SYNERGY: Rethinking Secure-Memory Design for Error-Correcting Memories

HPCA-2018

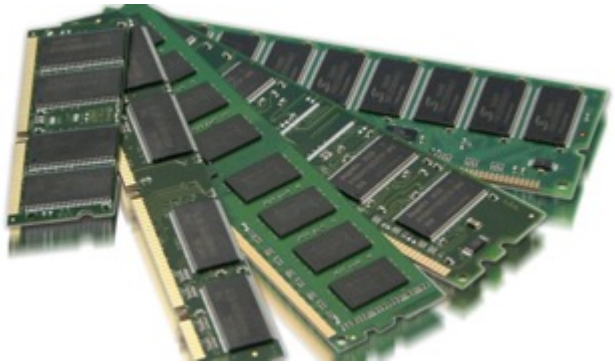
Vienna, Austria

Gururaj Saileshwar[1]

Prashant Nair[1] Prakash Ramrakhyani[2] Wendy Elsasser[2] Moinuddin Qureshi[1]

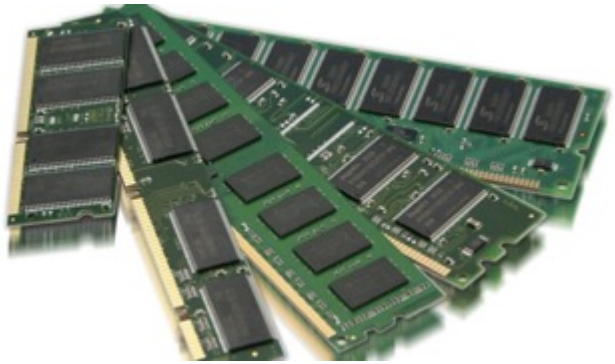
[1] Georgia Tech [2] arm Research

Designing Resilient Memories for Servers



Server memories need protection against **data corruption**

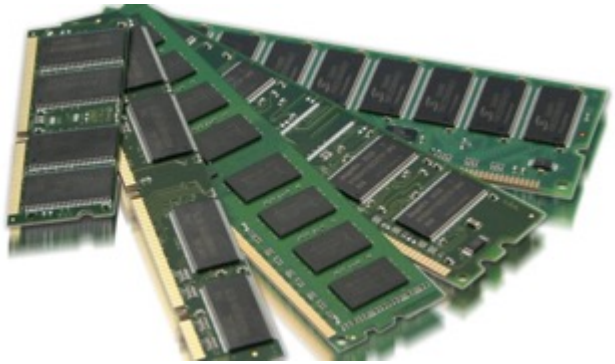
Designing Resilient Memories for Servers



by Malicious ***Attacks***

Server memories need protection against **data corruption**

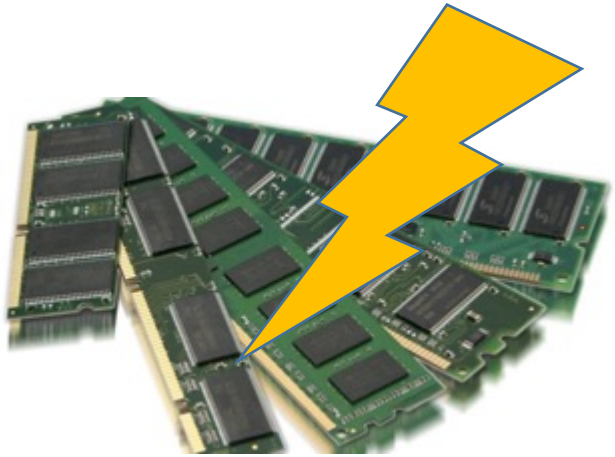
Designing Resilient Memories for Servers



by Malicious ***Attacks***

Server memories need protection against **data corruption**

Designing Resilient Memories for Servers



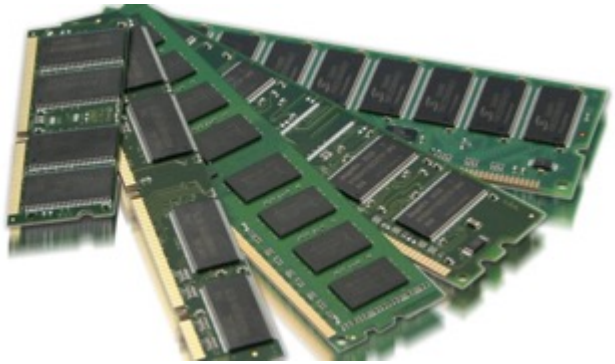
Server memories need protection against **data corruption**

by Malicious ***Attacks***

by Natural ***Errors***

Designing Resilient Memories for Servers

Resilient Memories



Server memories need protection against **data corruption**

Security

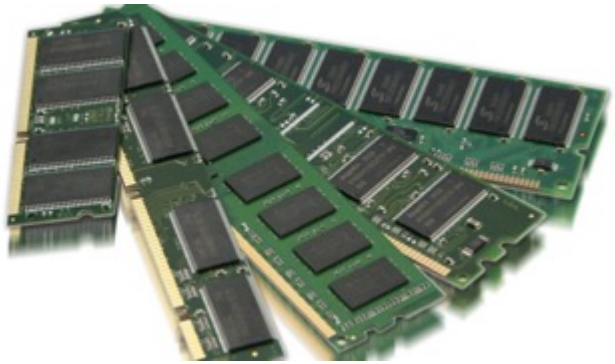
Reliability

by Malicious ***Attacks***

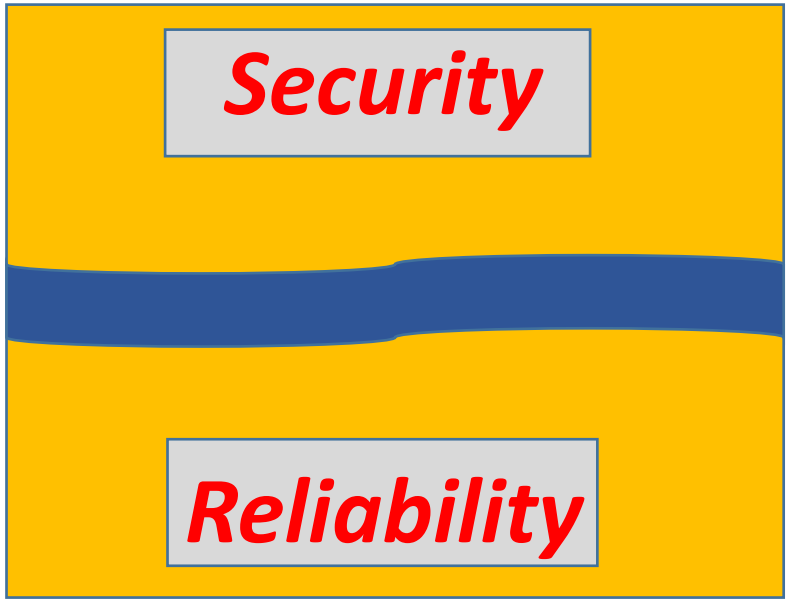
by Natural ***Errors***

Designing Resilient Memories for Servers

Resilient Memories

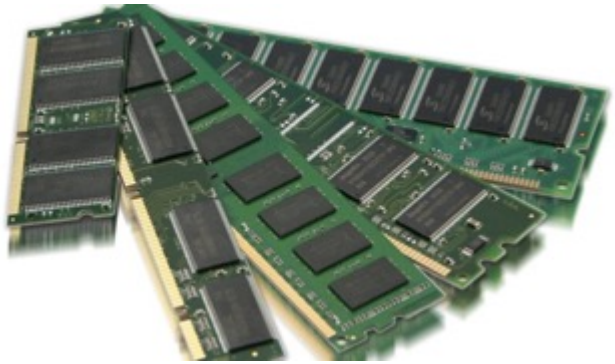


Server memories need protection against **data corruption**

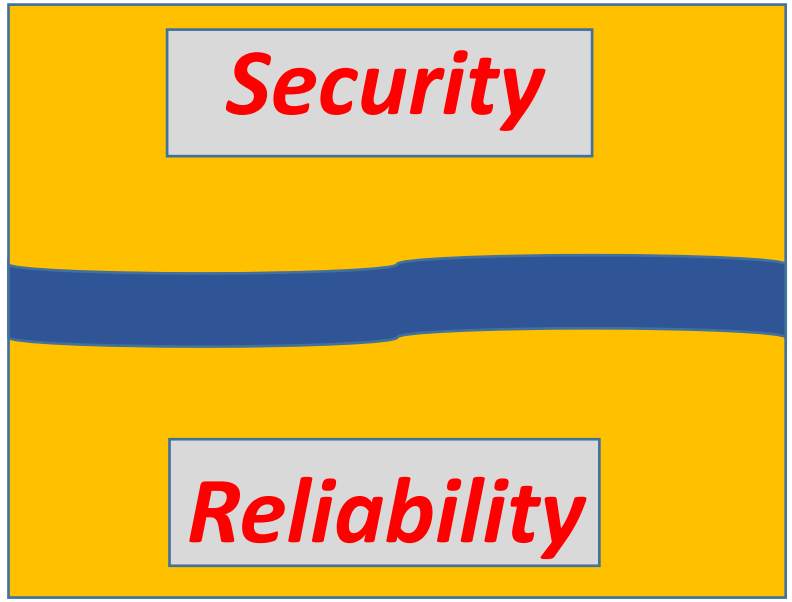


Designing Resilient Memories for Servers

Resilient Memories



Server memories need protection against **data corruption**



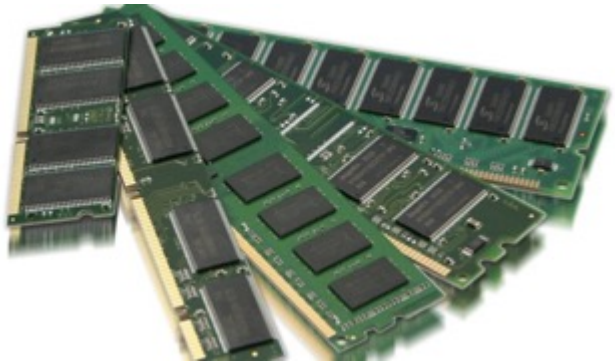
Attack Resilience – *Memory Security*

intel ***SGX***

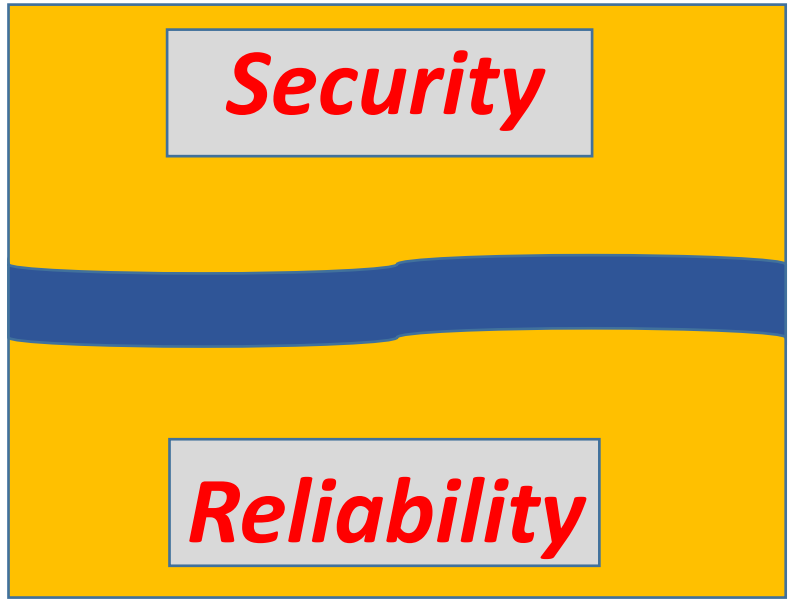
(Software Guard Extensions)

Designing Resilient Memories for Servers

Resilient Memories



Server memories need protection against **data corruption**

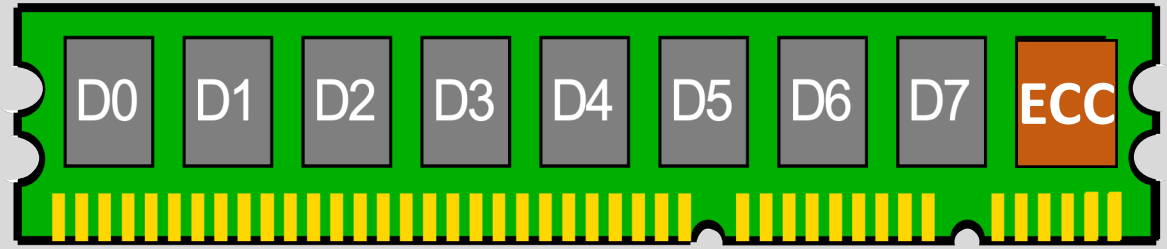


Attack Resilience – *Memory Security*

intel **SGX**

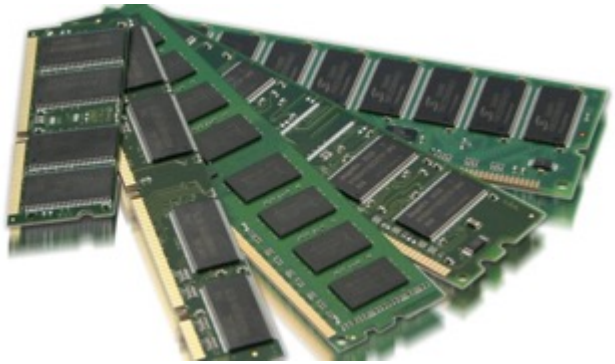
(Software Guard Extensions)

Error Resilience – *ECC-DIMM*

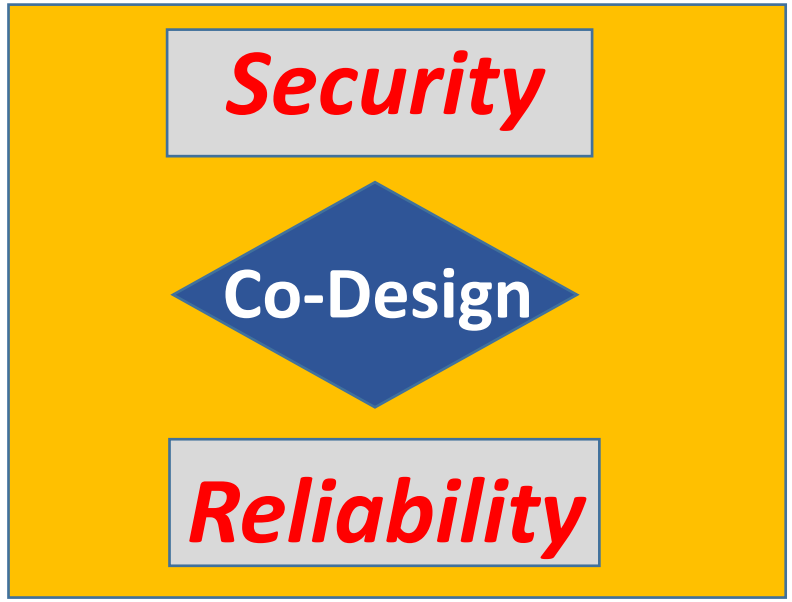


Designing Resilient Memories for Servers

Resilient Memories



Server memories need protection against **data corruption**

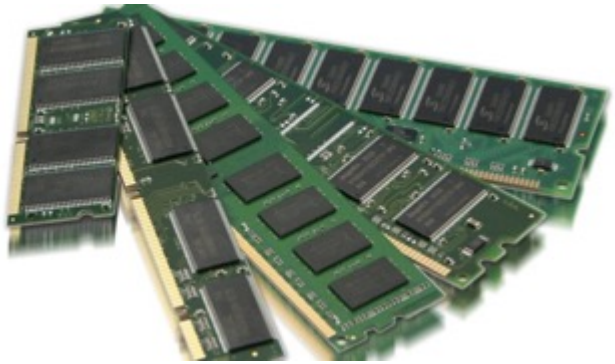


SYNERGY

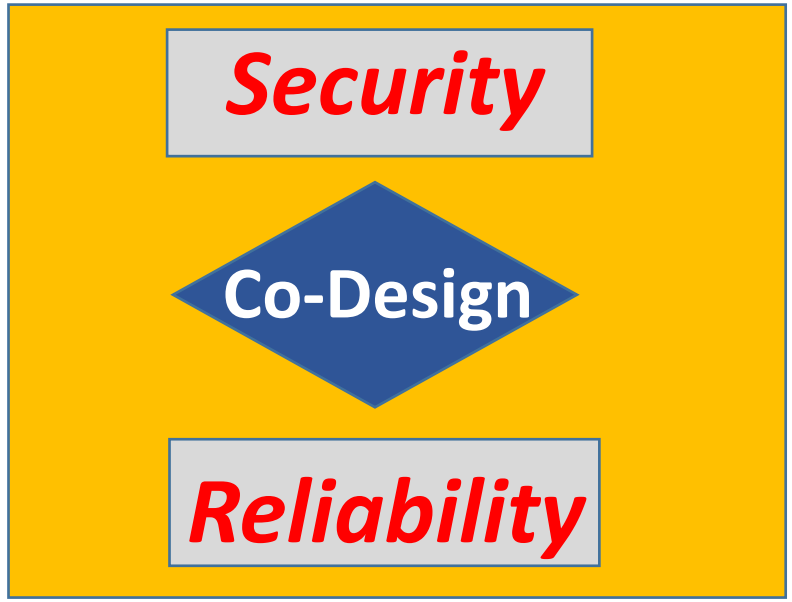
Co-Design of Security-Reliability

Designing Resilient Memories for Servers

Resilient Memories



Server memories need protection against **data corruption**



SYNERGY

Co-Design of Security-Reliability

- *Better* **Performance – *20%***
- *Better* **Reliability – *185X***
- *Maintaining* **Security**
- *No Extra* **Storage**

Threat Model for Memory Modules

Unauthorized Reads

Unauthorized Writes

Replay Attack

Threat Model for Memory Modules

Unauthorized Reads

Unauthorized Writes

Replay Attack

Cold Boot Attack

Threat Model for Memory Modules

Unauthorized Reads

Cold Boot Attack

Unauthorized Writes

DMA Attack

Replay Attack

Threat Model for Memory Modules

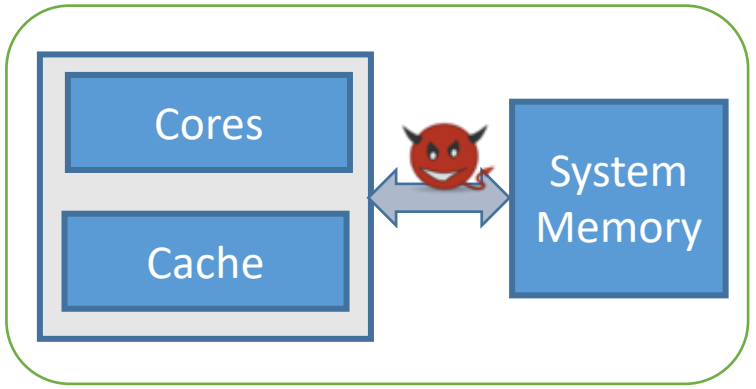
Unauthorized Reads

Cold Boot Attack

Unauthorized Writes

DMA Attack

Replay Attack



Man-in-the-middle Attack

Threat Model for Memory Modules

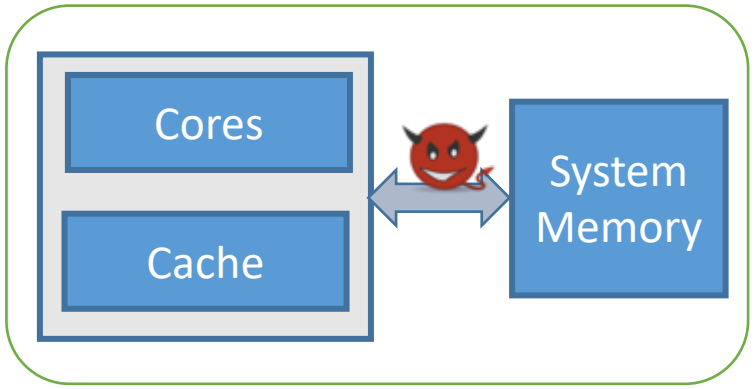
Unauthorized Reads

Cold Boot Attack

Unauthorized Writes

DMA Attack

Replay Attack



Man-in-the-middle Attack

Secure memory needs to protect against these attacks !

How Do Secure Memories Work ?

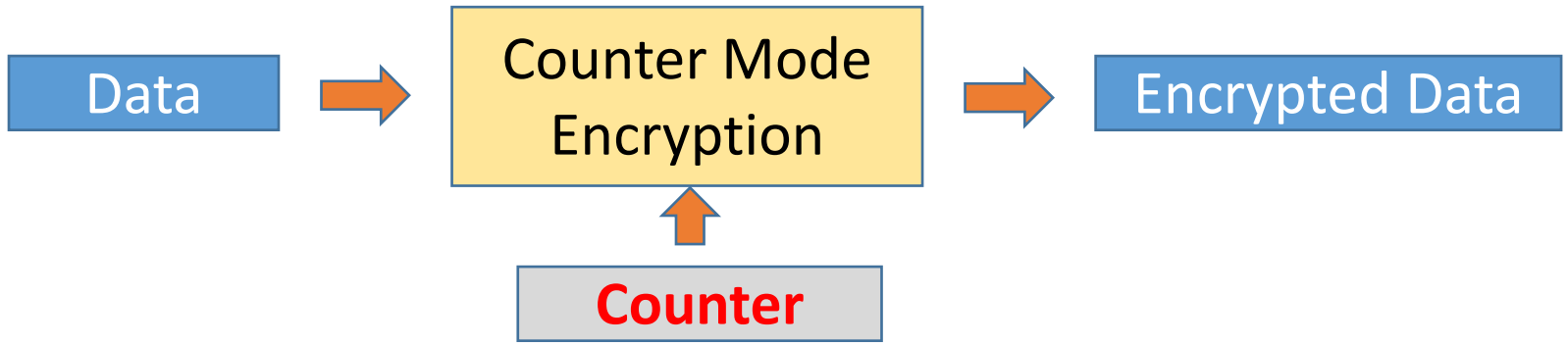
Unauthorized Reads

Unauthorized Writes

Replay Attack

How Do Secure Memories Work ?

Unauthorized Reads

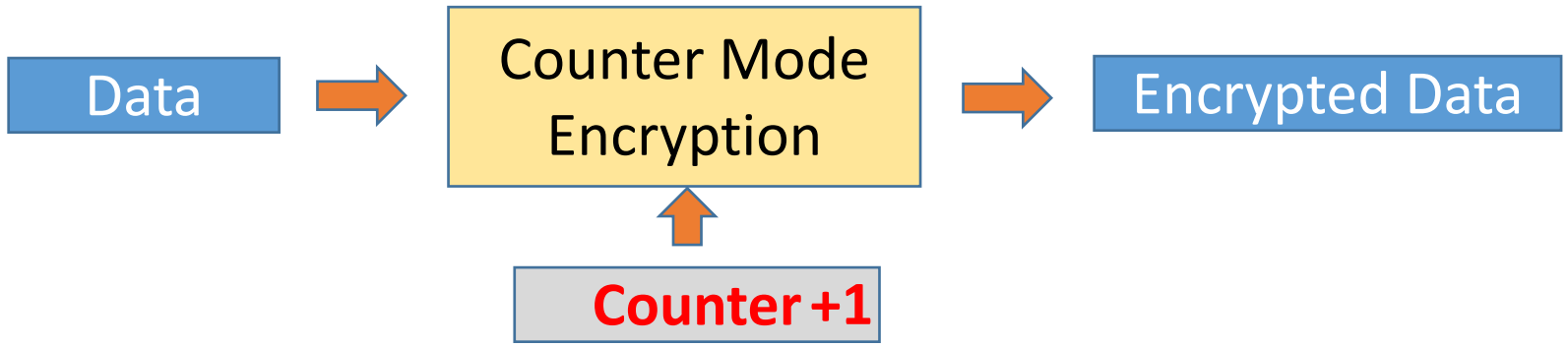


Unauthorized Writes

Replay Attack

How Do Secure Memories Work ?

Unauthorized Reads

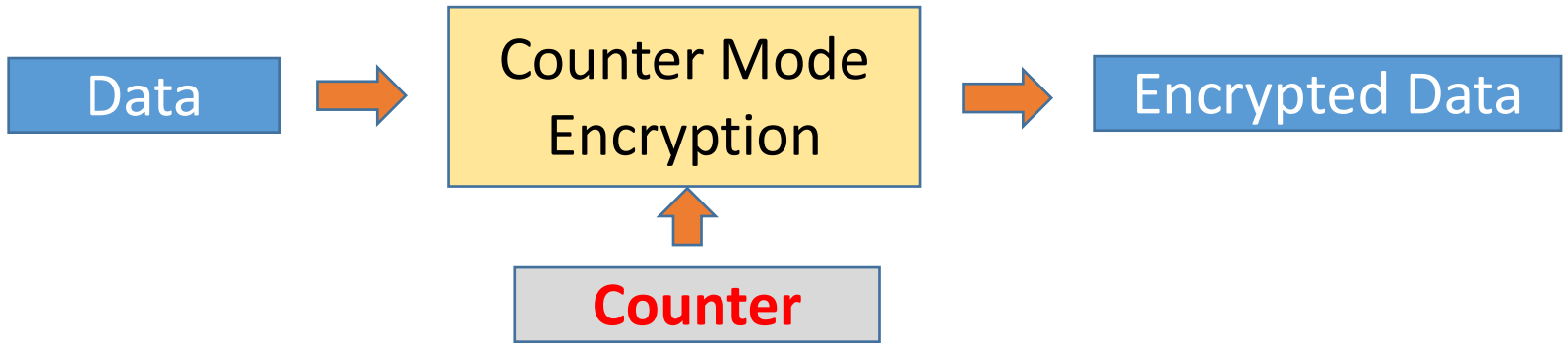


Unauthorized Writes

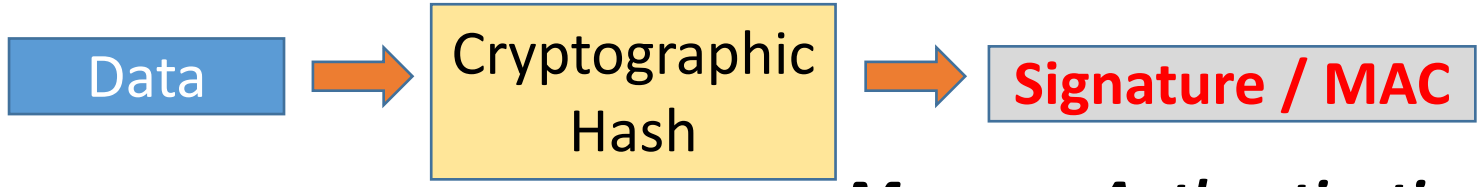
Replay Attack

How Do Secure Memories Work ?

Unauthorized Reads



Unauthorized Writes

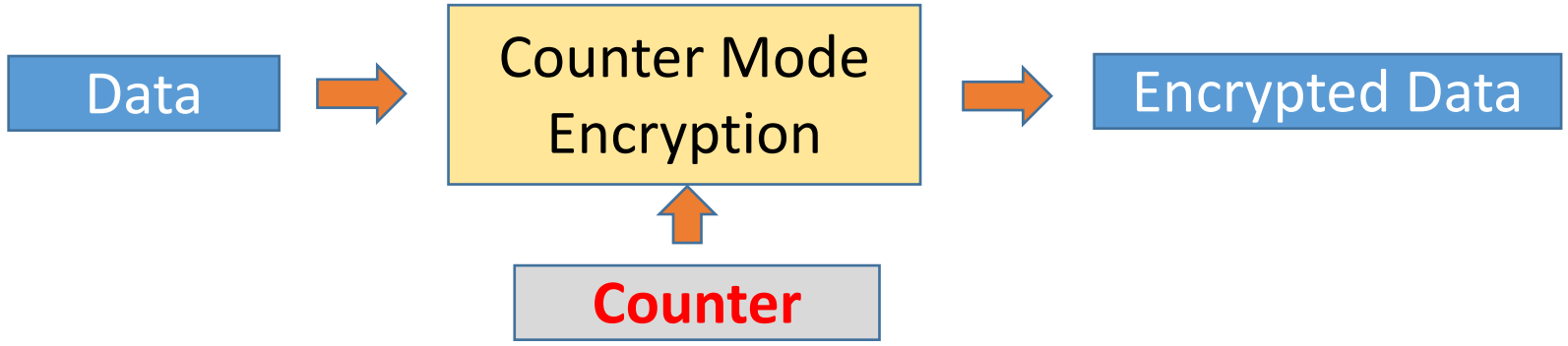


Message Authentication Code

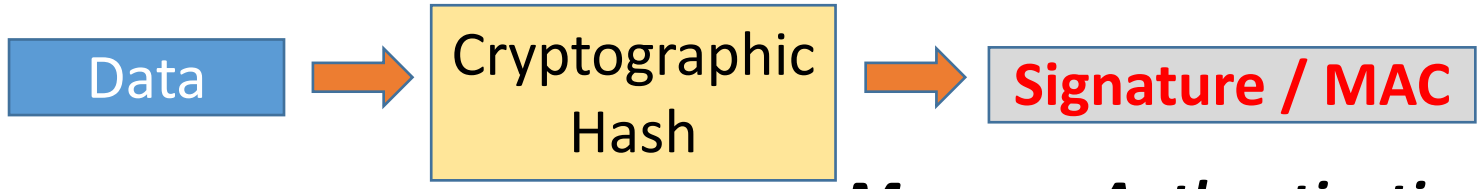
Replay Attack

How Do Secure Memories Work ?

Unauthorized Reads

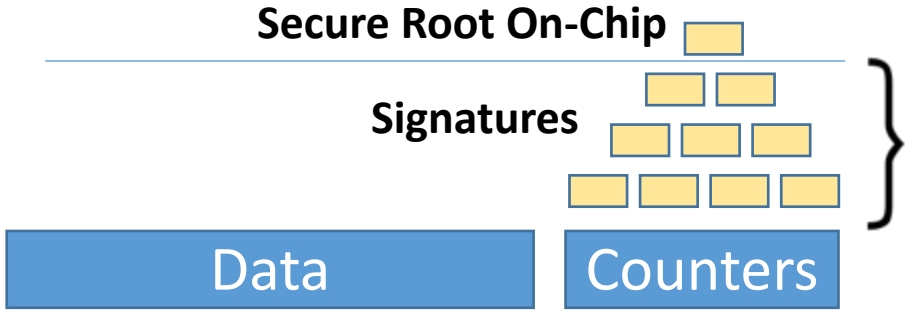


Unauthorized Writes



Message Authentication Code

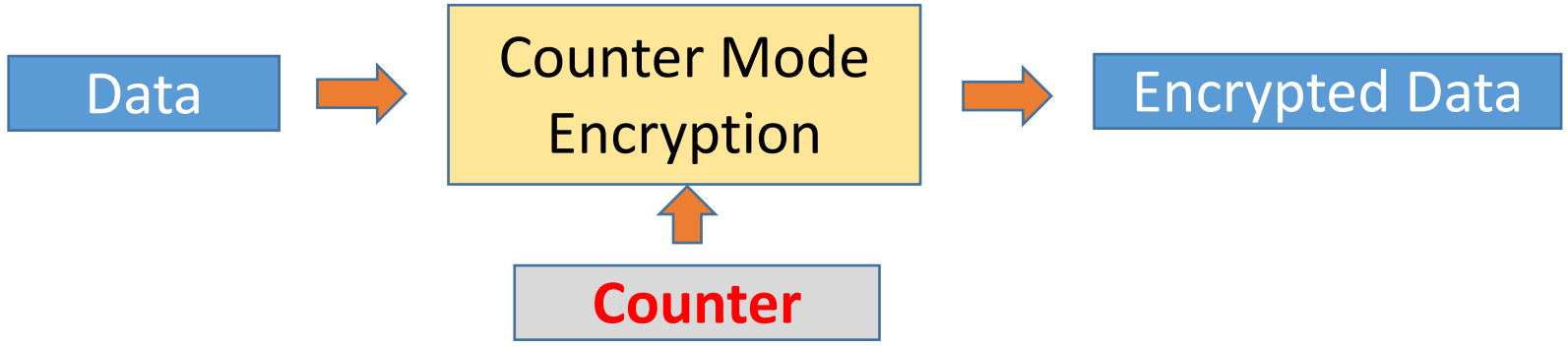
Replay Attack



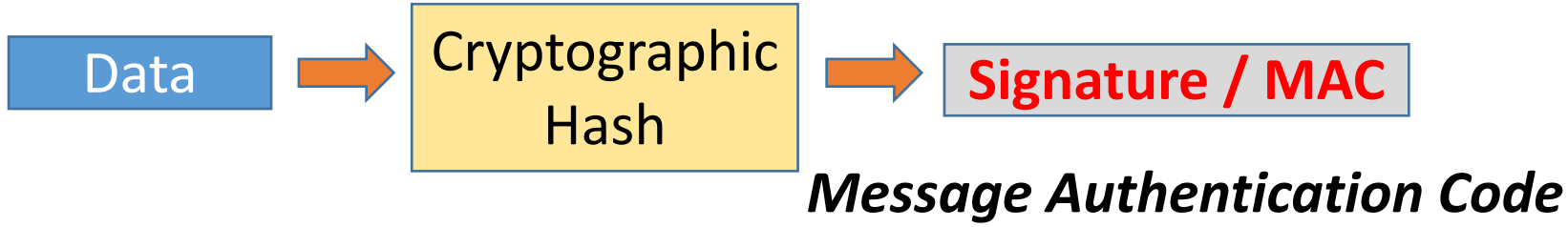
Integrity-Tree

How Do Secure Memories Work ?

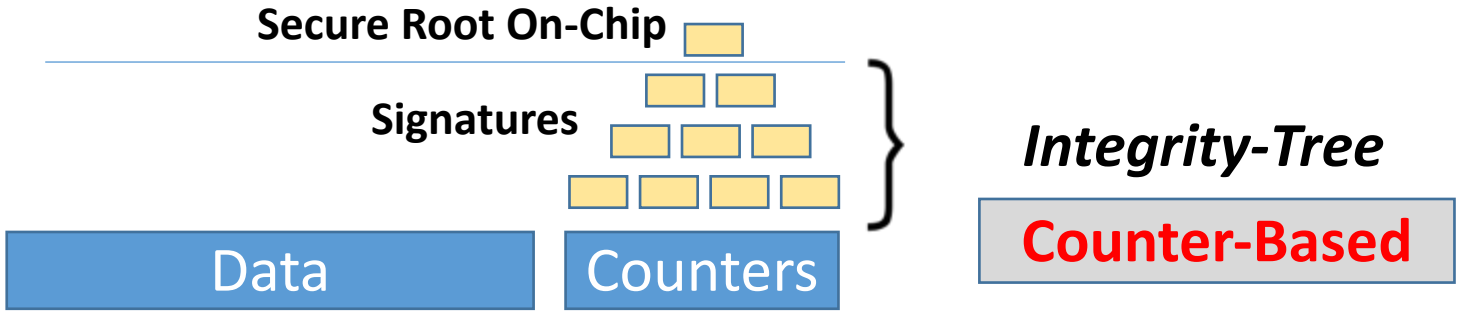
Unauthorized Reads



Unauthorized Writes

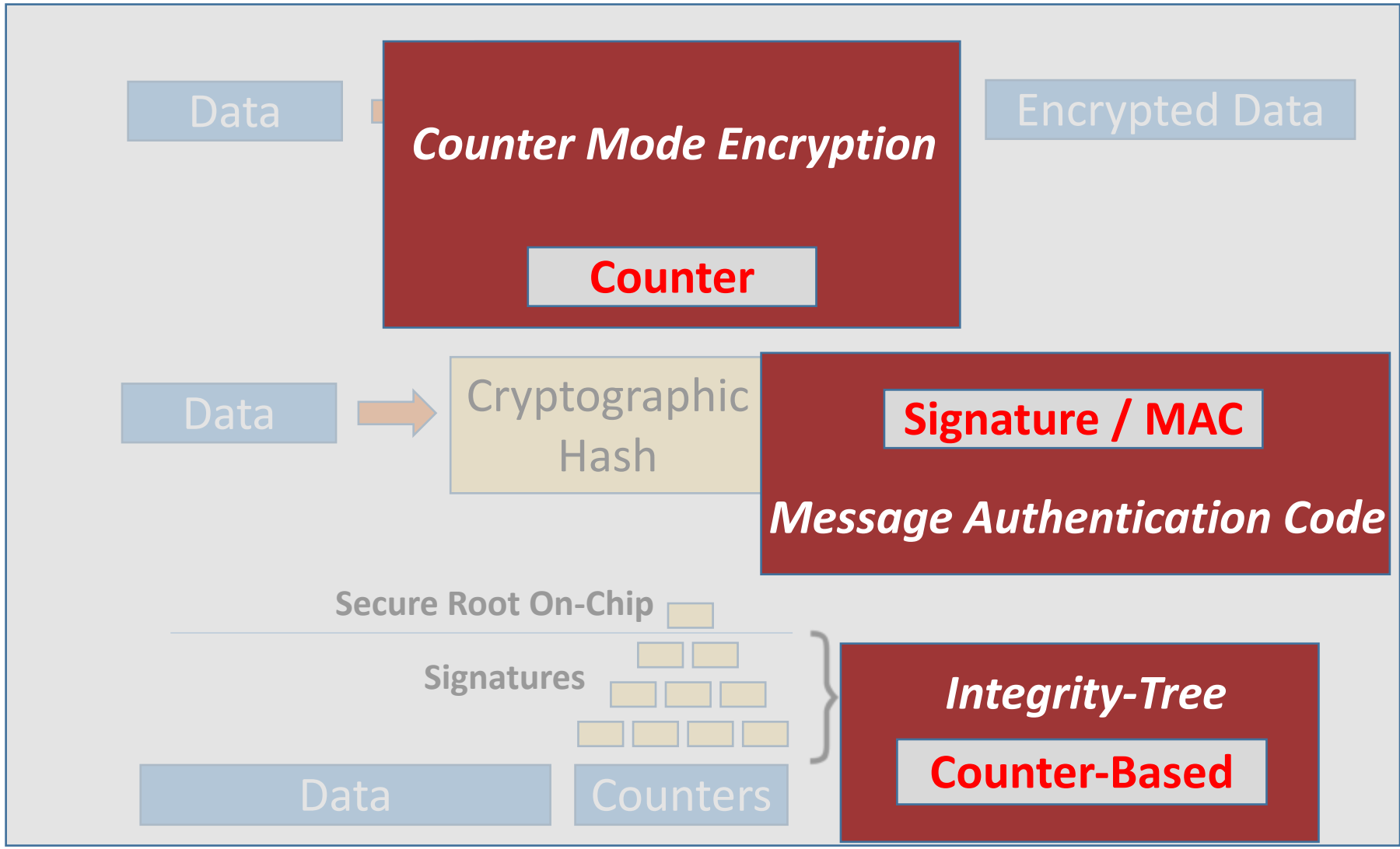


Replay Attack

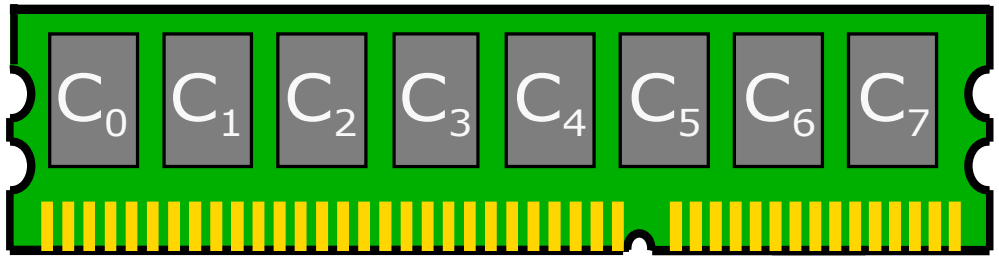


How Do Secure Memories Work ?

Intel SGX

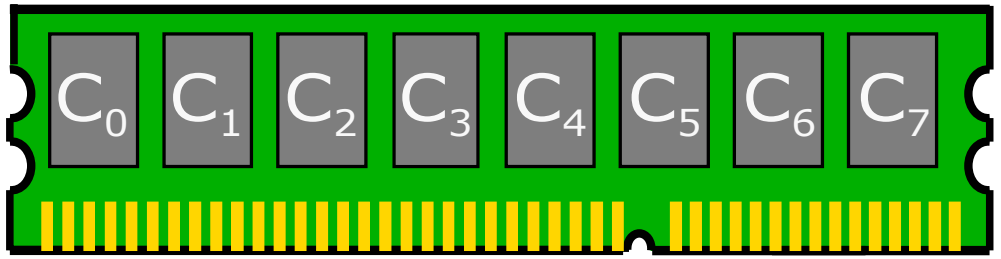


Problem: Metadata Accesses Consume Memory Bandwidth



Access Number

Problem: Metadata Accesses Consume Memory Bandwidth

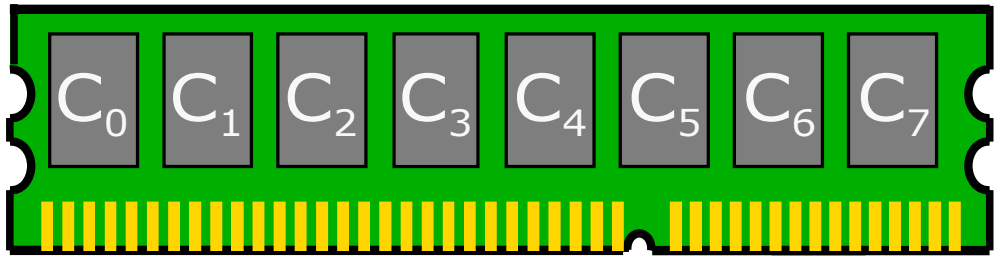


Access Number

1

Data Cacheline

Problem: Metadata Accesses Consume Memory Bandwidth

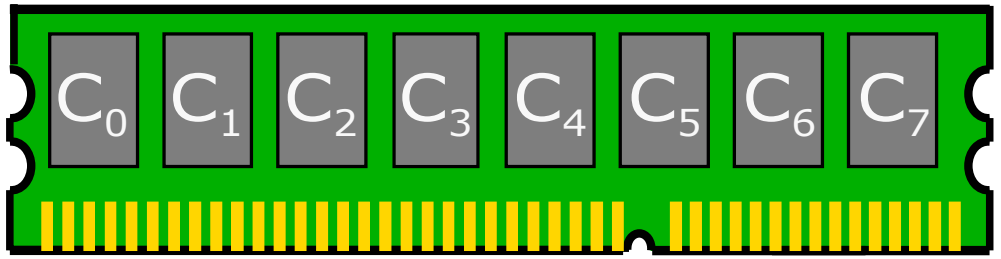


Access Number

1 Data Cacheline

2 Ctr Encryption Counter Cacheline — **Used to Encrypt Data**

Problem: Metadata Accesses Consume Memory Bandwidth



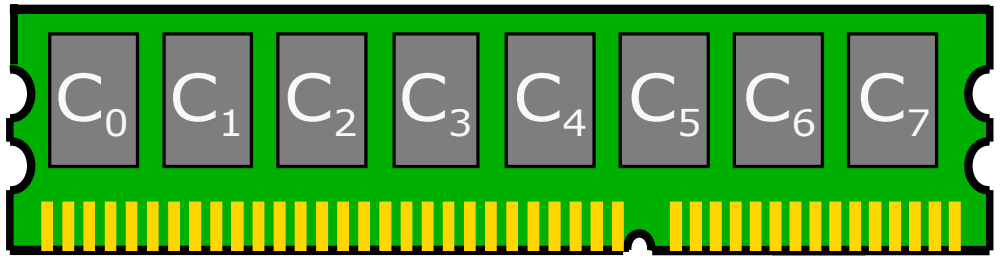
Access Number

1 Data Cacheline

3 MAC | MAC Cacheline — **Prevents Data Tampering**

2 Ctr | Encryption Counter Cacheline — **Used to Encrypt Data**

Problem: Metadata Accesses Consume Memory Bandwidth



Access Number

1 Data Cacheline

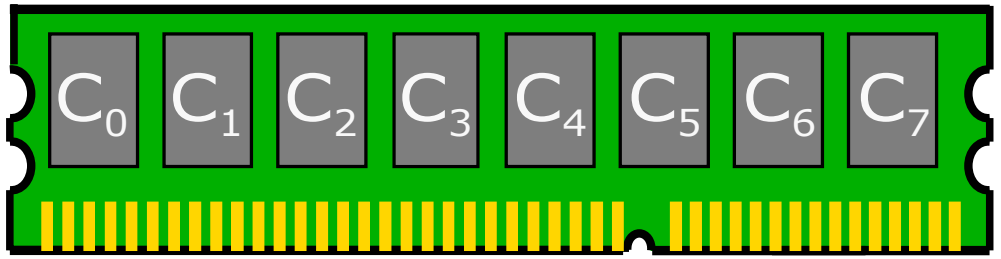
3 MAC | MAC Cacheline — **Prevents Data Tampering**

2 Ctr | Encryption Counter Cacheline — **Used to Encrypt Data**

4 Ctr | Integrity-Tree Counter Cacheline — **Prevents Replay-Attacks**

⋮

Problem: Metadata Accesses Consume Memory Bandwidth



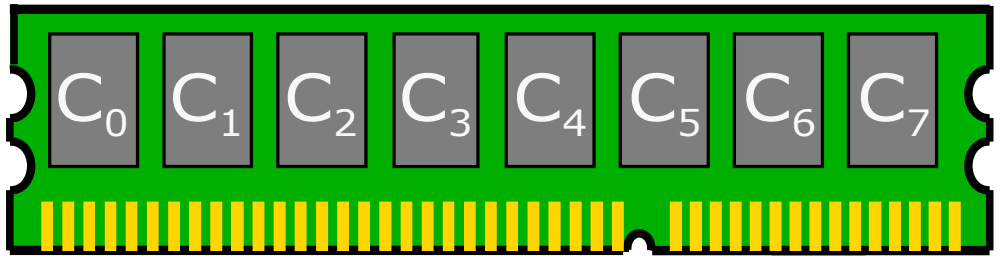
Secure Memory *can require*
4+ memory accesses *for each* ***Data access***

Access Number

1. Data Cacheline
3. MAC | MAC Cacheline — **Prevents Data Tampering**
2. Ctr | Encryption Counter Cacheline — **Used to Encrypt Data**
4. Ctr | Integrity-Tree Counter Cacheline — **Prevents Replay-Attacks**
⋮

Memory Access Bloat

Problem: Metadata Accesses Consume Memory Bandwidth



Secure Memory *can require*
4+ memory accesses *for each* ***Data access***

Access Number

1 — Data Cacheline

3 — MAC | MAC Cacheline — **Prevents Data Tampering**

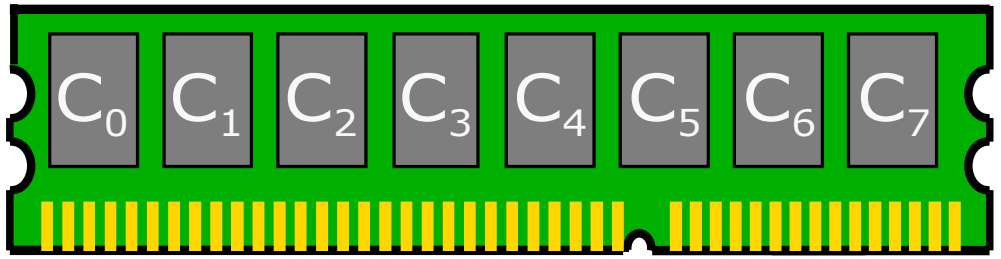
2 — Ctr | Encryption Counter Cacheline — **Used to Encrypt Data**

4 — Ctr | Integrity-Tree Counter Cacheline — **Prevents Replay-Attacks**

Cacheable On-Chip

Memory Access Bloat

Problem: Metadata Accesses Consume Memory Bandwidth



Secure Memory *can require*
4+ memory accesses *for each* ***Data access***

Access Number

1 Data Cacheline

3 MAC MAC Cacheline — **Prevents Data Tampering**

2 Ctr Encryption Counter Cacheline — Used to Encrypt Data

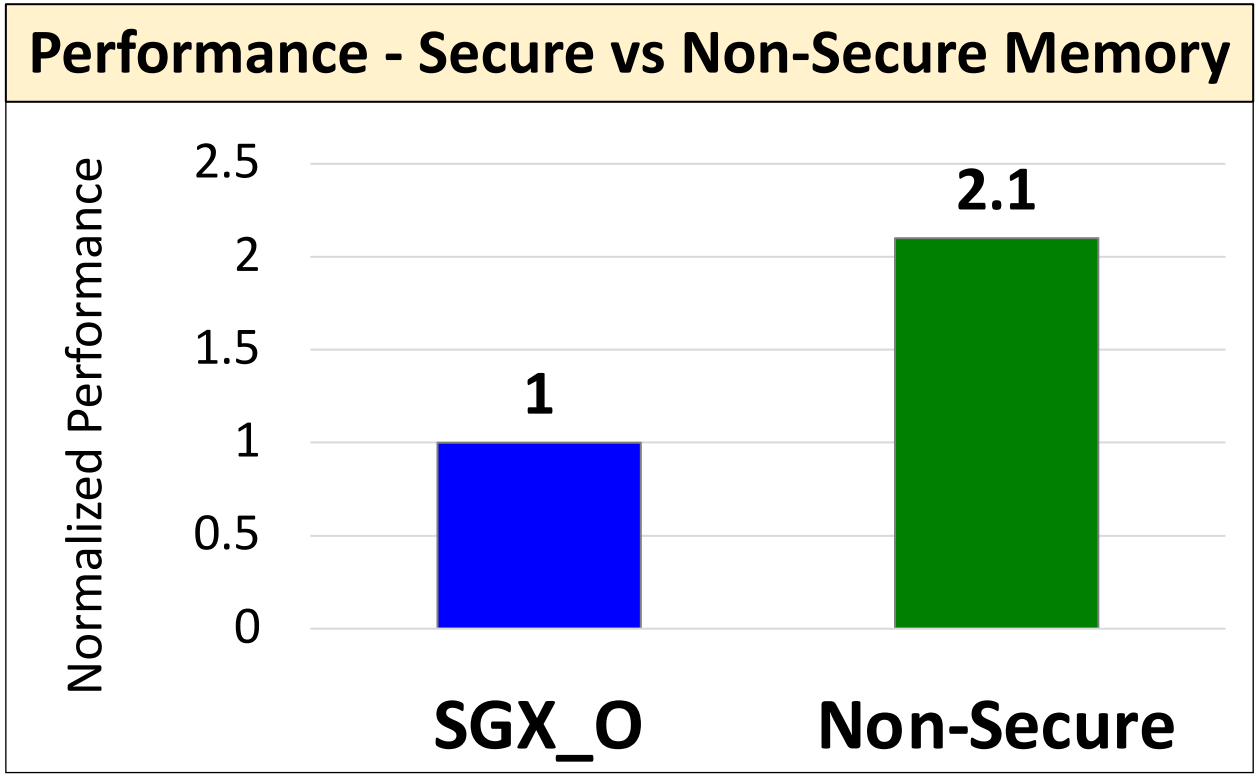
4 Ctr Integrity-Tree Counter Cacheline — Prevents Replay-Attacks

Cacheable On-Chip

Memory Access Bloat

Memory Access Bloat due to Security Metadata can lead to Performance Overheads

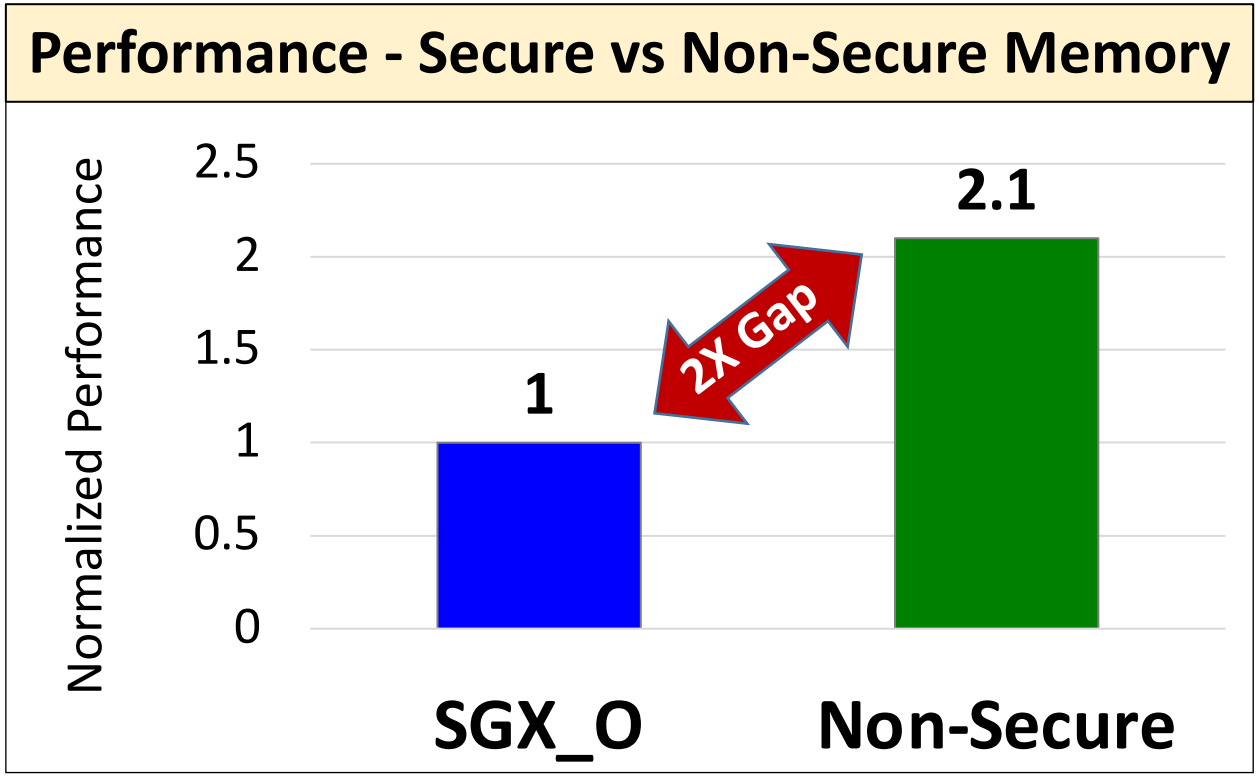
Dissecting Overheads For Secure Memory



SGX_O – *Enhanced Baseline for Secure Memory*

(SGX with LLC shared by Counters & Data)

Dissecting Overheads For Secure Memory

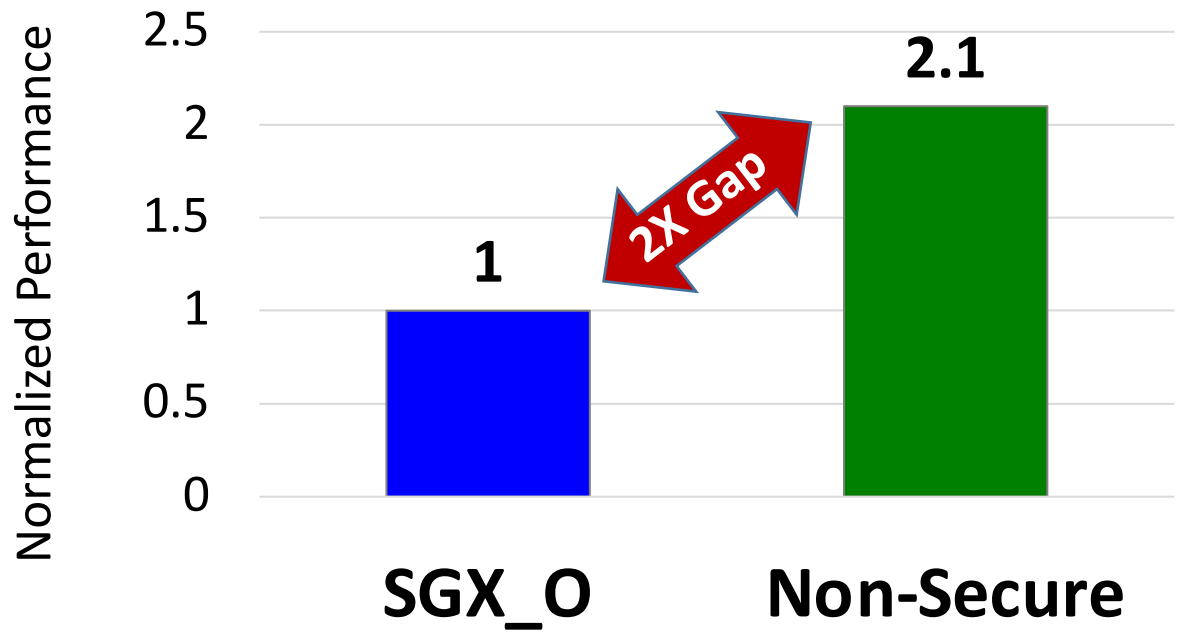


SGX_O – *Enhanced Baseline for Secure Memory*

(SGX with LLC shared by Counters & Data)

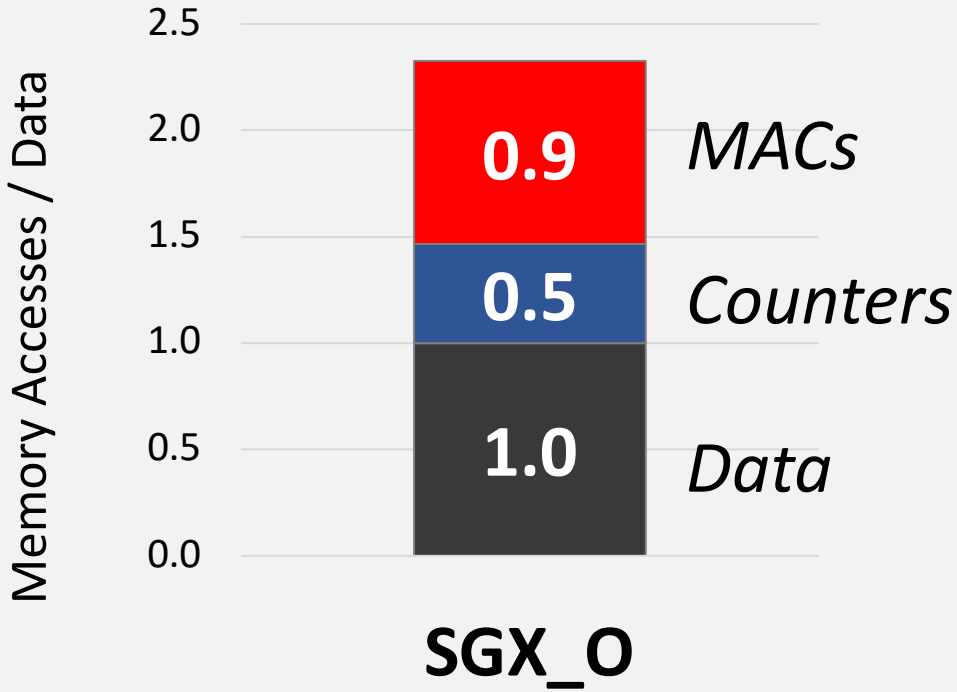
Dissecting Overheads For Secure Memory

Performance - Secure vs Non-Secure Memory



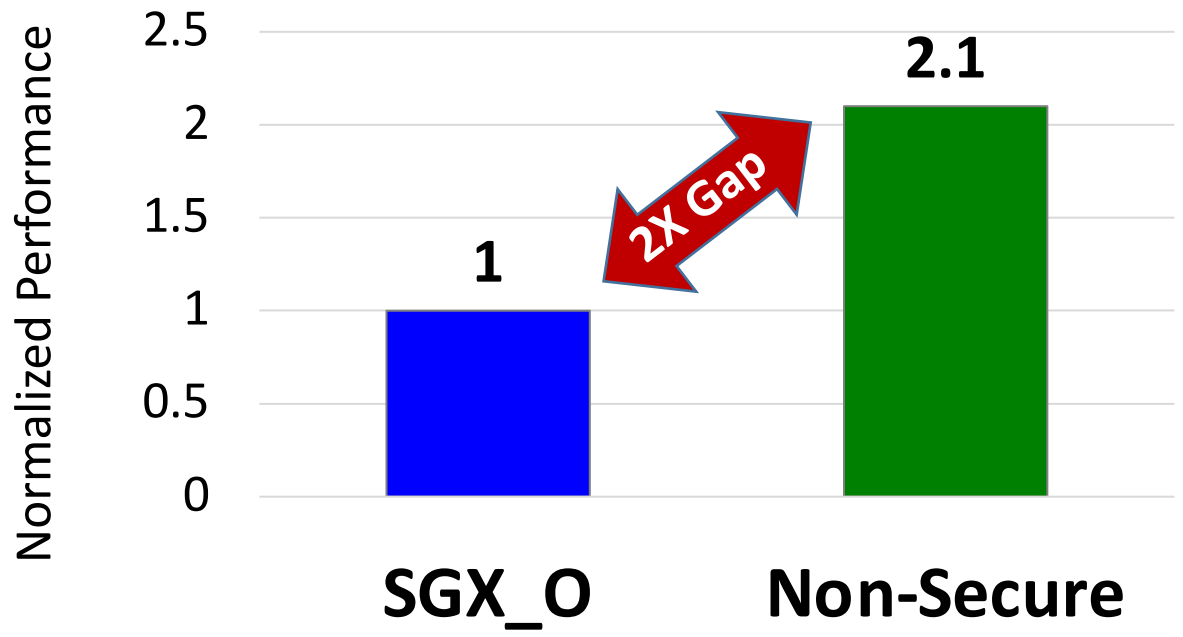
SGX_O – *Enhanced Baseline for Secure Memory (SGX with LLC shared by Counters & Data)*

Cause: Memory access bloat due to metadata accesses



Dissecting Overheads For Secure Memory

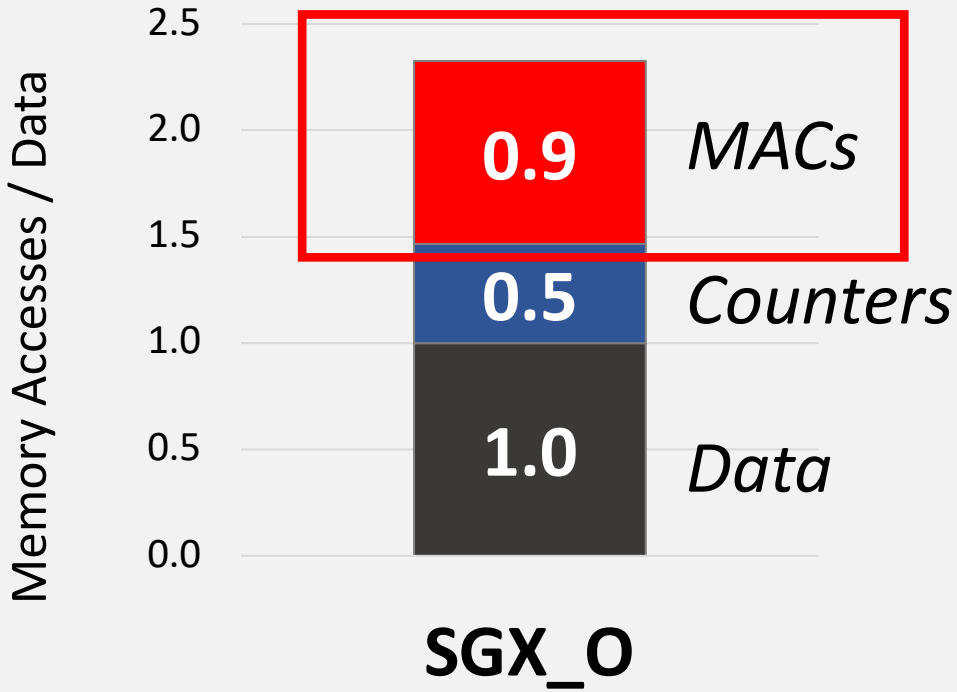
Performance - Secure vs Non-Secure Memory



SGX_O – *Enhanced Baseline for Secure Memory*

(SGX with LLC shared by Counters & Data)

Cause: Memory access bloat due to metadata accesses



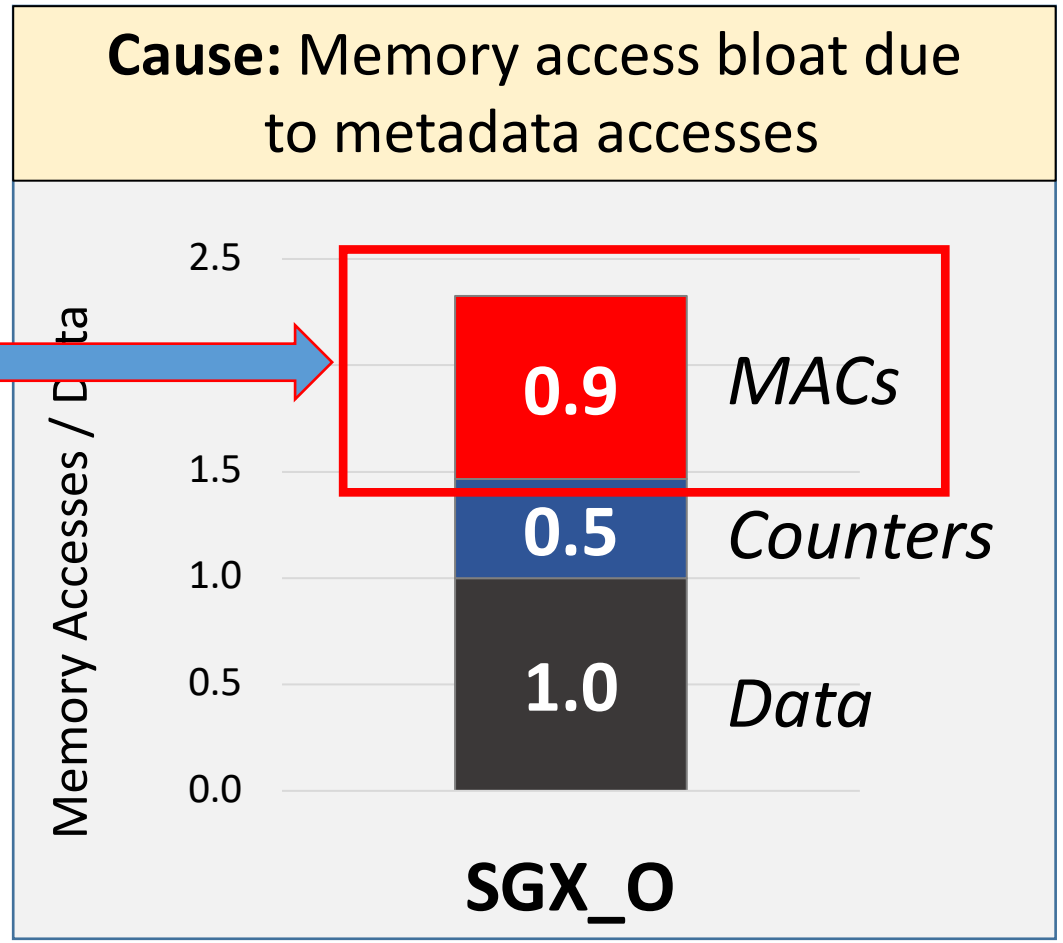
MACs cause 0.9x additional accesses – *focus of this paper !*

Dissecting Overheads For Secure Memory

Goal:

Reduce MAC access bloat to improve secure memory performance

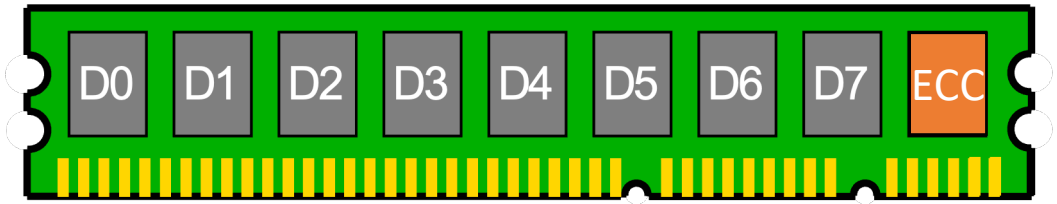
Cause: Memory access bloat due to metadata accesses



MACs cause 0.9x additional accesses – *focus of this paper !*

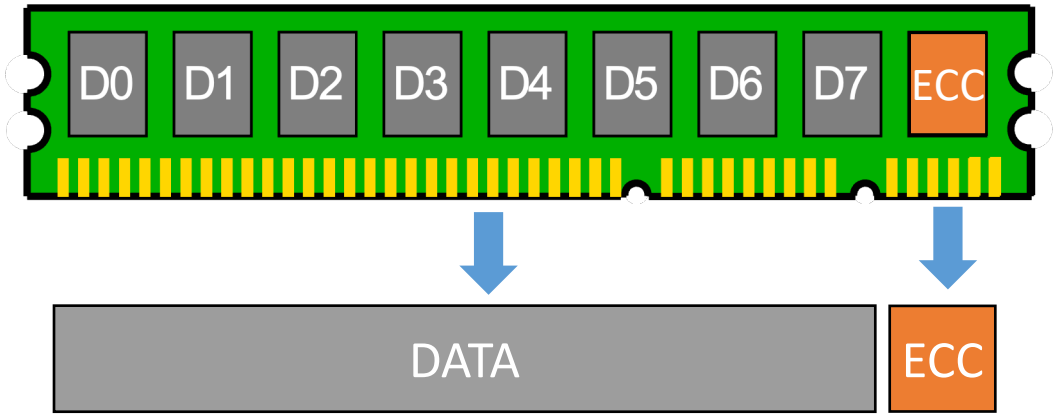
Insight: Leverage ECC-DIMM To Reduce MAC Overheads

ECC-DIMM for Reliability



Insight: Leverage ECC-DIMM To Reduce MAC Overheads

ECC-DIMM for Reliability



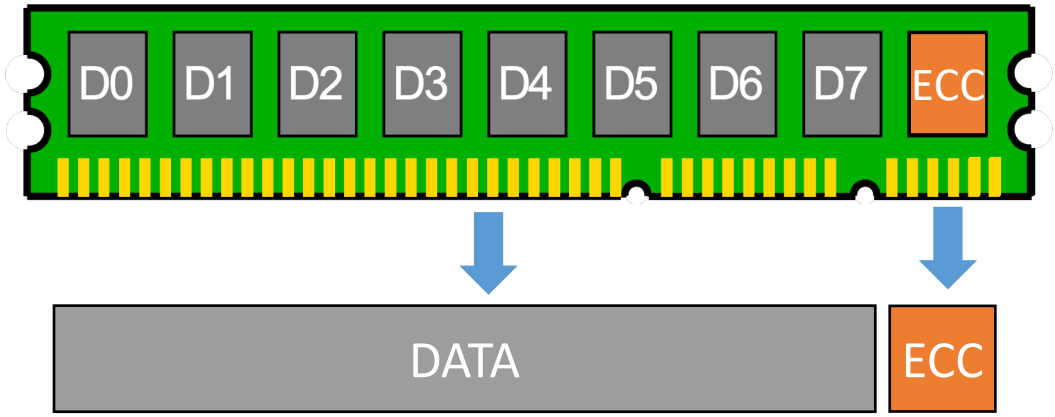
ECC

Fetched

Same Access as Data

Insight: Leverage ECC-DIMM To Reduce MAC Overheads

ECC-DIMM for Reliability



ECC

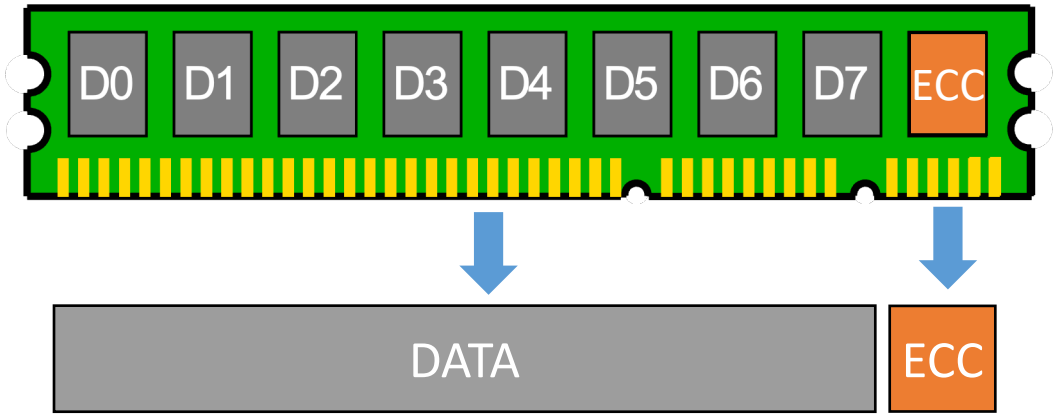
Fetched *Same Access as Data*

Detects *2-Bit Errors*

Corrects *1-Bit Error*

Insight: Leverage ECC-DIMM To Reduce MAC Overheads

ECC-DIMM for Reliability



ECC

Fetched — *Same Access as Data*

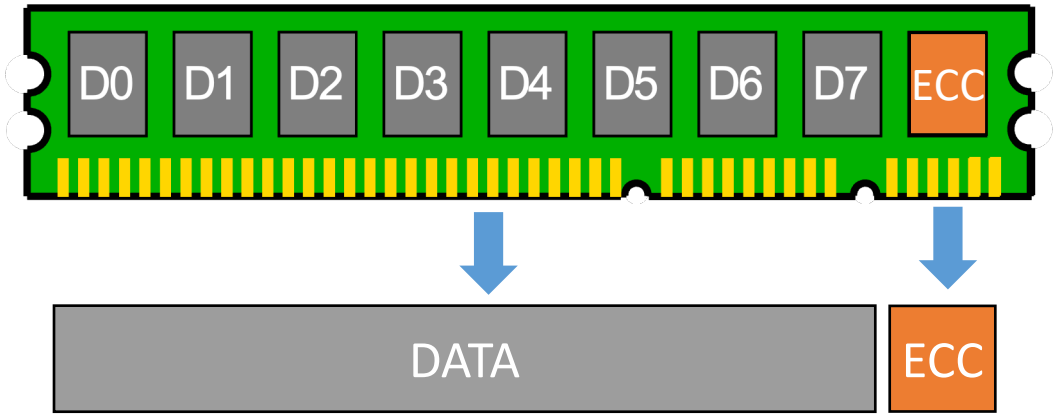
Detects — *2-Bit Errors*

Corrects — *1-Bit Error*

Rare

Insight: Leverage ECC-DIMM To Reduce MAC Overheads

ECC-DIMM for Reliability



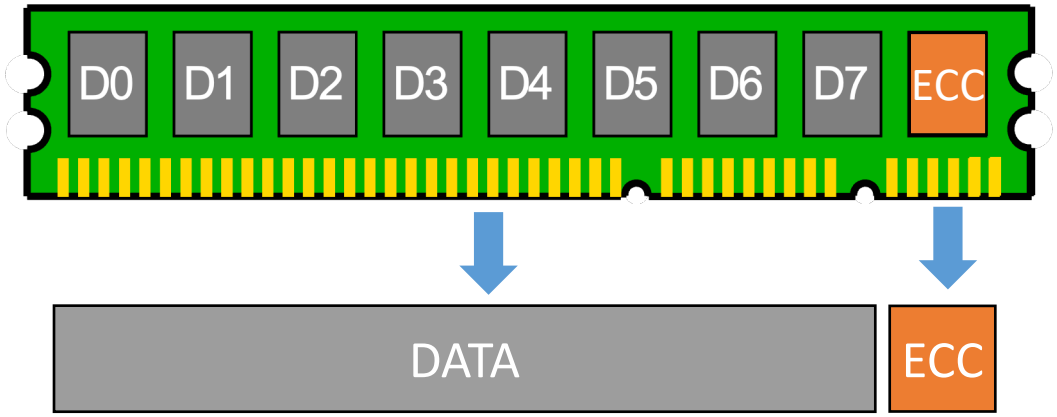
ECC

Fetched — *Same Access as Data*

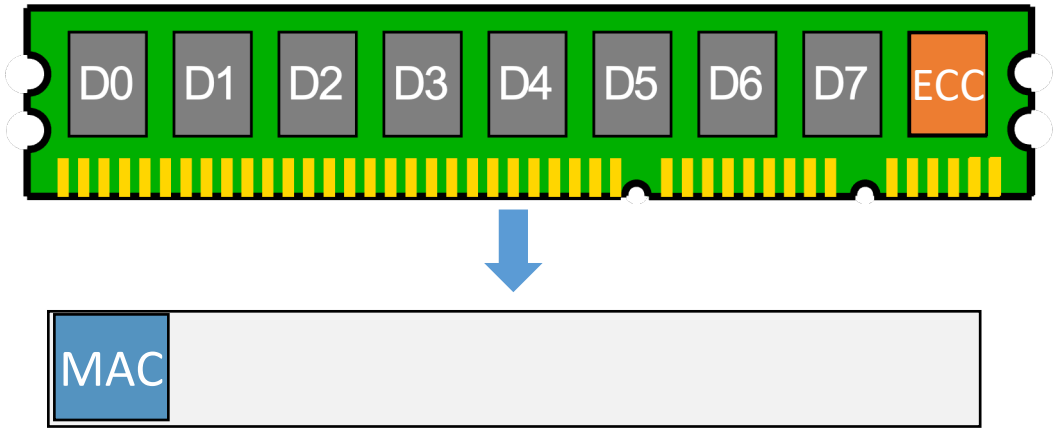
Detects — *2-Bit Errors*

Insight: Leverage ECC-DIMM To Reduce MAC Overheads

ECC-DIMM for Reliability



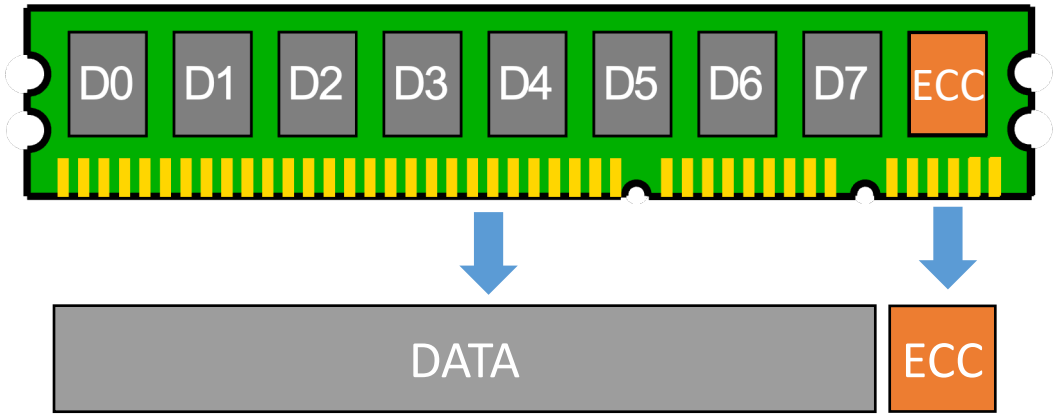
Security with MAC



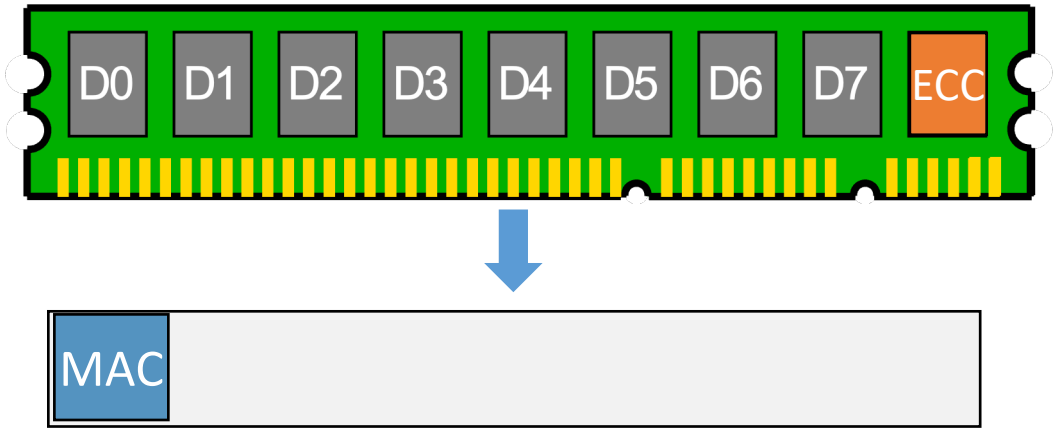
	ECC	MAC
Fetched	*Same Access as Data*	*Separate Access*
Detects	*2-Bit Errors*	

Insight: Leverage ECC-DIMM To Reduce MAC Overheads

ECC-DIMM for Reliability



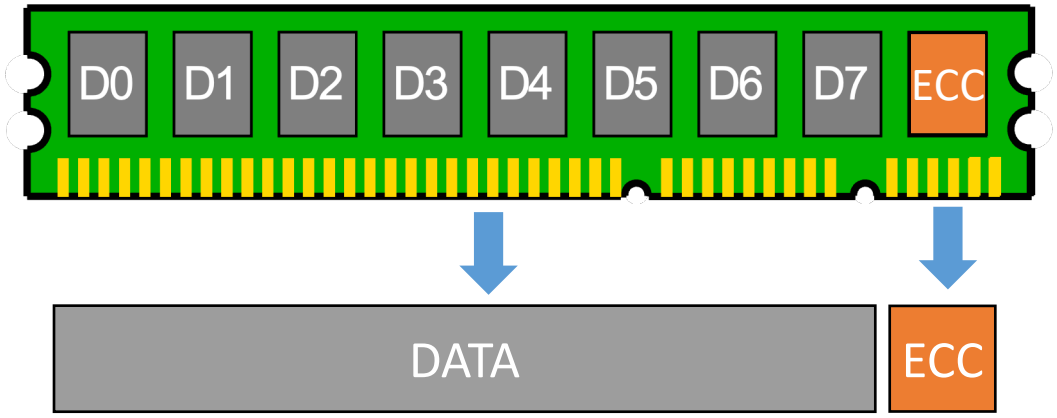
Security with MAC



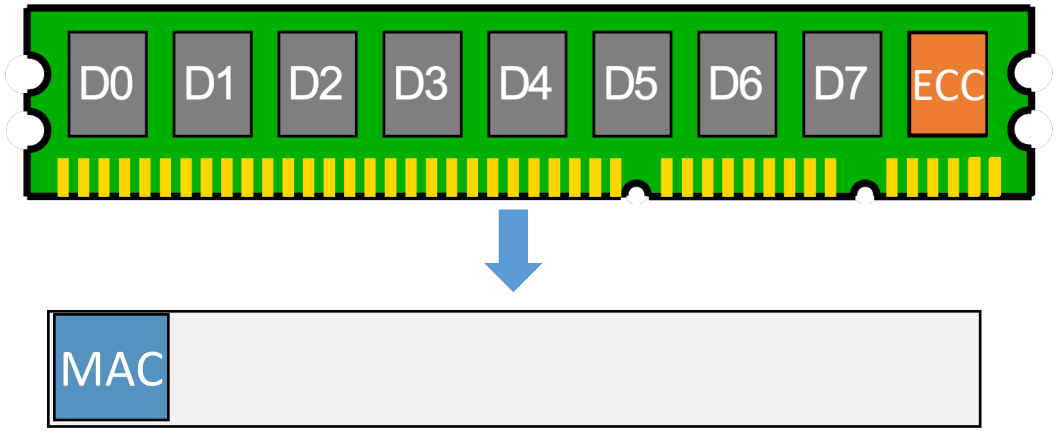
	ECC	MAC
Fetched	*Same Access as Data*	*Separate Access*
Detects	*2-Bit Errors*	*Any Modifications*

Insight: Leverage ECC-DIMM To Reduce MAC Overheads

ECC-DIMM for Reliability



Security with MAC

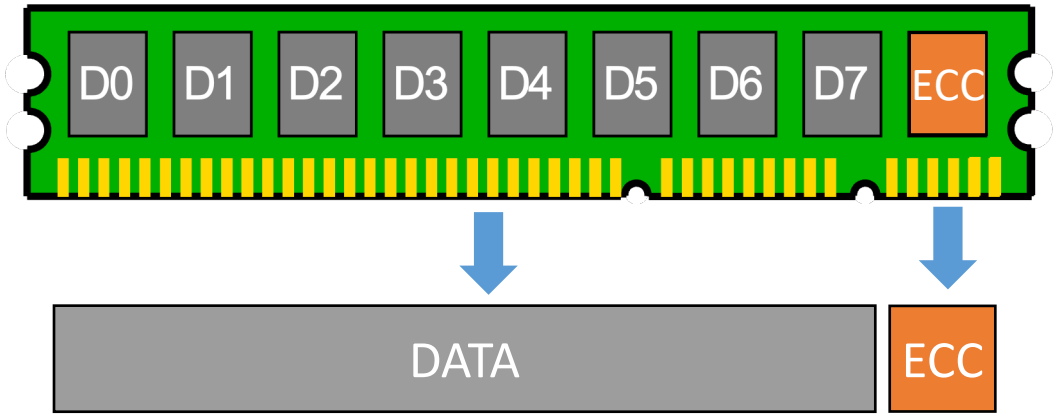


	ECC	MAC
Fetched	*Same Access as Data*	*Separate Access*
Detects	*2-Bit Errors*	*Any Modifications*

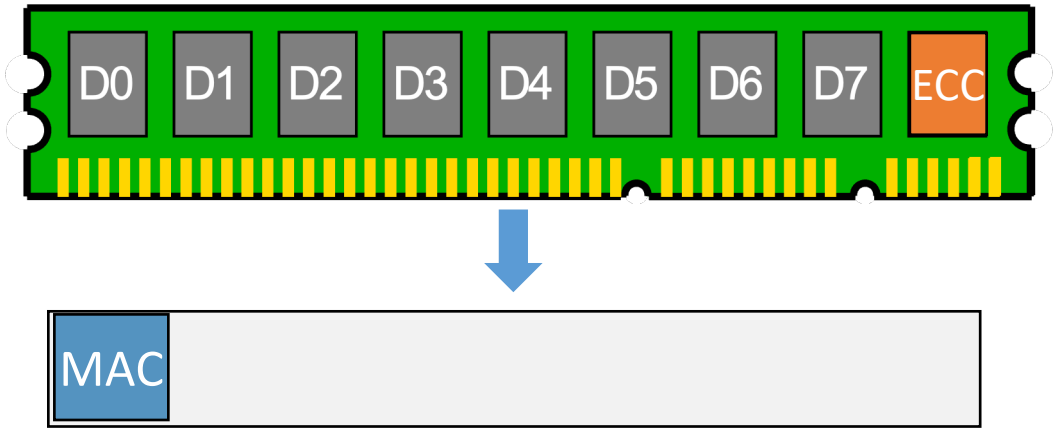
Replace

Insight: Leverage ECC-DIMM To Reduce MAC Overheads

ECC-DIMM for Reliability



Security with MAC



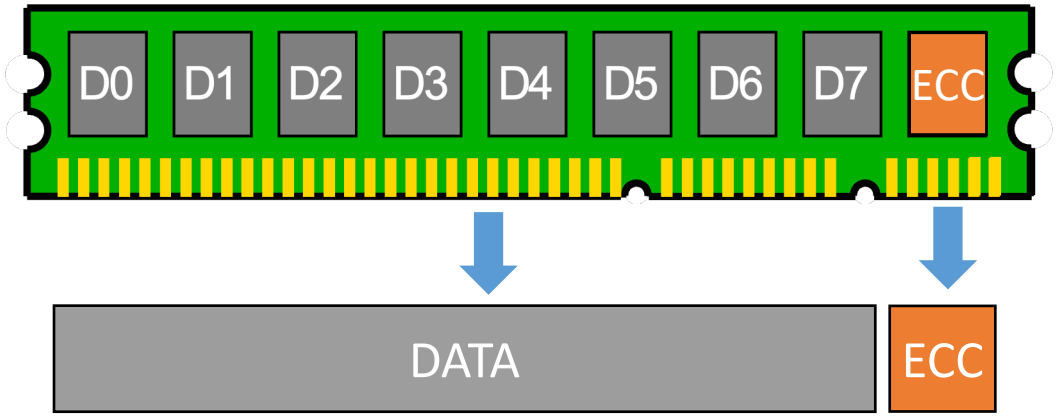
	ECC	MAC
Fetched	*Same Access as Data*	*Separate Access*
Detects	*2-Bit Errors*	*Any Modifications*

Replace

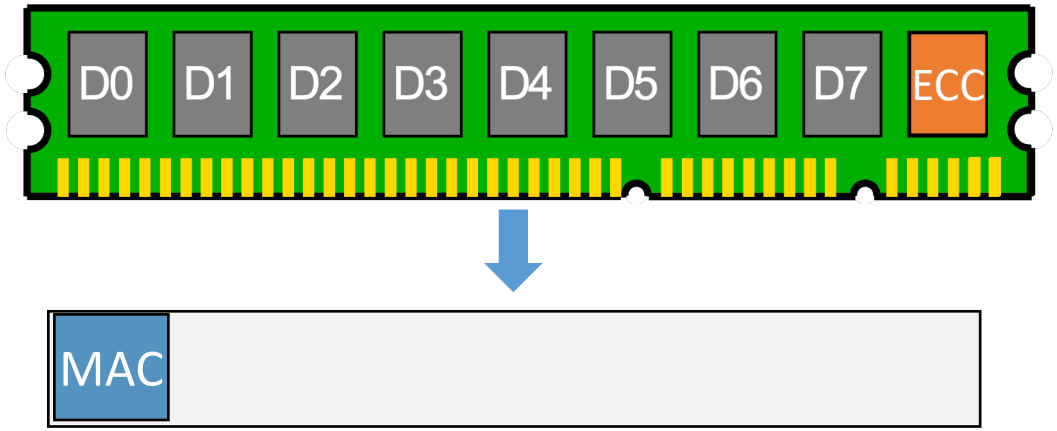
Still need ECC for Correction!

Insight: Leverage ECC-DIMM To Reduce MAC Overheads

ECC-DIMM for Reliability



Security with MAC



	ECC	MAC
Fetched	*Same Access as Data*	*Separate Access*
Detects	*2-Bit Errors*	*Any Modifications*

Co-Design

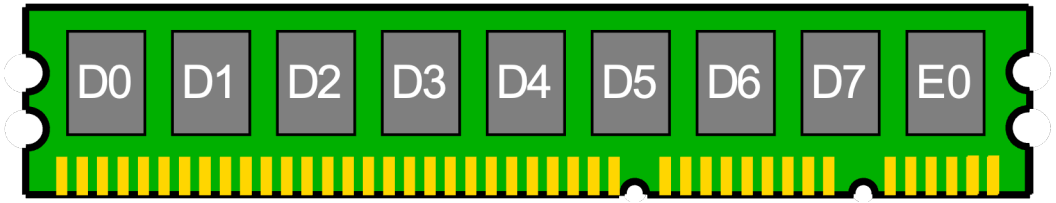
Still need ECC for Correction!

Agenda

- Introduction & Background
- **Synergy Design**
- Evaluation

SYNERGY Co-locates MAC & Data for Performance

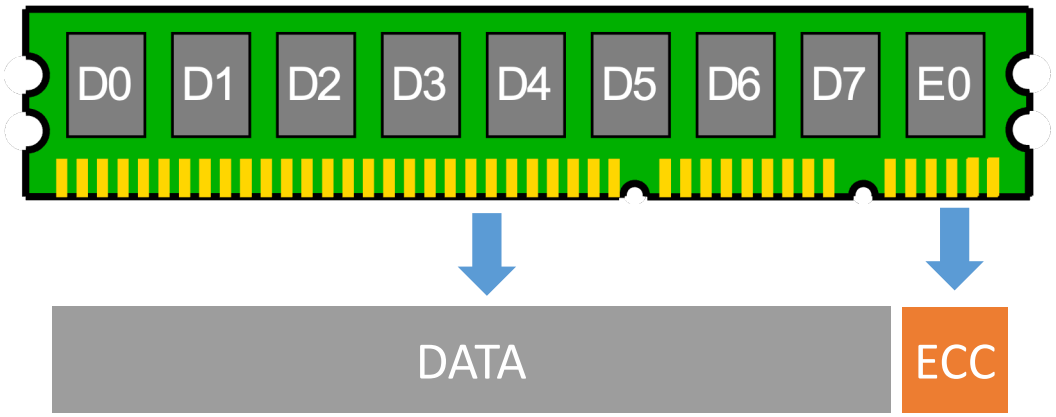
Data Cachelines



Non-Secure

SYNERGY Co-locates MAC & Data for Performance

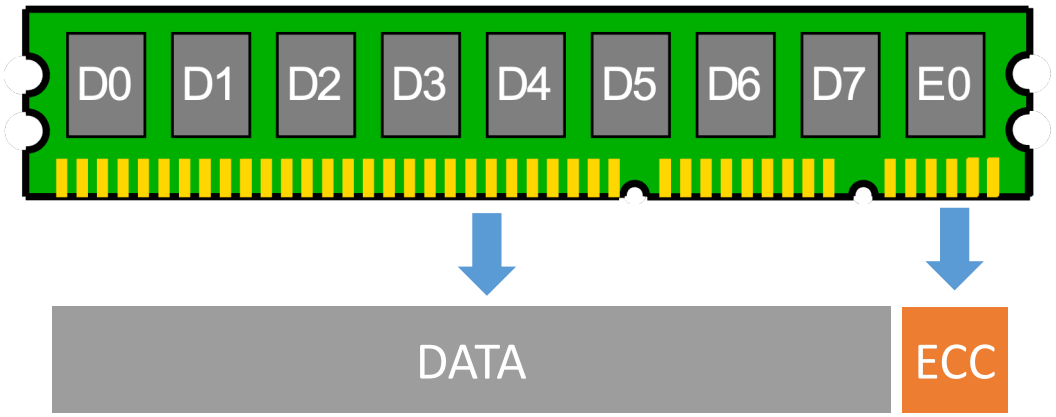
Data Cachelines



Non-Secure - 1 access

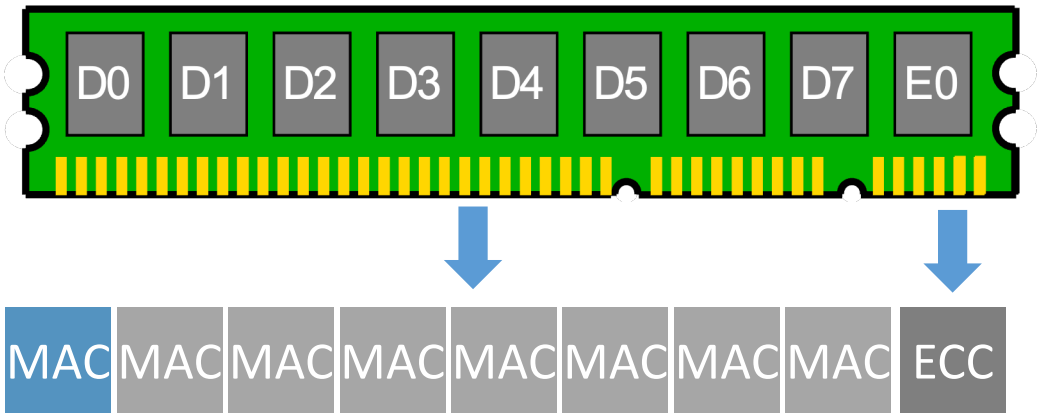
SYNERGY Co-locates MAC & Data for Performance

Data Cachelines



Non-Secure - 1 access

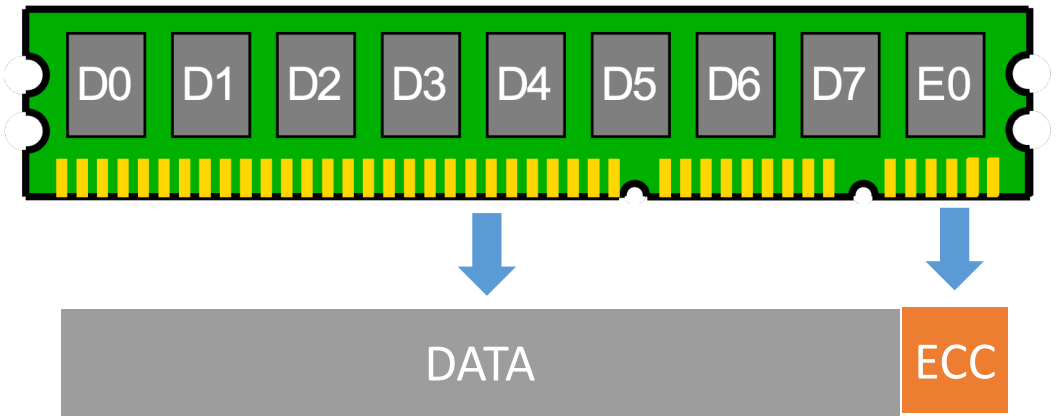
MAC Cachelines



Secure Memory

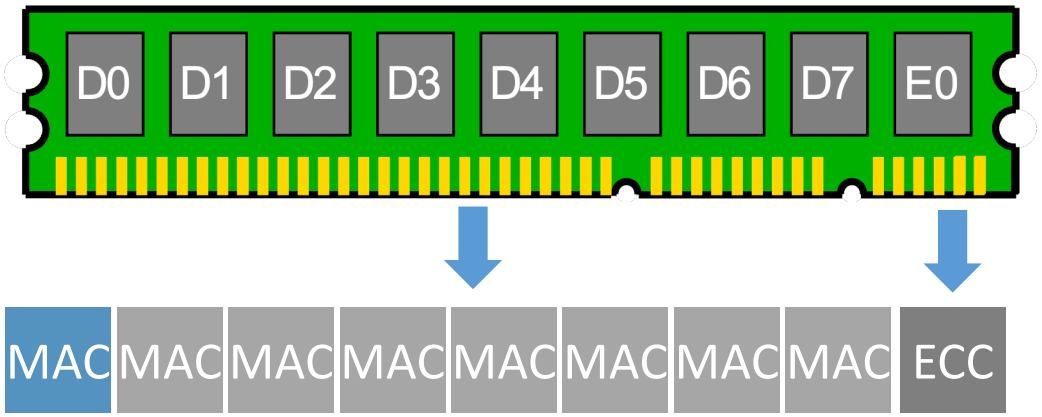
SYNERGY Co-locates MAC & Data for Performance

Data Cachelines



Non-Secure - 1 access

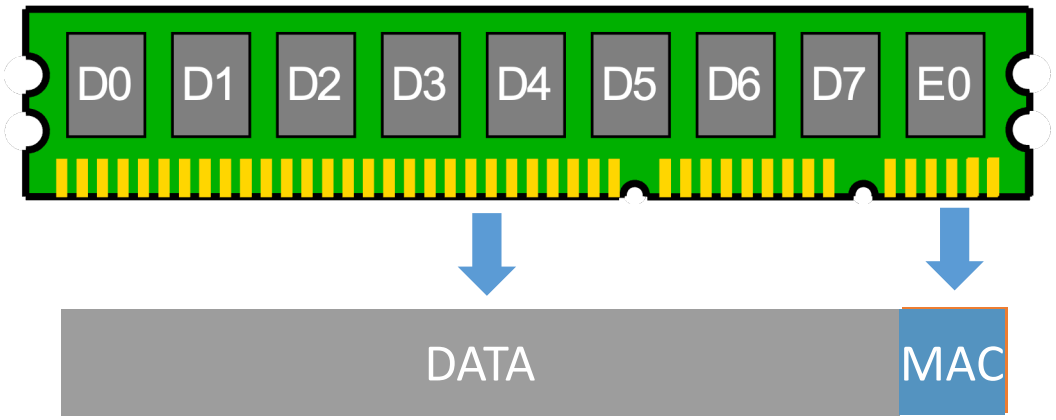
MAC Cachelines



Secure Memory - 2 accesses

SYNERGY Co-locates MAC & Data for Performance

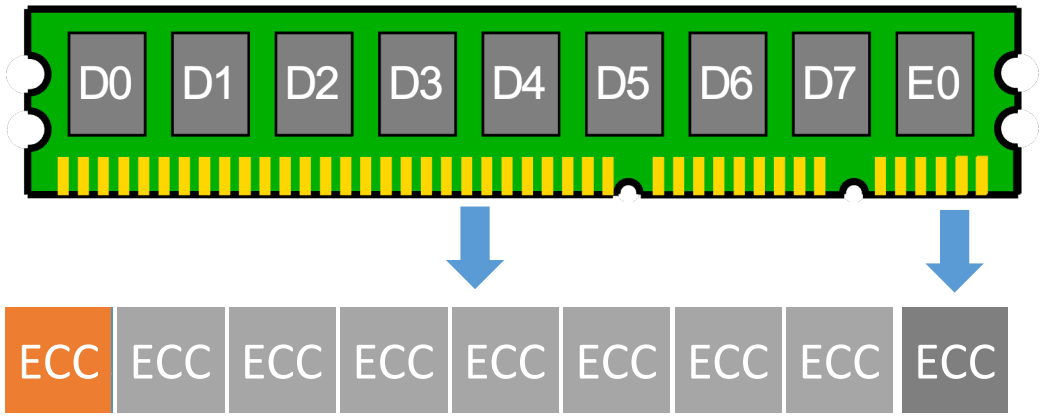
Data Cachelines



Non-Secure - 1 access

SYNERGY

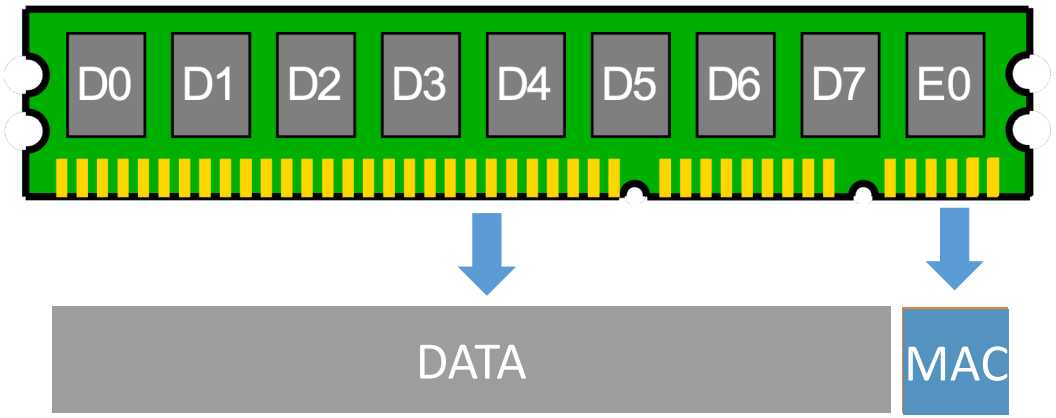
ECC Cachelines



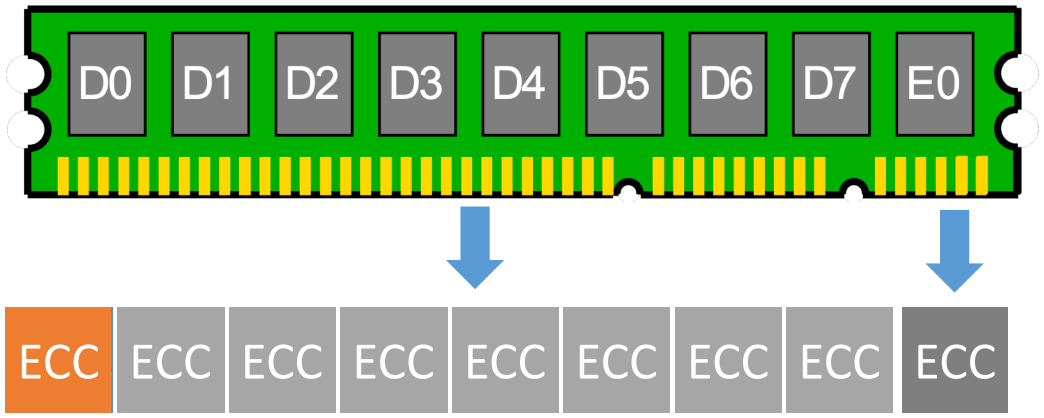
Secure Memory - 2 accesses

SYNERGY Co-locates MAC & Data for Performance

Data Cachelines



ECC Cachelines



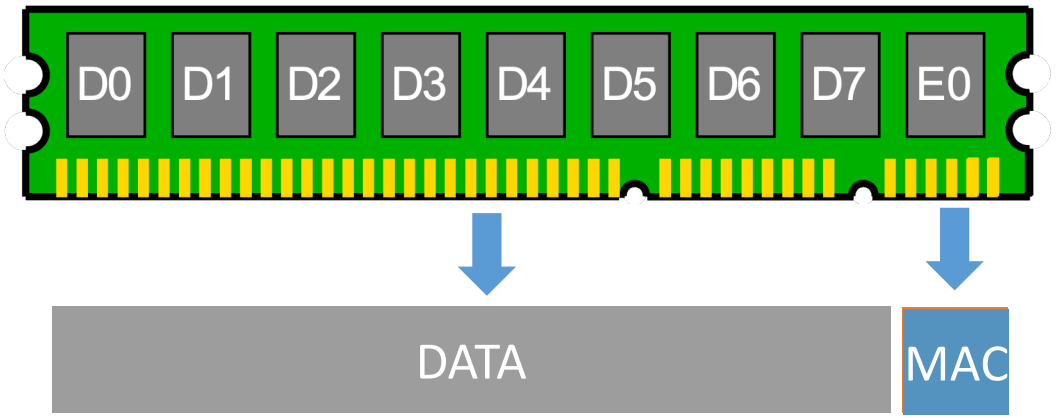
Non-Secure - 1 access

Secure Memory - 2 accesses

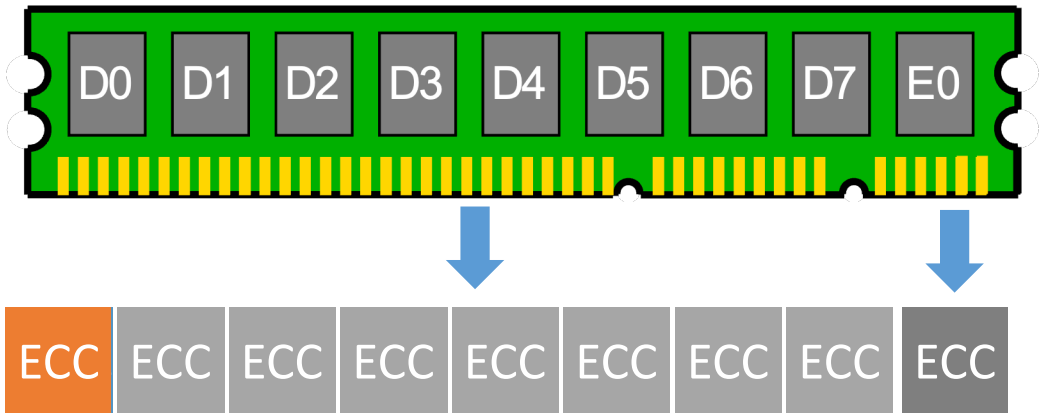
SYNERGY - 1 access for reads, unless error *(rare)*

SYNERGY Co-locates MAC & Data for Performance

Data Cachelines



ECC Cachelines



Non-Secure - 1 access

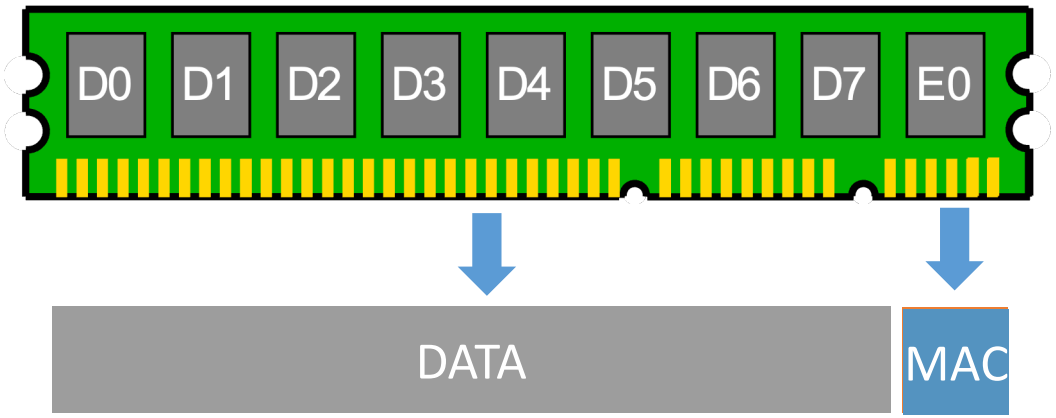
Secure Memory - 2 accesses

SYNERGY

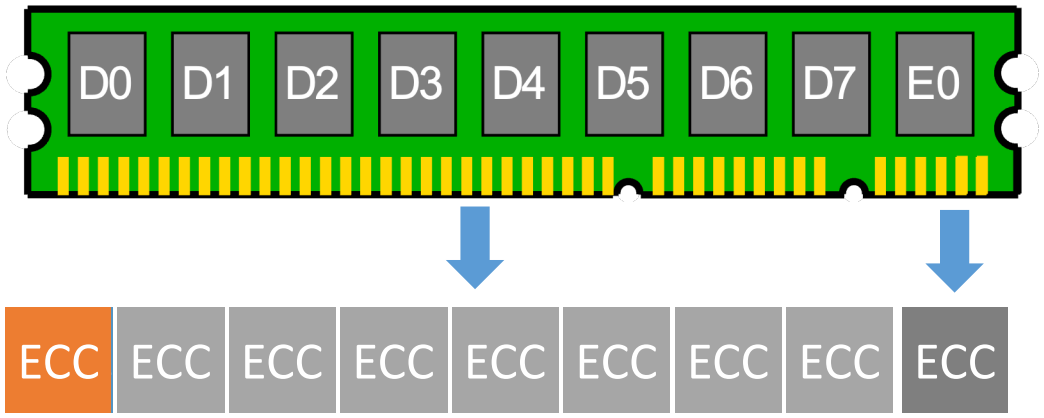
- 1 access for reads, unless error *(rare)*
- 2 accesses for writes

SYNERGY Co-locates MAC & Data for Performance

Data Cachelines



ECC Cachelines



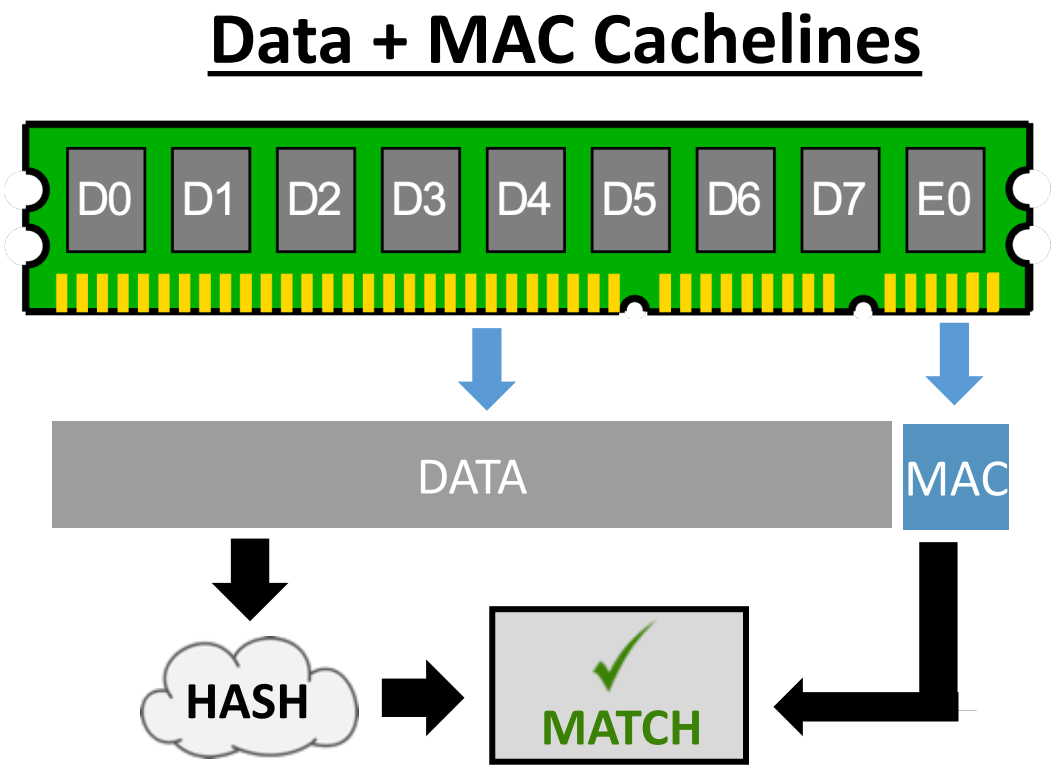
Non-Secure - 1 access

Secure Memory - 2 accesses

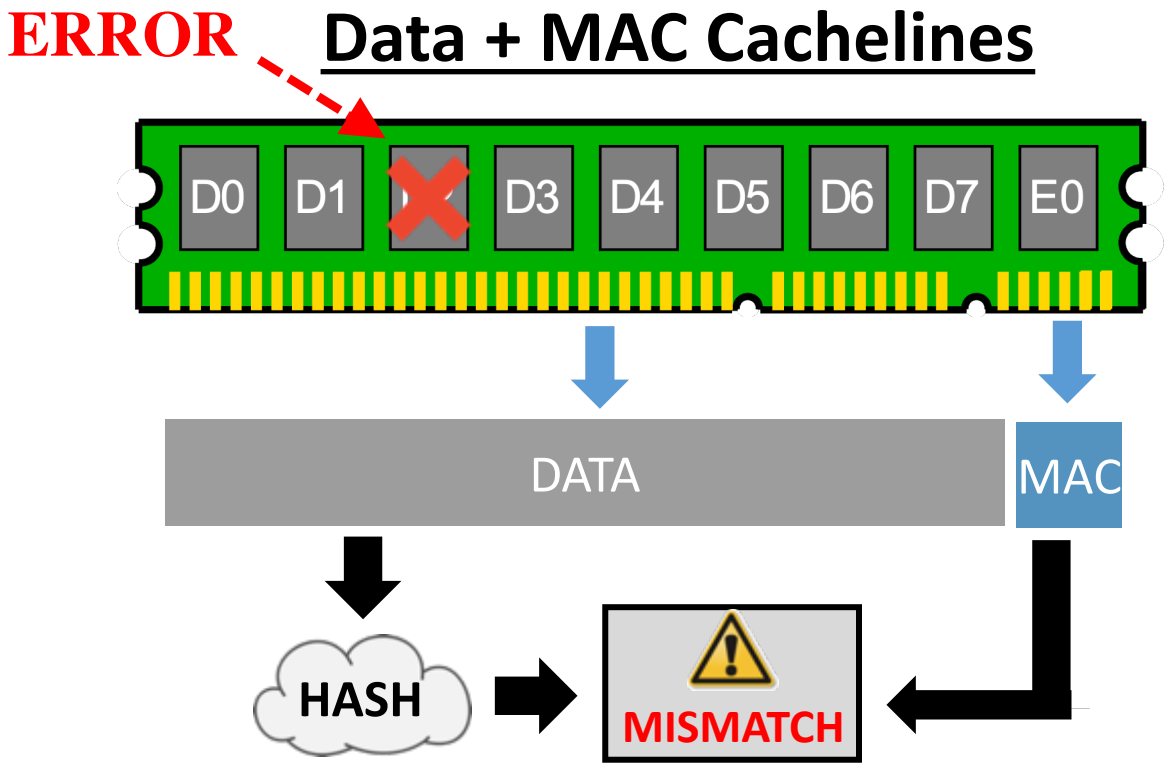
SYNERGY
- 1 access for reads, unless error *(rare)*
- 2 accesses for writes

Synergy avoids extra MAC lookups and improves performance, without any additional storage

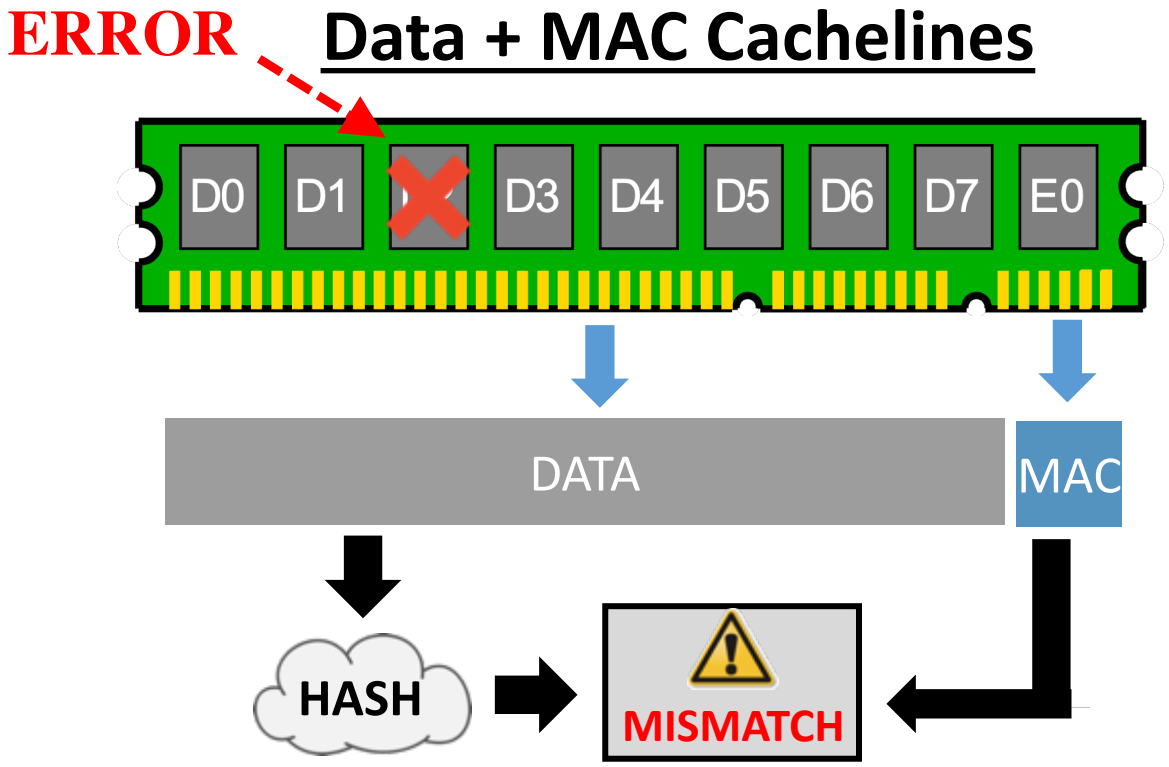
SYNERGY uses MAC for Error Detection



SYNERGY uses MAC for Error Detection

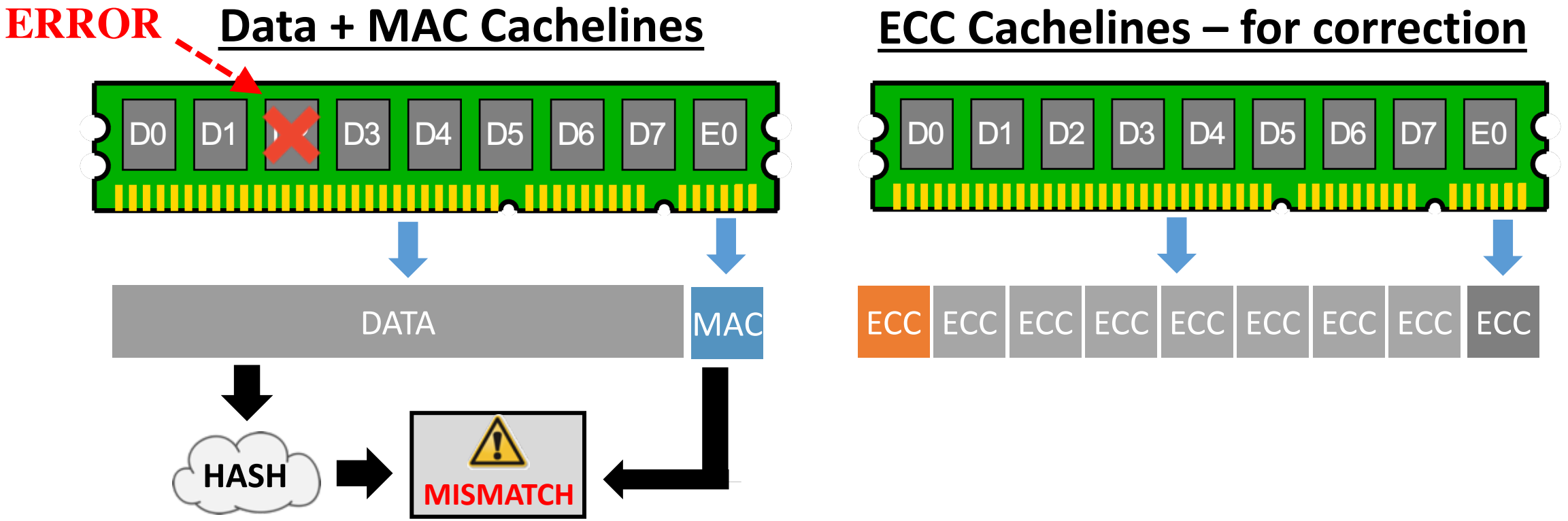


SYNERGY uses MAC for Error Detection



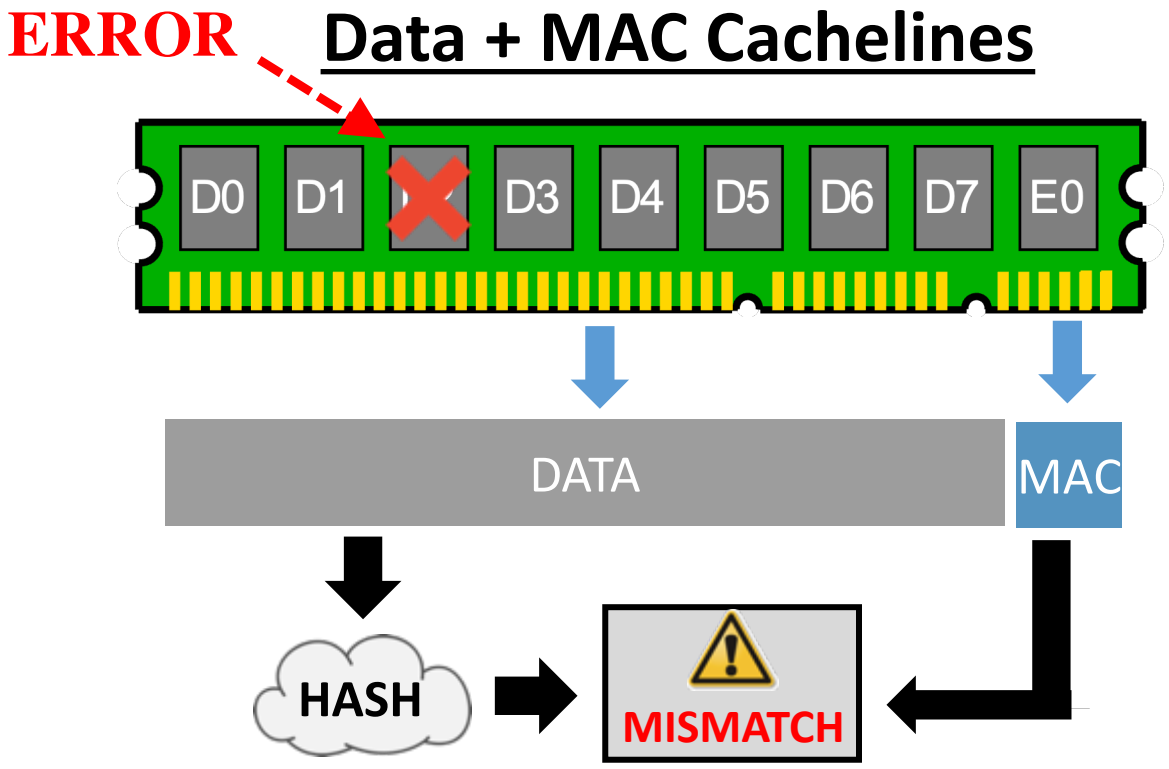
MACs have strong error detection ability

SYNERGY uses MAC for Error Detection

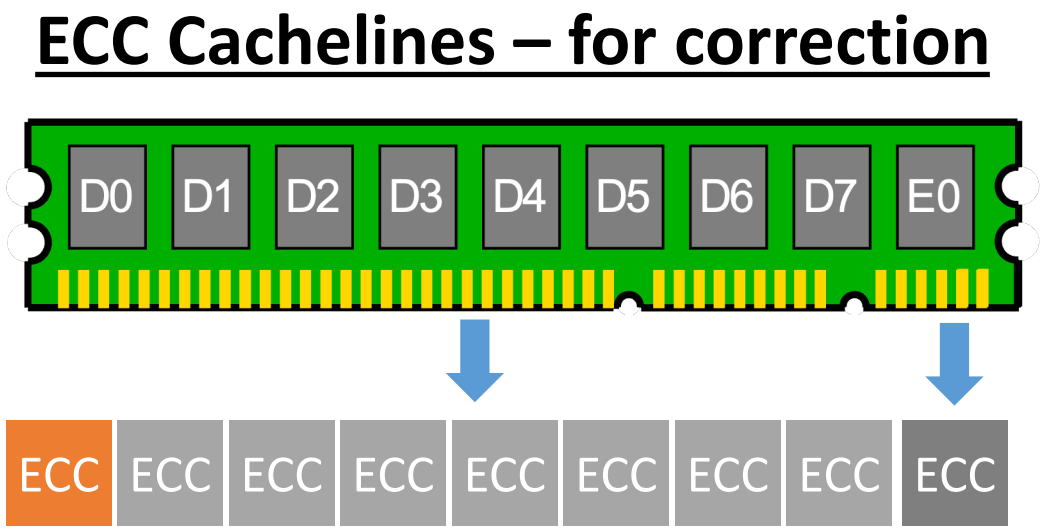


MACs have strong error detection ability

SYNERGY uses MAC for Error Detection

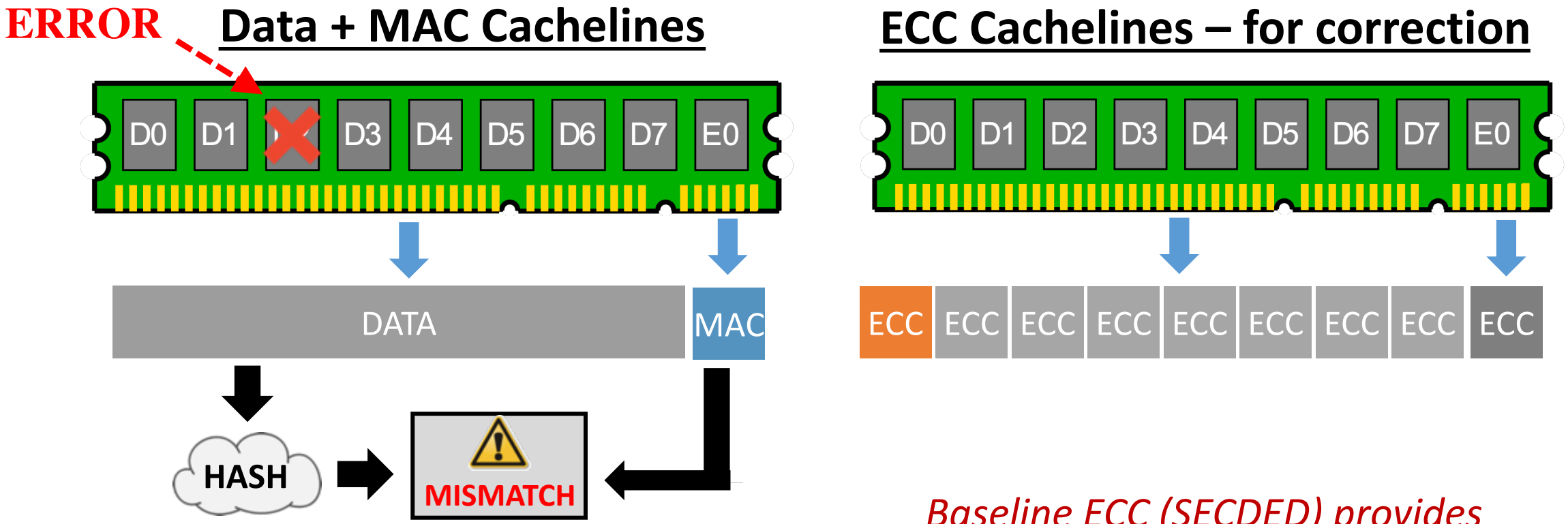


MACs have strong error detection ability



Baseline ECC (SECDED) provides single-bit error correction

SYNERGY uses MAC for Error Detection



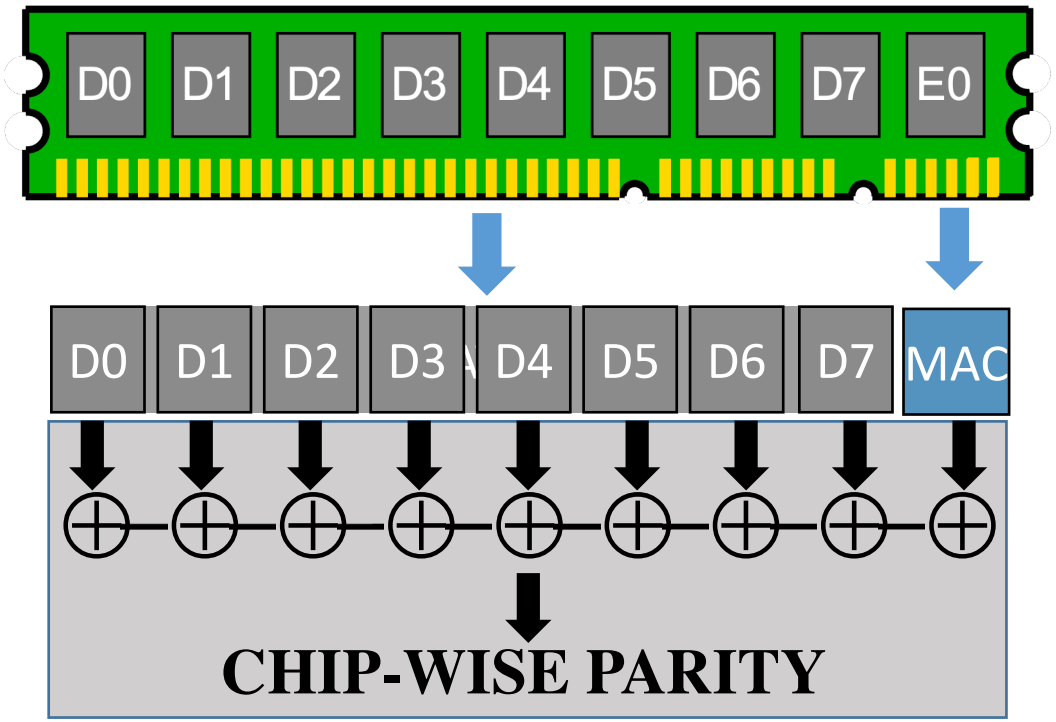
MACs have strong error detection ability

Baseline ECC (SECDED) provides single-bit error correction

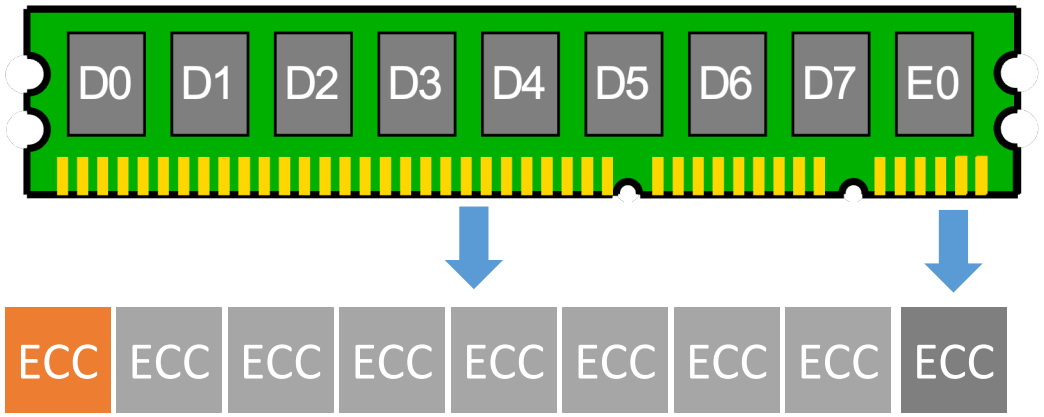
Can We Achieve Stronger Error-Correction with Co-Design?

Parity Can Reconstruct Chip With Failure

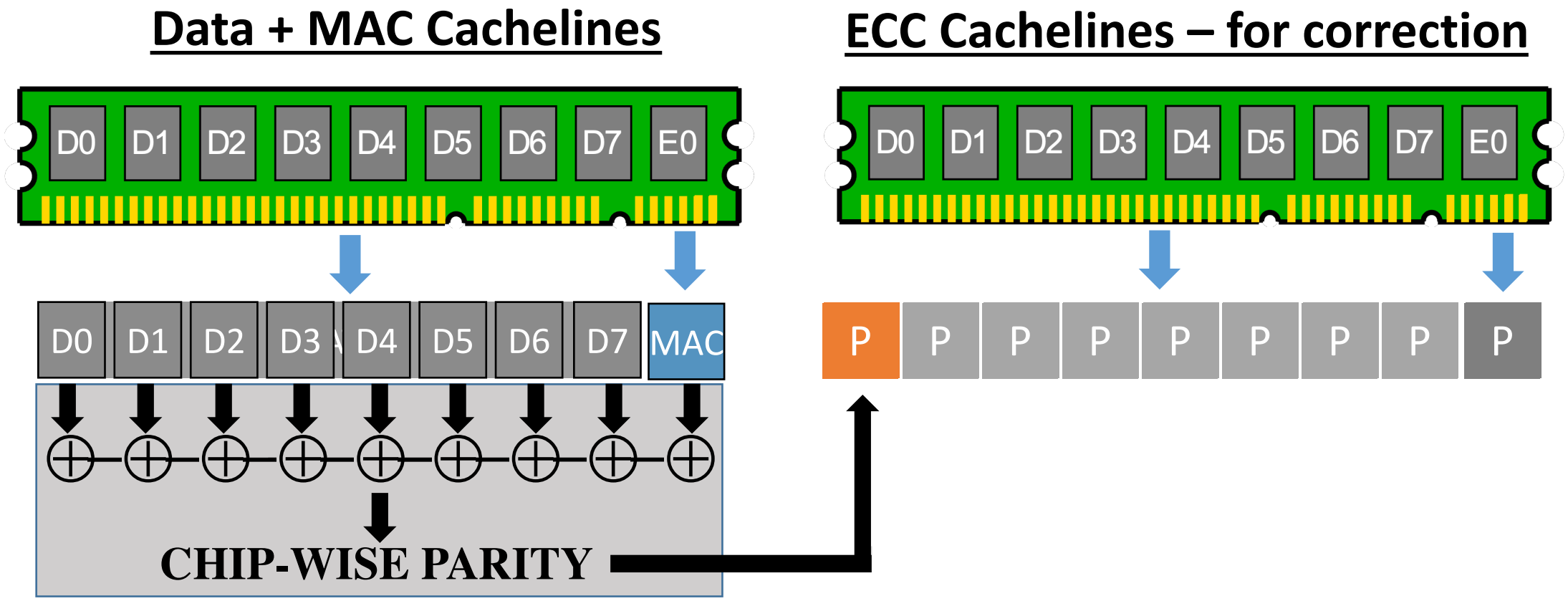
Data + MAC Cachelines



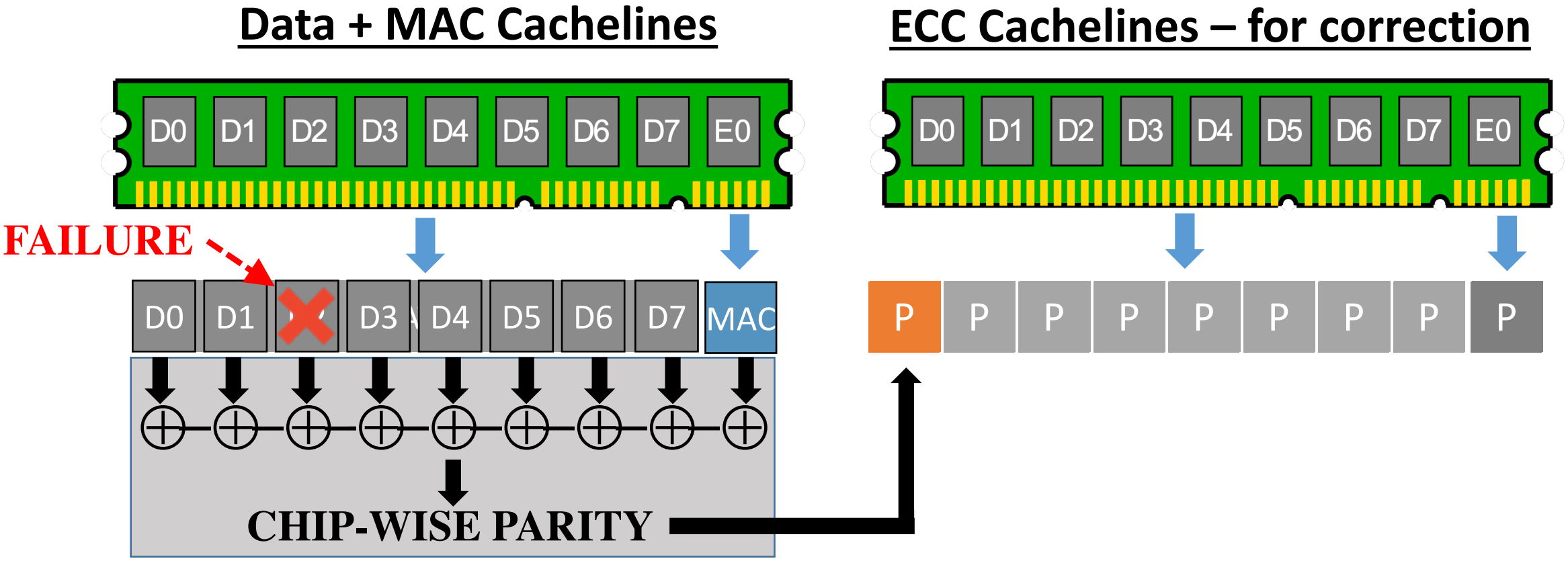
ECC Cachelines – for correction



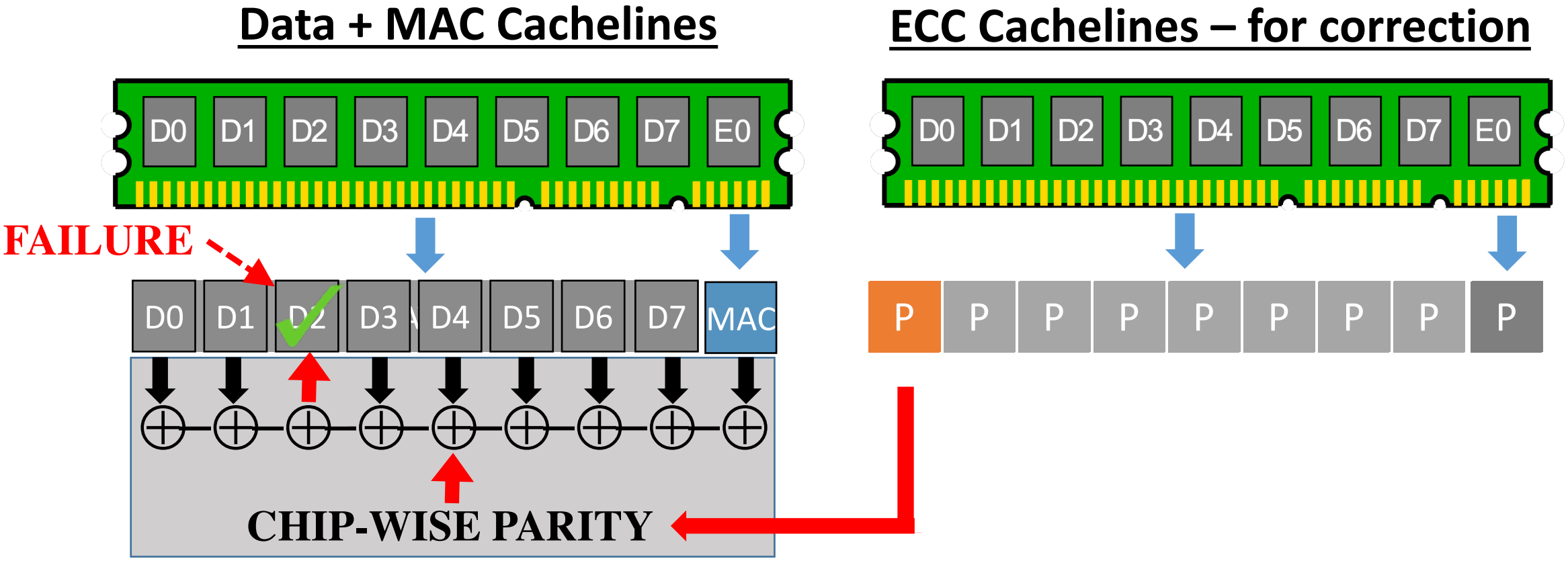
Parity Can Reconstruct Chip With Failure



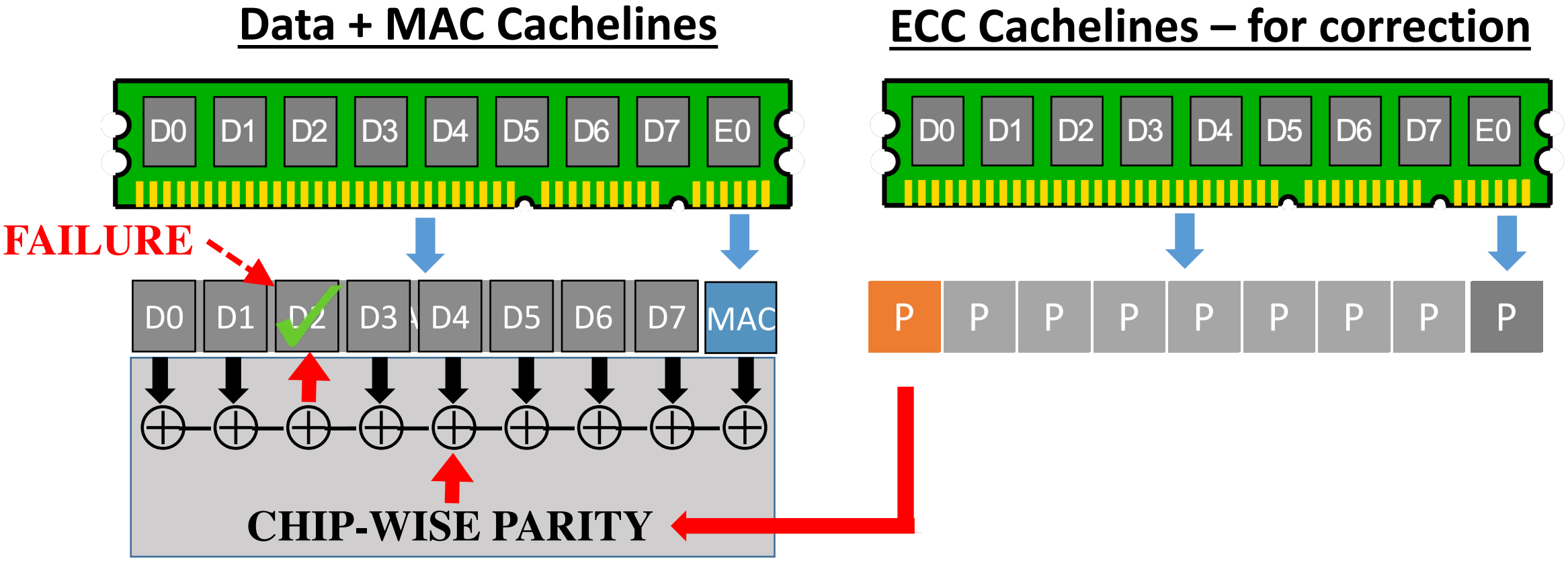
Parity Can Reconstruct Chip With Failure



Parity Can Reconstruct Chip With Failure

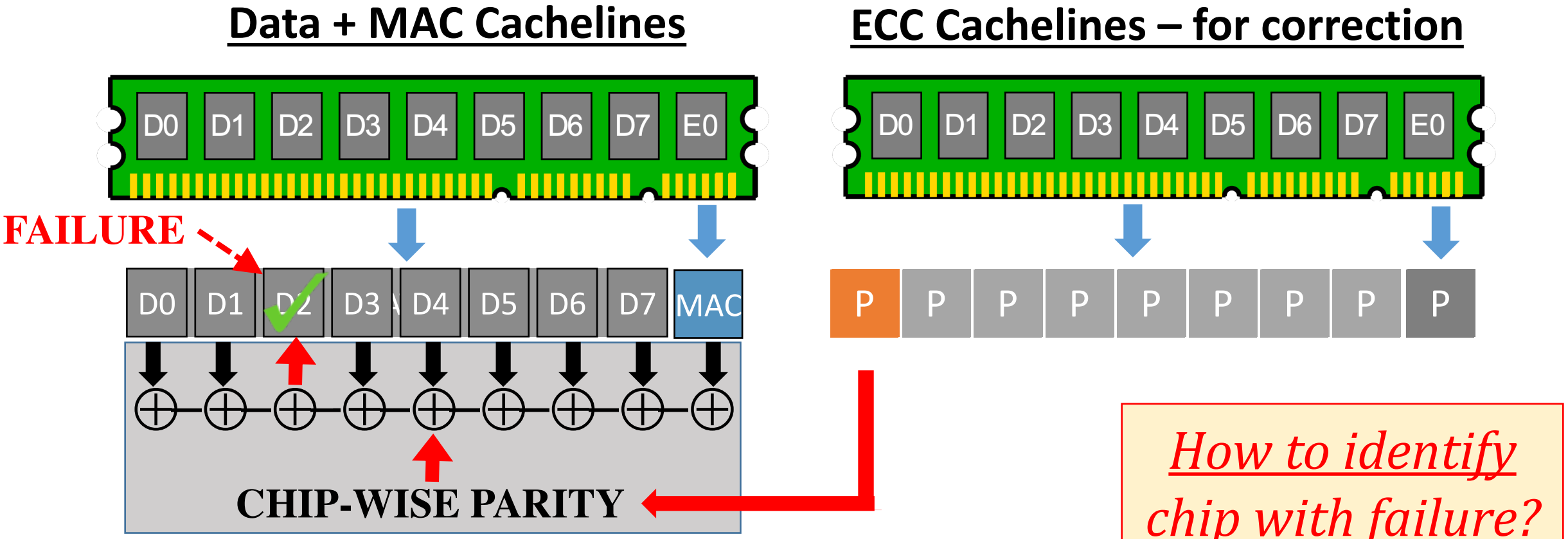


Parity Can Reconstruct Chip With Failure



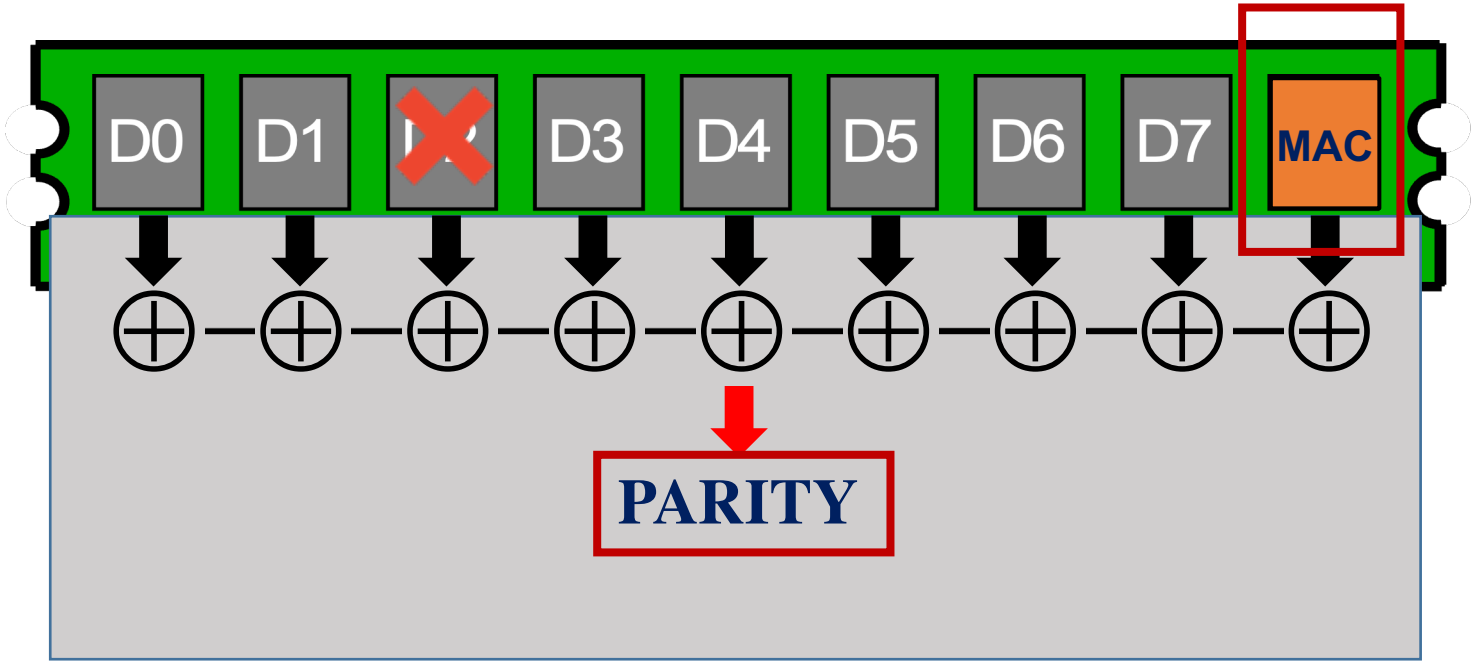
Parity Can Correct Large-Granularity Failures, If Chip With Failure Known

Parity Can Reconstruct Chip With Failure



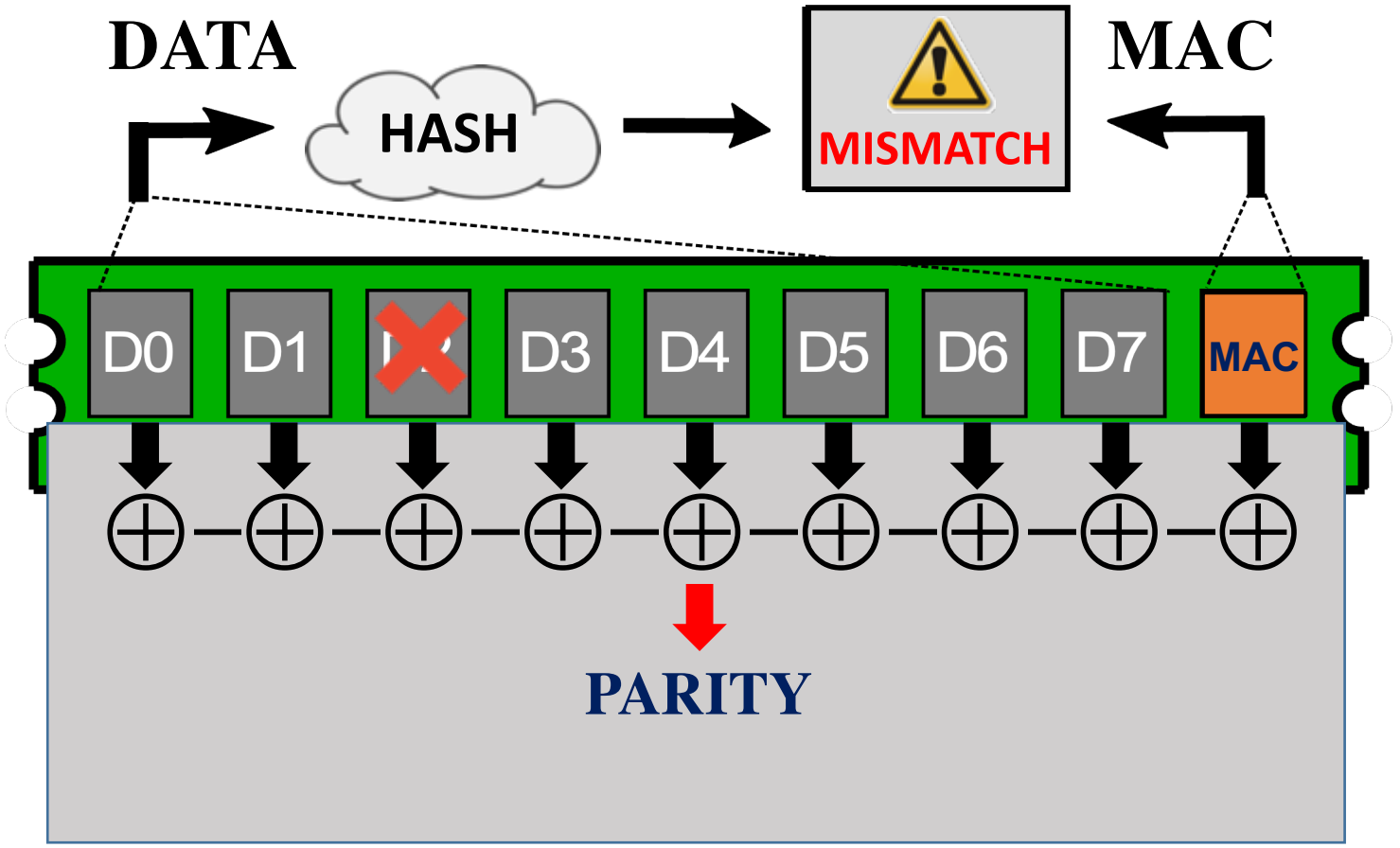
Parity Can Correct Large-Granularity Failures, If Chip With Failure Known

SYNERGY uses MAC + Parity for Error Correction



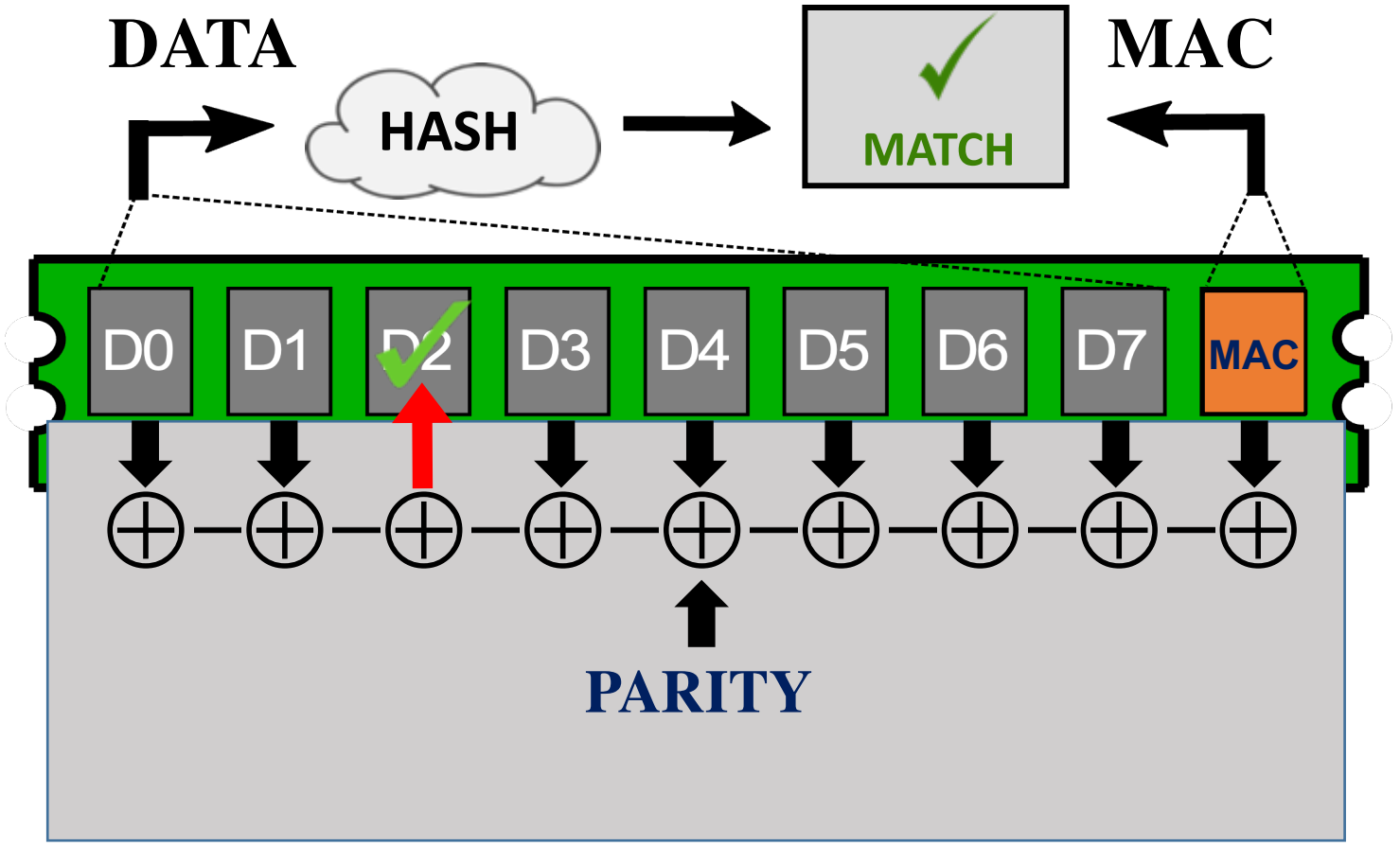
SYNERGY uses MAC + Parity for Error Correction

MAC can detect when data cacheline is free from error



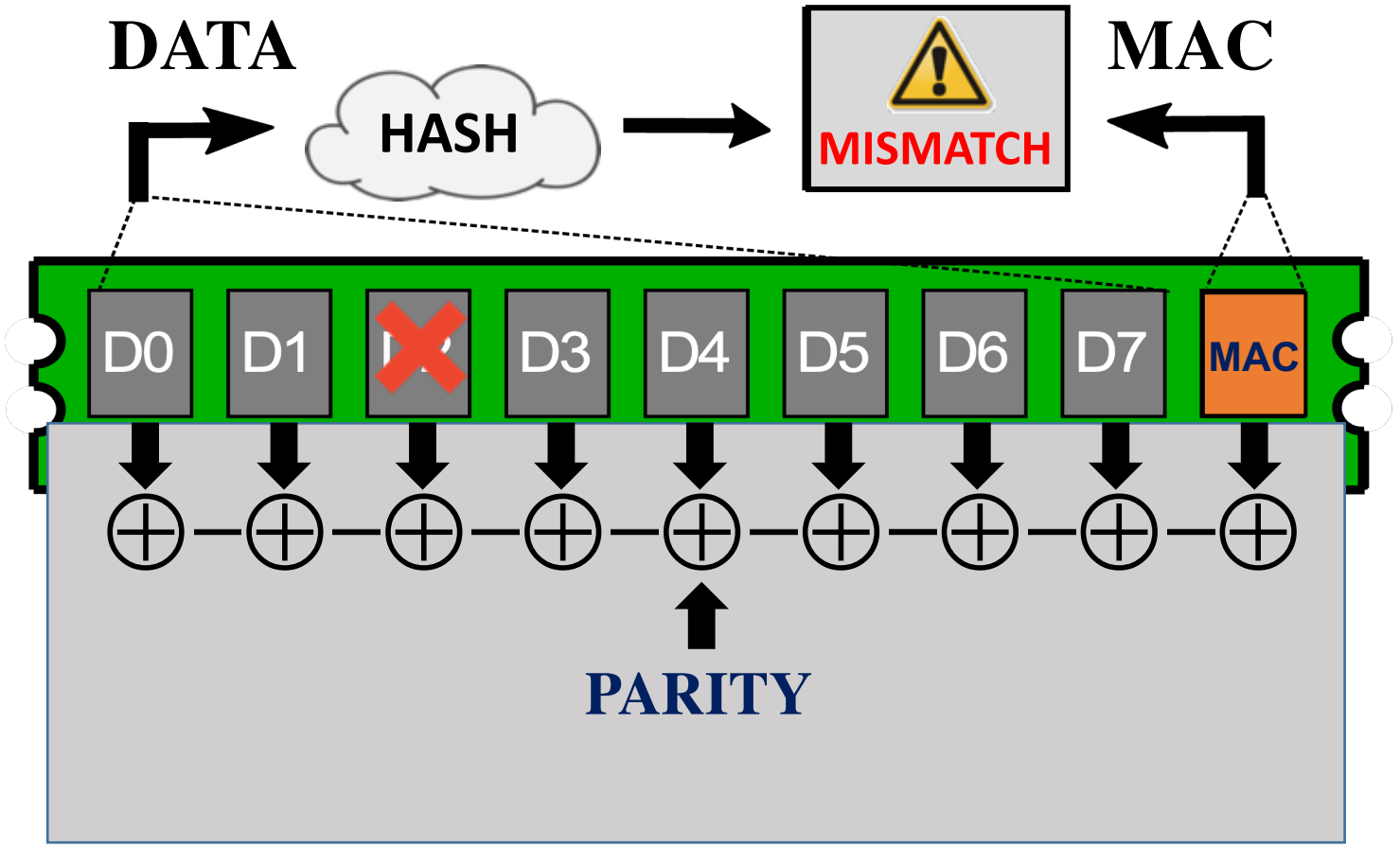
SYNERGY uses MAC + Parity for Error Correction

MAC can detect when data cacheline is free from error



SYNERGY uses MAC + Parity for Error Correction

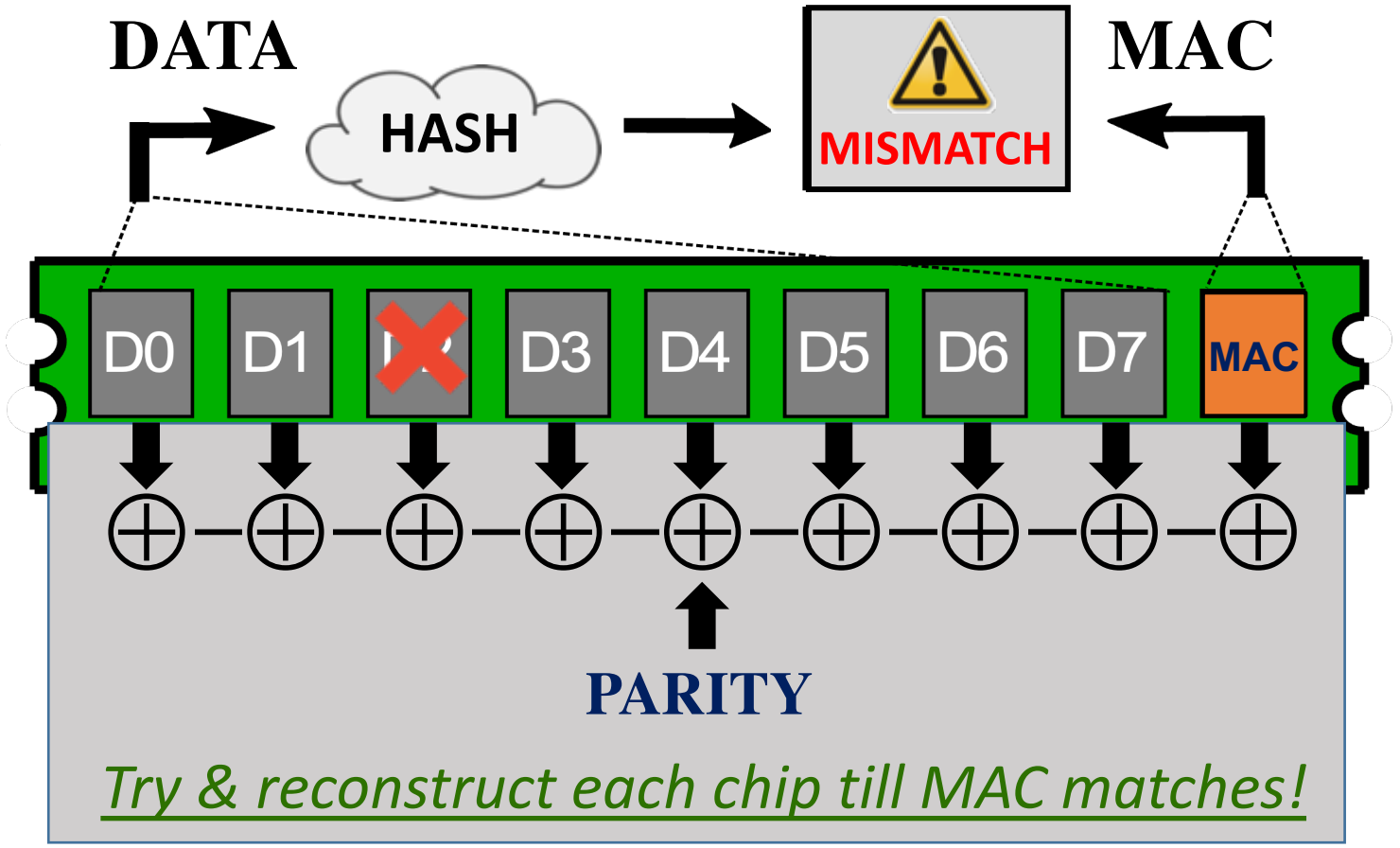
MAC can detect when data cacheline is free from error



How to identify chip with failure?

SYNERGY uses MAC + Parity for Error Correction

MAC can detect when data cacheline is free from error

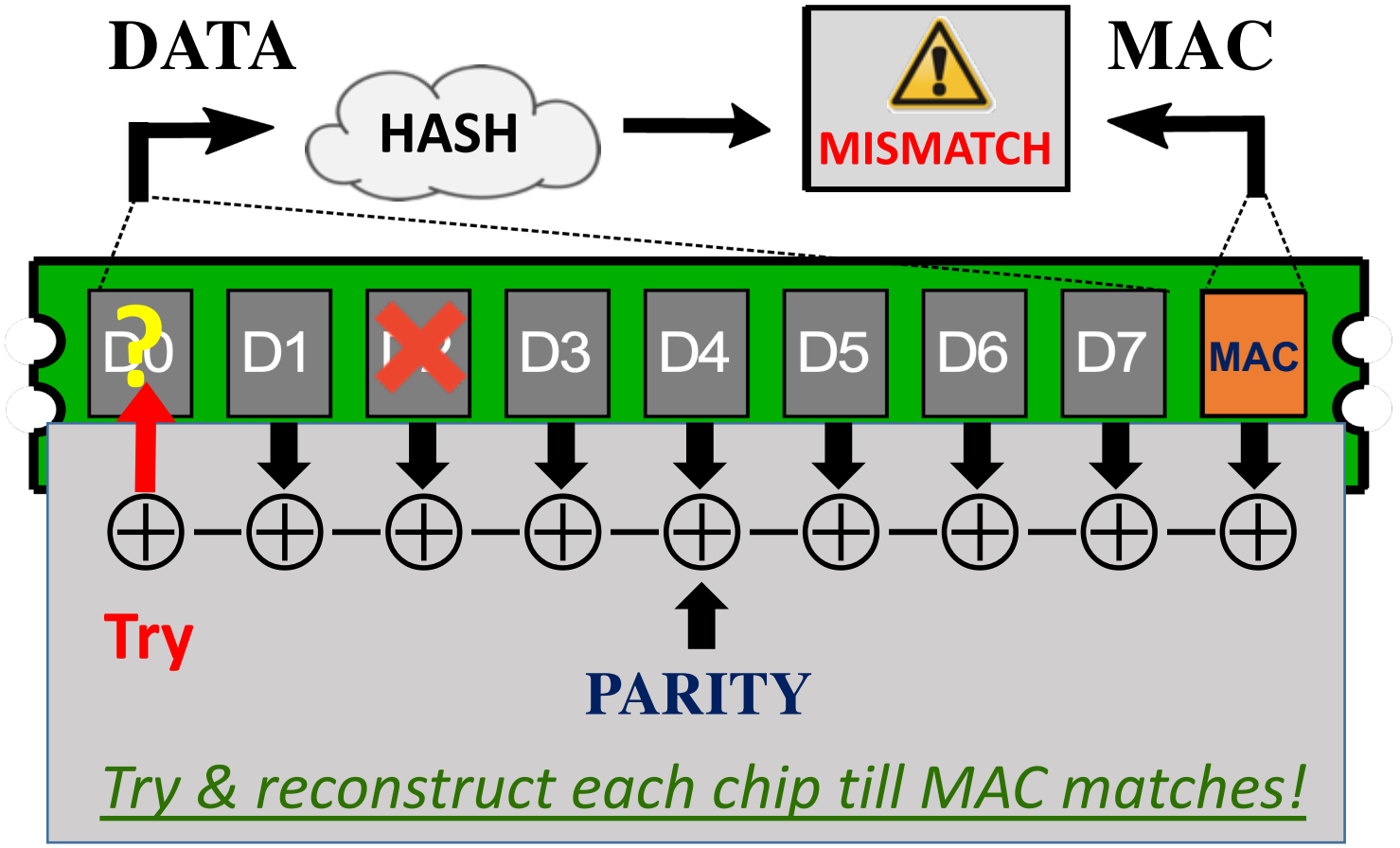


How to identify chip with failure?

Try & reconstruct each chip till MAC matches!

SYNERGY uses MAC + Parity for Error Correction

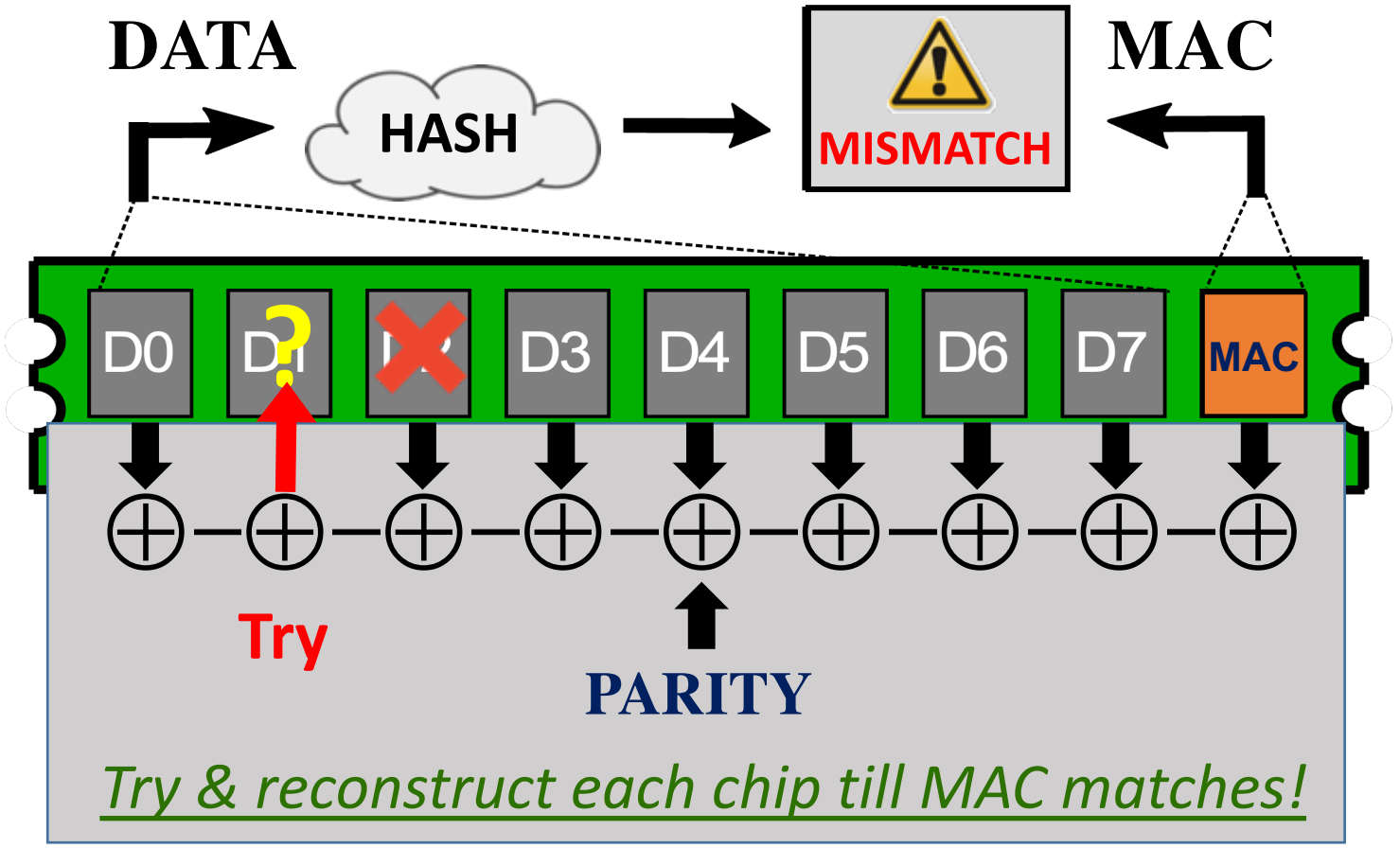
MAC can detect when data cacheline is free from error



How to identify chip with failure?

SYNERGY uses MAC + Parity for Error Correction

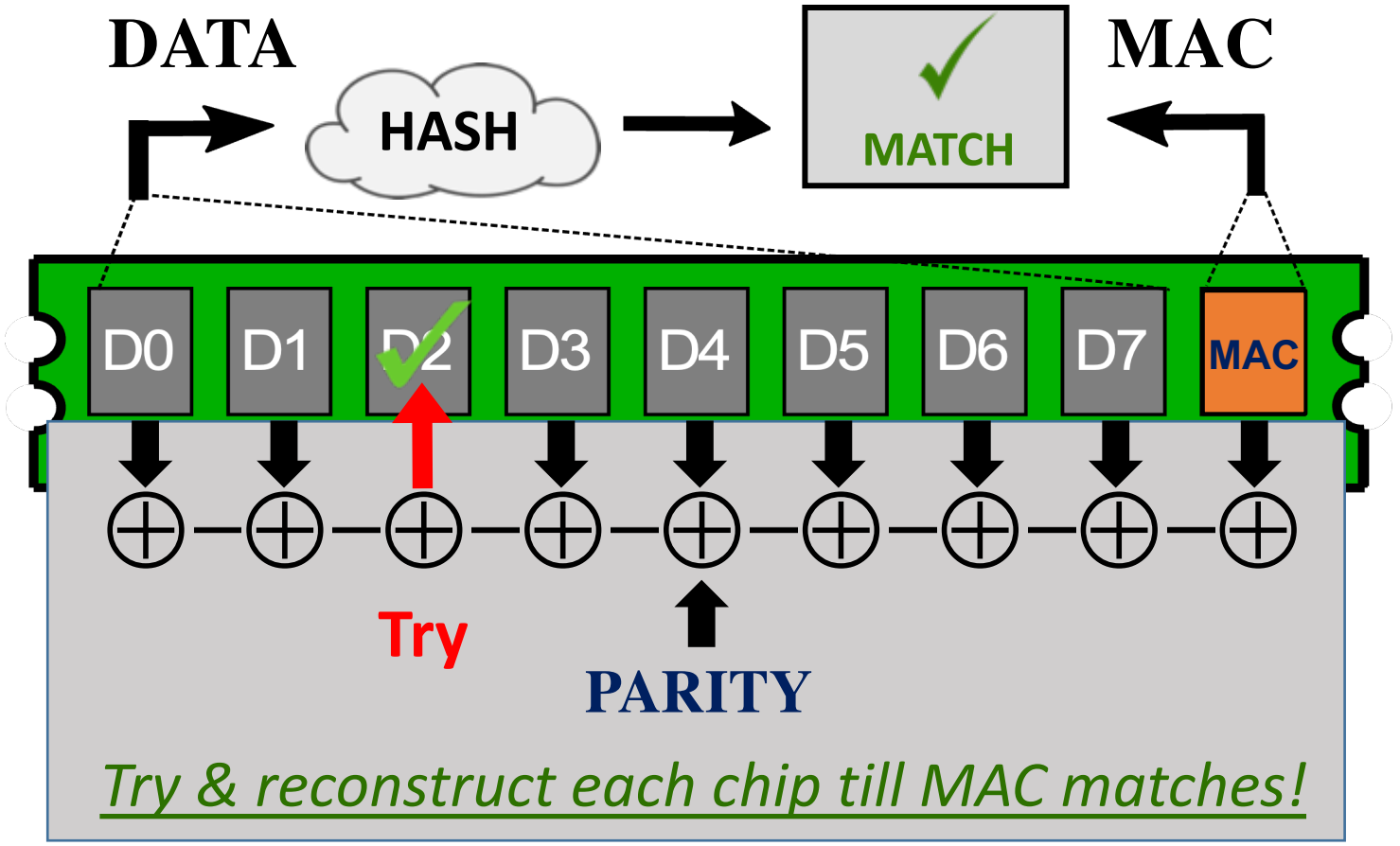
MAC can detect when data cacheline is free from error



How to identify chip with failure?

SYNERGY uses MAC + Parity for Error Correction

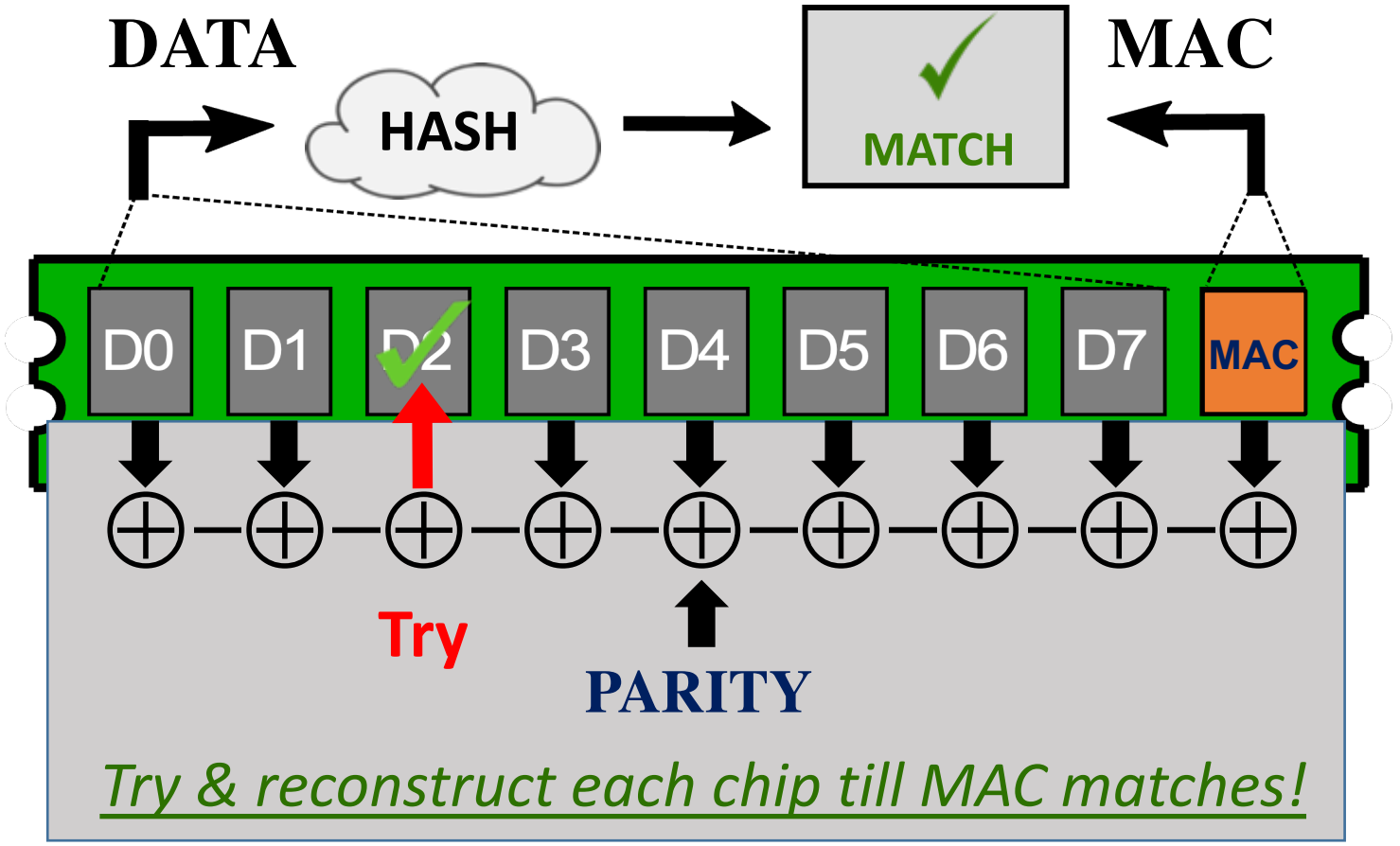
MAC can detect when data cacheline is free from error



How to identify chip with failure?

SYNERGY uses MAC + Parity for Error Correction

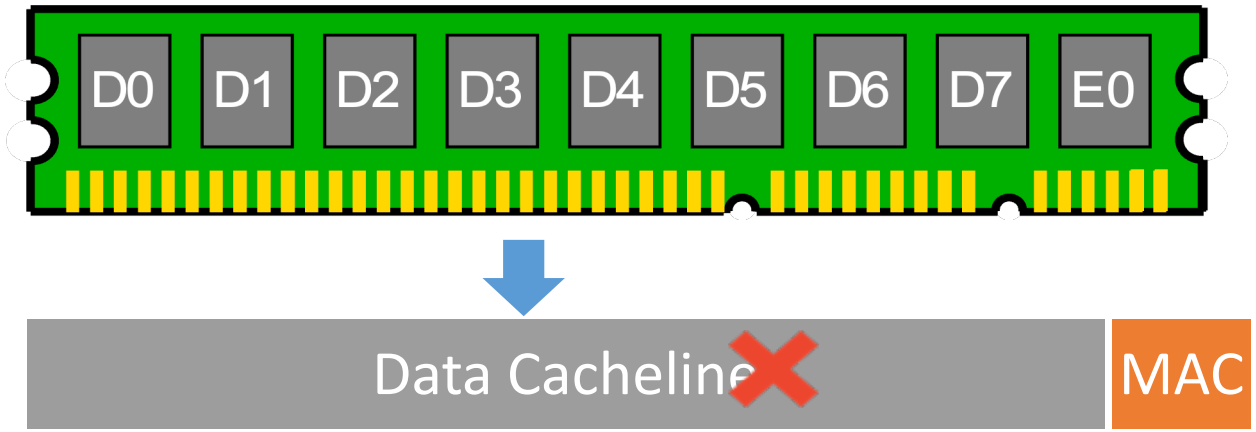
MAC can detect when data cacheline is free from error



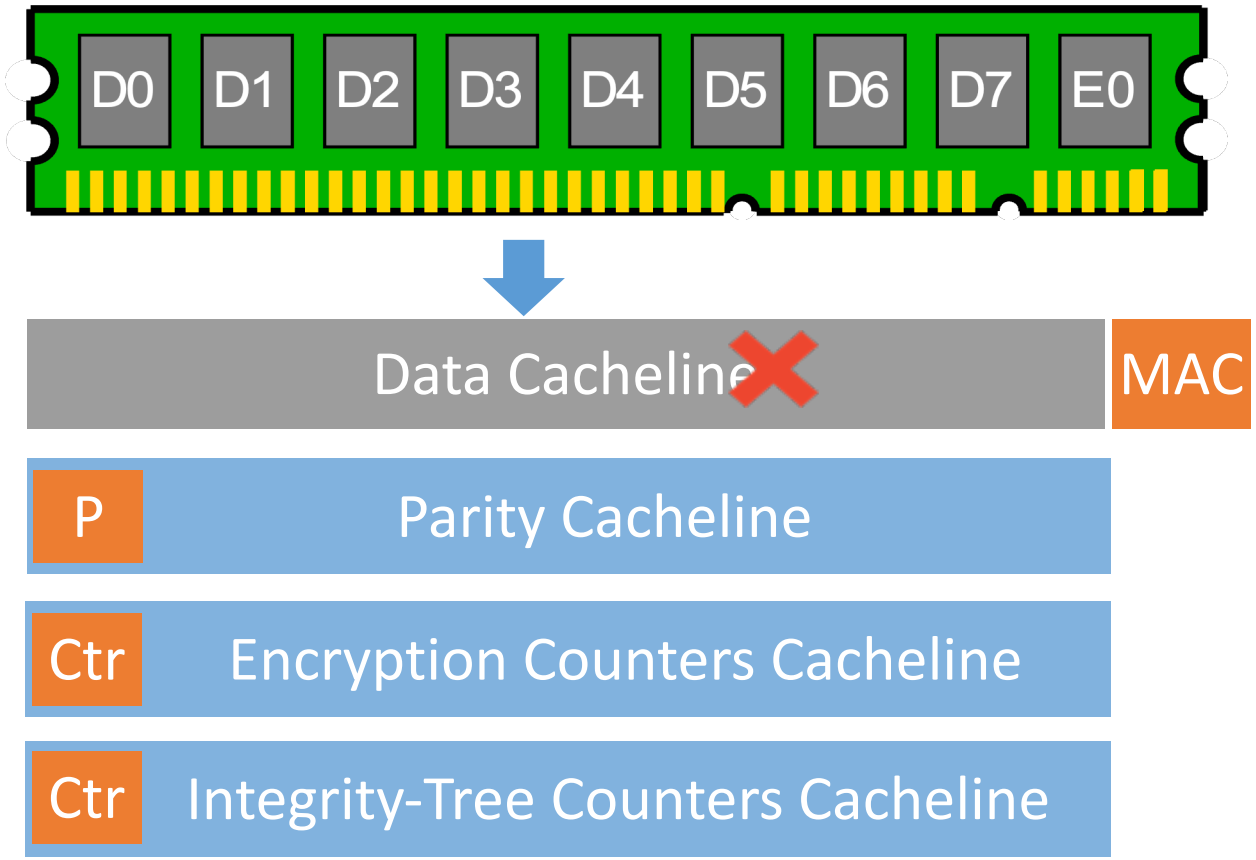
How to identify chip with failure?

Synergy can tolerate 1 chip with failure out of 9 chips, much stronger reliability than SECDED (Baseline)

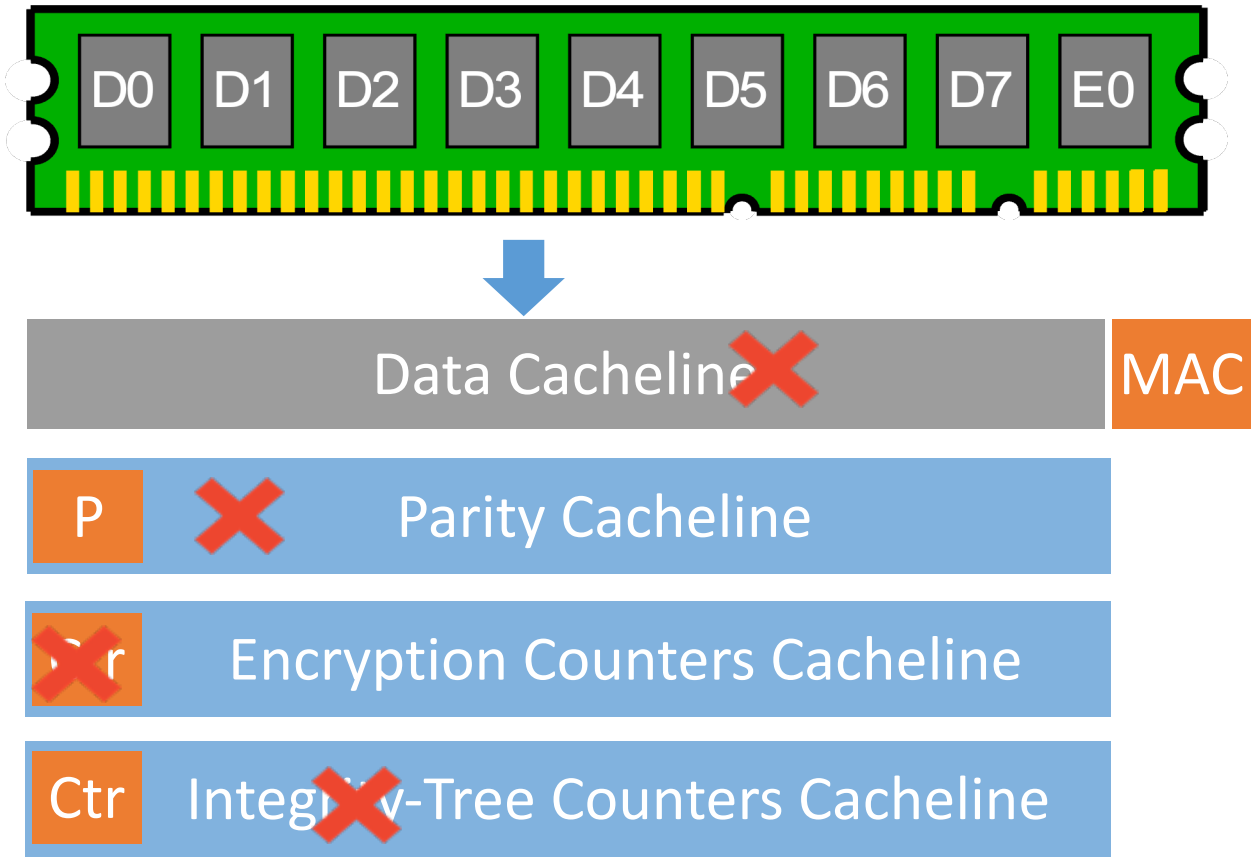
Tolerating Errors in Other Metadata



Tolerating Errors in Other Metadata

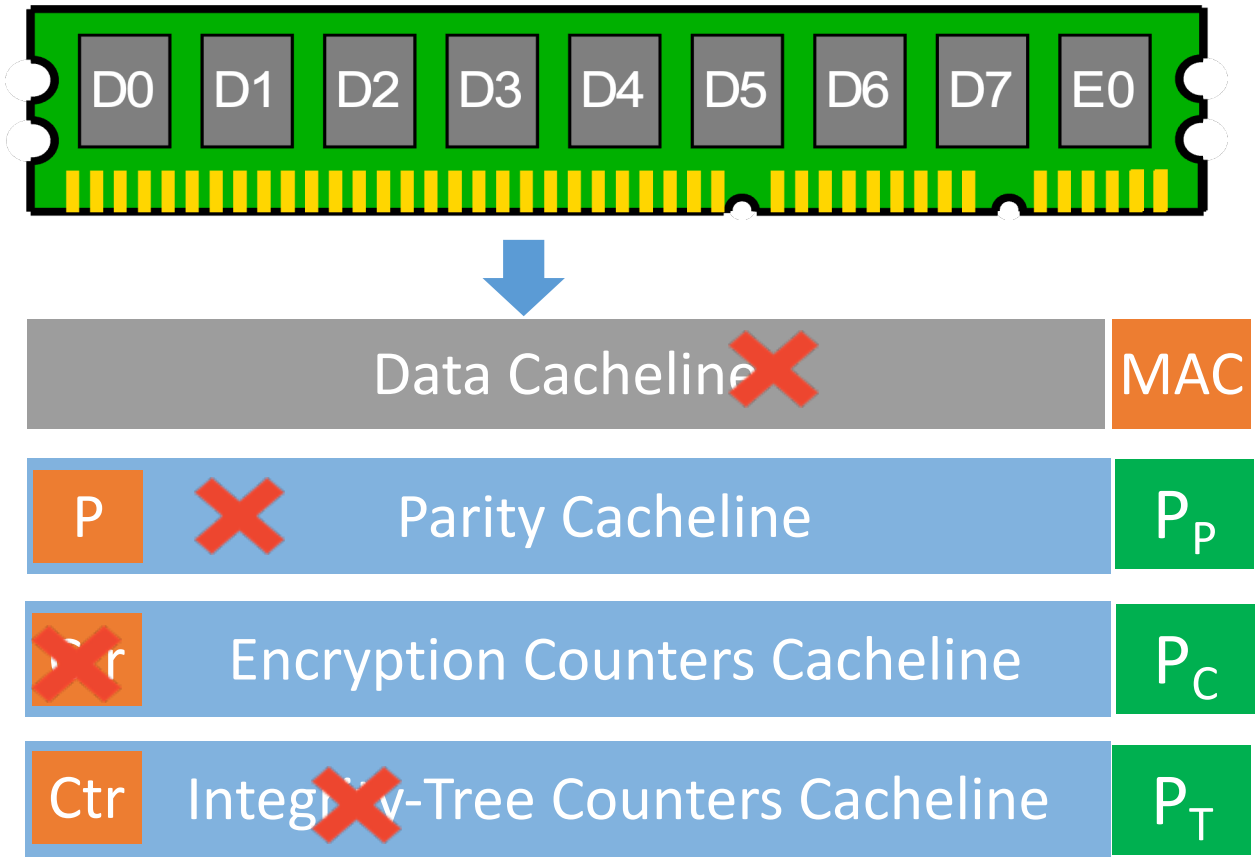


Tolerating Errors in Other Metadata



Errors can occur in any metadata stored in memory

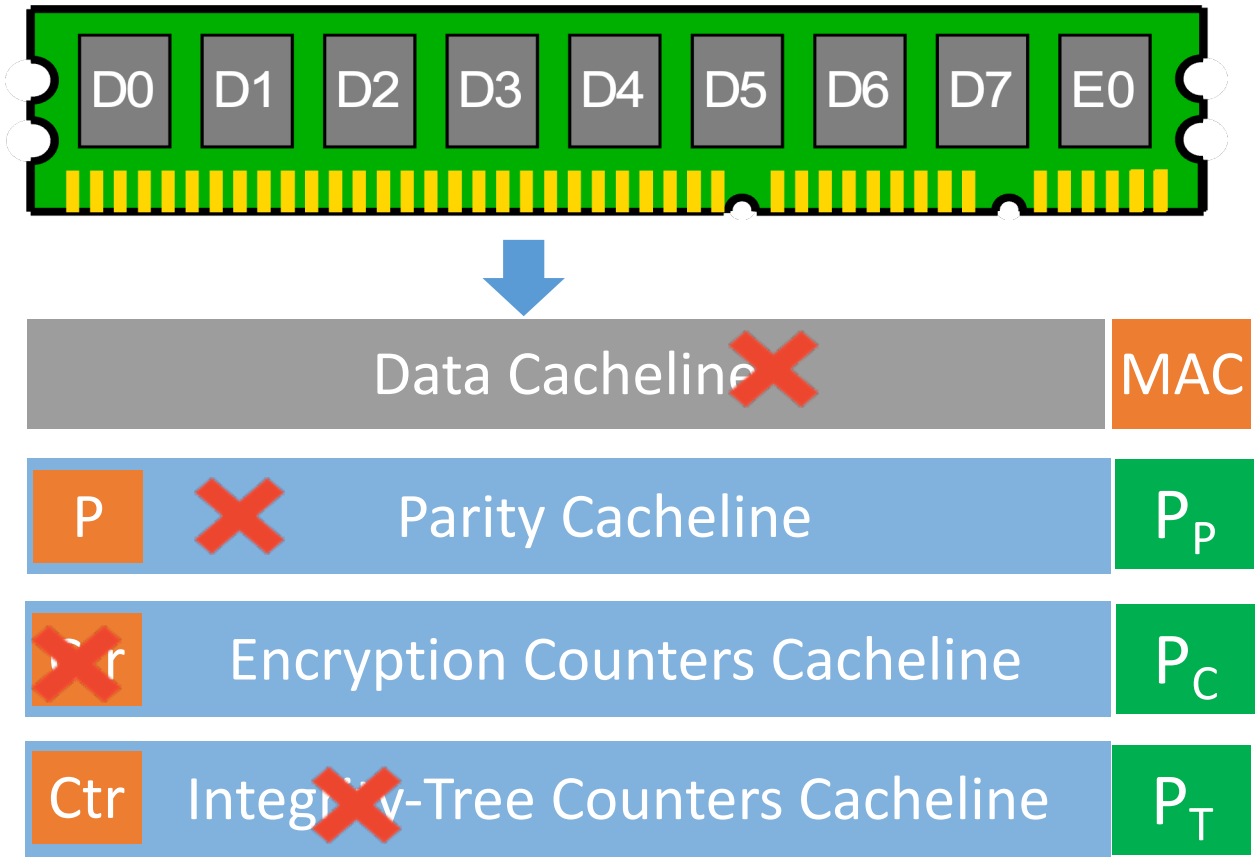
Tolerating Errors in Other Metadata



Errors can occur in any metadata stored in memory

SYNERGY stores Parity with each metadata to correct errors

Tolerating Errors in Other Metadata

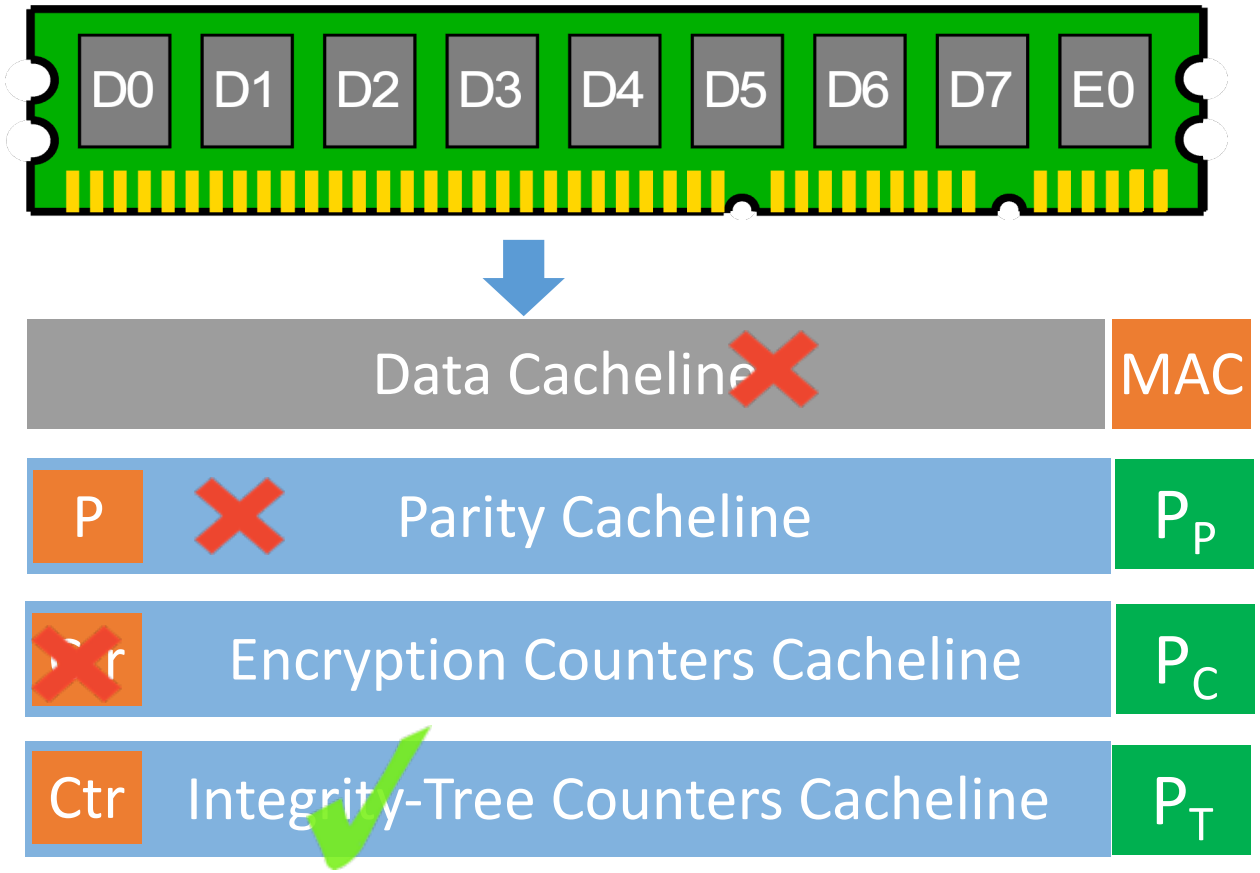


Errors can occur in any metadata stored in memory

SYNERGY stores Parity with each metadata to correct errors

Level-wise Error Correction Algorithm

Tolerating Errors in Other Metadata

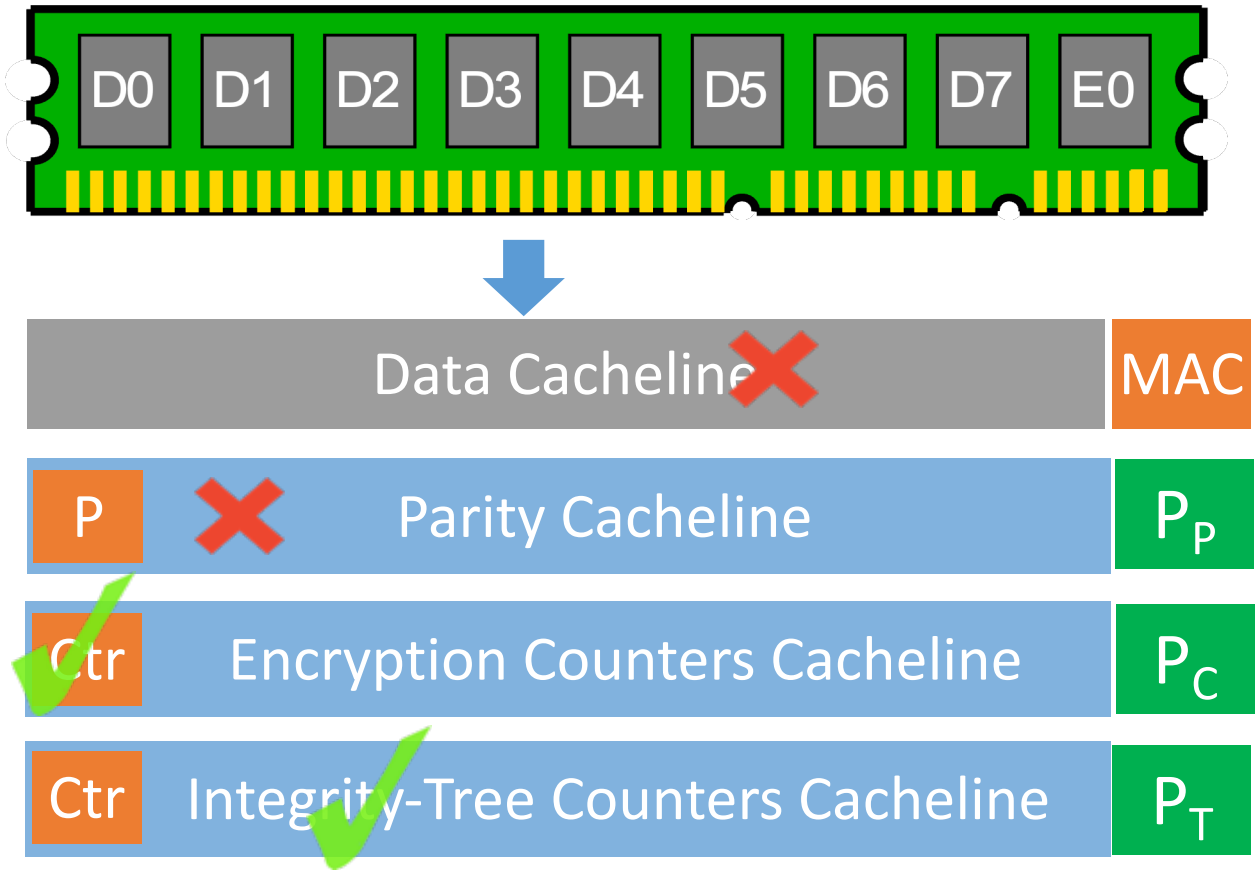


Errors can occur in any metadata stored in memory

SYNERGY stores Parity with each metadata to correct errors

Level-wise Error Correction Algorithm

Tolerating Errors in Other Metadata

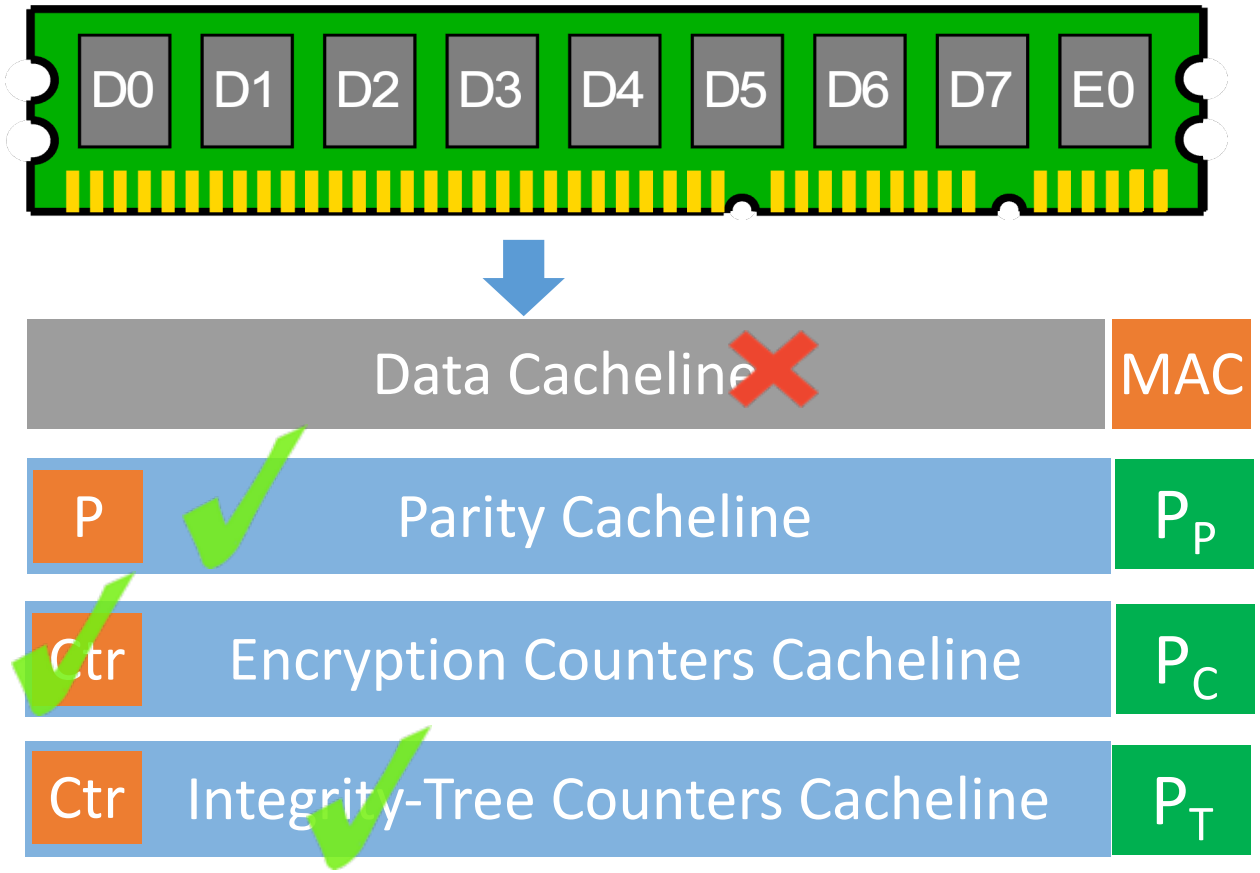


Errors can occur in any metadata stored in memory

SYNERGY stores Parity with each metadata to correct errors

Level-wise Error Correction Algorithm

Tolerating Errors in Other Metadata

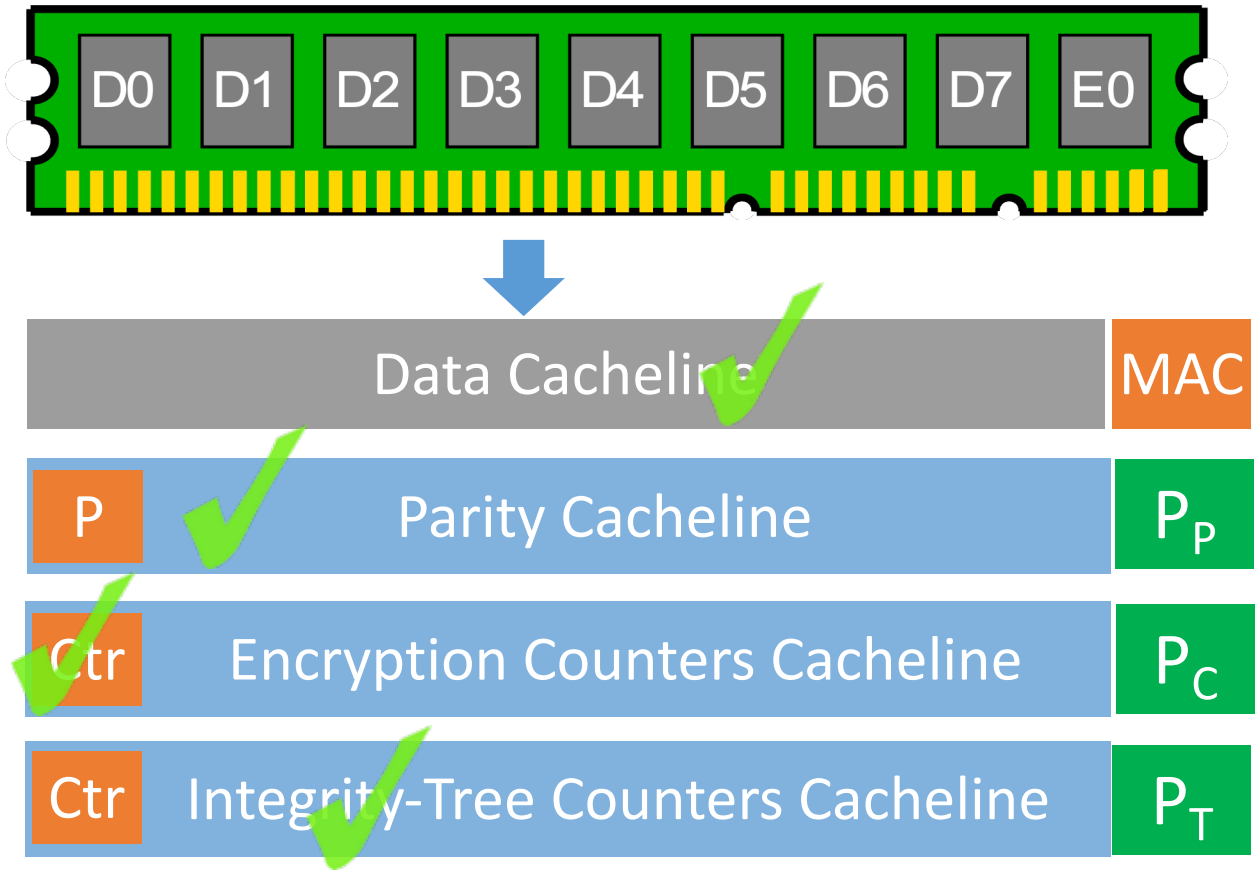


Errors can occur in any metadata stored in memory

SYNERGY stores Parity with each metadata to correct errors

Level-wise Error Correction Algorithm

Tolerating Errors in Other Metadata



Errors can occur in any metadata stored in memory

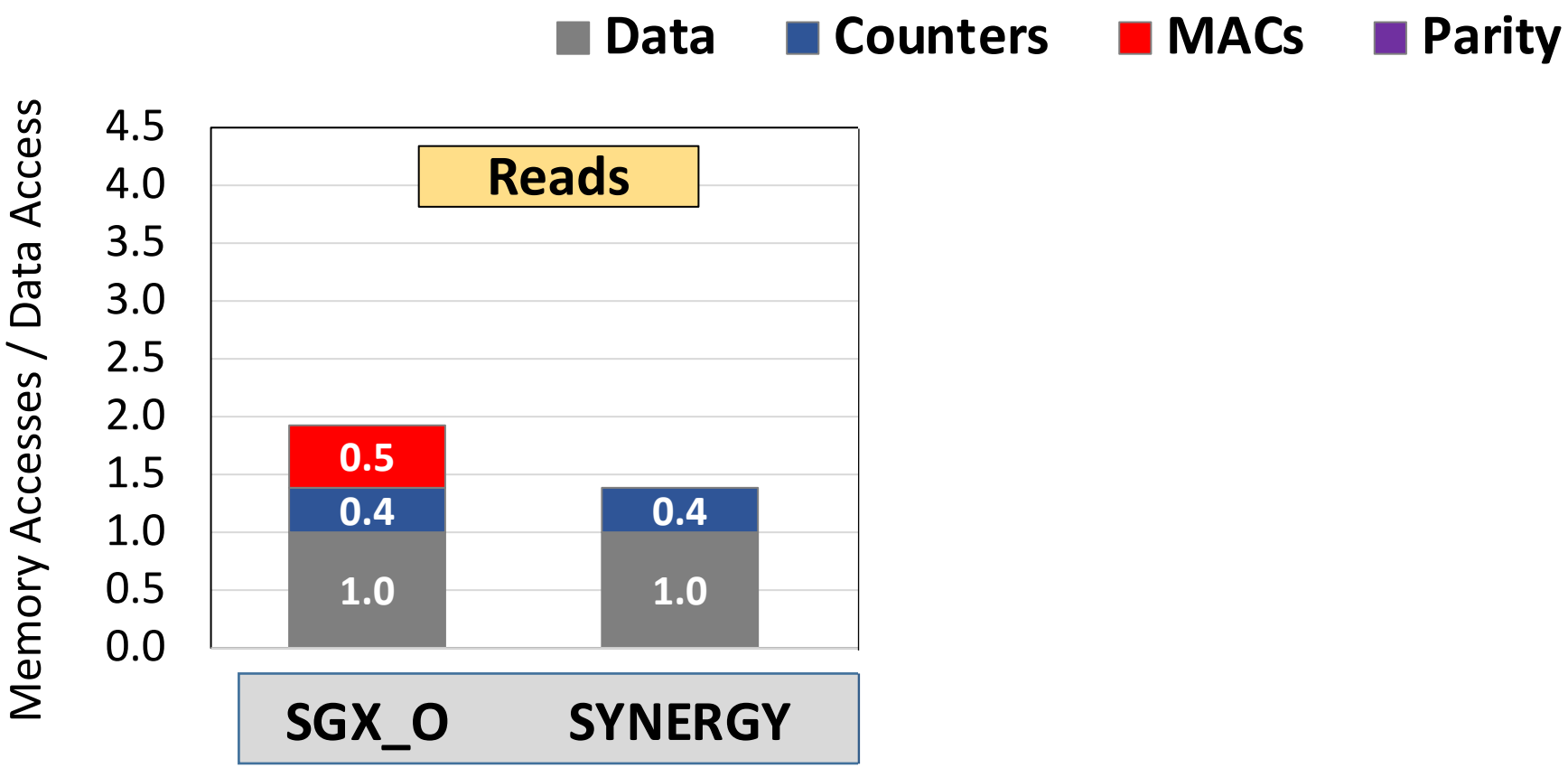
SYNERGY stores Parity with each metadata to correct errors

Level-wise Error Correction Algorithm

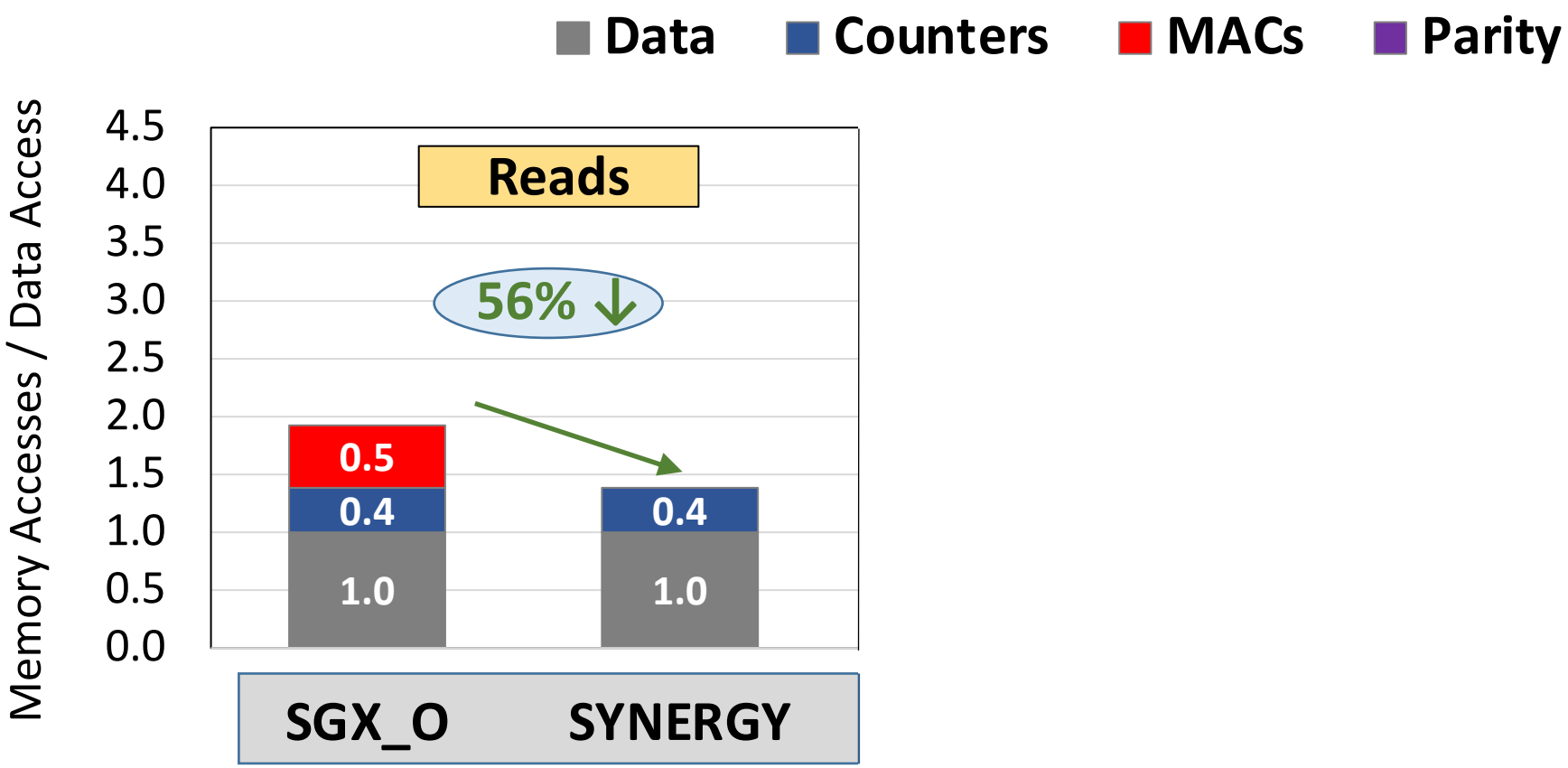
Agenda

- Introduction & Background
- Synergy Design
- **Evaluation**

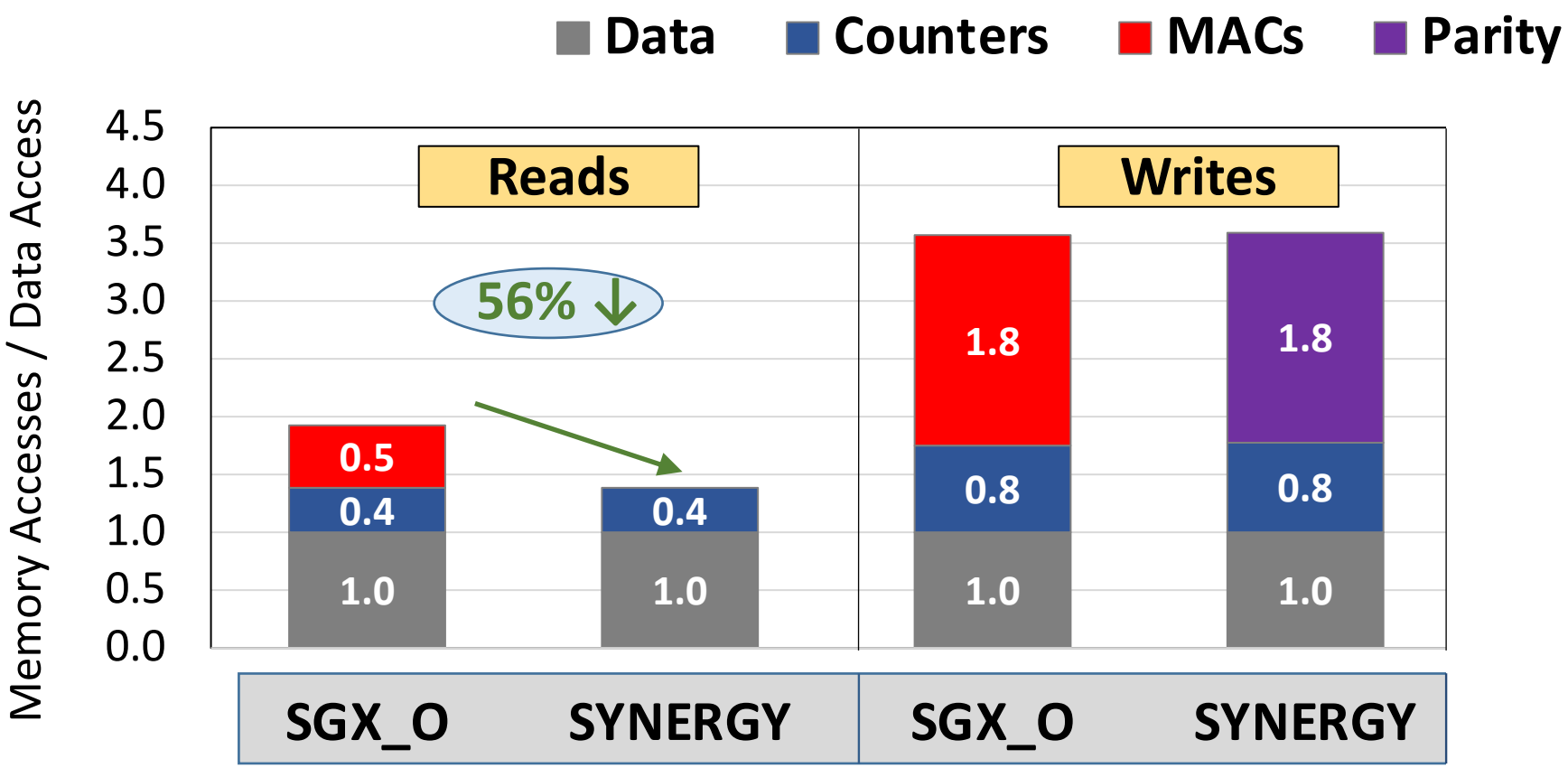
Benefit-1: Reduced Memory Traffic



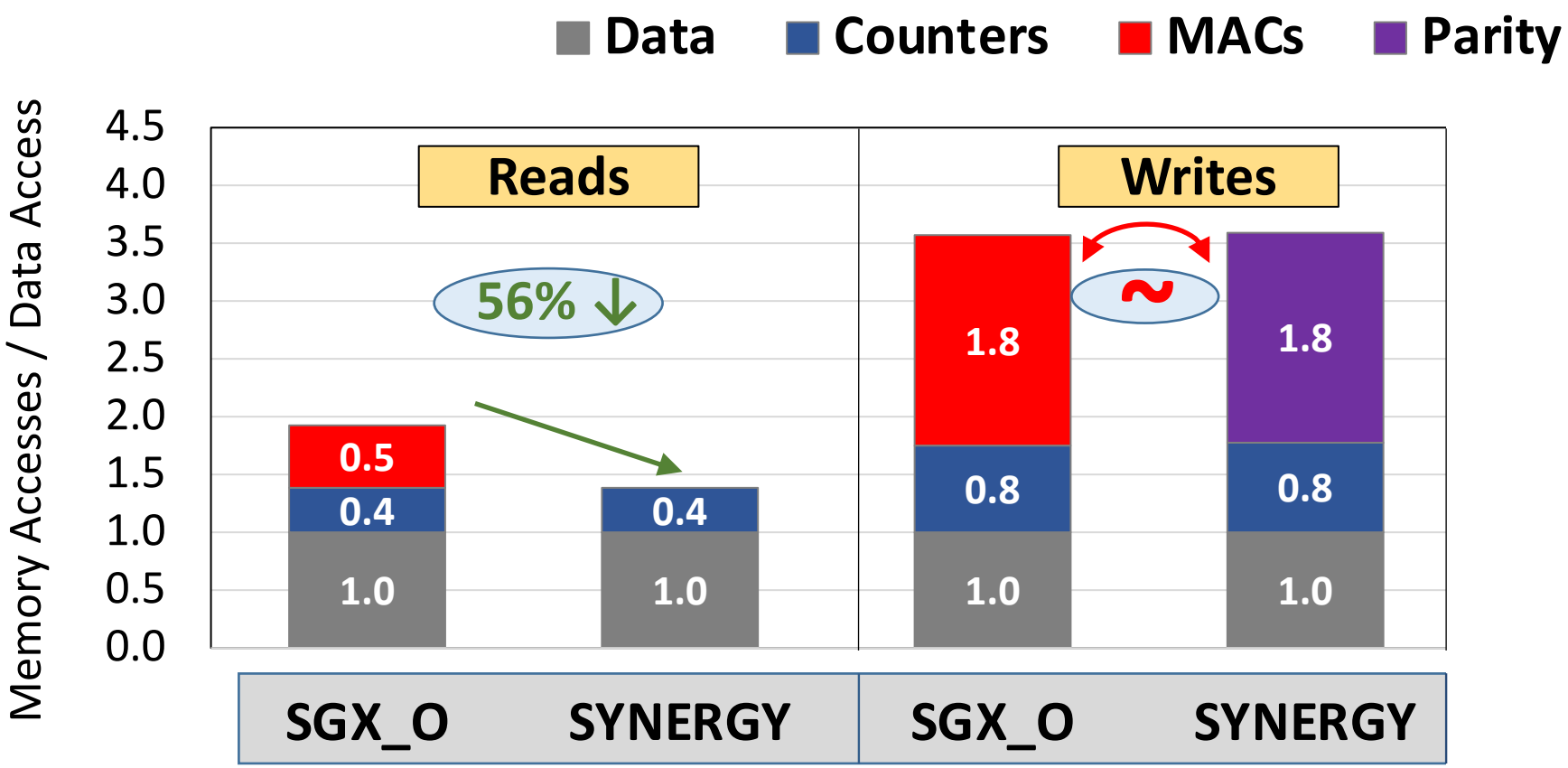
Benefit-1: Reduced Memory Traffic



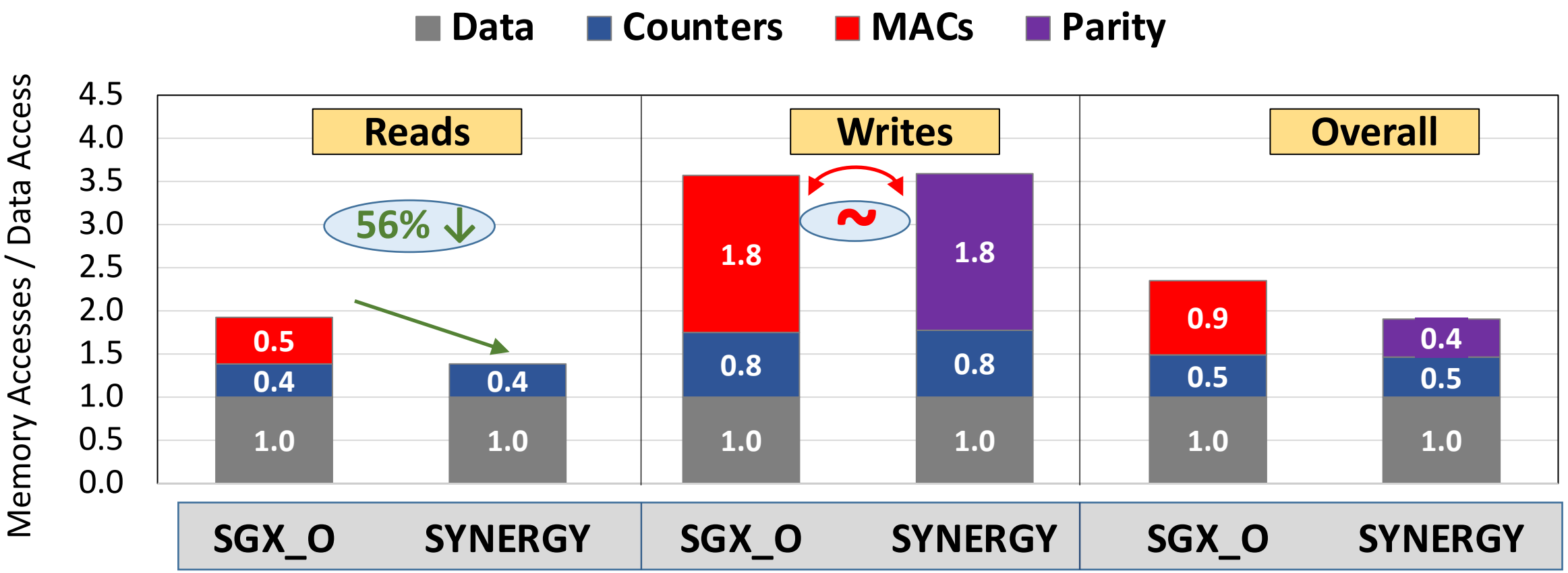
Benefit-1: Reduced Memory Traffic



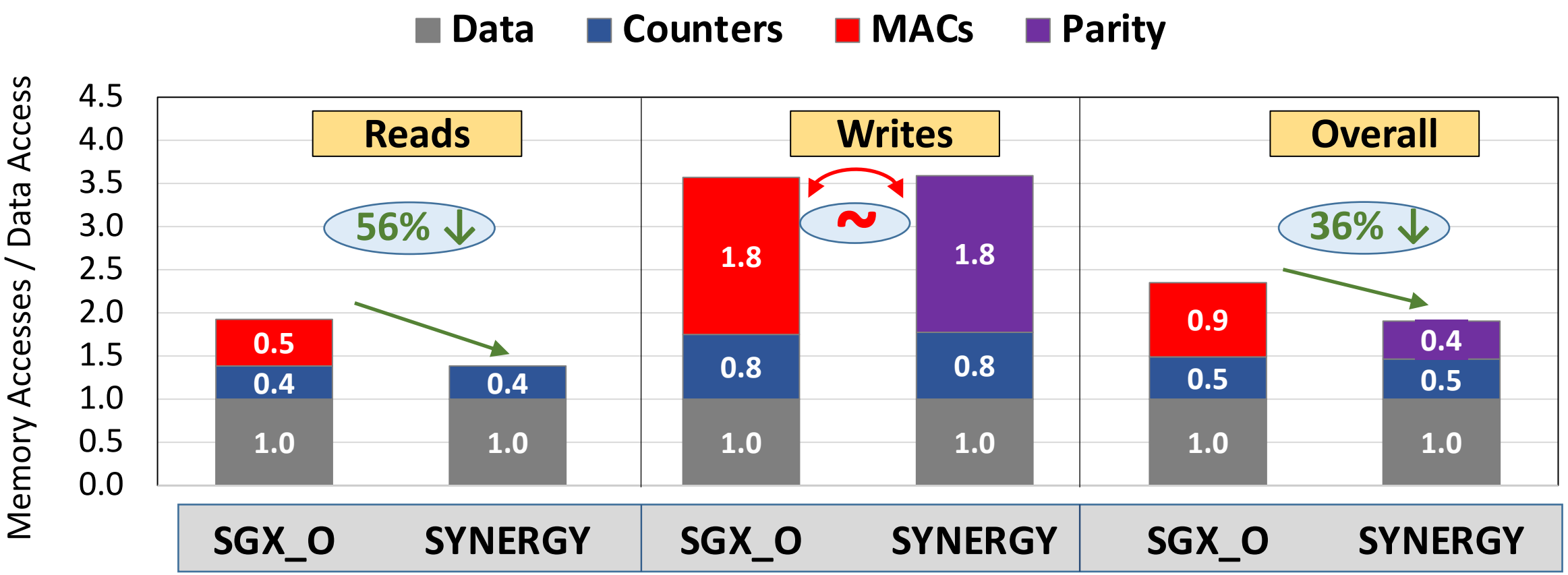
Benefit-1: Reduced Memory Traffic



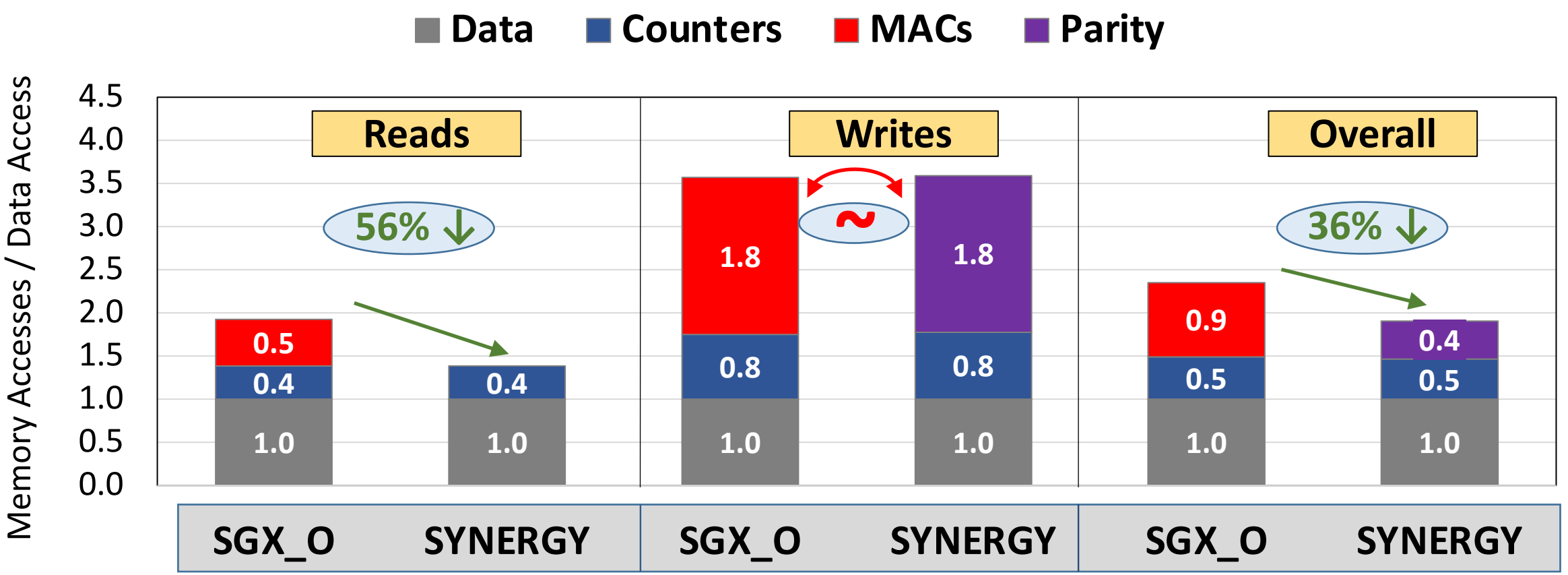
Benefit-1: Reduced Memory Traffic



Benefit-1: Reduced Memory Traffic



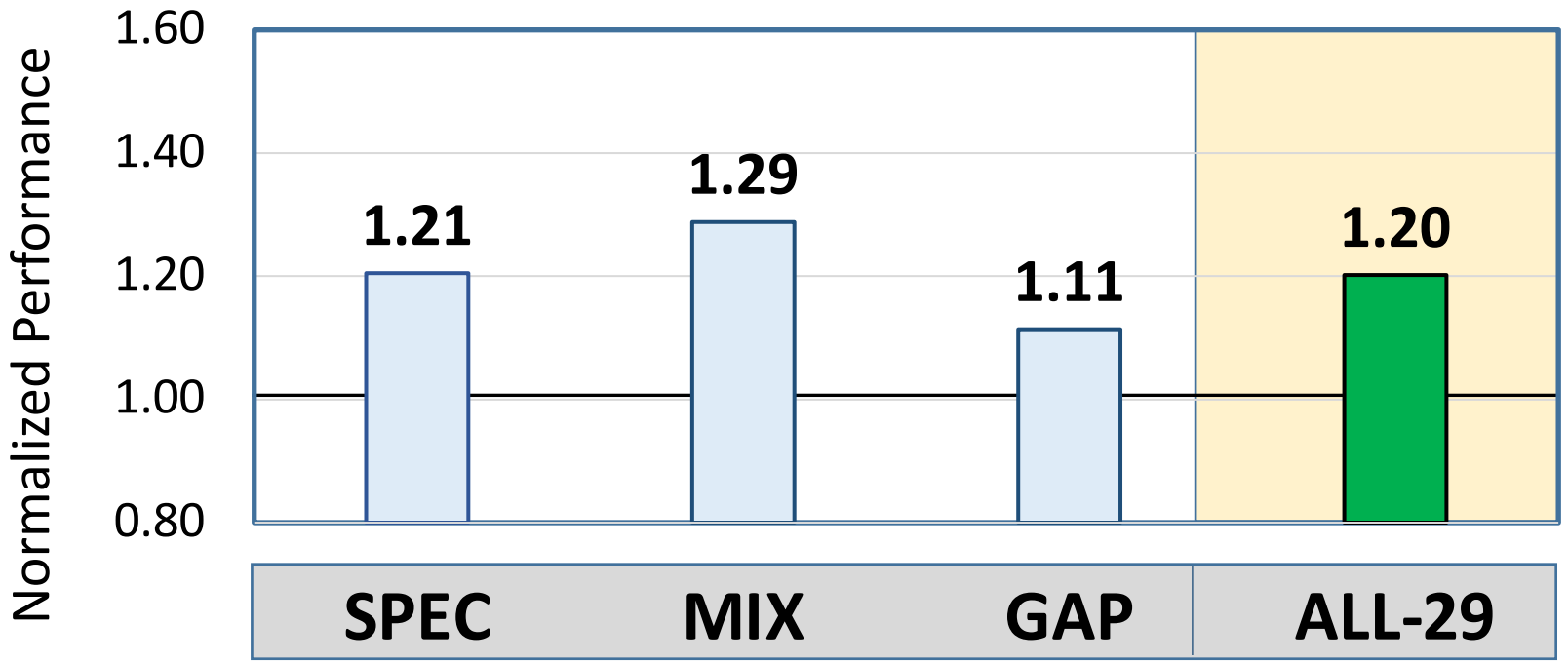
Benefit-1: Reduced Memory Traffic



SYNERGY Reduces Metadata Accesses by 36%

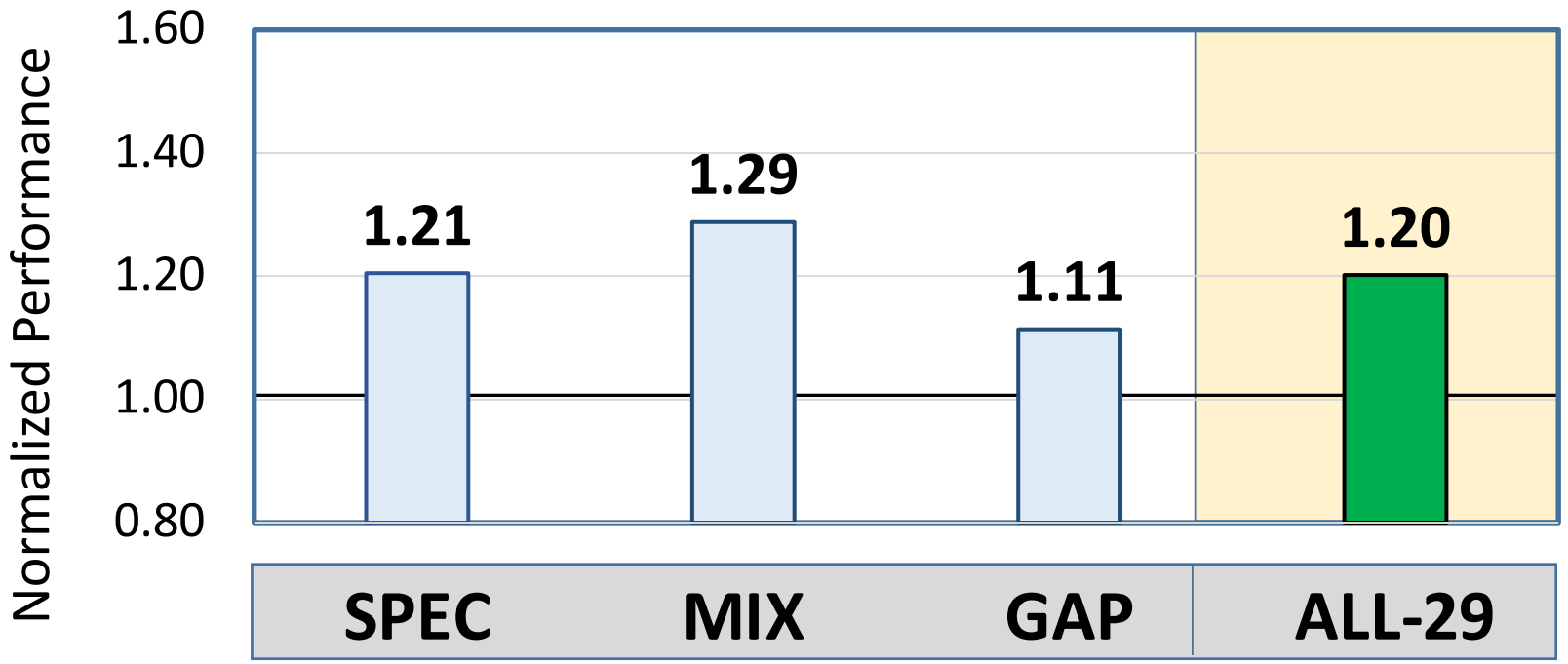
Benefit-2: Better Performance

Performance of SYNERGY normalized to SGX_O



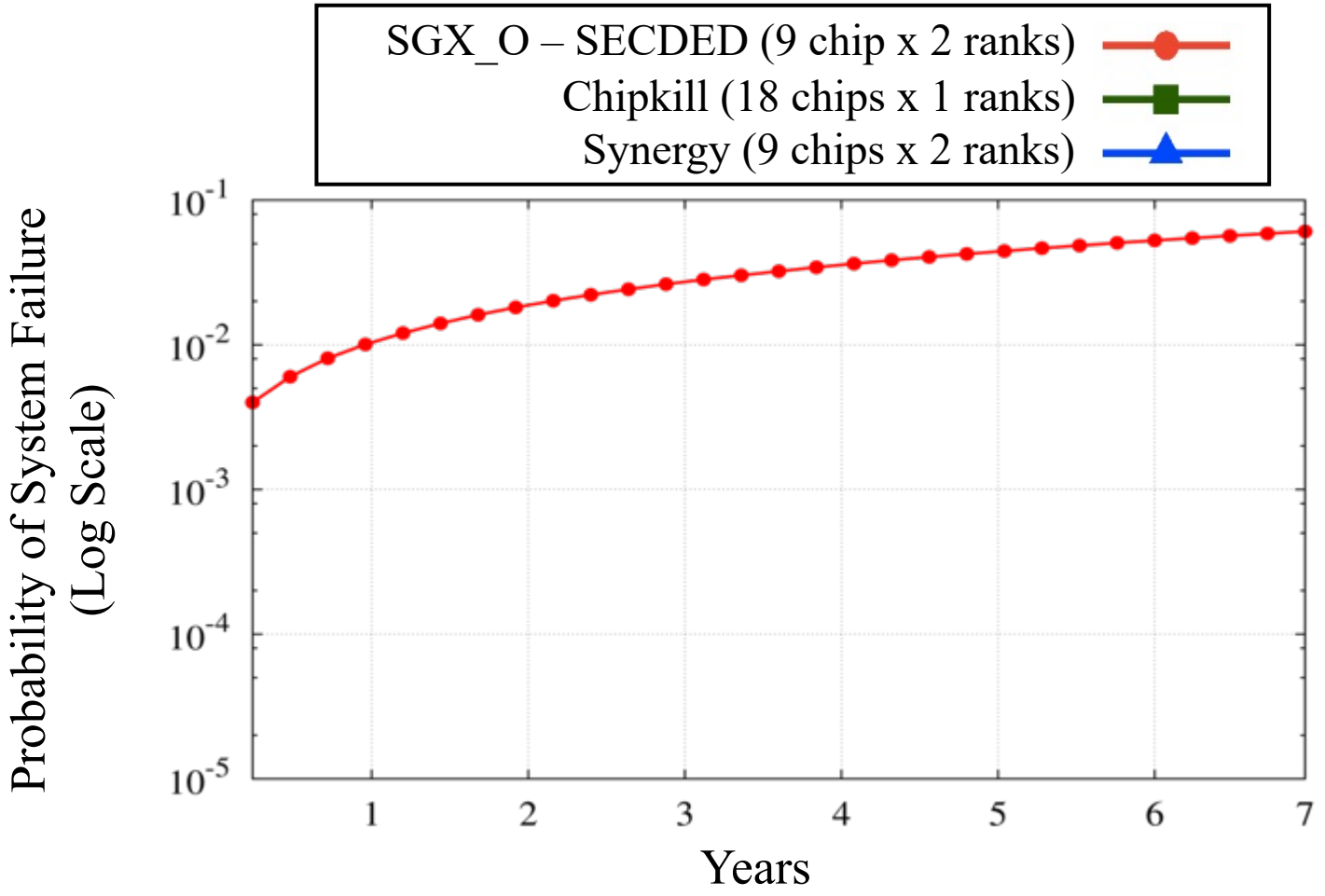
Benefit-2: Better Performance

Performance of SYNERGY normalized to SGX_O



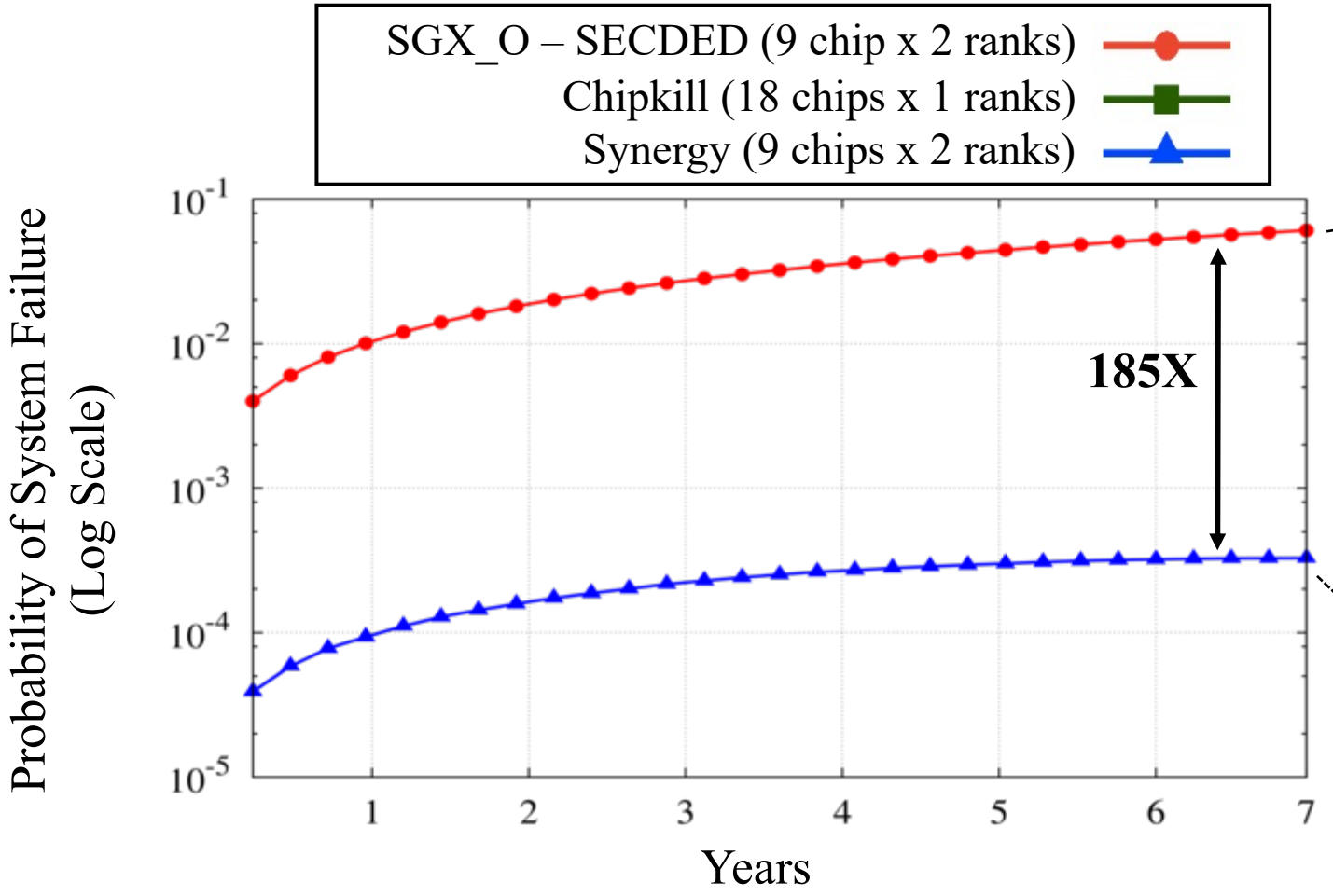
SYNERGY improves performance of secure memory by 20%, without any additional storage

Benefit-3: Stronger Reliability



SECDED only tolerates single-bit errors

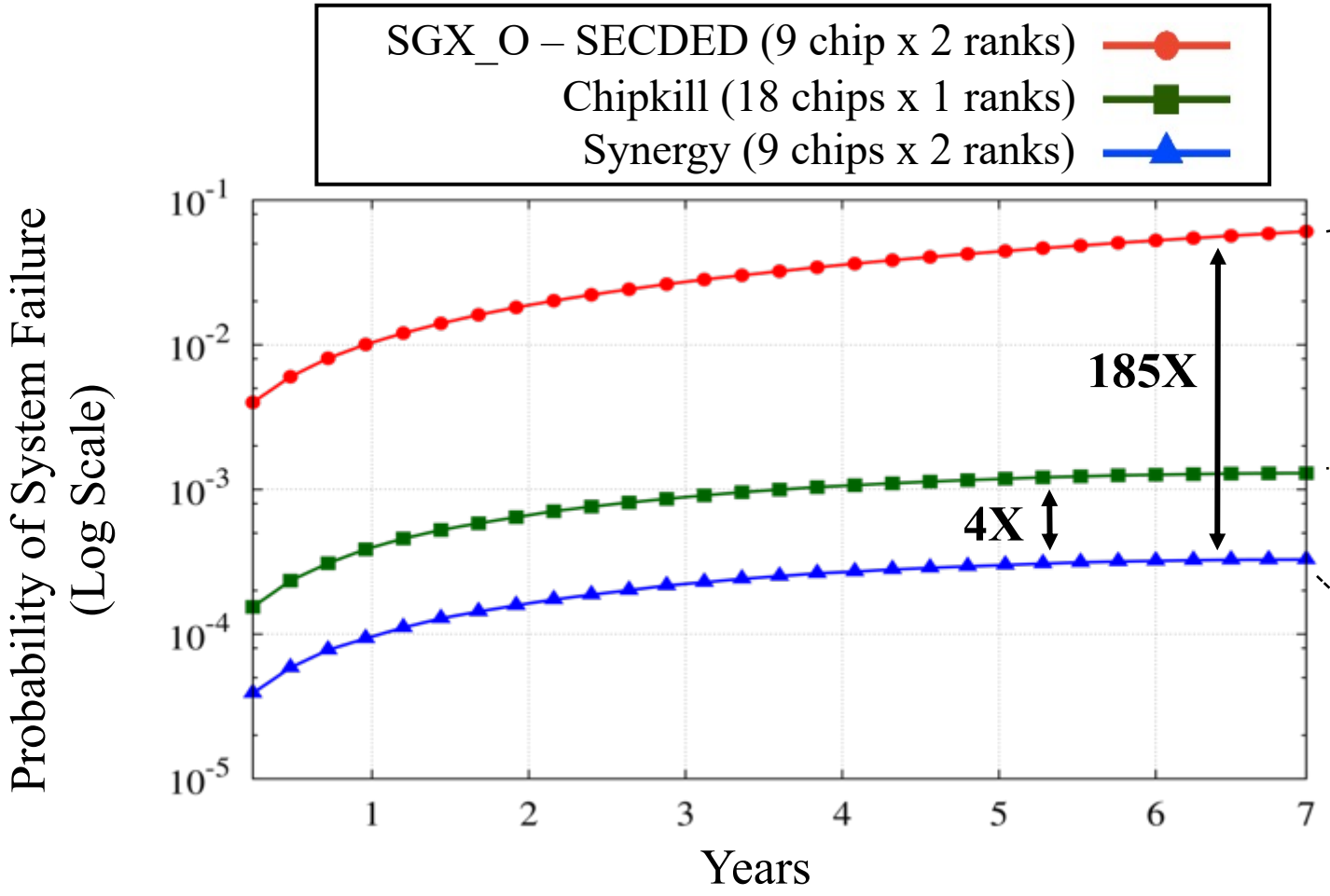
Benefit-3: Stronger Reliability



SECDED only tolerates single-bit errors

SYNERGY tolerates 1 chip failure out of 9 chips

Benefit-3: Stronger Reliability

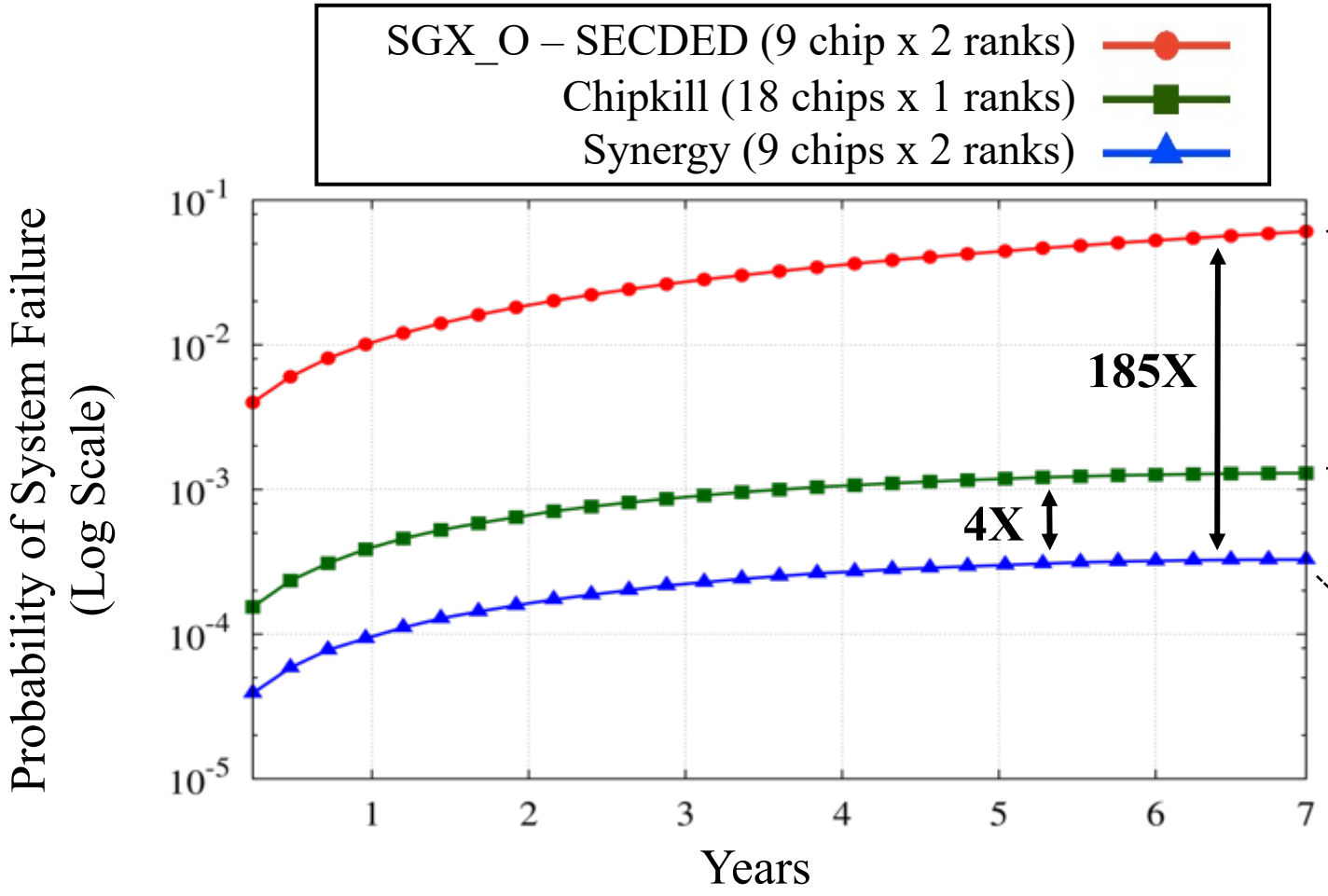


SECDED only tolerates single-bit errors

Chipkill tolerates 1 chip failure out of 18 chips

SYNERGY tolerates 1 chip failure out of 9 chips

Benefit-3: Stronger Reliability



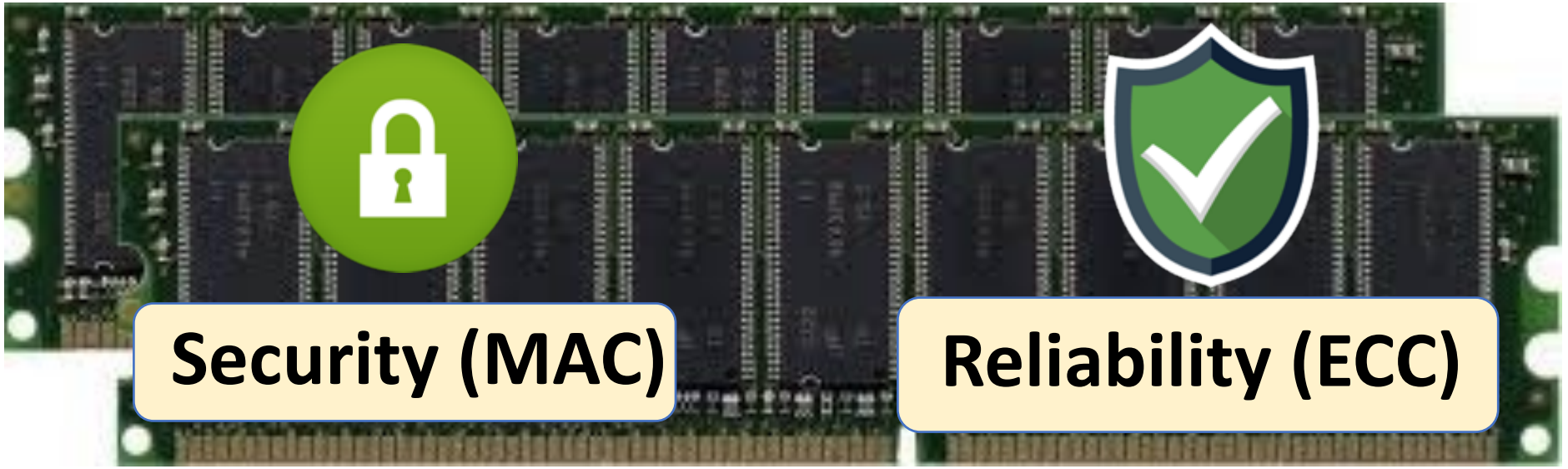
SECDED only tolerates single-bit errors

Chipkill tolerates 1 chip failure out of 18 chips

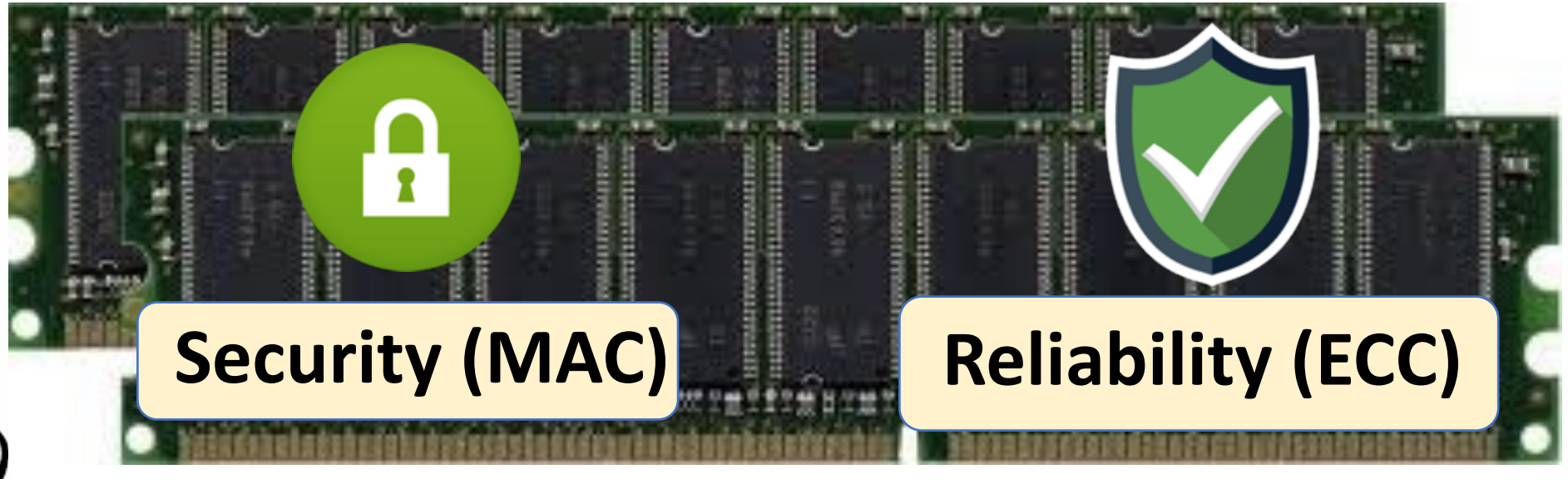
SYNERGY tolerates 1 chip failure out of 9 chips

SYNERGY has 185x higher reliability than Baseline ECC-DIMM

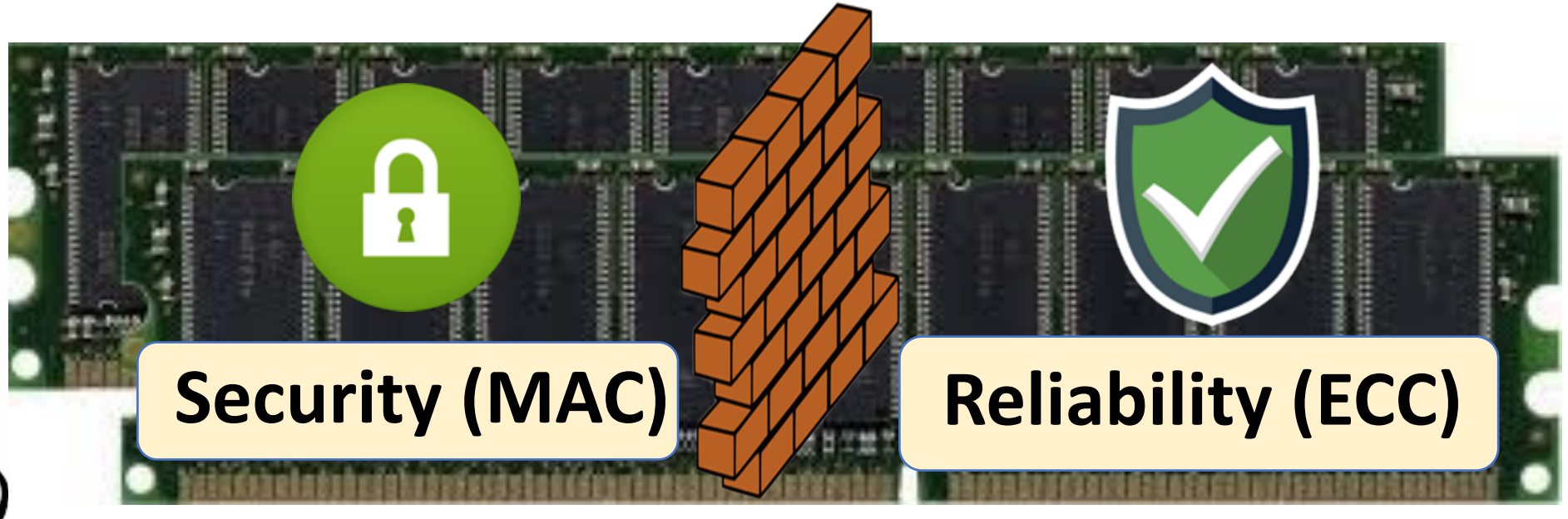
Conclusion



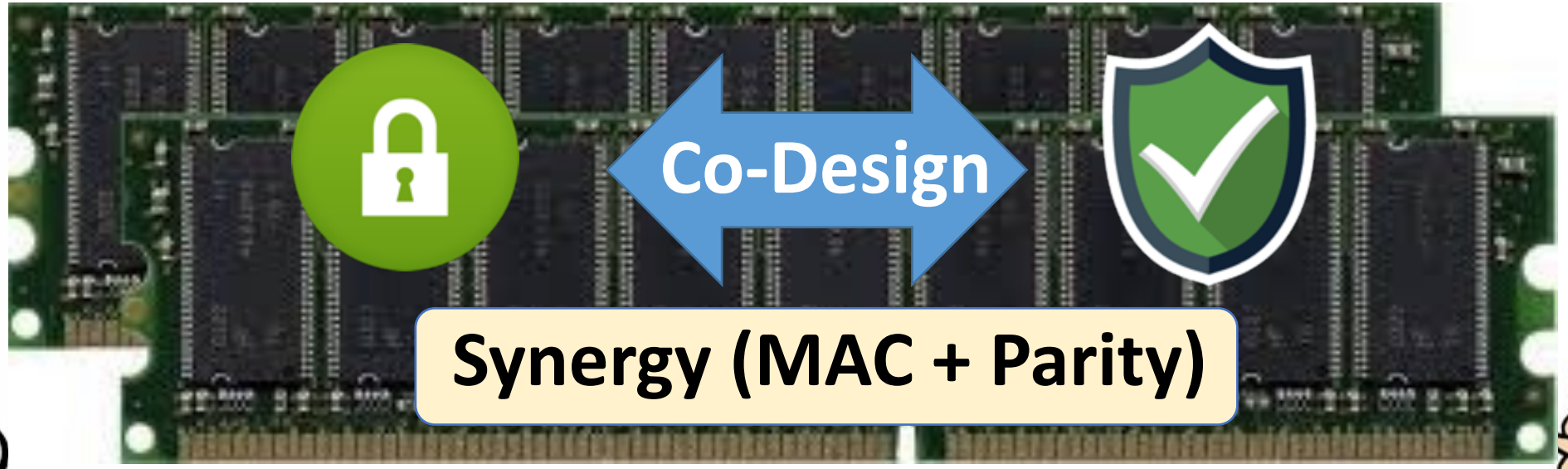
Conclusion

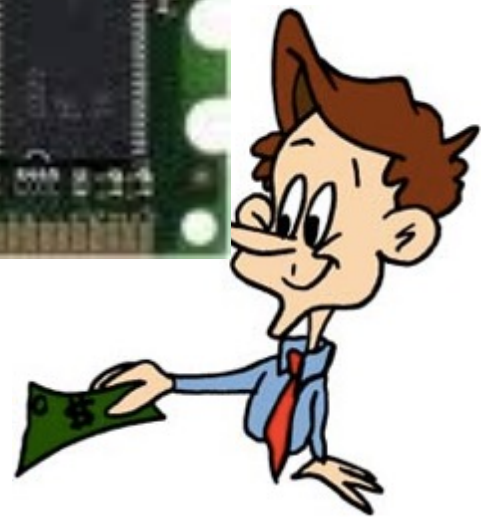


Conclusion

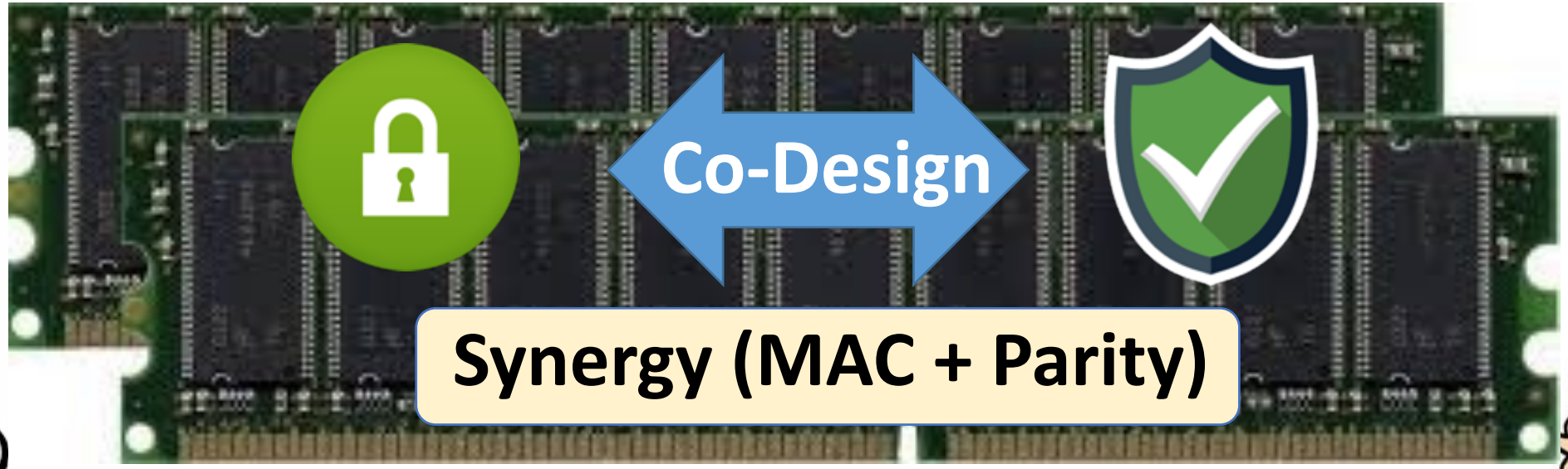


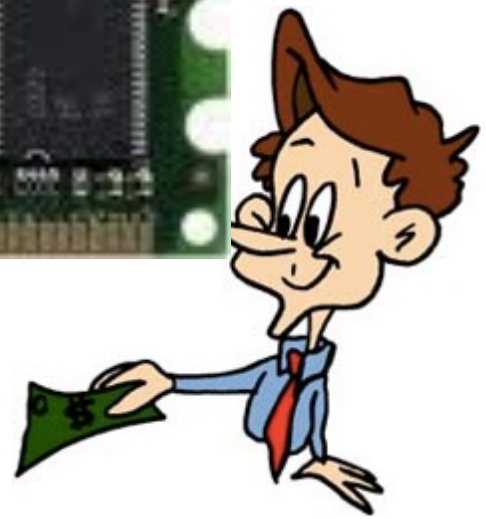
Conclusion





Conclusion





SYNERGY improves Performance by 20% and Reliability by 185x

Thanks and Questions

“The whole is greater than the sum of its parts” - *Aristotle*