How to Kill the Second Bird with One ECC: The Pursuit of Row Hammer Resilient DRAM

Michael Jaemin Kim[†], Minbok Wi[†], Jaehyun Park[†], Seoyoung Ko[†],
Jaeyoung Choi[†], Hwayong Nam[†], Nam Sung Kim[‡], Jung Ho Ahn[†], Eojin Lee[§]

Seoul National University[†], University of Illinois Urbana Champaign[‡], Inha University[§]

Presenter: Michael Jaemin Kim (michael604@scale.snu.ac.kr)

Overview

- Problem 1: **Scalability** problem of existing Row Hammer solutions.
- Problem 2: Limited success of **DRAM ECC** against Row Hammer.

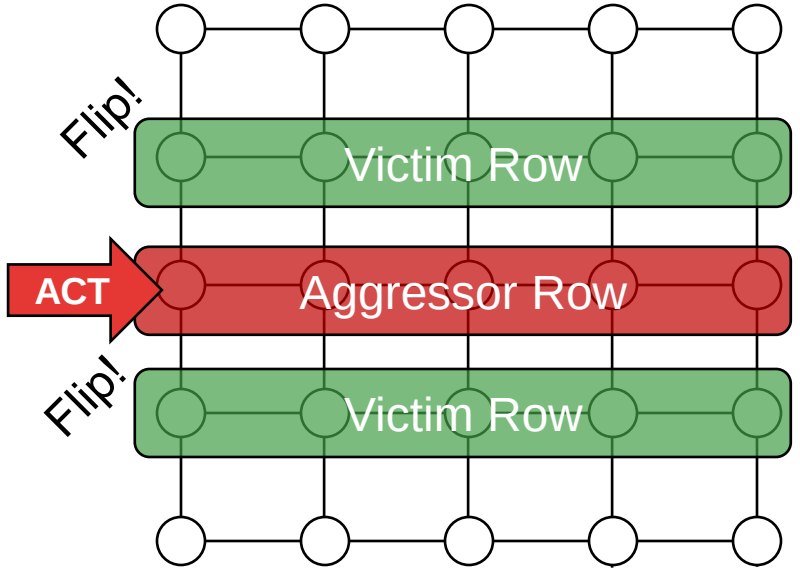
Cube

- Goal 1: Exploit the **ECC against Row Hammer**.
- Goal 2: **Cooperate** with the probabilistic Row Hammer solutions.

- Technique 1: Row address **scramble** for Chipkill against Row Hammer.
- Technique 2: Victim **diagnosis** using On-die ECC **error profile** against Row Hammer.

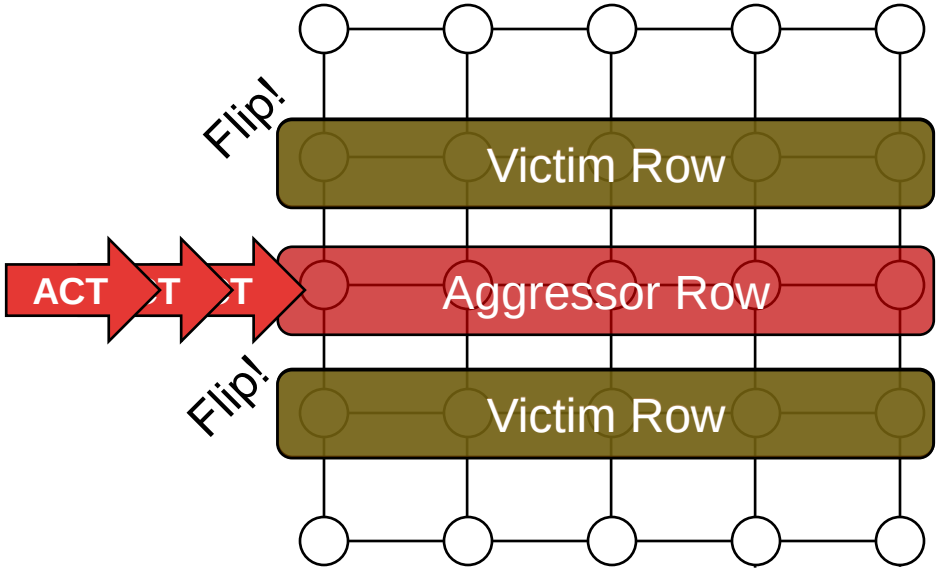
Row Hammer

- Row Hammer?
  - Row Hammer Threshold (T_{RH})

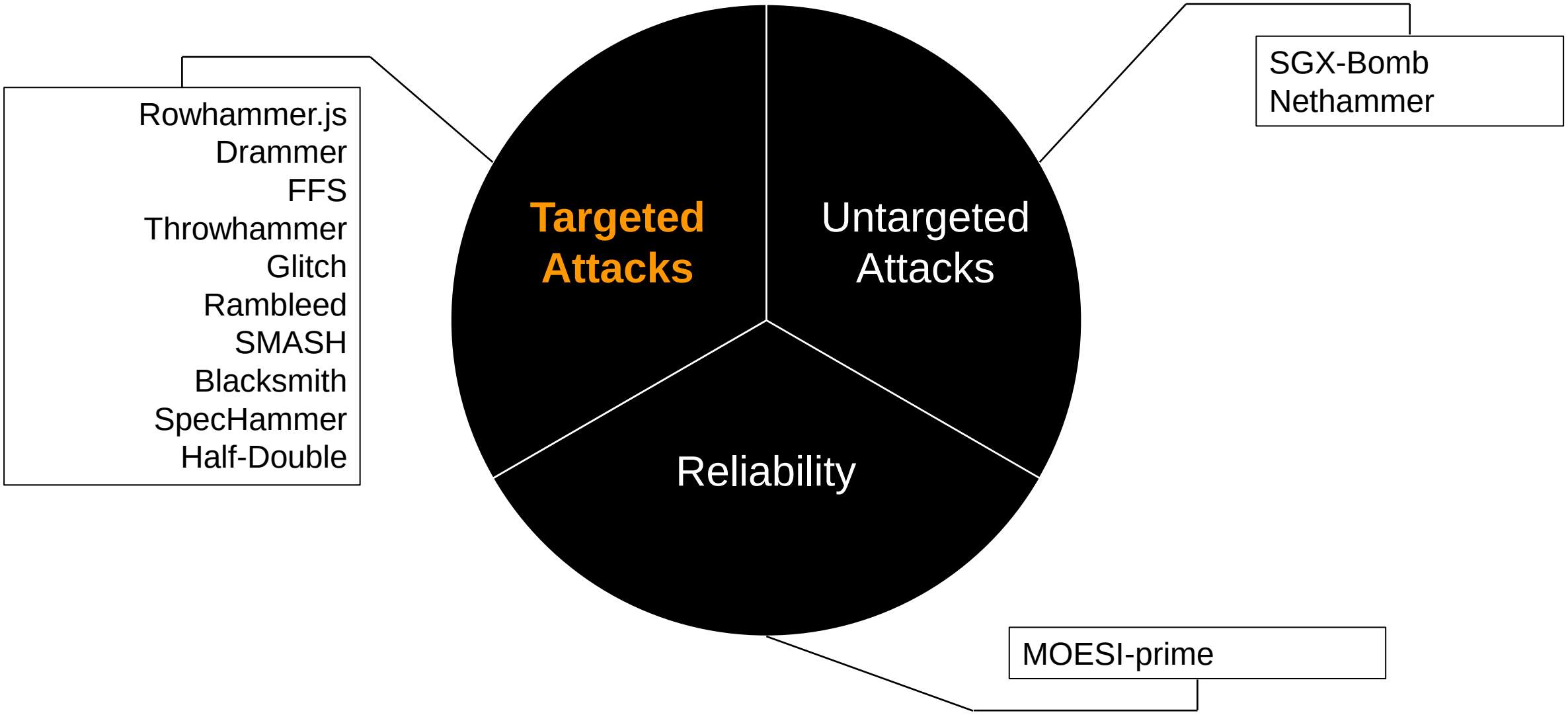


Row Hammer

- Row Hammer?
  - Row Hammer Threshold (T_{RH})

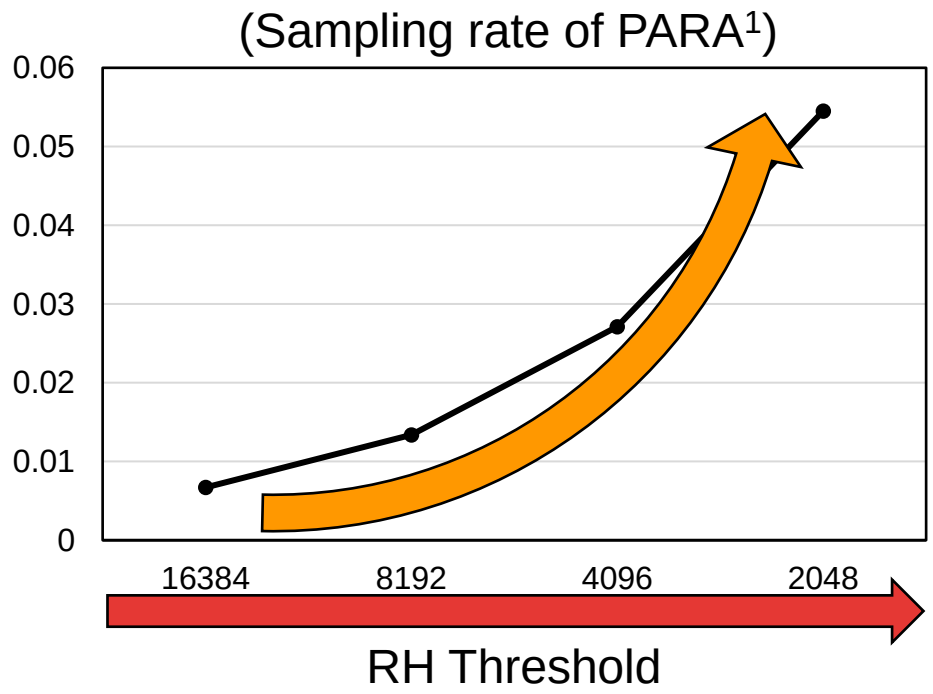


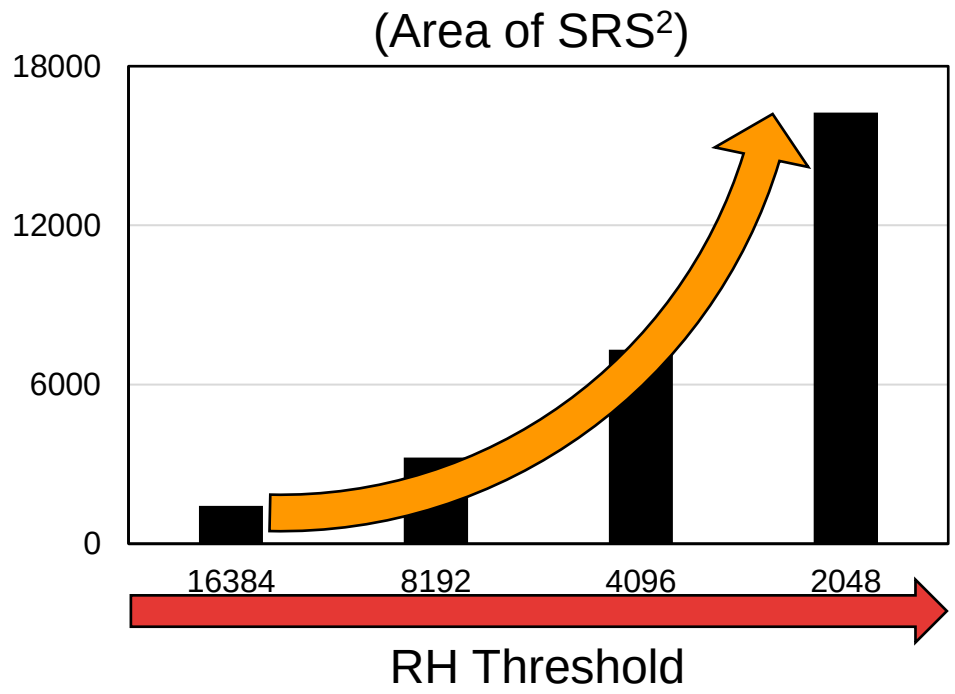
Row Hammer Attacks



Row Hammer Mitigation

- **Scalability** problem following lowering Row Hammer Threshold
  - Performance
  - Area

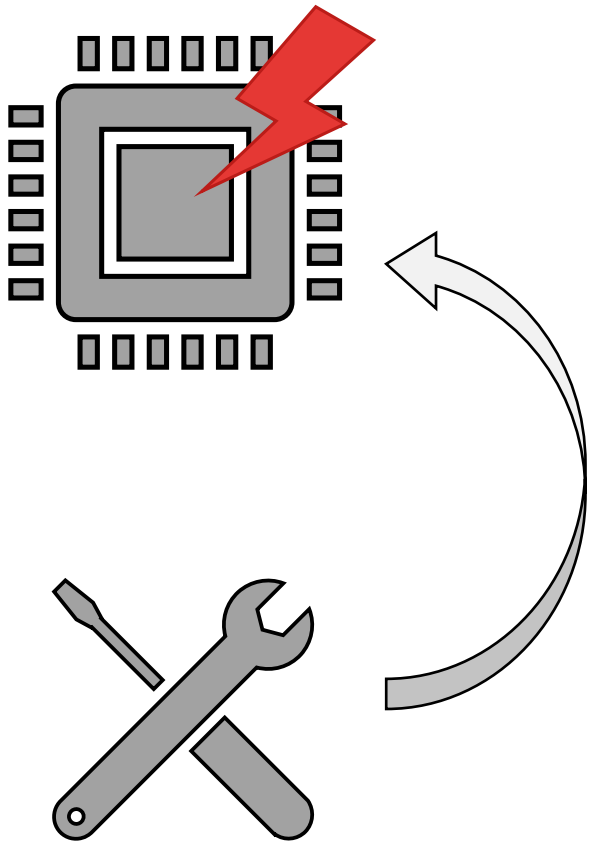




[1] Yoongu Kim, Ross Daly, Jeremie Kim, Chris Fallin, Ji Hye Lee, Donghyuk Lee, Chris Wilkerson, Konrad Lai, and Onur Mutlu. 2014. Flipping Bits in Memory Without Accessing Them: An Experimental Study of DRAM Disturbance Errors.
[2] Gururaj Saileshwar, Bolin Wang, Moinuddin Qureshi, and Prashant J. Nair. 2022. Randomized Row-Swap: Mitigating Row Hammer by Breaking Spatial Correlation between Aggressor and Victim Rows.

DRAM ECC

- DRAM errors
  - Single cell faults
    - Process scaling faults, cosmic rays, etc.
  - Large granularity row/column hard faults

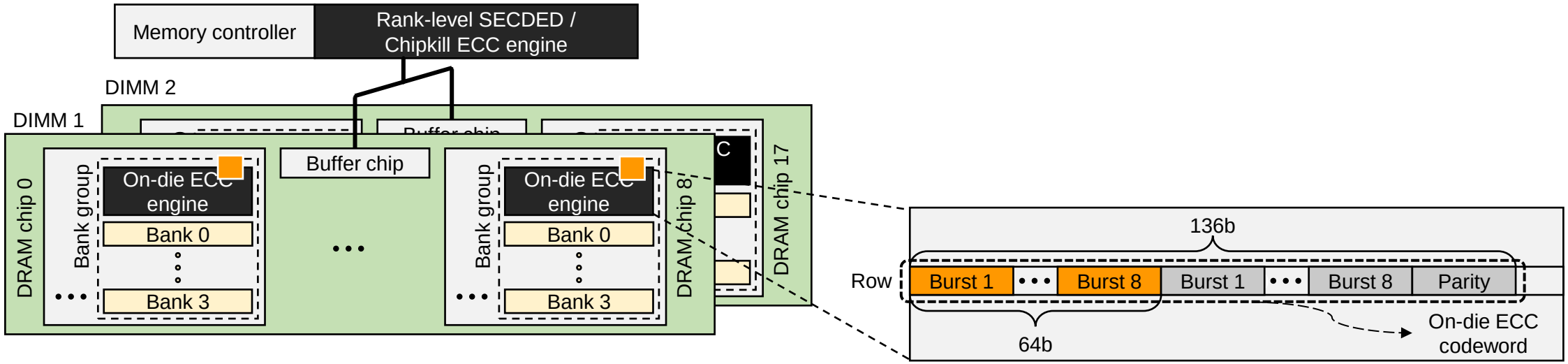


- Error correcting codes (ECC)
  - Redundancy / parity bits
    - Encoding
  - Correct or detect errors
    - Decoding

DRAM ECC

- Example
  - Two DIMMs per memory channel,
  - One rank per DIMM
  - ×8 chips with a burst length of eight

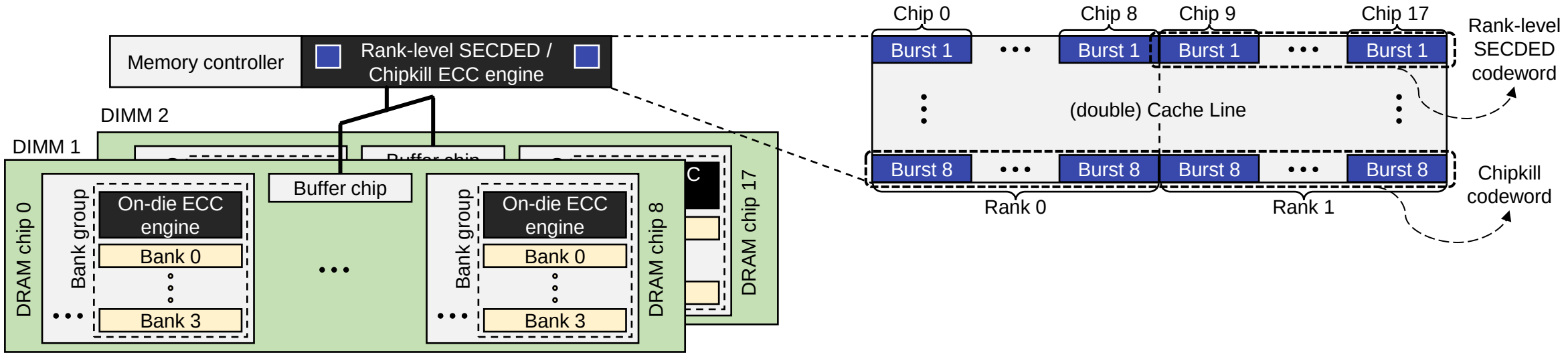
- Example
  - On-die ECC (OECC): Single Error Correction
  - Memory-Controller ECC: Chipkill ECC



DRAM ECC

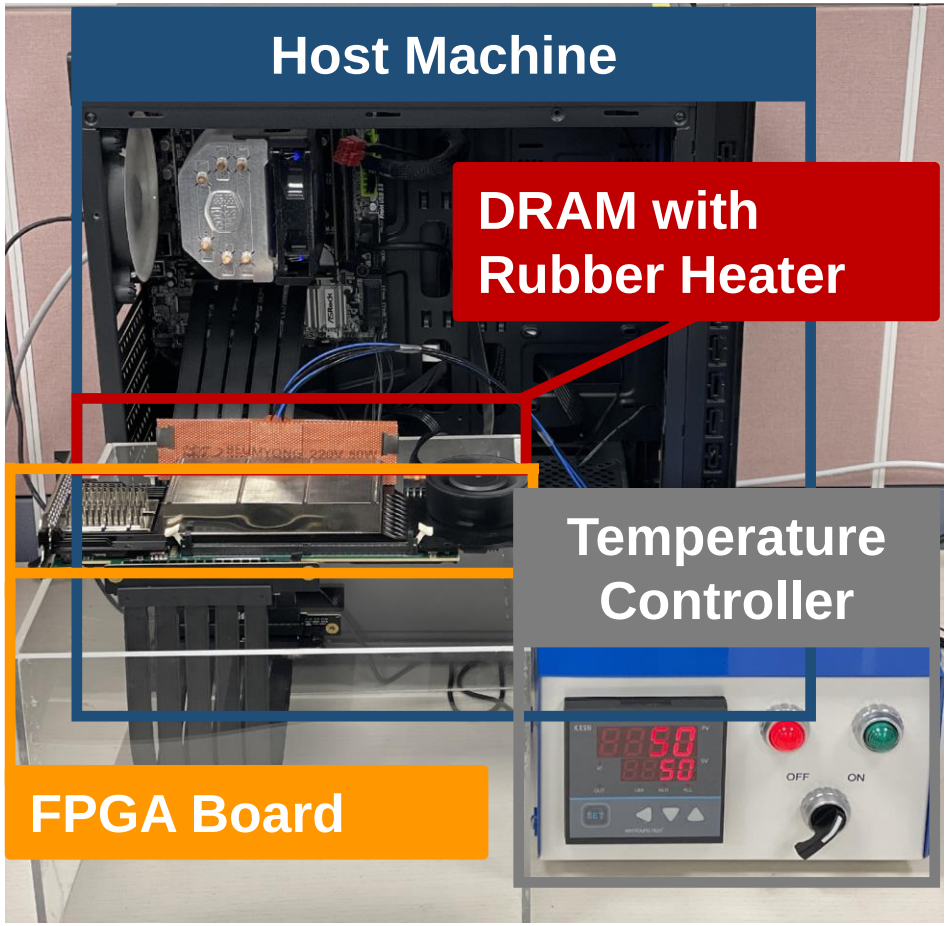
- Example
  - Two DIMMs per memory channel,
  - One rank per DIMM
  - ×8 chips with a burst length of eight

- Example
  - On-die ECC (OECC): Single Error Correction
  - Memory-Controller ECC: Chipkill ECC



DRAM ECC against Row Hammer Error

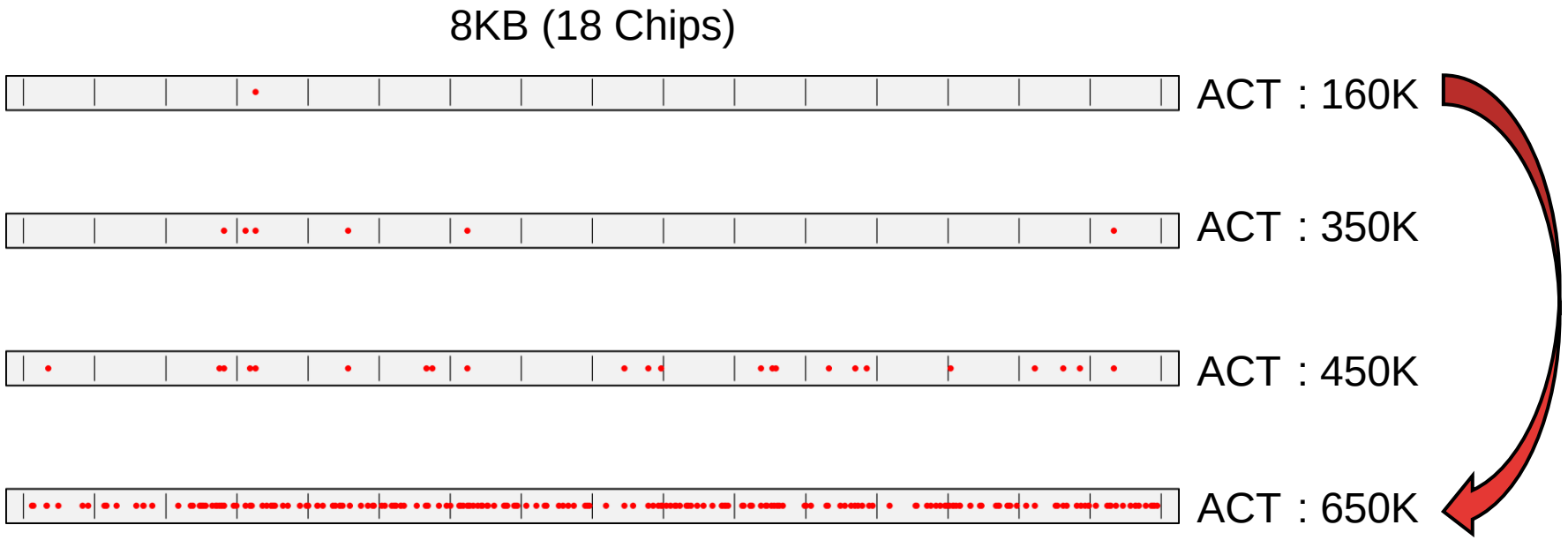
- Row Hammer experiment
  - SoftMC-based[1] **FPGA** environment
  - Xilinx Alveo U280
  - DDR4 RDIMM
    - 2 vendors
    - 14 DIMMS from 2016 to 2021
    - Total of 208 chips
    - No OECC
  - 8 intra-column data patterns
    - No cross-column patterns
  - Total of 16K rows, 1K rows per each bank
  - 50°C



[1] Hasan Hassan, Nandita Vijaykumar, Samira Khan, Saugata Ghose, Kevin Chang, Gennady Pekhimenko, Donghyuk Lee, Oğuz Ergin, and Onur Mutlu. 2017. SoftMC: A Flexible and Practical Open-Source Infrastructure for Enabling Experimental DRAM Studies.

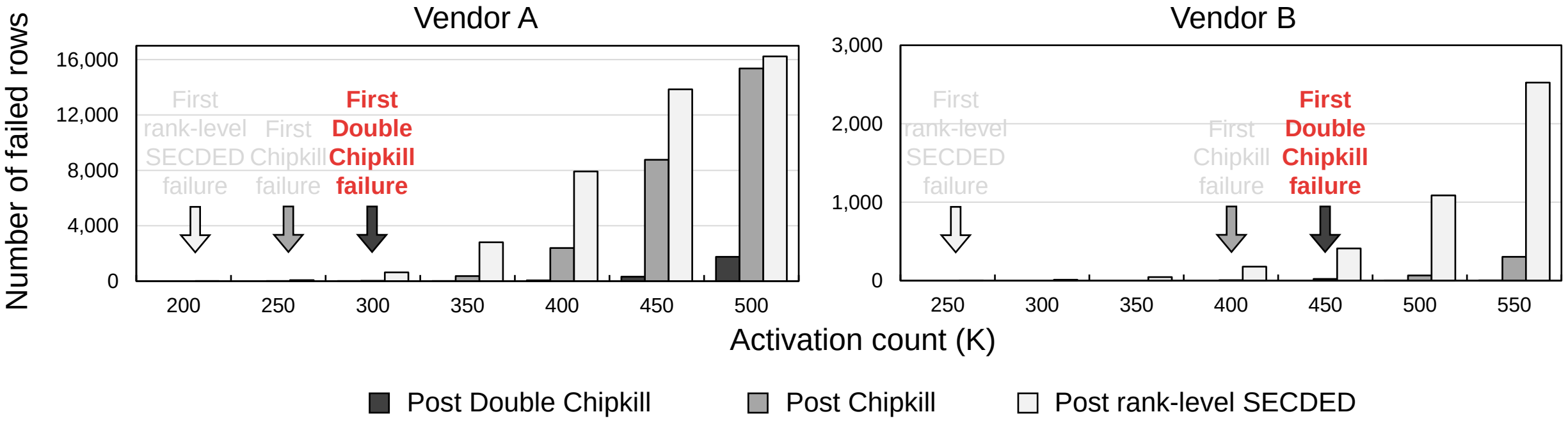
DRAM ECC again Row Hammer?

- Errors occur in a **bursty** way, as ACT count increases.
  - Example: row #10142



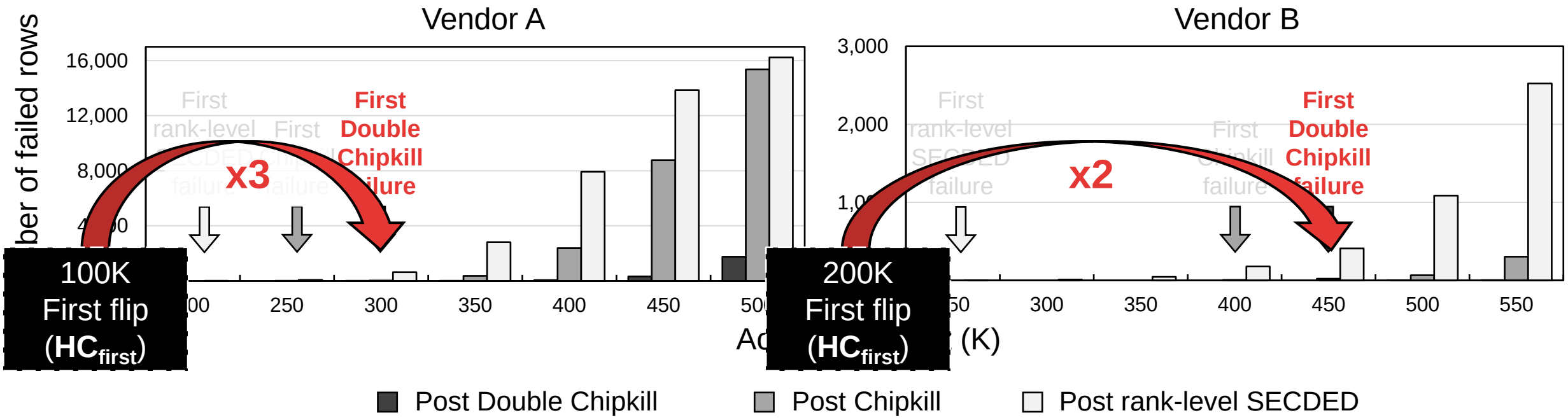
DRAM ECC again Row Hammer?

- ECC cannot tolerate victim row **concentrated** bursty error.
  - Double Chipkill-level ECC provides only a x3 increase in tolerable hammer count.



DRAM ECC again Row Hammer?

- ECC cannot tolerate victim row **concentrated** bursty error.
  - Double Chipkill-level ECC provides only a x3 increase in tolerable hammer count.



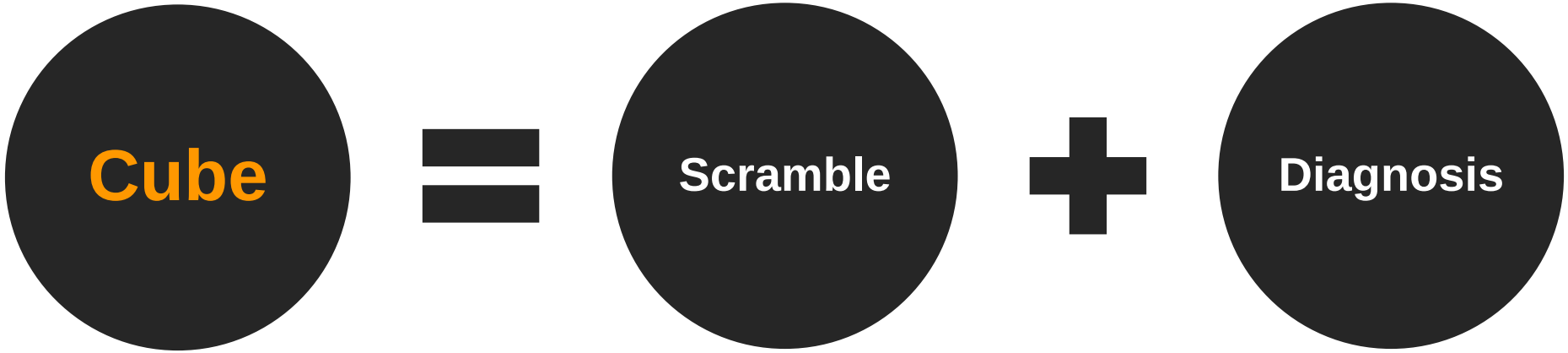
Cube

Cube

- Goal of Cube:
  - Better utilize the correction & detection capabilities of the **existing ECC against Row Hammer**.
  - Reduce the overall cost by **cooperating** with the probabilistic Row Hammer schemes.

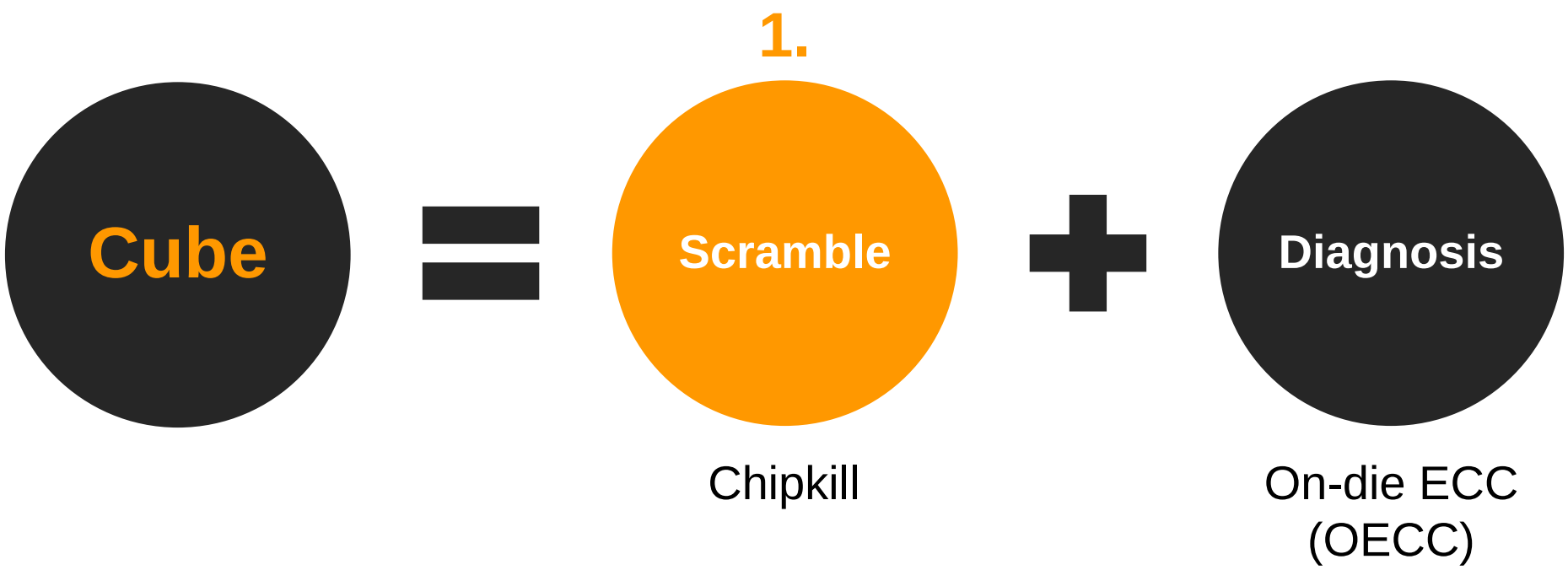
Cube

- Goal of Cube:
  - Better utilize the correction & detection capabilities of the **existing ECC against Row Hammer**.
  - Reduce the overall cost by **cooperating** with the probabilistic Row Hammer schemes.



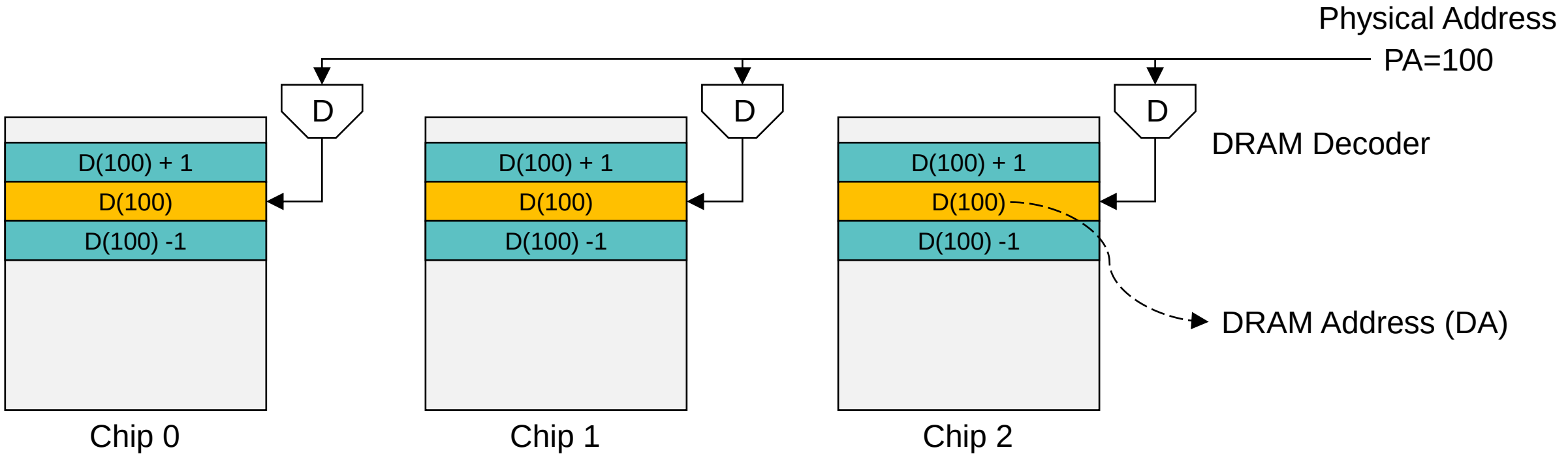
Cube

- Goal of Cube:
  - Better utilize the correction & detection capabilities of the **existing ECC against Row Hammer**.
  - Reduce the overall cost by **cooperating** with the probabilistic Row Hammer schemes.



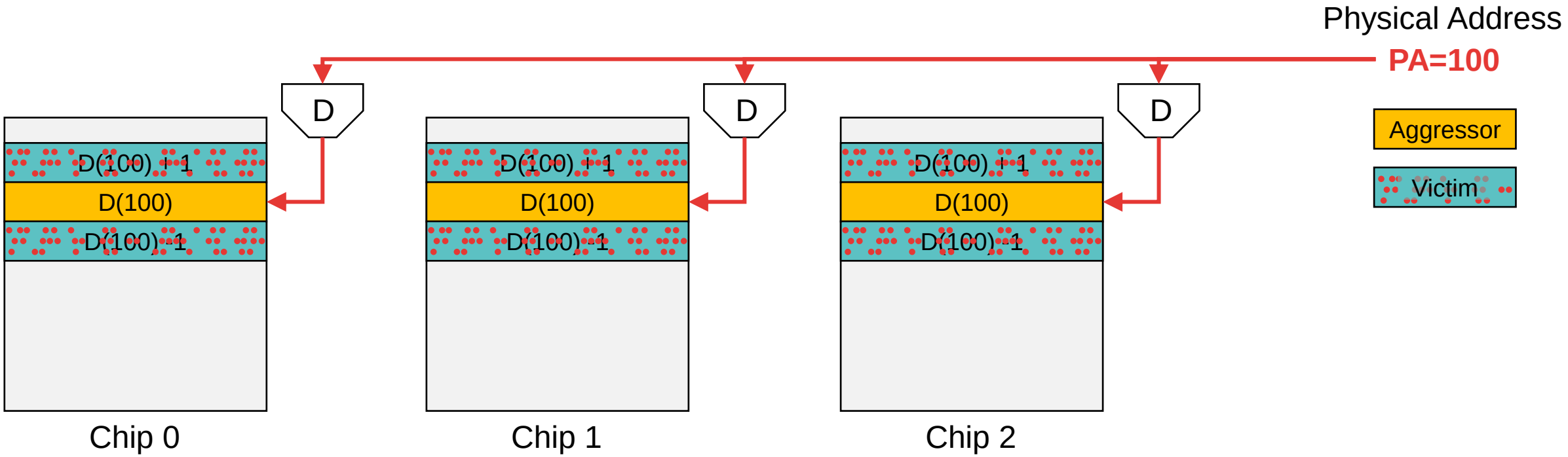
Cube 1 – Row Scramble

- Conventional addressing mechanism
  - Victims are **concentrated** to two Chipkill codewords



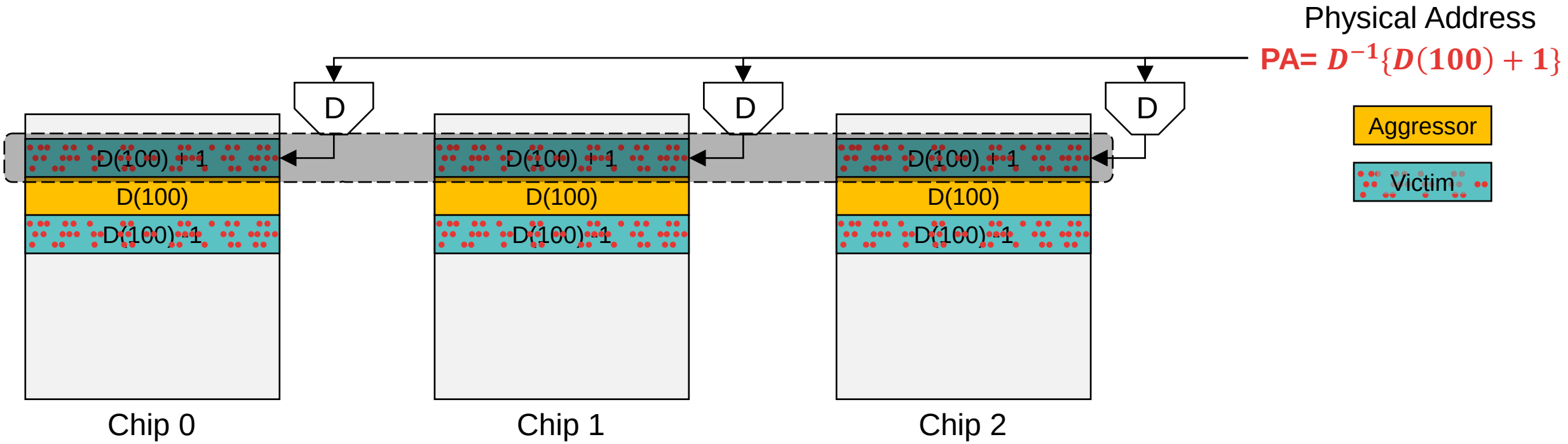
Cube 1 – Row Scramble

- Conventional addressing mechanism
  - Victims are **concentrated** to two Chipkill codewords



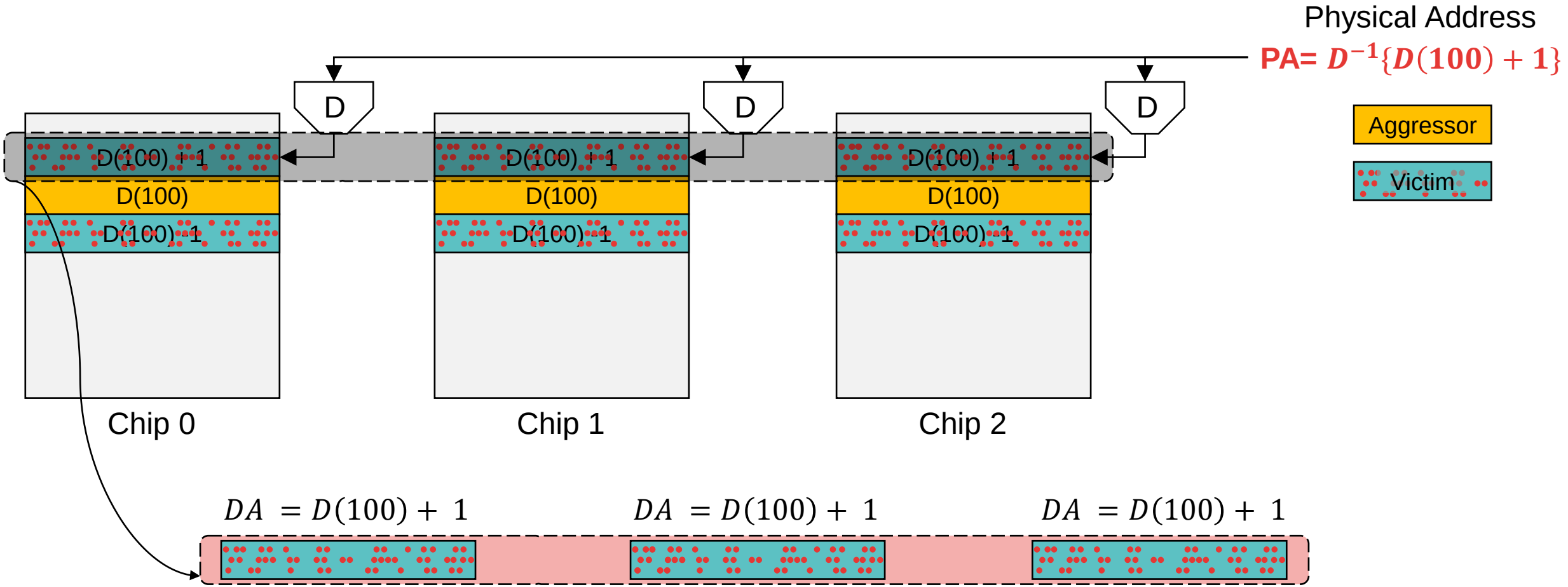
Cube 1 – Row Scramble

- Conventional addressing mechanism
  - Victims are **concentrated** to two Chipkill codewords



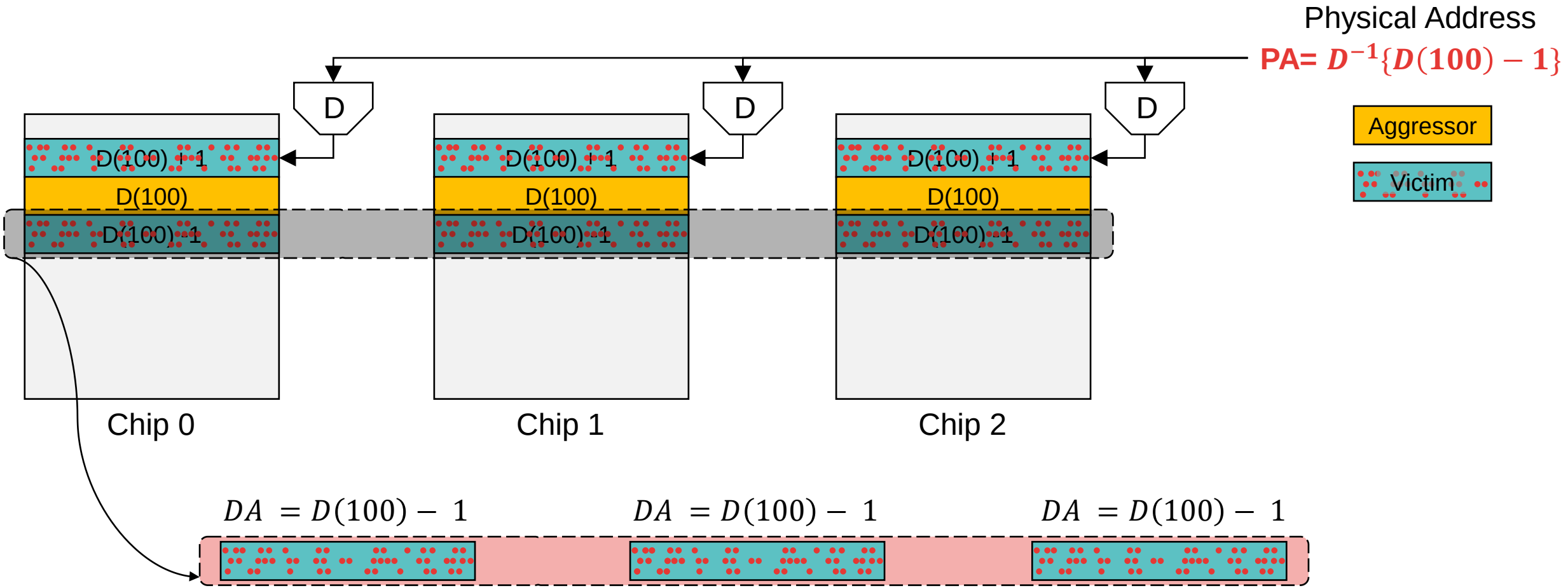
Cube 1 – Row Scramble

- Conventional addressing mechanism
  - Victims are **concentrated** to two Chipkill codewords



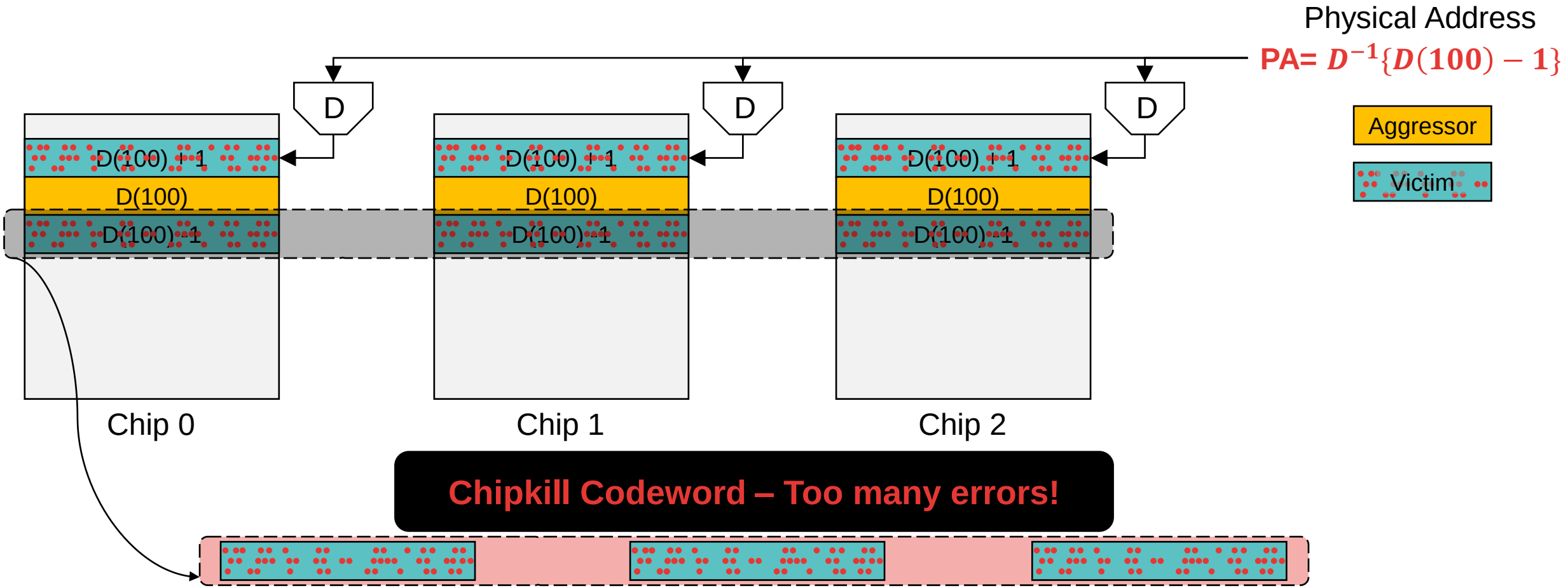
Cube 1 – Row Scramble

- Conventional addressing mechanism
  - Victims are **concentrated** to two Chipkill codewords



Cube 1 – Row Scramble

- Conventional addressing mechanism
  - Victims are **concentrated** to two Chipkill codewords

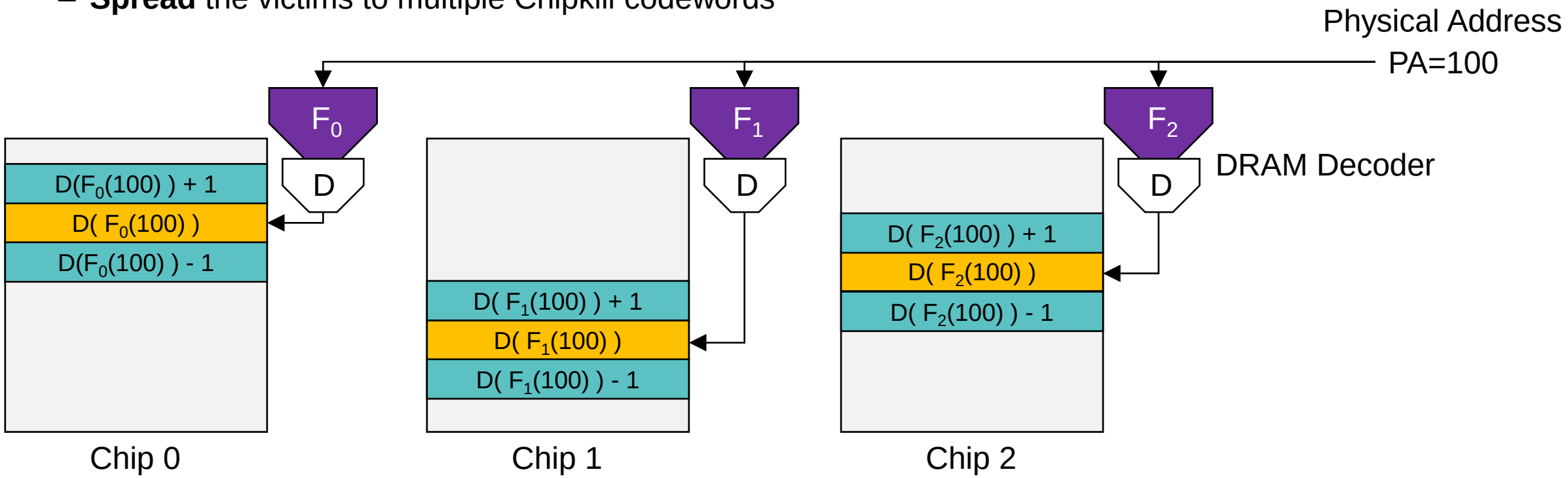


Cube 1 – Row Scramble

- Make every chip’s victim unique
  - **Spread** the victims to multiple Chipkill codewords

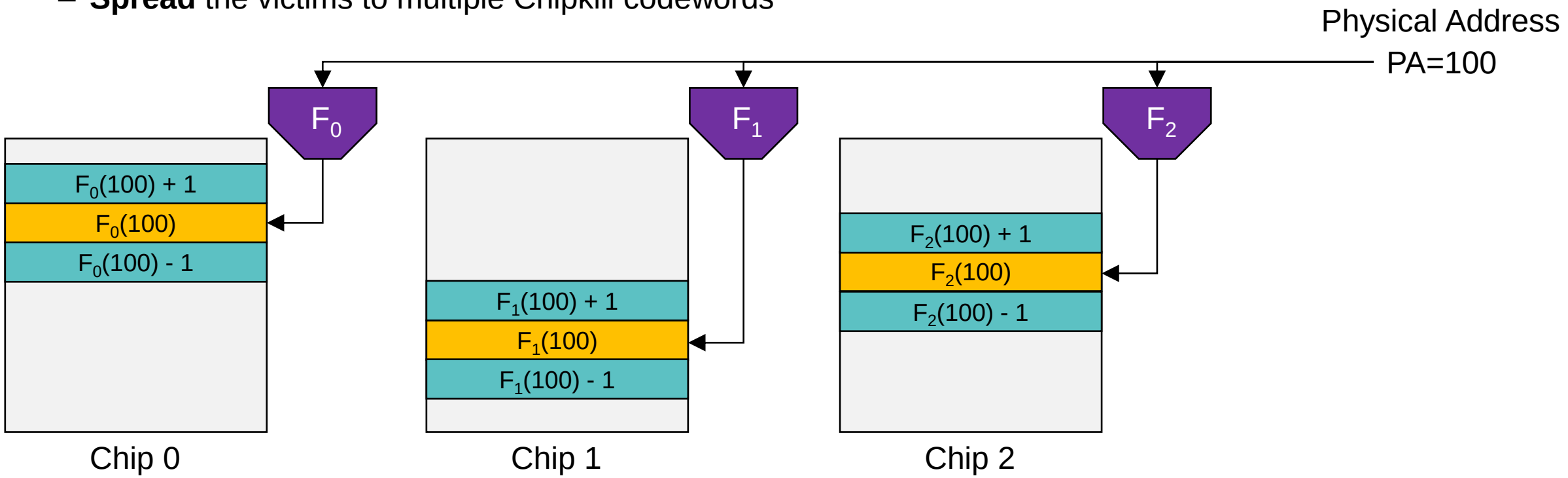
Cube 1 – Row Scramble

- Make every chip's victim unique
  - **Spread** the victims to multiple Chipkill codewords



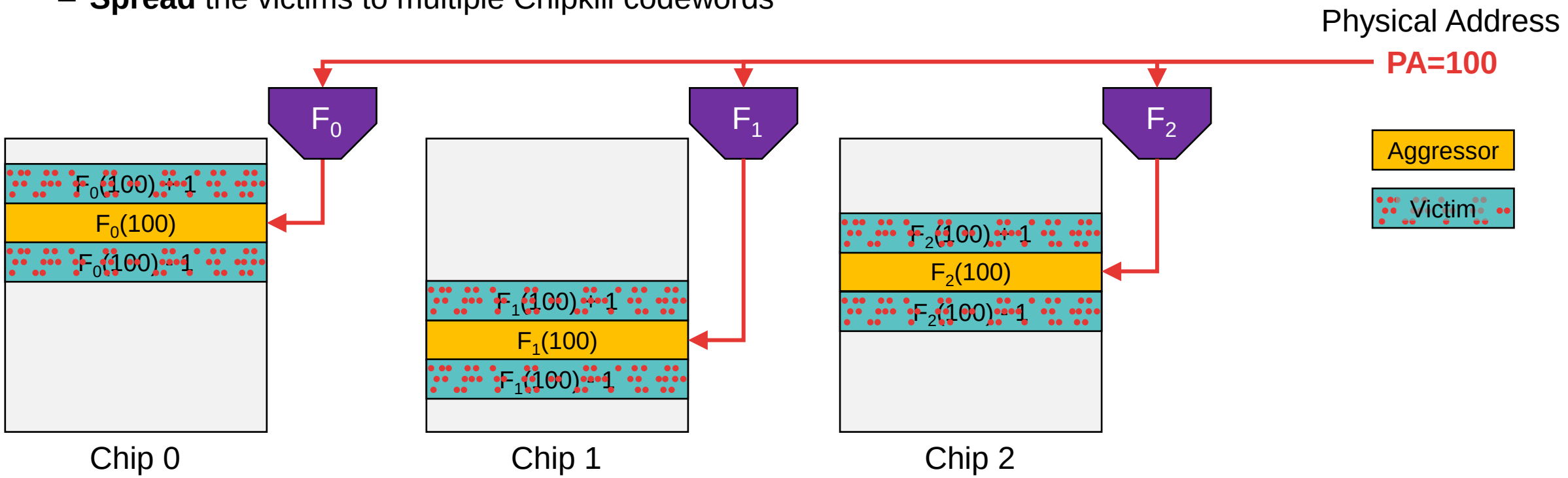
Cube 1 – Row Scramble

- Make every chip's victim unique
  - **Spread** the victims to multiple Chipkill codewords



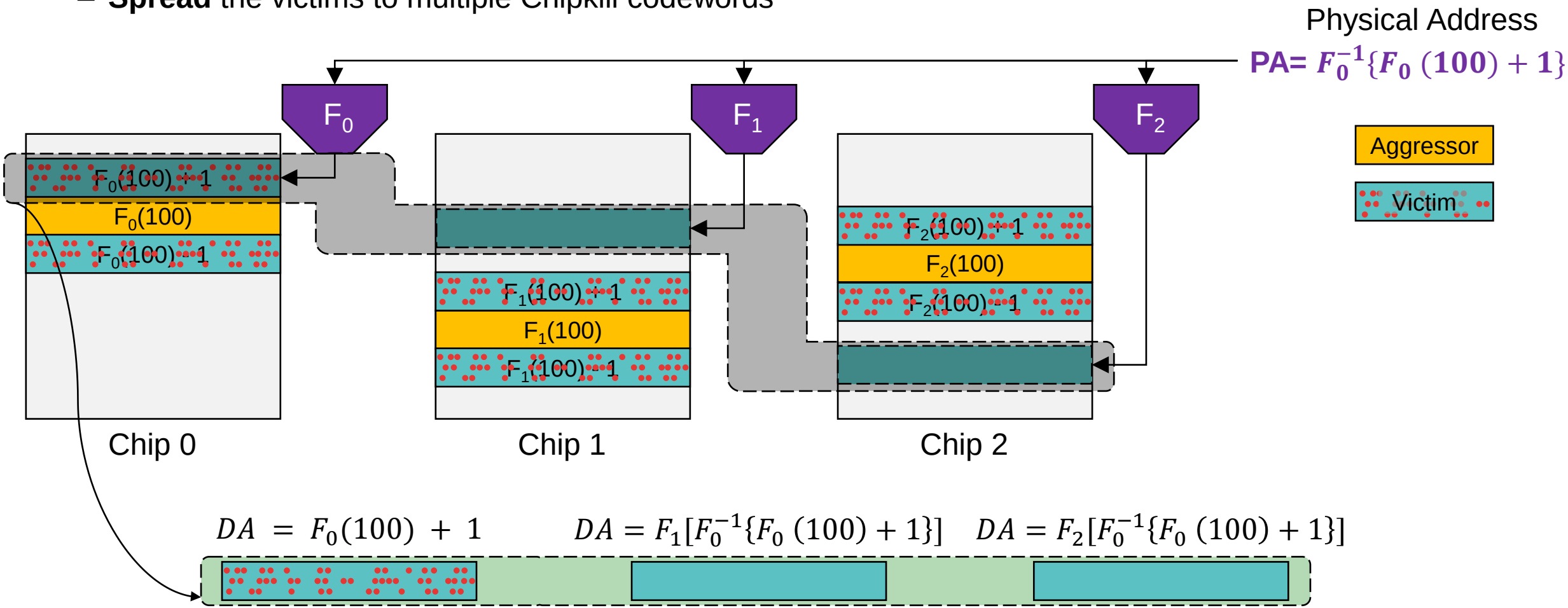
Cube 1 – Row Scramble

- Make every chip's victim unique
  - **Spread** the victims to multiple Chipkill codewords



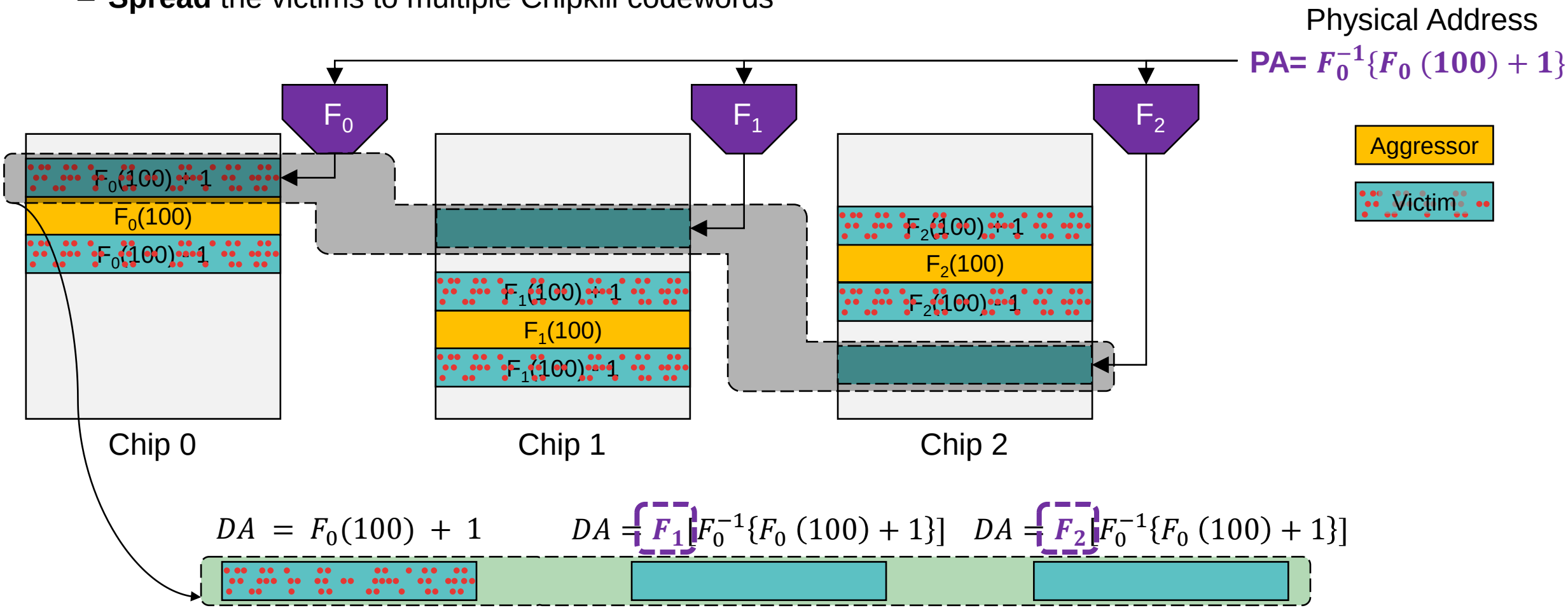
Cube 1 – Row Scramble

- Make every chip's victim unique
  - **Spread** the victims to multiple Chipkill codewords



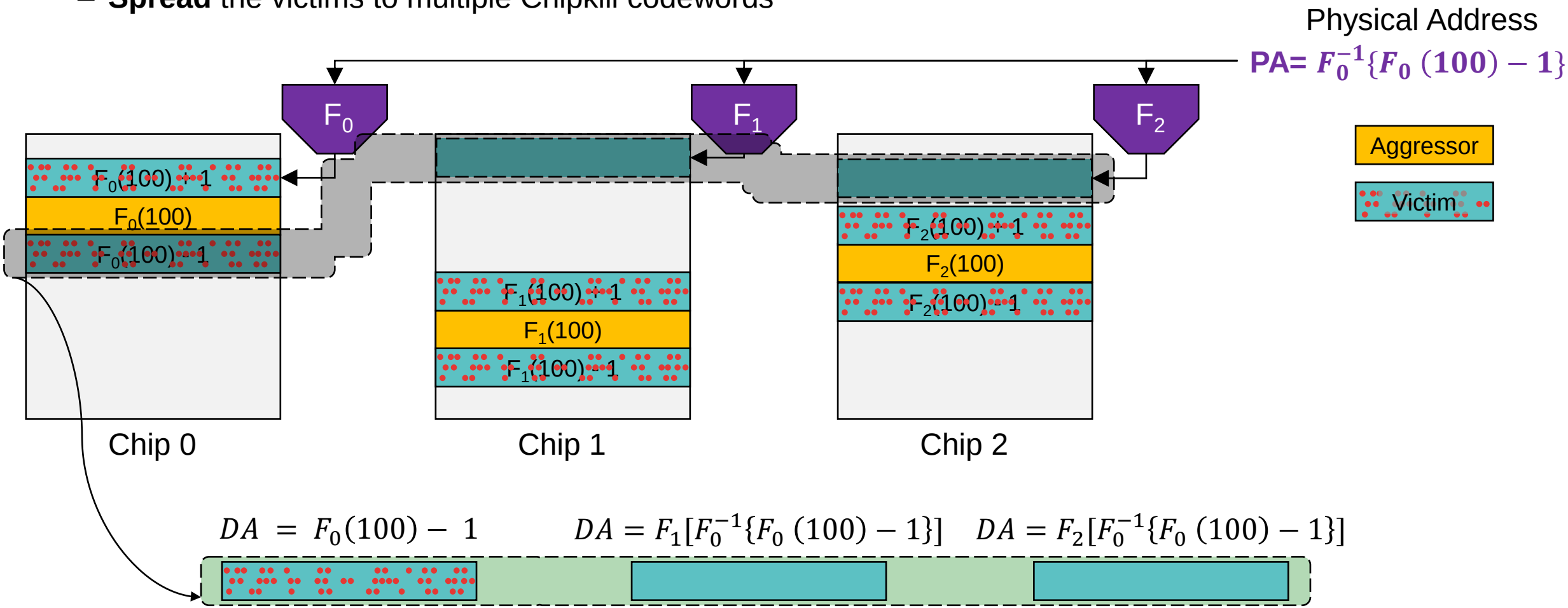
Cube 1 – Row Scramble

- Make every chip's victim unique
  - **Spread** the victims to multiple Chipkill codewords



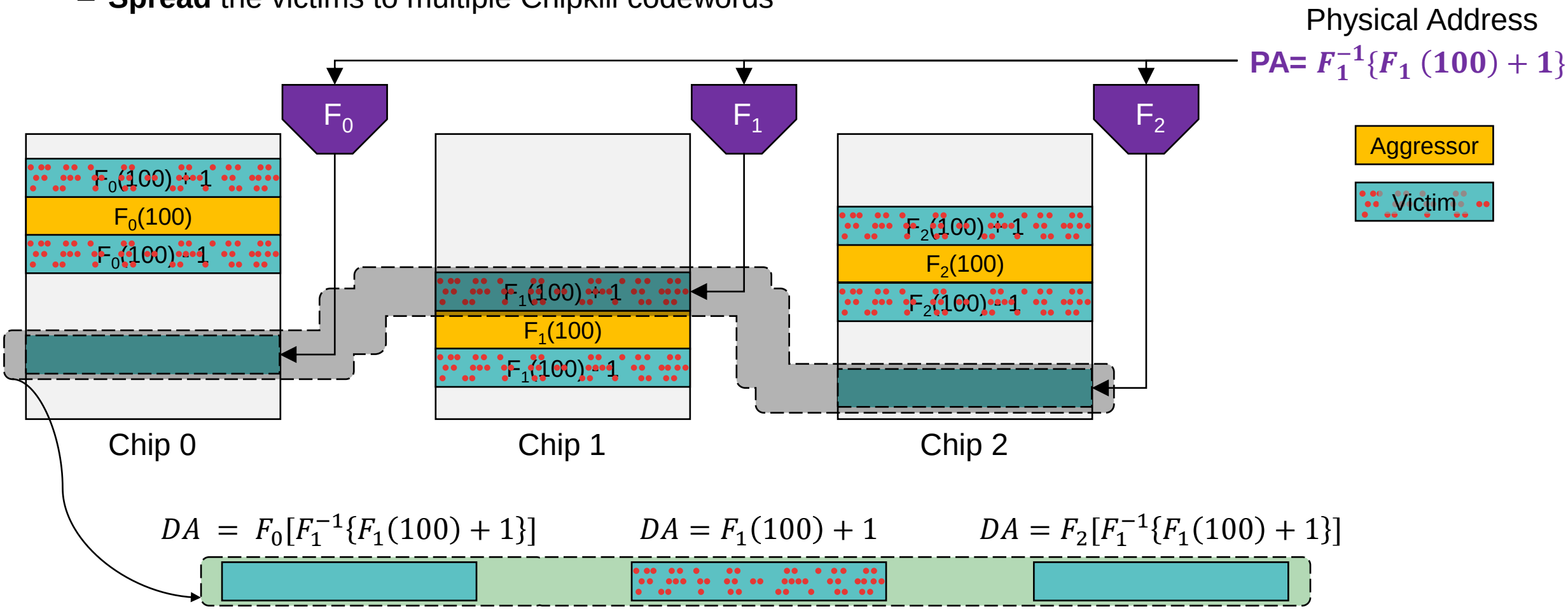
Cube 1 – Row Scramble

- Make every chip's victim unique
  - **Spread** the victims to multiple Chipkill codewords



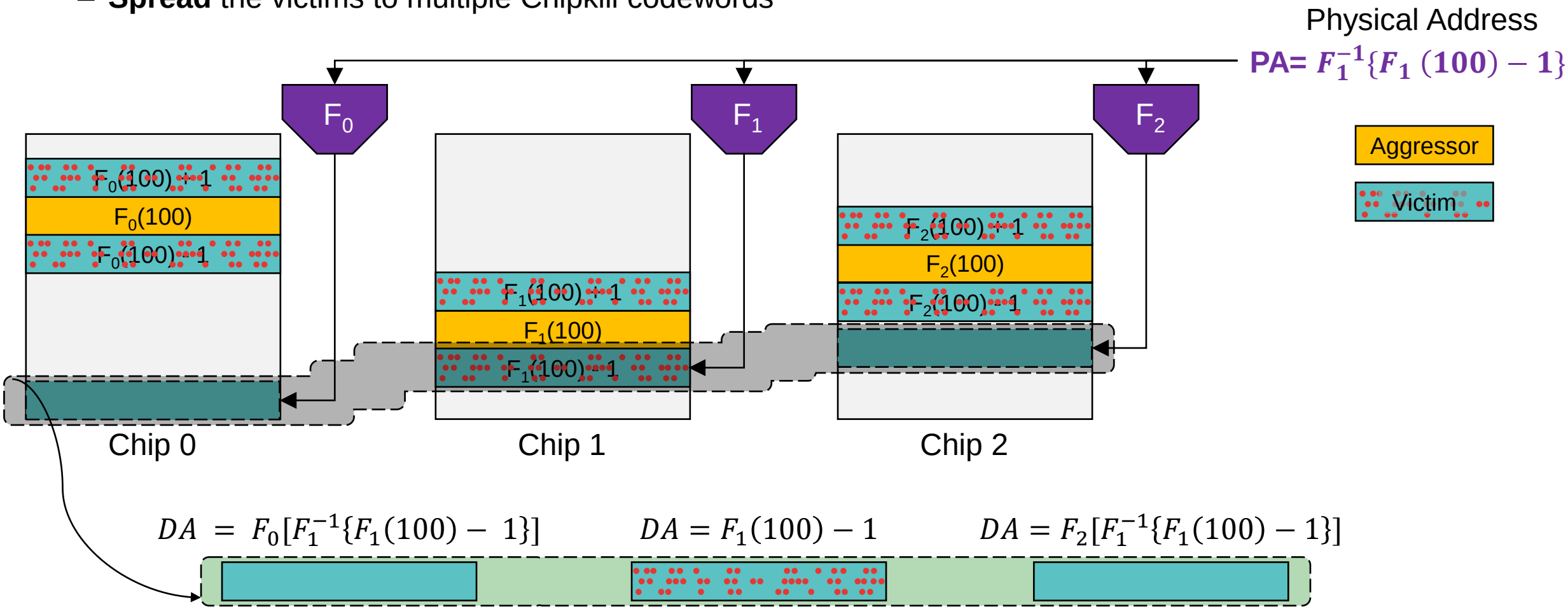
Cube 1 – Row Scramble

- Make every chip's victim unique
  - **Spread** the victims to multiple Chipkill codewords



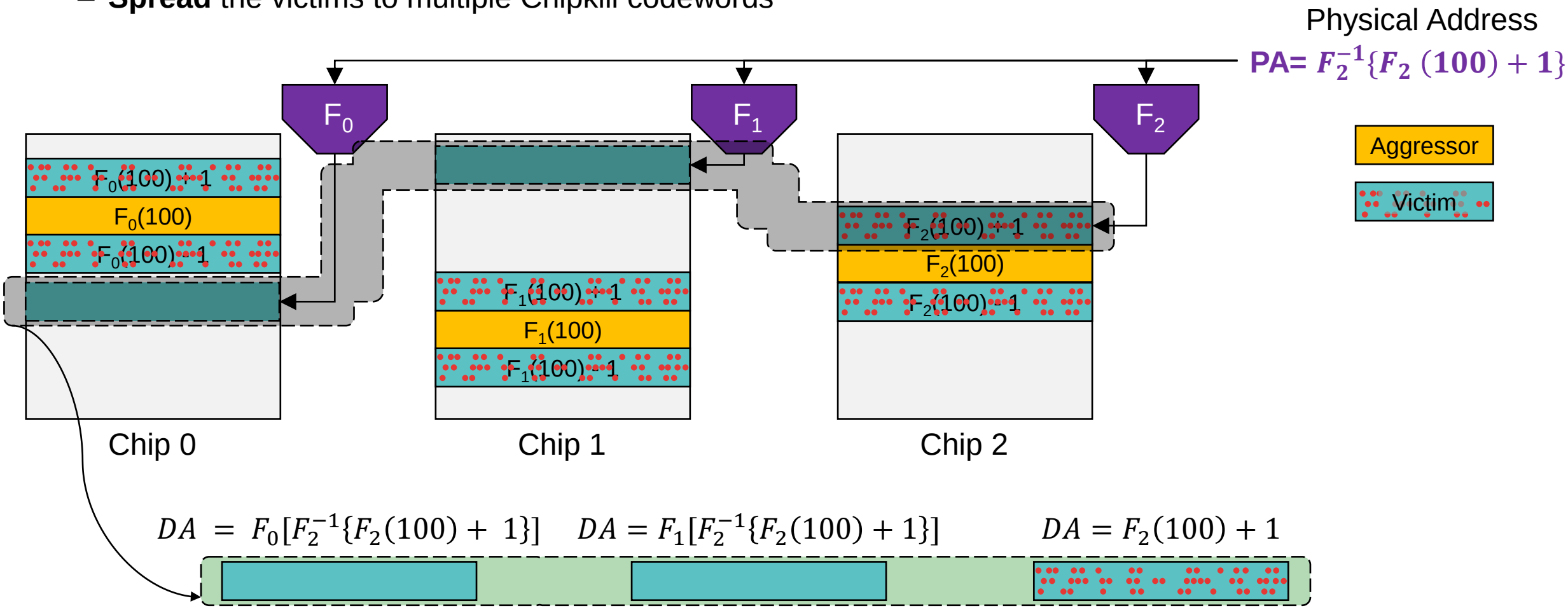
Cube 1 – Row Scramble

- Make every chip's victim unique
  - **Spread** the victims to multiple Chipkill codewords



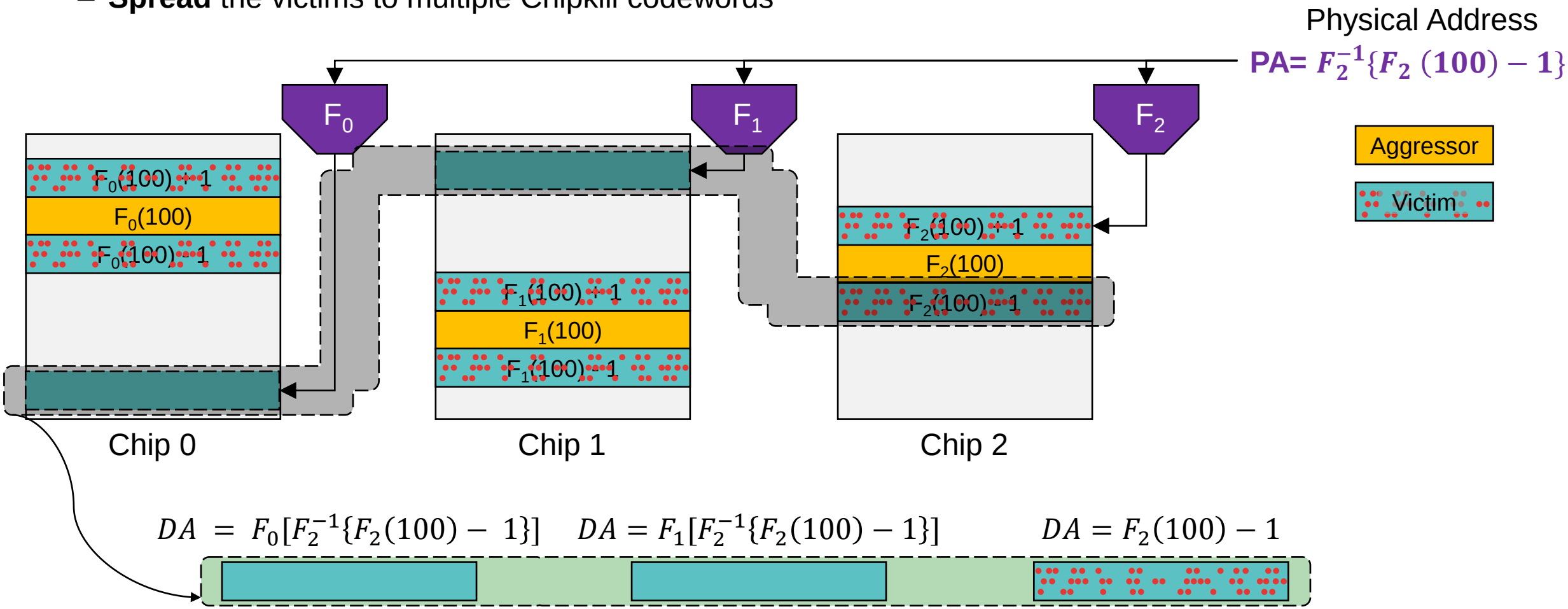
Cube 1 – Row Scramble

- Make every chip's victim unique
  - **Spread** the victims to multiple Chipkill codewords



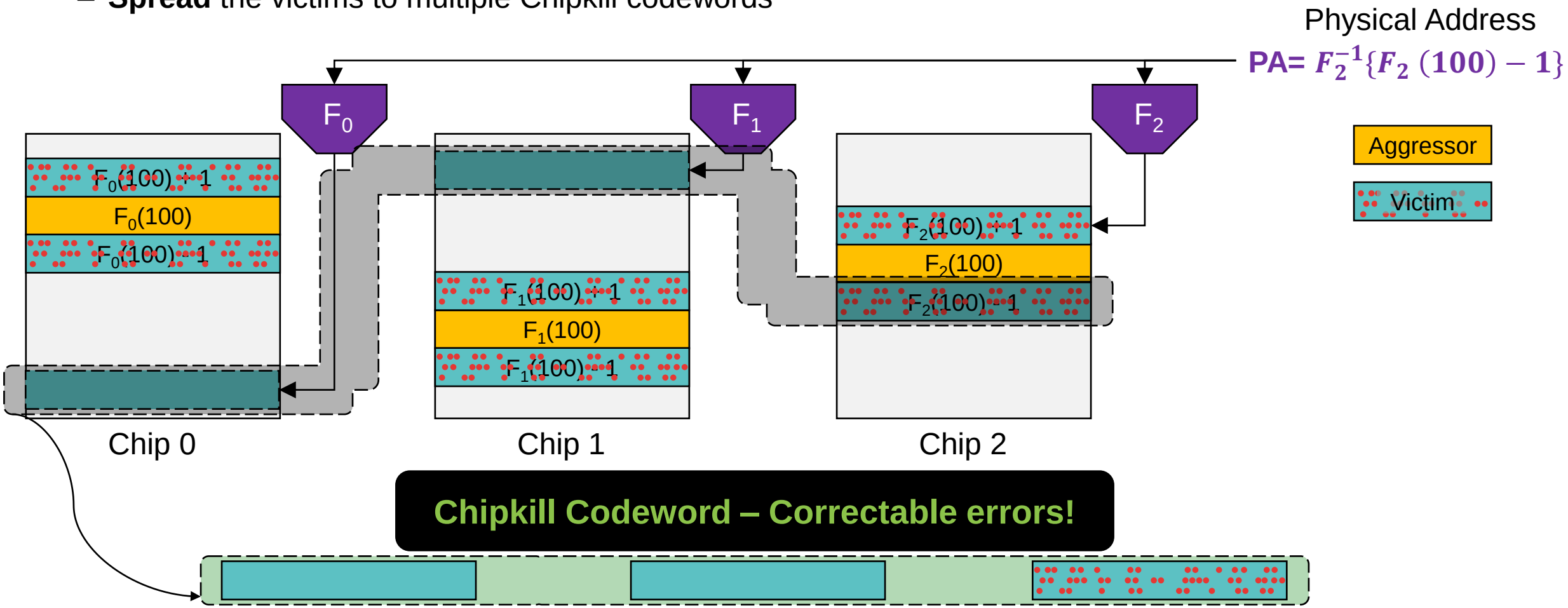
Cube 1 – Row Scramble

- Make every chip's victim unique
  - **Spread** the victims to multiple Chipkill codewords



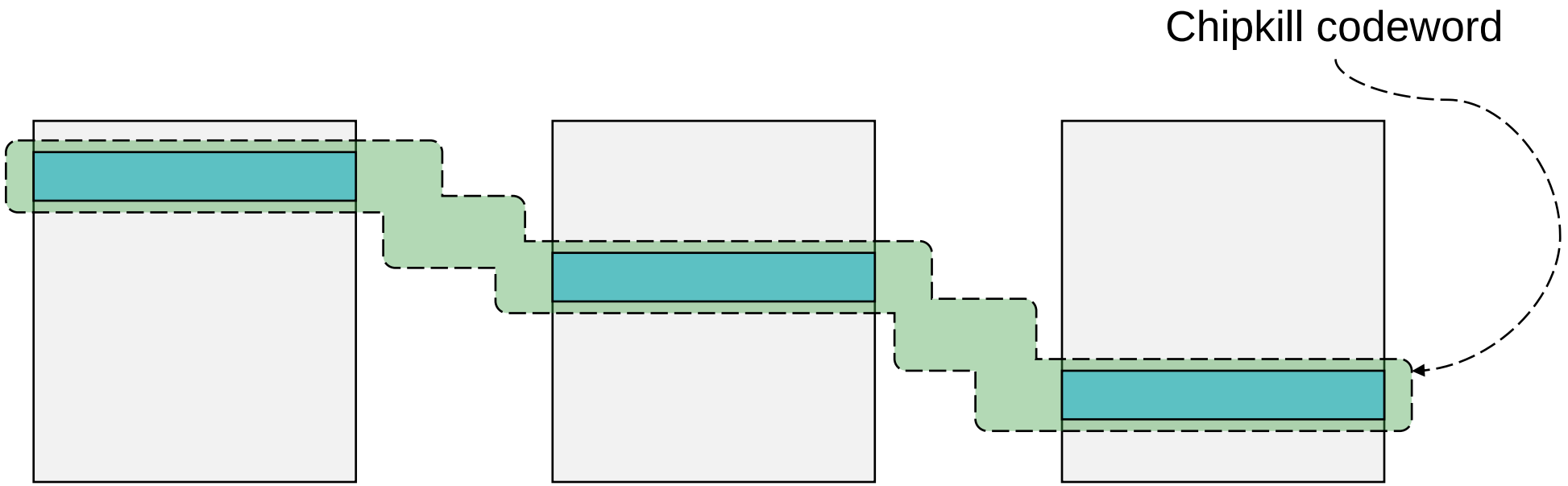
Cube 1 – Row Scramble

- Make every chip's victim unique
  - **Spread** the victims to multiple Chipkill codewords



Cube 1 – Row Scramble

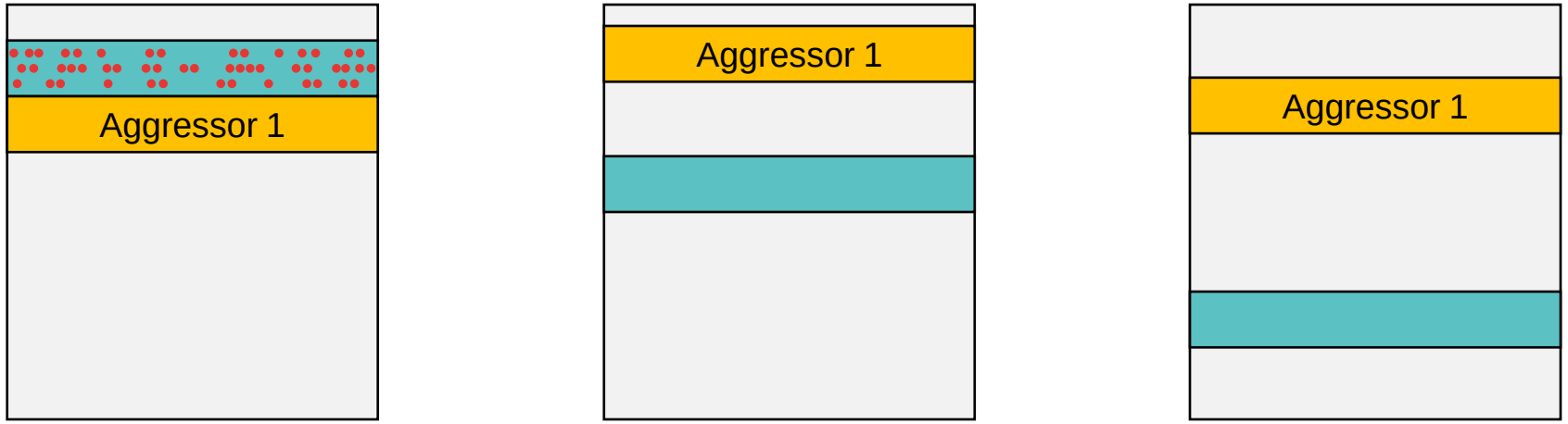
- Force the attacker to succeed in not only **one**, not only **two**, but **multiple Row Hammer attacks**!
  - Theoretically, two successful Row Hammer attack can be enough, but with low probability.



Cube 1 – Row Scramble

- Force the attacker to succeed in not only **one**, not only **two**, but **multiple Row Hammer attacks**!
  - Theoretically, two successful Row Hammer attack can be enough, but with low probability.

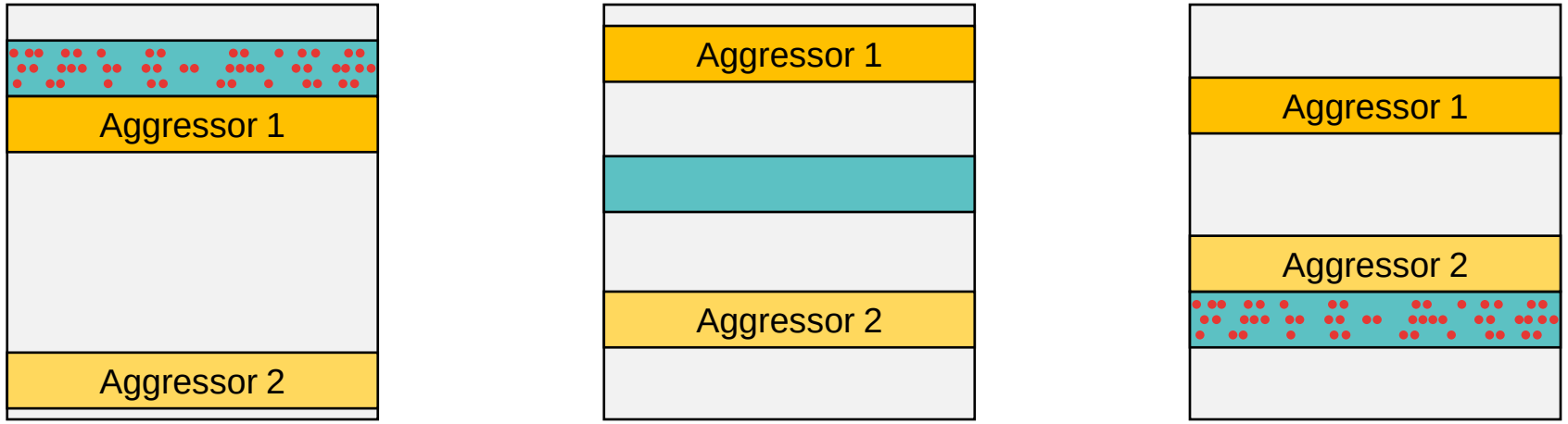
1. (Theoretical) Two successful Row Hammer attack



Cube 1 – Row Scramble

- Force the attacker to succeed in not only **one**, not only **two**, but **multiple Row Hammer attacks**!
  - Theoretically, two successful Row Hammer attack can be enough, but with low probability.

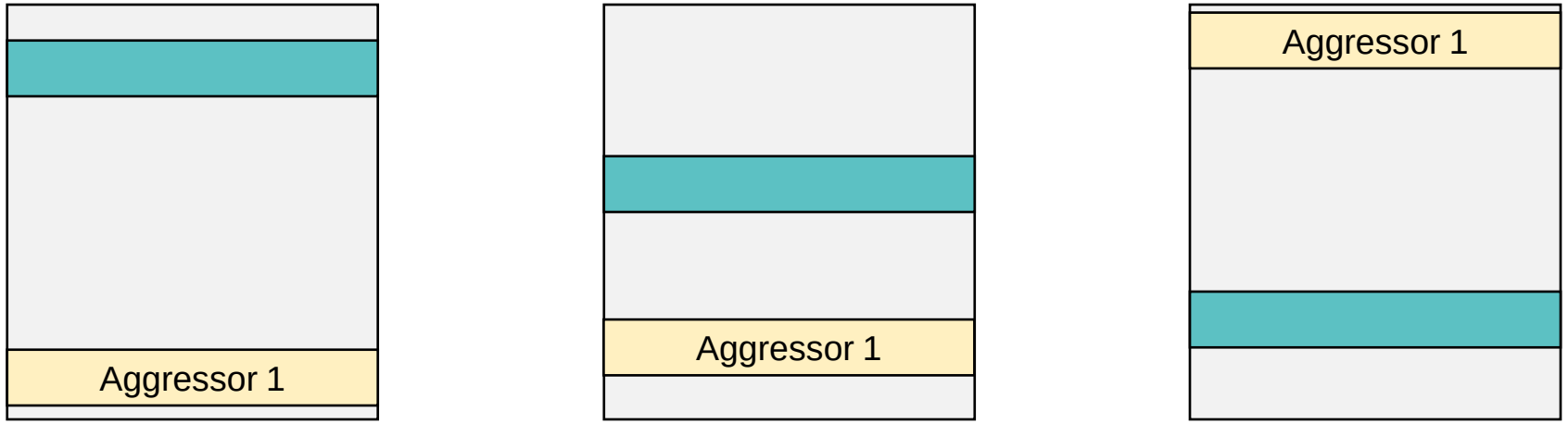
1. (Theoretical) Two successful Row Hammer attack



Cube 1 – Row Scramble

- Force the attacker to succeed in not only **one**, not only **two**, but **multiple Row Hammer attacks**!
  - Theoretically, two successful Row Hammer attack can be enough, but with low probability.
  - **Without knowing the mapping function F**, multiple Row Hammer attacks are likely needed.
    - "Birthday collision"

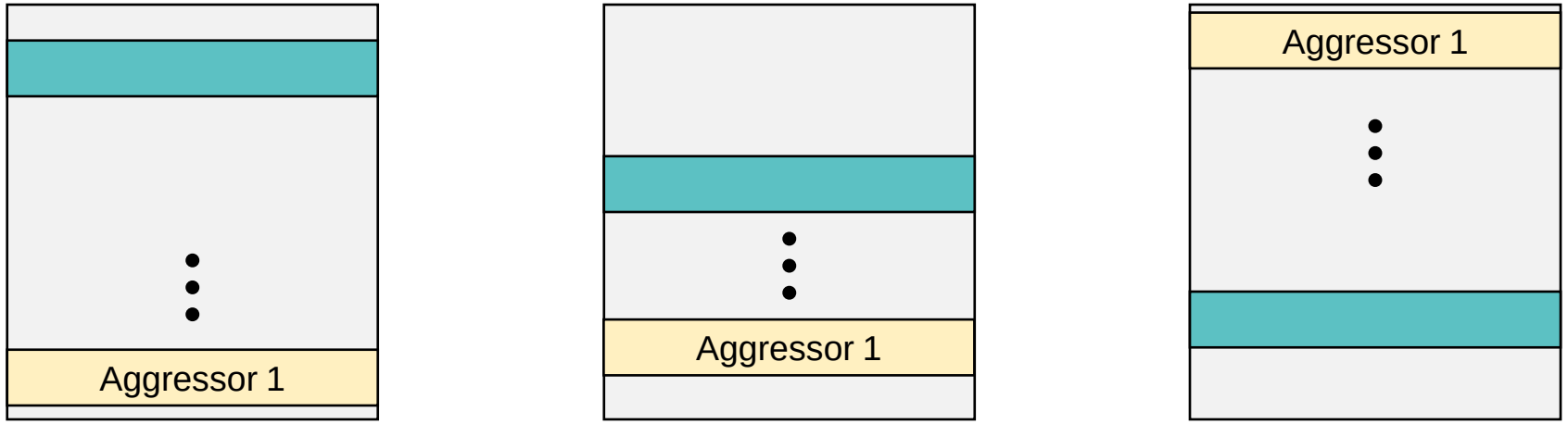
2. Multiple successful Row Hammer attack



Cube 1 – Row Scramble

- Force the attacker to succeed in not only **one**, not only **two**, but **multiple Row Hammer attacks**!
  - Theoretically, two successful Row Hammer attack can be enough, but with low probability.
  - **Without knowing the mapping function F**, multiple Row Hammer attacks are likely needed.
    - "Birthday collision"

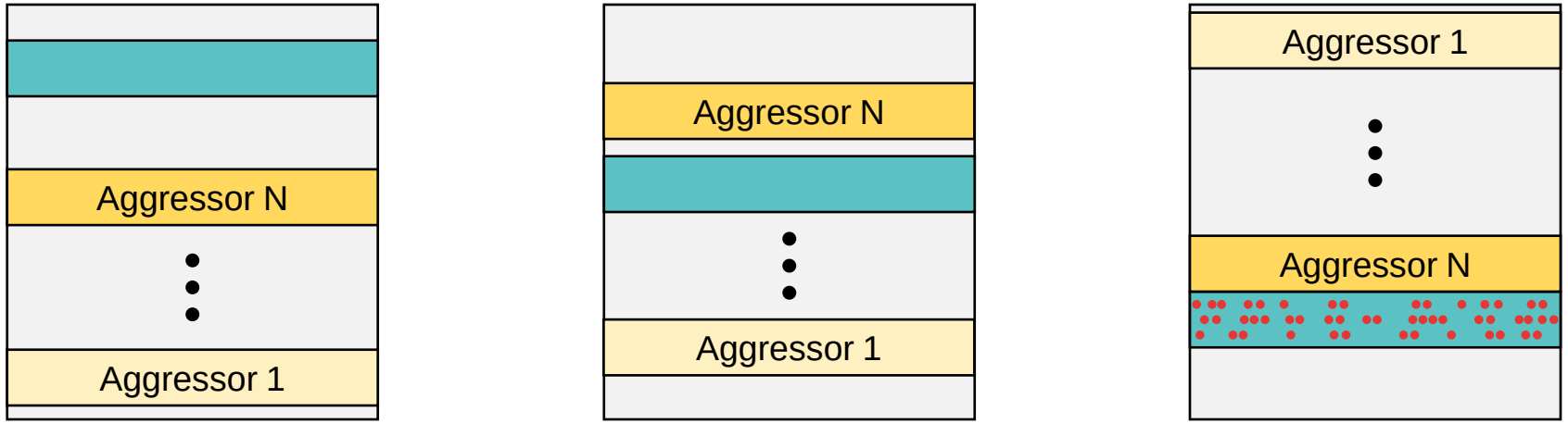
2. Multiple successful Row Hammer attack



Cube 1 – Row Scramble

- Force the attacker to succeed in not only **one**, not only **two**, but **multiple Row Hammer attacks**!
  - Theoretically, two successful Row Hammer attack can be enough, but with low probability.
  - **Without knowing the mapping function F**, multiple Row Hammer attacks are likely needed.
    - "Birthday collision"

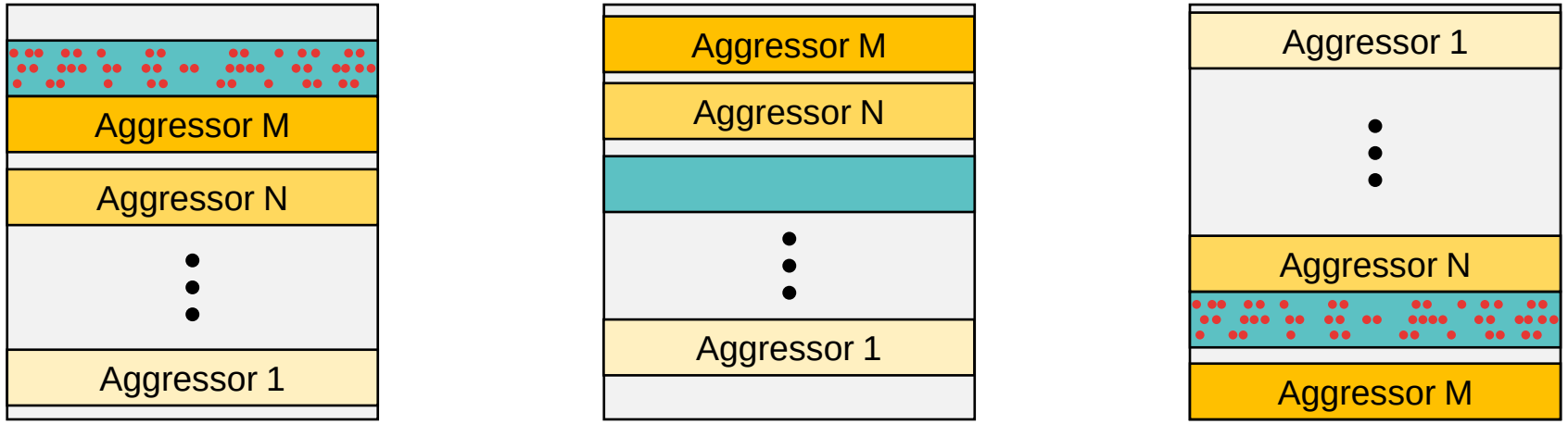
2. Multiple successful Row Hammer attack



Cube 1 – Row Scramble

- Force the attacker to succeed in not only **one**, not only **two**, but **multiple Row Hammer attacks**!
  - Theoretically, two successful Row Hammer attack can be enough, but with low probability.
  - **Without knowing the mapping function F**, multiple Row Hammer attacks are likely needed.
    - "Birthday collision"

2. Multiple successful Row Hammer attack



Cube 1 – Row Scramble

- Scramble function (F) requirements
  - *R-i)* For any given aggressor, victims must not collide.
  - *R-ii)* Must be fast, as it lies on the DRAM access critical path.
  - *R-iii)* No aliasing.
  - *R-iv)* Mapping function must be hidden from the attacker.

Cube 1 – Row Scramble

- Scramble function (F) requirements
  - *R-i)* For any given aggressor, victims must not collide.
    - All victim PAs must not collide within the same victim set.
  - *R-ii)* Must be fast, as it lies on the DRAM access critical path.
    - No AES or Fisher-Yates shuffle.
  - *R-iii)* No aliasing.
    - No hashing.
  - *R-iv)* Mapping function must be hidden from the attacker.

Cube 1 – Row Scramble

- Scramble function (F) requirements
  - ***R-i)*** For any given aggressor, victims must not collide.
    - All victim PAs must not collide within the same victim set.
  - ***R-ii)*** Must be fast, as it lies on the DRAM access critical path.
    - No AES or Fisher-Yates shuffle.
  - ***R-iii)*** No aliasing.
    - No hashing.
  - ***R-iv)*** Mapping function must be hidden from the attacker.

Vanilla Scramble Function for chip i,

$$\boldsymbol{F_i(PA) := (PA \times c_i) \bmod N_{row}}$$

$$R\text{-}i) \Rightarrow \{c_i \neq c_j \; and \; c_i \neq (N_{row} - c_j), 0 \leq i, j < 18\}$$

$$R\text{-}iii) \Rightarrow c_i \; is \; odd$$

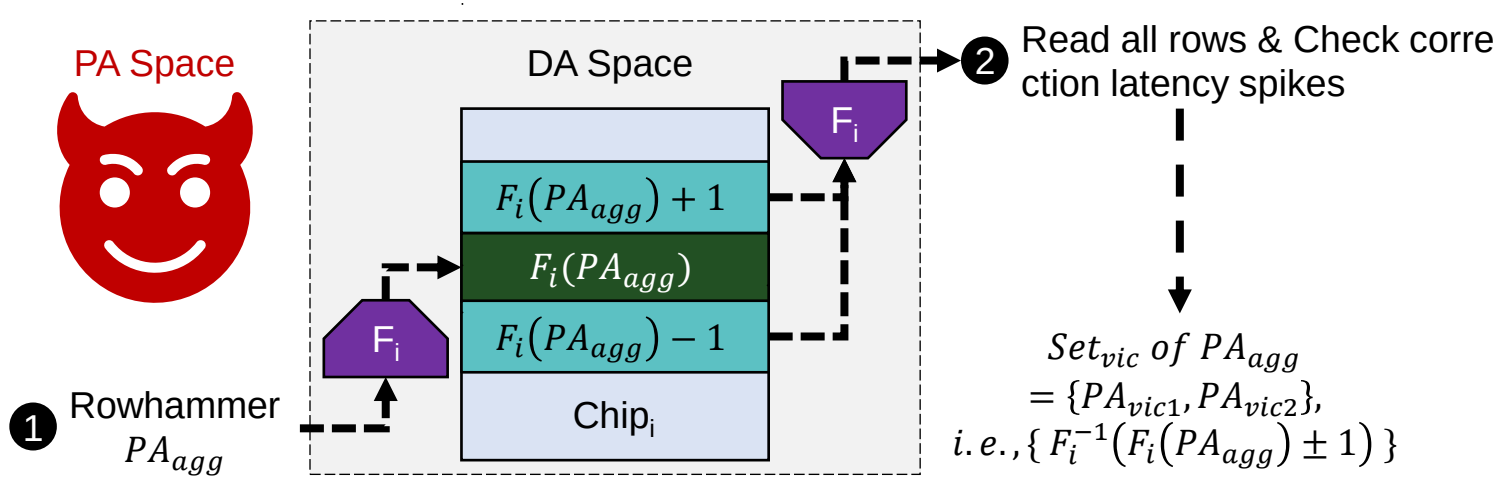
Cube 1 – Row Scramble

- Scramble function (F) requirements
  - *R-i)* For any given aggressor, victims must not collide.
  - *R-ii)* Must be fast, as it lies on the DRAM access critical path.
  - *R-iii)* No aliasing.
  - *R-iv)* Mapping function must be hidden from the attacker.
    - If not, two Row Hammer attacks are enough.
    - Possibility of side-channel from Chipkill correction latency[1].
      - A threat for **static scheme** (e.g., PARA)
      - Not an issue for dynamic scheme (e.g., SHADOW, SRS)

[1] Lucian Cojocar, Kaveh Razavi, Cristiano Giuffrida, and Herbert Bos. 2019. Exploiting Correcting Codes: On the Effectiveness of ECC Memory Against Rowhammer Attacks.

Cube 1 – Row Scramble

- Scramble function (F) requirements
  - *R-i)* For any given aggressor, victims must not collide.
  - *R-ii)* Must be fast, as it lies on the DRAM access critical path.
  - *R-iii)* No aliasing.
  - ***R-iv)*** Mapping function must be hidden from the attacker.
    - If not, two Row Hammer attacks are enough.
    - Possibility of side-channel from Chipkill correction latency[1].



[1] Lucian Cojocar, Kaveh Razavi, Cristiano Giuffrida, and Herbert Bos. 2019. Exploiting Correcting Codes: On the Effectiveness of ECC Memory Against Rowhammer Attacks.

Cube 1 – Row Scramble

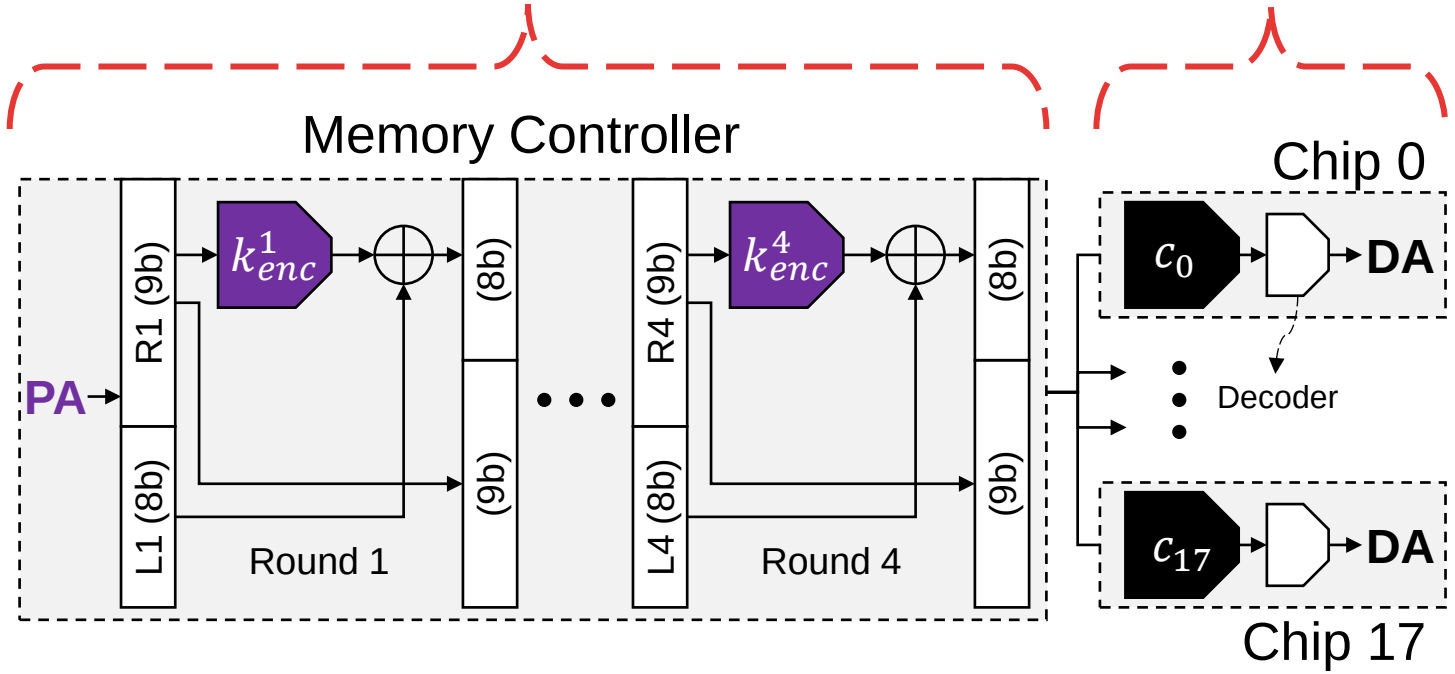
- Scramble function (F) requirements
  - *R-i)* For any given aggressor, victims must not collide.
  - *R-ii)* Must be fast, as it lies on the DRAM access critical path.
  - *R-iii)* No aliasing.
  - ***R-iv)*** Mapping function must be hidden from the attacker.
    - If not, two Row Hammer attacks are enough.
    - Possibility of side-channel from Chipkill correction latency[1].

(Optional) Encrypted Scramble Function $F_i(PA) := (Enc(PA) \times c_i) \bmod N_{row}$

[1] Lucian Cojocar, Kaveh Razavi, Cristiano Giuffrida, and Herbert Bos. 2019. Exploiting Correcting Codes: On the Effectiveness of ECC Memory Against Rowhammer Attacks.

Cube 1 – Row Scramble

- Scramble function (F) implementation
  - Global encryption scheme (LLBC[1])
  - Chip-wise fixed multiplication unit

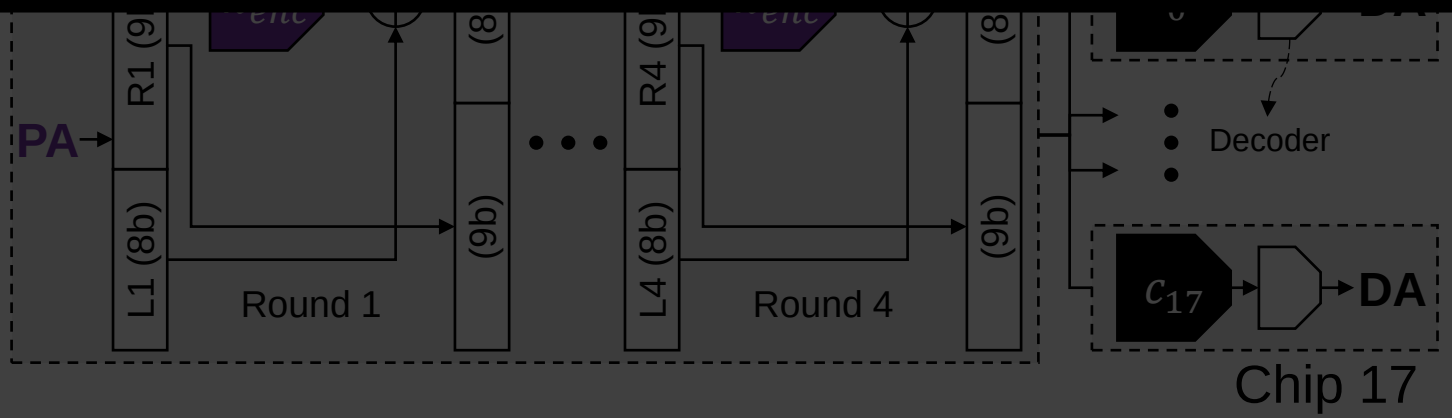


[1] Moinuddin K. Qureshi. 2018. CEASER: Mitigating Conflict-Based Cache Attacks via Encrypted-Address and Remapping.

Cube 1 – Row Scramble

- Scramble function (F) implementation
  - Chip-wise fixed multiplication unit
  - Global encryption scheme (LLBC[1])

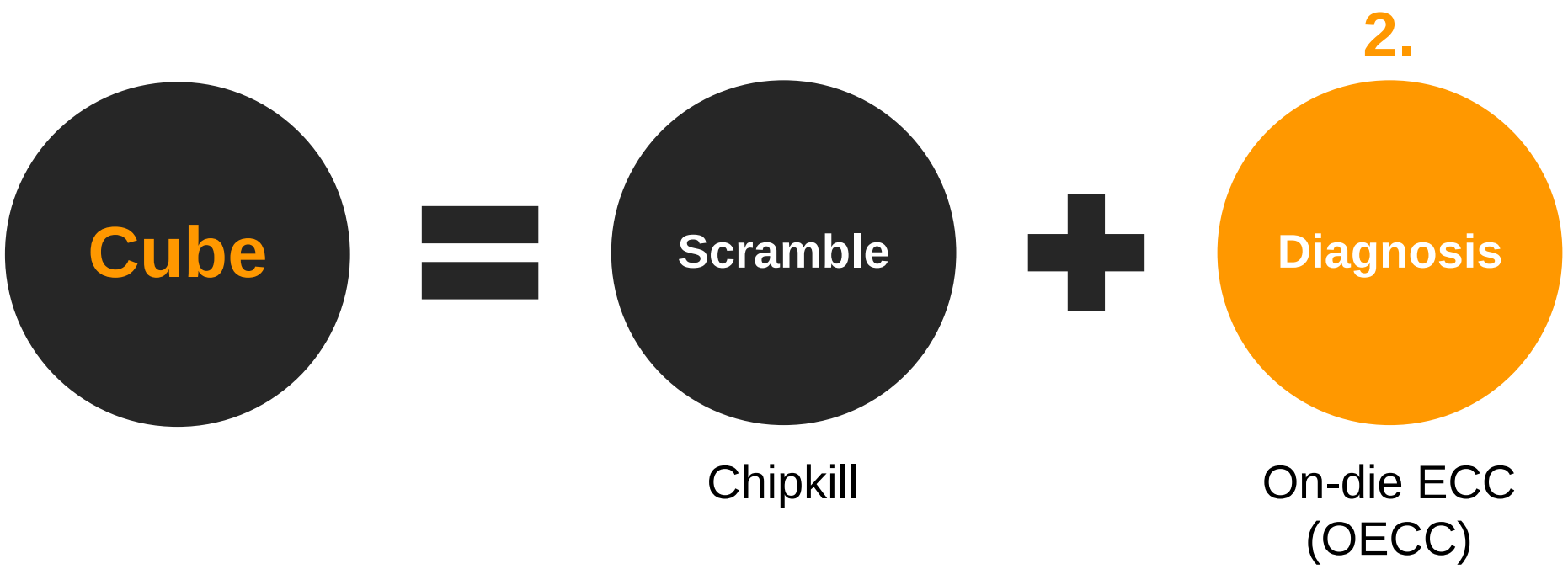
Please refer to the full paper for more detailed discussion on security and implementation.



[1] Soft-MC

Cube

- Goal of Cube:
  - Better utilize the correction & detection capabilities of the **existing ECC against Row Hammer**.
  - Reduce the overall cost by **cooperating** with the probabilistic Row Hammer schemes.

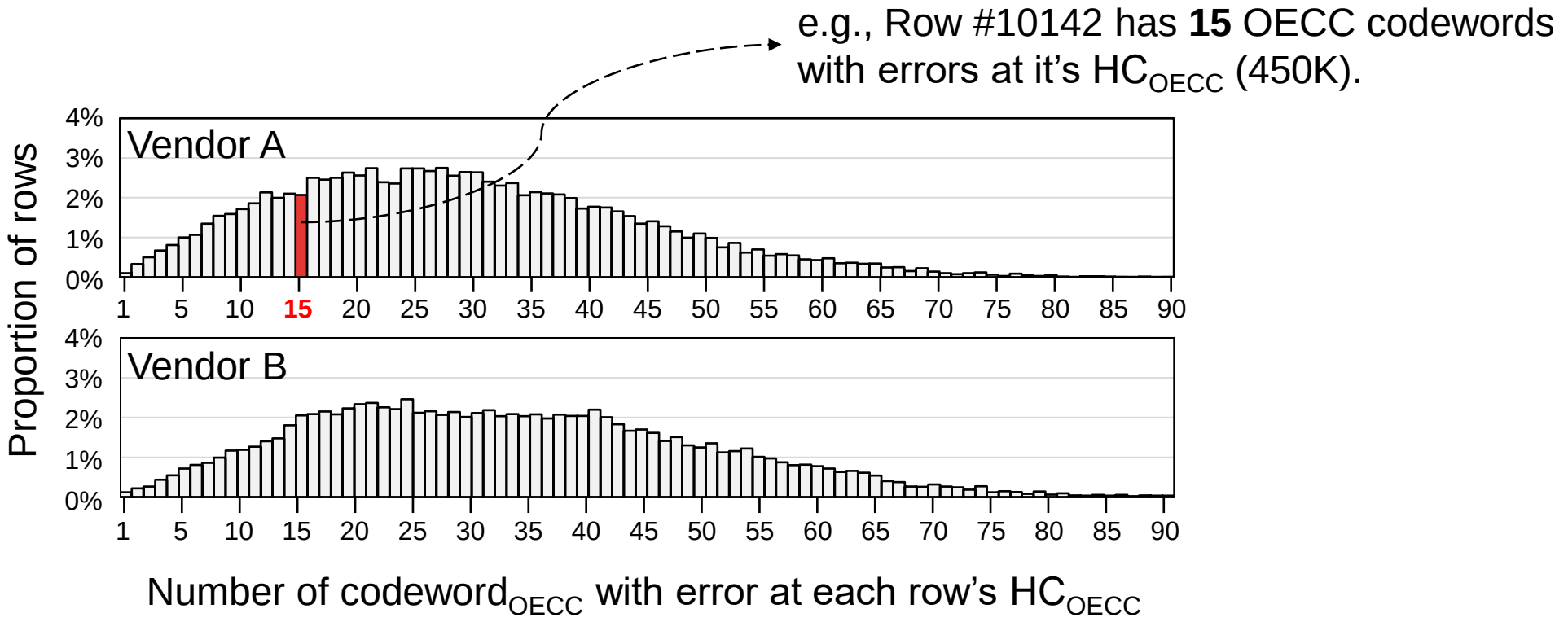


Cube 2 – Row Hammer Diagnosis with OECC Profile

- Row with a lot of **OECC errors** can indicate that it has been Row Hammer **victimized**.

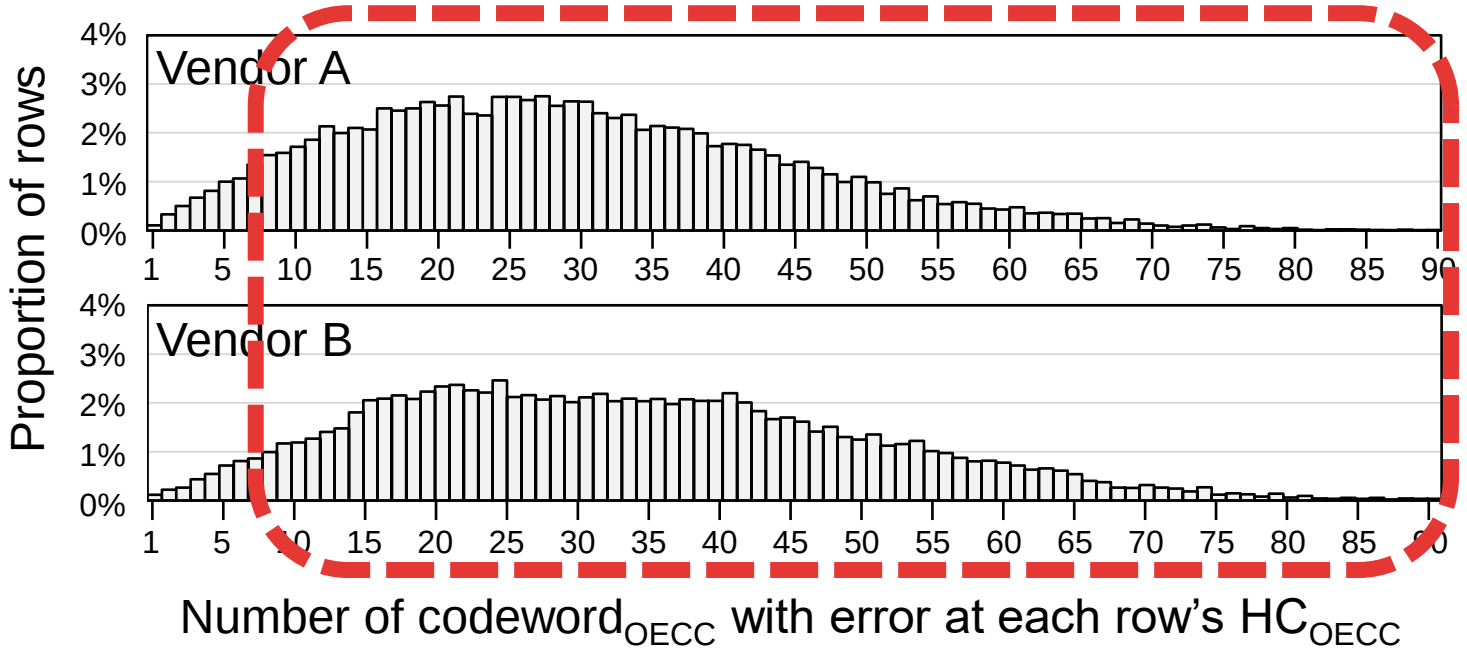
Cube 2 – Row Hammer Diagnosis with OECC Profile

- Row with a lot of **OECC errors** can indicate that it has been Row Hammer **victimized**.
  - At the first OECC **uncorrectable errors** (HC_{OECC}), many OECC **correctable errors** are likely to have occurred before.



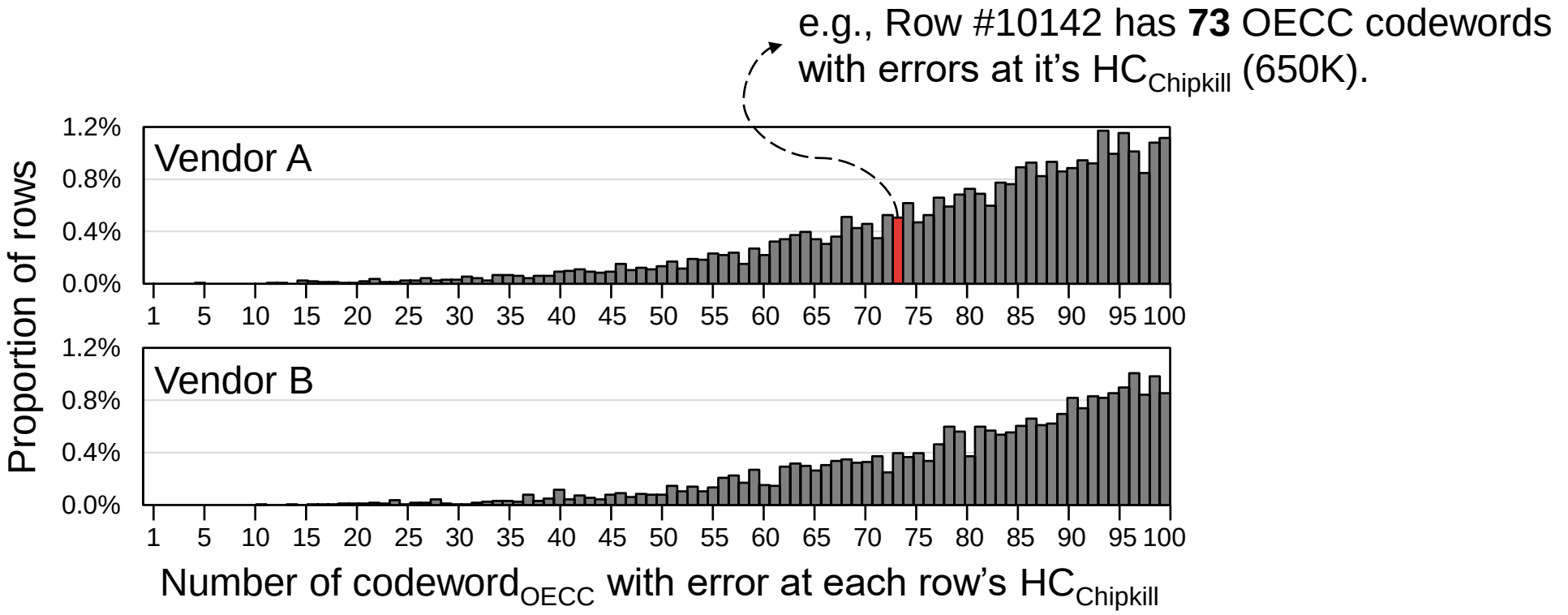
Cube 2 – Row Hammer Diagnosis with OECC Profile

- Row with a lot of **OECC errors** can indicate that it has been Row Hammer **victimized**.
  - At the first OECC **uncorrectable errors** (HC_{OECC}), many OECC **correctable errors** are likely to have occurred before.



Cube 2 – Row Hammer Diagnosis with OECC Profile

- Row with a lot of **OECC errors** can indicate that it has been Row Hammer **victimized**.
  - At the first **Chipkill uncorrectable errors** ($HC_{Chipkill}$), the phenomenon is more pronounced.

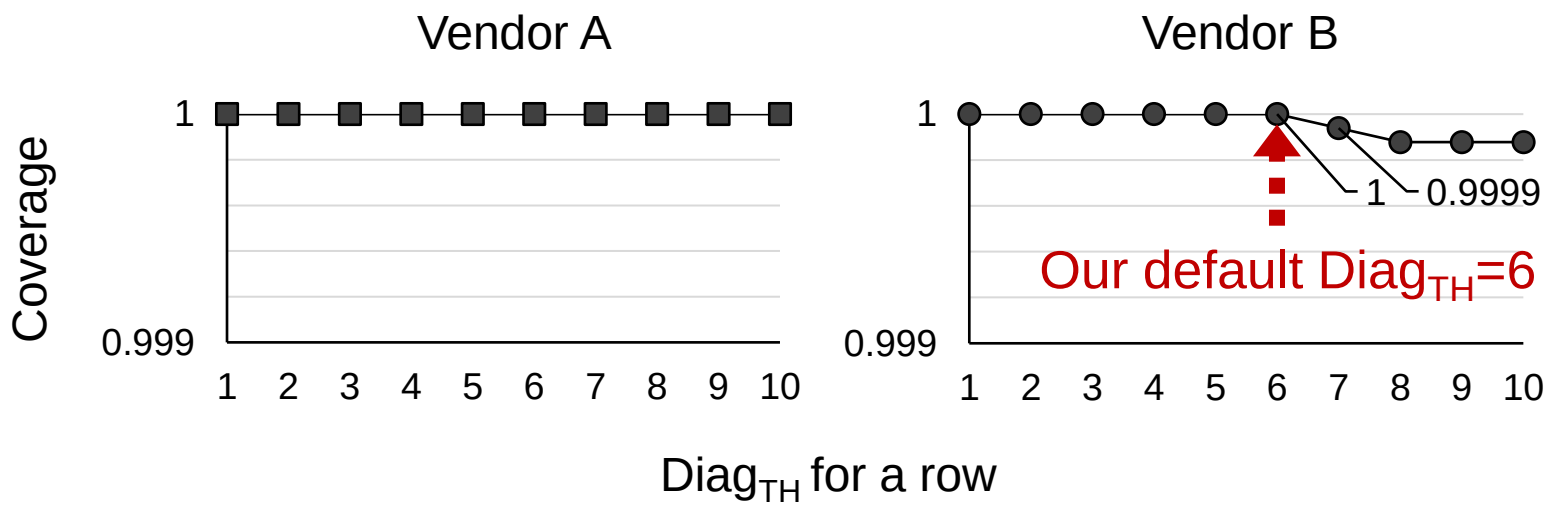


Cube 2 – Row Hammer Diagnosis with OECC Profile

- Cube diagnoses a row as Row Hammer victim, when it has more than **$Diag_{TH}$** number of OECC codeword with error.
  - When **diagnosed**, executes sequence of **Chipkill correction.**

Cube 2 – Row Hammer Diagnosis with OECC Profile

- Cube diagnoses a row as Row Hammer victim, when it has more than **$Diag_{TH}$** number of OECC codeword with error.
  - When **diagnosed**, executes sequence of **Chipkill correction.**
- **$Diag_{TH}$**
  - Too low $Diag_{TH}$ (e.g., 1) would incur too much **false positive** from non-Row Hammer errors.
  - Too high $Diag_{TH}$ (e.g., 100) would incur too much **false negative** and lose actual victims.



Cube 2 – Row Hammer Diagnosis with OECC Profile

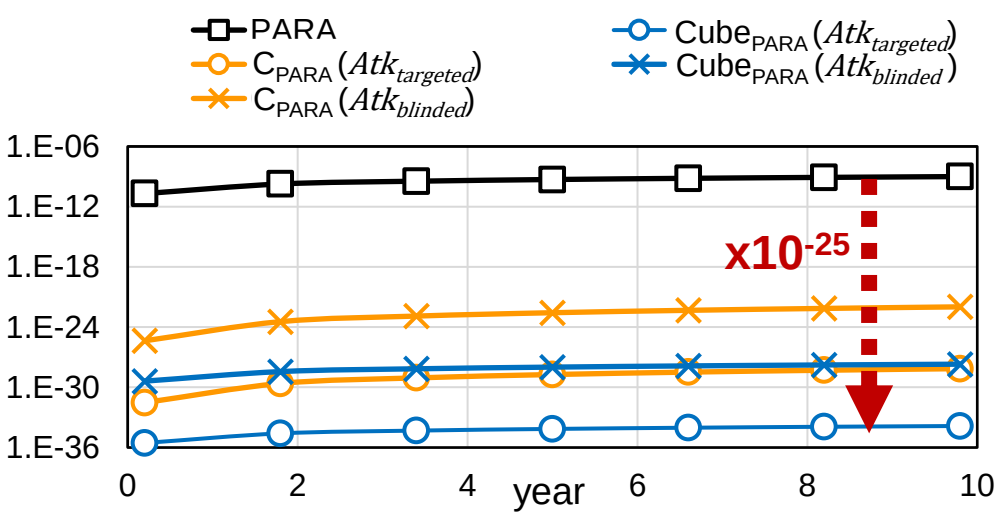
- OECC error profile can be collected from **patrol scrubbing** (e.g., ECS mode mode adopted for DDR5).

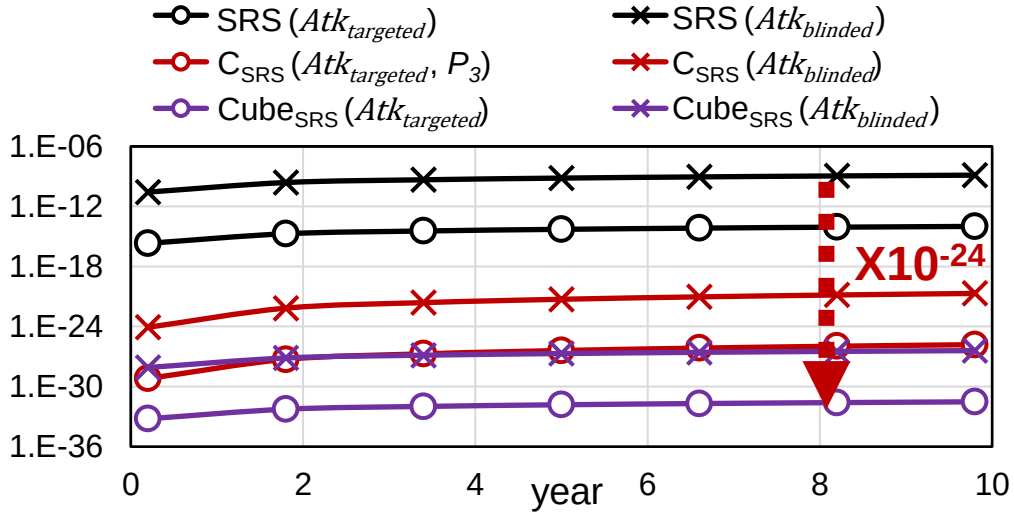
- We adopt a more optimized version of OECC scrubbing with a few additional registers.
  - Bank-group level parallelism
  - Scrub address register (17b)
  - 70B buffering register
  - Scrub all rows every 10 minutes
    - 1.37% reduced tREFI

Evaluation and Result

Cooperation with the Probabilistic Row Hammer Solutions

- Can be **co-run** with most probabilistic Row Hammer protection solutions.
  i. **Lowers the probability** of Row Hammer attack success.
  ii. **Reduces the overhead** of probabilistic Row Hammer solutions, when the target probability is fixed





Evaluation Methodology

- Pin-based cycle accurate simulator (McSimA+)
- SPEC CPU2017
  - Sampled using Simpoint
- Evaluate prior probabilistic schemes
  - PARA[1]
  - SHADOW[2]
  - SRS[3]
    - Did not considered the LLC pinning case, which is rather a deterministic solution.

Core Configurations (16 cores)	
Core	3.6 GHz 4-way O3 cores
LLC	16 MB (shared)
Memory System Configurations	
Module	DDR5-4800, 32Gb chips
Channel	2 channels
Configuration	1 rank; 32 banks per rank
Scheduling	BLISS [102]
Page-Policy	Minimalist-open [44]

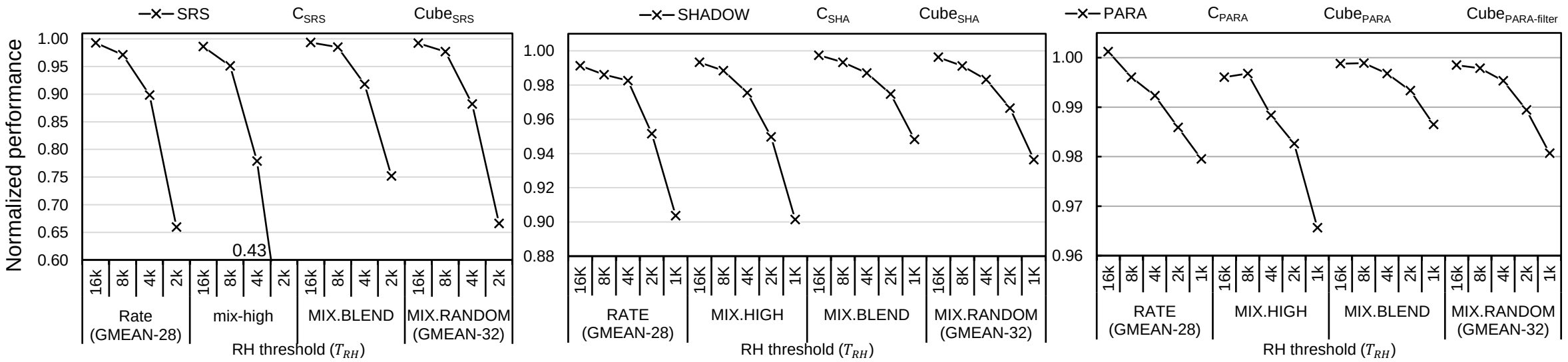
Parameters for architectural simulation

- Targets RH protection guarantee of **10^{-10} per rank per year**.
- **Conservatively** only consider **$Atk_{Blinded}$**.
  - Cube is stronger against $Atk_{Targeted}$ and most attacks belong to the latter.

[1] Yoongu Kim, Ross Daly, Jeremie Kim, Chris Fallin, Ji Hye Lee, Donghyuk Lee, Chris Wilkerson, Konrad Lai, and Onur Mutlu. 2014. Flipping Bits in Memory Without Accessing Them: An Experimental Study of DRAM Disturbance Errors.
[2] Minbok Wi, Jaehyun Park, Seoyoung Ko, Michael Jaemin Kim, Nam Sung Kim, Eojin Lee, and Jung Ho Ahn. 2023. SHADOW: preventing Row Hammer in DRAM with Intra-Subarray Row Shuffling.
[3] Gururaj Saileshwar, Bolin Wang, Moinuddin Qureshi, and Prashant J. Nair. 2022. Randomized Row-Swap: Mitigating Row Hammer by Breaking Spatial Correlation between Aggressor and Victim Rows.

Evaluation Results

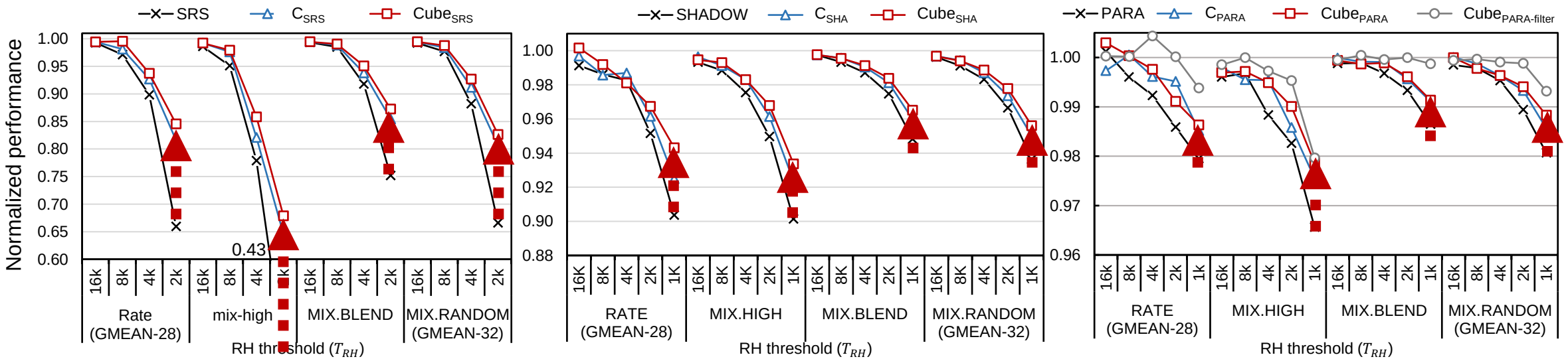
- Improves the performance degradation of all evaluated probabilistic Row Hammer protection schemes.
  - Up to 24.3% reduction.



(a) SRS, C_{SRS}, $Cube_{SRS}$ (b) SHADOW, C_{SHA}, $Cube_{SHA}$ (c) PARA, C_{PARA}, $Cube_{PARA}$, $Cube_{PARA\text{-}filter}$

Evaluation Results

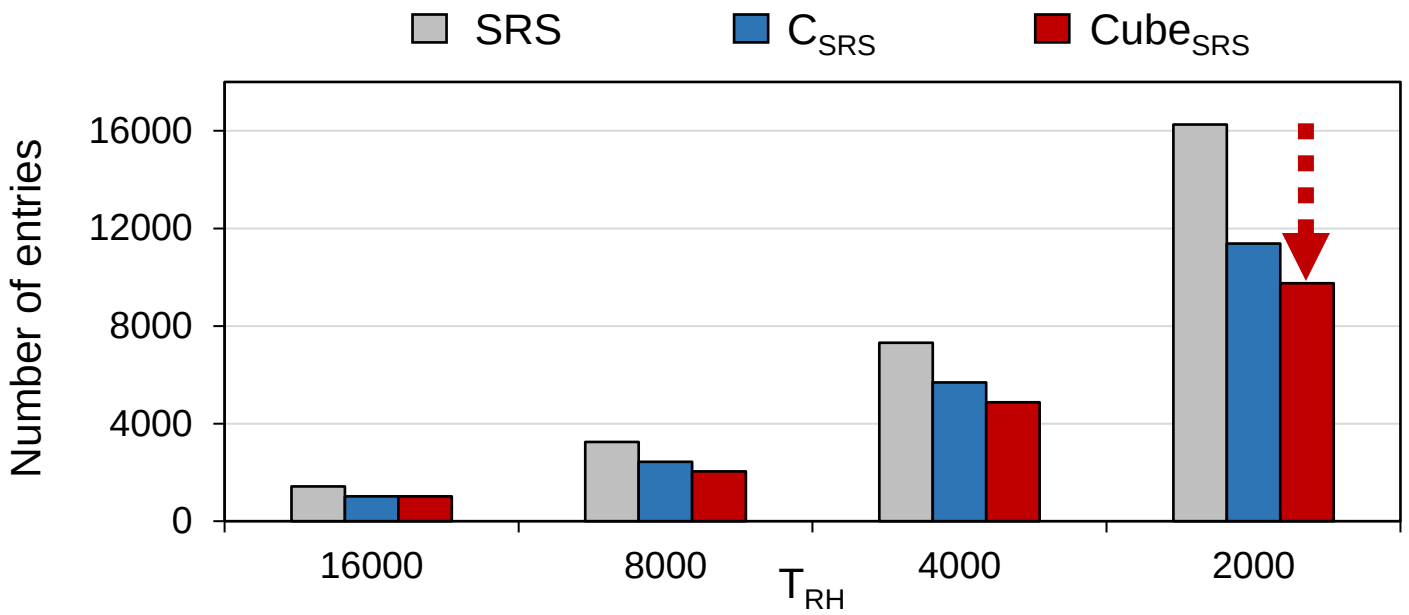
- Improves the performance degradation of all evaluated probabilistic Row Hammer protection schemes.
  - Up to 24.3% reduction.



(a) SRS, C_{SRS}, $Cube_{SRS}$ (b) SHADOW, C_{SHA}, $Cube_{SHA}$ (c) PARA, C_{PARA}, $Cube_{PARA}$, $Cube_{PARA\text{-}filter}$

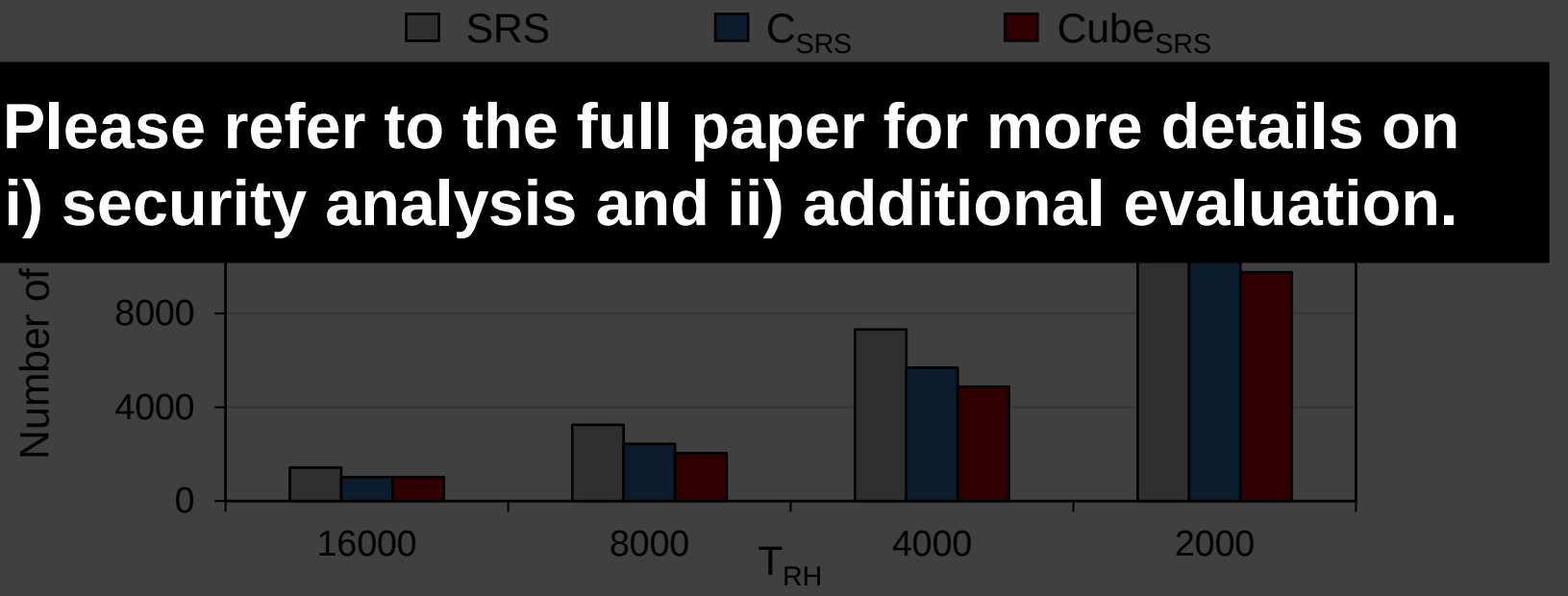
Evaluation Results

- Improves the area overhead of probabilistic scheme that requires counter table structure (SRS).
  - Up to 39.9% reduction.



Evaluation Results

- Improves the area overhead of probabilistic scheme that requires counter table structure (SRS).
  - Up to 39.9% reduction.



Please refer to the full paper for more details on i) security analysis and ii) additional evaluation.

Summary

Cube

1. Row address **scramble**
   – Spreads the Row Hammer victims across multiple Chipkill codewords

2. Victim **diagnosis**
   – Identify the Row Hammer victims and execute Chipkill correction

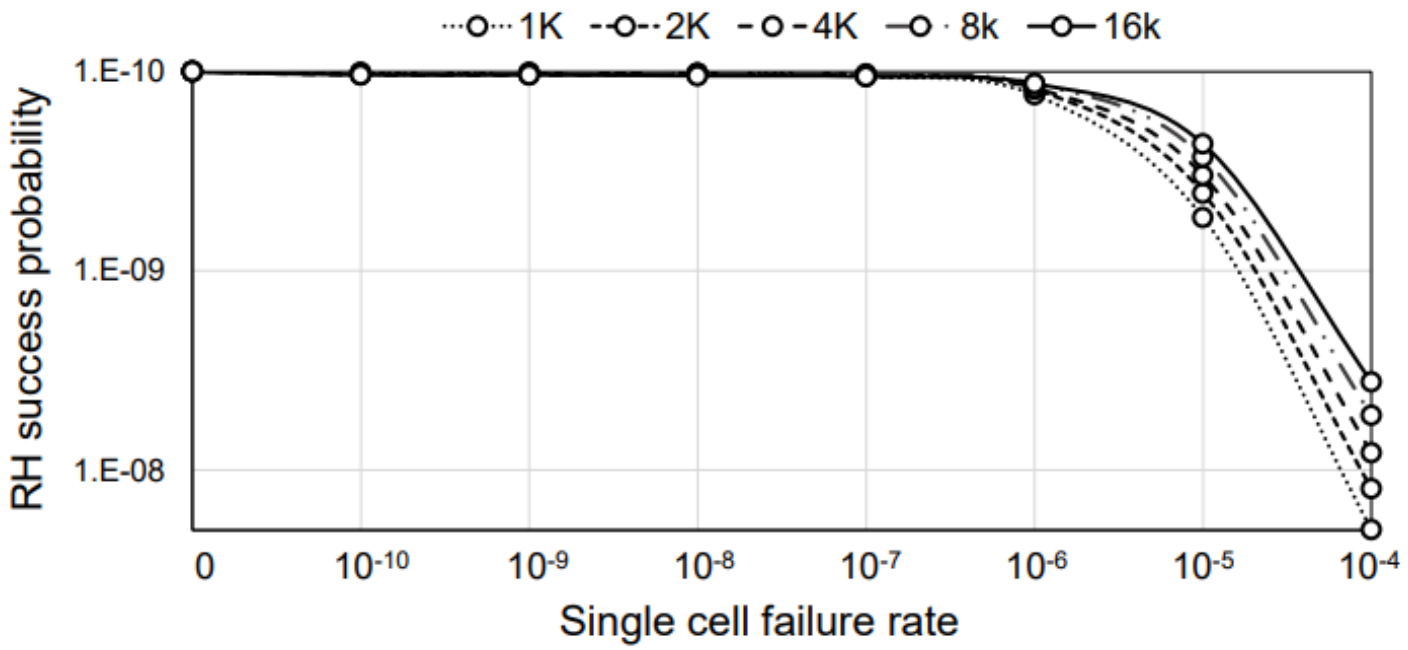
3. **Cooperation** with most existing probabilistic Row Hammer solutions
   i. Improves the **RH protection guarantee** of up to **10^{-25}**
   ii. Improves the **performance** degradation by up to **24.3%** or **area** overhead by up to **39.9%**

Thank you!

Backup: Non-Row Hammer Errors

- 1) Large granularity faults
  - Assumed to be handled by the higher-level RAS features, thus unlikely to coexist with the Row Hammer victim simultaneously.

- 2) Single cell faults



Backup: Configurations

- Fixed target RH protection guarantee of 10^{-10} per rank per year

Table 5: Cube_{SRS} and SRS k configurations, Cube_{SHA} and SHADOW RAAIMT configurations

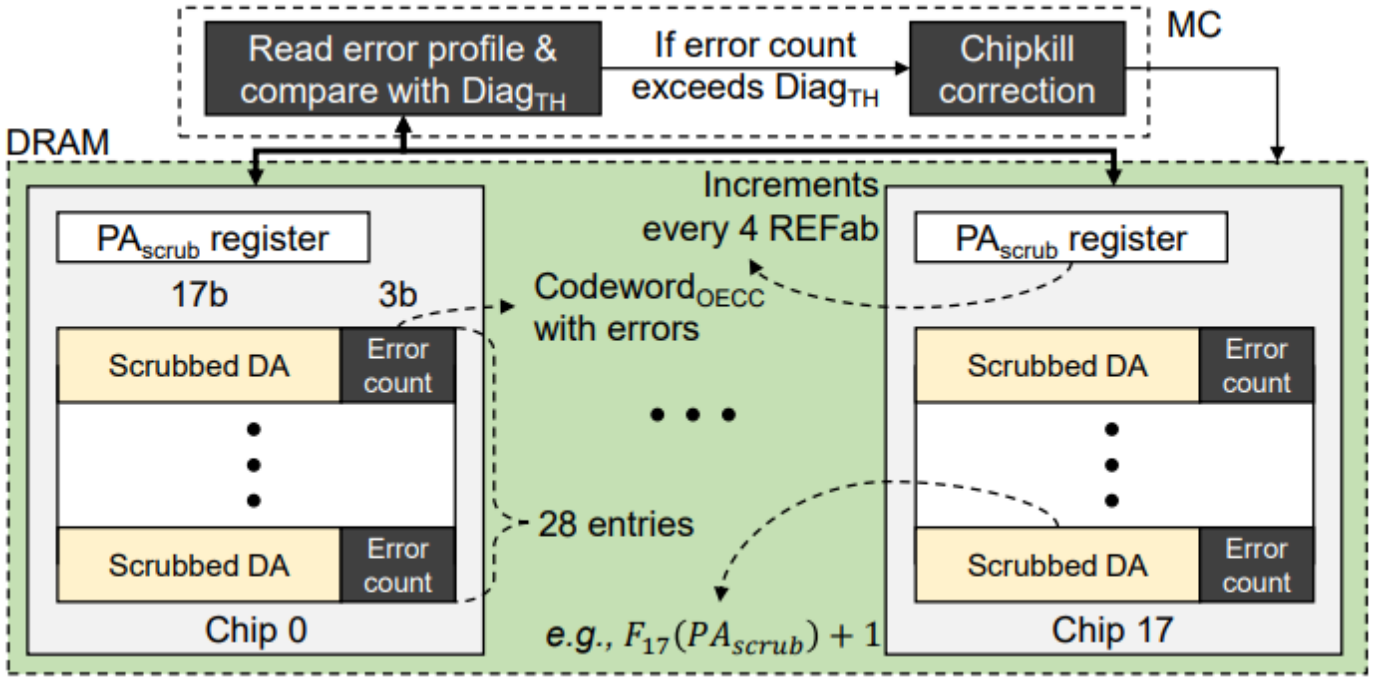
T_{RH}	SRS	C_{SRS}	Cube_{SRS}	SHADOW	C_{SHA}	Cube_{SHA}
1K	-	-	-	19	25	29
2K	10	7	6	39	51	60
4K	9	7	6	79	106	123
8K	8	6	5	161	218	255
16K	7	5	5	332	451	530

Table 6: $\text{Cube}_{\text{PARA}}$ and PARA p configurations

T_{RH}	PARA	C_{PARA}	$\text{Cube}_{\text{PARA}}$	$\text{Cube}_{\text{PARA-filter}}$
2K	0.0545	0.0388	0.0334	0.0554 & 400
4K	0.0271	0.0192	0.0164	0.0206 & 400
8K	0.0134	0.0094	0.0081	0.0092 & 500
16K	0.0067	0.0046	0.0039	0.0041 & 300

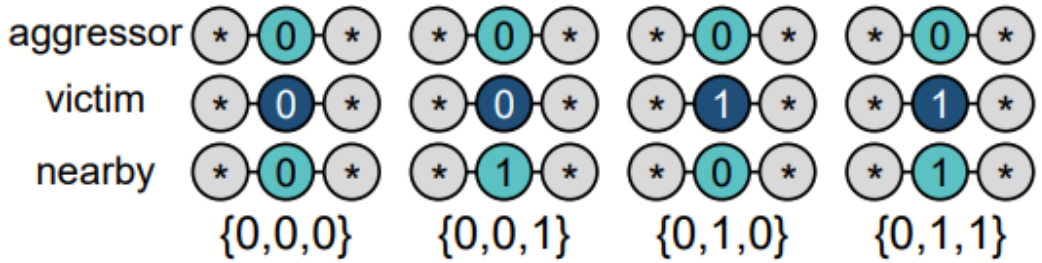
Backup: OECC Scrubbing Methodology

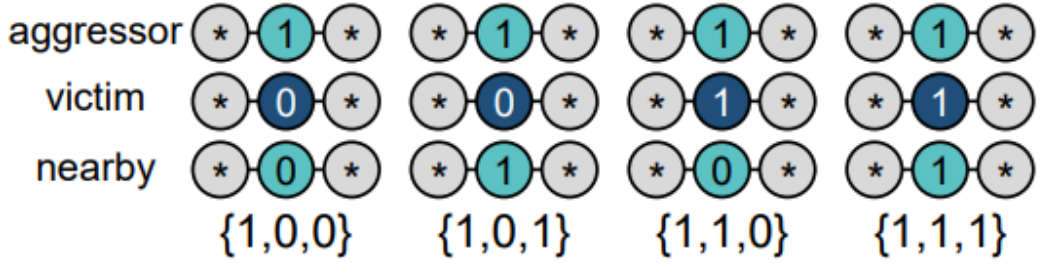
- Must be scrubbed at the **“victim set”** granularity.
  - For a given specific aggressor (PA), scan it’s victims (DA).



Backup: Experimented Data Pattern

- Only considered intra-column data patterns.
  - Did not consider the cross-column data patterns, referring to prior studies.
    - Lucian Cojocar, Kaveh Razavi, Cristiano Giuffrida, and Herbert Bos. 2019. Exploiting Correcting Codes: On the Effectiveness of ECC Memory Against Rowhammer Attacks.
    - Sangwoo Ji, Youngjoo Ko, Saeyoung Oh, and Jong Kim. 2019. Pinpoint Rowhammer: Suppressing Unwanted Bit Flips on Rowhammer Attacks.



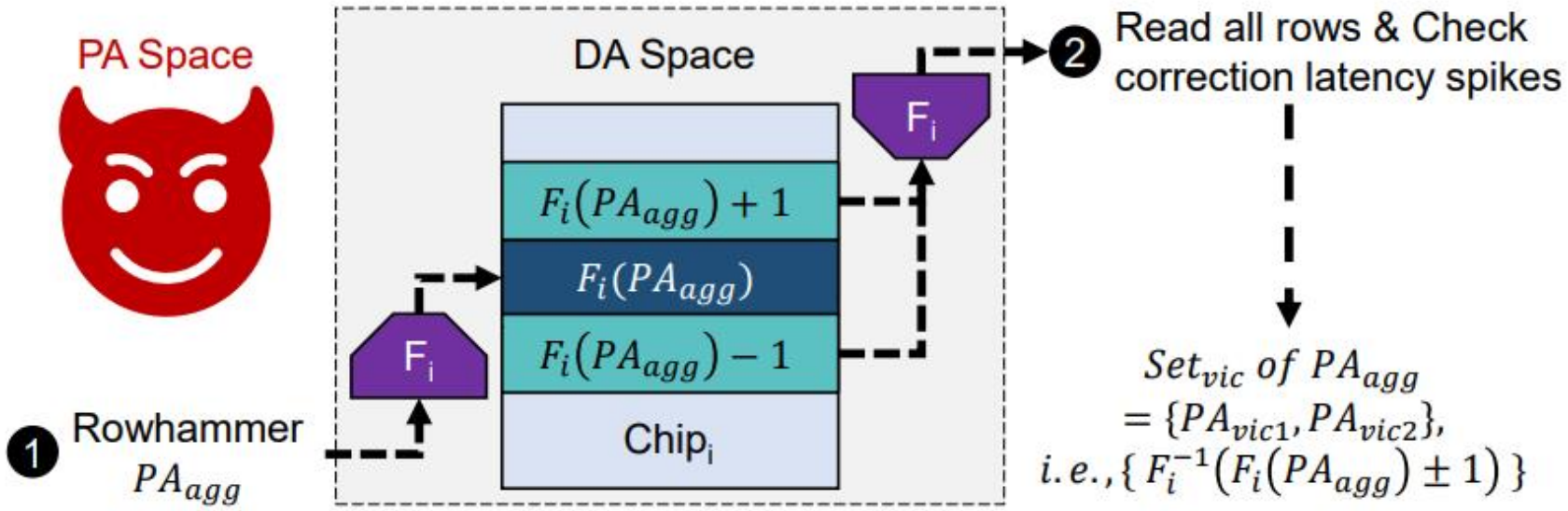


Backup: Implications of Row Scramble

- Targeted RH attacks require **templating** phase.
- Tailored data pattern attack requires **profiling** phase.

- Row scramble can actually prevent the attacker from executing such stages, by greatly limiting the possible knowledge gained by the attacker.
  - Only can get side-channel information about **the aggressor's PA, and the possible victims' PA**, even if you succeed in RH attack once, and be able to scan the whole DRAM space.
  - You cannot observe the **actual bitflip locations**, because they are going to be fixed.

Backup: Row Hammer Side-channel



Vanilla Function

$$\text{Two victim PAs} = c_i^{-1}(\text{PA}_{\text{agg}} \times c_i \pm 1)$$
$$\Rightarrow c_i^{-1}(\text{PA}_{\text{agg}} \times c_i + 1) - c_i^{-1}(\text{PA}_{\text{agg}} \times c_i - 1) = 2c_i^{-1}$$

Encrypted Function

$$\text{Enc. Scramble Function } F_i(\text{PA}) := (Enc(\text{PA}) \times c_i) \bmod N_{row},$$
$$\text{Two victim PAs} : Dec\{c_i^{-1}(Enc(\text{PA}_{\text{agg}}) \times c_i \pm 1)\}$$

Backup: LLBC Shortcut Attacks

For $Y = (Y_0 \| Y_1 | \cdots Y_n)$,

$$Y_i = F_i(X, K) = \mathcal{A}_i(X) \oplus \mathcal{B}_i(K)$$
$$= a_1 x_1 \oplus a_2 x_2 \oplus \cdots \oplus a_n x_n \oplus b_1 k_1 \oplus b_2 k_2 \oplus \cdots \oplus b_{mn} k_{mn}$$

if $\mathcal{A}_i(X_1) \oplus \mathcal{B}_i(K) = Y_i = \mathcal{A}_i(X_2) \oplus \mathcal{B}_i(K), \forall i \mid 1 \leq i \leq \lfloor log_2(L) \rfloor$

$\Rightarrow \mathcal{A}_i(X_1) = \mathcal{A}_i(X_2), \forall i \mid 1 \leq i \leq \lfloor log_2(L) \rfloor$, independent of K

Backup: LLBC Shortcut Attacks

- Redefine the shortcut attack as follows:
  - 1) Using the FPGA environment that bypasses in-DRAM TRR, find the set of of aggressor PAs (e.g., PA_{agg1} and PA_{agg2}) that hammers the same victim PA (e.g., PA_{vic}) on different chips (e.g., chip 1 and 2).

$$
\begin{aligned}
PA_{vic1} &= Dec[c_1^{-1} \cdot \{DA_{vic1}^{[1]}\}, K] \text{ for } PA_{agg1} \text{ on chip 1} \\
&= Dec[c_2^{-1} \cdot \{DA_{vic1}^{[2]}\}, K] \text{ for } PA_{agg2} \text{ on chip 2} \\
\text{where,} & \\
DA_{vic1}^{[1]} &= [c_1 \cdot \{Enc(PA_{agg1}, K)\} + 1] \ mod(2^{17}) \\
&= [c_1 \cdot \{F_1(PA_{agg1}, K) \| \cdots \| F_{17}(PA_{agg1}, K)\} + 1] \ mod(2^{17}) \\
&= [c_1 \cdot \{\mathcal{A}_1(PA_{agg1}) \oplus \mathcal{B}_1(K) \| \cdots\} + 1] \ mod(2^{17}) \\
DA_{vic1}^{[2]} &= [c_2 \cdot \{Enc(PA_{agg2}, K)\} + 1] \ mod(2^{17}) \\
&= [c_2 \cdot \{F_1(PA_{agg2}, K) \| \cdots \| F_{17}(PA_{agg2}, K)\} + 1] \ mod(2^{17}) \\
&= [c_2 \cdot \{\mathcal{A}_1(PA_{agg2}) \oplus \mathcal{B}_1(K) \| \cdots\} + 1] \ mod(2^{17})
\end{aligned}
$$

  - 2) The attacker can try to reuse the discovered set of aggressor PAs on a server environment with a new LLBC key (e.g., K').
- **Multiplication** and **Addition (from victim)**