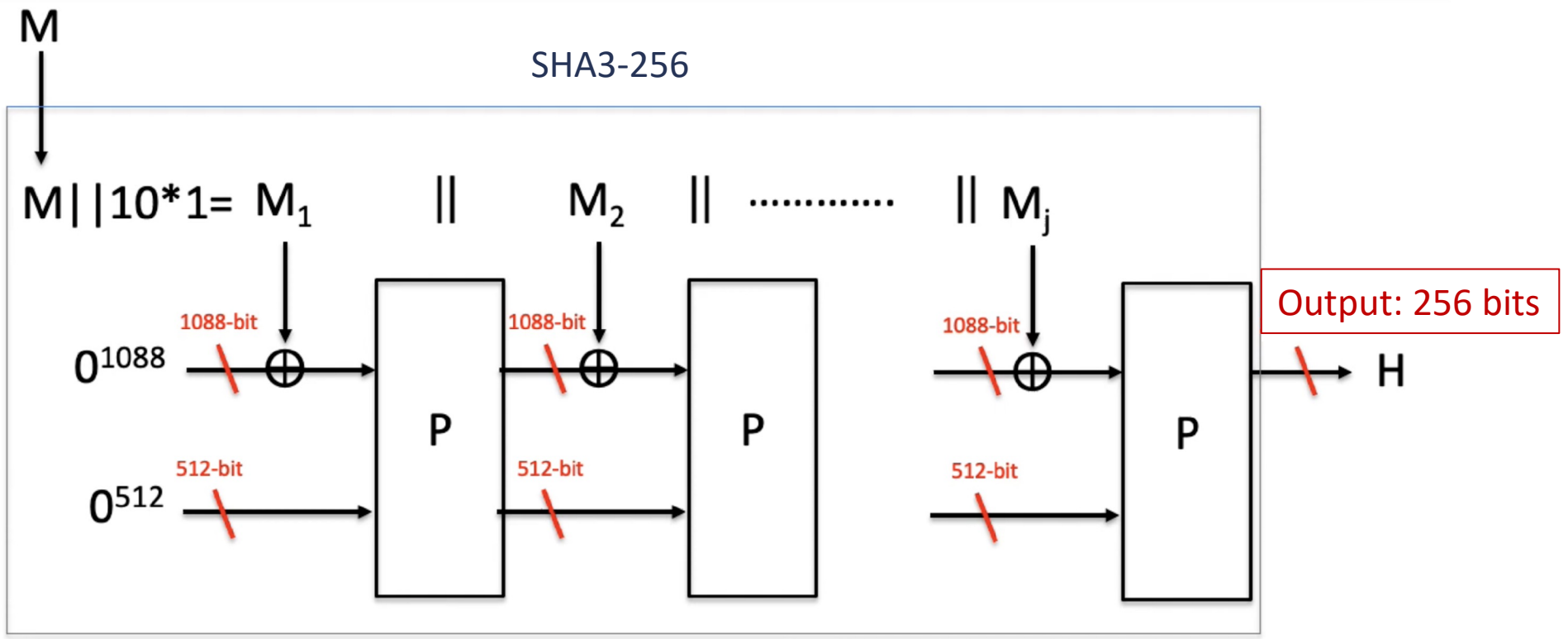
Inhale: Enabling High-Performance and Energy-Efficient In-SRAM Cryptographic Hash for IoT

Jingyao Zhang, Elaheh Sadredini

2022 IEEE/ACM International Conference on Computer-Aided Design

What is Cryptographic Hash

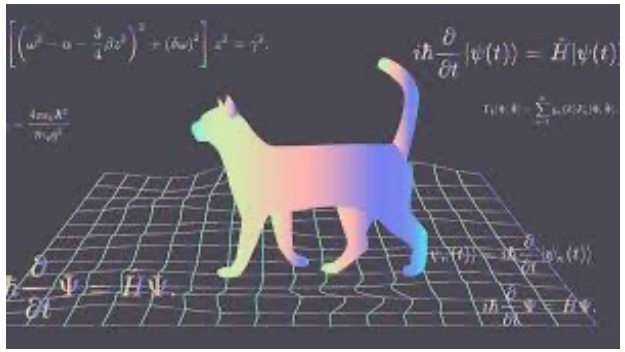
- Input M: a binary string of any bit-length
- Output H: a multi-bit string
- **Practically infeasible to invert or reverse the computation**



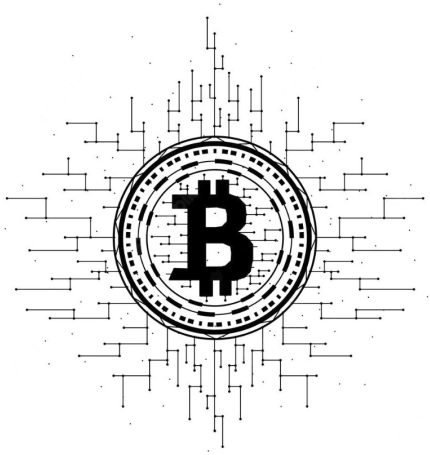
What is Cryptographic Hash

- Input M: a binary string of any bit-length
- Output H: a multi-bit string
- **Practically infeasible to invert or reverse the computation**

Transport Layer Security



Post-quantum Cryptography



Cryptocurrency

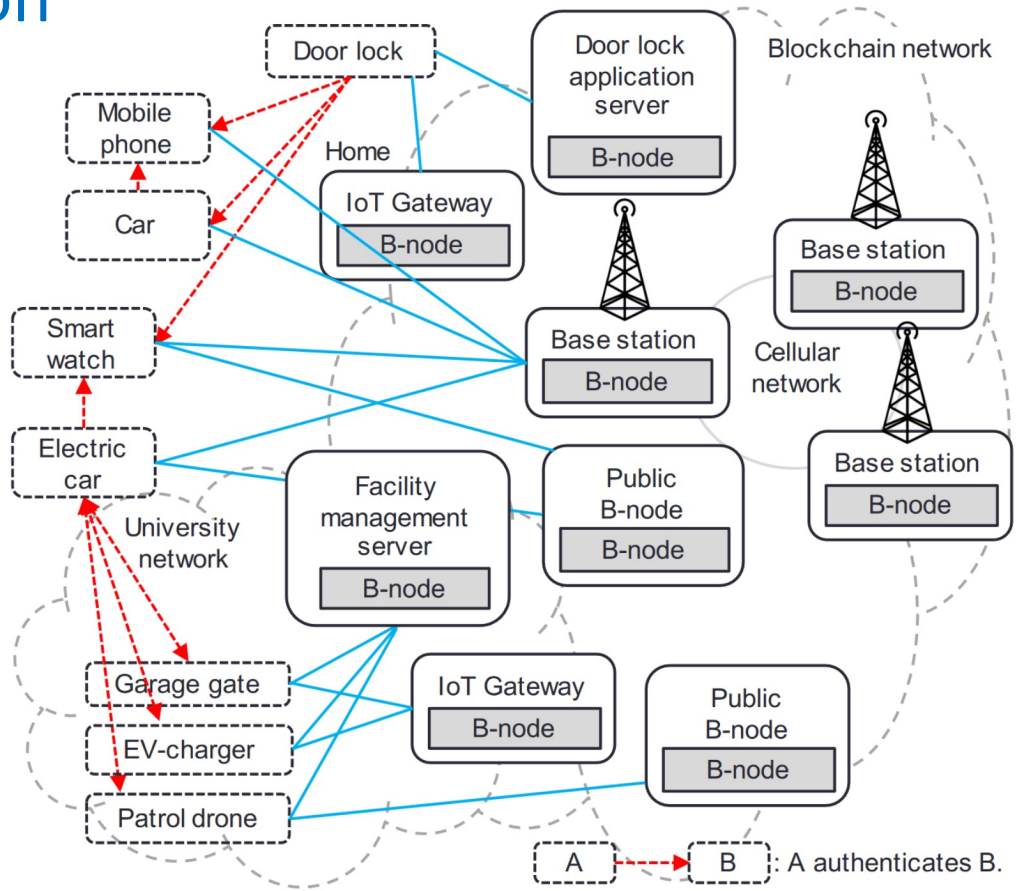
Motivating Example: Secure Communication

- Moreover, hashing can provide 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

More Vulnerability in IoT Era

- Attackers can **effortlessly** obtain physical access to edge devices
- Though **hash-based public key infrastructure** can be used for Data Integrity and Identity Authentication

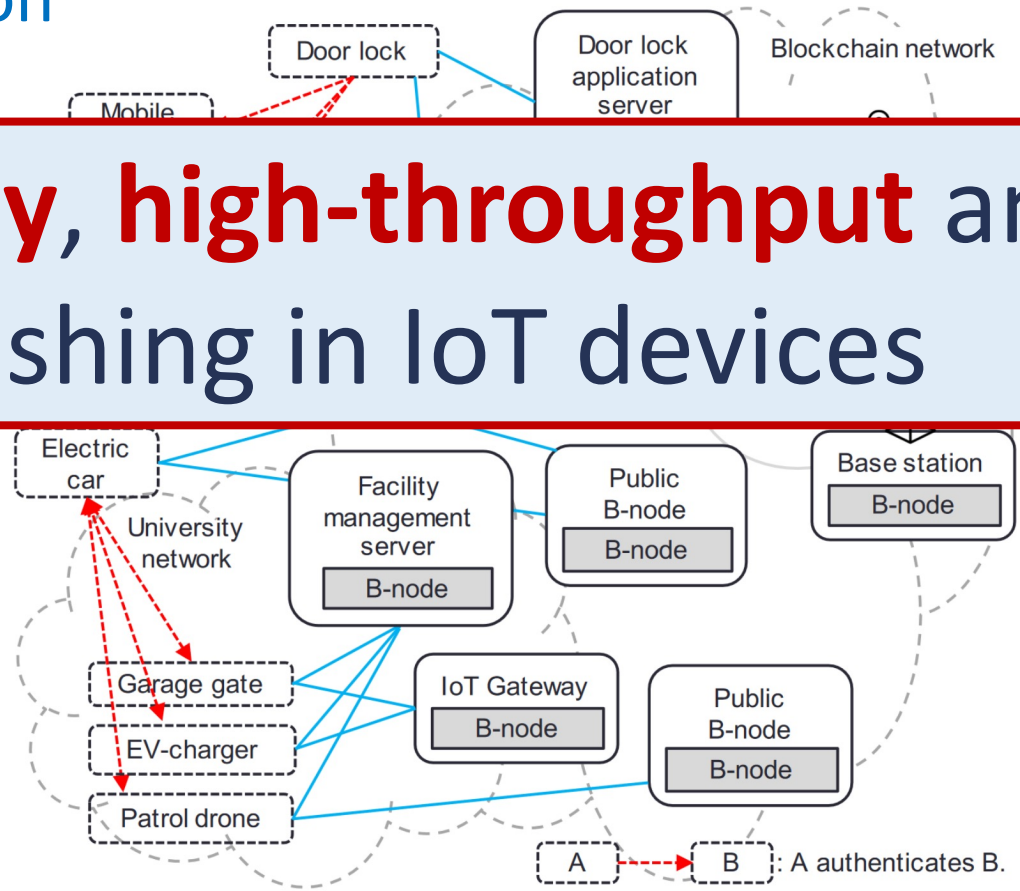


J. Won, et al., "Decentralized Public Key Infrastructure for Internet-of-Things," in MILCOM 2018

More Vulnerability in IoT Era

- Attackers can **effortlessly** obtain physical access to edge devices
- Though **hash-based public key infrastructure** can be used for Data Integrity and Identity Authentication

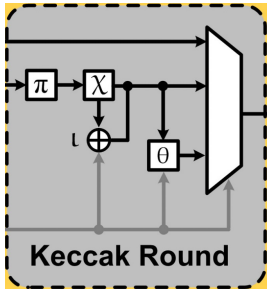
Demand for low-latency, high-throughput and energy-efficient hashing in IoT devices



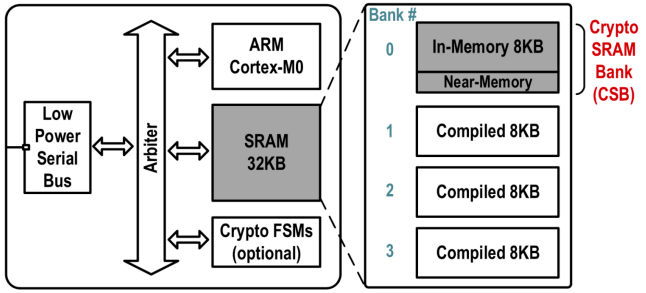
J. Won, et al., "Decentralized Public Key Infrastructure for Internet-of-Things," in MILCOM 2018

△ Challenges: Performance, Energy, Area

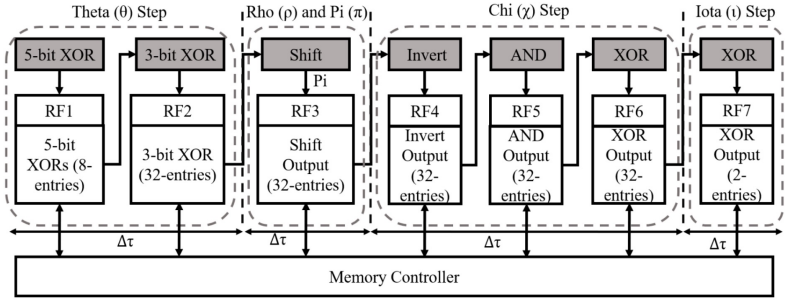
- **Dedicated hardware engine on chip (ISSCC'16)**
  - Low throughput
  - High area overhead on chip



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

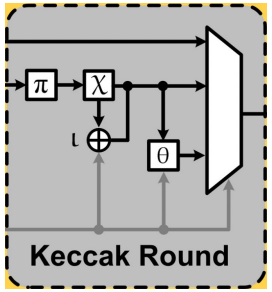


- **Dedicated in-memory acceleration (ISLPED'19)**
  - High area overhead
  - Low flexibility



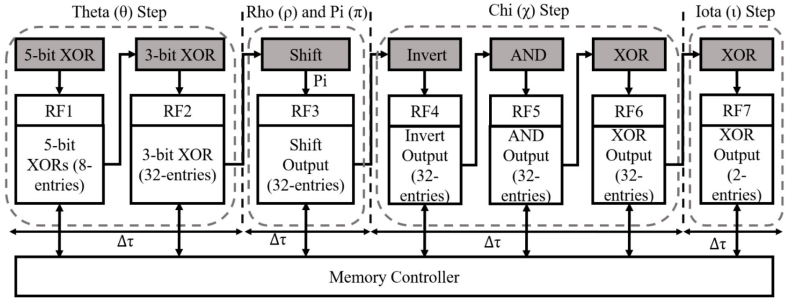
△ Challenges: Performance, Energy, Area

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



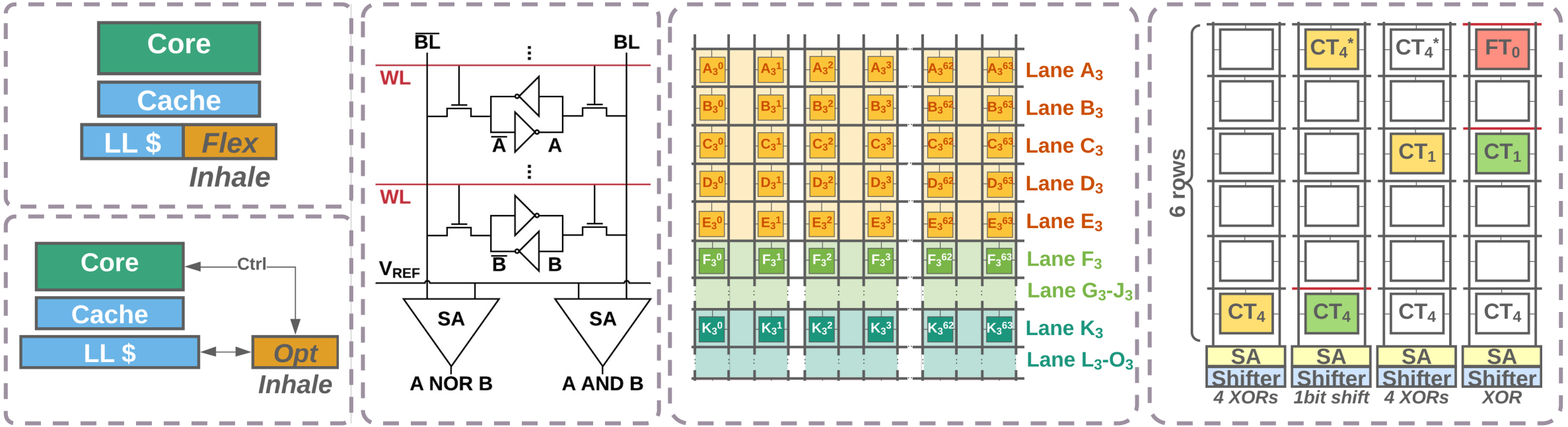
Demand for **low-latency**, **high-throughput**, **energy-efficient**, **low-overhead** hashing in IoT

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



Overview of Our Solution: *Inhale*

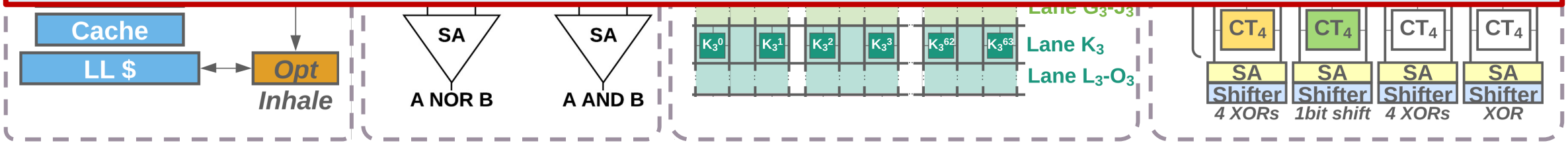
- On-chip Hashing -> high security level
- Bitline Computing -> high throughput
- Shift-optimized Data Alignment -> low latency, energy
- In-Place Read/Write Strategy -> low overhead



Overview of Our Solution: *Inhale*

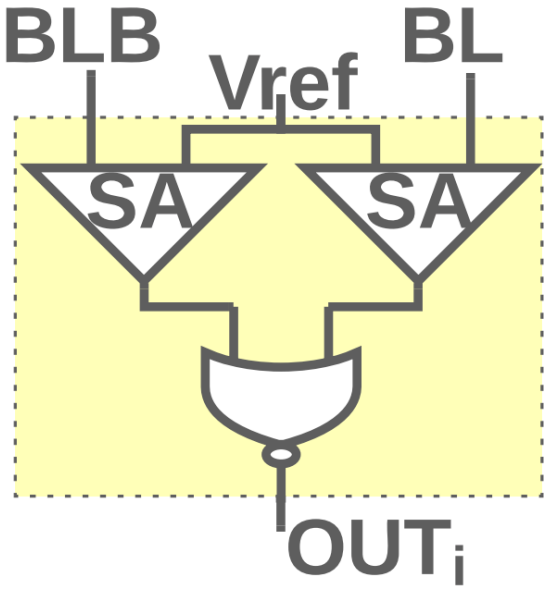
- On-chip Hashing -> high security level
- Bitline Computing -> high throughput
- Shift-optimized Data Alignment -> low latency, energy
- In-Place Read/Write Strategy -> low overhead

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



Inhale: Bitline Computing

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

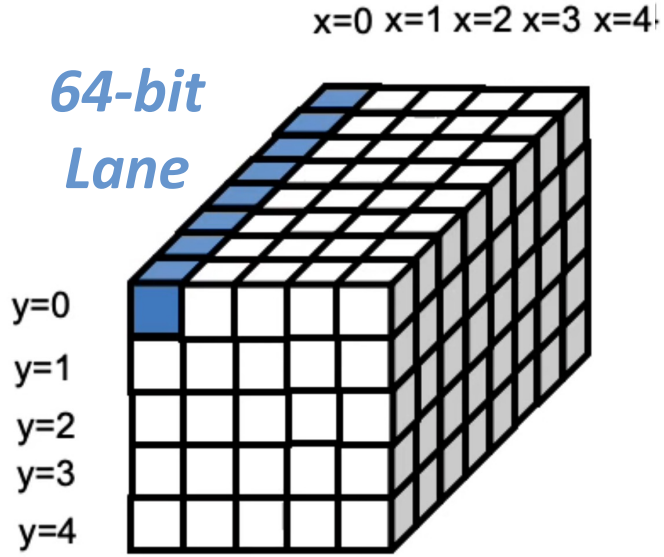


Sense Amplifier Design

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

Inhale: Shift-optimized Data Alignment

- **Existing Data Alignments**
  - JSSC'18:
    - hard for inter-lane and intra-lane shift
  - ISCA'18:
    - high latency, high control overhead (>10x JSSC'18)



1600-bit State

> 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				

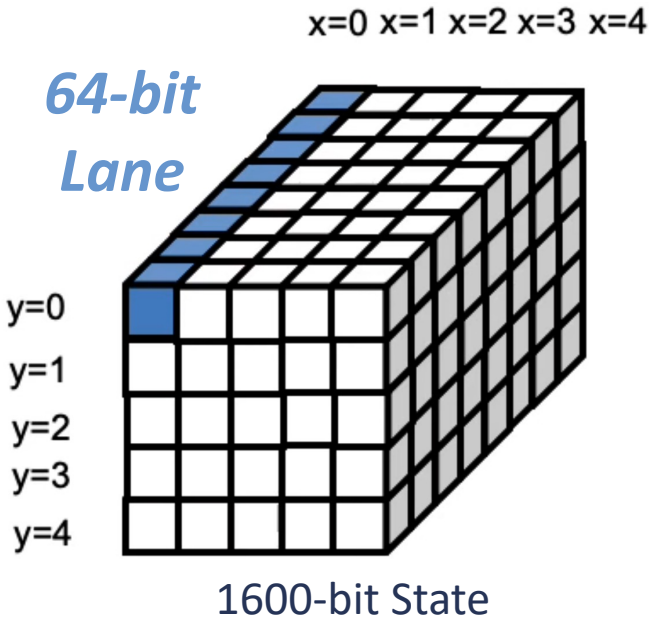
SRAM subarray (JSSC'18)

> 1600 bits

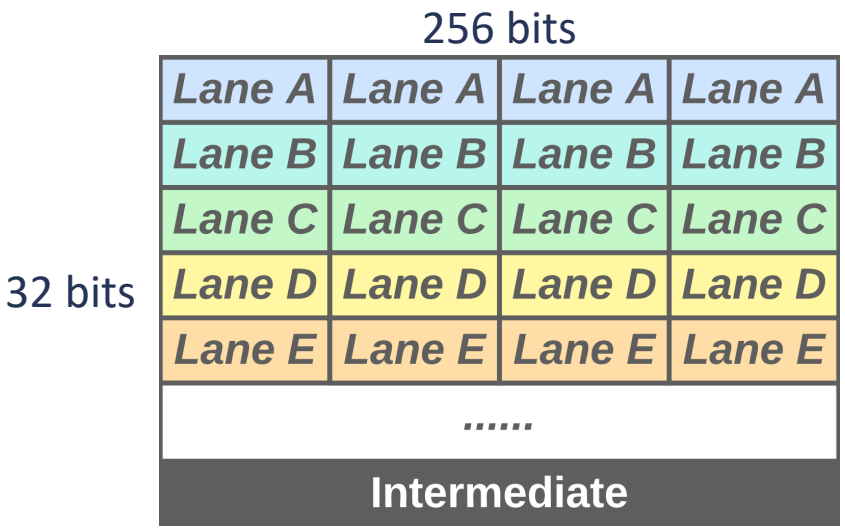
SRAM subarray (ISCA'18)

Inhale: Shift-optimized Data Alignment

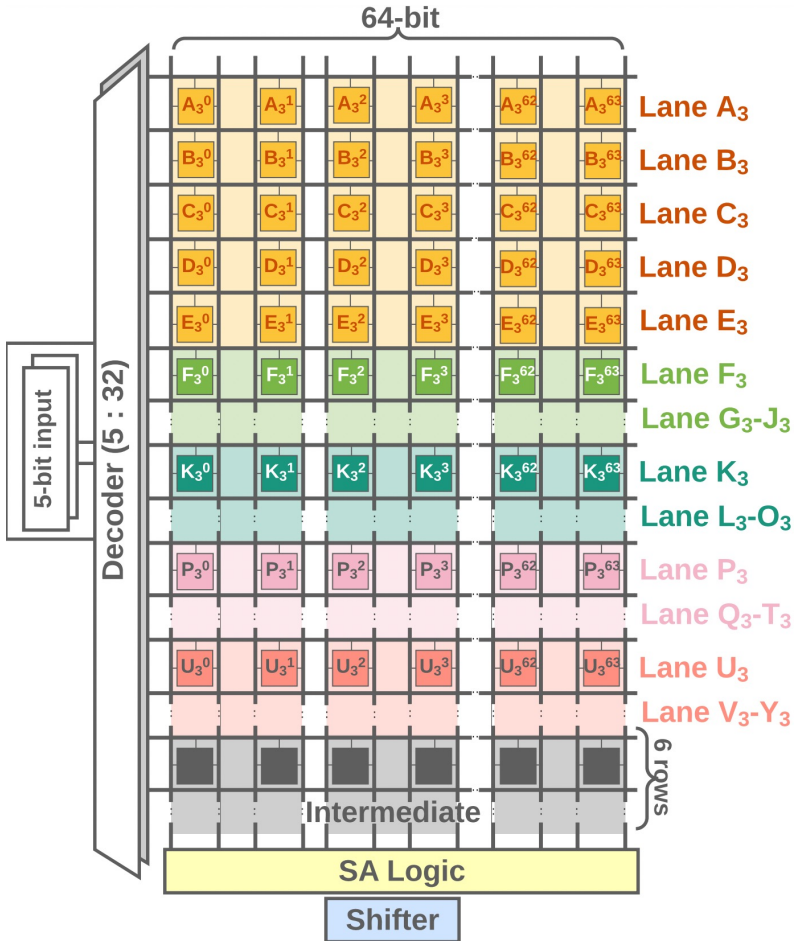
- **Shift-optimized Data Alignment**
  - Place lane per row
  - *Inter-lane* shifts are costless with the controller
  - *Intra-lane* shifts are performed with small shifters
  - Well balance the performance and overhead



1600-bit State



SRAM subarray (*Inhale*)

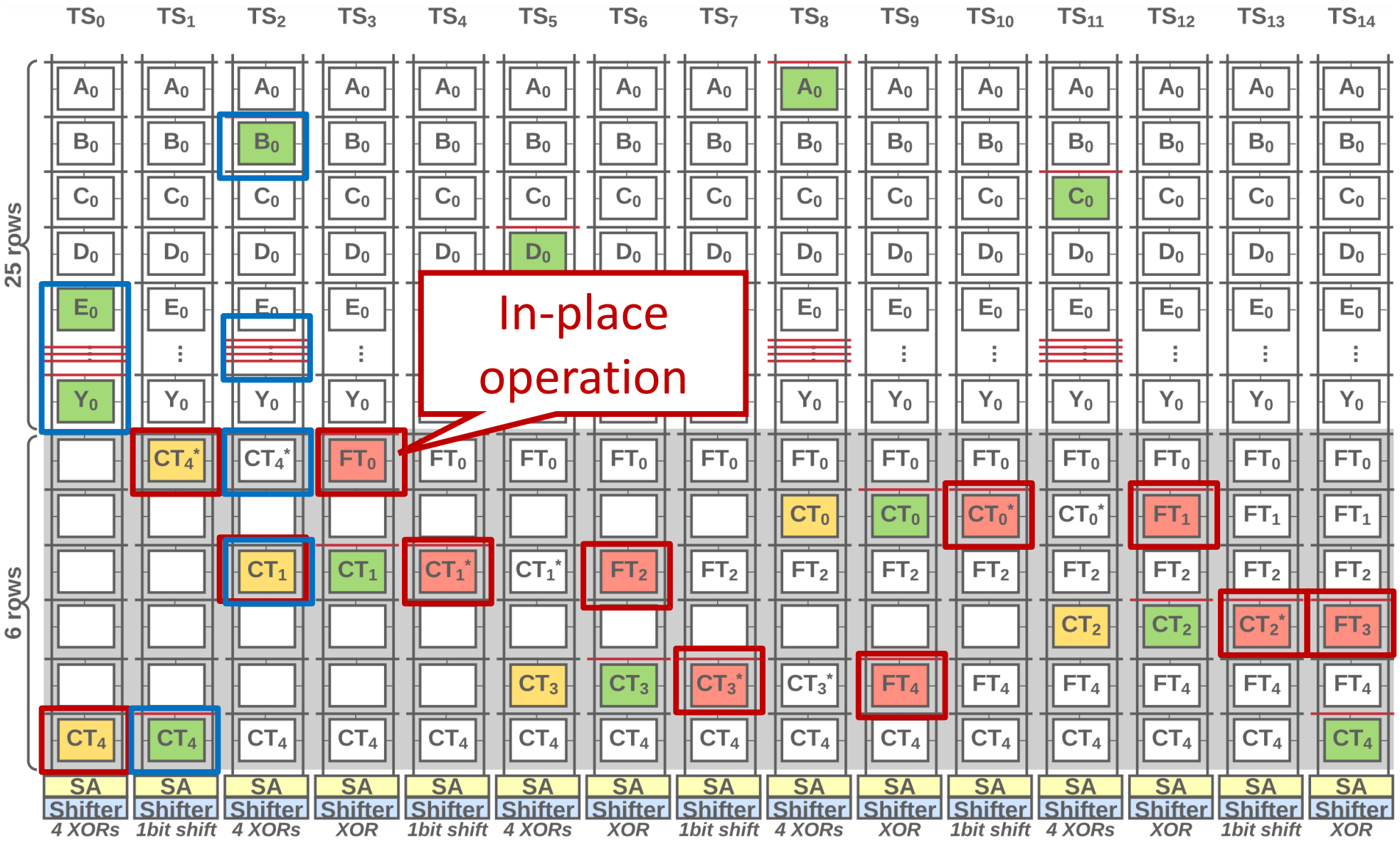


Detailed data alignment

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)
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^*)

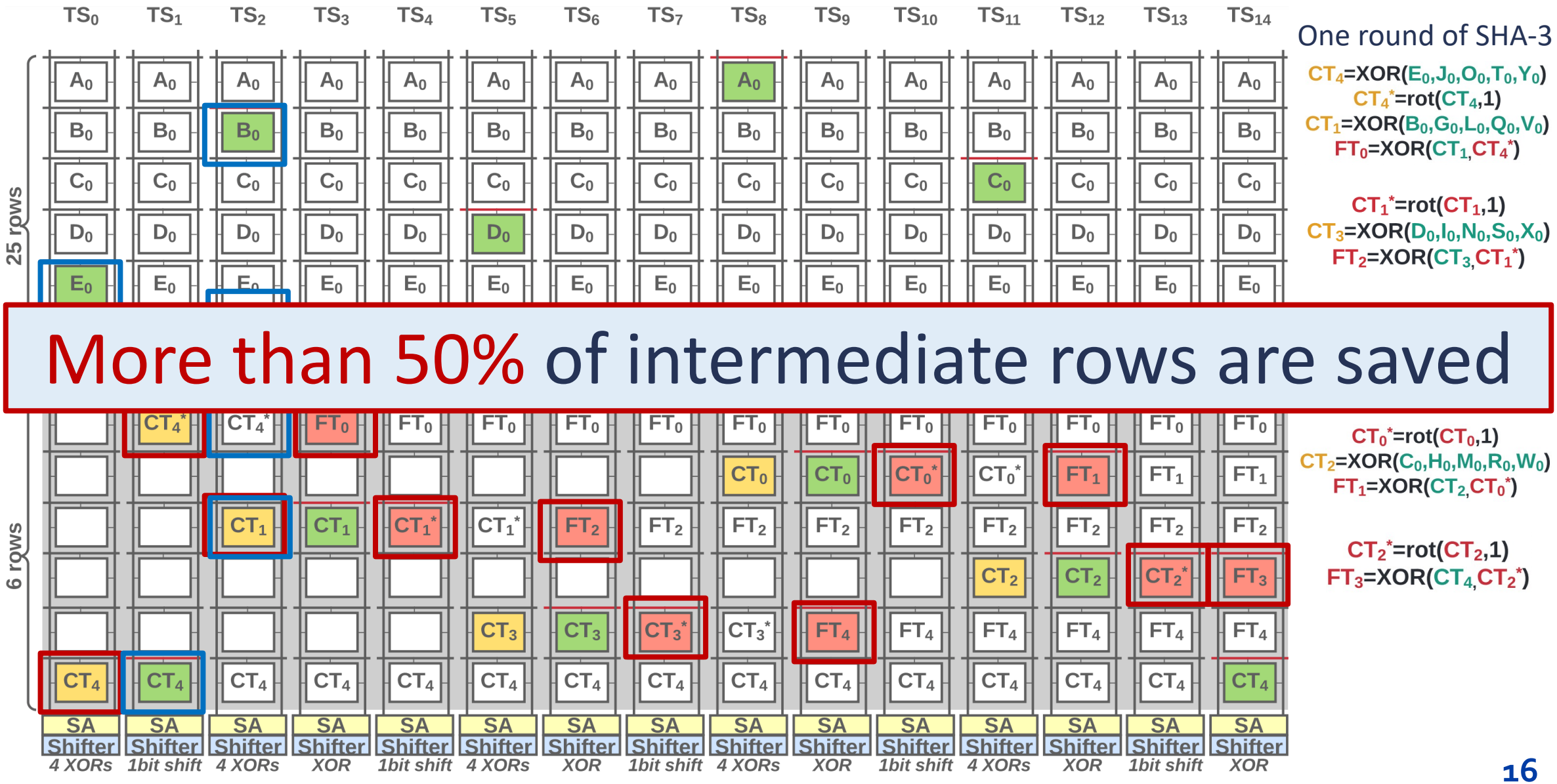
CT_1^*=rot(CT_1,1)
CT_3=XOR(D_0,I_0,N_0,S_0,X_0)
FT_2=XOR(CT_3,CT_1^*)

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



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*)

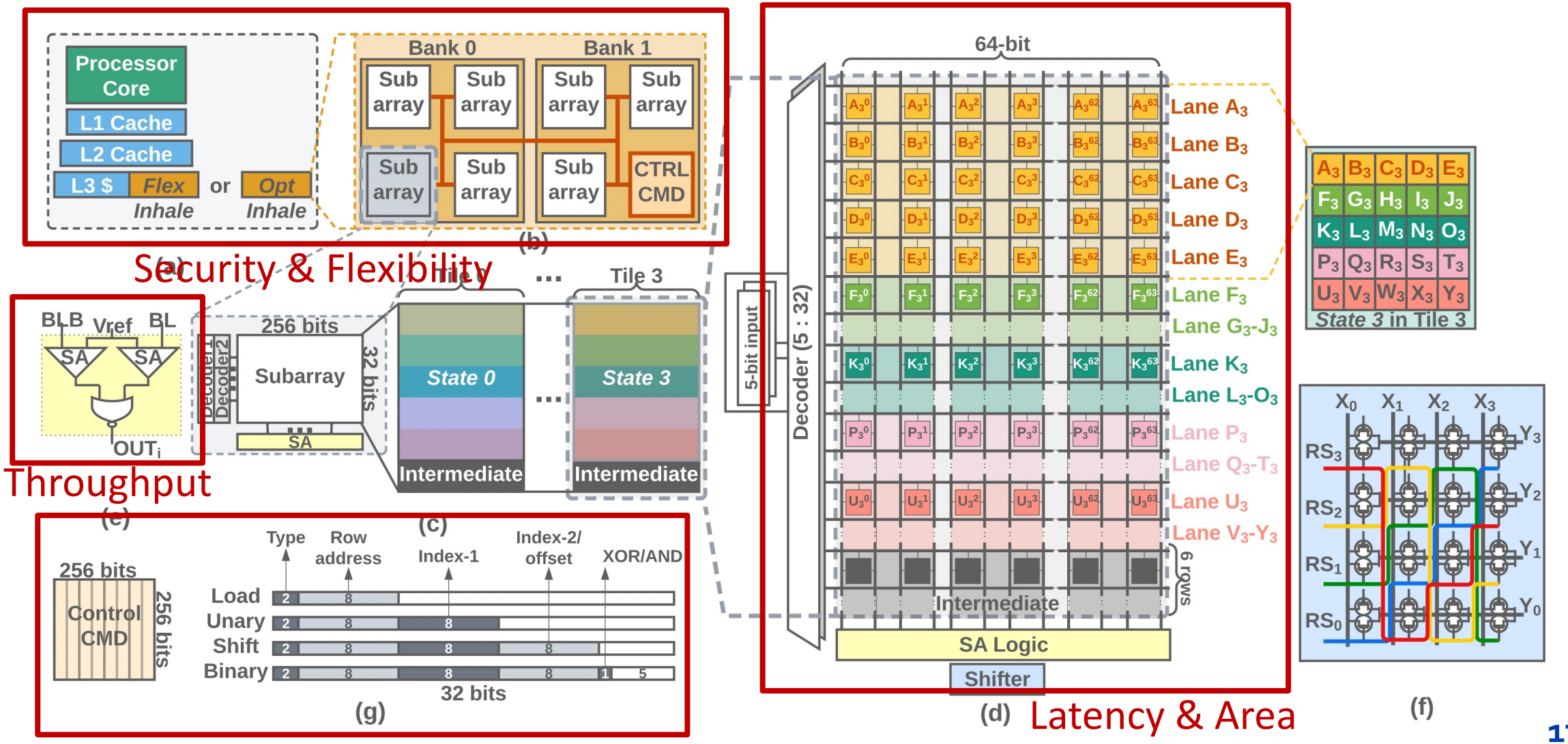
More than 50% of intermediate rows are saved

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



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

	Tech.	Max f (MHz)	Area (KGE)	Latency (cycles)	Latency (ns)	Tput. (Mbps)	Tput./Area (Mbps/KGE)	Energy (nJ)	Tput./Area/En. (Mbps/(KGE·nJ))
Inhale-Opt-SRAM	28nm	6.7K	**63.6**	564	83.6	52K	**818**	**0.456**	**1.8K**
Inhale-Flex-SRAM	28nm	6.1K	386	564	91.9	47.3K	123	0.596	206
				4	39.1	111K	225	-	-
				0	**20.7**	**210K**	293	-	-
				9	4.8K	226	0.377	2.03	0.186
Inhale-Opt-ReRAM	28nm	2.4K	**19.1**	564	235	18.6K	**970**	**0.348**	**2.79K**
Inhale-Flex-ReRAM	28nm	2.3K	56.3	564	240	18.1K	322	0.446	721
SHINE-1-ReRAM[19]	65nm	2K	494	264	132	33K	66.8	4.13	16.2
SHINE-2-ReRAM[19]	65nm	2K	717	140	70	62.2K	86.7	3.5	24.8
SHINE-1-ReRAM (projected)	28nm	4.6K	494	264	56.9	76.5K	155	-	-
SHINE-2-ReRAM (projected)	28nm	4.6K	717	140	**30.2**	**144K**	201	-	-
Akın[2]	90nm	455	10.5K	25	54.9	19.8K	1.89	>43.5	<0.043
Tillich[28]	180nm	488	56.3K	25	51.2	21.2K	0.377	>43.5	<0.009
Pessl-V1 [22]	130nm	1	5.5K	10.7K	10.7M	0.102	18.5e-6	>43.5	<4.25E-7
Pessl-V2 [22]	130nm	1	5.9K	7.4K	7.4M	0.147	24.9e-6	>43.5	<5.73E-7
Wong[30]	65nm	1K	105K	-	-	48K	0.457	>43.5	<0.011

In-memory computing brings advantages

Comparison of different designs

	Tech.	Max f (MHz)	Area (KGE)	Latency (cycles)	Latency (ns)	Tput. (Mbps)	Tput./Area (Mbps/KGE)	Energy (nJ)	Tput./Area/En. (Mbps/(KGE·nJ))
Inhale-Opt-SRAM	28nm	6.7K	**63.6**	564	83.6	52K	**818**	**0.456**	**1.8K**
Inhale-Flex-SRAM	28nm	6.1K	386	564	91.9	47.3K	123	0.596	206
SHINE-1-SRAM	28nm	6.7K	494	264	39.1				-
SHINE-2-SRAM	28nm	6.7K	717	140	**20.7**				-
Recryptor[34]	40nm	28.8	600	139	4.8K				0.186
Inhale-Opt-ReRAM	28nm	2.4K	**19.1**	564	235	18.6K	**970**	**0.348**	**2.79K**
Inhale-Flex-ReRAM	28nm	2.3K	56.3	564	240	18.1K	322	0.446	721
SHINE-1-ReRAM[19]	65nm	2K	494	264	132	33K	66.8	4.13	16.2
SHINE-2-ReRAM[19]	65nm	2K	717	140	70	62.2K	86.7	3.5	24.8
SHINE-1-ReRAM (projected)	28nm	4.6K	494	264	56.9	76.5K	155	-	-
SHINE-2-ReRAM (projected)	28nm	4.6K	717	140	**30.2**	**144K**	201	-	-
Akın[2]	90nm	455	10.5K	25	54.9	19.8K	1.89	>43.5	<0.043
Tillich[28]	180nm	488	56.3K	25	51.2	21.2K	0.377	>43.5	<0.009
Pessl-V1 [22]	130nm	1	5.5K	10.7K	10.7M	0.102	18.5e-6	>43.5	<4.25E-7
Pessl-V2 [22]	130nm	1	5.9K	7.4K	7.4M	0.147	24.9e-6	>43.5	<5.73E-7
Wong[30]	65nm	1K	105K	-	-	48K	0.457	>43.5	<0.011

Higher frequency

Comparison of different designs

	Tech.	Max f (MHz)	Area (KGE)	Latency (cycles)	Latency (ns)	Tput. (Mbps)	Tput./Area (Mbps/KGE)	Energy (nJ)	Tput./Area/En. (Mbps/(KGE·nJ))
Inhale-Opt-SRAM	28nm	6.7K	**63.6**	564	83.6	52K	**818**	**0.456**	**1.8K**
Inhale-Flex-SRAM	28nm	6.1K	386	564	91.9	47.3K	123	0.596	206
SHINE-1-SRAM	28nm	6.7K	494	264	39.1	111K	225	-	-
SHINE-2-SRAM	28nm	6.7K	717	140	**20.7**	**210K**	293	-	-
Recryptor[34]	40nm	28.8	600	139	4.8K				186
Inhale-Opt-ReRAM	28nm	2.4K	**19.1**	564	235				**79K**
Inhale-Flex-ReRAM	28nm	2.3K	56.3	564	240				21
SHINE-1-ReRAM[19]	65nm	2K	494	264	132	33K	66.8	4.13	16.2
SHINE-2-ReRAM[19]	65nm	2K	717	140	70	62.2K	86.7	3.5	24.8
SHINE-1-ReRAM (projected)	28nm	4.6K	494	264	56.9	76.5K	155	-	-
SHINE-2-ReRAM (projected)	28nm	4.6K	717	140	**30.2**	**144K**	201	-	-
Akın[2]	90nm	455	10.5K	25	54.9	19.8K	1.89	>43.5	<0.043
Tillich[28]	180nm	488	56.3K	25	51.2	21.2K	0.377	>43.5	<0.009
Pessl-V1 [22]	130nm	1	5.5K	10.7K	10.7M	0.102	18.5e-6	>43.5	<4.25E-7
Pessl-V2 [22]	130nm	1	5.9K	7.4K	7.4M	0.147	24.9e-6	>43.5	<5.73E-7
Wong[30]	65nm	1K	105K	-	-	48K	0.457	>43.5	<0.011

Dedicated multi-bit XOR logic

Comparison of different designs

	Tech.	Max f (MHz)	Area (KGE)	Latency (cycles)	Latency (ns)	Tput. (Mbps)	Tput./Area (Mbps/KGE)	Energy (nJ)	Tput./Area/En. (Mbps/(KGE·nJ))
Inhale-Opt-SRAM	28nm	6.7K	**63.6**	564	83.6	52K	**818**	**0.456**	**1.8K**
Inhale-Flex-SRAM	28nm	6.1K	386	564	91.9	47.3K	123	0.596	206
SHINE-1-SRAM	28nm	6.7K					225	-	-
SHINE-2-SRAM	28nm	6.7K					293	-	-
Recryptor[34]	40nm	28.8					0.377	2.03	0.186
Inhale-Opt-ReRAM	28nm	2.4K	**19.1**	564	235	K	**970**	**0.348**	**2.79K**
Inhale-Flex-ReRAM	28nm	2.3K	56.3	564	240	18.1K	322	0.446	721
SHINE-1-ReRAM[19]	65nm	2K	494	264	132	33K	66.8	4.13	16.2
SHINE-2-ReRAM[19]	65nm	2K	717	140	70	62.2K	86.7	3.5	24.8
SHINE-1-ReRAM (projected)	28nm	4.6K	494	264	56.9	76.5K	155	-	-
SHINE-2-ReRAM (projected)	28nm	4.6K	717	140	**30.2**	**144K**	201	-	-
Akın[2]	90nm	455	10.5K	25	54.9	19.8K	1.89	>43.5	<0.043
Tillich[28]	180nm	488	56.3K	25	51.2	21.2K	0.377	>43.5	<0.009
Pessl-V1 [22]	130nm	1	5.5K	10.7K	10.7M	0.102	18.5e-6	>43.5	<4.25E-7
Pessl-V2 [22]	130nm	1	5.9K	7.4K	7.4M	0.147	24.9e-6	>43.5	<5.73E-7
Wong[30]	65nm	1K	105K	-	-	48K	0.457	>43.5	<0.011

Inhale has smaller area

Comparison of different designs

	Tech.	Max f (MHz)	Area (KGE)	Latency (cycles)	Latency (ns)	Tput. (Mbps)	Tput./Area (Mbps/KGE)	Energy (nJ)	Tput./Area/En. (Mbps/(KGE·nJ))
Inhale-Opt-SRAM	28nm	6.7K	**63.6**	564	83.6	52K	**818**	**0.456**	**1.8K**
Inhale-Flex-SRAM	28nm	6.1K	386	564	91.9	47.3K	123	0.596	206
			494	264	39.1	111K	225	-	-
			717	140	**20.7**	**210K**	293	-	-
			600	139	4.8K	226	0.377	2.03	0.186
Inhale-Opt-ReRAM	28nm	[illegible]	**19.1**	564	235	18.6K	**970**	**0.348**	**2.79K**
Inhale-Flex-ReRAM	28nm	2.3K	56.3	564	240	18.1K	322	0.446	721
SHINE-1-ReRAM[19]	65nm	2K	494	264	132	33K	66.8	4.13	16.2
SHINE-2-ReRAM[19]	65nm	2K	717	140	70	62.2K	86.7	3.5	24.8
SHINE-1-ReRAM (projected)	28nm	4.6K	494	264	56.9	76.5K	155	-	-
SHINE-2-ReRAM (projected)	28nm	4.6K	717	140	**30.2**	**144K**	201	-	-
Akın[2]	90nm	455	10.5K	25	54.9	19.8K	1.89	>43.5	<0.043
Tillich[28]	180nm	488	56.3K	25	51.2	21.2K	0.377	>43.5	<0.009
Pessl-V1 [22]	130nm	1	5.5K	10.7K	10.7M	0.102	18.5e-6	>43.5	<4.25E-7
Pessl-V2 [22]	130nm	1	5.9K	7.4K	7.4M	0.147	24.9e-6	>43.5	<5.73E-7
Wong[30]	65nm	1K	105K	-	-	48K	0.457	>43.5	<0.011

70x fewer cells & 131x fewer peripheral logics

Comparison of different designs

	Tech.	Max f (MHz)	Area (KGE)	Latency (cycles)	Latency (ns)	Tput. (Mbps)	Tput./Area (Mbps/KGE)	Energy (nJ)	Tput./Area/En. (Mbps/(KGE·nJ))
Inhale-Opt-SRAM	28nm	6.7K	**63.6**	564	83.6	52K	**818**	**0.456**	**1.8K**
Inhale-Flex-SRAM	28nm	6.1K	386	564	91.9	47.3K	123	0.596	206
SHINE-1-SRAM	28nm	6.7K	494					-	-
SHINE-2-SRAM	28nm	6.7K	717					-	-
Recryptor[34]	40nm	28.8	600					2.03	0.186
Inhale-Opt-ReRAM	28nm	2.4K	**19.1**	564	235	18.6K	**970**	**0.348**	**2.79K**
Inhale-Flex-ReRAM	28nm	2.3K	56.3	564	240	18.1K	322	0.446	721
SHINE-1-ReRAM[19]	65nm	2K	494	264	132	33K	66.8	4.13	16.2
SHINE-2-ReRAM[19]	65nm	2K	717	140	70	62.2K	86.7	3.5	24.8
SHINE-1-ReRAM (projected)	28nm	4.6K	494	264	56.9	76.5K	155	-	-
SHINE-2-ReRAM (projected)	28nm	4.6K	717	140	**30.2**	**144K**	201	-	-
Akın[2]	90nm	455	10.5K	25	54.9	19.8K	1.89	>43.5	<0.043
Tillich[28]	180nm	488	56.3K	25	51.2	21.2K	0.377	>43.5	<0.009
Pessl-V1 [22]	130nm	1	5.5K	10.7K	10.7M	0.102	18.5e-6	>43.5	<4.25E-7
Pessl-V2 [22]	130nm	1	5.9K	7.4K	7.4M	0.147	24.9e-6	>43.5	<5.73E-7
Wong[30]	65nm	1K	105K	-	-	48K	0.457	>43.5	<0.011

Almost no inter-subarray data movement

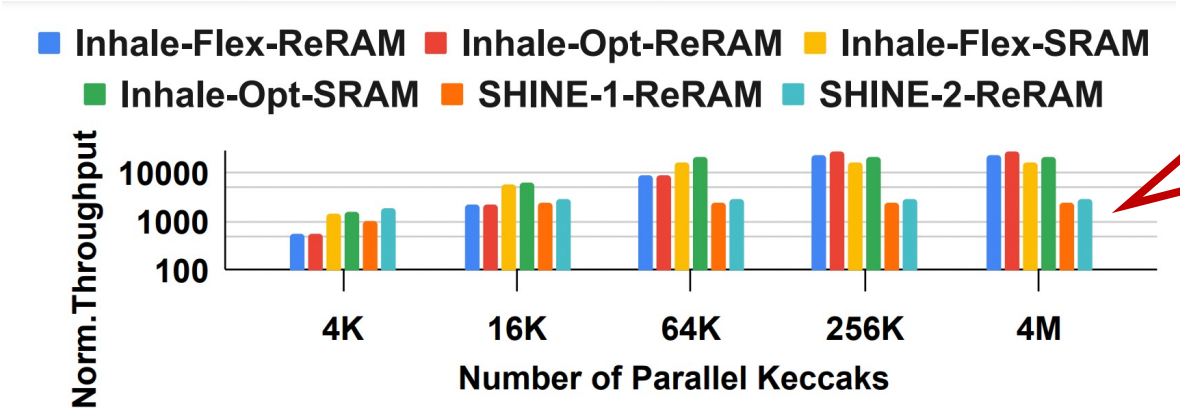
Comparison of different designs

	Tech.	Max f (MHz)	Area (KGE)	Latency (cycles)	Latency (ns)	Tput. (Mbps)	Tput./Area (Mbps/KGE)	Energy (nJ)	Tput./Area/En. (Mbps/(KGE·nJ))
Inhale-Opt-SRAM	28nm	6.7K						**0.456**	**1.8K**
Inhale-Flex-SRAM	28nm	6.1K						0.596	206
SHINE-1-SRAM	28nm	6.7K						-	-
SHINE-2-SRAM	28nm	6.7K						-	-
Recryptor[34]	40nm	28.8		139	4.8K	226	0.377	2.03	0.186
Inhale-Opt-ReRAM	28nm	2.4K	**19.1**	564	235	18.6K	**970**	**0.348**	**2.79K**
Inhale-Flex-ReRAM	28nm	2.3K	56.3	564	240	18.1K	322	0.446	721
SHINE-1-ReRAM[19]	65nm	2K	494	264	132	33K	66.8	4.13	16.2
SHINE-2-ReRAM[19]	65nm	2K	717	140	70	62.2K	86.7	3.5	24.8
SHINE-1-ReRAM (projected)	28nm	4.6K	494	264	56.9	76.5K	155	-	-
SHINE-2-ReRAM (projected)	28nm	4.6K	717	140	**30.2**	**144K**	201	-	-
Akın[2]	90nm	455	10.5K	25	54.9	19.8K	1.89	>43.5	<0.043
Tillich[28]	180nm	488	56.3K	25	51.2	21.2K	0.377	>43.5	<0.009
Pessl-V1 [22]	130nm	1	5.5K	10.7K	10.7M	0.102	18.5e-6	>43.5	<4.25E-7
Pessl-V2 [22]	130nm	1	5.9K	7.4K	7.4M	0.147	24.9e-6	>43.5	<5.73E-7
Wong[30]	65nm	1K	105K	-	-	48K	0.457	>43.5	<0.011

Fewer traversal time on bitlines

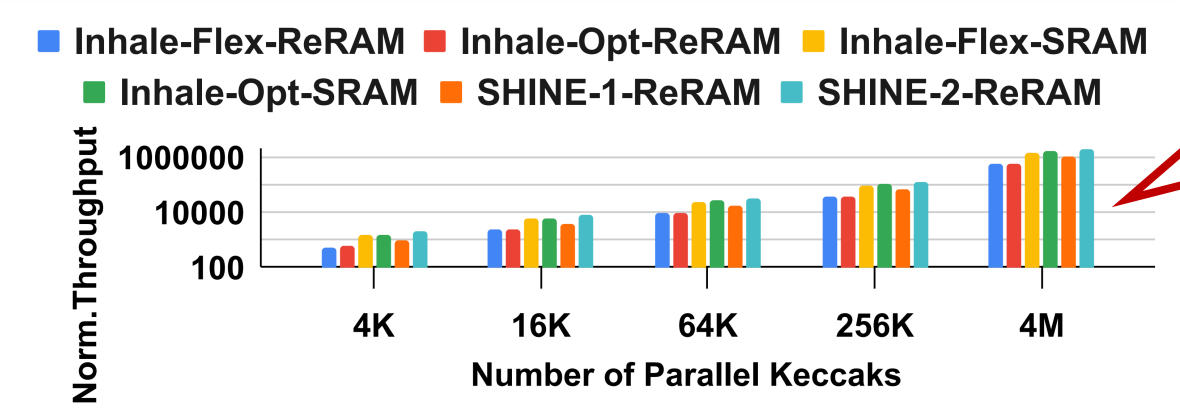
Performance Scaling

- With power constraint



SHINE hits power earlier than *Inhale*

- Without power constraint



IoT devices have tight power budget

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

Q&A