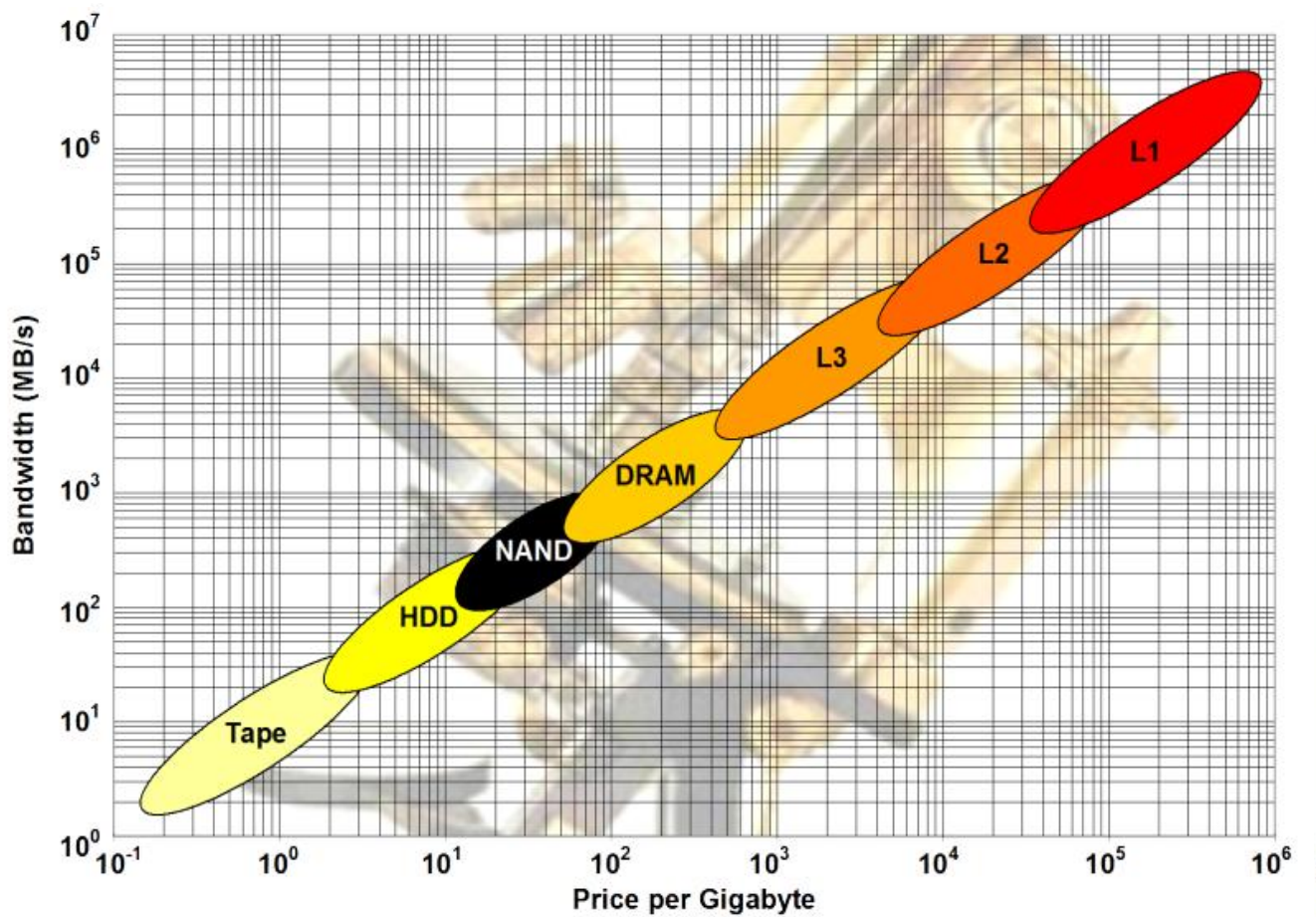


Flashing in the Memory Hierarchy

An Overview on Flash Memory Internals

Jalil Boukhobza, Stéphane Rubini
Lab-STICC, Université de Bretagne Occidentale

NAND Flash in the hierarchy

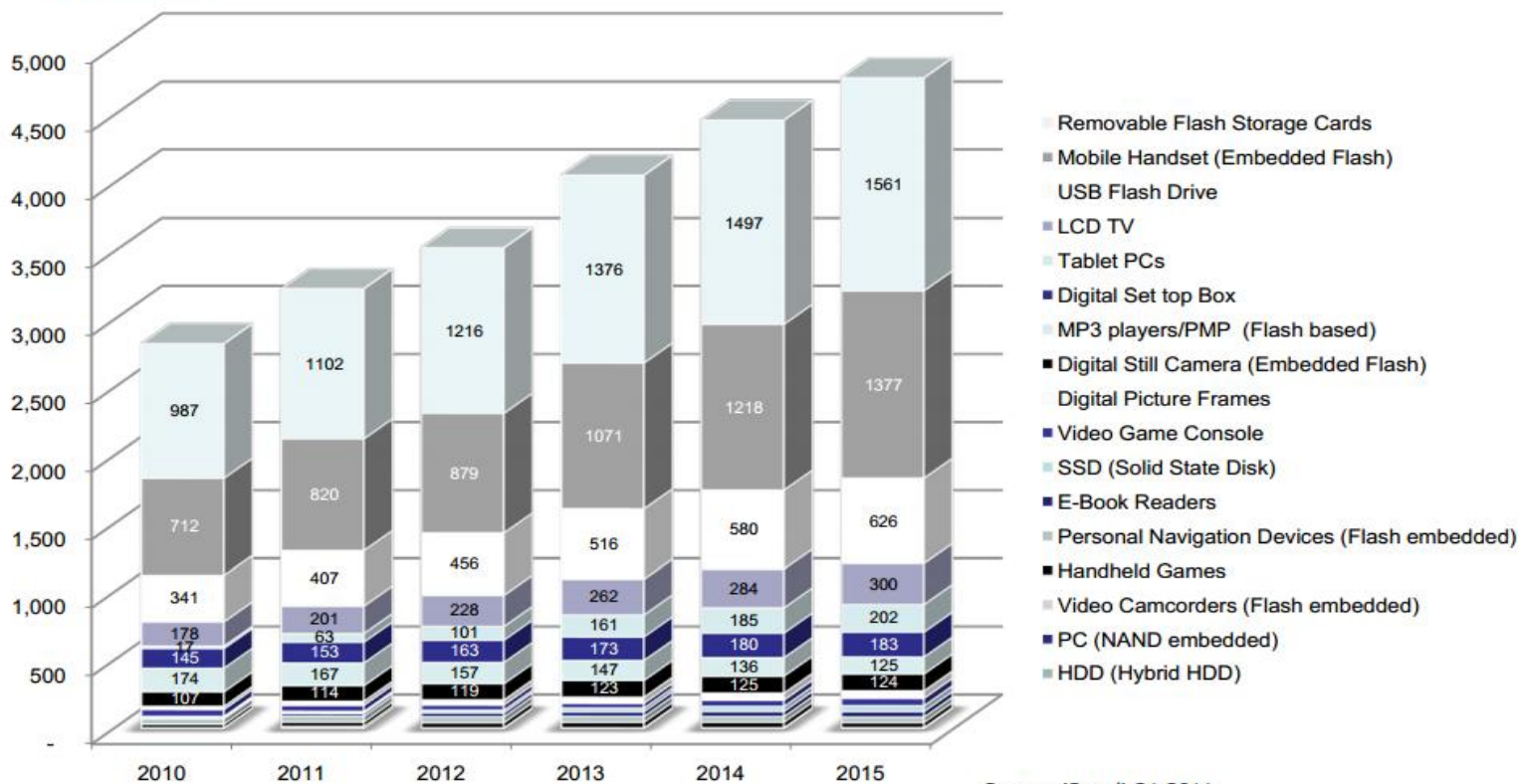


From Objective Analysis: *Are Hybrid Drives Finally Coming of Age?*

Where is the NAND flash memory ?



Number of Units (Millions) by Application



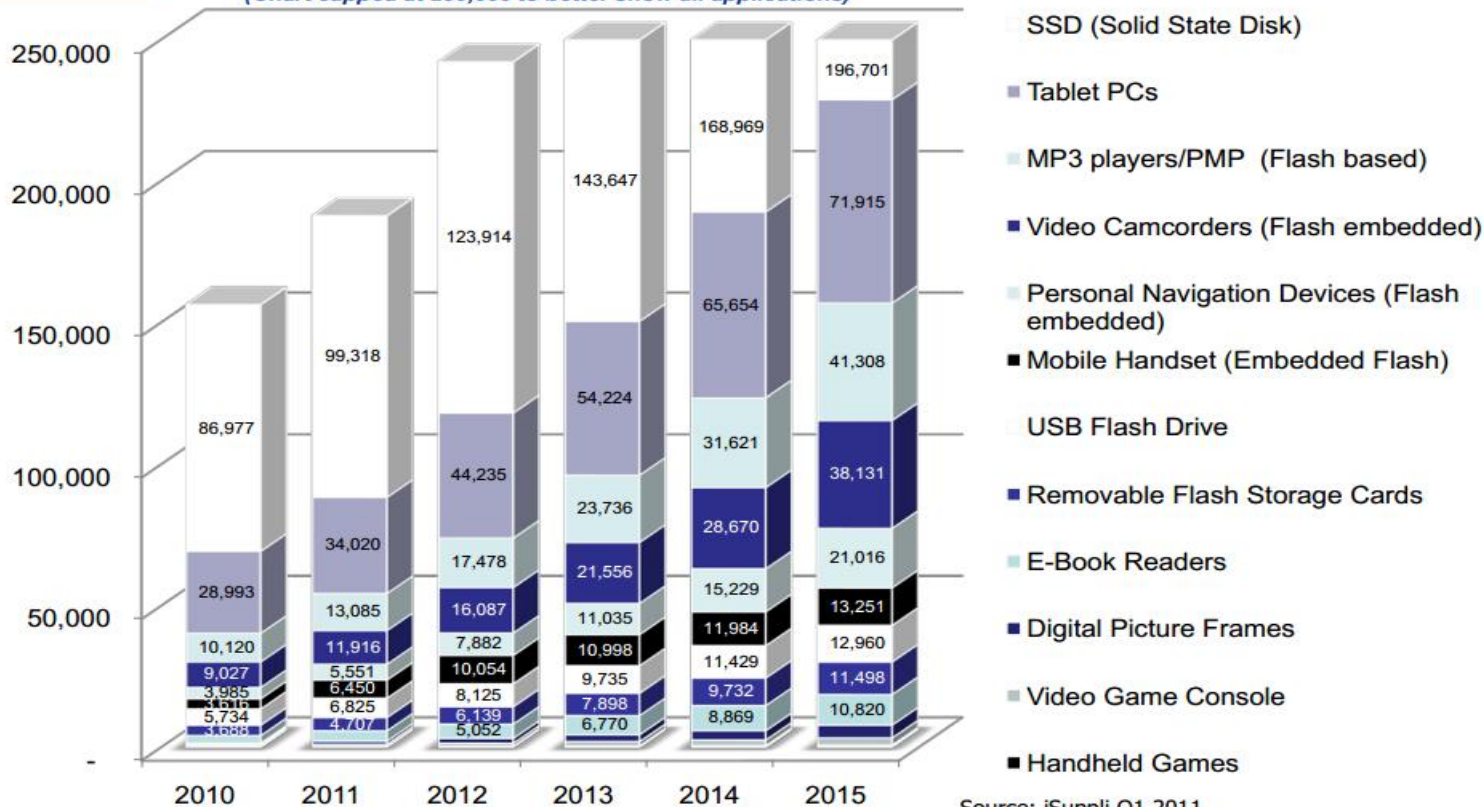
Flash Memory Summit 2011
Santa Clara, CA

Source: iSuppli Q1 2011



Megabytes (MB) per Unit by Application

(Chart capped at 250,000 to better show all applications)

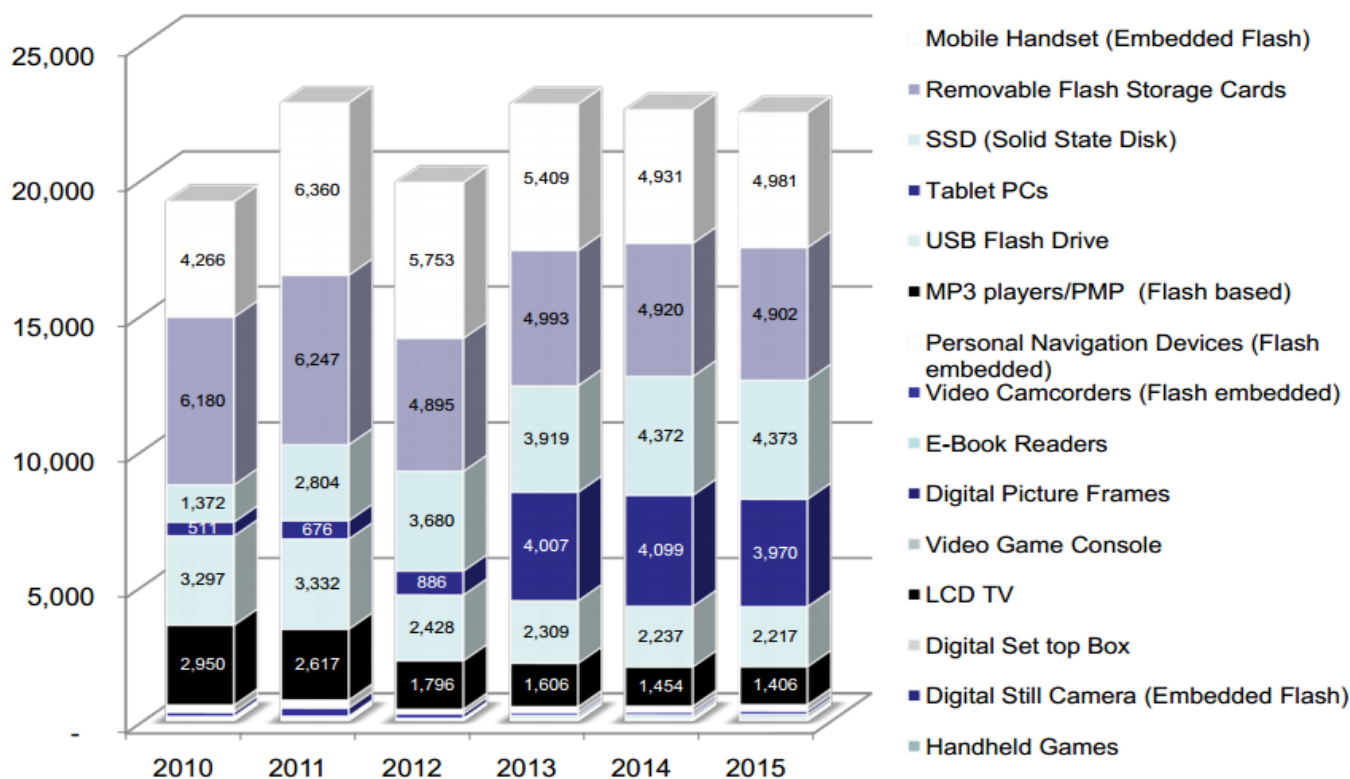


Flash Memory Summit 2011
Santa Clara, CA

Source: iSuppli Q1 2011



Millions of \$ by Application



Source: iSuppli Q1 2011

Flash Memory Summit 2011
Santa Clara, CA

Presentation outline

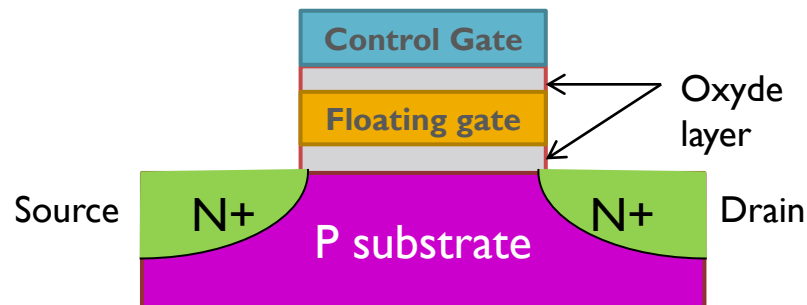
1. Flash memory basics
2. Flash memory characteristics
3. Flash memory support
 1. Mapping schemes
 2. Wear leveling
 3. Garbage collection
 4. Flash specific cache systems
 5. Some contributions
4. Performance & energy considerations
5. Flash memory interfacing
6. Conclusions & perspectives

Presentation outline

1. Flash memory basics
2. Flash memory characteristics
3. Flash memory support
 1. Mapping schemes
 2. Wear leveling
 3. Garbage collection
 4. Flash specific cache systems
 5. Some contributions
4. Performance & energy considerations
5. Flash memory interfacing
6. Conclusions & perspectives

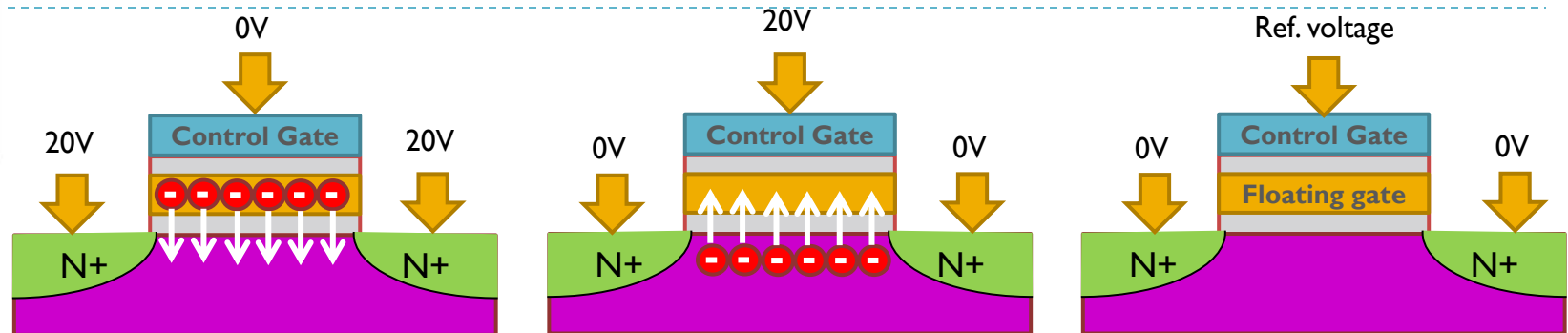
Flash memory cells

- ▶ Invented by F. Masuoka Toshiba 1980
- ▶ Introduced by Intel in 1988
- ▶ Type of EEPROM (Electrically **Erasable** & **Programmable Read** Only Memory)
- ▶ Use of Floating gate transistors
- ▶ Electrons pushed in the floating gate are trapped



- ▶ 3 operations: program (write), erase, and read

Flash memory operations



Erase operation

- ▶ FN (Fowler-Nordheim) tunneling: Apply high voltage to substrate (compared to the operating voltage of the chip - usually between 7–20V)
- ▶ → electrons off the floating gate
- ▶ Logic « 1 » in SLC

Program / write operation

- ▶ Apply high voltage to the control gate
- ▶ → electrons get trapped into the floating gate
- ▶ Logic « 0 »

Read operation

- ▶ Apply reference voltage to the control gate:
 - ▶ If floating gate charged: no current flow
 - ▶ If not charged; current flow

NOR Vs NAND

► NOR

- Byte random access
- Low density
- Higher cost (/bit)
- Fast read
- Slow write
- Slow erase
- Code storage (XIP – eXecute In Place)

► NAND

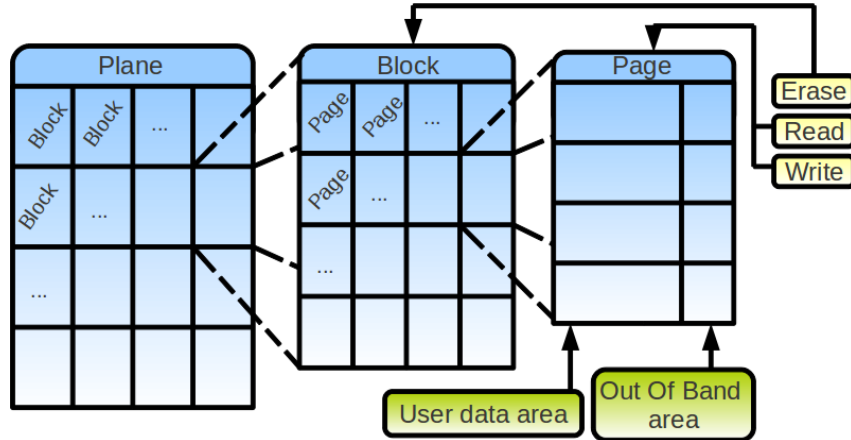
- Page access
- High density
- Slower read
- Faster write
- Faster erase (block granularity)
- Data storage

► Other types: DiNOR, AND, ...

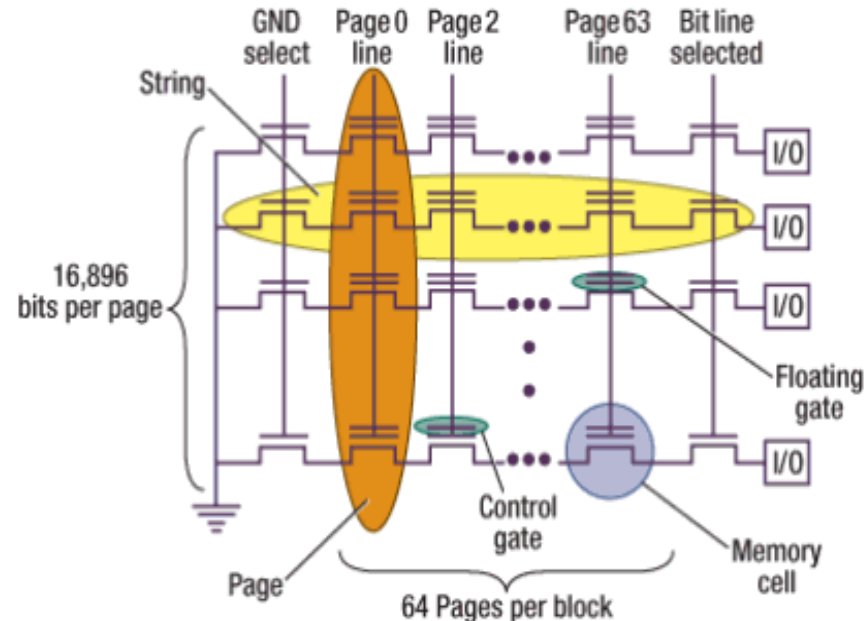
	NOR	NAND
Cell Array		
Cell Size	$10F^2$	$4F^2$

Source EETimes: <http://www.eetimes.com/design/memory-design/4009410/Flash-memory-101-An-Introduction-to-NAND-flash>

NAND flash memory architecture

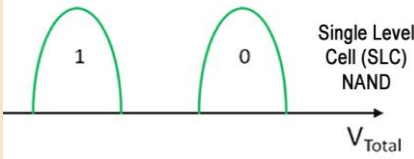
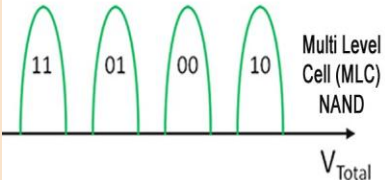
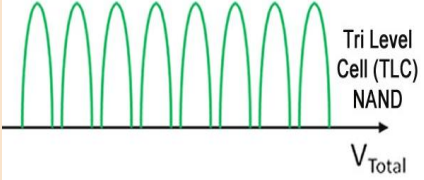


- ▶ Read/Write → page
- ▶ Erasures → blocks
- ▶ Page: 2-8KB
- ▶ Block: 128-1024 KB

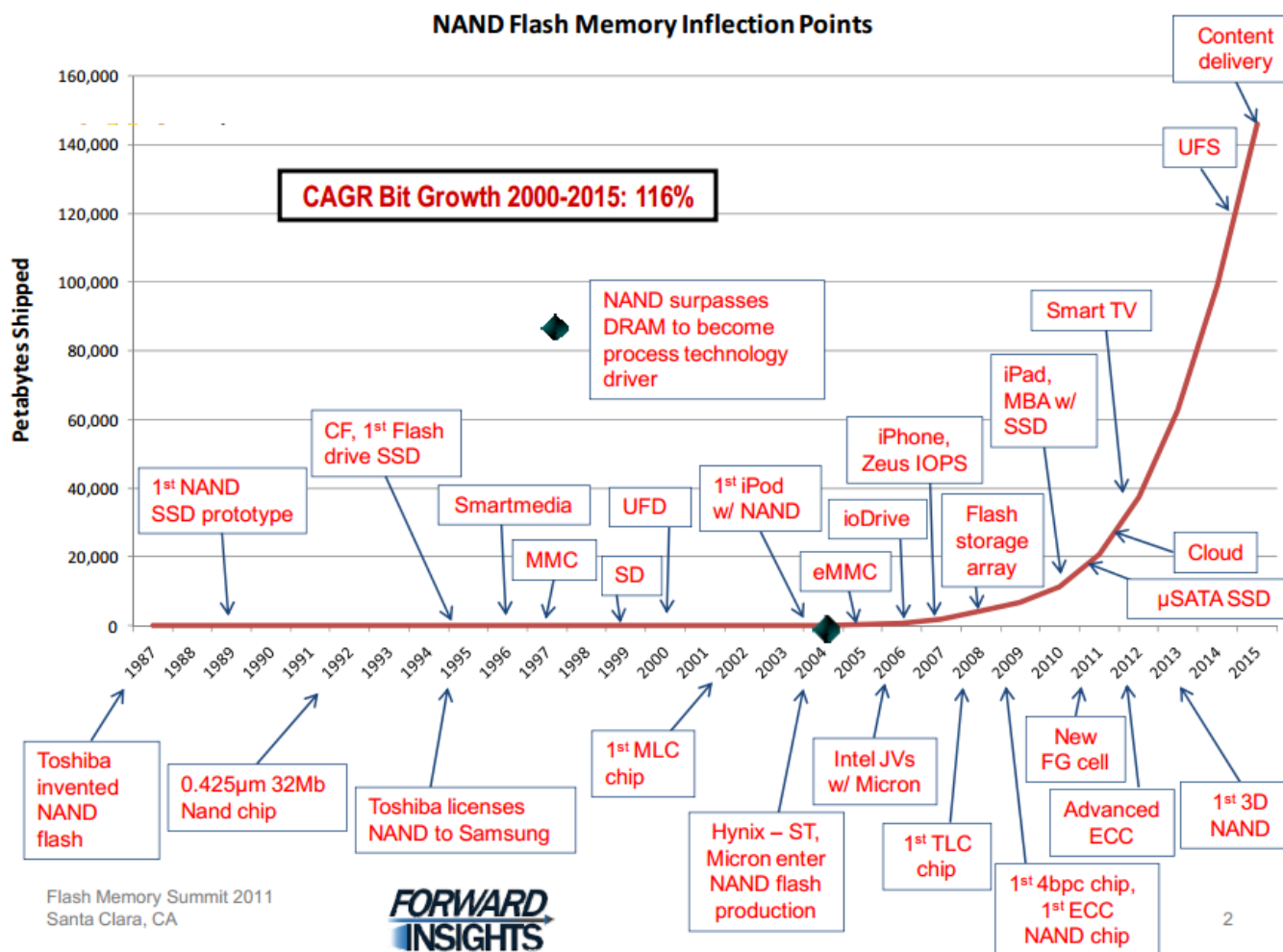


Source: <http://www.electroiq.com/articles/sst/2011/05/solid-state-drives.html>

Different densities: SLC, MLC, TLC

	SLC (Single Level Cell)	MLC (Multi Level Cell)	TLC (Tri Level Cell)
	 <p>Single Level Cell (SLC) NAND</p>	 <p>Multi Level Cell (MLC) NAND</p>	 <p>Tri Level Cell (TLC) NAND</p>
Storage	1 bit / cell	2 bits / cell	3 bits /cell
Performance	+++	++	+
Density	+	++	+++
Lifetime (P/E cycles)	~ 100 000	~ 10 000	~5 000
ECC complexity	+	++	+++
Applications	Embedded and industrial applications (high end SSDs...)	Most consumer applications (e.g. memory cards)	Low-end consumer applications not needing data updates (e.g. mobile GPS)

Compound Annual Growth Rate (CAGR)

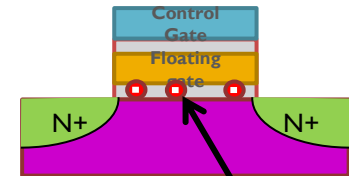
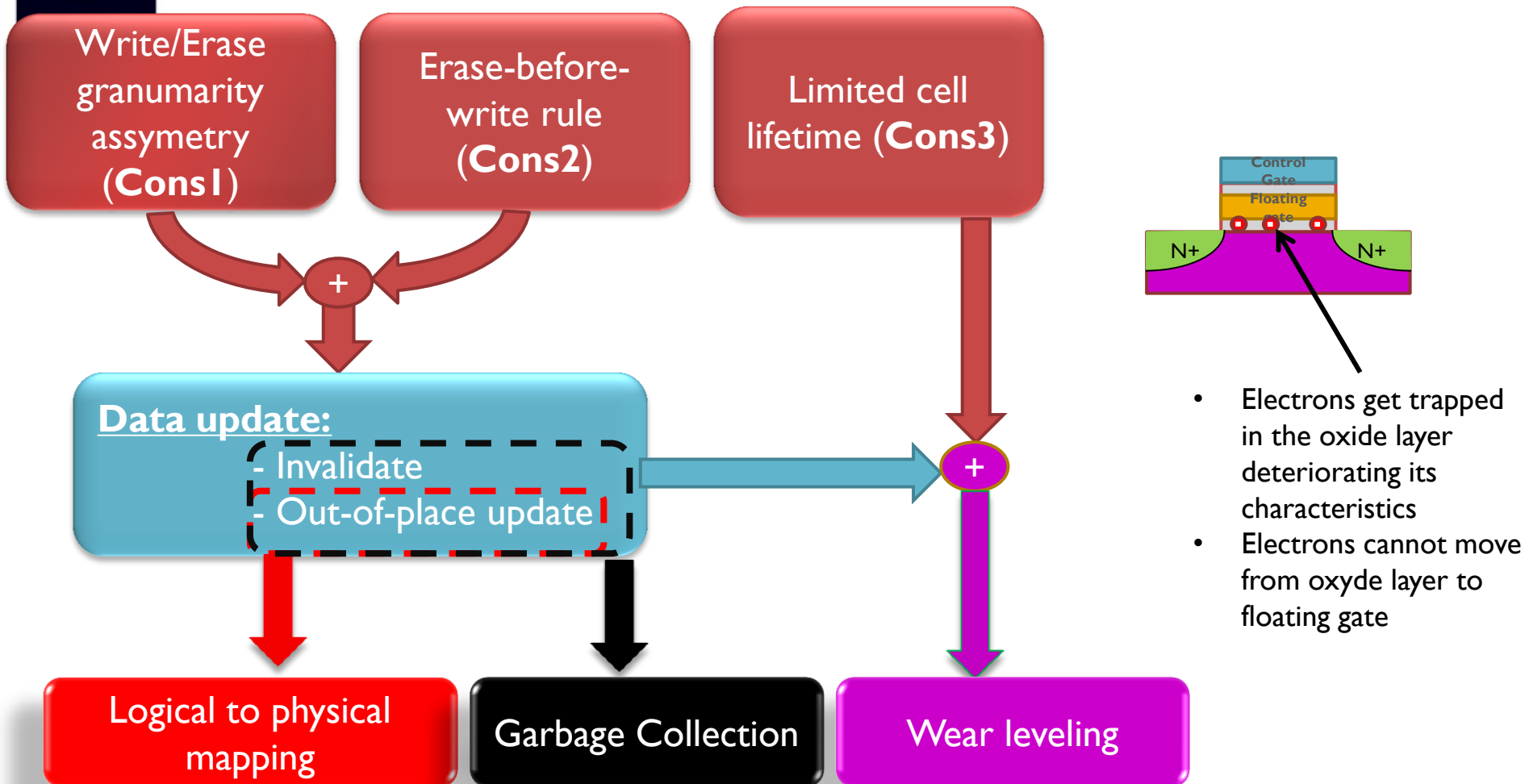


Source: G. Wong, « Inflection points », Flash Summit 2011

Presentation outline

1. Flash memory basics
2. **Flash memory characteristics**
3. Flash memory support
 1. Mapping schemes
 2. Wear leveling
 3. Garbage collection
 4. Flash specific cache systems
 5. Some contributions
4. Performance & energy considerations
5. Flash memory interfacing
6. Conclusions & perspectives

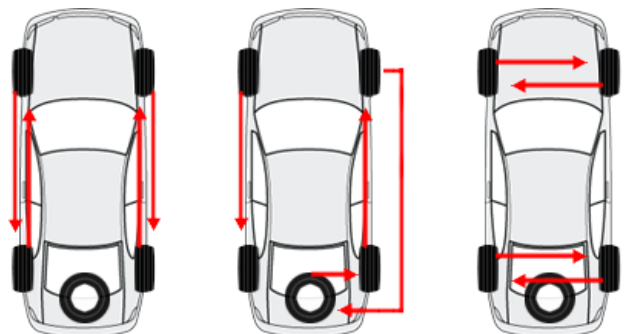
Flash memory constraints



- Electrons get trapped in the oxide layer deteriorating its characteristics
- Electrons cannot move from oxide layer to floating gate

Wear leveling

- ▶ You already do that with your tyres ...



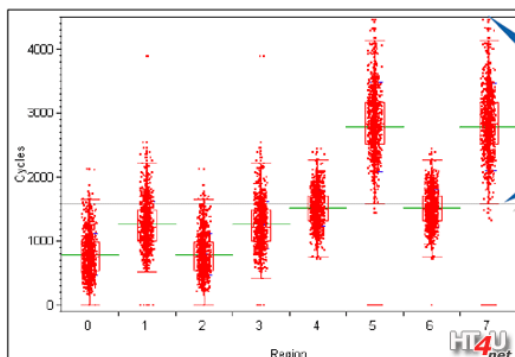
4 tyre rotation
all are same size

5 tyre rotation
all are same size

Tyre rotation when size
is different front & rear

Pierelli
Courtesy

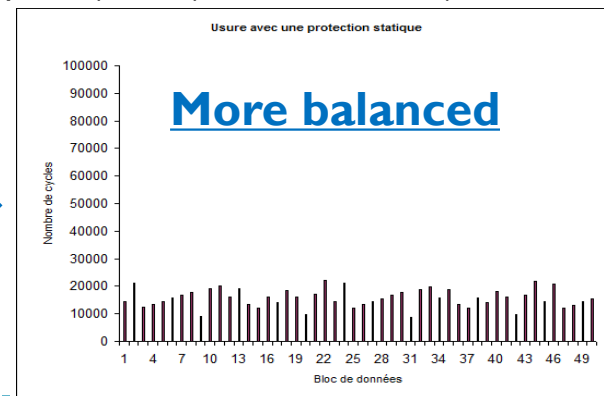
- ▶ Keeping a balanced erasures' distribution over flash memory blocks.



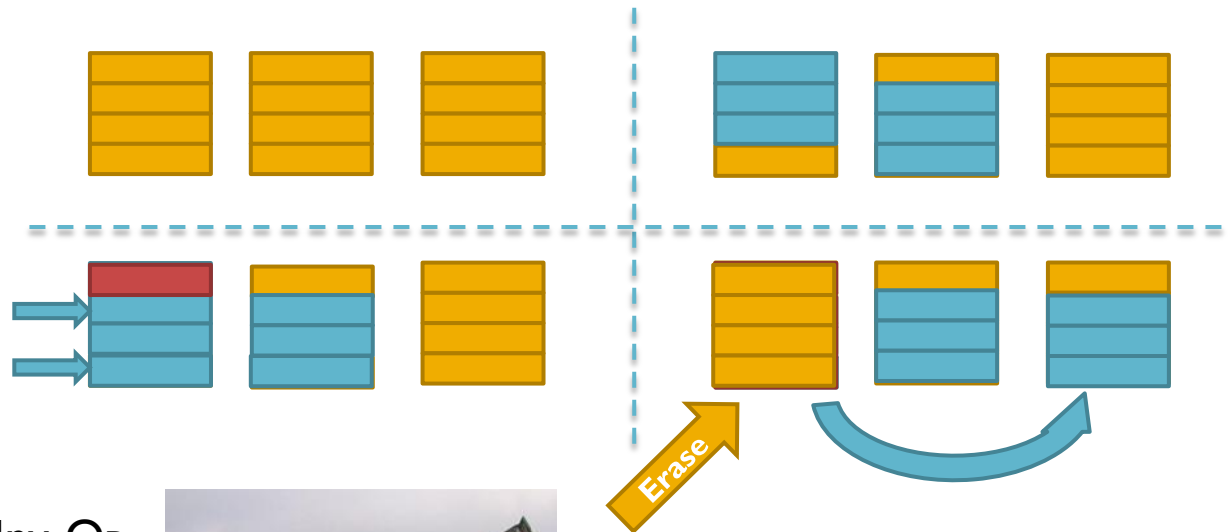
Block with max
cycles much
higher than
average block



<http://www.presence-pc.com/tests/ssd-flash-disques-22675/5/>



Garbage Collection



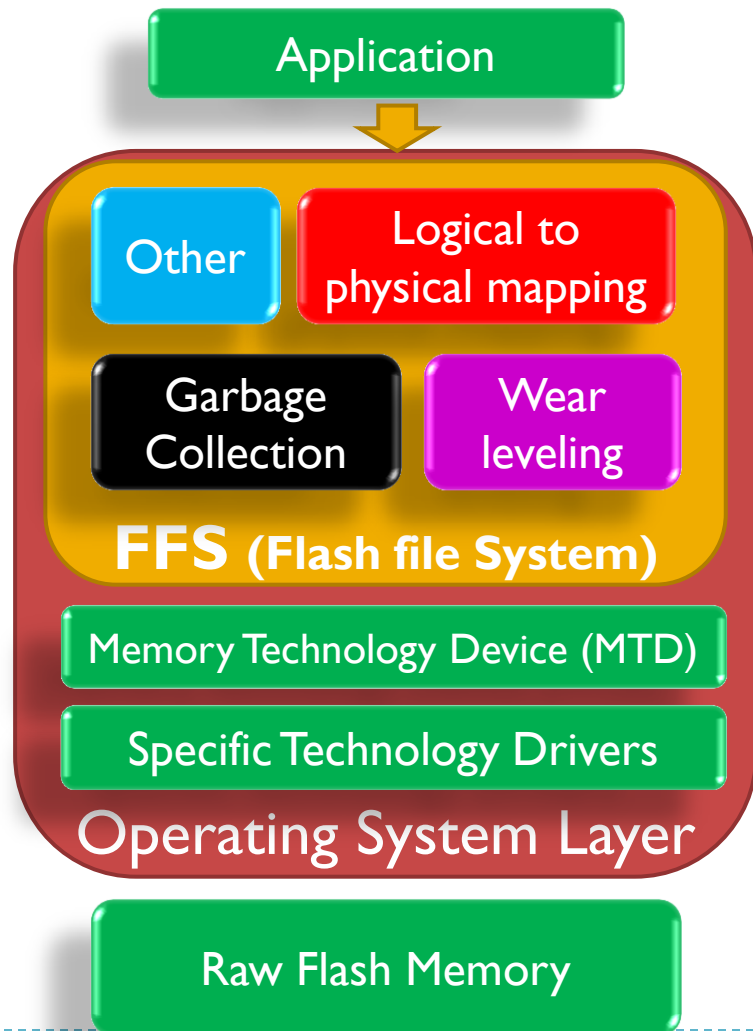
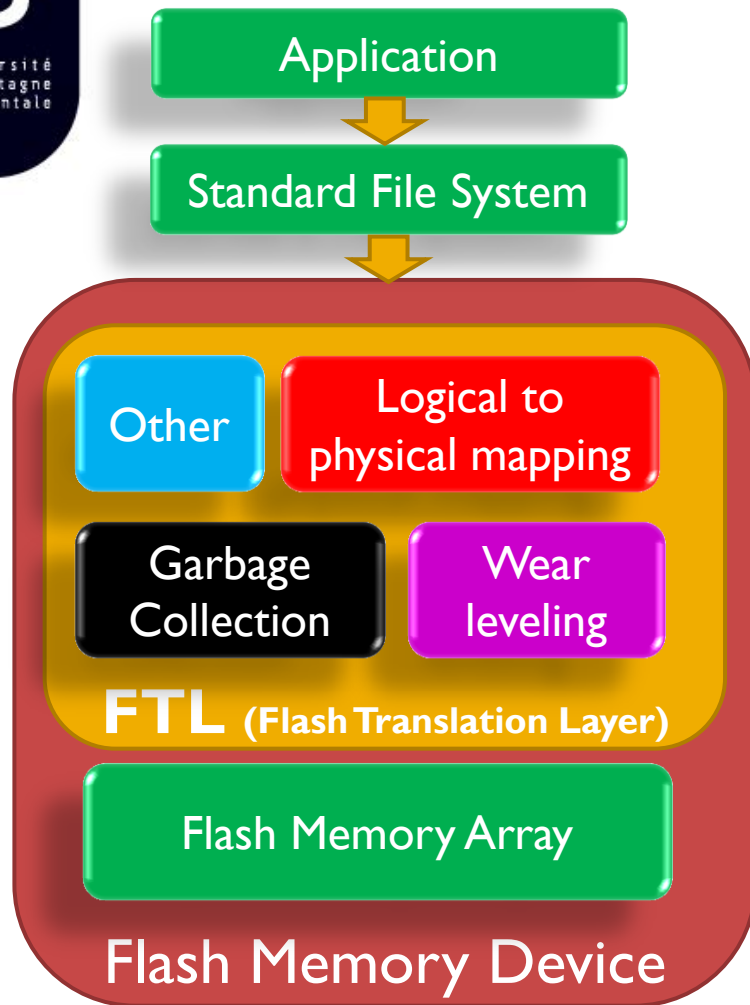
Inv. Op.



Moving people to a new city
and « erasing » the old one
to reuse the space !!!

- Moving valid pages from blocks containing invalid data and then erase/recycle the blocks

Flash memory structure



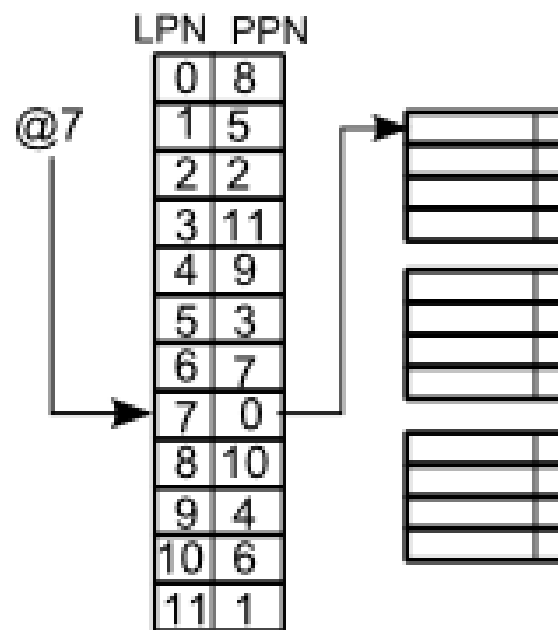
Presentation outline

1. Flash memory basics
2. Flash memory characteristics
3. **Flash memory support**
 1. Mapping schemes
 2. Wear leveling
 3. Garbage collection
 4. Flash specific cache systems
 5. Some contributions
4. Performance & energy considerations
5. Flash memory interfacing
6. Conclusions & perspectives

Basic mapping schemes

a- Page mapping ideal scheme

- ▶ Each page mapped independently
- ▶ High flexibility
- ▶ Ideal performance
- ▶ High RAM usage → unfeasible
 - ▶ 32GB flash memory, 2KB per page and 8 bytes/table entry → 128MB table !!!
- ▶ Optimal performance reference

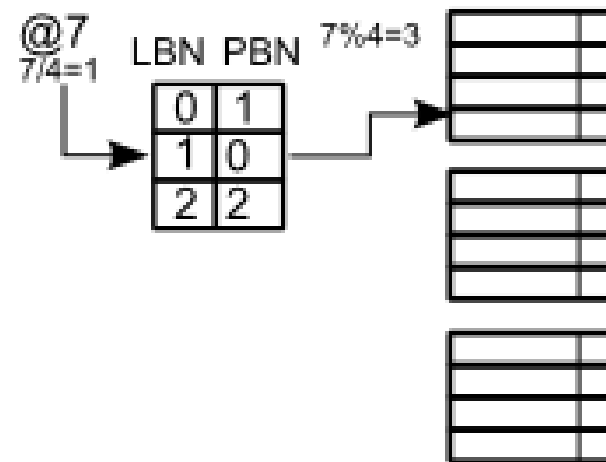


(a) Page mapping table

Basic mapping schemes

b- Block mapping scheme

- ▶ Only blocks numbers in the mapping table
- ▶ Page offsets remain unchanged
- ▶ Small mapping table (memory footprint)
- ▶ Very bad performance for write updates
- ▶ Same config. with 64 pages/block: 2MB page table

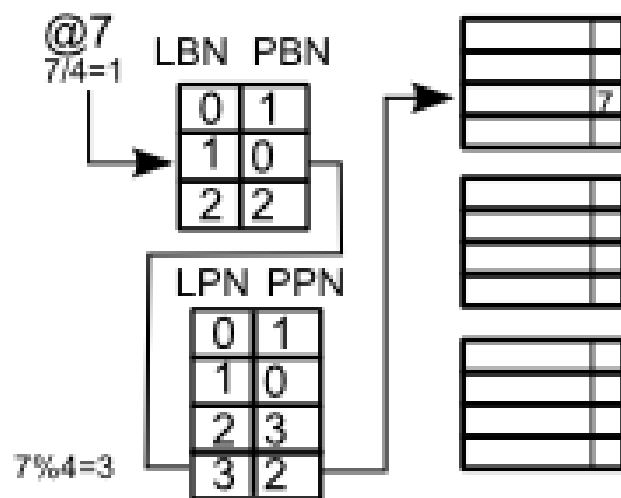


(b) Block mapping table

Basic mapping schemes

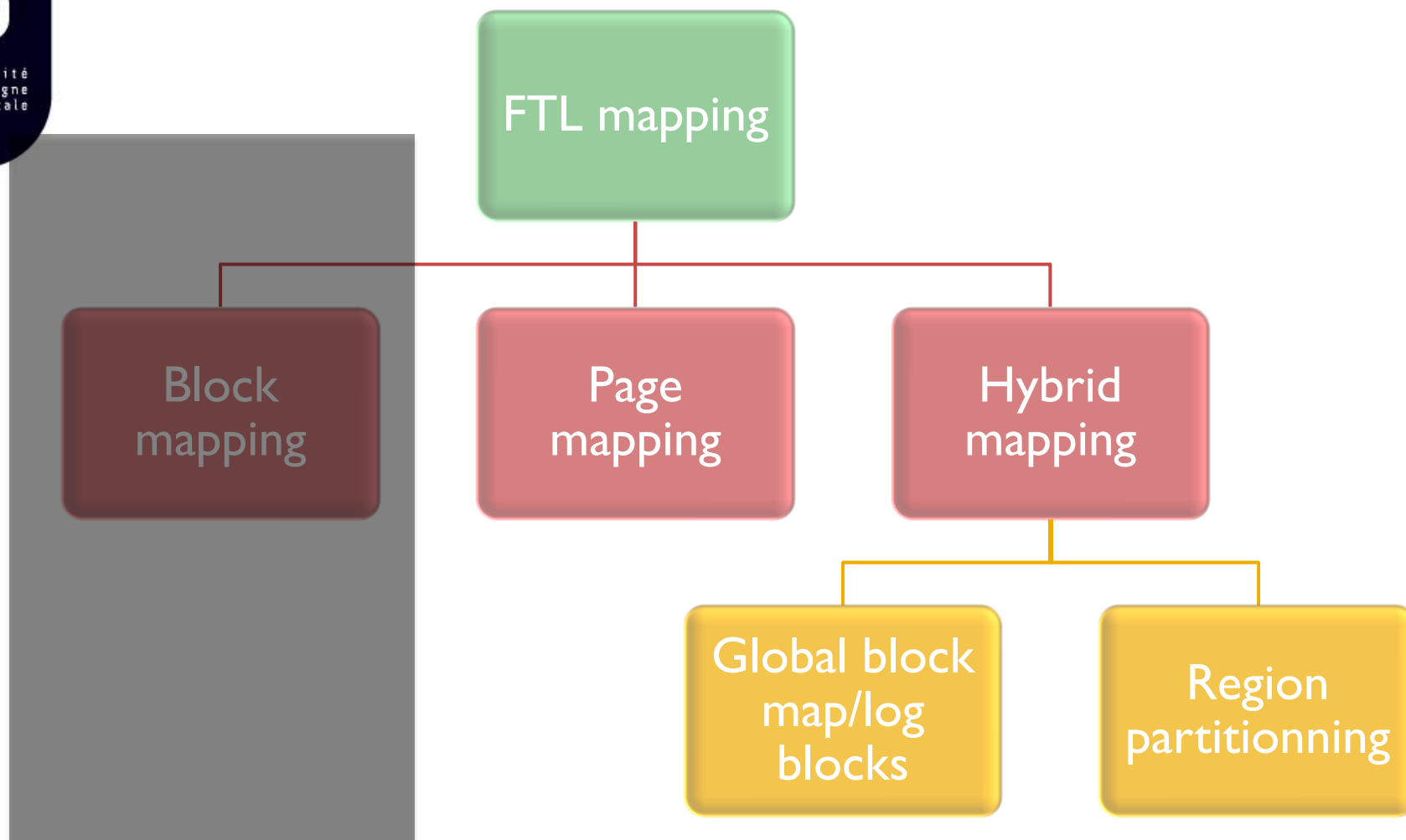
c- Hybrid mapping scheme

- ▶ Use of both block and page mappings
- ▶ Example:
 - ▶ Block mapping (BM)
 - ▶ Some data blocks are page mapped (PM)
- ▶ Current designs use whether:
 - ▶ One global BM and use log blocks
 - ▶ Partition flash memory in one BM region and a PM region
- ▶ Performance of PM with a memory footprint approaching BM



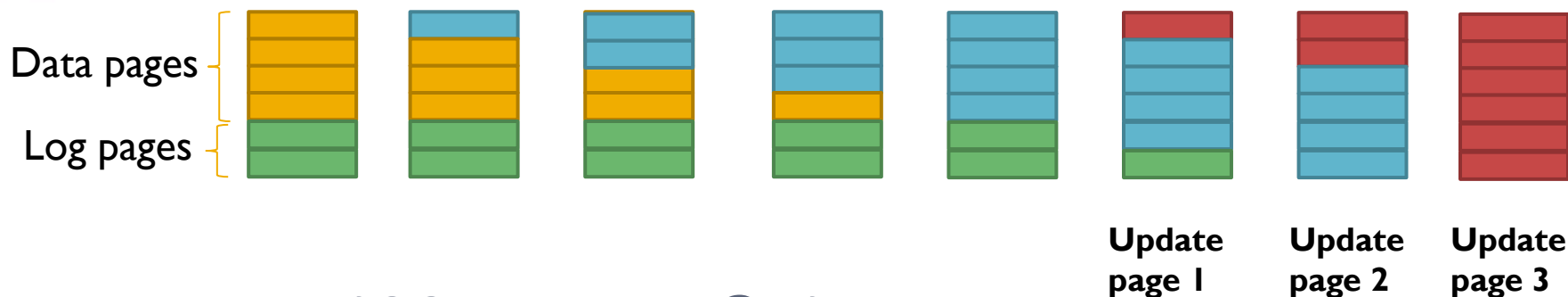
(c) Hybrid mapping tables

FTL complex mapping schemes



Global block map / log block based FTL

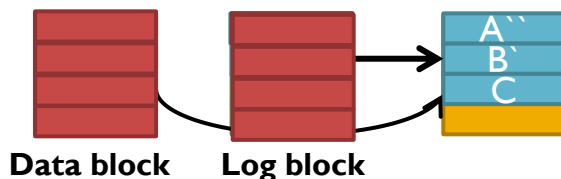
- ▶ PB with block mapping:
 - ▶ **Each page update:** one block erase op. + pages copy for each page update
- ▶ [Shinohara99] use of log pages in each block



- ▶ Use of OOB to save the @ of the data page written to the spare area
- ▶ [Ban99] use of log blocks: blocks dedicated to absorb data update
 - ▶ ANAND : respecting the page offset in log blocks (cons: no more than one same page update before merging data and log blocks)
 - ▶ FMAX: Allowing associativity in log blocks (use of OOB area)
 - ▶ 1 to 1 data/log block correspondance

Log block based FTL -2-

► Merge operations:



**Merge data and log
block into a free block**

- [RNFTL10] Reuse-Aware NAND FTL: only 46% of the data blocks are full before merge operation happen
 - ▶ erase only log block and use the rest of data block as log pages for other blocks
- [BAST02] Block Associative Sector Translation FTL
 - ▶ FMAX with \searrow log blocks
 - ▶ Log blocks managed by page mapping table
 - ▶ One log block for a given data block

Log block based FTL -3-

- ▶ **[FAST07] Fully Associative Sector Translation FTL**
 - ▶ Log blocks poorly filled because of associativity
 - ▶ Divide log blocks in 2 regions (spatial locality):
 - ▶ One sequentially written log block
 - ▶ Randomly accessed log blocks → fully associative → page mapped
- ▶ **[LAST08] Locality Aware Sector Translation FTL**
 - ▶ FAST: less merges operations but very costly because pages coming from many different blocks
 - ▶ As for FAST 2 regions
 - ▶ Big writes → sequential log blocks
 - ▶ Small write → random log blocks
 - In random region: hot and cold region to avoid costly merge operations

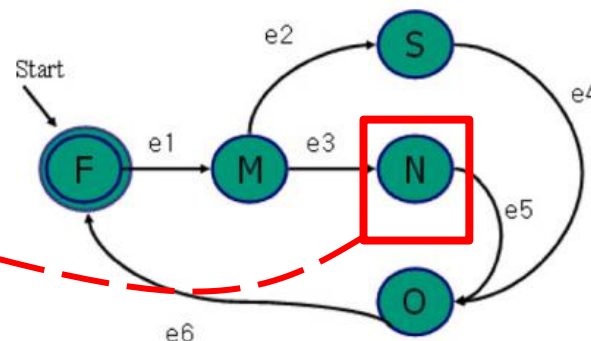
Log block based FTL -4-

- ▶ [EAST08] Efficient and Advanced Space management Technique
 - ▶ FAST: bad usage of log blocks ... yet not full
 - ▶ No sequential and random regions but:
 - ▶ **In-place** data updates in log blocks in first page
 - ▶ **Out-of-place** data updates if more updates are achieved
 - ▶ Fix the number of log blocks that can be dedicated to one data blocks according to flash characteristics
- ▶ [KAST09] K-Associative Sector Translation FTL
 - ▶ FAST: costly merge operation for the random region
 - ▶ Limit the associativity of log blocks (K)
 - ▶ Many sequential log blocks
 - ▶ Migration between random and sequential region

Hybrid FTLs, region partitioning

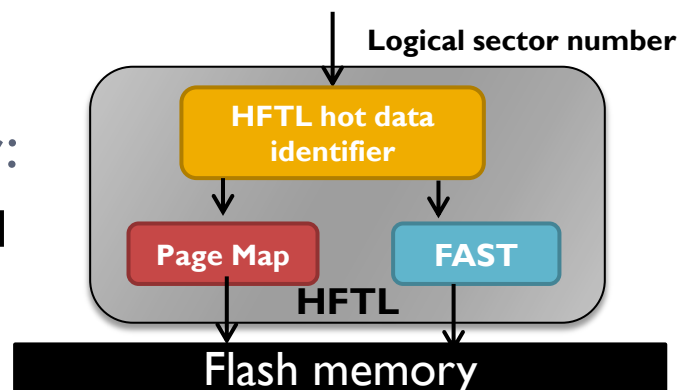
▶ [STAFF07] State Transition Applied Fast FTL

- ▶ Block: (F)ree state, (M)odified in-place, complete in-place (S)tate, modified out-of-place (N), or in (O)bsolute state (no valid data)
- ▶ Page mapping table for blocks in the N state.
- ▶ RAM usage unpredictability because of N



▶ [HFTL09] Hybrid FTL

- ▶ Use of hot data identifier:
 - ▶ Hot data are page mapped
 - ▶ Cold data use FAST



Hybrid FTLs, region partitioning -2-

► [WAFTLII] Workload adaptive FTL

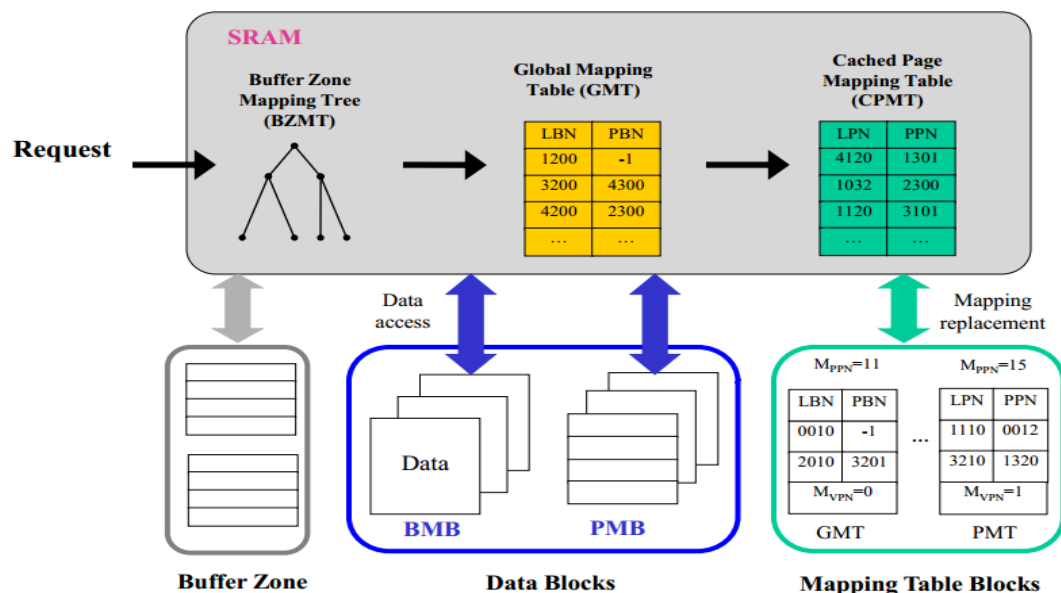
► Page mapped region

- Random data and partial updates

► Block mapped region

- Sequential data and mapping tables

Layout and In-memory Data Structure

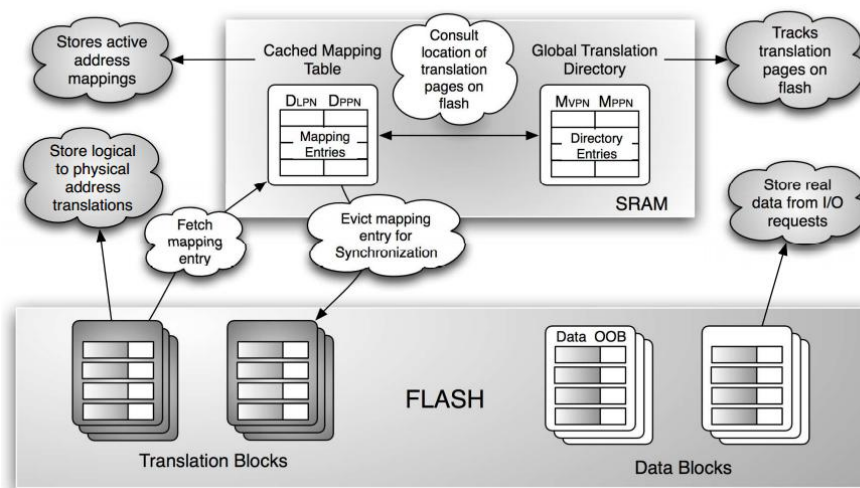


Source: <http://storageconference.org/2011/Presentations/Research/6.Wei.pdf>

Page mapping FTL

► [DFTL09] Demand based FTL

- Idea : use page mapping and keep only part of the mapping table in RAM
 - Rest of the mapping table stored in the flash



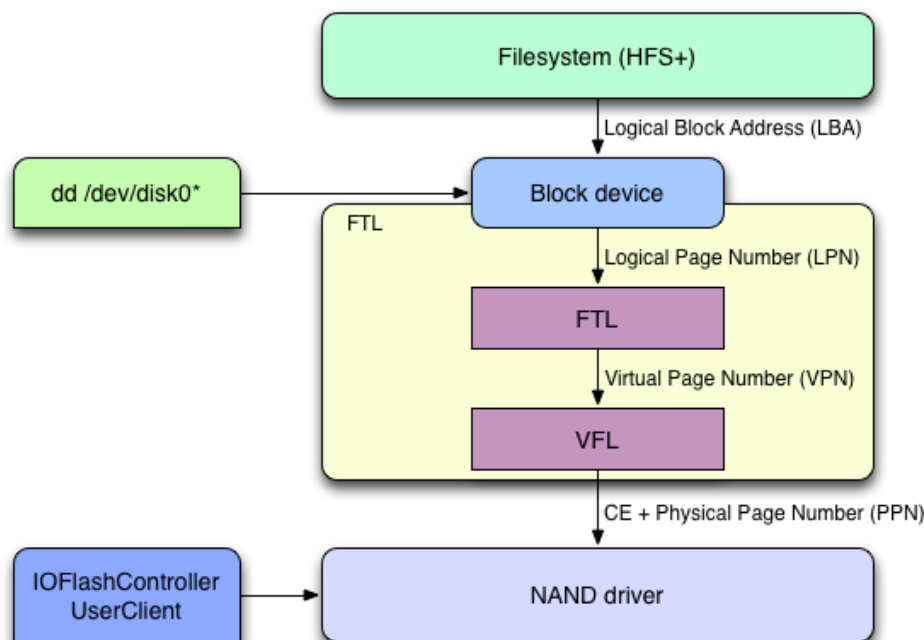
Source: [DFTL09]

► [SFTL11] Spatial locality FTL

- Reduces the size of the mapping table by keeping track of sequential accesses and use only one table entry.

Tablet Apple iPad

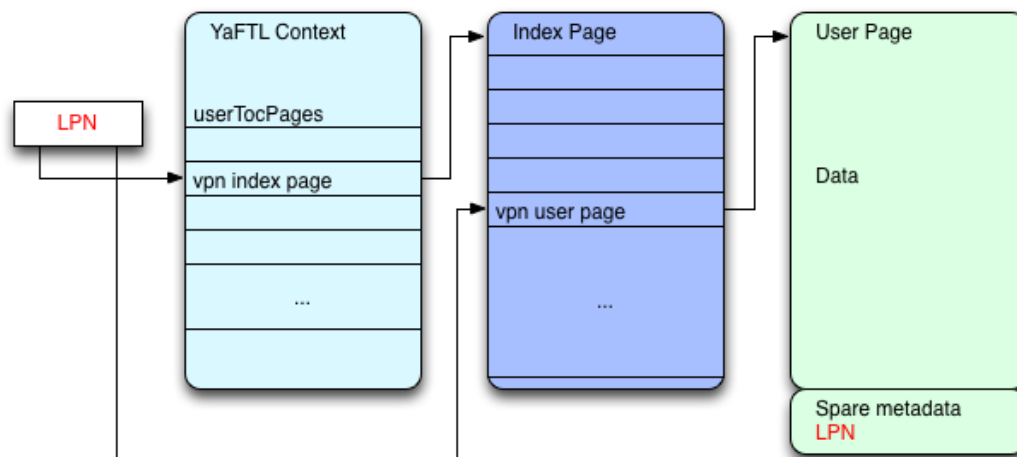
- ▶ Virtual Flash Layer (VFL): remaps bad blocks and presents an error-free NAND to the FTL layer
- ▶ FTL → YaFTL (Yet another FTL)



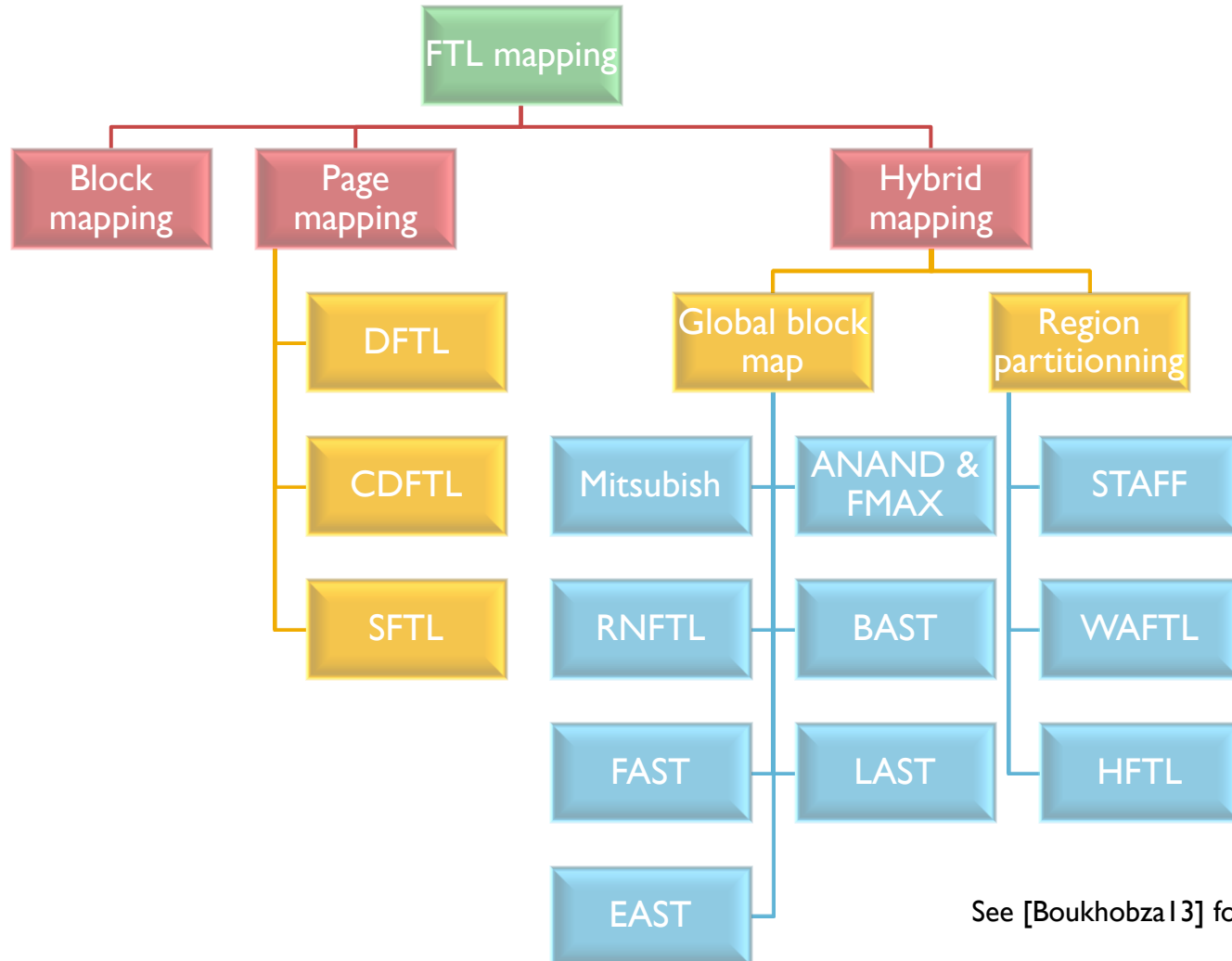
<http://esec-lab.sogeti.com/post/Low-level-iOS-forensics>

YaFTL

- ▶ Page mapping (implemented as a walk table)
 - ▶ DFTL principles: a part of the map is stored on the Flash
- ▶ Splits the virtual address space into *superblocks*
 - ▶ 3 types of superblocks
 - ▶ User data page
 - ▶ Index page (page map)
 - ▶ Context (index page+ erase count)



FTL (very partial) taxonomy



See [Boukhobza13] for more details

Presentation outline

1. Flash memory basics
2. Flash memory characteristics
3. **Flash memory support**
 1. Mapping schemes
 2. **Wear leveling**
 3. Garbage collection
 4. Flash specific cache systems
 5. Some contributions
4. Performance & energy considerations
5. Flash memory interfacing
6. Conclusions & perspectives

Wear leveling

- ▶ **Objective**: keep all the flash memory space usable as long as possible
- ▶ Based on the number of **erasures** or **writes** performed on a block
 - ▶ < mean value : cold block
 - ▶ > mean value: hot block
 - ▶ Maintain the gap between hot and cold block as small as possible
 - ▶ Swap data from hot blocks to cold blocks (costly)
 - ▶ Which blocks are concerned: only free ? All of them ?
- ▶ **[DualPool95]** adds 2 additional block mapping tables
 - ▶ Hot and cold table: free block taken from cold blocks
 - ▶ Periodically the mean erase number is recalculated and hot and cold tables updated

Wear leveling -2-

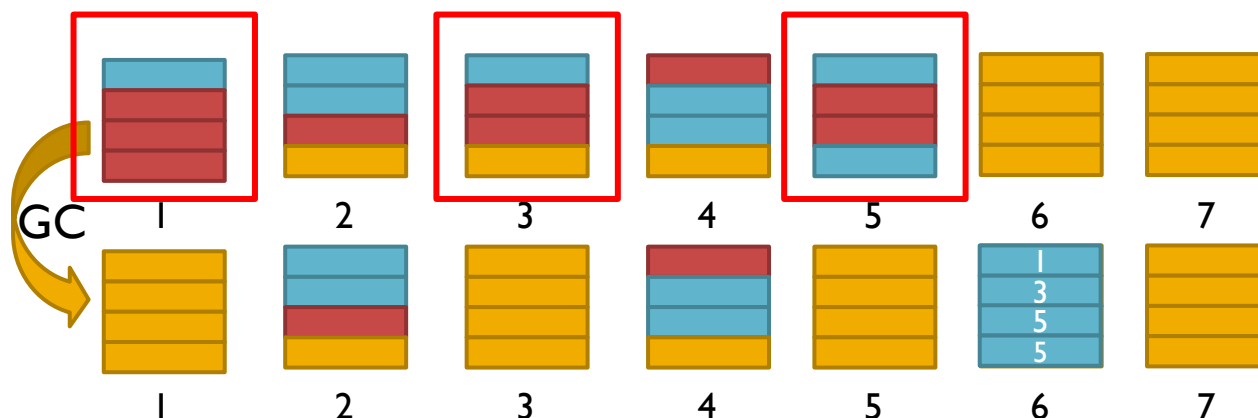
- ▶ Write count based wear leveler
- ▶ [Achiwa99]: erase count maintained in RAM in addition to write count
 - ▶ Put the more written data into the less erased blocks
- ▶ [Chang07]: like the dual pool but according to the number of writes (pages level)
- ▶ [Kwon11] considering groups of blocks reducing the RAM usage of the wear leveler

Presentation outline

1. Flash memory basics
2. Flash memory characteristics
3. **Flash memory support**
 1. Mapping schemes
 2. Wear leveling
 3. **Garbage collection**
 4. Flash specific cache systems
 5. Some contributions
4. Performance & energy considerations
5. Flash memory interfacing
6. Conclusions & perspectives

Garbage Collection / cleaning policy

- ▶ Process that recycles free space from previously invalidated pages in different blocks.
- ▶ Answers the questions:
 1. When should it be launched ?
 2. Which blocks to choose and how many ?
 3. How should valid data be written ?
 4. (where to write the new data ?) → wear leveler



1. Free blocks < 3
2. Containing the most invalid pages to free 3 blocks (1,3,5)
3. Respecting pages' placement

Garbage Collection -2-

- ▶ Minimizing **cleaning cost** while maximizing cleaned space
- ▶ **Cleaning cost:**
 - ▶ Number of erase operations
 - ▶ Number of valid pages copy
- ▶ Main considered metric: ratio of dirty pages in blocks
 - ▶ Maintaining counters in RAM or in metadata area
- ▶ Separate cold and hot data when performing valid pages copy (Q. 3 In previous slide)
 - ▶ Otherwise GC frequently launched (due to hot data updates)
 - ▶ [DAC08] Dynamic dAta Clustering: partitioning flash memory into many regions depending on update frequency

Garbage Collection -3-

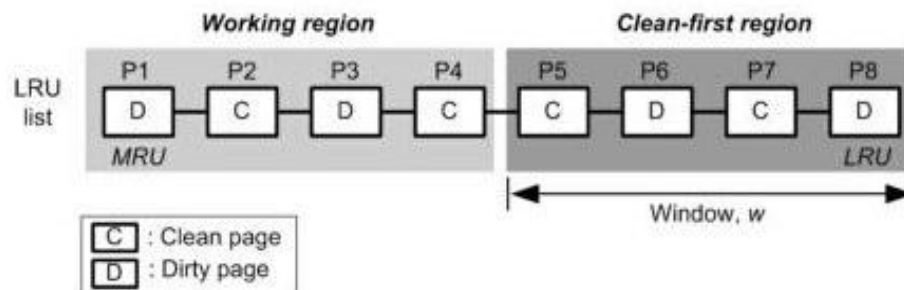
- ▶ Which blocks to choose (Q. 2)
 - ▶ Greedy policy: blocks with the most dirty pages
 - ▶ Efficient if flash memory accessed uniformly
- ▶ [Kawaguchi95]
 - ▶ Space recycled: $(1-u)$, cost of read and write valid data: $(2u)$
 - ▶ Elapsed time since the last modification: age
 - ▶ $\text{Age} * (1-u)/2u$ block with the highest score is chosen
- ▶ [CAT99]: Cost AgeTimes
 - ▶ Hot blocks are given more time to accumulate more invalid data
→ direct erase without GC
 - ▶ $\text{CleaningCost} * (1/\text{age}) * \text{NumberOfCleaning}$
 - ▶ CleaningCost: $u/(1-u)$ u : percentage of valid data in the block
 - ▶ NumberOfCleaning: number of generated erases
 - ▶ The less the score, the more chances to be cleaned

Presentation outline

1. Flash memory basics
2. Flash memory characteristics
3. **Flash memory support**
 1. Mapping schemes
 2. Wear leveling
 3. Garbage collection
 4. **Flash specific cache systems**
 5. Some contributions
4. Performance & energy considerations
5. Flash memory interfacing
6. Conclusions & perspectives

Flash specific cache systems

- ▶ They are mostly write specific and try to:
 - ▶ absorb most page/block write operations at the cache level
 - ▶ reveal sequentiality by buffering write operations and reorganizing them
- ▶ [CFLRU06] Clean First LRU (caches reads & writes)
 - ▶ LRU list divided into two regions:
 - ▶ A working region: recently accessed pages
 - ▶ A clean first region: candidate for eviction
 - ▶ It evicts first clean pages that do not generate any write (e.g. P7, P5, P8, then P6.)



Flash specific cache systems -2-

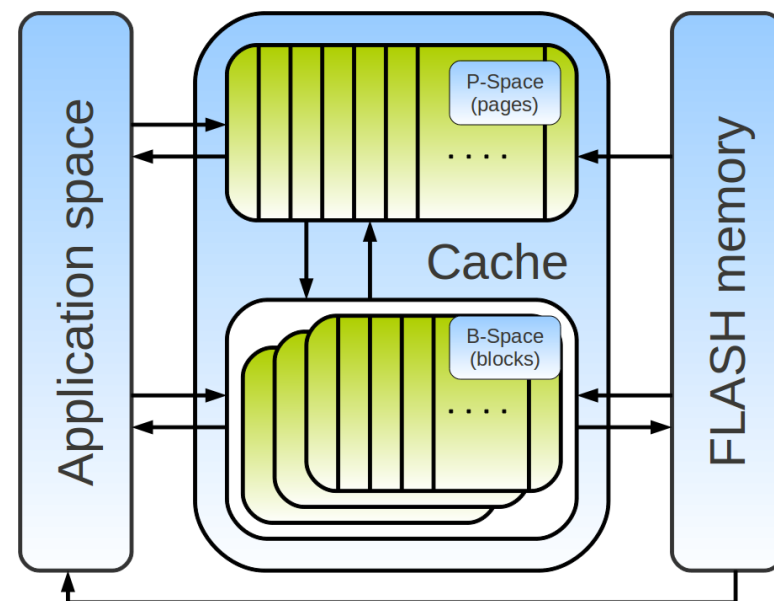
- ▶ **[FAB06] Flash Aware Buffer**
 - ▶ Flushes the largest groups of pages belonging to the same block (minimize merges)
 - ▶ If the same number of pages: uses LRU
- ▶ **[BPLRU08] Block Padding LRU**
 - ▶ Block level LRU scheme
 - ▶ Page padding: read lacking pages and flush full blocks
 - ▶ LRU compensation: sequentially written data are moved to the end of the LRU queue
- ▶ **LRU like algorithms, page/block granularity, & double objective : caching, reducing erasures**

Presentation outline

1. Flash memory basics
2. Flash memory characteristics
3. Flash memory support
 1. Mapping schemes
 2. Wear leveling
 3. Garbage collection
 4. Flash specific cache systems
 5. **Some contributions**
4. Performance & energy considerations
5. Flash memory interfacing
6. Conclusions & perspectives

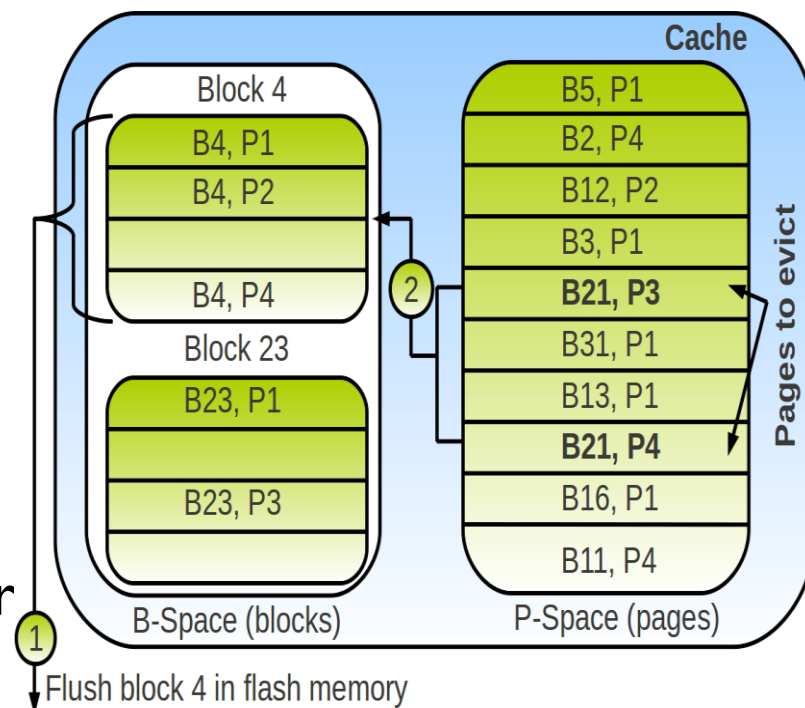
C-lash: a cache for flash [C-lash11]

- ▶ Hierarchical cache
- ▶ Cache with no WL or GC
- ▶ 2 regions:
 - ▶ Page region (P-space)
 - ▶ Block region (B-space)
- ▶ Read operations
 - ▶ Hit: Read from cache
 - ▶ Miss: no copy to the cache
- ▶ Write operations
 - ▶ Hit: update in the cache
 - ▶ Miss: write in the P-spaces



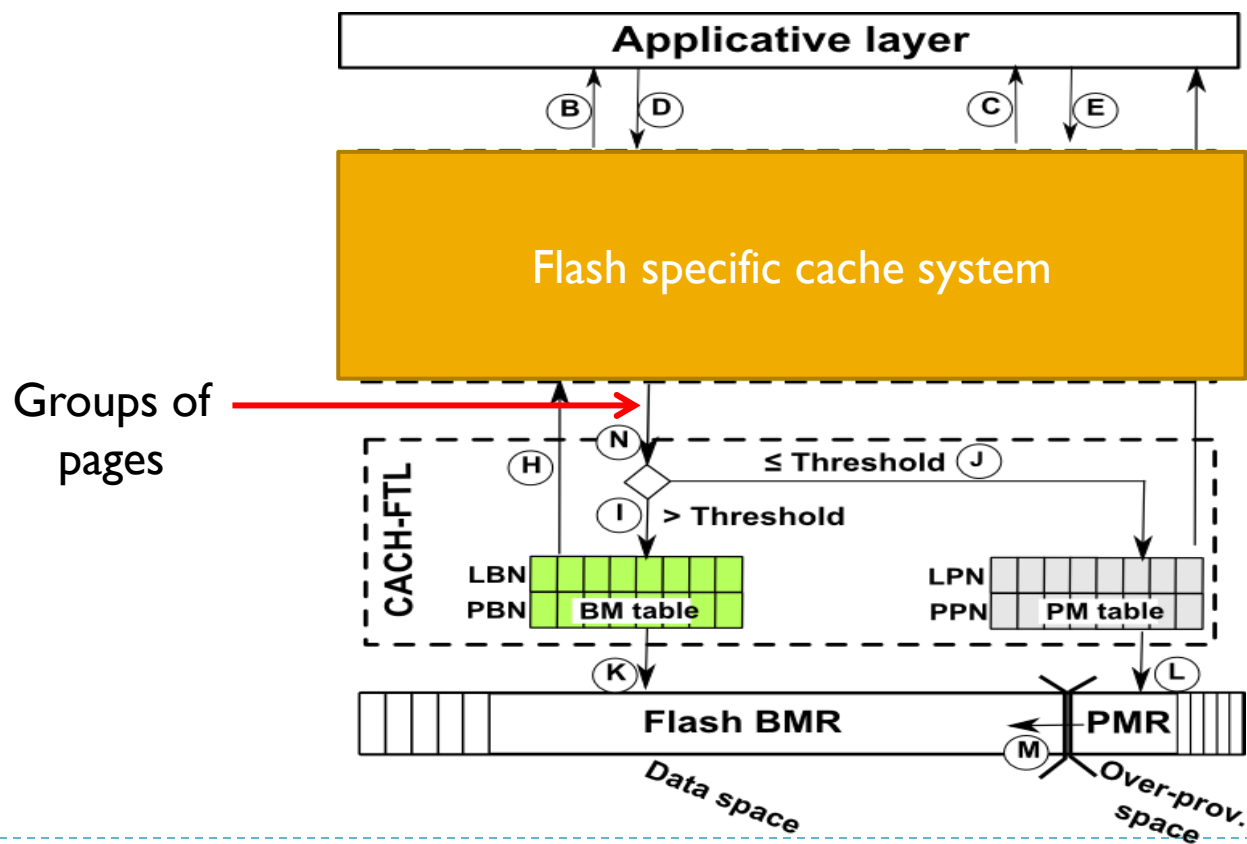
C-lash: a cache for flash -2-

- ▶ 2 eviction policies
 - ▶ P-space → B-space: largest set of pages from the same block (spatial locality)
 - ▶ B-space → flash: LRU (temporal locality)
- ▶ Early and late cache merge in B-space.
- ▶ Switch (p-space/b-space)
- ▶ Proved good performance for mostly sequential workload
 - ▶ Bad for very random ones



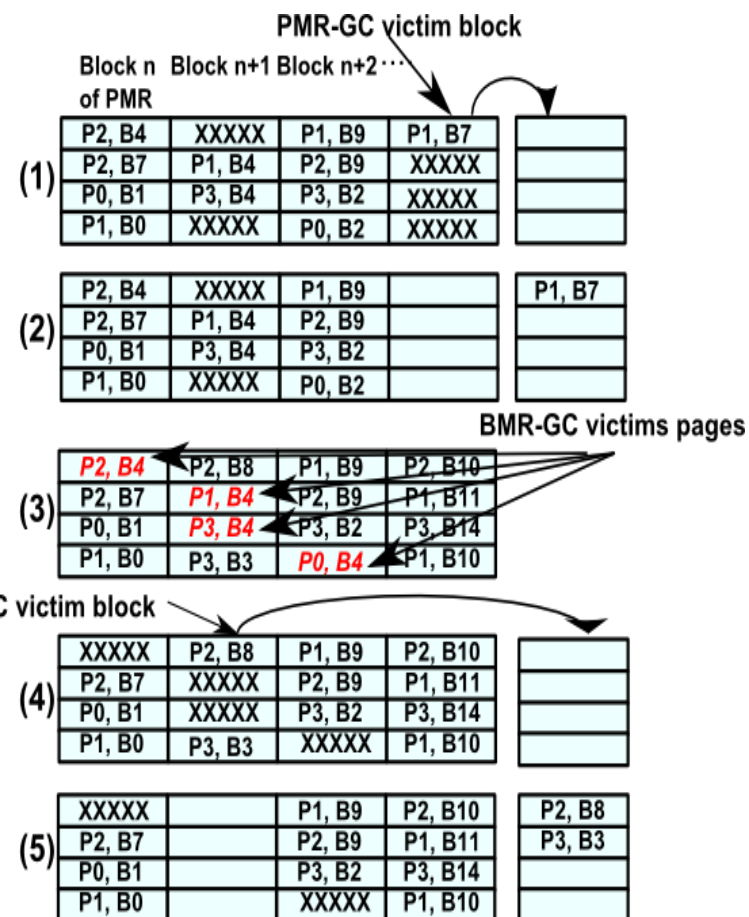
CACH-FTL Cache-Aware Configurable Hybrid FTL [CACH-FTL13]

- ▶ Most flash systems have cache mechanisms on top of FTL
- ▶ Most flash specific cache systems flush groups of pages

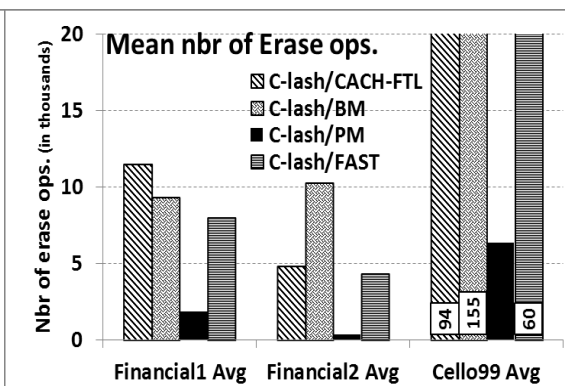
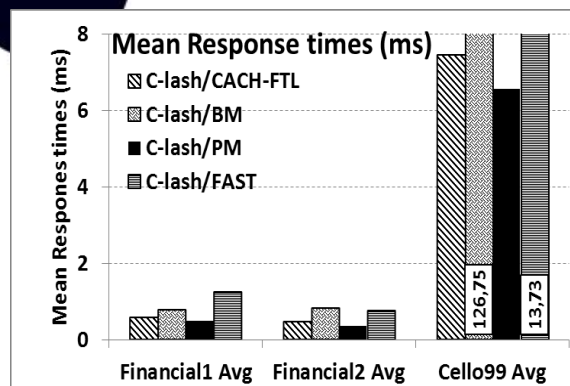


CACH-FTL -2-

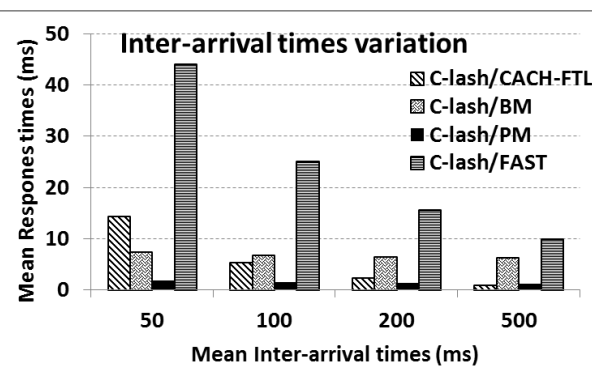
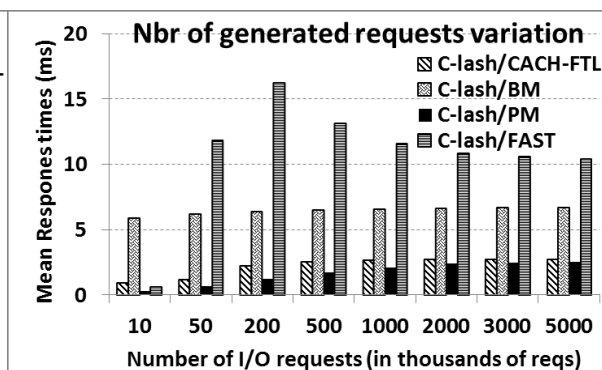
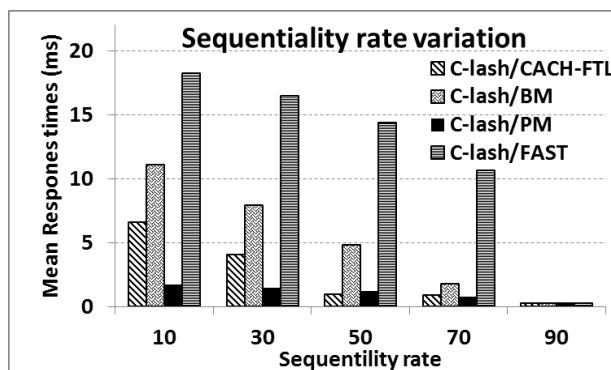
- ▶ **PMR (Page Mapped Region)**
garbage collection:
 - ▶ Launched when the number of free blocks in the PMR goes under a predefined threshold
 - ▶ Greedy reclamation algorithm (least number of valid pages)
- ▶ **BMR (Block Mapped Region)**
garbage collections
 - ▶ PMR-GC cannot find any physical block containing enough invalid pages to recycle a block
 - ▶ Greedy reclamation algorithm selecting the largest group of PMR pages belonging to the same data block



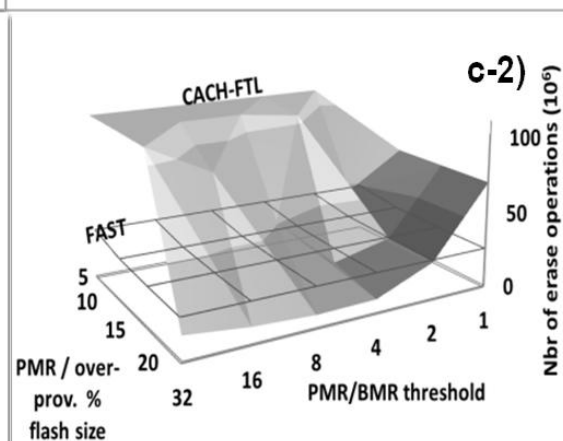
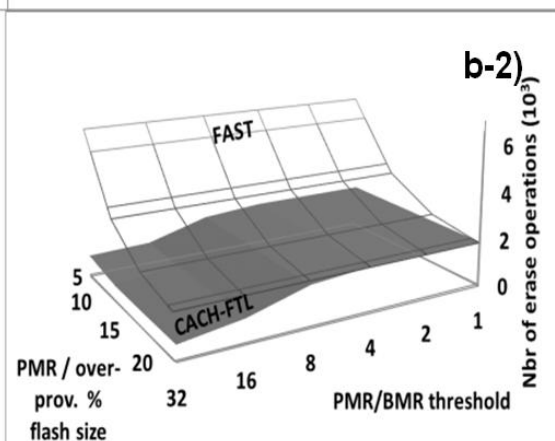
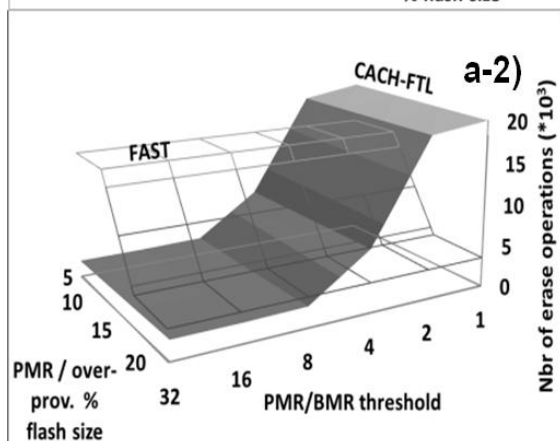
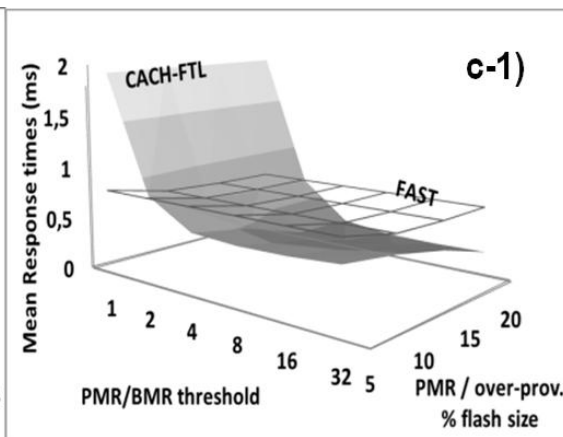
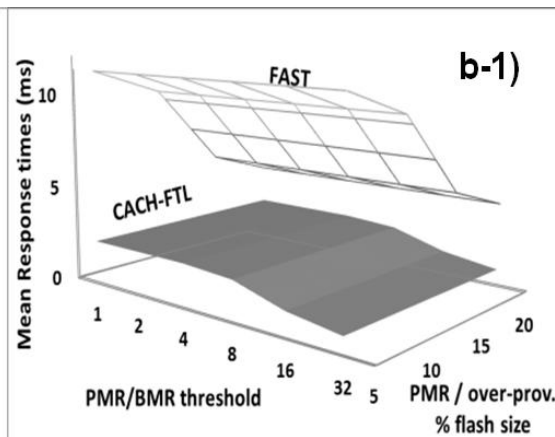
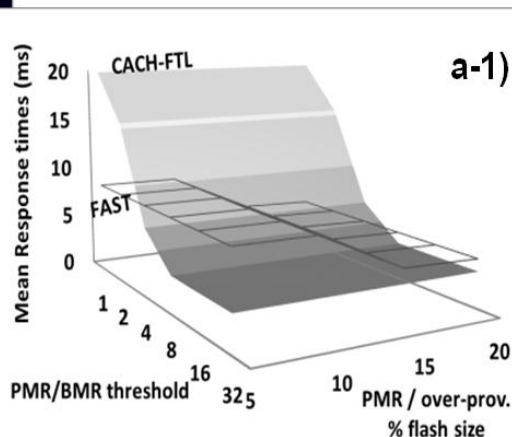
CACH-FTL -3-



- Flashsim+Disksim simulator
- CACH-FTL configuration:
 - threshold = 8 pages
 - Over-prov. of 10%
- OLTP real traces + Synth. traces



CACH-FTL -4-



Presentation outline

1. Flash memory basics
2. Flash memory characteristics
3. Flash memory support
 1. Mapping schemes
 2. Wear leveling
 3. Garbage collection
 4. Flash specific cache systems
 5. Some contributions
4. **Performance & energy considerations**
5. Flash memory interfacing
6. Conclusions & perspectives

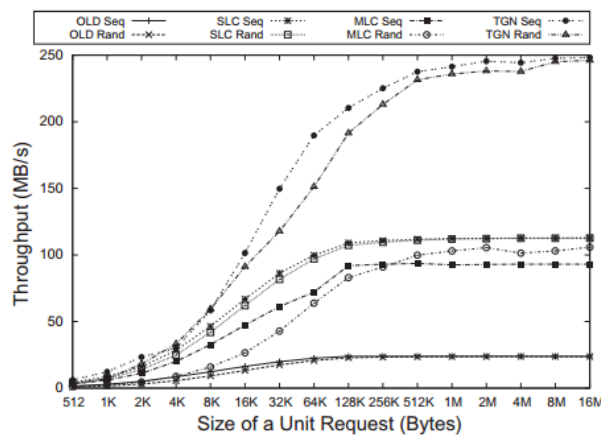
Performance and energy considerations

- ▶ 2006 → 2012: nearly exponential growth of published work on flash memory
- ▶ “*Tape is dead, disk is tape, flash is disk, RAM locality is King*” Jim Gray 2006
- ▶ **Flash disks outperform hard disk drives (HDD)**
 - ▶ Sequential reads and writes
 - ▶ Random reads (no mechanical elements)
 - ▶ Random writes → Achilles' heel
 - ▶ Depends on flash intricacies
 - ▶ Flash disks are generally more energy efficient
 - ▶ More than 5x less energy in some cases [Park11]

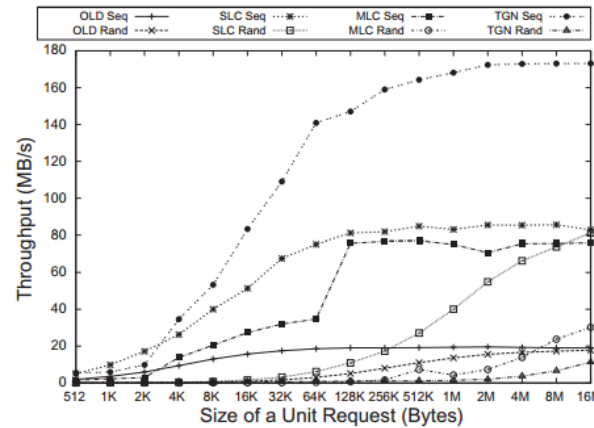
Performance and energy -2-

► Flash disk performance is heterogeneous

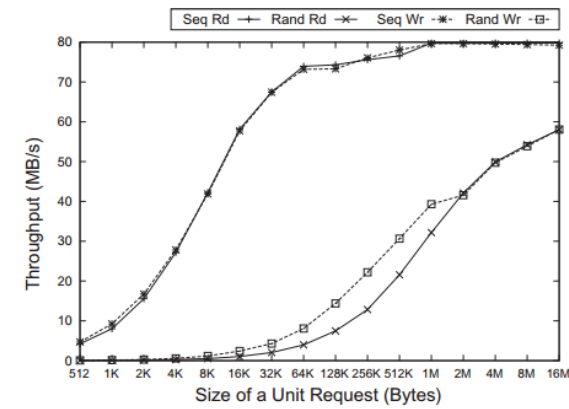
- Depend on internal structure and workload
- Performance disparities between SSDs from the same constructor and between different technologies are significant [Park I I]



(a) SSD read



(b) SSD write



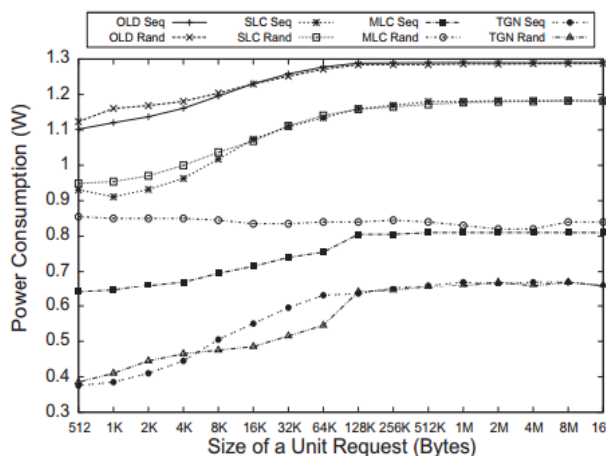
(c) HDD read/write

Specifications of the storage devices used in our work.

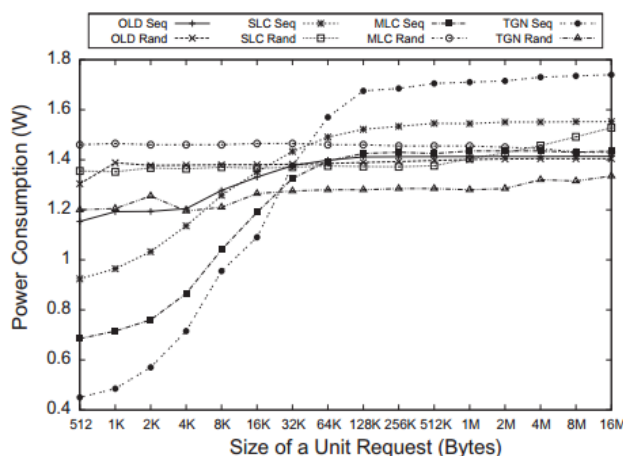
	OLD	MLC	SLC	TGN	HDD
Model	FSD32GB25M	1C32G	MSP7000	MMCRE28G5	WD1600BEKT
Vendor	Super talent	OCZ	MTron	Samsung	Western digital
Form factor	2.5 in.	2.5 in.	2.5 in.	2.5 in.	2.5 in.
Flash type/RPM	SLC	MLC	SLC	MLC	7200
Capacity	32 GB	32 GB	16 GB	128 GB	160 GB
Rd./Wr. Perf. (MB/s)	60/45	143/93	120/90	220/200	NA/NA

Performance and energy -3-

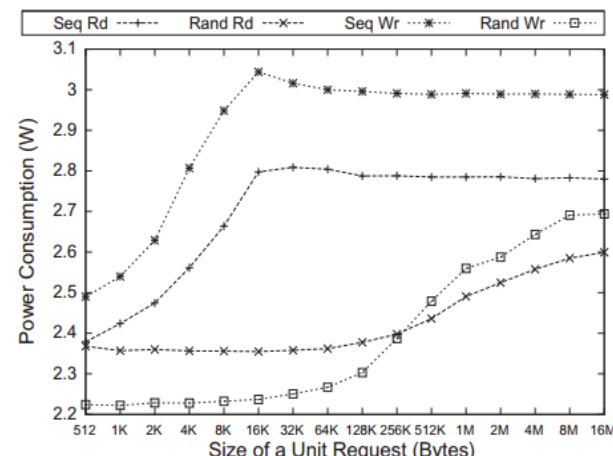
- ▶ A wide performance and energy asymmetry between reads and writes [Park11]
- ▶ The more free space the better the write performance



(a) SSD read



(b) SSD write



(c) HDD read/write

Specifications of the storage devices used in our work.

	OLD	MLC	SLC	TGN	HDD
Model	FSD32GB25M	1C32G	MSP7000	MMCRE28G5	WD1600BEKT
Vendor	Super talent	OCZ	MTTron	Samsung	Western digital
Form factor	2.5 in.	2.5 in.	2.5 in.	2.5 in.	2.5 in.
Flash type/RPM	SLC	MLC	SLC	MLC	7200
Capacity	32 GB	32 GB	16 GB	128 GB	160 GB
Rd./Wr. Perf. (MB/s)	60/45	143/93	120/90	220/200	NA/NA

Performance and energy -4-

Flash performance needs time to reach steady state.

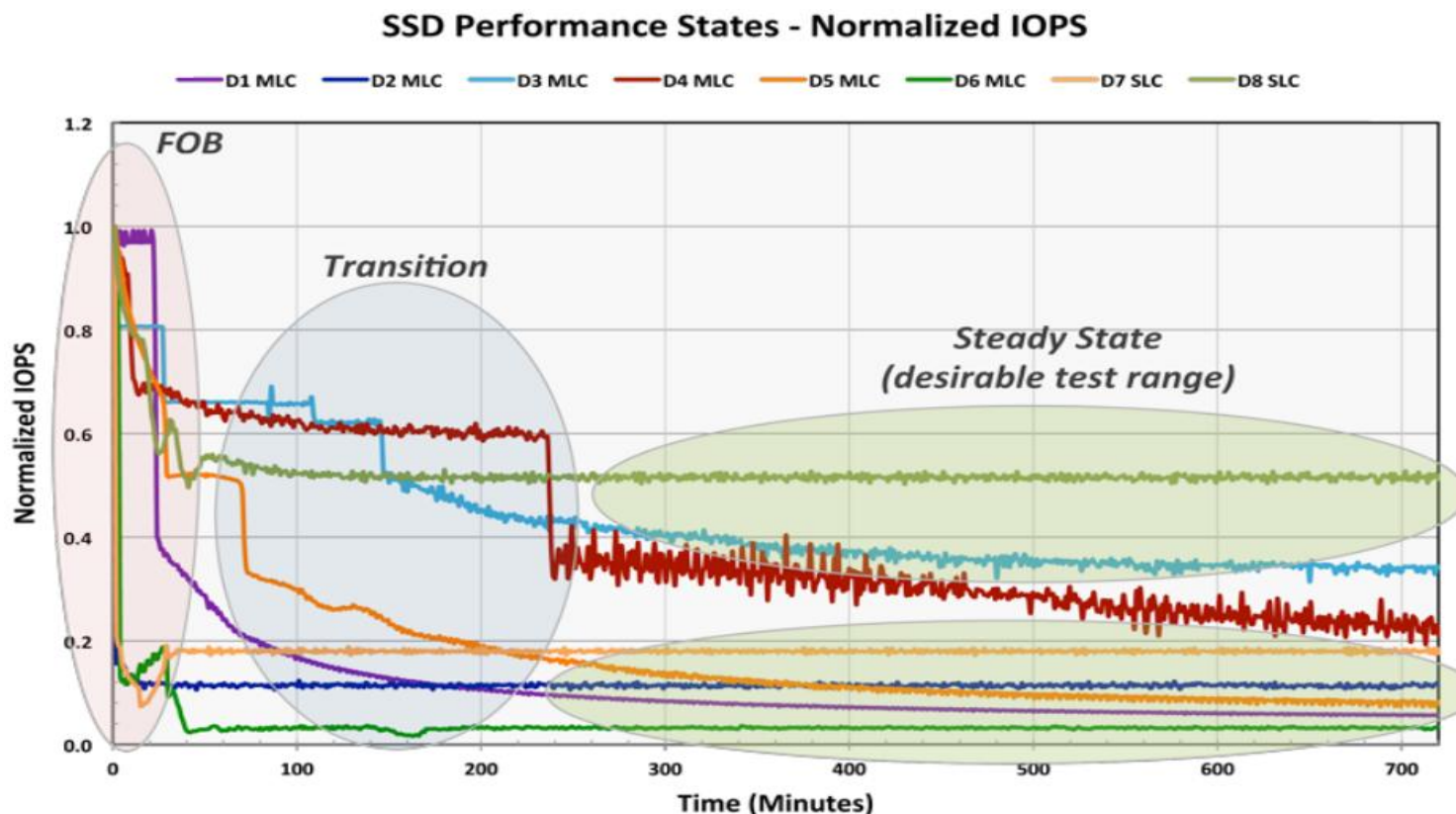


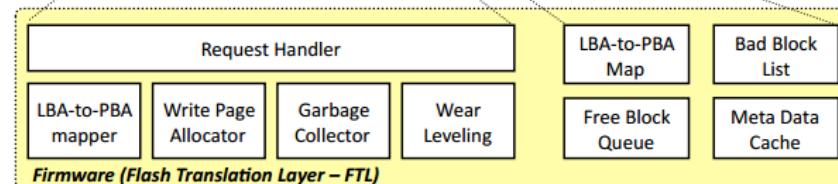
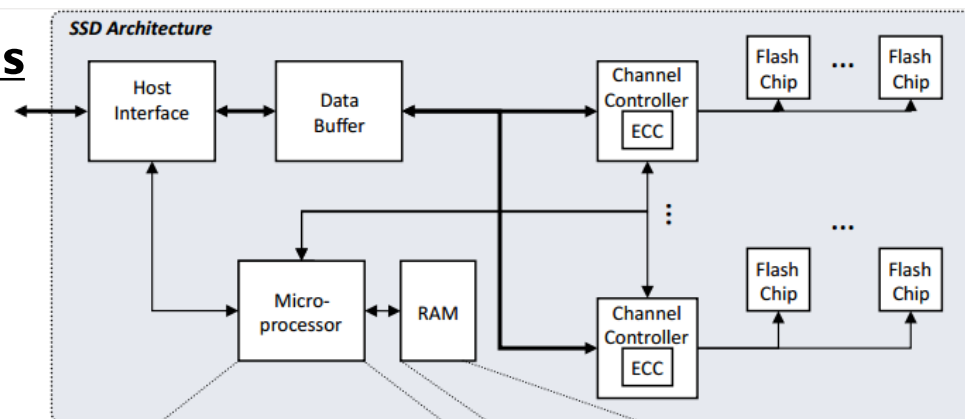
Figure 1-1 – NAND-based SSS Performance States (RND 4KiB Writes)

Source: <http://snia.org/sites/default/files/SSS%20PTS%20Client%20-%20v1.1.pdf>

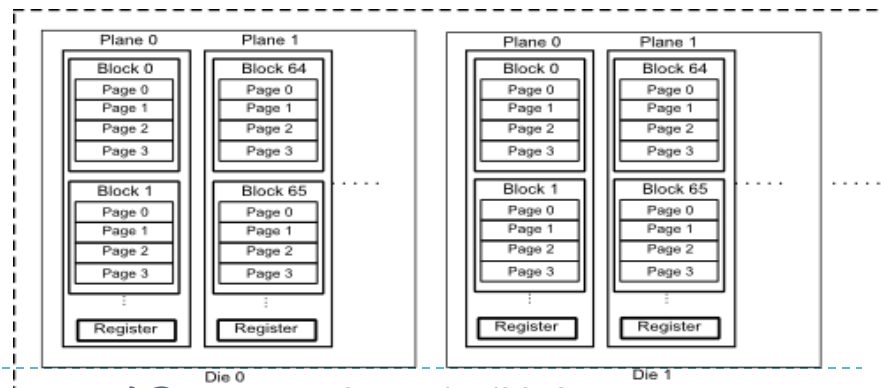
Performance and energy -5-

Flash memory design space is large !

- ▶ Different FTLs (mapping, WL, GC)
- ▶ Degree of concurrency [Agrawal08]:
 - ▶ **Parallel requests:** parallel requests to each element of the flash array, a queue per element.
 - ▶ **Ganging:** using a gang of flash elements in synchrony to optimize multi-page request
 - ▶ **Interleaving:** within a die
 - ▶ **Background cleaning**
- ▶ Capacity > 2TB, cost 0.7€/GB



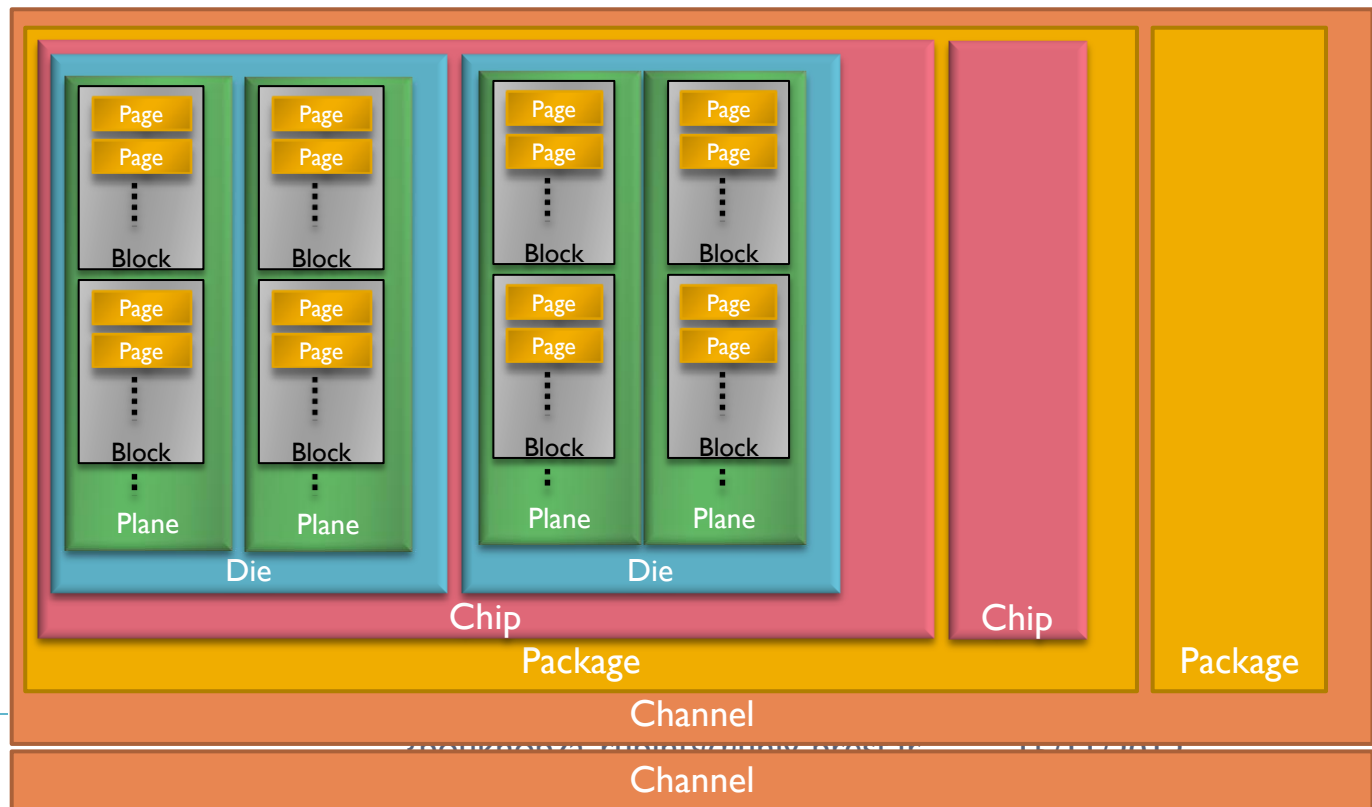
Source: Bonnet, Bouganin, Koltsidas, Viglas, VLDB 2011



Performance and energy -6-

► Intra SSD performance:

- SSD System level: among channels
- Chip-level: among chips in a channel
- Dies level: among dies in a chip
- Among planes in a die



Performance and energy -6-

Off-the-shelf SSDs

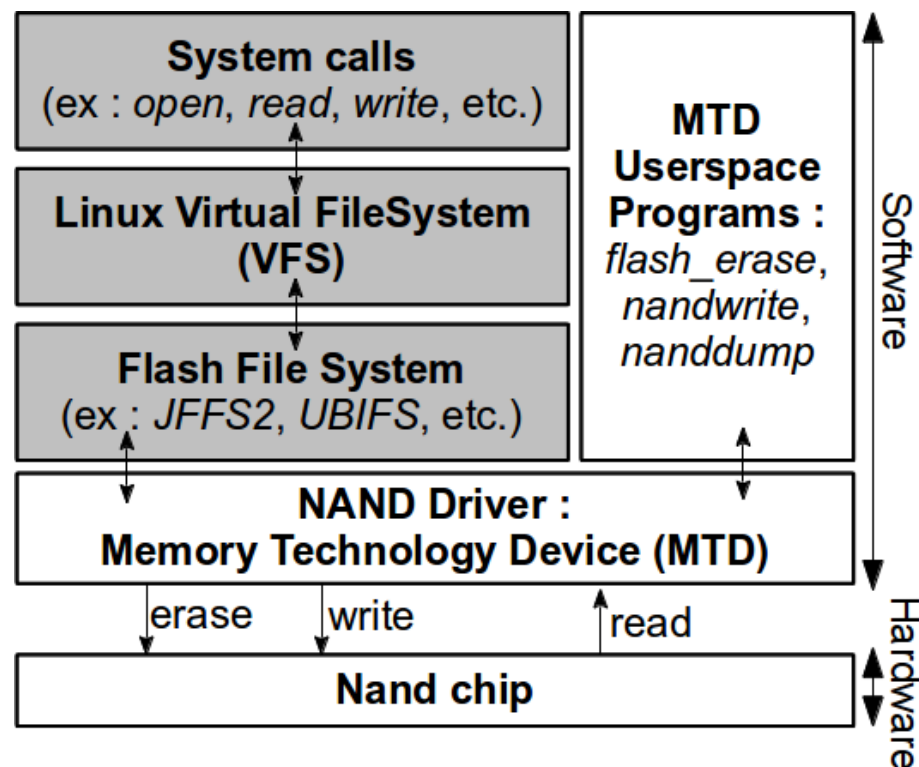
15k RPM SAS HDD: ~250-300 IOPS
7.2k RPM SATA HDD: ~80 IOPS

	A	B	C	D	E
Form Factor	PATA Drive Consumer	SATA Drive Consumer	SATA Drive Consumer	SAS Drive Enterprise	PCI-e card Enterprise
Flash Chips	MLC	MLC	MLC	SLC	SLC
Capacity	32 GB	100 GB	160GB	140GB	450 GB
Read Bandwidth	53 MB/s	285 MB/s	250 MB/s	220 MB/s	700 MB/s
Write Bandwidth	28 MB/s	250 MB/s	100 MB/s	115 MB/s	500 MB/s
Random 4kB Read IOPS	3.5k	30k	35k	45k	140k
Random 4kB Write IOPS	0.01k	10k	0.6k	16k	70k
Street Price:	~ 15 \$/GB (2007)	~ 4 \$/GB (2010)	~ 2.5 \$/GB (2010)	~18 \$/GB (2011)	~ 38 \$/GB (2009)

Contributions

Power consumption & performance modeling of embedded systems with an embedded OS (Pierre Olivier PhD)

- ▶ Performance and energy consumption at different layers.
 - ▶ Microbenchmarking different FFS / different initial states
- ▶ Simple, atomic access. Legacy NAND commands :
 - ▶ Read and write (page)
 - ▶ Erase (block)



Linux :

MTD Device versus Block Device

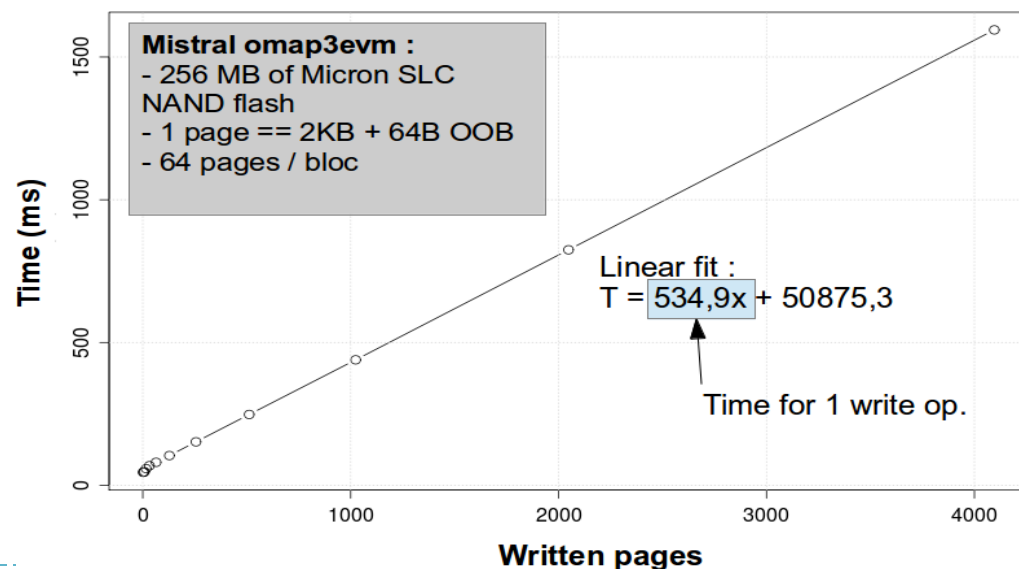
Block device	MTD device
Consists of sectors	Consists of eraseblocks
Sectors are small (512, 1024 bytes)	Eraseblocks are larger (typically 128KiB)
Maintains 2 main operations: read sector and write sector	Maintains 3 main operations: read from block , write to eraseblock , and erase eraseblock
Bad sectors are re-mapped and hidden by hardware	Bad eraseblocks are not hidden and should be dealt with in software
Sectors are devoid of the wear-out property	Eraseblocks wear-out and become bad and unusable

Source: <http://www.linux-mtd.infradead.org/>

Performance

- ▶ Tests programs:
 - ▶ Kernel level : modules (MTD low level calls)
 - ▶ MTD-userspace : shell scripts
- ▶ Ex : **writes** (MTD kernel level)

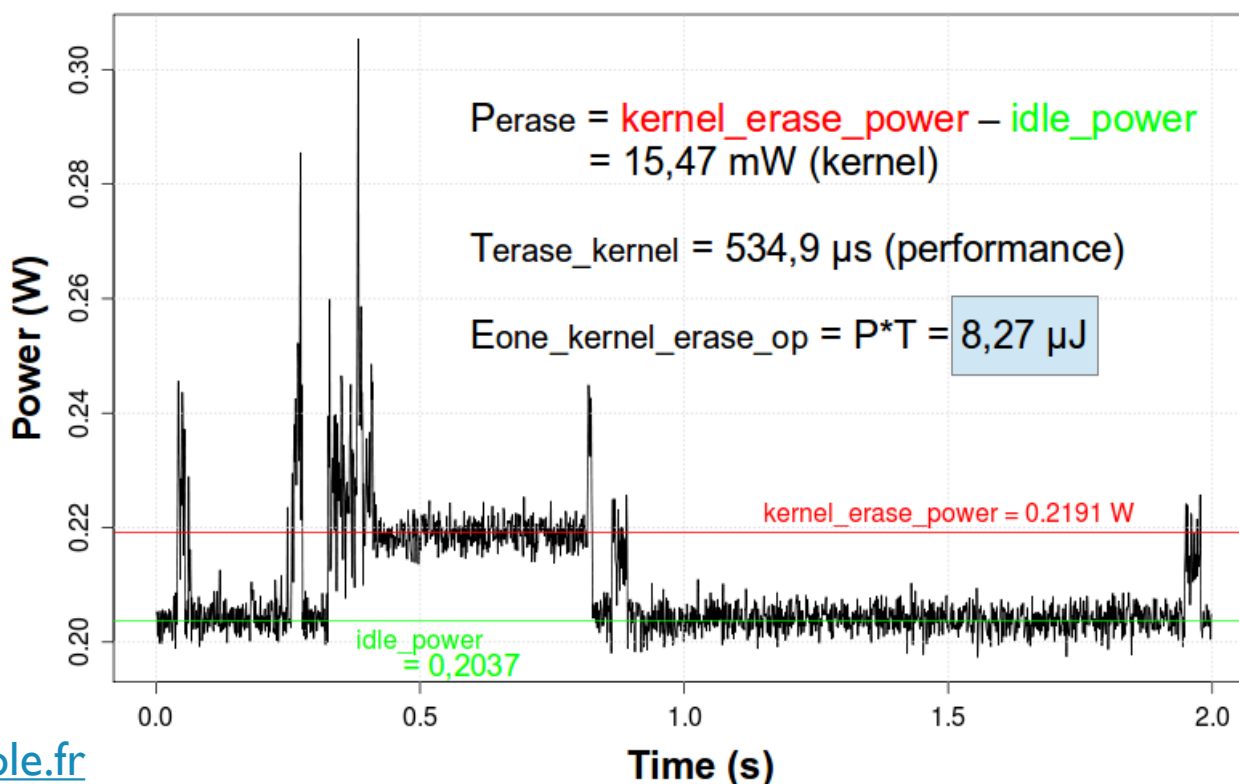
```
/* ex : test program for write operations */
for(i=0 ; i< N ; i++)
{
    offset= i*flash_page_size
    flash_write(offset, flash_page_size, buf)
}
```



Power consumption

- ▶ Same test programs, max access on test partition
- ▶ Ex : **erases**(MTD kernel level)

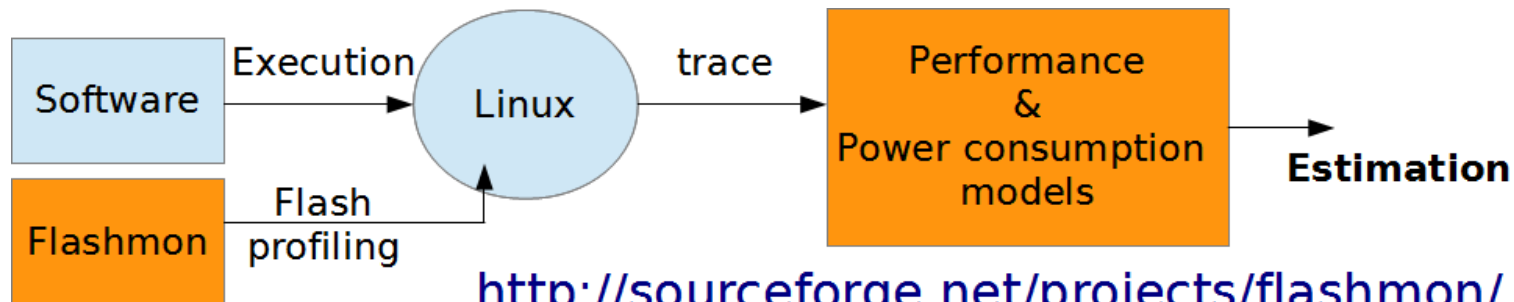
Kernel erase test module power consumption



<https://www.open-people.fr>

Used tool: Flashmon [Flashmon11]

- ▶ Flash access profiler (kernel module) for raw flash-based Linux embedded systems
 - ▶ Monitors and log events (read, write, erase)
 - ▶ Stores access counter for each block of the monitored flash memory
- ▶ Linux programs profiled with Flashmon
- ▶ Flash access numbers injected into the models to estimate Flash I/O time and power consumption



<http://sourceforge.net/projects/flashmon/>

Flashmon -2-

```

Applications Raccourcis Système
root@as3: /home/pierre/Bureau/code/trunk
Fichier Edition Affichage Terminal Aide

root@as3:/home/pierre/Bureau/code/trunk# insmod flashmon.ko
root@as3:/home/pierre/Bureau/code/trunk# dmesg | tail -n 13
[453500.943605] Flashmon : Flashmon module loaded
[453500.943608] Flashmon : Read Jprobe on : f857c750, handler addr : fcc
[453500.943609] Flashmon : Write Jprobe on : f857de60, handler addr : fcc
[453500.943610] Flashmon : Erase Jprobe on : f857ede0, handler addr : fcc
[453500.943611] Flashmon : Flash device :
[453500.943612] Flashmon :      Total size : 134217728 bytes
[453500.943613] Flashmon :      Blocks num : 8192 blocks
[453500.943614] Flashmon :      Pages num : 262144 pages
[453500.943615] Flashmon :      Block size : 16384 bytes
[453500.943616] Flashmon :      Page size : 512 pages
[453500.943617] Flashmon :      Pages per block : 32 pages
[453500.943681] Flashmon : No PID for userland notification
[453500.943687] Flashmon : /proc/flashmon created
root@as3:/home/pierre/Bureau/code/trunk# flash_erase /dev/mtd0 0 10
Erase Total 10 Units
Performing Flash Erase of length 16384 at offset 0x24000 done
root@as3:/home/pierre/Bureau/code/trunk# cat /proc/flashmon_log
1352898743.785077603;E;0
1352898743.785086888;E;1
1352898743.785090711;E;2
1352898743.785094288;E;3
1352898743.785097672;E;4
1352898743.785101035;E;5
1352898743.785104344;E;6
1352898743.785107864;E;7
1352898743.785111205;E;8
1352898743.785114523;E;9
root@as3:/home/pierre/Bureau/code/trunk#

Applications Raccourcis Système
root@as3: /home/pierre/Bureau/code/trunk
Fichier Edition Affichage Terminal Aide

root@as3:/home/pierre/Bureau/code/trunk# cat /proc/flashmon | head -n 15
0 0 1
0 0 1
0 0 1
0 0 1
0 0 1
0 0 1
0 0 1
0 0 1
0 0 1
0 0 1
0 0 1
0 0 0
0 0 0
0 0 0
0 0 0
root@as3:/home/pierre/Bureau/code/trunk#

```

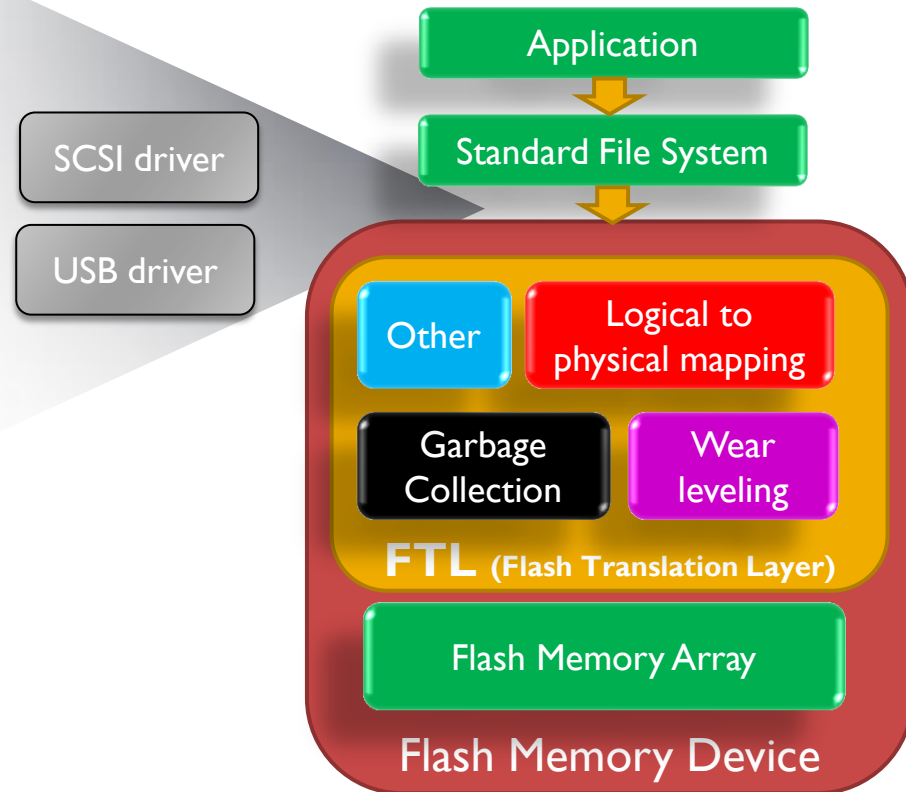
Presentation outline

1. Flash memory basics
2. Flash memory characteristics
3. Flash memory support
 1. Mapping schemes
 2. Wear leveling
 3. Garbage collection
 4. Flash specific cache systems
 5. Some contributions
4. Performance & energy considerations
5. **Flash memory interfacing**
6. Conclusions & perspectives

Flash based subsystems

► Flash based subsystems

- Flash memory +
- Controller (FTL, wear leveler) + (buffer) +
- Hardware interface, software API
 - Solid State Drive: ATA, SATA, PCIe
 - USB mass storage
 - ...



Memory Card & Drive

Card Type	Interface (command)
CompactFlash	N + PCMCIA, PC_Card (PATA)
MMC/SD	N + SPI (MMC)
XQD	N + PCIeexpress
USB flash drive	USB (SCSI)
UFS	UniPro (SCSI)
CFAST	N+ SATA (SCSI)

N=Native

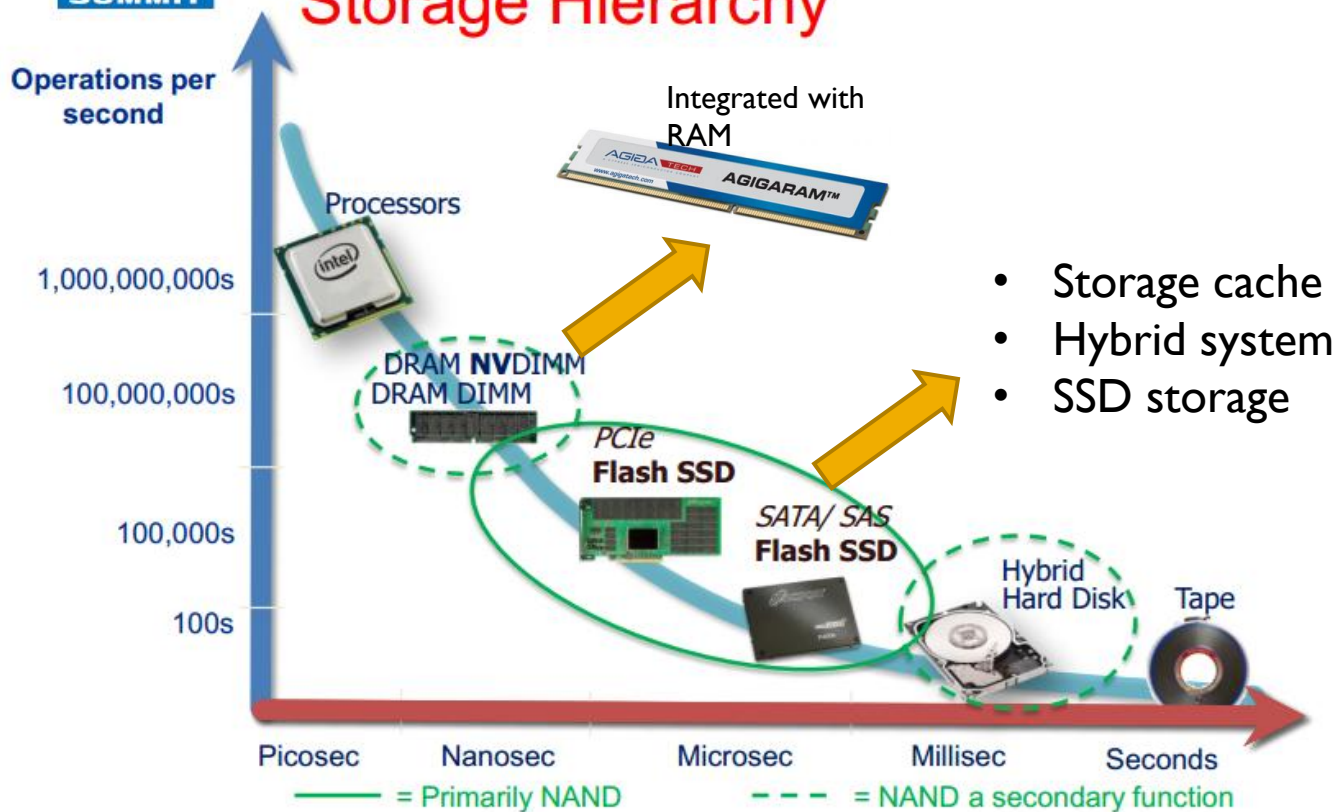
SSD Drives: SATA, SAS,
NVMeexpress,



Flash memory integration



NAND Uses in the Memory / Storage Hierarchy



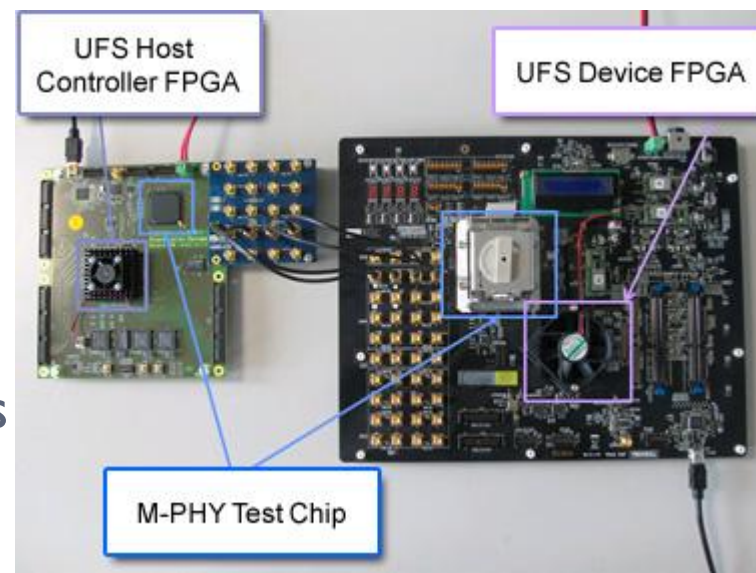
- Storage cache
- Hybrid system
- SSD storage

Flash Memory Summit 2012
Santa Clara, CA

Source: J. Cooke, Micron, Flash Summit 2012

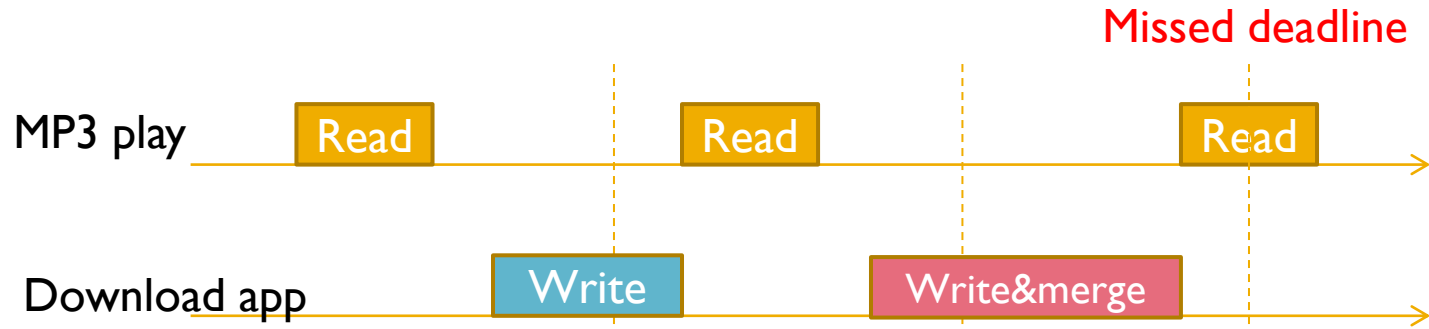
Universal Flash Storage (UFS)

- ▶ JEDEC standard VI.1, 2012
- ▶ Features:
 - ▶ Multiple command queues, multiple partitions parallel functions, boot partition
 - ▶ Max interface speed: 5.8 Gbps
 - ▶ Command set: SCSI+UFS specific
 - ▶ Compatible eMMC
- ▶ Serial interconnection (unipro), chain topology

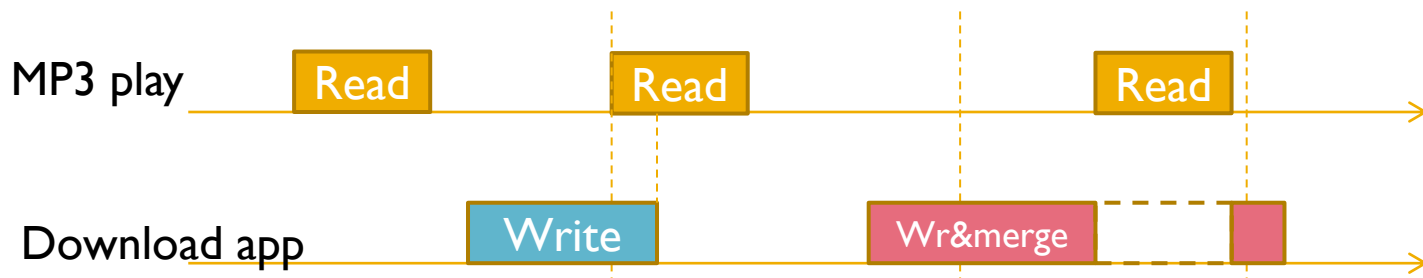


<http://www.toshiba-components.com/ufs/index.html>

Real-Time Constraints

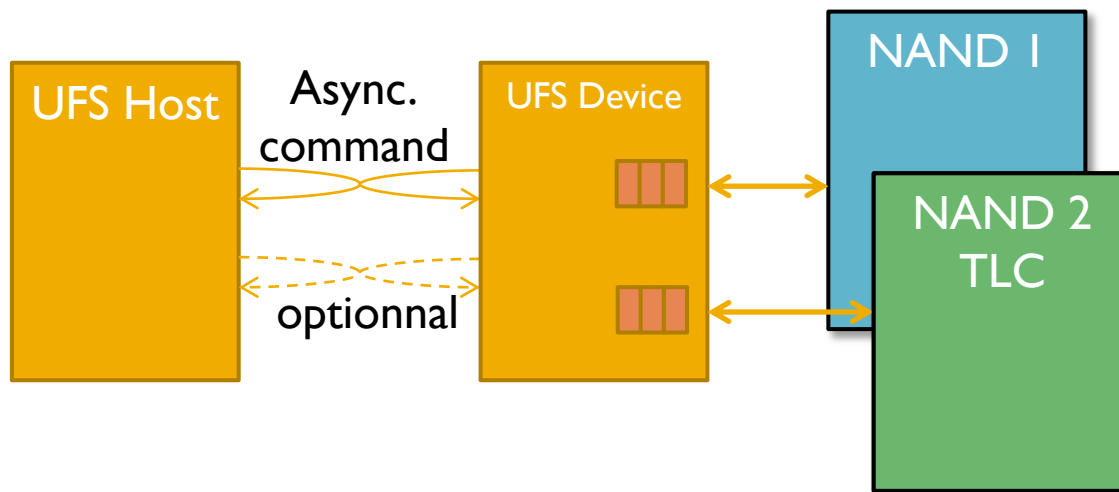


- ▶ Full-duplex host interface
- ▶ High Priority Interrupt



Multiple NAND channels & partitions

- ▶ Full utilization of interleaving across NAND channels
- ▶ Read-while-write (full-duplex) and dual write across multiple channels



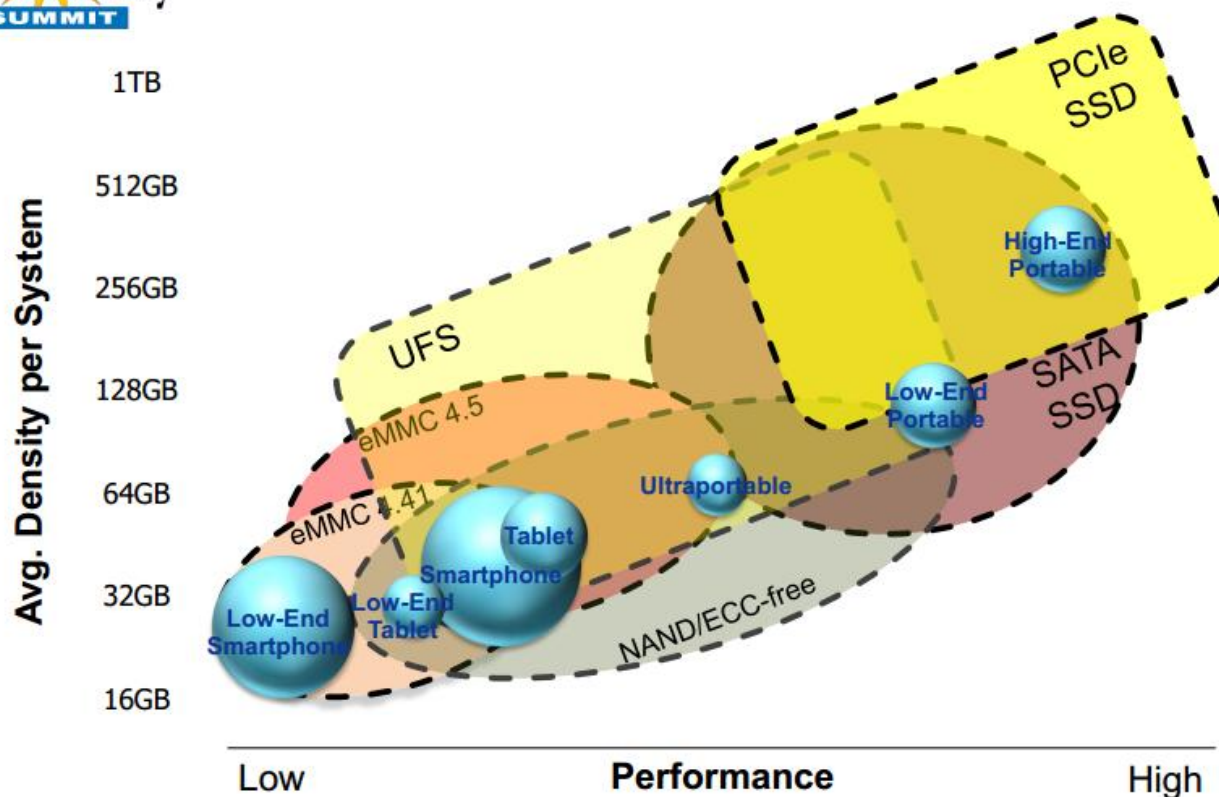
Multiple partitions

Technology mix:
Part 1: SLC
Part 2: TLC

Interface / product



Product Positioning for Client Markets



Source: Micron Marketing, Gartner, IDC, iSuppli, Forward Insights

Flash Memory Summit 2012
Santa Clara, CA

Size of application bubble = 2016 Market Size (end system units)

High/Low End Portable = Ultrathin/Notebook

Presentation outline

1. Flash memory basics
2. Flash memory characteristics
3. Flash memory support
 1. Mapping schemes
 2. Wear leveling
 3. Garbage collection
 4. Flash specific cache systems
 5. Some contributions
4. Performance & energy considerations
5. Flash memory interfacing
6. **Conclusions & perspectives**

Conclusions & perspectives

NV Memory Technology Comparison Ed Grochowski



	HDD	Flash (SLC)	Flash (MLC)	FeRAM	MRAM	PCM	ReRAM	STT-RAM	Race-Track	Molecular
Cell Structure	None	1T/0C	1T/0C	1T/1R	1T/1R	1T/1R	1T/1R	1T/1R	None	1T/1R
Cell size (F ²)	0.5	3.5	3.5	15	16=30	5-7	6-10	6-20	0.5e	10-5
Read time (ns)	2000	50	50	20-80	3-20	20-50	20-50	2-20	500	100e
Write / Erase time (ns)	1000	1 ms / 0.1 ms	1 ms / 0.1 ms	50/50	3-20	100	20-50	2-20	200	300e
Endurance	10 ¹⁶	10 ⁵	10 ³	10 ¹²	>10 ¹⁵	10 ¹²	10 ⁸	>10 ¹⁵	10 ¹⁶ e	10 ⁸ e
Write power	Low	Very high	Very high	Low	High	High	Low	Low	Low	Low
Max. Areal Density Gbits/in ²	750-1000	150	550	0.1	10	200e	200e	300e	>1000e	400e
Voltage required	3-5V	12 V	12 V	2-3 V	3 V	1.5-3 V	1.5-3 V	<1.5 V	3-5V	3-5V
Existing products						Prototype				22

Flash Memory Summit 2012
Santa Clara, CA



Just one of a team !

Source: E. Grochowski, "Future Technology Challenges For NAND Flash And HDD Products", FlashSummit 2012

Conclusions & perspectives -2-

- ▶ Integration of the non-volatile memory in operating system stack (not in the storage stack).
- ▶ Non volatile memories' coexistence
- ▶ Toward the predictability of access times in flash memory:
 - ▶ Real time systems
 - ▶ Data base cost models
- ▶ Leveraging data center's energy/performance bottleneck:
 - ▶ Storage represents 20% to 40% of the total energy consumption [Carter10]
 - ▶ EMC forecasted that the amount of digital information created annually will grow by a factor of 44 from 2009 to 2020 [Farmer10]
 - ▶ Microsoft, Google, and Yahoo are showing the way ...
 - ▶ → **Energy proportionality**
- ▶ Architectural design space exploration tools
- ▶ ...

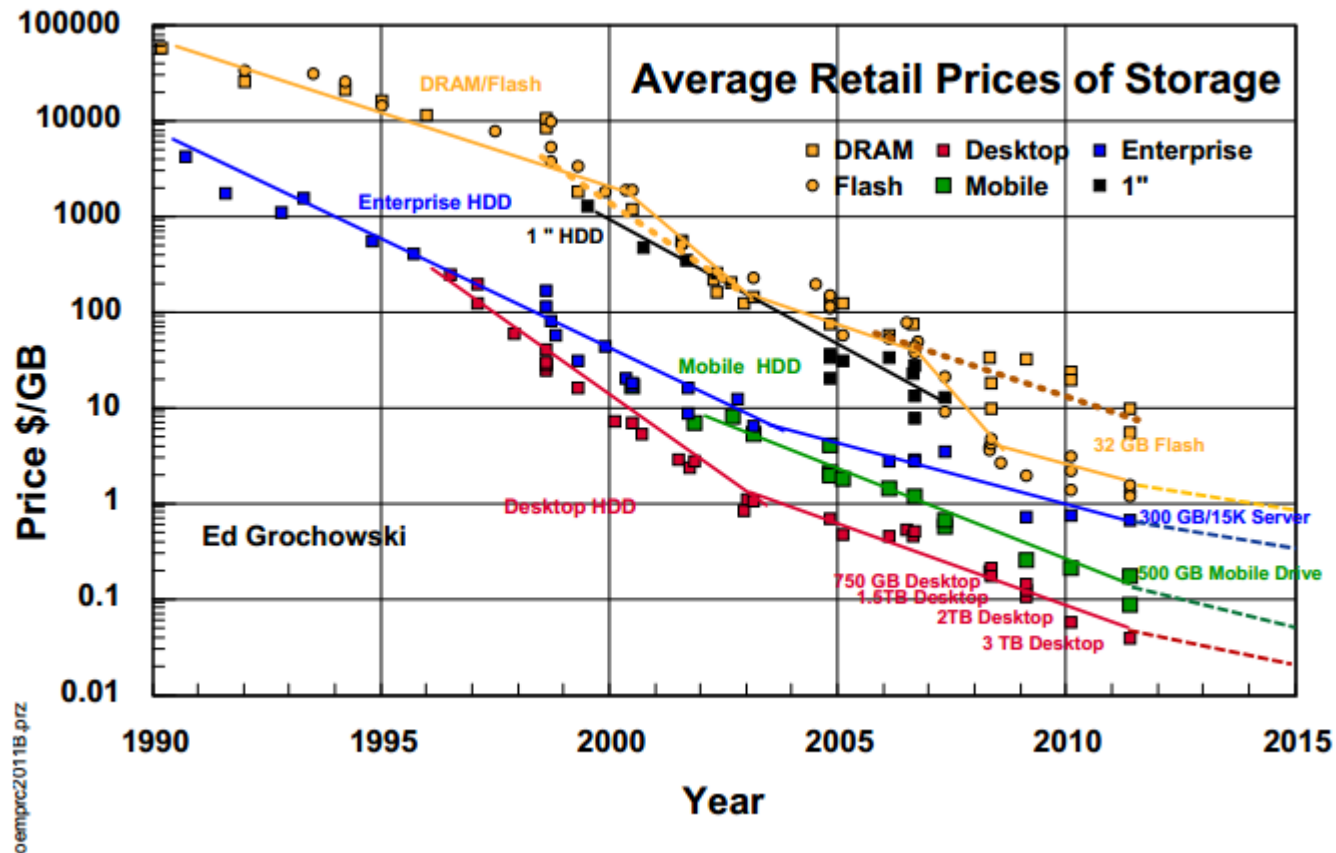
References

- [Shinohara99] Shinohara, T. (1999), Flash Memory Card with Block Memory Address Arrangement, United States Patent, No 5,905,993.
- [Ban99] Ban, A. (1999), Flash File System Optimized for Page-mode Flash Technologies, United States Patent, No 5,937,425.
- [RNFTL10] Wang, Y., Liu, D., Wang, M., Qin, Z., Shao, Z., & Guan, Y. (2010), RNFTL: a Reuse-aware NAND Flash Translation Layer for Flash Memory, *In Proceedings of the ACM SIGPLAN/SIGBED 2010 conference on Languages, compilers, and tools for embedded systems (LCTES)*.
- [BAST02] Kim, J. Kim, J. M., Noh, S. H., Min, S. L., & Cho, Y. (2002), A Space-Efficient Flash Translation Layer for Compact Flash Systems, *IEEE Transactions on Consumer Electronics*, 48(2), 366-375.
- [FAST07] Lee, S., Park, D., Chung, T., Lee, D., Park, S., & Song, H. (2007), A Log Buffer Based Flash Translation Layer Using Fully Associative Sector Translation, *ACM Transactions on Embedded Computing Systems*, 6(3), 1-27.
- [LAST08] Lee, S., Shin, D., Kim, Y., & Kim, J. (2008), LAST: Locality Aware Sector Translation for NAND Flash Memory Based Storage Systems, *ACM SIGOPS Operating Systems Review*, 42(6), 36-42.
- [EAST08] Kwon, S. J., & Chung, T. (2008), An Efficient and Advanced Space-management Technique for Flash Memory Using Reallocation Blocks, *IEEE Transactions on Consumer Electronics*, 54(2), 631-638.
- [KAST09] Cho, H., Shin, D., & Eom, Y. I. (2009), KAST: K-associative sector translation for NAND flash memory in real-time systems. *In Proceedings of Design, Automation, and Test in Europe (DATE)*, 507-512.
- [STAFF07] Chung, T. S., & Park, H. S. (2007), STAFF: a Flash Driver Algorithm Minimizing Block Erasures, *Journal of Systems Architectures*, 53(12), 889-901.
- [HFTL09] Lee, H., Yun, H., & Lee, D. (2009), HFTL: Hybrid Flash Translation Layer Based on Hot Data Identification for Flash Memory, *IEEE Transactions on Consumer Electronics*, 55(4), 2005-2011.
- [WAFTL11] Wei, Q., Gong, B., Pathak, S., Veeravalli, B., Zeng, L., & Okada, K. (2011), WAFTL: A Workload Adaptive Flash Translation Layer with Data Partition, *In Proceedings of 2011 IEEE 27th Symposium on Mass Storage Systems and Technologies (MSST)*.
- [DFTL09] Gupta, A., Kim, Y., & Urgaonkar, B. (2009), DFTL: a Flash Translation Layer Employing Demand-based Selective Caching of Page-level Address mappings, *In Proceedings of the 14th international conference on Architectural support for programming languages and operating systems (ASPLOS)*.
- [SFTL11] Jiang, S., Zhang, L., Yuan, X., Hu, H., & Chen, Y. (2011), SFTL: An Efficient Address Translation for Flash Memory by Exploiting Spatial Locality, *In Proceedings of 2011 IEEE 27th Symposium on Mass Storage Systems and Technologies (MSST)*.
- [Boukhobza13] Boukhobza, J. (2013), Flashing in the Cloud: Shedding some Light on NAND Flash Memory Storage System, chapter to appear in *Data Intensive Storage Services for Cloud Environment*, IGI Global.

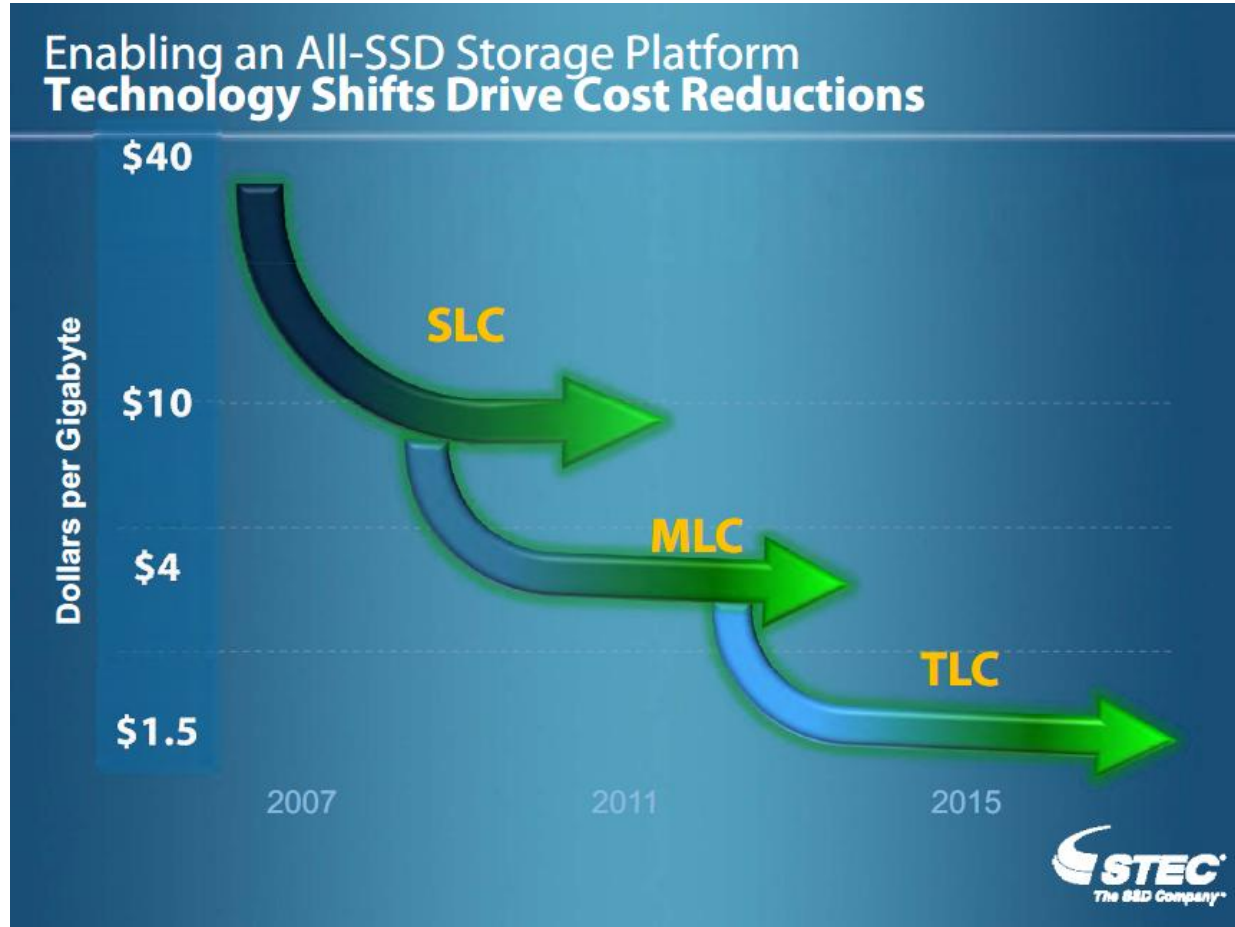
References -2-

- [DualPool95] Assar, M., Namazie, S., & Estakhri, P. (1995), Flash Memory Mass Storage Architecture Incorporation Wear Leveling Technique, United States Patent, No 5,479,638.
- [Achiwa99] Achiwa, K., Yamamoto, A., & Yamagata, O. (1999), Memory Systems Using a Flash Memory and Method for Controlling the Memory System, United States Patents, No 5,930,193
- [Chang07] Chang, L. (2007), On Efficient Wear Leveling for Large-scale Flash-Memory Storage Systems, *In Proceedings of the 2007 ACM Symposium on Applied Computing (SAC)*.
- [Kwon11] Kwon S. J., Ranjitkar, A., Ko, Y., & Chung, T. (2011), FTL Algorithms for NAND-type Flash Memories, *Design Automation for Embedded Systems*, 15(3-4), 191-224.
- [DAC08] Chiang, M., & Chang, R. C. (1999), Cleaning Policies in Mobile Computers Using Flash Memories, *Journal of Systems and Software*, 48(3), 213-231.
- [Kawaguchi95] Kawaguchi, A., Nishioka, S., & Motoda, H. (1995), A Flash Memory Based File System, *In Proceedings of the USENIX 1995 Annual Technical Conference (ATC)*.
- [CAT99] Chiang, M., & Chang, R. C. (1999), Cleaning Policies in Mobile Computers Using Flash Memories, *Journal of Systems and Software*, 48(3), 213-231.
- [CFLRU06] Park, S., Jung, D., Kang, J., Kim, J., & Lee, J. (2006), CFLRU: a Replacement Algorithm for Flash Memory, *In Proceedings of the 2006 International conference on Compilers, architecture and synthesis for embedded systems (CASES)*.
- [FAB06] Jo, H., Kang, J., Park, S., Kim, J., & Lee, J. (2006), FAB: a Flash-aware Buffer Management Policy for Portable Media Players, *IEEE Transactions on Consumer Electronics*, 52(2), 485-493.
- [BPLRU08] Kim, H., & Ahn, S. (2008), BPLRU: A Buffer Management Scheme for Improving Random Writes in Flash Storage, *In Proceedings of the 6th USENIX Conference on File and Storage Technologies (FAST)*.
- [C-lash11] Boukhobza, J., Olivier, P., & Rubini, S. (2011), A Cache Management Strategy To Replace Wear Leveling Techniques for Embedded Flash Memory, *2011 International Symposium on Performance Evaluation of Computer & Telecommunication Systems (SPECTS)*.
- [CACH-FTL13] Boukhobza, J., Olivier, P., & Rubini, S., A Cache-Aware Configurable Hybrid Flash Translation Layer , To appear in the 21st Euromicro International Conference on Parallel, Distributed, and Network-Based Processing (PDP).
- [Park11] Park, S., Kim, Y., Urgaonkar, B., Lee, J., & Seo, E. (2011), A Comprehensive Study on Energy Efficiency of Flash Memory Storages, *Journal of Systems Architecture (JSA)*, 57(4), 354-365.
- [Agrawal08] Agrawal, N., Prabhakaran, V., Wobber, T., Davis, J. D., Manasse, M., & Panigrahy, R. (2008), Design Tradeoffs for SSD Performance, *In Proceedings of the USENIX Annual Technical Conference (ATC)*.
- [Flashmon11] B., Khetib, I., and Olivier, P., (2011). Flashmon : un outil de trace pour les accès à la mémoire flash NAND, in *Proceedings of the Embed With linux Workshop, France, 2011*,

Price of storage

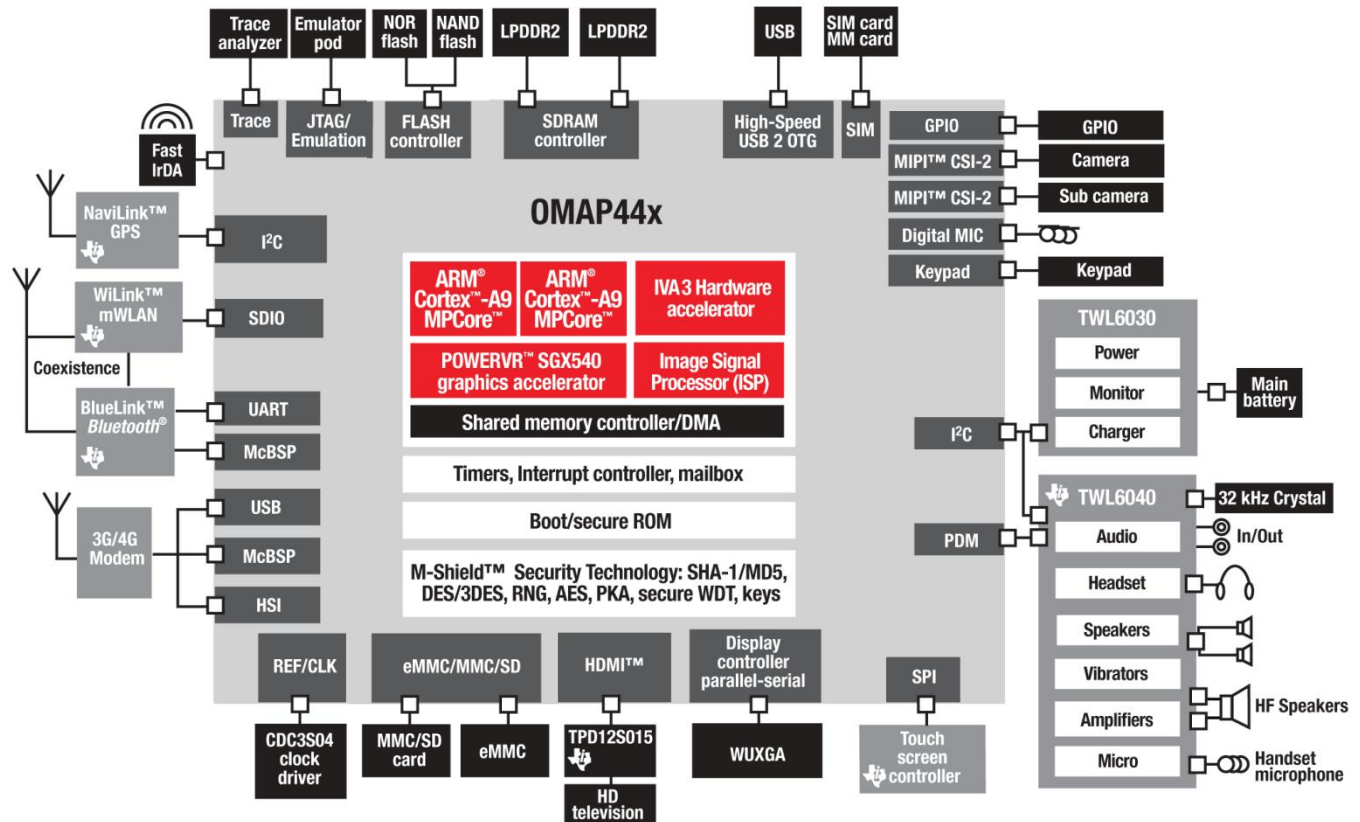


Price of NAND storage

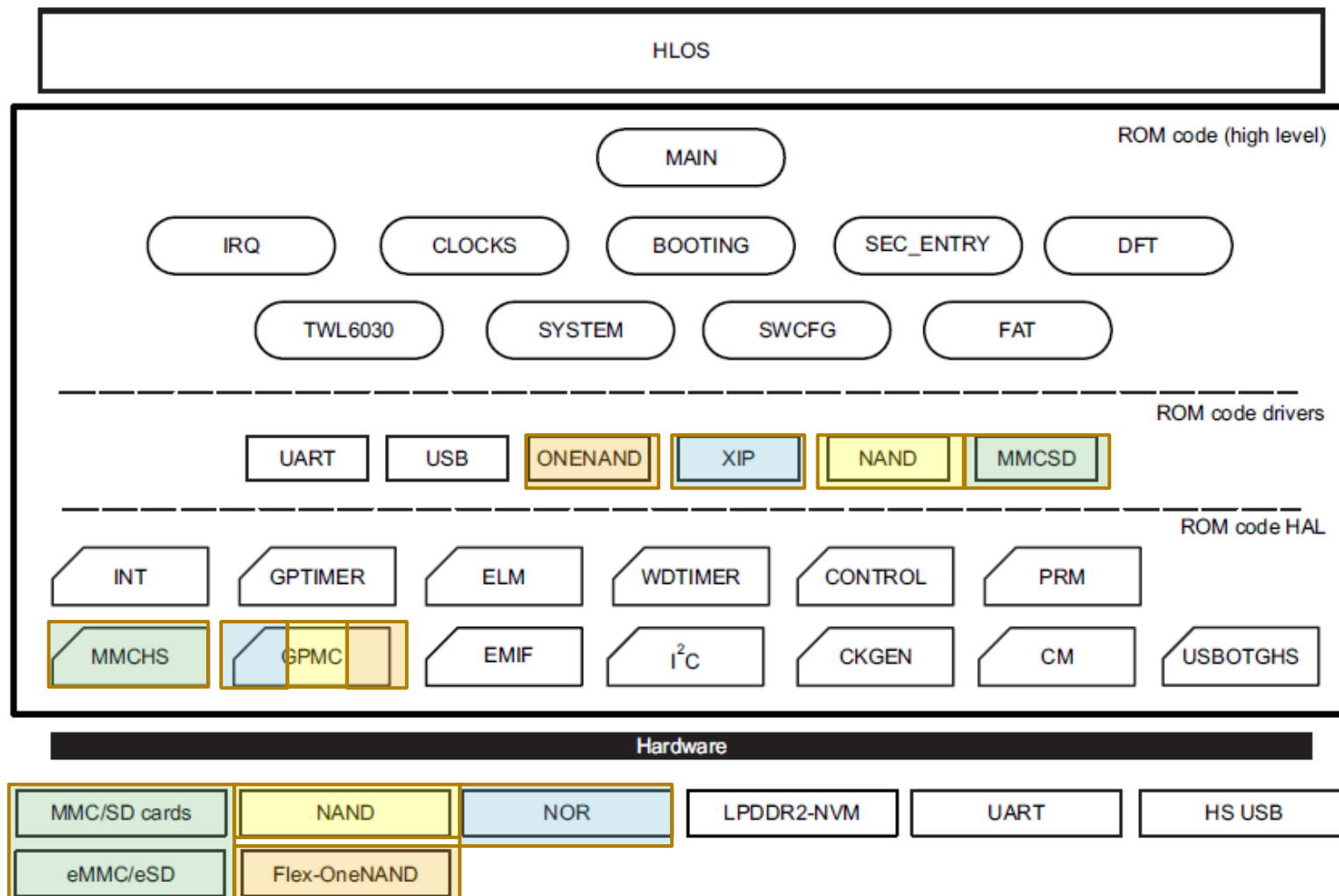


SoC TI OMAP4 Mobil Application Platform

- Development of planned features for the Smartphones and Mobile Internet Devices



OMAP Boot System



Flash in the boot process

▶ Flash NOR

- ▶ boot code, executable, configuration data
- ▶ XIP : eXecute In Place capability

▶ Flash NAND:

- ▶ eMMC, USB mass storage
- ▶ Copy the boot code into RAM before being executed

▶ Pre-Flashing: the code provided by a peripheral is automatically stored into a Flash for the next boots.

NAND Flash with NOR interface

- ▶ Samsung OneNAND
- ▶ Simplest interface (NOR NAND), XIP, prefetch

