Inhale: Enabling High-Performance and Energy-Efficient In-SRAM Cryptographic Hash for IoT

Jingyao Zhang, Elaheh Sadredini

2022 IEEE/ACM International Conference on Computer-Aided Design

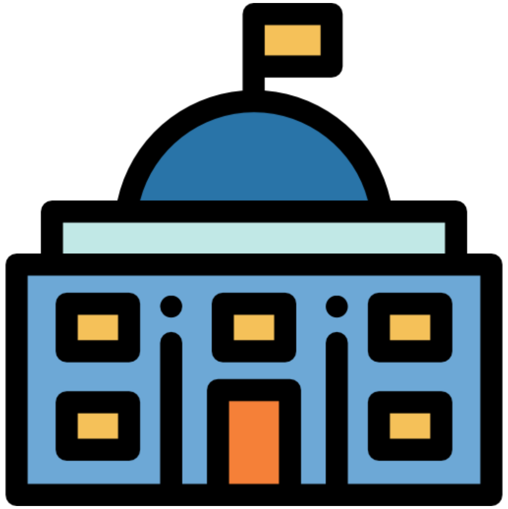
UC RIVERSIDE

Why IoT security is crucial

Why IoT security is crucial

Healthcare industry

Home

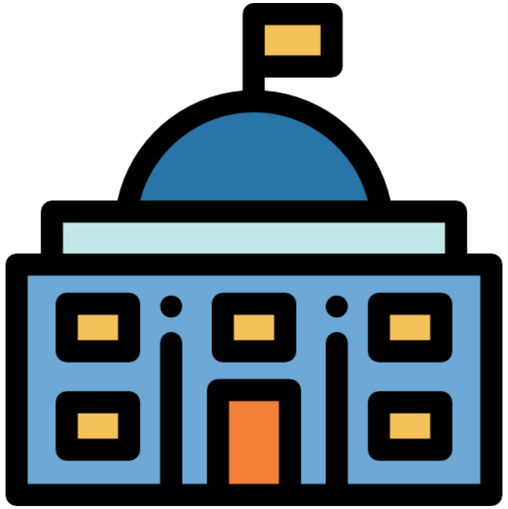


Government

Why IoT security is crucial

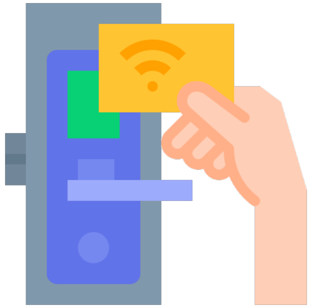
Healthcare industry

Home



Government

ECG monitor

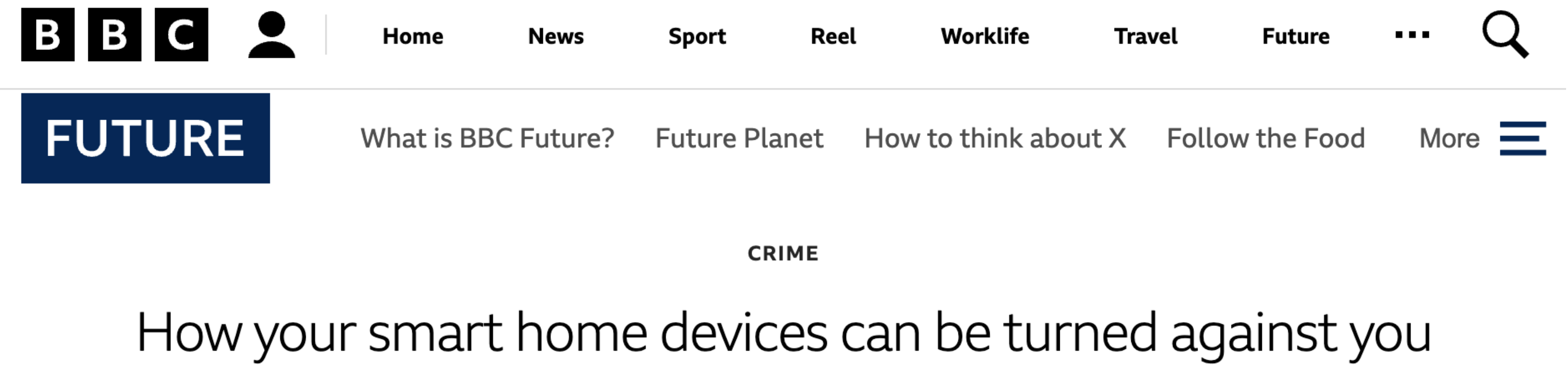


Smart lock

Security camera

IoT attacks are happening now!

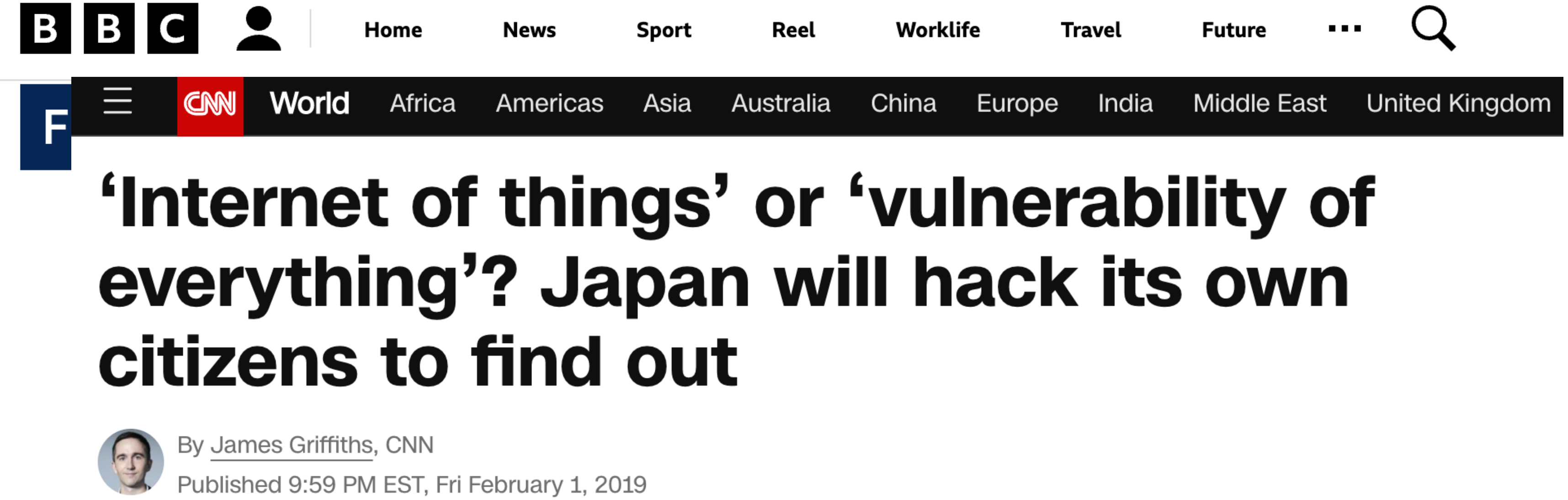
IoT attacks are happening now!



CRIME

How your smart home devices can be turned against you

IoT attacks are happening now!

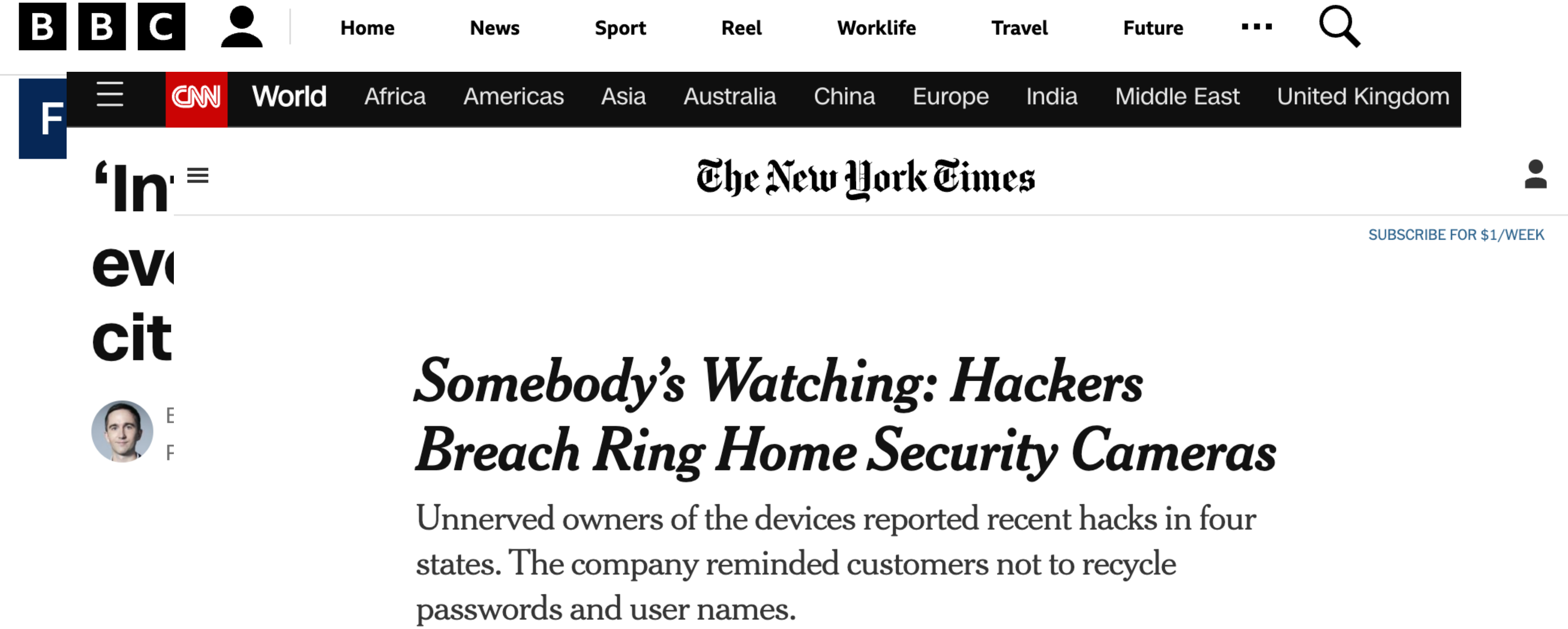


‘Internet of things’ or ‘vulnerability of everything’? Japan will hack its own citizens to find out

By James Griffiths, CNN

Published 9:59 PM EST, Fri February 1, 2019

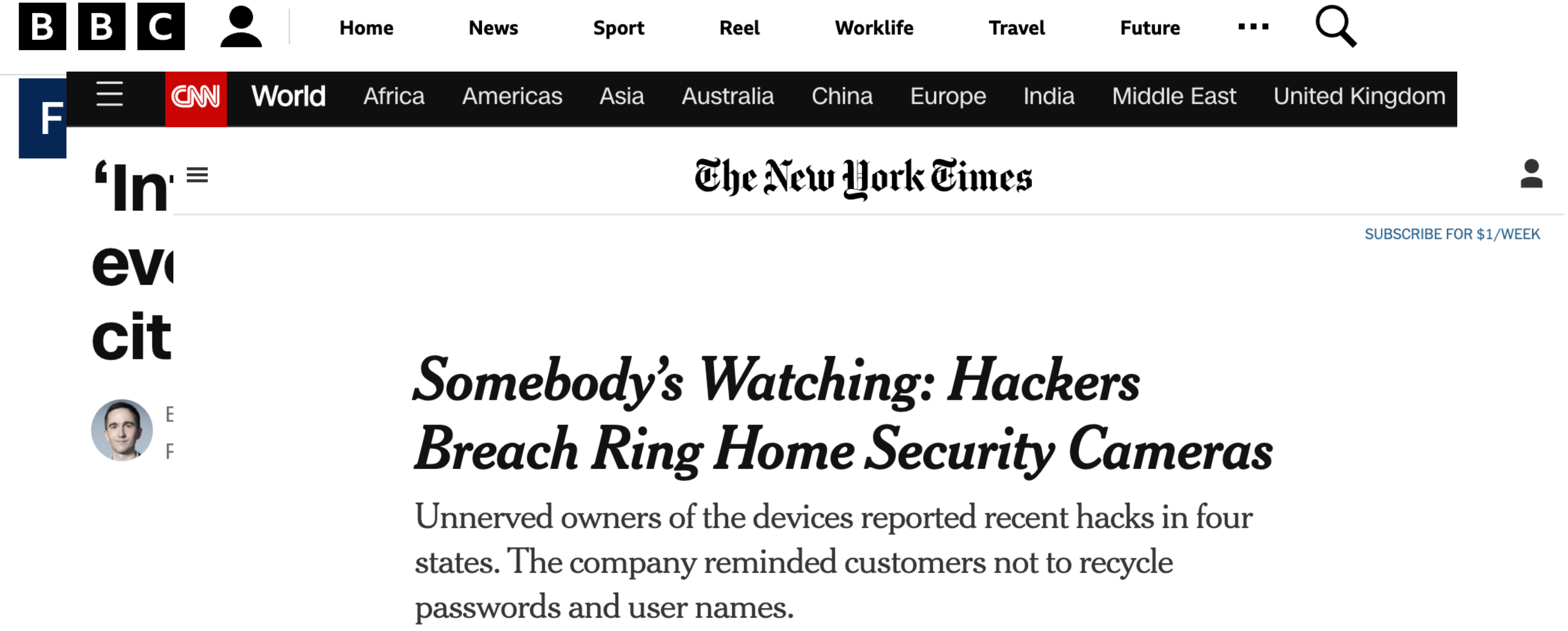
IoT attacks are happening now!



Somebody's Watching: Hackers Breach Ring Home Security Cameras

Unnerved owners of the devices reported recent hacks in four states. The company reminded customers not to recycle passwords and user names.

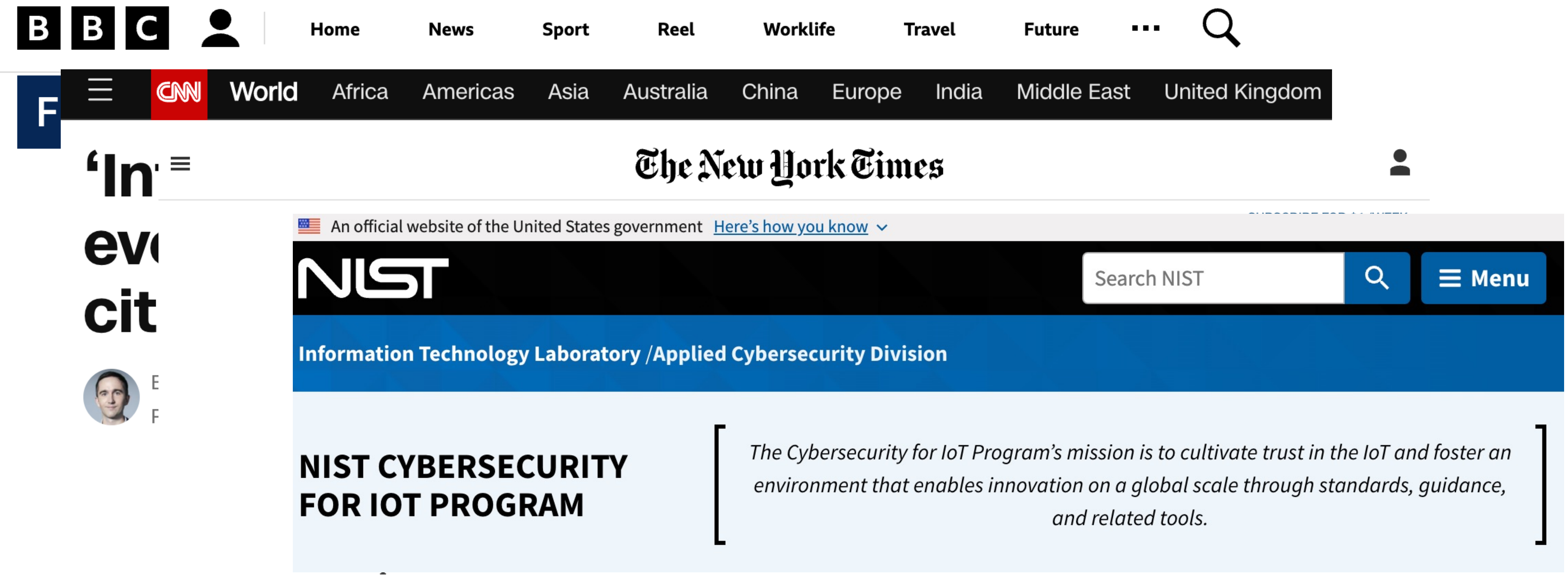
IoT attacks are happening now!
IoT security is urgent!



Somebody’s Watching: Hackers Breach Ring Home Security Cameras

Unnerved owners of the devices reported recent hacks in four states. The company reminded customers not to recycle passwords and user names.

IoT attacks are happening now!
IoT security is urgent!



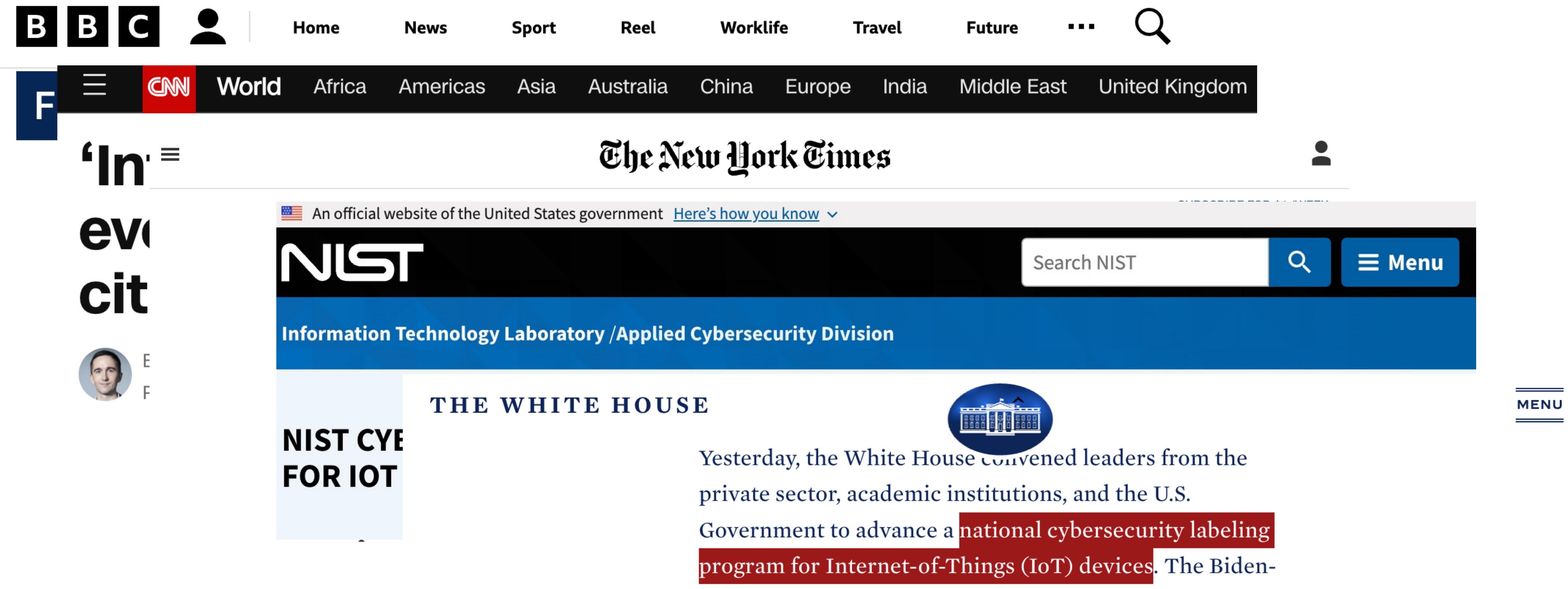
An official website of the United States government Here's how you know

Information Technology Laboratory /Applied Cybersecurity Division

NIST CYBERSECURITY FOR IOT PROGRAM

The Cybersecurity for IoT Program's mission is to cultivate trust in the IoT and foster an environment that enables innovation on a global scale through standards, guidance, and related tools.

IoT attacks are happening now!
IoT security is urgent!



Yesterday, the White House convened leaders from the private sector, academic institutions, and the U.S. Government to advance a national cybersecurity labeling program for Internet-of-Things (IoT) devices. The Biden-

Foundation of IoT security

Foundation of IoT security

- IoT security highly relies on data integrity to authenticate identity

Foundation of IoT security

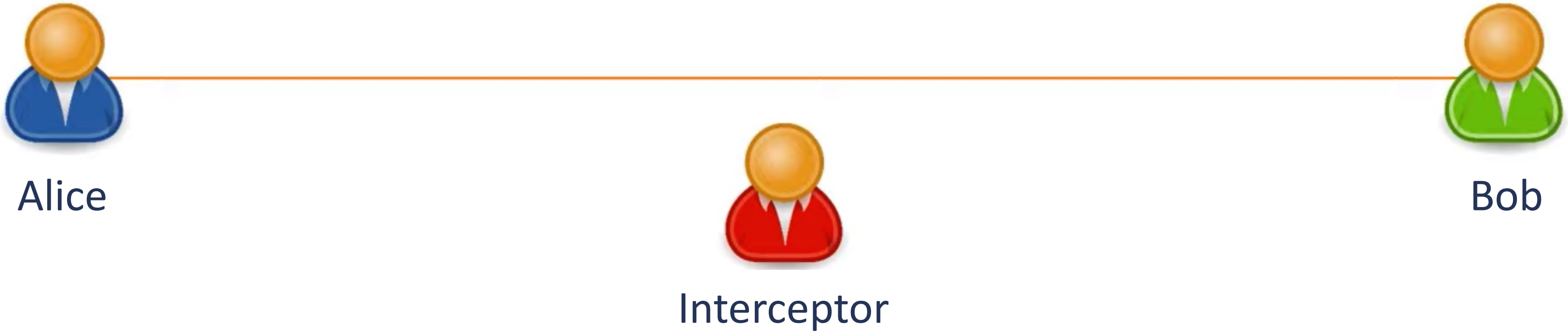
- IoT security highly relies on data integrity to authenticate identity
- In engineering, cryptographic hash algorithm is adopted for data integrity

Foundation of IoT security

- IoT security highly relies on data integrity to authenticate identity
- In engineering, cryptographic hash algorithm is adopted for data integrity
- **Practically infeasible to invert or reverse the hash computation**

Example: Secure Communication

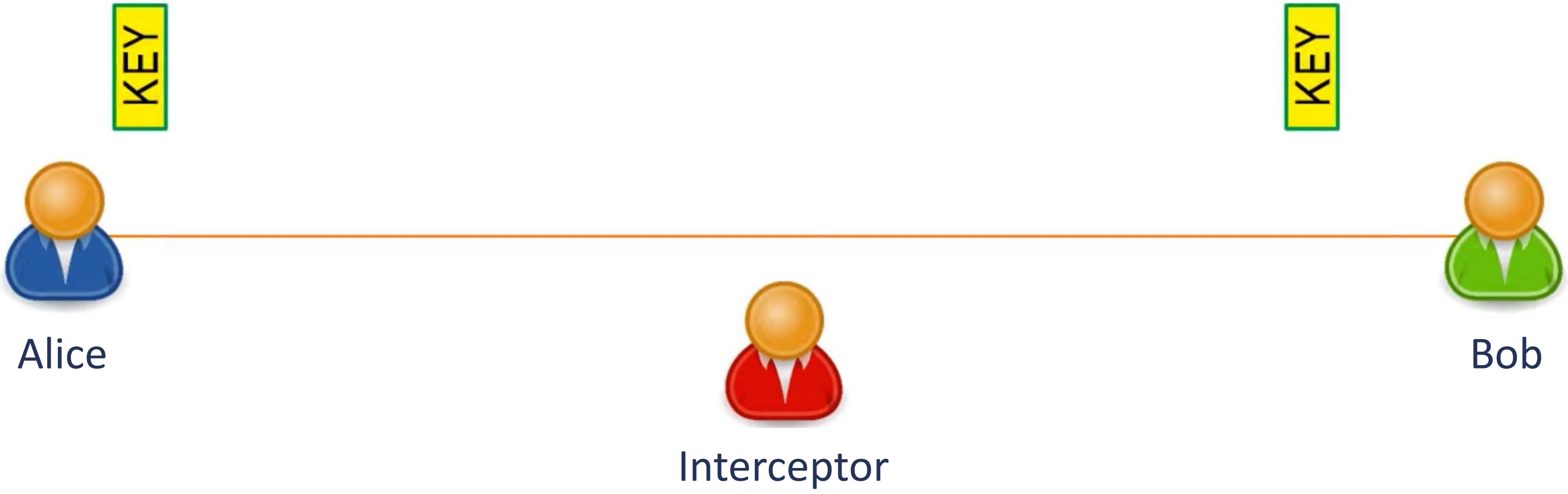
- Hashing can provide Data Integrity and Identity Authentication



https://www.youtube.com/watch?v=doN3lzzNEIM&t=96s

Example: Secure Communication

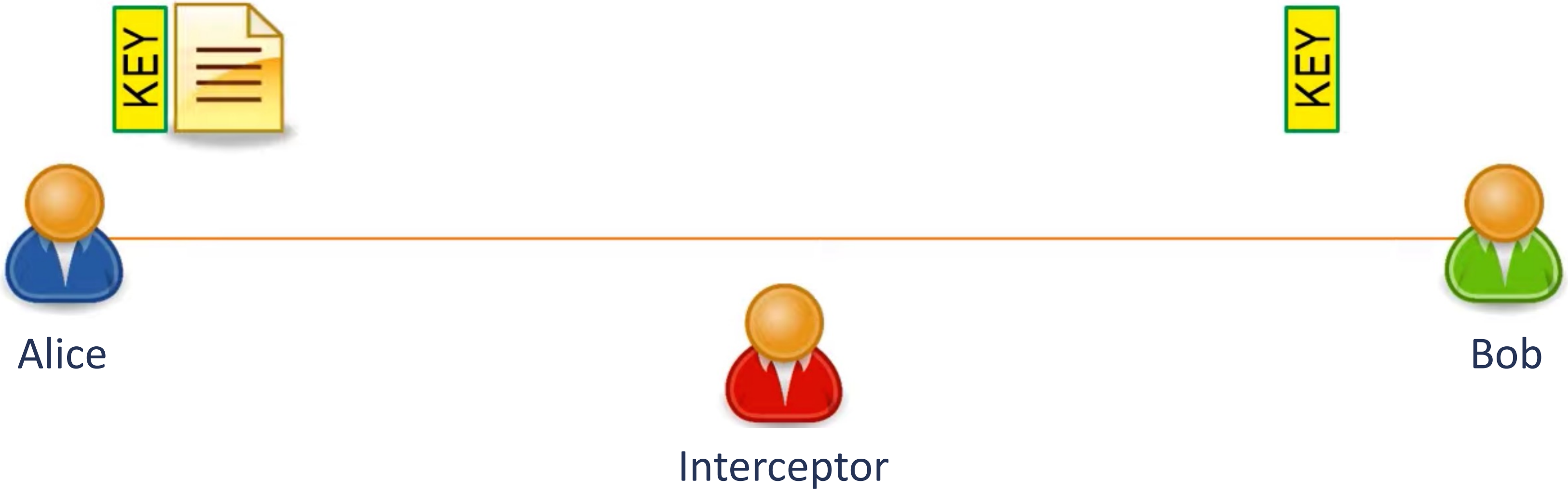
- Hashing can provide Data Integrity and Identity Authentication
  - They establish a mutual *Secret Key* with **key encapsulation mechanism (KEM)**



https://www.youtube.com/watch?v=doN3lzzNEIM&t=96s

Example: Secure Communication

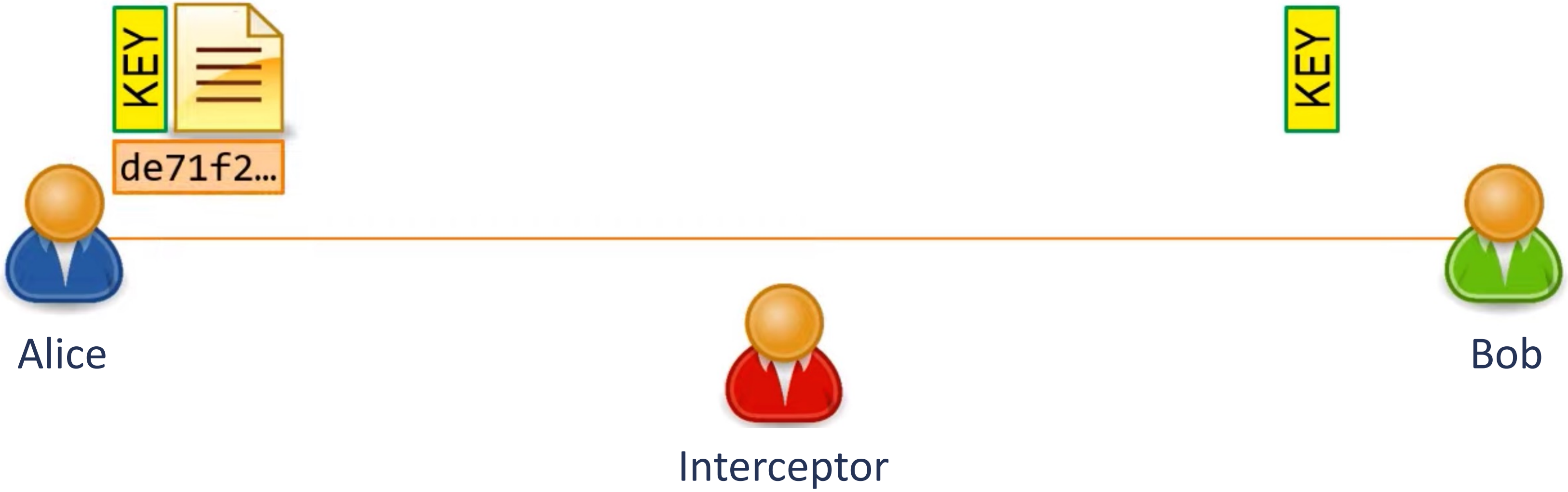
- Hashing can provide Data Integrity and Identity Authentication
  - They establish a mutual *Secret Key* with **key encapsulation mechanism (KEM)**



https://www.youtube.com/watch?v=doN3lzzNEIM&t=96s

Example: Secure Communication

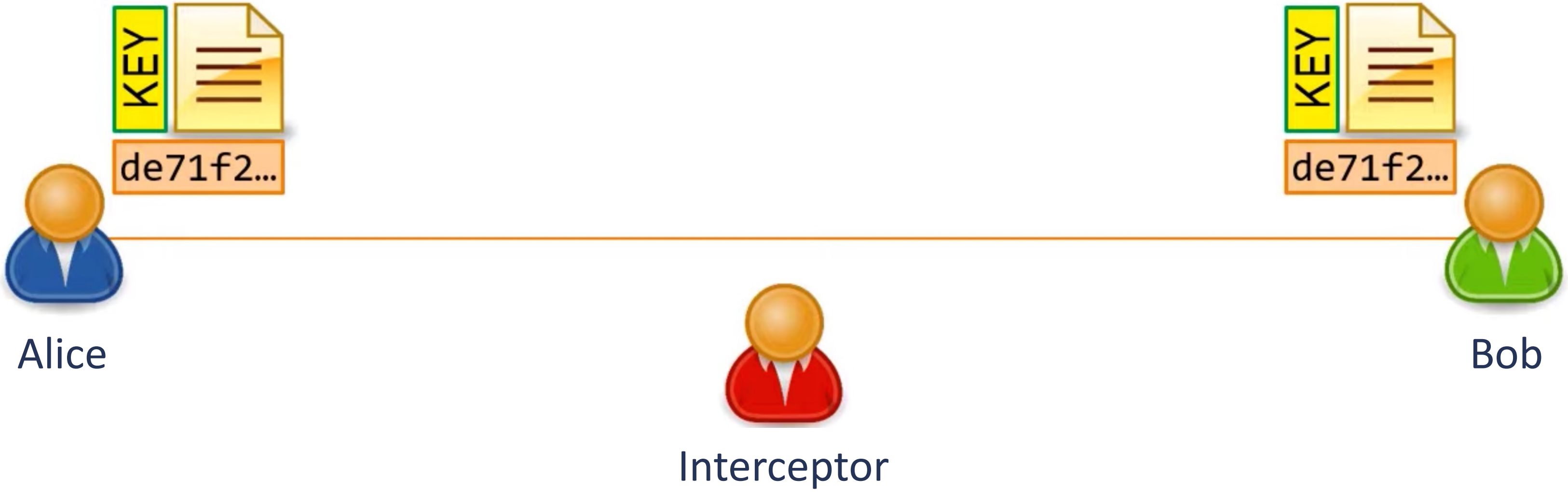
- Hashing can provide Data Integrity and Identity Authentication
  - They establish a mutual *Secret Key* with **key encapsulation mechanism (KEM)**
  - Alice combines *Message* + *Secret Key* to create *Digest* by **Hashing**



https://www.youtube.com/watch?v=doN3lzzNEIM&t=96s

Example: Secure Communication

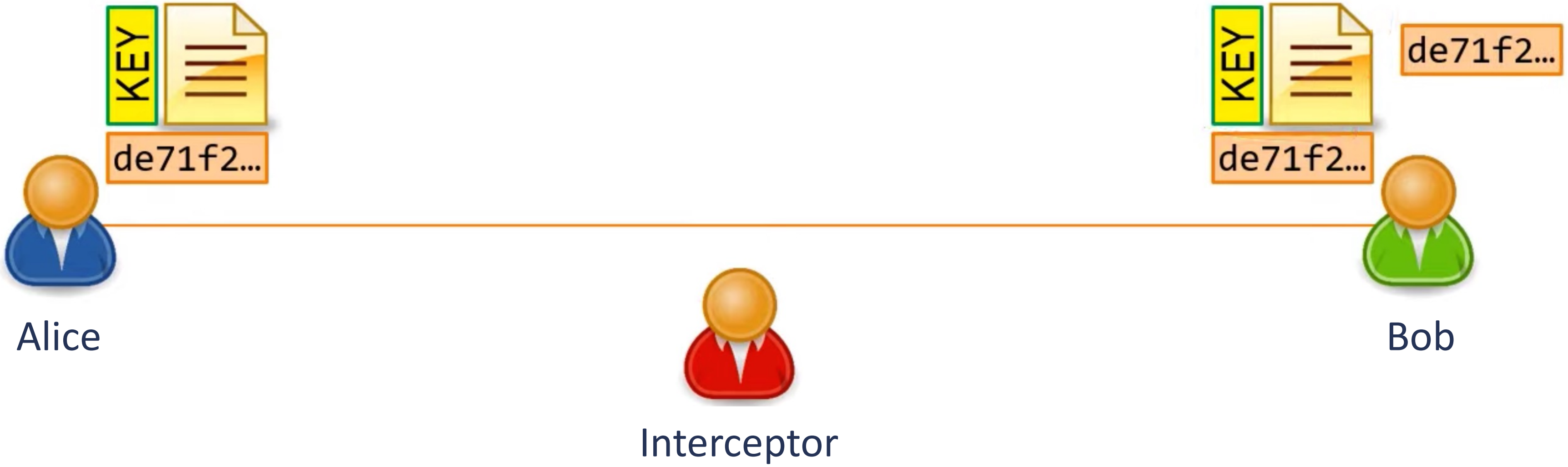
- Hashing can provide Data Integrity and Identity Authentication
  - They establish a mutual *Secret Key* with **key encapsulation mechanism (KEM)**
  - Alice combines *Message* + *Secret Key* to create *Digest* by **Hashing**



https://www.youtube.com/watch?v=doN3lzzNEIM&t=96s

Example: Secure Communication

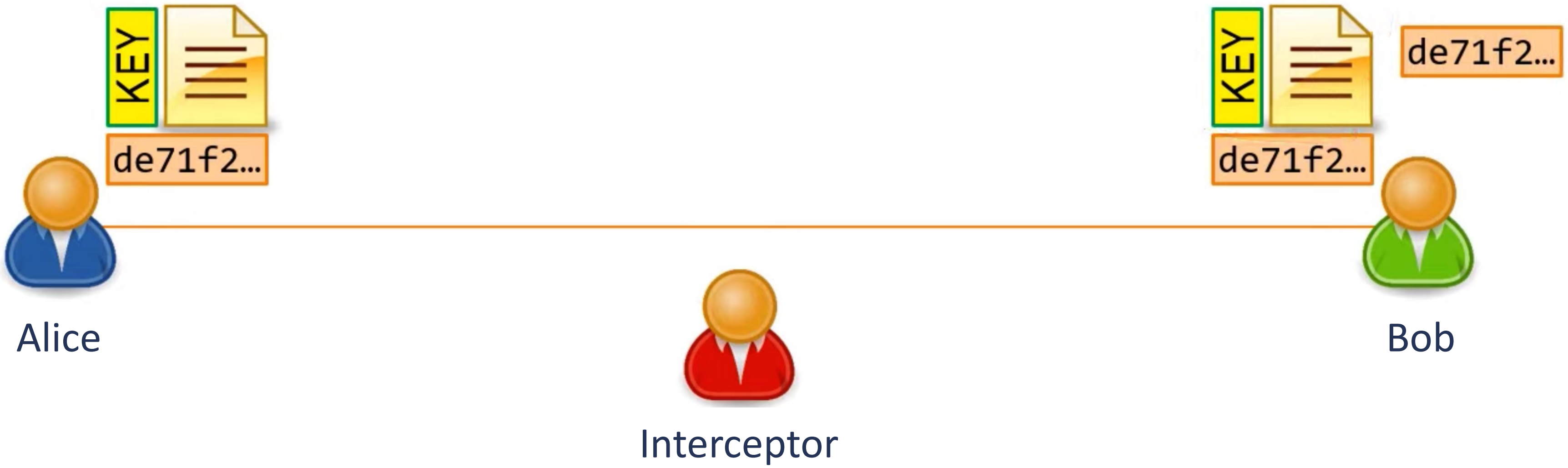
- Hashing can provide Data Integrity and Identity Authentication
  - They establish a mutual *Secret Key* with **key encapsulation mechanism (KEM)**
  - Alice combines *Message* + *Secret Key* to create *Digest* by **Hashing**
  - Bob verifies by calculating **Hash** of *Message* + *Secret Key*



https://www.youtube.com/watch?v=doN3lzzNEIM&t=96s

Example: Secure Communication

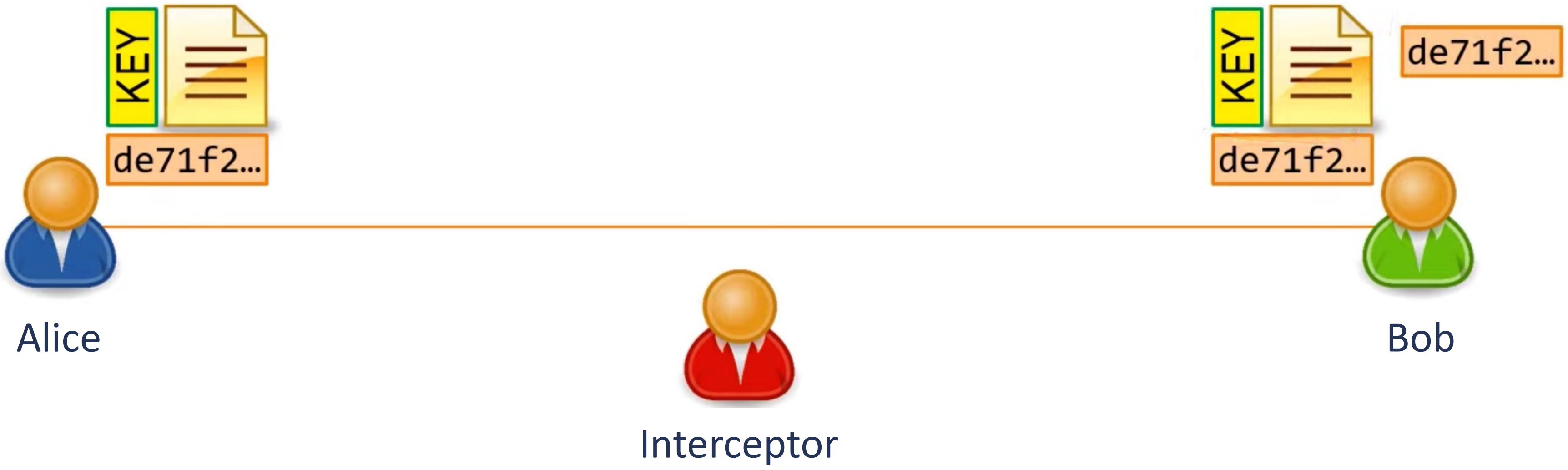
- Hashing can provide Data Integrity and Identity Authentication
  - They establish a mutual *Secret Key* with **key encapsulation mechanism (KEM)**
  - Alice combines *Message* + *Secret Key* to create *Digest* by **Hashing**
  - Bob verifies by calculating **Hash** of *Message* + *Secret Key*
    - *Message* was not modified in transit ------ **Integrity**



https://www.youtube.com/watch?v=doN3lzzNEIM&t=96s

Example: Secure Communication

- Hashing can provide Data Integrity and Identity Authentication
  - They establish a mutual *Secret Key* with **key encapsulation mechanism (KEM)**
  - Alice combines *Message* + *Secret Key* to create *Digest* by **Hashing**
  - Bob verifies by calculating **Hash** of *Message* + *Secret Key*
    - *Message* was not modified in transit ------ **Integrity**
    - Alice had the identical *Secret Key* ------ **Authentication**



https://www.youtube.com/watch?v=doN3lzzNEIM&t=96s

Cryptographic hash algorithm

- ❑ IoT security highly relies on data integrity to authenticate identity
- ❑ In engineering, cryptographic hash algorithm is adopted for data integrity
- ❑ **Practically infeasible to invert or reverse the hash computation**

Cryptographic hash algorithm

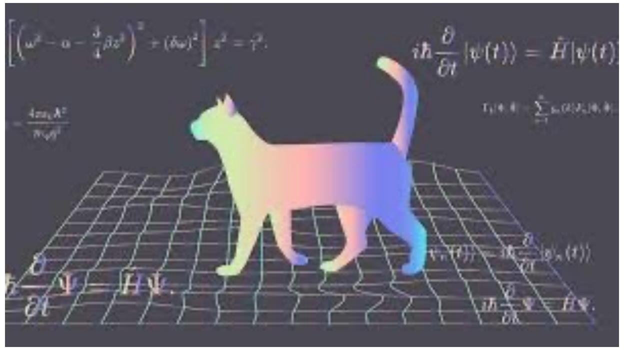
- IoT security highly relies on data integrity to authenticate identity
- In engineering, cryptographic hash algorithm is adopted for data integrity
- **Practically infeasible to invert or reverse the hash computation**

Transport Layer Security
in IoT (Amazon IoT Core)

Cryptographic hash algorithm

- IoT security highly relies on data integrity to authenticate identity
- In engineering, cryptographic hash algorithm is adopted for data integrity
- **Practically infeasible to invert or reverse the hash computation**

Transport Layer Security
in IoT (Amazon IoT Core)



Quantum-resistant TLS
in IoT

More challenges in IoT Era

More challenges in IoT Era

- Attackers can effortlessly obtain physical access to edge devices

More challenges in IoT Era

- ❑ Attackers can effortlessly obtain physical access to edge devices
- ❑ Hardware resources are highly constrained in IoT devices

More challenges in IoT Era

- Attackers can effortlessly obtain physical access to edge devices
- Hardware resources are highly constrained in IoT devices
- Performance matters especially in internet of vehicles

More challenges in IoT Era

- ❑ Attackers can effortlessly obtain physical access to edge devices
- ❑ Hardware resources are highly constrained in IoT devices
- ❑ Performance matters especially in internet of vehicles
- ❑ Energy consumption matters since IoT is powered by battery

More challenges in IoT Era

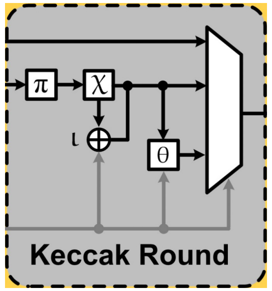
- ❑ Attackers can effortlessly obtain physical access to edge devices
- ❑ Hardware resources are highly constrained in IoT devices
- ❑ Performance matters especially in internet of vehicles
- ❑ Energy consumption matters since IoT is powered by battery

Demand for **low-latency**, **high-throughput** and **energy-efficient** hashing in IoT devices

△ Challenges: Performance, Energy, Area

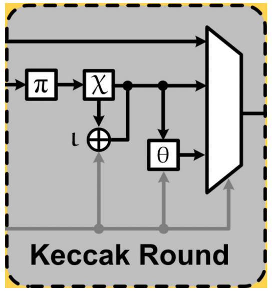
△ Challenges: Performance, Energy, Area

- Dedicated hardware engine on chip (ISSCC'16)



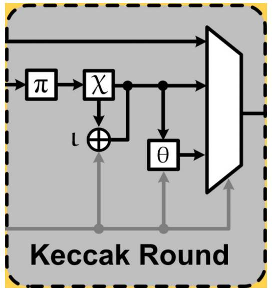
△ Challenges: Performance, Energy, Area

- **Dedicated hardware engine on chip (ISSCC’16)**
  - Low throughput



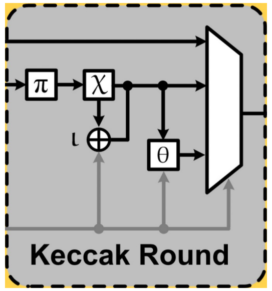
△ Challenges: Performance, Energy, Area

- **Dedicated hardware engine on chip (ISSCC'16)**
  - Low throughput
  - High area overhead on chip



△ Challenges: Performance, Energy, Area

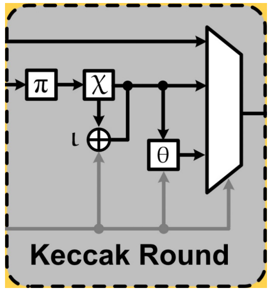
- ❑ **Dedicated hardware engine on chip (ISSCC’16)**
  - ○ Low throughput
  - ○ High area overhead on chip



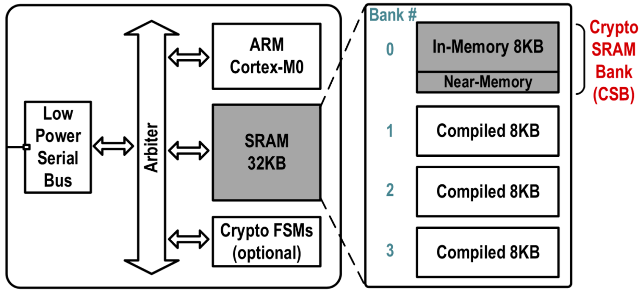
- ❑ **General-purpose in-memory acceleration (JSSC’18)**

△ Challenges: Performance, Energy, Area

- ❑ **Dedicated hardware engine on chip (ISSCC'16)**
  - ○ Low throughput
  - ○ High area overhead on chip

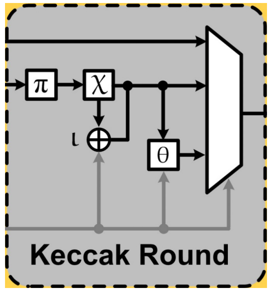


- ❑ **General-purpose in-memory acceleration (JSSC'18)**
  - ○ High latency

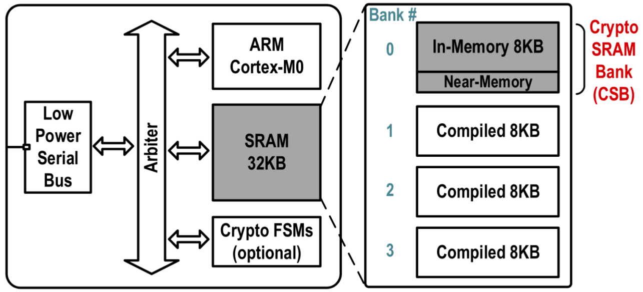


△ Challenges: Performance, Energy, Area

- **Dedicated hardware engine on chip (ISSCC'16)**
  - Low throughput
  - High area overhead on chip

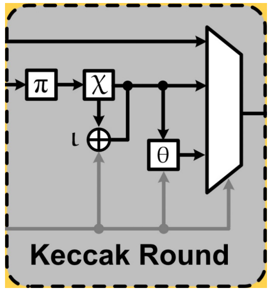


- **General-purpose in-memory acceleration (JSSC'18)**
  - High latency
  - Low throughput per unit area

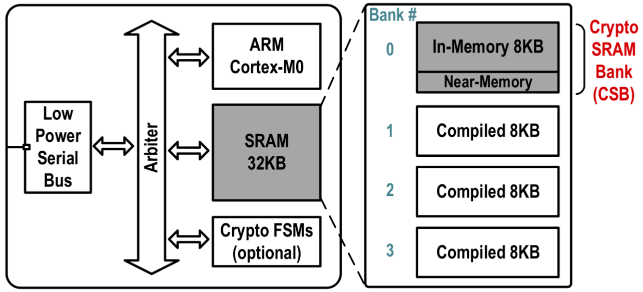


△ Challenges: Performance, Energy, Area

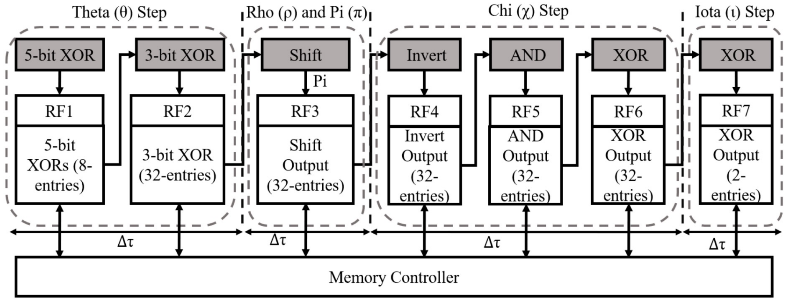
- ❑ **Dedicated hardware engine on chip (ISSCC'16)**
  - ○ Low throughput
  - ○ High area overhead on chip



- ❑ **General-purpose in-memory acceleration (JSSC'18)**
  - ○ High latency
  - ○ Low throughput per unit area

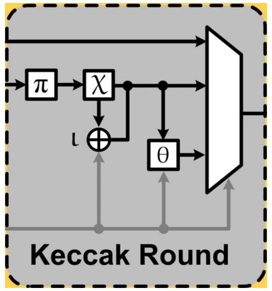


- ❑ **Dedicated in-memory acceleration (ISLPED'19)**

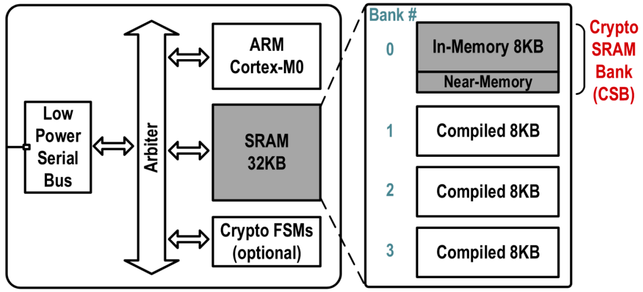


△ Challenges: Performance, Energy, Area

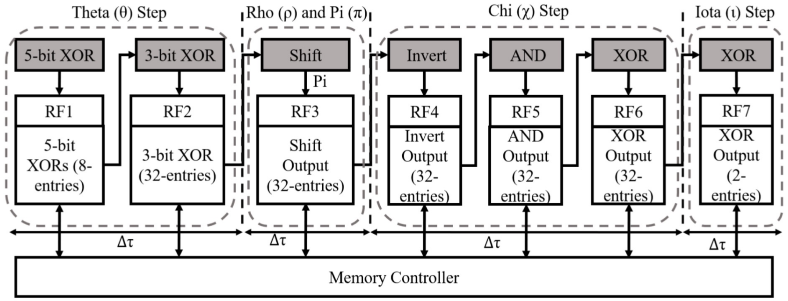
- **Dedicated hardware engine on chip (ISSCC’16)**
  - Low throughput
  - High area overhead on chip



- **General-purpose in-memory acceleration (JSSC’18)**
  - High latency
  - Low throughput per unit area

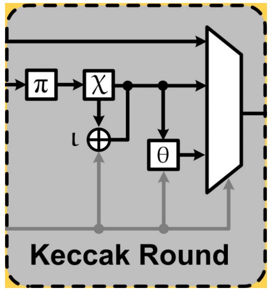


- **Dedicated in-memory acceleration (ISLPED’19)**
  - High area overhead

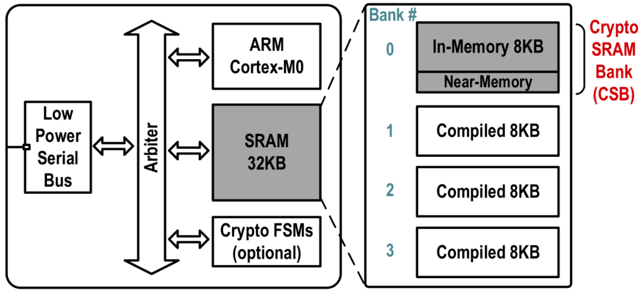


△ Challenges: Performance, Energy, Area

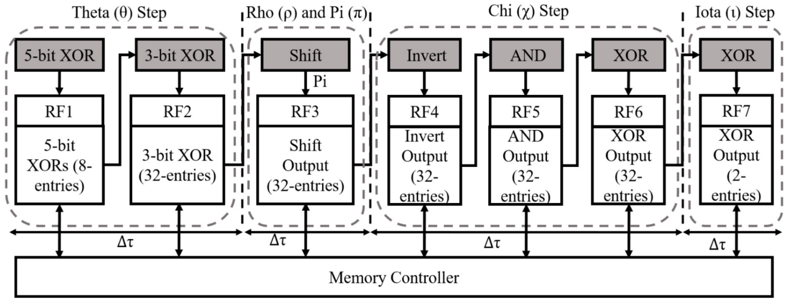
- ❑ **Dedicated hardware engine on chip (ISSCC'16)**
  - Low throughput
  - High area overhead on chip



- ❑ **General-purpose in-memory acceleration (JSSC'18)**
  - High latency
  - Low throughput per unit area

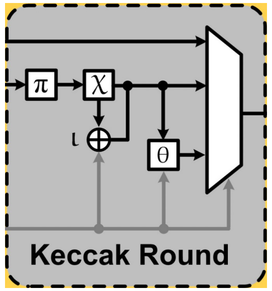


- ❑ **Dedicated in-memory acceleration (ISLPED'19)**
  - High area overhead
  - Low generality



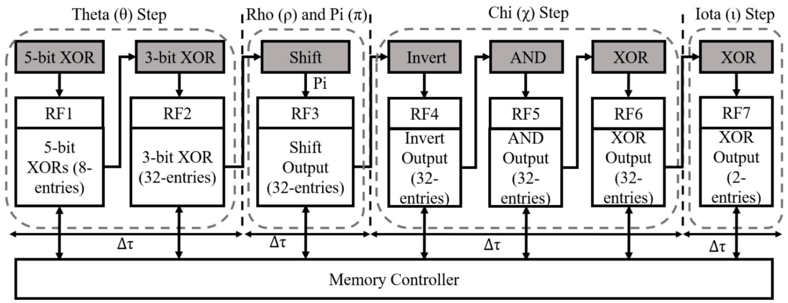
△ Challenges: Performance, Energy, Area

- **Dedicated hardware engine on chip (ISSCC’16)**
  - Low throughput
  - High area overhead on chip



Demand for low-latency, high-throughput, energy-efficient, low-overhead hashing in IoT

- **Dedicated in-memory acceleration (ISLPED’19)**
  - High area overhead
  - Low generality



Overview of Our Solution: *Inhale*

Overview of Our Solution: *Inhale*

- ❑ **On-chip Hashing**
  - ○ Perform all the operations within the chip (trusted computing base)

Overview of Our Solution: *Inhale*

- **On-chip Hashing -> high security level**
  - Perform all the operations within the chip (trusted computing base)

Overview of Our Solution: *Inhale*

- ❑ **On-chip Hashing -> high security level**
  - ○ Perform all the operations within the chip (trusted computing base)

- ❑ **Bitline Computing**
  - ○ Repurpose SRAM subarrays into active large vector computation units

Overview of Our Solution: *Inhale*

- **On-chip Hashing -> high security level**
  - Perform all the operations within the chip (trusted computing base)

- **Bitline Computing -> high throughput**
  - Repurpose SRAM subarrays into active large vector computation units

Overview of Our Solution: *Inhale*

- ❑ **On-chip Hashing -> high security level**
  - ○ Perform all the operations within the chip (trusted computing base)

- ❑ **Bitline Computing -> high throughput**
  - ○ Repurpose SRAM subarrays into active large vector computation units

- ❑ **Shift-optimized Data Alignment**
  - ○ Implicitly perform inter-lane shift operations via the controller

Overview of Our Solution: *Inhale*

- ❑ **On-chip Hashing -> high security level**
  - ○ Perform all the operations within the chip (trusted computing base)

- ❑ **Bitline Computing -> high throughput**
  - ○ Repurpose SRAM subarrays into active large vector computation units

- ❑ **Shift-optimized Data Alignment -> low latency, energy**
  - ○ Implicitly perform inter-lane shift operations via the controller

Overview of Our Solution: *Inhale*

- ❑ **On-chip Hashing -> high security level**
  - ○ Perform all the operations within the chip (trusted computing base)

- ❑ **Bitline Computing -> high throughput**
  - ○ Repurpose SRAM subarrays into active large vector computation units

- ❑ **Shift-optimized Data Alignment -> low latency, energy**
  - ○ Implicitly perform inter-lane shift operations via the controller

- ❑ **In-Place Read/Write Strategy**
  - ○ Carefully design read/write order and address to save memory capacity

Overview of Our Solution: *Inhale*

- ❑ **On-chip Hashing -> high security level**
  - ○ Perform all the operations within the chip (trusted computing base)

- ❑ **Bitline Computing -> high throughput**
  - ○ Repurpose SRAM subarrays into active large vector computation units

- ❑ **Shift-optimized Data Alignment -> low latency, energy**
  - ○ Implicitly perform inter-lane shift operations via the controller

- ❑ **In-Place Read/Write Strategy -> low overhead**
  - ○ Carefully design read/write order and address to save memory capacity

Overview of Our Solution: *Inhale*

- **On-chip Hashing -> high security level**
  - Perform all the operations within the chip (trusted computing base)

Inhale can achieve **up to 14x** throughput-per-area, **172x** throughput-per-area-per-energy than state-of-the-art

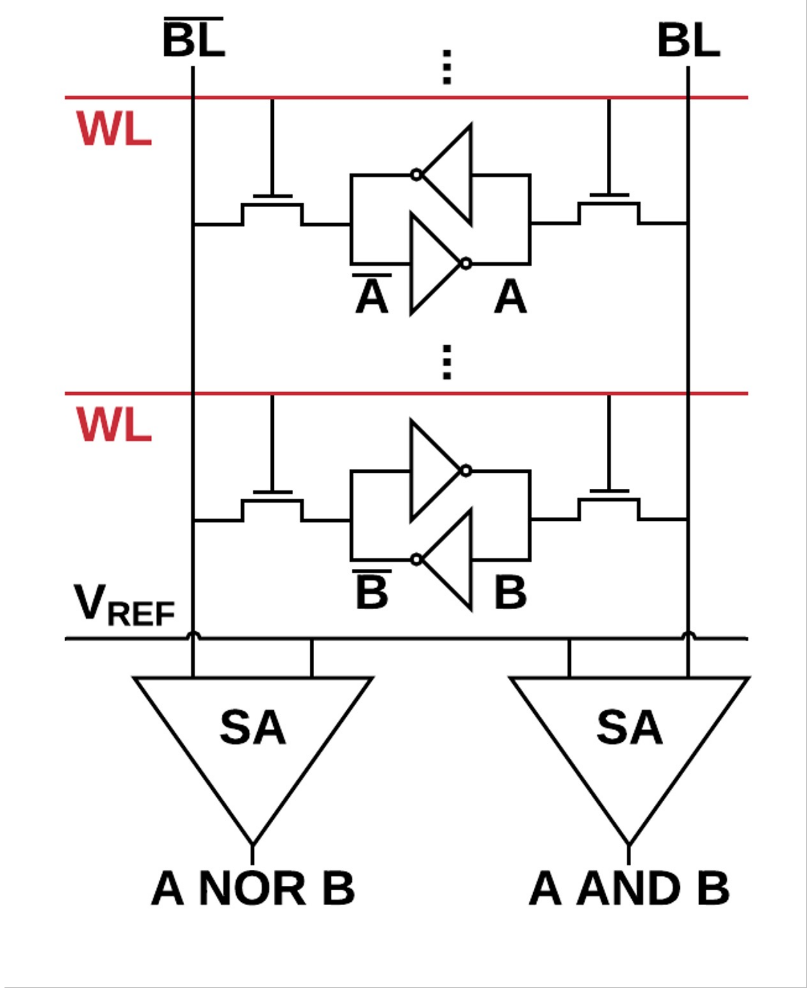
- **In-Place Read/Write Strategy -> low overhead**
  - Carefully design read/write order and address to save memory capacity

Bitline Computing used in *Inhale*

[1] Aga, Shaizeen, et al. “Compute caches.” HPCA 2017

Bitline Computing used in *Inhale*

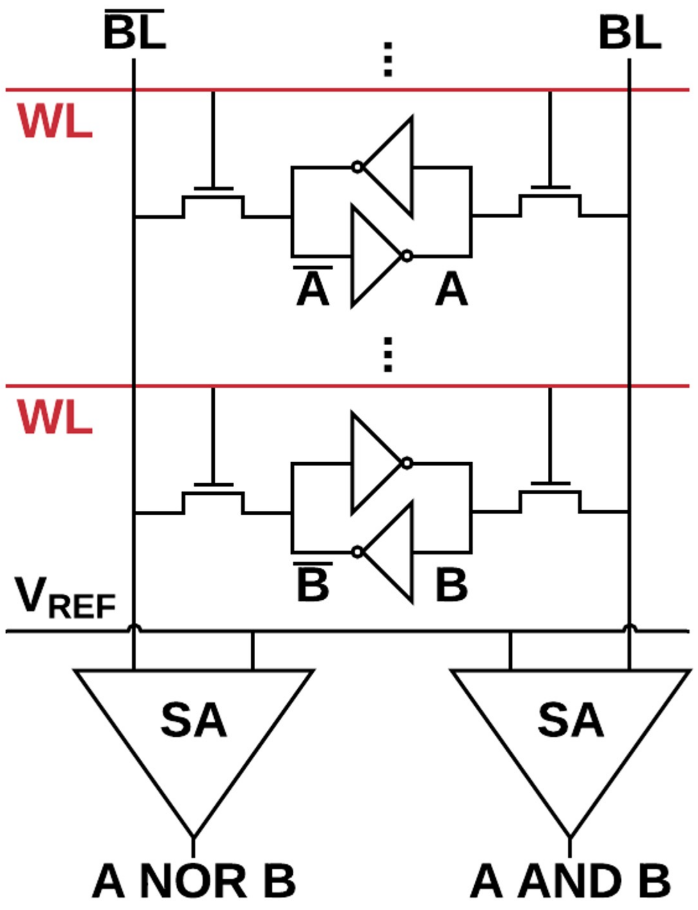
- **Bitline Computing [1]**



[1] Aga, Shaizeen, et al. "Compute caches." HPCA 2017

Bitline Computing used in *Inhale*

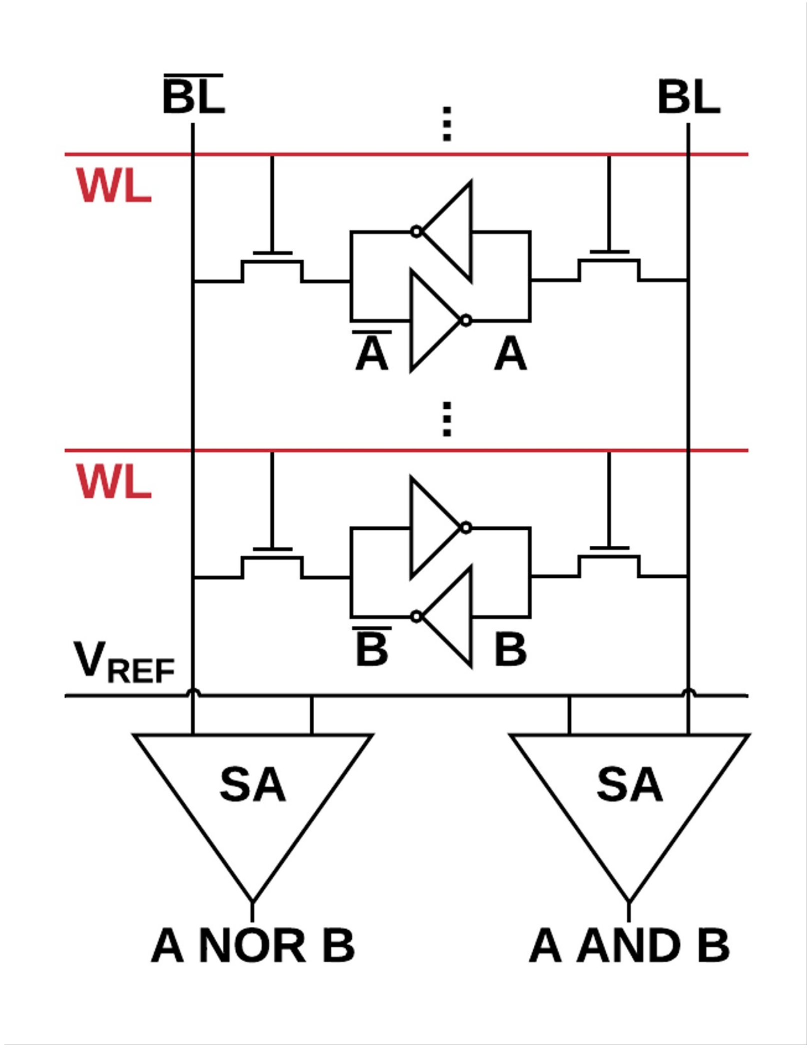
- **Bitline Computing [1]**
  - Activate two wordlines simultaneously



[1] Aga, Shaizeen, et al. "Compute caches." HPCA 2017

Bitline Computing used in *Inhale*

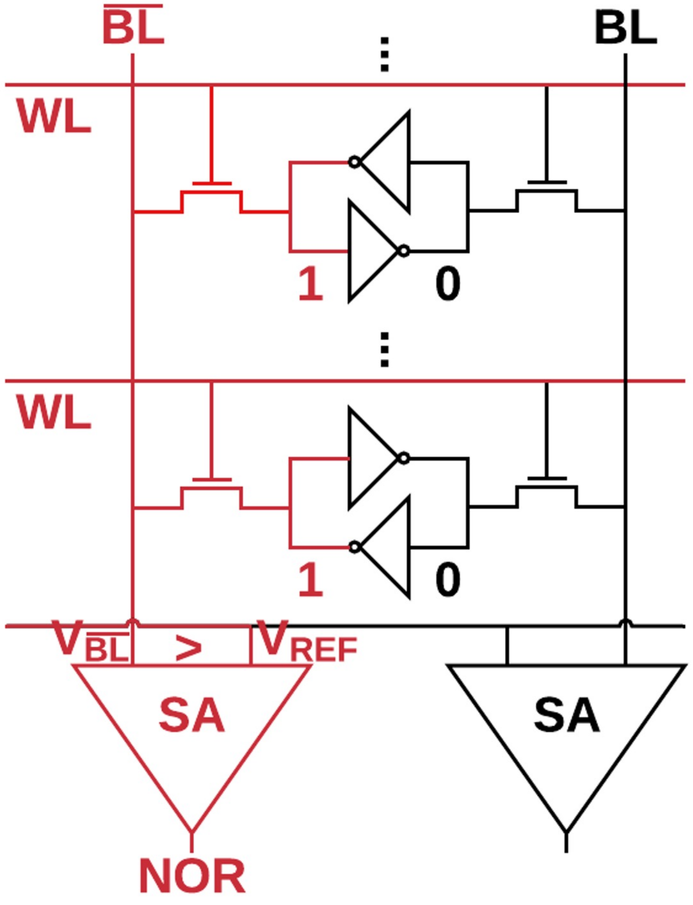
- **Bitline Computing [1]**
  - Activate two wordlines simultaneously
  - Inherently perform logic operations



[1] Aga, Shaizeen, et al. “Compute caches.” HPCA 2017

Bitline Computing used in *Inhale*

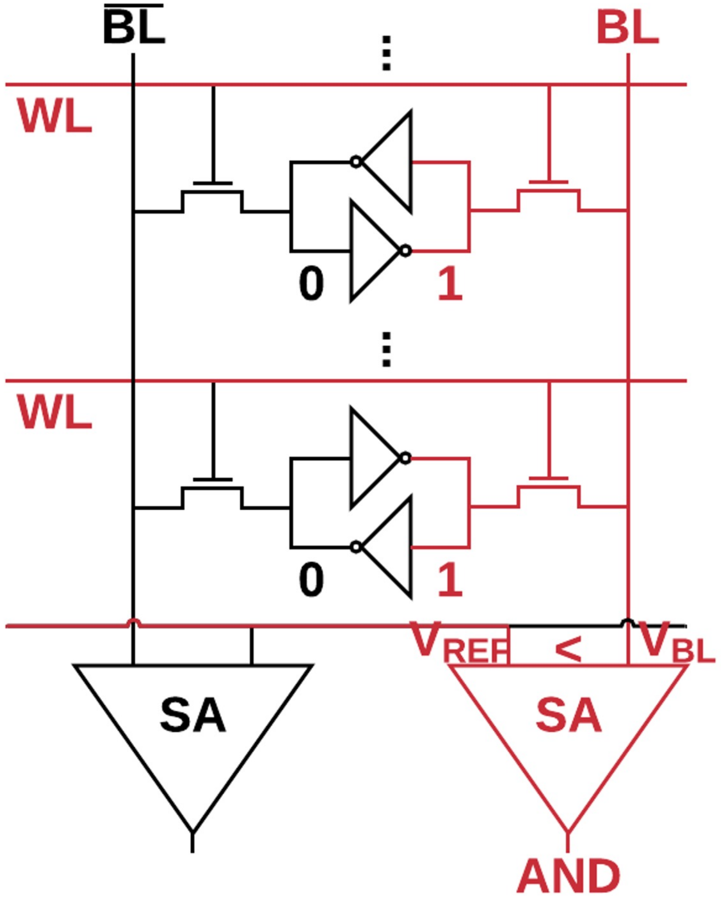
- **Bitline Computing [1]**
  - Activate two wordlines simultaneously
  - Inherently perform logic operations
    - NOR



[1] Aga, Shaizeen, et al. "Compute caches." HPCA 2017

Bitline Computing used in *Inhale*

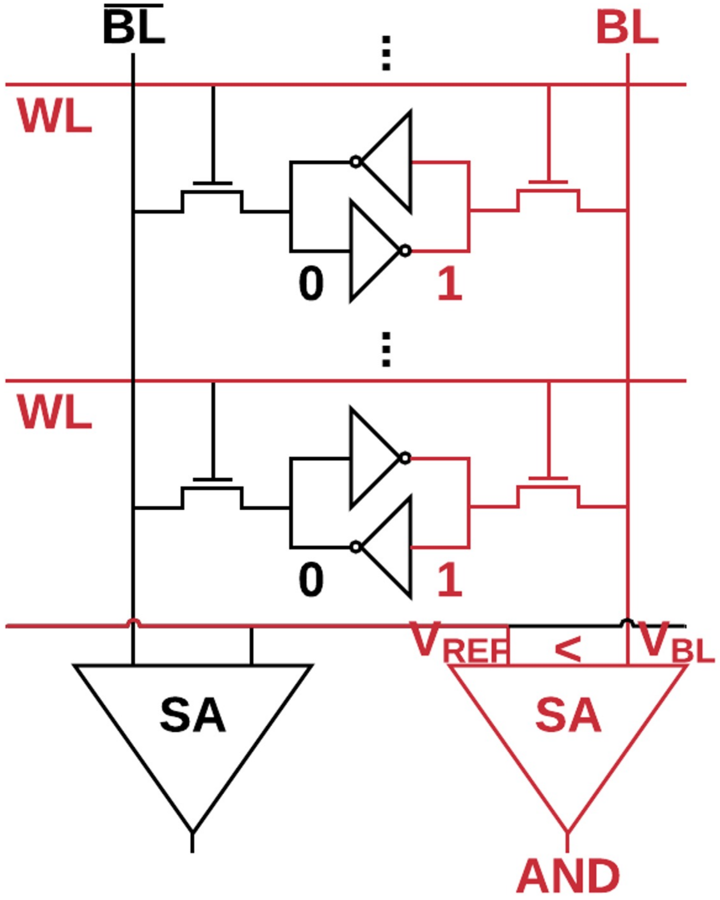
- **Bitline Computing [1]**
  - Activate two wordlines simultaneously
  - Inherently perform logic operations
    - NOR
    - AND



[1] Aga, Shaizeen, et al. "Compute caches." HPCA 2017

Bitline Computing used in *Inhale*

- **Bitline Computing [1]**
  - Activate two wordlines simultaneously
  - Inherently perform logic operations
    - NOR
    - AND
  - Additionally support other logic operations

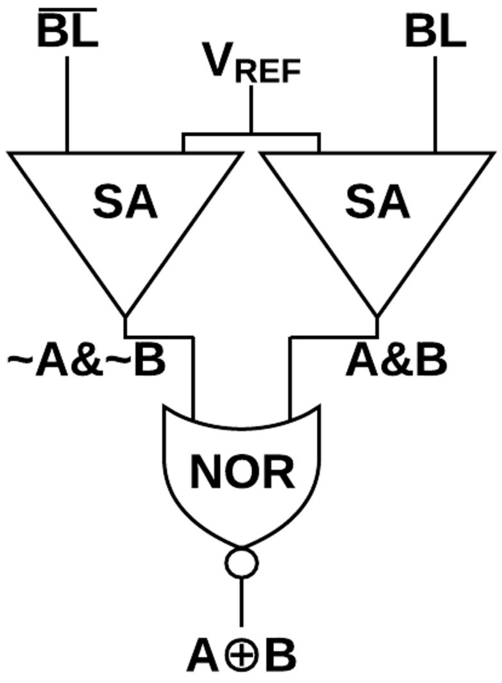


[1] Aga, Shaizeen, et al. "Compute caches." HPCA 2017

Bitline Computing used in *Inhale*

❑ Bitline Computing [1]

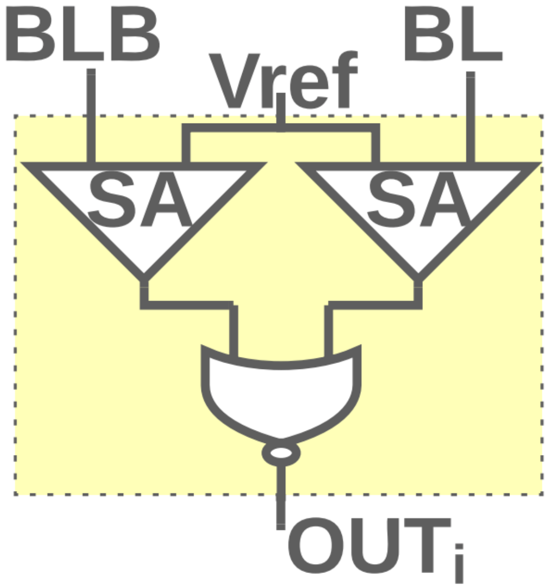
- Activate two wordlines simultaneously
- Inherently perform logic operations
  - NOR
  - AND
- Additionally support other logic operations
  - XOR



[1] Aga, Shaizeen, et al. “Compute caches.” HPCA 2017

Bitline Computing used in *Inhale*

- **Bitline Computing [1]**
  - Activate two wordlines simultaneously
  - Inherently perform logic operations
    - NOR
    - AND
  - Additionally support other logic operations
    - XOR

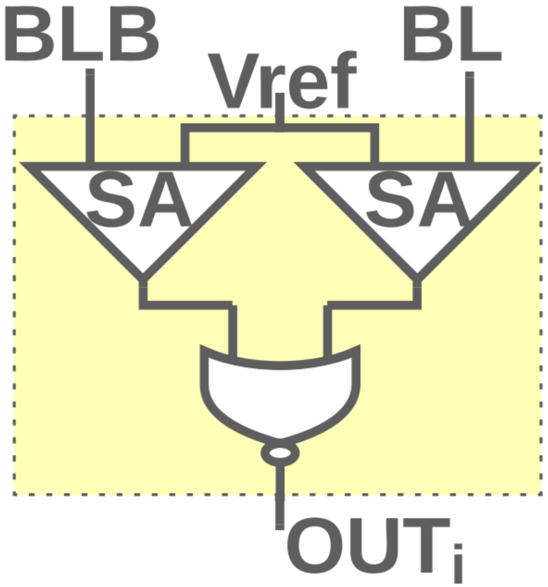


Sense Amplifier Design

[1] Aga, Shaizeen, et al. "Compute caches." HPCA 2017

Bitline Computing used in *Inhale*

- **Bitline Computing [1]**
  - Activate two wordlines simultaneously
  - Inherently perform logic operations
    - NOR
    - AND
  - Additionally support other logic operations
    - XOR
  - Provide high parallelism



Sense Amplifier Design

[1] Aga, Shaizeen, et al. “Compute caches.” HPCA 2017

Motivation for shift-optimization

Motivation for shift-optimization

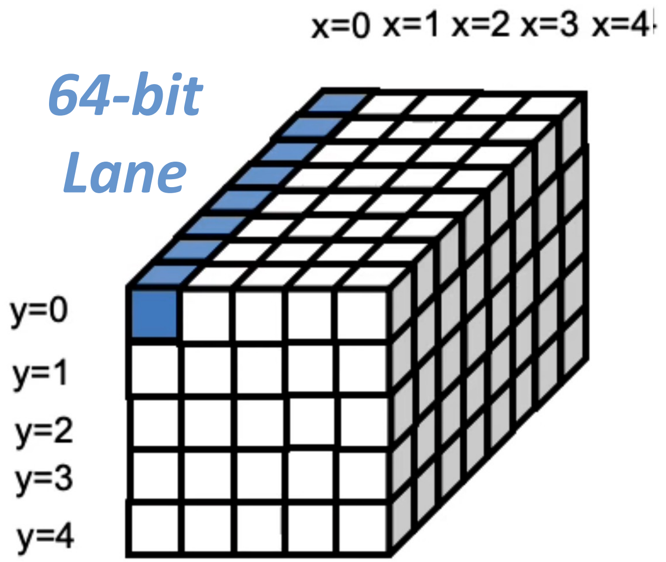
- 76% operations of SHA-3 are shifting in a vanilla PIM architecture

Motivation for shift-optimization

- 76% operations of SHA-3 are shifting in a vanilla PIM architecture
- 90% shifting operations are inter-lane

Motivation for shift-optimization

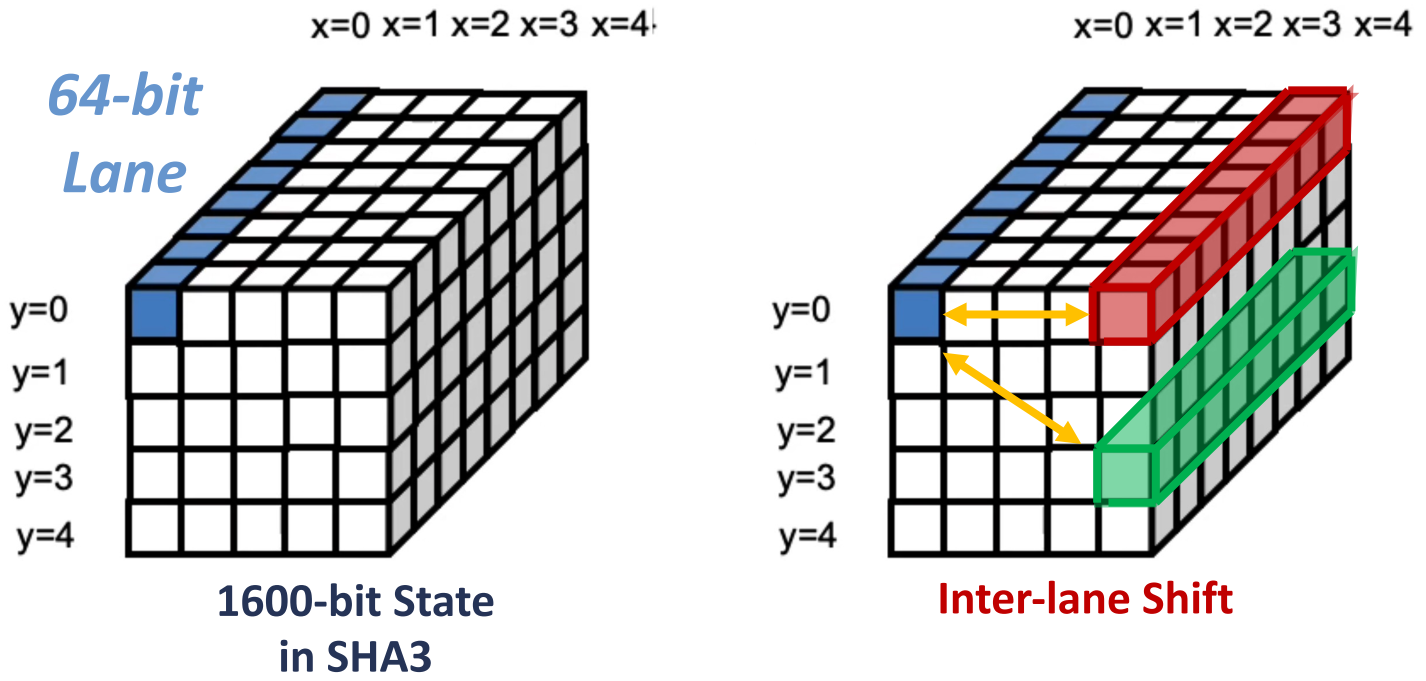
- 76% operations of SHA-3 are shifting in a vanilla PIM architecture
- 90% shifting operations are inter-lane



1600-bit State in SHA3

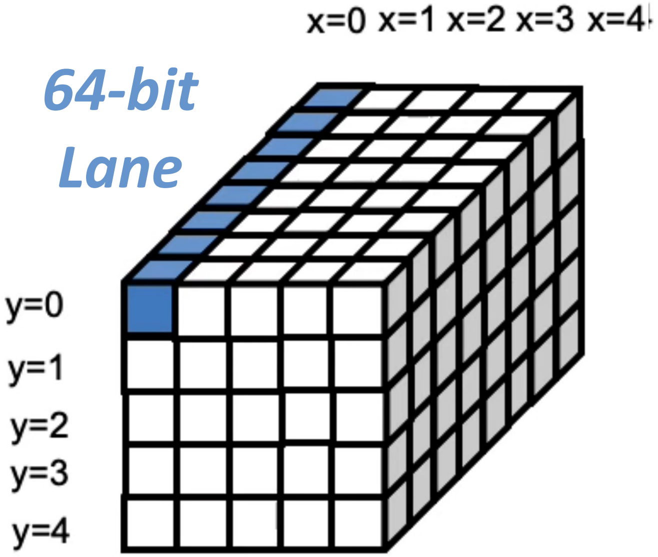
Motivation for shift-optimization

- 76% operations of SHA-3 are shifting in a vanilla PIM architecture
- 90% shifting operations are inter-lane

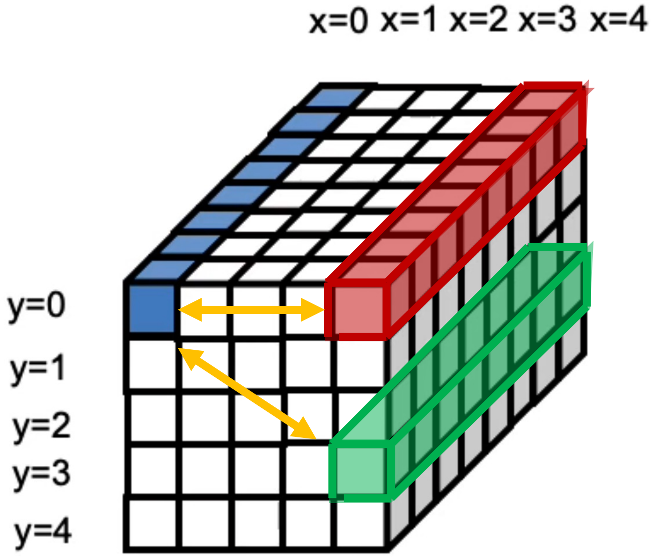


Motivation for shift-optimization

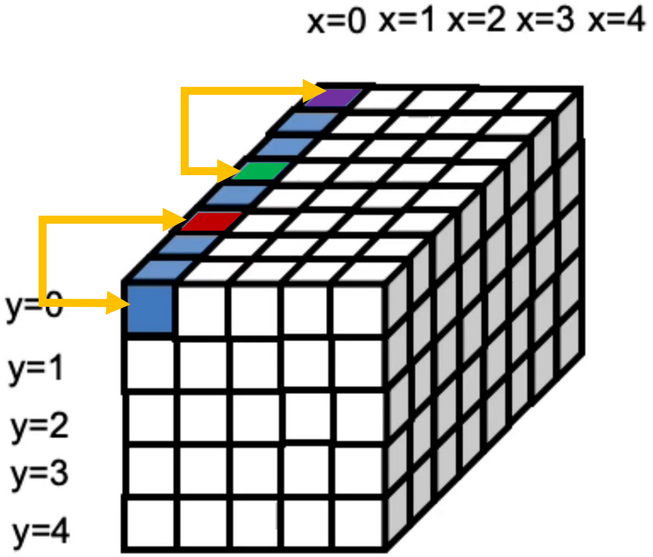
- 76% operations of SHA-3 are shifting in a vanilla PIM architecture
- 90% shifting operations are inter-lane



1600-bit State in SHA3



Inter-lane Shift



Intra-lane Shift

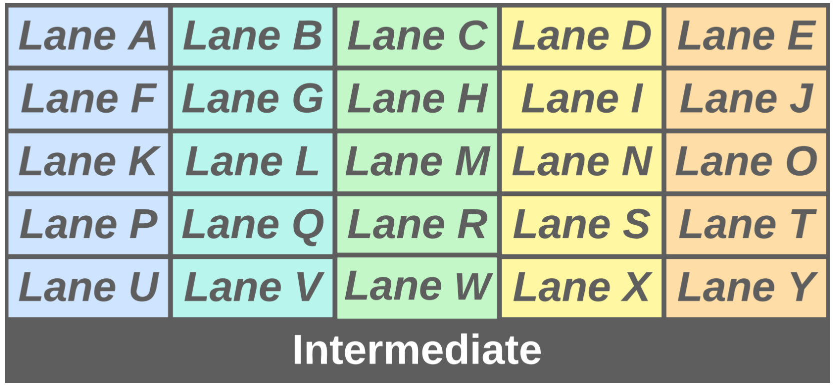
Prior works

Prior works

- **Existing Data Alignments**

Prior works

- ❑ **Existing Data Alignments**
  - ○ JSSC’18:



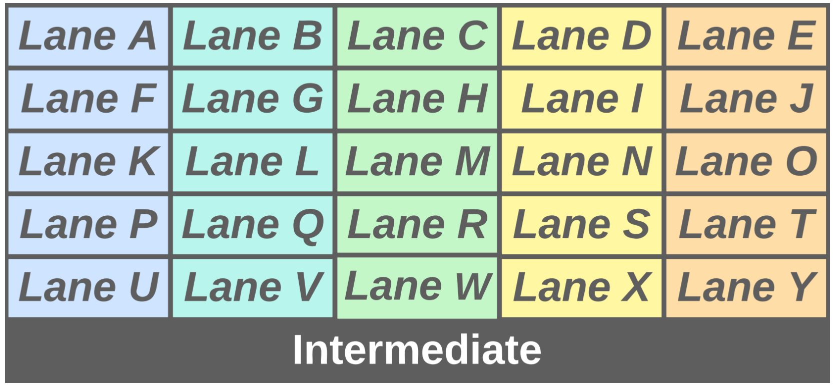
Lane A	Lane B	Lane C	Lane D	Lane E
Lane F	Lane G	Lane H	Lane I	Lane J
Lane K	Lane L	Lane M	Lane N	Lane O
Lane P	Lane Q	Lane R	Lane S	Lane T
Lane U	Lane V	Lane W	Lane X	Lane Y
Intermediate				

SRAM subarray (JSSC’18)

Prior works

❑ Existing Data Alignments

- JSSC'18:
  - highly utilize the parallelism



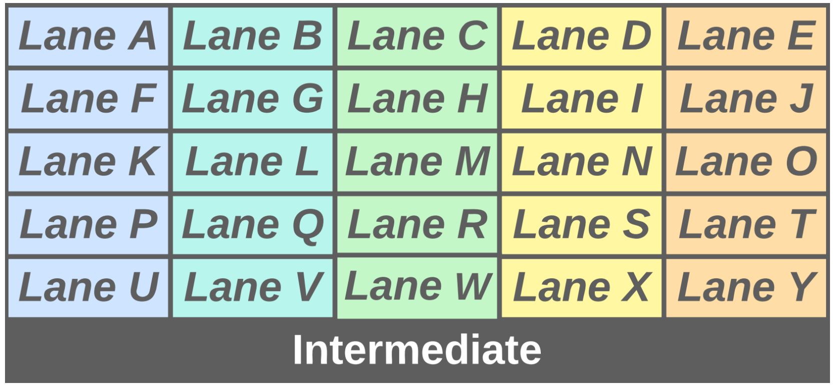
Lane A	Lane B	Lane C	Lane D	Lane E
Lane F	Lane G	Lane H	Lane I	Lane J
Lane K	Lane L	Lane M	Lane N	Lane O
Lane P	Lane Q	Lane R	Lane S	Lane T
Lane U	Lane V	Lane W	Lane X	Lane Y

Intermediate

SRAM subarray (JSSC'18)

Prior works

- ❑ **Existing Data Alignments**
  - JSSC’18:
    - highly utilize the parallelism
    - hard for inter-lane and intra-lane shift



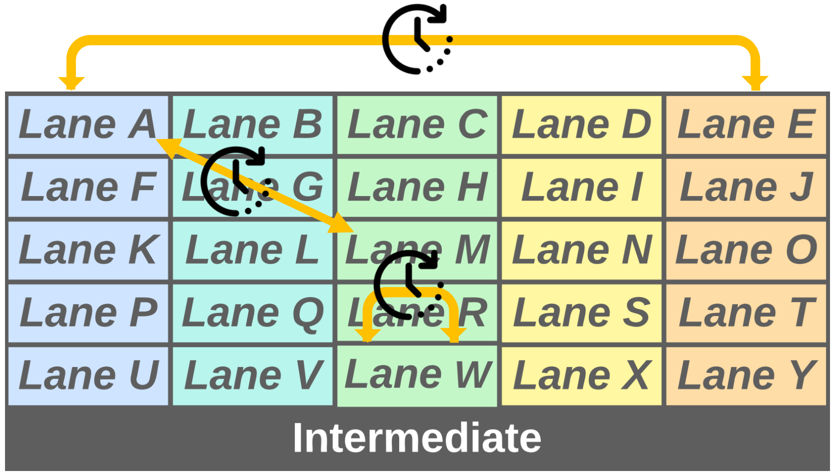
Lane A	Lane B	Lane C	Lane D	Lane E
Lane F	Lane G	Lane H	Lane I	Lane J
Lane K	Lane L	Lane M	Lane N	Lane O
Lane P	Lane Q	Lane R	Lane S	Lane T
Lane U	Lane V	Lane W	Lane X	Lane Y
Intermediate				

SRAM subarray (JSSC’18)

Prior works

❑ Existing Data Alignments

- JSSC’18:
  - highly utilize the parallelism
  - hard for inter-lane and intra-lane shift



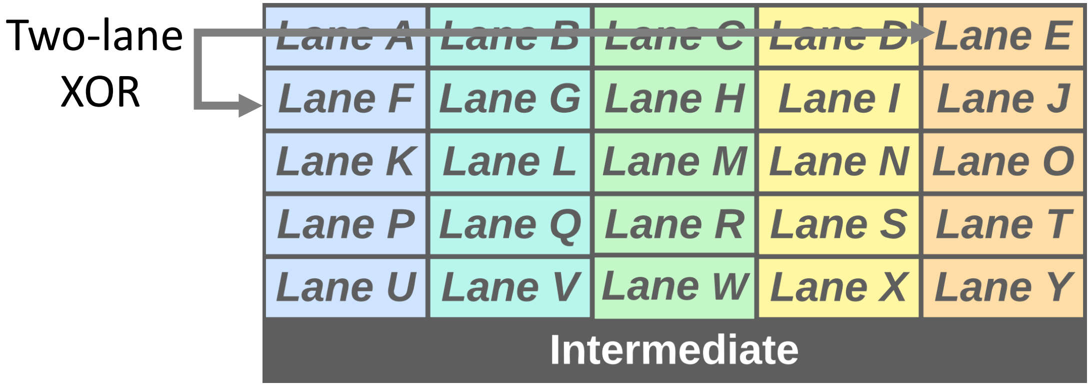
Lane A	Lane B	Lane C	Lane D	Lane E
Lane F	Lane G	Lane H	Lane I	Lane J
Lane K	Lane L	Lane M	Lane N	Lane O
Lane P	Lane Q	Lane R	Lane S	Lane T
Lane U	Lane V	Lane W	Lane X	Lane Y

Intermediate

SRAM subarray (JSSC’18)

Prior works

- ❑ **Existing Data Alignments**
  - JSSC’18:
    - highly utilize the parallelism
    - hard for inter-lane and intra-lane shift

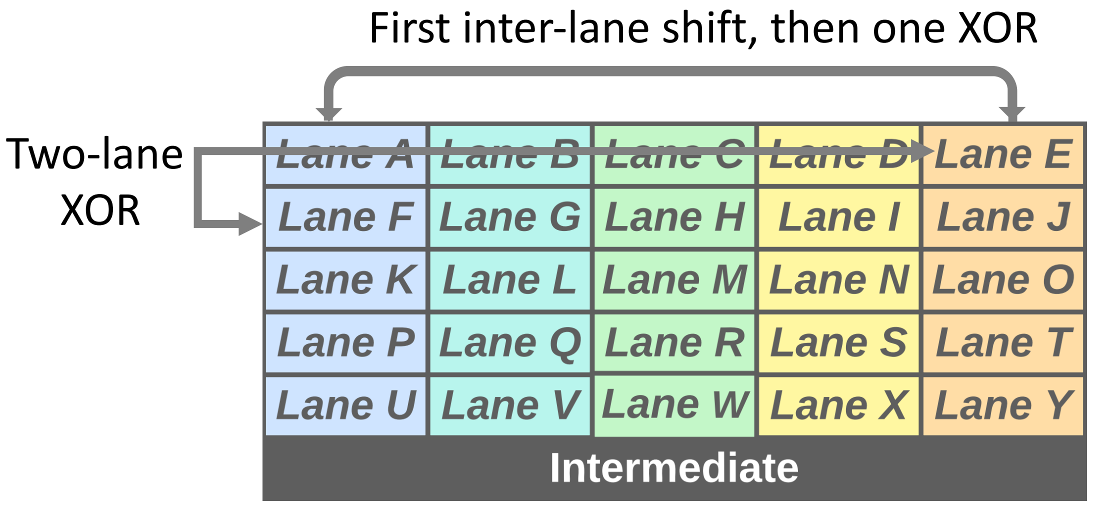


Lane A	Lane B	Lane C	Lane D	Lane E
Lane F	Lane G	Lane H	Lane I	Lane J
Lane K	Lane L	Lane M	Lane N	Lane O
Lane P	Lane Q	Lane R	Lane S	Lane T
Lane U	Lane V	Lane W	Lane X	Lane Y
Intermediate				

SRAM subarray (JSSC’18)

Prior works

- **Existing Data Alignments**
  - JSSC’18:
    - highly utilize the parallelism
    - hard for inter-lane and intra-lane shift

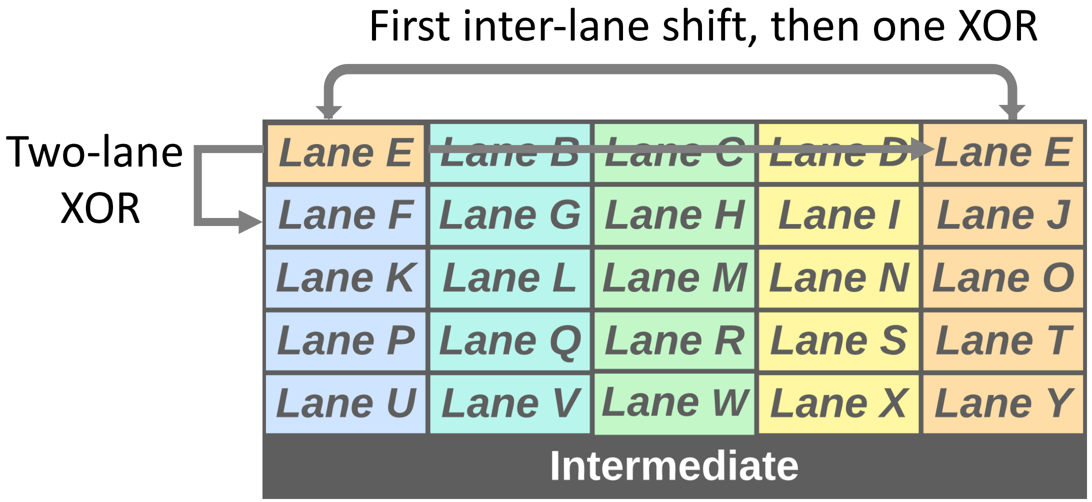


Lane A	Lane B	Lane C	Lane D	Lane E
Lane F	Lane G	Lane H	Lane I	Lane J
Lane K	Lane L	Lane M	Lane N	Lane O
Lane P	Lane Q	Lane R	Lane S	Lane T
Lane U	Lane V	Lane W	Lane X	Lane Y

SRAM subarray (JSSC’18)

Prior works

- **Existing Data Alignments**
  - JSSC’18:
    - highly utilize the parallelism
    - hard for inter-lane and intra-lane shift



Lane E	Lane B	Lane C	Lane D	Lane E
Lane F	Lane G	Lane H	Lane I	Lane J
Lane K	Lane L	Lane M	Lane N	Lane O
Lane P	Lane Q	Lane R	Lane S	Lane T
Lane U	Lane V	Lane W	Lane X	Lane Y

SRAM subarray (JSSC’18)

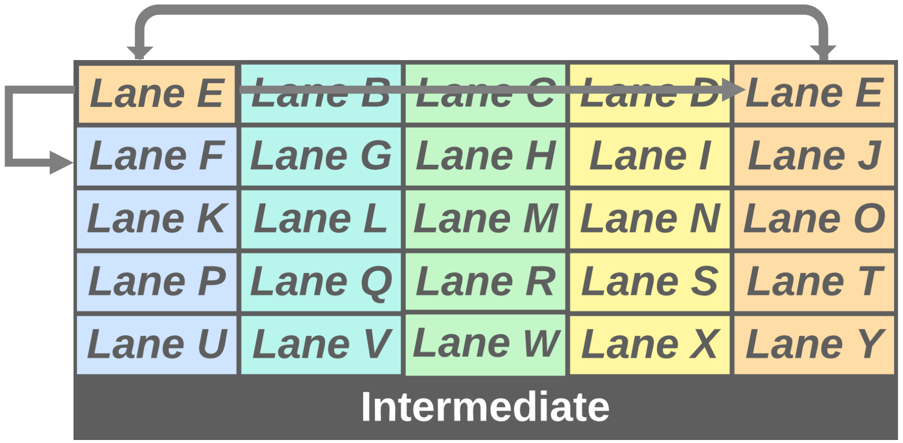
Prior works

❑ Existing Data Alignments

- JSSC’18:
  - highly utilize the parallelism
  - hard for inter-lane and intra-lane shift

First inter-lane shift, then one XOR

One XOR

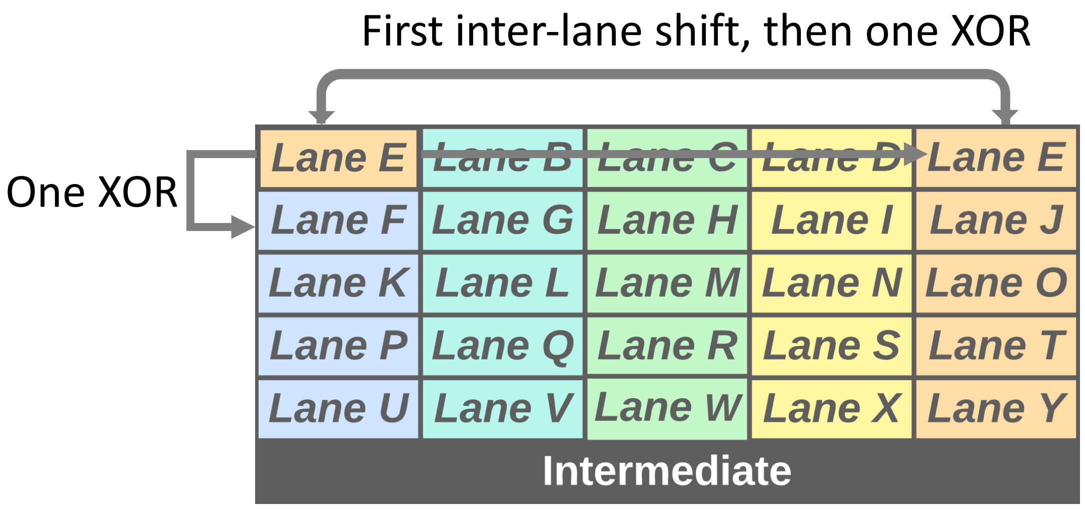


Lane E	Lane B	Lane C	Lane D	Lane E
Lane F	Lane G	Lane H	Lane I	Lane J
Lane K	Lane L	Lane M	Lane N	Lane O
Lane P	Lane Q	Lane R	Lane S	Lane T
Lane U	Lane V	Lane W	Lane X	Lane Y
Intermediate				

SRAM subarray (JSSC’18)

Prior works

- **Existing Data Alignments**
  - JSSC'18:
    - highly utilize the parallelism
    - hard for inter-lane and intra-lane shift
  - ISCA'18:

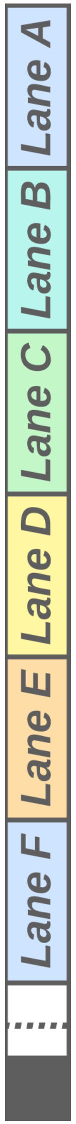


First inter-lane shift, then one XOR

One XOR

Lane E	Lane B	Lane C	Lane D	Lane E
Lane F	Lane G	Lane H	Lane I	Lane J
Lane K	Lane L	Lane M	Lane N	Lane O
Lane P	Lane Q	Lane R	Lane S	Lane T
Lane U	Lane V	Lane W	Lane X	Lane Y
Intermediate				

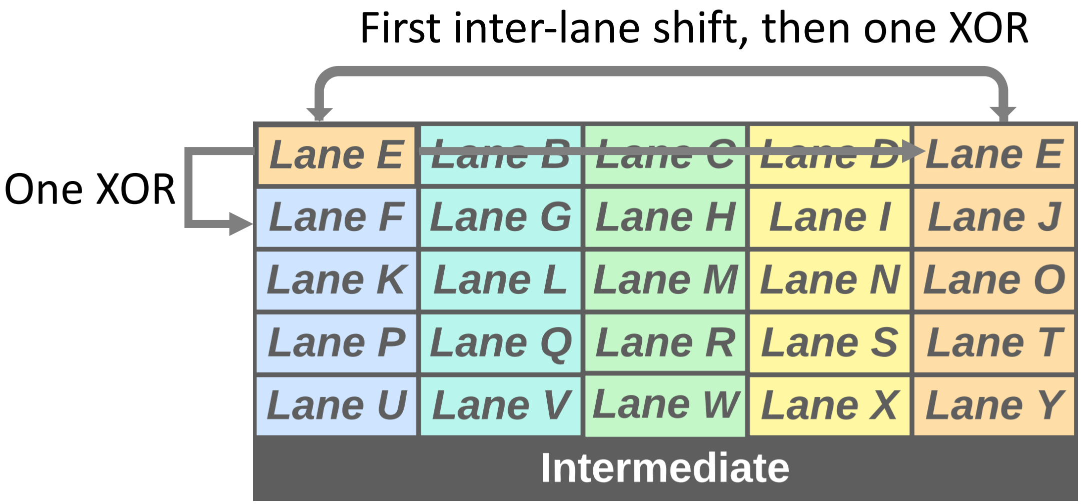
SRAM subarray (JSSC'18)



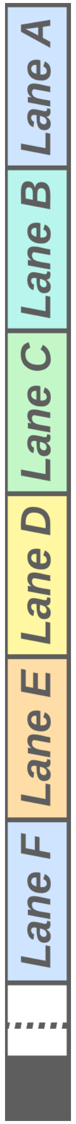
SRAM subarray (ISCA'18)

Prior works

- ❑ **Existing Data Alignments**
  - JSSC’18:
    - highly utilize the parallelism
    - hard for inter-lane and intra-lane shift
  - ISCA’18:
    - shift implicitly



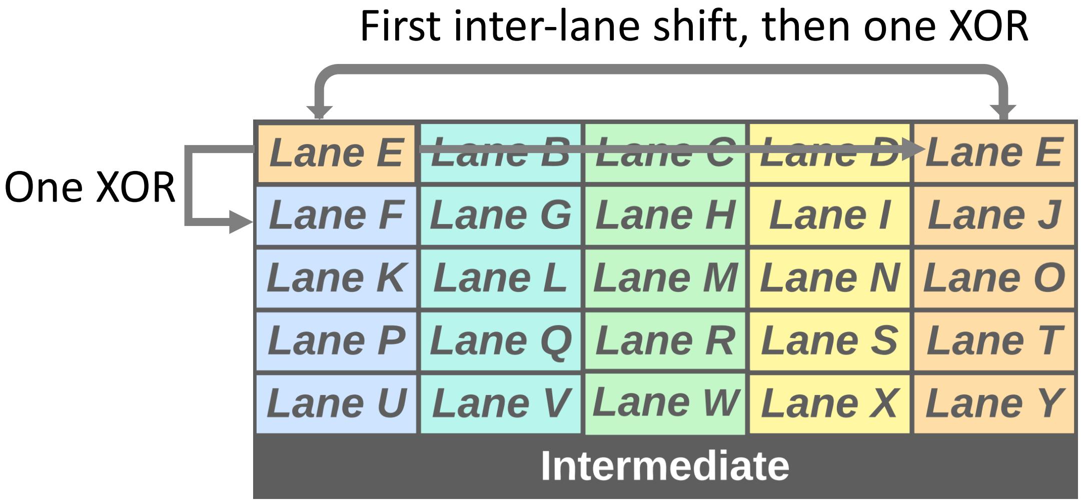
SRAM subarray (JSSC’18)



SRAM subarray (ISCA’18)

Prior works

- **Existing Data Alignments**
  - JSSC’18:
    - highly utilize the parallelism
    - hard for inter-lane and intra-lane shift
  - ISCA’18:
    - shift implicitly



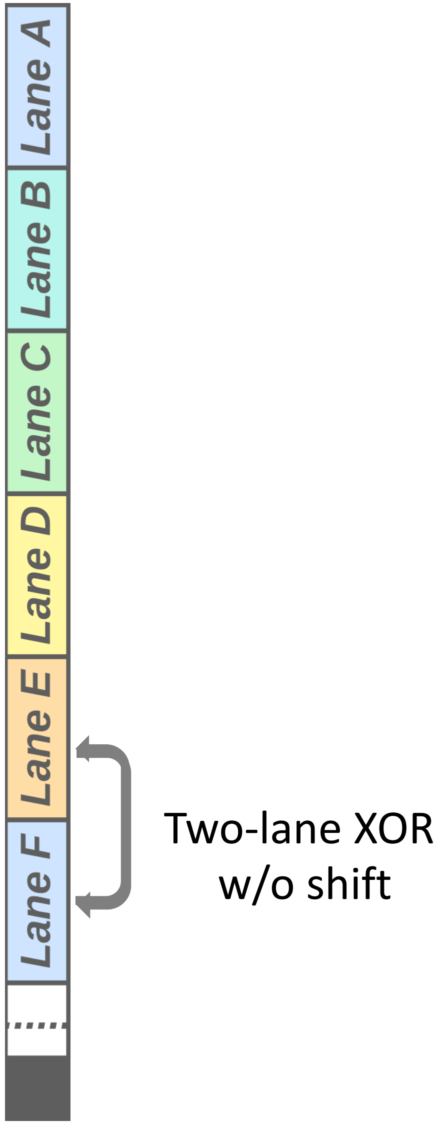
First inter-lane shift, then one XOR

One XOR

Lane E	~~Lane B~~	~~Lane C~~	~~Lane D~~	Lane E
Lane F	Lane G	Lane H	Lane I	Lane J
Lane K	Lane L	Lane M	Lane N	Lane O
Lane P	Lane Q	Lane R	Lane S	Lane T
Lane U	Lane V	Lane W	Lane X	Lane Y

Intermediate

SRAM subarray (JSSC’18)



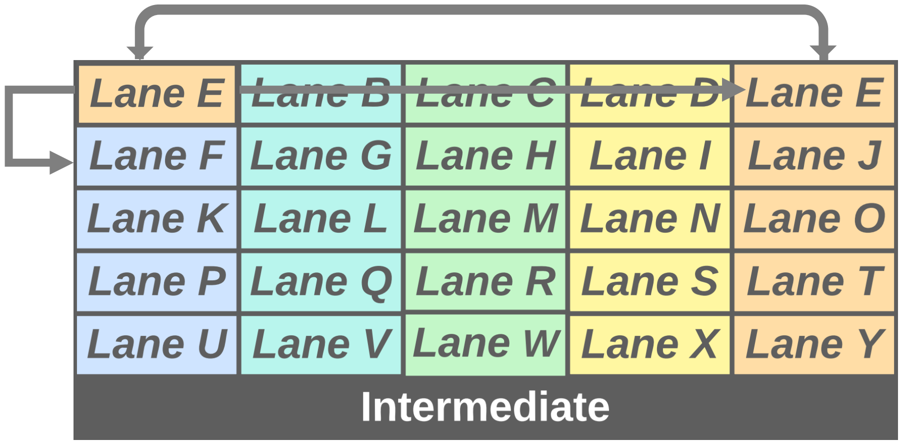
Two-lane XOR w/o shift

SRAM subarray (ISCA’18)

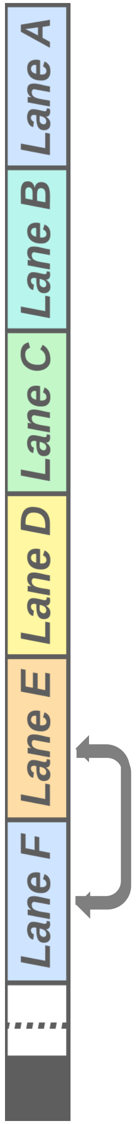
Prior works

❑ Existing Data Alignments

- JSSC’18:
  - highly utilize the parallelism
  - hard for inter-lane and intra-lane shift
- ISCA’18:
  - shift implicitly
  - high latency (>10x JSSC’18)



SRAM subarray (JSSC’18)

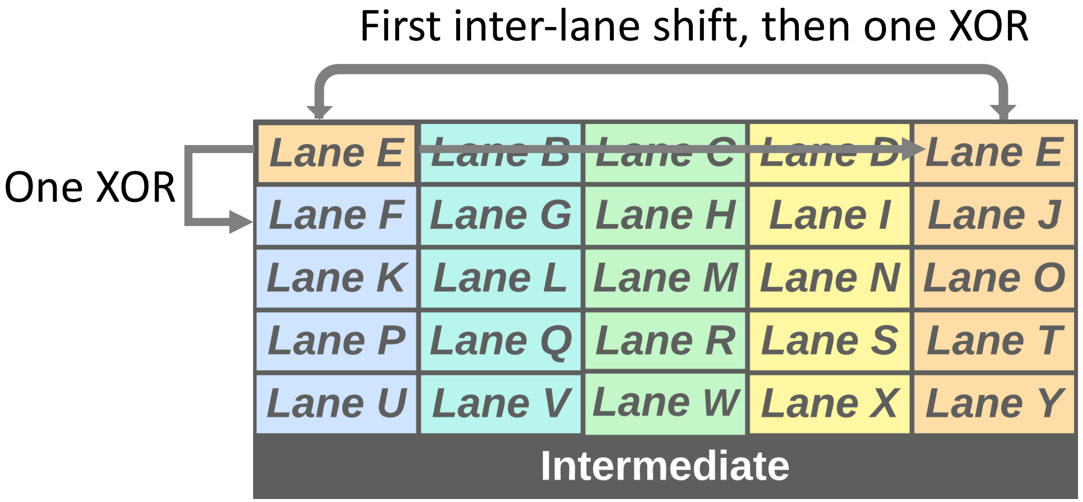


Two-lane XOR w/o shift

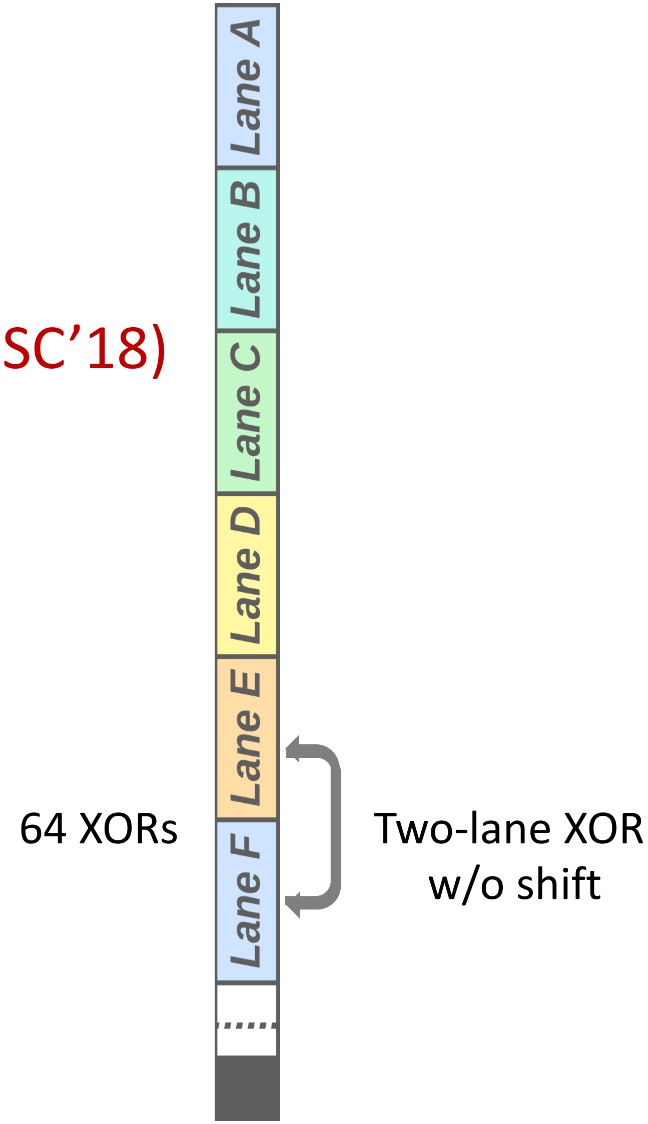
SRAM subarray (ISCA’18)

Prior works

- ❑ **Existing Data Alignments**
  - JSSC’18:
    - highly utilize the parallelism
    - hard for inter-lane and intra-lane shift
  - ISCA’18:
    - shift implicitly
    - high latency (>10x JSSC’18)


First inter-lane shift, then one XOR
One XOR
Lane E | Lane B | Lane C | Lane D | Lane E
Lane F | Lane G | Lane H | Lane I | Lane J
Lane K | Lane L | Lane M | Lane N | Lane O
Lane P | Lane Q | Lane R | Lane S | Lane T
Lane U | Lane V | Lane W | Lane X | Lane Y
Intermediate

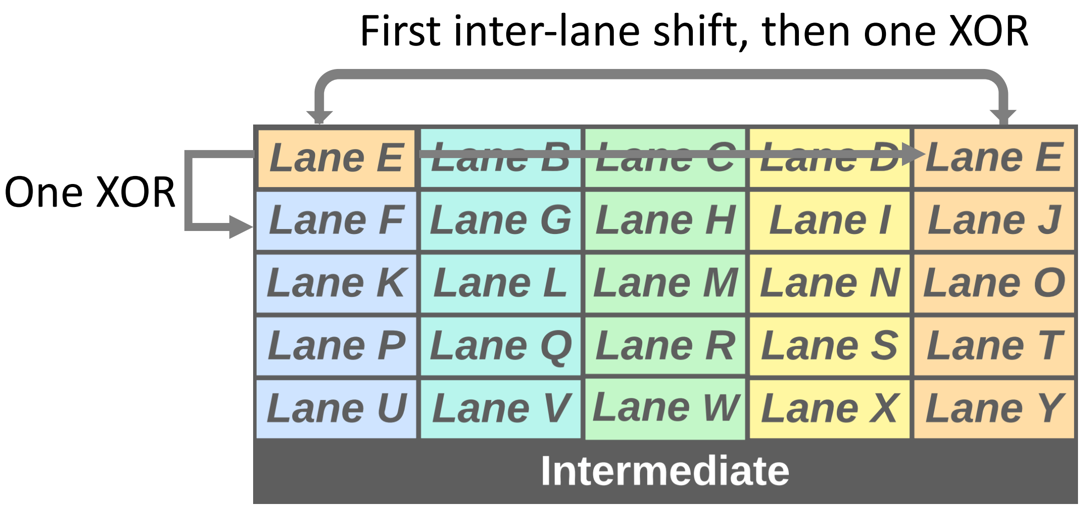
SRAM subarray (JSSC’18)



SRAM subarray (ISCA’18)

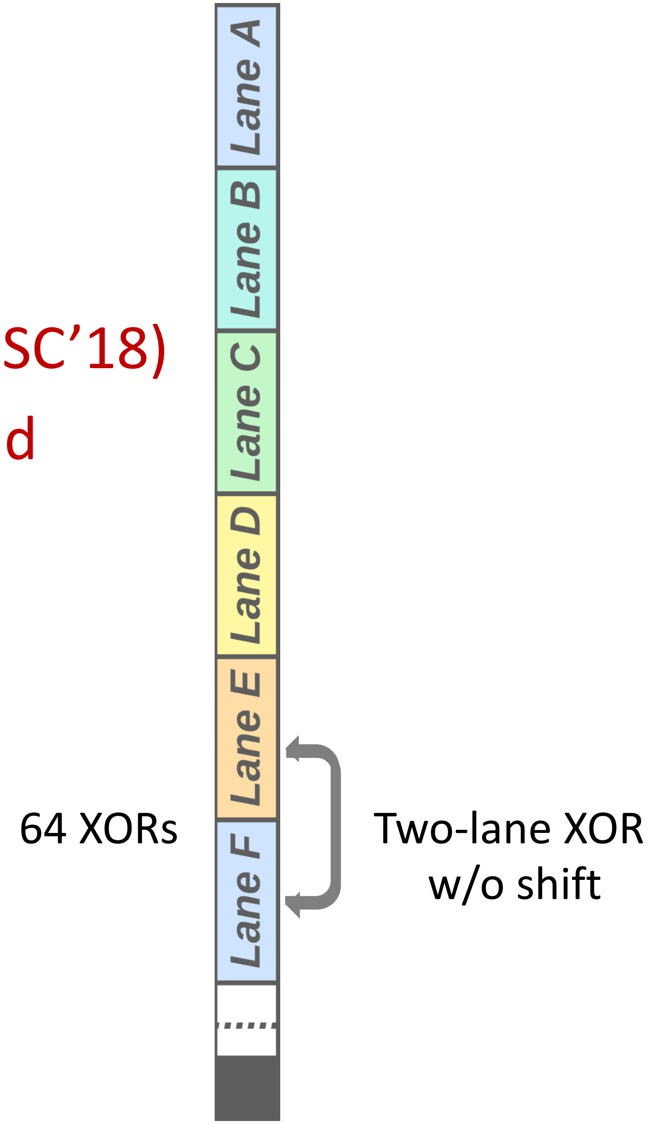
Prior works

- ❑ **Existing Data Alignments**
  - JSSC’18:
    - highly utilize the parallelism
    - hard for inter-lane and intra-lane shift
  - ISCA’18:
    - shift implicitly
    - high latency (>10x JSSC’18)
    - high control overhead



Lane E	Lane B	Lane C	Lane D	Lane E
Lane F	Lane G	Lane H	Lane I	Lane J
Lane K	Lane L	Lane M	Lane N	Lane O
Lane P	Lane Q	Lane R	Lane S	Lane T
Lane U	Lane V	Lane W	Lane X	Lane Y

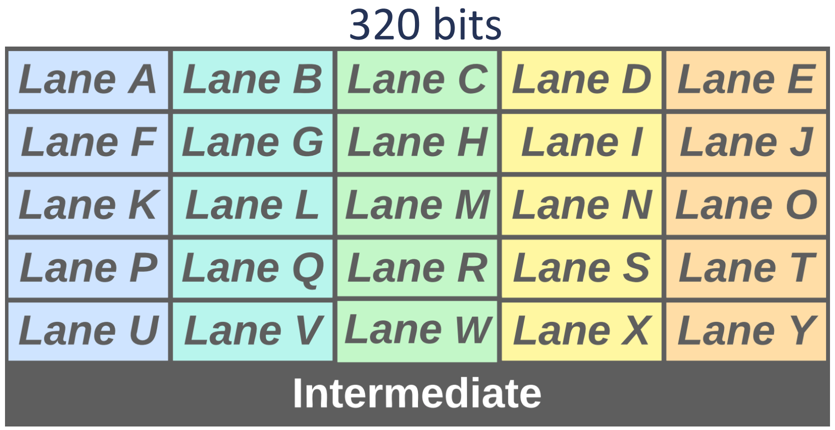
SRAM subarray (JSSC’18)



SRAM subarray (ISCA’18)

Inhale: Shift-optimized Data Alignment

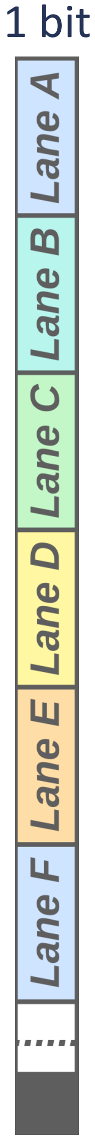
- ❑ **Shift-optimized Data Alignment**



320 bits

Lane A	Lane B	Lane C	Lane D	Lane E
Lane F	Lane G	Lane H	Lane I	Lane J
Lane K	Lane L	Lane M	Lane N	Lane O
Lane P	Lane Q	Lane R	Lane S	Lane T
Lane U	Lane V	Lane W	Lane X	Lane Y
Intermediate				

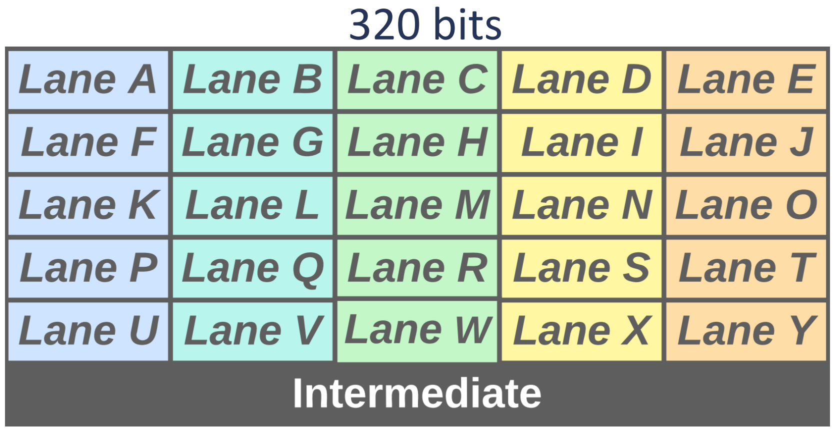
JSSC’18



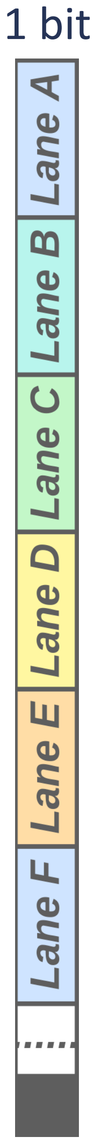
ISCA’18

Inhale: Shift-optimized Data Alignment

- ❑ **Shift-optimized Data Alignment**
  - ○ **Place lane per row**



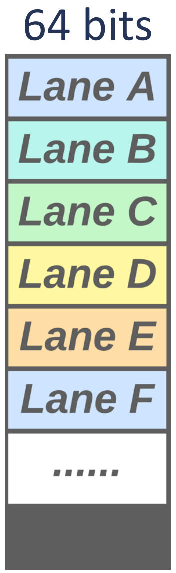
JSSC’18



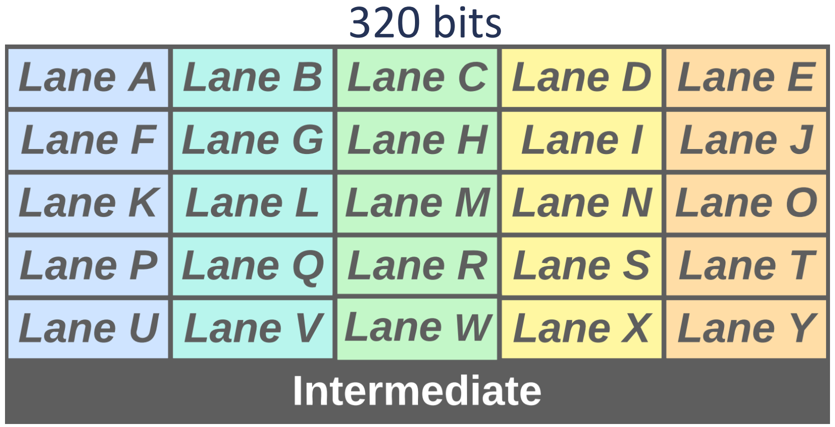
ISCA’18

Inhale: Shift-optimized Data Alignment

- ❑ **Shift-optimized Data Alignment**
  - ○ **Place lane per row**

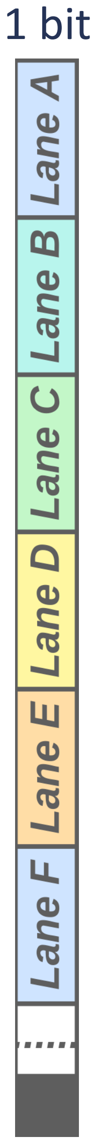


Proposed *Inhale*



Lane A	Lane B	Lane C	Lane D	Lane E
Lane F	Lane G	Lane H	Lane I	Lane J
Lane K	Lane L	Lane M	Lane N	Lane O
Lane P	Lane Q	Lane R	Lane S	Lane T
Lane U	Lane V	Lane W	Lane X	Lane Y
Intermediate				

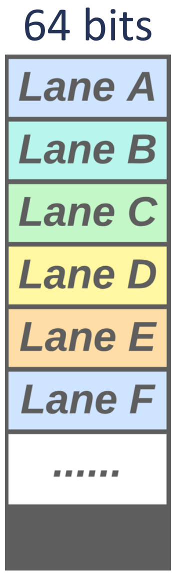
JSSC'18



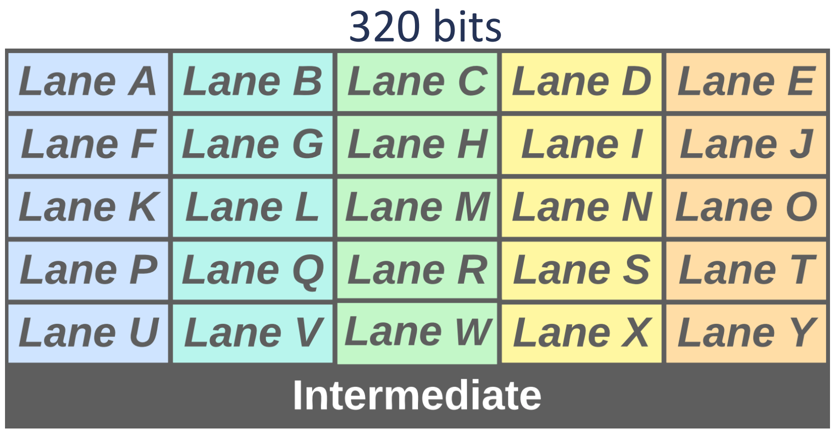
ISCA'18

Inhale: Shift-optimized Data Alignment

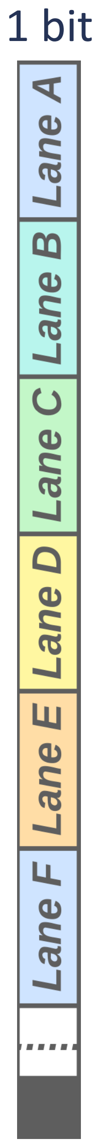
- **Shift-optimized Data Alignment**
  - **Place lane per row**
  - *Inter-lane* shifts are costless with the controller



Proposed *Inhale*



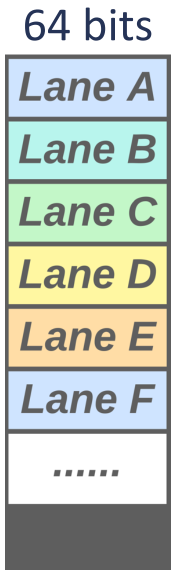
JSSC'18



ISCA'18

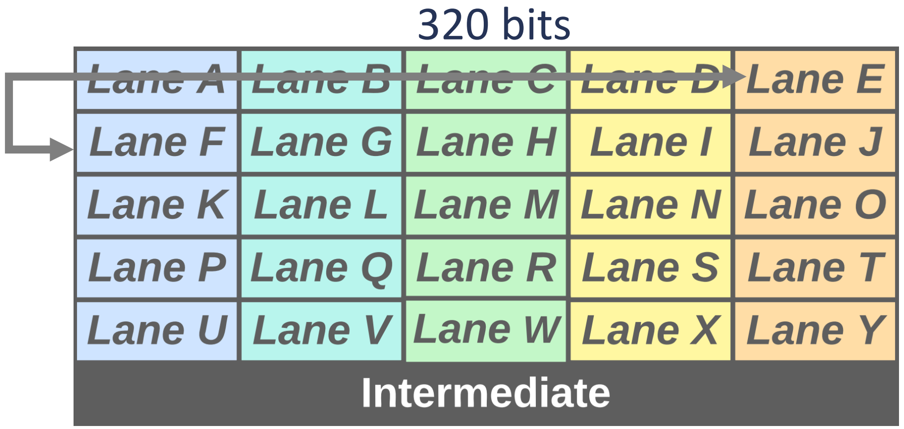
Inhale: Shift-optimized Data Alignment

- **Shift-optimized Data Alignment**
  - **Place lane per row**
  - *Inter-lane* shifts are costless with the controller



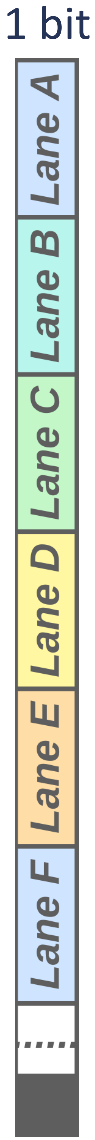
Proposed *Inhale*

First inter-lane shift, then one XOR



~~Lane A~~	~~Lane B~~	~~Lane C~~	~~Lane D~~	Lane E
Lane F	Lane G	Lane H	Lane I	Lane J
Lane K	Lane L	Lane M	Lane N	Lane O
Lane P	Lane Q	Lane R	Lane S	Lane T
Lane U	Lane V	Lane W	Lane X	Lane Y
Intermediate				

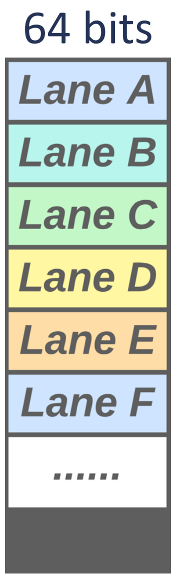
JSSC'18



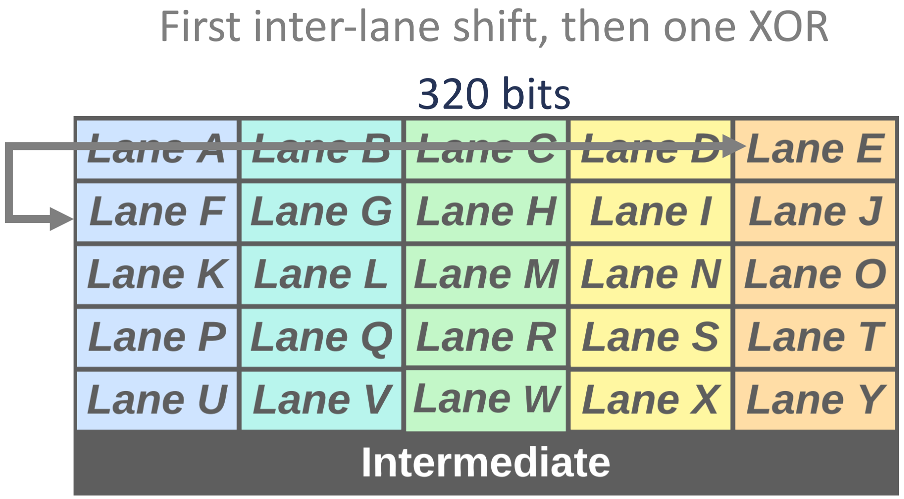
ISCA'18

Inhale: Shift-optimized Data Alignment

- **Shift-optimized Data Alignment**
  - **Place lane per row**
  - *Inter-lane* shifts are costless with the controller

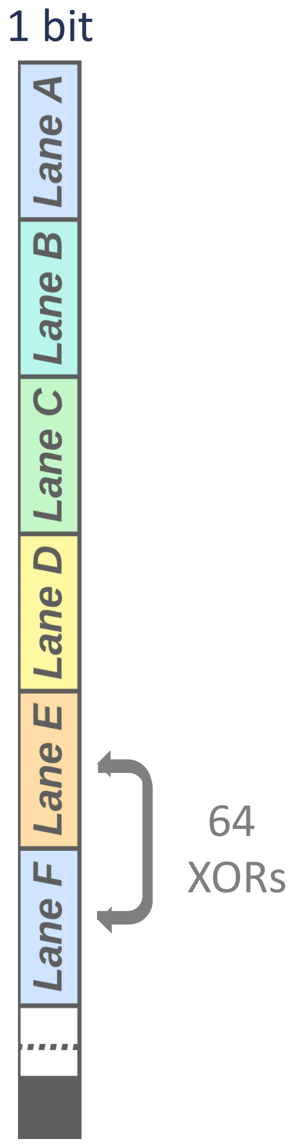


Proposed *Inhale*



Lane A	Lane B	Lane C	Lane D	Lane E
Lane F	Lane G	Lane H	Lane I	Lane J
Lane K	Lane L	Lane M	Lane N	Lane O
Lane P	Lane Q	Lane R	Lane S	Lane T
Lane U	Lane V	Lane W	Lane X	Lane Y

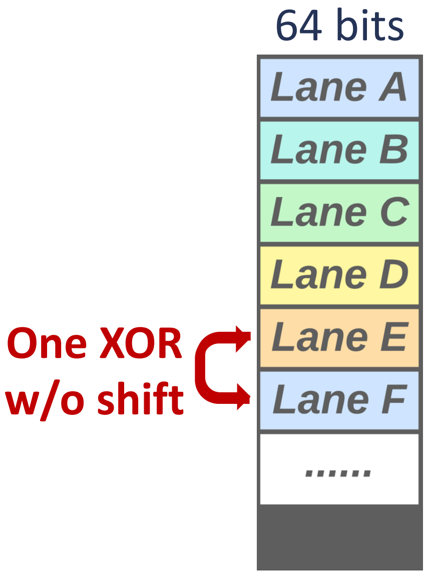
JSSC'18



ISCA'18

Inhale: Shift-optimized Data Alignment

- **Shift-optimized Data Alignment**
  - **Place lane per row**
  - *Inter-lane* shifts are costless with the controller



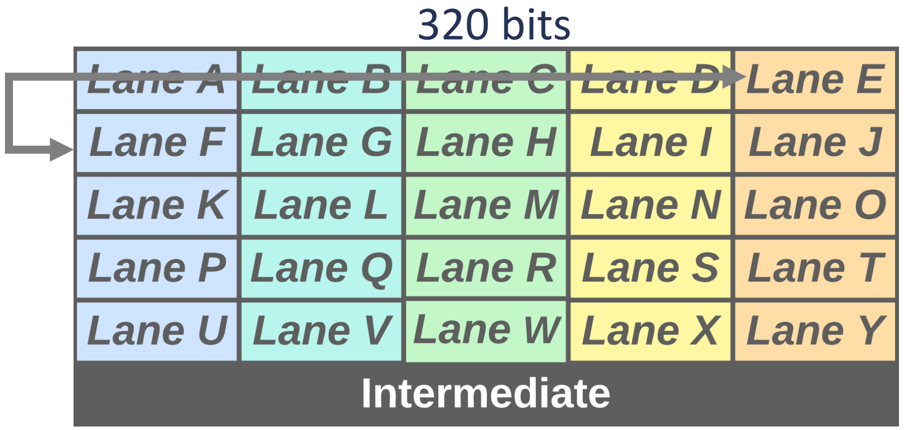
64 bits

Lane A
Lane B
Lane C
Lane D
Lane E
Lane F
……

One XOR w/o shift

Proposed *Inhale*

First inter-lane shift, then one XOR

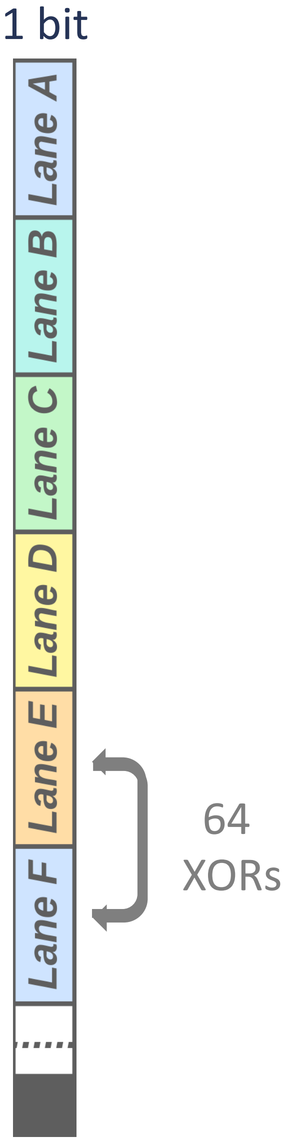


320 bits

~~Lane A~~ ~~Lane B~~ ~~Lane C~~ ~~Lane D~~ Lane E
Lane F Lane G Lane H Lane I Lane J
Lane K Lane L Lane M Lane N Lane O
Lane P Lane Q Lane R Lane S Lane T
Lane U Lane V Lane W Lane X Lane Y

Intermediate

JSSC'18



1 bit

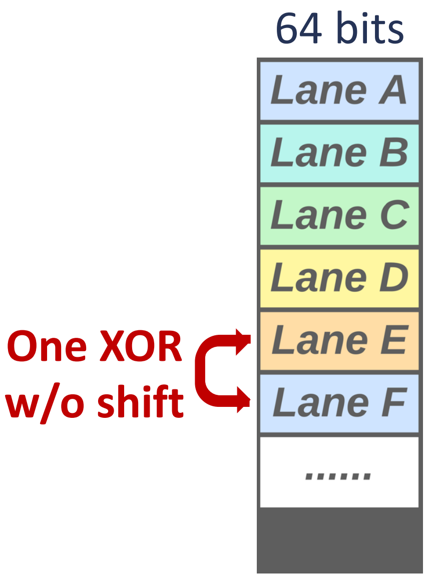
Lane A
Lane B
Lane C
Lane D
Lane E
Lane F
……

64 XORs

ISCA'18

Inhale: Shift-optimized Data Alignment

- **Shift-optimized Data Alignment**
  - **Place lane per row**
  - *Inter-lane* shifts are costless with the controller
  - *Intra-lane* shifts are performed with small shifters


64 bits

Lane A

Lane B

Lane C

Lane D

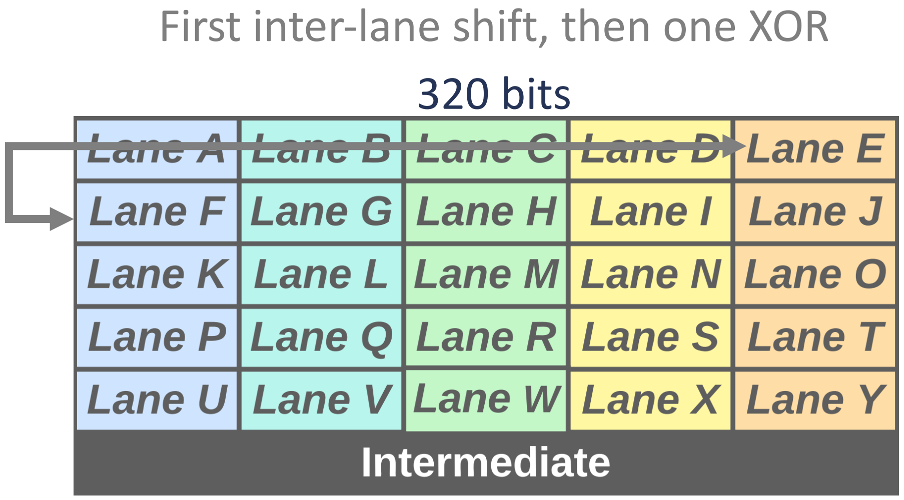
Lane E

Lane F

……

One XOR w/o shift

Proposed *Inhale*

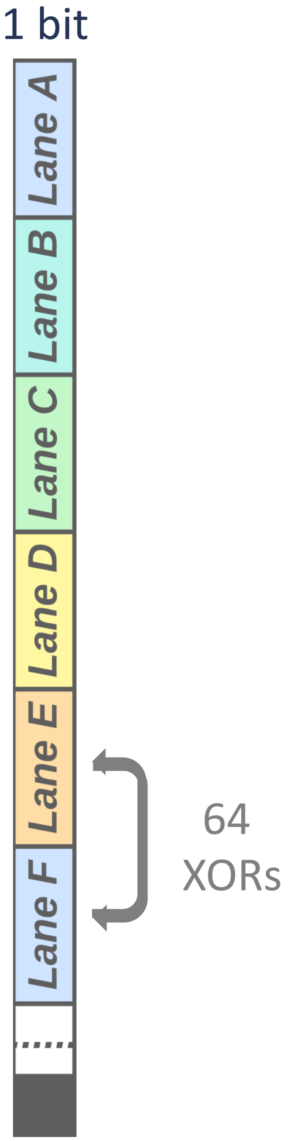

First inter-lane shift, then one XOR

320 bits

~~Lane A~~	~~Lane B~~	~~Lane C~~	~~Lane D~~	Lane E
Lane F	Lane G	Lane H	Lane I	Lane J
Lane K	Lane L	Lane M	Lane N	Lane O
Lane P	Lane Q	Lane R	Lane S	Lane T
Lane U	Lane V	Lane W	Lane X	Lane Y

Intermediate

JSSC'18


1 bit

Lane A

Lane B

Lane C

Lane D

Lane E

Lane F

……

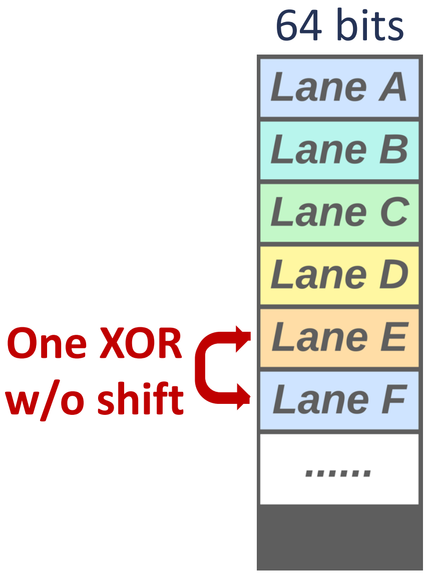
64 XORs

ISCA'18

Inhale: Shift-optimized Data Alignment

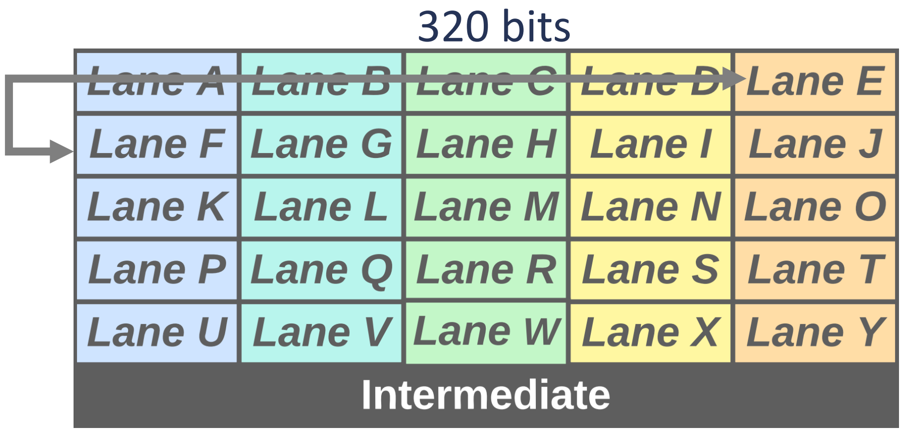
❑ Shift-optimized Data Alignment

- **Place lane per row**
- *Inter-lane* shifts are costless with the controller
- *Intra-lane* shifts are performed with small shifters
- Well balance the performance and overhead



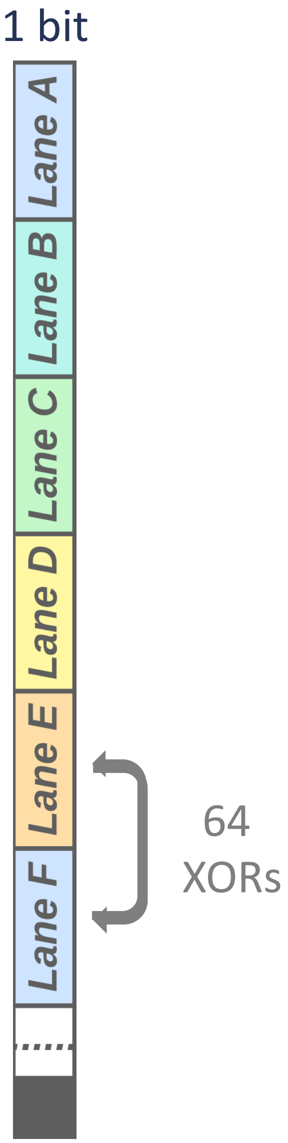
Proposed *Inhale*

First inter-lane shift, then one XOR



Lane A	Lane B	Lane C	Lane D	Lane E
Lane F	Lane G	Lane H	Lane I	Lane J
Lane K	Lane L	Lane M	Lane N	Lane O
Lane P	Lane Q	Lane R	Lane S	Lane T
Lane U	Lane V	Lane W	Lane X	Lane Y

JSSC’18



ISCA’18

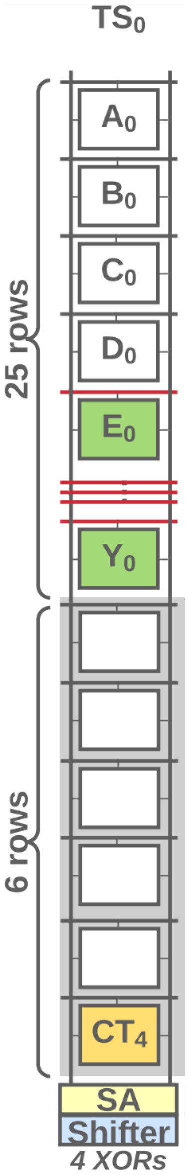
Inhale: In-place read/write strategy

- **In-place read/write strategy**

Inhale: In-place read/write strategy

- **In-place read/write strategy**
  - Read/write order and address are carefully designed to save memory capacity and maintain generality of our solution in varied IoT devices

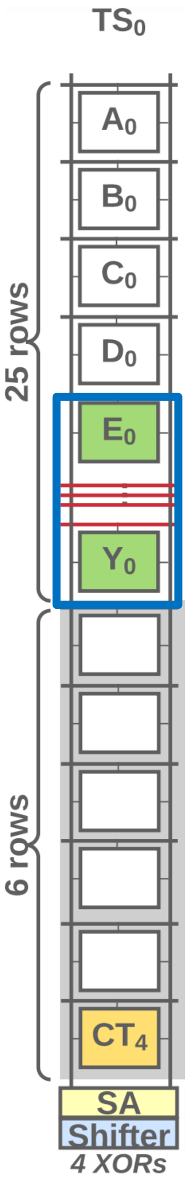
Inhale: In-place read/write strategy



One round of SHA-3

$CT_4 = XOR(E_0, J_0, O_0, T_0, Y_0)$

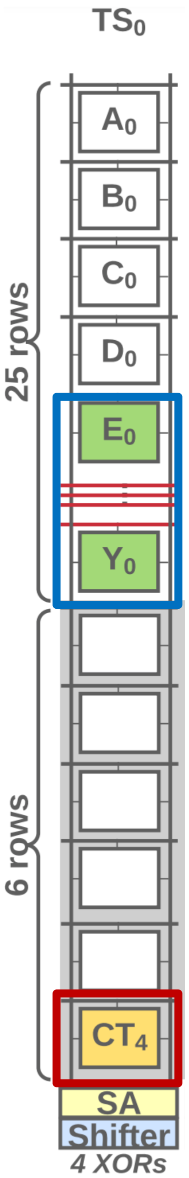
Inhale: In-place read/write strategy



One round of SHA-3

$CT_4 = XOR(E_0, J_0, O_0, T_0, Y_0)$

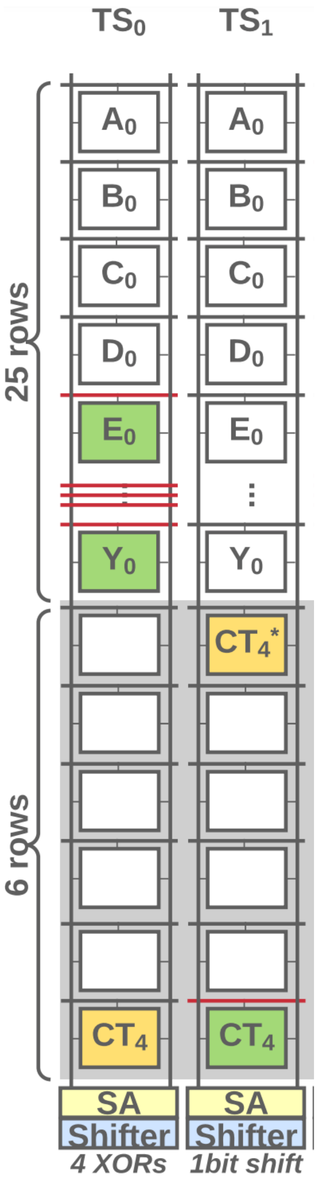
Inhale: In-place read/write strategy



One round of SHA-3

CT_4=XOR(E_0,J_0,O_0,T_0,Y_0)

Inhale: In-place read/write strategy

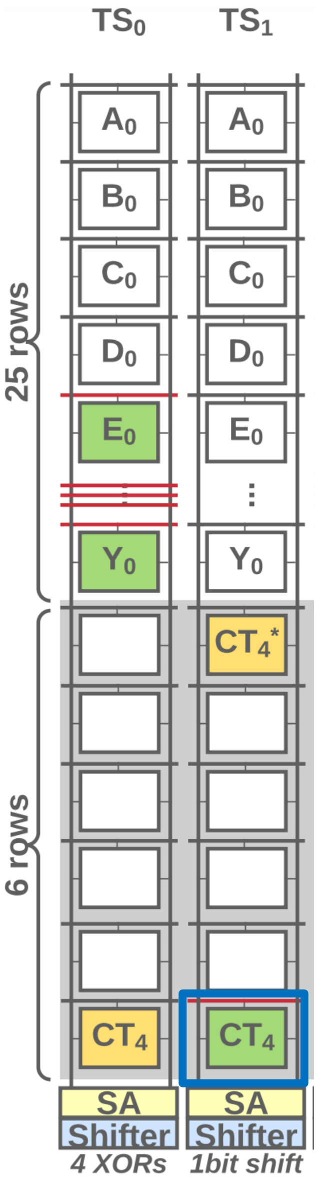


One round of SHA-3

$CT_4 = XOR(E_0, J_0, O_0, T_0, Y_0)$

$CT_4^* = rot(CT_4, 1)$

Inhale: In-place read/write strategy

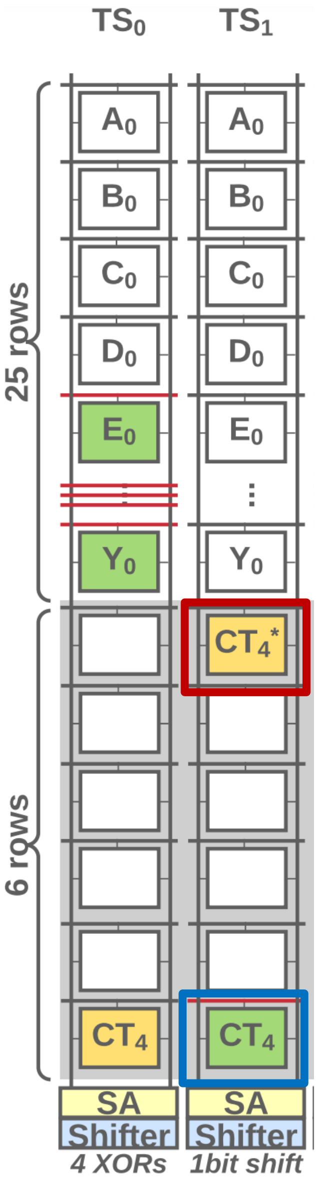


One round of SHA-3

$CT_4 = XOR(E_0, J_0, O_0, T_0, Y_0)$

$CT_4^* = rot(CT_4, 1)$

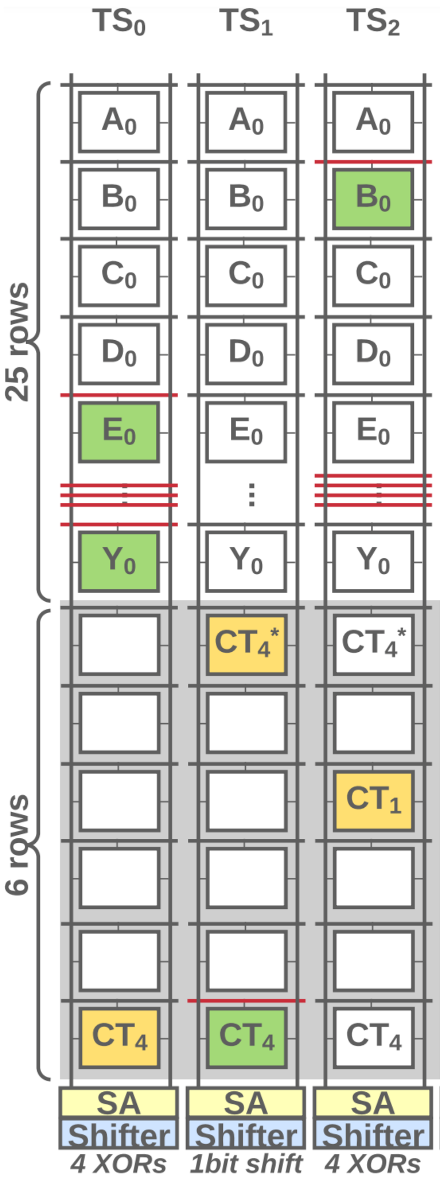
Inhale: In-place read/write strategy



One round of SHA-3

CT_4=XOR(E_0,J_0,O_0,T_0,Y_0)
CT_4^*=rot(CT_4,1)

Inhale: In-place read/write strategy



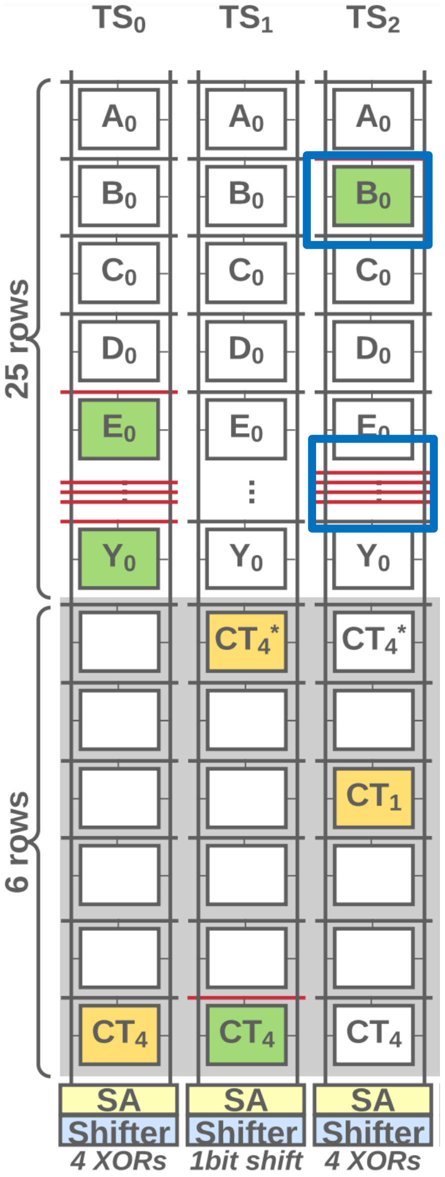
One round of SHA-3

CT_4=XOR(E_0,J_0,O_0,T_0,Y_0)

CT_4^*=rot(CT_4,1)

CT_1=XOR(B_0,G_0,L_0,Q_0,V_0)

Inhale: In-place read/write strategy



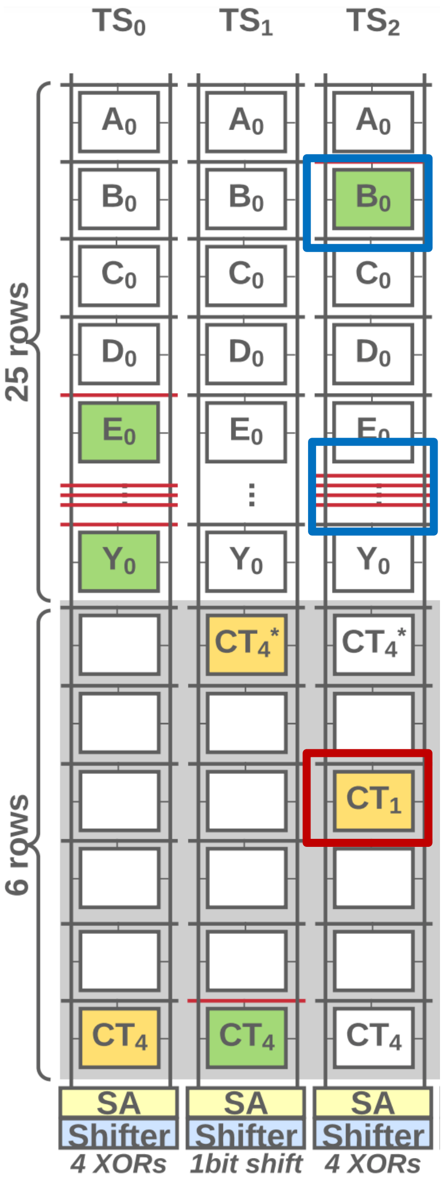
One round of SHA-3

CT_4=XOR(E_0,J_0,O_0,T_0,Y_0)

CT_4^*=rot(CT_4,1)

CT_1=XOR(B_0,G_0,L_0,Q_0,V_0)

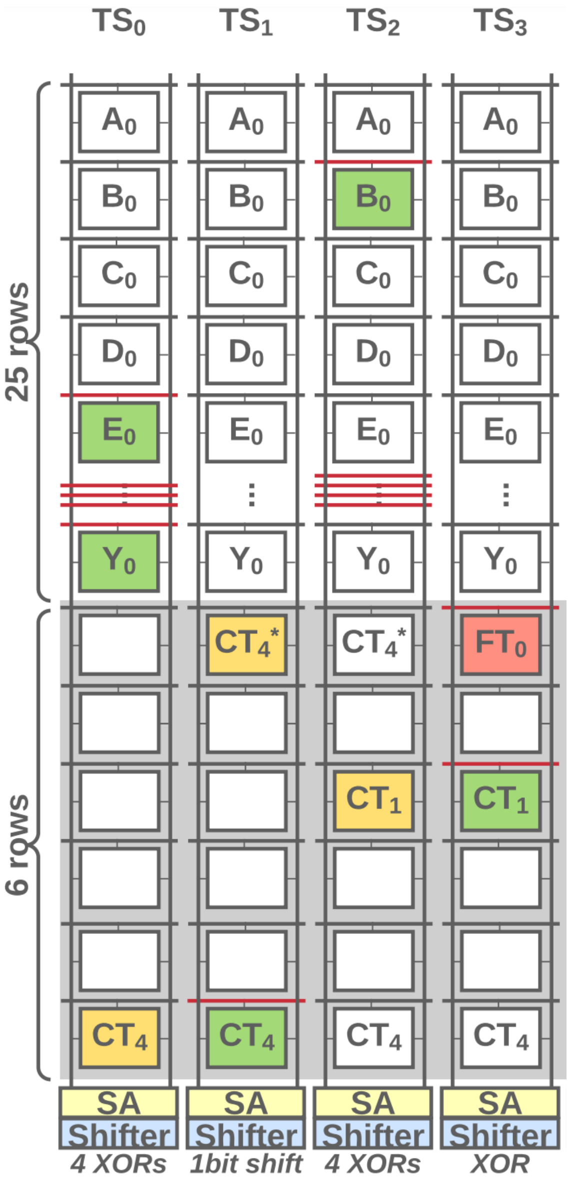
Inhale: In-place read/write strategy



One round of SHA-3

CT_4=XOR(E_0,J_0,O_0,T_0,Y_0)
CT_4^*=rot(CT_4,1)
CT_1=XOR(B_0,G_0,L_0,Q_0,V_0)

Inhale: In-place read/write strategy



One round of SHA-3

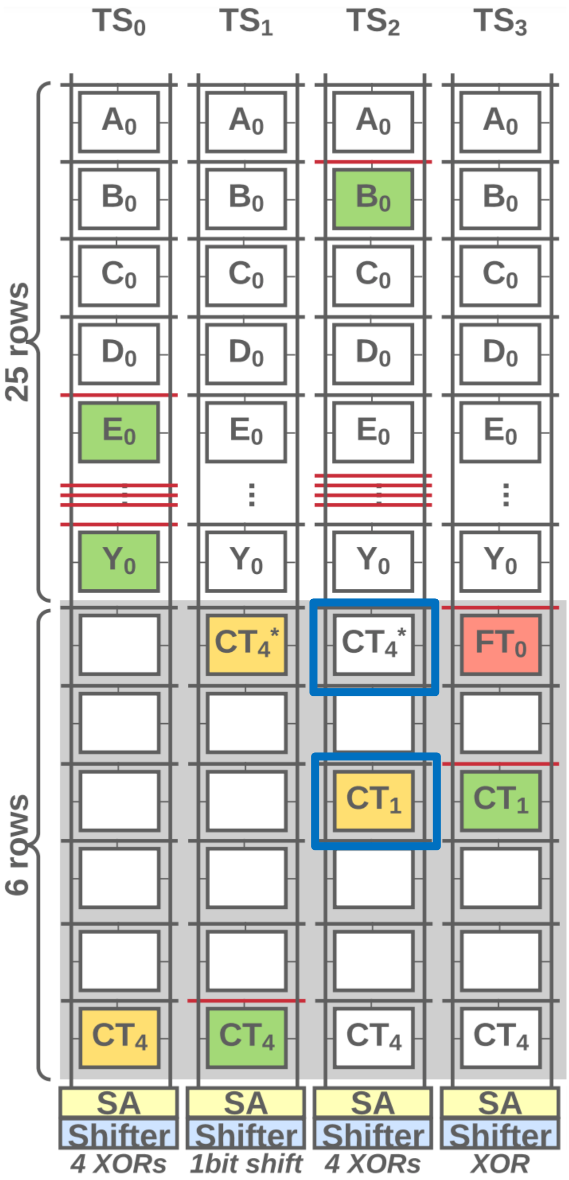
CT_4=XOR(E_0,J_0,O_0,T_0,Y_0)

CT_4^*=rot(CT_4,1)

CT_1=XOR(B_0,G_0,L_0,Q_0,V_0)

FT_0=XOR(CT_1,CT_4^*)

Inhale: In-place read/write strategy



One round of SHA-3

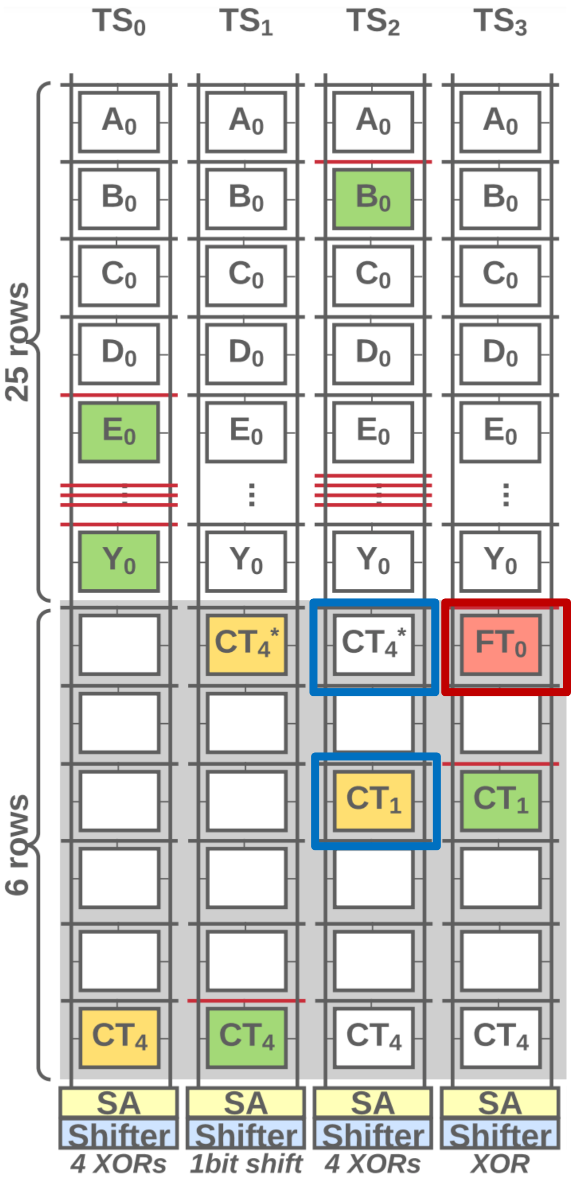
CT_4=XOR(E_0,J_0,O_0,T_0,Y_0)

CT_4^*=rot(CT_4,1)

CT_1=XOR(B_0,G_0,L_0,Q_0,V_0)

FT_0=XOR(CT_1,CT_4^*)

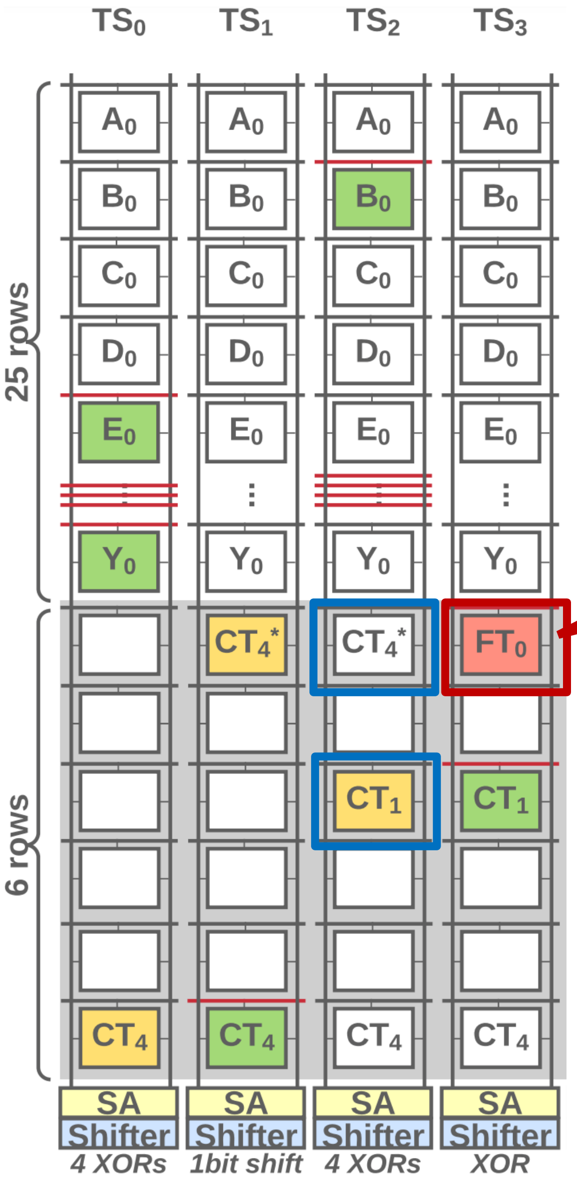
Inhale: In-place read/write strategy



One round of SHA-3

CT_4=XOR(E_0,J_0,O_0,T_0,Y_0)
CT_4^*=rot(CT_4,1)
CT_1=XOR(B_0,G_0,L_0,Q_0,V_0)
FT_0=XOR(CT_1,CT_4^*)

Inhale: In-place read/write strategy



In-place operation

One round of SHA-3

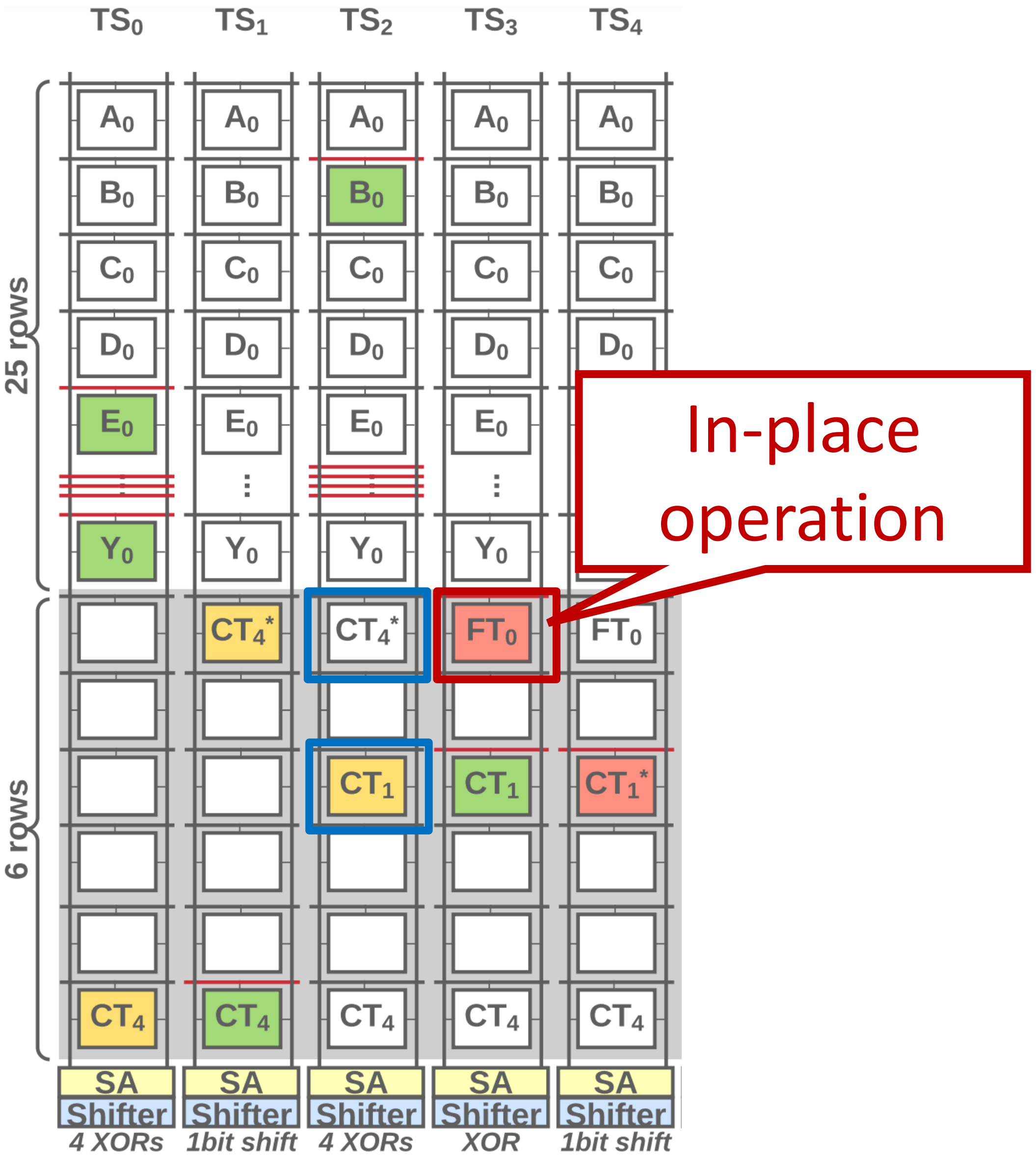
CT_4=XOR(E_0,J_0,O_0,T_0,Y_0)

CT_4^*=rot(CT_4,1)

CT_1=XOR(B_0,G_0,L_0,Q_0,V_0)

FT_0=XOR(CT_1,CT_4^*)

Inhale: In-place read/write strategy



One round of SHA-3

CT_4=XOR(E_0,J_0,O_0,T_0,Y_0)

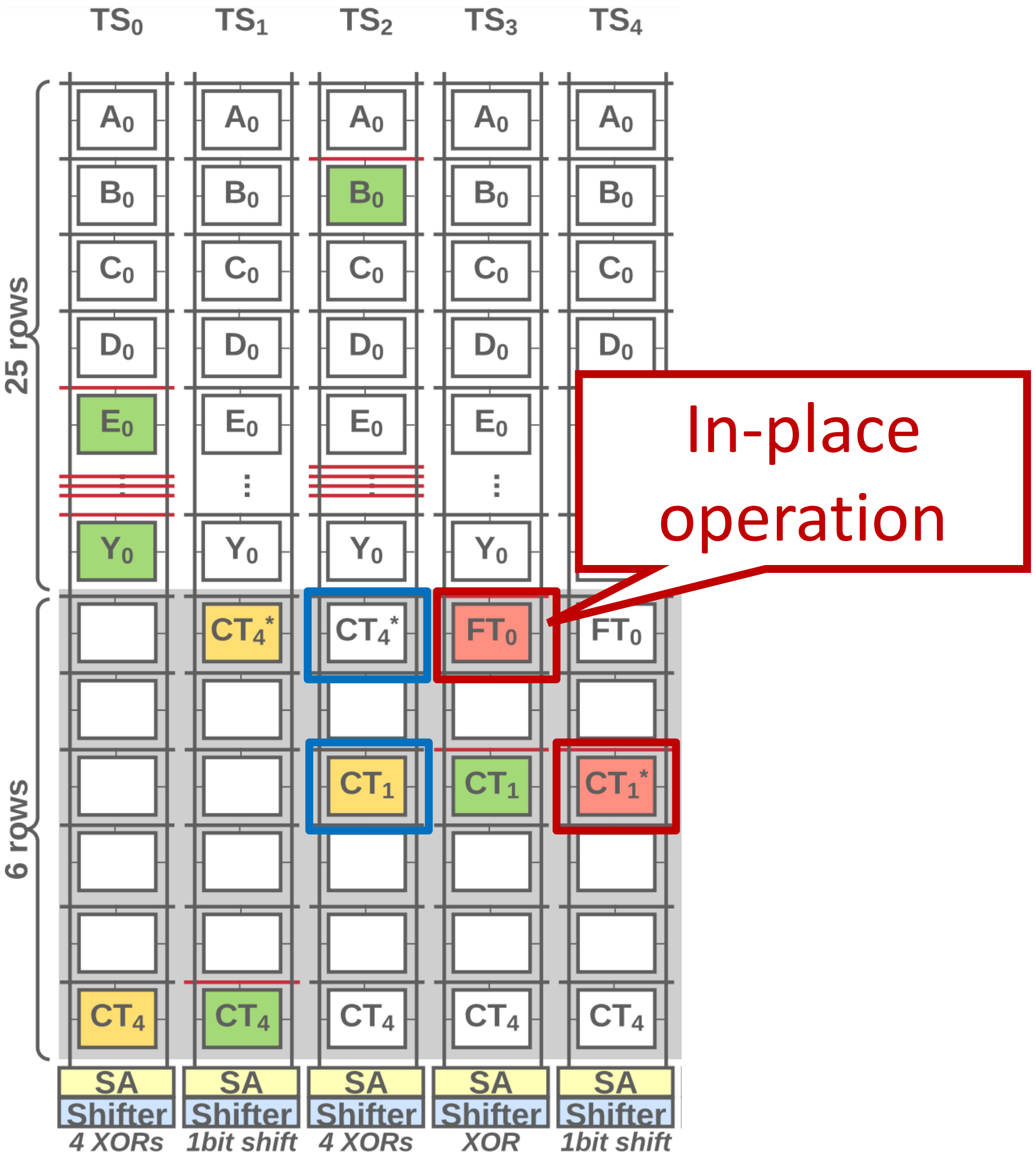
CT_4^*=rot(CT_4,1)

CT_1=XOR(B_0,G_0,L_0,Q_0,V_0)

FT_0=XOR(CT_1,CT_4^*)

CT_1^*=rot(CT_1,1)

Inhale: In-place read/write strategy



In-place operation

One round of SHA-3

CT_4=XOR(E_0,J_0,O_0,T_0,Y_0)

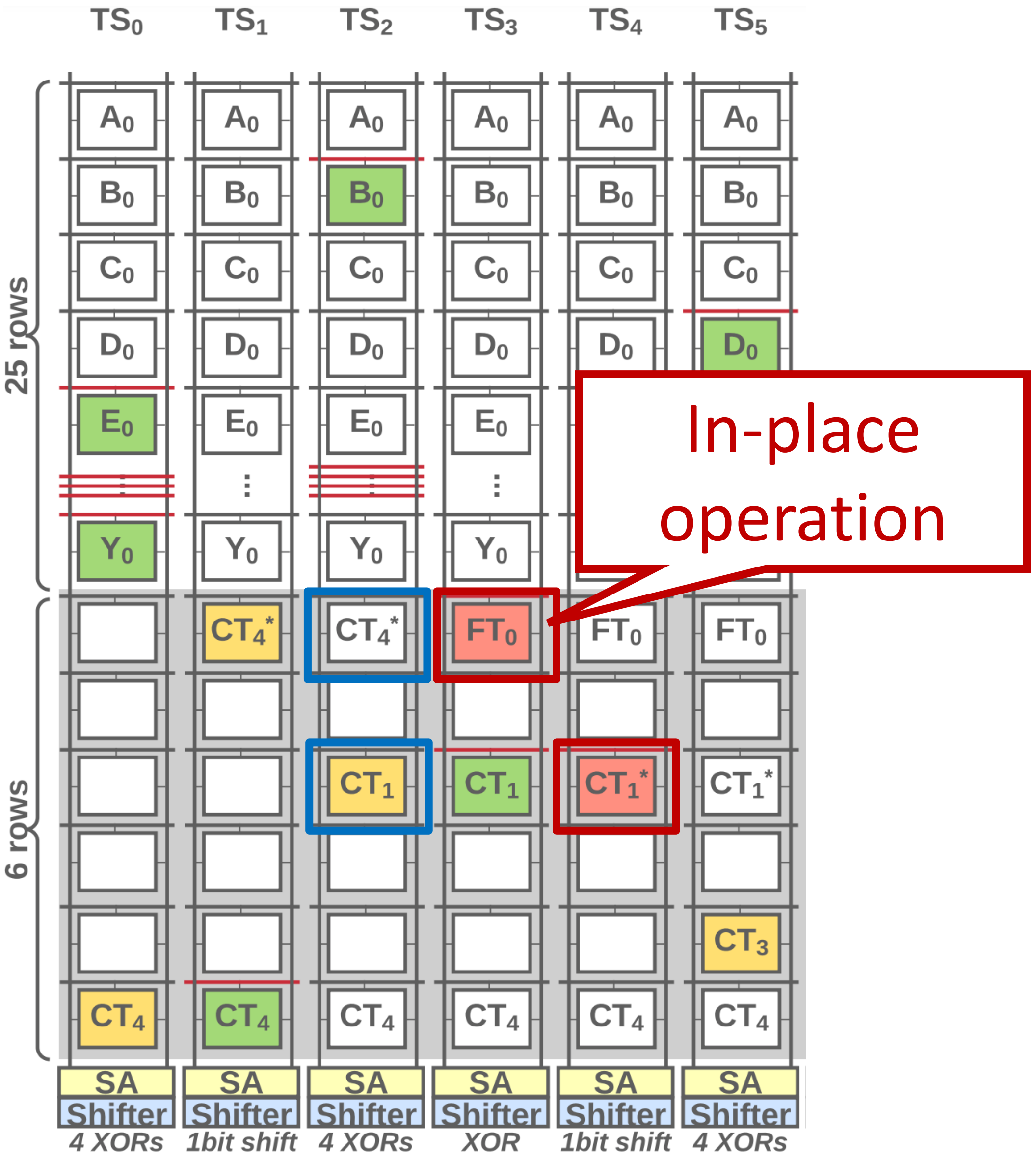
CT_4^*=rot(CT_4,1)

CT_1=XOR(B_0,G_0,L_0,Q_0,V_0)

FT_0=XOR(CT_1,CT_4^*)

CT_1^*=rot(CT_1,1)

Inhale: In-place read/write strategy



	TS_0	TS_1	TS_2	TS_3	TS_4	TS_5
25 rows	A_0	A_0	A_0	A_0	A_0	A_0
	B_0	B_0	B_0	B_0	B_0	B_0
	C_0	C_0	C_0	C_0	C_0	C_0
	D_0	D_0	D_0	D_0	D_0	D_0
	E_0	E_0	E_0	E_0		
	⋮	⋮	⋮	⋮		
	Y_0	Y_0	Y_0	Y_0		
6 rows		CT_4^*	CT_4^*	FT_0	FT_0	FT_0
			CT_1	CT_1	CT_1^*	CT_1^*
						CT_3
	CT_4	CT_4	CT_4	CT_4	CT_4	CT_4
	SA	SA	SA	SA	SA	SA
	Shifter	Shifter	Shifter	Shifter	Shifter	Shifter
	4 XORs	*1bit shift*	*4 XORs*	*XOR*	*1bit shift*	*4 XORs*

In-place operation

One round of SHA-3

CT_4=XOR(E_0,J_0,O_0,T_0,Y_0)

CT_4^*=rot(CT_4,1)

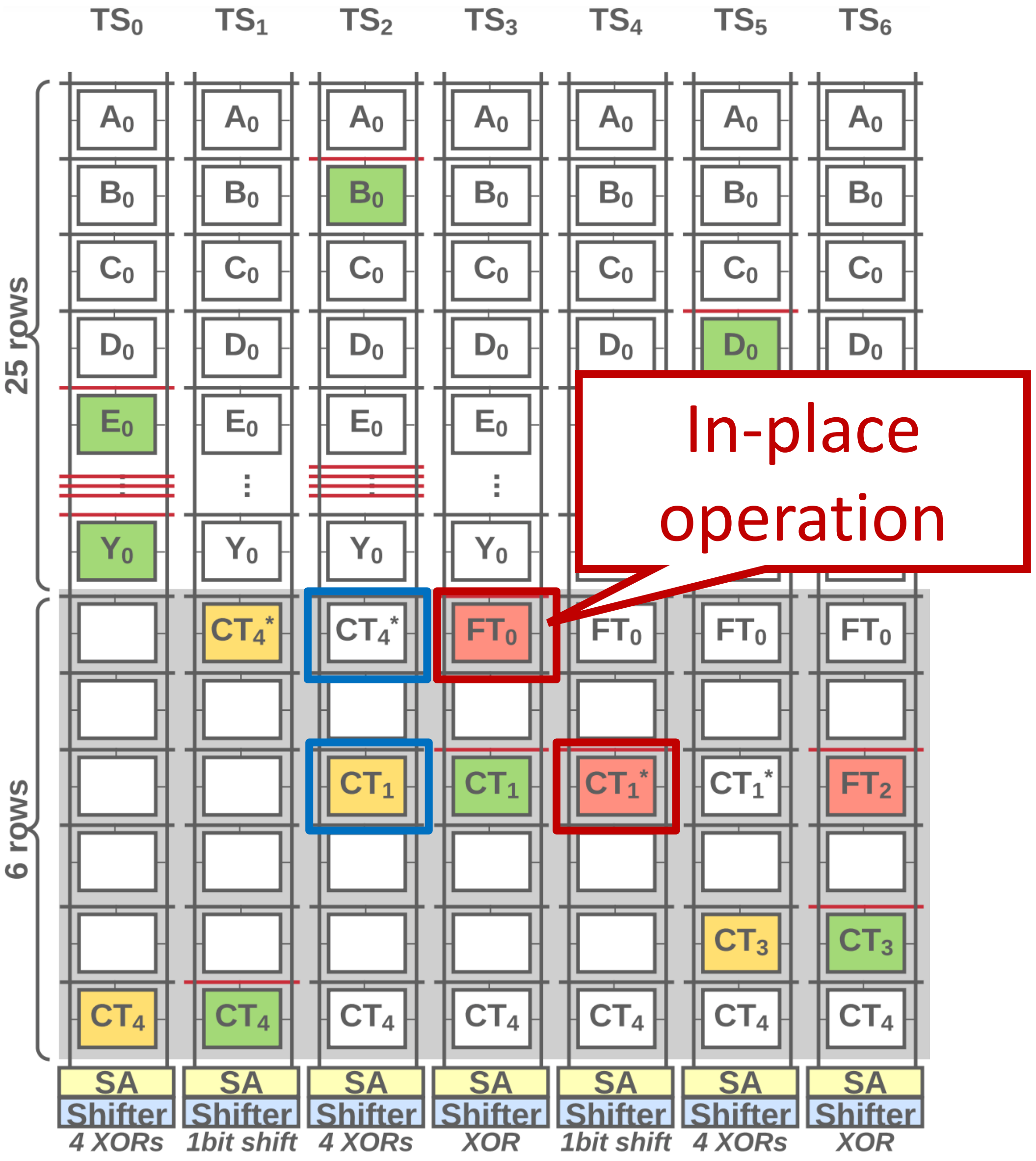
CT_1=XOR(B_0,G_0,L_0,Q_0,V_0)

FT_0=XOR(CT_1,CT_4^*)

CT_1^*=rot(CT_1,1)

CT_3=XOR(D_0,I_0,N_0,S_0,X_0)

Inhale: In-place read/write strategy



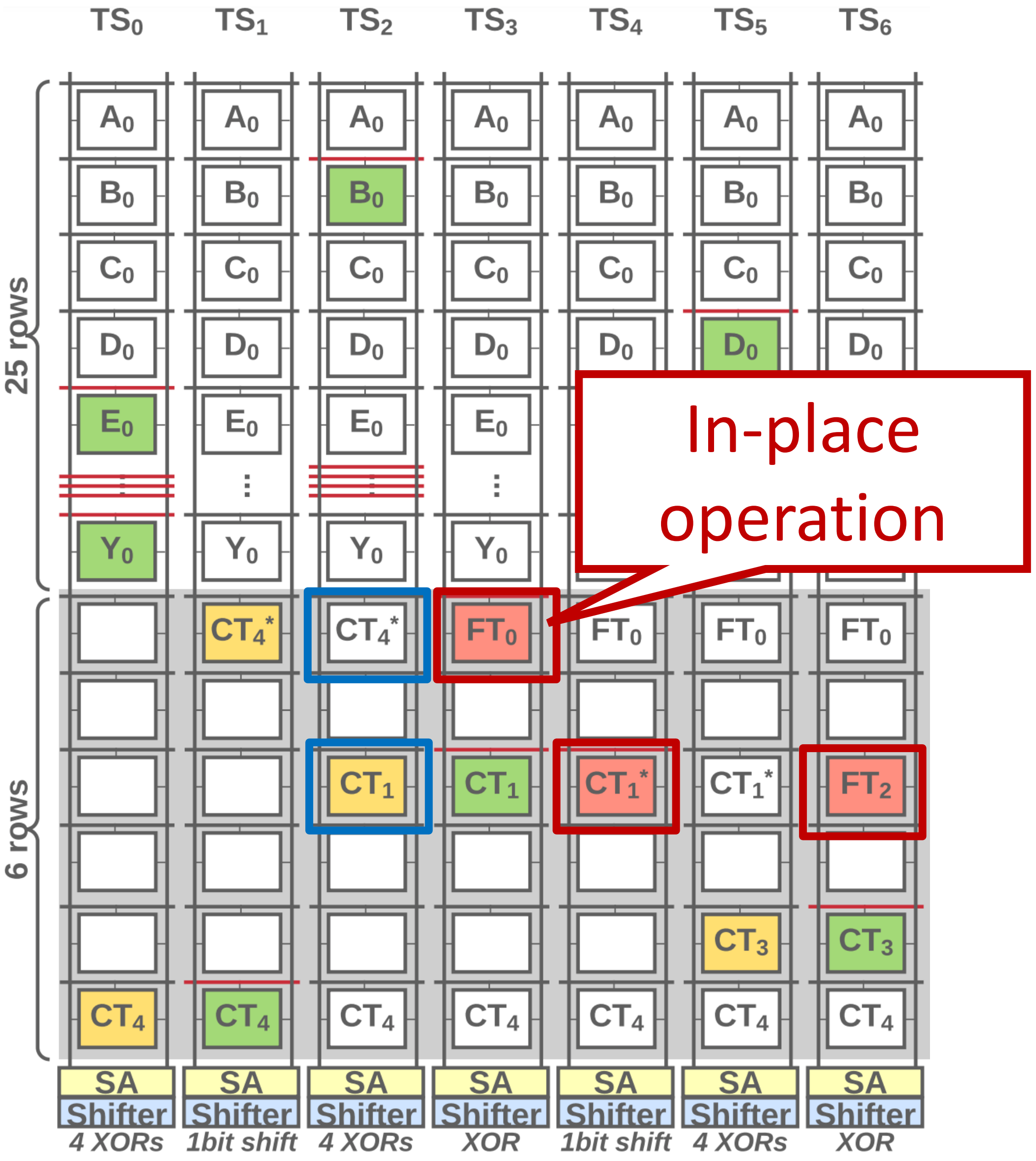
TS$_0$	TS$_1$	TS$_2$	TS$_3$	TS$_4$	TS$_5$	TS$_6$
A_0	A_0	A_0	A_0	A_0	A_0	A_0
B_0	B_0	B_0	B_0	B_0	B_0	B_0
C_0	C_0	C_0	C_0	C_0	C_0	C_0
D_0	D_0	D_0	D_0	D_0	D_0	D_0
E_0	E_0	E_0	E_0			
⋮	⋮	⋮	⋮			
Y_0	Y_0	Y_0	Y_0			
	CT_4^*	CT_4^*	FT_0	FT_0	FT_0	FT_0
		CT_1	CT_1	CT_1^*	CT_1^*	FT_2
					CT_3	CT_3
CT_4	CT_4	CT_4	CT_4	CT_4	CT_4	CT_4
SA	SA	SA	SA	SA	SA	SA
Shifter	Shifter	Shifter	Shifter	Shifter	Shifter	Shifter
4 XORs	*1bit shift*	*4 XORs*	*XOR*	*1bit shift*	*4 XORs*	*XOR*

One round of SHA-3

CT_4=XOR(E_0,J_0,O_0,T_0,Y_0)
CT_4^*=rot(CT_4,1)
CT_1=XOR(B_0,G_0,L_0,Q_0,V_0)
FT_0=XOR(CT_1,CT_4^*)

CT_1^*=rot(CT_1,1)
CT_3=XOR(D_0,I_0,N_0,S_0,X_0)
FT_2=XOR(CT_3,CT_1^*)

Inhale: In-place read/write strategy



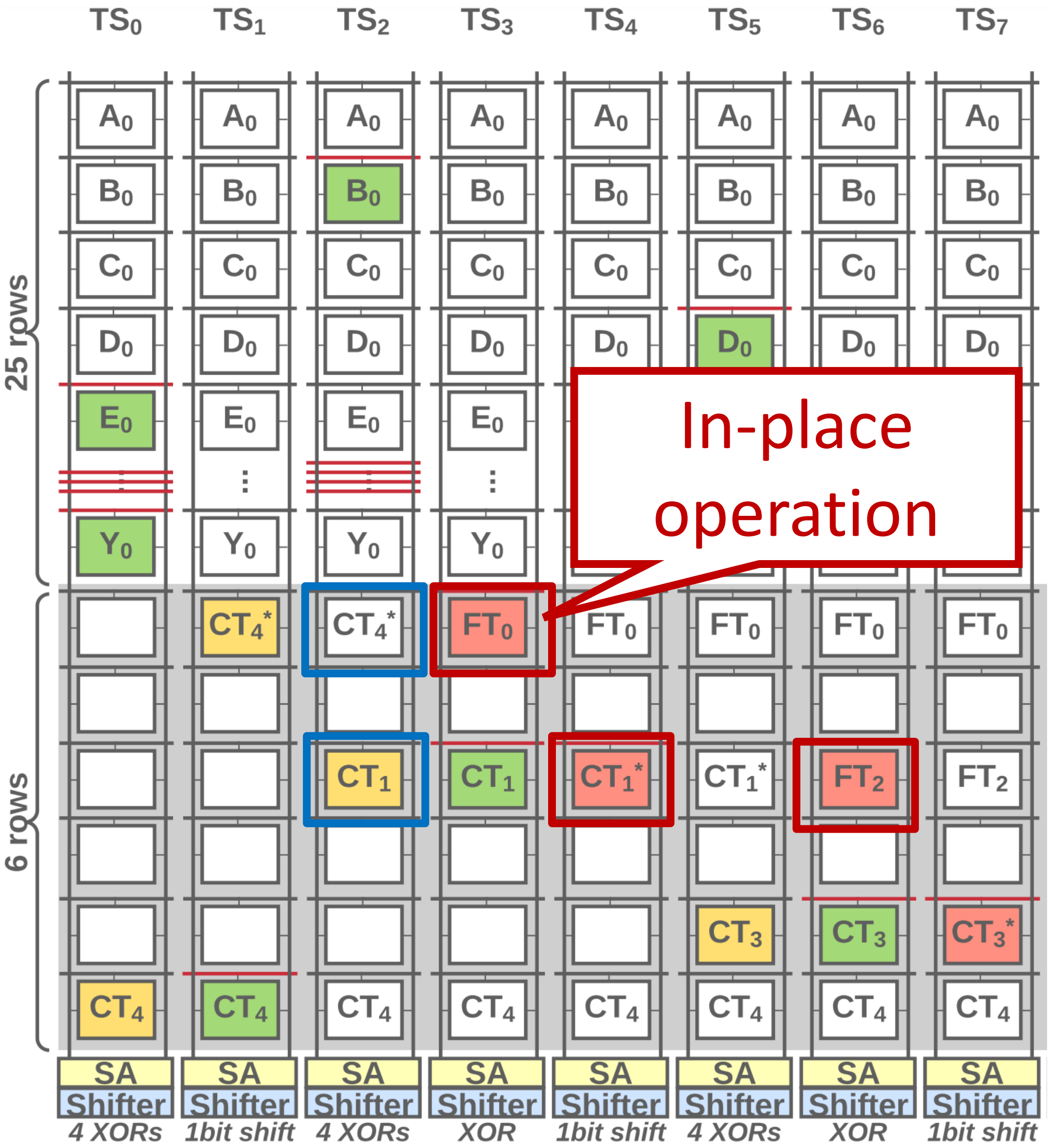
In-place operation

One round of SHA-3

CT_4=XOR(E_0,J_0,O_0,T_0,Y_0)
CT_4^*=rot(CT_4,1)
CT_1=XOR(B_0,G_0,L_0,Q_0,V_0)
FT_0=XOR(CT_1,CT_4^*)

CT_1^*=rot(CT_1,1)
CT_3=XOR(D_0,I_0,N_0,S_0,X_0)
FT_2=XOR(CT_3,CT_1^*)

Inhale: In-place read/write strategy



	TS_0	TS_1	TS_2	TS_3	TS_4	TS_5	TS_6	TS_7
25 rows	A_0	A_0	A_0	A_0	A_0	A_0	A_0	A_0
	B_0	B_0	B_0	B_0	B_0	B_0	B_0	B_0
	C_0	C_0	C_0	C_0	C_0	C_0	C_0	C_0
	D_0	D_0	D_0	D_0	D_0	D_0	D_0	D_0
	E_0	E_0	E_0	E_0				
	⋮	⋮	⋮	⋮				
	Y_0	Y_0	Y_0	Y_0				
6 rows		CT_4^*	CT_4^*	FT_0	FT_0	FT_0	FT_0	FT_0
			CT_1	CT_1	CT_1^*	CT_1^*	FT_2	FT_2
						CT_3	CT_3	CT_3^*
	CT_4	CT_4	CT_4	CT_4	CT_4	CT_4	CT_4	CT_4
	SA	SA	SA	SA	SA	SA	SA	SA
	Shifter	Shifter	Shifter	Shifter	Shifter	Shifter	Shifter	Shifter
	4 XORs	*1bit shift*	*4 XORs*	*XOR*	*1bit shift*	*4 XORs*	*XOR*	*1bit shift*

In-place operation

One round of SHA-3

CT_4=XOR(E_0,J_0,O_0,T_0,Y_0)

CT_4^*=rot(CT_4,1)

CT_1=XOR(B_0,G_0,L_0,Q_0,V_0)

FT_0=XOR(CT_1,CT_4^*)

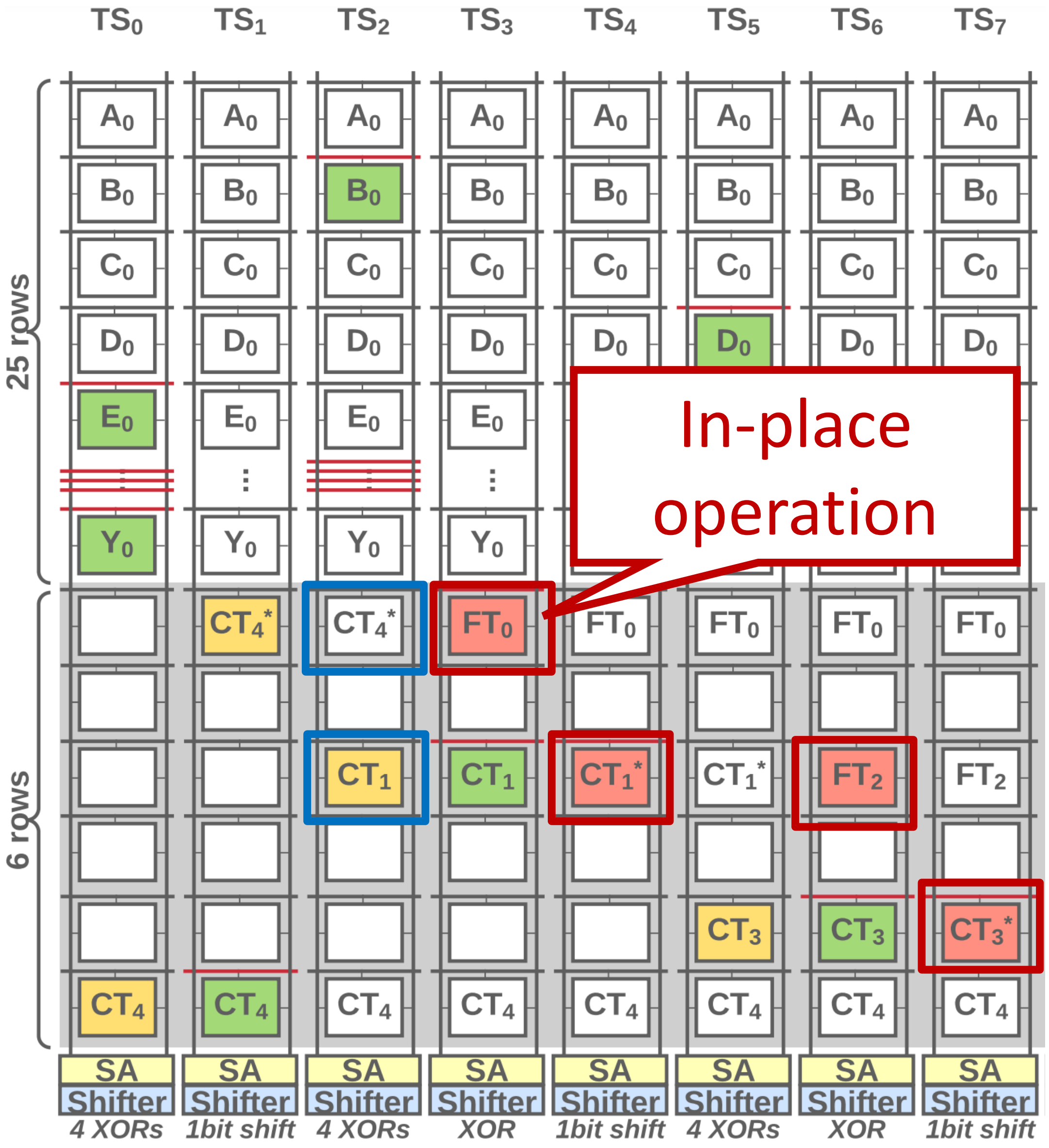
CT_1^*=rot(CT_1,1)

CT_3=XOR(D_0,I_0,N_0,S_0,X_0)

FT_2=XOR(CT_3,CT_1^*)

CT_3^*=rot(CT_3,1)

Inhale: In-place read/write strategy



In-place operation

One round of SHA-3

CT_4=XOR(E_0,J_0,O_0,T_0,Y_0)

CT_4^*=rot(CT_4,1)

CT_1=XOR(B_0,G_0,L_0,Q_0,V_0)

FT_0=XOR(CT_1,CT_4^*)

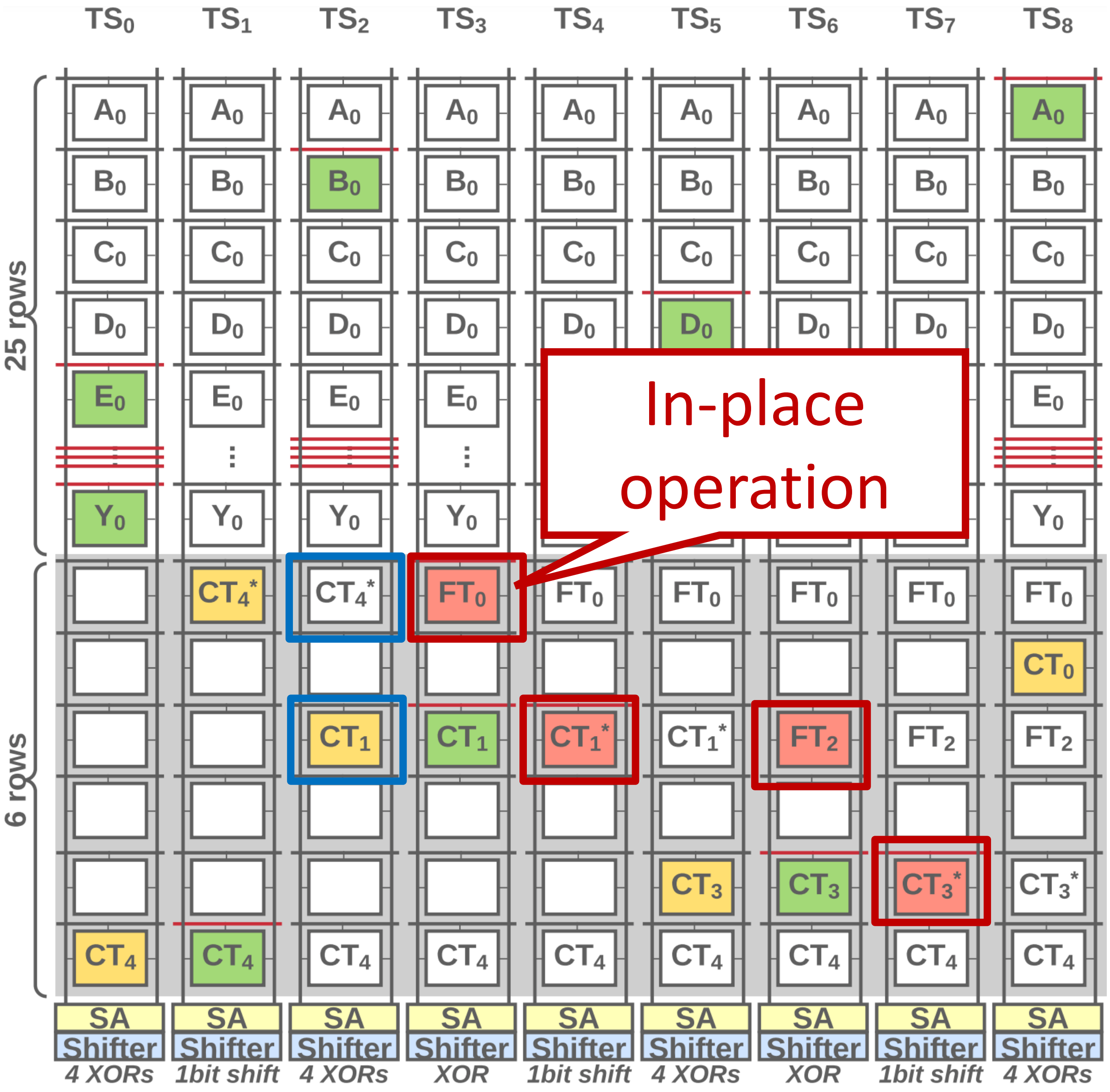
CT_1^*=rot(CT_1,1)

CT_3=XOR(D_0,I_0,N_0,S_0,X_0)

FT_2=XOR(CT_3,CT_1^*)

CT_3^*=rot(CT_3,1)

Inhale: In-place read/write strategy



	TS_0	TS_1	TS_2	TS_3	TS_4	TS_5	TS_6	TS_7	TS_8
25 rows	A_0	A_0	A_0	A_0	A_0	A_0	A_0	A_0	A_0
	B_0	B_0	B_0	B_0	B_0	B_0	B_0	B_0	B_0
	C_0	C_0	C_0	C_0	C_0	C_0	C_0	C_0	C_0
	D_0	D_0	D_0	D_0	D_0	D_0	D_0	D_0	D_0
	E_0	E_0	E_0	E_0					E_0
	⋮	⋮	⋮	⋮					⋮
	Y_0	Y_0	Y_0	Y_0					Y_0
6 rows		CT_4^*	CT_4^*	FT_0	FT_0	FT_0	FT_0	FT_0	FT_0
									CT_0
			CT_1	CT_1	CT_1^*	CT_1^*	FT_2	FT_2	FT_2
						CT_3	CT_3	CT_3^*	CT_3^*
	CT_4	CT_4	CT_4	CT_4	CT_4	CT_4	CT_4	CT_4	CT_4
	SA	SA	SA	SA	SA	SA	SA	SA	SA
	Shifter	Shifter	Shifter	Shifter	Shifter	Shifter	Shifter	Shifter	Shifter
	4 XORs	*1bit shift*	*4 XORs*	*XOR*	*1bit shift*	*4 XORs*	*XOR*	*1bit shift*	*4 XORs*

In-place operation

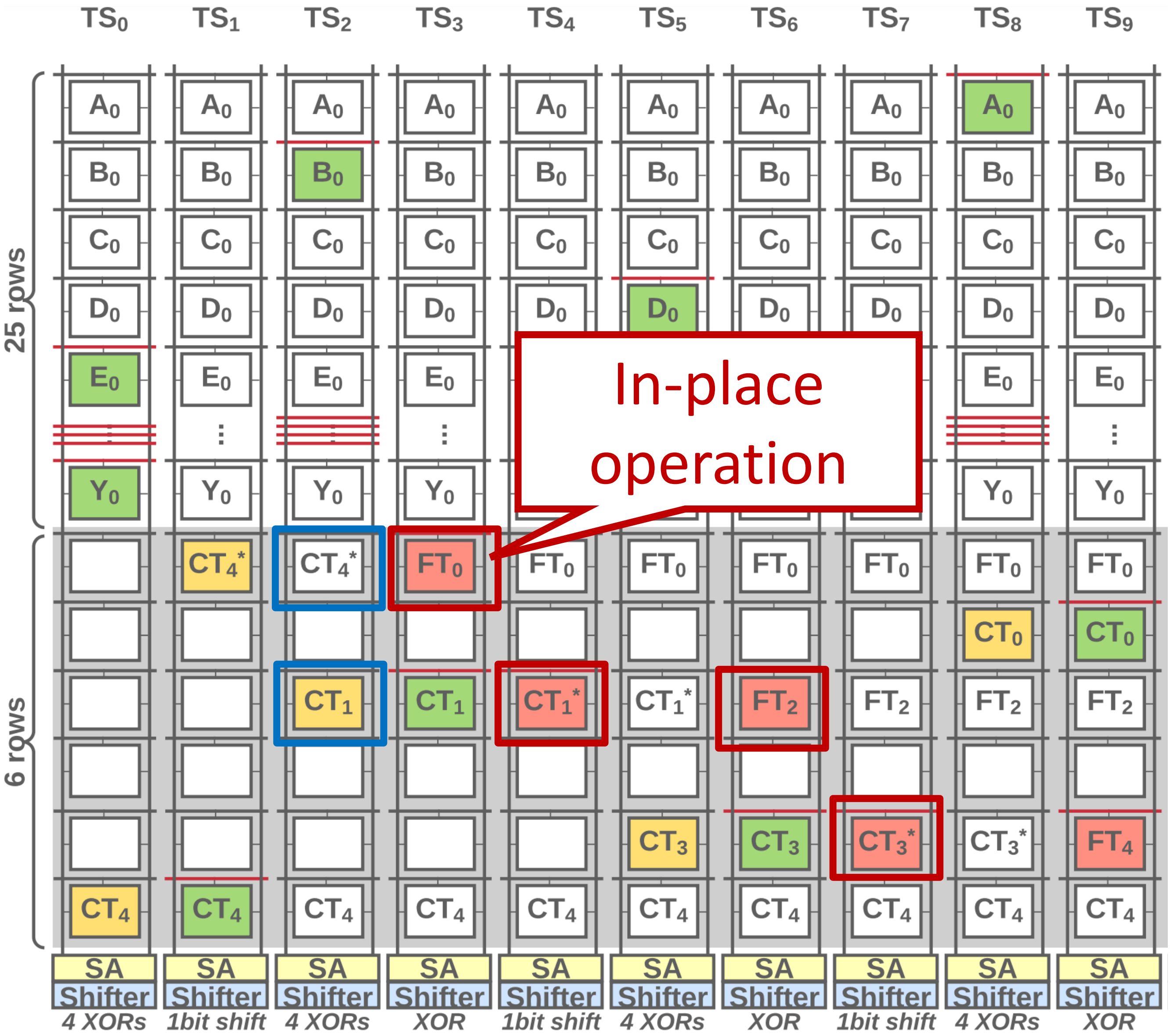
One round of SHA-3

CT_4=XOR(E_0,J_0,O_0,T_0,Y_0)
CT_4^*=rot(CT_4,1)
CT_1=XOR(B_0,G_0,L_0,Q_0,V_0)
FT_0=XOR(CT_1,CT_4^*)

CT_1^*=rot(CT_1,1)
CT_3=XOR(D_0,I_0,N_0,S_0,X_0)
FT_2=XOR(CT_3,CT_1^*)

CT_3^*=rot(CT_3,1)
CT_0=XOR(A_0,F_0,K_0,P_0,U_0)

Inhale: In-place read/write strategy



	TS$_0$	TS$_1$	TS$_2$	TS$_3$	TS$_4$	TS$_5$	TS$_6$	TS$_7$	TS$_8$	TS$_9$
	A$_0$	A$_0$	A$_0$	A$_0$	A$_0$	A$_0$	A$_0$	A$_0$	A$_0$	A$_0$
	B$_0$	B$_0$	B$_0$	B$_0$	B$_0$	B$_0$	B$_0$	B$_0$	B$_0$	B$_0$
	C$_0$	C$_0$	C$_0$	C$_0$	C$_0$	C$_0$	C$_0$	C$_0$	C$_0$	C$_0$
	D$_0$	D$_0$	D$_0$	D$_0$	D$_0$	D$_0$	D$_0$	D$_0$	D$_0$	D$_0$
	E$_0$	E$_0$	E$_0$	E$_0$					E$_0$	E$_0$
	⋮	⋮	⋮	⋮					⋮	⋮
	Y$_0$	Y$_0$	Y$_0$	Y$_0$					Y$_0$	Y$_0$
		CT$_4^*$	CT$_4^*$	FT$_0$	FT$_0$	FT$_0$	FT$_0$	FT$_0$	FT$_0$	FT$_0$
									CT$_0$	CT$_0$
			CT$_1$	CT$_1$	CT$_1^*$	CT$_1^*$	FT$_2$	FT$_2$	FT$_2$	FT$_2$
						CT$_3$	CT$_3$	CT$_3^*$	CT$_3^*$	FT$_4$
	CT$_4$	CT$_4$	CT$_4$	CT$_4$	CT$_4$	CT$_4$	CT$_4$	CT$_4$	CT$_4$	CT$_4$
	SA	SA	SA	SA	SA	SA	SA	SA	SA	SA
	Shifter	Shifter	Shifter	Shifter	Shifter	Shifter	Shifter	Shifter	Shifter	Shifter
	4 XORs	*1bit shift*	*4 XORs*	*XOR*	*1bit shift*	*4 XORs*	*XOR*	*1bit shift*	*4 XORs*	*XOR*

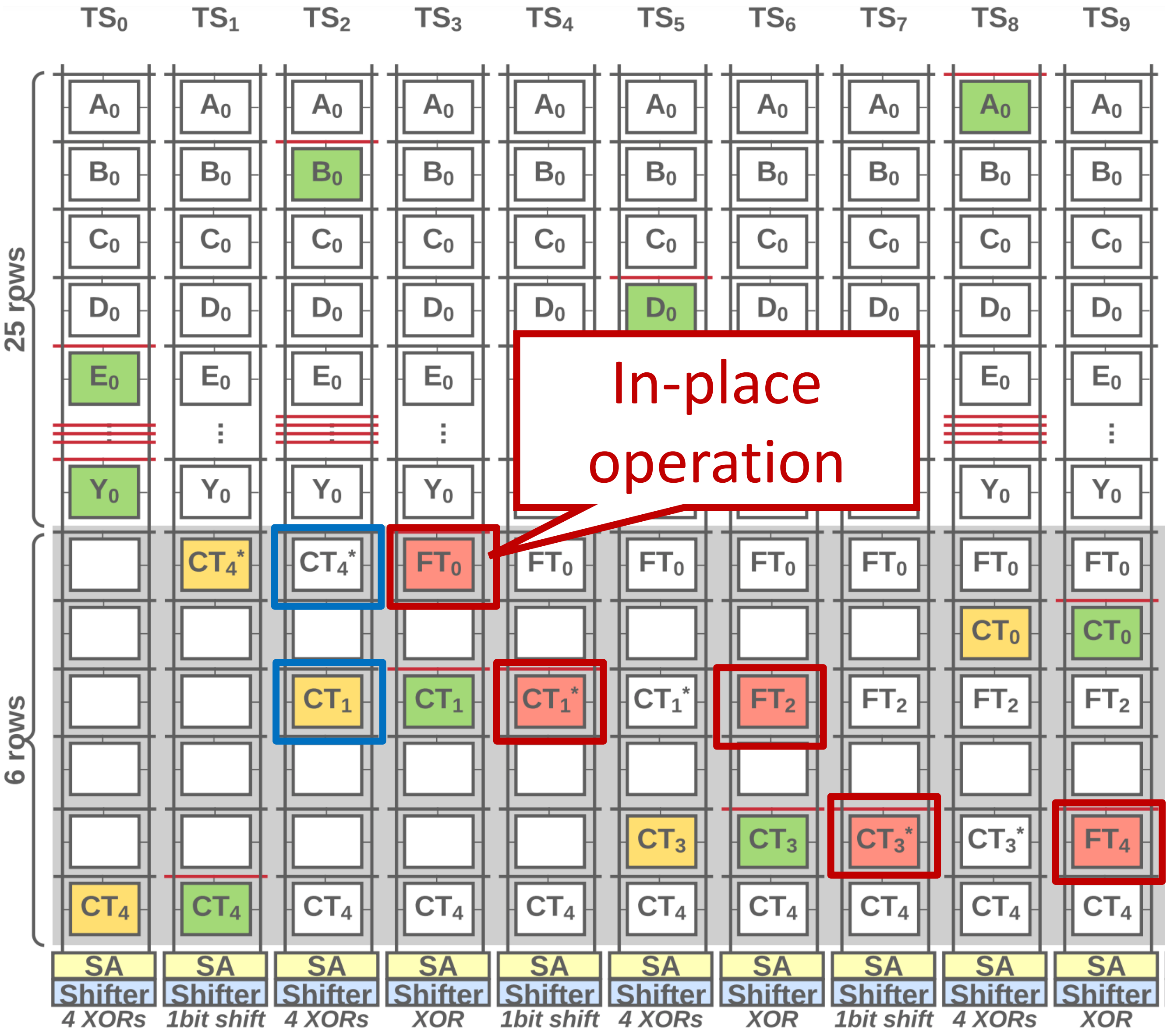
One round of SHA-3

CT_4=XOR(E_0,J_0,O_0,T_0,Y_0)
CT_4^*=rot(CT_4,1)
CT_1=XOR(B_0,G_0,L_0,Q_0,V_0)
FT_0=XOR(CT_1,CT_4^*)

CT_1^*=rot(CT_1,1)
CT_3=XOR(D_0,I_0,N_0,S_0,X_0)
FT_2=XOR(CT_3,CT_1^*)

CT_3^*=rot(CT_3,1)
CT_0=XOR(A_0,F_0,K_0,P_0,U_0)
FT_4=XOR(CT_0,CT_3^*)

Inhale: In-place read/write strategy



	TS_0	TS_1	TS_2	TS_3	TS_4	TS_5	TS_6	TS_7	TS_8	TS_9
25 rows	A_0	A_0	A_0	A_0	A_0	A_0	A_0	A_0	A_0	A_0
	B_0	B_0	B_0	B_0	B_0	B_0	B_0	B_0	B_0	B_0
	C_0	C_0	C_0	C_0	C_0	C_0	C_0	C_0	C_0	C_0
	D_0	D_0	D_0	D_0	D_0	D_0	D_0	D_0	D_0	D_0
	E_0	E_0	E_0	E_0					E_0	E_0
	⋮	⋮	⋮	⋮					⋮	⋮
	Y_0	Y_0	Y_0	Y_0					Y_0	Y_0
6 rows		CT_4^*	CT_4^*	FT_0	FT_0	FT_0	FT_0	FT_0	FT_0	FT_0
									CT_0	CT_0
			CT_1	CT_1	CT_1^*	CT_1^*	FT_2	FT_2	FT_2	FT_2
						CT_3	CT_3	CT_3^*	CT_3^*	FT_4
	CT_4	CT_4	CT_4	CT_4	CT_4	CT_4	CT_4	CT_4	CT_4	CT_4
	SA	SA	SA	SA	SA	SA	SA	SA	SA	SA
	Shifter	Shifter	Shifter	Shifter	Shifter	Shifter	Shifter	Shifter	Shifter	Shifter
	4 XORs	*1bit shift*	*4 XORs*	*XOR*	*1bit shift*	*4 XORs*	*XOR*	*1bit shift*	*4 XORs*	*XOR*

In-place operation

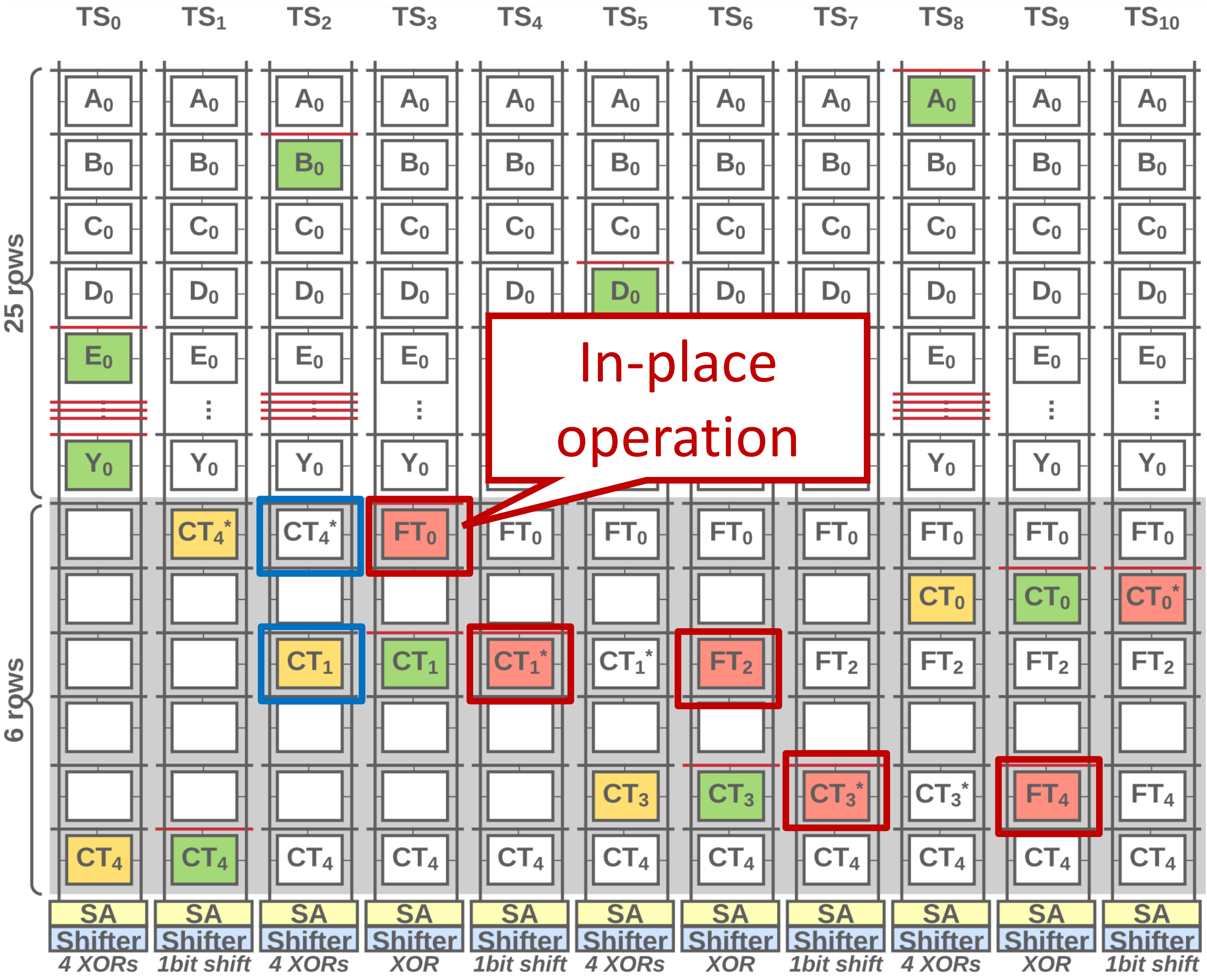
One round of SHA-3

CT_4=XOR(E_0,J_0,O_0,T_0,Y_0)
CT_4^*=rot(CT_4,1)
CT_1=XOR(B_0,G_0,L_0,Q_0,V_0)
FT_0=XOR(CT_1,CT_4^*)

CT_1^*=rot(CT_1,1)
CT_3=XOR(D_0,I_0,N_0,S_0,X_0)
FT_2=XOR(CT_3,CT_1^*)

CT_3^*=rot(CT_3,1)
CT_0=XOR(A_0,F_0,K_0,P_0,U_0)
FT_4=XOR(CT_0,CT_3^*)

Inhale: In-place read/write strategy



	TS$_0$	TS$_1$	TS$_2$	TS$_3$	TS$_4$	TS$_5$	TS$_6$	TS$_7$	TS$_8$	TS$_9$	TS$_{10}$
25 rows	A$_0$	A$_0$	A$_0$	A$_0$	A$_0$	A$_0$	A$_0$	A$_0$	A$_0$	A$_0$	A$_0$
	B$_0$	B$_0$	B$_0$	B$_0$	B$_0$	B$_0$	B$_0$	B$_0$	B$_0$	B$_0$	B$_0$
	C$_0$	C$_0$	C$_0$	C$_0$	C$_0$	C$_0$	C$_0$	C$_0$	C$_0$	C$_0$	C$_0$
	D$_0$	D$_0$	D$_0$	D$_0$	D$_0$	D$_0$	D$_0$	D$_0$	D$_0$	D$_0$	D$_0$
	E$_0$	E$_0$	E$_0$	E$_0$					E$_0$	E$_0$	E$_0$
	⋮	⋮	⋮	⋮					⋮	⋮	⋮
	Y$_0$	Y$_0$	Y$_0$	Y$_0$					Y$_0$	Y$_0$	Y$_0$
6 rows		CT$_4^*$	CT$_4^*$	FT$_0$	FT$_0$	FT$_0$	FT$_0$	FT$_0$	FT$_0$	FT$_0$	FT$_0$
									CT$_0$	CT$_0$	CT$_0^*$
			CT$_1$	CT$_1$	CT$_1^*$	CT$_1^*$	FT$_2$	FT$_2$	FT$_2$	FT$_2$	FT$_2$
						CT$_3$	CT$_3$	CT$_3^*$	CT$_3^*$	FT$_4$	FT$_4$
	CT$_4$	CT$_4$	CT$_4$	CT$_4$	CT$_4$	CT$_4$	CT$_4$	CT$_4$	CT$_4$	CT$_4$	CT$_4$
	SA	SA	SA	SA	SA	SA	SA	SA	SA	SA	SA
	Shifter	Shifter	Shifter	Shifter	Shifter	Shifter	Shifter	Shifter	Shifter	Shifter	Shifter
	4 XORs	*1bit shift*	*4 XORs*	*XOR*	*1bit shift*	*4 XORs*	*XOR*	*1bit shift*	*4 XORs*	*XOR*	*1bit shift*

In-place operation

One round of SHA-3

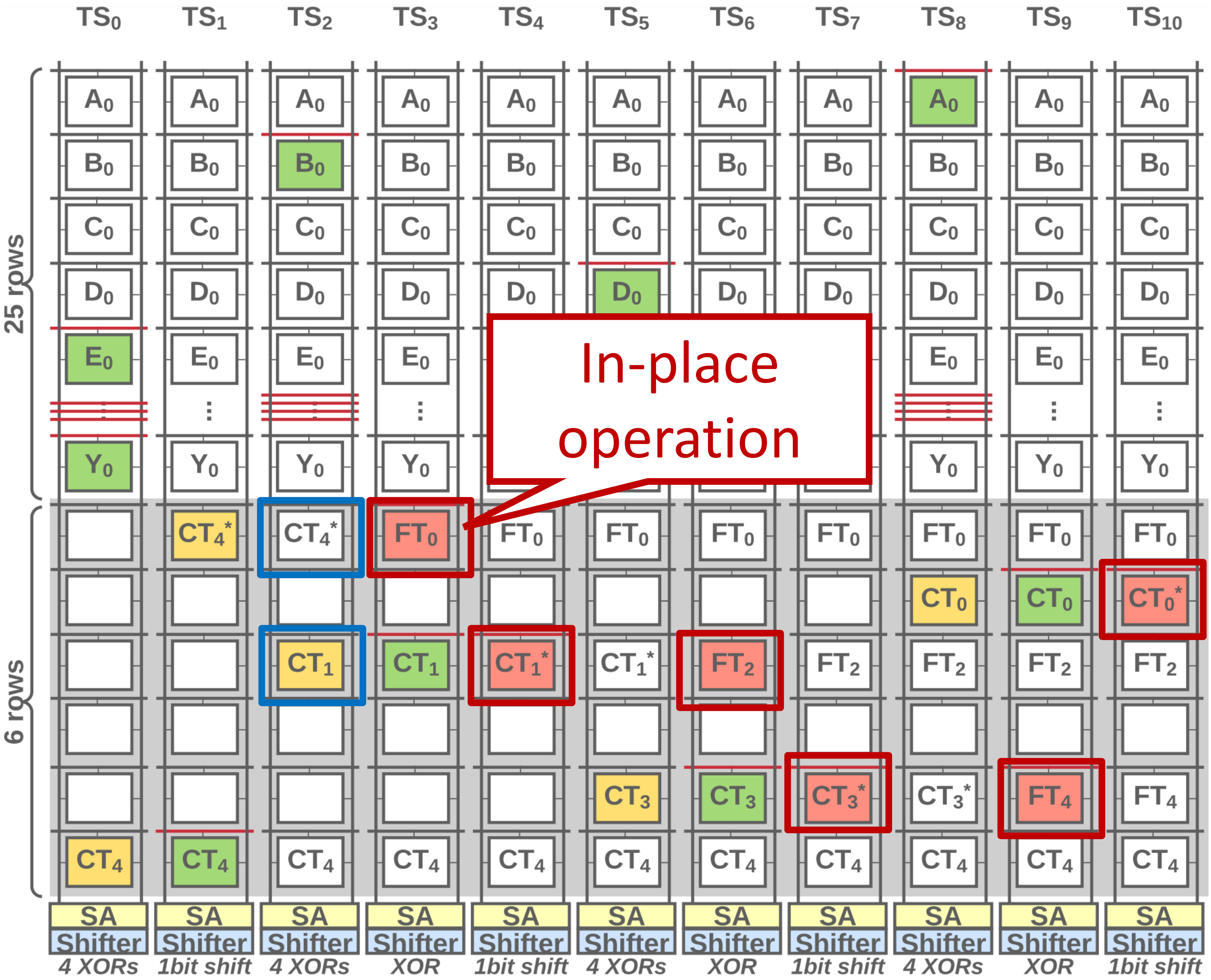
CT_4=XOR(E_0,J_0,O_0,T_0,Y_0)
CT_4^*=rot(CT_4,1)
CT_1=XOR(B_0,G_0,L_0,Q_0,V_0)
FT_0=XOR(CT_1,CT_4^*)

CT_1^*=rot(CT_1,1)
CT_3=XOR(D_0,I_0,N_0,S_0,X_0)
FT_2=XOR(CT_3,CT_1^*)

CT_3^*=rot(CT_3,1)
CT_0=XOR(A_0,F_0,K_0,P_0,U_0)
FT_4=XOR(CT_0,CT_3^*)

CT_0^*=rot(CT_0,1)

Inhale: In-place read/write strategy



	TS$_0$	TS$_1$	TS$_2$	TS$_3$	TS$_4$	TS$_5$	TS$_6$	TS$_7$	TS$_8$	TS$_9$	TS$_{10}$
	A$_0$	A$_0$	A$_0$	A$_0$	A$_0$	A$_0$	A$_0$	A$_0$	A$_0$	A$_0$	A$_0$
	B$_0$	B$_0$	B$_0$	B$_0$	B$_0$	B$_0$	B$_0$	B$_0$	B$_0$	B$_0$	B$_0$
	C$_0$	C$_0$	C$_0$	C$_0$	C$_0$	C$_0$	C$_0$	C$_0$	C$_0$	C$_0$	C$_0$
	D$_0$	D$_0$	D$_0$	D$_0$	D$_0$	D$_0$	D$_0$	D$_0$	D$_0$	D$_0$	D$_0$
25 rows	E$_0$	E$_0$	E$_0$	E$_0$					E$_0$	E$_0$	E$_0$
		⋮		⋮						⋮	⋮
	Y$_0$	Y$_0$	Y$_0$	Y$_0$					Y$_0$	Y$_0$	Y$_0$
		CT$_4^*$	CT$_4^*$	FT$_0$	FT$_0$	FT$_0$	FT$_0$	FT$_0$	FT$_0$	FT$_0$	FT$_0$
									CT$_0$	CT$_0$	CT$_0^*$
6 rows			CT$_1$	CT$_1$	CT$_1^*$	CT$_1^*$	FT$_2$	FT$_2$	FT$_2$	FT$_2$	FT$_2$
						CT$_3$	CT$_3$	CT$_3^*$	CT$_3^*$	FT$_4$	FT$_4$
	CT$_4$	CT$_4$	CT$_4$	CT$_4$	CT$_4$	CT$_4$	CT$_4$	CT$_4$	CT$_4$	CT$_4$	CT$_4$
	SA	SA	SA	SA	SA	SA	SA	SA	SA	SA	SA
	Shifter	Shifter	Shifter	Shifter	Shifter	Shifter	Shifter	Shifter	Shifter	Shifter	Shifter
	4 XORs	*1bit shift*	*4 XORs*	*XOR*	*1bit shift*	*4 XORs*	*XOR*	*1bit shift*	*4 XORs*	*XOR*	*1bit shift*

One round of SHA-3

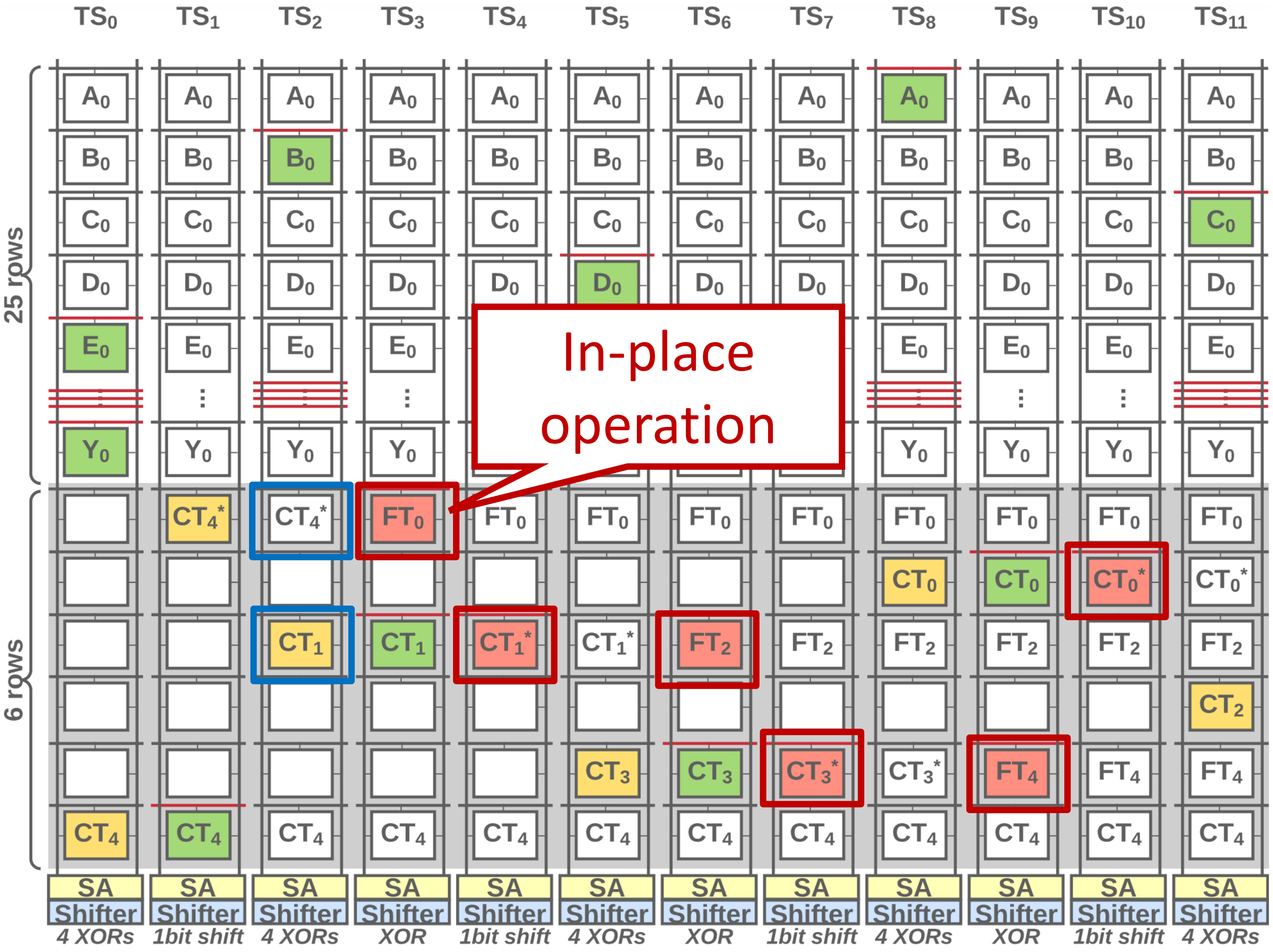
CT_4=XOR(E_0,J_0,O_0,T_0,Y_0)
CT_4^*=rot(CT_4,1)
CT_1=XOR(B_0,G_0,L_0,Q_0,V_0)
FT_0=XOR(CT_1,CT_4^*)

CT_1^*=rot(CT_1,1)
CT_3=XOR(D_0,I_0,N_0,S_0,X_0)
FT_2=XOR(CT_3,CT_1^*)

CT_3^*=rot(CT_3,1)
CT_0=XOR(A_0,F_0,K_0,P_0,U_0)
FT_4=XOR(CT_0,CT_3^*)

CT_0^*=rot(CT_0,1)

Inhale: In-place read/write strategy



TS_0	TS_1	TS_2	TS_3	TS_4	TS_5	TS_6	TS_7	TS_8	TS_9	TS_{10}	TS_{11}
A_0	A_0	A_0	A_0	A_0	A_0	A_0	A_0	A_0	A_0	A_0	A_0
B_0	B_0	B_0	B_0	B_0	B_0	B_0	B_0	B_0	B_0	B_0	B_0
C_0	C_0	C_0	C_0	C_0	C_0	C_0	C_0	C_0	C_0	C_0	C_0
D_0	D_0	D_0	D_0	D_0	D_0	D_0	D_0	D_0	D_0	D_0	D_0
E_0	E_0	E_0	E_0					E_0	E_0	E_0	E_0
⋮	⋮	⋮	⋮					⋮	⋮	⋮	⋮
Y_0	Y_0	Y_0	Y_0					Y_0	Y_0	Y_0	Y_0
	CT_4^*	CT_4^*	FT_0	FT_0	FT_0	FT_0	FT_0	FT_0	FT_0	FT_0	FT_0
								CT_0	CT_0	CT_0^*	CT_0^*
		CT_1	CT_1	CT_1^*	CT_1^*	FT_2	FT_2	FT_2	FT_2	FT_2	FT_2
											CT_2
					CT_3	CT_3	CT_3^*	CT_3^*	FT_4	FT_4	FT_4
CT_4	CT_4	CT_4	CT_4	CT_4	CT_4	CT_4	CT_4	CT_4	CT_4	CT_4	CT_4
SA	SA	SA	SA	SA	SA	SA	SA	SA	SA	SA	SA
Shifter	Shifter	Shifter	Shifter	Shifter	Shifter	Shifter	Shifter	Shifter	Shifter	Shifter	Shifter
4 XORs	*1bit shift*	*4 XORs*	*XOR*	*1bit shift*	*4 XORs*	*XOR*	*1bit shift*	*4 XORs*	*XOR*	*1bit shift*	*4 XORs*

In-place operation

One round of SHA-3

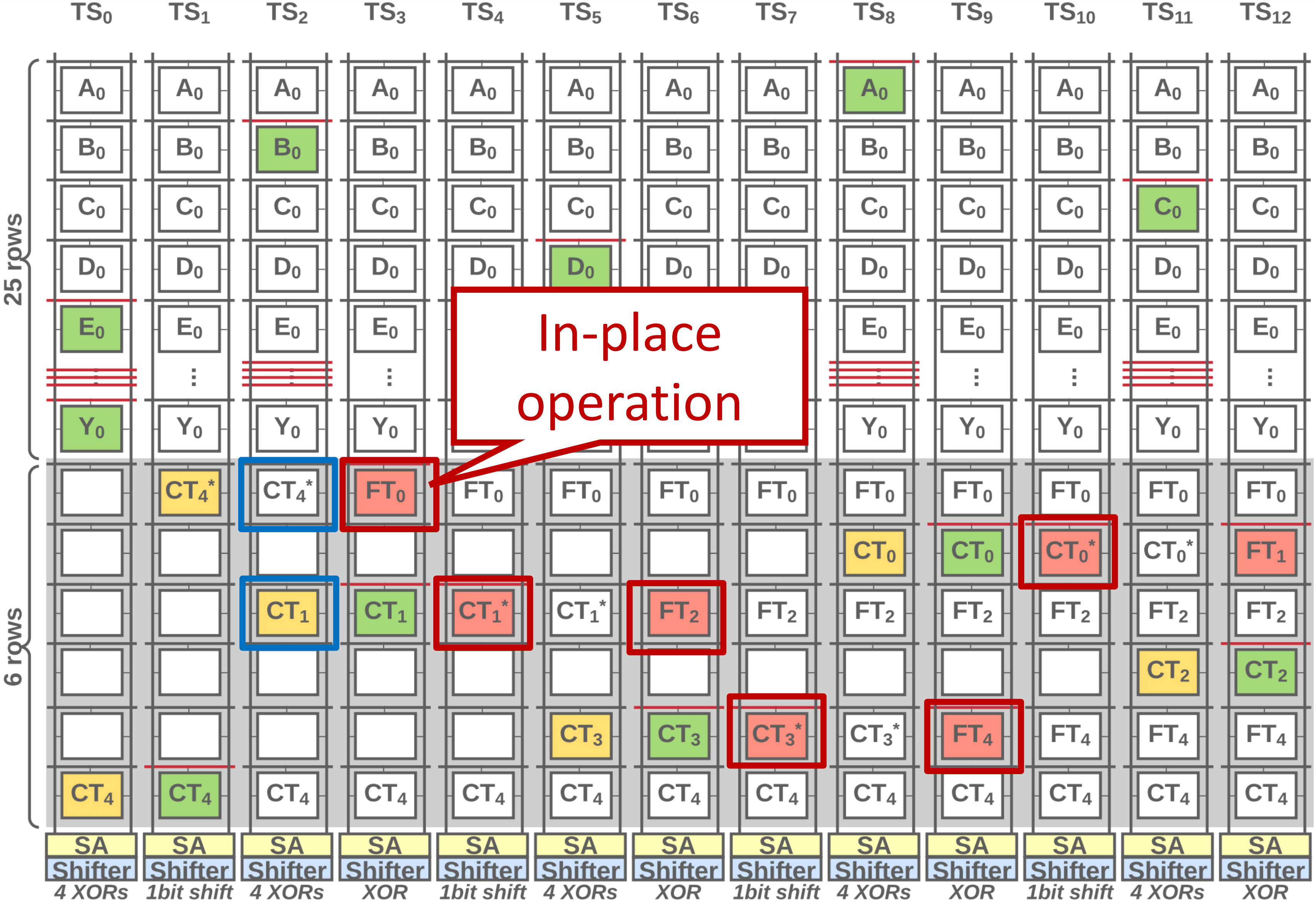
$CT_4=XOR(E_0,J_0,O_0,T_0,Y_0)$
$CT_4^*=rot(CT_4,1)$
$CT_1=XOR(B_0,G_0,L_0,Q_0,V_0)$
$FT_0=XOR(CT_1,CT_4^*)$

$CT_1^*=rot(CT_1,1)$
$CT_3=XOR(D_0,I_0,N_0,S_0,X_0)$
$FT_2=XOR(CT_3,CT_1^*)$

$CT_3^*=rot(CT_3,1)$
$CT_0=XOR(A_0,F_0,K_0,P_0,U_0)$
$FT_4=XOR(CT_0,CT_3^*)$

$CT_0^*=rot(CT_0,1)$
$CT_2=XOR(C_0,H_0,M_0,R_0,W_0)$

Inhale: In-place read/write strategy



	TS$_0$	TS$_1$	TS$_2$	TS$_3$	TS$_4$	TS$_5$	TS$_6$	TS$_7$	TS$_8$	TS$_9$	TS$_{10}$	TS$_{11}$	TS$_{12}$
25 rows	A$_0$	A$_0$	A$_0$	A$_0$	A$_0$	A$_0$	A$_0$	A$_0$	A$_0$	A$_0$	A$_0$	A$_0$	A$_0$
	B$_0$	B$_0$	B$_0$	B$_0$	B$_0$	B$_0$	B$_0$	B$_0$	B$_0$	B$_0$	B$_0$	B$_0$	B$_0$
	C$_0$	C$_0$	C$_0$	C$_0$	C$_0$	C$_0$	C$_0$	C$_0$	C$_0$	C$_0$	C$_0$	C$_0$	C$_0$
	D$_0$	D$_0$	D$_0$	D$_0$	D$_0$	D$_0$	D$_0$	D$_0$	D$_0$	D$_0$	D$_0$	D$_0$	D$_0$
	E$_0$	E$_0$	E$_0$	E$_0$					E$_0$	E$_0$	E$_0$	E$_0$	E$_0$
	⋮	⋮	⋮	⋮					⋮	⋮	⋮	⋮	⋮
	Y$_0$	Y$_0$	Y$_0$	Y$_0$					Y$_0$	Y$_0$	Y$_0$	Y$_0$	Y$_0$
6 rows		CT$_4^*$	CT$_4^*$	FT$_0$	FT$_0$	FT$_0$	FT$_0$	FT$_0$	FT$_0$	FT$_0$	FT$_0$	FT$_0$	FT$_0$
									CT$_0$	CT$_0$	CT$_0^*$	CT$_0^*$	FT$_1$
			CT$_1$	CT$_1$	CT$_1^*$	CT$_1^*$	FT$_2$	FT$_2$	FT$_2$	FT$_2$	FT$_2$	FT$_2$	FT$_2$
												CT$_2$	CT$_2$
						CT$_3$	CT$_3$	CT$_3^*$	CT$_3^*$	FT$_4$	FT$_4$	FT$_4$	FT$_4$
	CT$_4$	CT$_4$	CT$_4$	CT$_4$	CT$_4$	CT$_4$	CT$_4$	CT$_4$	CT$_4$	CT$_4$	CT$_4$	CT$_4$	CT$_4$
	SA	SA	SA	SA	SA	SA	SA	SA	SA	SA	SA	SA	SA
	Shifter	Shifter	Shifter	Shifter	Shifter	Shifter	Shifter	Shifter	Shifter	Shifter	Shifter	Shifter	Shifter
	4 XORs	*1bit shift*	*4 XORs*	*XOR*	*1bit shift*	*4 XORs*	*XOR*	*1bit shift*	*4 XORs*	*XOR*	*1bit shift*	*4 XORs*	*XOR*

One round of SHA-3

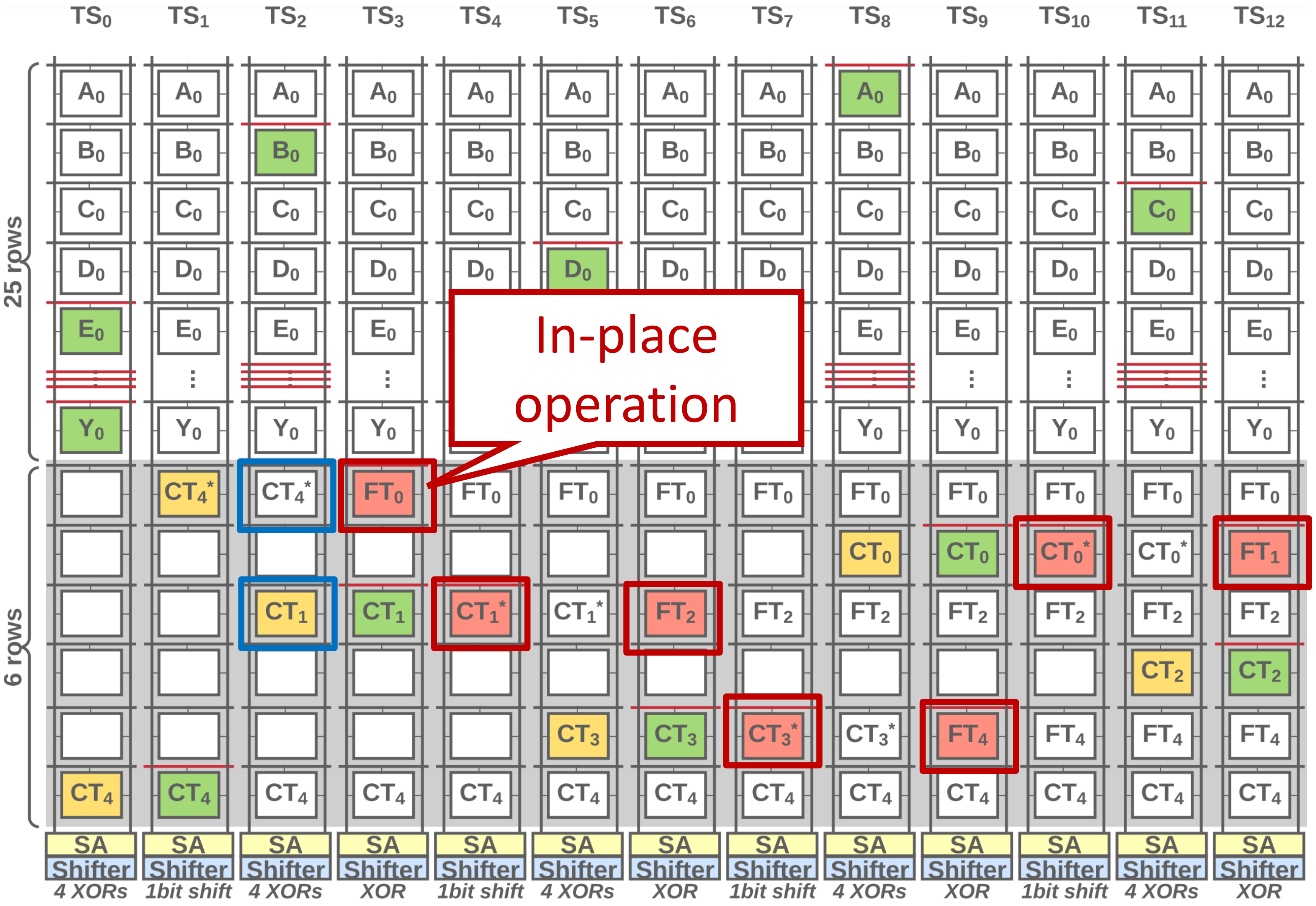
CT_4=XOR(E_0,J_0,O_0,T_0,Y_0)
CT_4^*=rot(CT_4,1)
CT_1=XOR(B_0,G_0,L_0,Q_0,V_0)
FT_0=XOR(CT_1,CT_4^*)

CT_1^*=rot(CT_1,1)
CT_3=XOR(D_0,I_0,N_0,S_0,X_0)
FT_2=XOR(CT_3,CT_1^*)

CT_3^*=rot(CT_3,1)
CT_0=XOR(A_0,F_0,K_0,P_0,U_0)
FT_4=XOR(CT_0,CT_3^*)

CT_0^*=rot(CT_0,1)
CT_2=XOR(C_0,H_0,M_0,R_0,W_0)
FT_1=XOR(CT_2,CT_0^*)

Inhale: In-place read/write strategy



TS	TS_0	TS_1	TS_2	TS_3	TS_4	TS_5	TS_6	TS_7	TS_8	TS_9	TS_{10}	TS_{11}	TS_{12}
25 rows	A_0	A_0	A_0	A_0	A_0	A_0	A_0	A_0	A_0	A_0	A_0	A_0	A_0
	B_0	B_0	B_0	B_0	B_0	B_0	B_0	B_0	B_0	B_0	B_0	B_0	B_0
	C_0	C_0	C_0	C_0	C_0	C_0	C_0	C_0	C_0	C_0	C_0	C_0	C_0
	D_0	D_0	D_0	D_0	D_0	D_0	D_0	D_0	D_0	D_0	D_0	D_0	D_0
	E_0	E_0	E_0	E_0					E_0	E_0	E_0	E_0	E_0
	⋮	⋮	⋮	⋮					⋮	⋮	⋮	⋮	⋮
	Y_0	Y_0	Y_0	Y_0					Y_0	Y_0	Y_0	Y_0	Y_0
6 rows		CT_4^*	CT_4^*	FT_0	FT_0	FT_0	FT_0	FT_0	FT_0	FT_0	FT_0	FT_0	FT_0
									CT_0	CT_0	CT_0^*	CT_0^*	FT_1
			CT_1	CT_1	CT_1^*	CT_1^*	FT_2	FT_2	FT_2	FT_2	FT_2	FT_2	FT_2
												CT_2	CT_2
						CT_3	CT_3	CT_3^*	CT_3^*	FT_4	FT_4	FT_4	FT_4
	CT_4	CT_4	CT_4	CT_4	CT_4	CT_4	CT_4	CT_4	CT_4	CT_4	CT_4	CT_4	CT_4
	SA	SA	SA	SA	SA	SA	SA	SA	SA	SA	SA	SA	SA
	Shifter	Shifter	Shifter	Shifter	Shifter	Shifter	Shifter	Shifter	Shifter	Shifter	Shifter	Shifter	Shifter
	4 XORs	*1bit shift*	*4 XORs*	*XOR*	*1bit shift*	*4 XORs*	*XOR*	*1bit shift*	*4 XORs*	*XOR*	*1bit shift*	*4 XORs*	*XOR*

One round of SHA-3

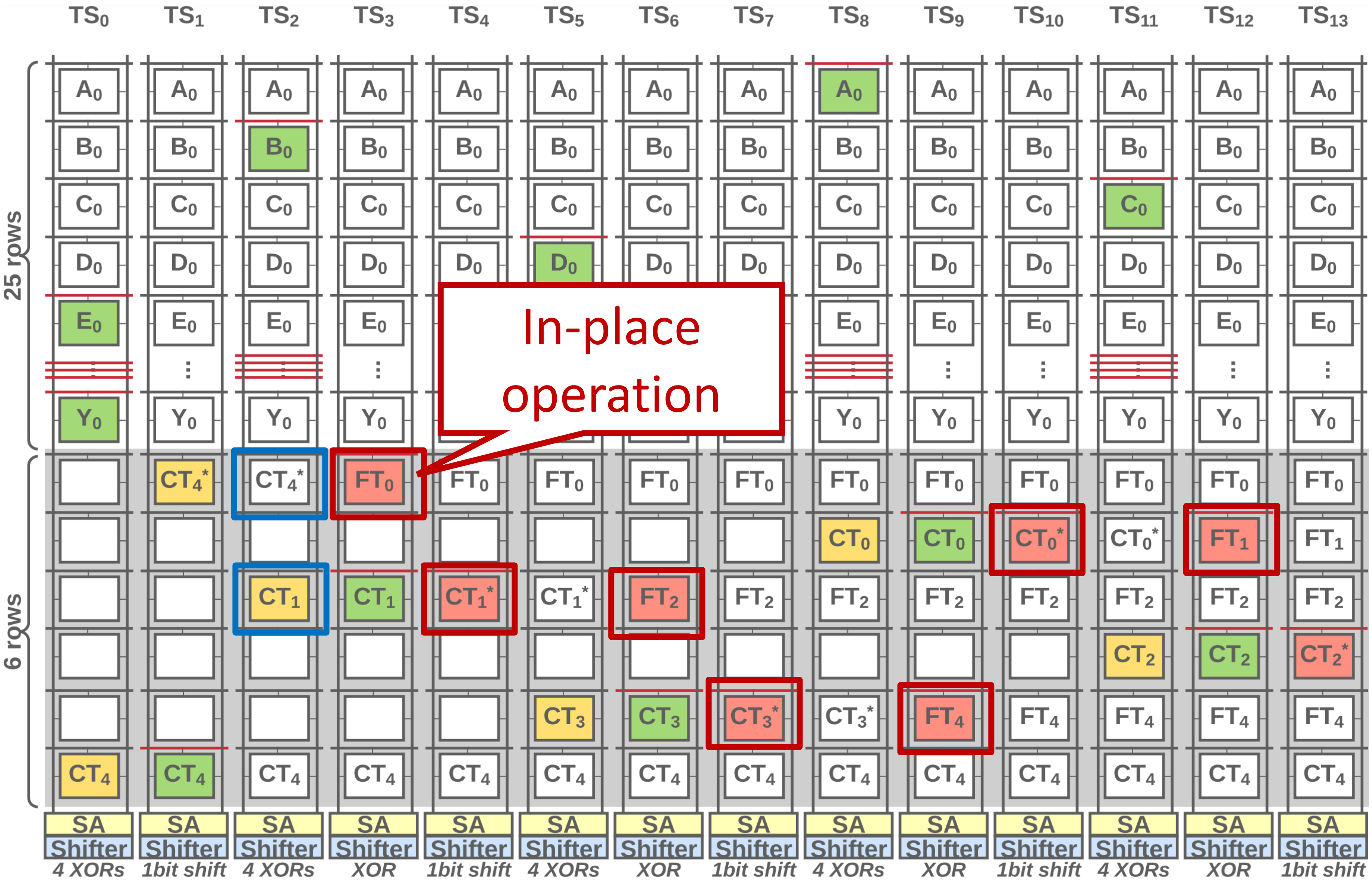
CT_4=XOR(E_0,J_0,O_0,T_0,Y_0)
CT_4^*=rot(CT_4,1)
CT_1=XOR(B_0,G_0,L_0,Q_0,V_0)
FT_0=XOR(CT_1,CT_4^*)

CT_1^*=rot(CT_1,1)
CT_3=XOR(D_0,I_0,N_0,S_0,X_0)
FT_2=XOR(CT_3,CT_1^*)

CT_3^*=rot(CT_3,1)
CT_0=XOR(A_0,F_0,K_0,P_0,U_0)
FT_4=XOR(CT_0,CT_3^*)

CT_0^*=rot(CT_0,1)
CT_2=XOR(C_0,H_0,M_0,R_0,W_0)
FT_1=XOR(CT_2,CT_0^*)

Inhale: In-place read/write strategy



	TS$_0$	TS$_1$	TS$_2$	TS$_3$	TS$_4$	TS$_5$	TS$_6$	TS$_7$	TS$_8$	TS$_9$	TS$_{10}$	TS$_{11}$	TS$_{12}$	TS$_{13}$
25 rows	A$_0$	A$_0$	A$_0$	A$_0$	A$_0$	A$_0$	A$_0$	A$_0$	A$_0$	A$_0$	A$_0$	A$_0$	A$_0$	A$_0$
	B$_0$	B$_0$	B$_0$	B$_0$	B$_0$	B$_0$	B$_0$	B$_0$	B$_0$	B$_0$	B$_0$	B$_0$	B$_0$	B$_0$
	C$_0$	C$_0$	C$_0$	C$_0$	C$_0$	C$_0$	C$_0$	C$_0$	C$_0$	C$_0$	C$_0$	C$_0$	C$_0$	C$_0$
	D$_0$	D$_0$	D$_0$	D$_0$	D$_0$	D$_0$	D$_0$	D$_0$	D$_0$	D$_0$	D$_0$	D$_0$	D$_0$	D$_0$
	E$_0$	E$_0$	E$_0$	E$_0$					E$_0$	E$_0$	E$_0$	E$_0$	E$_0$	E$_0$
	⋮	⋮	⋮	⋮					⋮	⋮	⋮	⋮	⋮	⋮
	Y$_0$	Y$_0$	Y$_0$	Y$_0$					Y$_0$	Y$_0$	Y$_0$	Y$_0$	Y$_0$	Y$_0$
6 rows		CT$_4^*$	CT$_4^*$	FT$_0$	FT$_0$	FT$_0$	FT$_0$	FT$_0$	FT$_0$	FT$_0$	FT$_0$	FT$_0$	FT$_0$	FT$_0$
									CT$_0$	CT$_0$	CT$_0^*$	CT$_0^*$	FT$_1$	FT$_1$
			CT$_1$	CT$_1$	CT$_1^*$	CT$_1^*$	FT$_2$	FT$_2$	FT$_2$	FT$_2$	FT$_2$	FT$_2$	FT$_2$	FT$_2$
												CT$_2$	CT$_2$	CT$_2^*$
						CT$_3$	CT$_3$	CT$_3^*$	CT$_3^*$	FT$_4$	FT$_4$	FT$_4$	FT$_4$	FT$_4$
	CT$_4$	CT$_4$	CT$_4$	CT$_4$	CT$_4$	CT$_4$	CT$_4$	CT$_4$	CT$_4$	CT$_4$	CT$_4$	CT$_4$	CT$_4$	CT$_4$
	SA	SA	SA	SA	SA	SA	SA	SA	SA	SA	SA	SA	SA	SA
	Shifter	Shifter	Shifter	Shifter	Shifter	Shifter	Shifter	Shifter	Shifter	Shifter	Shifter	Shifter	Shifter	Shifter
	4 XORs	*1bit shift*	*4 XORs*	*XOR*	*1bit shift*	*4 XORs*	*XOR*	*1bit shift*	*4 XORs*	*XOR*	*1bit shift*	*4 XORs*	*XOR*	*1bit shift*

One round of SHA-3

CT_4=XOR(E_0,J_0,O_0,T_0,Y_0)
CT_4^*=rot(CT_4,1)
CT_1=XOR(B_0,G_0,L_0,Q_0,V_0)
FT_0=XOR(CT_1,CT_4^*)

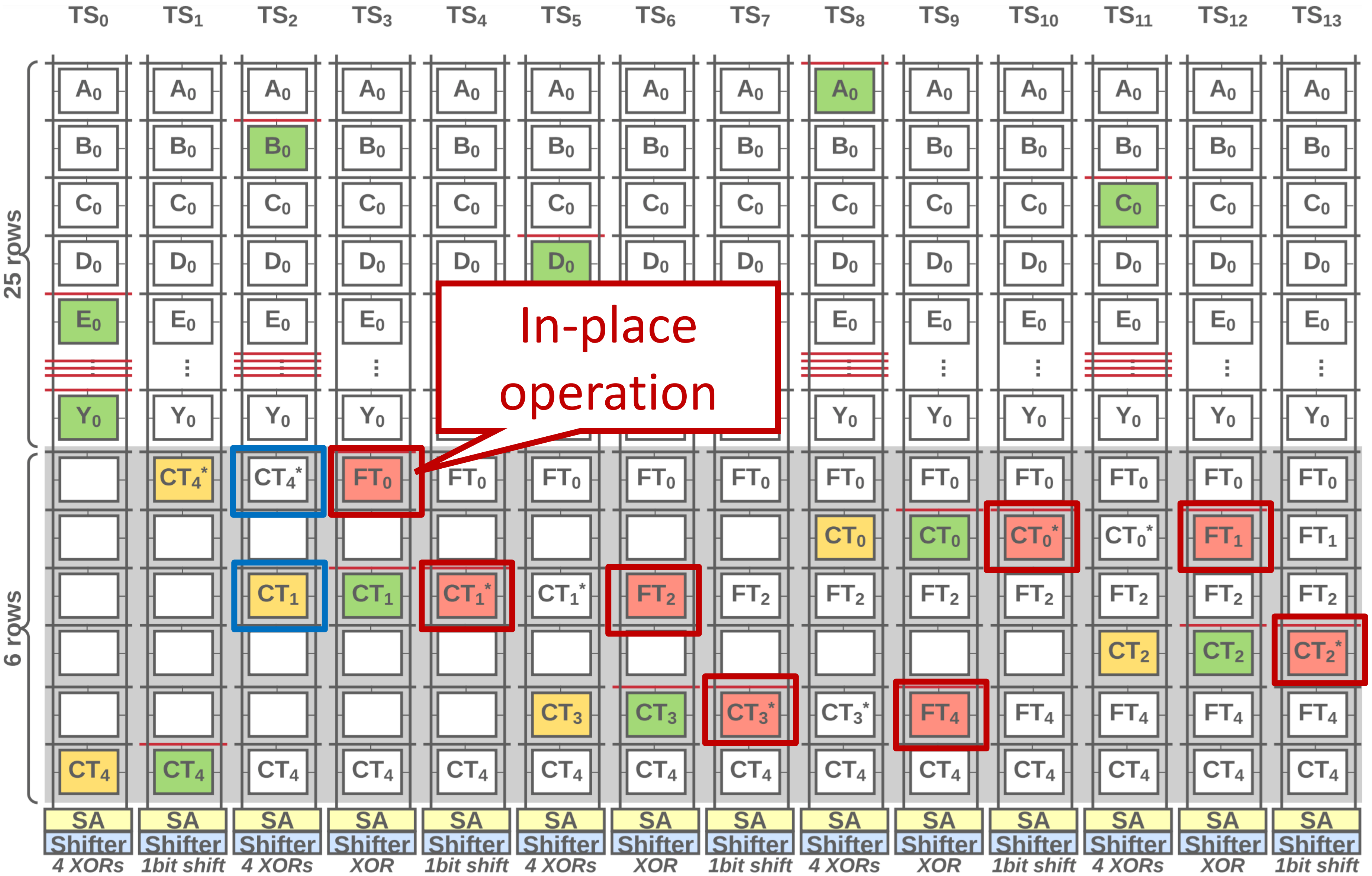
CT_1^*=rot(CT_1,1)
CT_3=XOR(D_0,I_0,N_0,S_0,X_0)
FT_2=XOR(CT_3,CT_1^*)

CT_3^*=rot(CT_3,1)
CT_0=XOR(A_0,F_0,K_0,P_0,U_0)
FT_4=XOR(CT_0,CT_3^*)

CT_0^*=rot(CT_0,1)
CT_2=XOR(C_0,H_0,M_0,R_0,W_0)
FT_1=XOR(CT_2,CT_0^*)

CT_2^*=rot(CT_2,1)

Inhale: In-place read/write strategy



	TS

One round of SHA-3

CT_4=XOR(E_0,J_0,O_0,T_0,Y_0)
CT_4^*=rot(CT_4,1)
CT_1=XOR(B_0,G_0,L_0,Q_0,V_0)
FT_0=XOR(CT_1,CT_4^*)

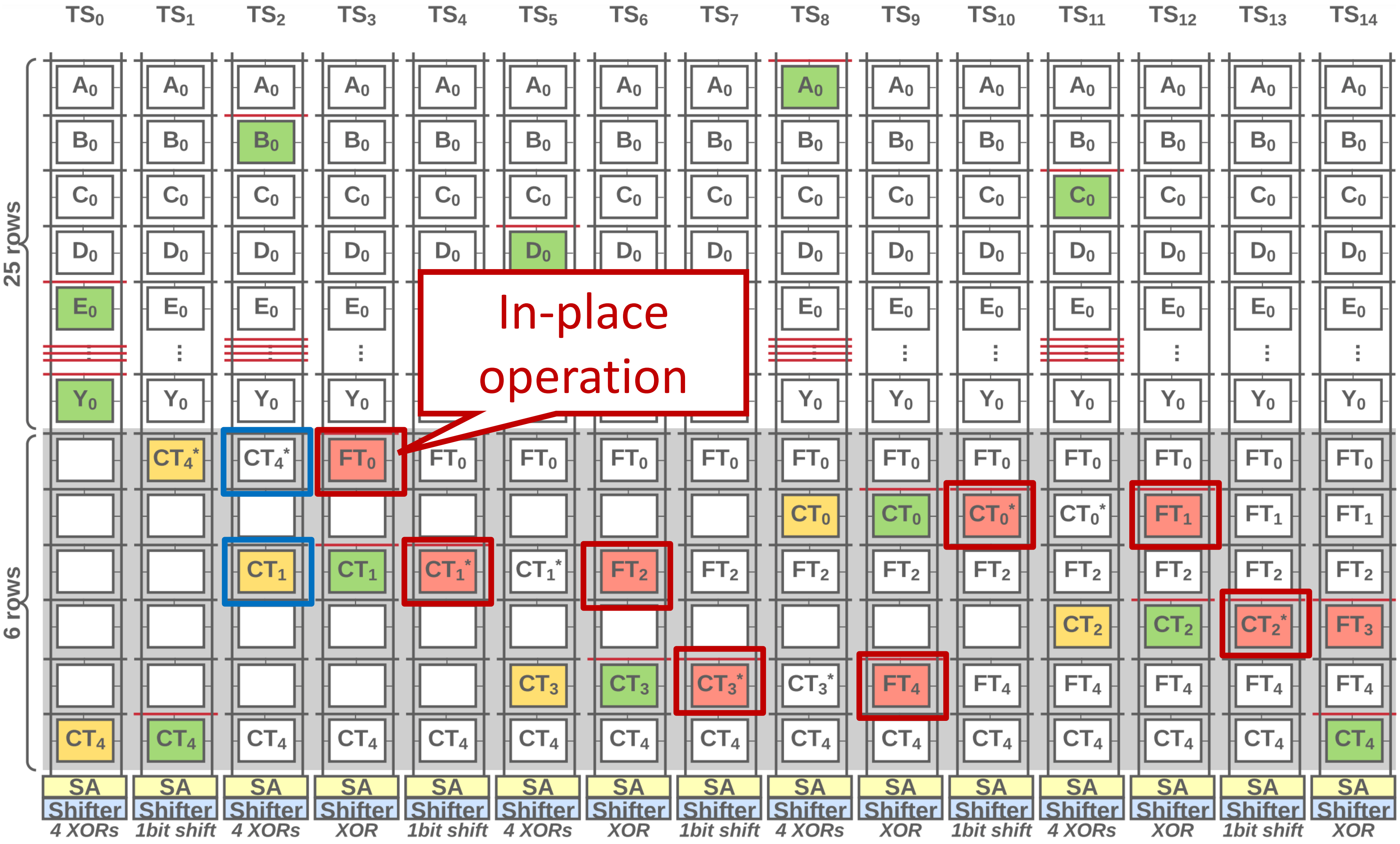
CT_1^*=rot(CT_1,1)
CT_3=XOR(D_0,I_0,N_0,S_0,X_0)
FT_2=XOR(CT_3,CT_1^*)

CT_3^*=rot(CT_3,1)
CT_0=XOR(A_0,F_0,K_0,P_0,U_0)
FT_4=XOR(CT_0,CT_3^*)

CT_0^*=rot(CT_0,1)
CT_2=XOR(C_0,H_0,M_0,R_0,W_0)
FT_1=XOR(CT_2,CT_0^*)

CT_2^*=rot(CT_2,1)

Inhale: In-place read/write strategy



TS$_0$	TS$_1$	TS$_2$	TS$_3$	TS$_4$	TS$_5$	TS$_6$	TS$_7$	TS$_8$	TS$_9$	TS$_{10}$	TS$_{11}$	TS$_{12}$	TS$_{13}$	TS$_{14}$
A$_0$	A$_0$	A$_0$	A$_0$	A$_0$	A$_0$	A$_0$	A$_0$	A$_0$	A$_0$	A$_0$	A$_0$	A$_0$	A$_0$	A$_0$
B$_0$	B$_0$	B$_0$	B$_0$	B$_0$	B$_0$	B$_0$	B$_0$	B$_0$	B$_0$	B$_0$	B$_0$	B$_0$	B$_0$	B$_0$
C$_0$	C$_0$	C$_0$	C$_0$	C$_0$	C$_0$	C$_0$	C$_0$	C$_0$	C$_0$	C$_0$	C$_0$	C$_0$	C$_0$	C$_0$
D$_0$	D$_0$	D$_0$	D$_0$	D$_0$	D$_0$	D$_0$	D$_0$	D$_0$	D$_0$	D$_0$	D$_0$	D$_0$	D$_0$	D$_0$
E$_0$	E$_0$	E$_0$	E$_0$					E$_0$	E$_0$	E$_0$	E$_0$	E$_0$	E$_0$	E$_0$
⋮	⋮	⋮	⋮					⋮	⋮	⋮	⋮	⋮	⋮	⋮
Y$_0$	Y$_0$	Y$_0$	Y$_0$					Y$_0$	Y$_0$	Y$_0$	Y$_0$	Y$_0$	Y$_0$	Y$_0$
	CT$_4^*$	CT$_4^*$	FT$_0$	FT$_0$	FT$_0$	FT$_0$	FT$_0$	FT$_0$	FT$_0$	FT$_0$	FT$_0$	FT$_0$	FT$_0$	FT$_0$
								CT$_0$	CT$_0$	CT$_0^*$	CT$_0^*$	FT$_1$	FT$_1$	FT$_1$
		CT$_1$	CT$_1$	CT$_1^*$	CT$_1^*$	FT$_2$	FT$_2$	FT$_2$	FT$_2$	FT$_2$	FT$_2$	FT$_2$	FT$_2$	FT$_2$
											CT$_2$	CT$_2$	CT$_2^*$	FT$_3$
					CT$_3$	CT$_3$	CT$_3^*$	CT$_3^*$	FT$_4$	FT$_4$	FT$_4$	FT$_4$	FT$_4$	FT$_4$
CT$_4$	CT$_4$	CT$_4$	CT$_4$	CT$_4$	CT$_4$	CT$_4$	CT$_4$	CT$_4$	CT$_4$	CT$_4$	CT$_4$	CT$_4$	CT$_4$	CT$_4$
SA	SA	SA	SA	SA	SA	SA	SA	SA	SA	SA	SA	SA	SA	SA
Shifter	Shifter	Shifter	Shifter	Shifter	Shifter	Shifter	Shifter	Shifter	Shifter	Shifter	Shifter	Shifter	Shifter	Shifter
4 XORs	*1bit shift*	*4 XORs*	*XOR*	*1bit shift*	*4 XORs*	*XOR*	*1bit shift*	*4 XORs*	*XOR*	*1bit shift*	*4 XORs*	*XOR*	*1bit shift*	*XOR*

One round of SHA-3

CT_4=XOR(E_0,J_0,O_0,T_0,Y_0)
CT_4^*=rot(CT_4,1)
CT_1=XOR(B_0,G_0,L_0,Q_0,V_0)
FT_0=XOR(CT_1,CT_4^*)

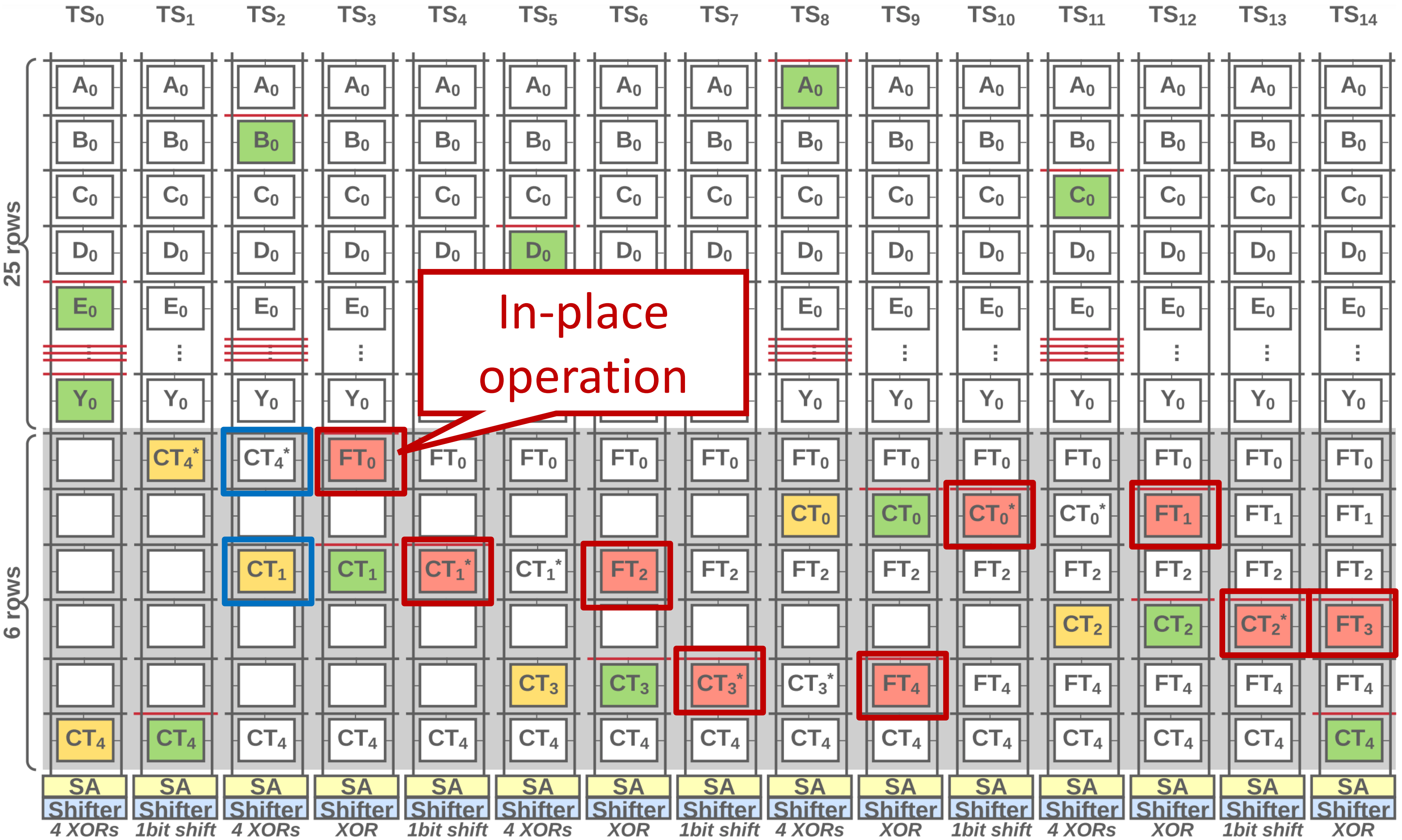
CT_1^*=rot(CT_1,1)
CT_3=XOR(D_0,I_0,N_0,S_0,X_0)
FT_2=XOR(CT_3,CT_1^*)

CT_3^*=rot(CT_3,1)
CT_0=XOR(A_0,F_0,K_0,P_0,U_0)
FT_4=XOR(CT_0,CT_3^*)

CT_0^*=rot(CT_0,1)
CT_2=XOR(C_0,H_0,M_0,R_0,W_0)
FT_1=XOR(CT_2,CT_0^*)

CT_2^*=rot(CT_2,1)
FT_3=XOR(CT_4,CT_2^*)

Inhale: In-place read/write strategy



TS_0	TS_1	TS_2	TS_3	TS_4	TS_5	TS_6	TS_7	TS_8	TS_9	TS_{10}	TS_{11}	TS_{12}	TS_{13}	TS_{14}
A_0	A_0	A_0	A_0	A_0	A_0	A_0	A_0	A_0	A_0	A_0	A_0	A_0	A_0	A_0
B_0	B_0	B_0	B_0	B_0	B_0	B_0	B_0	B_0	B_0	B_0	B_0	B_0	B_0	B_0
C_0	C_0	C_0	C_0	C_0	C_0	C_0	C_0	C_0	C_0	C_0	C_0	C_0	C_0	C_0
D_0	D_0	D_0	D_0	D_0	D_0	D_0	D_0	D_0	D_0	D_0	D_0	D_0	D_0	D_0
E_0	E_0	E_0	E_0					E_0	E_0	E_0	E_0	E_0	E_0	E_0
⋮	⋮	⋮	⋮					⋮	⋮	⋮	⋮	⋮	⋮	⋮
Y_0	Y_0	Y_0	Y_0					Y_0	Y_0	Y_0	Y_0	Y_0	Y_0	Y_0
	CT_4^*	CT_4^*	FT_0	FT_0	FT_0	FT_0	FT_0	FT_0	FT_0	FT_0	FT_0	FT_0	FT_0	FT_0
								CT_0	CT_0	CT_0^*	CT_0^*	FT_1	FT_1	FT_1
		CT_1	CT_1	CT_1^*	CT_1^*	FT_2	FT_2	FT_2	FT_2	FT_2	FT_2	FT_2	FT_2	FT_2
											CT_2	CT_2	CT_2^*	FT_3
					CT_3	CT_3	CT_3^*	CT_3^*	FT_4	FT_4	FT_4	FT_4	FT_4	FT_4
CT_4	CT_4	CT_4	CT_4	CT_4	CT_4	CT_4	CT_4	CT_4	CT_4	CT_4	CT_4	CT_4	CT_4	CT_4
SA	SA	SA	SA	SA	SA	SA	SA	SA	SA	SA	SA	SA	SA	SA
Shifter	Shifter	Shifter	Shifter	Shifter	Shifter	Shifter	Shifter	Shifter	Shifter	Shifter	Shifter	Shifter	Shifter	Shifter
4 XORs	*1bit shift*	*4 XORs*	*XOR*	*1bit shift*	*4 XORs*	*XOR*	*1bit shift*	*4 XORs*	*XOR*	*1bit shift*	*4 XORs*	*XOR*	*1bit shift*	*XOR*

One round of SHA-3

CT_4=XOR(E_0,J_0,O_0,T_0,Y_0)
CT_4^*=rot(CT_4,1)
CT_1=XOR(B_0,G_0,L_0,Q_0,V_0)
FT_0=XOR(CT_1,CT_4^*)

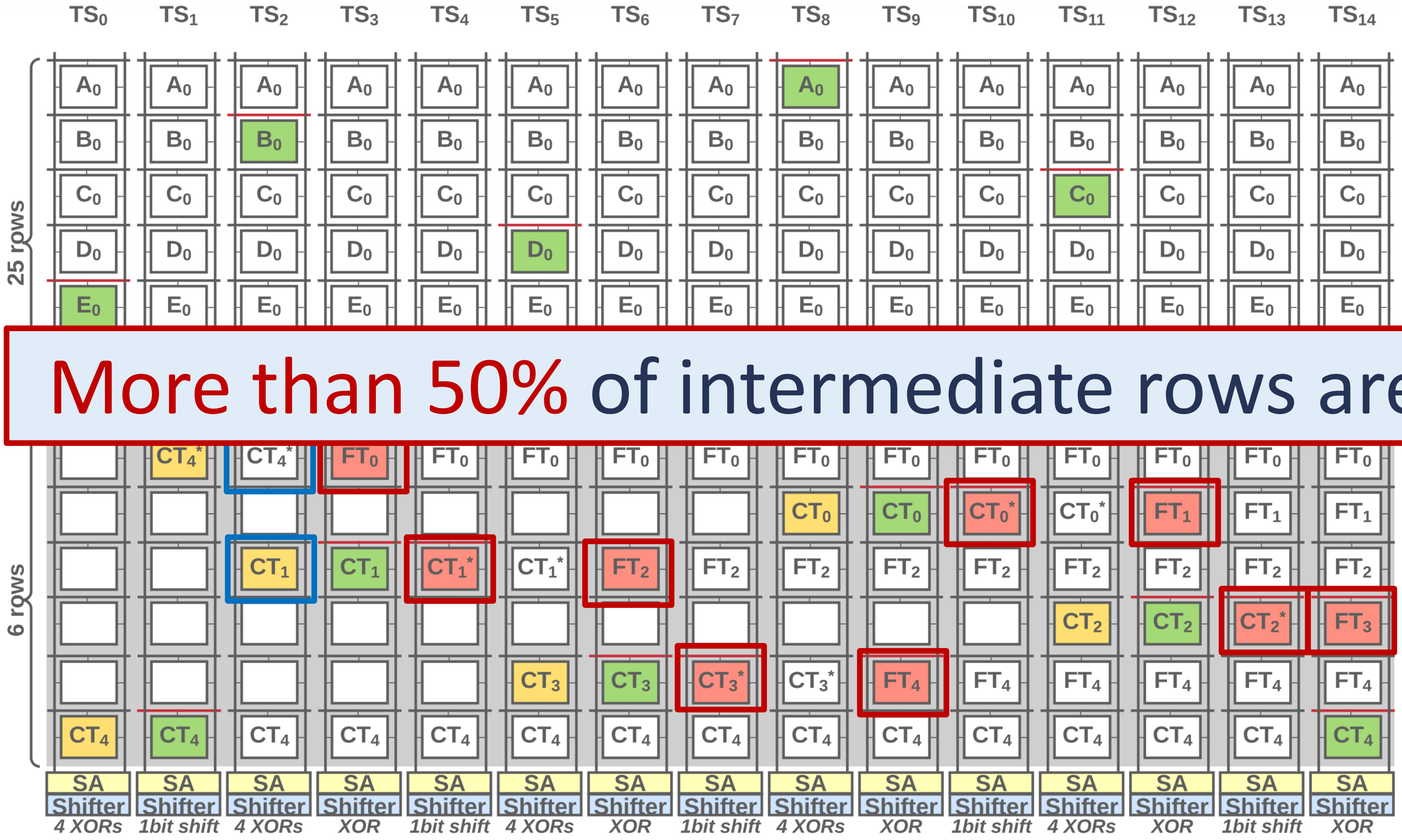
CT_1^*=rot(CT_1,1)
CT_3=XOR(D_0,I_0,N_0,S_0,X_0)
FT_2=XOR(CT_3,CT_1^*)

CT_3^*=rot(CT_3,1)
CT_0=XOR(A_0,F_0,K_0,P_0,U_0)
FT_4=XOR(CT_0,CT_3^*)

CT_0^*=rot(CT_0,1)
CT_2=XOR(C_0,H_0,M_0,R_0,W_0)
FT_1=XOR(CT_2,CT_0^*)

CT_2^*=rot(CT_2,1)
FT_3=XOR(CT_4,CT_2^*)

Inhale: In-place read/write strategy



TS_0	TS_1	TS_2	TS_3	TS_4	TS_5	TS_6	TS_7	TS_8	TS_9	TS_{10}	TS_{11}	TS_{12}	TS_{13}	TS_{14}
A_0	A_0	A_0	A_0	A_0	A_0	A_0	A_0	A_0	A_0	A_0	A_0	A_0	A_0	A_0
B_0	B_0	B_0	B_0	B_0	B_0	B_0	B_0	B_0	B_0	B_0	B_0	B_0	B_0	B_0
C_0	C_0	C_0	C_0	C_0	C_0	C_0	C_0	C_0	C_0	C_0	C_0	C_0	C_0	C_0
D_0	D_0	D_0	D_0	D_0	D_0	D_0	D_0	D_0	D_0	D_0	D_0	D_0	D_0	D_0
E_0	E_0	E_0	E_0	E_0	E_0	E_0	E_0	E_0	E_0	E_0	E_0	E_0	E_0	E_0
	CT_4^*	CT_4^*	FT_0	FT_0	FT_0	FT_0	FT_0	FT_0	FT_0	FT_0	FT_0	FT_0	FT_0	FT_0
								CT_0	CT_0	CT_0^*	CT_0^*	FT_1	FT_1	FT_1
		CT_1	CT_1	CT_1^*	CT_1^*	FT_2	FT_2	FT_2	FT_2	FT_2	FT_2	FT_2	FT_2	FT_2
											CT_2	CT_2	CT_2^*	FT_3
					CT_3	CT_3	CT_3^*	CT_3^*	FT_4	FT_4	FT_4	FT_4	FT_4	FT_4
CT_4	CT_4	CT_4	CT_4	CT_4	CT_4	CT_4	CT_4	CT_4	CT_4	CT_4	CT_4	CT_4	CT_4	CT_4
SA	SA	SA	SA	SA	SA	SA	SA	SA	SA	SA	SA	SA	SA	SA
Shifter	Shifter	Shifter	Shifter	Shifter	Shifter	Shifter	Shifter	Shifter	Shifter	Shifter	Shifter	Shifter	Shifter	Shifter
4 XORs	*1bit shift*	*4 XORs*	*XOR*	*1bit shift*	*4 XORs*	*XOR*	*1bit shift*	*4 XORs*	*XOR*	*1bit shift*	*4 XORs*	*XOR*	*1bit shift*	*XOR*

More than 50% of intermediate rows are saved

One round of SHA-3

CT_4=XOR(E_0,J_0,O_0,T_0,Y_0)
CT_4^*=rot(CT_4,1)
CT_1=XOR(B_0,G_0,L_0,Q_0,V_0)
FT_0=XOR(CT_1,CT_4^*)

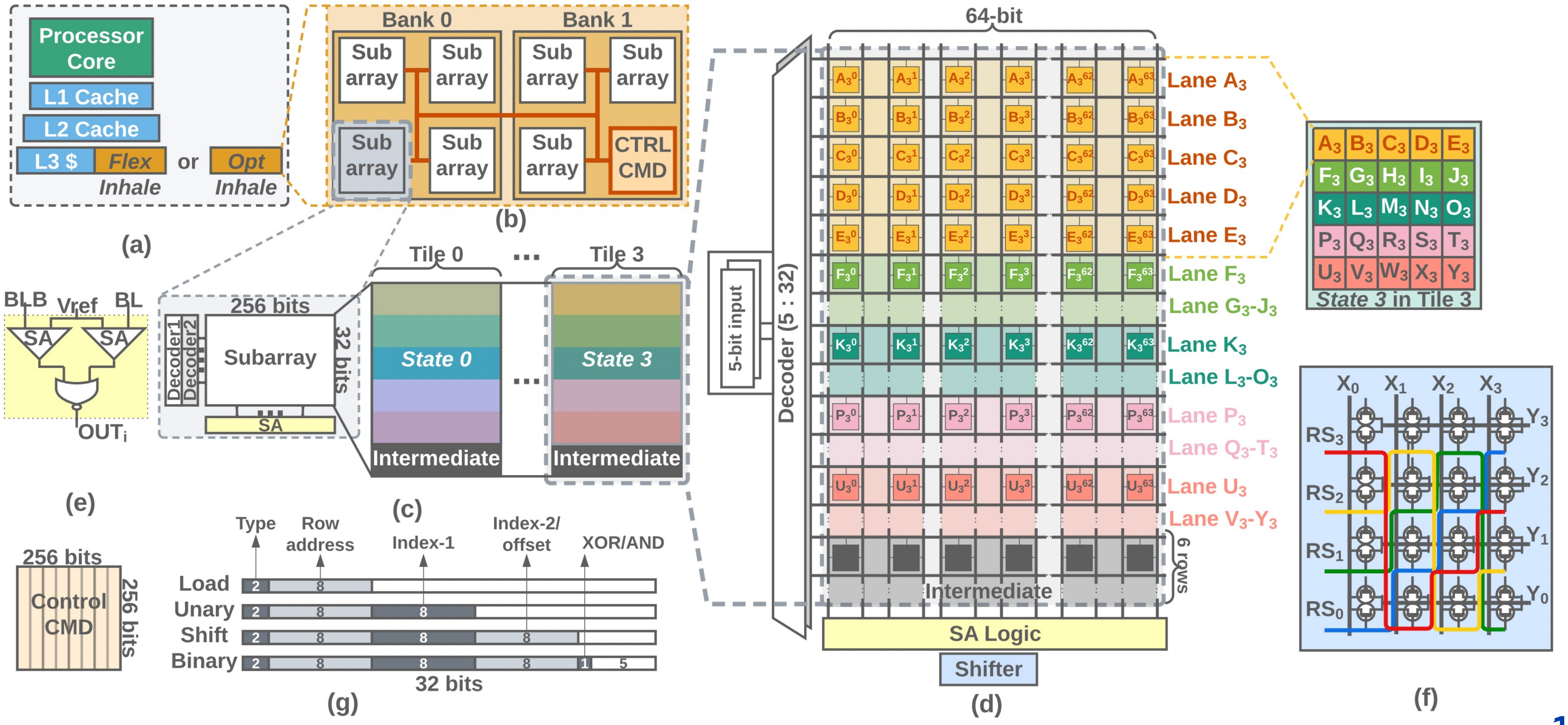
CT_1^*=rot(CT_1,1)
CT_3=XOR(D_0,I_0,N_0,S_0,X_0)
FT_2=XOR(CT_3,CT_1^*)

CT_0^*=rot(CT_0,1)
CT_2=XOR(C_0,H_0,M_0,R_0,W_0)
FT_1=XOR(CT_2,CT_0^*)

CT_2^*=rot(CT_2,1)
FT_3=XOR(CT_4,CT_2^*)

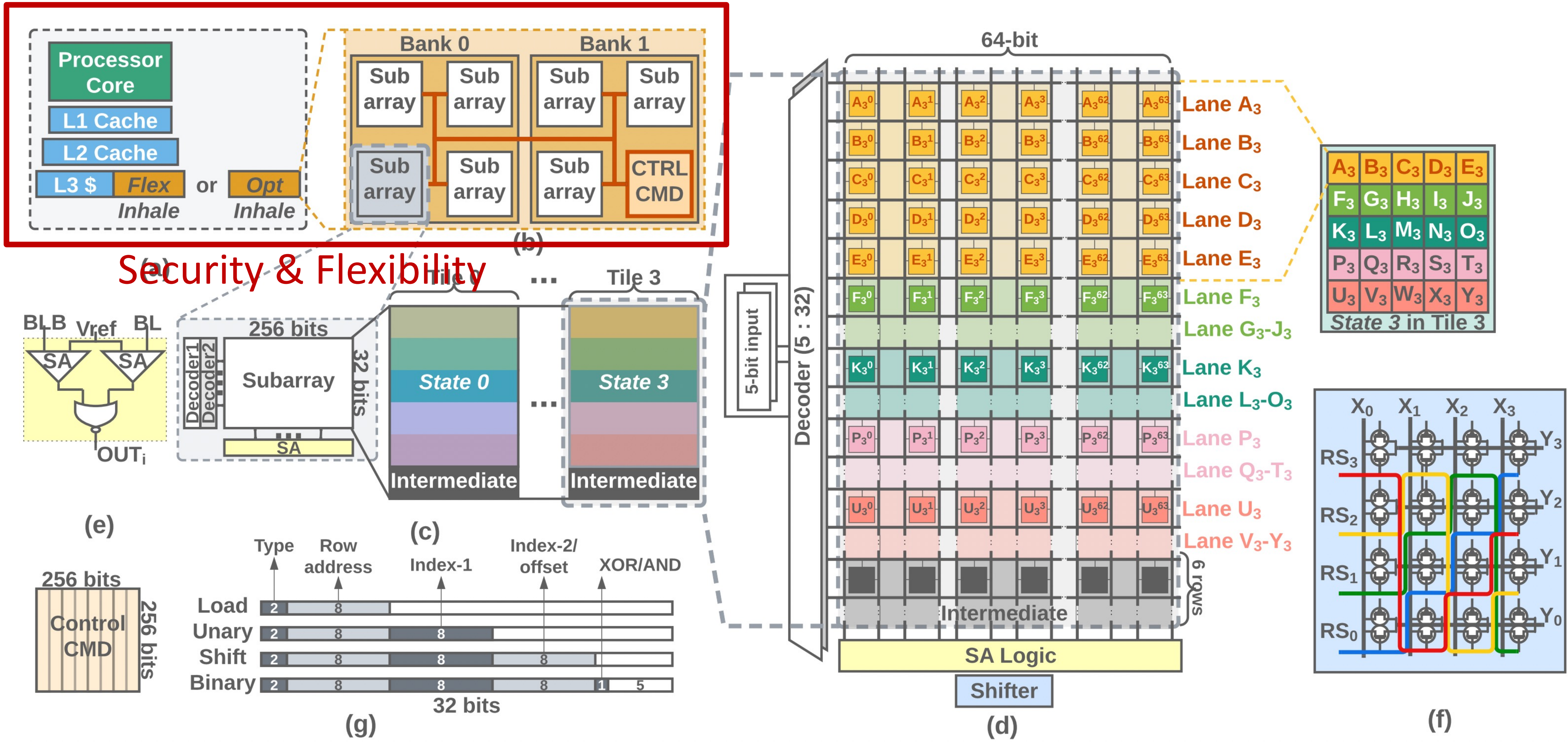
Inhale: Overall Architecture

High-performance, energy-efficient and low-overhead hashing engine



Inhale: Overall Architecture

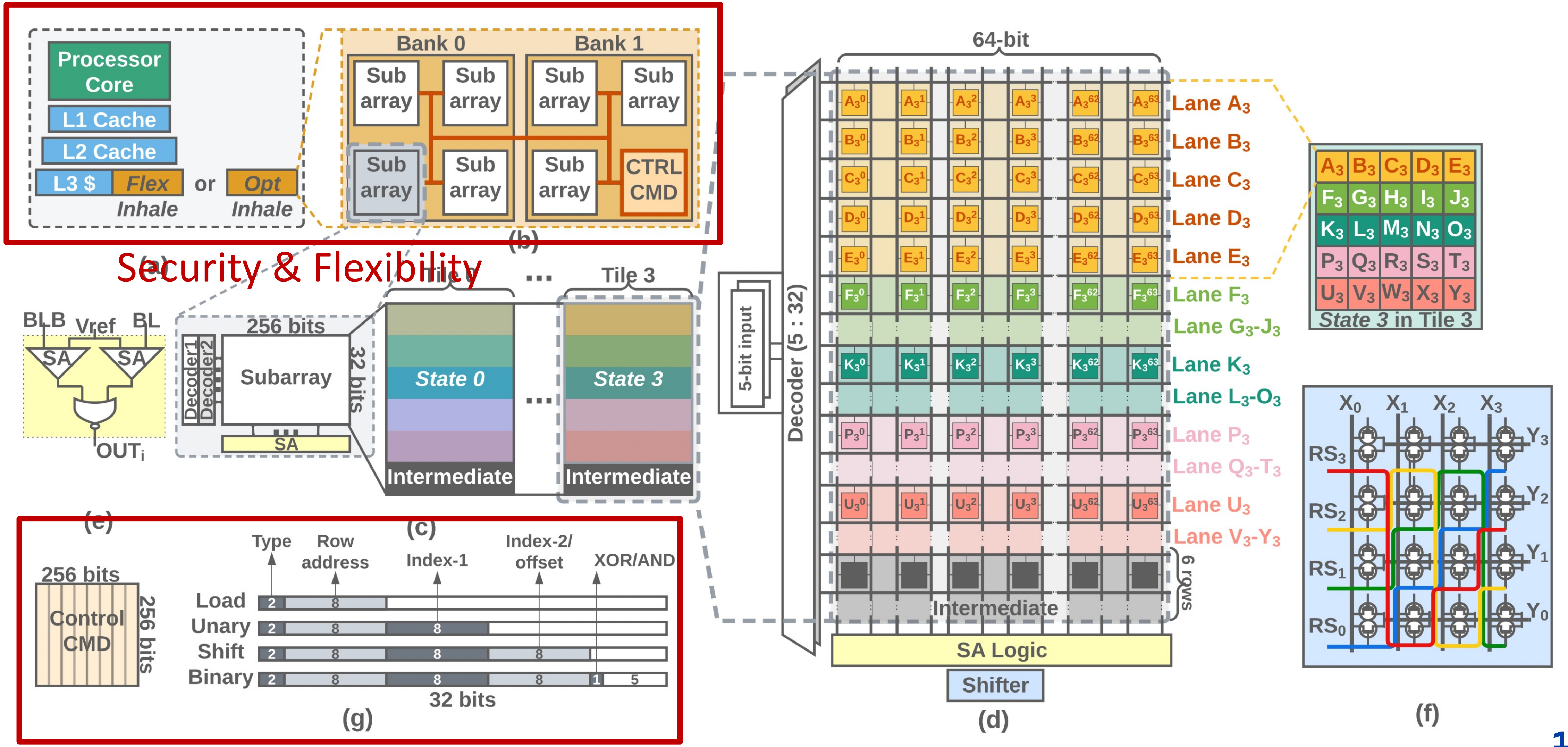
High-performance, energy-efficient and low-overhead hashing engine



Security & Flexibility

Inhale: Overall Architecture

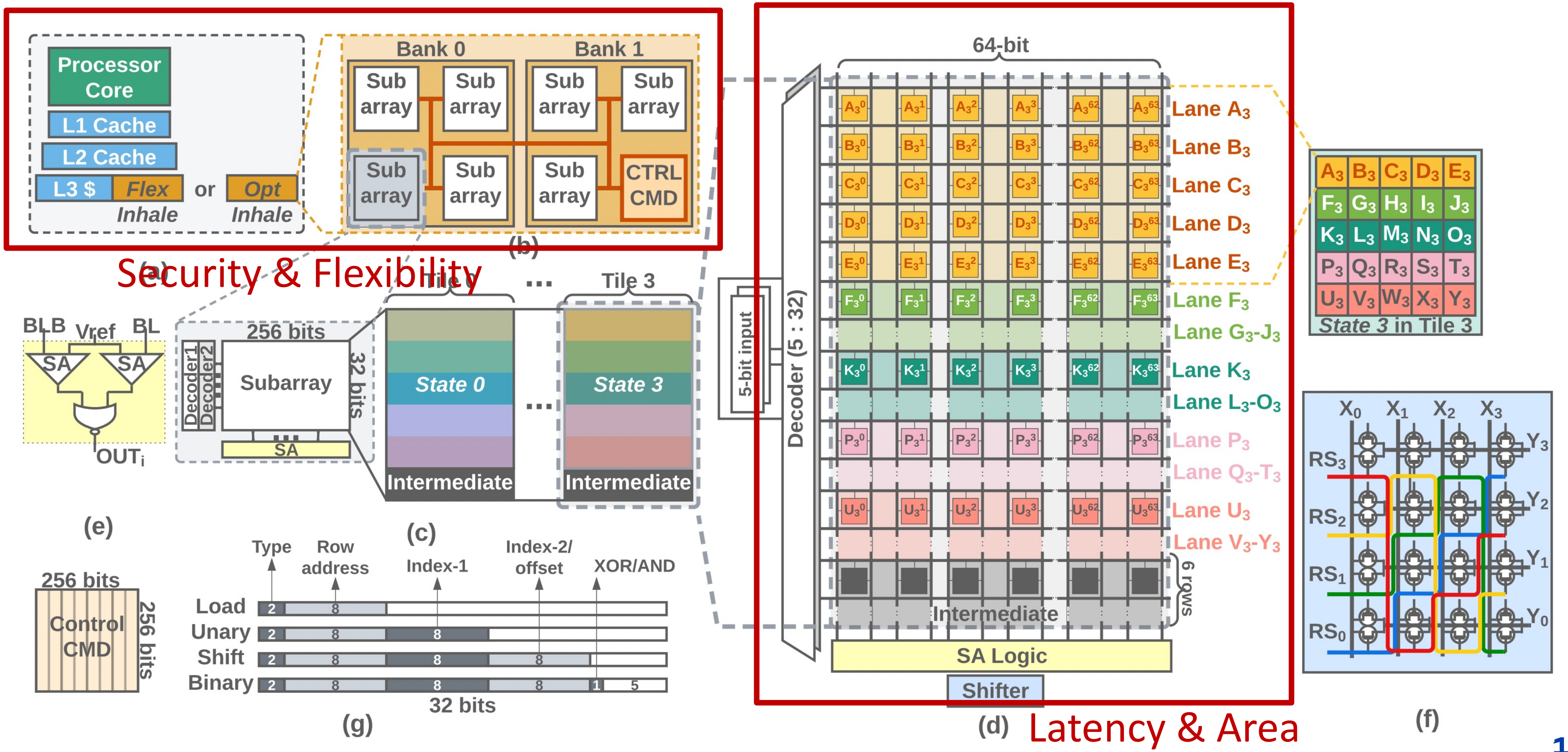
High-performance, energy-efficient and low-overhead hashing engine


Processor Core
L1 Cache
L2 Cache
L3 $
Flex Inhale
or
Opt Inhale
(a)
Bank 0
Bank 1
Sub array
CTRL CMD
(b)
Tile 0
...
Tile 3
256 bits
Decoder1
Decoder2
Subarray
32 bits
SA
State 0
State 3
Intermediate
(c)
BLB
Vref
BL
SA
OUT_i
(e)
64-bit
5-bit input
Decoder (5 : 32)
A_3^0 A_3^1 A_3^2 A_3^3 A_3^{62} A_3^{63}
B_3^0 B_3^1 B_3^2 B_3^3 B_3^{62} B_3^{63}
C_3^0 C_3^1 C_3^2 C_3^3 C_3^{62} C_3^{63}
D_3^0 D_3^1 D_3^2 D_3^3 D_3^{62} D_3^{63}
E_3^0 E_3^1 E_3^2 E_3^3 E_3^{62} E_3^{63}
F_3^0 F_3^1 F_3^2 F_3^3 F_3^{62} F_3^{63}
K_3^0 K_3^1 K_3^2 K_3^3 K_3^{62} K_3^{63}
P_3^0 P_3^1 P_3^2 P_3^3 P_3^{62} P_3^{63}
U_3^0 U_3^1 U_3^2 U_3^3 U_3^{62} U_3^{63}
Lane A_3
Lane B_3
Lane C_3
Lane D_3
Lane E_3
Lane F_3
Lane G_3-J_3
Lane K_3
Lane L_3-O_3
Lane P_3
Lane Q_3-T_3
Lane U_3
Lane V_3-Y_3
Intermediate
6 rows
SA Logic
Shifter
(d)
A_3 B_3 C_3 D_3 E_3
F_3 G_3 H_3 I_3 J_3
K_3 L_3 M_3 N_3 O_3
P_3 Q_3 R_3 S_3 T_3
U_3 V_3 W_3 X_3 Y_3
State 3 in Tile 3
X_0 X_1 X_2 X_3
RS_3
RS_2
RS_1
RS_0
Y_3
Y_2
Y_1
Y_0
(f)
256 bits
Control CMD
256 bits
Type
Row address
Index-1
Index-2/ offset
XOR/AND
Load
Unary
Shift
Binary
2
8
1
5
32 bits
(g)

Security & Flexibility

Inhale: Overall Architecture

High-performance, energy-efficient and low-overhead hashing engine

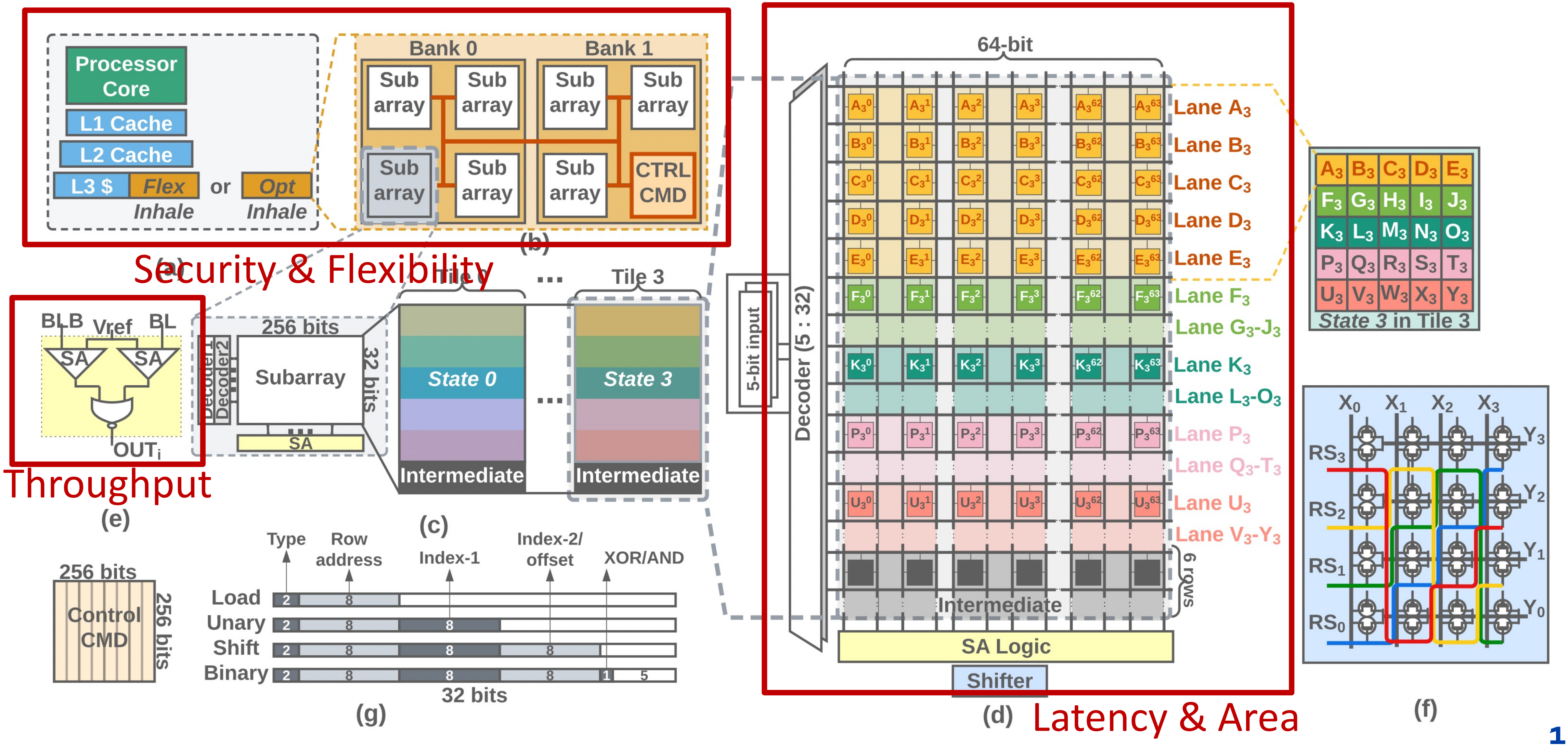


Security & Flexibility

Latency & Area

Inhale: Overall Architecture

High-performance, energy-efficient and low-overhead hashing engine



Security & Flexibility

Throughput

Latency & Area

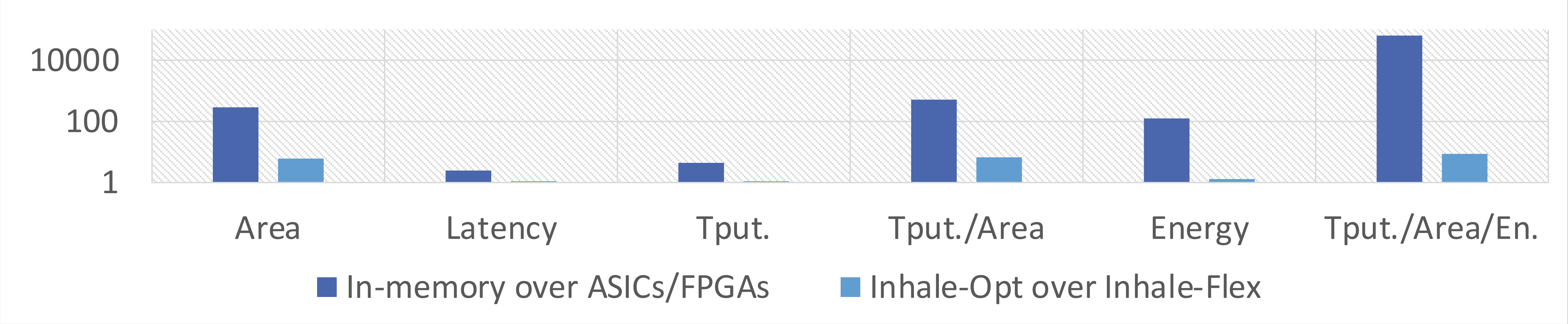
Evaluation Methodology

- Read and write latency:
  - PyMTL3 and OpenRAM for generating SRAM arrays
  - Synopsys Design Compiler for extracting latencies
  - Latencies of ReRAM array from DESTINY simulator
- Area and energy numbers simulated by DESTINY simulator
  - Kilo Gate Equivalent (KGE) is used to decouple the area overhead from the technology node
- For apples-to-apples comparison between different designs
  - *Inhale* and SHINE in 28nm ReRAM and SRAM are all evaluated

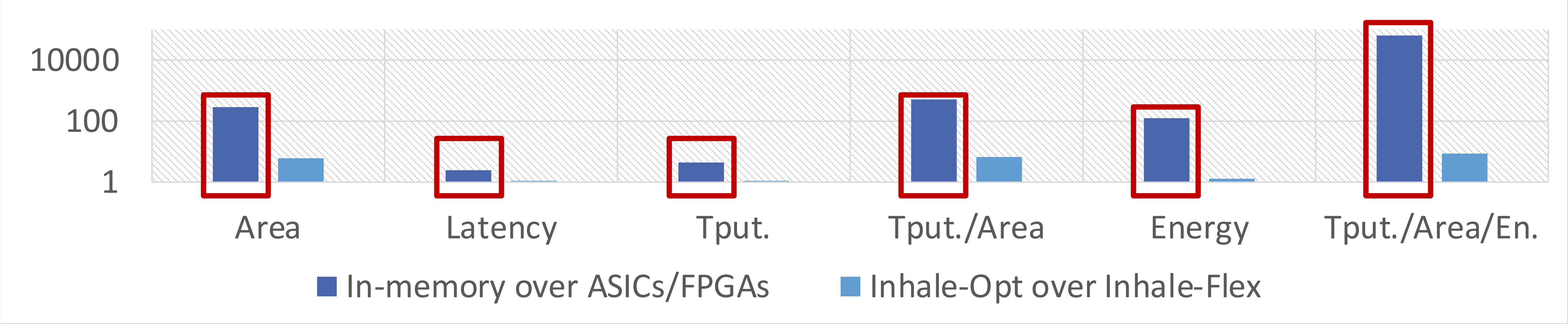
Jiang, Shunning, et al. “PyMTL3: A Python framework for open-source hardware modeling, generation, simulation, and verification.” MICRO’20.
Guthaus, Matthew R., et al. "OpenRAM: An open-source memory compiler." ICCAD’16.
Poremba, Matt, et al. "Destiny: A tool for modeling emerging 3d nvm and edram caches." DATE’15.
Nagarajan, Karthikeyan, et al. "SHINE: A novel SHA-3 implementation using ReRAM-based in-memory computing." ISLPED’19

Comparison of different designs

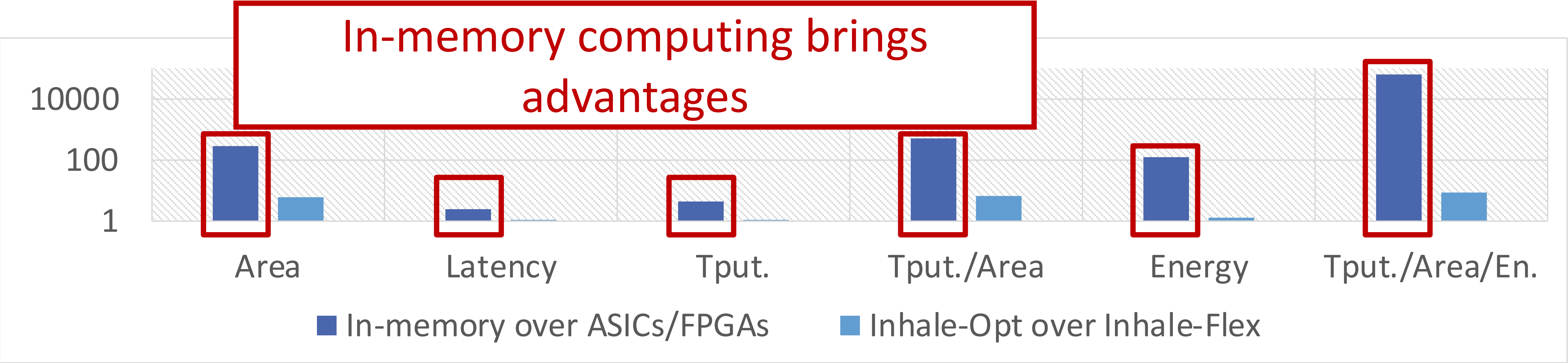
Comparison of different designs



Comparison of different designs

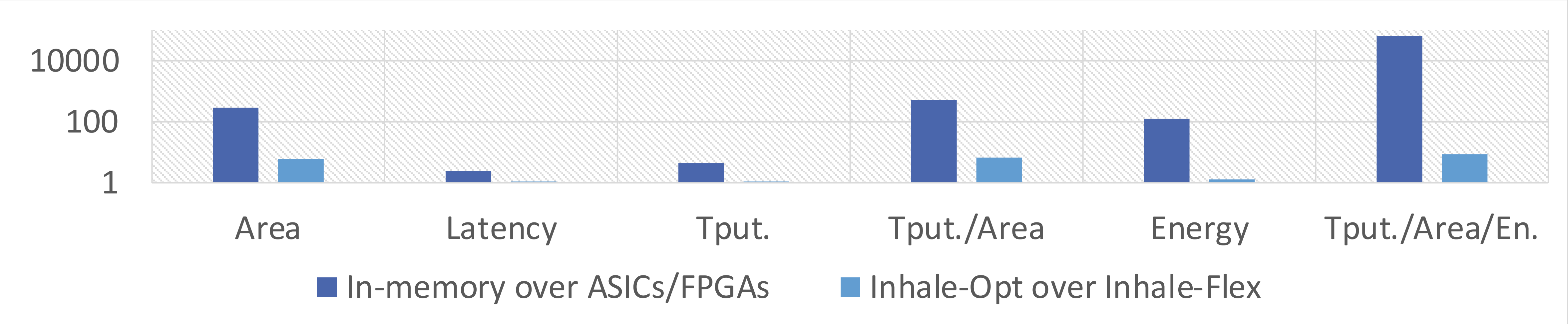


Comparison of different designs

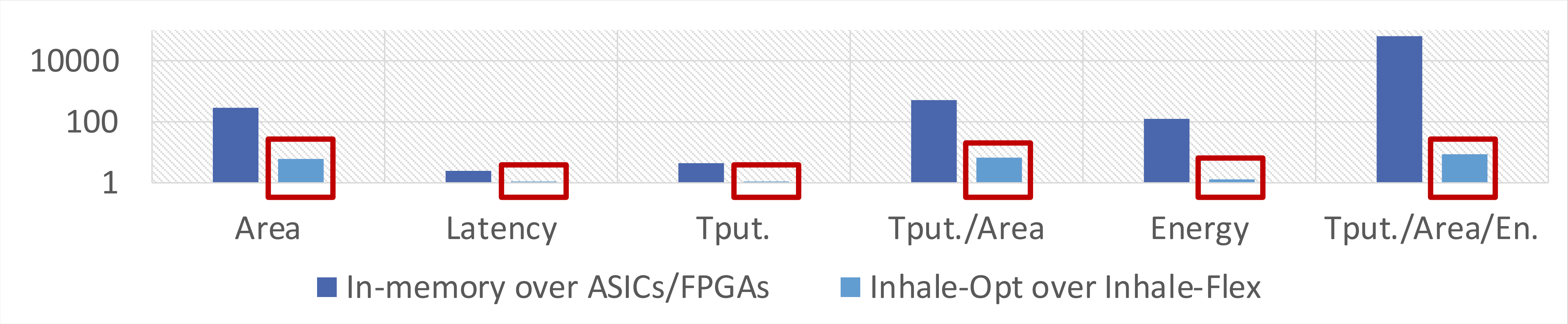


In-memory computing brings advantages

Comparison of different designs

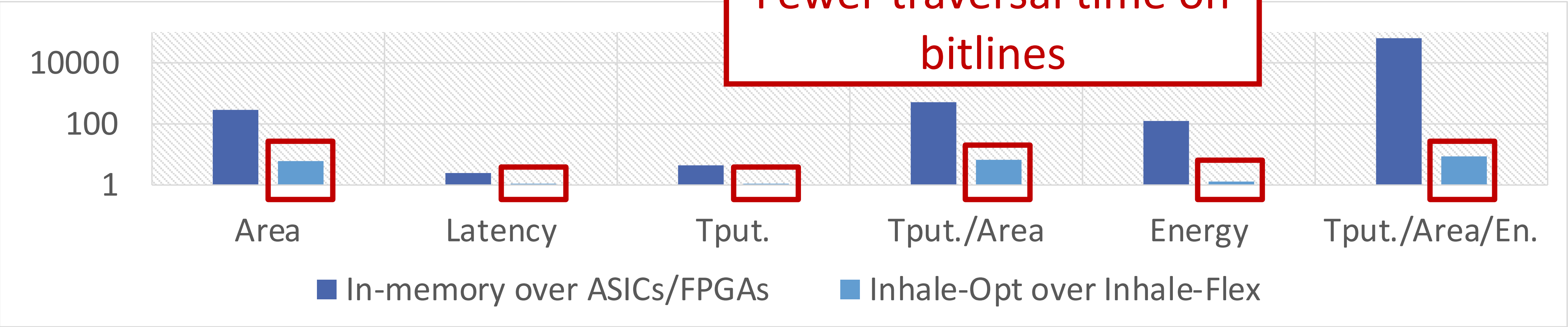


Comparison of different designs

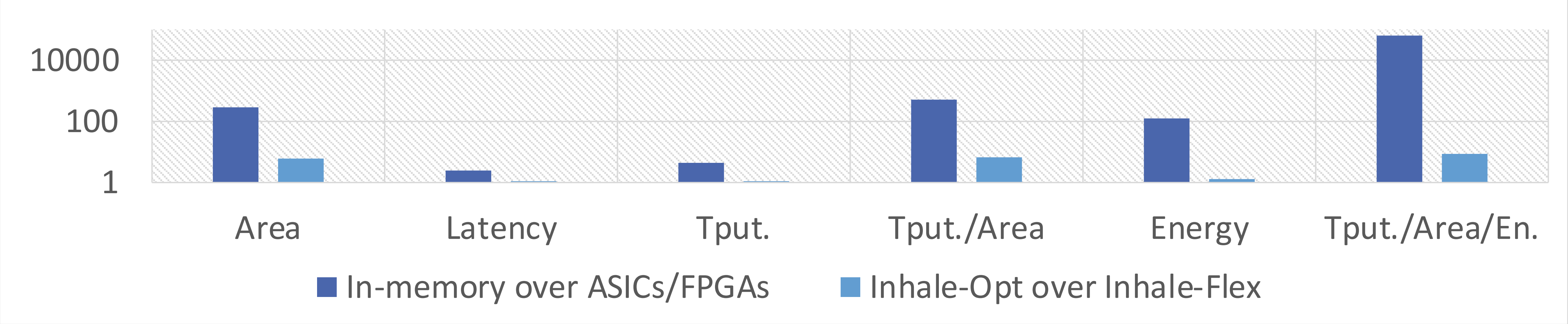


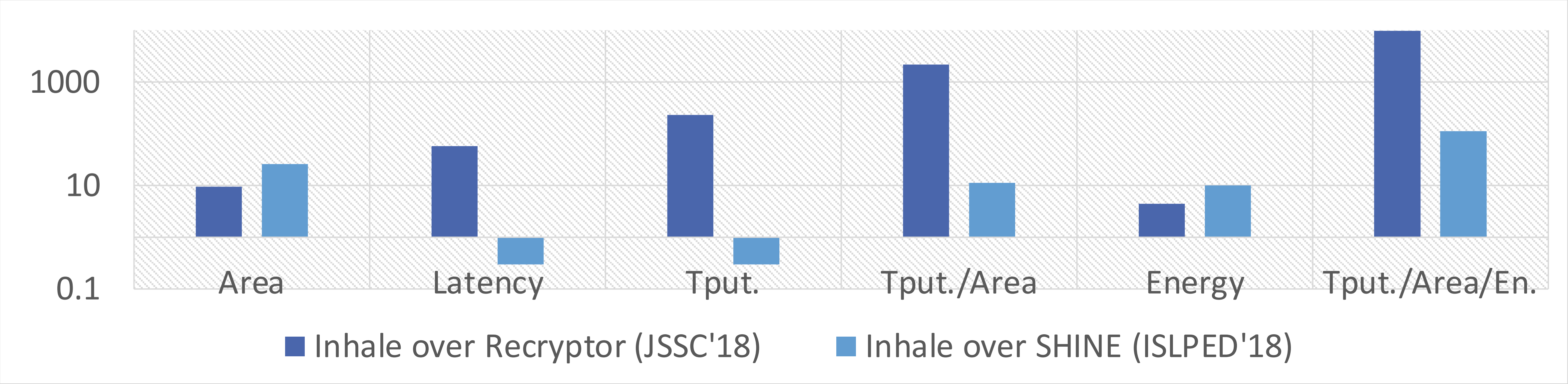
Comparison of different designs

Fewer traversal time on bitlines

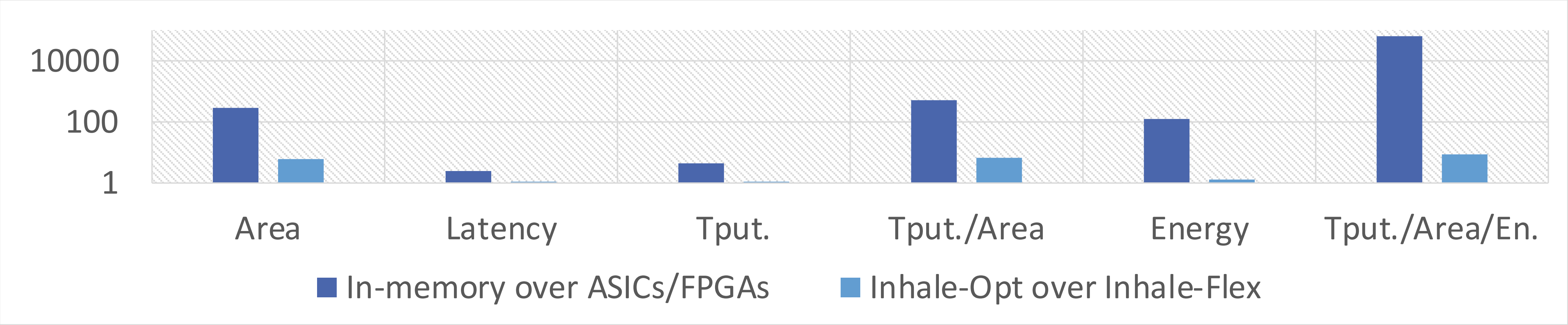


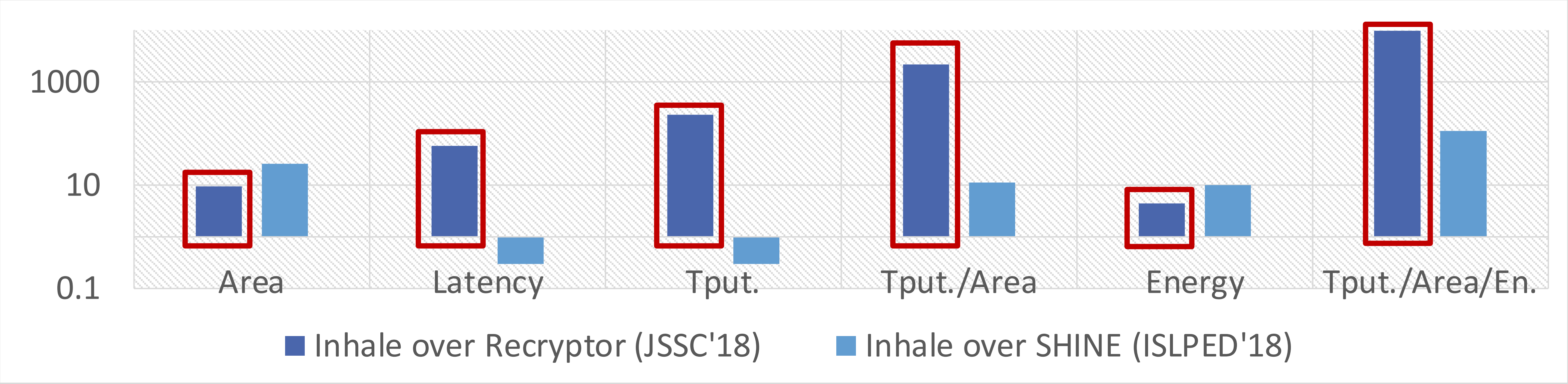
Comparison of different designs



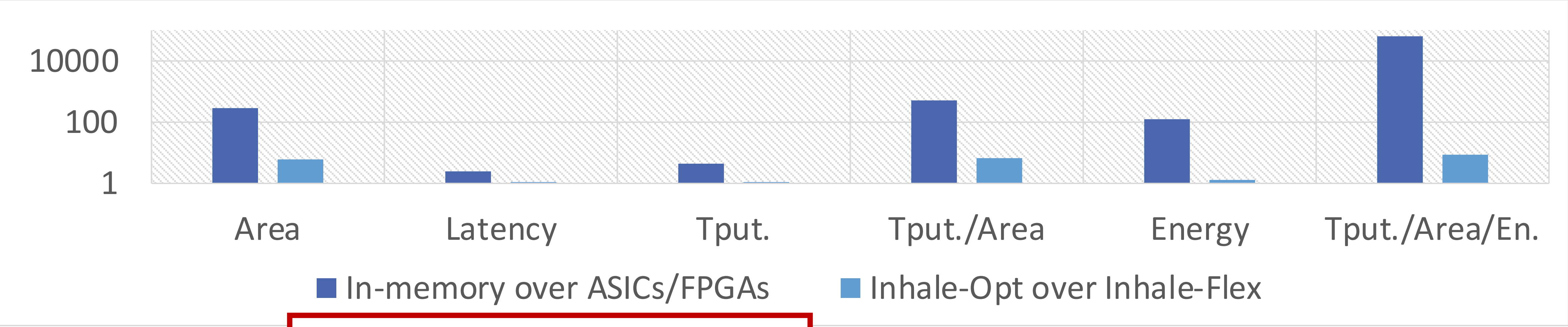


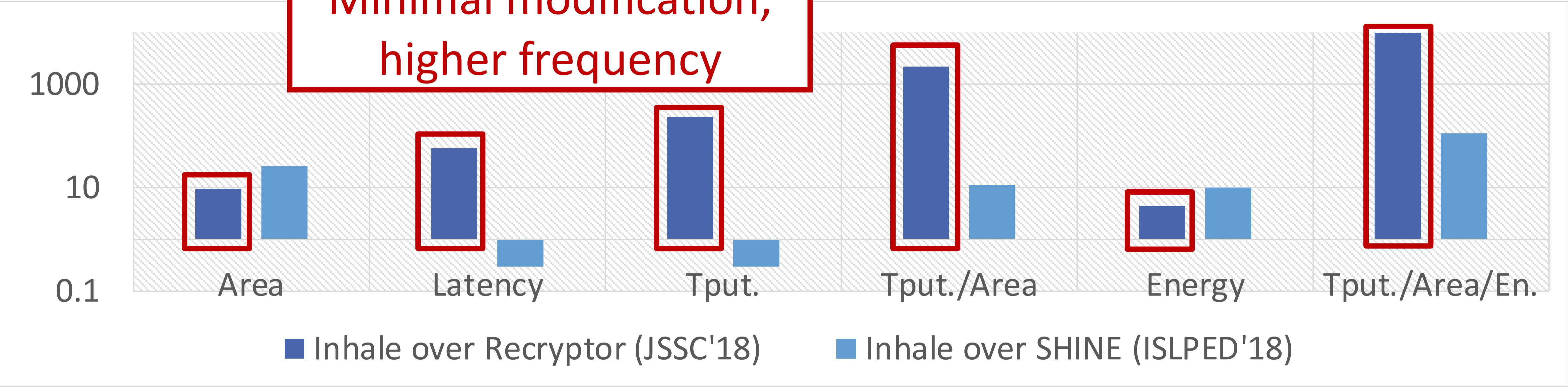
Comparison of different designs





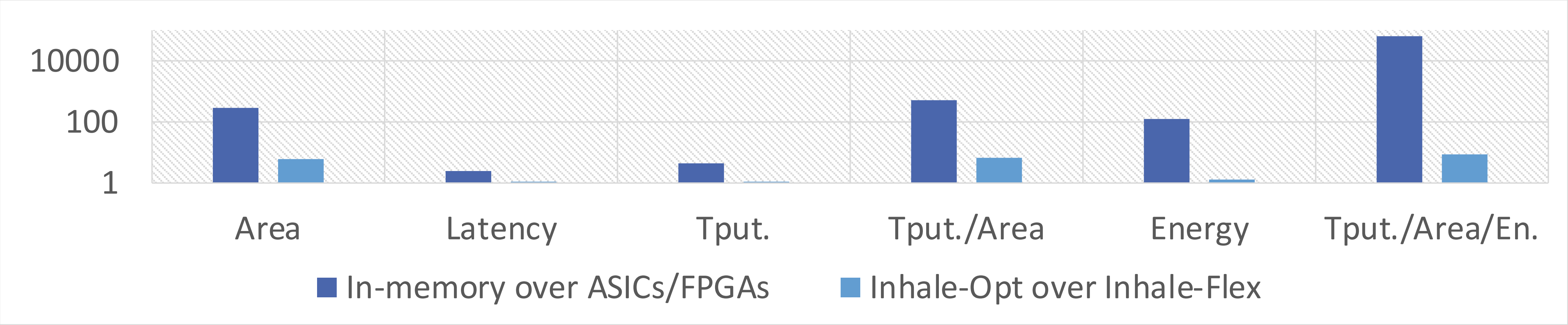
Comparison of different designs

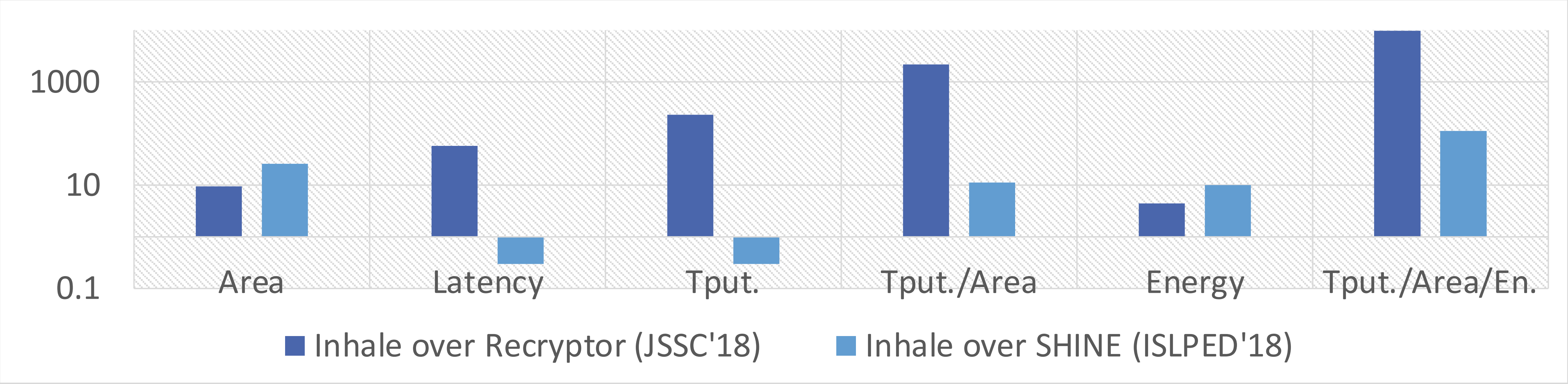




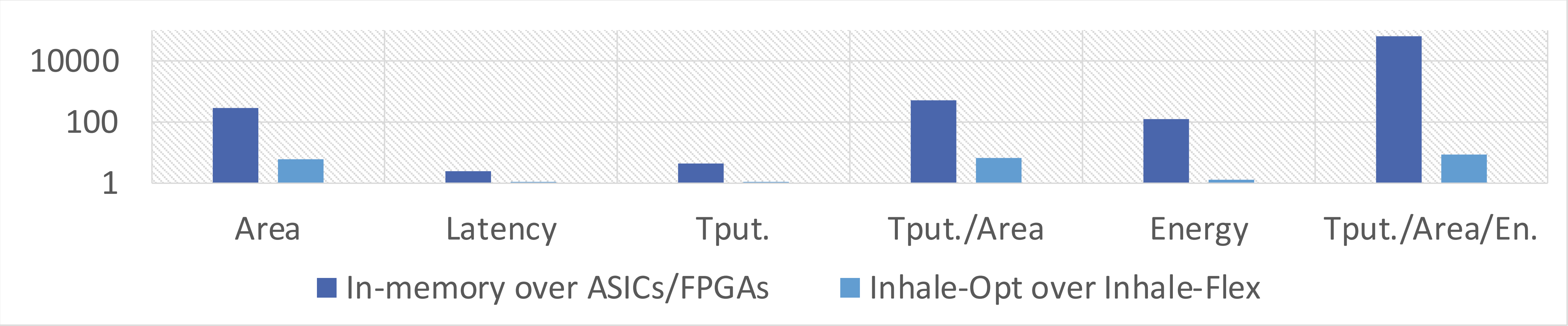
Minimal modification, higher frequency

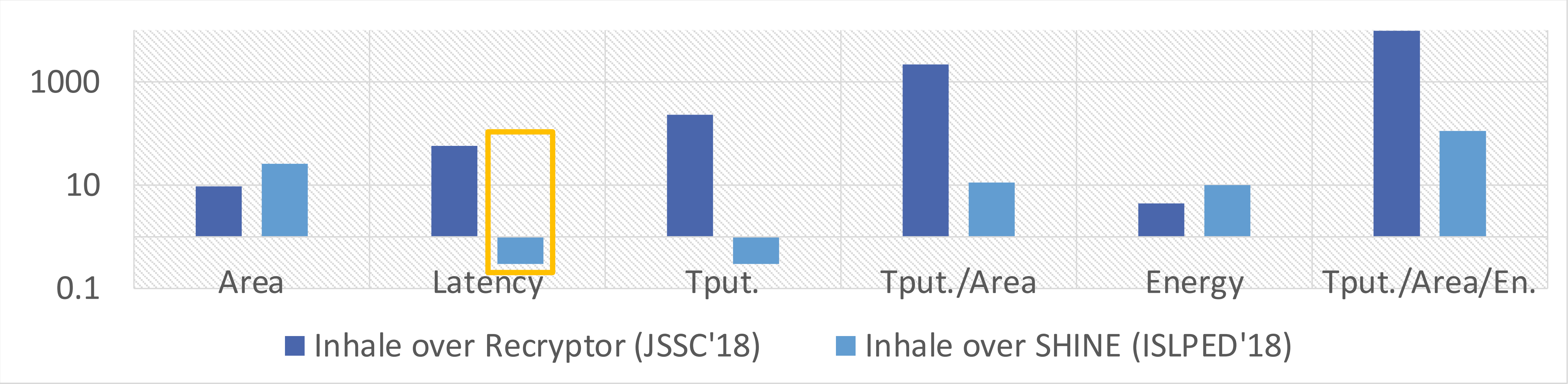
Comparison of different designs



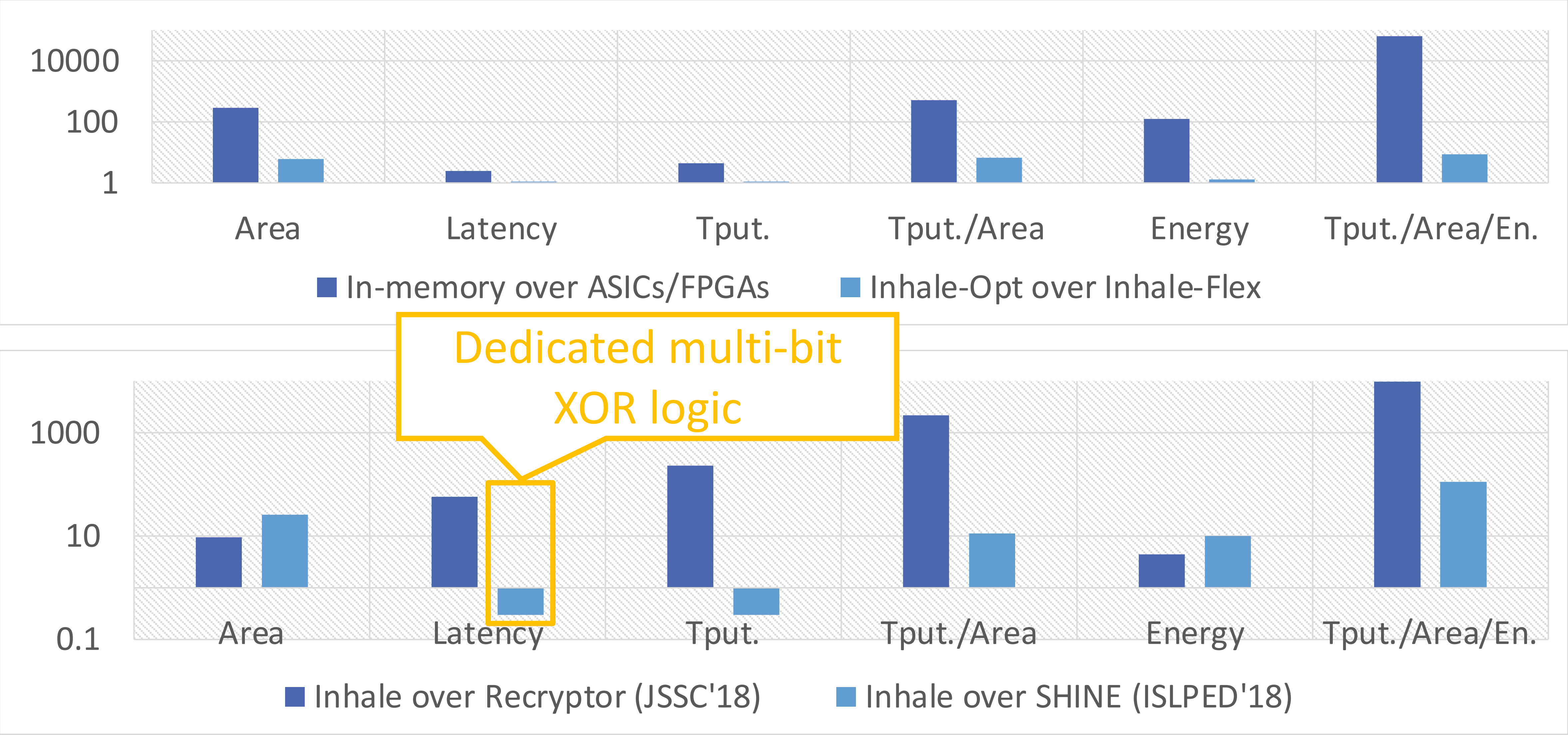


Comparison of different designs



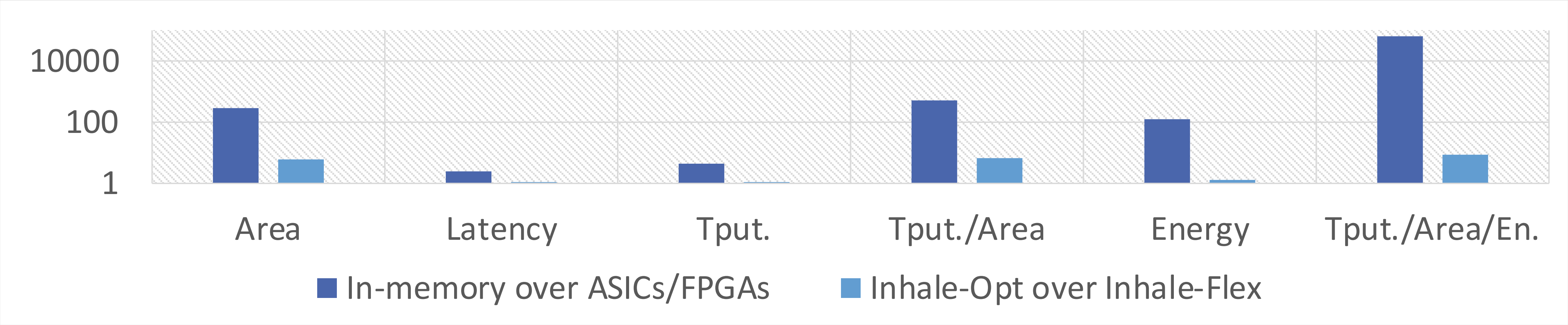


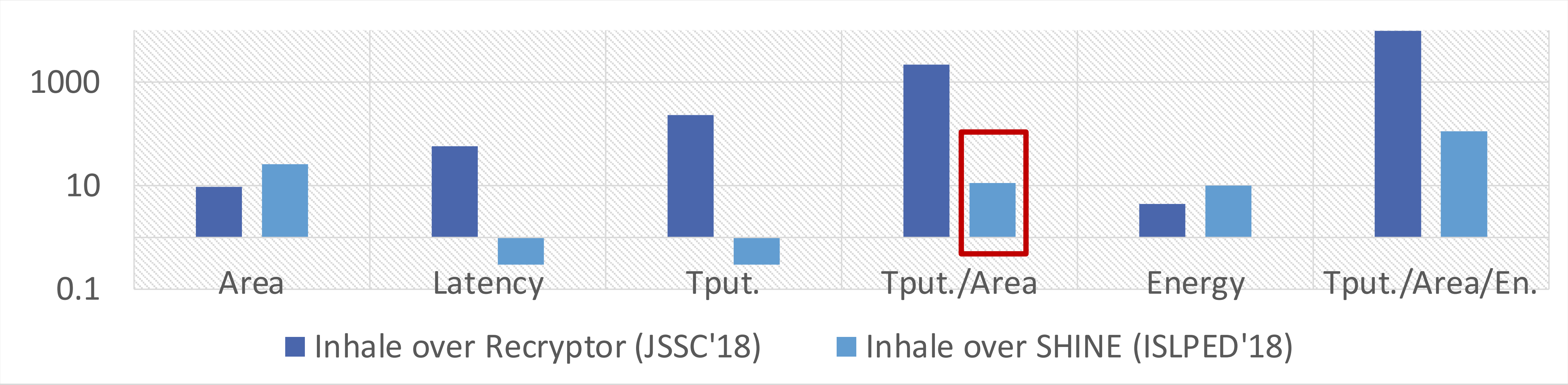
Comparison of different designs



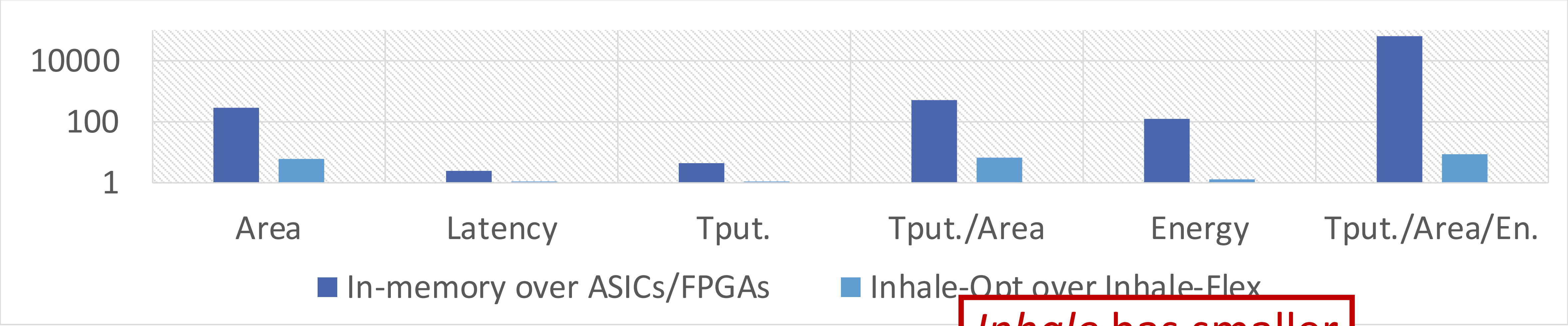
Dedicated multi-bit XOR logic

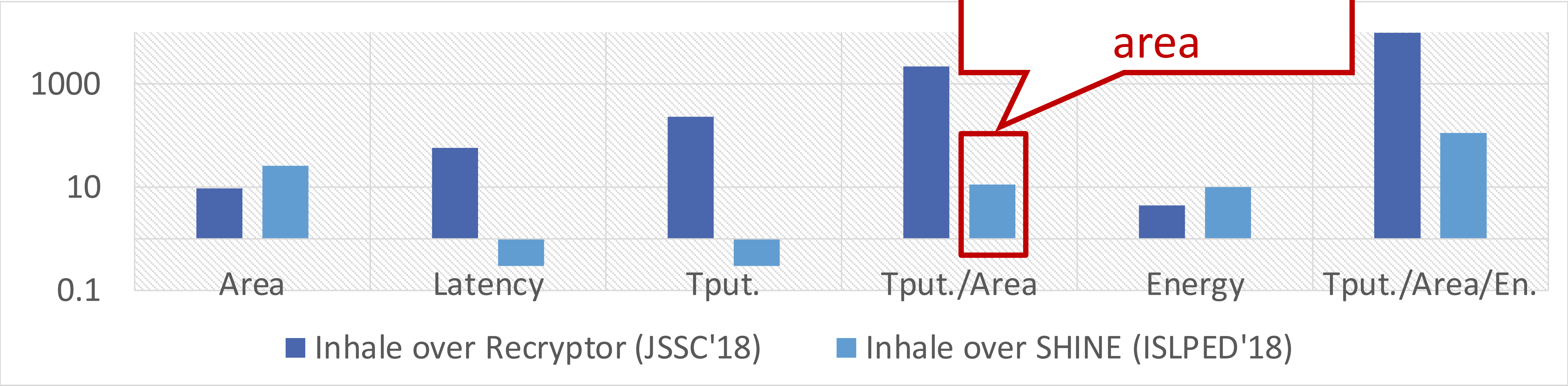
Comparison of different designs





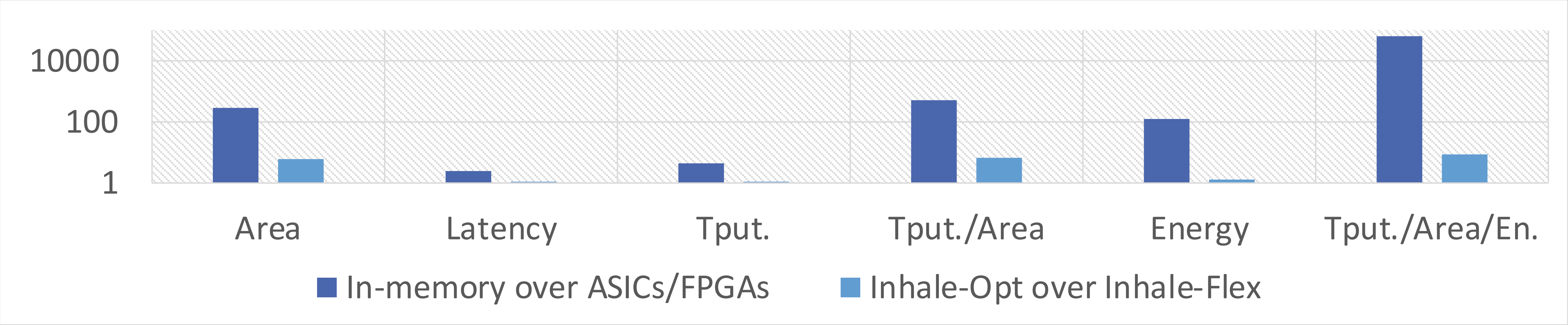
Comparison of different designs

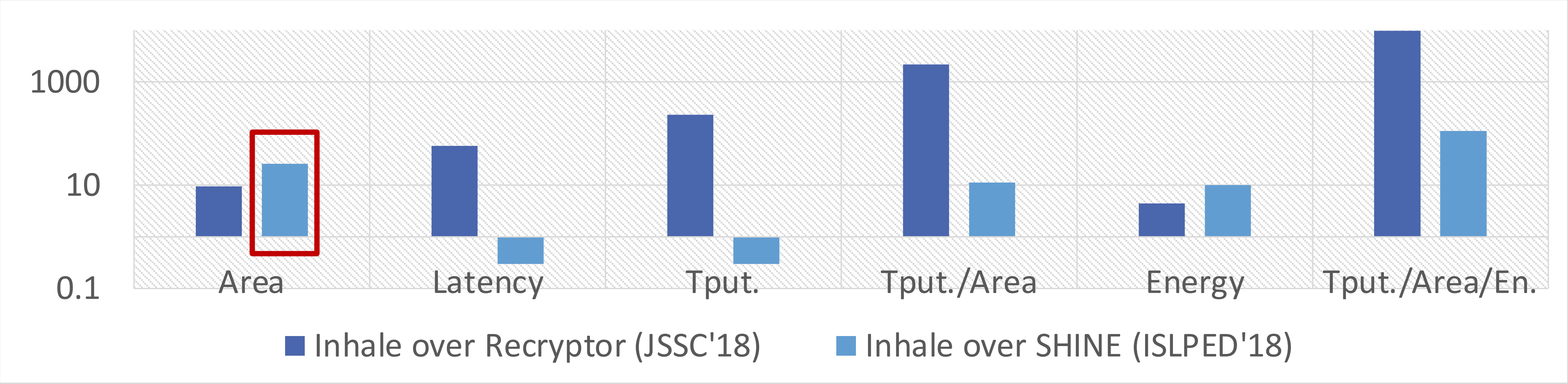




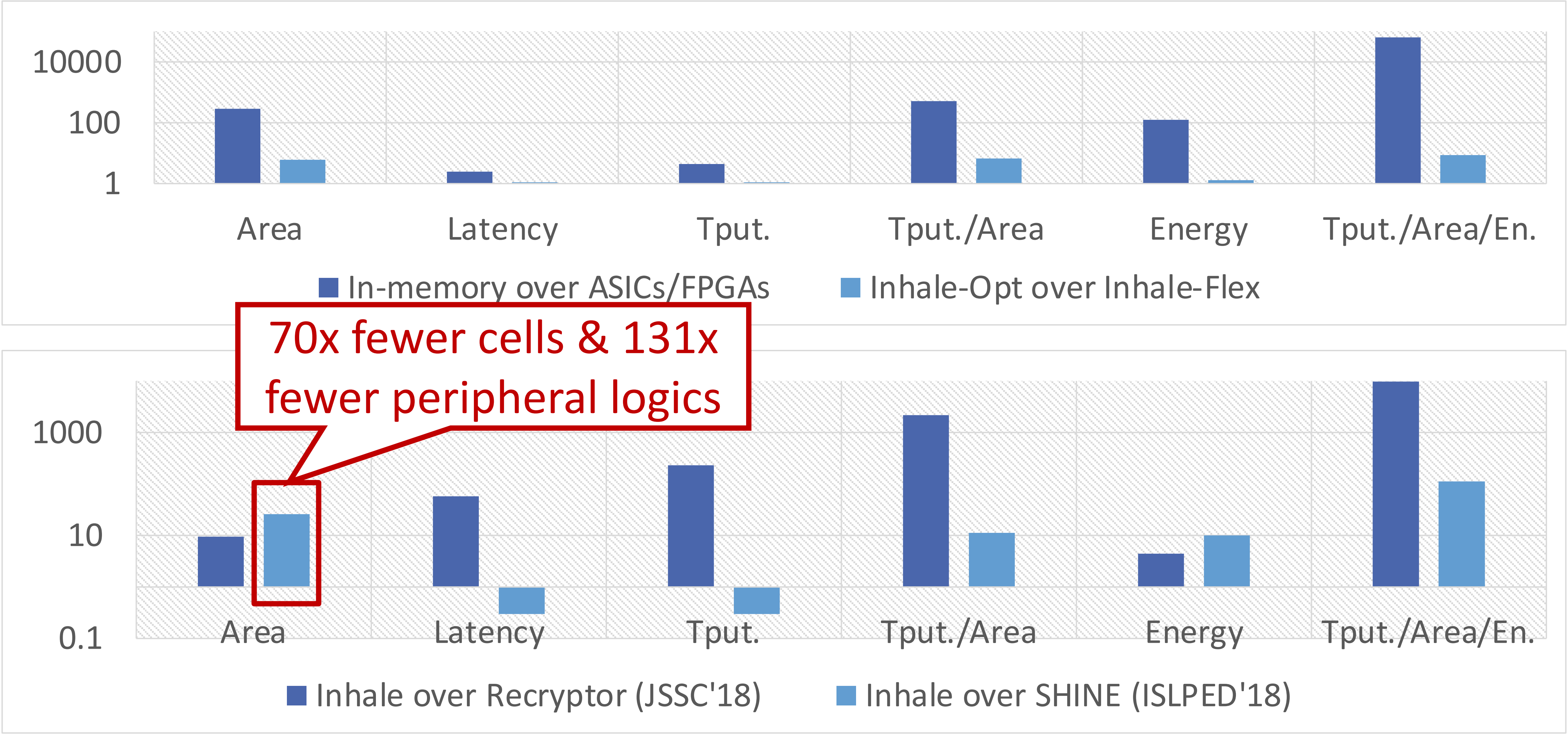
Inhale has smaller area

Comparison of different designs



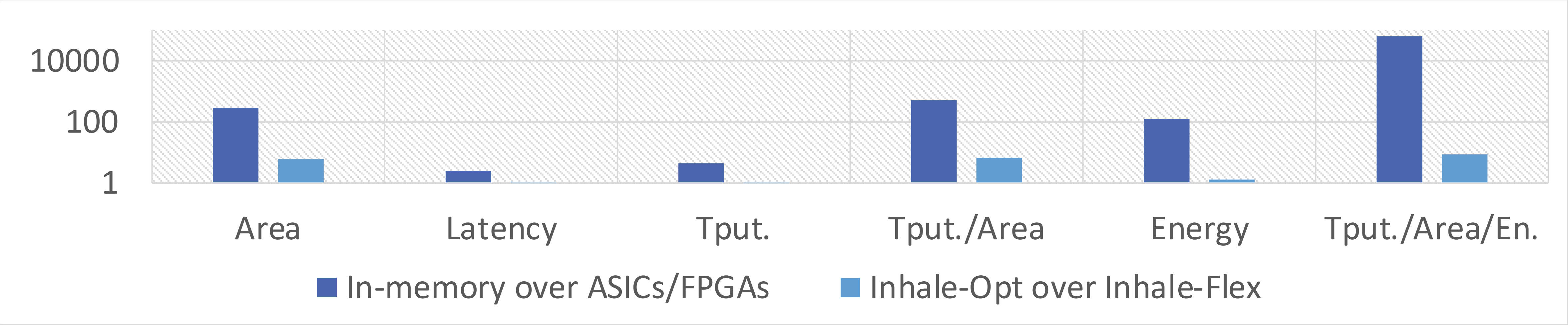


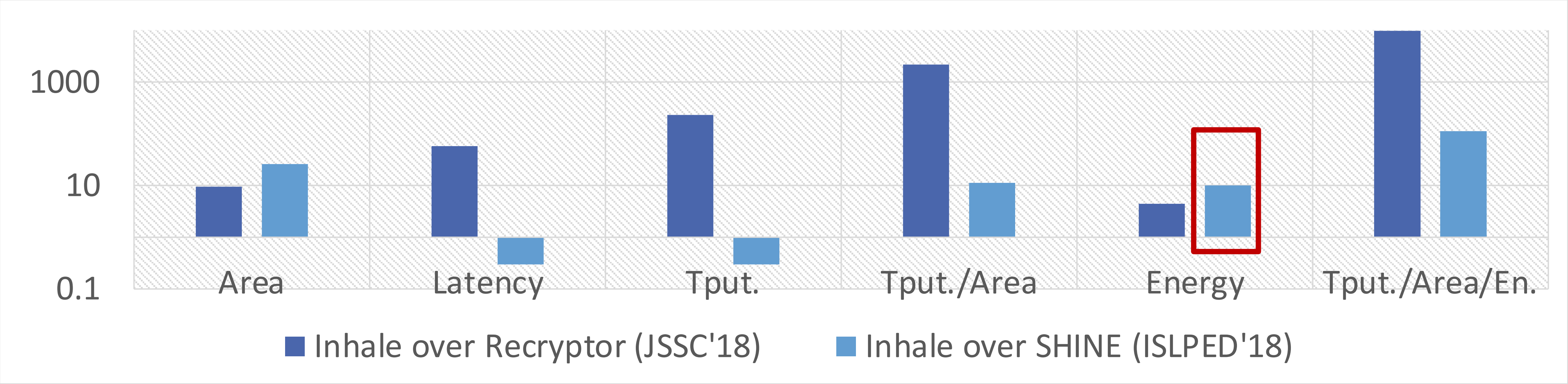
Comparison of different designs



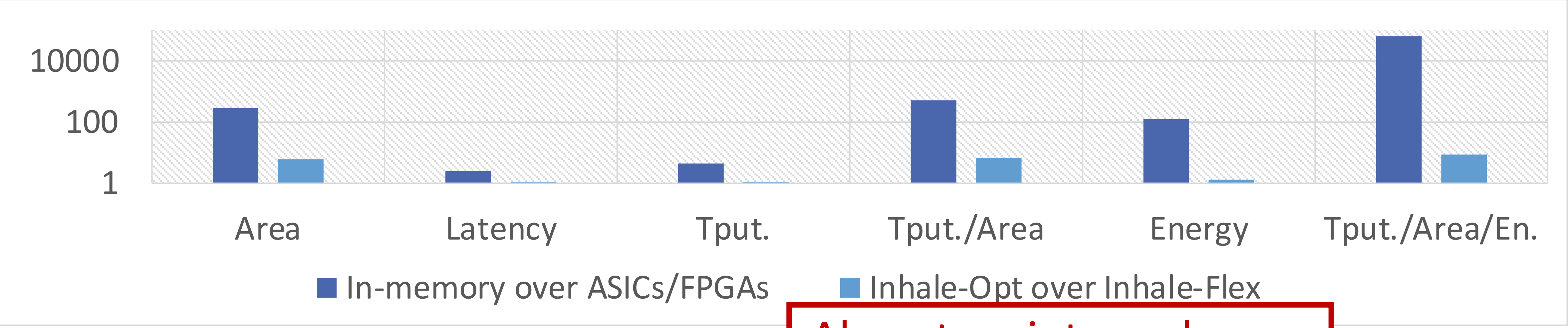
70x fewer cells & 131x fewer peripheral logics

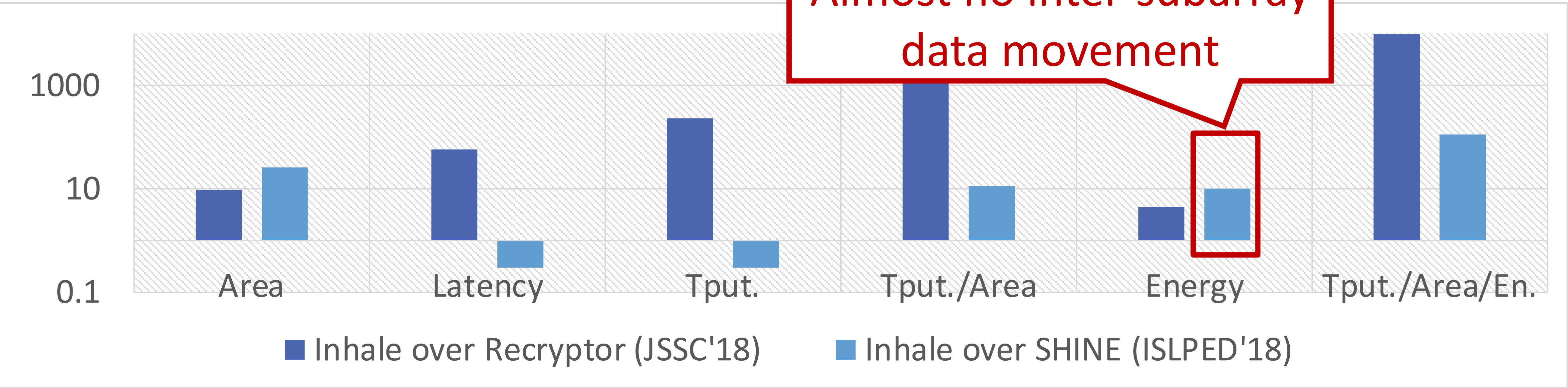
Comparison of different designs





Comparison of different designs





Almost no inter-subarray data movement

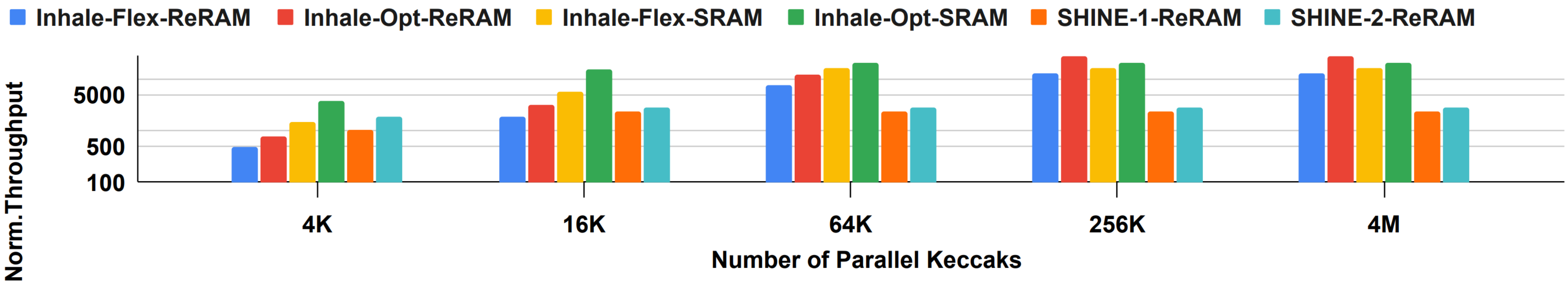
Performance Scaling

Performance Scaling

- With power constraint

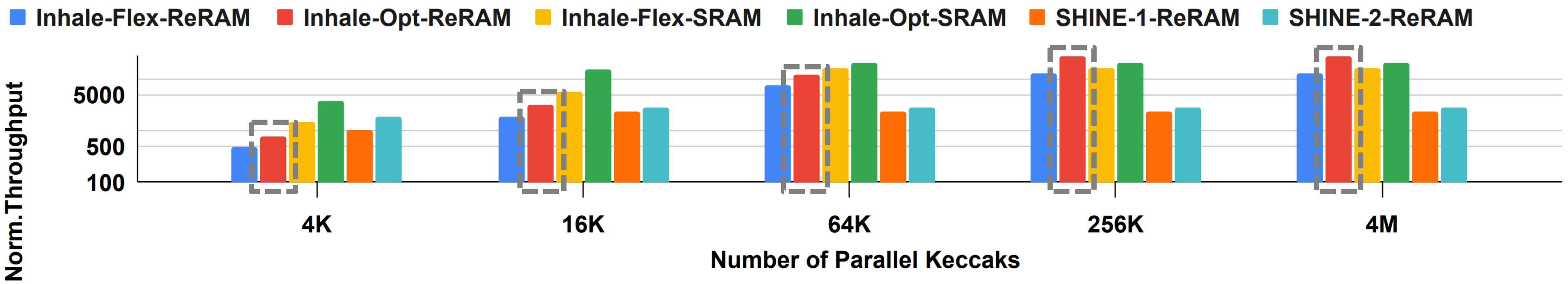
Performance Scaling

- With power constraint



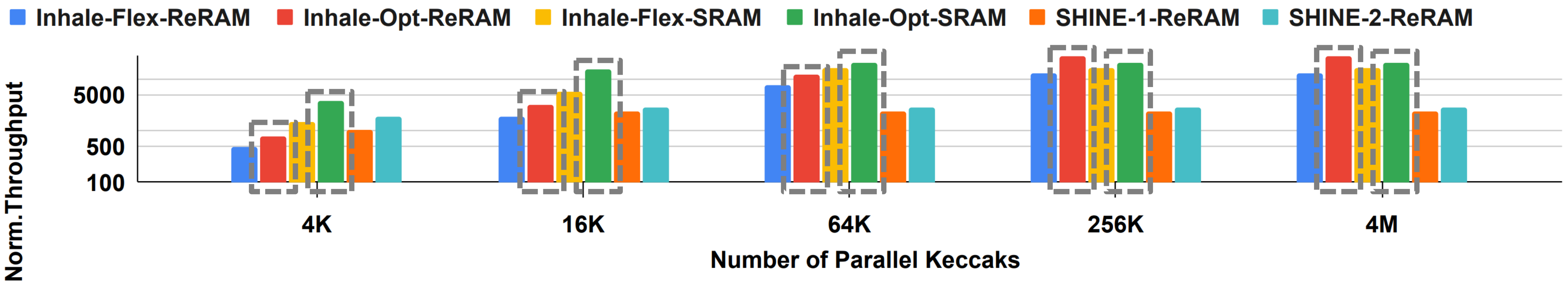
Performance Scaling

- With power constraint



Performance Scaling

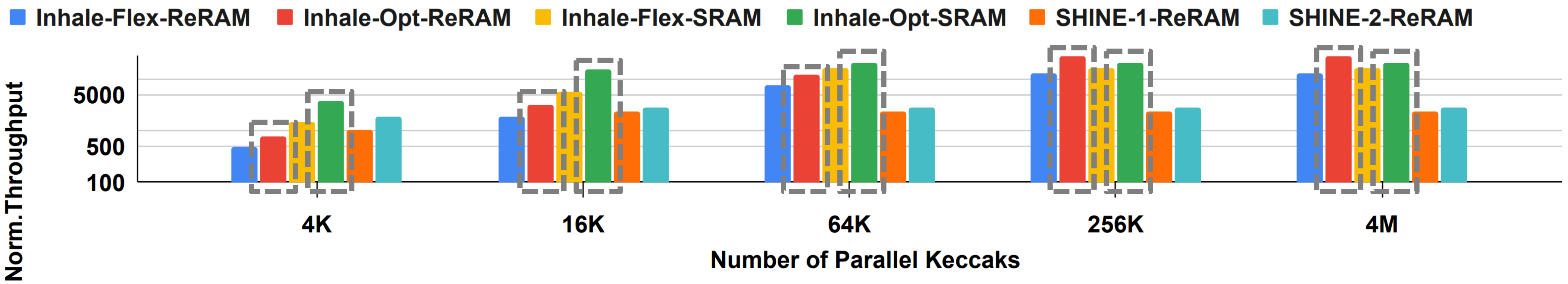
- ❑ With power constraint



Performance Scaling

- With power constraint

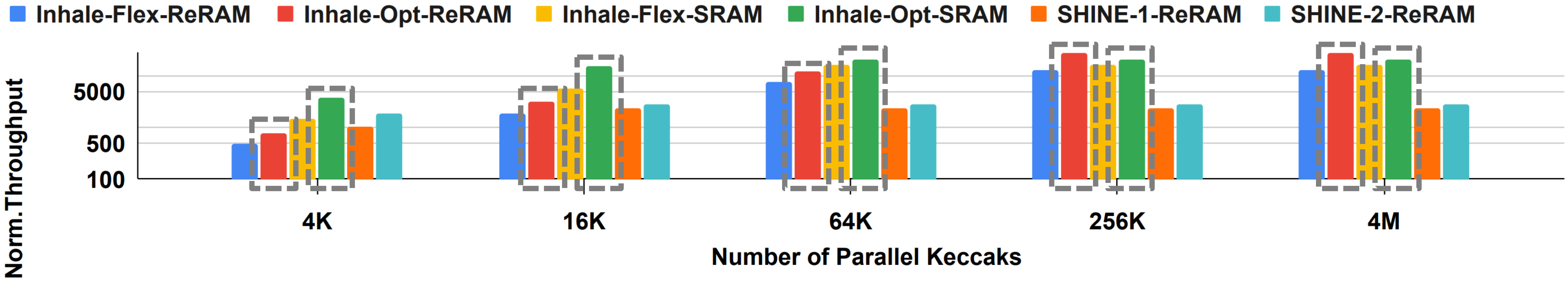
SHINE hits power earlier than *Inhale*



Performance Scaling

- ❑ With power constraint

SHINE hits power earlier than *Inhale*

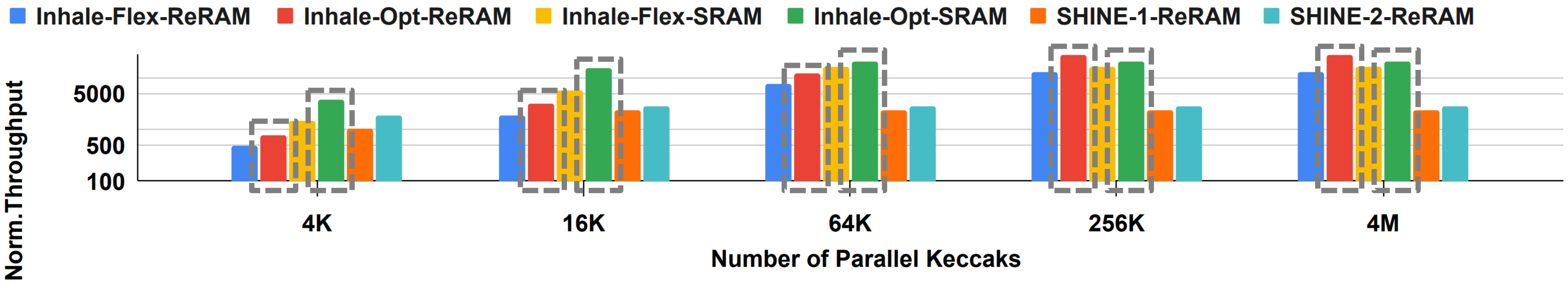


- ❑ Without power constraint

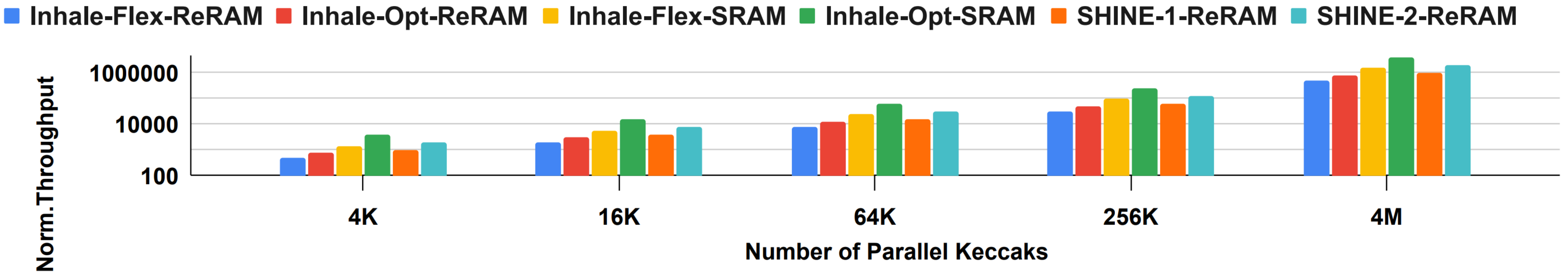
Performance Scaling

SHINE hits power earlier than *Inhale*

- ❑ With power constraint



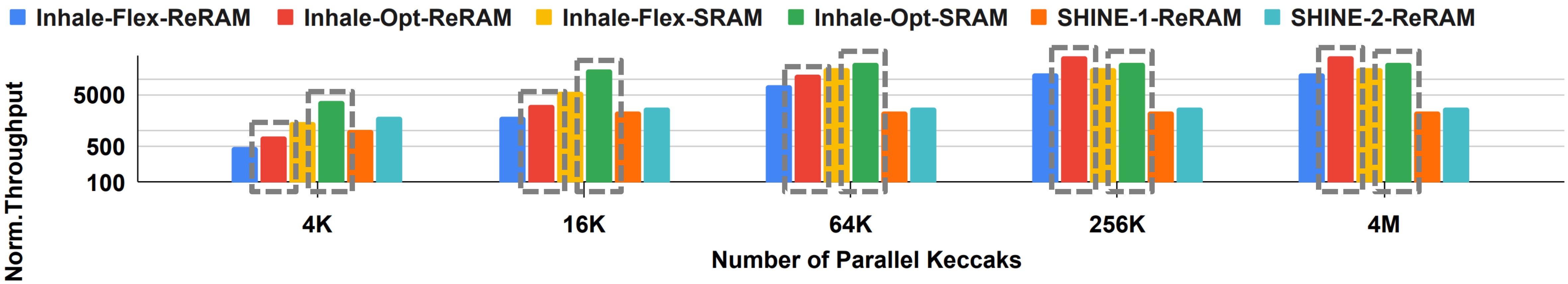
- ❑ Without power constraint



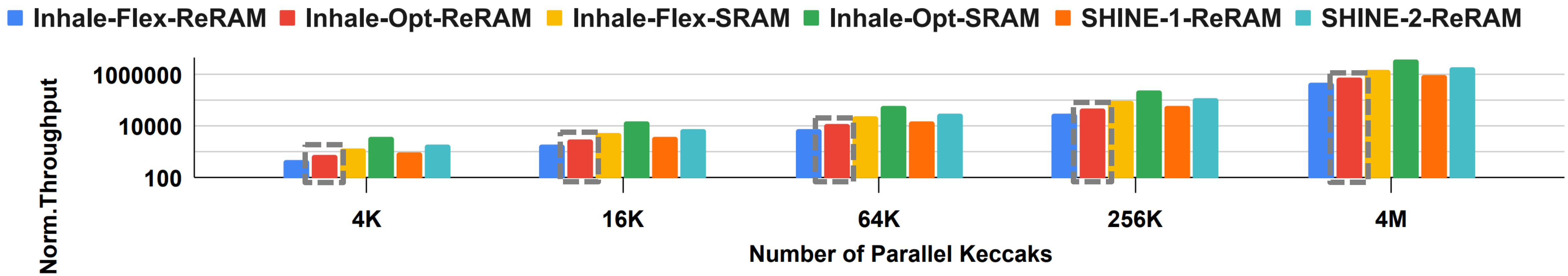
Performance Scaling

SHINE hits power earlier than *Inhale*

- ❑ With power constraint



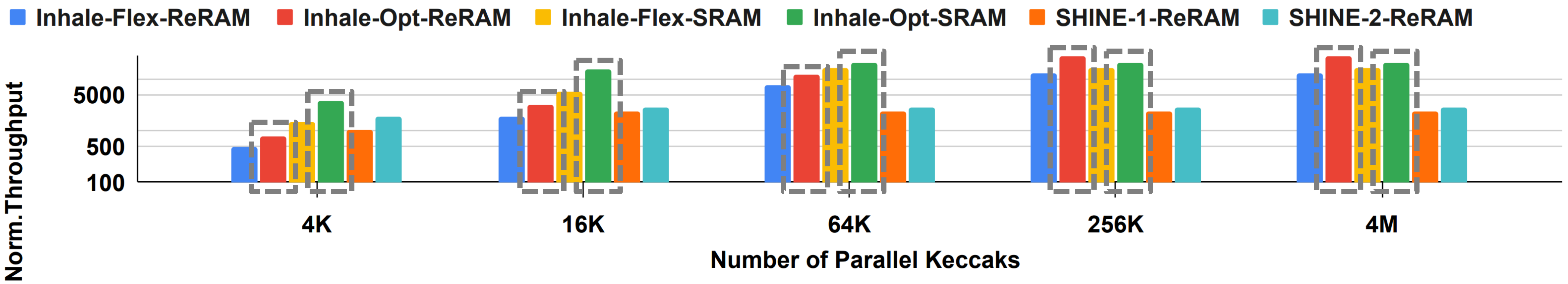
- ❑ Without power constraint



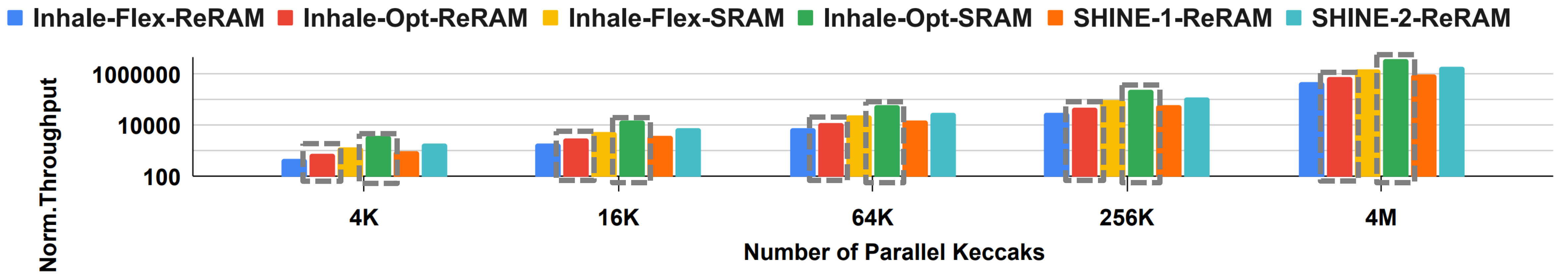
Performance Scaling

SHINE hits power earlier than *Inhale*

- With power constraint



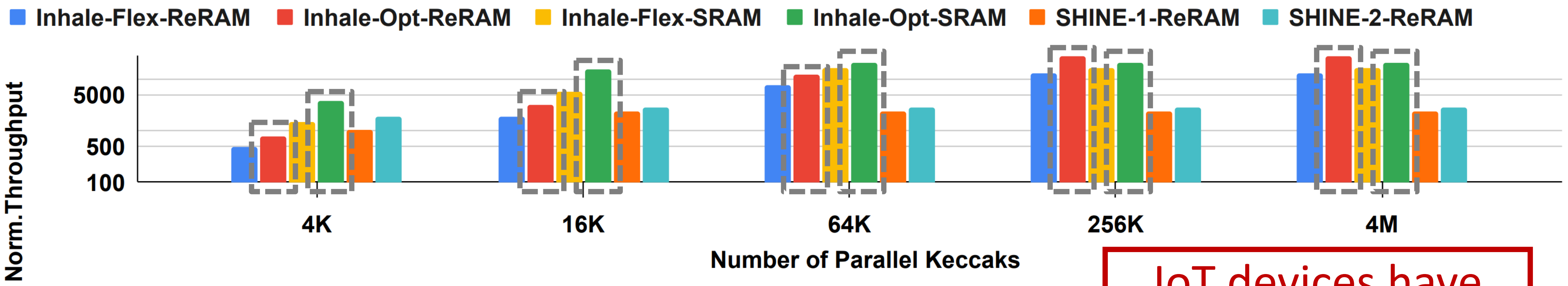
- Without power constraint



Performance Scaling

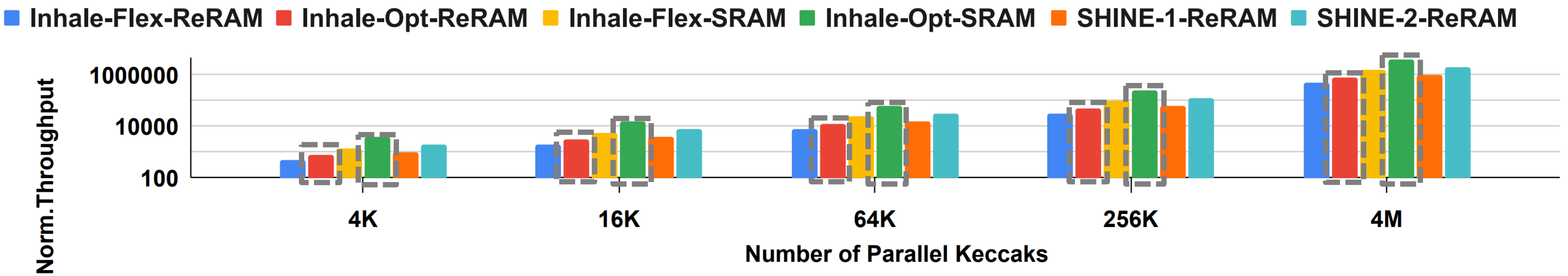
SHINE hits power earlier than *Inhale*

- With power constraint



IoT devices have tight power budget

- Without power constraint



Conclusion

Conclusion

- ❑ ***Inhale*** provides **high performance**, **energy efficiency**, **low overhead** all by proposing an in-SRAM **hashing** engine

Conclusion

- ❑ ***Inhale*** provides **high performance**, **energy efficiency**, **low overhead** all by proposing an in-SRAM **hashing** engine

- ❑ **Shift-optimized data alignment** and **in-place read/write strategy** are proposed to efficiently map the algorithm to the ***Inhale*** architecture

Conclusion

- ❑ ***Inhale*** provides **high performance**, **energy efficiency**, **low overhead** all by proposing an in-SRAM **hashing** engine

- ❑ **Shift-optimized data alignment** and **in-place read/write strategy** are proposed to efficiently map the algorithm to the ***Inhale*** architecture

- ❑ ***Inhale*** can achieve **up to 14x** throughput-per-area, **172x** throughput-per-area-per-energy than state-of-the-art

Conclusion

- ❑ ***Inhale*** provides **high performance**, **energy efficiency**, **low overhead** all by proposing an in-SRAM **hashing** engine

- ❑ **Shift-optimized data alignment** and **in-place read/write strategy** are proposed to efficiently map the algorithm to the ***Inhale*** architecture

- ❑ ***Inhale*** can achieve **up to 14x** throughput-per-area, **172x** throughput-per-area-per-energy than state-of-the-art

- ❑ Future work is providing an end-to-end solution for IoT security, and supporting other cryptographic operations

Q&A