FLARES: An Aging-Aware Algorithm to Autonomously Adapt the Error Correction Capability in NAND Flash Memories

Davide Bertozzi, Cristian Zambelli, Piero Olivo

(University of Ferrara, Italy)

Stefano di Carlo, Salvatore Galfano, Marco Indaco,

Paolo Prinetto

(Politecnico di Torino, Italy)

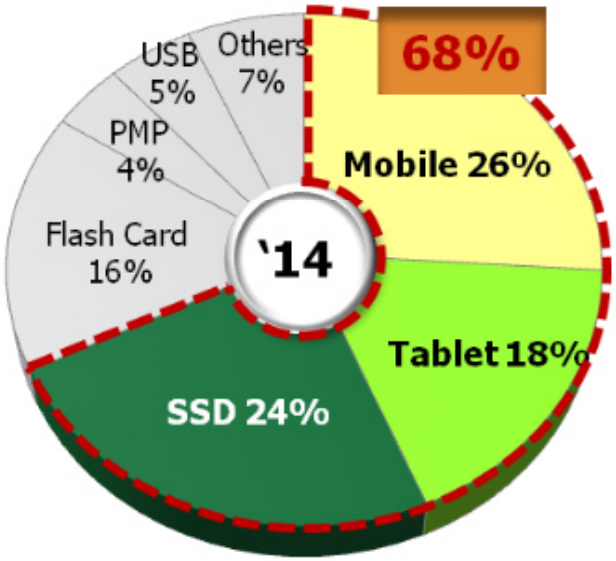
università di ferrara

DA SEICENTO ANNI GUARDIAMO AVANTI.

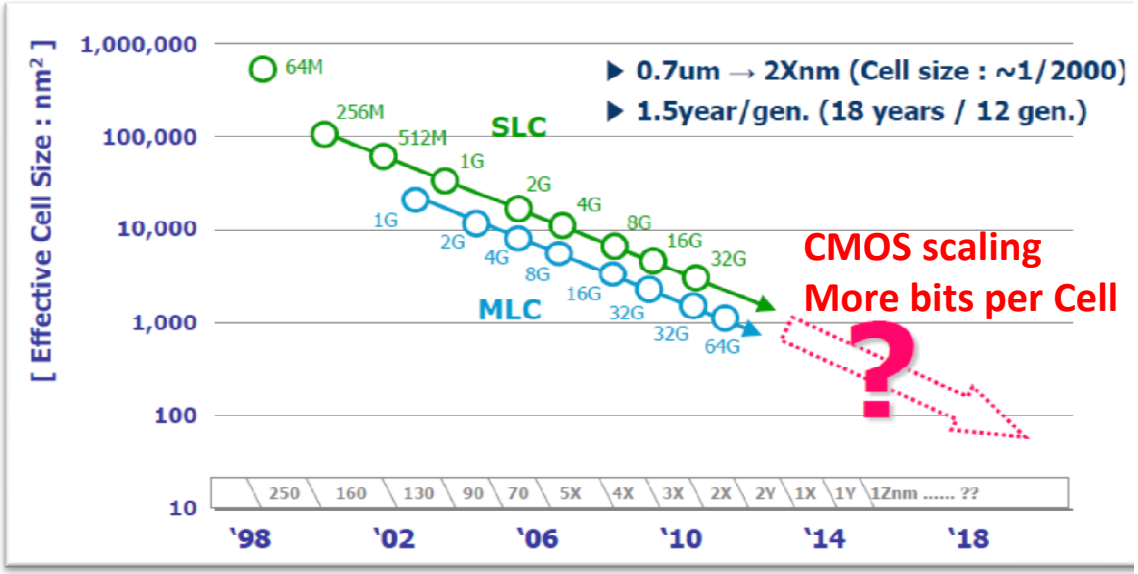
Clereco

High-Performance Data Storage

- Driven by the demand for high-performance data storage, NAND Flash memory has become one of the fastest-growing segments in the global semiconductor industry



Source: Hynix Marketing 2011



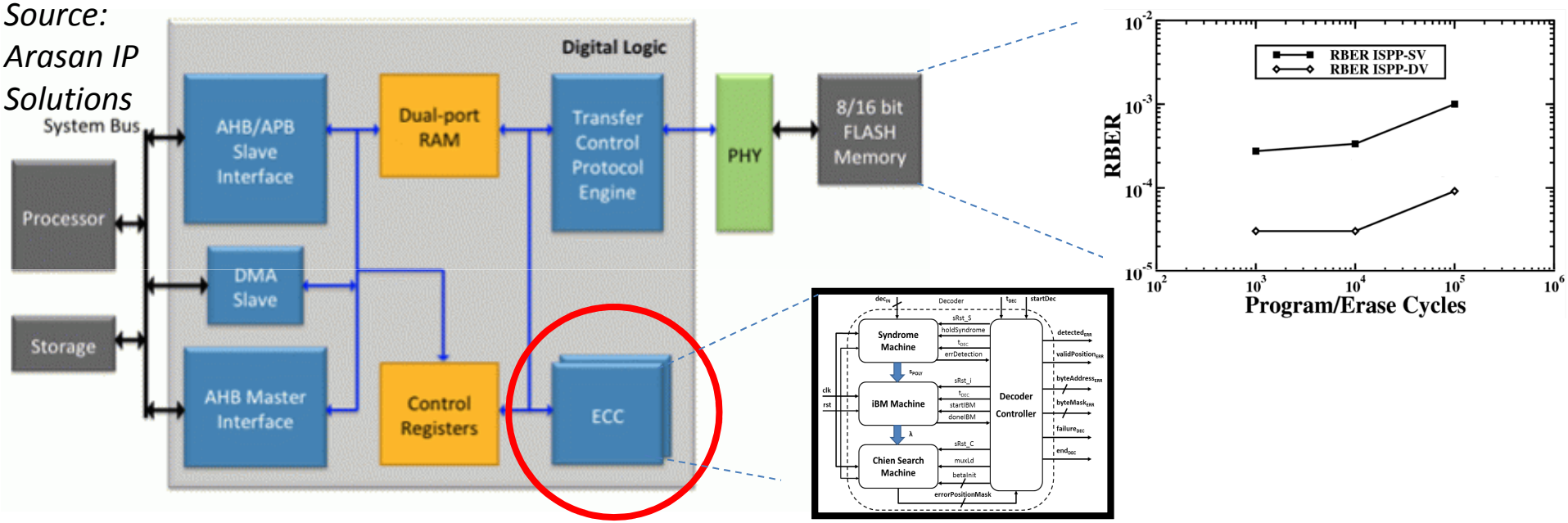
Seaung Suk Lee, "Emerging Challenges in NAND Flash Technology", Flash Summit 2011 (Hynix)

System management algorithms face serious issues to maintain product reliability under endurance and performance constraints.

Ecc correction capability

- An ECC can correct data corrupted by aging or read/program disturbs (e.g., RS, BCH, LDPC,..)

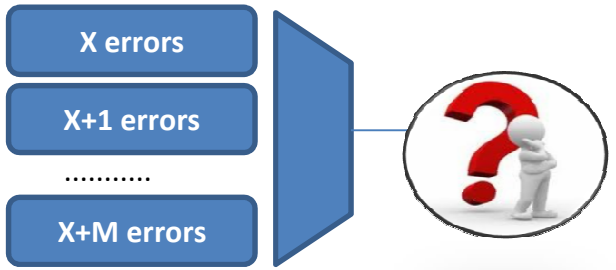
Source: Arasan IP Solutions



- The reliability of a NAND Flash is not constant
  - It decreases over time due to aging effects
  - ECCs with programmable correction capabilities are widely implemented to adapt the controller to the error rate changes over P/E cycles

Runtime Adaptation

A strategy is needed to decide which correction capability to use at runtime



FLARES

Algorithm capable of predicting at runtime the optimal correction capability for each page of a NAND Flash

NAND Flash BER estimation model

Real measurements of errors

PREDICTION

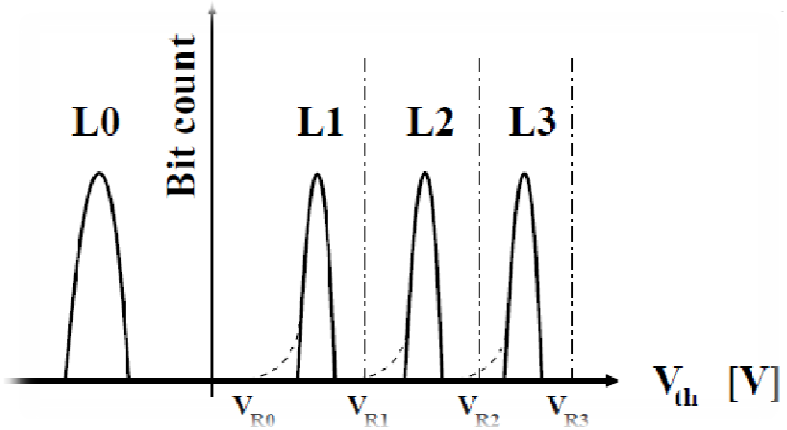
CORRECTION STRENGTH TUNED TO MEET TARGET UBER WHILE MAXIMIZING PERFORMANCE

- **Generic enough to work with several types of ECC, provided the programming and monitoring feature are provided**
- **Validated at first by MATLAB and then in a real environment by instrumenting the YAFFS2 file system**

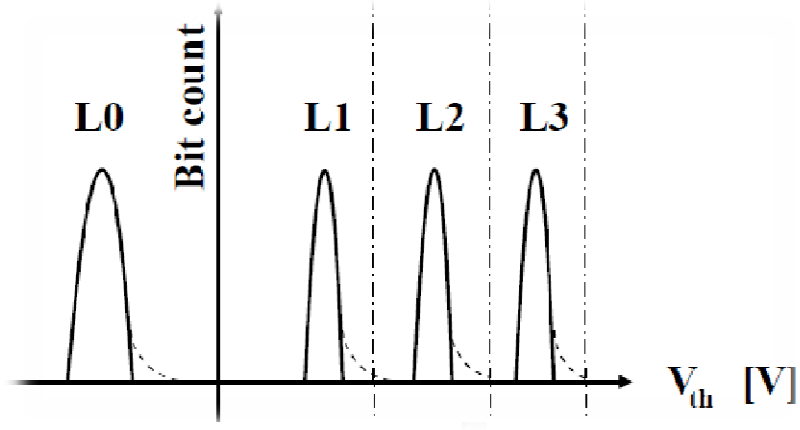
BER Modelling

- We focus on two dominant types of MLC NAND Flash errors

Retention Errors



Program Errors



UBER model is a reliability metric used to specify the data corruption rate accepted in the target application (e.g., 10^-11 or 10^-16)

- ✓ UBER is the probability of having E>p errors in the page divided by num. bits
- ✓ Program and retention errors have been proven to be topologically non-correlated

$$UBER = \frac{1}{n} \sum_{i=p+1}^{n} \binom{n}{i} \cdot RBER^{i} \cdot (1 - RBER)^{n-i}$$

Modeling RBER is mandatory for correction capability prediction

BER Modelling (2)

- We rely on empirical and statistical characterization efforts of real devices (*Mielke et al. 2008, Micheloni et al. 2010,..*)
- *Sun et al. 2011* quantified RBER as a function of the program/erase (PE) cycle for a variety of vendors and technologies
  - Empirical data then curve-fitted by an exponential growth model

$$RBER(PE) = [A \cdot e^{B \cdot PE} + C]$$

- **Drawbacks**

- **RBER under/overestimated for low PE values**
- **Does not include any time-related term**

BER Modelling (3)

- A model extension has been proposed by JEDEC.
- A power-law model is used to describe error rate as a function of elapsed time AND program/erase cycles:

$$RBER(t_{ret}, PE) = RBERwr + B_0(PE^n \cdot t_{ret})^m$$

- **RBER increases as retention time increases**
- **This increment increases with larger program/erase cycles.**
- **Slightly overestimates the retention RBER in fresh devices**

- **RBERwr is error rate at *tret=0* (i.e., program errors only).**
  - **We can use the binomial distribution**

BER Modelling (4)

- Final analytical model
- structured into two contributions
  - RBERwr
  - RBERrd

$$RBER(t_{ret}, PE) = \underbrace{[A \cdot e^{B \cdot PE} + C]}_{\textbf{RBERwr}} + \underbrace{B_0 (PE^n \cdot t_{ret})^m}_{\textbf{RBERrd}}$$

Runtime RBER Estimation

- To perform RBER runtime estimation, the following **page profile (PPROF)** must be **stored and constantly updated**:
  - **Actual correction capability used to encode page content;**
  - **Correction capability to use at the next page programming to meet the target UBER;**
  - **PE cycles;**
  - **The timestamp of the last program operation;**
  - **The total number of errors detected when reading the page over a *window* of operations;**
  - **The total number of ECC decoding failures detected over a *window* of operations.**

PPROF data are enough to feed

the derived analytical model.

Dynamic effects

What if runtime and environmental operating conditions (e.g., unforeseen stress conditions) cause error rate variations with respect to the UBER model?

- FLARE **dynamically** adapts error correction capability by complementing:
  - **theoretical RBER estimation;**
  - **runtime RBER estimation obtained by the ECC subsystem.**

FLARES Algorithm

1. Retention time assessment

ret_time=current_time – PPROF.writestamp

If ret_time > search (max_ret_time (Pcur, PPROF.pecycles))
then
alarm(“page rewrite required”)

FLARES Algorithm

2. Assess contribution of PE cycles to page reliability

Empirical RBER as a function of PE cycling

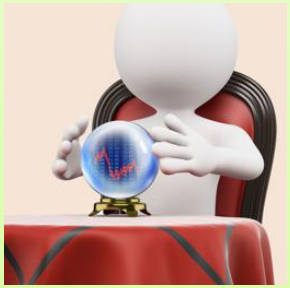
meas_rber'=(PPROF.errc/PAGESIZE)/WSIZE

meas_rber = meas_rber'-RBERrd(PPROF-pecycles, ret_time)

Combine empirical and theoretical RBER into an average RBER

model_rber=RBERwr(PPROF.pecycles)

avg_rber=MIX*meas_rber+(1-MIX)*model_rber



Project RBER to account for retention time requirements

Proj_rber=avg_rber+RBERrd(PPROF.pecycles, REQ_RET_TIME)

(min_rber …. max_rber, p) = search (corr_table, proj_rber)

The correction capability **p** *for next program operation is derived*

FLARES Algorithm

3. Course of action

- **Fast Zone:** a correction capability higher than the current correction is required;

Next correction capability updated to the computed value right away

- **Overcorrection Zone:** a correction capability lower than the current correction is required;
- **Critical Zone:** the same correction capability is required but it approaches the correction limit of the current code;
- **Failure Zone:** detected failures require higher correction capability;
- **Safe Zone:** the current correction capability is proper and no action must be taken.

FLARES Algorithm

3. Course of action

- **Fast Zone:** a correction capability higher than the current correction is required;
- **Overcorrection Zone:** a correction capability lower than the current correction is required;
- **Critical Zone:** the same correction capability is required but it approaches the correction limit of the current code;
- **Failure Zone:** detected failures require higher correction capability;
- **Safe Zone:** the current correction capability is proper and no action must be taken.

Count occurrence of this condition, and lower correction strength once threshold is exceeded

FLARES Algorithm

3. Course of action

- **Fast Zone:** a correction capability higher than the current correction is required;
- **Overcorrection Zone:** a correction capability lower than the current correction is required;
- **Critical Zone:** the same correction capability is required but it approaches the correction limit of the current code;

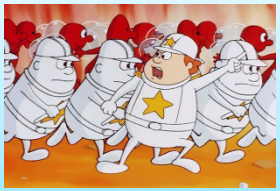
Count occurrence of this condition, and increase next correction strength by 1 only when threshold exceeded

- **Failure Zone:** detected failures require higher correction capability;
- **Safe Zone:** the current correction capability is proper and no action must be taken.

FLARES Algorithm

3. Course of action

- **Fast Zone:** a correction capability higher than the current correction is required;
- **Overcorrection Zone:** a correction capability lower than the current correction is required;
- **Critical Zone:** the same correction capability is required but it approaches the correction limit of the current code;
- **Failure Zone:** detected failures require higher correction capability;



Increase correction capability right away to cope with the unforeseen situation

- **Safe Zone:** the current correction capability is proper and no action must be taken.

FLARES Algorithm

3. Course of action

- **Fast Zone:** a correction capability higher than the current correction is required;
- **Overcorrection Zone:** a correction capability lower than the current correction is required;
- **Critical Zone:** the same correction capability is required but it approaches the correction limit of the current code;
- **Failure Zone:** detected failures require higher correction capability;
- **Safe Zone:** the current correction capability is proper and no action must be taken.

Nothing to do

FLARES Algorithm

3. Course of action

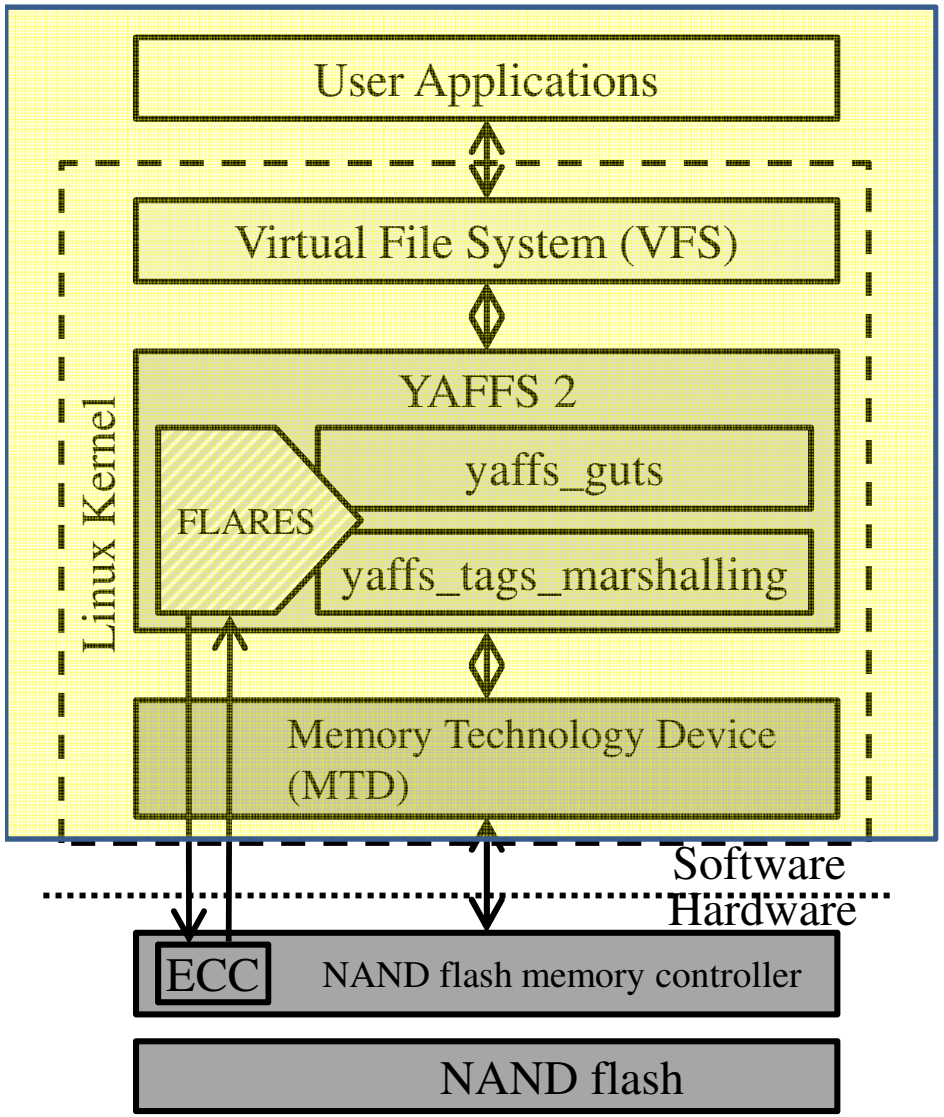
- **Fast Zone:** a correction capability higher than the current correction is required;
- **Overcorrection Zone:** a correction capability lower than the current correction is required;
- **Critical Zone:** the same correction capability is required but it approaches the correction limit of the current code;
- **Failure Zone:** detected failures require higher correction capability;
- **Safe Zone:** the current correction capability is proper and no action must be taken.

Should be scheduled at every page access, but also periodically to check retention errors in pages not accessed for a long time

PPROF information must be stored in a reliable way but without impacting flash endurance and performance

IMPLEMENTATION

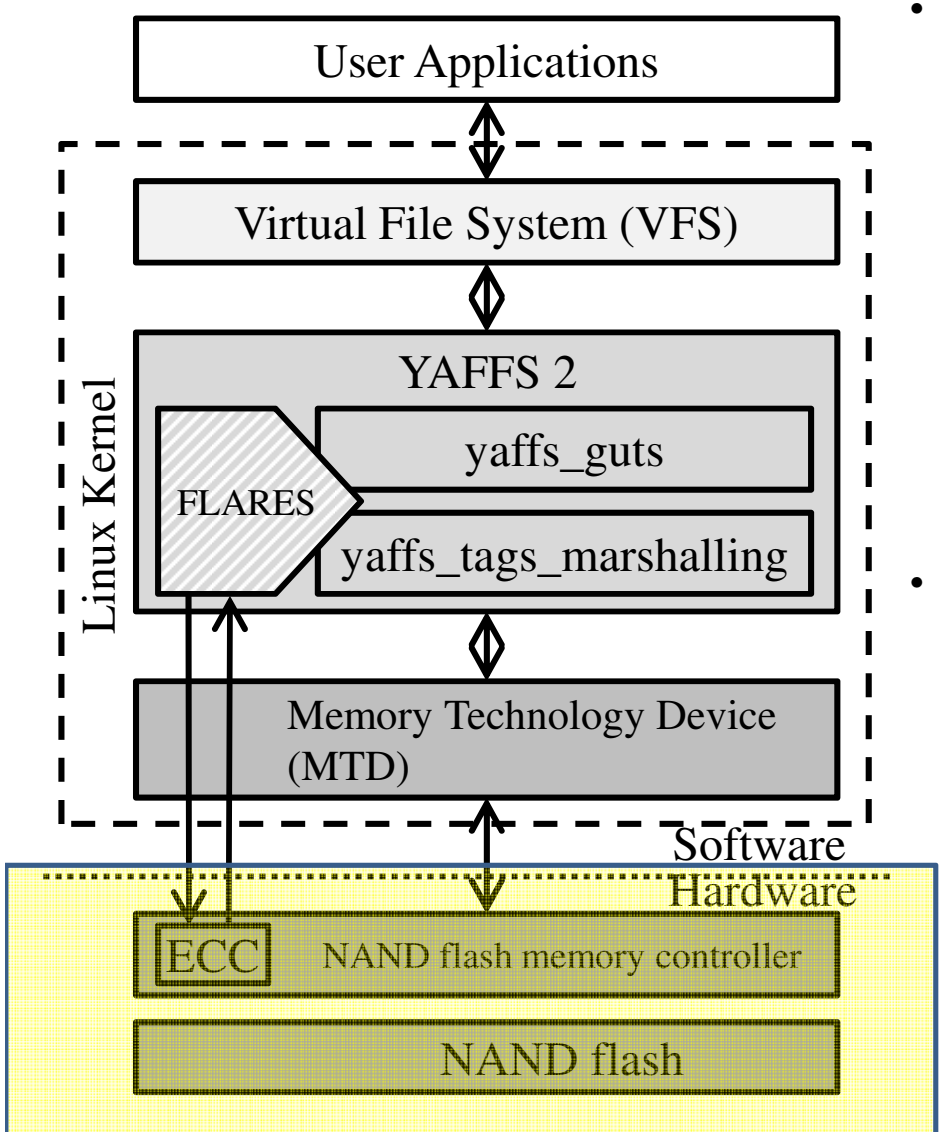
We implemented FLARES as part of the YAFFS2 file system



- **Applications run on top of the VFS that makes them independent of the specific file system (FLARES-agnostic)**
- **FLARES is implemented as an additional YAFFS 2 module, at the tags' marshalling level**
- **The Linux Memory Technology Device serves as a driver interfacing the file system with the NAND flash controller**

IMPLEMENTATION (2)

We implemented FLARES as part of the YAFFS2 file system



- **The full HW layer (NAND Flash memory + controller + ECC engine) has been EMULATED at MTD level.**
  - *2-bit MLC NAND Memory (Cai et al., 2012), 3x nm manufacturing process, 4096 blocks of 128 pages*

Page write time (AVG) @ cycle 1	800us
Page read time	75us
Write operation power consumptions	0.164 W
Read operation power consumptions	40 mW
Maximum considered P/E cycles	10,000
Page Size	4 kB + 224B

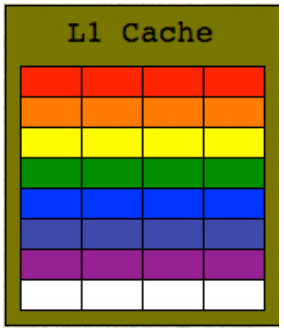
- **Emulated ECC architecture from Zambelli et al. 2012.**
  - *Fast parallel BCH coding engine with tunable correction capability*
  - *ECC latency model from RTL simulation runs Latency = f(correction capability, number of errors in page, error location)*
  - *From 83.9 us (1 error) to 194 us (50 errors)*
  - *ECC power model from post-P&R netlist*
- **Parametric per-page random error injection function modelled in the MTD**

PPROF INFORMATION CACHING

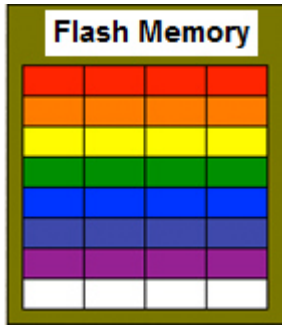
One of the most critical activities performed by FLARES is maintaining updated per-page PPROF information

Similarly, YAFFS2 needs to manage per-page tag information to build the file system. Same policy is used.

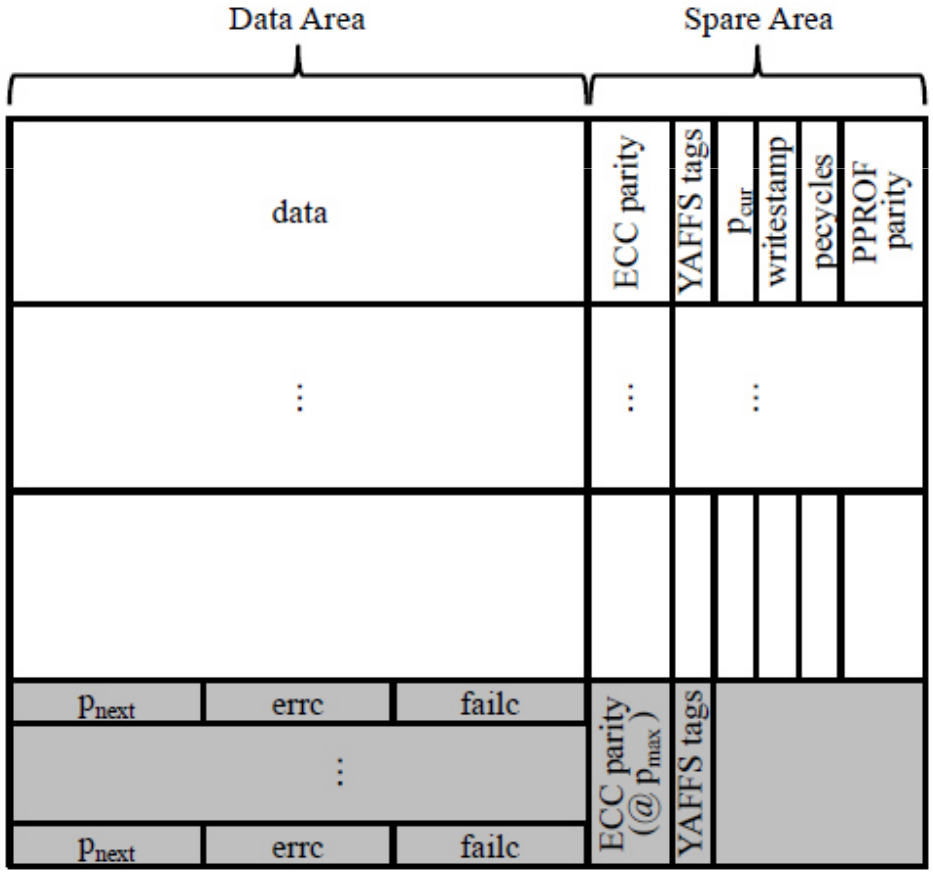
- Each PPROF must be stored in flash to store the information across system reboots
- It must also be cached in RAM to enable fast access and to minimize page programming that may impact the flash endurance.



The coherence of the two copies must always be guaranteed.



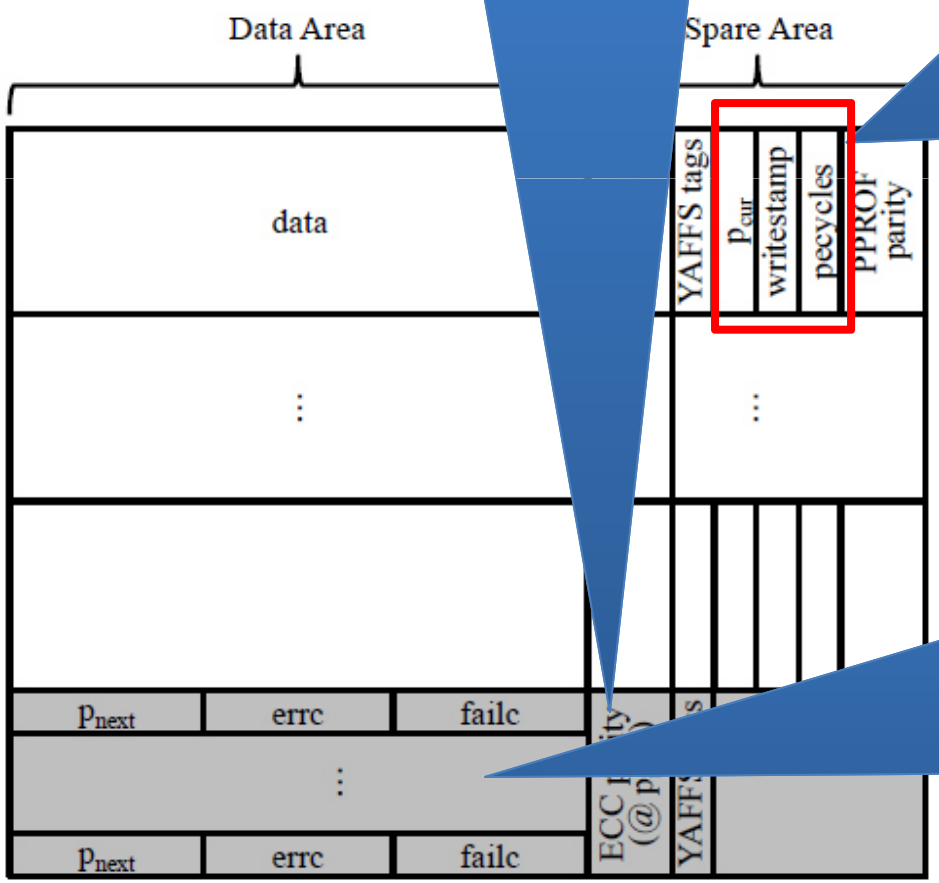
PPROF DATA ORGANIZATION



PPROF DATA ORGANIZATION

- Worst-Case ECC protection
- Rarely accessed pages

- What if flush fails at FS unmount? Lost information recomputed by FLARES at runtime



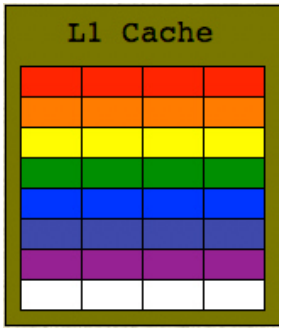
- Essential to decode the page and to keep track of the page wearout
- Updated only when a page is programmed
- Can be saved in the page spare area every time a page is written
- They are protected by a dedicated ECC designed for worst case scenario
- Decoded only once when PPROF is cached

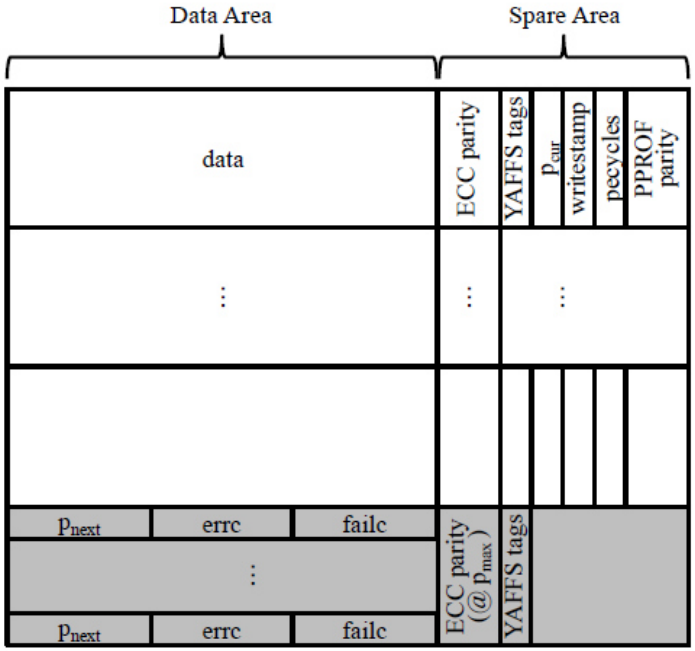
- The other PPROF fields must be continuously updated at runtime
- Cache-flash misalignment unavoidable
- To avoid memory wearout this information is cached, updated in RAM and flushed when the disk is unmounted
- Data are saved in dedicated pages similarly to the YAFFS2 file system info

PPROF CACHING POLICY

The PPROF cache policy is important to guarantee high system performance.

Full PPROFs data caching at FS mounting is impractical!

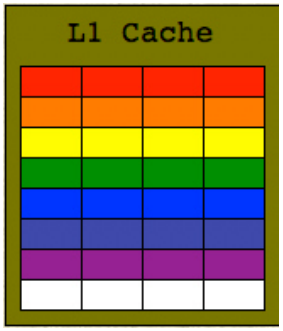




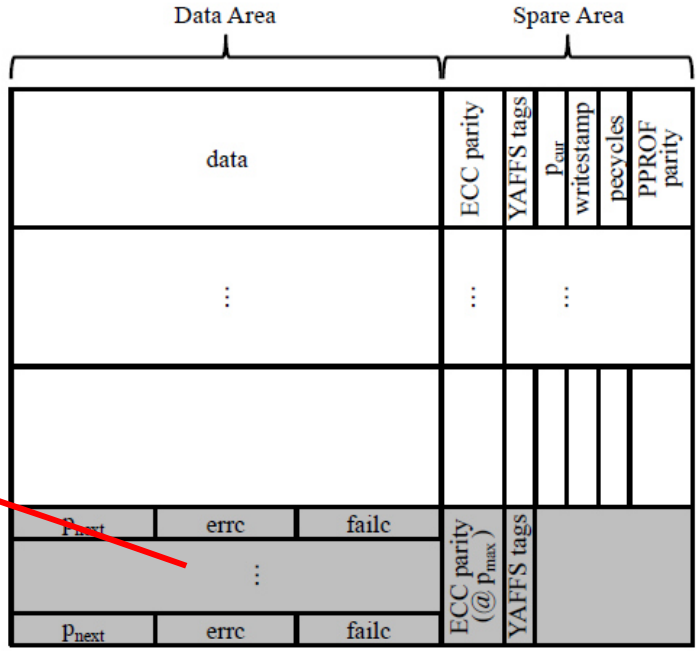
PPROF CACHING POLICY

The PPROF cache policy is important to guarantee high system performance.

Full PPROFs data caching at FS mounting is impractical!



reserved pages cached when the file system is mounted (only a few pages)



PPROF CACHING POLICY

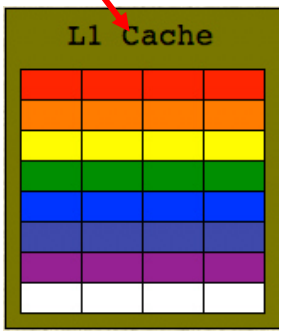
The PPROF cache policy is important to guarantee high system performance.

Full PPROFs data caching at FS mounting is impractical!

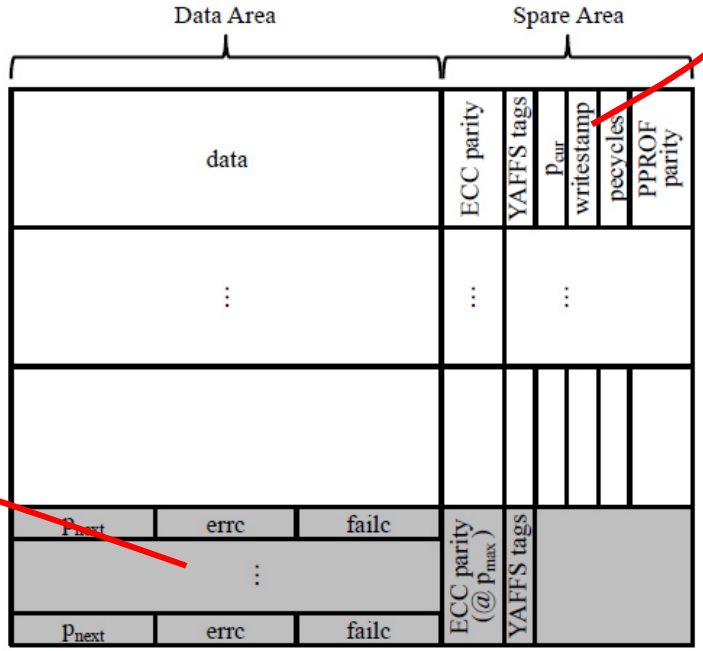
PPROF section in spare area cached on-demand

When a page is accessed for the first time after FS mounting:

- **Read operation**. *Both the page content and the spare area where the PPROF is stored are read from the flash.*
- **Program operation.** *An additional read operation is performed before the write operation to access the page spare area and read the page PPROF*



reserved pages cached when the file system is mounted (only a few pages)



PPROF MEMORY OVERHEAD

Quantity	Occupation
Pcur	6 bits
Pnext	6 bits
Pecycles	14 bits
Writestamp	32 bits
errc	13 bits
failc	2 bits
PPROF parity	35

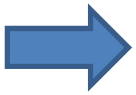
Main BCH ECC needs from 5B to 81B for parity

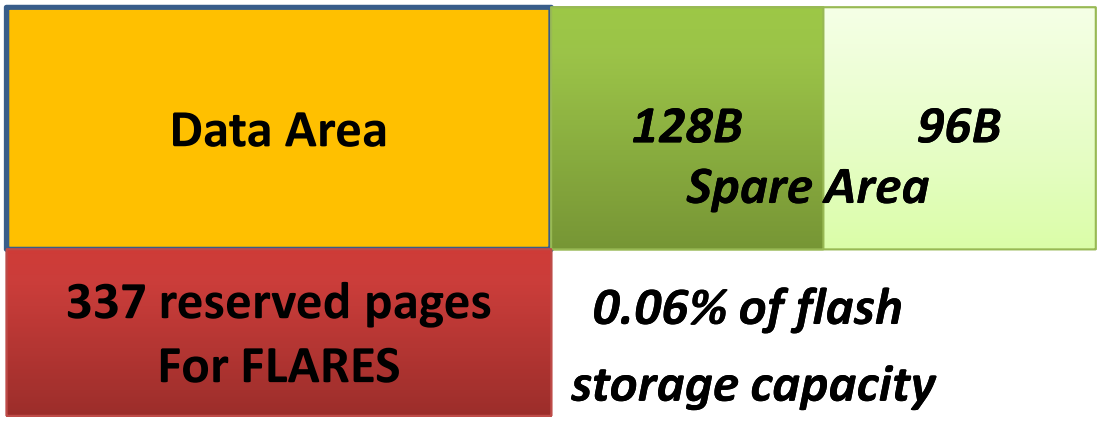
Main ECC parity	**81B**
Pcur	**6 bits**
Pecycles	**14 bits**
Writestamp	**32 bits**
PPROF parity	**4B**
YAFFS tags	**36B**

Left for system-level management policies

errc	**13 bits**
failc	**2 bits**
Pnext	**6 bits**

For each page



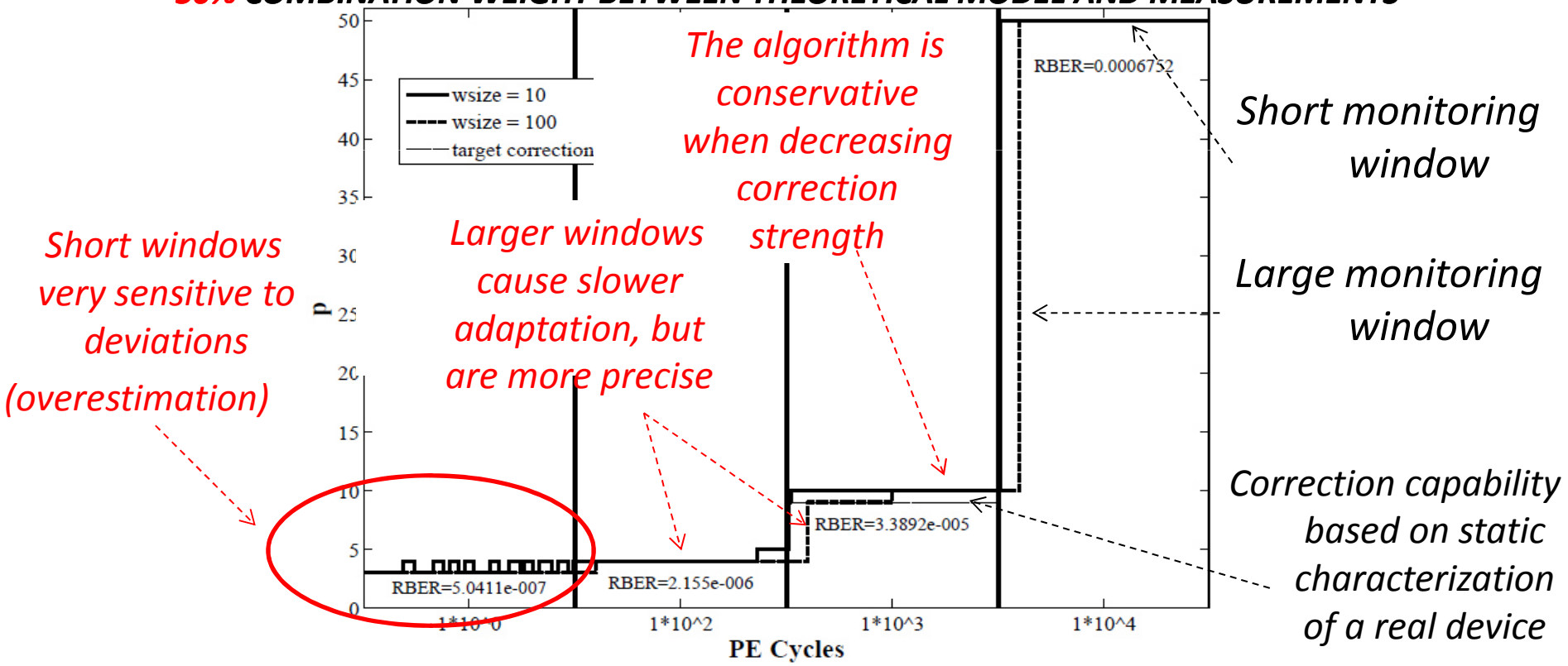


0.06% of flash storage capacity

FLARES Validation

- 4 operating points of a page simulated (different PE cycling)
- Around each operating point, 1000 read operations performed
- At each operation, errors injected based on RBER from a real device = f(PE cycle, retention time) + **gaussian variability**

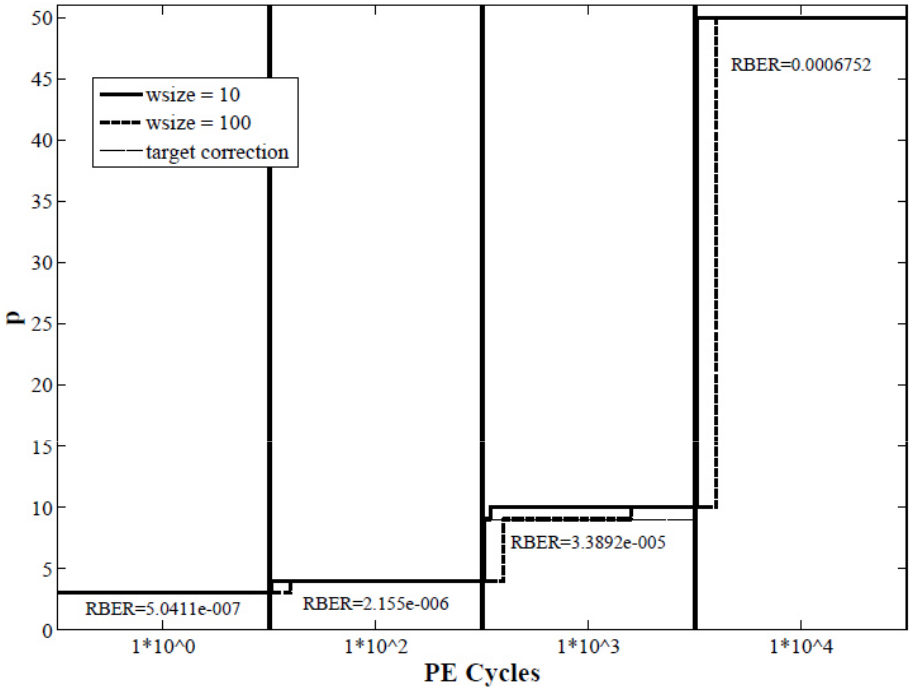
50% COMBINATION WEIGHT BETWEEN THEORETICAL MODEL AND MEASUREMENTS



Window size spans the response time vs. adaptation precision trade-off

FLARES Validation (2)

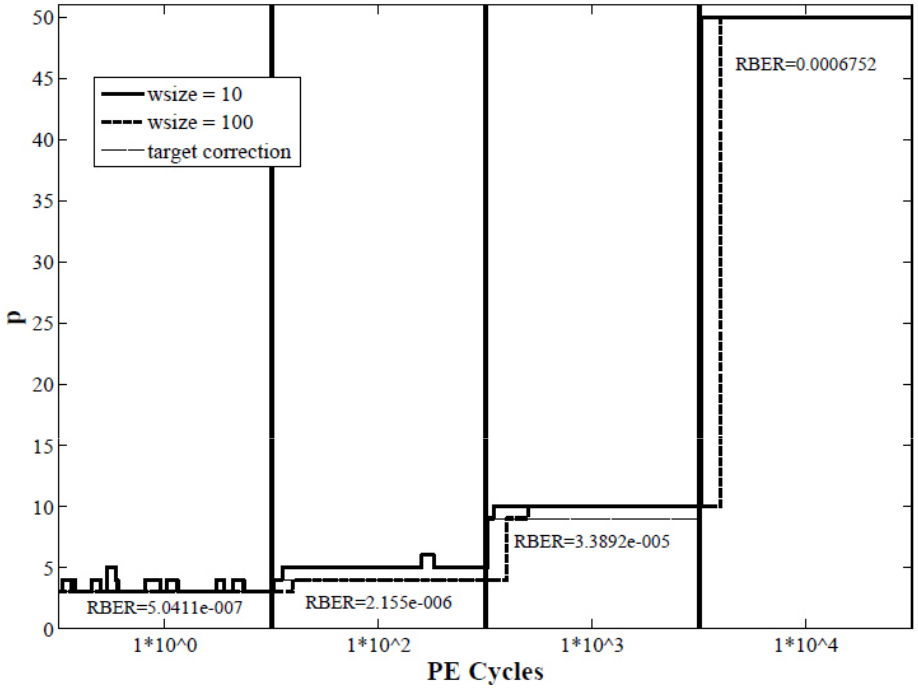
THEORETICAL MODEL-BASED PREDICTION



FLARES CORRECTLY PREDICTS THE ERROR CORRECTION CAPABILITY OF THE FLASH

BUT LOOSES REACTION CAPABILITY TO RUNTIME RBER CHANGES

MEASUREMENTS-BASED PREDICTION



FLARES TOO SENSITIVE TO LOCAL CHANGES IN MEASURED RBER

THUS LIKELY INCURRING PERFORMANCE DROPS

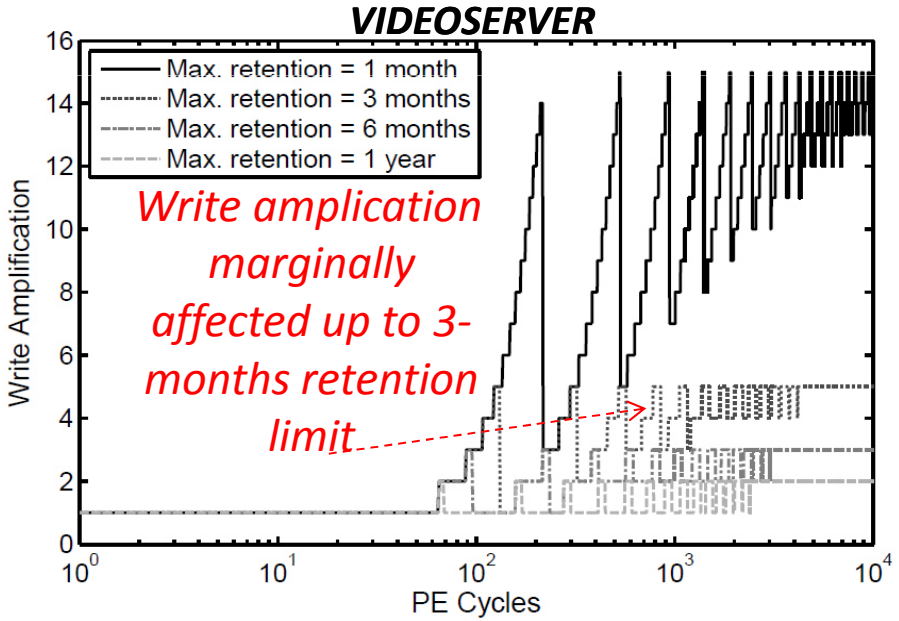
This demonstrates the importance of combining the modelled RBER with the measured one

Wearout Implications

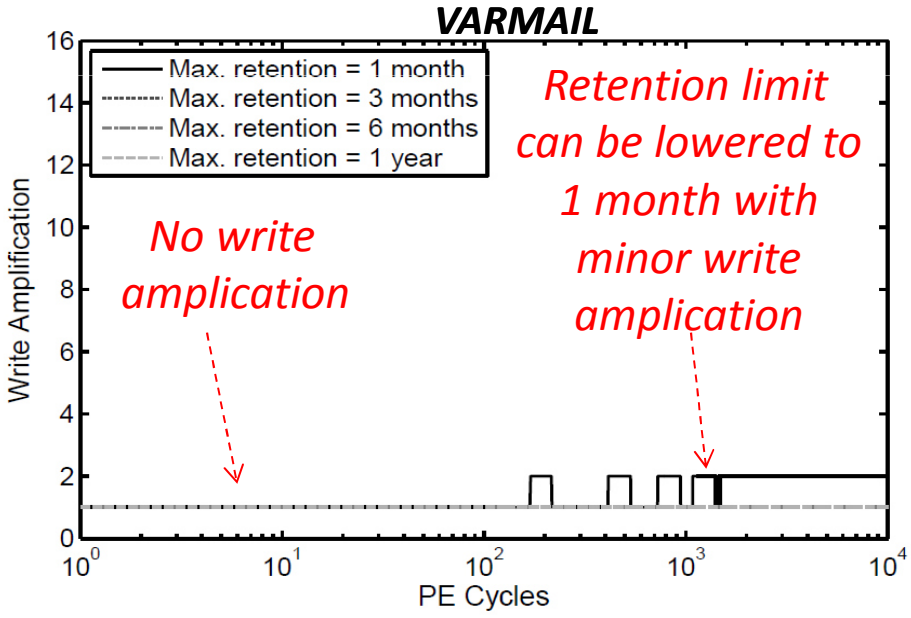
- FLARES introduces a mechanism to issue a page refresh command for those pages that are reaching their maximum retention time.
  - Reduction of the impact of retention errors
  - May introduce additional programming operations that have an impact over the endurance of the Flash
- 2 page refresh techniques are feasible and compatible with FLARES:
  a) Remap the page to a different location
     i. Same policy for wear-leveling: already supported by controllers
     ii. Causes additional write operations, affecting flash wearout
  b) Reprogramming the cell in its original location
     i. Recharging floating gates of leaky cells
     ii. No wearout implications
     iii. Induces program errors in neighboring cells, hence worsening flash RBER

Write Amplification

- Memory wearout measured by the write amplification:
  - Total writes **with** FLARES/Tot. writes **without** FLARES
- 3 applications from the Filebench file system benchmark suite
  - Videoserver (read-intensive)
  - Webserver (balanced read/writes)
  - Varmail (write-intensive)



AVERAGE PAGE RETENTION TIME IS SLIGHLY LARGER THAN 1-YEAR, THUS TRIGGERING SOME REFRESH OPERATIONS

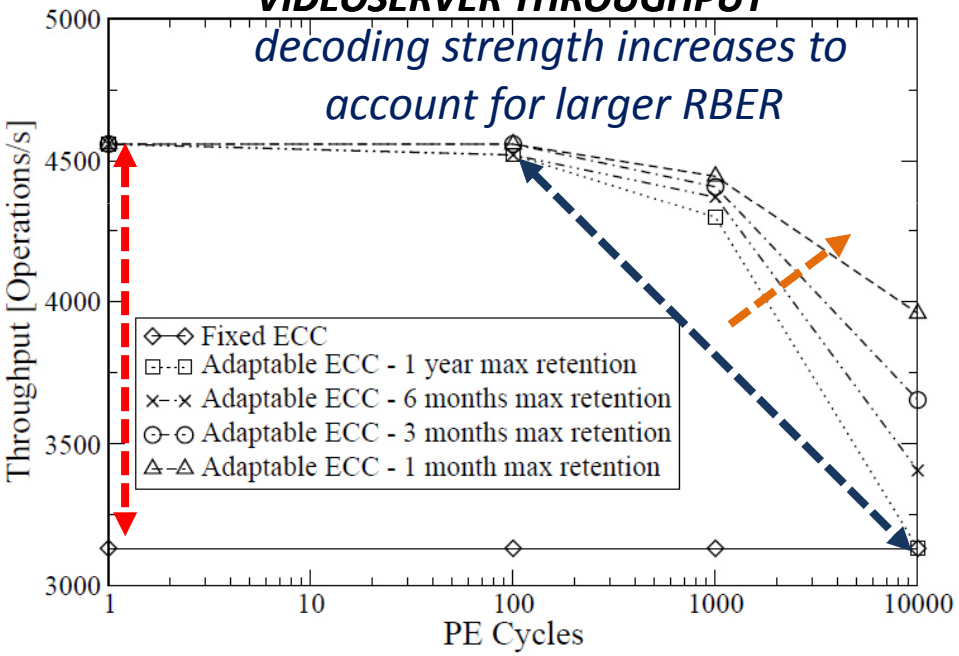


PAGES HAVE HIGH PROBABILITY TO BE REWRITTEN AND IN GENERAL DO NOT REACH THE 1-YEAR RETENTION LIMIT

Performance Implications

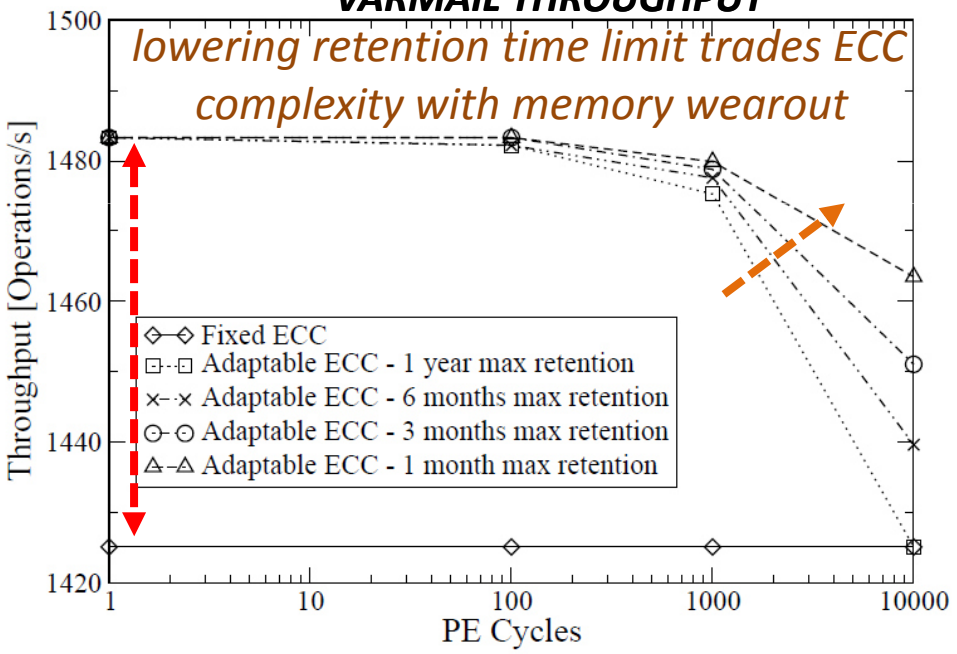
- Throughput of parametric FLARES configurations compared with fixed-, WC-ECC
- Random injection of errors based on device RBER curves and random page refresh triggered based on collected write amplification statistics

VIDEOSERVER THROUGHPUT



Large speedups given by low RBER and by the read-intensive nature of the benchmark

VARMAIL THROUGHPUT



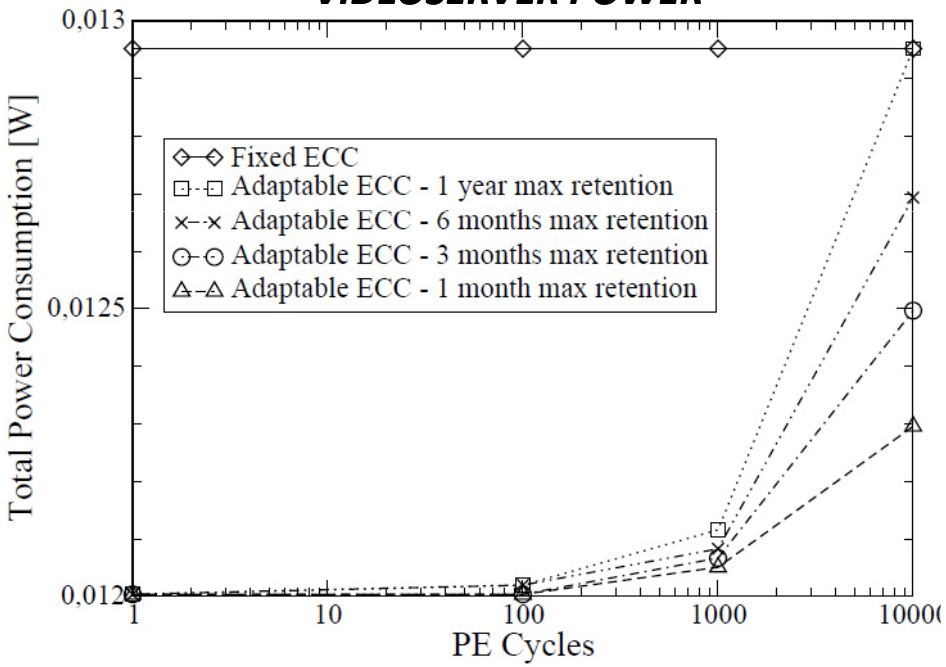
5% improvement despite the write-intensive nature of the benchmark

The high performance gain of lowering maximum retention time limit comes with marginal wearout degradation: outstanding memory performance boost technique

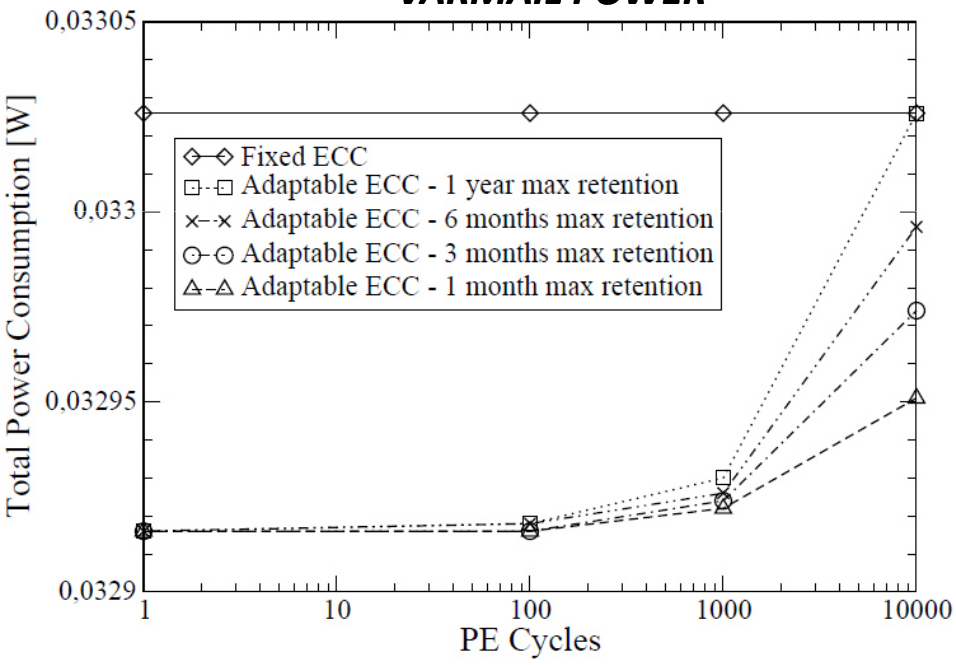
Power Implications

- Power consumption of NAND Flash Array + ECC sub-system
- STM 45 nm technology library

VIDEOSERVER POWER



VARMAIL POWER



Techniques capable of reducing ECC decoding complexity (correction capability adaptation, retention time limit lowering) yield power savings in addition to performance boosts

Conclusions

- We propose FLARES: an algorithm able to estimate at runtime the best ECC correction capability to apply to each page of a flash-based storage system.
  - *Key idea: improving performnace of the flash by carefully tuning the reliability level to the actual wearout conditions of flash pages.*
  - *Key idea: capability to allocate additional correction power at runtime only when and where needed*
- FLARES has been fully implemented within the YAFFS2 file system under the Linux operating system
- FLARES yields strong improvements in overall application throughput for real-life benchmarks, confirming the power of adaptation
- FLARES predictions have been demonstrated to be in general accurate, thus enabling one to fit the reliability requirements imposed by real applications