11.2), that are block-wise algorithms, BP is a *bit-wise* decoding algorithm, working iteratively. More specifically, at the end of each decoding iteration a separate decision is taken about each bit in the codeword, and then it is checked whether the currently decoded hard-decision sequence is a codeword or it is not. Letting $\boldsymbol{y} = [y_0, y_1, \ldots, y_{n-1}]$ denote the sequence outcoming from the communication channel, the decision about encoded bit c_i, $i = 0, \ldots, n-1$, is taken according to its a posteriori likelihood ratio (LR), namely,

$$L(c_i|\boldsymbol{y}) = \frac{\Pr(c_i = 0|\boldsymbol{y})}{\Pr(c_i = 1|\boldsymbol{y})} \underset{\hat{c}_i = 1}{\overset{\hat{c}_i = 0}{\gtrless}} 1.$$

Unfortunately, the only information available at variable node i at the beginning of the decoding process is the a priori LR

$$L(c_i|y_i) = \frac{\Pr(c_i = 0|y_i)}{\Pr(c_i = 1|y_i)}$$

i.e., the LR conditioned only to the local observation, not the a posteriori LR $L(c_i|\boldsymbol{y})$ as required. Indeed, the task of the BP decoder consists of calculating the a posteriori LR for each variable node, starting from the individual a priori LRs, exploiting an iterative exchange of information among the nodes of the bipartite graph. In the following description of the BP decoder, we will not make any assumption on the communication channel, but that the channel is memory-less with binary input and equally likely input values.

11.5.2 Preliminaries

We start with some preliminary material that will be useful to properly describe BP decoding of LDPC codes.

Let us consider a Bernoulli random variable B taking the values 0 and 1 with equal probabilities. As depicted in Fig. 11.11, assume that N random experiments are performed to get information about the value assumed by B and that all these experiments are independent. The outcome of the n-th experiment (n-th observation) is denoted by ω_n, while the vector of N observables by $\boldsymbol{\omega} = [\omega_1, \omega_2, \ldots, \omega_N]$. We define the likelihood ratio (LR) of B conditioned to the observation ω_n as

$$L(B|\omega_n) = \frac{\Pr(B = 0|\omega_n)}{\Pr(B = 1|\omega_n)} \tag{11.11}$$

and the *a posteriori* likelihood ratio of B (i.e., conditioned to the whole set of N independent observations), as

$$L(B|\boldsymbol{\omega}) = \frac{\Pr(B = 0|\boldsymbol{\omega})}{\Pr(B = 1|\boldsymbol{\omega})}. \tag{11.12}$$

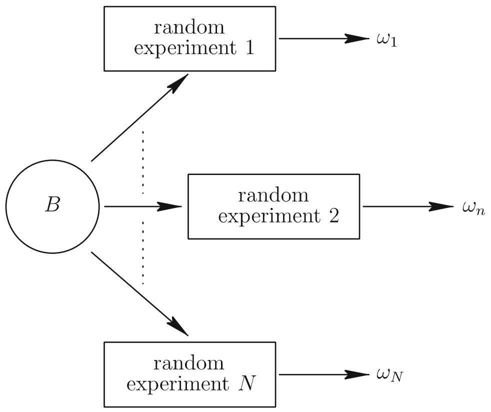


Fig. 11.11 N random experiments are conducted to obtain some information about the value of a Bernoulli random variable B. The observation associated with the n-th random experiment is ω_n

We now seek for an expression of the a posteriori LR, $L(B|\boldsymbol{\omega})$, as a function of the individual LRs, each conditioned to a specific observation. By Bayes rule we have

$$L\left(B|\boldsymbol{\omega}\right) = \frac{p\left(\boldsymbol{\omega}|B=0\right)}{p\left(\boldsymbol{\omega}|B=1\right)}$$
$$= \prod_{n=1}^{N} \frac{p\left(\omega_n|B=0\right)}{p\left(\omega_n|B=1\right)}$$
$$= \prod_{n=1}^{N} L\left(B|\omega_n\right), \tag{11.13}$$

where the second equality follows from independence of the random experiments.

We also observe that, through Eq. (11.12) and the relationship $\Pr\left(B=0|\boldsymbol{\omega}\right) + \Pr\left(B=1|\boldsymbol{\omega}\right) = 1$, the probabilities $\Pr\left(B=0|\boldsymbol{\omega}\right)$ and $\Pr\left(B=1|\boldsymbol{\omega}\right)$ may be expressed as functions of the a posteriori LR as follows:

$$\Pr\left(B=0|\boldsymbol{\omega}\right) = \frac{L\left(B|\boldsymbol{\omega}\right)}{1+L\left(B|\boldsymbol{\omega}\right)}, \tag{11.14}$$

$$\Pr\left(B=1|\boldsymbol{\omega}\right) = \frac{1}{1+L\left(B|\boldsymbol{\omega}\right)}. \tag{11.15}$$

This is sometimes referred to as *soft bit*. Analogous relationships may be derived for $\Pr(B=0|\omega_n)$ and $\Pr(B=1|\omega_n)$.

Next, consider n statistically independent Bernoulli random variables $B_1, B_2, \ldots, B_n$ each taking its value in $\{0, 1\}$. We allow $\Pr\left(B_k=1\right) \neq \Pr\left(B_l=1\right)$ if $k \neq l$. We ask what is the probability that the n variables sum to 0 (in binary algebra), i.e., the probability that an even number of such random variables take value 1. This problem was solved in [15], where it was shown that

$$\Pr(B_1 + B_2 + \ldots + B_n = 0) = \frac{1 + \prod_{\mathrm{k}=1}^{n} (1 - 2\Pr(B_k = 1))}{2}. \tag{11.16}$$

Consider now n Bernoulli random variables $B_1, B_2, \ldots, B_n$ fulfilling a parity constraint $B_1 + B_2 + \ldots + B_n = 0$. Moreover, assume that some reliability information is known about variables $B_1, \ldots, B_{i-1}, B_{i+1}, \ldots, B_n$, in terms of LRs $L(B_k)$, $k \in \{1, \ldots, i-1, i+1, \ldots, n\}$ and that $B_1, \ldots, B_{i-1}, B_{i+1}, \ldots, B_n$ are statistically independent. We seek for an expression of the LR $L(B_i)$, conditional on all available information about the other $n-1$ variables. Since $\Pr(B_i = 0) = \Pr(B_1 + \ldots B_{i-1} + B_{i+1} + \ldots + B_n = 0)$, through Eq. (11.16) we obtain

$$\begin{aligned}&\Pr(B_i = 0 | L(B_1), \ldots, L(B_{i-1}), L(B_{i+1}), \ldots, L(B_n))\\ &= \frac{1 + \prod_{\mathrm{k} \neq \mathrm{i}} (1 - 2\Pr(B_k = 1))}{2}\end{aligned}$$

and, consequently,

$$\begin{aligned}&\Pr(B_i = 1 | L(B_1), \ldots, L(B_{i-1}), L(B_{i+1}), \ldots, L(B_n))\\ &= \frac{1 - \prod_{\mathrm{k} \neq \mathrm{i}} (1 - 2\Pr(B_k = 1))}{2}.\end{aligned}$$

Note that each term $\Pr(B_k = 1)$ involved in the multiplication may be expressed in terms of the corresponding $L(B_k)$ through Eq. (11.15). From the term-by-term ratio between these two latter equations, we obtain

$$\mathrm{L}(B_i | L(B_1), \ldots, L(B_{i-1}), L(B_{i+1}), \ldots, L(B_n)) = \frac{1 + \prod_{\mathrm{k} \neq \mathrm{i}} (1 - 2\Pr(B_k = 1))}{1 - \prod_{\mathrm{k} \neq \mathrm{i}} (1 - 2\Pr(B_k = 1))}.$$

Through Eq. (11.15), after a few calculations this leads to

$$\mathrm{L}(B_i | L(B_1), \ldots, L(B_{i-1}), L(B_{i+1}), \ldots, L(B_n)) = \frac{\prod_{k \neq i} \frac{L(B_k)+1}{L(B_k)-1} + 1}{\prod_{k \neq i} \frac{L(B_k)+1}{L(B_k)-1} - 1}. \tag{11.17}$$

11.5.3 Algorithm Description

11.5.3.1 Overview

For ease of presentation, in the description of the algorithm we omit the decoding iteration index. We denote by r_i^j the message sent by variable node V_i, $i = 0, \ldots, n-1$, to check node C_j, $j = 0, \ldots, m-1$ during the current iteration,

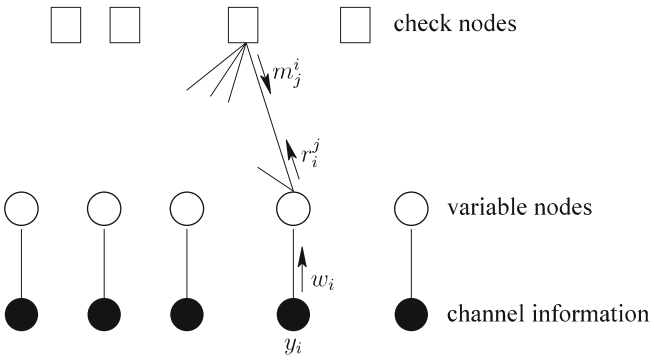


Fig. 11.12 Tanner graph of an LDPC code. The message sent by variable node V_i to check node C_j and the message sent by check node C_j to variable node V_i are denoted by r_i^j and m_j^i, respectively

and by m_j^i the message sent back by check node C_j, to variable node V_i, during the same iteration. For $i = 0, \ldots, n-1$, we also denote by w_i the a priori LR for variable node V_i, i.e.,

$$w_i = \frac{\Pr(c_i = 0|y_i)}{\Pr(c_i = 1|y_i)}.$$

This is illustrated in Fig. 11.12.

Belief-propagation decoding is composed of four steps, namely[4]:

- initialization;
- horizontal step;
- vertical step;
- hard decision and stopping criterion step.

Out of them, the initialization step is executed only once, at the beginning of decoding. The other three steps are executed iteratively, until a termination condition is verified or a maximum number of iterations, denoted by I_{max}, is reached. Each decoding iteration is split into two half-iterations. During the first half-iteration (horizontal step), check nodes process messages incoming from their neighboring variable nodes. Then, each check node sends one message along every edge incident on it. Thus, every check node sends one message per iteration to each of its neighboring variable nodes. During the second half-iteration (vertical step) variable nodes process messages incoming from their neighboring check nodes. Similar to the previous half-iteration, at the end of this processing each variable node sends one message along each edge incident on it. Thus, every variable node sends one

[4]The words "horizontal" and "vertical" remind us that the check nodes and the variable nodes are associated with the rows and the columns of the parity-check matrix, respectively.

message per iteration to each of its neighboring check nodes. At the end of the two half-iterations, a hard decision is taken in each variable node, about the value of the corresponding encoded bit.

The message transmitted by check node C_j, $j = 0, \dots, m-1$, to variable node V_i, $i = 0, \dots, n-1$, where V_i belongs to the neighborhood of C_j, may be interpreted as the best estimate C_j has about the value of V_i up to the current iteration. This is the estimate of the value of V_i given all information about V_i the check node has got from the variable nodes connected to it *other than* V_i. This is known as *extrinsic information*. Analogously, the message sent back by variable node V_i to check node C_j may be interpreted as the best estimate V_i has about itself up to the current iteration. This is the estimate of its value given all information the variable node has got from the communication channel and from the check nodes connected to it *other than* C_j (extrinsic information). All messages exchanged between variable nodes and check nodes are LRs or, equivalently, soft bits.

At the end of the vertical step, each variable node takes a hard decision about the value of its associated bit, based on the a priori information incoming from the channel and on all estimates incoming from the check nodes connected to it. If the obtained hard-decision binary sequence $\hat{\boldsymbol{c}}$ is a codeword of the LDPC code, i.e., if every check node is connected to an even number of variable nodes whose current estimate is 1, then a decoding success is declared, decoding is terminated, and $\hat{\boldsymbol{c}}$ is returned as the decoded codeword. Otherwise, a new iteration is started, unless the maximum number of iterations has been reached. In this latter case, no codeword has been found and a decoding failure is declared. LDPC codes decoded via belief propagation are then characterized by two different error events: detected errors and undetected errors. A detected error takes place whenever no codeword is found up to the maximum number of iterations. An undetected error takes place whenever, at some iteration, the hard-decision sequence $\hat{c}$ is a codeword but not the transmitted one. Undetected errors may be extremely dangerous in some contexts, including NAND Flash memories (Chap. 10).

11.5.3.2 Initialization

At the beginning, each variable node broadcasts to all its neighboring check nodes the a priori LR received from the communication channel. Hence, we have

$$r_i^j = w_i$$

for all $j \in N(i)$, where $N(i)$ is the set of indexes of check nodes connected to V_i. The expression of w_i depends on the nature of the channel. For example, it is easy to check that over a BSC with error probability p and antipodal mapping $x_i = 1 - 2c_i \in \{-1, +1\}$, we have

$$w_i = \begin{cases} \dfrac{1-p}{p} & \text{if} \quad y_i = +1 \\ \dfrac{p}{1-p} & \text{if} \quad y_i = -1. \end{cases} \tag{11.18}$$

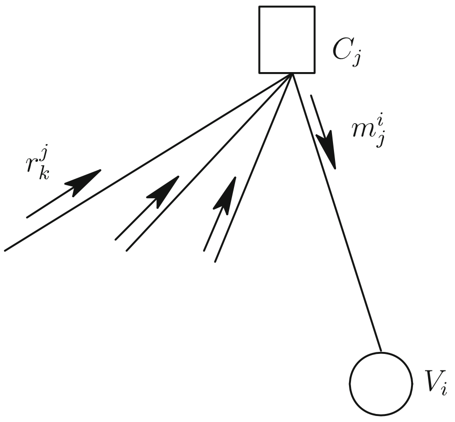


Fig. 11.13 Check node processing of incoming messages during the horizontal step

As another example, over a Bi-AWGN channel and again antipodal mapping $x_i = 1 - 2c_i$, (meaning E_s normalized to 1) we have

$$w_i = e^{(2/\sigma^2)y_i}. \tag{11.19}$$

Importantly, the initialization step requires a knowledge of the channel. For instance, in the case of a BSC the error probability p must be known, as well as the noise power σ^2 in the Bi-AWGN case.

11.5.3.3 Horizontal Step

For $j = 0, \ldots, m-1$, check node C_j, of degree h_j, sends to each of the h_j variable nodes connected to it its current estimate of the corresponding bit. If variable node V_i is connected to C_j, the message from C_j to V_i is the LR of bit c_i, conditional on the information available at C_j incoming from all its neighboring variable nodes, except the information incoming from V_i. A pictorial representation of this process is provided in Fig. 11.13. Note that two different variable nodes connected to C_j will receive, in general, different messages.

The message m_j^i from C_j to V_i can be calculated exploiting one of the results introduced in Sect. 11.5.2. In fact, each of the h_j incoming messages is the LR of a specific bit on which the check node imposes a parity constraint. Hence, under independence hypothesis, denoting by $N(j)\backslash\{i\}$ the set of indexes of variable nodes connected to C_j except V_i, from Eq. (11.17) we immediately obtain

$$m_j^i = \frac{\prod_{k \in N(j)\backslash\{i\}} \frac{r_k^j+1}{r_k^j-1} + 1}{\prod_{k \in N(j)\backslash\{i\}} \frac{r_k^j+1}{r_k^j-1} - 1}. \tag{11.20}$$

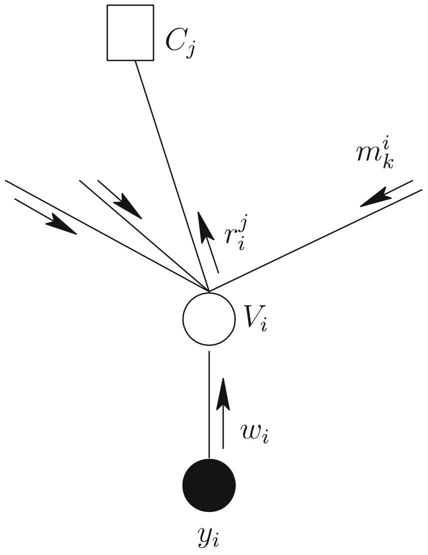


Fig. 11.14 Variable node processing of incoming messages during the vertical step

Note that the independence hypothesis is fulfilled only during the first $g/2$ decoding iterations, where g is the girth of the Tanner graph. On the other hand, it represents an approximation during all subsequent iterations.

11.5.3.4 Vertical Step

For $i = 0, \ldots, n-1$, variable node V_i, of degree d_i, sends to each of its d_i neighboring check nodes its current estimate of the associated bit. With reference to Fig. 11.14, the message r_i^j sent to check node C_j is the LR about bit c_i, conditional on the a priori information available from the communication channel and on the information incoming from all check nodes connected to it, except C_j. Again, two different check nodes connected to V_i will receive, in general, different messages.

The message r_i^j that variable node V_i sends to check node C_j connected to it can be easily computed based on the result in Sect. 11.5.2. In fact, each of the d_i messages incoming towards the variable node (including the message w_i incoming from the channel), represents the LR of c_i conditioned to some observation. Under the hypothesis of independence for the d_i observations, denoting by $N(i)\backslash\{j\}$ the set of indexes check nodes connected to V_i except check node of index j, we have

$$r_i^j = w_i \prod_{k \in N(i)\backslash\{j\}} m_k^i. \tag{11.21}$$

(Again, the independence hypothesis is valid rigorously only during the first $g/2$ decoding iterations.)

11.5.3.5 Hard Decision and Stopping Criterion

At the last step of each iteration, every variable node takes a decision about its associated encoded bit. This decision is based on all currently available information about the bit, i.e., on the a priori information from the communication channel and on *all* messages incoming from the check nodes. Let $\boldsymbol{m}^i$ denote the list of all messages incoming towards the variable node V_i. Applying again the result developed in Sect. 11.5.2 under the hypothesis of independence of the incoming messages, we may write

$$L\left(c_i|w_i,\boldsymbol{m}^i\right) = w_i \prod_{k\in N(i)} m_k^i. \tag{11.22}$$

(Again, the independence hypothesis is fulfilled rigorously only during the first $g/2$ decoding iterations.) The decision about encoded bit c_i at the end of the generic iteration is then

$$L\left(c_i|w_i,\boldsymbol{m}^i\right) \underset{\hat{c}_i=1}{\overset{\hat{c}_i=0}{\gtrless}} 1.$$

If the current hard-decision sequence $\hat{\boldsymbol{c}}$ is a codeword ($\hat{\boldsymbol{c}}\boldsymbol{H}^T = 0$, where $\boldsymbol{H}$ is any parity-check matrix of the code) then the algorithm is terminated and $\hat{\boldsymbol{c}}$ is returned as the decoded codeword. Else, if $\hat{\boldsymbol{c}}$ is not a codeword and the maximum number of iterations I_{max} has been reached, the algorithm is terminated and a failure is reported. Else, a new iteration is started jumping to the horizontal step. Belief propagation decoding of LDPC codes may be summarized as follows.

Belief-Propagation Decoding of LDPC Codes

```
1: set $I = 1$. For $i = 0,\dots,n-1$, for $j \in N(i)$, set $r_i^j = w_i$;
2: for $j = 0,\dots,m-1$
        for $i \in N(j)$ calculate $m_j^i$ according to Eq. (11.20);
3: for $i = 0,\dots,n-1$
        for $j \in N(i)$ calculate $r_i^j$ according to Eq. (11.21);
4: for $i = 0,\dots,n-1$ {
        calculate $L\left(c_i|w_i,\boldsymbol{m}^i\right)$ according to Eq. (11.22);
        if $L\left(c_i|w_i,\boldsymbol{m}^i\right) \geq 1$ then set $\hat{c}_i = 0$;
        else set $\hat{c}_i = 1$;
   }
   if $\hat{\boldsymbol{c}}\boldsymbol{H}^T = 0$ then return $\hat{\boldsymbol{c}}$;
   else {
        if $I = I_{max}$ exit;
        else {
             $I = I + 1$;
             goto 2;
        }
   }
```

11.5.4 Log-Domain BP Decoder

The main issue when implementing BP decoding described in Sect. 11.5.3 is represented by the need to handle and combine, through multiplications and divisions, likelihood ratios whose values may differ by several orders of magnitude. For this reason, a log-domain implementation is usually preferred from an implementation viewpoint. In the log-domain version of BP decoding, log-likelihood ratios (LLRs) of the encoded bits are exchanged between variable and check nodes. Next, we discuss how the above-described BP decoding shall be modified in the log-domain. All logarithms are assumed to be natural logarithms. Moreover, $\operatorname{sgn}(x)$ will denote the sign function, i.e., $\operatorname{sgn}(x) = +1$ if $x \geq 0$ and $\operatorname{sgn}(x) = -1$ otherwise.

The initialization step remains the same, the only difference being that the first message each variable node sends to all its neighboring check nodes is the a priori LLR of the corresponding encoded bit. Neglecting again the iteration index and denoting by R_i^j the message sent from variable node $i \in \{0, \dots, n-1\}$ to check node $j \in N(i)$, we have

$$R_i^j = W_i,$$

where $W_i = \log w_i$. For instance, assuming antipodal mapping $x_i = 1 - 2c_i$, over a BSC with error probability p we have

$$W_i = \begin{cases} \log \frac{1-p}{p} & \text{if} \quad y_i = +1 \\ \log \frac{p}{1-p} & \text{if} \quad y_i = -1 \end{cases} \tag{11.23}$$

while, over a Bi-AWGN channel,

$$W_i = \frac{2}{\sigma^2} y_i. \tag{11.24}$$

The development of check node message processing (horizontal step) in the log domain is more involved. Denoting $R_i^j = \log r_i^j$ and $M_j^i = \log m_j^i$, from Eq. (11.20) we may write

$$\begin{aligned} M_j^i &= \log \frac{\prod_{k \in N(j) \setminus \{i\}} \frac{e^{R_k^j}+1}{e^{R_k^j}-1} + 1}{\prod_{k \in N(j) \setminus \{i\}} \frac{e^{R_k^j}+1}{e^{R_k^j}-1} - 1} \\ &= \log \frac{\prod_{k \in N(j) \setminus \{i\}} \operatorname{sgn}\left(R_k^j\right) \cdot \prod_{k \in N(j) \setminus \{i\}} \frac{e^{\left|R_k^j\right|}+1}{e^{\left|R_k^j\right|}-1} + 1}{\prod_{k \in N(j) \setminus \{i\}} \operatorname{sgn}\left(R_k^j\right) \cdot \prod_{k \in N(j) \setminus \{i\}} \frac{e^{\left|R_k^j\right|}+1}{e^{\left|R_k^j\right|}-1} - 1}, \end{aligned}$$

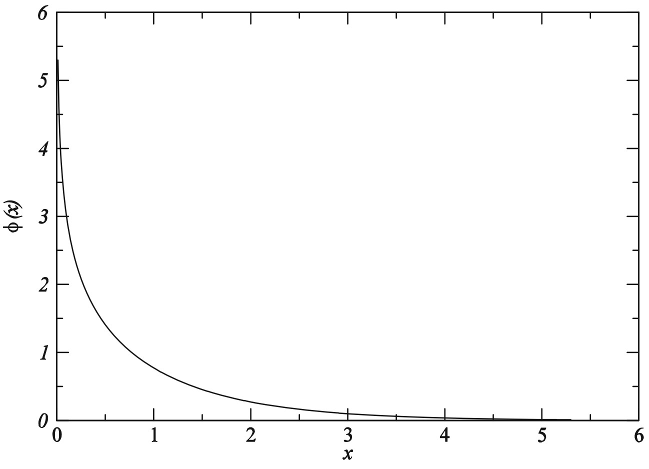


Fig. 11.15 Plot of function $\varphi(x) = -\log(\tanh(x/2))$

where we have exploited the fact that any odd function fulfills $f(x) = \text{sgn}(x) f(|x|)$ and the fact that $f(x) = (e^x + 1)/(e^x - 1)$ is odd. The obtained expression of M_j^i can be further developed through the identity $\log((x+1)/(x-1)) = \text{sgn}(x) \cdot \log((|x|+1)/(|x|-1))$ and through the fact that $e^{|R|} \geq 1$. This yields

$$
\begin{aligned}
M_j^i &= \prod_{k \in N(j)\setminus\{i\}} \text{sgn}\left(R_k^j\right) \cdot \log \frac{\prod_{k \in N(j)\setminus\{i\}} \frac{e^{\left|R_k^j\right|}+1}{e^{\left|R_k^j\right|}-1} + 1}{\prod_{k \in N(j)\setminus\{i\}} \frac{e^{\left|R_k^j\right|}+1}{e^{\left|R_k^j\right|}-1} - 1} \\
&= \prod_{k \in N(j)\setminus\{i\}} \text{sgn}\left(R_k^j\right) \cdot \log \frac{e^{\sum_{k \in N(j)\setminus\{i\}} \log \frac{e^{\left|R_k^j\right|}+1}{e^{\left|R_k^j\right|}-1}} + 1}{e^{\sum_{k \in N(j)\setminus\{i\}} \log \frac{e^{\left|R_k^j\right|}+1}{e^{\left|R_k^j\right|}-1}} - 1} \\
&= \prod_{k \in N(j)\setminus\{i\}} \text{sgn}\left(R_k^j\right) \cdot \varphi\left(\sum_{k \in N(j)\setminus\{i\}} \varphi\left(\left|R_k^j\right|\right)\right)
\end{aligned} \tag{11.25}
$$

where, for $x > 0$, we have introduced the nonlinear function

$$\varphi(x) = \log \frac{e^x + 1}{e^x - 1} = -\log(\tanh(x/2)).$$

A plot of this function is depicted in Fig. 11.15. Note that the function coincides with its inverse, i.e., $\varphi(\varphi(x)) = x$.

The transposition of the variable node processing (vertical step) to the logarithmic domain is much simpler. In fact, from Eq. (11.21) we immediately obtain

$$R_i^j = W_i + \sum_{k \in N(i) \setminus \{j\}} M_k^i. \tag{11.26}$$

Analogously, Eq. (11.22) shall be updated as

$$\log L\left(c_i | W_i, \boldsymbol{M}^i\right) = W_i + \sum_{k \in N(i)} M_k^i. \tag{11.27}$$

The algorithm may be then summarized as follows.

Log-Domain Belief-Propagation Decoding of LDPC Codes

```
1: set $I = 1$. For $i = 0, \ldots, n-1$, for $j \in N(i)$, set $R_i^j = W_i$;
2: for $j = 0, \ldots, m-1$
       for $i \in N(j)$ calculate $M_j^i$ according to Eq. (11.25);
3: for $i = 0, \ldots, n-1$
       for $j \in N(i)$ calculate $R_i^j$ according to Eq. (11.26);
4: for $i = 0, \ldots, n-1$ {
       calculate $\log L\left(c_i | W_i, \boldsymbol{M}^i\right)$ according to Eq. (11.27);
       if $\log L\left(c_i | W_i, \boldsymbol{M}^i\right) \geq 0$ then set $\hat{c}_i = 0$;
       else set $\hat{c}_i = 1$;
   }
   if $\hat{\boldsymbol{c}} \boldsymbol{H}^T = 0$ then return $\hat{\boldsymbol{c}}$;
   else {
       if $I = I_{max}$ exit;
       else {
           $I = I + 1$;
           goto 2;
       }
   }
```

Although an enhanced numerical stability is achieved operating on log-likelihood ratios, as well as a lower complexity (as, for instance, products in Eqs. (11.21) and (11.22) are transformed in sums in Eqs. (11.25) and (11.26), respectively), check node processing in the log-domain imposes the evaluation of the nonlinear function φ. For a single check node C_j of degree h_j, this function should in principle be evaluated $(h_j)^2$ times per iteration (even if techniques to limit the number of φ evaluations exist). The calculation of function φ is typically performed by means of lookup-tables. Note that, however, for small x the graph of $\varphi(x)$ is very steep, thus requiring a very fine (in general, nonuniform) discretization of the corresponding region of the function domain, and that the implementation of $\varphi(x)$ through a lookup table may be quite inconvenient in hardware implementation. For these reasons, extensive work has been carried out to develop either approximations of the log-domain BP decoder or other reduced-complexity decoding schemes. All of these decoders offer a reduced error correction capability than actual BP. However, they also exhibit a lower decoding complexity and, hence, a higher decoding speed.

11.6 Reduced-Complexity Decoders

So far we have focused on the BP decoder (both in probability domain and log-domain) originally developed by Gallager. Next, we present a few reduced-complexity, implementation-friendly decoders for LDPC codes. It must be pointed out that a large amount of reduced-complexity decoding schemes for LDPC codes have been developed in the last decade [7]. Most of these decoding schemes may be seen as approximations of the BP decoder, in the sense that they are characterized by approximations of the most complex step of BP decoding, namely, the horizontal step (consisting of the calculation of extrinsic messages from the check nodes to the variable nodes). As such, these approximate BP decoding algorithms can be formalized via the same pseudo-code we have adopted for the log-domain BP decoder, with a difference in step 2.

We only present the most famous approximation of the BP decoder, called the Min-Sum (MS) decoder. We then move to describe decoders exhibiting an even lower complexities. More specifically, we present a binary message-passing algorithm known as "Gallager B" (and originally proposed in [15]) and a class of non-message-passing decoders named "flipping algorithms" (the idea of bit flipping appears again in [15]). These very low complexity decoding algorithms (along with some of their modifications, not addressed in this chapter) are of interest in NAND Flash memories at the beginning of the memory life, when the raw bit error probability is extremely low.

11.6.1 Min-Sum Decoder

The MS decoder can be directly developed from the log-domain BP decoder as follows. From Fig. 11.15 observe that the graph of function $\varphi(x)$ is very steep for small values of x. Then, when x assumes small values, a small perturbation in terms of x determines a large deviation in terms of $\varphi(x)$. For this reason, if at least one of the magnitudes $|R_k^j|$ in the summation appearing in (11.25) is sufficiently small, the corresponding value of $\varphi(|R_k^j|)$ dominates the other summands. Hence, we can write

$$\begin{aligned} M_j^i &= \prod_{k \in N(j)\backslash\{i\}} \operatorname{sgn}\left(R_k^j\right) \cdot \varphi\left(\sum_{k \in N(j)\backslash\{i\}} \varphi\left(\left|R_k^j\right|\right)\right) \\ &\approx \prod_{k \in N(j)\backslash\{i\}} \operatorname{sgn}\left(R_k^j\right) \cdot \varphi\left(\max_{k \in N(j)\backslash\{i\}} \max_{k \in N(j)\backslash\{i\}} \varphi\left(\left|R_k^j\right|\right)\right) \\ &= \prod_{k \in N(j)\backslash\{i\}} \operatorname{sgn}\left(R_k^j\right) \cdot \min_{k \in N(j)\backslash\{i\}} \left|R_k^j\right| \end{aligned} \tag{11.28}$$

where the last equality follows from $\varphi(x)$being self-invertible (i.e., $\varphi\left(\varphi(x)\right) = x$) and monotonically decreasing. The MS decoding algorithm is summarized next.

Min-Sum Decoding of LDPC Codes

```
1: set $I = 1$. For $i = 0, \ldots, n-1$, for $j \in N(i)$, set $R_i^j = W_i$;
2: for $j = 0, \ldots, m-1$
        for $i \in N(j)$ calculate $M_j^i$ according to Eq. (11.28);
3: for $i = 0, \ldots, n-1$
        for $j \in N(i)$ calculate $R_i^j$ according to Eq. (11.26);
4: for $i = 0, \ldots, n-1$ {
        calculate $\log L\left(c_i | W_i, \boldsymbol{M}^i\right)$ according to Eq. (11.27);
        if $\log L\left(c_i | W_i, \boldsymbol{M}^i\right) \geq 0$ then set $\hat{c}_i = 0$;
        else set $\hat{c}_i = 1$;
   }
   if $\hat{\boldsymbol{c}}\boldsymbol{H}^T = 0$ then return $\hat{\boldsymbol{c}}$;
   else {
        if $I = I_{max}$ exit;
        else {
                $I = I + 1$;
            goto 2;
            }
   }
```

Several improvements to the MS decoder have been proposed in the literature, to reduce the gap between its performance and that of BP decoding, at the expense of a small increase in terms of computational cost. These refinements are out of the scope of this book. Interested readers may refer, for example, to [6, 33].

11.6.2 Gallager B Decoder

The BP and MS decoders are characterized by real-valued (properly quantized, in hardware implementation) messages exchanged between the variable nodes and the check nodes. Moreover, as previously emphasized, both algorithms remain unchanged over a wide range of communication channels. In contrast, Gallager B decoder, first proposed in [15], is a message-passing decoding algorithm for LDPC codes characterized by binary-valued messages and is specifically tailored for the BSC (i.e., no soft information is available at the decoder input). Although its performance is poor compared with that of BP and MS algorithms over the BSC, it has been proved that it represents the optimum LDPC decoder over the BSC when the extrinsic messages are constrained to be binary.

The algorithm works as follows. Assuming transmission over a BSC with error probability p and input and output alphabets $\mathcal{X} = \mathcal{Y} = \{0, 1\}$, for $i = 0, \ldots, n-1$ variable node V_i is fed with the corresponding binary symbol $y_i \in \{0, 1\}$ received from the channel. (In contrast, to perform BP decoding over the BSC variable node i is initialized according to Eq. (11.18) or to its logarithmic version Eq. (11.23).) The symbol y_i is broadcasted by variable node V_i to each of its neighboring check nodes. The algorithm is then structured in a similar way as BP or MS, where the horizontal, vertical, and stopping criterion steps are specified as follows.

During the horizontal step, for $j = 0, \ldots, m-1$ the message propagating from check node C_j to variable node V_i, $i \in N(j)$, is simply the modulo-2 summation of all binary messages incoming from variable nodes connected to C_j but the message incoming from V_i. Hence, we can write

$$m_j^i = \sum_{k \in N(j) \setminus \{i\}} r_k^j \tag{11.29}$$

where the summation is modulo-2. (Note that $r_i^j = y_i$ for all $i = 0, \ldots, n-1$ at the first iteration.) During the vertical step, for $i = 0, \ldots, n-1$ the message from variable node V_i to check node C_j, $j \in N(i)$, is equal to the modulo-2 complement of y_i if the number of incoming extrinsic messages different from y_i is above some threshold, and is equal to y_i otherwise. Letting

$$X_j^i = \left|\left\{m_k^i \neq y_i s.t. k \in N(i) \setminus \{j\}\right\}\right|$$

and $T^{(i)}$ be the number of such extrinsic messages and the threshold at the current iteration, respectively, and letting $C(y_i)$ be the modulo-2 complement of y_i, we have

$$r_i^j = \begin{cases} C(y_i) & \text{if } X_j^i \geq T^{(i)} \\ y_i & \text{otherwise.} \end{cases} \tag{11.30}$$

At the end of each decoding iteration, for each variable node V_i the decision about the current value of the local bit $\hat{c}_i$ is taken according to a majority policy. More specifically, if the variable node degree d_i is even, then $\hat{c}_i$ is set equal to the value assumed by the majority of the incoming messages m^i_j and of y_i. On the other hand, if the variable node degree is odd, then $\hat{c}_i$ is set equal to the value assumed by the majority of the incoming messages m^i_j (y_i is not considered).

Gallager B Decoding of LDPC Codes

```
1: set $I = 1$. For $i = 0, \ldots, n-1$, for $j \in N(i)$, set $r_i^j = y_i$;
2: for $j = 0, \ldots, m-1$
       for $i \in N(j)$ calculate $m^i_j$ according to Eq. (11.29);
3: for $i = 0, \ldots, n-1$
       for $j \in N(i)$ calculate $r_i^j$ according to Eq. (11.30);
4: for $i = 0, \ldots, n-1$ {
      if $d_i \mod 2 = 0$ then set $\hat{c}_i$ to the value assumed by the majority of the incoming messages
        $m^i_j$ and of $y_i$;
      else set $\hat{c}_i$ to the value assumed by the majority of the incoming messages $m^i_j$;
   }
   if $\hat{\boldsymbol{c}}\boldsymbol{H}^T = 0$ then return $\hat{\boldsymbol{c}}$;
   else {
       if $I = I_{max}$ exit;
       else {
                  $I = I + 1$;
           goto 2;
           }
     }
```

Appropriate values for the threshold $T^{(i)}$ range between $(d_i - 1)/2$ and d_i, as the number of incoming extrinsic messages enforcing an outgoing message different from $\tilde{y}_i$ must be sufficiently high. Note that in principle, for irregular codes the value of the threshold may be different for two different variable nodes, even during the same iteration. Also note that, for the same variable node, the value of the threshold may not remain constant with the iteration index, as it may be adjusted dynamically. In [15] it was shown that for a regular (d, h) LDPC code, the optimum value of the threshold (the same for all variable nodes at the same iteration) is the smallest integer T for which the inequality

$$\frac{1-p}{p} \leq \left(\frac{1 + (1-2\varepsilon)^{h-1}}{1 - (1-2\varepsilon)^{h-1}} \right)^{2T-d+1} \tag{11.31}$$

is fulfilled, where p is the BSC error probability and ε is the extrinsic error probability. This latter parameter represents the average probability that an edge in the Tanner graph carries an error message from the variable node set to the check node set at the considered iteration, and varies over iterations. In the asymptotic setting where the Tanner graph is assumed to be cycle-free, the update equation for ε for regular LDPC codes is [15]

$$\begin{aligned}\varepsilon_{\ell+1} = p - p \sum_{z=T_\ell}^{d-1} \binom{d-1}{z} \left[\frac{1+(1-2\varepsilon_\ell)^{h-1}}{2}\right]^z \left[\frac{1-(1-2\varepsilon_\ell)^{h-1}}{2}\right]^{d-1-z} \\ +(1-p)\sum_{z=T_\ell}^{d-1} \binom{d-1}{z} \left[\frac{1-(1-2\varepsilon_\ell)}{2}\right]^z \left[\frac{1+(1-2\varepsilon_\ell)^{h-1}}{2}\right]\end{aligned} \tag{11.32}$$

where $\ell \geq 0$ is the iteration index and where $\varepsilon_0 = p$.

Example 5 Equation (11.32) represents density evolution recursion for Gallager B decoding of regular unstructured (d, h) LDPC code ensembles. The asymptotic decoding threshold p^* for this ensemble under Gallager B decoding is then the sup of the set of all $p > 0$ such that $\lim_{\ell\to\infty} \varepsilon_\ell = 0$. For given d and h, whether or not some p is above or below threshold can be easily checked by running the recursion (with starting point $\varepsilon_0 = p$), adapting the value of T_ℓ at each iteration according to Eq. (11.31) for the current value of ε_ℓ. For example, for $d = 4$ and $h = 40$ (which corresponds to a rate $R = 9/10$ ensemble) we obtain a threshold $p^* = 0.0041$. Through Eq. (11.8) and $E_s = RE_b$, this corresponds to a threshold $(E_b/N_o)^* = 5.892$ dB, about 1.5 dB away from the Shannon limit relevant to the one-bit quantized Bi-AWGN channel.

11.6.3 Flipping Algorithms

Flipping algorithms are a class of low-complexity, iterative decoding algorithms for LDPC codes over the BSC different from message-passing ones. The decoding strategy consists of flipping, at the end of each decoding iteration, the current value of a subset of variable nodes for which a certain flipping condition is fulfilled. If the obtained binary sequence is a codeword, decoding is stopped and the codeword is returned. Otherwise, a new iteration is started. The process continues until a codeword is found or a maximum number of iterations is reached. Different flipping algorithms are characterized by different criteria to identify the variable nodes to be flipped.

A popular flipping algorithm, hereafter referred to simply as bit-flipping (BF) algorithm, consists of flipping at each iteration those variable nodes for which the number u of unsatisfied check nodes is maximum. A BSC with input and output alphabets $\mathcal{X} = \mathcal{Y} = \{0, 1\}$ is assumed.

Bit-Flipping Decoding of LDPC Codes

```
1: set $I = 1$. For $i = \{0, \ldots, n-1\}$, set $\hat{c}_i = y_i$;
2: if $\hat{\boldsymbol{c}}\boldsymbol{H}^T = 0$ then return $\hat{\boldsymbol{c}}$;
3: for $i = 0, \ldots, n-1$
       calculate $u_i$;
4: calculate $u_{max}$;
5: for each $i$ such that $u_i = u_{max}$ set $\hat{c}_i = (\hat{c}_i + 1)\, mod 2$;
5: if $\hat{\boldsymbol{c}}\boldsymbol{H}^T = 0$ then return $\hat{\boldsymbol{c}}$;
   else {
       if $I = I_{max}$ exit;
       else {
           $I = I + 1$;
           goto 2;
       }
   }
```

11.7 Numerical Example

In this section, we present some numerical results aimed at comparing the performance of LDPC and BCH codes, with the purpose to highlight the potential of LDPC codes in Flash memories applications. We assume an SLC memory as the reference channel model. We compare the performance of a regular QC-LDPC code, under several decoding algorithms offering different tradeoffs between performance and complexity, with the performance of a narrowsense binary BCH code with similar parameters, decoded via bounded distance decoding.

The LDPC code is characterized by a length $n_{\text{LDPC}} = 8200$ and a dimension $k_{\text{LDPC}} = 7379$ bits, and therefore by a code rate R very close to 9/10. Its minimum distance, estimated with the impulse method proposed in [16], is equal to $d_{\text{LDPC}} = 114$. All variable nodes of the LDPC code have degree 4, and all of its check nodes have degree 40. Its $820 \times 8{,}200$ parity-check matrix is in block circulant form, where the generic block is a 205×205 circulant permutation matrix, and has been constructed according to a block circulant version of the progressive edge-growth (PEG) algorithm. The performance of this code has been evaluated via Monte Carlo software simulation, under BP, MS, and BF decoding algorithms. The performance curves under both BP and MS decoding have been obtained under two different settings, namely, soft-decision and hard-decision decoding. These two settings correspond to assuming the Bi-AWGN channel with unquantized output (Example 2) and with one-bit quantized output (Example 3), respectively, as the channel model. The first setting is equivalent to assuming an SLC memory with an infinite number of reads per bit, while the second one to assuming an SLC memory with one read per bit. The variable nodes are initialized according to Eq. (11.19) in the unquantized case and according to Eq. (11.18) in the quantized one. In the quantized case, the raw bit error rate of the channel can be obtained from E_b/N_0 according to Eq. (11.8), where $E_s/N_0 = RE_b/N_0$. For instance,

$E_b/N_0 = 5\text{dB}$ corresponds to a raw bit error rate $p = 8.5\,10^{-3}$. The Shannon limit for the unquantized case and for the one-bit quantized case are also evaluated, for benchmarking purposes.

The competitor BCH code has nominal parameters $n_{\text{BCH}} = 8191$, $k_{\text{BCH}} = 7372$, $t = 63$ (error correction capability), and minimum distance $d_{\text{BCH}} = 127$. Its code rate is approximately equal to 9/10, similar to the code rate of the QC-LDPC code. The codeword error rate (CER) and the bit error rate (BER) of the BCH code under hard decision bounded distance decoding have been evaluated analytically according to the relationships

$$P_e = \sum_{r=t+1}^{n_{\text{BCH}}} \binom{n_{\text{BCH}}}{r} p^r (1-p) n_{\text{BCH}}{}^{-r} \tag{11.33}$$

And

$$P_b \approx \frac{d_{\text{BCH}}}{k} \cdot P_s \tag{11.34}$$

respectively.

With reference to Fig. 11.16, we see that over the hard-decision channel (SLC with one read) the BCH code exhibits nearly the same performance as the QC-LDPC code decoded via BP and that its performance is even slightly better at low error rates. This is not surprising, as BCH codes are well known to offer very good performances over hard-decision channels, especially at high code rates. As opposed to BCH codes, however, LDPC codes can handle in a very natural way soft information incoming from the communication channel, which allows to attain substantial performance improvements over the error correction capabilities achievable with hard-decision decoding. In our example, when the LDPC decoder is fed with unquantized soft information, its coding gain with respect to that achieved under hard-decision decoding is improved by about 1.6 dB under both BP and MS decoding algorithms at CER $= 10^{-4}$. Moreover, again at CER $= 10^{-4}$, the LDPC code under unquantized BP decoding performs only 0.8 dB away from the corresponding Shannon limit, in terms of BER.

For the same decoding algorithm (BP or MS), the performance curves of the LDPC code labeled as "soft" and "hard" represent the two extreme cases in which unconstrained soft information is available at the decoder, and no soft information is available. In general, when a finite number of cell reads is performed with different read voltage values, the corresponding performance curve will lie between the two extreme curves: The larger the number of cell reads, the closer the performance curve to the "soft" one. Therefore, LDPC codes can largely outperform BCH codes in Flash memory applications, provided a sufficient amount of soft information is available at the decoder. It is also pointed out that the design of appropriate QC irregular LDPC codes can favor an even larger coding gain with respect to BCH codes.

We also highlight how very simple decoding algorithms of LDPC codes such as BF (or Gallager B) decoding, can be of interest at the beginning of the memory life,

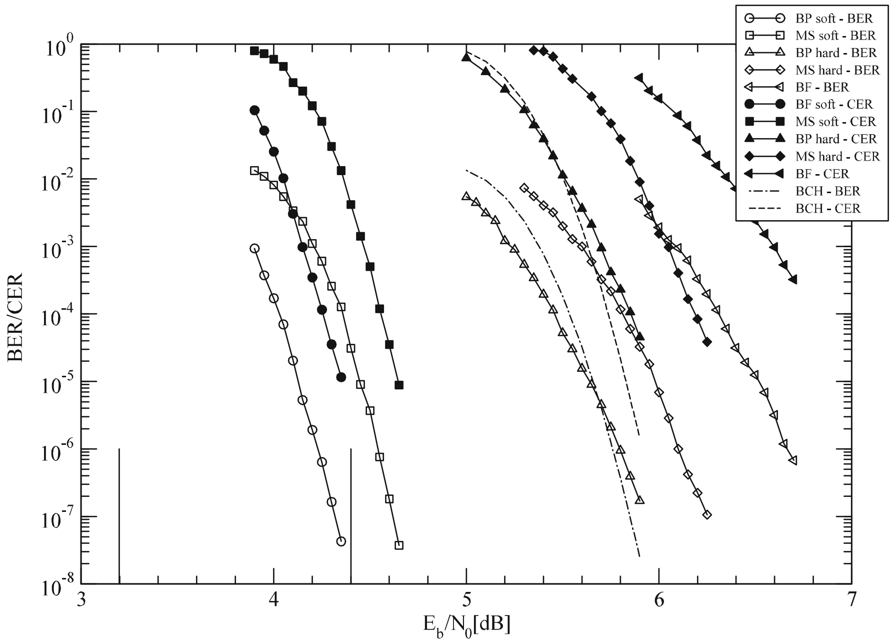


Fig. 11.16 Bit and codeword error rates for an (8191,7372) QC-LDPC code (under different decoding algorithms) and an (8191,7372), t = 63 narrowsense binary BCH code under bounded distance decoding, over an SLC Flash memory channel. Curves corresponding to *filled* and *empty symbols* illustrate the codeword error rates and the bit error rates of the LDPC code, respectively. The *dashed* and *dot-dashed lines* illustrate the codeword error rate and the bit error rate of the BCH code, respectively. The two *straight solid lines* are the Shannon limits for rate R = 9/10 under soft-decision and hard-decision decoding, respectively

i.e., when the raw bit error rate is very small. For example, as from Fig. 11.16, BF decoding could become of interest for values of E_b/N_0 larger of 70 dB, corresponding to a raw bit error rate smaller than $1.3{\cdot}10^{-3}$.

Acknowledgments The author wishes to thank Prof. M. Chiani (University of Bologna) and Dr. G. Liva (DLR) for their useful feedback. He also wishes to thank Ing. R. Micheloni and A. Marelli (IDT Italy) for their careful proof check of this chapter.

References

1. S. Abu-Surra, D. Divsalar, W.E. Ryan, Enumerators for protograph-based ensembles of LDPC and generalized LDPC codes. IEEE Trans. Inf. Theory **57**(February), 858–886 (2011)
2. C. Berrou, A. Glavieux, P. Thitimajshima, Near shannon limit error-correcting coding and decoding: turbo-codes, in *Proceedings of the 2003 International Conference on Communication*, vol. 2, May 1993, Geneva, Switzerland, pp. 1064–1070

3. N. Bonello, S. Chen, L. Hanzo, Low-density parity-check codes and their rateless relatives. IEEE Commun. Surveys & Tutorials **13**(February), 3–26 (2011)
4. D. Cavus, C. Haymes, Low BER performance estimation of LDPC codes via application of importance sampling to trapping sets. IEEE Trans. Commun. **57**(July), 1886–1888 (2009)
5. L. Chen, J. Xu, I. Djurdjevic, S. Lin, Near Shannon limit quasi cyclic low-density parity-check codes. IEEE Trans. Commun. **52**(July), 1038–1042 (2004)
6. J. Chen, M. Tanner, C. Jones, Y. Li, Improved min-sum decoding algorithms for irregular LDPC codes, in *Proceedings of the 2005 IEEE International Symposium on Information Theory*, Sept 2005, Adelaide, Australia, pp. 449–453
7. J. Chen, A. Dholakia, E. Eleftheriou, M. Fossorier, X.-Y. Hu, Reduced-complexity decoding of LDPC codes. IEEE Trans. Commun. **53**(August), 1288–1299 (2005)
8. M. Chiani, A. Ventura, Design and performance evaluation of some high-rate irregular low-density parity-check codes, in *Proceedings of the 2001 Global Telecommunication Conference*, San Antonio, TX, Nov 2001, pp. 990–994
9. S.-Y. Chung, G.D. Forney Jr., T. Richardson, R. Urbanke, On the design of low-density parity-check codes within 0.0045 dB of the Shannon limit. IEEE Commun. Lett. **5**(February), 58–60 (2001)
10. T.M. Cover, J.A. Thomas, *Elements of Information Theory* (Wiley, New York, 1991)
11. L. Dolecek et al., Predicting error floors of structured LDPC codes: Deterministic bounds and estimates. IEEE J. Sel. Areas Commun. **27**(August), 908–917 (2009)
12. M. Flanagan, E. Paolini, M. Chiani, M. Fossorier, On the growth rate of the weight distribution of irregular doubly-generalized LDPC codes. IEEE Trans. Inf. Theory **57**(June), 3721–3737 (2011)
13. M. Fossorier, Quasi-cyclic low-density parity-check codes from circulant permutation matrices. IEEE Trans. Inf. Theory **50**(August), 1788–1793 (2004)
14. R.D. Fowler, L. Nordheim, Electron emission in intense electric fields. Proc. Royal Soc. London **119**(May), 173–181 (1928)
15. R.G. Gallager, *Low-Density Parity-Check Codes* (MIT Press, Cambridge, 1963)
16. X.-Y. Hu, M. Fossorier, E. Eleftheriou, On the computation of the minimum distance of low-density parity-check codes, in *Proceedings of the 2004 International Conference on Communication*, June 2004, Paris, France, pp. 767–771
17. S. Li, T. Zhang, Improving multi-level NAND flash memory storage reliability using concatenated BCH-TCM coding. IEEE Trans. VLSI **18**(October), 1412–1420 (2010)
18. Z. Li, L. Chen, L. Zeng, S. Lin, W. Fong, Efficient encoding of low-density parity-check codes. IEEE Trans. Commun. **54**(January), 71–81 (2006)
19. G. Liva, M. Chiani, Protograph LDPC codes design based on EXIT analysis, in *Proceedings of the 2007 IEEE Global Telecommunications Conference*, Washington, DC, Nov 2007, pp. 3250–3254
20. M. Mansour, High-performance decoders for regular and irregular repeat-accumulate codes, in *Proceedings of the IEEE 2004 IEEE Global Telecommunications Conference*, Nov/Dec 2004, Dallas, TX, USA, pp. 2583–2588
21. R. Micheloni, L. Crippa, A. Marelli (eds.), *Inside NAND Flash Memories* (Springer, Berlin, 2010)
22. N. Mielke et al., Bit error rate in NAND Flash memories, in *Proceedings of the 2008 IEEE International Symposium on Reliability Physics*, Phoenix, AZ, Apr/May 2008, pp. 9–19
23. T. Richardson, Error floors of LDPC codes, in *Proceedings of the 41st Annual Allerton Conference on Communication, Control and Computing*, (Monticello, IL, USA, 2003)
24. T. Richardson, M. Shokrollahi, R. Urbanke, Design of capacity-approaching irregular low-density parity-check codes. IEEE Trans. Inf. Theory **47**(February), 619–637 (2001)
25. T. Richardson, R. Urbanke, The capacity of low-density parity-check codes under message-passing decoding. IEEE Trans. Inf. Theory **47**(February), 599–618 (2001)
26. T. Richardson, R. Urbanke, The renaissance of Gallager's low-density parity-check codes. IEEE Commun. Mag. **41**(August), 126–131 (2003)

27. H. Tang, J. Xu, Y. Kou, S. Lin, K. Abdel-Ghaffar, On algebraic construction of Gallager and circulant low density parity-check codes. IEEE Trans. Inf. Theory **50**(June), 1269–1279 (2004)
28. M. Tanner, A recursive approach to low complexity codes. IEEE Trans. Inf. Theory **27**(September), 533–547 (1981)
29. S. ten Brink, Convergence behavior of iteratively decoded parallel concatenated codes. IEEE Trans. Commun. **49**(October), 1727–1737 (2001)
30. J. Thorpe, Low-density parity-check (LDPC) codes constructed from protographs. JPL INP, Tech. Rep. August 42–154 (2003)
31. J. Xu, L. Chen, L. Zeng, L. Lan, S. Lin, Construction of low-density parity-check codes by superposition. IEEE Trans. Commun. **53**(February), 243–251 (2005)
32. J. Wang, T. Courtade, H. Shankar, R. Wesel, Soft information for LDPC decoding in flash: Mutual-information optimized quantization, in *Proceedings of the 2011 IEEE Global Telecommunication Conference*, Houston, TX, Dec 2011
33. J. Zhao, F. Zarkeshvari, A. Banihashemi, On implementation of min-sum algorithm and its modifications for decoding low-density parity-check (LDPC) codes. IEEE Trans. Commun. **53**(April), 549–554 (2005)

Chapter 12
Protecting SSD Data Against Attacks

A. Marelli and R. Micheloni

Abstract When a drive is broken and we have to throw it away, we want to be sure that no hackers can recover the data stored in that disk, especially in the enterprise environment where sensitive date are stored on the drive, such as financial transactions or military applications.

As the SSD market is growing, the security issue must be carefully considered. Some methods used with HDDs, such as degaussian, are not applicable to SSDs, due to the different storage technique. Recent studies indicate that encryption is the necessary step to protect data stored in SSD against hackers attacks.

This chapter describes the SSD security approach in comparison to HDD, then it walks the reader through the encryption world: how a cryptosystem is built, how a cryptosystem is broken, different encryption applications, and then the AES cryptosystem as it is the most used in SSDs; finally, it addresses the security applications in SSDs.

12.1 Challenges of SSD Security vs. HDD

Hard Disk Drives as well as Solid State Disks contain a number of sensible data that must be kept secret. When a disk is thrown away or stolen, it is very important that nobody can access these data.

The purpose of HDD is to store data and protect them from corruption or accidental erase. In this latter case, procedures like folder or un-erase are used. In addition, data erasure is unlikely to occur because it takes a lot of time, hence reducing performances. The drawback is that user data are vulnerable to recovery

A. Marelli (✉) • R. Micheloni
Integrated Device Technology, Enterprise Computing Division, Agrate Brianza (MB), Italy
e-mail: alessiamarelli@gmail.com; rino.micheloni@ieee.org

by unauthorized person. Increased storage of sensitive data, combined with rapid technological change and the shorter lifespan of IT assets, has driven the need for permanent data erasure of electronic devices as they are retired.

If data erasure does not occur when a disk is retired or lost, an organization or a user faces the possibility that data will be stolen and compromised, leading to identity theft, loss of corporate reputation, threats to regulatory compliance and financial impacts. There are well-known cases of sensible data loss such as CardSystems Solutions where Credit card breach exposed 40 million accounts in 2005. In addition, government laws oblige disk makers to have a method to secure data. Nowadays, there are four methods to secure data:

- physical drive destruction;
- degaussian;
- secure erase;
- encryption.

In the following we will see what these methods are and how they are applied to SSD and HDD.

To prevent data from recovery, disks can be broken up to microscopic pieces. However, such physical destruction is not absolute if any remaining disk pieces are larger than a single 512-byte block. In case of HDD this is not easy and a magnetic microscopy is able to recover the data. In case of SSD it is easier to destroy the physical component but this method is old and not used.

Degaussian uses magnetic field to erase data stored on HDD. Degaussers create high intensity magnetic fields that erase all the magnetic recordings in a hard disk drive, including the sector header information on drive data tracks. Like physical destruction, once this procedure is applied, the disk is no longer usable. However, as the storage density increases, higher magnetic fields are required, so that old degaussers cannot be reused in modern HDD. In addition, new perpendicular recording drives my not be erasable by present degaussers designed for past longitudinal recording drives. Due to the different physical media, degaussian procedure is not applicable to SSD. However, there are companies [1] that build a self-destructive SSD by applying an over-current to the NAND Flash memories.

As regards erase, four security levels are defined: weak erase (deleting files), block erase (overwrite by external software), secure erase and fast secure erase. There is a big difference in terms of security achieved and time required by these four levels as depicted in Fig. 12.1.

Sanitation of HDD through erase is not easy, because when we delete a file, we just remove its name from the directory structure. The user data remain on the drive where they can be retrieved until the sectors are overwritten by new data. Even reformatting the drive only file directories and links among sectors are cleared, but the user data remain and can be recovered. Moreover, software utilities that overwrite files are susceptible to error or malicious virus attack and require constant update.

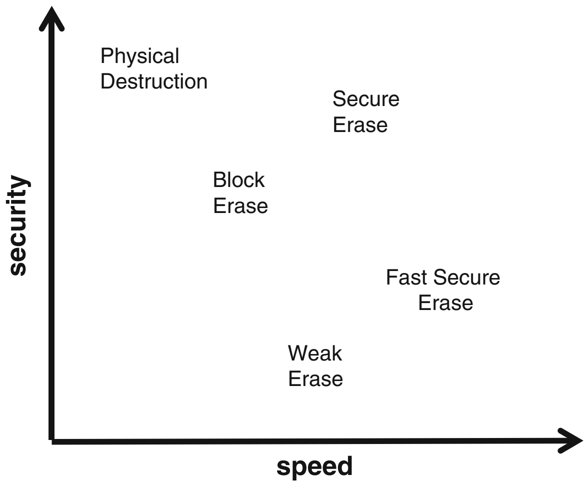


Fig. 12.1 Trade-off between speed and security among different security levels

Secure Erase (SE) is the name given to a set of commands available in PATA and SATA hard drives. The Secure Erase commands are used as data sanitization method to completely overwrite all the data on a hard drive. The method is very simple: it writes a binary one or zero in all the locations.

After SE file recovery programs will not be able to extract data from the drive. Secure Erase is a simple addition to the existing "format drive" command and adds no cost to hard disk drives. Usually, HDDs ask for multiple SE operations; in the SSD case a single erase should be enough because data are erased in blocks. The bad news is that the operating system is not aware of where data are physically stored; only the Flash controller inside the SSD knows the logical-to-physical mapping (Chap. 2). Recently it has been published [2] a study on limitations about secure erase applied to SSDs.

- First, ATA and SCSI built-in commands are effective, but manufacturers sometimes implement them incorrectly. Moreover, sometimes they are not implemented in SSDs.
- Second, overwriting the entire visible address space of an SSD twice is usually, but not always, sufficient to sanitize the drive. In addition, due to the Firmware Transaction Layer (FTL) (Chap. 2) the procedure is more complex and time consuming compared to HDD.
- Third, none of the existing hard drive techniques for individual file sanitization are effective on SSDs.

Even if there is a lot of effort on developing a stronger secure erase for SSDs, nowadays encryption is the preferred method. Encryption should be used on the drive since beginning of life: when we want to destroy data, it is enough to delete all the keys in order to be sure that all the data are un-recoverable.

The next section walks the reader through the encryption world before discussing encryption applied to SSDs.

12.2 Introduction to Cryptography

The fascinating art of cryptography was born as soon as the civilized man began to communicate information to another man. In fact, quite at the beginning he felt the need of secrecy or privacy, so that if Alessia wants to send a message to Rino, she doesn't want Kam, who heard the message, to understand its meaning.

There are evidence of cryptographic schemes in the ancient Jew population and their atbash schemes, the Spartans with their scytale (Fig. 12.2) but the first "published" encryption scheme is the Caesar ciphrary invented by emperor Caius Julius Caesar. From then a number of different schemes were used during the ages, till the popular Enigma during the Second World War, used by the Germans to send encrypted messages to U-boots (Fig. 12.3).

Together with encryption methods, more and more efforts were put on the opposite side of the story: the codebreakers that invented the science of cryptanalysis. The most famous were the scientists of Bletchey park (Alan Touring was one of them) that were able to decrypt the messages sent with Enigma. This was a key point in the defeat of Germany in the Second World War.

Modern encryption science was born in 1949 with Claude Shannon [3], the father of Information theory with the paper "Communication Theory of Secrecy Systems". After that, the encryption science was pushed by military industry and then applied to telephone lines, computer networks, financial transactions and so on.

More and more complex schemes were discovered and then analyzed to find their weakness. The next sub-sections introduce the basic concept of a cryptographic system and how it is possible to find if it is secure or not. Last sub-section describes encryption applied to MAC (Message Authentication Code) and digital signatures.

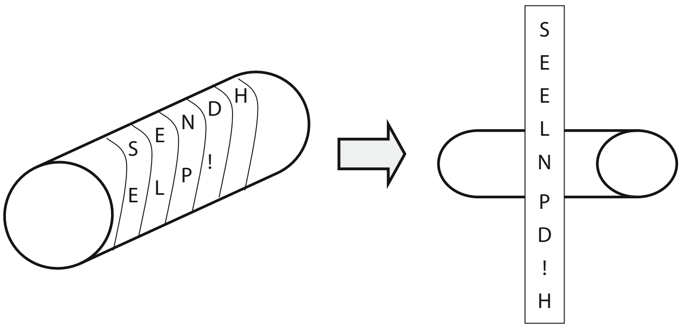


Fig. 12.2 The scytale used by spartans to encrypt codes: it was a wooden stick used to roll the message to be encrypted

Fig. 12.3 The enigma machine

12.2.1 Basic Concepts

As the name cryptography suggests (from the greek kryptos = hidden and graphia = written language) the purpose of this science is to hide an information under an apparent random message.

Let's say Alessia wants to send a message to Rino and be sure that listener Kam doesn't understand the message (Fig. 12.4). The message Alessia wants to send is called *plaintext*. She applies an *encryption* function, that generally involves a *key*, to the plaintext in order to get a *ciphertext* to be sent to Rino. On the other side, Rino receives the ciphertext and applies his *decryption* function, that generally involves another key, in order to recover the original plaintext. If Kam hears the ciphertext, he is unable to recover the plaintext because he hasn't the key.

A basic example of an encryption scheme is based on letter substitution. Figure 12.5 shows the Caesar code (Fig. 12.5). The key is the width of the rotation, 3 in this example.

Alessia wants to send the plaintext "Caesar" to Rino. She uses her key (rotation of 3 positions) to obtain the cyphertext "Zxbpxo". Rino receives the message and rotates back of 3 positions to read the original message.

This is a very simple example where Rino and Alessia have the same key. Of course, a number of modifications have been introduced in order to have a different number of rotations for each letter of the message (Vigenère codes) or different keys for Alessia and Rino, or different encryption methods.

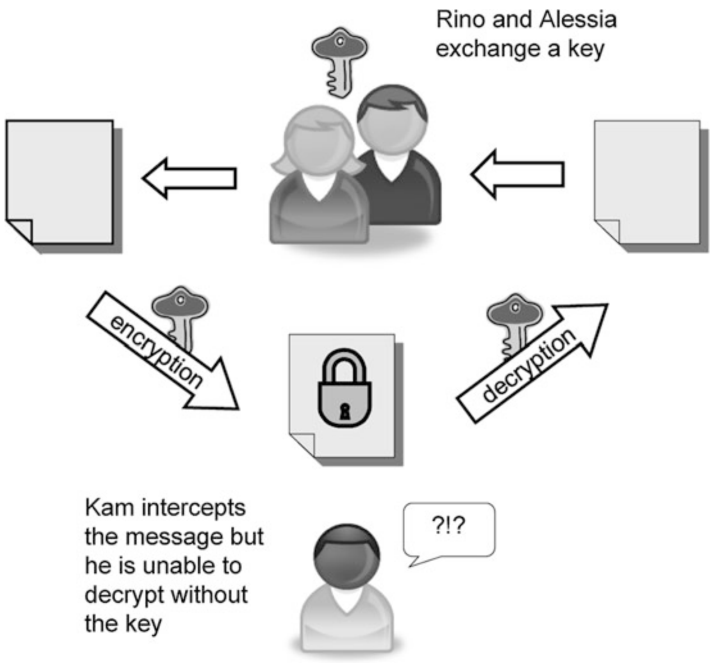
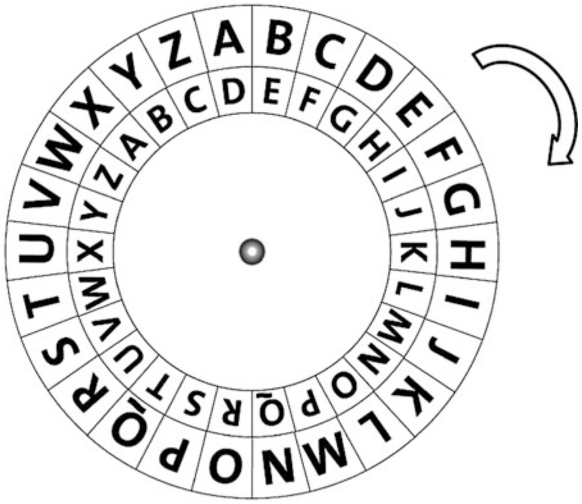

Rino and Alessia exchange a key
encryption
decryption
Kam intercepts the message but he is unable to decrypt without the key
?!?

Fig. 12.4 A cryptographic system

Fig. 12.5 Caesar codes



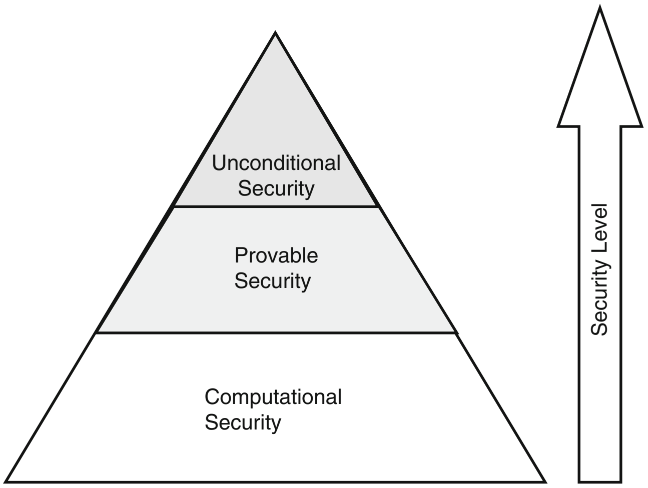


Fig. 12.6 The pyramid of security

However, it is necessary to have a "metric" to evaluate the security of a cryptosystem.

Shannon was able to give a mathematical structure to the encryption science, first of all by evaluating the secrecy of a system. In fact there are different levels of security of a cryptosystem as shown in Fig. 12.6.

The low level of security is the *computational security*. This is a measure of the computational effort required to break a cryptosystem; in other words a system is considered computational secure if it requires at least *N* operations. However, given the speed of the technology evolution, what is secure today will unlikely be secure tomorrow. Moreover, there aren't any practical secure cryptosystems based on this definition. The problem is that people study the computational security of a system under a specific attack, but this does not guarantee its security under another attack.

The second level of security is the *provable security*. A cryptosystem is said to be provable secure if its construction is based on a very difficult mathematical problem, not yet theoretically solved. For example, as it will be discussed later, RSA system is based on integer factorization. Until now, there aren't any methods that can easily factorize an integer. If some day a method will be found, RSA will be easily broken, but until then it is provable secure.

The highest level of security is the *unconditional security*. In this case there are no bounds on the computational effort that Kam can use: the cryptosystem can't be broken even with infinite computational resources.

We won't go through all the mathematical description of this analysis, but we report here only an interesting result: the Vigenère cipher is unconditional secure if the keyword has the same length of the plaintext. It is even more secure if the key is used only once.

```
          H         E         L         L         O  message
   7  (H)    4  (E)   11  (L)   11  (L)   14  (O) message
+ 23  (X)   12  (M)    2  (C)   10  (K)   11  (L) key
= 30        16        13        21        25      message + key
=  4  (E)   16  (Q)   13  (N)   21  (V)   25  (Z) message + key
(mod 26)
       E         Q         N         V         Z  → ciphertext
```

Fig. 12.7 Example of One-Time pad encryption

Nowadays, there is only one cryptosystem known as unconditional secure: the One-Time Pad. Historically, this encryption method was used by KGB agents. The system was so secure that some messages have been decrypted only when agents re-used the same key more than once or some spies have been arrested and revealed the keys. We explain this method with an example.

Alessia wants to send the message "hello" to Rino. They have the same pads of keys to be used only one time and they decided for "xmckl" (same length of the message). The encryption method follows Fig. 12.7. Based on the alphabet, letters are translated in numbers, and the sum of message and key gives the ciphertext.

The sum is performed mod(26). Alessia immediately destroys the key. Rino receives the message "eqnvz", translates it in numbers, and subtracts the key to obtain the original message. At this point Rino destroys the key.

If Kam hears the cipthertext and tries to decrypt it with infinite computing power, he fill find "xmckl" as key but also "tquri" that gives the word "later" with same probability.

This is a very simple and fast encryption method, easily performed by xoring the key with either the plaintext (during encryption) or ciphertext (during decryption).

Difficulties arise in the key management: the key must be as long as the message, it must be random, it must be used only once and destroyed immediately after use. In addition, it is very difficult to distribute keys among multiple users. Especially the requirement on the key length is so difficult to achieve that different encryption methods are preferred, such as AES (Sect. 12.3) even if not unconditional secure.

This discussion leads us to the problem of the key. In fact, till few years ago all those methods were based on a symmetric encryption [4, 5]. In other words it is very easy to understand the key that Rino has, given Alessia's key. In particular most of the time the key is the same. It follows that this key must be secret otherwise all the messages will be decrypted by Kam.

This leads us to some kind of paradox: we want to send secret messages but we must exchange a secure key over a secure channel. This is what happens in internet, when we are accessing a secure channel (e.g. home banking, credit card payment, etc.): we are exchanging a secure key to encrypt and decrypt messages. In financial transactions, however, we have the logistic problem of keys distribution. In other words, a bank must provide a different key to each user: handling of all these keys is translated in time and cost. It's not the purpose of this chapter to address the

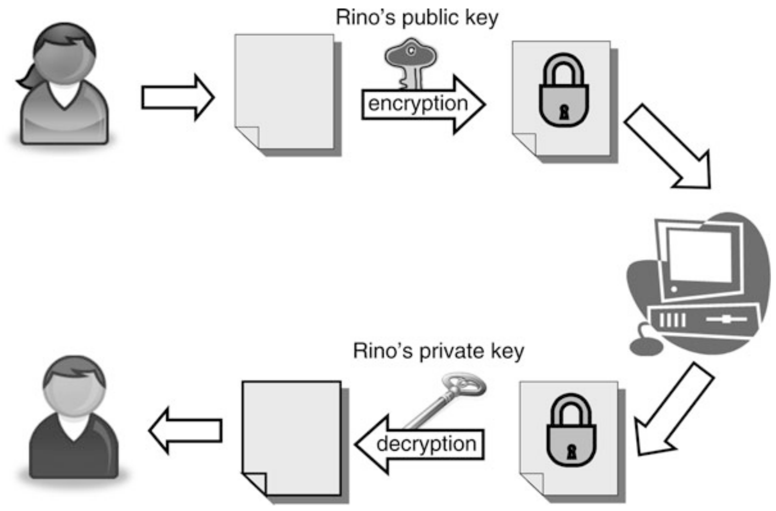


Fig. 12.8 The asymmetric cryptosystem

problem of keys distribution. One way to solve it is the use of the Diffie-Hellman algorithm, i.e. an asymmetric encryption. The interested reader can refer to [5–8].

In order to overcome the problem of the key exchange, the *public-key cryptosystem* has been developed. The idea behind is that it might be unfeasible to find out Alessia's key *d*, given Rino's key *k*. It follows that Rino can publish his key and Alessia uses it to encrypt the message (Fig. 12.8). The sent message is received by Rino, that now uses his private key to decrypt the message.

Observe that this method can also be reversed, that is Rino can use his private key to encrypt the message and sends it to Alessia. Alessia uses Rino's public key to decrypt the message. In this case everyone can decrypt the message, since Rino's key is public, but we are sure of the authenticity of the message, because it was encrypted using Rino's private key (Sect. 12.2.3).

The advantage of the public-key cryptosystems is that Alessia can send messages using the public key without any secret prior exchange of keys and be sure that only Rino is able to decrypt the message.

The public-key cryptosystem was first discussed by Rivest, Shamir and Adleman in 1978 with the very famous system called RSA [9]. Several systems have then be proposed, but their security remains computational. In fact, asymmetric encryption could never provide unconditional security. When Kam intercepts the ciphertext *y,* he can encrypt each possible plaintext using the public encryption rule until he finds the unique solution so that $y = e(x)$. This x is the decryption of y.

Public-key cryptosystem is based on one-way functions which are very easy to compute but very difficult to be inverted. There are a lot of functions that are believed to be one-way but never proven.

An example of such function is the factorization of an integer into two prime numbers, used in RSA. This cryptosystem can be summarized as follows:

- Rino picks up two large prime numbers p and q,
- Rino sends the number $n = p\ x\ q$ to Alessia. Everyone can see it;
- Alessia uses n to encrypt the message;
- Alessia sends the cipthertext to Rino. Everyone can see it but nobody can decrypt it;
- Rino receives the message and, knowing p and q, is able to decrypt it.

The difficulty of this algorithm is the primality test of large integers. Today only numbers with al least 300 ciphers are considered secure [10, 11].

Asymmetric cryptosystems are used in a number of different protocols like SSH, Internet Key Exchange and PGP. The main advantage is that the generation of the key pair solves the logistic problem of key distribution and the problem of authentication (Sect. 12.2.3).

These systems are not broadly used because they are too slow and can limit performances in most of the cases, like in SSDs. A solution that sometimes is adopted is to transmit the keys with a public-key cryptosystems and then switch to a symmetric cryptosystem.

12.2.2 Cryptanalysis

Let’s analyze the cryptosystem from Kam’s side. Kam is not the bad guy of the story; of course, he could be a hacker that wants to intercept our credit card but he could also be a secret agent that needs to intercept a terroristic attack. This is the reason why the government puts a lot of effort and money in finding a good code but also in breaking codes.

Cryptoanalysis science, as the name suggests (from Greek cryptos = “hidden” and analyein = “to untie”) has the purpose to break codes.

Generally speaking, we suppose that Kam knows the cryptosystem in use: this is known as the Kerckhoffs’ principle. Hence, given a ciphertext, Kam’s goal is to understand the key of the system.

Different attacks are based on the amount of information that Kam has.

- *Ciphertext only* attack: Kam knows a ciphertext or a part of it.
- *Known plaintext* attack: Kam knows plaintexts and their corresponding ciphertexts.
- *Chosen plaintext* attack: Kam can choose a set of plaintexts and encrypt them.
- *Chosen ciphertext* attack: Kam can choose a set of ciphertexts and decrypt them.

The attacks are based on available resources, i.e. computing power, storage memory, and time.

At this point we need to clarify what “break the code” means. Generally speaking, Kam wants to know the key, but if he is unable to recover the key, he could attempt a partial break of the code.

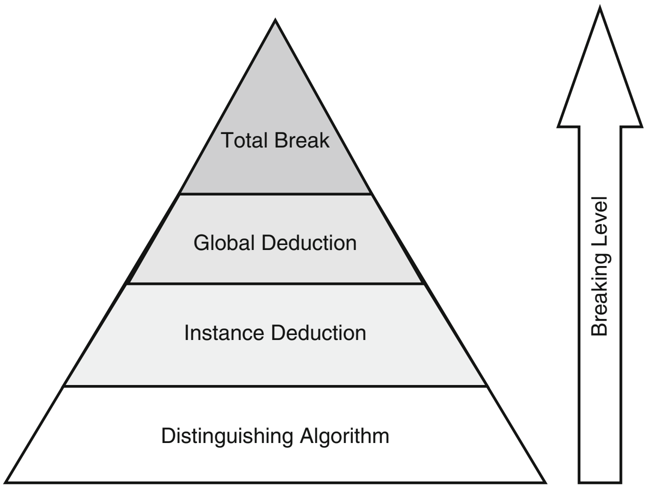


Fig. 12.9 The pyramid of codebreaking

In the pyramid of Fig. 12.9 the highest level is the *total break* where Kam understands the key. The second level is the *global deduction*: Kam does not know the key but he discovers a functionally equivalent encryption and decryption method. Then we have *instance deduction*: Kam produces additional plaintexts or ciphertexts. Finally, we have *distinguishing algorithm*: Kam is able to distinguish a ciphertext from a random permutation.

For example, if we want to discover a key of a ciphertext obtained with a Caesar code (and all the substitution cryptosystems) we can use an attack called *Frequency Analysis*. This attack is based on the analysis of the frequency of letters or group of letters in a particular language. Typical distribution of letters in English language is shown in Fig. 12.10.

When Kam intercepts a message, he can easily find out the most frequent letter and decrypts it as either E or A or T, but unlikely as Z. By analyzing letter's frequency, and group of letters together, he can recover the plaintext.

An evolution of this attack, used in more complex cryptosystems, as Vigenère codes, is called *Kasiski method*. The purpose of this attack is to understand the length of the key and then reduce the ciphertext to a cipher substitution that can be analyzed with frequency analysis attack. The method was discovered by Kasiski in 1863 and independently by Babbage in 1846.

The method is based on these observations:

1. two identical segments of plaintext will be encrypted to the same ciphertext whenever their occurrence in the plaintext is d position apart;
2. if we observe two identical segments of ciphertext, each of length at least 3, there is a good chance that they correspond to identical segment of plaintext.

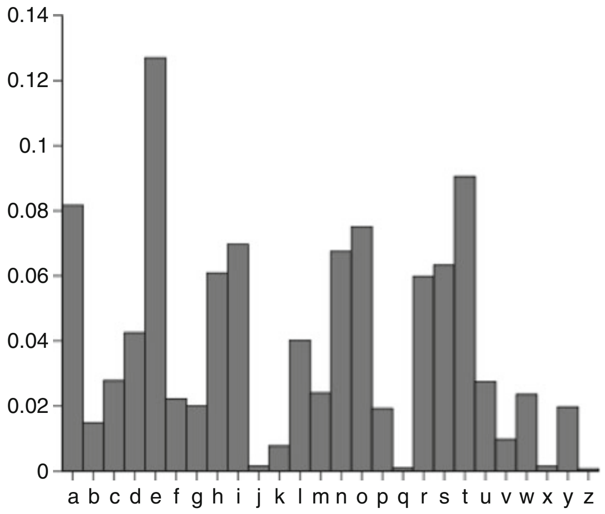


Fig. 12.10 Typical frequency distribution of letters in an English text

Hence, the first thing to do is to find groups of equal characters, of at least 3 letters, and record their position. Suppose that in a text we have the same group of 3 letters separated of 165, 235, 275 and 285 positions. The greatest common divisor is 5 and it is very likely to be the keyword length. Now that the length is known, every group of 5 letters can be broken via the frequency analysis attack.

These two attacks (Frequency analysis and Kasiski method) are based on linguistic statistics, but as the cryptosystems complexity increases, more mathematics and computational power are required.

There are cases where codes are broken not because of the weakness of the code itself, but because of an erroneous or insecure usage. For example, encrypting two messages with the same key is an insecure process, the messages are said to be *in depth*: Kam gains a lot of information by analyzing more than one ciphertext encrypted with the same key.

Another weakness that historically helped breaking a code is the *indicator* transmission with the Enigma machine.

The key was kept constant for a period of time, generally a day. However, a different rotor position (Fig. 12.11) was used for each message, a sort of initialization message.

The starting position of these rotors was transmitted just before the ciphertext. It was design weakness and operator sloppiness in this indicator procedure that broke Enigma. The procedure works as follows: the operator sets the rotor as indicated by his list to the initial setting, i.e. to some specific combination of letters (e.g. RDKP) visible in the rotor window. Then the operator chooses a starting position for his message which becomes the indicator to be sent with the message (e.g. ABGY). He then types ABGY two times in the machine so that the message is encoded twice, for example in SWTHNQLM. He transmits this string and then the encrypted plaintext.

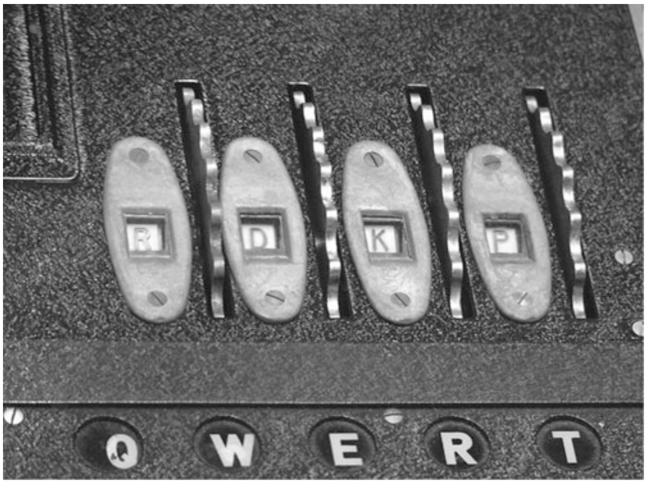


Fig. 12.11 An example of a rotor position to send an indicator for the Enigma machine

At the receiver side, the operator sets the rotor in the initial settings and then types SWTHNQLM. Immediately RDKP pops up, the receiver sets the rotors in that position and starts typing the ciphertext to obtain the plaintext.

The weakness of this scheme is that it is used as a worldwide setting. Moreover, the repetition of this value causes a security flaw.

The attacks used for symmetric-type cryptosystems are based on difficult mathematical problems. The most obvious way to attack this system is solving those mathematical problems. In case of RSA cryptosystem, 3 algorithms seem to be the most effective to factorize integers: quadratic sieve, elliptic-curve factorization and number field sieve [12–15].

Today Cryptanalysis tries to break RSA encryption by using a huge computational power. In 1980 10^{12} CPU operations were required to factor a number of 50 digits. The same number of operations was required to factor a number of 75 digits in 1984. Nowadays, it is possible to factor a number of 150 digits. Given the speed trend of CPUs, more and more digits are required to secure RSA cryptosystems.

12.2.3 Hash Functions

The previous sections described the use of encryption for the general case where Alessia wants to send a message to Rino and doesn't want Kam, who hears the message, to understand. However, encryption is used to solve also a number of other issues in telecommunication world. In this section we address these issues and how they are solved.

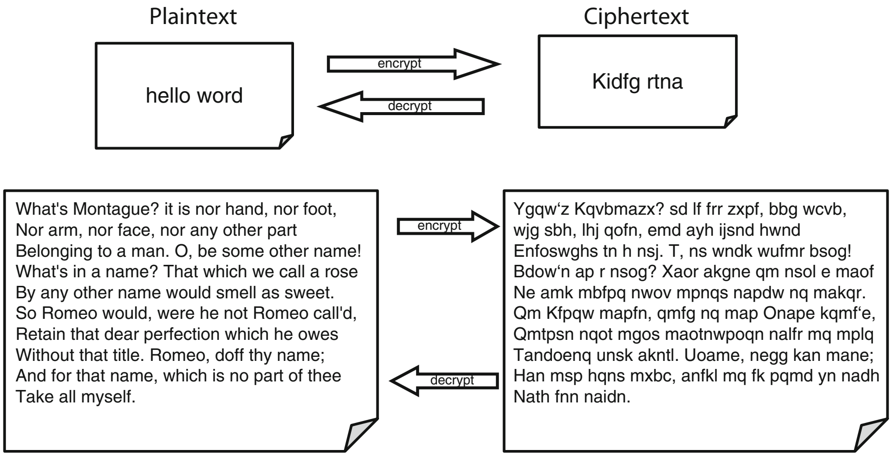

Plaintext
hello word
encrypt
decrypt
Ciphertext
Kidfg rtna
What's Montague? it is nor hand, nor foot,
Nor arm, nor face, nor any other part
Belonging to a man. O, be some other name!
What's in a name? That which we call a rose
By any other name would smell as sweet.
So Romeo would, were he not Romeo call'd,
Retain that dear perfection which he owes
Without that title. Romeo, doff thy name;
And for that name, which is no part of thee
Take all myself.
encrypt
decrypt
Ygqw'z Kqvbmazx? sd lf frr zxpf, bbg wcvb,
wjg sbh, lhj qofn, emd ayh ijsnd hwnd
Enfoswghs tn h nsj. T, ns wndk wufmr bsog!
Bdow'n ap r nsog? Xaor akgne qm nsol e maof
Ne amk mbfpq nwov mpnqs napdw nq makqr.
Qm Kfpqw mapfn, qmfg nq map Onape kqmf'e,
Qmtpsn nqot mgos maotnwpoqn nalfr mq mplq
Tandoenq unsk akntl. Uoame, negg kan mane;
Han msp hqns mxbc, anfkl mq fk pqmd yn nadh
Nath fnn naidn.

Fig. 12.12 General properties of encryption

The hash function is any algorithm that maps a large bunch of data of variable length to a smaller set of data of fixed length [16, 17]. A cryptographic hash function is used to provide data integrity: in some way, it builds a fingerprint of data, so that when data change, the fingerprint is not valid anymore. It is also used when data are stored in an insecure location: fingerprints are re-computed from time to time to verify that they have not changed. This fingerprint is usually called *digest*. With a good hash function it is easy to compute a digest given a message, but it is unfeasible to find the message given the hash; in other words it is an unidirectional function. This is very different from encryption where we encrypt and decrypt, and the ciphertext has the same length of plaintext (Fig. 12.12). On the contrary, hash functions are unidirectional and the length of the digest is fixed despite the length of the message (Fig. 12.13).

Being unidirectional is a basic requirement for security. If a hash function hasn't this property, we say that it has *preimage resistance*. Another bad property is called *collision resistance*: given a message and its digest, there is another message with the same digest.

These bad properties imply that somebody can change a message without changing its digest. As discussed, hash functions are used to verify data integrity. For example, when we download a file from the web, our PC computes the hash function and compares it with the one published on the website as data integrity check [18, 19]. Please note that the digest is not visible on the screen but embedded in the properties of the file.

Another application is the password storage (Fig. 12.14).

PCs do not store cleartext password, because it would be too dangerous if the personal computer is stolen or somebody has access to its storage area. Therefore, the hash function of the password is stored, since it is unfeasible to recover the cleartext password from the hash. On the following login, the system re-computes

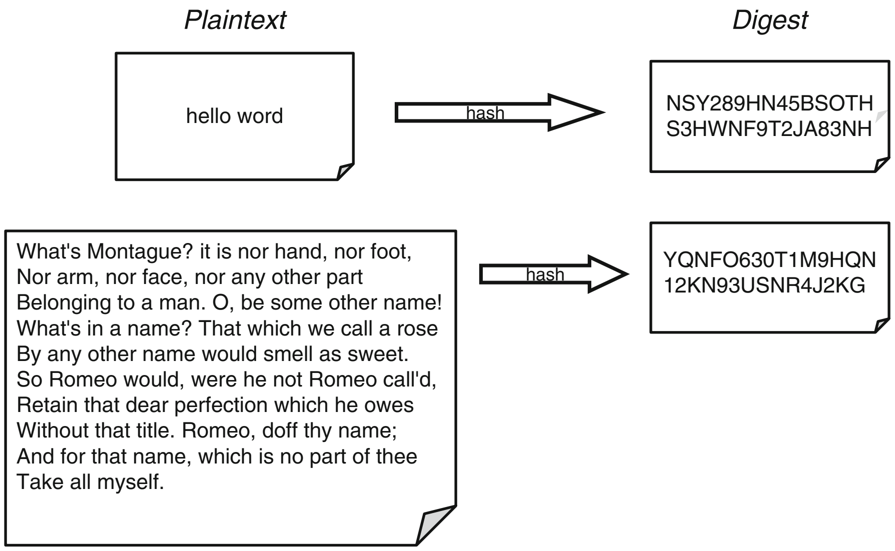

Plaintext
Digest
hello word
hash
NSY289HN45BSOTH S3HWNF9T2JA83NH
What's Montague? it is nor hand, nor foot,
Nor arm, nor face, nor any other part
Belonging to a man. O, be some other name!
What's in a name? That which we call a rose
By any other name would smell as sweet.
So Romeo would, were he not Romeo call'd,
Retain that dear perfection which he owes
Without that title. Romeo, doff thy name;
And for that name, which is no part of thee
Take all myself.
hash
YQNFO630T1M9HQN 12KN93USNR4J2KG

Fig. 12.13 General properties of hash function

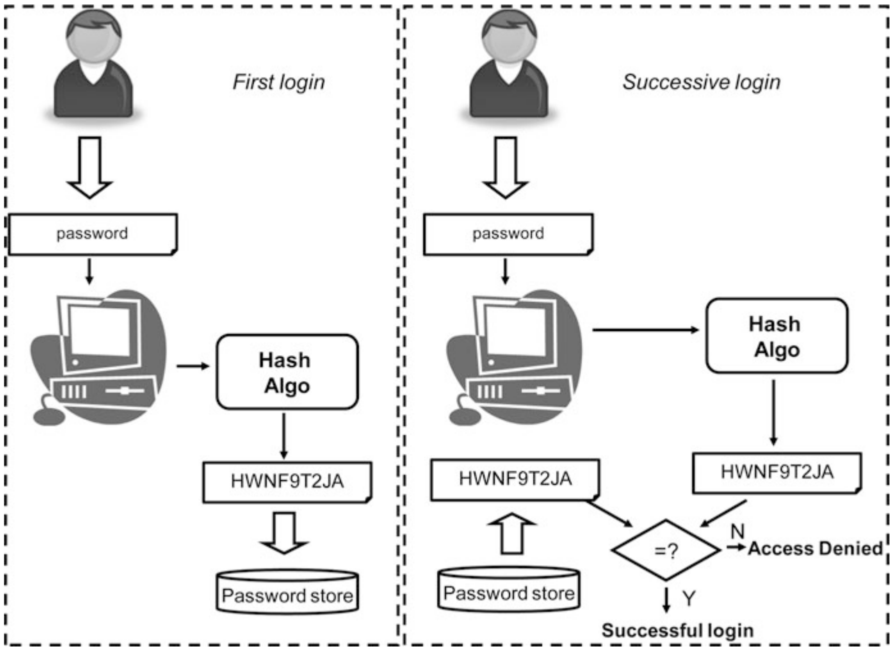


Fig. 12.14 Hash function in the password storage

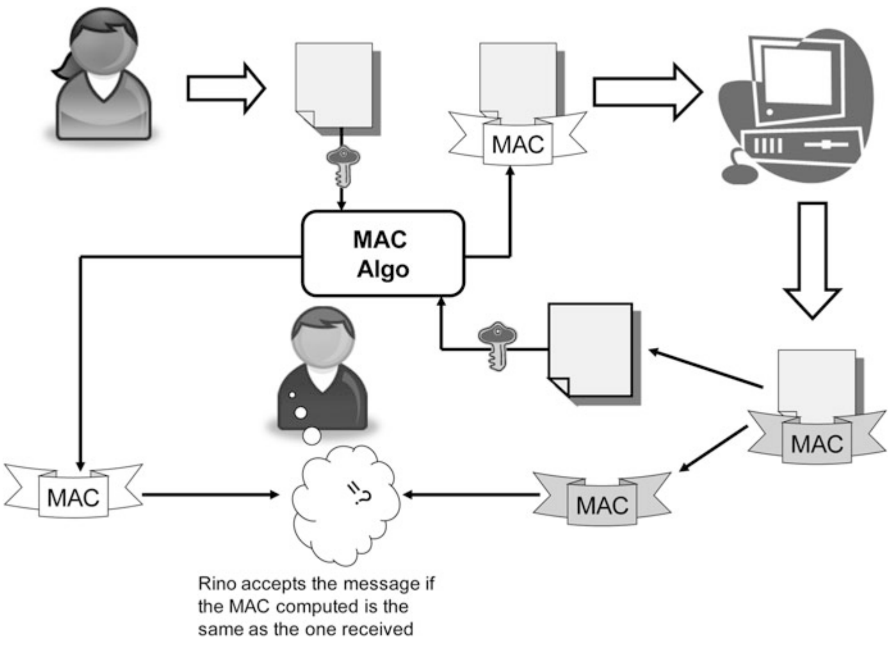

MAC
Algo
MAC
MAC
MAC
MAC
=?
Rino accepts the message if the MAC computed is the same as the one received

Fig. 12.15 Block scheme of the MAC usage

the hash for the cleartext password and compares it with the stored one. Since it is impossible to have two messages with the same hash, the user must have typed the correct password to login in the system.

A special application is the *Message Authentication Code* (MAC). In this case keyed cryptographic functions are used [20–22]. These functions have stringent security requirements: specifically, even if the attacker is able to generate MACs for some messages, the attacker cannot guess the MAC for other messages without performing unfeasible amounts of computations.

The power of the MAC is that it guarantees both data integrity and authenticity of the message (Fig. 12.15). Moreover, since MACs require the same key for both receiver and sender, MAC functions are similar to symmetric encryption functions.

Another important usage of the hash function is the *digital signature*. There are three reasons to use a digital signature.

1. *Authentication*: this is the same reason why we sign documents. We want to authenticate the source of the messages. This is especially true in financial transactions.
2. *Integrity*: sender and receiver want to be sure that the message has not been corrupted during transition, even if it has been encrypted. Since there is no valid way to change a message without changing its signature, a non-valid signature detects a corrupted message.
3. *Non-repudiation*: once we have signed a document, we can't later deny it.

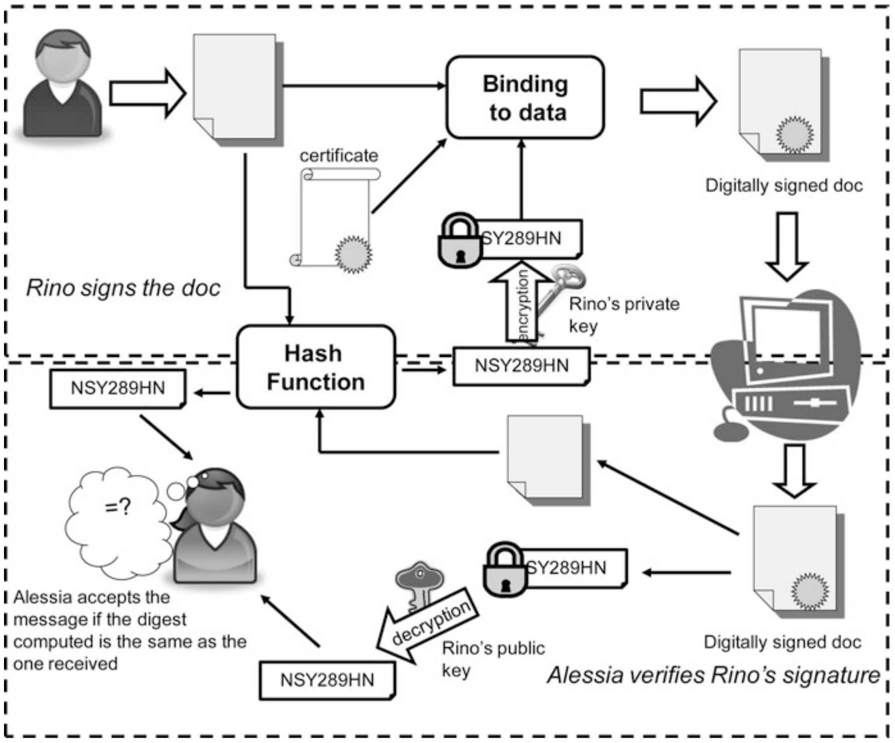


Fig. 12.16 Block diagram of the digital signature

Although the discussion is very complex about how to digitally sign a document, high level blocks are sketched in Fig. 12.16.

First of all, the asymmetric encryption is used. This is because everybody should be able to decrypt, but nobody could modify the signed document. The hash function is computed on data. At this point the resulting digest is encrypted using Rino's private key, in order to produce the signature. Finally, there is a certificate that binds the signature to the document so that they can't be split.

On the receiver side, Alessia reads the signed message and decrypts the digest using Rino's public key to obtain the received cleartext digest.

She computes the hash function on the received document and compares the result with the obtained cleartext digest. If they are equal, she accepts the message from Rino, otherwise she repudiates it.

12.3 AES

Advanced Encryption Standard (AES), or its variant XTS-AES, is the encryption system generally used in Solid State Disks. It is an iterative symmetric encryption method, it supports 128, 192 and 256 bits as key length, and is available worldwide on a royalty-free basis. The algorithm was originally proposed by Daemen and Rijmen (called Rijndael) and it was published in the Federal Register on December 4, 2001 [23].

AES is iterative and the number of iterations (rounds) Nr depends on the key length: $Nr = 10$ if the key length is 128, $Nr = 12$ if the key length is 192 and $Nr = 14$ if the key length is 256.

AES works on a basic unit called *state*. Each state consists of a matrix: 4×4 bytes in the 128 case, and 4×8 bytes in the 256 case. We can split the algorithm in two parts: key generation and core algorithm [24–27].

12.3.1 Key Generator

Every key is split in 32-bit word. We have 8 words in the 256 case. At iteration i we have 32-bit word as input and 32-bit word as output. The algorithm proceeds as follows:

1. copy the input over the output;
2. *rotate operation* to rotate 8 bits to the left;
3. apply *S-box* to the 4 bytes individually;
4. on the first (leftmost) byte of the output word, XOR the byte with $2^{(i\text{-}1)}$. In other words, perform the *rcon operation* with i as the input, and XOR the *rcon* output with the first byte of the output word.

As the name may suggest, the rotate operation cyclically shifts bytes to the left:

$$\text{rotate}\,(B_0, B_1, B_2, B_3) = (B_1, B_2, B_3, B_0).$$

The *rcon* operation is equal to

$$rcon(i) = x^{i\text{-}1} \text{ in } GF(2^8) \text{ or } rcon(i) = x^{i\text{-}1} \bmod x^8 + x^4 + x^3 + x + 1 \text{ in } GF(2).$$

For example rcon(1) = 1, rcon(4) = 3 and rcon(9) = 27.

Finally, we define the S-box in Fig. 12.17: it indicates a substitution to be made for each byte combination.

For example: S-box(9c) = de or S-box(f2) = 89.

12.3.2 AES Algorithm Core

Once we have defined how the keys and sub-keys are computed we now describe the AES algorithm.

- State = plaintext. Perform the *AddRoundKey* operation between the state and the key.
- For each iteration:
  - Execute *SubBytes*
  - Execute *ShiftRows*

	x0	x1	x2	x3	x4	x5	x6	x7	x8	x9	xa	xb	xc	xd	xe	xf
0x	63	7c	77	7b	f2	6b	6f	c5	30	01	67	2b	fe	d7	ab	76
1x	ca	82	c9	7d	fa	59	47	f0	ad	d4	a2	af	9c	a4	72	c0
2x	b7	fd	93	26	36	3f	f7	cc	34	a5	e5	f1	71	d8	31	15
3x	04	c7	23	c3	18	96	05	9a	07	12	80	e2	eb	27	b2	75
4x	09	83	2c	1a	1b	6e	5a	a0	52	3b	d6	b3	29	e3	2f	84
5x	53	d1	00	ed	20	fc	b1	5b	6a	cb	be	39	4a	4c	58	cf
6x	d0	ef	aa	fb	43	4d	33	85	45	f9	02	7f	50	3c	9f	a8
7x	51	a3	40	8f	92	9d	38	f5	bc	b6	da	21	10	ff	f3	d2
8x	cd	0c	13	ec	5f	97	44	17	c4	a7	7e	3d	64	5d	19	73
9x	60	81	4f	dc	22	2a	90	88	46	ee	b8	14	de	5e	0b	db
ax	e0	32	3a	0a	49	06	24	5c	c2	d3	ac	62	91	95	e4	79
bx	e7	c8	37	6d	8d	d5	4e	a9	6c	56	f4	ea	65	7a	ae	08
cx	ba	78	25	2e	1c	a6	b4	c6	e8	dd	74	1f	4b	bd	8b	8a
dx	70	e3	b5	66	48	03	f6	0e	61	35	57	b9	86	c1	1d	9e
ex	e1	f8	98	11	69	d9	8e	94	9b	1e	87	e9	ce	55	28	df
fx	8c	a1	89	0d	bf	e6	42	68	41	99	2d	0f	b0	54	bb	16

Fig. 12.17 S-box for AES

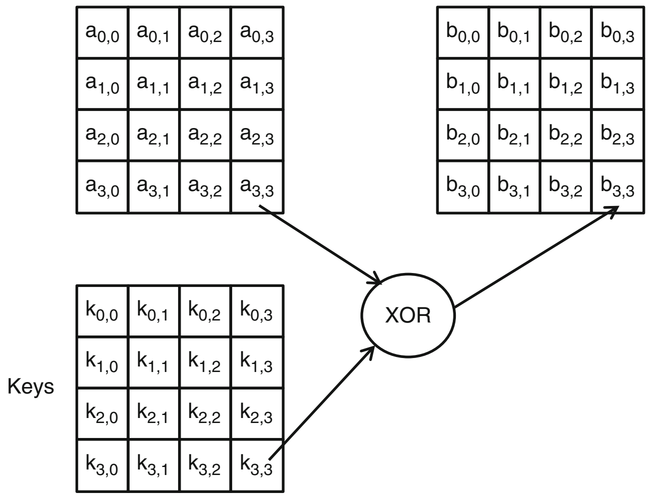

$a_{0,0}$ $a_{0,1}$ $a_{0,2}$ $a_{0,3}$
$a_{1,0}$ $a_{1,1}$ $a_{1,2}$ $a_{1,3}$
$a_{2,0}$ $a_{2,1}$ $a_{2,2}$ $a_{2,3}$
$a_{3,0}$ $a_{3,1}$ $a_{3,2}$ $a_{3,3}$
$b_{0,0}$ $b_{0,1}$ $b_{0,2}$ $b_{0,3}$
$b_{1,0}$ $b_{1,1}$ $b_{1,2}$ $b_{1,3}$
$b_{2,0}$ $b_{2,1}$ $b_{2,2}$ $b_{2,3}$
$b_{3,0}$ $b_{3,1}$ $b_{3,2}$ $b_{3,3}$
XOR
Keys
$k_{0,0}$ $k_{0,1}$ $k_{0,2}$ $k_{0,3}$
$k_{1,0}$ $k_{1,1}$ $k_{1,2}$ $k_{1,3}$
$k_{2,0}$ $k_{2,1}$ $k_{2,2}$ $k_{2,3}$
$k_{3,0}$ $k_{3,1}$ $k_{3,2}$ $k_{3,3}$

Fig. 12.18 Representation of the AddRoundKey operation

  - Execute *MixColumns*
  - Execute *AddRoundKey*
  - Execute *SubBytes*
  - Execute *ShiftRows*
  - Execute *AddRoundKey*

- The resulting ciphertext = State.

The AddRoundKey operation is simply the XOR (Fig. 12.18) between the State and the subkey obtained at that point using the key generator.

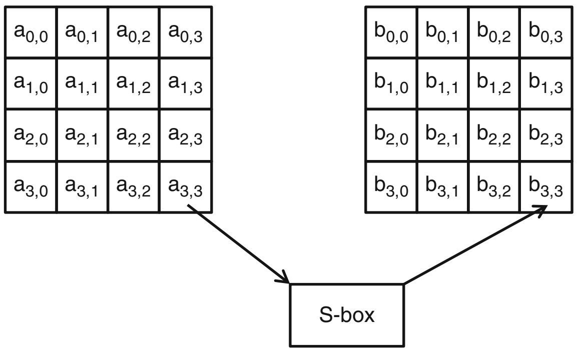

$a_{0,0}$ $a_{0,1}$ $a_{0,2}$ $a_{0,3}$
$a_{1,0}$ $a_{1,1}$ $a_{1,2}$ $a_{1,3}$
$a_{2,0}$ $a_{2,1}$ $a_{2,2}$ $a_{2,3}$
$a_{3,0}$ $a_{3,1}$ $a_{3,2}$ $a_{3,3}$
$b_{0,0}$ $b_{0,1}$ $b_{0,2}$ $b_{0,3}$
$b_{1,0}$ $b_{1,1}$ $b_{1,2}$ $b_{1,3}$
$b_{2,0}$ $b_{2,1}$ $b_{2,2}$ $b_{2,3}$
$b_{3,0}$ $b_{3,1}$ $b_{3,2}$ $b_{3,3}$
S-box

Fig. 12.19 Representation of the SubBytes operation

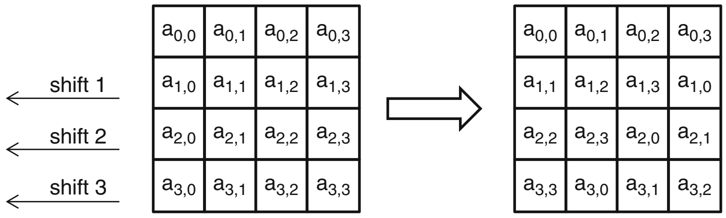

shift 1
shift 2
shift 3
$a_{0,0}$ $a_{0,1}$ $a_{0,2}$ $a_{0,3}$
$a_{1,0}$ $a_{1,1}$ $a_{1,2}$ $a_{1,3}$
$a_{2,0}$ $a_{2,1}$ $a_{2,2}$ $a_{2,3}$
$a_{3,0}$ $a_{3,1}$ $a_{3,2}$ $a_{3,3}$
$a_{0,0}$ $a_{0,1}$ $a_{0,2}$ $a_{0,3}$
$a_{1,1}$ $a_{1,2}$ $a_{1,3}$ $a_{1,0}$
$a_{2,2}$ $a_{2,3}$ $a_{2,0}$ $a_{2,1}$
$a_{3,3}$ $a_{3,0}$ $a_{3,1}$ $a_{3,2}$

Fig. 12.20 Representation of the ShiftRow operation

In the SubBytes step, each byte in the state matrix is replaced with a SubByte using an 8-bit substitution box, the S-box (Fig. 12.19). This operation provides the non-linearity in the cipher.

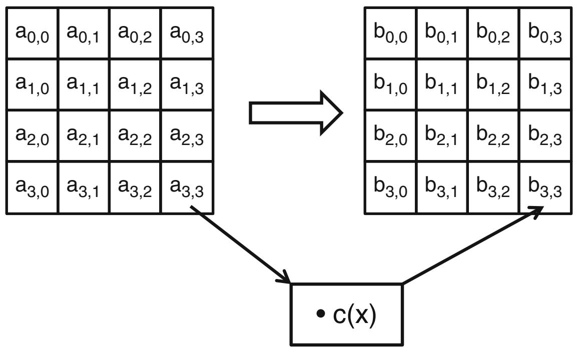
The ShiftRow operation operates on the rows of the State. Each byte of the row is cyclically shifted to the left by some locations. The first row does not shift, the second row shifts by one location, the third row by two locations and so on and so forth (Fig. 12.20).

In the MixColumn operation four bytes of each column of the state are combined using an invertible linear transformation. Together with the ShiftRow operation it provides *diffusion*, i.e. non-uniformity of the ciphertext.

Each column is multiplied by a known matrix which is

$$\begin{pmatrix} 2 & 3 & 1 & 1 \\ 1 & 2 & 3 & 1 \\ 1 & 1 & 2 & 3 \\ 3 & 1 & 1 & 2 \end{pmatrix}$$

in the 128 case. Multiplication by 1 means no change, multiplication by 2 means shifting to the left, and multiplication by 3 means shifting to the left and then performing XOR with the initial unshifted value. After shifting, a conditional XOR with 0x1B should be performed if the shifted value is larger than 0xFF. This operation is represented in Fig. 12.21.



$a_{0,0}$ $a_{0,1}$ $a_{0,2}$ $a_{0,3}$
$a_{1,0}$ $a_{1,1}$ $a_{1,2}$ $a_{1,3}$
$a_{2,0}$ $a_{2,1}$ $a_{2,2}$ $a_{2,3}$
$a_{3,0}$ $a_{3,1}$ $a_{3,2}$ $a_{3,3}$
$b_{0,0}$ $b_{0,1}$ $b_{0,2}$ $b_{0,3}$
$b_{1,0}$ $b_{1,1}$ $b_{1,2}$ $b_{1,3}$
$b_{2,0}$ $b_{2,1}$ $b_{2,2}$ $b_{2,3}$
$b_{3,0}$ $b_{3,1}$ $b_{3,2}$ $b_{3,3}$
• c(x)

Fig. 12.21 Representation of the MixColumn operation

So far there aren't any known successful attacks to AES. Especially AES-256 is considered very secure, because all the operations are studied to mix data and avoid any linearity or uniformity.

12.4 SSD Security and Applications

As described in Sect. 12.1, SSDs are gaining popularity, but security is a hard matter. More and more companies build military-grade SSDs, protecting sensitive data from environmental and human threats. In fact, this is a very important issue in defense applications or financial applications where sensitive data are treated.

SSD security is so difficult because they are based on industry-standard NAND Flash chips that were designed for cameras and MP3 players: these memories have no physical security hooks that prevent them from being removed from enclosures. A hacker could easily unsolder NAND chips and read data using a standard Flash programmer. Once raw data are read, corresponding files could be reassembled using data recovery software.

When the SSD is broken, we want data to be erased or unreadable before throwing away the SSD. Secure Erase command exists but it has its own drawbacks. First of all, if the SSD is broken, it could be possible that some blocks become un-erasable, but a hacker can read back data from those blocks.

In addition there isn't a mechanism to erase single files, but the entire SSD must be erased.

The logical-to-physical mapping of SSDs makes files even harder to be completely erased. In fact, the erase operation is a slow operation in NAND Flash, so it happens that files are not really erased but just "marked" as erased to avoid a drop in performance. The problem is that the file-system does not know the real blocks where data are stored. Logical-to-physical mapping is managed by the Flash controller inside the SSD. In other words, it's like saying that the file-system hasn't a full control on the block locations. In this context, the most common way to increase security is encryption, and it must be done within the SSD itself.

Here is what happens. Data are input by the host, encrypted by the Flash controller, and then stored in NAND. During read operation, data are read from NAND, decrypted and output to the host. Encryption and key generation are completely transparent to the host. In this way, when we want to make data unreadable, it is enough to erase the locations where keys are stored. This location can be a NAND block or a RAM block in the Flash controller.

As already pointed out, AES-256 or the XTS-AES-256 are generally used in SSDs. The firmware running on the Flash controller sets the first key; following keys are computed by the key generator described in Sect. 12.3. All the keys are stored in specific NAND blocks.

Finally, we can state that encryption is the first step to secure data on SSDs, and the sooner we use it the more secure system we have. While it is easy to encrypt data already stored on a HDD, because we can re-write encrypted data in the same locations, this is not so easy with SSDs. NAND storage doesn't allow to re-write data on the same locations: actually, encrypted data are stored in different locations (logical-to-physical mapping). At the end of the day, encryption must be activated when the device is fresh in order to secure data from external attacks.

References

1. www.runcore.com
2. M.Wei, L.M. Grupp, F.E. Spada, S. Swanson, Reliably erasing data from flash-based solid state drives, in *Usenix FAST 11 Conference,* San Jose, 2011
3. C. Shannon, Communication theory of secrecy systems. Bell Syst. Tech. J. **27**, 379–423 (1949)
4. O. Goldreich, *Foundations of Criptography: Basic Tools* (Cambridge University Press, Cambridge, 2001)
5. D.R. Stinson, *Cryptography: Theory and Practice* (Chapman & Hall/CRC, London, 2006)
6. W. Diffie, M.E. Hellman, Multiuser cryptographic techniques. Fed. Inf. Process. Stand. Conf. Proc. **45**, 109–112 (1979)
7. U. Maurer, S. Wolf, The Diffie-Hellman protocol. Des. Codes Cryptogr. **19**, 147–171 (2000)
8. B. Schneier, *Secrets and Lies: Digital Security in a Networked World* (Wiley, New York, 2000)
9. R.L. Rivest, A. Shamir, L. Adleman, A method for obtaining digital signatures and public key cryptosystems. Commun. ACM **21**, 120–126 (1978)
10. A.K. Lenstra, E.R. Verheaul, Selecting cryptographic key sizes. J. Cryptolo. **14**, 255–293 (2001)
11. M.O. Rabin, Probabilistic algorithms for testing primality. J. Number Theory **12**, 128–138 (1980)
12. M.J. Wiener, Cryptoanalysis of short RSA secret exponents. IEEE Trans. Inf. Theory **36**, 553–558 (1990)
13. A.K. Lenstra, Integer factoring. Des. Codes Cryptogr. **19**, 101–128 (2000)
14. D. Boneh, G. Durfee, Cryptoanalysis of RSA with private key d less than $N^{0.292}$. IEEE Trans. Inf. Theory **46**, 1339–1349 (2000)
15. D. Boneh, Twenty years of attacks on the RSA cryptosystem. Not. Am. Math. Soc. **46**, 203–213 (1999)
16. N. Ferguson, B. Schneier, *Practical Cryptography* (Wiley, New York, 2003)
17. H. Delfs, H. Knebl, *Introduction to Cryptography: Principles and Applications* (Springer, New York, Berlin, 2002)

18. R. Churchhouse, *Codes and Ciphers: Julius Caesar, the Enigma and the Internet* (Cambridge University Press, Cambridge, 2002)
19. M. Bellare, R. Canetti, H. Krawczyk, Keying hash function for message authentication. Lect. Notes Comput. Sci. **1109**, 1–15 (1996)
20. P. Preneel, P.C. Van Oorschot, On the security of iterated message authentication codes. IEEE Trans. Inf. Theory **45**, 188–199 (1999)
21. D. Pointcheval, J. Stern, Security arguments for signature schemes and blind signatures. J. Cryptol. **13**, 361–396 (2000)
22. T.P. Pedersen, Signing contracts and paying electronically. Lect.Notes Comp. Sci. **1561**, 134–157 (1999)
23. Advanced Encryption Standard in Federal Information Processing Standard (FIPS) Publication 197 (2001)
24. J. Nechvatal, E. Barker, L. Bassham, W. Burr, M. Dworkin, J. Foti, E. Roback, Report on the Development of the Advanced Encryption Standard (AES), 2 Oct 2000
25. S. Murphy, M.J.B. Robshaw, Essential algebraic structure within AES. Lect. Notes Comp. Sci. **2442**, 1–16 (2002)
26. S. Landau, Polynomials in the nation's service: using algebra to design the Advanced Encryption Standard. Am. Math. Mon. **111**, 89–117 (2004)
27. S. Landau, Standing the test of time: the data encryption standard. Not. Am. Math. Soc. **47**, 341–349 (2000)

Chapter 13
Flash Signal Processing and NAND/ReRAM SSD

K. Takeuchi

Abstract The widespread use of NAND Flash memories in SSDs has unleashed new avenues of innovation for the enterprise and client computing. System-wide architectural changes are required to make full use of the advantages of SSDs in terms of performance, reliability and power. Signal processing technologies are becoming more and more popular to countermeasure all the parasitic effects of a Flash NAND array: the first part of this chapter deals with such techniques.

On the other side, the emerging storage class memories (SCM) such as PCRAM, FeRAM, ReRAM and MRAM are becoming a viable alternative to commonly used volatile and nonvolatile memories. Being bit-alterable like DRAM and nonvolatile like a Flash memory, together with CMOS-process compatibility, these non-volatile random access memories have a potential to revolutionize various aspects of the computing platform architectures. A 3D TSV-integrated SSD with hybrid memory configuration which uses storage class memories (SCMs) and NAND Flash memories is a promising solution for the future memory system. This chapter describes the signal processing technologies and data management which realizes the high speed operation, low power consumption and high reliability of the SCM and NAND Flash integrated hybrid SSDs.

13.1 Error Prediction (EP) LDPC [1]

As the design rule shrinks, the floating gate (FG)-FG capacitive coupling among neighboring memory cells seriously degrades the memory cell reliability [2]. To enhance the error correction capability, an LDPC ECC is proposed for 1X nm

K. Takeuchi (✉)
Department of Electrical, Electronic and Communication Engineering,
Chuo University, Tokyo, Japan
e-mail: takeuchi@takeuchi-lab.org

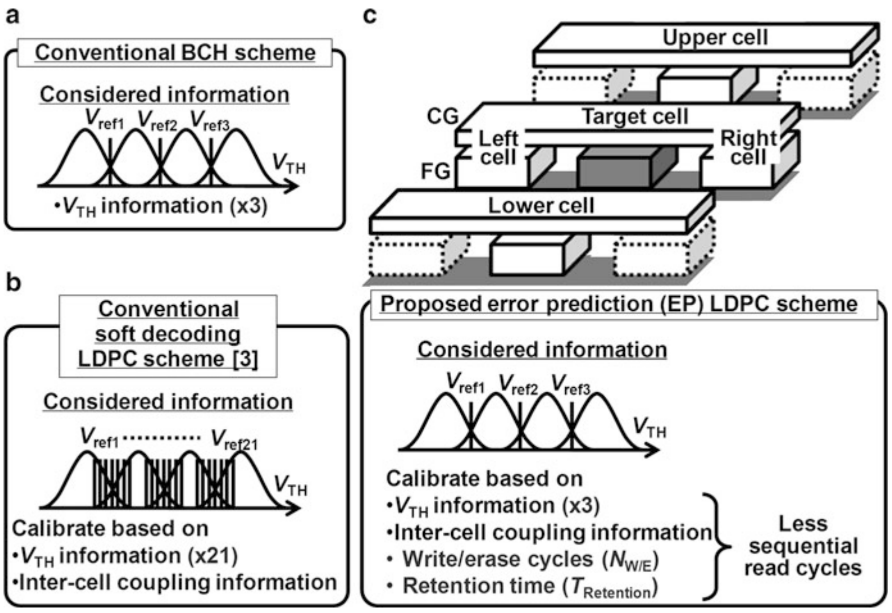

a
Conventional BCH scheme
Considered information
V_{ref1} V_{ref2} V_{ref3}
V_{TH}
•V_{TH} information (x3)
b
Conventional soft decoding LDPC scheme [3]
Considered information
V_{ref1} .... V_{ref21}
V_{TH}
Calibrate based on
•V_{TH} information (x21)
•Inter-cell coupling information
c
Upper cell
CG
FG
Target cell
Left cell
Right cell
Lower cell
Proposed error prediction (EP) LDPC scheme
Considered information
V_{ref1} V_{ref2} V_{ref3}
V_{TH}
Calibrate based on
•V_{TH} information (x3)
•Inter-cell coupling information
• Write/erase cycles ($N_{W/E}$)
• Retention time ($T_{Retention}$)
Less sequential read cycles

Fig. 13.1 Comparison of the conventional and the EP-LDPC ECC. EP-LDPC realizes the higher reliability with the minimum sequential read cycles. The best reliability is achieved by calibrating the memory cell information based on the V_{TH}, the inter-cell coupling, the write/erase cycles and the data retention time

Flash memories instead of the Bose-Chaudhuri-Hocquenghem (BCH) ECC [3]. In the 2 bit/cell, 3 reference voltages (Vref) are needed for the BCH (Fig. 13.1a). The conventional LDPC requires many, e.g. 21 [4], Vref to get accurate V_{TH} information (Fig. 13.1b). The inter-cell coupling is also considered to calibrate the interference [5].

However, the increase in Vref number requires more sequential read cycles. Assuming 50 μs cell read time and 21 Vref, the read access time is as much as 1050 μs. In case of the 3 bit/cell or 4 bit/cell, the read access time increases by twice or five-times, which is unacceptably long.

To realize both fast read and high reliability, the error prediction LDPC (EP-LDPC) utilizing only 3 Vref can be used (Fig. 13.1c).

The read is 7-times faster than the conventional LDPC. The EP-LDPC corrects errors most effectively because in addition to the V_{TH} and the inter-cell coupling, the write/erase cycles ($N_{W/E}$) and the retention time ($T_{Retention}$) are considered for the calibration. Figure 13.2 shows the hardware architecture of the SSD. The error prediction sequence is realized with the simple logic gates in the NAND controller. The additional NAND controller circuit area to the conventional LDPC is negligibly small.

The key technology to realize the EP-LDPC is to estimate $T_{Retention}$ by measuring errors and referring to the pre-recorded tables. During the program, the number of

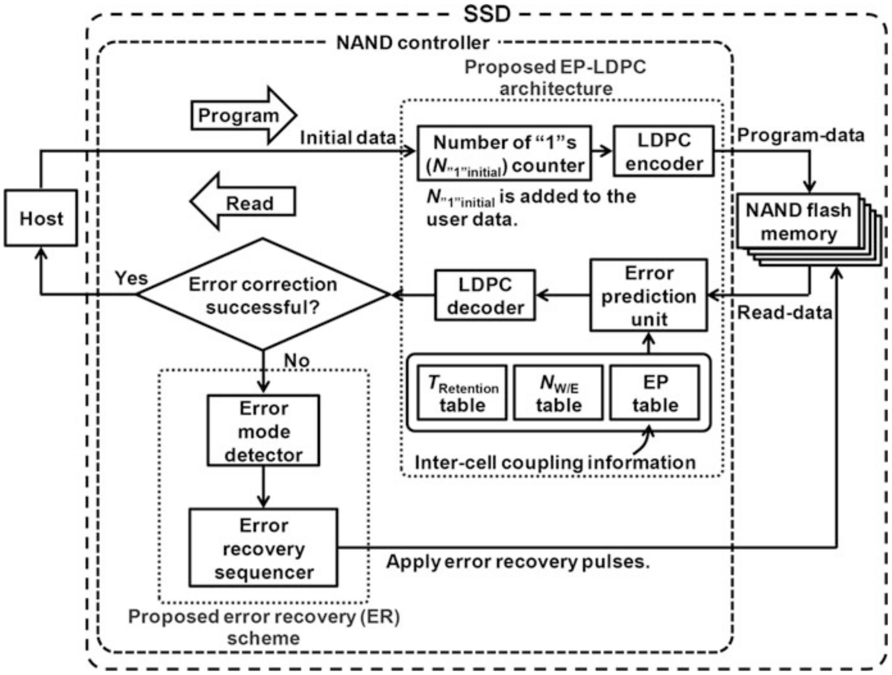

SSD
NAND controller
Proposed EP-LDPC architecture
Program
Initial data
Number of "1"s ($N_{\text{"1"initial}}$) counter
LDPC encoder
Program-data
Read
$N_{\text{"1"initial}}$ is added to the user data.
Host
NAND flash memory
Yes
Error correction successful?
LDPC decoder
Error prediction unit
Read-data
No
$T_{\text{Retention}}$ table
$N_{\text{W/E}}$ table
EP table
Error mode detector
Inter-cell coupling information
Error recovery sequencer
Apply error recovery pulses.
Proposed error recovery (ER) scheme

Fig. 13.2 Hardware architecture of SSD with EP-LDPC and error recovery scheme. Memory cell errors are corrected with the EP-LDPC. If the error correction fails, the proposed error recovery sequencer decreases the memory cell errors by applying the error recovery pulses

"1"s in the initial lower page write-data ($N_{\text{"1"initial}}$) is counted and added to the user data. During the read, the error prediction unit estimates the bit-error-rate (BER) by referring to the pre-recorded tables. These tables are stored in a Flash memory during the test process before shipment. The estimated BER is transferred to the LDPC decoder. If the error correction succeeds, the corrected data are output to the host. If the error correction fails, the error recovery (ER) scheme reduces errors. After the errors are recovered, the error correction is performed again.

Figure 13.3 shows the algorithm of the EP-LDPC.

Figure 13.4 shows the measurement results. The EP-LDPC during the sequential read consists of 5 steps. In Step1, the number of "1"s ($N_{\text{"1"}}$) in the lower page is counted ($N_{\text{"1"measured}}$). To secure the high reliability, triplicated $N_{\text{"1"}}$ data are stored with BCH ECC because the EP-LDPC is not performed for $N_{\text{"1"}}$. In Step2, the BER of the lower page (BER_{Lower}) is estimated from $|N_{\text{"1"measured}} - N_{\text{"1"initial}}|/N_P$. N_P denotes the page size. In lower pages, the data retention error increases $N_{\text{"1"}}$ because the electron ejection from FG decreases the V_{TH} [6]. On the other hand, in the program disturb error, the electron injection to FG increases the V_{TH} and $N_{\text{"1"}}$ decreases [7]. For the worst-case maximum BER which requires the powerful ECC, two scenarios are considered where the program disturb error or the data retention

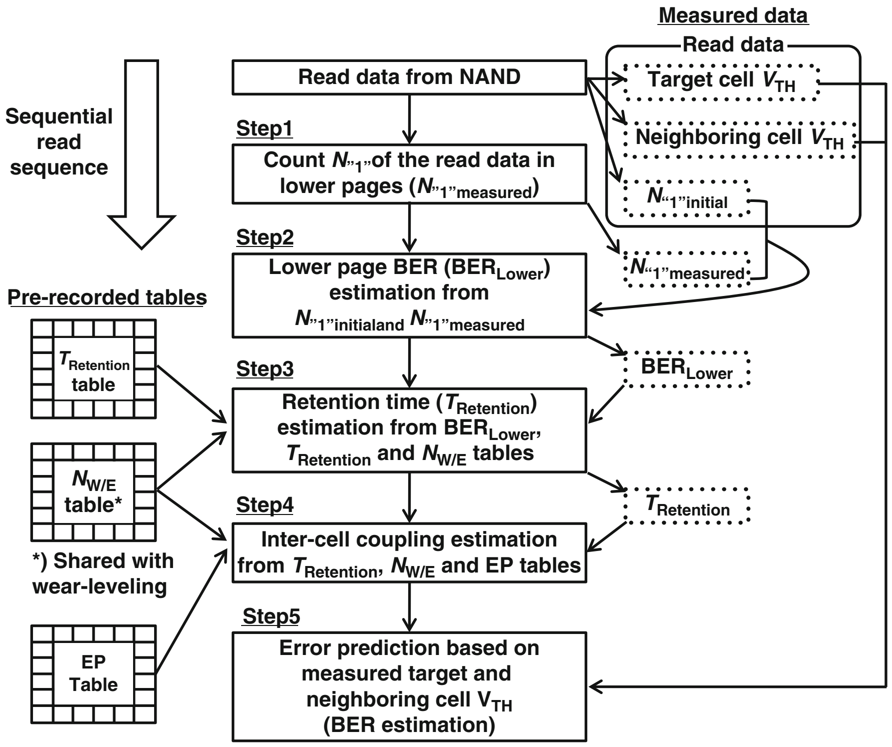

Measured data
Read data
Target cell V_{TH}
Neighboring cell V_{TH}
$N_{“1”initial}$
$N_{“1”measured}$
BER_{Lower}
$T_{Retention}$
Sequential read sequence
Read data from NAND
Step1
Count $N_{“1”}$ of the read data in lower pages ($N_{“1”measured}$)
Step2
Lower page BER (BER_{Lower}) estimation from $N_{“1”initial}$ and $N_{“1”measured}$
Step3
Retention time ($T_{Retention}$) estimation from BER_{Lower}, $T_{Retention}$ and $N_{W/E}$ tables
Step4
Inter-cell coupling estimation from $T_{Retention}$, $N_{W/E}$ and EP tables
Step5
Error prediction based on measured target and neighboring cell V_{TH} (BER estimation)
Pre-recorded tables
$T_{Retention}$ table
$N_{W/E}$ table*
*) Shared with wear-leveling
EP Table

Fig. 13.3 Calculation algorithm of the EP-LDPC. BER is estimated with a simple logic by referring to the pre-recorded $T_{Retention}$ table and EP table instead of increasing the number of V_{ref}. The memory capacity to store the EP table with 34 bit accuracy for each BER is 267 kByte, which is negligibly small compared with the overall SSD capacity, 512 GByte

error dominates. When $|N_{“1”measured} - N_{“1”initial}|$ is close to 0, the shift of the V_{TH} due to the program disturb and the data retention are compensated and the BER becomes smaller, which does not require a strong ECC.

In Step3, the data retention time $T_{Retention}$ is estimated by BER_{Lower}, $N_{W/E}$, and the $T_{Retention}$ table. The $T_{Retention}$ table represents the relation among BER_{Lower}, $N_{W/E}$, and the data retention time. In Step4, the inter-cell coupling is estimated from $T_{Retention}$ and the pre-recorded EP table which stores the dependence of the target cell BER on the neighboring cell data. In Step5, the BER of each data is estimated from the inter-cell coupling with the target and neighboring cell V_{TH} data read from the Flash memory. After that, the LDPC decoding is applied. The corrected data is finally output to the host if the decoding is successful. The error recovery sequence is applied if the decoding fails. In the real-world failure, various effects happen at once. The capacitive interference increasing the V_{TH} and the data retention failure decreasing the V_{TH} occur simultaneously. Moreover, the V_{TH} shift due to the data retention strongly depends on $N_{W/E}$ and $T_{Retention}$. The proposed scheme is most

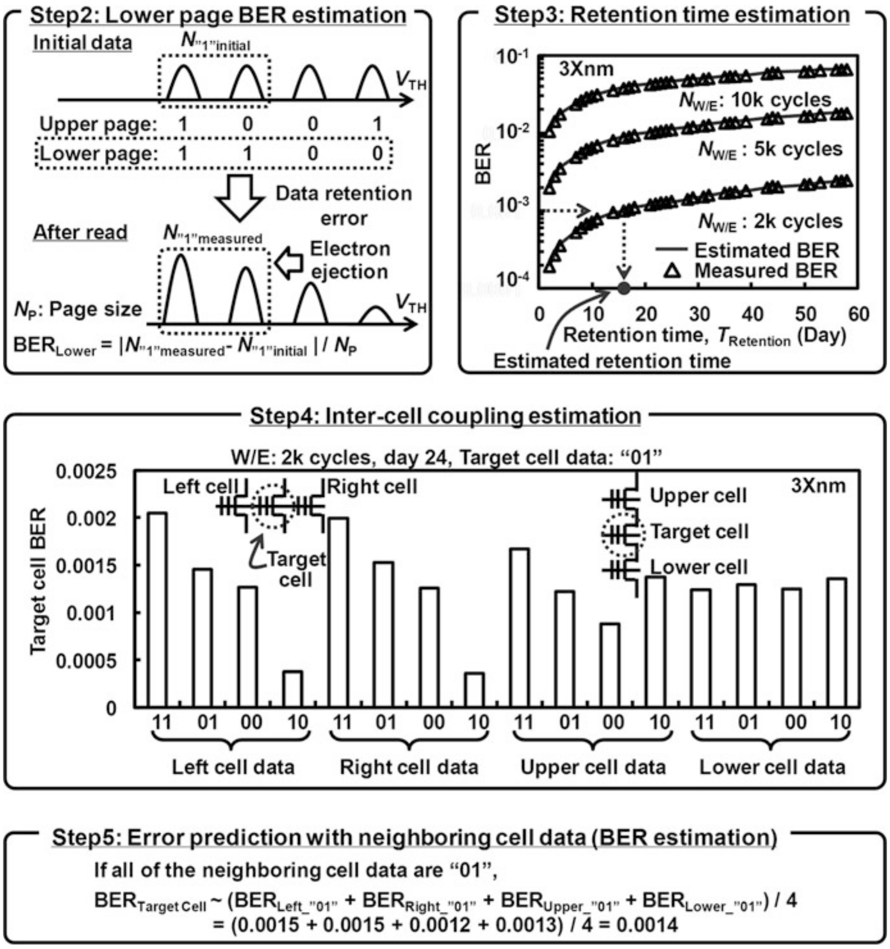


Fig. 13.4 Measurement results of the EP-LDPC

efficient because these various effects are considered by estimating $T_{Retention}$ and using different EP table values for each $N_{W/E}$ and $T_{Retention}$.

Figure 13.5 shows the measured reliability improvement. n-bit codeword consists of k-bit user data and (n-k)-bit parity. The code rate (CR) is defined as k/n. The lower code rate corrects more errors. Figure 13.5a shows the SSD lifetime at 85°C. The measured lifetime is extended by 11-times and 2.8-times when the CR is 2/3 and 9/10, respectively. The acceptable BER increases by 3.7-times and 1.8-times (Fig. 13.5b), and the acceptable $N_{W/E}$ improves by 1.6-times and 1.5-times (Fig. 13.5c) when the CR is 2/3 and 9/10, respectively. These measured results demonstrate that the EP-LDPC is more effective for the smaller CR and thus is more useful for the future reliability degraded Flash memories.

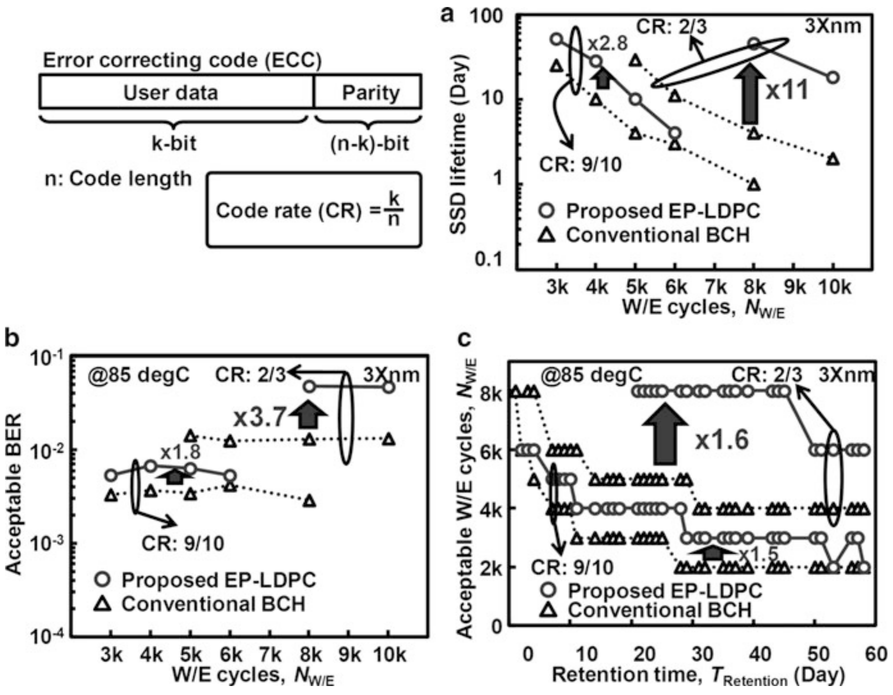


Fig. 13.5 Measured reliability improvement with the proposed EP-LDPC. With the proposed architecture, over 10-times longer SSD lifetime, 3.7-times higher acceptable BER, and 1.6-times higher acceptable W/E cycles are realized

13.2 Error Recovery Scheme [1]

Figure 13.6 describes the error recovery scheme. First, the error mode is determined to the program disturb or the data retention error. The error mode depends on the location of memory cell (word-line number), $N_{W/E}$ and $T_{Retention}$. If the program disturb error is dominant ($N_{\text{“1”measured}} < N_{\text{“1”initial}}$), the program disturb error recovery pulse (PDRP) is applied to the memory cells. In the case of $N_{\text{“1”measured}} > N_{\text{“1”initial}}$, the data retention error is dominant and the data retention error recovery pulse (DRRP) is applied to the memory cells. The error recovery mechanism is illustrated in Fig. 13.6a. Electrons at the interface between control gate (CG) and inter-poly dielectric (IPD) are de-trapped with PDRP. PDRP decreases the V_{TH} and improves the program disturb error. With DRRP, electrons are injected into the floating gate. DRRP increases the V_{TH} and mitigates the data retention error.

Figure 13.6b shows the measured program disturb BER vs. the time after PDRP. The BER with PDRP converges to the BER without PDRP in 200 ms. This

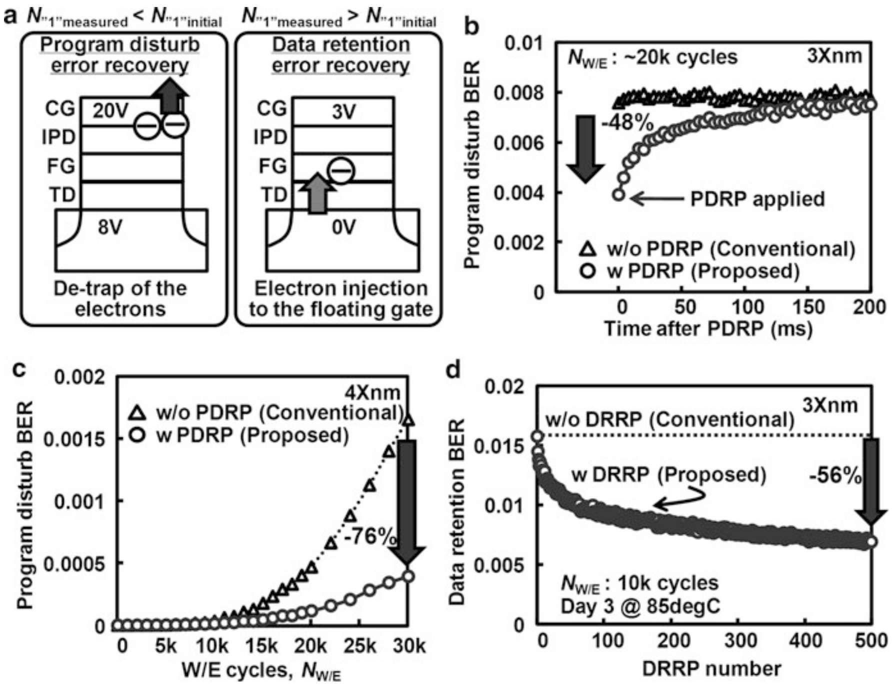


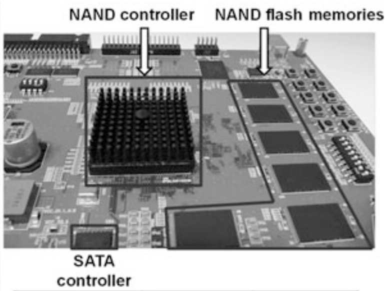
Fig. 13.6 Error recovery (*ER*) scheme and the measured results. The error reduction pulse is applied to the memory cells only when the error correction fails. If N“1”measured is smaller or larger than N“1”initial, the memory cell errors are dominated by the program disturb or the data retention errors, respectively. The program disturb error is reduced by 76% with the program disturb error recovery pulse (*PDRP*). The data retention error de-creases by 56% with the data retention error recovery pulse (*DRRP*)

convergence implies that because the electrons are easily re-trapped, the de-trapping is from the interface between CG and IPD and not from the inside of IPD as reported [4].

The ECC is performed for data just after PDRP is applied. The PDRP realizes 76% recovery of the program disturb error (Fig. 13.6c). The data retention BER is recovered by 56% with 500 times DRRPs (Fig. 13.6d).

Figure 13.7 depicts the photograph of the measured SSD system. The gate count of the LDPC ECC in the NAND controller is similar to the conventional LDPC [8] and is twice as large as the BCH ECC. By using sophisticated analog-digital mixed logic, the circuit size of the LDPC decoder can be reduced by one-tenth [9]. The SSD lifetime is extended by over 10-times compared with the conventional BCH. The read is 7-times faster than the conventional LDPC. The error recovery scheme realizes 76% program disturb error recovery and 56% data retention error recovery.

	Conventional BCH	Conventional LDPC (Soft decoding)	Proposed EP-LDPC
Considered information	V_{TH}	V_{TH} Inter-cell coupling [4]	V_{TH} Inter-cell coupling W/E cycles Retention time
Read reference voltage number	3	21 [3]	3
Sequential read cycles	x1	x7	x1
Gate count of ECC circuits in the controller	x1	x2 [7]	x2
SSD lifetime @85 degC (W/E cycles: 8k)	4 days	-	45 days (>x10)
Acceptable BER	1.3%	-	4.8% (x3.7)
Acceptable W/E cycles	5k	-	8k (x1.6)



	Conventional SSD	Proposed SSD
Program disturb error recovery	None	PDRP (-76%)
Data retention error recovery	None	DRRP (-56%)

Fig. 13.7 Photograph of the measured SSD system. The EP-LDPC scheme realizes the high reliability with the minimum sequential read cycles. The error recovery scheme reduces the memory cell error by 56–76%

13.3 3D TSV-Integrated Hybrid ReRAM/MLC NAND SSD [10]

There is a growing demand for a high performance, highly reliable and low power SSD. A 3D TSV-integrated SSD with hybrid memory configuration which uses storage class memories (SCMs) and NAND Flash memories is a promising solution. Among various SCMs, ReRAM is the best candidate due to its high speed, low power operation and potentially high scalability [11, 12]. This section describes detailed specifications for the ReRAM and architecture for the hybrid SSD. Requirements for the ReRAM in the 3D TSV-integrated hybrid ReRAM/MLC NAND SSD are clarified. Suitable SSD data management algorithms are also described and the SSD performance, energy and endurance are evaluated.

ReRAM specifications are discussed. First, I/F (interface) specifications are discussed. The block diagram of the SSD is shown in Fig. 13.8a.

The ReRAM uses NAND-like I/F. Figure 13.8b shows the physical image of the SSD with TSVs. The reasons for adopting NAND I/F to the ReRAM are shown in Fig. 13.9d). The set/reset characteristics of 50 nm ReRAM cells are measured. The set pulse is 50 ns wide and 2 V high (Fig. 13.9a).

Figure 13.9b is the measured program/erase (P/E) cycles of a ReRAM cell without write verification. The cell fails at less than 100 program/erase (P/E) cycles. By using the write verification (Fig. 13.9c), more than 10^6 P/E cycles are achieved. Figure 13.9d shows the verification cycles performed in Fig. 13.9c. In fact, multiple verifications rarely occur but can be more than 10 times in the worst case. This means that the strict worst latency definition used in a DRAM I/F is not suitable for ReRAM. Moreover, the limited P/E cycles in ReRAM require logical-physical address translation and wear leveling tables in the SSD controller. The overhead for

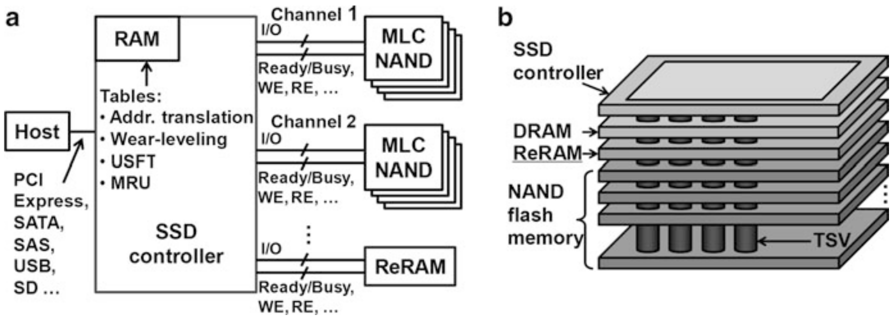


Fig. 13.8 (**a**) Block diagram of the 3D TSV–integrated hybrid ReRAM/MLC NAND SSD. ReRAM uses NAND-like I/F. (**b**) Physical image of the SSD

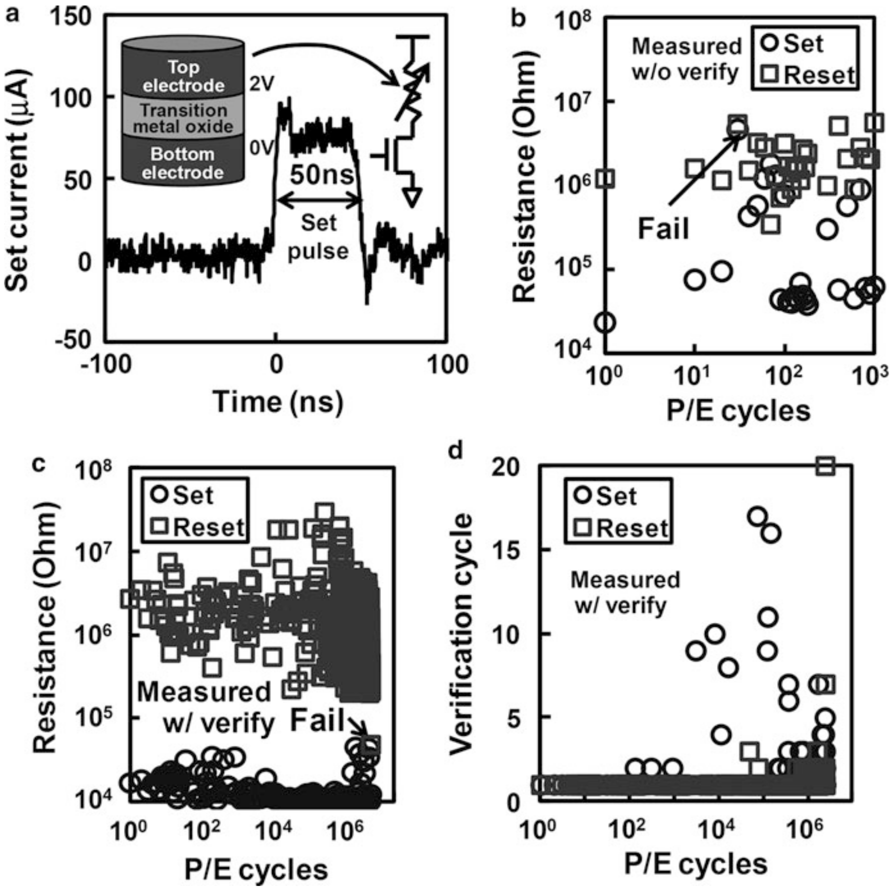


Fig. 13.9 (**a**) Measured set pulse of the 50 nm ReRAM. (**b**) Measured ReRAM P/E cycles without verify. (**c**) Measured ReRAM P/E cycles with verification. (**d**) Verification cycle performed in (**c**). Reset pulse is -2 V, 20 ns

Table 13.1 SLC/MLC NAND and ReRAM specifications

	SLC/MLC NAND	ReRAM
Read latency (Max.)	85us/page	<3 μs/sector
Write latency (Typical)	Lower page 400 μs (SLC) Upper page 2,800 μs	(Set/Reset) <3 μs/sector
Erase latency (Typical)	8,500 μs/block	Unnecessary
I/O: Toggle/ONFi DDR	400 MHz	1,066 MHz
V_{DD} (Core/I/O)	3.3 V/1.8 V	1.8 V/1.2 V
Access unit	Page (16 KiB)	Sector (512 B)
Partial write/overwrite	Impossible (erase required)	Possible
Required endurance	3×10^3	10^5

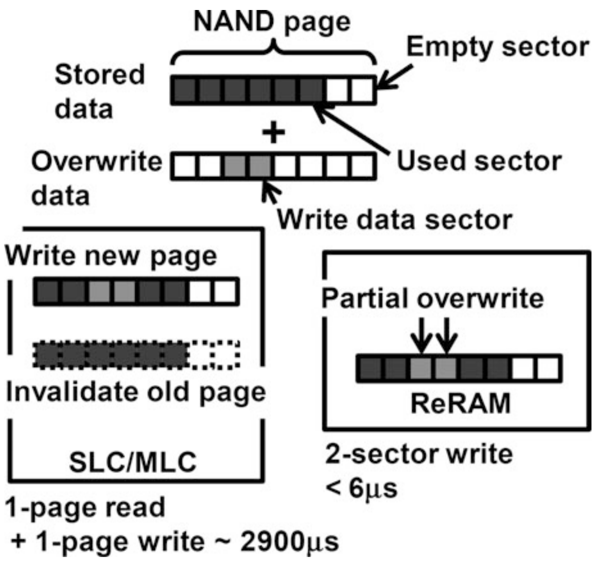


Fig. 13.10 Partial write or overwrite policy of the SLC/MLC NAND and ReRAM

referring these tables also makes ReRAM difficult to use byte-accessible DRAM I/F. On the other hand, polling (Ready/Busy status), which is used in NAND I/F, allows a variable access time. Therefore, NAND-like I/F is suitable for the ReRAM because the write latency increases with verification.

Secondly, the overwrite policy is discussed. One of the critical issues in SSDs is the data fragmentation due to random writes [13]. Table 13.1 summarizes the single-level-cell (SLC) NAND, MLC NAND and ReRAM specifications. Since NAND Flash memory writes in a page unit (16 KB) and erases in a block unit (4 MB), overwrite is not allowed. Therefore, a random overwrite with size less than a page (partial overwrite) requires 1-page read from the old page and 1-page write to a new page as shown in Fig. 13.10. Then the old page is invalidated. Thus, frequent random overwrites create many invalid pages. As a result, serious performance, power and reliability degradation are induced due to the increase of block copy which takes more than 100 ms [14].

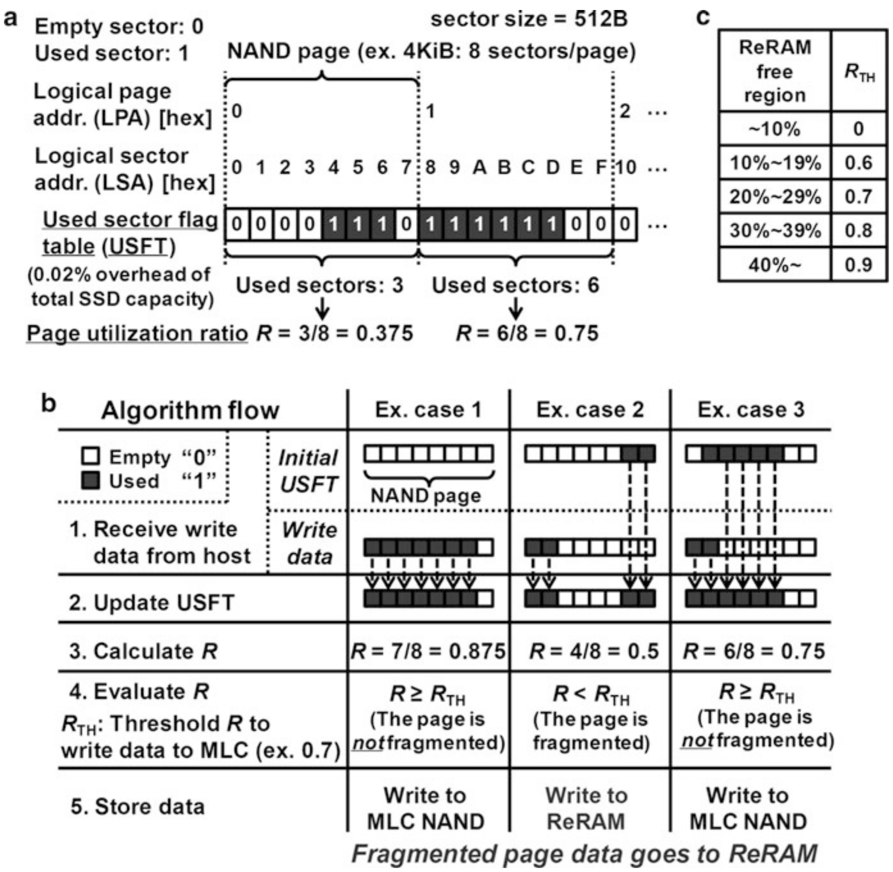

a
Empty sector: 0
Used sector: 1
sector size = 512B
NAND page (ex. 4KiB: 8 sectors/page)
Logical page addr. (LPA) [hex]
0
1
2 …
Logical sector addr. (LSA) [hex]
0 1 2 3 4 5 6 7 8 9 A B C D E F 10 …
Used sector flag table (USFT)
(0.02% overhead of total SSD capacity)
0 0 0 0 1 1 1 0 1 1 1 1 1 1 0 0 0 …
Used sectors: 3
Used sectors: 6
Page utilization ratio R = 3/8 = 0.375
R = 6/8 = 0.75
c
ReRAM free region
R_TH
~10% 0
10%~19% 0.6
20%~29% 0.7
30%~39% 0.8
40%~ 0.9
b
Algorithm flow
Ex. case 1
Ex. case 2
Ex. case 3
Empty "0"
Used "1"
Initial USFT
NAND page
1. Receive write data from host
Write data
2. Update USFT
3. Calculate R
R = 7/8 = 0.875
R = 4/8 = 0.5
R = 6/8 = 0.75
4. Evaluate R
R_TH: Threshold R to write data to MLC (ex. 0.7)
R ≥ R_TH (The page is not fragmented)
R < R_TH (The page is fragmented)
R ≥ R_TH (The page is not fragmented)
5. Store data
Write to MLC NAND
Write to ReRAM
Write to MLC NAND
Fragmented page data goes to ReRAM

Fig. 13.11 Anti-fragmentation (*AF*) algorithm. (**a**) Introduction of used sector flag table (*USFT*) and page utilization ratio R. (**b**) AF algorithm. (**c**) Dynamic R_{TH} control. R_{TH} decreases when free ReRAM region decreases to evict more data to MLC NAND

On the other hand, the ReRAM access unit is a sector (512B) and partial overwrite is possible. The fragmentation problems can be solved by the suitable hybrid SSD architecture and data management algorithms, which are presented in the next section.

13.3.1 Data Management Algorithms [10]

Three data management algorithms are described for the 3D hybrid SSD. The key idea is to store hot fragmented data less than the page size to ReRAM and use MLC NAND for sequential data. Two key concepts are introduced as shown in Fig. 13.11a. The used sector flag table (USFT) located in the SSD controller stores information of the used logical sector address (LSA). A flag bit is prepared for each LSA. If the LSA is used, its used sector flag is switched to 1, otherwise 0. By using USFT, page utilization ratio R is calculated. R is a ratio of the number of used LSAs to the number of total sectors in a page. For instance, the page size is 4 KB. A page has three used LSAs in the logical page address (LPA) 0 in Fig. 13.11a. Then, $R = 3/(4\ \text{KB}/512\ \text{B}) = 3/8 = 0.375$. Here, the sector size is 512 B. The overhead of USFT is only 0.02% of the total SSD capacity.

First, the anti-fragmentation (AF) algorithm is described in Fig. 13.11b. When write request is received from a host, USFT of the target LPA is updated. The used sector flags of the newly used LSAs are turned to 1. Then R is calculated and evaluated. If the R is larger than the threshold value R_{TH}, the data are written to MLC NAND as a non-fragmented page. Otherwise, it is written to ReRAM as a fragmented page. Therefore, only fragmented pages are written in ReRAM.

The MLC NAND data fragmentation due to small data write accesses is avoided. Afterwards, as data accumulates in the ReRAM, page data in ReRAM become not fragmented. When R becomes higher than R_{TH}, the data is evicted from ReRAM to MLC NAND. To avoid ReRAM overflow as well as make full use of the limited ReRAM capacity, dynamic R_{TH} control is used (Fig. 13.11c). When ReRAM free region decreases, R_{TH} also decreases to enhance data eviction.

Second, to increase the performance boost of AF, *Most-Recently-Used* (MRU) algorithm is proposed (Fig. 13.12). LPAs of the write request from the host are stored in a MRU table in a FIFO order. The algorithm first evaluates R as AF.

However, if $R \geq R_{TH}$, MRU table is searched instead of storing the write data to MLC NAND. If the write LPA is found in the MRU table, the data go to ReRAM. If not, the data is stored in MLC NAND.

The SSD performance is significantly enhanced because hot page data are always kept in ReRAM even when the data are not fragmented.

Finally, reconsider-as-a-fragmentation (RAAF) algorithm is proposed. This algorithm suppresses overwrites to the MLC NAND after the data eviction from ReRAM to MLC NAND. As shown in Fig. 13.13, once data are evicted from ReRAM to MLC NAND, R is permanently greater than R_{TH}. The data stay in MLC NAND.

However, a page in MLC NAND can become hot even after the data eviction. As a result, small data overwrites could frequently occur to the page of MLC NAND. This induces data fragmentation again in MLC NAND. To solve this problem, RAAF is proposed to store back the data from MLC NAND to ReRAM. In this scheme, the target LPA USFT is reset to 0 when the data are written to the MLC NAND. Small size overwrite to the MLC NAND page is recognized as fragmented data again. Thus, AF works again to store hot small data in ReRAM. Note that actual page data written back to the ReRAM must maintain data consistency. The target page data are read out from MLC NAND to the SSD controller and merged with the new overwrite data. Then, the newly merged data are written to ReRAM.

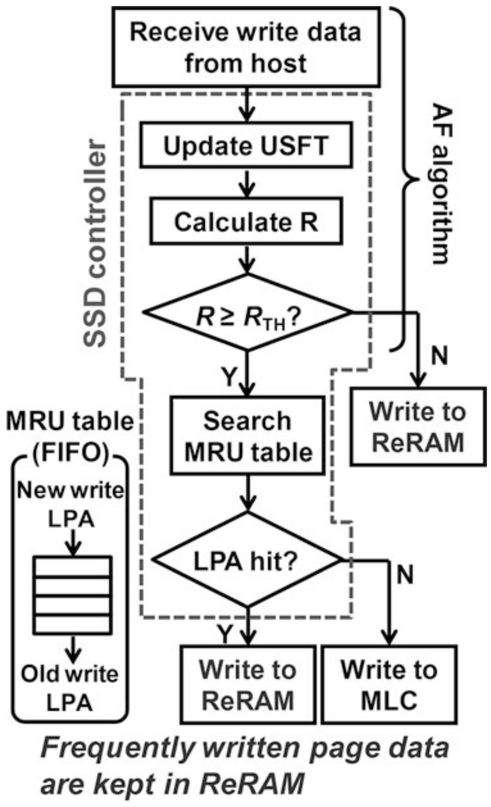


Fig. 13.12 Most-Recently-Used (*MRU*) algorithm. Recently accessed LPAs are stored in the MRU table. The frequently written hot data permanently stays in ReRAM

13.3.2 Performance, Power and Reliability [10]

The 3D hybrid NAND SSD is evaluated. A TLM (*Transaction Level Modeling*)-based SSD emulator that can comprehensively simulate performance, energy consumption and P/E cycles has been developed. The profile of the write data input obtained from a financial server [15] is shown in Fig. 13.14.

For comparison, conventional MLC NAND and hybrid SLC/MLC NAND SSDs are also evaluated. SLC NAND is used instead of ReRAM.

The results for the write performance, write energy and average P/E cycles are shown in Fig. 13.15. Compared with the conventional MLC NAND SSD, the SSD with AF, MRU and RAAF algorithms shows 11 times higher performance and 79% lower write energy (Fig. 13.15a, b). By using 3D TSV interconnects, the I/O energy is reduced by 27 times because the huge capacitance of the wire bonding is almost eliminated. As a result, the total SSD energy reduction reaches 93%.

The speed and power overhead of the USFT reference are negligibly small. Furthermore, the slope of the average MLC NAND P/E cycles is decreased by 6.9 times in Fig. 13.15c by the SSD. This directly corresponds to a reduction in the replacement cost of a SSD storage system because the slope determines the aging speed of the SSD.

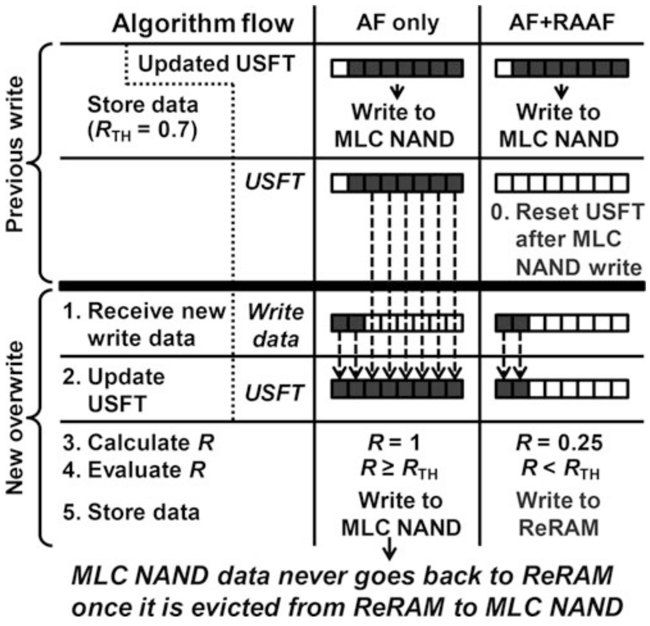
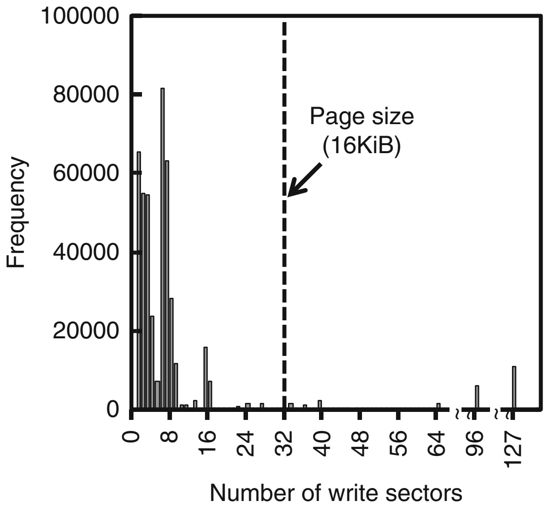
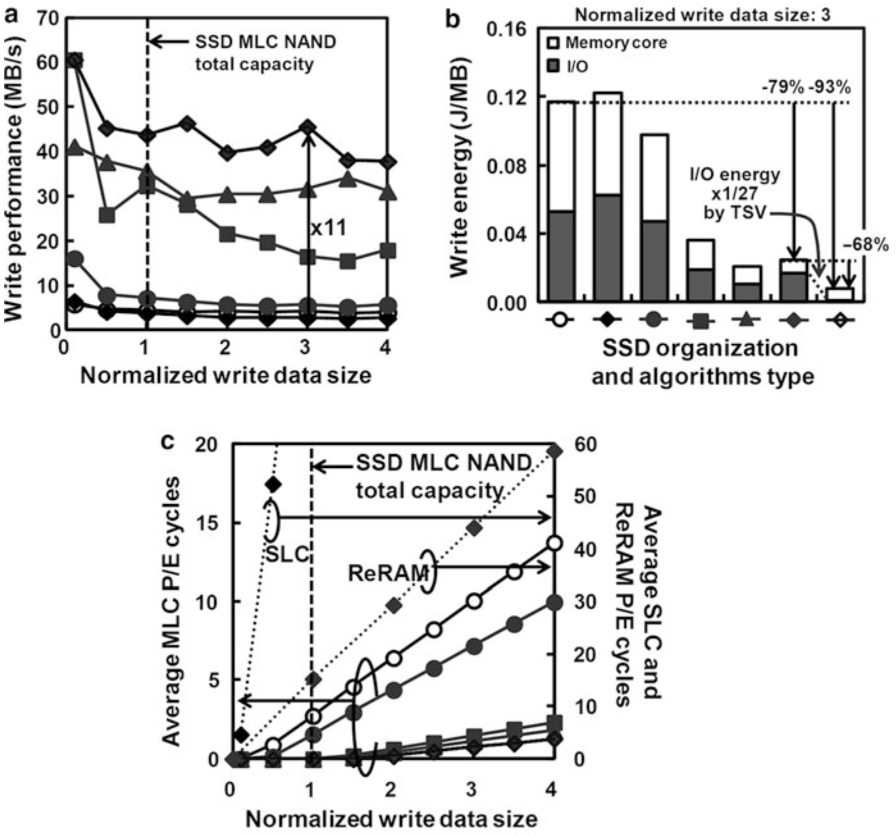


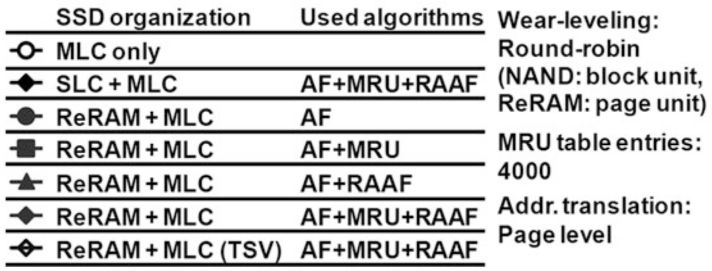
Fig. 13.13 Reconsider-as-a-fragmentation (RAAF) algorithm. The data evicted from ReRAM to MLC NAND is stored back to ReRAM

Fig. 13.14 Distribution of the write data size used in the SSD evaluation [15]



Although the MLC NAND P/E cycles of the SLC/MLC NAND SSD also decreases by the algorithms, the slope of the P/E cycles of the SLC NAND become 250 times of that of the MLC NAND, which is unacceptably high. This is because serious data fragmentation is induced in the SLC NAND. The frequent block copy





SSD organization	Used algorithms
-O- MLC only	
-◆- SLC + MLC	AF+MRU+RAAF
-●- ReRAM + MLC	AF
-■- ReRAM + MLC	AF+MRU
-▲- ReRAM + MLC	AF+RAAF
-◆- ReRAM + MLC	AF+MRU+RAAF
-◇- ReRAM + MLC (TSV)	AF+MRU+RAAF

Wear-leveling: Round-robin (NAND: block unit, ReRAM: page unit)
MRU table entries: 4000
Addr. translation: Page level

Fig. 13.15 (**a**) Write performance, (**b**) write energy and (**c**) average P/E cycles of the evaluated SSDs. The *horizontal axis* for (**a**) and (**c**) is the data size written to the SSD normalized by the SSD MLC NAND total capacity. P/E cycles for SLC NAND and ReRAM use AF + MRU + RAAF. 100 ns/sector is assumed for the ReRAM write and read latency

of SLC NAND degrades the performance and the energy to the level of MLC only SSD. In ReRAM, such a data fragmentation does not occur because the partial overwrite is possible. As a result, the slope of the ReRAM P/E cycles is limited to

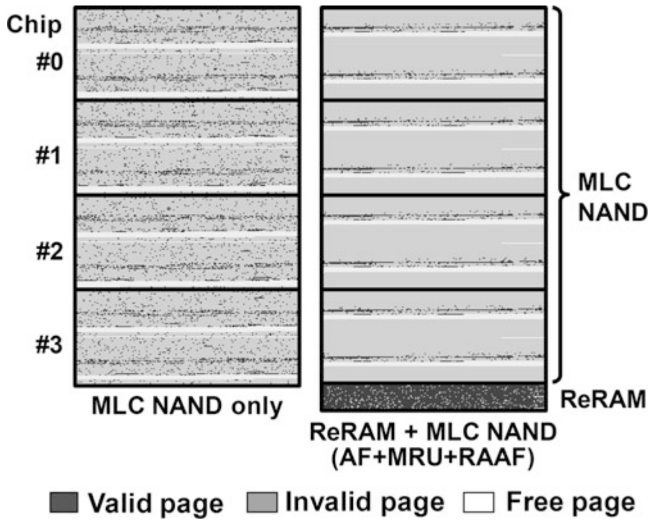


Fig. 13.16 Comparison of the SSD valid page location

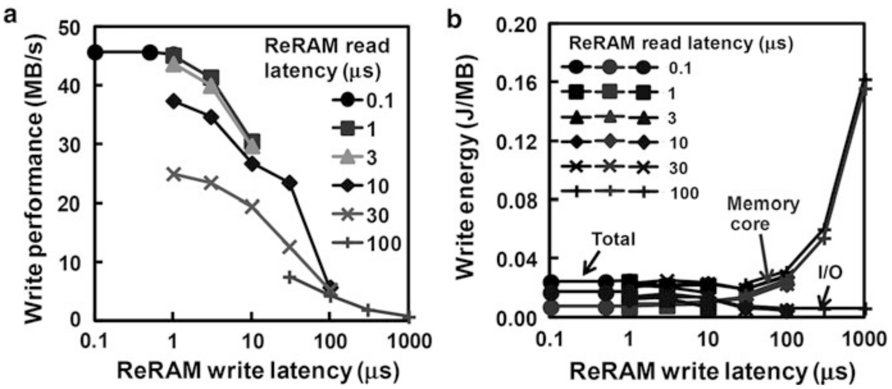


Fig. 13.17 (**a**) Write performance and (**b**) write energy of the proposed SSD with various ReRAM write and read latency. AF + MRU + RAAF is used

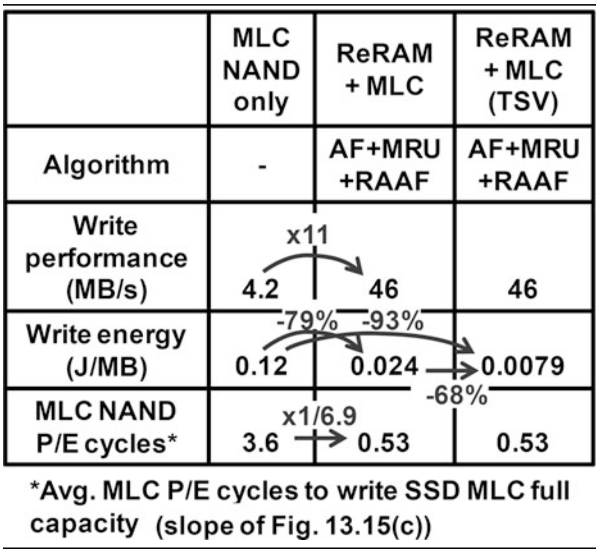
28 times of that of the MLC NAND in the SSD. Assuming MLC NAND endurance of 3×10^3, the required P/E cycles for ReRAM is less than 10^5, which is acceptable for the ReRAM device characteristics.

Figure 13.16 shows the valid page map of the conventional and proposed SSD. The valid pages are scattered in the conventional SSD indicating that frequent overwrites have occurred to the MLC NAND.

On the other hand, the hybrid SSD efficiently uses ReRAM and shows less fragmentation of MLC NAND because overwrites to MLC NAND are suppressed.

The required ReRAM latency to obtain sufficient improvements by the proposed algorithm is also investigated. Figure 13.17 show the proposed SSD write

Table 13.2 Summary of the 3D TSV-integrated hybrid ReRAM/MLC NAND SSD



	MLC NAND only	ReRAM + MLC	ReRAM + MLC (TSV)
Algorithm	-	AF+MRU +RAAF	AF+MRU +RAAF
Write performance (MB/s)	4.2	46	46
Write energy (J/MB)	0.12	0.024	0.0079
MLC NAND P/E cycles*	3.6	0.53	0.53

*Avg. MLC P/E cycles to write SSD MLC full capacity (slope of Fig. 13.15(c))

x11
-79%
-93%
-68%
x1/6.9

performance and energy as a function of the ReRAM write latency. ReRAM read latency is also varied. From the figures, both ReRAM write and read latency should be less than 3 us to maintain high performance and low power operation. Considering 50 ns write pulse, the 3 us access is achievable for ReRAM in write verify operation.

Table 13.2 summarizes the hybrid SSD. The proposed 3D hybrid SSD shows 11 times higher performance, 93% lower write energy and 6.9 times higher endurance.

13.4 Conclusions

In this chapter, the signal processing technologies which realizes the high speed operation, low power consumption and high reliability of the SCM and NAND Flash integrated hybrid SSDs are described. First, an error prediction (EP) low density parity check (LDPC) error correcting code (ECC) is described which realizes an over 10-times extended lifetime without access time penalty. Errors are efficiently predicted by the write/erase cycles, data retention time and the neighboring cell data. Second, an error recovery (ER) scheme is discussed, which decreases the program disturb error and the data retention error by 74 and 56%, respectively. When the ECC cannot correct errors, the ER scheme can temporarily recover the failures.

Finally, a 3D TSV-integrated hybrid ReRAM/MLC NAND SSD architecture has been proposed. Data fragmentation of MLC NAND is suppressed and efficient MLC NAND usage is realized by storing small hot data to ReRAM using the proposed

SSD data management algorithms. The proposed 3D TSV hybrid SSD realizes 11 times performance increase, 6.9 times endurance enhancement and 93% write energy reduction from the conventional MLC NAND SSD. Moreover, 68% energy reduction is achieved by the 3D-TSV interconnects. ReRAM specifications are also proposed. NAND-like ReRAM I/F, sector-access overwrite policy are necessary. Both ReRAM write and read latency should be less than 3 us and required endurance for ReRAM is 10^5.

References

1. S. Tanakamaru et al., Over 10-times extended lifetime, 76% reduced error Solid-State Drives (SSDs) with error prediction LDPC architecture and error recovery scheme, in *IEEE International Solid-State Circuits Conference (ISSCC)*, 2012, San Francisco, CA, USA, pp. 424–425
2. J.-D. Lee et al., Effects of floating-gate interference on NAND flash memory cell operation. IEEE Electron Device Lett. **23**(5), 264–266 (2002)
3. R. Motwani et al., Low Density Parity Check (LDPC) Codes and the NEED for Stronger ECC, in Flash *Memory Summit* (2011)
4. C. Kim et al., A 21 nm high performance 64 Gb MLC NAND flash memory with 400 MB/s asynchronous toggle DDR interface, in *Symposium on VLSI Circuits Dig. Tech. Papers*, 2011, Kyoto, Japan, pp. 196–197
5. G. Dong et al., On the use of soft-decision error-correction codes in NAND flash memory. IEEE Trans. Circuits Syst. I **58**(2), 429–439 (2011)
6. A. Serov et al., Statistical retention modeling in floating-gate cell: ONO scaling, in *IRPS*, 2009, Montreal, QC, Canada, pp. 887–890
7. S. Tanakamaru et al., 95%-lower-BER 43%-lower-power intelligent Solid-State Drive (SSD) with asymmetric coding and stripe pattern elimination algorithm, in *ISSCC Dig. Tech. Papers*, 2011, pp. 204–205
8. J. Yang, High-efficiency SSD for reliable data storage systems, in *Flash Memory Summit* (2011)
9. B. Vigoda, LDPC error correction using probability processing circuits, in *Flash Memory Summit* (2010)
10. H. Fujii et al., x11 performance increase, x6.9 endurance enhancement, 93% energy reduction of 3D TSV-integrated hybrid ReRAM/MLC NAND SSDs by data fragmentation suppression, in *IEEE Symposium on VLSI Circuits*, Hawaii, 2012
11. Y.S. Chen et al., Challenges and opportunities for HfOX based resistive random access memory, *IEDM*, pp. 717–720 (2011)
12. K. Higuchi et al., 50 nm HfO_2 ReRAM with 50-times endurance enhancement by set/reset turnback pulse & verify scheme, in *SSDM*, 2011, pp. 1011–1012
13. T. Hatanaka et al., A 60% higher write speed, 4.2 Gbps, 24-channel 3D-Solid State Drive (SSD) with NAND flash channel number detector and intelligent program-voltage booster, in *IEEE Symposium on VLSI Circuits*, University of Tokyo, Tokyo, Japan, June 2010, pp. 233–234
14. K. Takeuchi et al., A 56 nm CMOS 99 mm^2 8 Gb Multi–level NAND flash memory with 10 Mbyte/s program throughput. IEEE J. Solid-State Circuits **42**(1), 219–232 (2007)
15. http://traces.cs.umass.edu/index.php/Storage/Storage

Index