Resilient Memory Architectures

A very short tutorial on ECC and repair

Dong Wan Kim

Jungrae Kim

Mattan Erez

The University of Texas at Austin

Are DRAM errors rare?

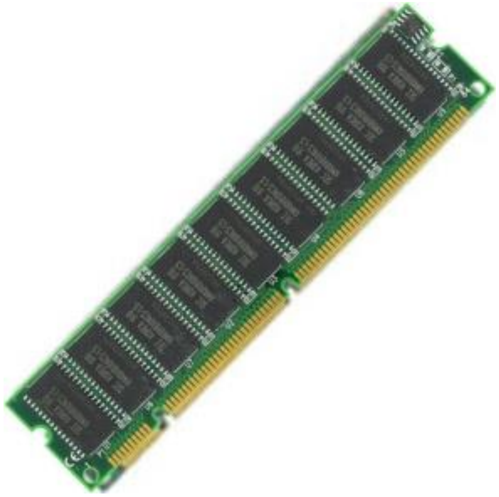
Many errors per minute

(100k nodes, 1.5PB mem)

Detect → Correct → Continue

– Error checking and correcting (ECC) codes

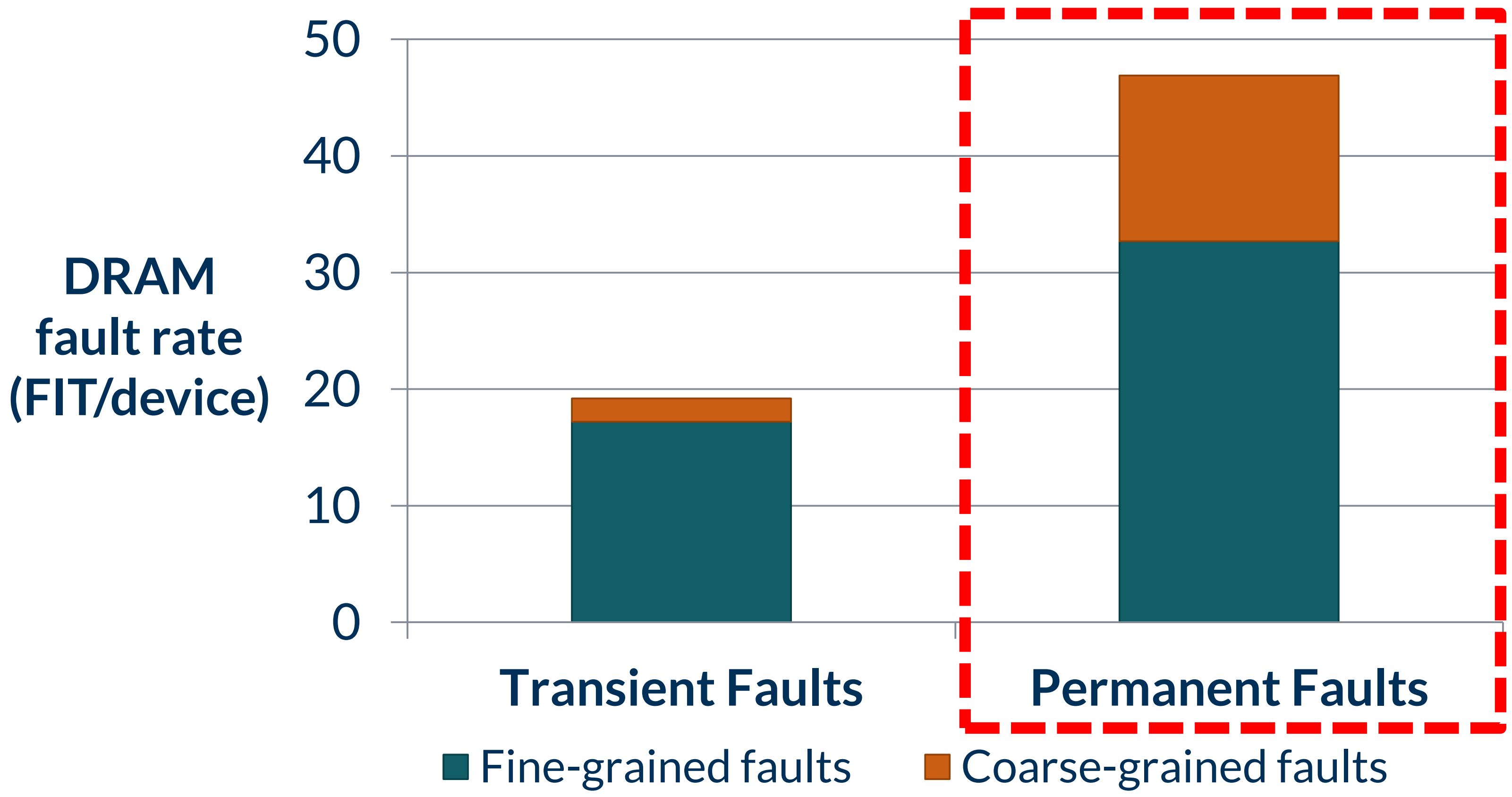
Are DRAM faults rare?



200+ years

~5 hours
(100k nodes, 1.5PB mem)

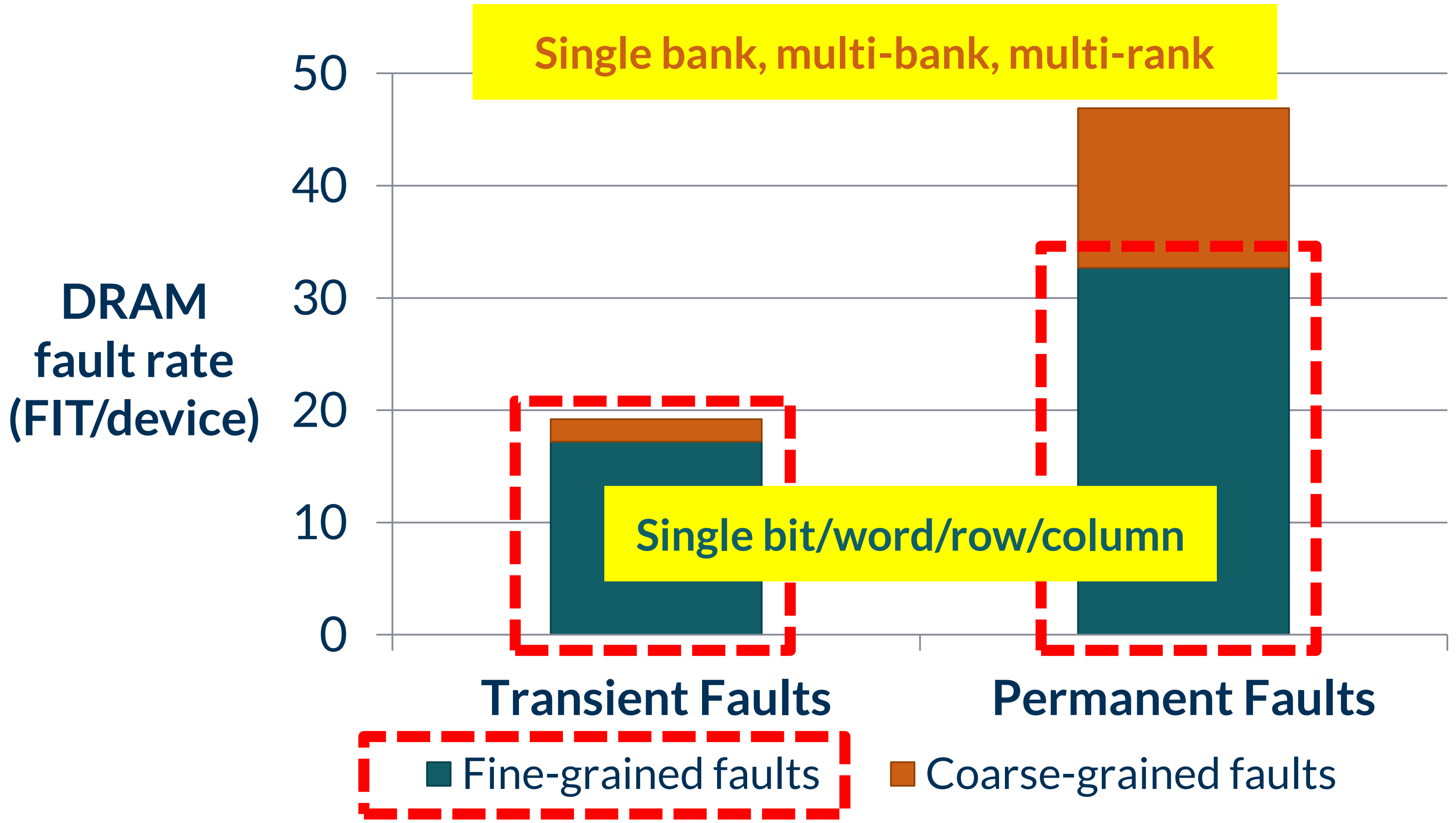
Permanent faults are a big problem



*** FIT (Failure in Time):***
Number of failures expected in **1 billion device-hours of operation.**

* Vilas Sridharan and Dean Liberty, “A Study of DRAM Failures in the Field”, SC 2012

Most faults affect a *small memory region*



**** FIT (Failure in Time):***
Number of failures expected in **1 billion device-hours of operation.**

* Vilas Sridharan and Dean Liberty, “A Study of DRAM Failures in the Field”, SC 2012

ECC for permanent faults?

- Strong ECC correction possible
- Latency and energy overheads?
  - Problematic if errors very frequent
- Detection compromised when redundancy relied on for correction?
  - Will eventually fail

Detect → Correct → **Repair** → Continue

– "Fixing" broken memory eliminates ECC deficiencies

Outline

- ECC
  - “Theory”
  - Common DRAM ECC organizations
  - Single- vs. multi-tier ECC
- Repair
  - Coarse-grained repair
  - Fine-grained repair

Error checking and correcting requires redundancy

Easy start – parity

- Guaranteed to detect any 1-bit error
  - In fact, any odd number of errors (more on this later)

```
1 1 0 1   1
```

$$1\ 0\ 1\ 0 \xrightarrow{?} 1$$

Can parity correct that 1 bit?

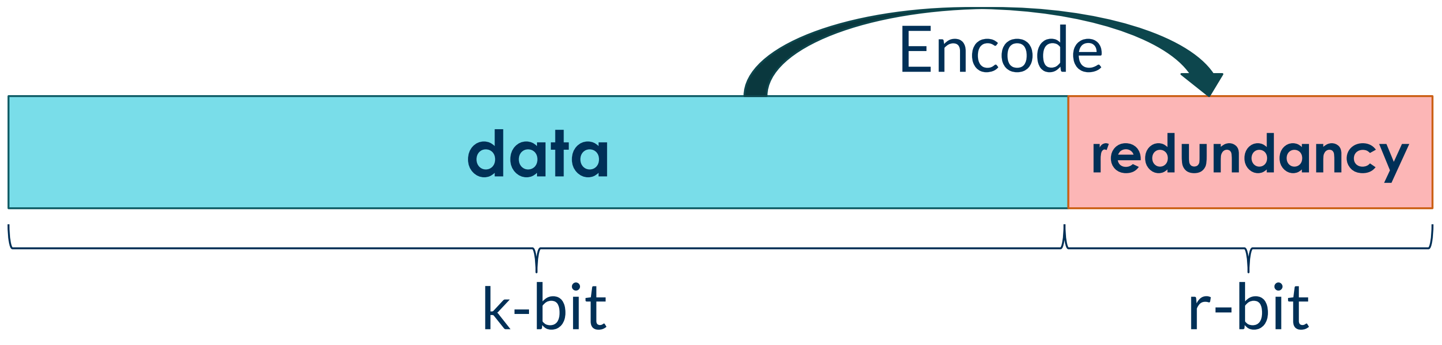
– Yes, if error is location is known by some other means

Erasure decoding

```
1 1 0 1   1


1 X 0 1   1
1   0 1   1
```

Error **Checking** & Correcting (ECC) Codes



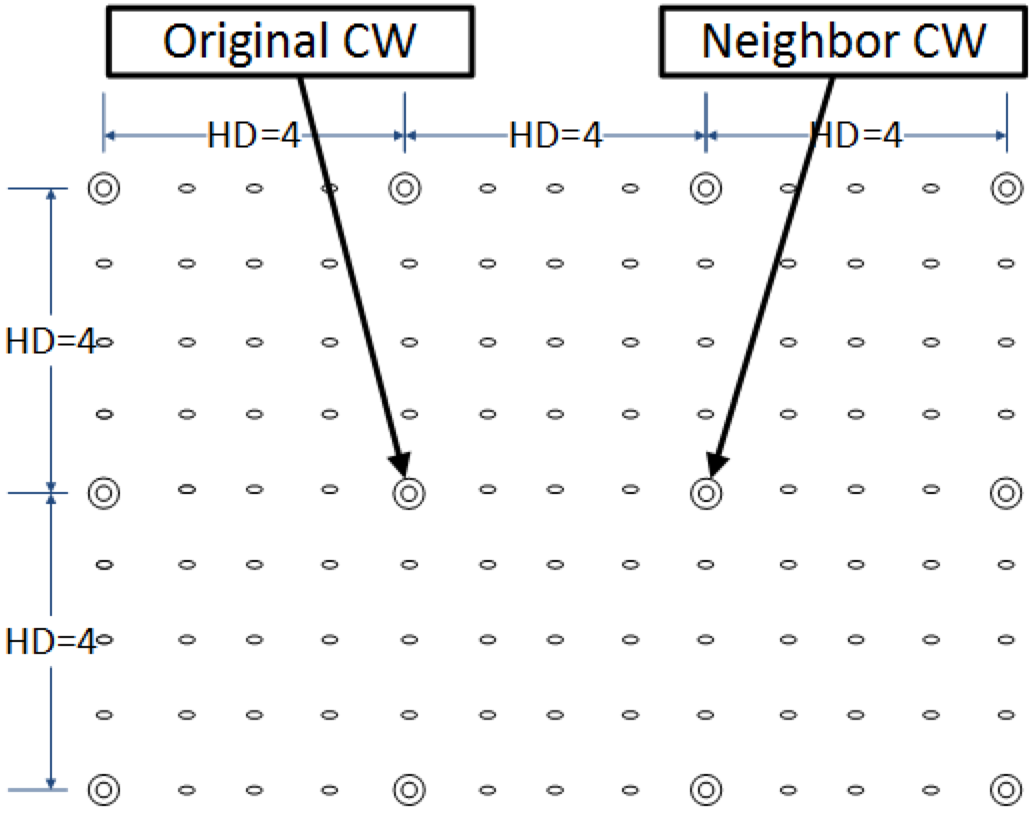
A ***codeword (CW)*** : a valid pair of data and redundancy

* the figure represents a *systematic* code
** (k+r, r) code

How does ECC Correct/Detect an Error?

Code distance (***d***)

– Any CW is at least *d*-symbols different from any other CW



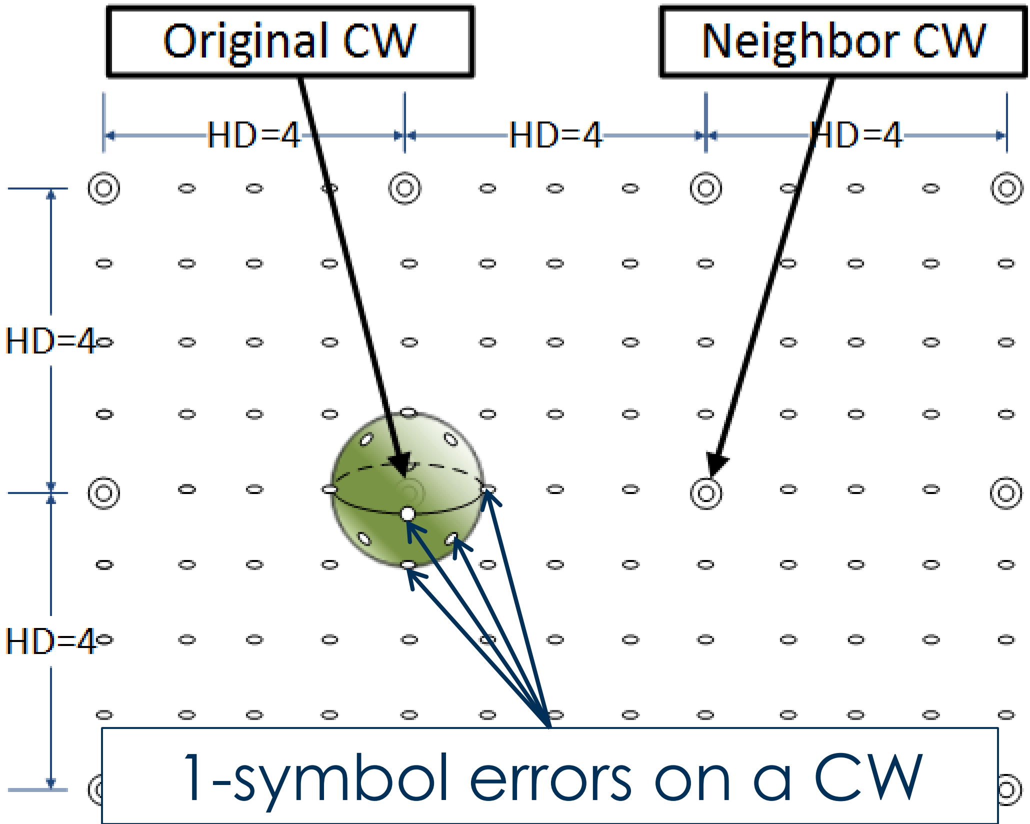
Conceptual code space with *d*=4

1-symbol Error in a Code with *d*=4

Can be detected (not a CW)

Can be corrected by finding the nearest CW

➔ **Detectable and Correctable Error (DCE)**



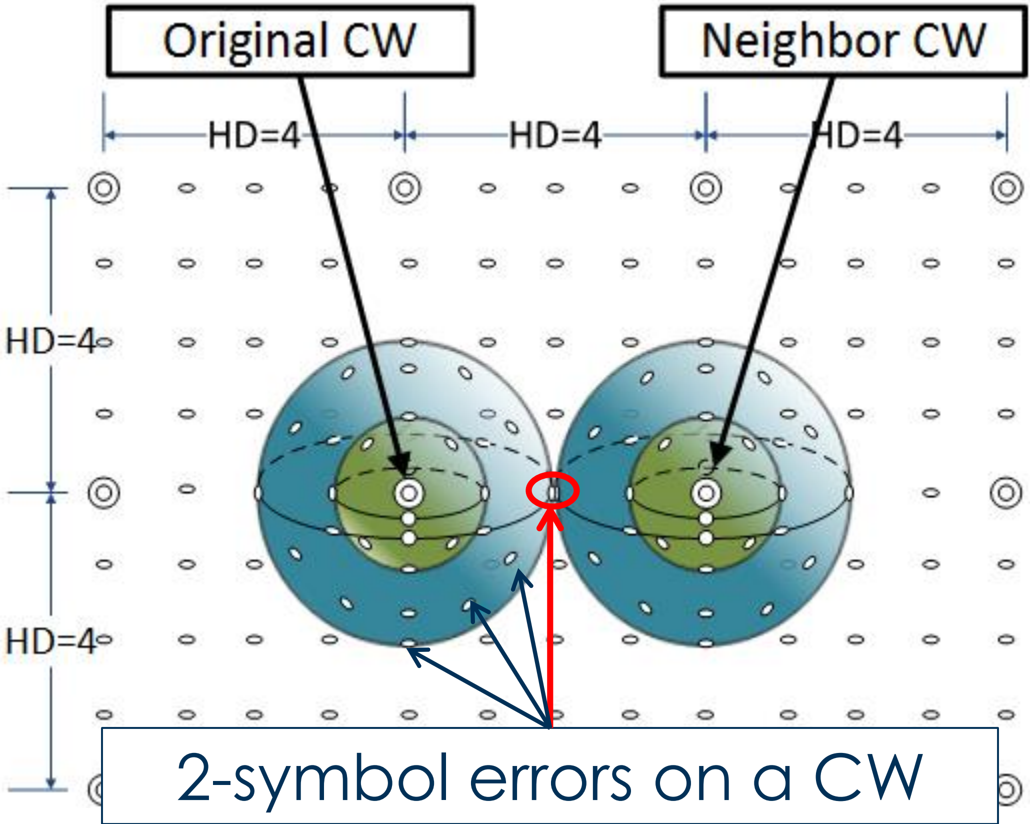
2-symbol Error in a Code with *d*=4

Can be detected (not a CW)

Cannot be corrected due to multiple nearest CWs

Detectable but Uncorrectable Error (**DUE**)

Code with distance 4
➔ Single symbol correcting
– double symbol detecting
(**SSC-DSD**) codes



The challenges are:

- Finding implementable decoders and encoders
- Matching the code properties to DRAM properties and fault/error characteristics

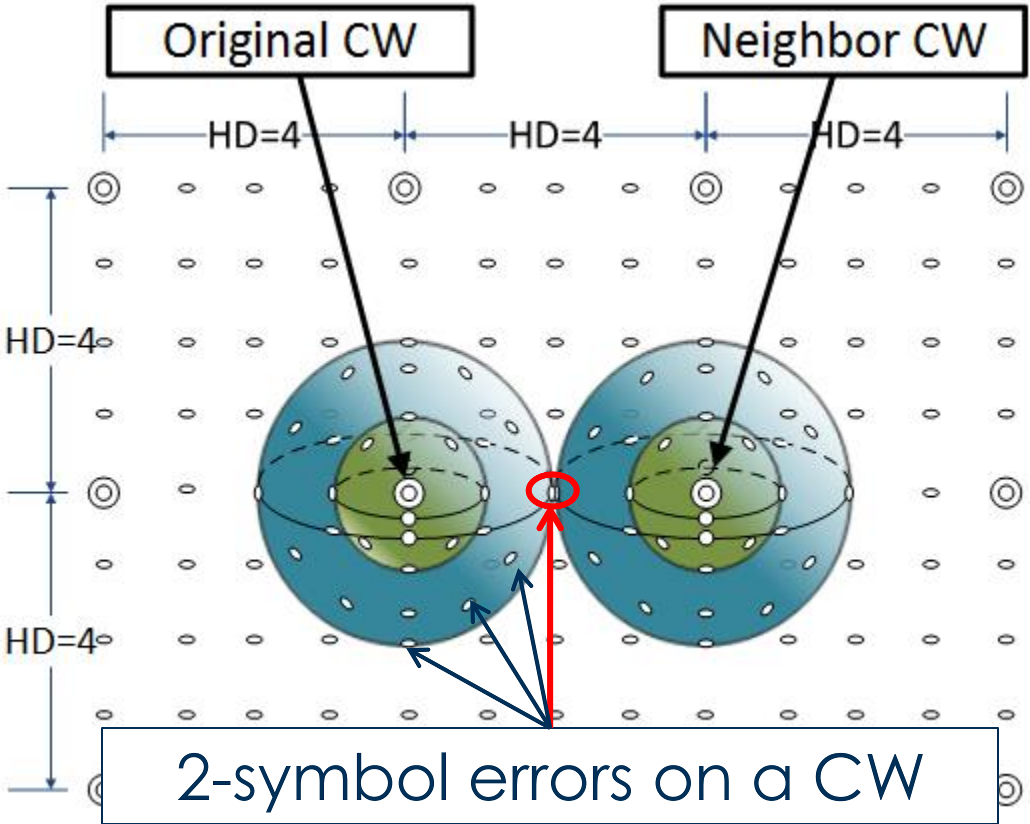
2-symbol Error in a Code with *d*=4

Can be detected (not a CW)

Cannot be corrected due to multiple nearest CWs

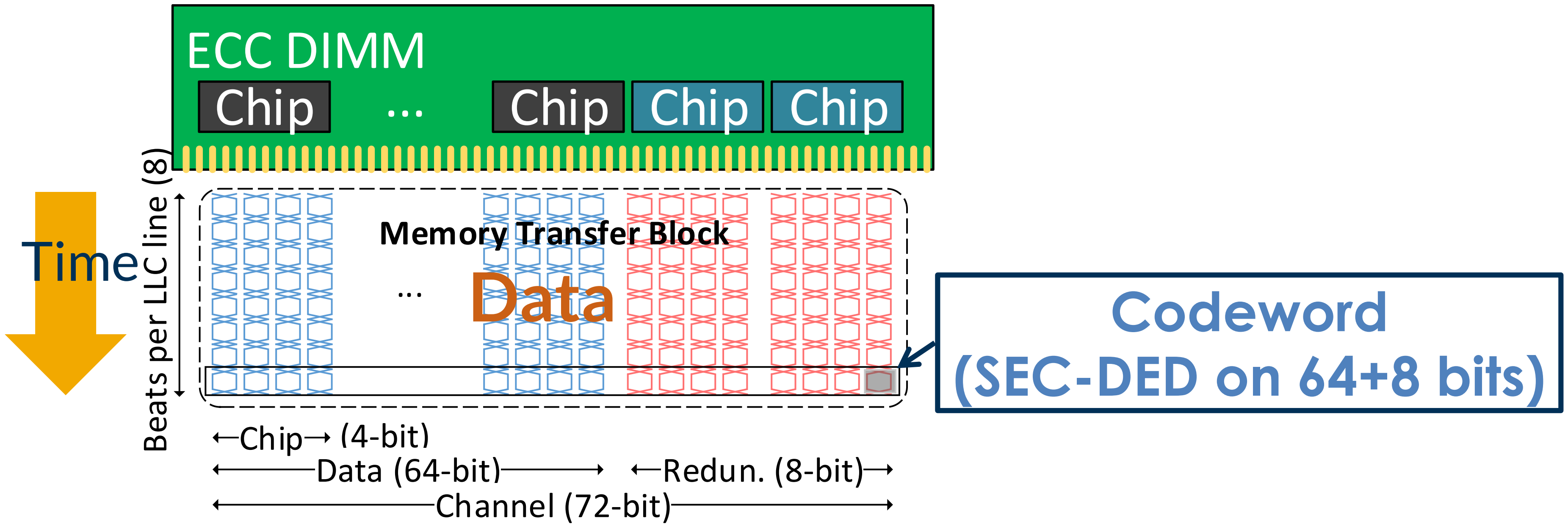
Detectable but Uncorrectable Error (DUE)

Code with distance 4
➔ Single symbol correcting – double symbol detecting (**SSC-DSD**) codes



SEC-DED (Single Error Correcting – Double Error Detecting)

- On single ECC-DIMM
  - 64-bit data + 8-bit redundancy
- Bit-level (weak) protection
  - **Hamming codes**



Reed Solomon codes

- A code with distance ***d***
  - Guaranteed to detect and correct up to $\frac{d-1}{2}$ errors
  - Minimum redundancy = (***d*-1**) symbols
- Reed-Solomon (RS) codes
  - Can provide minimum redundancy

8-bit sym RS codes: ***2t***-sym redundancy for ***t***-sym correction

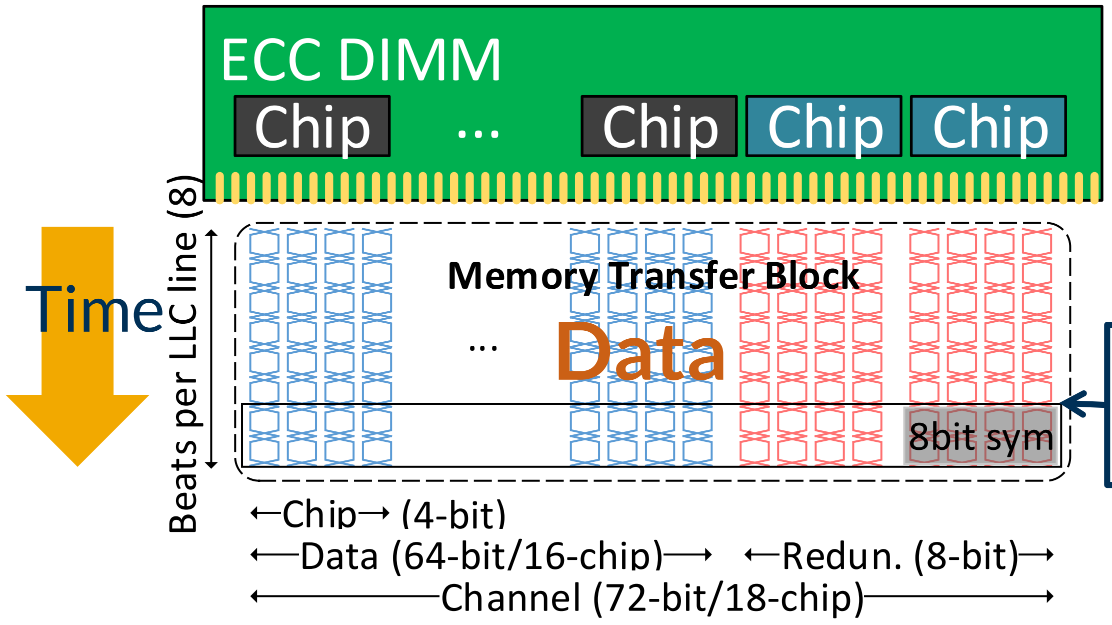
Min. Redun.	*d*	Correction	Detection	Name
2	3	**1-sym**	1-sym	Single symbol correcting (**SSC**)
3	4	1-sym	2-sym	Single symbol correcting – double symbol detecting (**SSC-DSD**)
4	5	**2-sym**	2-sym	Double symbol correcting (**DSC**)
		…		

Stronger DRAM ECC

Chipkill-correct

- Can restore data from a completely failed chip
- 99.94% correction of errors (vs. 91% in SEC-DED)[1]

Chipkill-level corrects most chip errors (not 100%)



AMD Chipkill-correct on single ECC-DIMM (baseline)

[1] "Chipkill Memory", IBM, 2001

Even stronger codes?

Effective ECC coverage much better than guarantee

What happens beyond detection coverage?

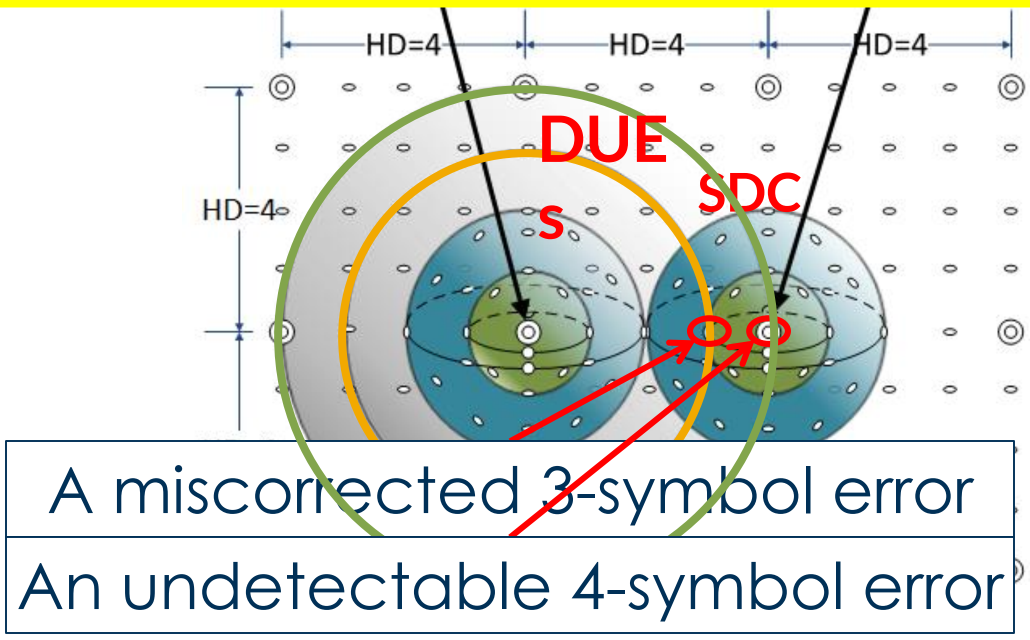
- E.g. 3+ symbol errors on SSC-DSD

Silent Data Corruption (**SDC**)

- Miscorrected or undetected

Larger code distance exponentially decreases SDC probability by making code space more sparse.

DUE otherwise

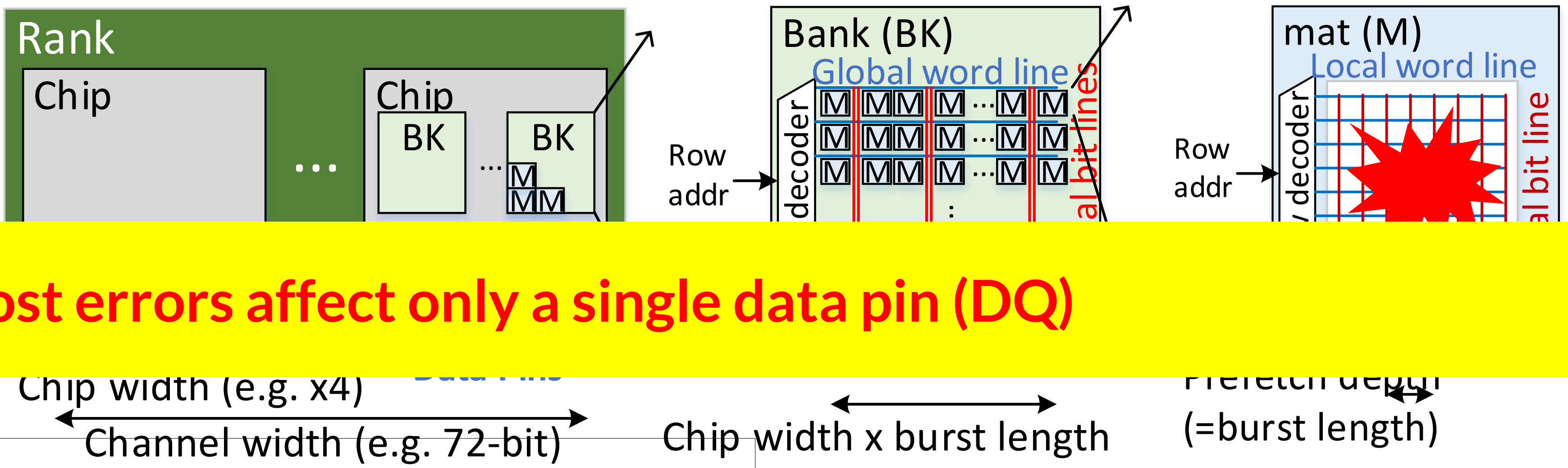


A miscorrected 3-symbol error

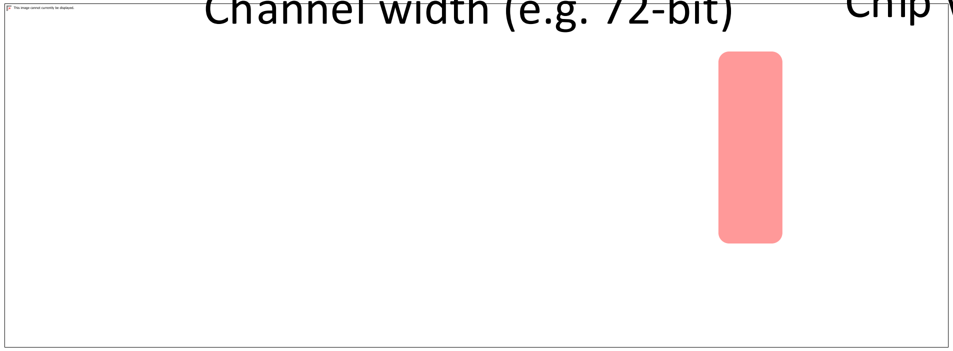
An undetectable 4-symbol error

DRAM error patterns are not random

DRAM internal structure

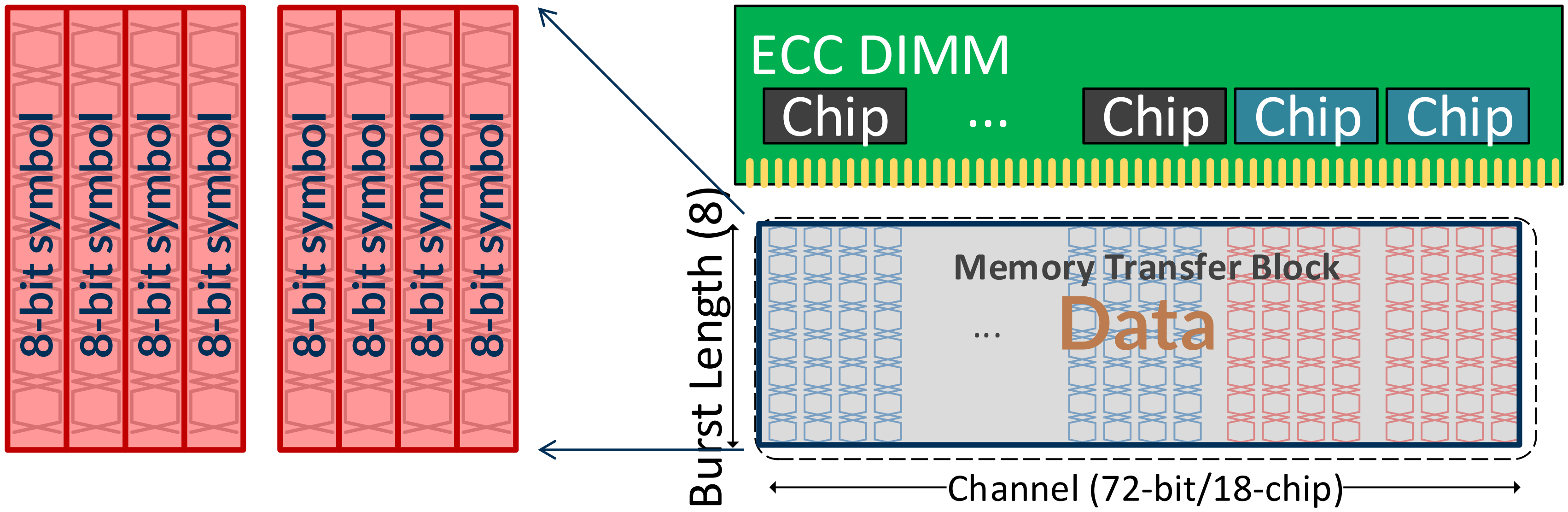


Most errors affect only a single data pin (DQ)



Bamboo ECC (J. Kim et al. HPCA'15)

ECC layout change



Per-pin symbol

➔ Aligned to frequent per-pin errors

Larger codeword

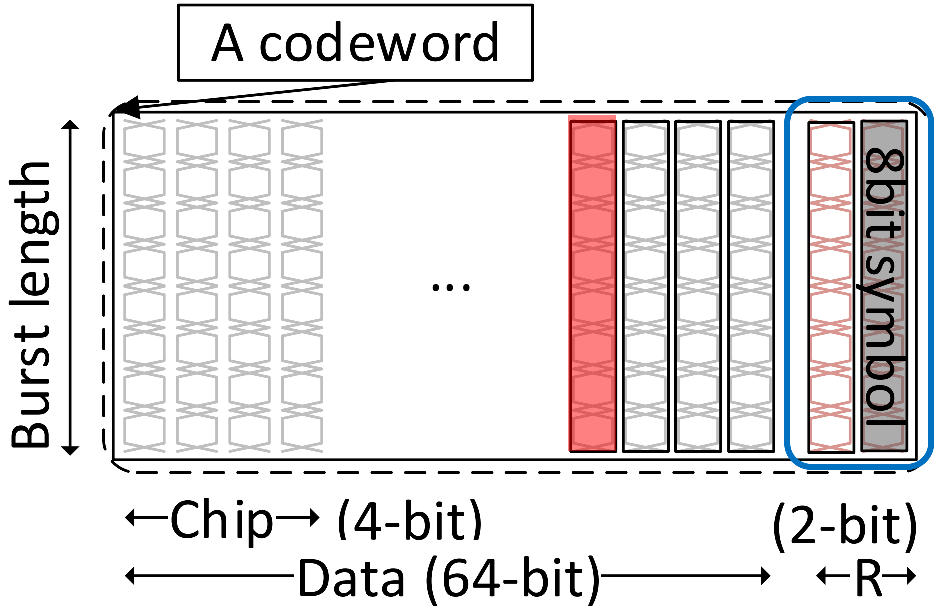
➔ Larger code distance

A family of codes

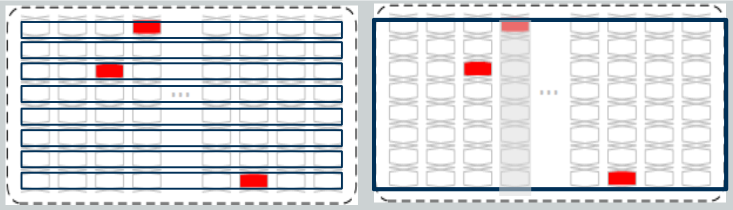
First Bamboo - Single Pin Correcting (SPC)

Corrects 1 pin error
- Using **2** pin redundancy

Compared to SEC-DED
- 1/4 redundancy (3.1% vs. 12.5%)
- Fewer uncorrectable errors from fewer raw errors



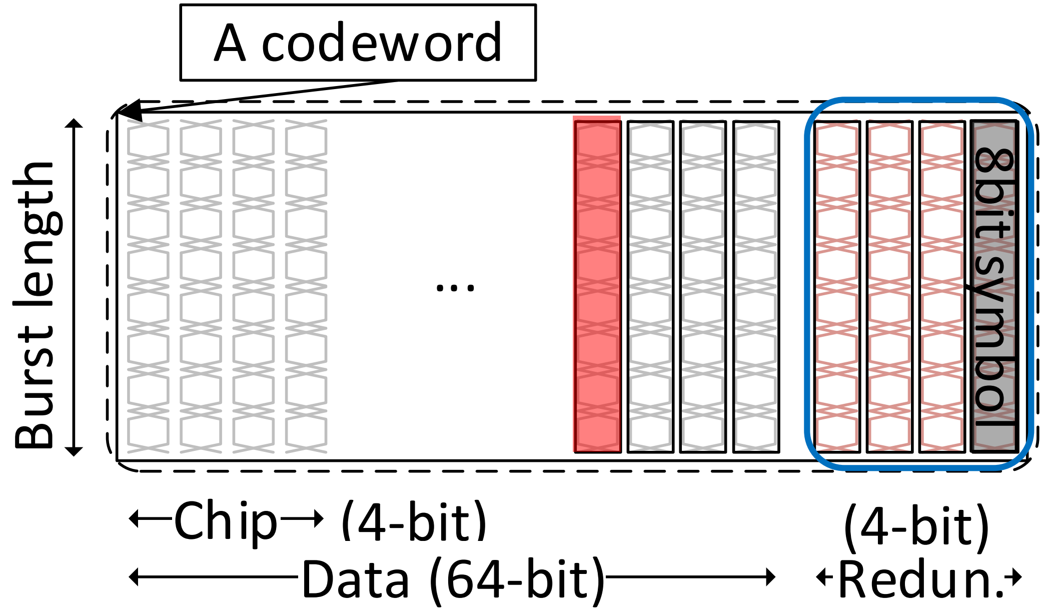
ECC is tricky: SPC vs. SEC-DED

	Which one is better?	Note
Correction probability (A)	**SEC-DED**	 SEC-DED SPC
Raw error rate (B)	**SPC**	~91% (16.5 chips vs. 18 chips)
Overall: uncorrected errors (B x (1-A))	**SPC**	~95%

Single Pin Correcting – Triple Pin Detecting

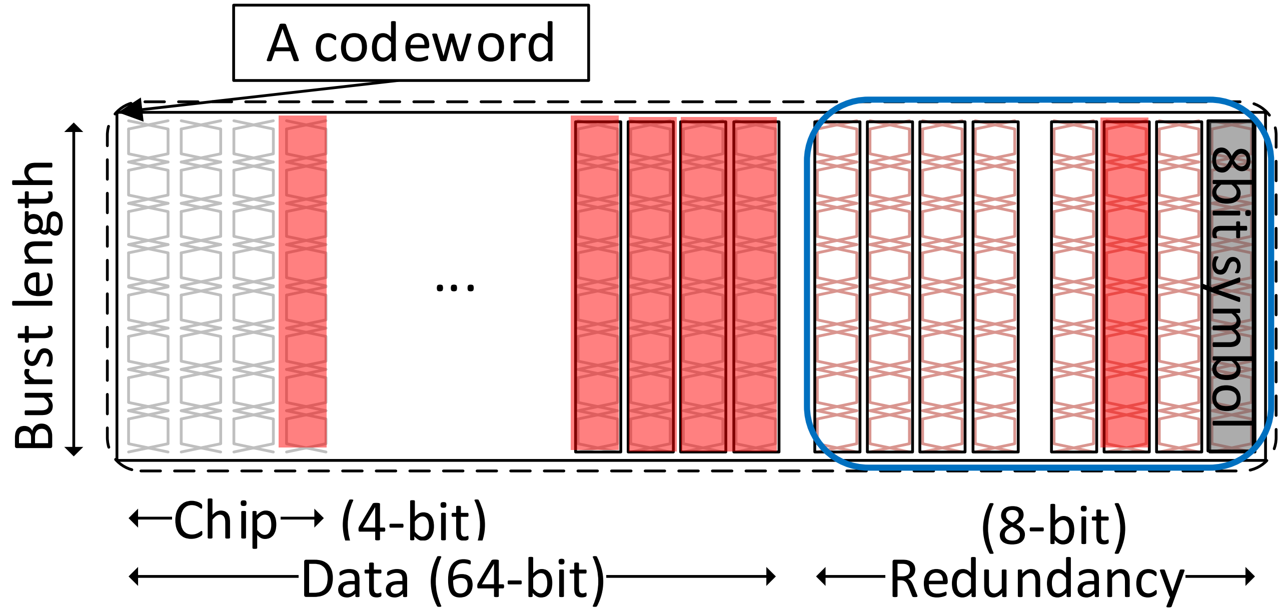
(SPC-TPD)

- Corrects 1 pin error / detects up to 3 pin errors
  - Using **4** pin redundancy
- Compared to SEC-DED
  - 1/2 redundancy (6.3%)
  - Fewer uncorrectable errors
  - Safer detection (0.0004% SDC vs. up to 51%)



Quadruple Pin Correcting (QPC/4PC)

- Corrects one x4 chip error
  - Using **8** pin redundancy
- Compared to AMD Chipkill
  - Same redundancy (8 pins / 12.5%)
  - Stronger correction (e.g. two concurrent pin errors)
  - Safer detection (99.9999%)



But, always tradeoffs …

- Bamboo correction somewhat more complex
- But some other benefits …

Some tricks of the trade

1. Know your faults and errors

– And the difference between them

2. Erasures are your friend

– If you know what’s broken, no need to try and fix it

2*corrections + erasures < code-distance

Fine-grained Retirement using Bamboo

Permanent faults

- 71% of faults / 99.9% of errors
- Common solution: **retirement**

Graceful downgrade of Bamboo family

- Can retire up to 6 sequential pin faults

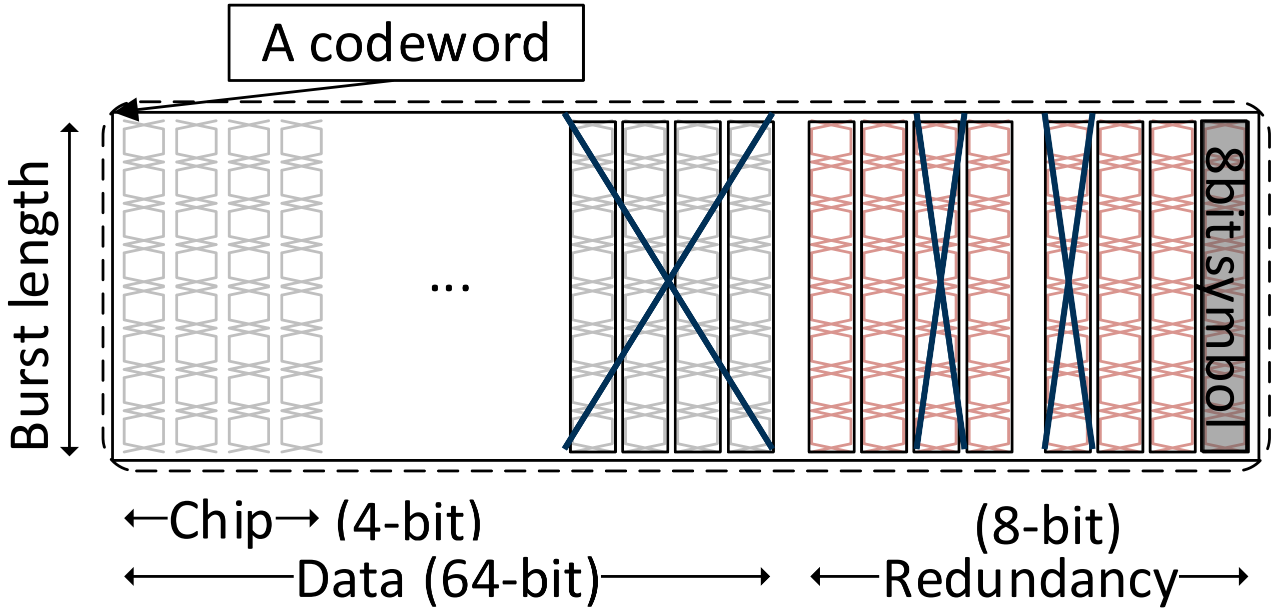
QPC (8-pin redun.)

↓

SPC-TPD (4-pin)

↓

SPC (2-pin)



3. Multi-tier and concatenated codes

– When one code just isn’t enough

Channel constrains organization

- Granularity
- Chip kill
- …

Split the code up to fit the constraints

Common example:

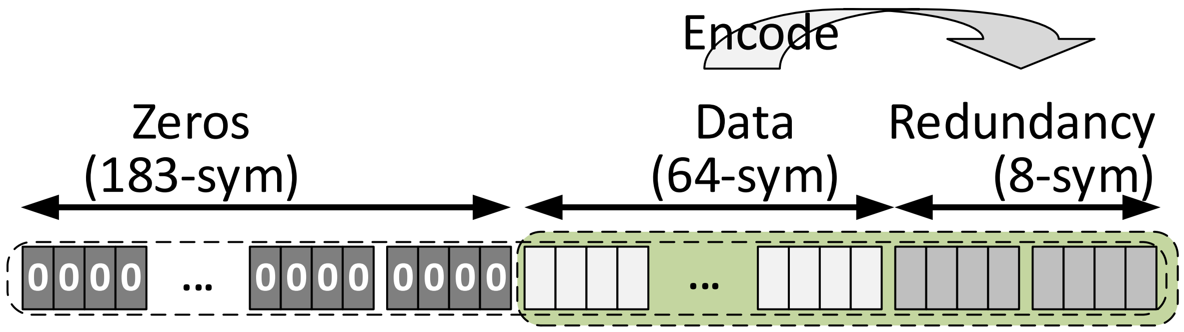
- Detection as the *inner* first-tier code
- Correction as the *outer* second-tier code

- E.g., CRC + parity

4. Shortened codes have "spare capacity"

Maximum CW size of 8-bit RS codes

– 255-symbol
– Only parts are used and the rest are regarded as zeros



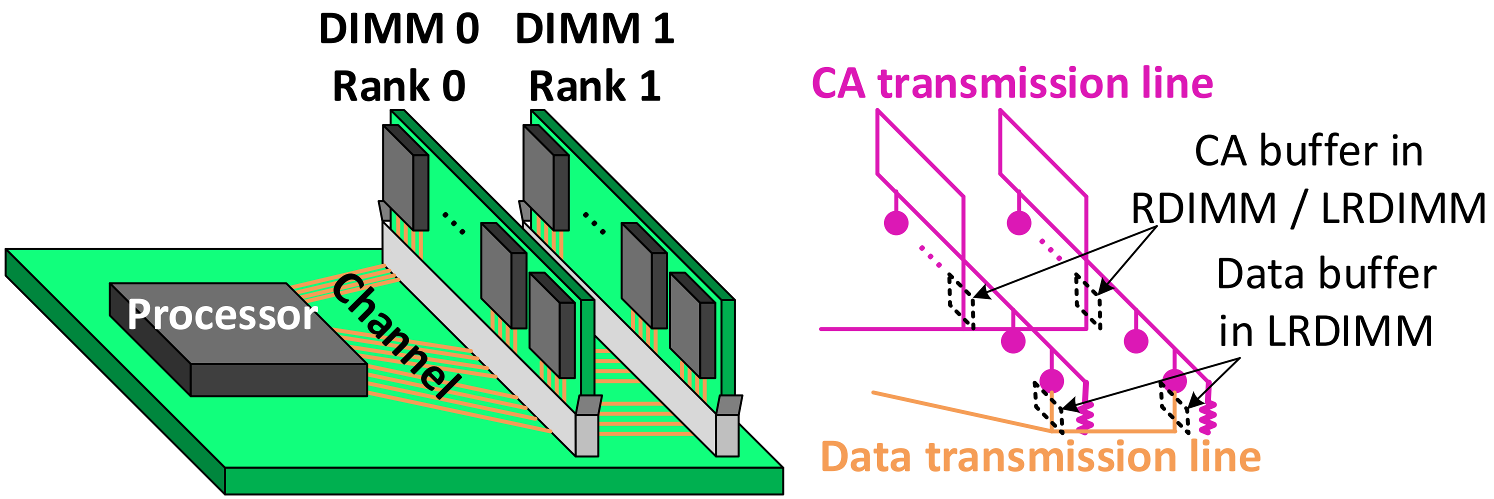
Data ECC using ***shortened code***

- Only 72 symbols out of 255 are used
- Remaining symbols are regarded as zeros

Example: transmission errors

- Other examples do exist

DDR DIMM Topology

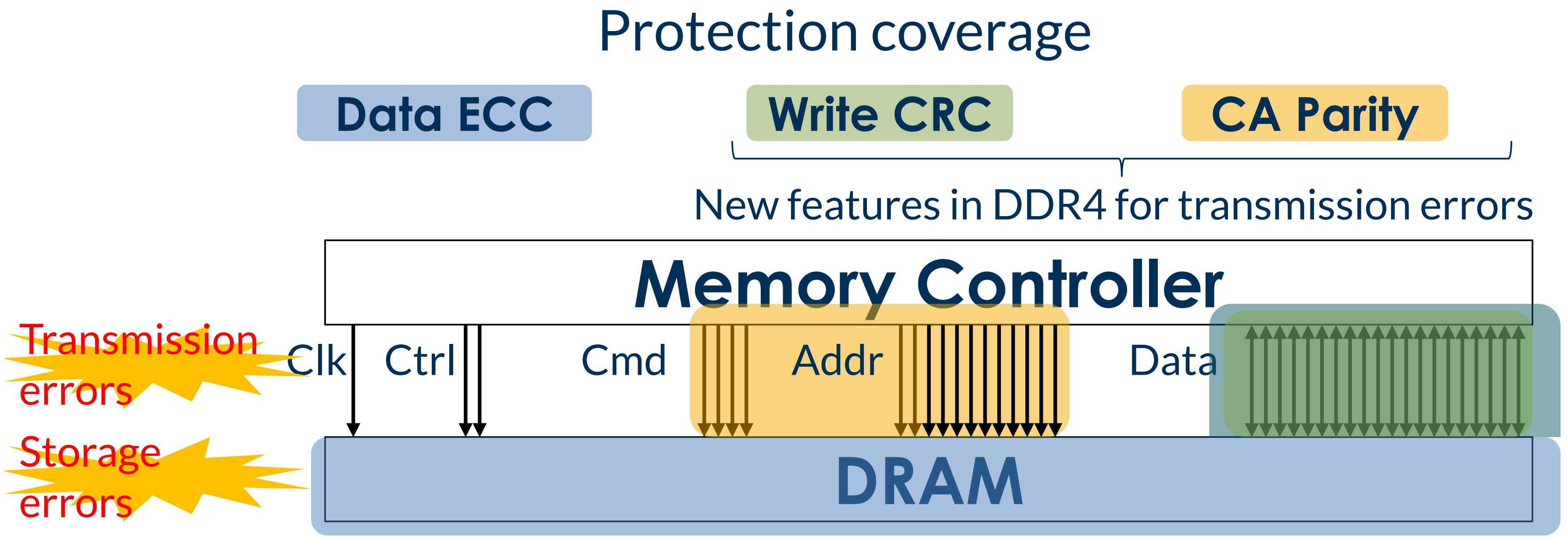


Current Transmission Protection

Data is strongly protected

Address/control

- Cmd/Addr are **weakly protected** by even parity in DDR4
- Clk/Ctrl are **unprotected**

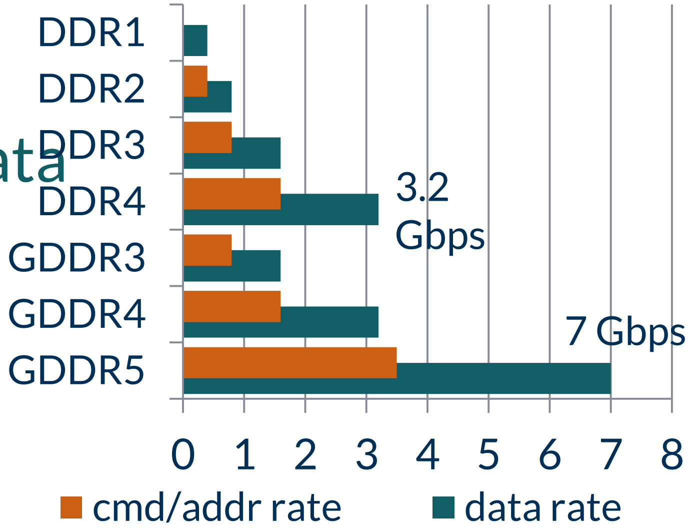


5. Retry often works
   - Repair and correction not always necessary
   - Still need to detect the errors first

CCCA Transmission Errors

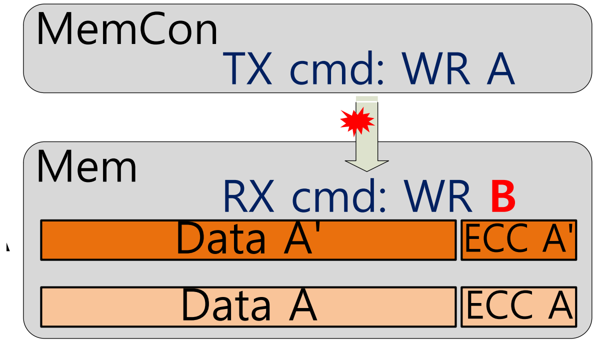
Clk/Ctrl/Cmd/Addr (CCCA)

- Half transfer rate than data
- **More transmission errors** than data
  - Due to DDR3/4 DIMM topology



CCCA error example

Write address error



@ address A: **obsolete CW**

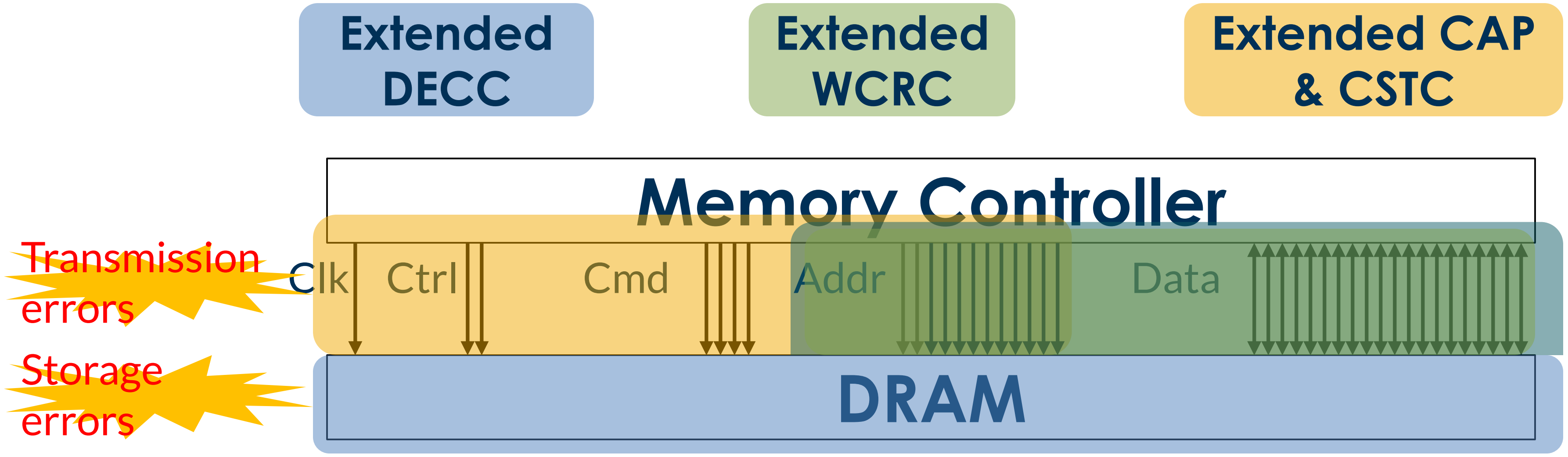
@ address B: **overwritten by another CW**

All-Inclusive ECC [J. Kim et al. ISCA'16]

Augments current protection schemes
- No extra storage or pins

Strong protection against CCCA errors
- Virtually 100% detection

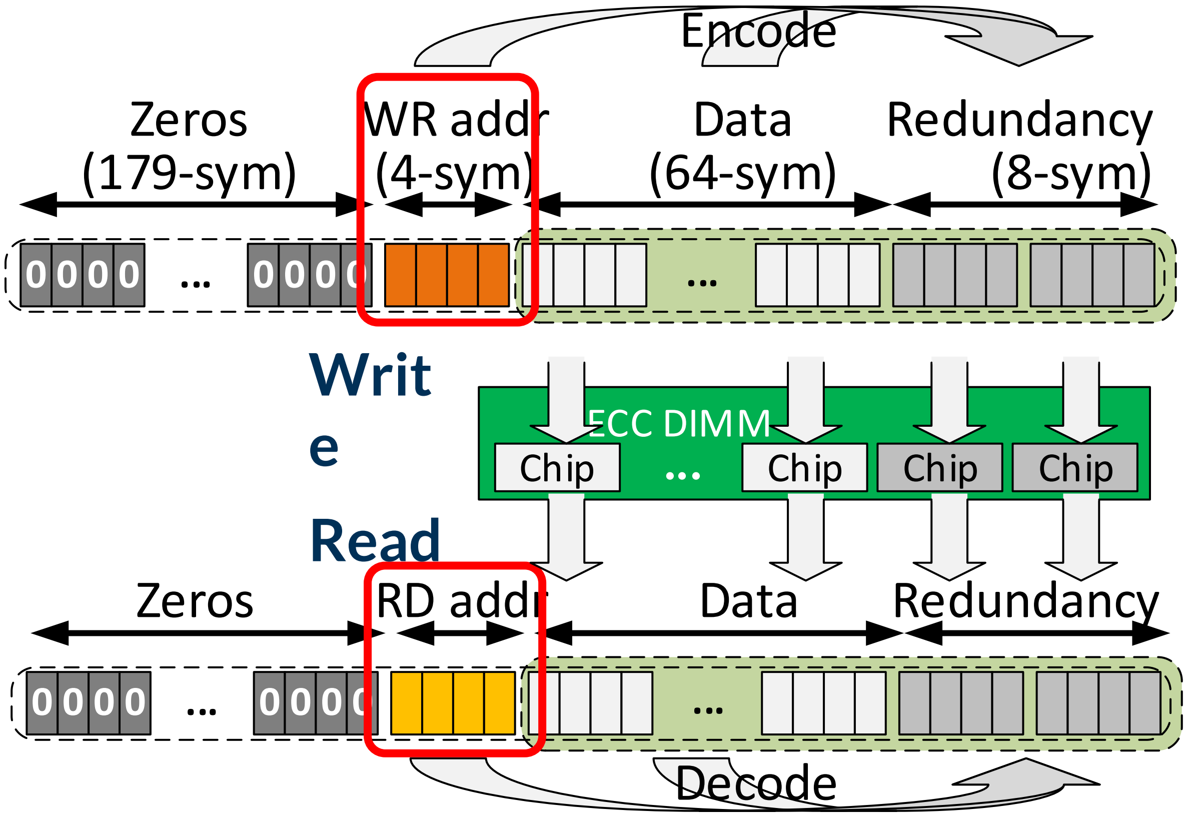
Protection coverage



Address Protection – eDECC

Extended Data ECC (**eDECC**)

- Augment DECC to protect (RD/WR) address
- ECC Redundancy over data and **address**

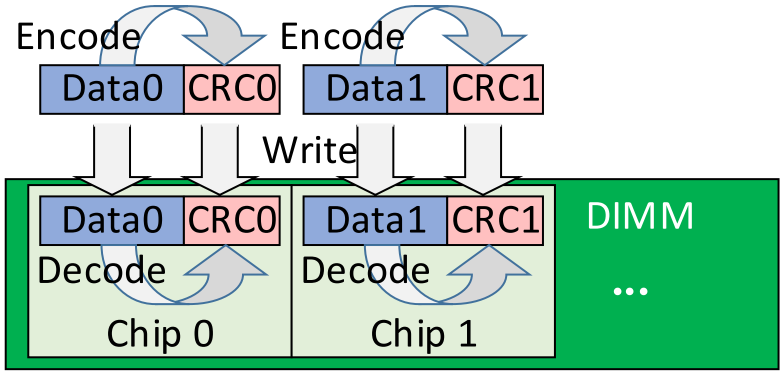


Address Protection – eWCRC

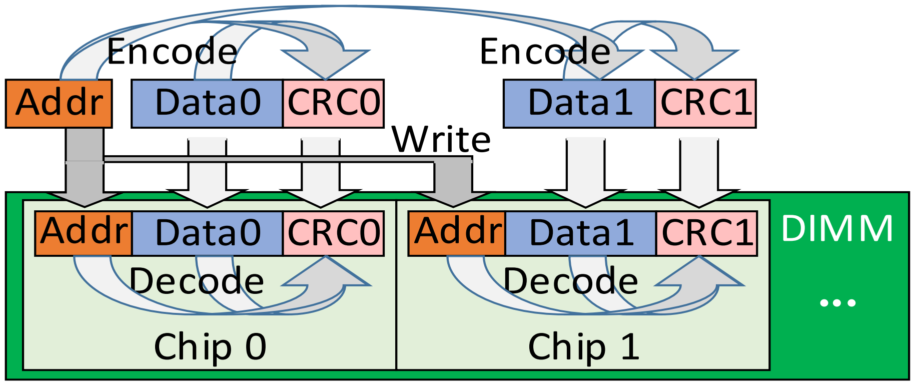
Pitfall of eDECC
- Late detection of WR address error

Extended Write CRC (**eWCRC**)
- Augment WCRC to detect WR address error **early**
- 8-bit CRC over data and address

Write CRC



Extended Write CRC



Clk/Ctrl/Cmd Protection

- Command State & Timing Checker (CSTC)
  - Detects errors by tracking DRAM state and timing
- Extended CA Parity (eCAP)
  - Detects missing write by tracking sent/received cmds
  - Even-parity over {WrCmdToggle, CA}

Command error detection mechanism

Command	Error Type		
	Extra	Missing	Timing
Activate	CSTC	CSTC	CSTC
Precharge	CSTC	CSTC	CSTC
Read	eDECC	eDECC	CSTC
Write	eWCRC	eCAP	CSTC
Refresh	NoErr		~~CSTC~~

May not affect correctness

Memory Repair

Coarse-grained

MEMORY REPAIR MECHANISM

Node/Channel/Rank Retirement

Node retirement

- Simple, and effective when
  - Fault rate leading to this is low enough
  - No critical data is lost
- Challenges
  - Very high overhead
  - Specific node's availability can be critical

Channel/Rank retirement

- Reduce memory capacity and possibly bandwidth
- Impact performance and power efficiency

DRAM Device Retirement

Utilize a part of ECC redundancy

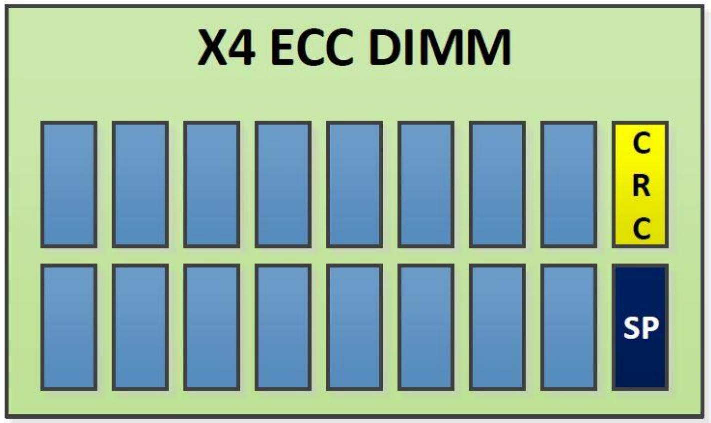
- Bit-steering of IBM's Memory ProteXion
- Bamboo ECC[1]
- Intel's DDDC (*Double Device Data Correction*)

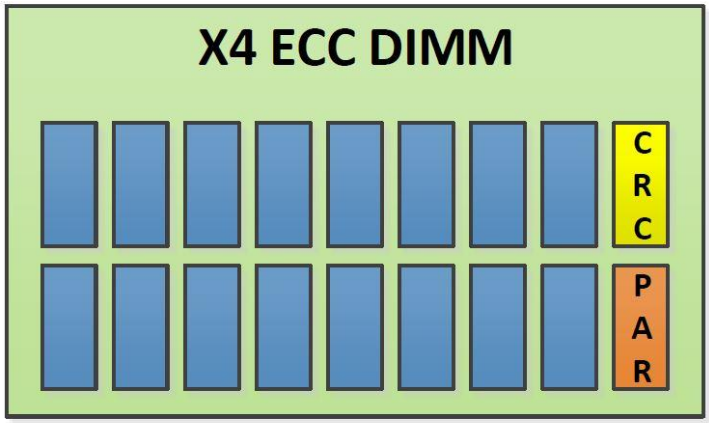
[1] Jungrae Kim, Michael Sullivan, and Mattan Erez,, "Bamboo ECC: Strong, Safe, and Flexible Codes for Reliable Computer Memory", HPCA 2015

DRAM Device Retirement

Utilize a part of ECC redundancy

- Bit-steering of IBM's Memory ProteXion
- Bamboo ECC[1]
- **Intel's DDDC (*Double Device Data Correction*)**





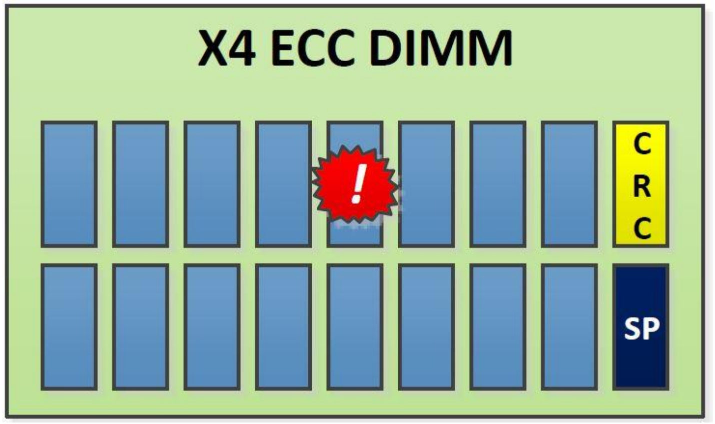
*** SP: spare, PAR: parity**

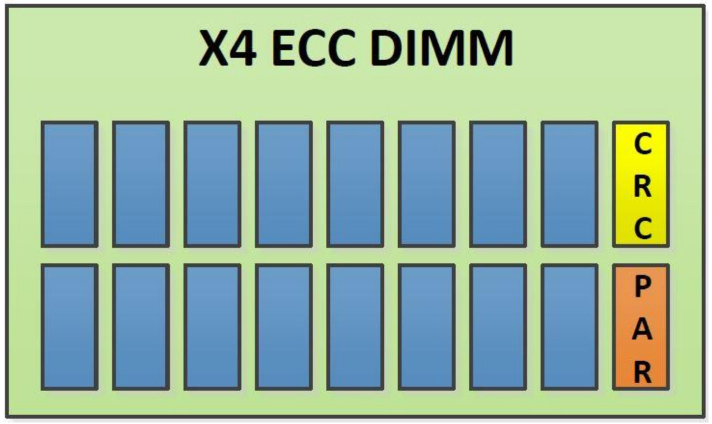
[1] Jungrae Kim, Michael Sullivan, and Mattan Erez,, "Bamboo ECC: Strong, Safe, and Flexible Codes for Reliable Computer Memory", HPCA 2015

DRAM Device Retirement

Utilize a part of ECC redundancy

- Bit-steering of IBM's Memory ProteXion
- Bamboo ECC[1]
- **Intel's DDDC (*Double Device Data Correction*)**





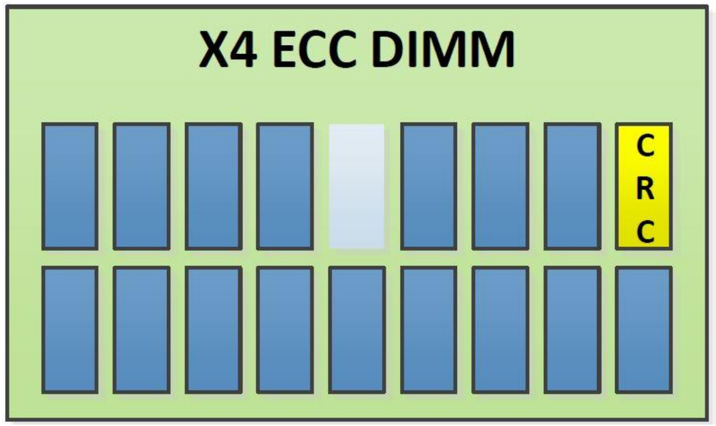
*** SP: spare, PAR: parity**

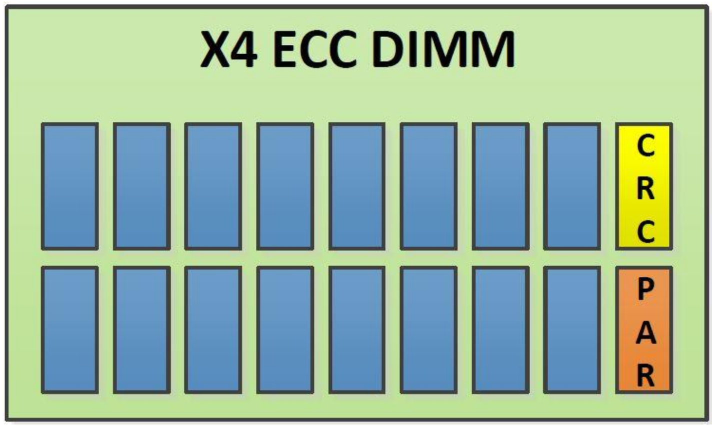
[1] Jungrae Kim, Michael Sullivan, and Mattan Erez,, "Bamboo ECC: Strong, Safe, and Flexible Codes for Reliable Computer Memory", HPCA 2015

DRAM Device Retirement

Utilize a part of ECC redundancy

- Bit-steering of IBM's Memory ProteXion
- Bamboo ECC[1]
- **Intel's DDDC (*Double Device Data Correction*)**





*** SP: spare, PAR: parity**

[1] Jungrae Kim, Michael Sullivan, and Mattan Erez,, "Bamboo ECC: Strong, Safe, and Flexible Codes for Reliable Computer Memory", HPCA 2015

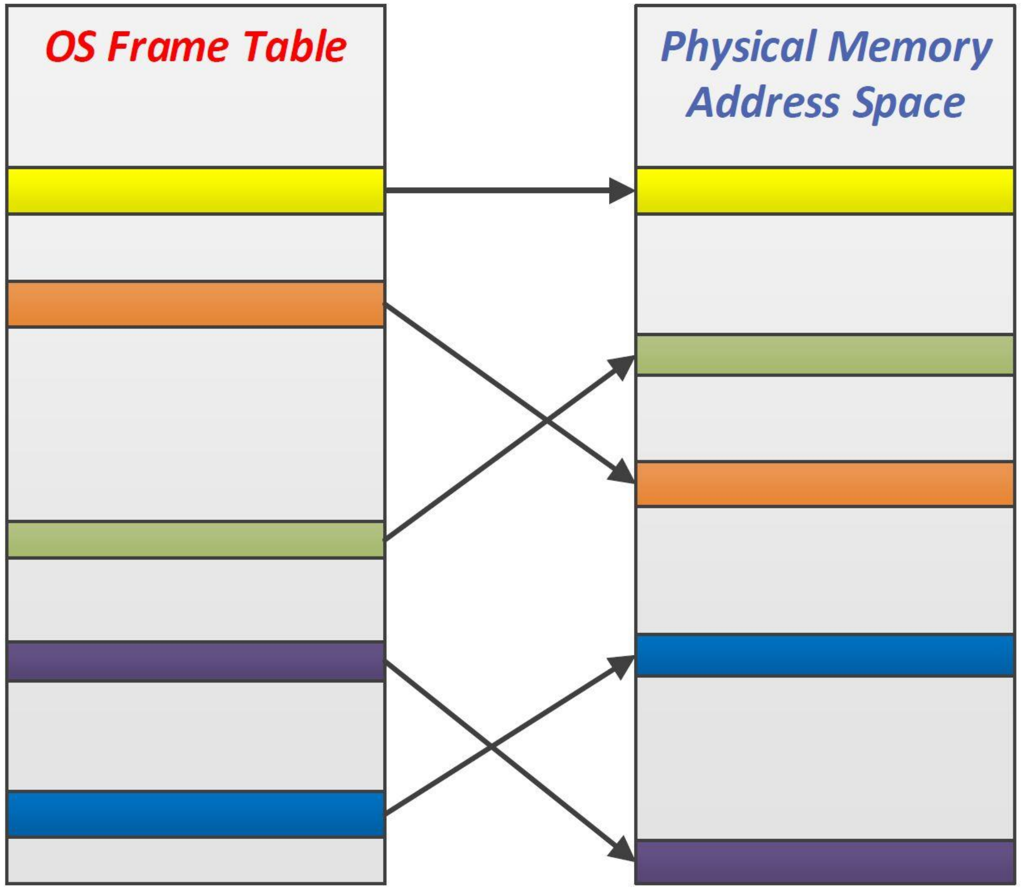
Fine-grained

MEMORY REPAIR MECHANISMS

Memory Frame Retirement (Virtual Memory)

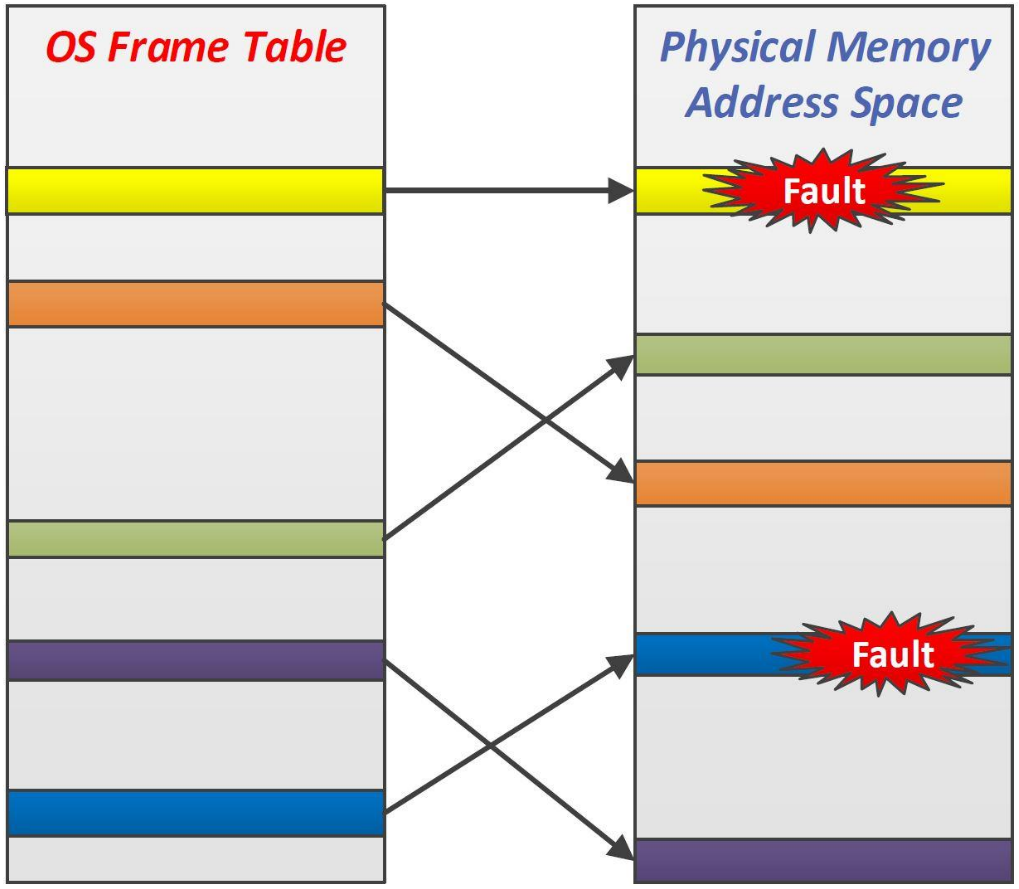
Exploit page-based virtual memory support

– OS (or device driver) retires frames affected by faults



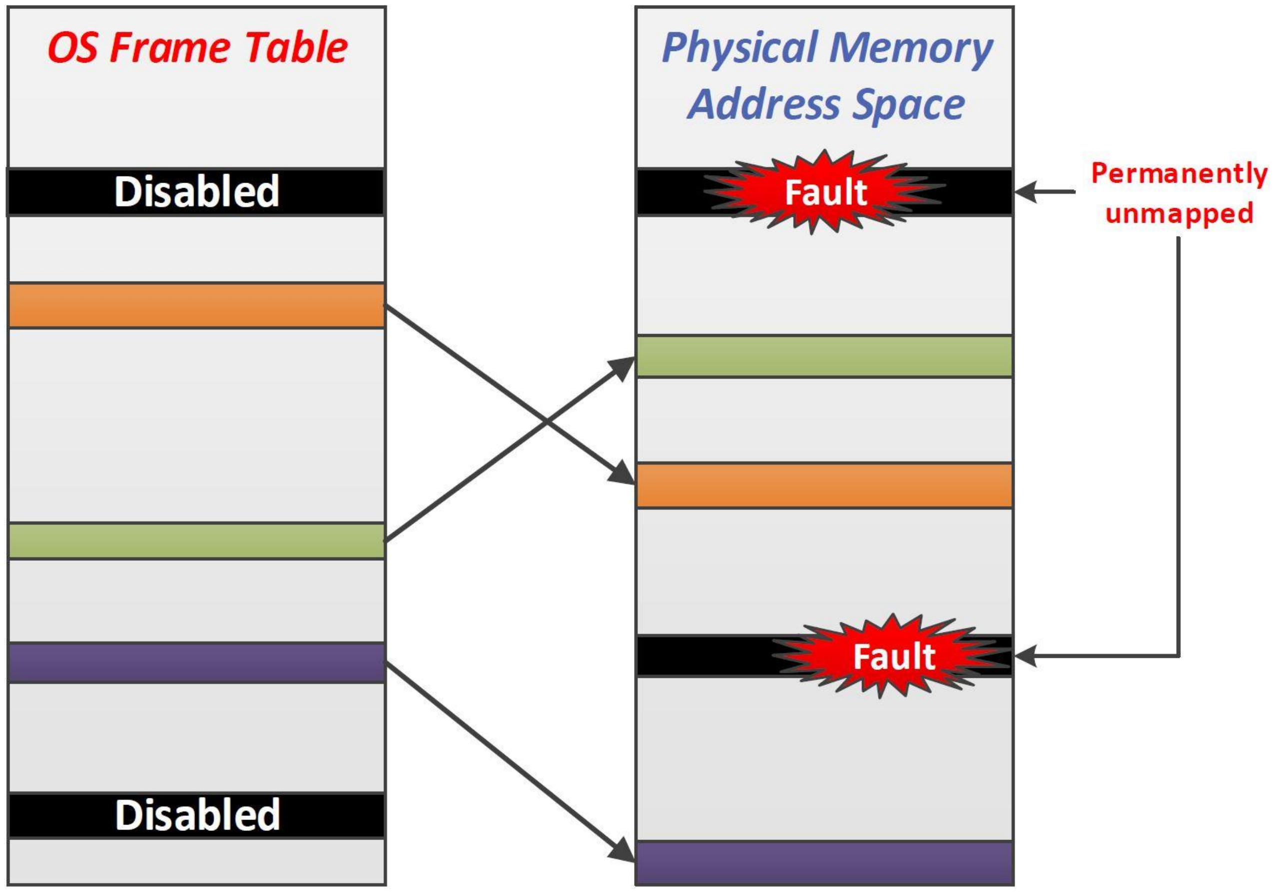
Memory Frame Retirement (Virtual Memory)

- Exploit page-based virtual memory support
  - OS (or device driver) retires frames affected by faults



Memory Frame Retirement (Virtual Memory)

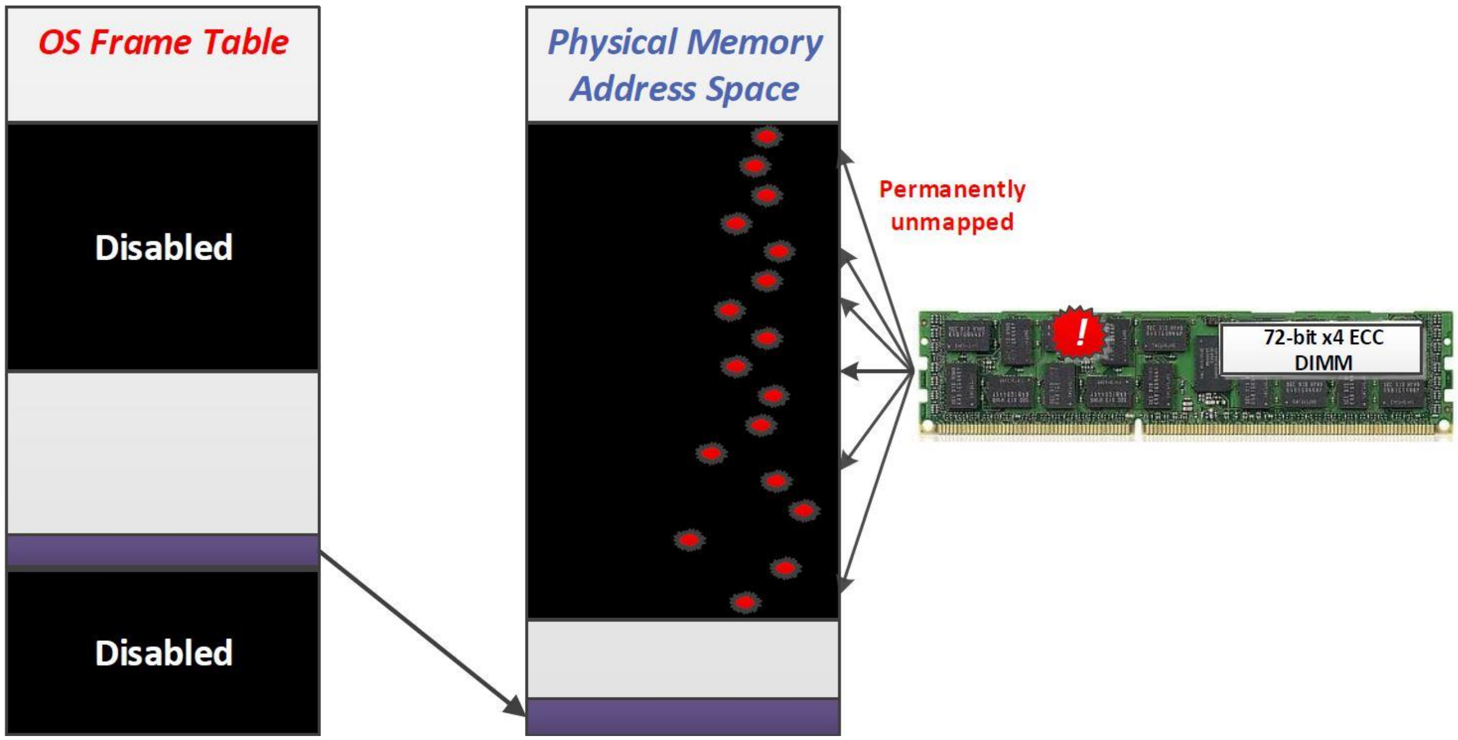
- Exploit page-based virtual memory support
  - OS (or device driver) retires frames affected by faults



Memory Frame Retirement (Virtual Memory)

Challenge 1

- Physical to DRAM address mapping increases footprint of many DRAM faults

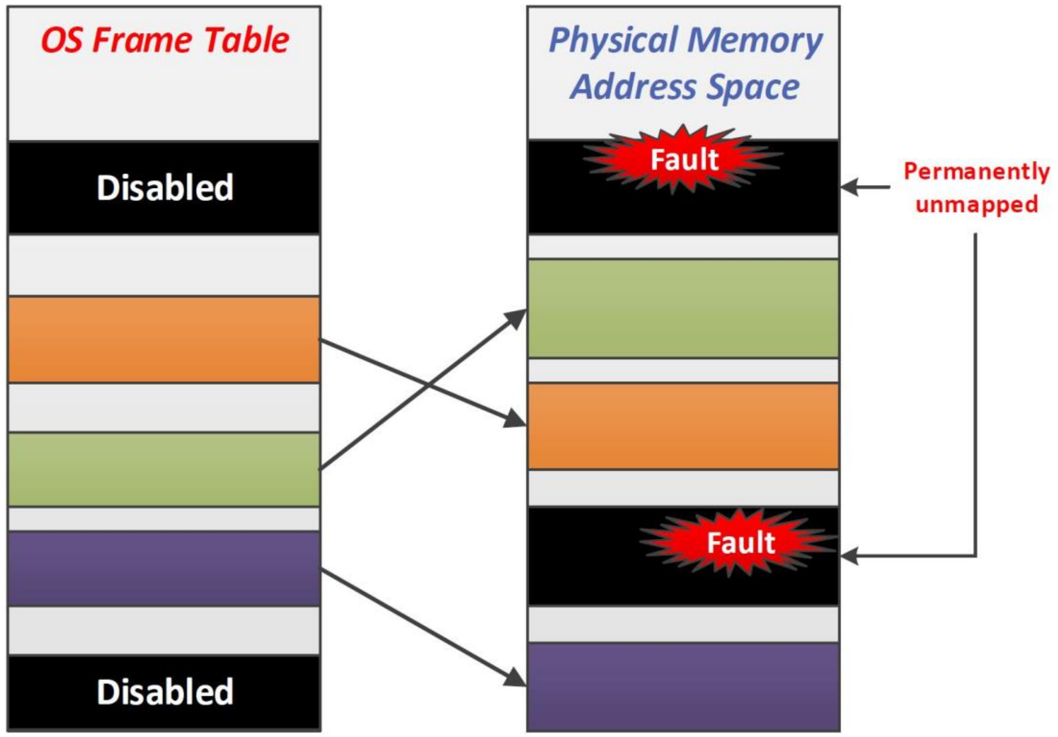


E.g. Single Column faults in DRAM span many OS pages

Memory Frame Retirement (Virtual Memory)

Challenge 2

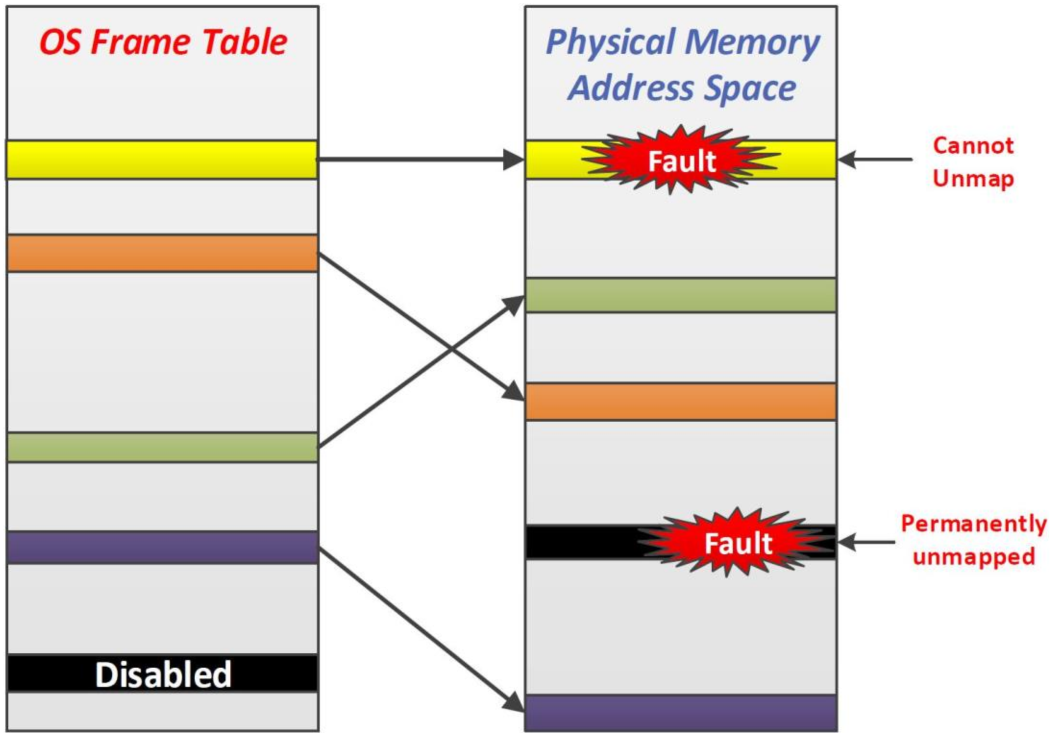
– Huge-pages or segmentation



Memory Frame Retirement (Virtual Memory)

Challenge 3

- May be limited to a specific address space
  - Some OS components and peripheral devices do not fully utilize virtual memory facilities (e.g. IBM's AIX)



Remapping With Redundancy

- Row/Column sparing
  - Available during manufacture/test/integration time
  - PPR (*Post Package Repair*) in DDR4 and LPDDR4 repairs a faulty row in-the-fields
    - Repair at most one row per bank or bank group
    - Repair one-time only (fuse)
- Structure external to arrays
  - Either adding storage (e.g. Fault Cache[1] and CiDRA[2]) or using a fraction of DRAM storage (e.g. ArchShield[3])
  - Add substantial cost
    - Each device or DIMM requires redundant storage and remapping logic

[1] A. Tanabe et al., "A 30-ns 64-Mb DRAM with built-in self-test and self-repair function," JSSC 1992
[2] Y. H. Son et al., "CiDRA: A cache-inspired DRAM resilience architecture", HPCA 2015
[3] P. J. Nair et al., "ArchShield: architectural framework for assisting DRAM scaling by tolerating high error rates," ISCA 2013

Remapping With Redundancy

- Row/Column sparing
  - Available during manufacture/test/integration time
  - PPR (*Post Package Repair*) in DDR4 and LPDDR4 repairs a faulty row in-the-fields
    - Repair at most one row per bank or bank group
    - Repair one-time only (fuse)
- Structure external to arrays
  - Either **adding storage (e.g. Fault Cache[1] and CiDRA[2])** or using a fraction of DRAM storage (e.g. ArchShield[3])
  - Add substantial cost
    - Each device or DIMM requires redundant storage and remapping logic

[1] A. Tanabe et al., "A 30-ns 64-Mb DRAM with built-in self-test and self-repair function," JSSC 1992
[2] Y. H. Son et al., "CiDRA: A cache-inspired DRAM resilience architecture", HPCA 2015
[3] P. J. Nair et al., "ArchShield: architectural framework for assisting DRAM scaling by tolerating high error rates," ISCA 2013

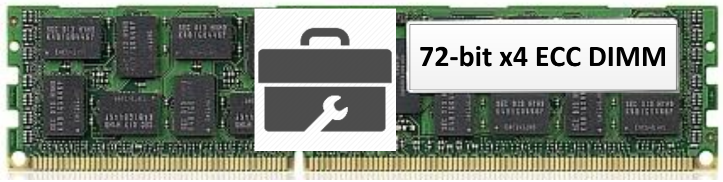
Remapping With Redundancy

Row/Column sparing

- Available during manufacture/test/integration time
- PPR (*Post Package Repair*) in DDR4 and LPDDR4 repairs a faulty row in-the-fields
  - Repair at most one row per bank or bank group
  - Repair one-time only (fuse)

Structure external to arrays

- Either adding storage (e.g. Fault Cache[1] and CiDRA[2]) or **using a fraction of DRAM storage (e.g. ArchShield[3])**
- Add substantial cost
  - Each device or DIMM requires redundant storage and remapping logic



[1] A. Tanabe et al., "A 30-ns 64-Mb DRAM with built-in self-test and self-repair function," JSSC 1992
[2] Y. H. Son et al., "CiDRA: A cache-inspired DRAM resilience architecture", HPCA 2015
[3] P. J. Nair et al., "ArchShield: architectural framework for assisting DRAM scaling by tolerating high error rates," ISCA 2013

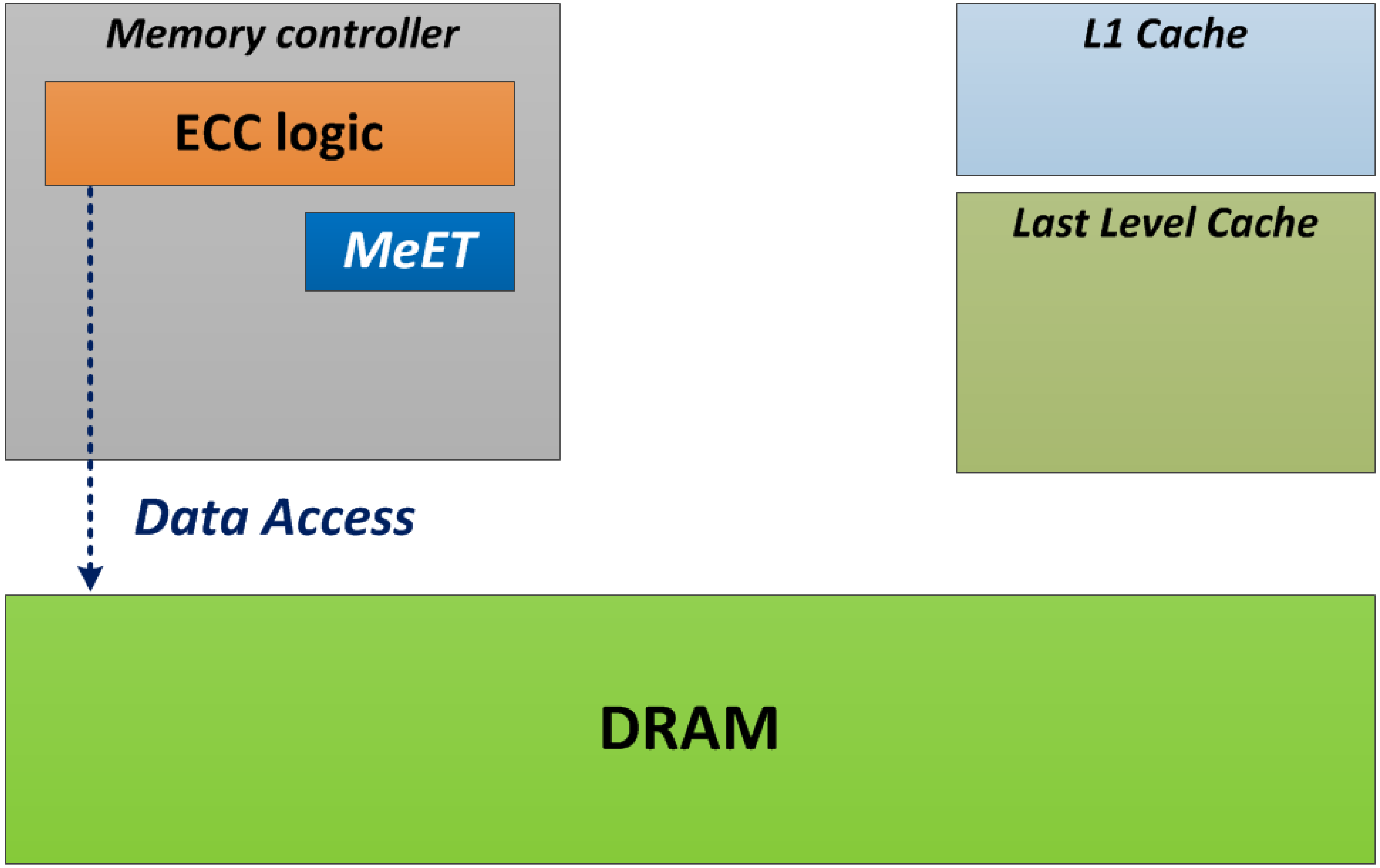
Remapping Without Redundancy

Utilize microarchitectural structures [Kim and Erez, HPCA’15][1]

- Caches as redundant storage
  - A small fraction of LLC is used to remap data from faulty regions in DRAM
- Mechanism is transparent
- Remapping storage size is determined as needed
- Due to limited size of LLC, it should be limited to repair faults that affect only small regions of DRAM

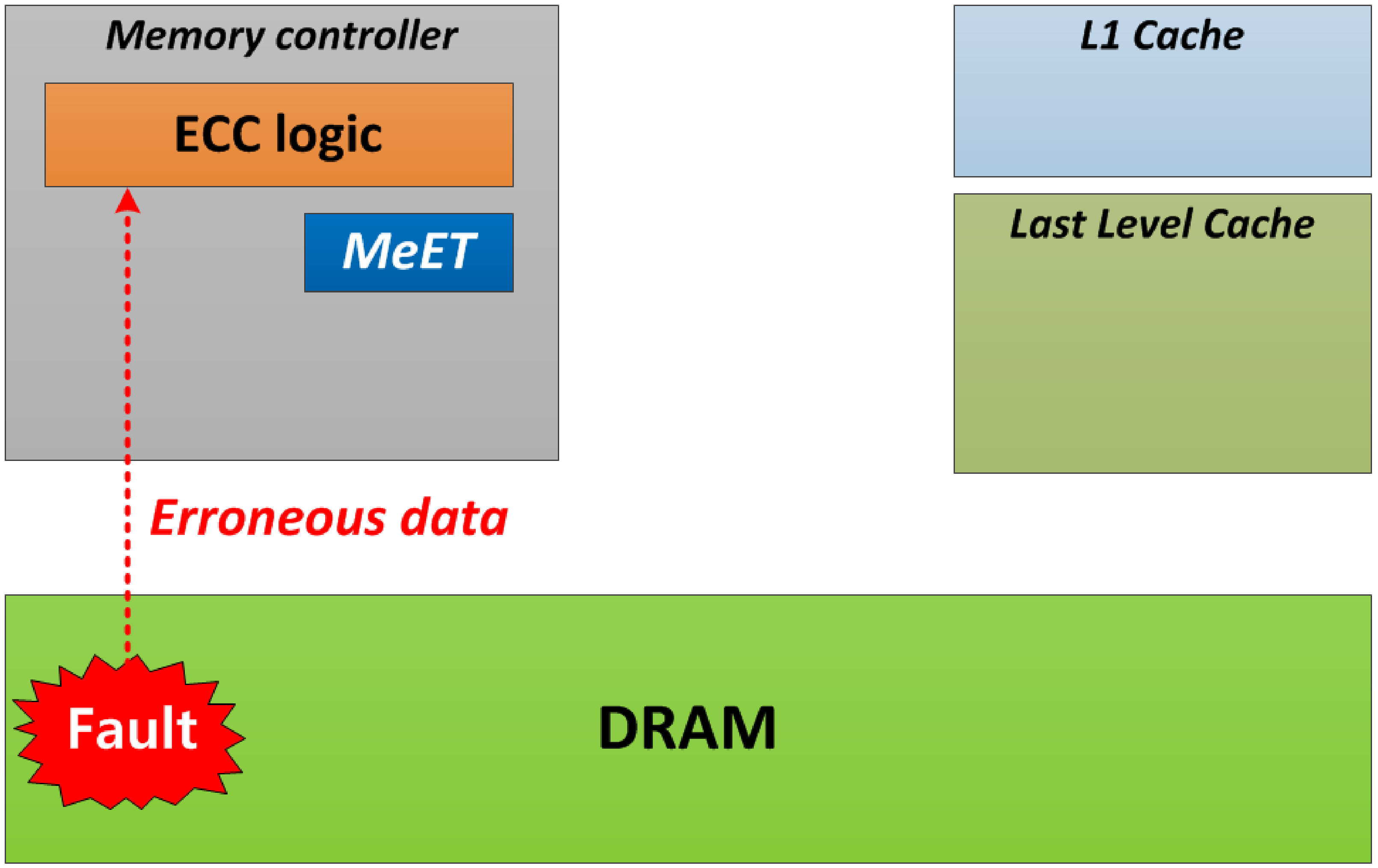
[1] Dong Wan Kim and Mattan Erez, “Balancing reliability, cost, and performance tradeoffs with FreeFault," *HPCA*, 2015

FreeFault Architecture [DW Kim and M. Erez HPCA'15, ISCA'16]



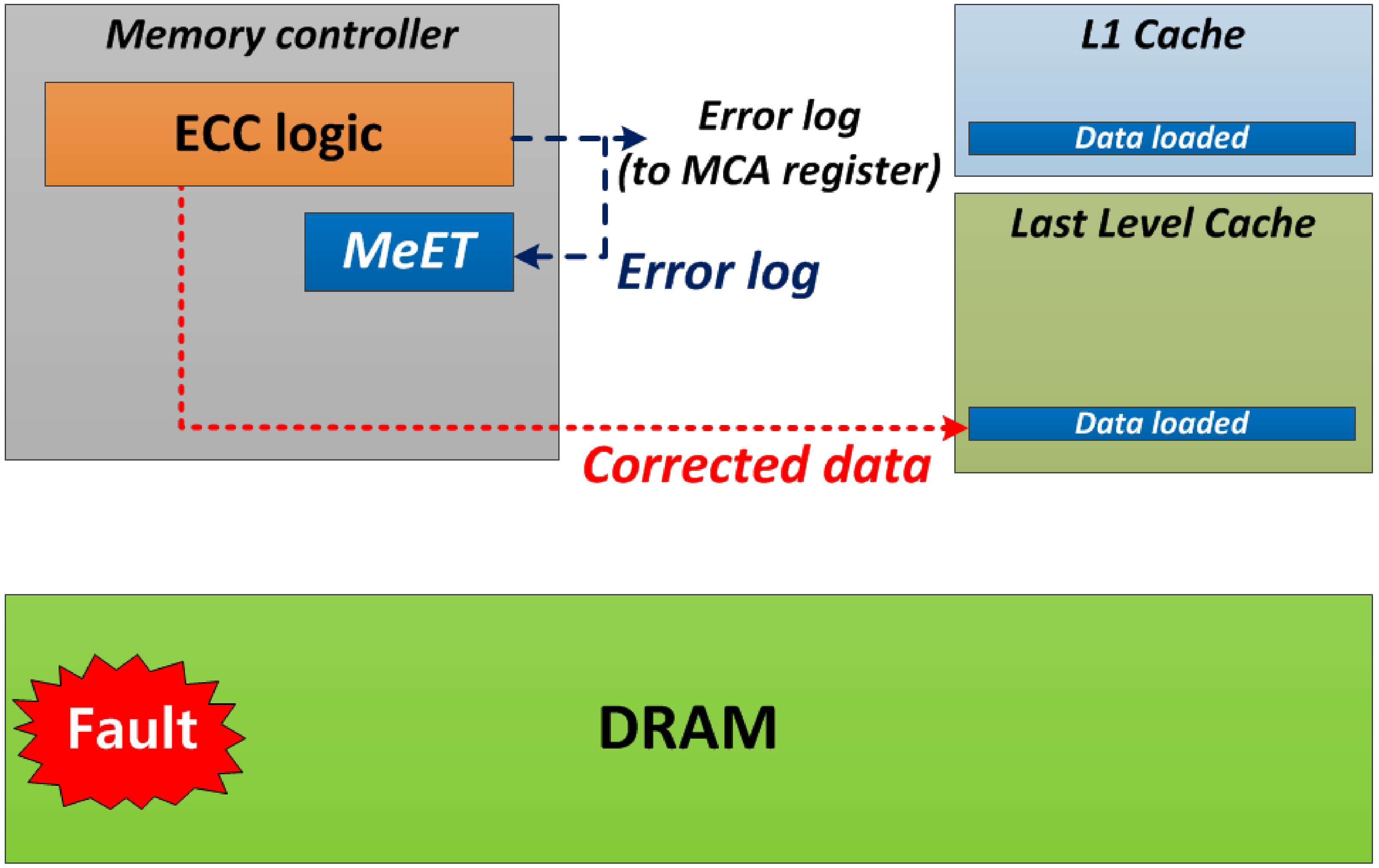
* MeET: Memory Error Tracker

FreeFault Architecture



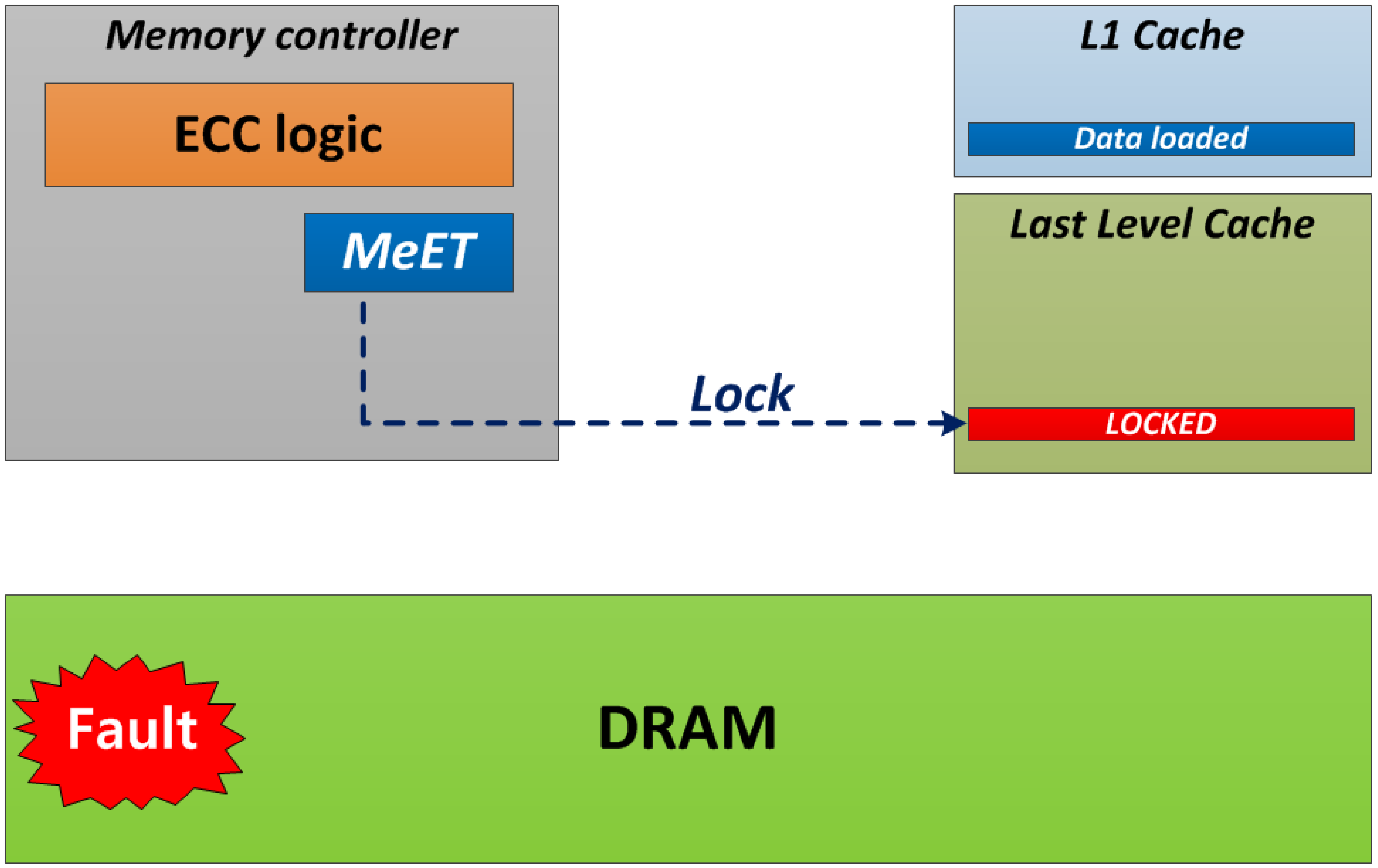
* MeET: Memory Error Tracker

FreeFault Architecture



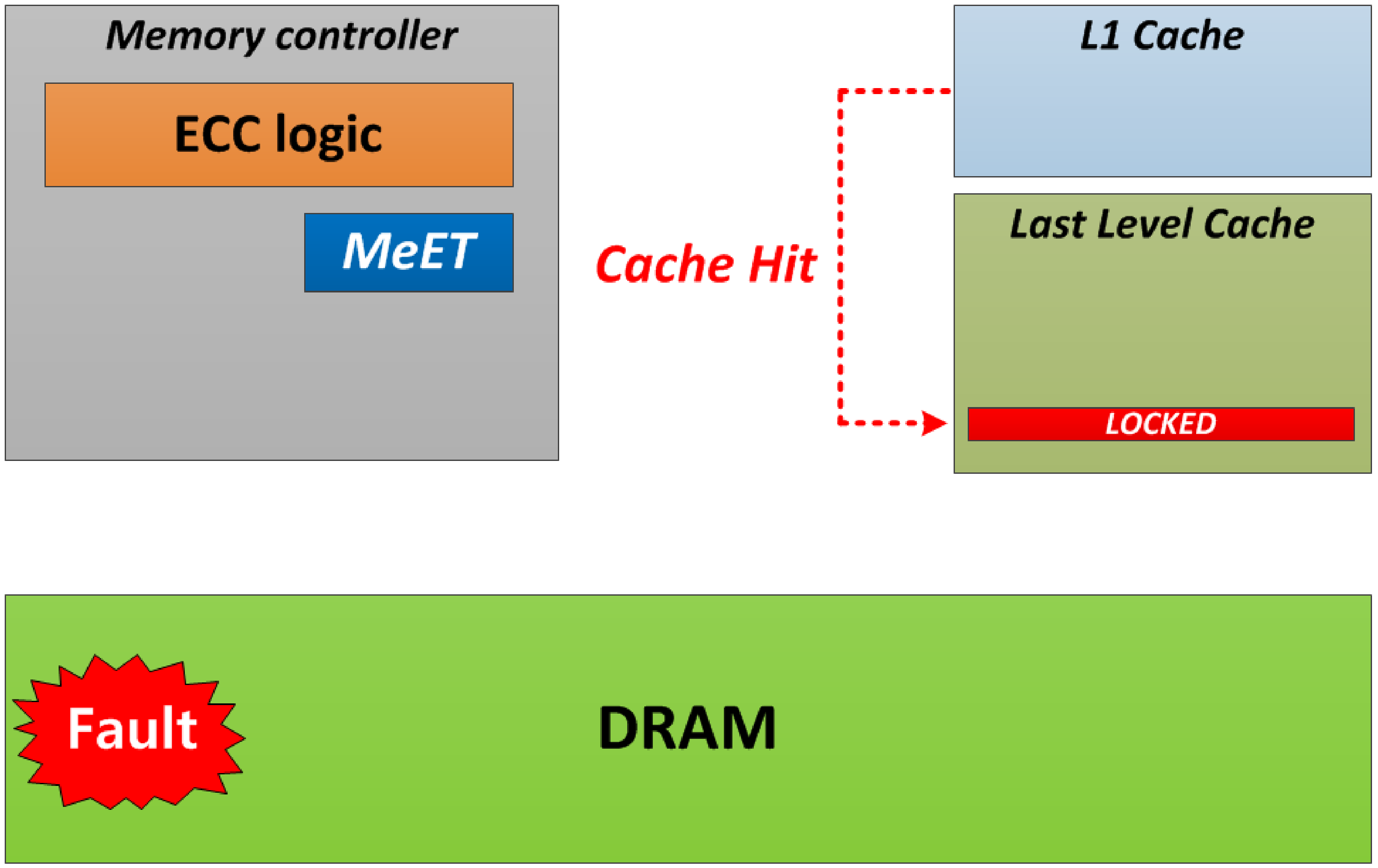
* MeET: Memory Error Tracker

FreeFault Architecture



* MeET: Memory Error Tracker

FreeFault Architecture



* MeET: Memory Error Tracker

7 key takeaway

0. Know the requirements
   - SDCs? Availability? Graceful downgrade? Time-to-replace?
1. Know your faults and errors
   - Transient, intermittent, permanent, single-bit, single-pin, single chip, multi-chip/single-bit, …
   - Inform ECC scheme, concatenation, repair needs, …
     - If a bear is chasing you, you only need to run faster than slowest person – no such thing as perfectly reliable – just reliable enough
2. Erasures are your friend
   - Knowing/assuming where the error is increases code correction capability – significantly reduces redundancy
3. Concatenated/multi-tier codes overcome constraints
   - Separate detection and correction
   - Separate storage
4. Long codes have spare “coverage capacity”
   - Longer symbols and longer codewords reduce SDCs
   - Can add implicit information (e.g., for address error detection)
5. Retry can beat redundancy
   - Transmission and read errors can simply be retried
     - Possibly with different access parameters
6. Repair is very effective and can be very cheap
   - Trade off redundancy of repair with redundancy for ECC

Backup:

DRAM RELIABILITY EVALUATION

Evaluation methodology

System configuration

– 8 DIMMs per node
– SSC (*Single symbol correction*) chipkill level ECC

DRAM reliability metric

– DUE, SDC rate, and number of replaced DIMMs

Fault mode	Transient fault	Permanent fault
Single-bit	14.2	18.6
Single-word	1.4	0.3
Single-row	0.2	8.2
Single-column	1.4	5.6
Single-bank	0.8	10.0
Multiple-bank	0.3	1.4
Multiple-rank	0.9	2.8

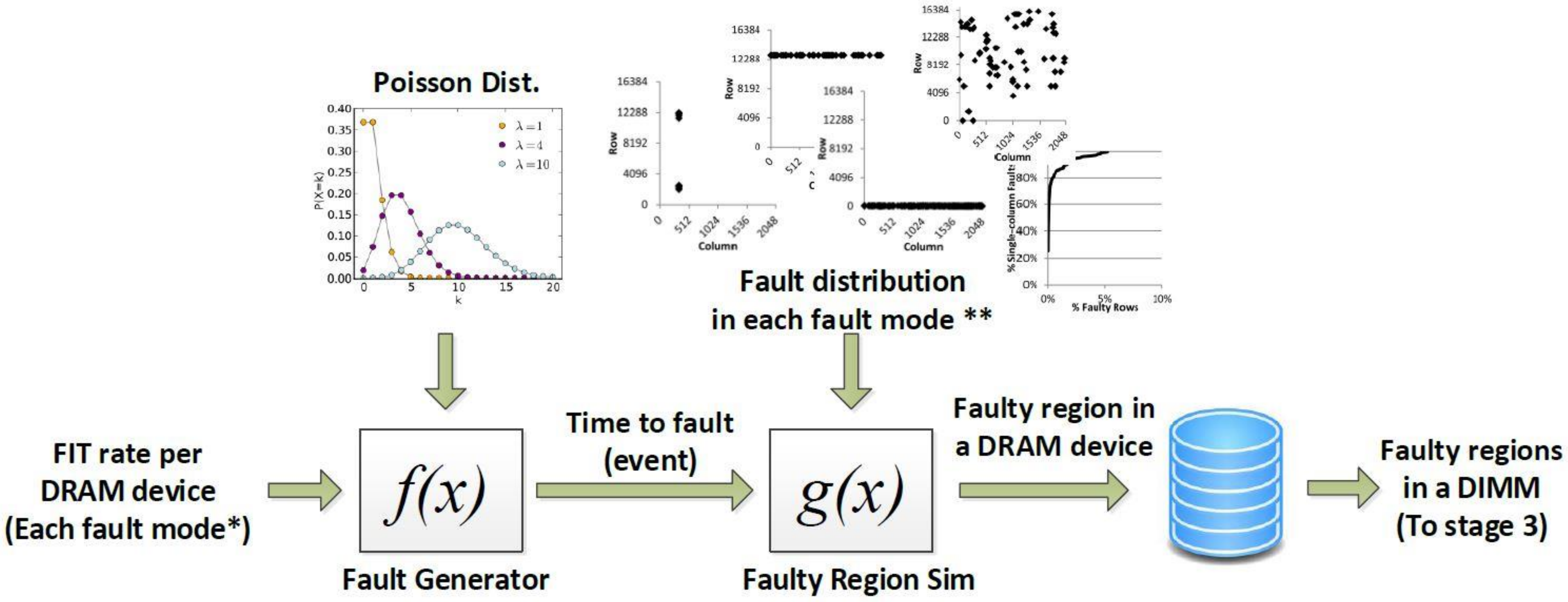
Table. Failure rate of DRAM device (FIT/device) [1]

[1] Vilas Sridharan and Dean Liberty, "A Study of DRAM Failures in the Field", SC 2012

Fault simulation

Multi-stage Monte-Carlo fault simulator

– This is basically FaultSim

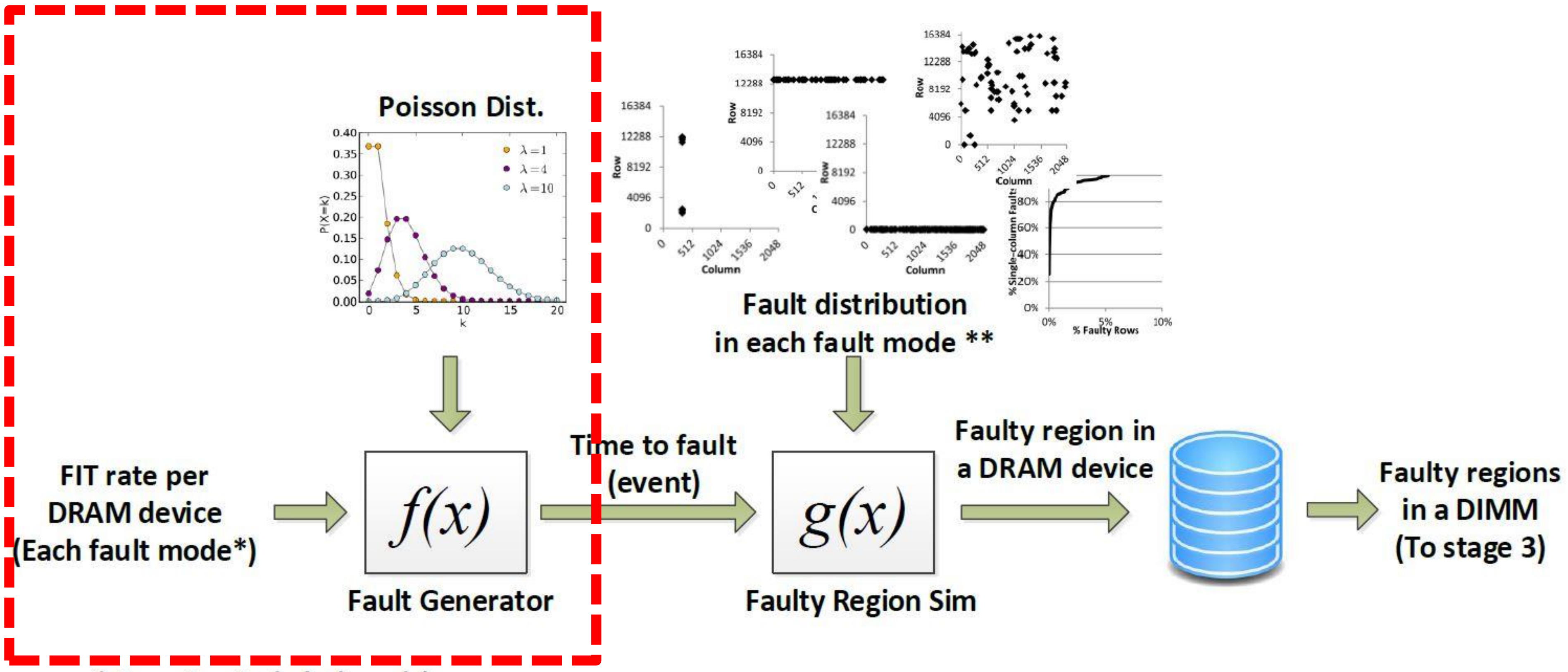

Poisson Dist.
λ=1
λ=4
λ=10
P(X=k)
k
Fault distribution in each fault mode **
Row
Column
% Single-column Faults
% Faulty Rows
FIT rate per DRAM device (Each fault mode*)
f(x)
Fault Generator
Time to fault (event)
g(x)
Faulty Region Sim
Faulty region in a DRAM device
Faulty regions in a DIMM (To stage 3)

** Assuming single fault model*
*** Estimated from previous literatures, e.g. Sridharan et al., A Study of DRAM Failure in the Field, SC 2012*

Fault simulation

Multi-stage Monte-Carlo fault simulator

– **Stage 1: Faults are simulated as a *Poisson process***



Poisson Dist.
P(X=k)
k
λ=1
λ=4
λ=10
Fault distribution in each fault mode **
Row
Column
% Single-column Faults
% Faulty Rows
FIT rate per DRAM device (Each fault mode*)
f(x)
Fault Generator
Time to fault (event)
g(x)
Faulty Region Sim
Faulty region in a DRAM device
Faulty regions in a DIMM (To stage 3)

** Assuming single fault model*

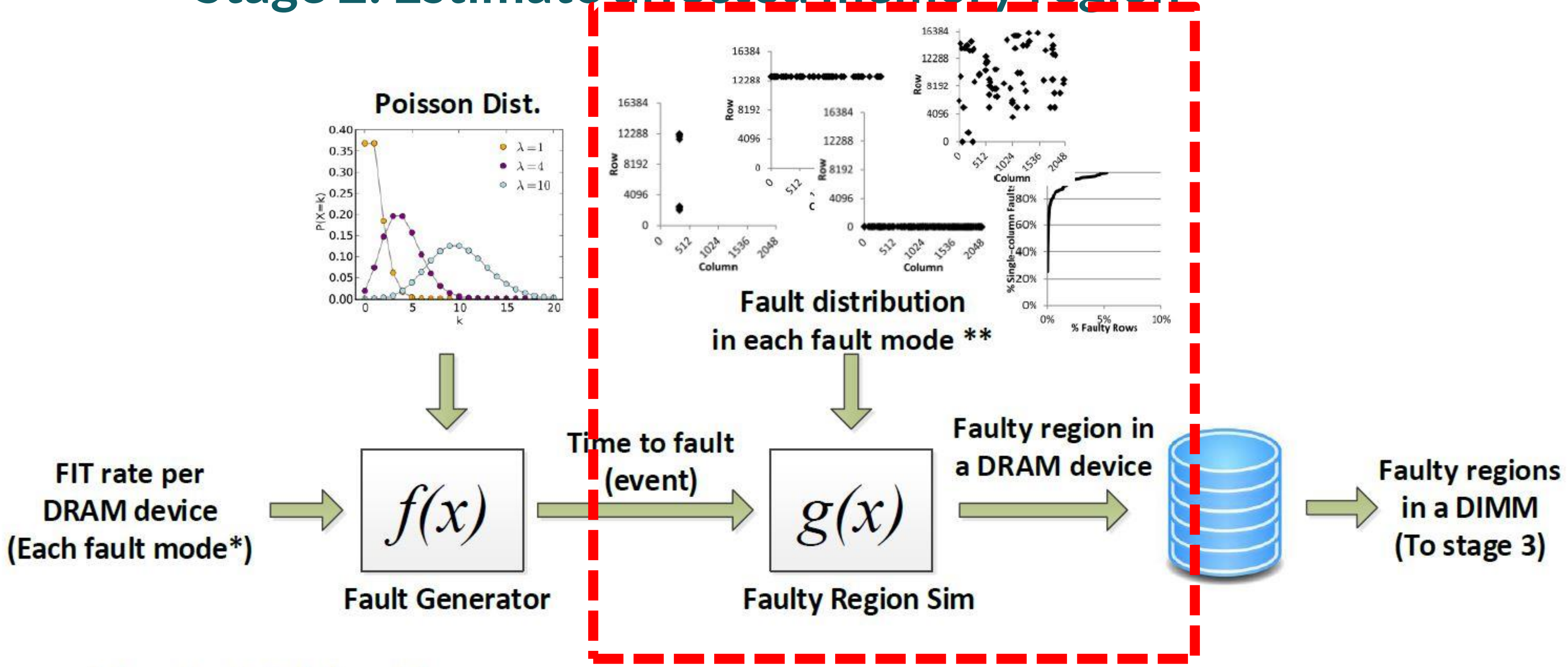
*** Estimated from previous literatures, e.g. Sridharan et al., A Study of DRAM Failure in the Field, SC 2012*

Fault simulation

Multi-stage Monte-Carlo fault simulator

– Stage 1: Faults are simulated as a *Poisson process*

– **Stage 2: Estimate affected memory region**



Poisson Dist.
λ=1
λ=4
λ=10
P(X=k)
k
Fault distribution in each fault mode **
Row
Column
% Single-column Faults
% Faulty Rows
FIT rate per DRAM device (Each fault mode*)
f(x)
Fault Generator
Time to fault (event)
g(x)
Faulty Region Sim
Faulty region in a DRAM device
Faulty regions in a DIMM (To stage 3)

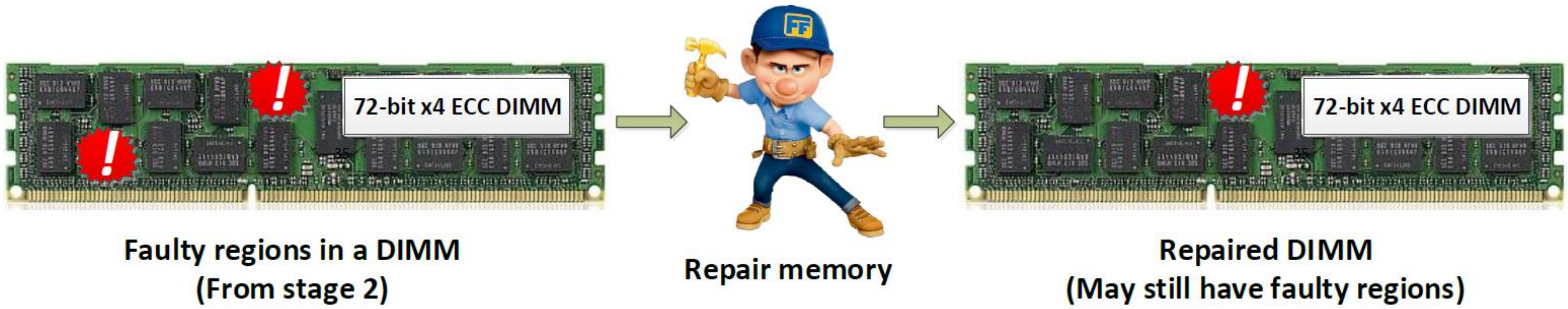
** Assuming single fault model*

*** Estimated from previous literatures, e.g. Sridharan et al., A Study of DRAM Failure in the Field, SC 2012*

Fault simulation

Multi-stage Monte-Carlo fault simulator

- Stage 1: Faults are simulated as a *Poisson process*
- Stage 2: Estimate affected memory region
- **Stage 3: Repair memory**



Fault simulation [DW Kim et al. ISCA'16]

Multi-stage Monte-Carlo fault simulator

- Stage 1: Faults are simulated as a *Poisson process*
- Stage 2: Estimate affected memory region/capacity
- Stage 3: Repair memory
- **Stage 4: Estimate DUE/SDC**
  - Use single symbol correction Chipkill-correct ECC
  - Double device errors are detected with very high likelihood
  - DUE/SDC is probabilistically determined by simulation results[1]

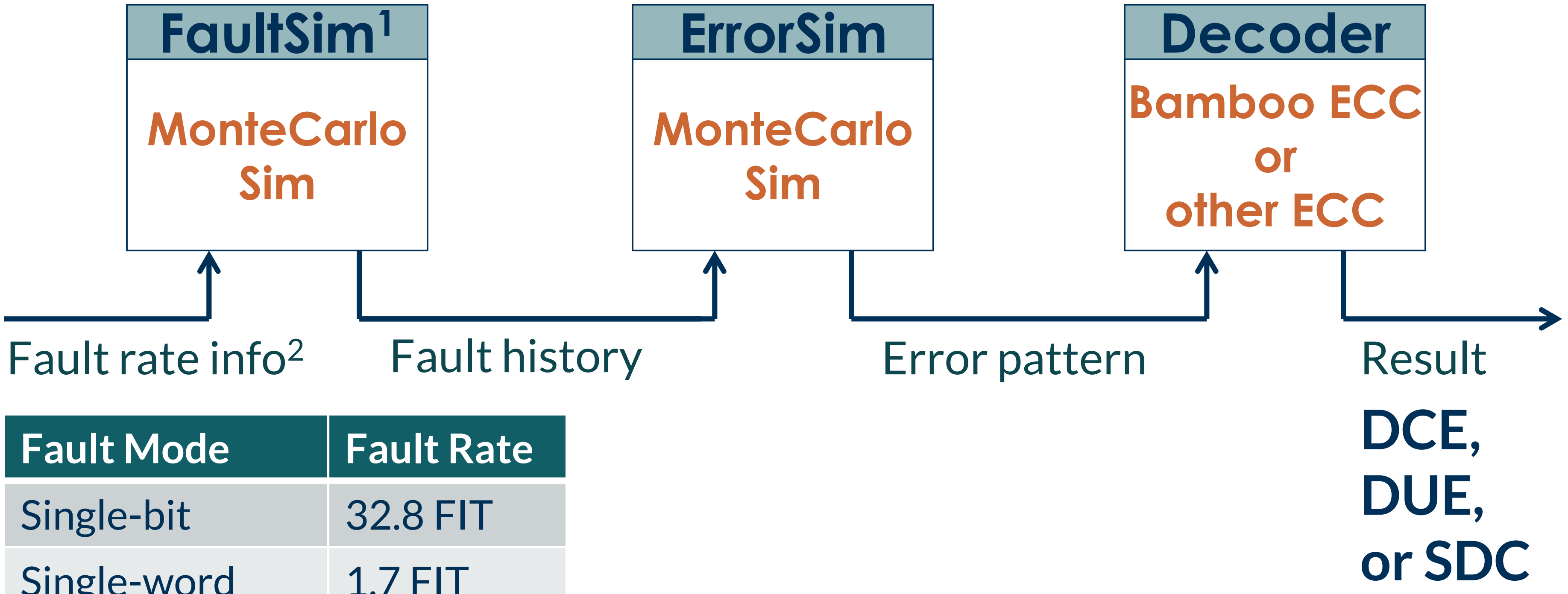
[1] Jungrae Kim, Michael Sullivan, and Mattan Erez,, "Bamboo ECC: Strong, Safe, and Flexible Codes for Reliable Computer Memory", HPCA 2015

FaultSim isn’t enough – misses true coverage of ECC

Evaluation: Reliability with expected reliability

[J. Kim et al. HPCA15]

3-stage simulation

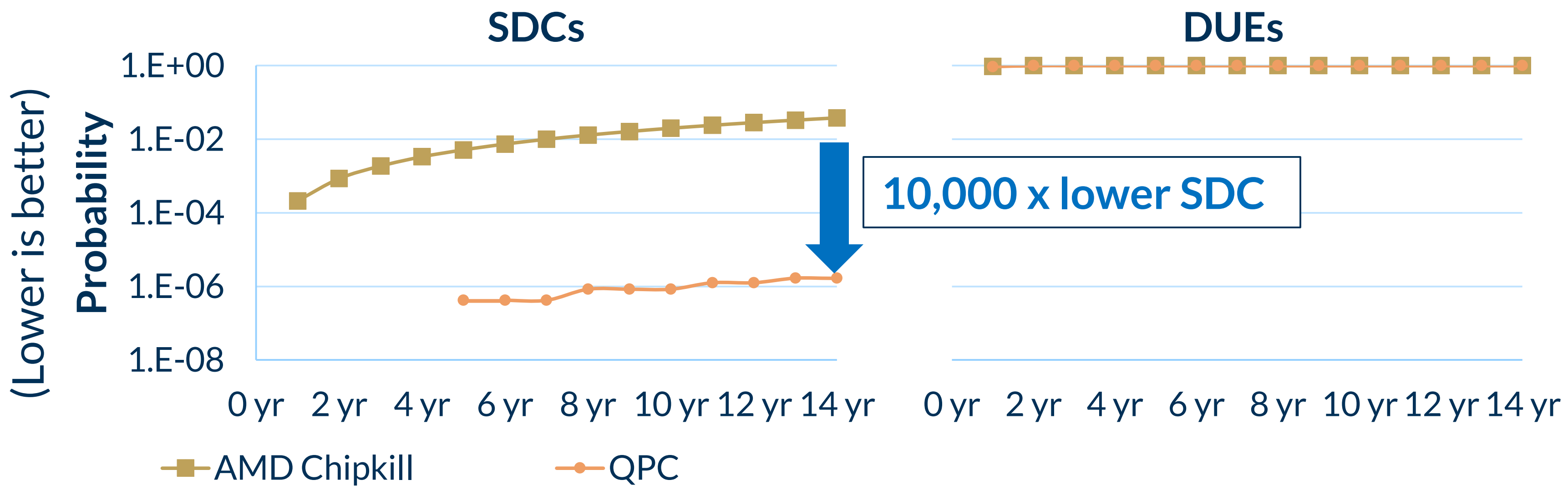


Fault Mode	Fault Rate
Single-bit	32.8 FIT
Single-word	1.7 FIT
Single-column	7.0 FIT
…	…

[1] "FAULTSIM: A fast, configurable memory-resilience simulator", Roberts et al., The Memory Forum '14
[2] "A study of DRAM failures in the field", Sridharan and Liberty, SC'12

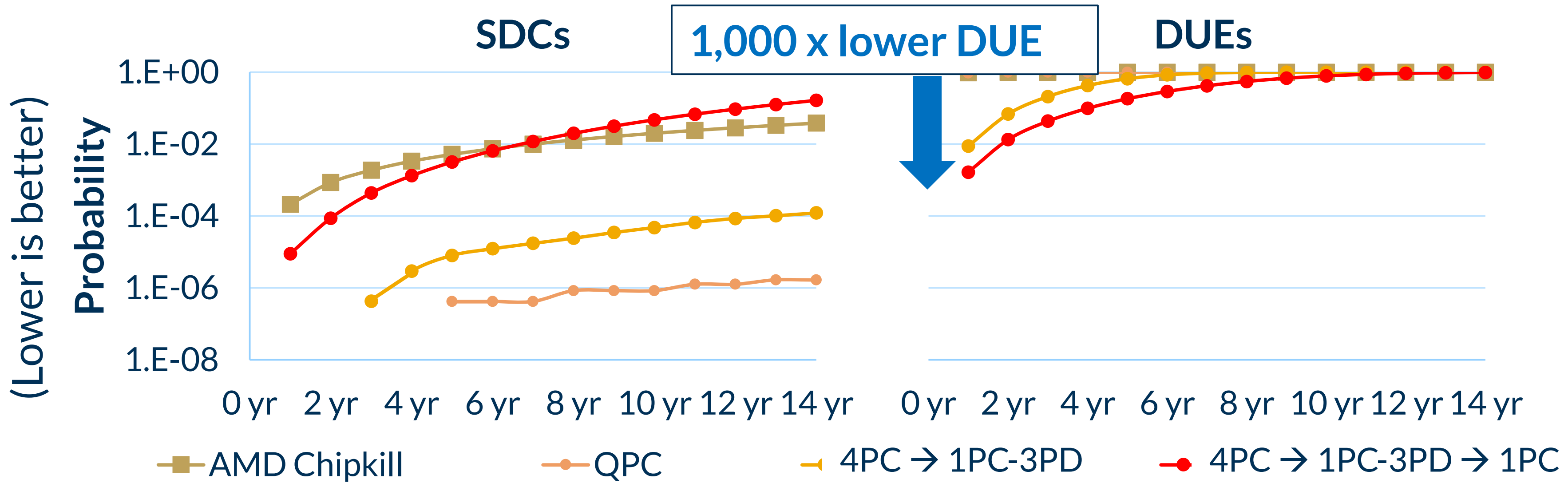
Example:
Bamboo System-level Reliability (72b channel)

A system with 100K ranks (18 x4 chips/rank)



Example: Bamboo System-level Reliability (72b channel)

A system with 100K ranks (18 x4 chips/rank)



Don’t forget about power, area, and latency: Bamboo Overheads

Logic overheads	AMD Chipkill	Bamboo QPC	Comparison
Area (NAND2 gates)	1,600	25,000	**16 x**
Latency (XOR2 gates)	8	10	**+ 2**

Logic overheads of encoder and decoder (error detection part), each

Performance

- **2%** (H. mean) execution time increases[1]
- Due to +3 read/write latency to wait block transfer

Energy

- **<1%** (H.mean) DRAM energy increases[1]

[1] SPECcpu2006 on Gem5 simulator (2GHz 1-core / 2M LLC / 2GB (64+8)b DDR3-1600)