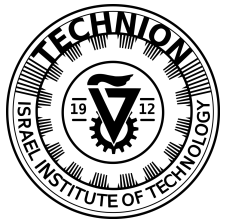
The Devil is in the Details: Implementing Flash Page Reuse with WOM Codes

Fabio Margaglia, Gala Yadgar, Eitan Yaakobi, Yue Li,
Assaf Schuster, André Brinkmann
Johannes Gutenberg University, Mainz
Technion
California Institute of Technology



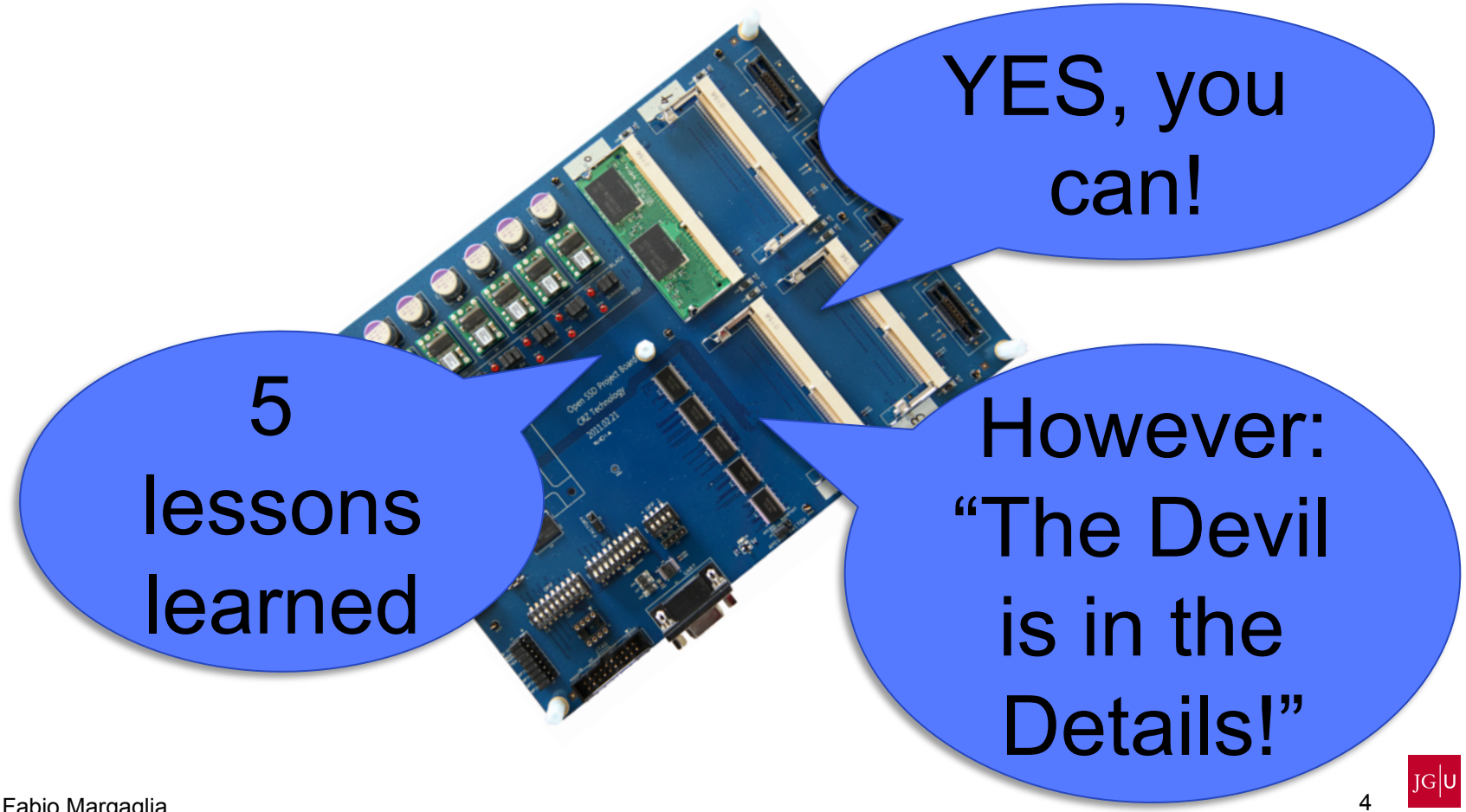
JOHANNES GUTENBERG UNIVERSITÄT MAINZ

JG|U

Flash Page Reuse and WOM Codes are a popular topic

- Grupp et al. “Characterizing flash memory: Anomalies, observations, and applications.” MICRO‘09.
- Jagmohan et al. “Write amplification reduction in NAND flash through multi-write coding.” MSST’10
- Yaakobi et al. “Error characterization and coding schemes for flash memories.” GLOBECOM ’10
- Jacobvitz et al. “Writing cosets of a convolutional code to increase the lifetime of flash memory.” Allerton ’12
- Luojie et al. “WOM codes reduce write amplification in NAND flash memory.” GLOBECOM ’12
- Kaiser et al. “Extending SSD lifetime in database applications with page overwrites.” SYSTOR ‘13
- Berman et al. “Retired-page utilization in write-once memory – a coding perspective.” ISIT’13.
- Odeh et al. “NAND flash architectures reducing write amplification through multi-write codes.” MSST ‘14
- Gad et al. “Rewriting flash memories by message passing.” ISIT ’15
- Yaakobi et al. “When do WOM codes improve the erasure factor in flash memories?” ISIT ’15
- Margaglia et al. “Improving MLC flash performance and endurance with extended P/E cycles.” MSST ’15
- Yadgar et al. “Write once, get 50% free: Saving SSD erase costs using WOM codes.” FAST ‘15

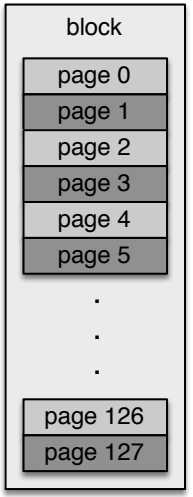
Image courtesy of lekkyjustdoit at FreeDigitalPhotos.net


YES, you can!
5 lessons learned
However: “The Devil is in the Details!”

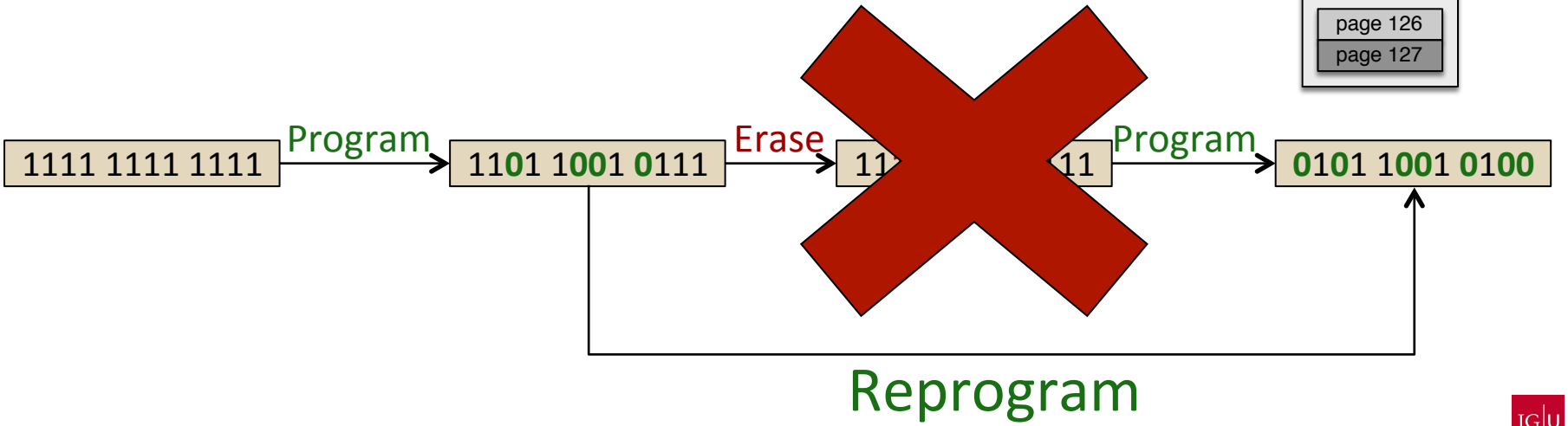
Background

NAND Flash traditional assumptions:

Operation	Granularity	Effect
Program	Page	Clear arbitrary bits
Erase	Block	Reset all bits



To save the erase operation reprogram with: $0 \nrightarrow 1$



Write Once Memory Codes

WOM Codes ensure $0 \nrightarrow 1$

Plain bits	1st gen	2nd gen
00	111	000
01	110	001
10	101	010
11	011	100

[Rivest and Shamir, '82]

- encode *n* bits with *m* bits, where *m* > *n*
- encodings organized into reprogram compatible generations

Example:

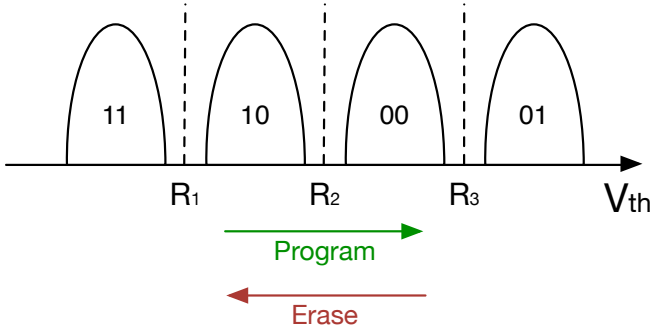
What about MLC?

MLC – Multi Level Cell

In MLC 1 physical cell represents 2 bits

Every MLC is characterized by V_{th}

- 4 states
- program operations increase V_{th}
- erase operations reset V_{th}



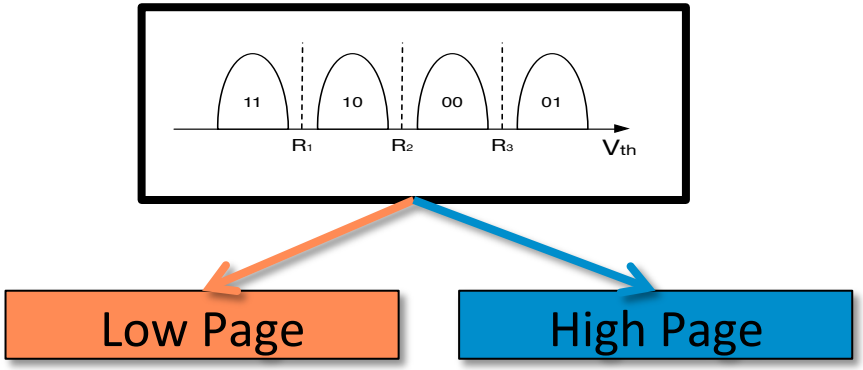
- Reprogramming may cause program disturbance between bits

  [Grupp et. al, MICRO 2009][Margaglia et. al, MSST 2014]

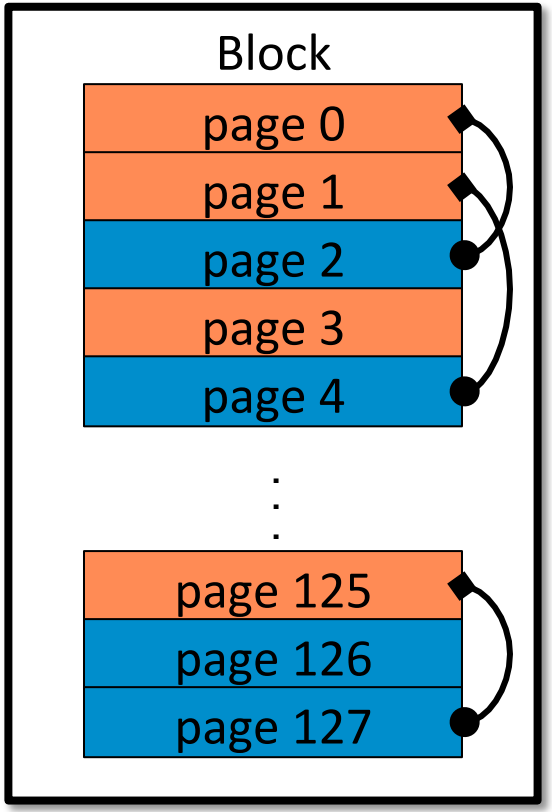
WOM codes are not enough

Mapping bits to pages

The bits of a single MLC are mapped to 2 independent pages



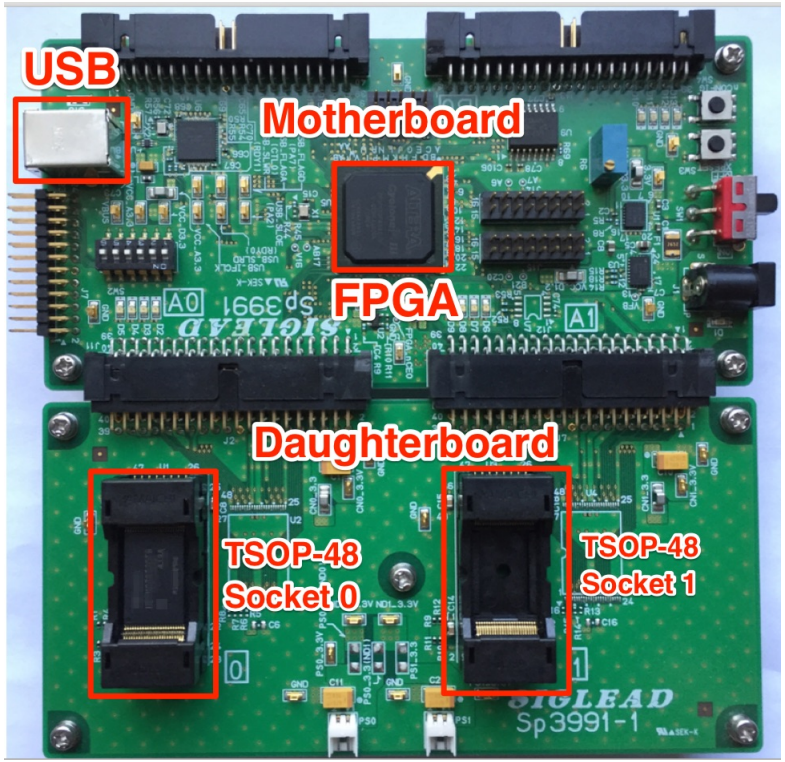
Program disturbance across pages



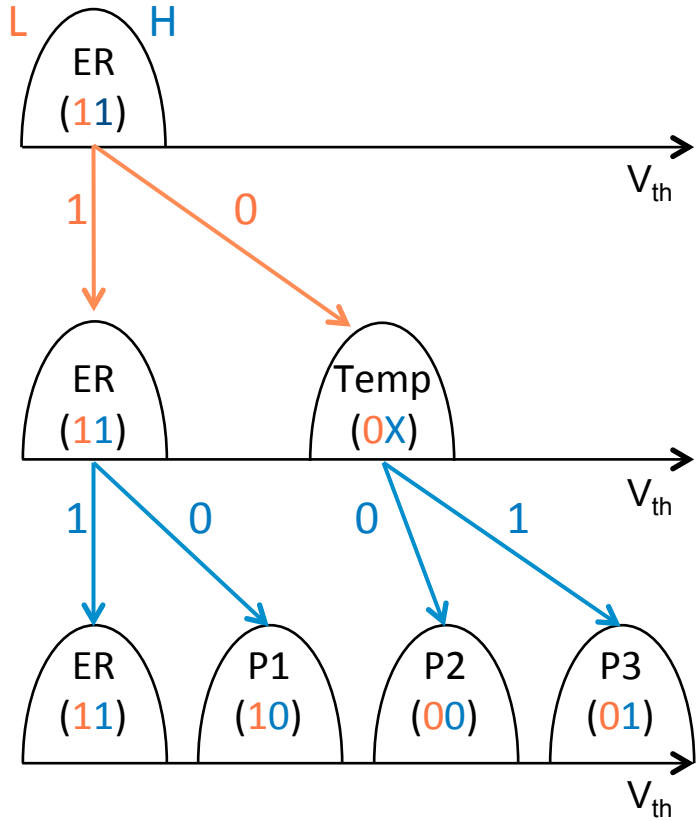
Setup for the analysis of MLC chips

Siglead SigNASII NAND Tester

- Multiple manufacturers
- Multiple feature sizes



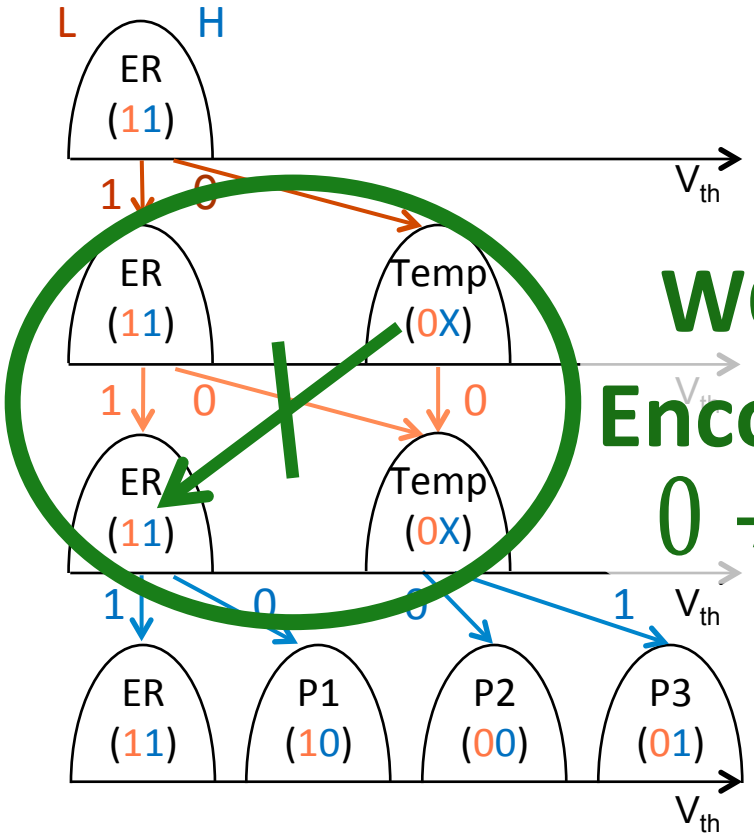
Low-High (LH) Programming scheme



Low Bit Program

High Bit Program

Low-Low-High (LLH) Reprogramming Scheme



WOM Encoding

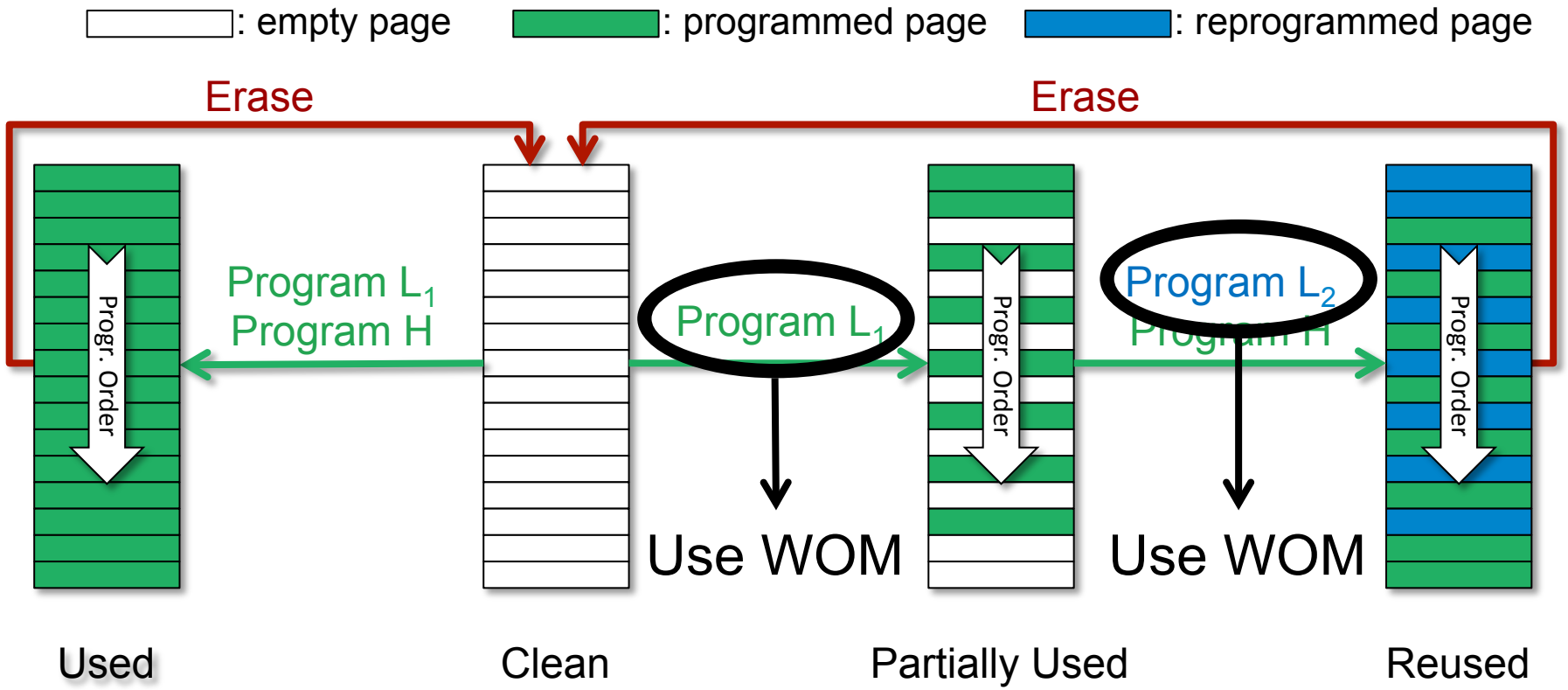
$0 \nrightarrow 1$

Low Bit Program (1)

Low Bit Program (2)

High Bit Program

Block Life Cycle with LLH Reprogramming

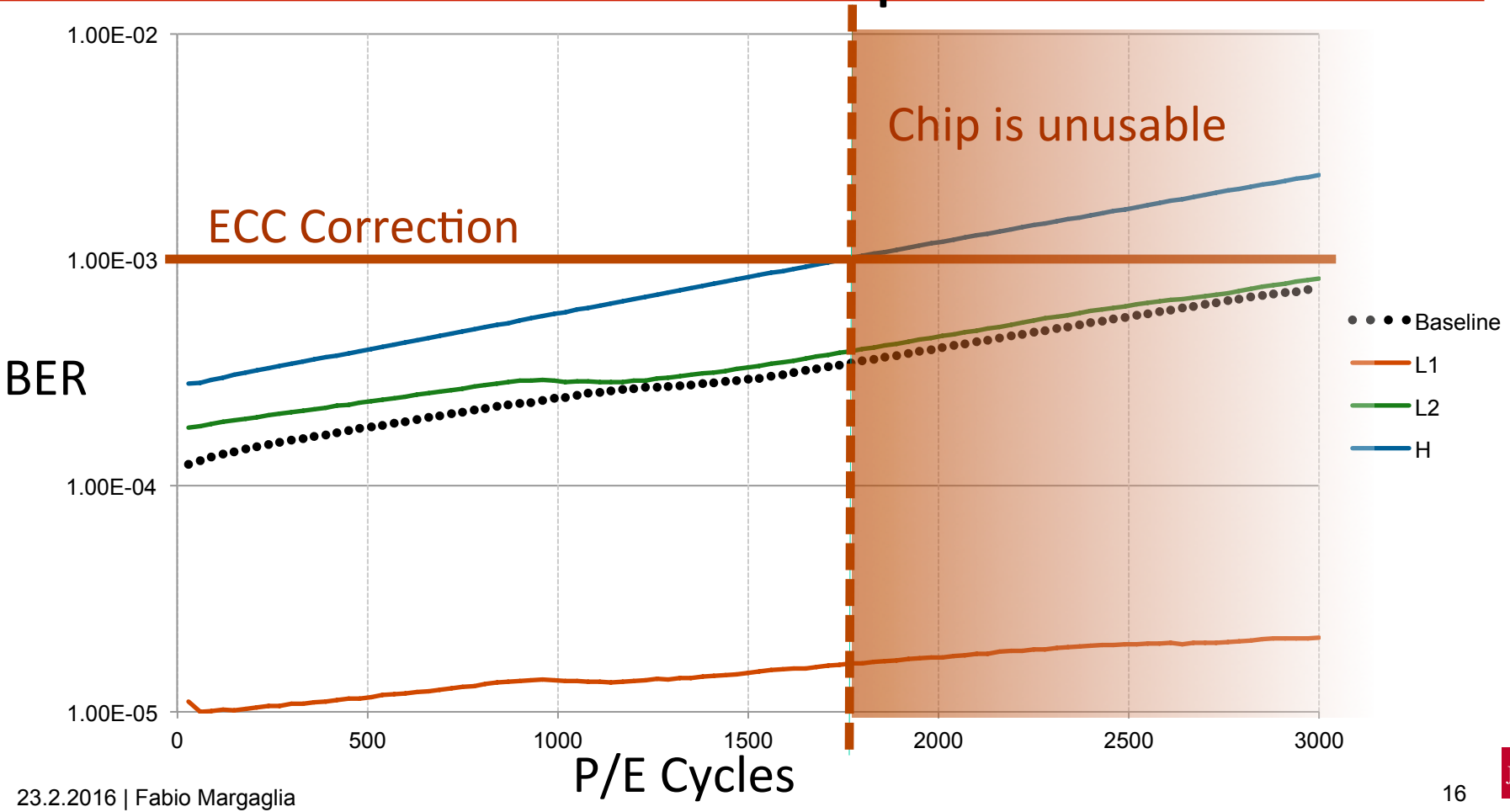


Lesson I

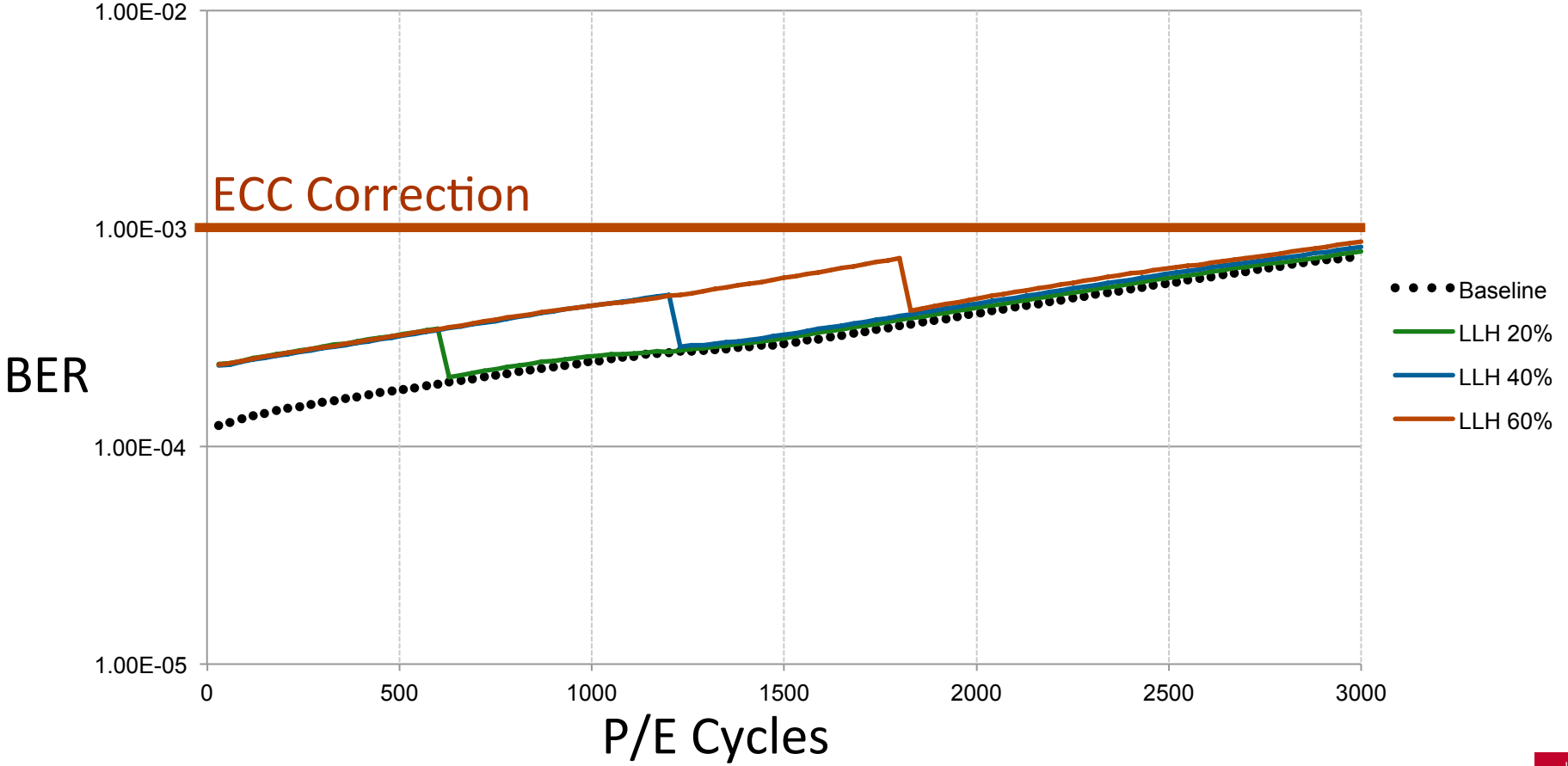
Page reprogram in MLC is possible,
but only for half of the pages.

LLH Reprogramming and cell wear

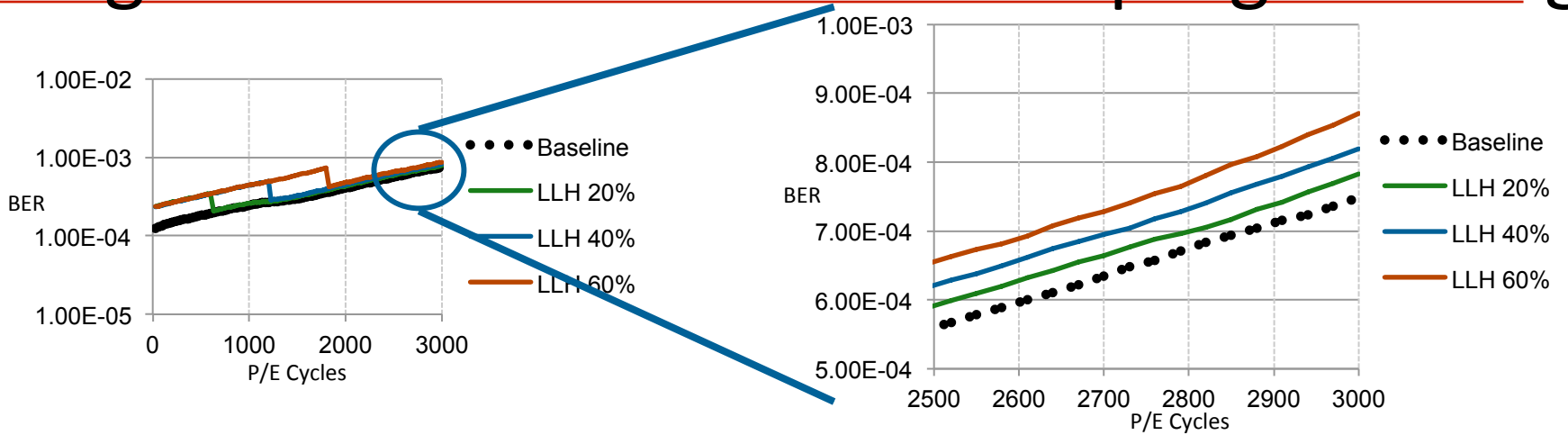
BER for LLH Scheme – Chip A 16nm



BER for LLH partial application – Chip A 16nm



Long-term cell wear with LLH Reprogramming



Relative reduction of P/E cycles to reach BER=10^{-3}

Part. application	A16	B16	B27	B29
60%	8.5%	7%	8%	8.5%
40%	5.2%	5%	6%	5.5%
20%	1%	3%	2.5%	3%

Lesson II

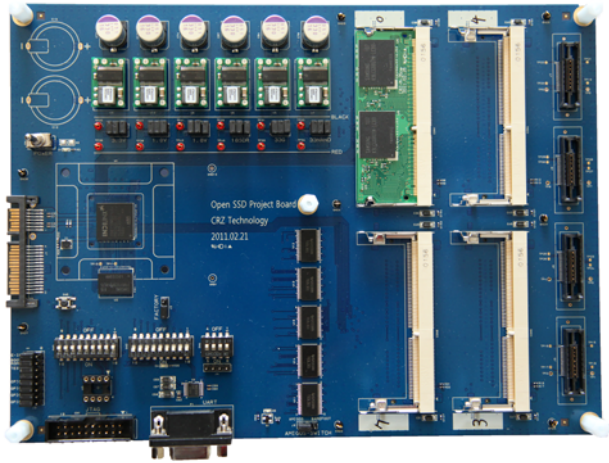
LLH reprogramming applied for a portion of the lifetime causes only minor long-term cell wear

FTL Architecture

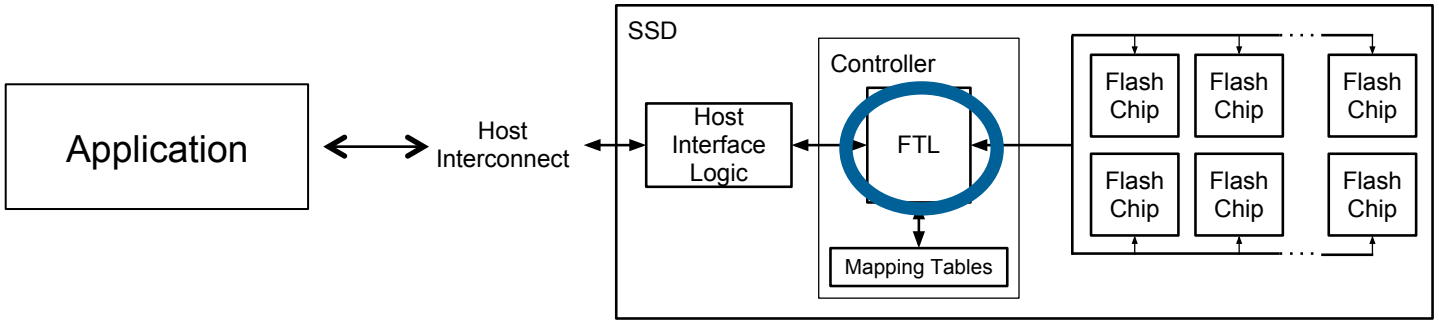
Prototype

Jasmine OpenSSD board
www.openssd-project.org

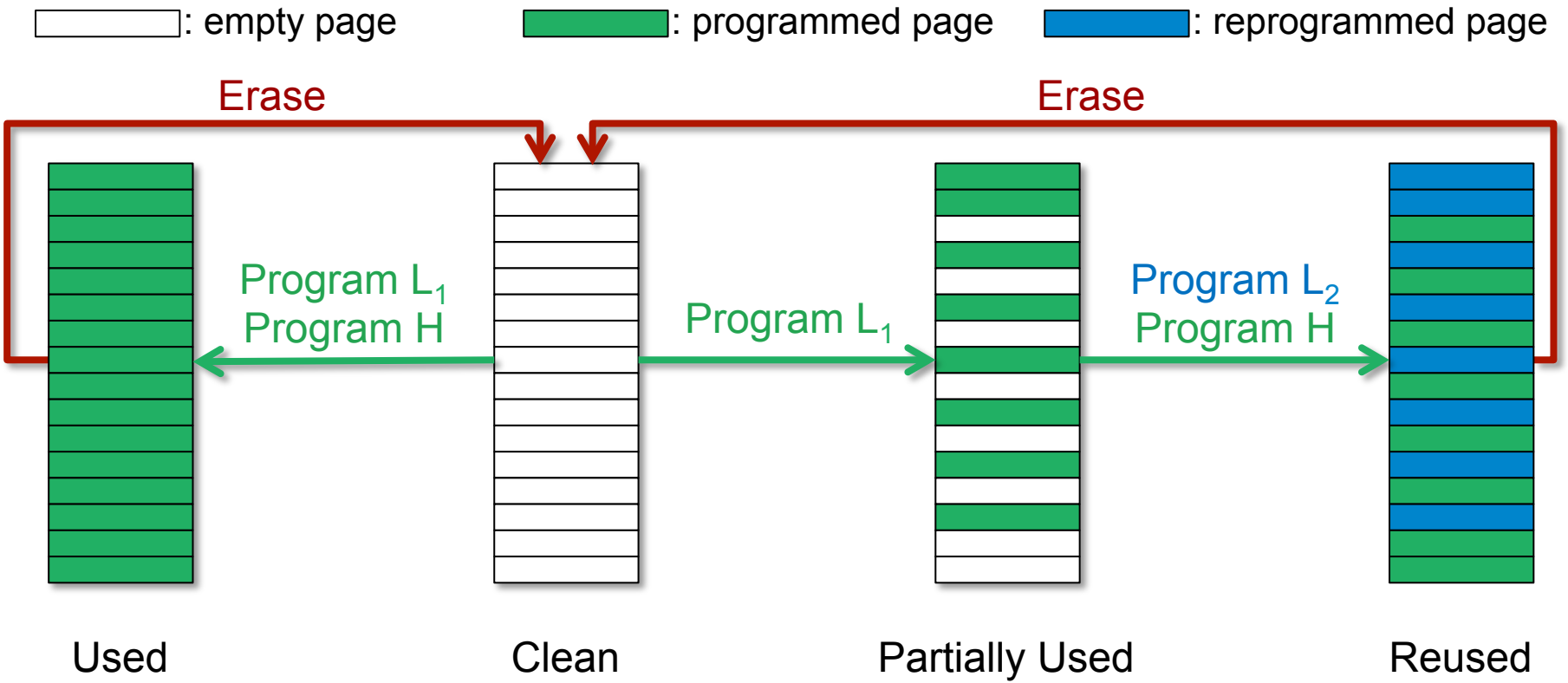
- ARM7TDMI single core
- SATA interface
- 64MB DRAM
- Samsung 35nm MLC



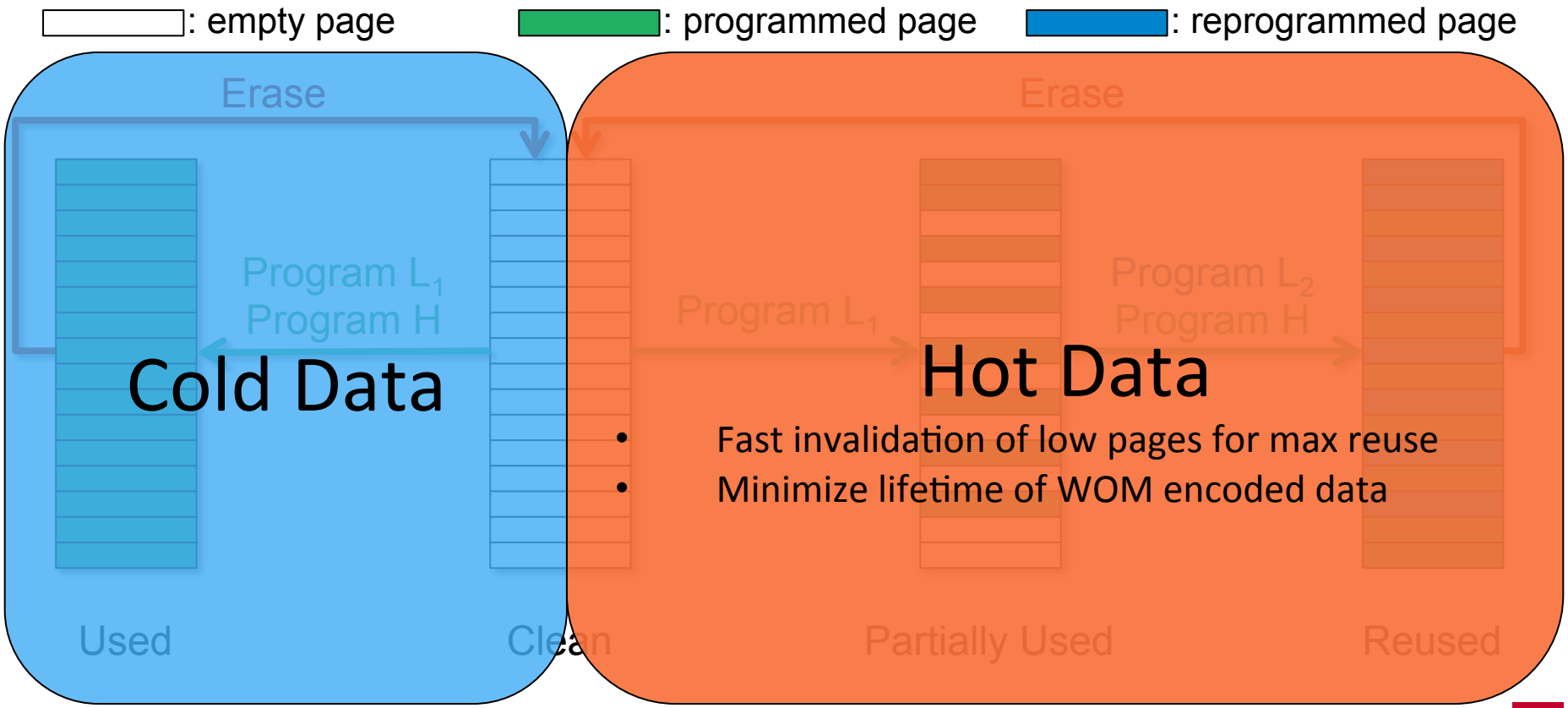
Modify only the FTL



Block Life Cycle with LLH Reprogramming



Block Life Cycle with LLH Reprogramming

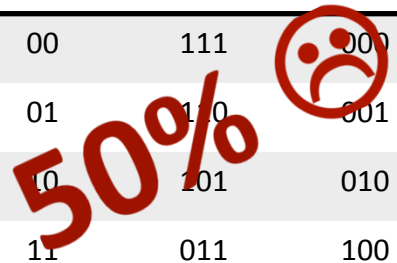


- Fast invalidation of low pages for max reuse
- Minimize lifetime of WOM encoded data

WOM overhead

- Polar WOM Codes
- DRAM Buffers

Plain bits	1st gen	2nd gen
00	111	000
01	110	001
10	101	010
11	011	100



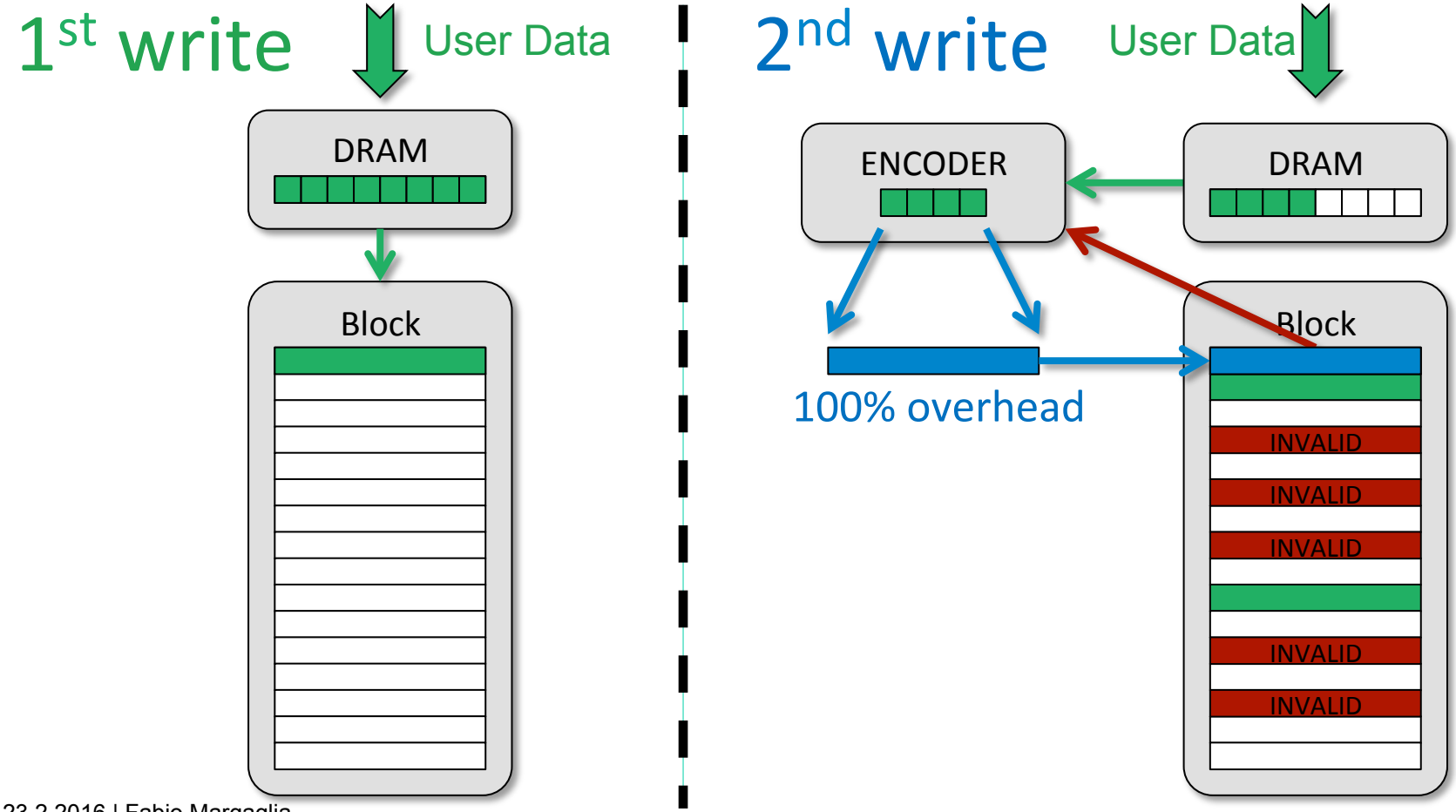
Polar WOM Codes

Probabilistic codes introduced in [Burshtein and Strugatski, IT `13]

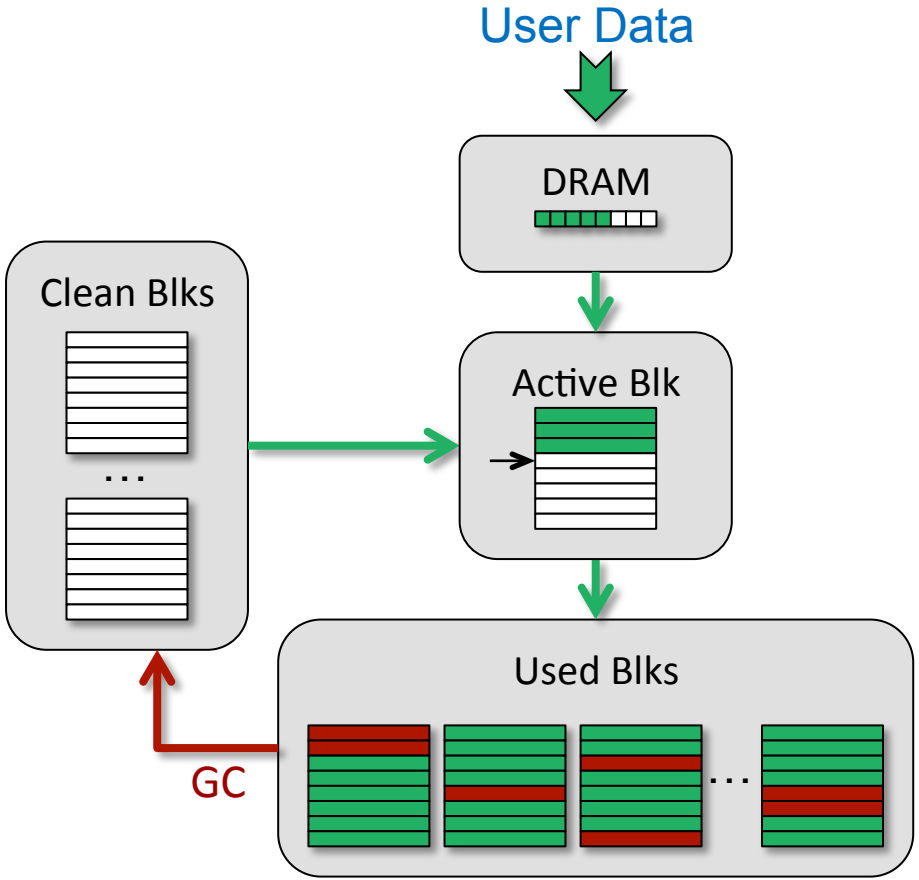
Already proposed for Reusable SSD in [Yadgar et. al, FAST `15]

- Complexity ≅ LDPC ECC (could be accelerated in HW)
- 1st write ➔ no overhead
- 2nd write ➔ 100% overhead

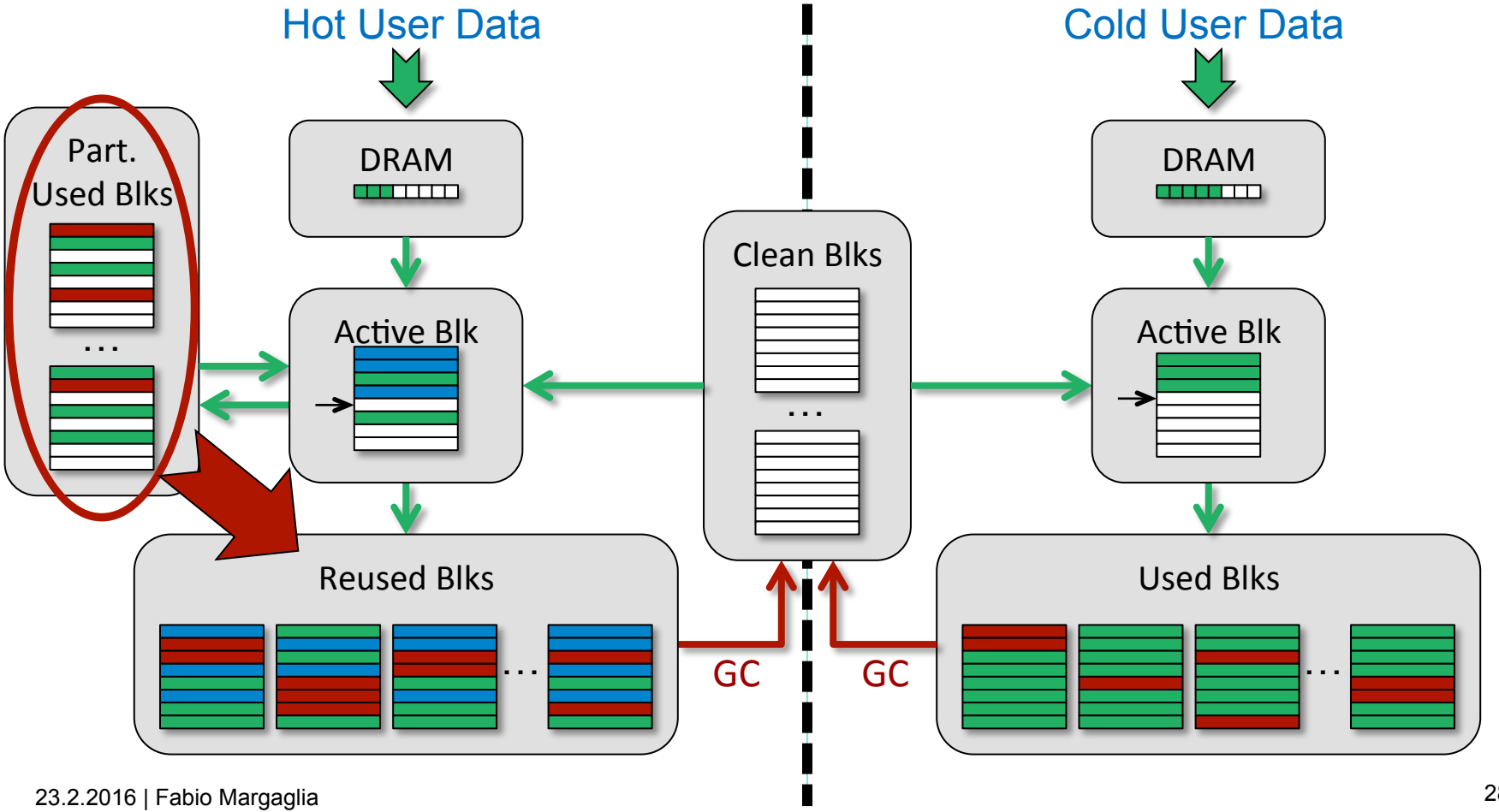
DRAM Buffers



Putting it all together..

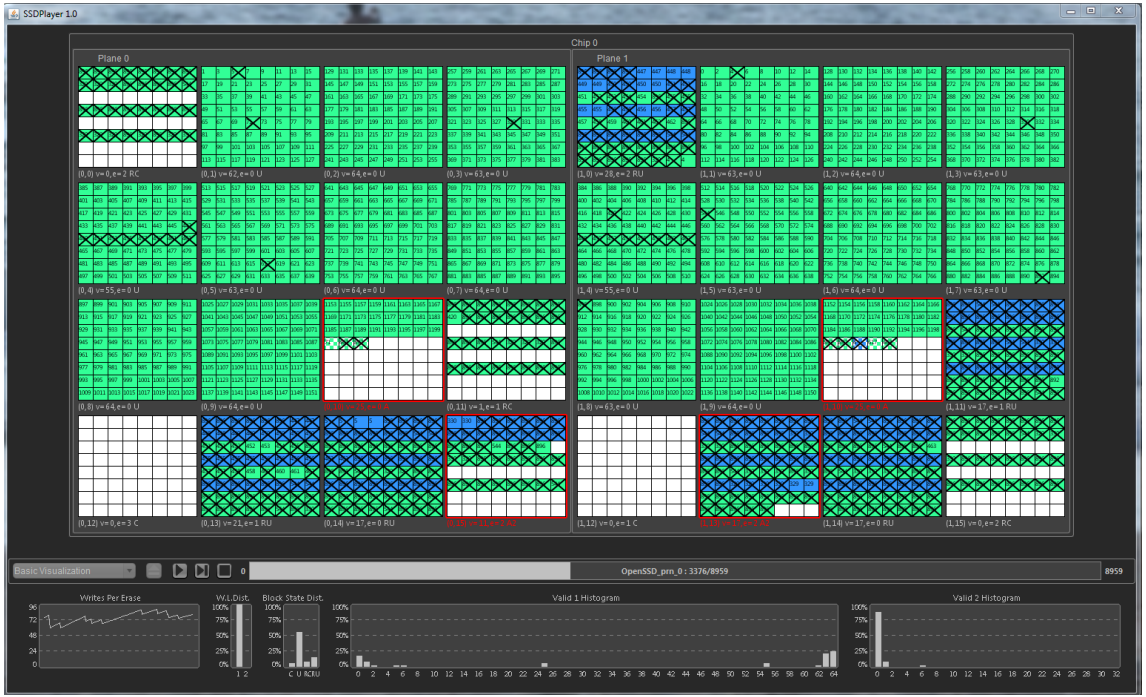


Putting it all together..



Demo

- Visualization of the SSD internal status
- SSDPlayer [Yadgar, Shor, Yaakobi, and Shuster, HotStorage ’15]
- Replay of OpenSSD run of MSR Cambridge trace
- Small device



http://www.cs.technion.ac.il/~gala/LLH-FTL-MSR/LLH-FTL-MSR.html

Evaluation:

Number of Erasures and Lifetime

Best Case Reduction in Erasures

Standard SSD

Erasures ← $E = \frac{M}{N}$ → Logical Writes

→ Pages per Block

Reusable SSD [Yadgar et. al, FAST 2015]

Reuse all pages with 100% WOM overhead

$$E' = \frac{M}{N + \frac{N}{2}} = \frac{2}{3}E$$

➔33% Reduction

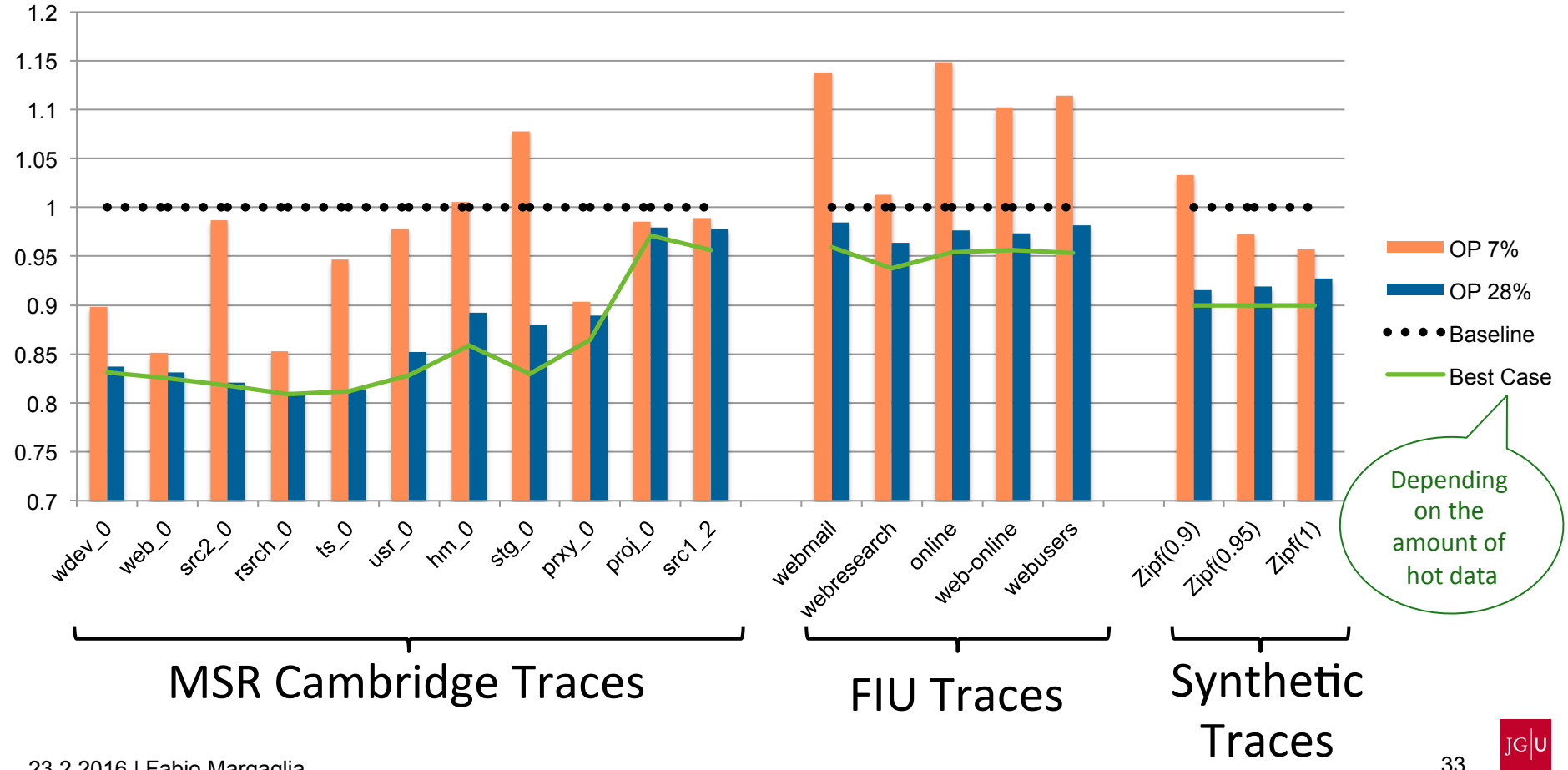
The Devil is in the Details

Reuse only low pages with 100% WOM overhead

$$E'' = \frac{M}{N + \frac{N}{4}} = \frac{4}{5}E$$

➔20% Reduction

Erasures Reduction normalized to Baseline



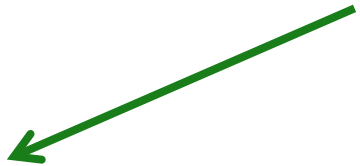
Lifetime

Best case when using LLH Reprogramming for 40% lifetime (T):

$$Writes\ per\ Block\ with\ LLH = N + \frac{N}{4}$$

$$Writes\ per\ Block\ with\ LH = N$$

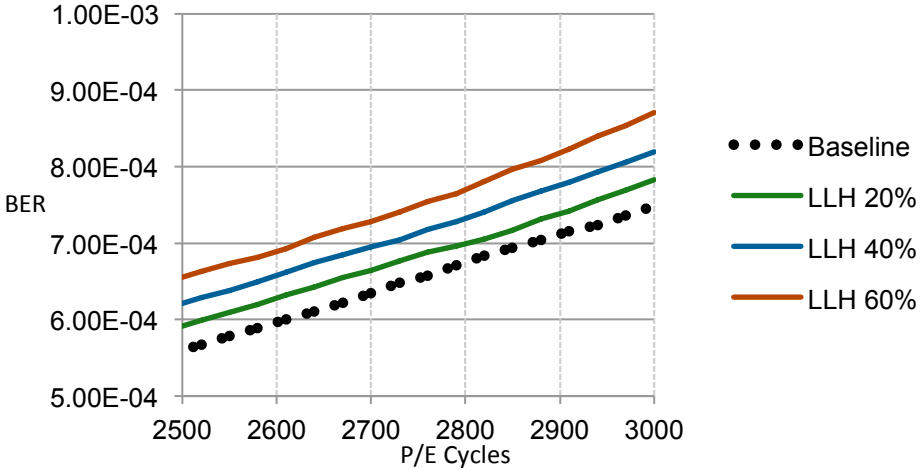
$$Total\ Writes\ per\ Block = 0.4 \cdot T \cdot \left(N + \frac{N}{4}\right) + 0.6 \cdot T \cdot N = 1.1 \cdot T \cdot N$$



10% more writes over the lifetime

Erasures Reduction and Lifetime

Long-term cell wear



Lifetime reduction due to increased BER

Part. application	A16	B16	B27	B29
40%	5.2%	5%	6%	5.5%

10% increase ➔ < 5%

Lesson III

LLH reprogramming reduces erasures, but this does not necessarily translate to a substantial lifetime increase.

Evaluation:

Energy and Latency

Single Operations

We have less erasures, but single read and write operations become more expensive (both for energy and for latency):

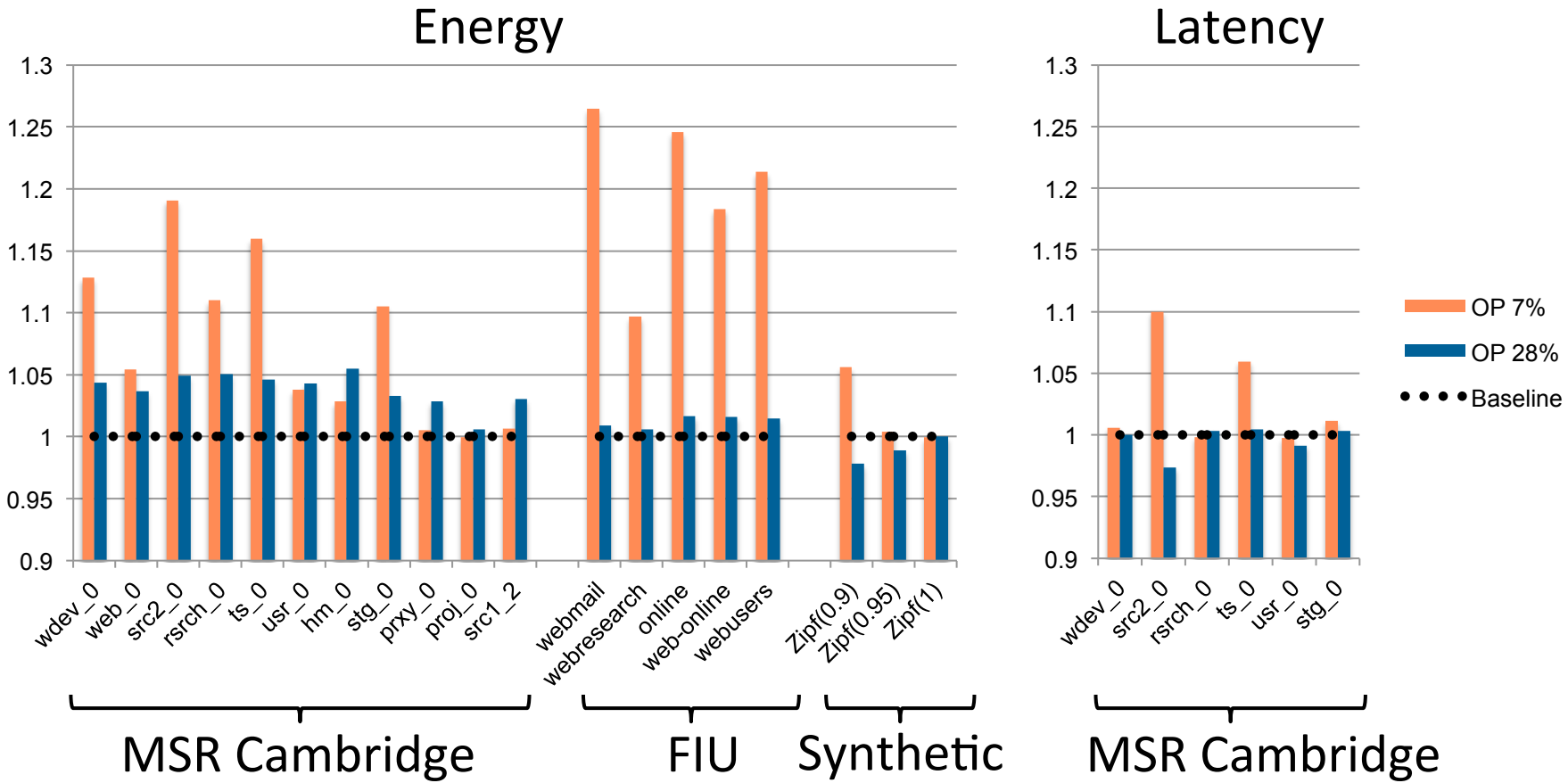
Write WOM encoded data

- Requires read of previous content
- WOM overhead for second writes (use half flash bandwidth)

Read WOM encoded data

- WOM overhead for data written with second write (use half flash bandwidth)

Energy and Latency normalized to Baseline



Lesson IV

The extra energy consumed by read and write operations is larger than that saved in erase operations.

Lesson V

The extra time consumed by read and write operations does not increase latency.

On the contrary, small latency reductions are possible

Conclusions

Lessons learned:

I. Reprogramming possible in MLC

II. Modest long-term wear

III. Less erasures,
limited lifetime increase

IV. Energy consumption increases

V. Limited effect on latency

Where to go from here?

- Manufacturers support
- Application specific reprogramming
  [Margaglia et al. MSST 2015]
- MLC specific WOM codes
- TLC, one-shot programming?

Thanks for Your Attention

Questions?

JOHANNES GUTENBERG
UNIVERSITÄT MAINZ