# Aegis: Partitioning Data Block for Efficient Recovery of Stuck-at-Faults in Phase Change Memory

Jie Fan, Jiwu Shu,
Youhui Zhang, and Weimin Zheng

Song Jiang

清華大學
Tsinghua University

WAYNE STATE
UNIVERSITY

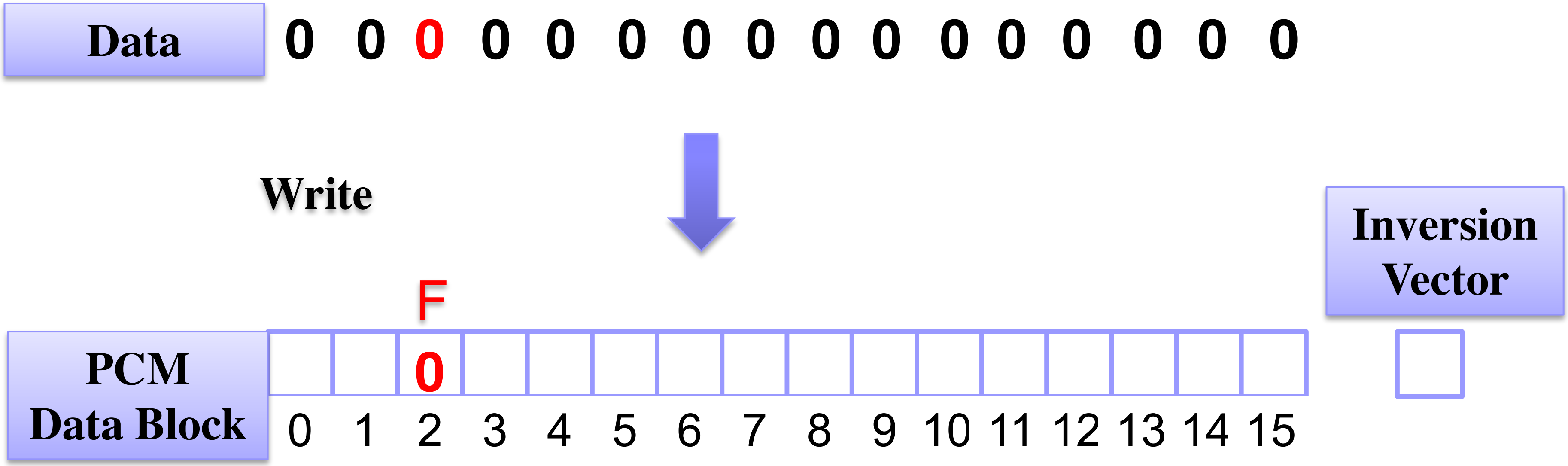
# Stuck-at Faults in PCM

- PCM has limited endurance.

- Stuck-at fault occurs when memory cell fails to change its value.
  - It is a major type of errors in PCM.
  - Values in such faulty cells can still be read.
  - The faults are permanent and accumulate.

- Two general error correction approaches at the chip level.
  - Pointer-based correction: Record the address of each faulty bit and its replacement bit (e.g., ECP).
  - Inversion-based correction: Partition data block into a number of groups and exploit the fact that stuck-at values are still readable (e.g., SAFER).

# Inversion-based Correction (Stuck-at-Right)



Data 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0

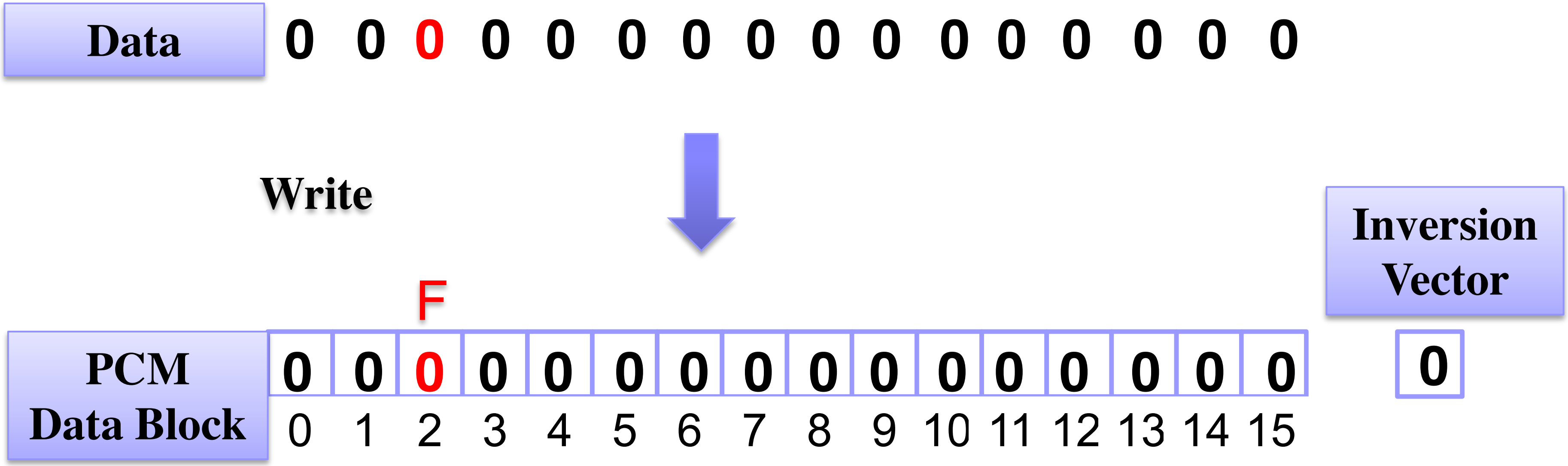
Write

Inversion Vector

PCM Data Block

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
|  |  | F |  |  |  |  |  |  |  |  |  |  |  |  |  |
|  |  | 0 |  |  |  |  |  |  |  |  |  |  |  |  |  |
| 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 | 12 | 13 | 14 | 15 |

# Inversion-based Correction (Stuck-at-Right)



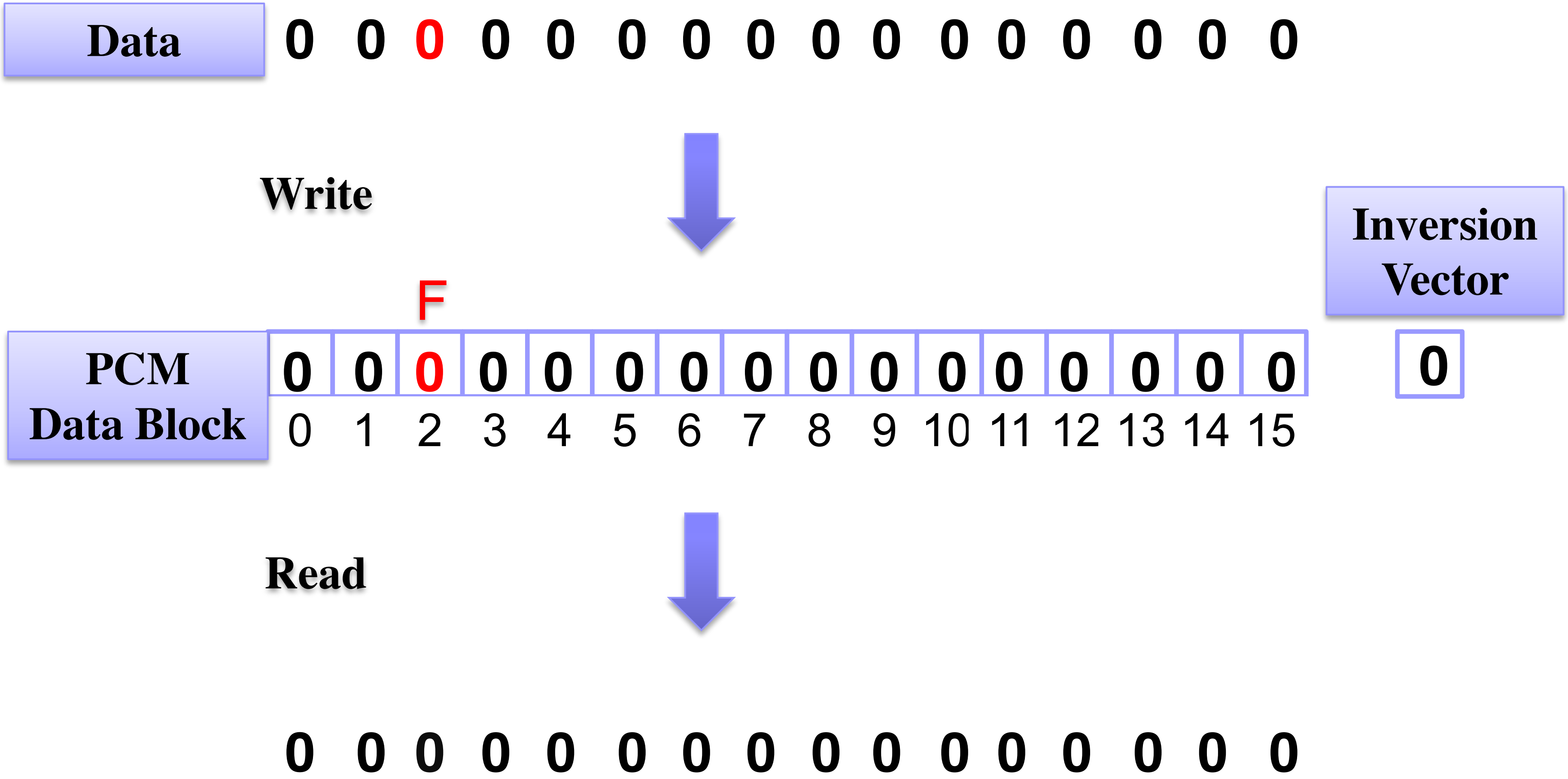
| | | | | | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Data | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| | | | F | | | | | | | | | | | | | |
| PCM Data Block | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 | 12 | 13 | 14 | 15 |

Write

Inversion Vector

0

# Inversion-based Correction (Stuck-at-Right)



Data 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0

Write

Inversion Vector

0

F

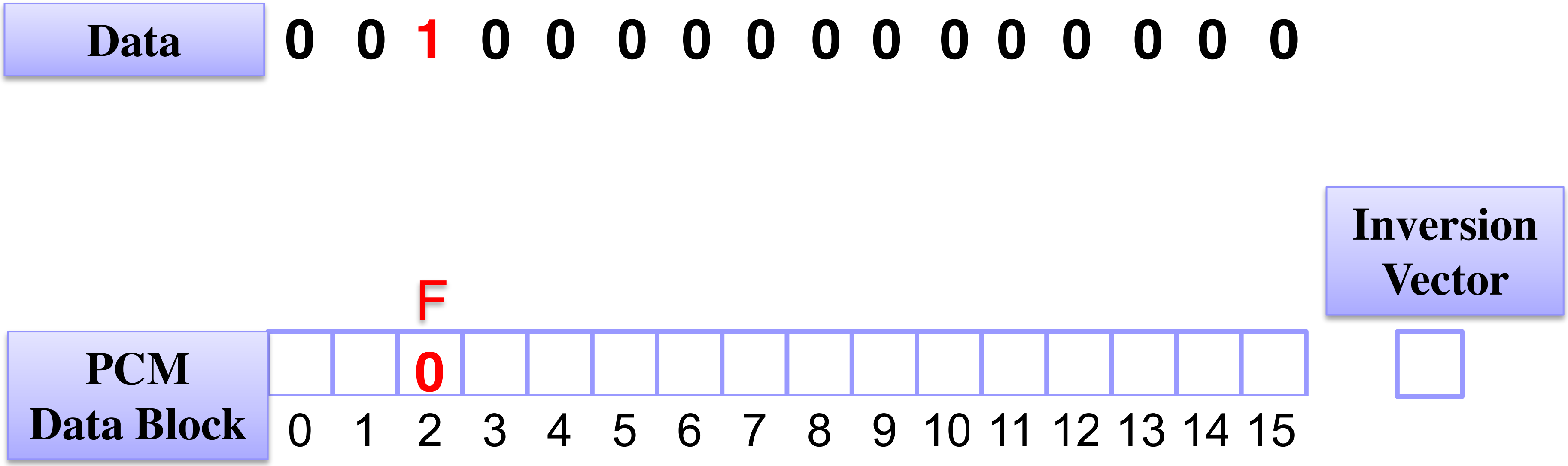
PCM Data Block

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 | 12 | 13 | 14 | 15 |

Read

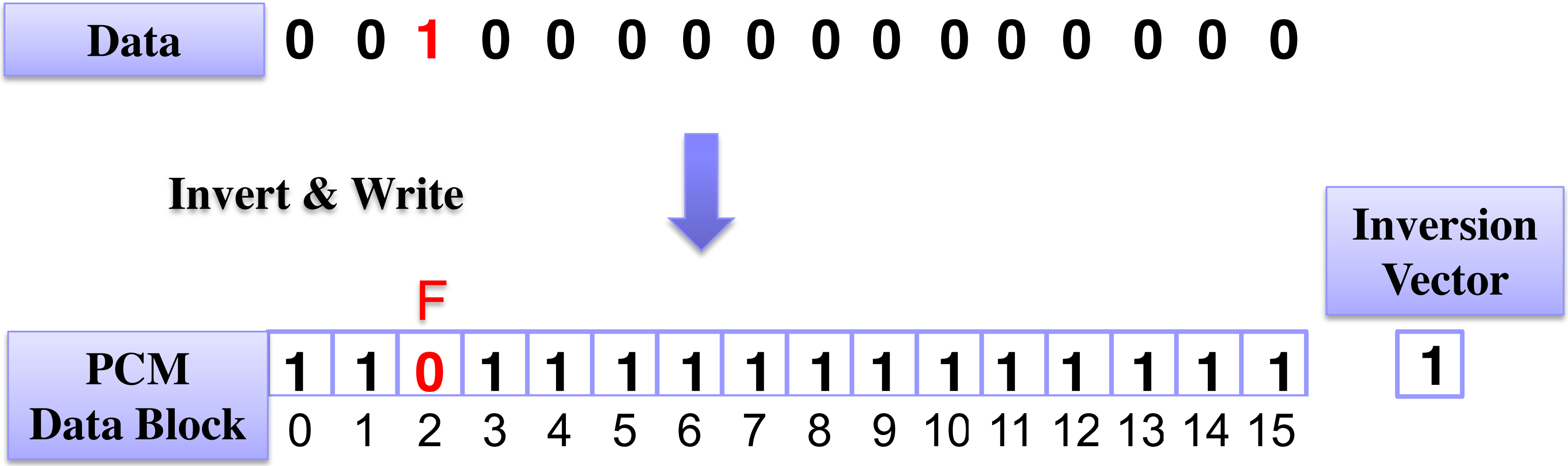
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0

# Inversion-based Correction (Stuck-at-Wrong)



| | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 | 12 | 13 | 14 | 15 | Inversion Vector |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Data | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | |
| | | | F | | | | | | | | | | | | | | |
| PCM Data Block | | | 0 | | | | | | | | | | | | | | |

# Inversion-based Correction (Stuck-at-Wrong)



| Data | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|

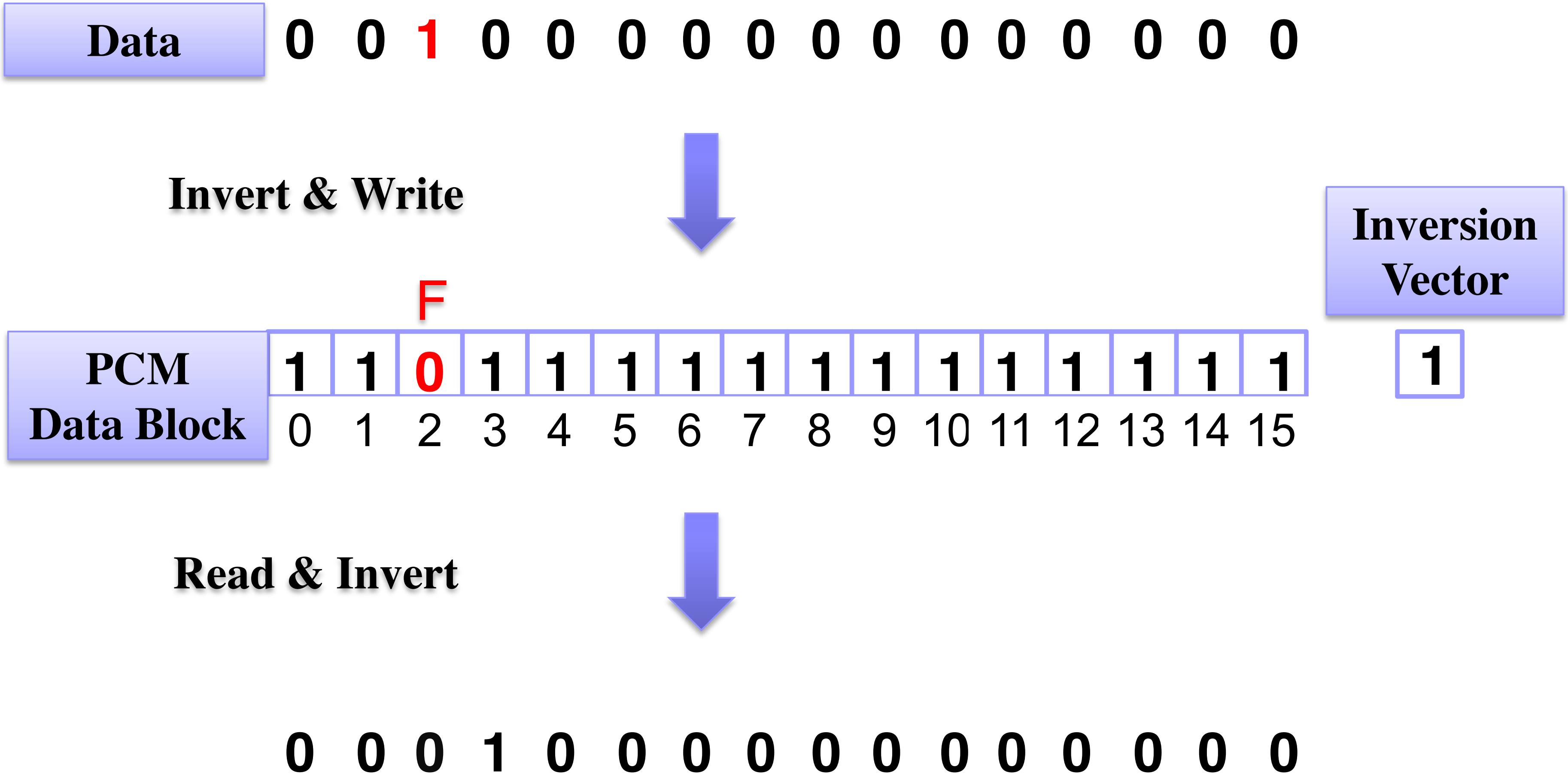
Invert & Write

Inversion Vector

1

|  |  |  | F |  |  |  |  |  |  |  |  |  |  |  |  |  |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| PCM Data Block | 1 | 1 | 0 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 |
|  | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 | 12 | 13 | 14 | 15 |

# Inversion-based Correction (Stuck-at-Wrong)



| Data | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|

Invert & Write

Inversion Vector

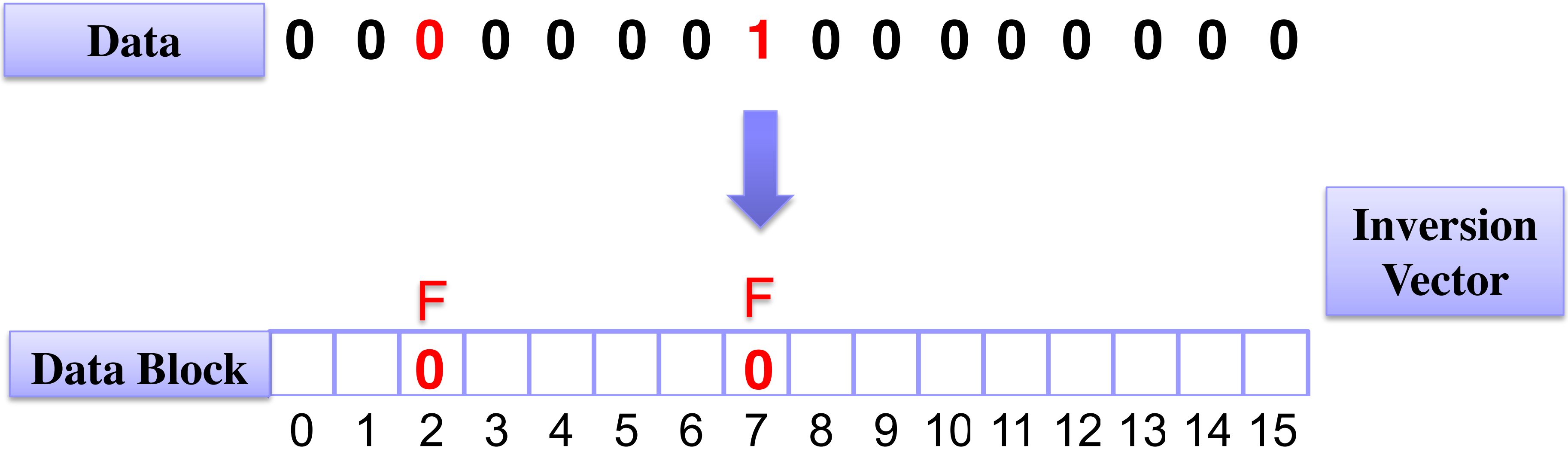
| | | | F | | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| PCM Data Block | 1 | 1 | 0 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 |
| | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 | 12 | 13 | 14 | 15 |

| Inversion Vector |
|---|
| 1 |

Read & Invert

0 0 0 1 0 0 0 0 0 0 0 0 0 0 0 0

# Inversion-based Correction: Two Faults

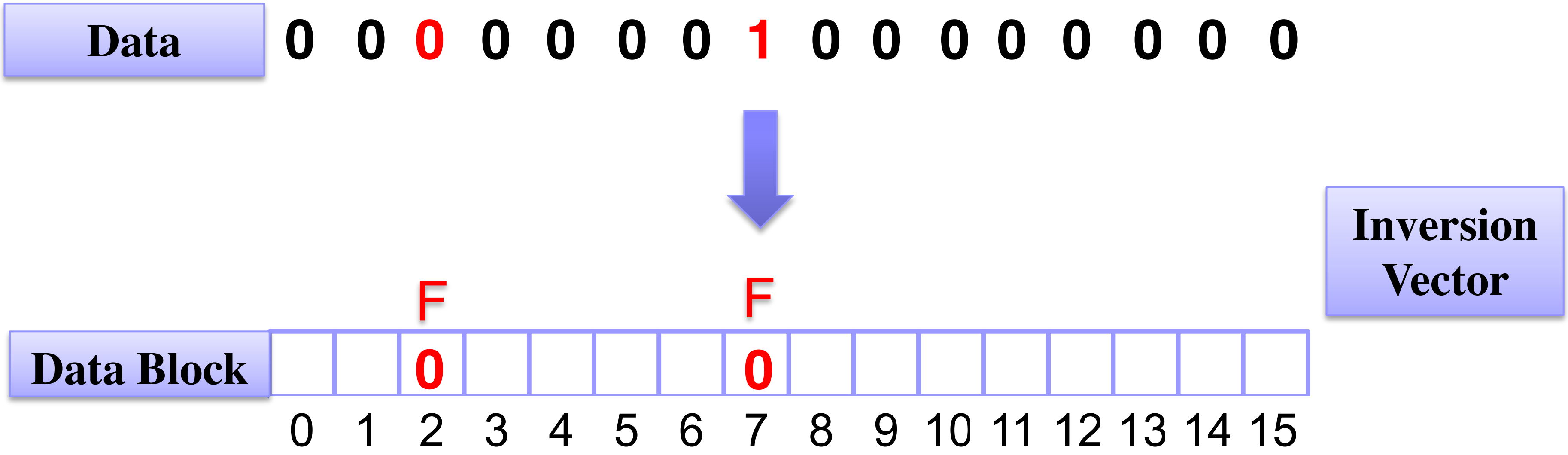


Data 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0

Inversion Vector

| | | | | | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | | | F | | | | | F | | | | | | | | |
| Data Block | | | 0 | | | | | 0 | | | | | | | | |
| | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 | 12 | 13 | 14 | 15 |

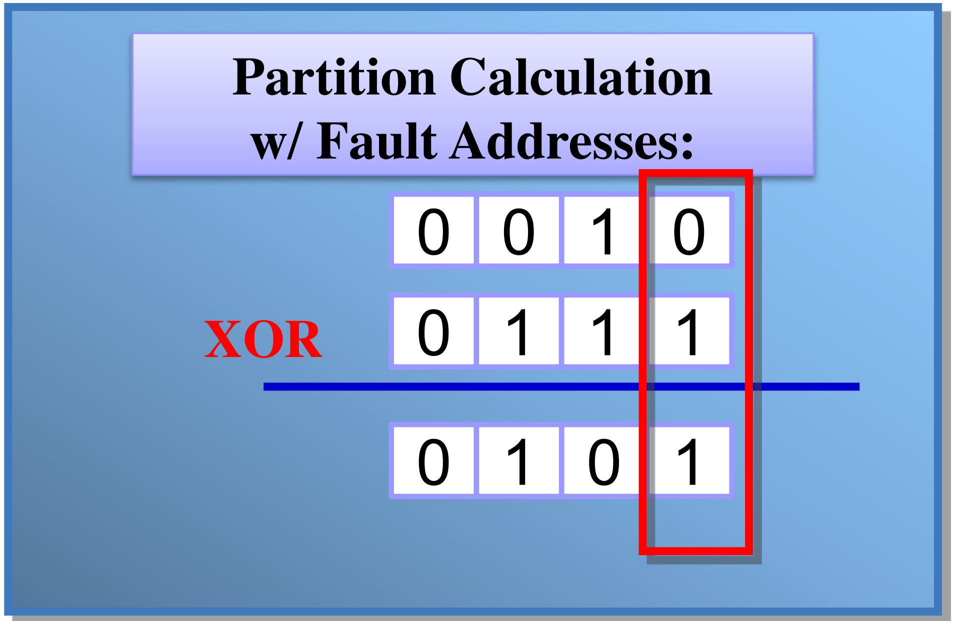
# Inversion-based Correction: Two Faults



Data 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0

Inversion Vector

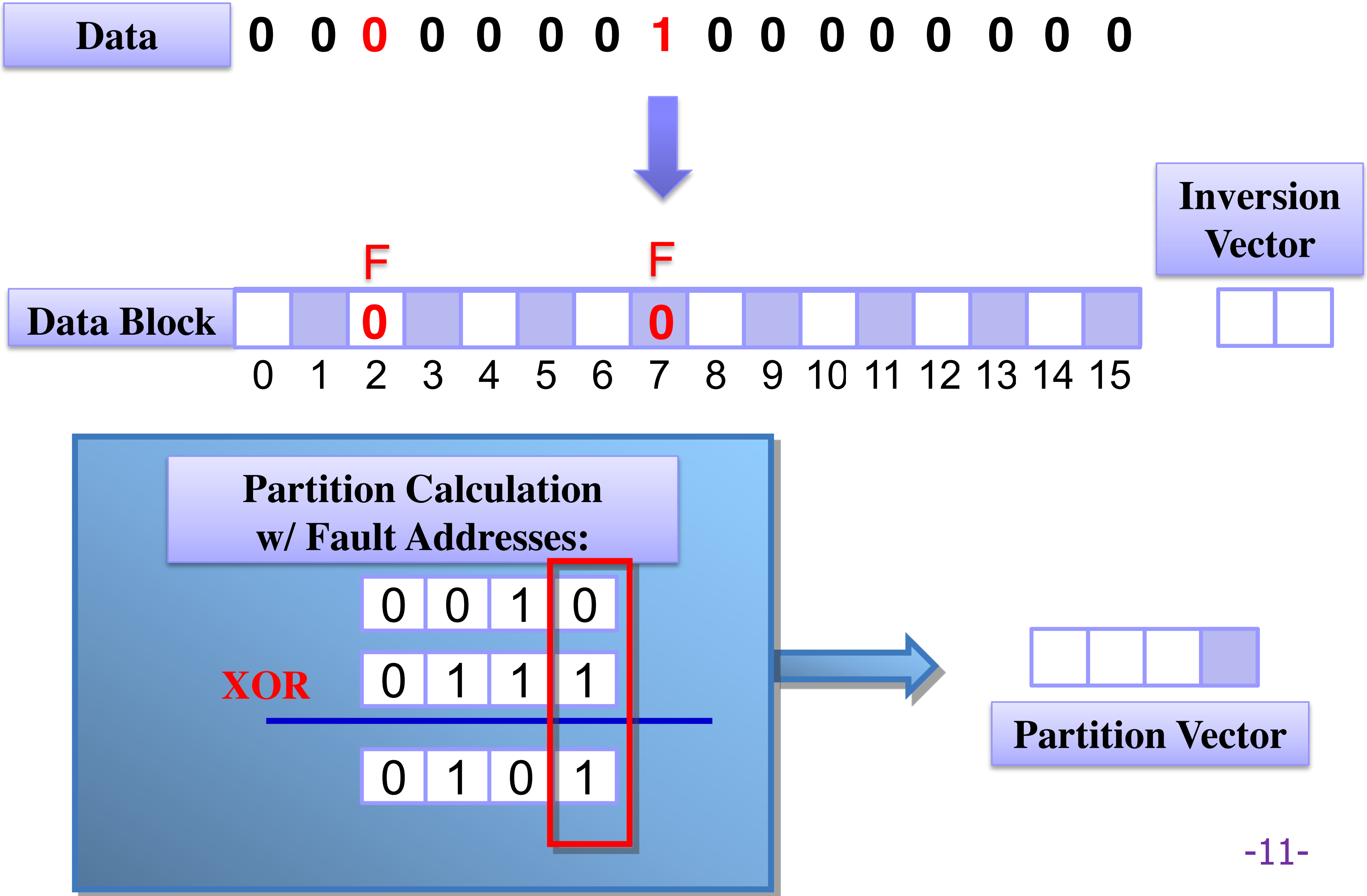
|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
|  |  |  | F |  |  |  |  | F |  |  |  |  |  |  |  |  |
| Data Block |  |  | 0 |  |  |  |  | 0 |  |  |  |  |  |  |  |  |
|  | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 | 12 | 13 | 14 | 15 |



Partition Calculation w/ Fault Addresses:

|  |  |  |  |  |
|---|---|---|---|---|
|  | 0 | 0 | 1 | 0 |
| XOR | 0 | 1 | 1 | 1 |
|  | 0 | 1 | 0 | 1 |

# Inversion-based Correction: Two Faults



Data 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0

Inversion Vector

Data Block

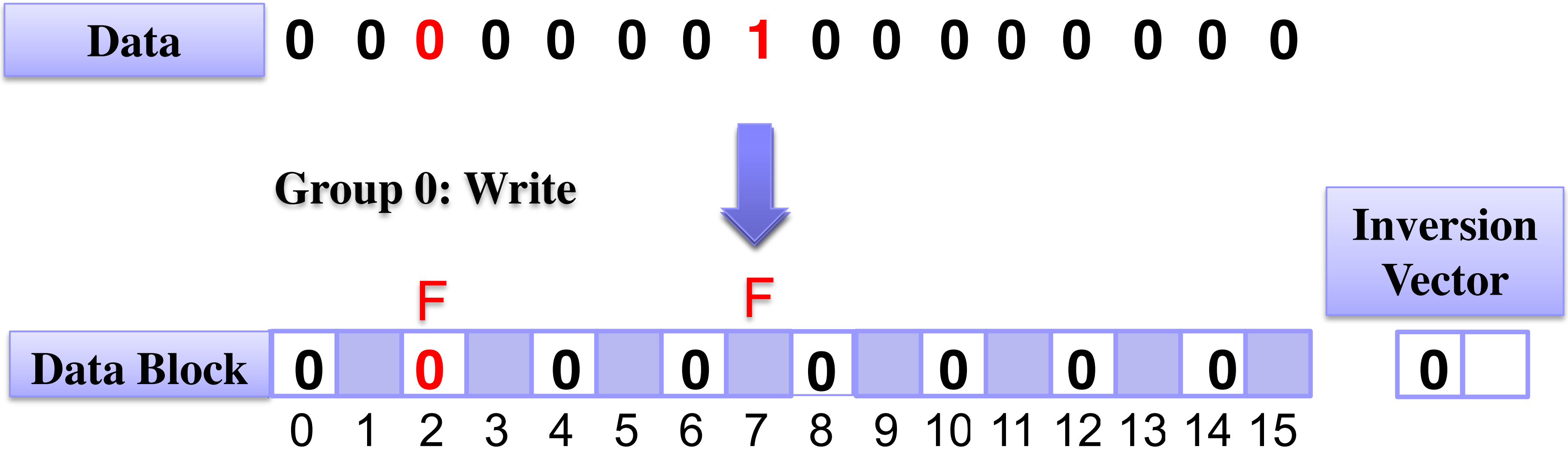
| 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 | 12 | 13 | 14 | 15 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
|  |  | 0 |  |  |  |  | 0 |  |  |  |  |  |  |  |  |

Partition Calculation w/ Fault Addresses:

```
    0 0 1 0
XOR 0 1 1 1
    -------
    0 1 0 1
```

Partition Vector

# Inversion-based Correction: Two Faults



Data 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0

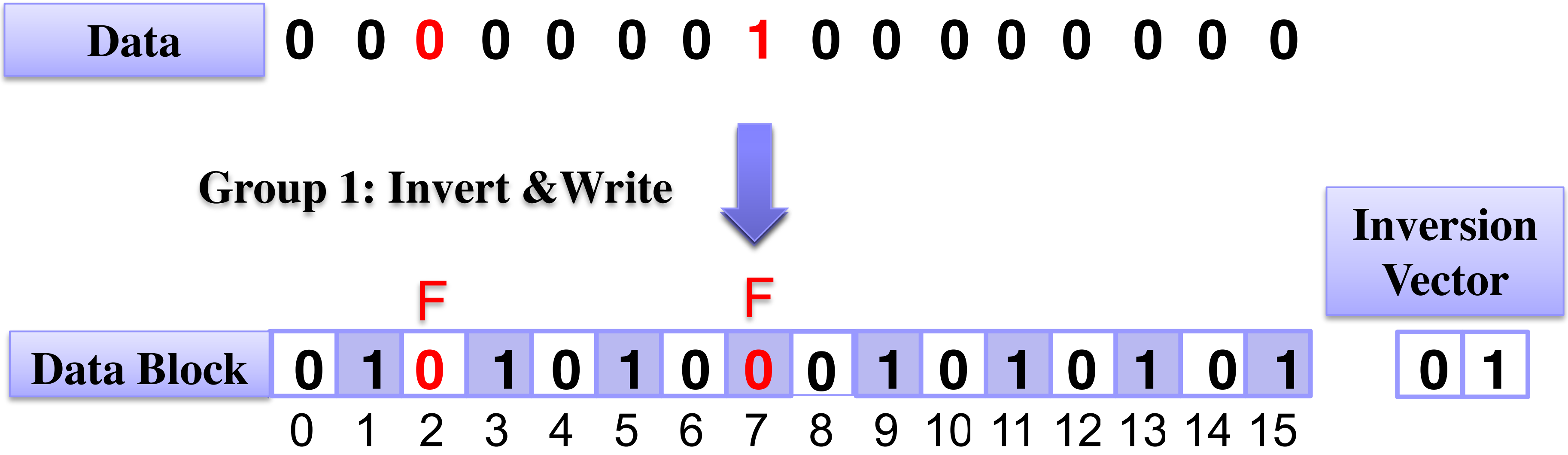
Group 0: Write

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
|  |  |  | F |  |  |  |  | F |  |  |  |  |  |  |  |  |
| Data Block | 0 |  | 0 |  | 0 |  | 0 |  | 0 |  | 0 |  | 0 |  | 0 |  |
|  | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 | 12 | 13 | 14 | 15 |

Inversion Vector

| 0 |  |
|---|---|

# Inversion-based Correction: Two Faults



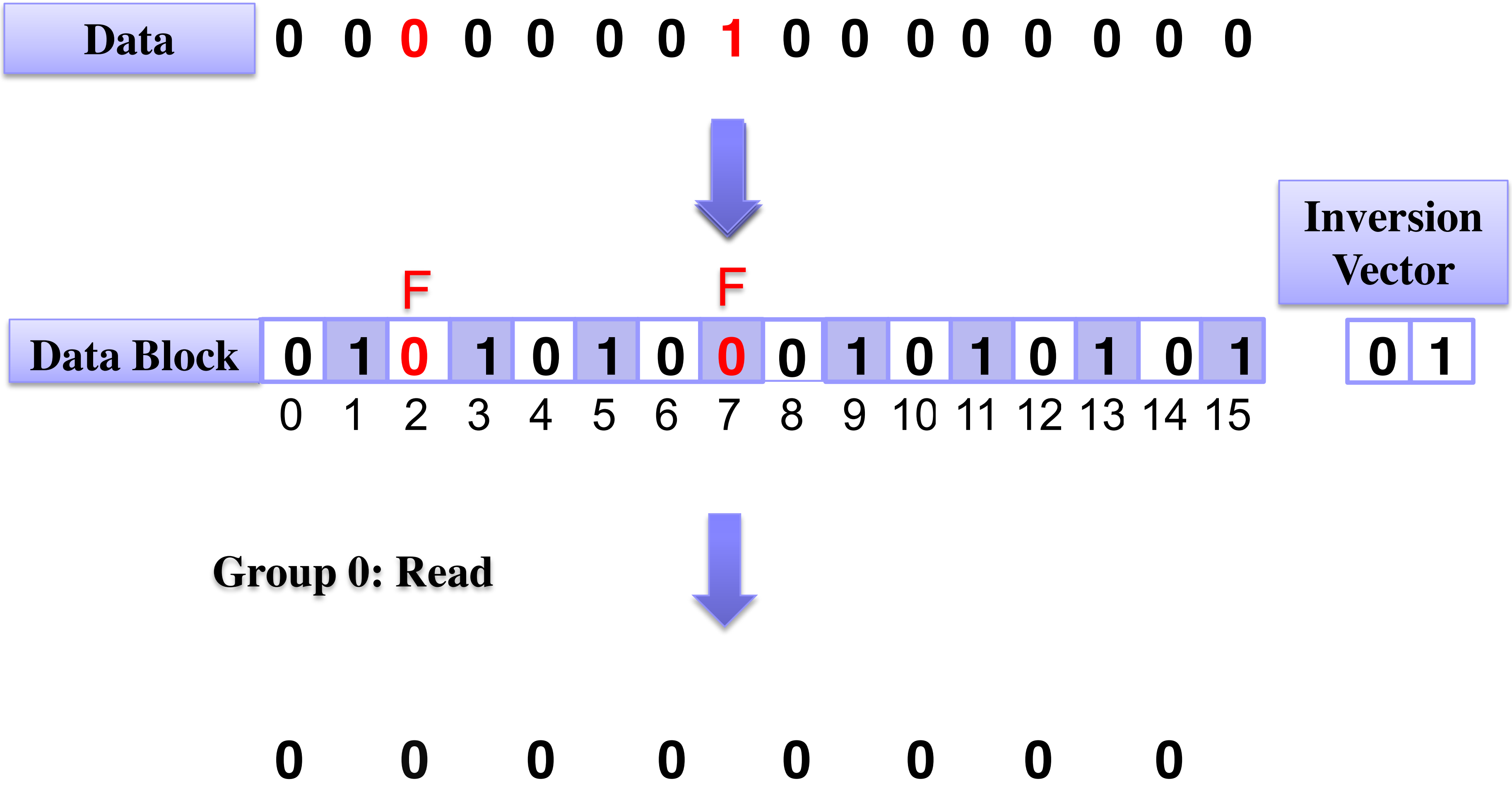
| Data | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|

Group 1: Invert &Write

|  |  |  | F |  |  |  |  | F |  |  |  |  |  |  |  |  |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Data Block | 0 | 1 | 0 | 1 | 0 | 1 | 0 | 0 | 0 | 1 | 0 | 1 | 0 | 1 | 0 | 1 |
|  | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 | 12 | 13 | 14 | 15 |

| Inversion Vector |  |
|---|---|
| 0 | 1 |

# Inversion-based Correction: Two Faults



Data 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0

Inversion Vector

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
|  |  |  | F |  |  |  |  | F |  |  |  |  |  |  |  |  |
| Data Block | 0 | 1 | 0 | 1 | 0 | 1 | 0 | 0 | 0 | 1 | 0 | 1 | 0 | 1 | 0 | 1 |
|  | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 | 12 | 13 | 14 | 15 |

0 1

Group 0: Read

0 0 0 0 0 0 0 0

# Inversion-based Correction: Two Faults

| Data | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|

Inversion Vector

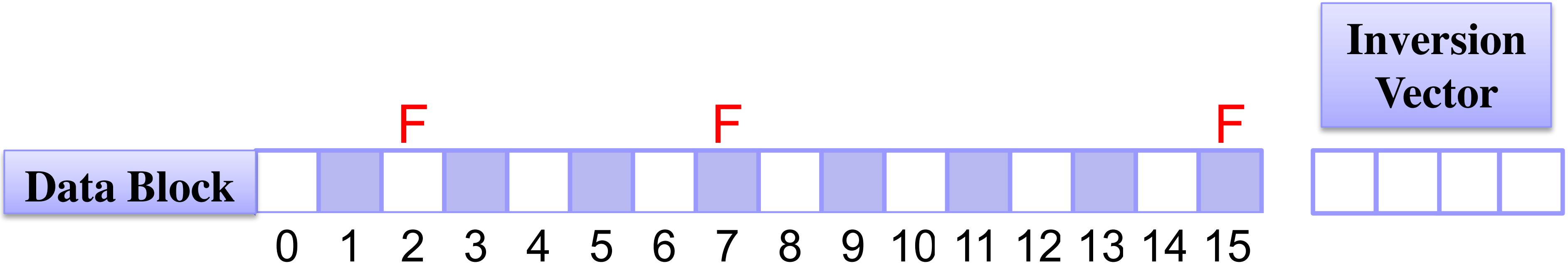
|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
|  |  |  | F |  |  |  |  | F |  |  |  |  |  |  |  |  |
| Data Block | 0 | 1 | 0 | 1 | 0 | 1 | 0 | 0 | 0 | 1 | 0 | 1 | 0 | 1 | 0 | 1 |
|  | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 | 12 | 13 | 14 | 15 |

| 0 | 1 |
|---|---|

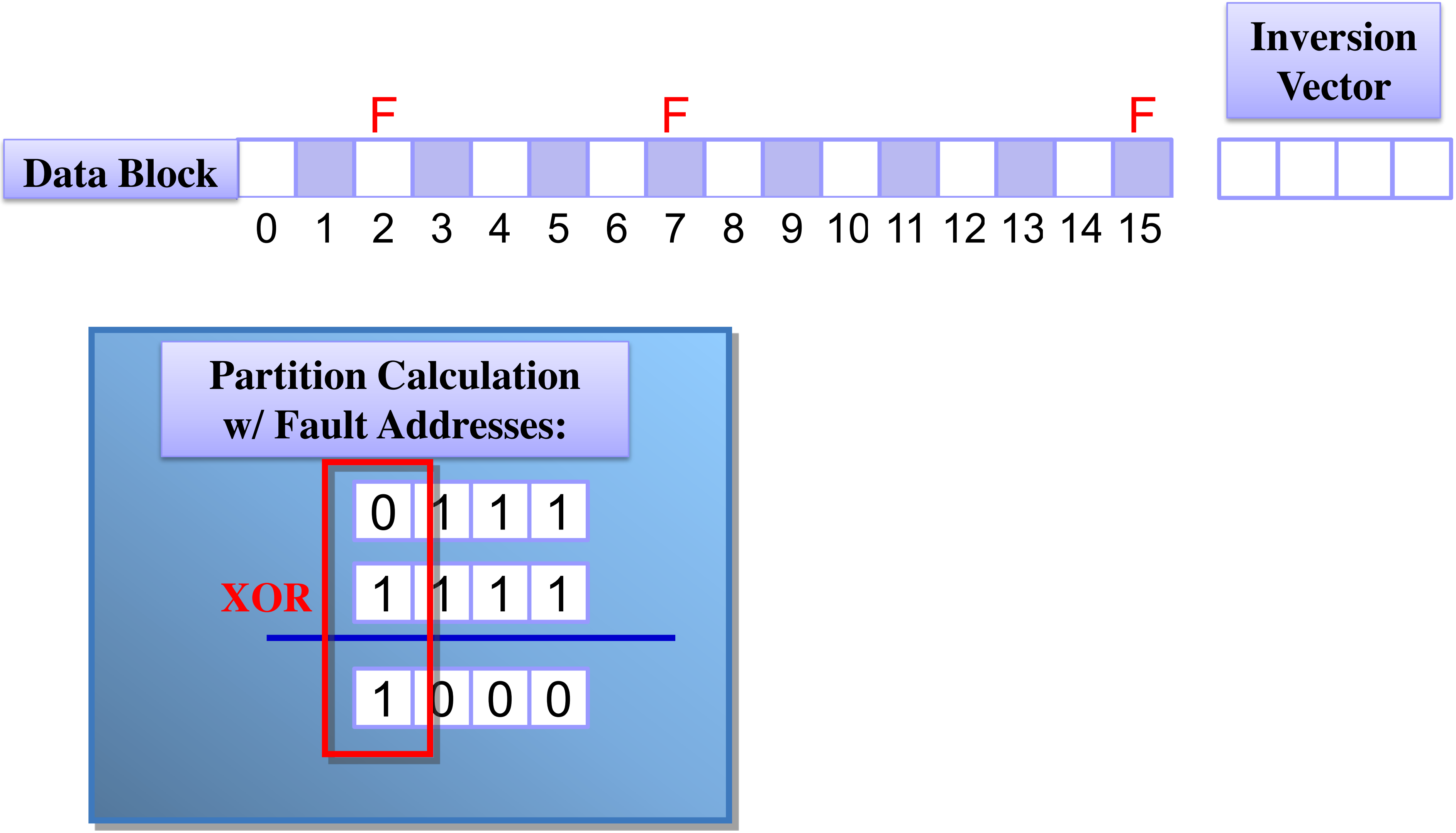
Group 1: Read & Invert

0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0

# Inversion-based Correction: the Third Fault



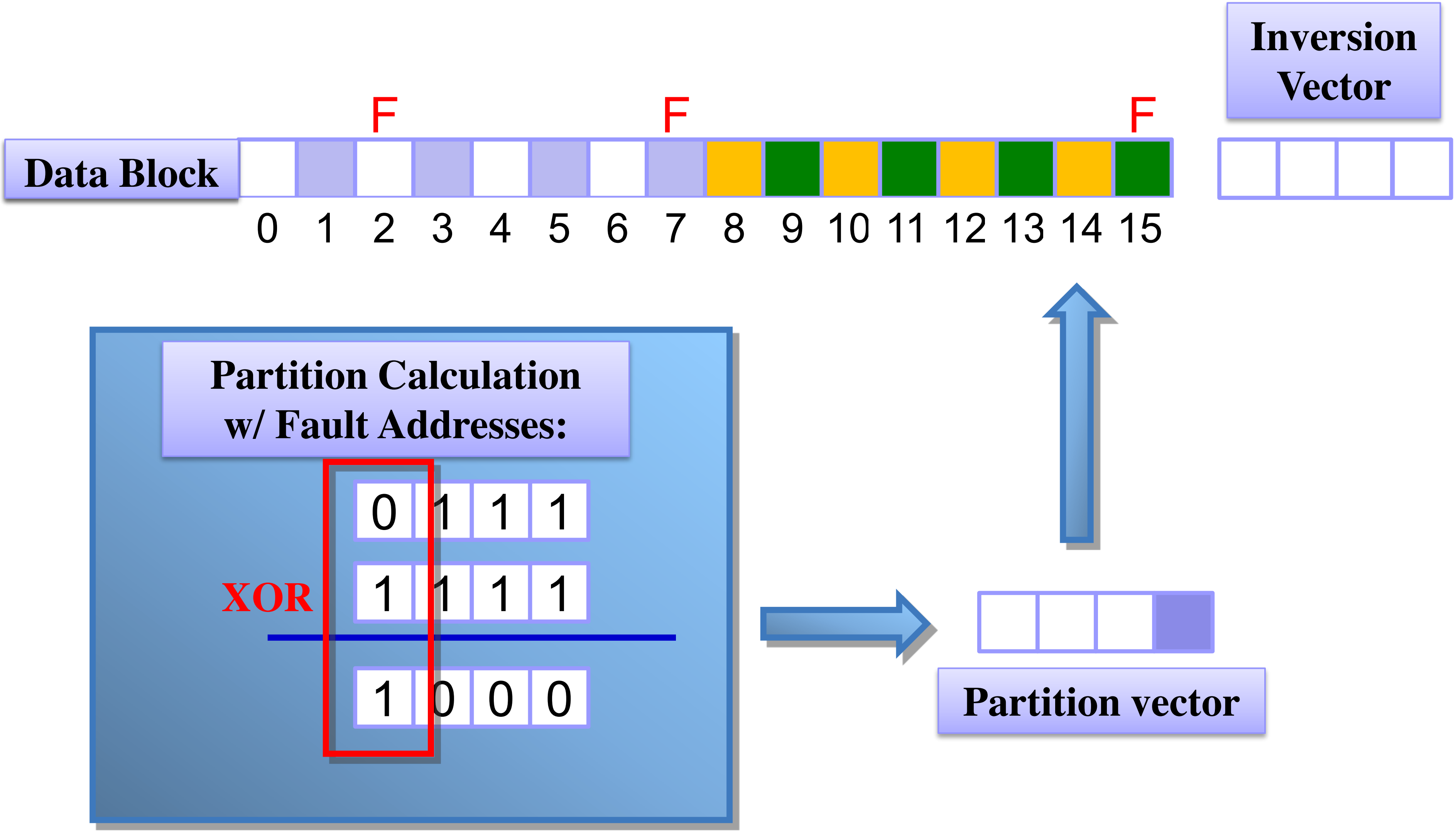
# Inversion-based Correction: the Third Fault



**Partition Calculation w/ Fault Addresses:**

```
      0 1 1 1
XOR   1 1 1 1
      -------
      1 0 0 0
```

# Inversion-based Correction: the Third Fault



Inversion Vector

Data Block

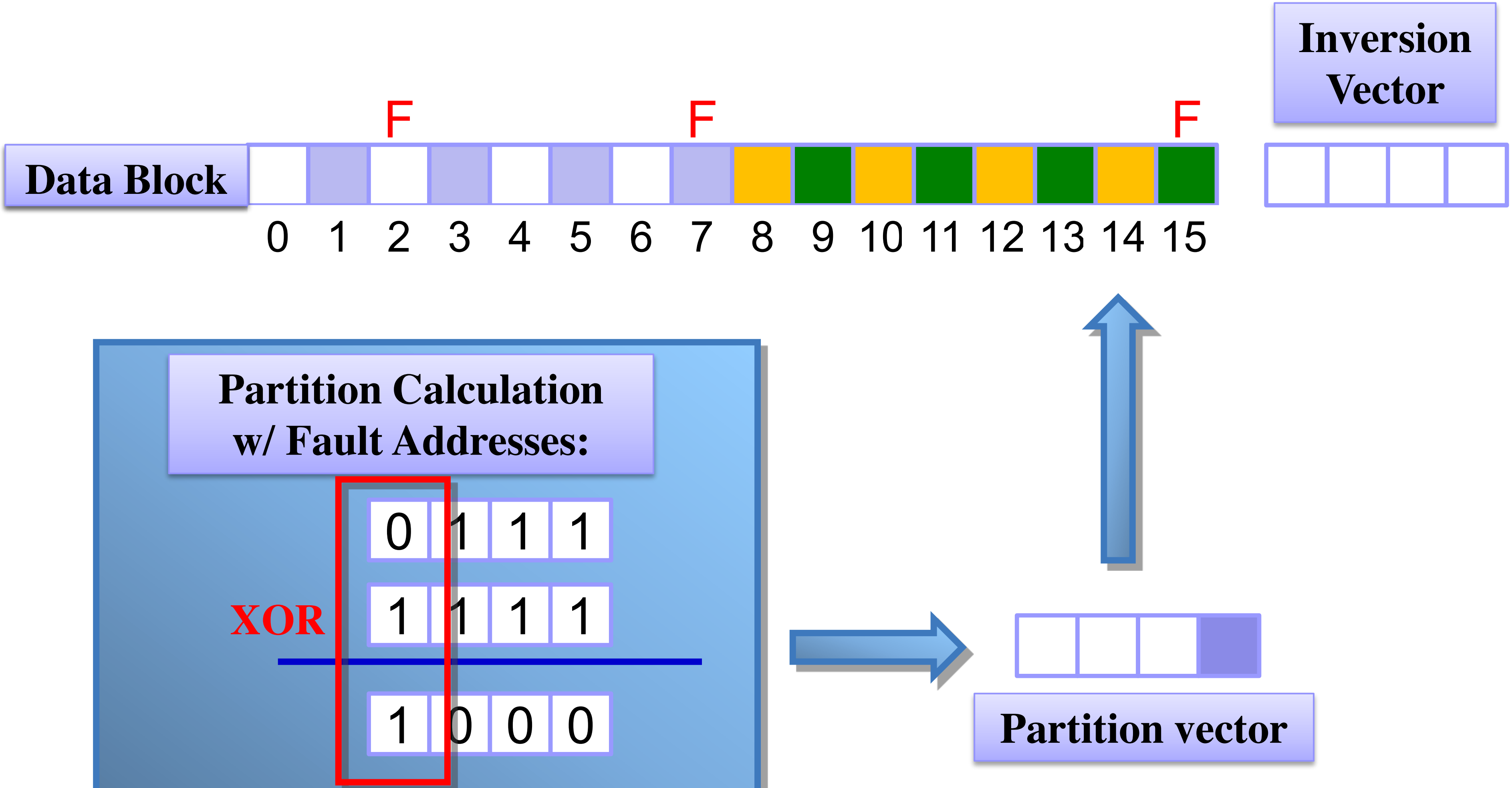
| | | | | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | | F | | | | | F | | | | | | | | F |
| 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 | 12 | 13 | 14 | 15 |

Partition Calculation w/ Fault Addresses:

| | | | | |
|---|---|---|---|---|
| | 0 | 1 | 1 | 1 |
| XOR | 1 | 1 | 1 | 1 |
| | 1 | 0 | 0 | 0 |

Partition vector

# Inversion-based Correction: the Third Fault



Inversion Vector

F F F

Data Block

0 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15

Partition Calculation w/ Fault Addresses:

| | | | | |
|---|---|---|---|---|
| | 0 | 1 | 1 | 1 |
| XOR | 1 | 1 | 1 | 1 |
| | 1 | 0 | 0 | 0 |

Partition vector

**In the worst scenario, with only five faults the block cannot be furthered partitioned!**

# Issues with the State-of-the-art Partition Scheme

❑ For a given data block of $n$ bits, there are only $\log_2 \boldsymbol{n}$ partition configurations available to resolve fault collisions.

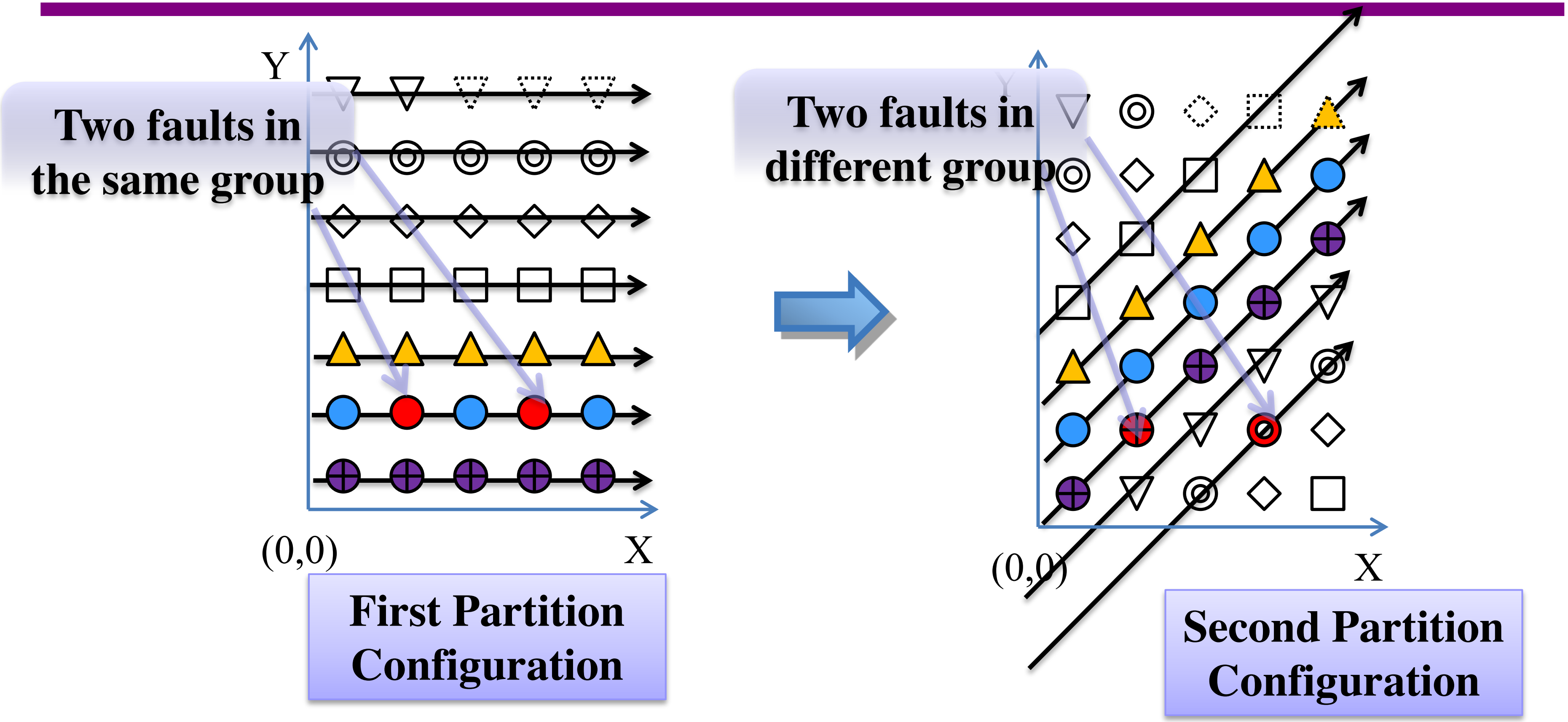
❑ In the worst case, group count can increase **exponentially** with accumulating faults.

❑ Only $\log_2 \boldsymbol{n}$ faults could exhaust the configurations and essentially demand an inversion vector as large as the data block itself.

# Design Objectives of Aegis

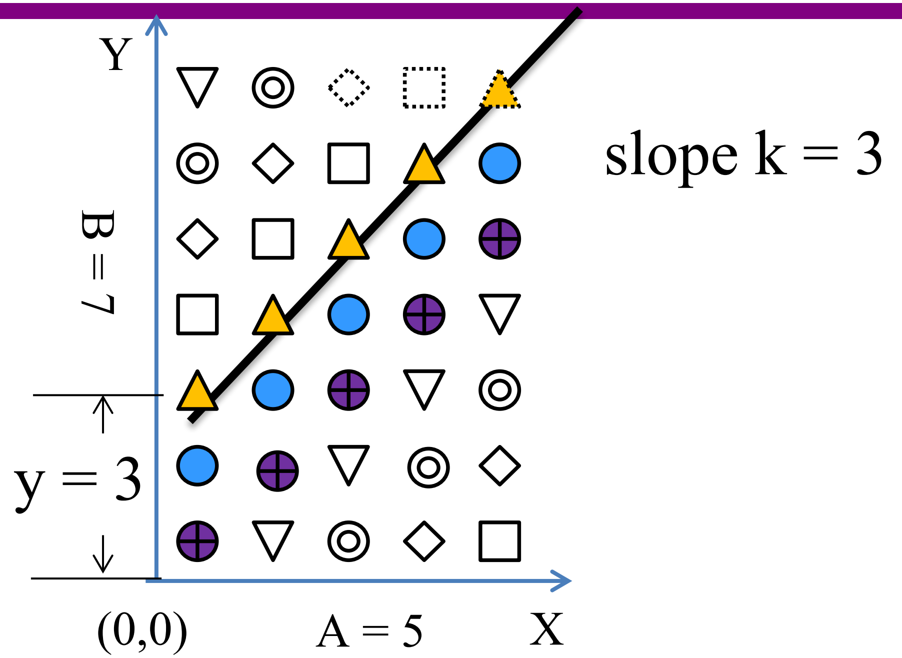
- A larger set of partition configurations for resolving fault collision to tolerate more faults.
  - A new configuration is needed whenever two faults collide in a group.
  - More candidate configurations mean more tolerable faults.
- A smaller number of groups in each configuration to reduce space overhead.
  - Group count mainly determines space overhead.
- Actively shuffling bits among groups to even out cell wears.
  - Cells in a group with faults wear out faster.

# Design of Aegis: an Observation


Two faults in the same group
Y
(0,0)
X
First Partition Configuration
Two faults in different group
Y
(0,0)
X
Second Partition Configuration

- ❑ Bits of a data block are placed on the Cartesian plane.
- ❑ A set of parallel lines defines a partition configuration.

# Aegis's Group Partition Scheme



- ❑ Aegis arranges bits of an $n$-bit data block on an $A \times B$ rectangle on the Cartesian plane.
- ❑ Bits $(a, b)$, where $\boxed{b = (a \times k + y) \,\%\, B}$, for a given slope $k$ and a given line $y$ , are in the same group.
- ❑ Each $k \in [0, B-1]$ corresponds to a partition configuration, and each y $\in [0, B-1]$ corresponds to a group in the configuration.

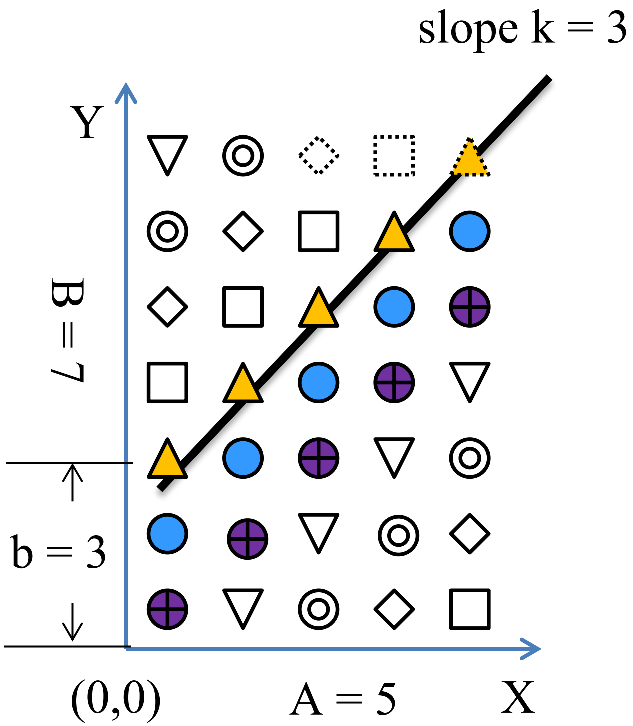
# Principle of Fault Collision Resolving

We have proved that under the Aegis partition scheme:

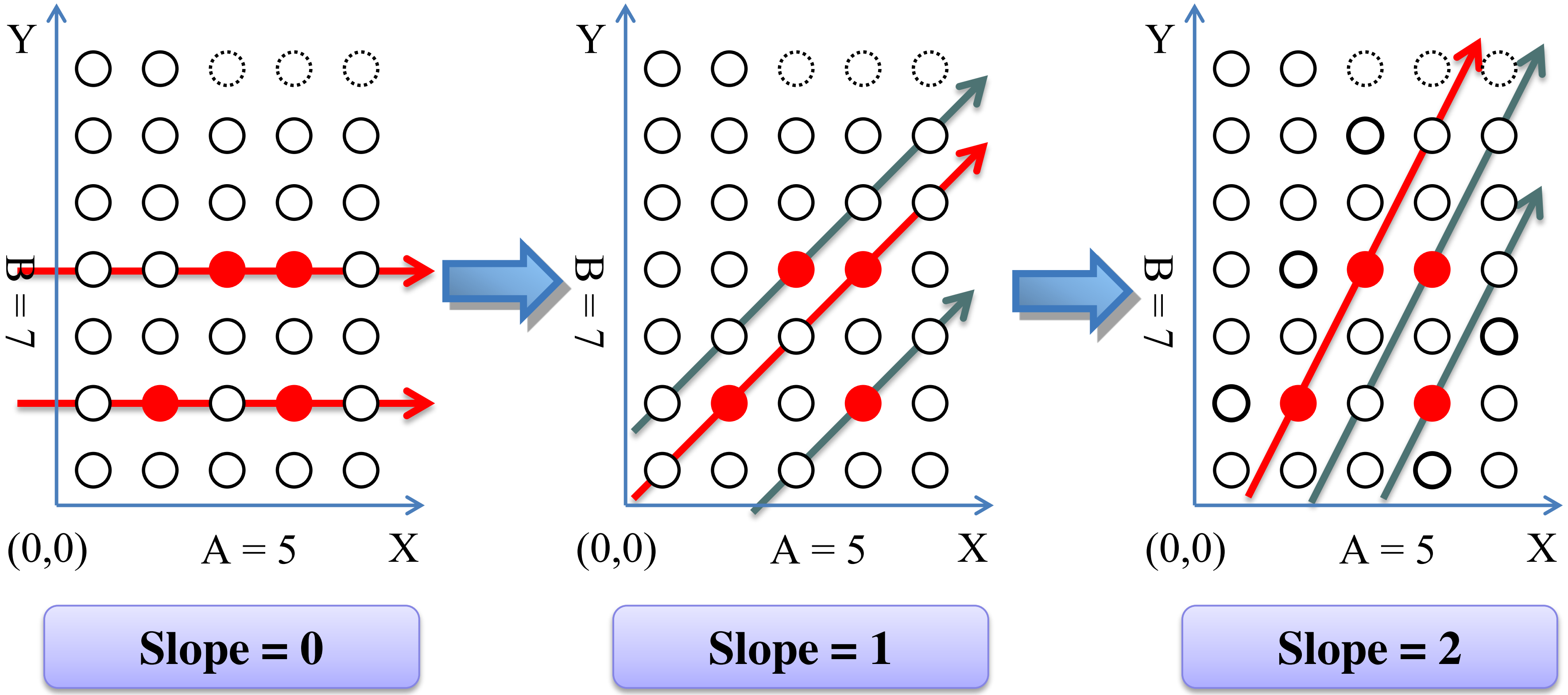
*Any two bits in the same group of a data block in a partition configuration will not be in the same group in a different partition configuration as long as:*

❑ $B$ is a prime number.

❑ $A \leq B$



# A Concern: How about Collisions after Re-partitions?



**Slope = 0**

**Slope = 1**

**Slope = 2**

There is possibility that multiple re-partitions are needed to reach a configuration without any fault collisions.

# Aegis Guarantees a Collision-free Configuration

- ❑ Collision of any pair of faults appear in only one partition configuration
- ❑ A data block of $f$ faults can generate at most $\boxed{\binom{f}{2}}$, or $\frac{f\times(f+1)}{2}$, different collisions of fault pairs.
- ❑ Each re-partition eliminates at least one such collision.
- ❑ As long as number of configurations in a partition scheme, B, is larger than $\boxed{\frac{f\times(f+1)}{2}}$, there exists at least one collision-free configuration.
- ❑ For a set of known faults, a pre-wired logic can be used to compute collision-free configuration(s).

# Aegis's Advantages

**To guarantee a tolerance of *f* faults:**

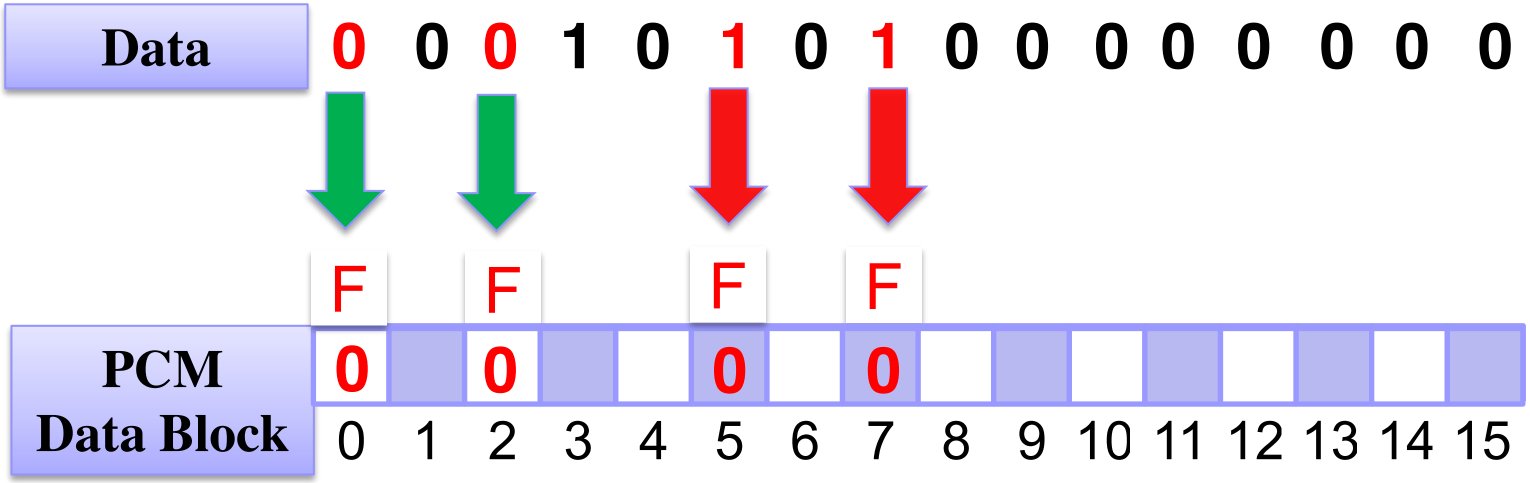
- ❑ Aegis provides B partition configurations to resolve collisions. (B is the minimal prime number satisfying $\boxed{\binom{f}{2} < B}$)
  - ➢ SAFER provides only $\boxed{f}$ usable configurations.
- ❑ Aegis has only $\boxed{B}$ groups in a configuration.
  - ➢ SAFER has $\boxed{\mathbf{2}^f}$ groups in a configuration.
- ❑ Aegis can have a much smaller space overhead.

# Comparison of Space Cost

**To guarantee a tolerance of *f* faults in a 512-bit data block:**

| ***f* (# of faults)** | **1** | **2** | **3** | **4** | **5** | **6** | **7** | **8** | **9** | **10** |
|---|---|---|---|---|---|---|---|---|---|---|
| ECP | 11 | 21 | 31 | 41 | 51 | 61 | 71 | 81 | 91 | 101 |
| SAFER | 1 | 7 | 14 | 22 | 35 | 55 | 91 | 159 | 292 | 552 |
| Aegis | 23 | 24 | 25 | 26 | 27 | 27 | 28 | 34 | 43 | 53 |

# Aegis-rw: Tolerate More faults



- Assume we know the distinction between stuck-at-wrong and stuck-at-right faults before the actual write.

- Use a fail cache to record fault locations and stuck-at values.

- Aegis-rw: allowing multiple W faults or R faults in a group.
  - Only $f_w \times f_r + 1$ partition configurations are required.

# Experimental Setup

❑ Cell lifetime follows the normal distribution with a mean lifetime of $10^8$ writes and a 25% coefficient of variance.

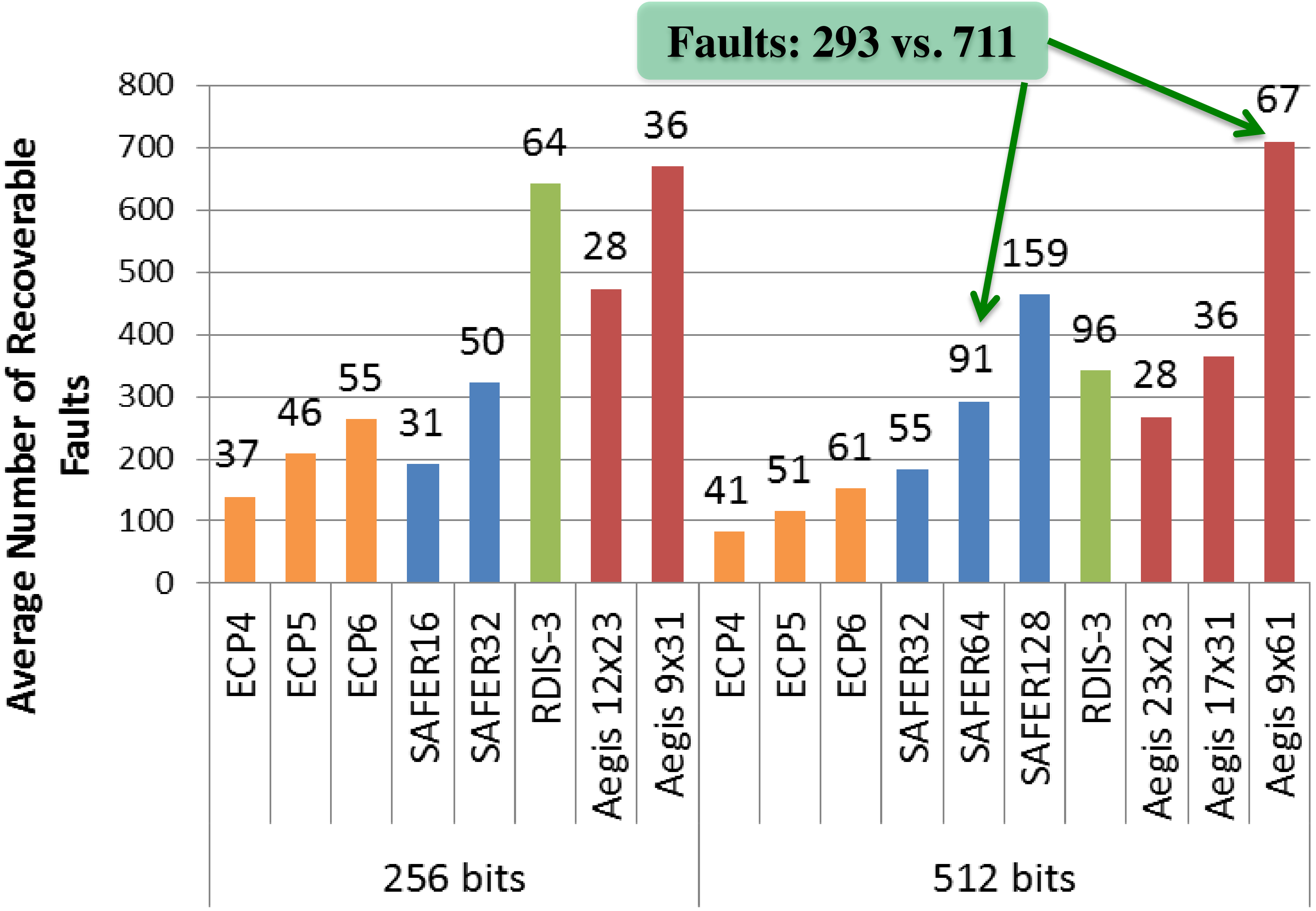
❑ A perfect wear leveling is assumed.

❑ A cell has a $50\%$ probability to be updated in serving a write request.

❑ Compare with ECP, SAFER, and RDIS. SAFER may use a cache to avoid the second writes.

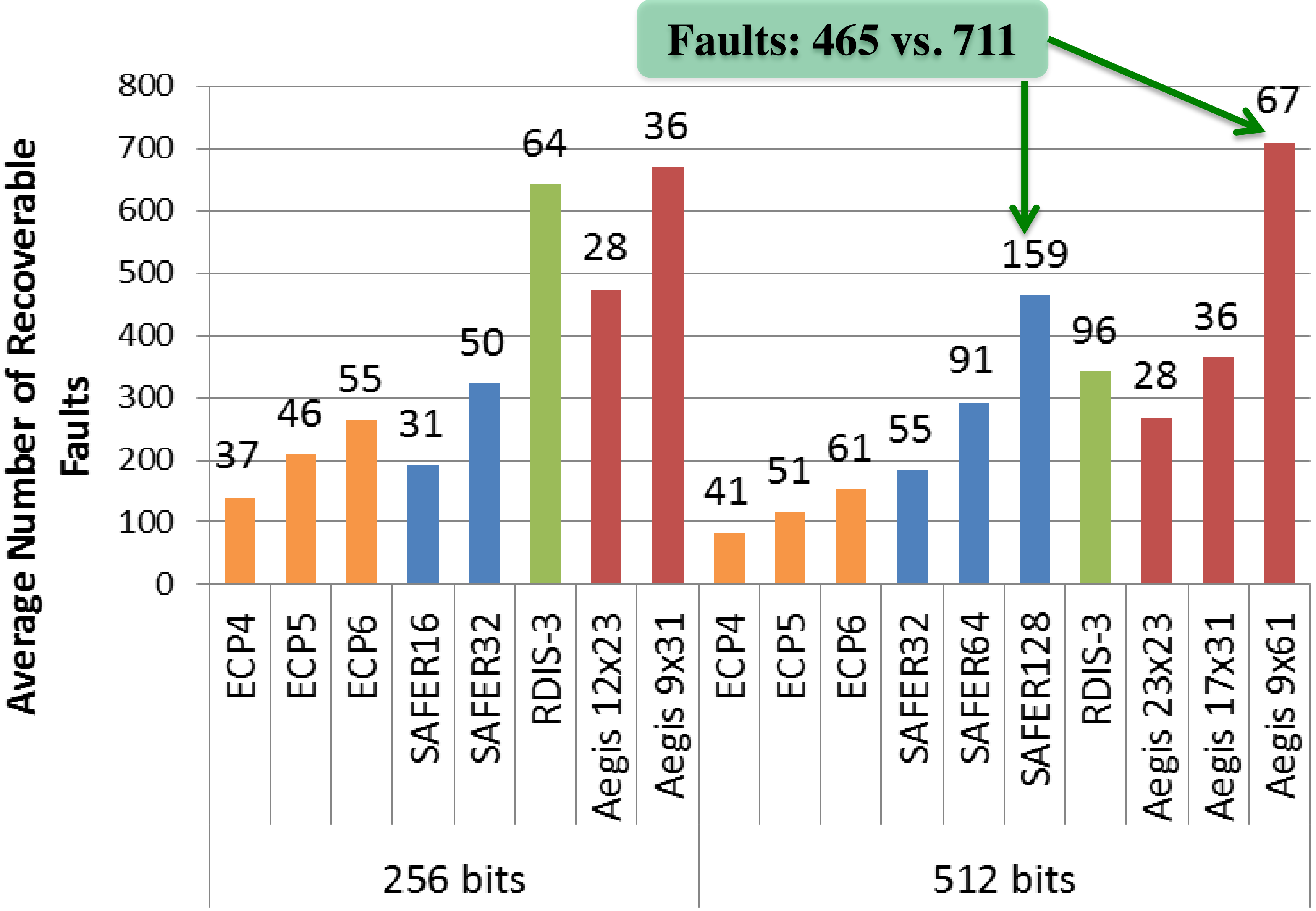
❑ We continuously issue page (4KB) writes to a 8MB PCM memory until all memory blocks are dead.

# Average Number of Recoverable Faults in a 4KB Page



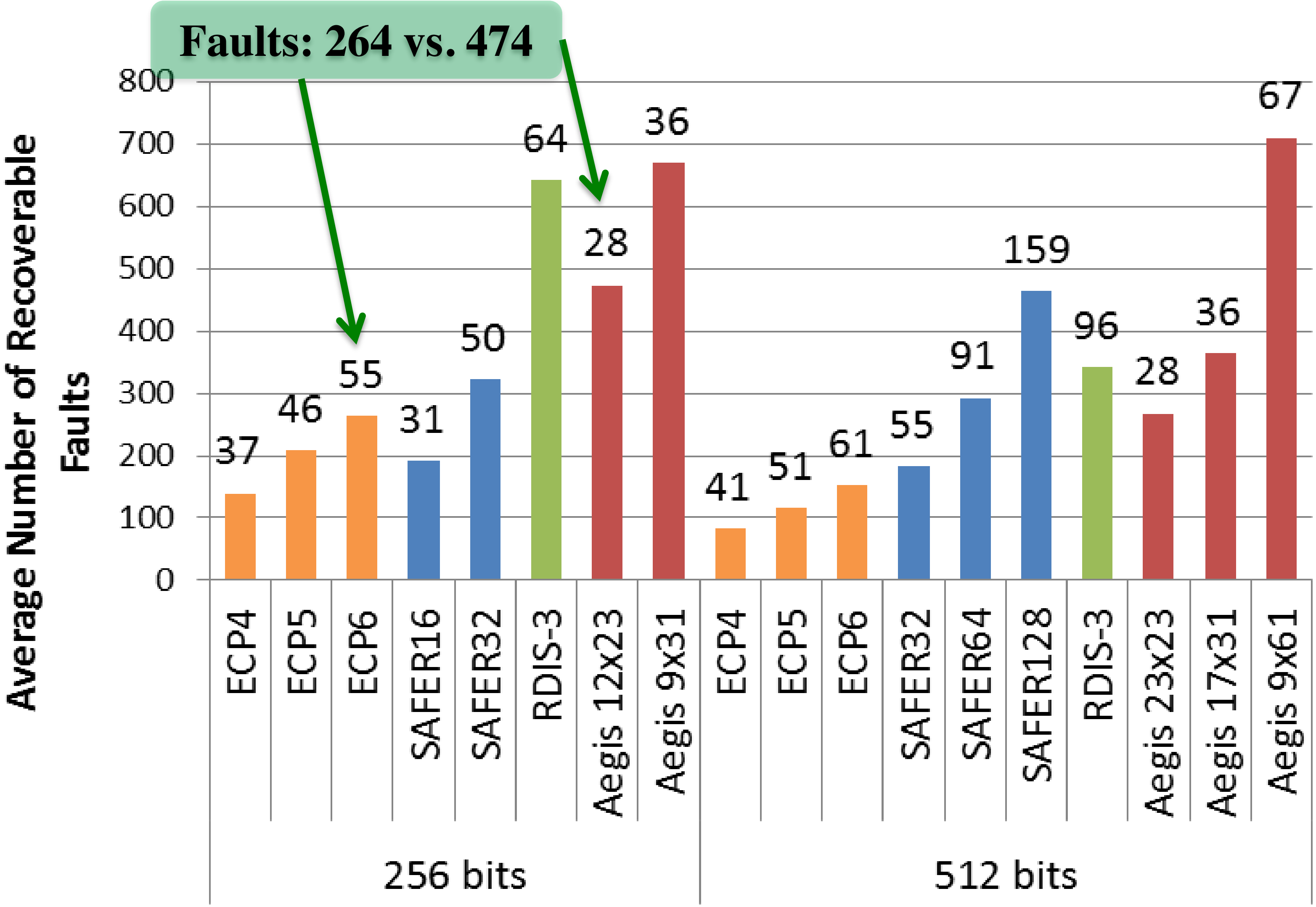
Faults: 293 vs. 711

# Average Number of Recoverable Faults in a 4KB Page

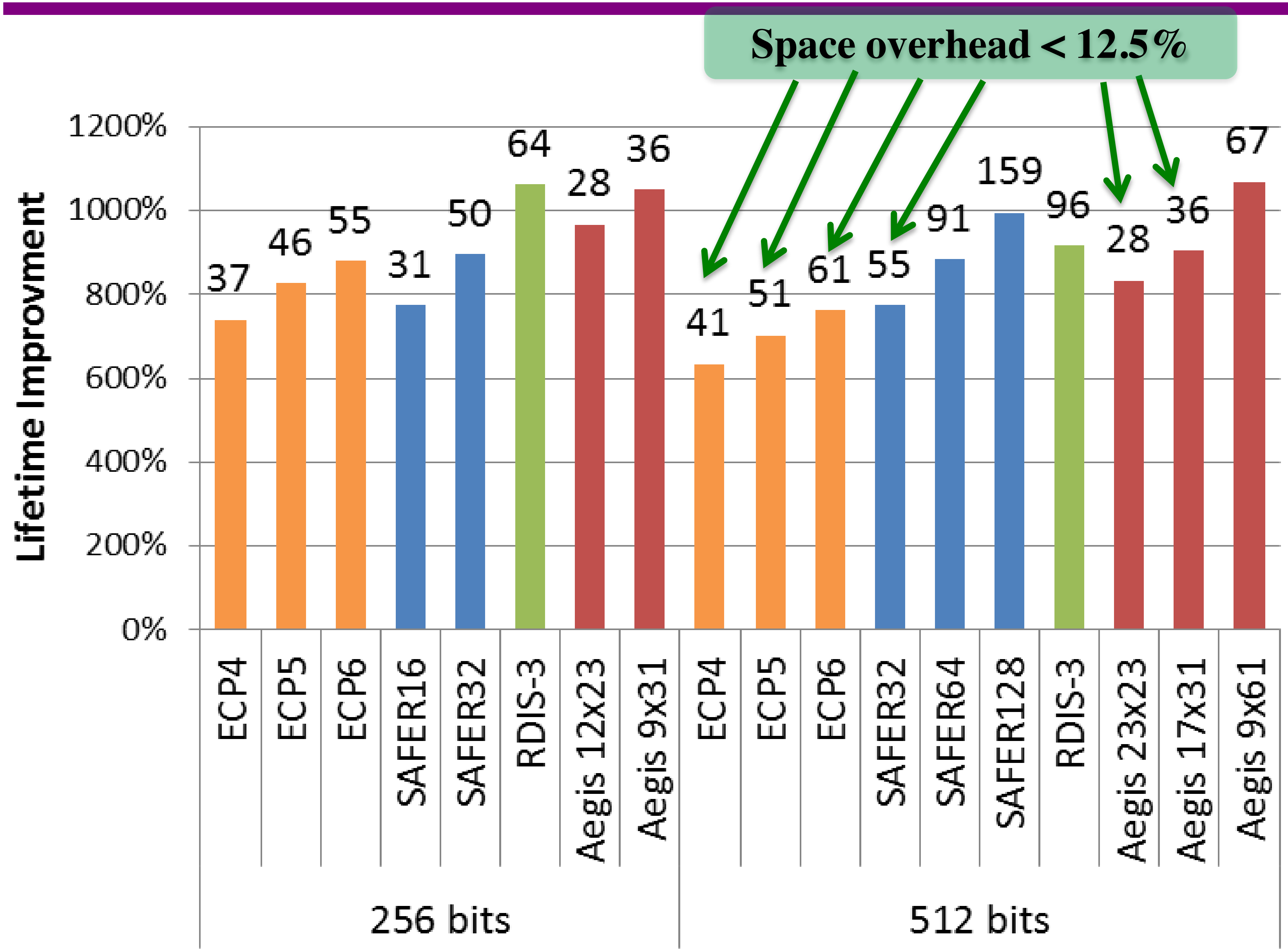


Faults: 465 vs. 711

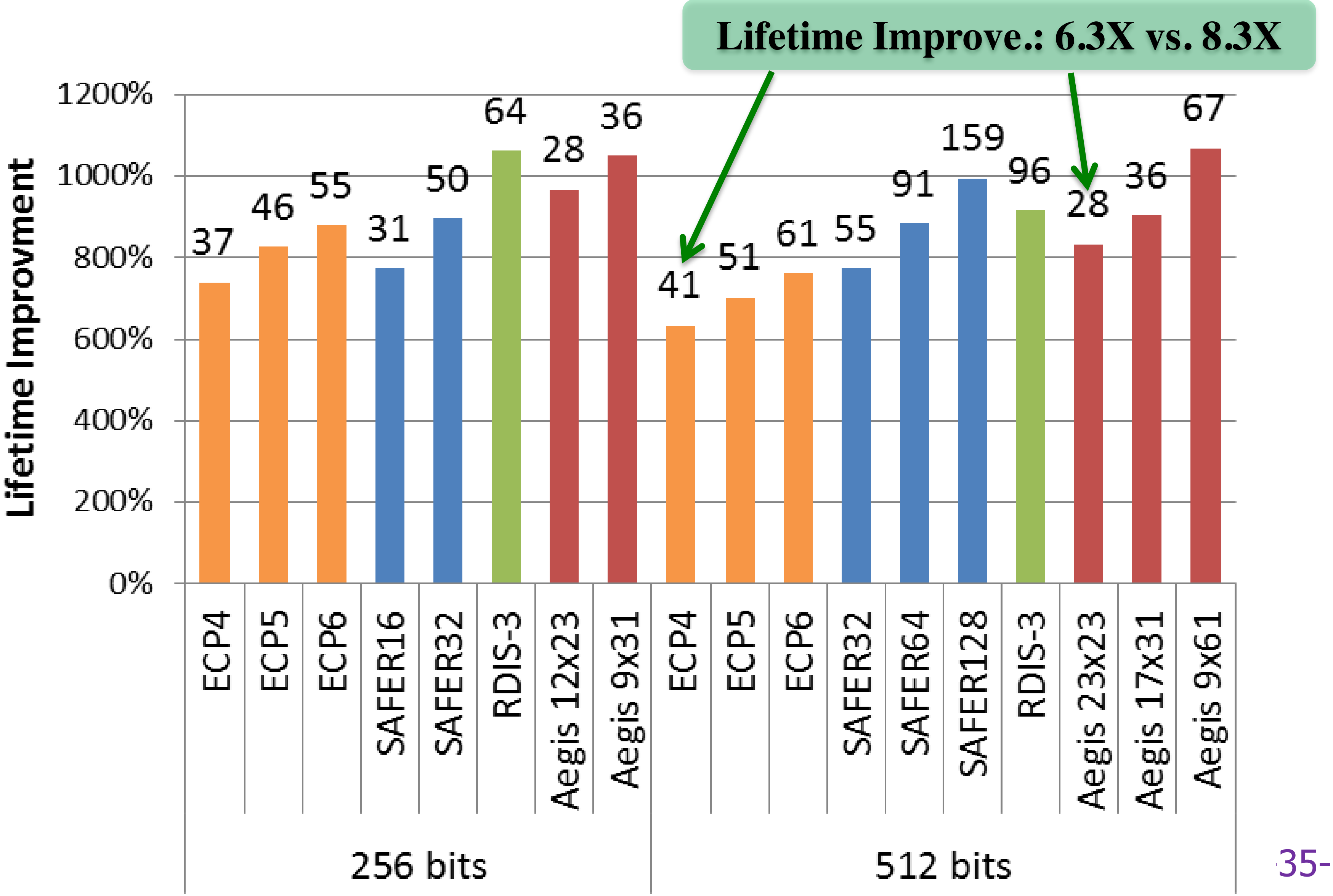
# Average Number of Recoverable Faults in a 4KB Page


Faults: 264 vs. 474

# Improvement of 4KB-page's Lifetime

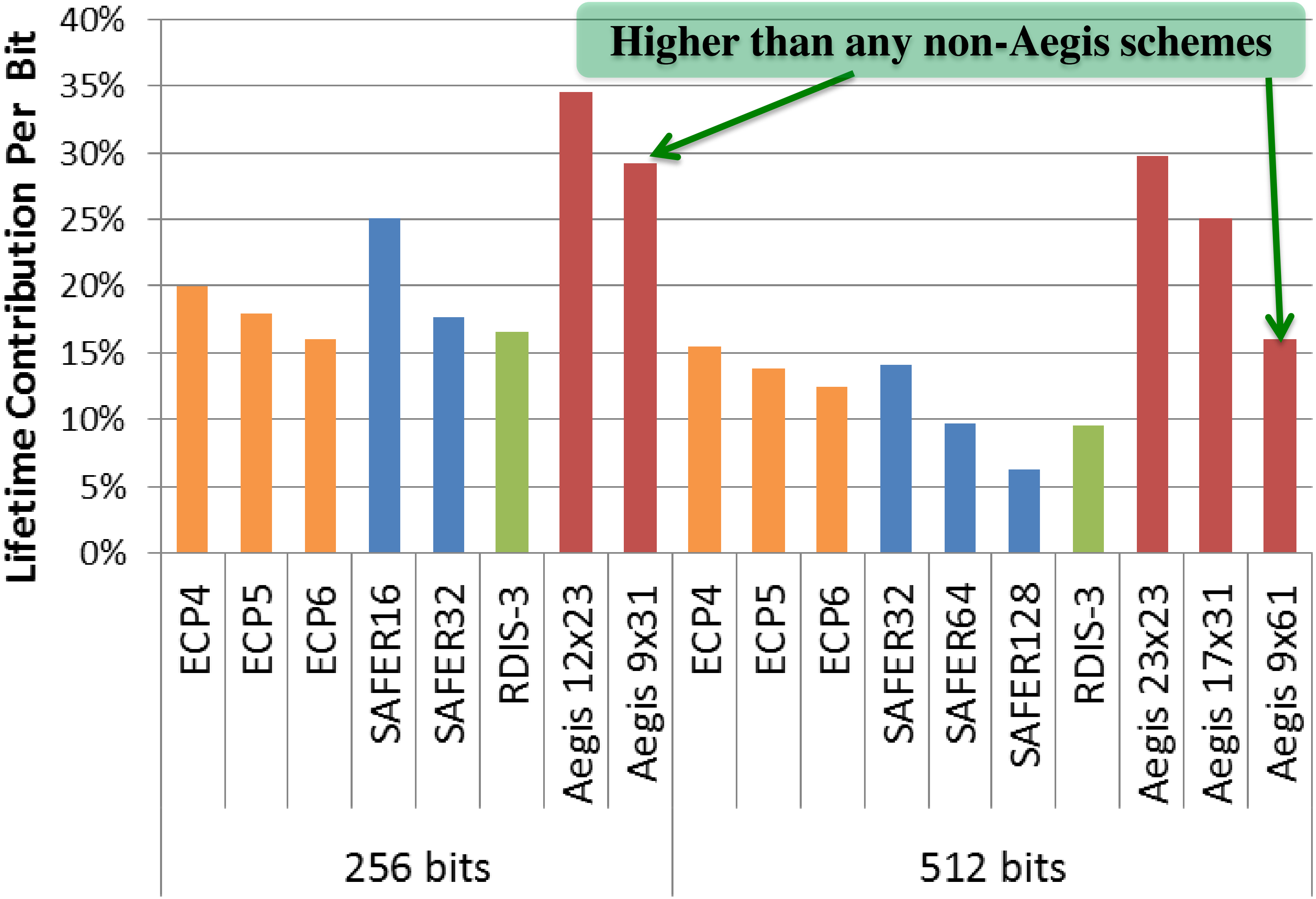

Space overhead < 12.5%

# Improvement of 4KB-page's Lifetime



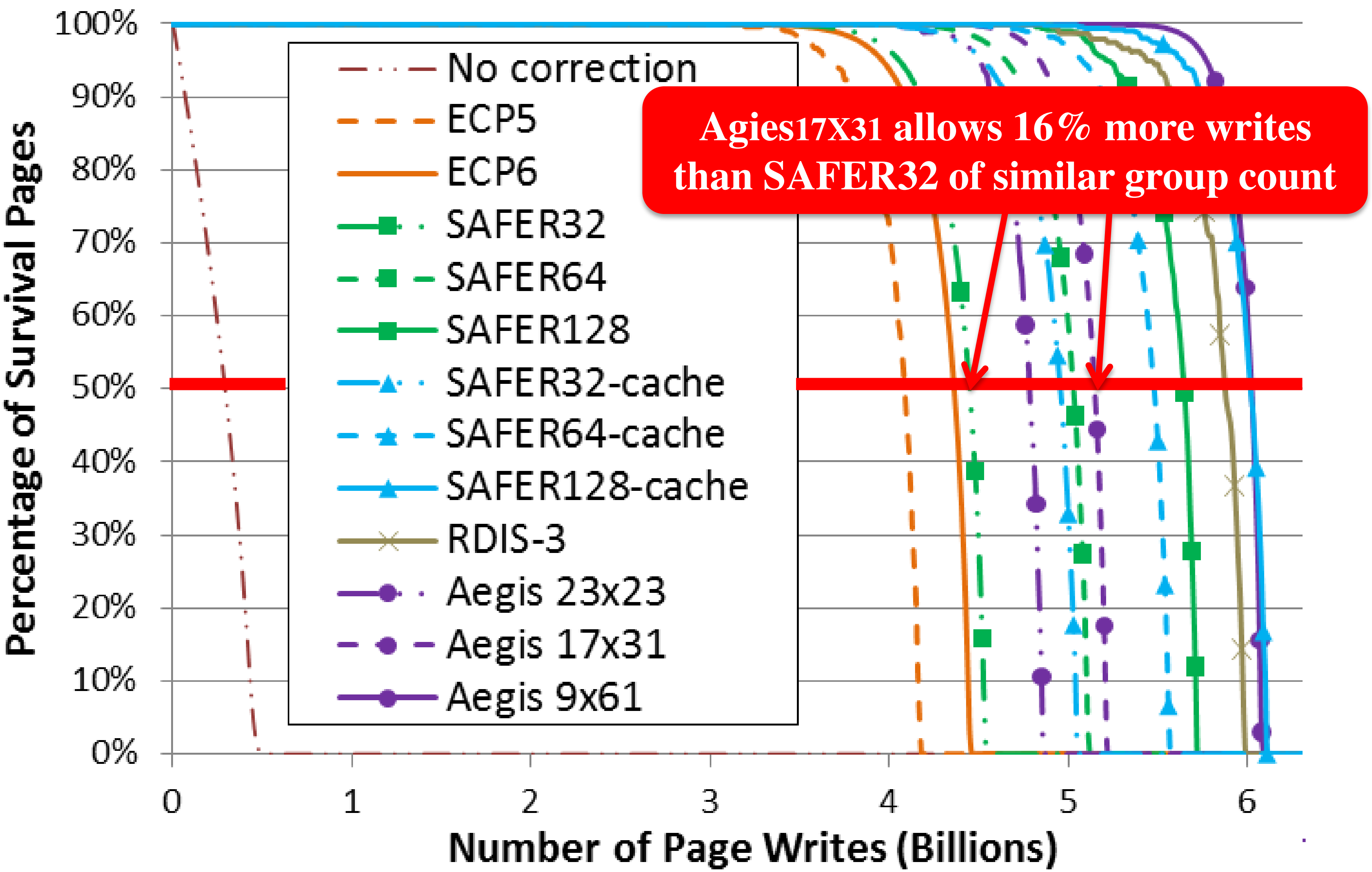
Lifetime Improve.: 6.3X vs. 8.3X

# Each Bit's Contribution to the Lifetime Improvement



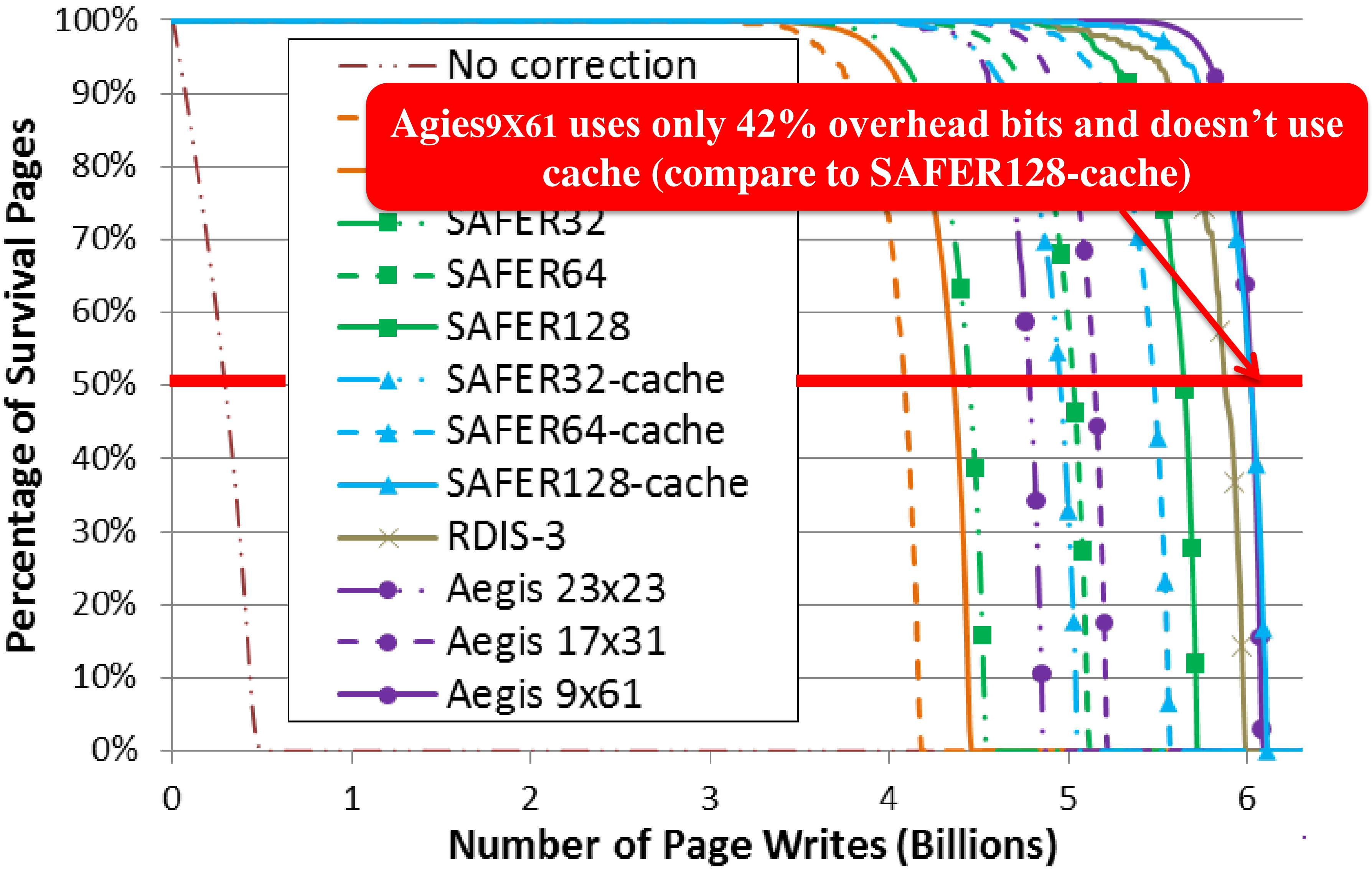
Higher than any non-Aegis schemes

# Survival Rate of a 4KB-page



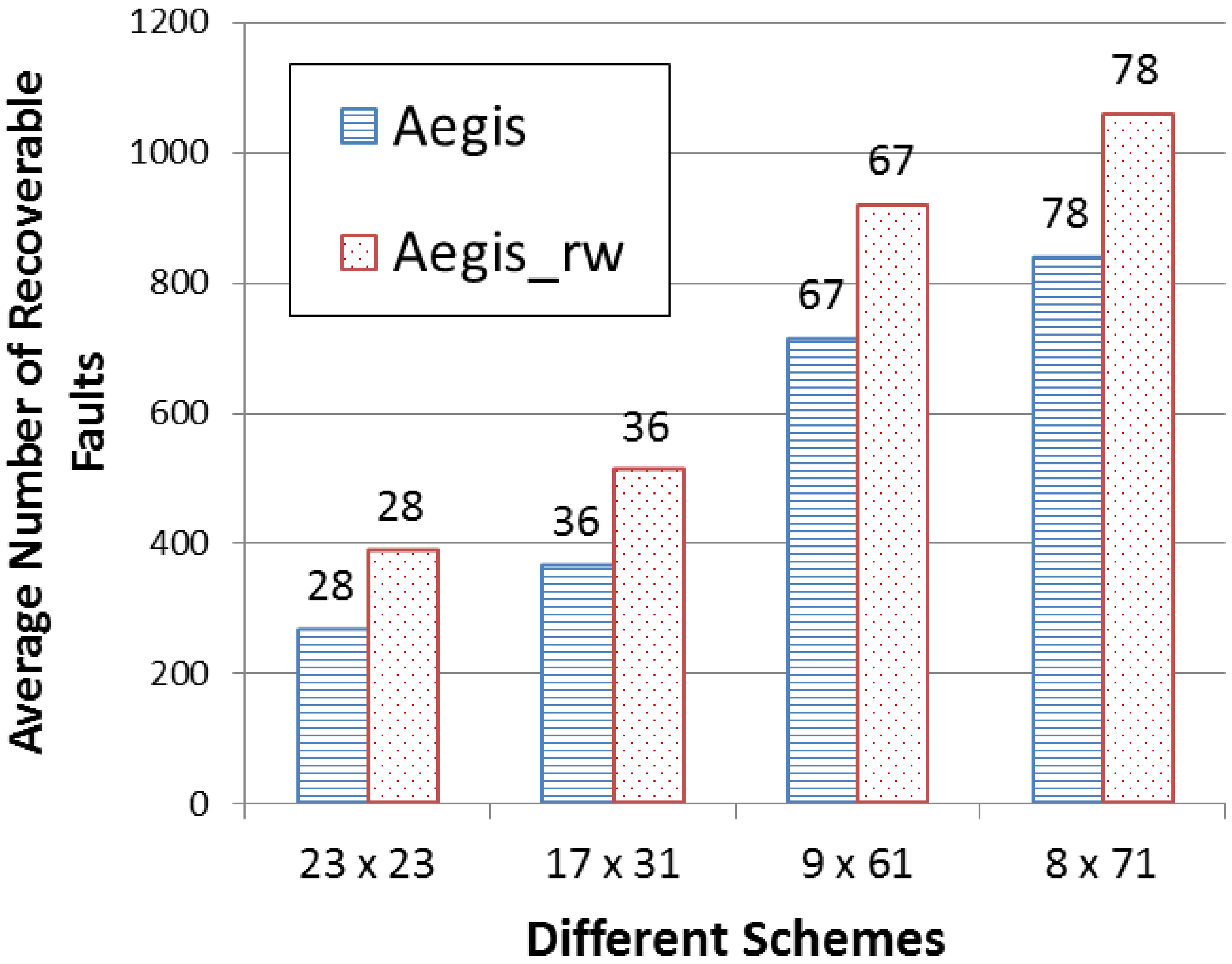
Agies17X31 allows 16% more writes than SAFER32 of similar group count

# Survival Rate of a 4KB-page



Agies9X61 uses only 42% overhead bits and doesn’t use cache (compare to SAFER128-cache)

# Compare Aegis with Aegis-rw



# Conclusions

❑ To meet the demand on PCM’s high fault tolerance, Aegis effectively separates many faults in different groups for inversion-based recovery.

❑ To minimize space overhead, Aegis provides a large number of partition configurations and a small number of groups in each configuration.

❑ Extensive experiments show Aegis provides substantially higher fault tolerance, longer lifetime, and lower cost.