http://synergy.ece.gatech.edu

SIGMA: A Sparse and Irregular GEMM Accelerator with Flexible Interconnects for DNN Training

ERIC QIN[*], ANANDA SAMAJDAR[*], HYOUKJUN KWON[*],
VINEET NADELLA[*], SUDARSHAN SRINIVASAN[#], DIPANKAR DAS[#],
BHARAT KAUL[#], TUSHAR KRISHNA[*]

[*] GEORGIA TECH
[#] INTEL

Outline

- Motivation
    - GEMMs in Deep Learning
    - Utilization on TPU (Systolic Array)
- Accelerator Requirements
- SIGMA
    - Interconnects Implementations
    - Full System Design
- Evaluation
- Conclusion

Outline

- **Motivation**
  - **GEMMs in Deep Learning**
  - Utilization on TPU (Systolic Array)
- Accelerator Requirements
- SIGMA
  - Interconnects Implementations
  - Full System Design
- Evaluation
- Conclusion

Deep Learning Applications

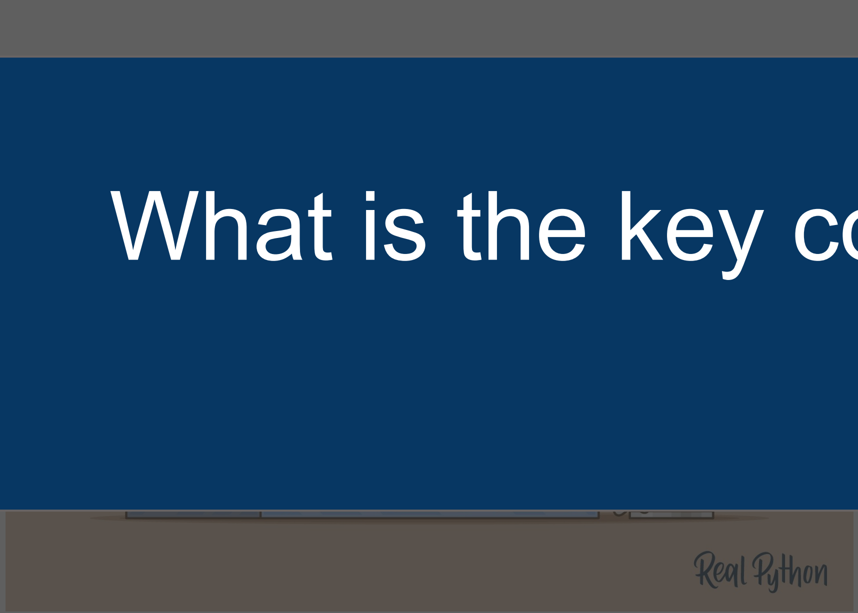
Speech Recognition

Language Understanding

Recommender Systems

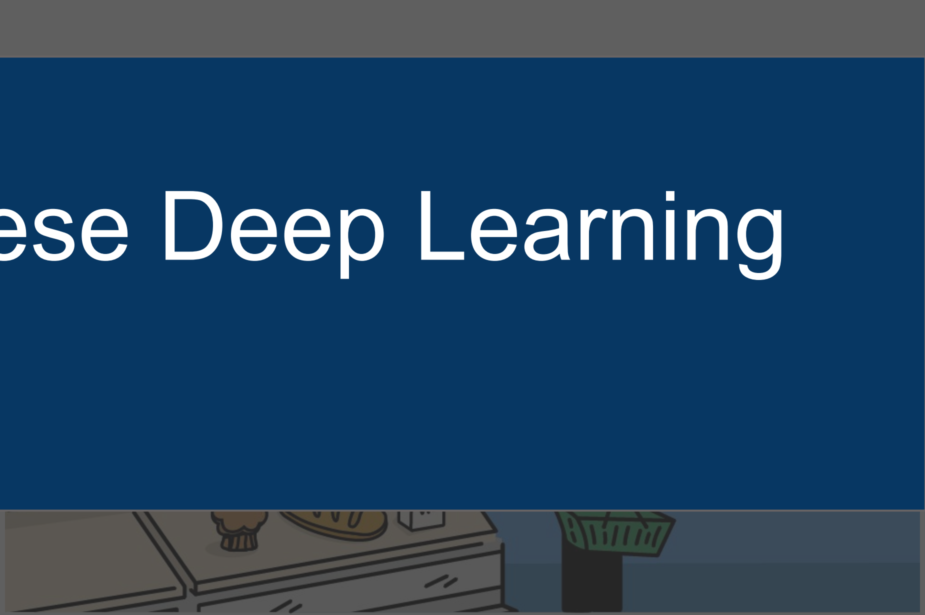
Deep Learning Applications

What is the key computation for these Deep Learning applications?



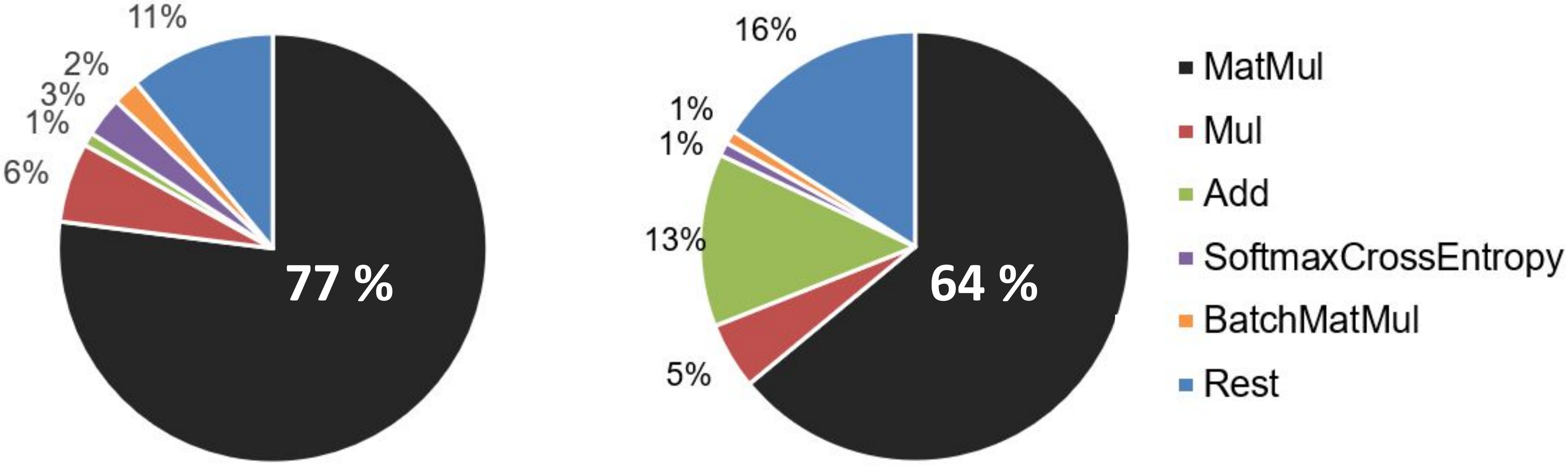
Speech Recognition

Language Understanding



Recommender Systems

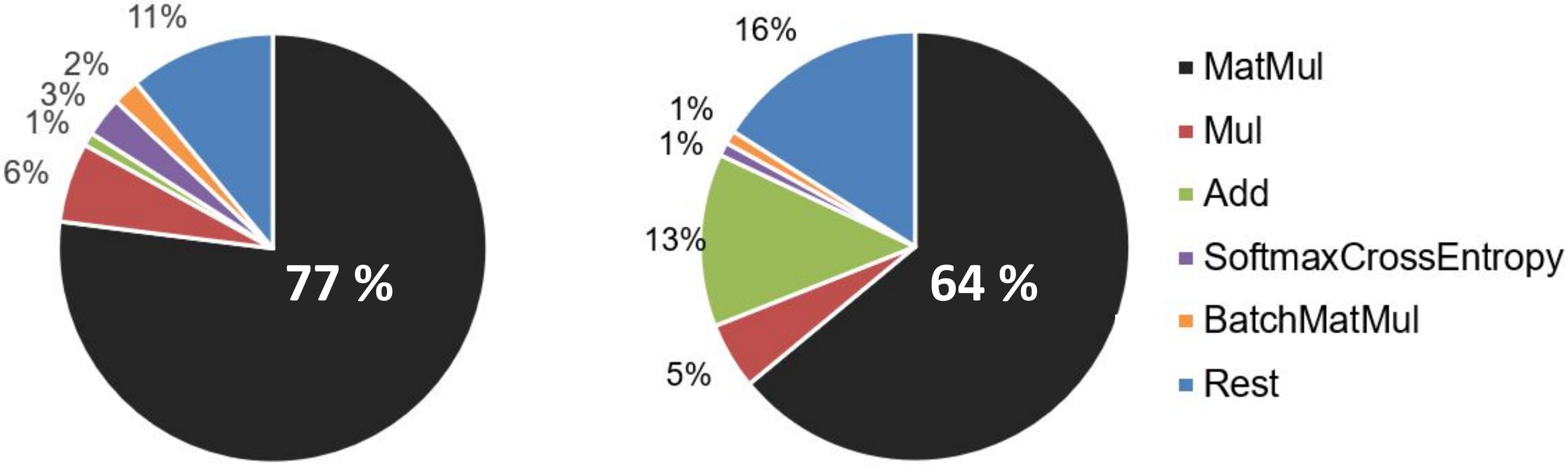
Runtime breakdown on V100 GPU



Transformer
(Language Understanding)

GNMT
(Machine Translation)

Runtime breakdown on V100 GPU



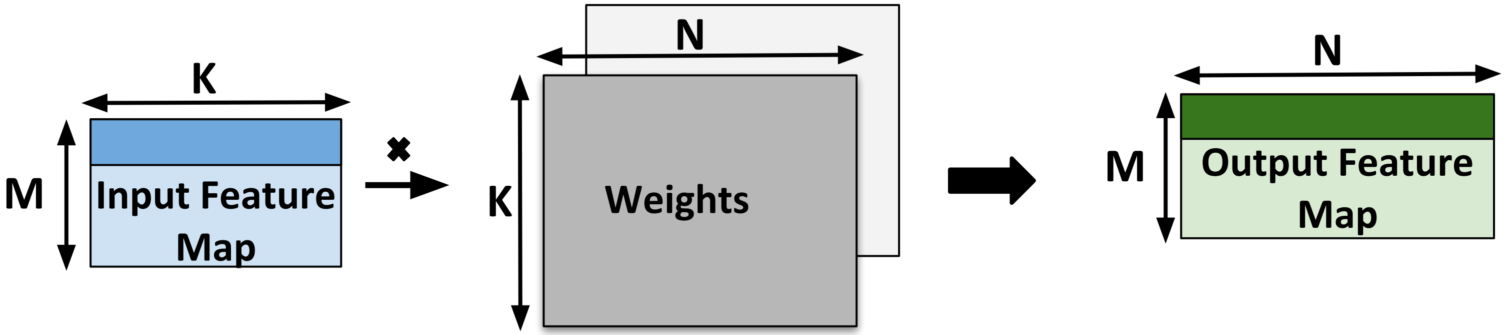
Transformer
(Language Understanding)

GNMT
(Machine Translation)

Matrix multiplications (GEMMs) consume around **70%** of the total runtime when training modern deep learning workloads.

GEMMs in Deep Learning

Forward Pass
(Inference and Training)



GEMM MNK Dimension Representation

M dim: batch size

N dim: number of channels in the next layer

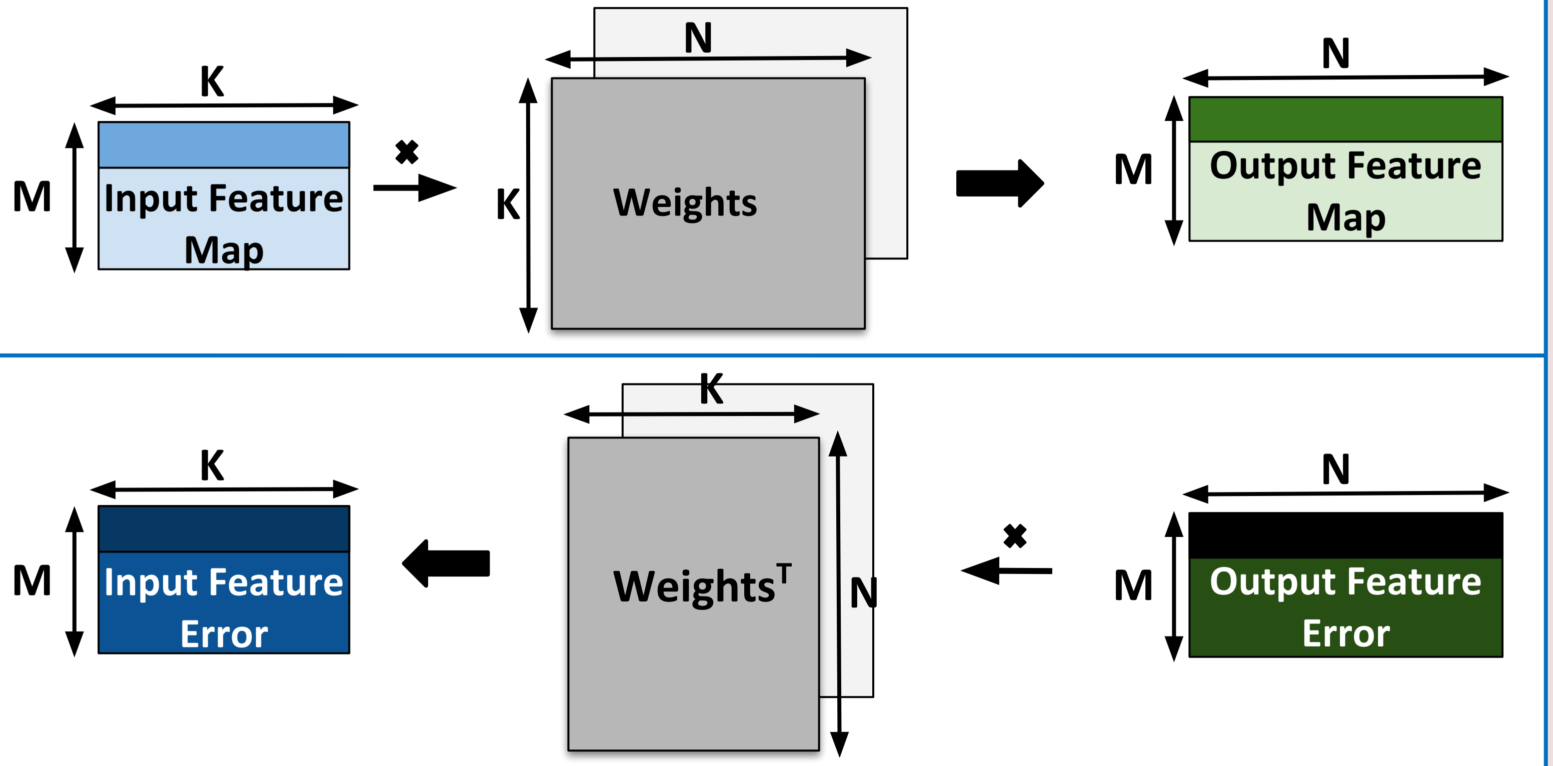
K dim: [H * W * C]

Figure derived from *HyPar: Towards Hybrid Parallelism for Deep Learning Accelerator Array, HPCA 2019, Song et al.*

GEMMs in Deep Learning

Forward Pass
(Inference and Training)

Backward Pass
(Training)



GEMM MNK Dimension Representation

M dim: batch size

N dim: number of channels in the next layer

K dim: [H * W * C]

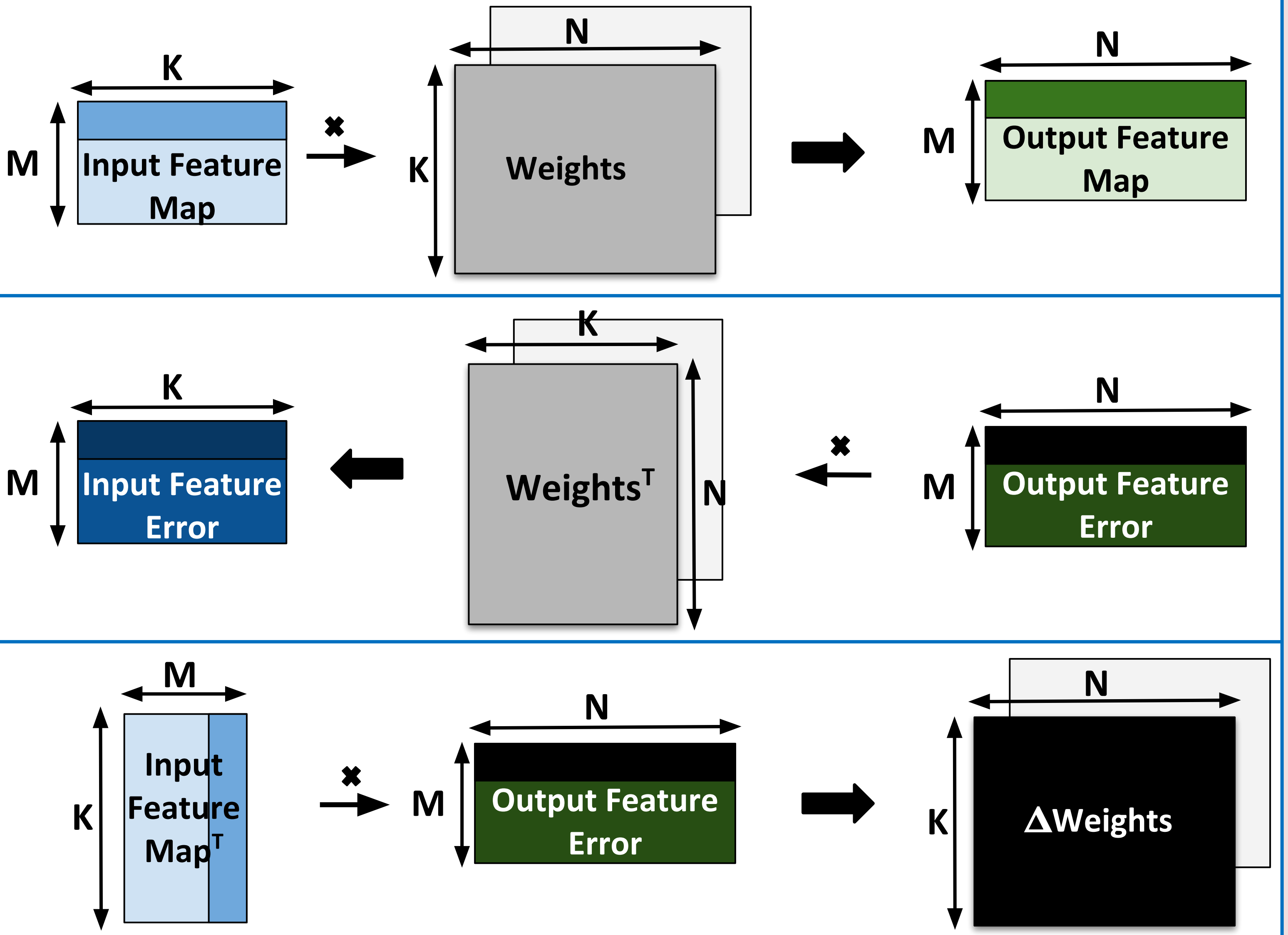
Figure derived from *HyPar: Towards Hybrid Parallelism for Deep Learning Accelerator Array, HPCA 2019, Song et al.*

GEMMs in Deep Learning

Forward Pass
(Inference and Training)

Backward Pass
(Training)

Gradient Computation
(Training)



GEMM MNK Dimension Representation

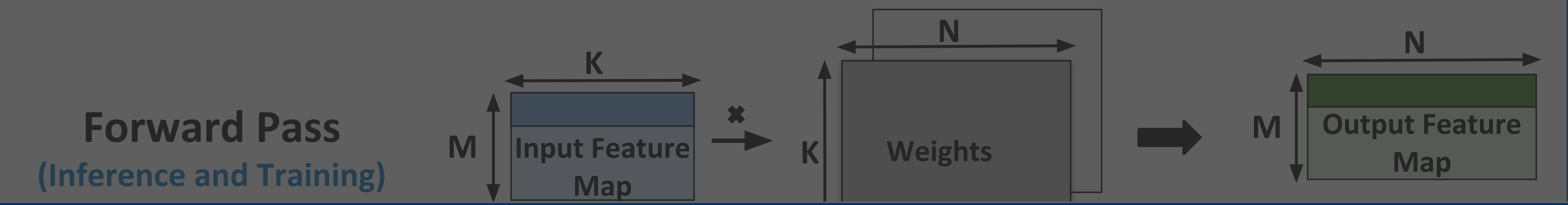
M dim: batch size

N dim: number of channels in the next layer

K dim: [H * W * C]

Figure derived from *HyPar: Towards Hybrid Parallelism for Deep Learning Accelerator Array, HPCA 2019, Song et al.*

GEMMs in Deep Learning



GEMM MNK Dimension Representation

M dim: batch size

GEMM is a key compute primitive to accelerate in hardware to speed up training.

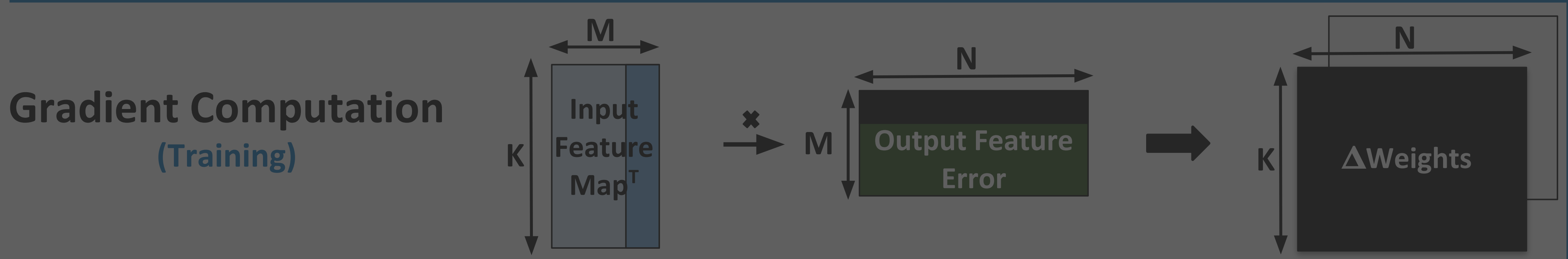


Figure derived from *HyPar: Towards Hybrid Parallelism for Deep Learning Accelerator Array, HPCA 2019, Song et al.*

Hardware for Accelerating GEMMs

SIMT Architectures

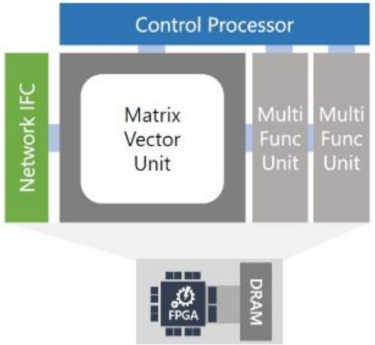
Nvidia GTX GPUs

Hardware for Accelerating GEMMs

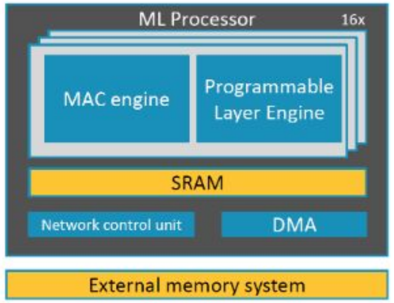
SIMT Architectures

Nvidia GTX GPUs

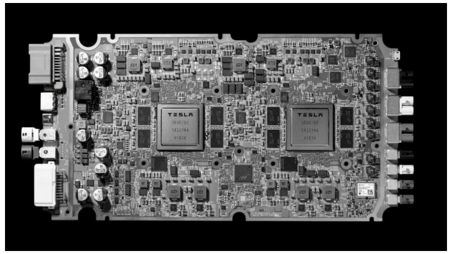
SIMD Architectures



Microsoft Brainwave



ARM Trillium



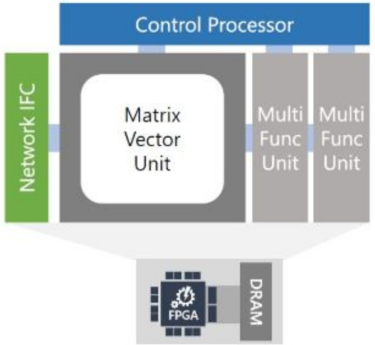
Tesla FSDC

Hardware for Accelerating GEMMs

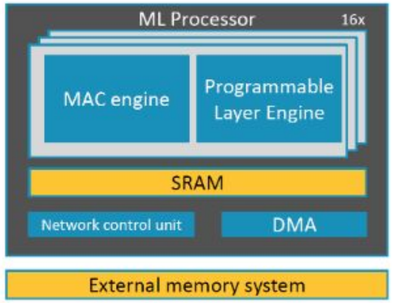
SIMT Architectures

Nvidia GTX GPUs

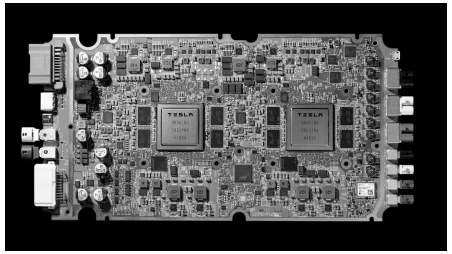
SIMD Architectures



Microsoft Brainwave

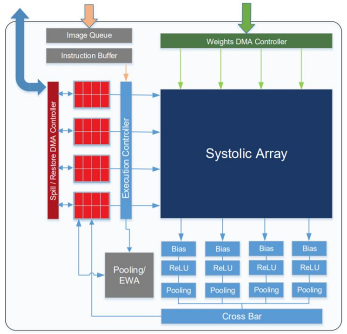


ARM Trillium

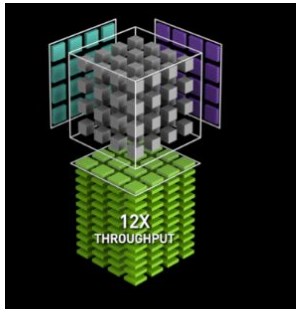


Tesla FSDC

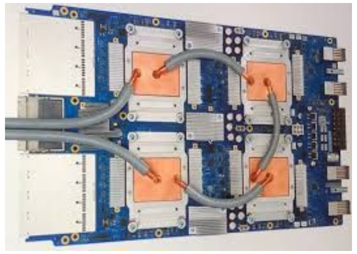
Systolic Architectures



Xilinx xDNN



Nvidia Tensor Cores



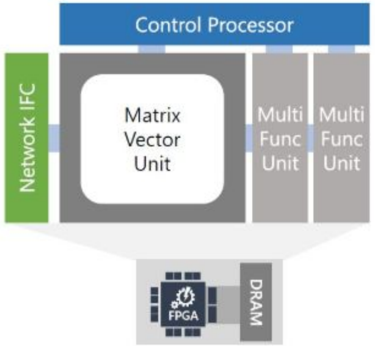
Google TPU

Hardware for Accelerating GEMMs

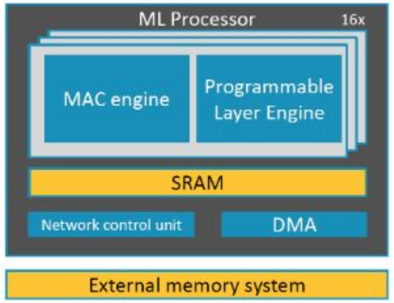
SIMT Architectures

Nvidia GTX GPUs

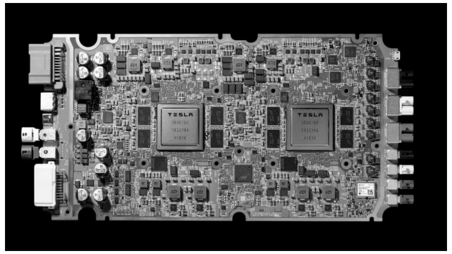
SIMD Architectures



Microsoft Brainwave

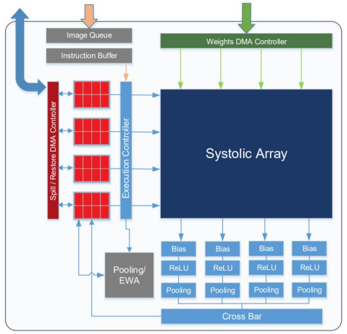


ARM Trillium

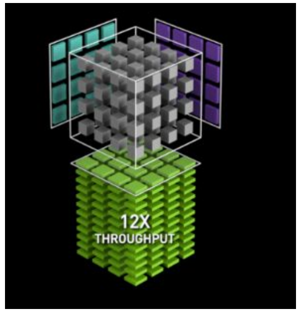


Tesla FSDC

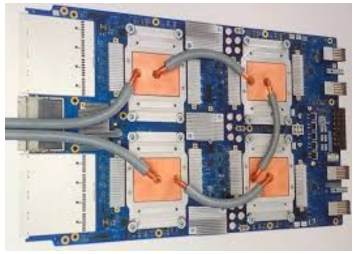
Systolic Architectures



Xilinx xDNN



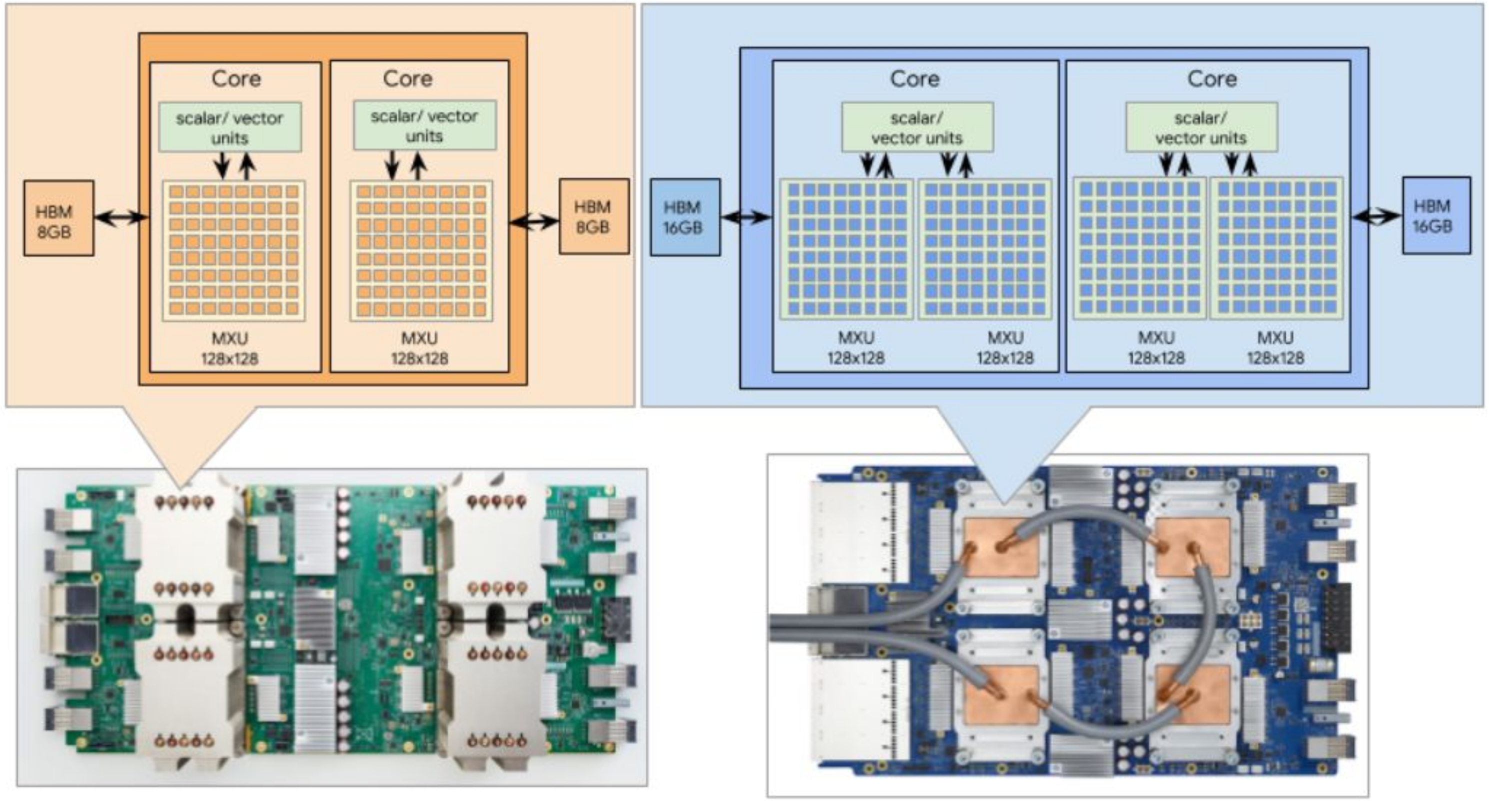
Nvidia Tensor Cores



Google TPU

Recently, systolic array based architectures are popular for accelerating GEMMs.

Target comparison: Google TPU



TPU v2 - 4 chips, 2 cores per chip

TPU v3 - 4 chips, 2 cores per chip

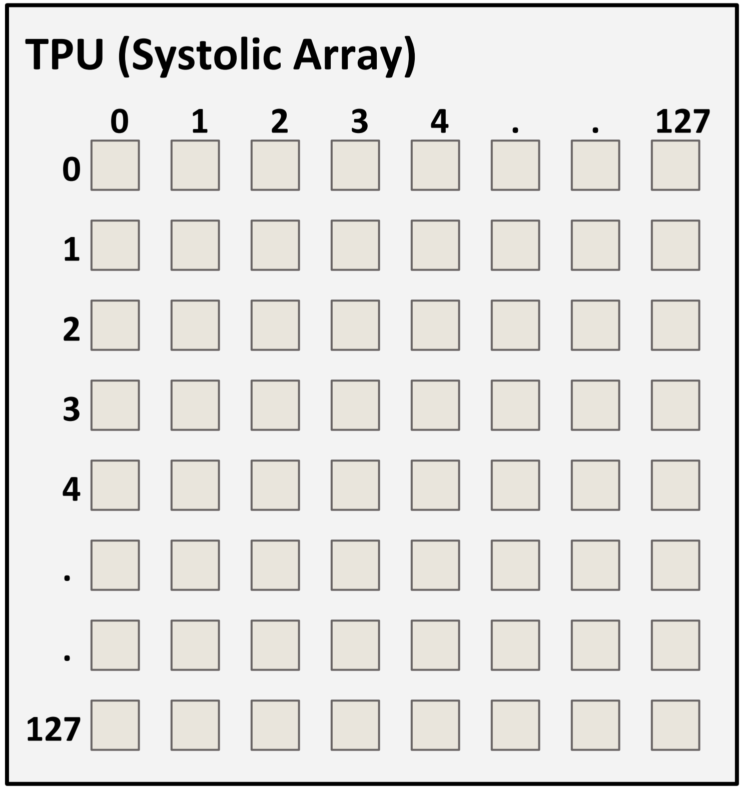
Our target comparison is with the Google TPU, which uses **128 x 128** systolic arrays.

TPU figure from https://cloud.google.com/tpu/docs/system-architecture

Outline

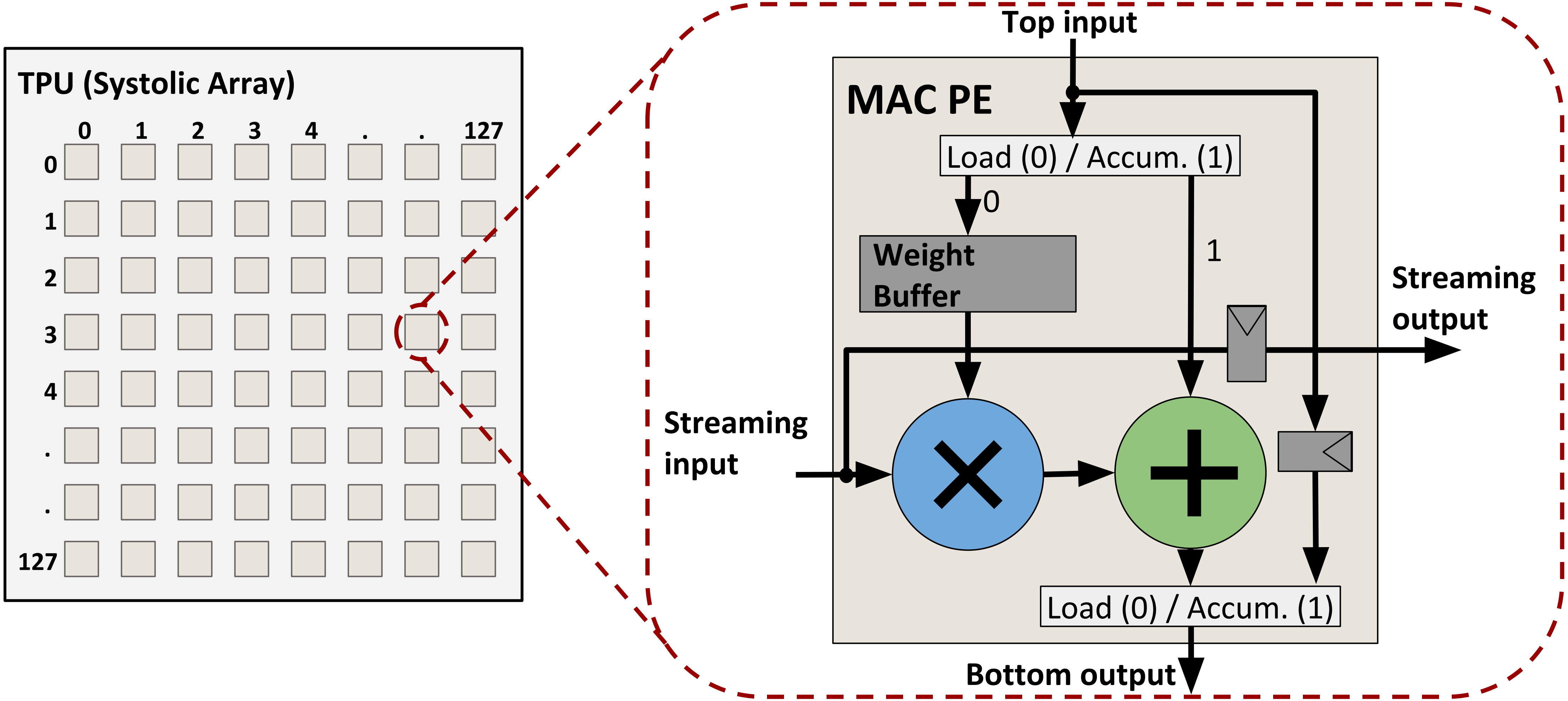
- **Motivation**
  - GEMMs in Deep Learning
  - **Utilization on TPU (Systolic Array)**
- Accelerator Requirements
- SIGMA
  - Interconnects Implementations
  - Full System Design
- Evaluation
- Conclusion

Systolic Array Architectures

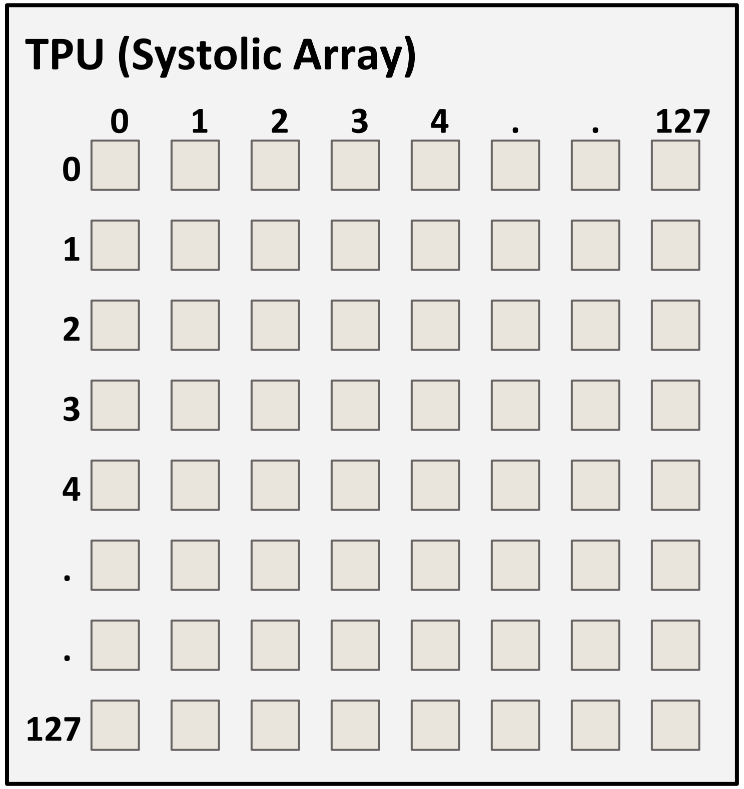


TPU (Systolic Array)

Systolic Array Architectures



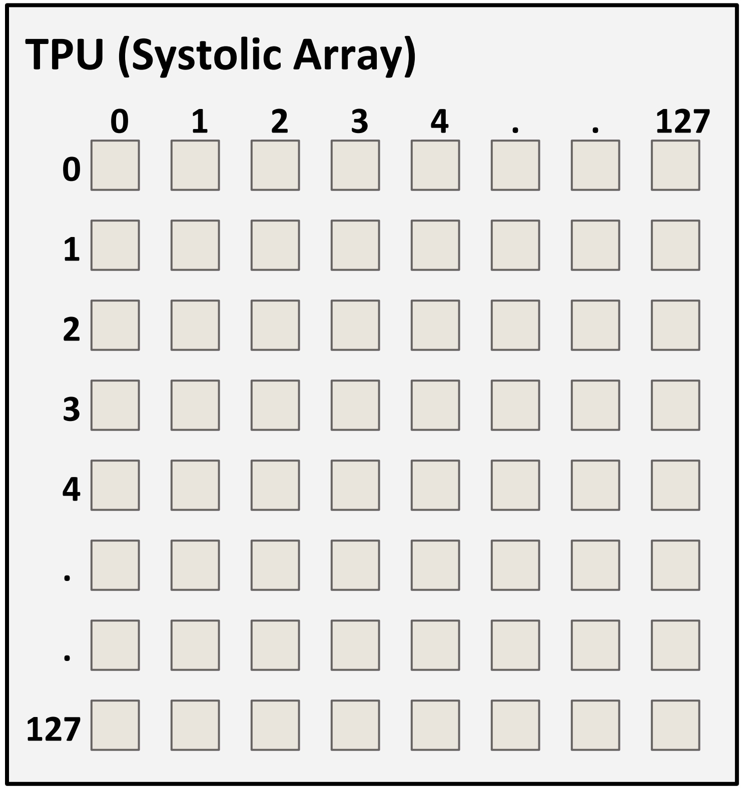
Mapping GEMMs onto TPUs



Workload	Application	Example Dimensions		
		M	N	K
GNMT	Machine Translation	128	2048	4096
		320	3072	4096
		1632	36548	1024
		2048	4096	32
DeepBench	General Workload	1024	16	500000
		35	8457	2560
Transformer	Language Understanding	31999	1024	84
		84	1024	4096
NCF	Collaborative Filtering	2048	1	128
		256	256	2048

GEMMs used for evaluation.

Mapping GEMMs onto TPUs

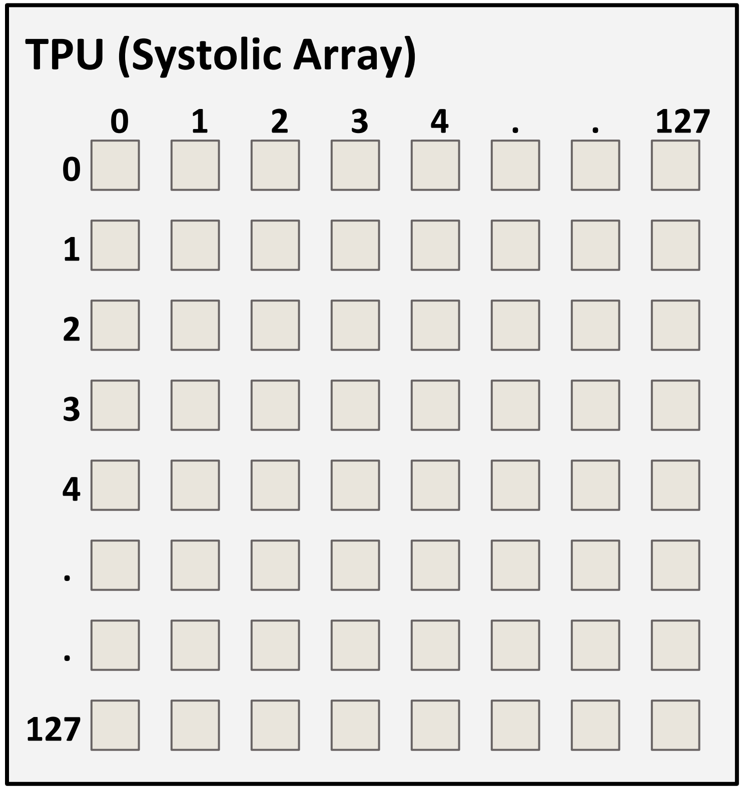


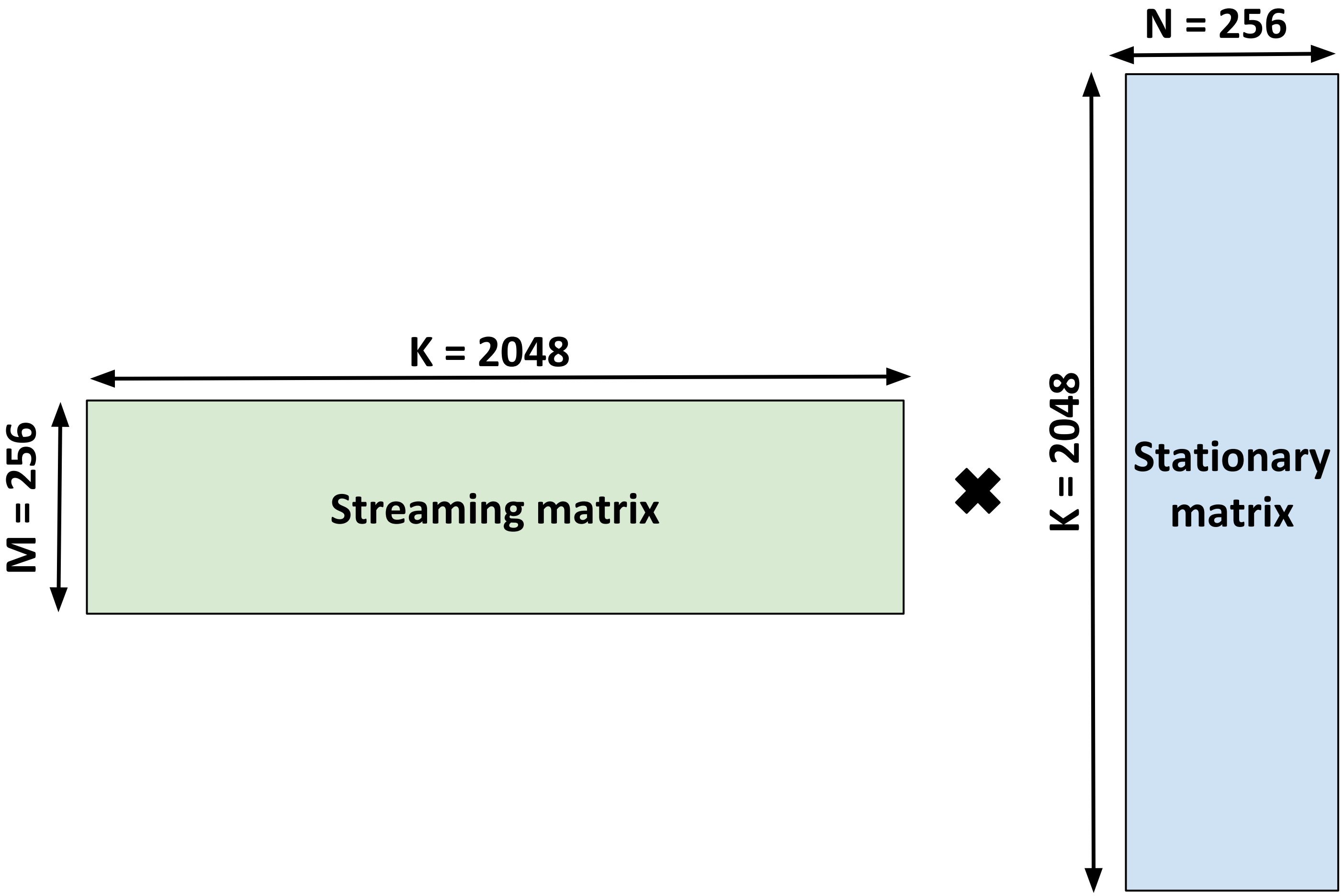
Workload	Application	Example Dimensions		
		M	N	K
GNMT	Machine Translation	128	2048	4096
		320	3072	4096
		1632	36548	1024
		2048	4096	32
DeepBench	General Workload	1024	16	500000
		35	8457	2560
Transformer	Language Understanding	31999	1024	84
		84	1024	4096
NCF	Collaborative Filtering	2048	1	128
		256	256	2048

GEMMs used for evaluation.

Let's map this GEMM!

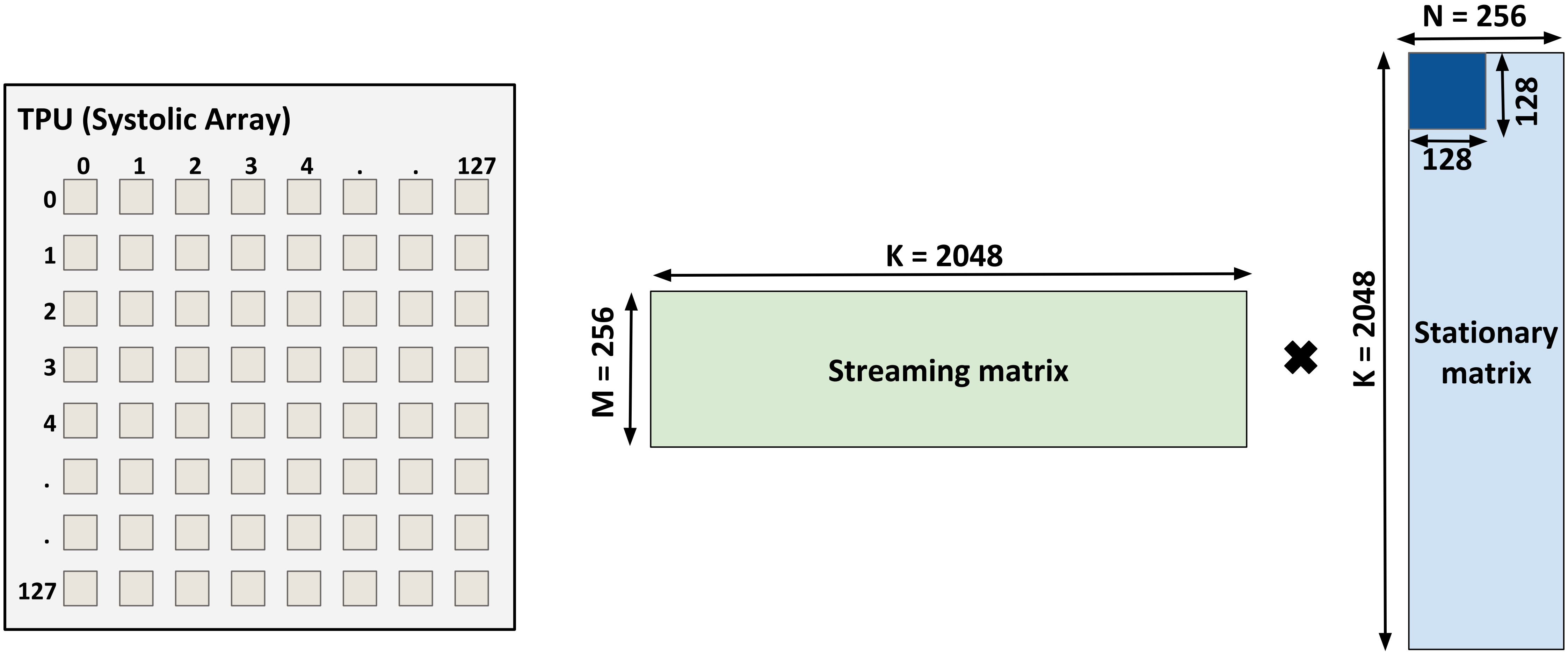
Mapping GEMMs onto TPUs





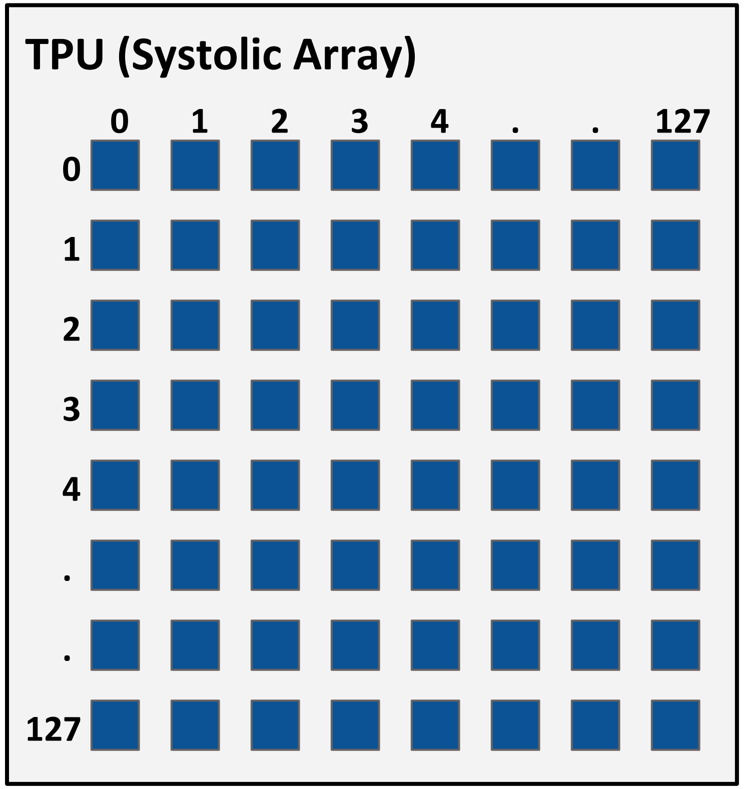
***** Assuming MK matrix is streaming and KN matrix is stationary. (aka weight stationary)***

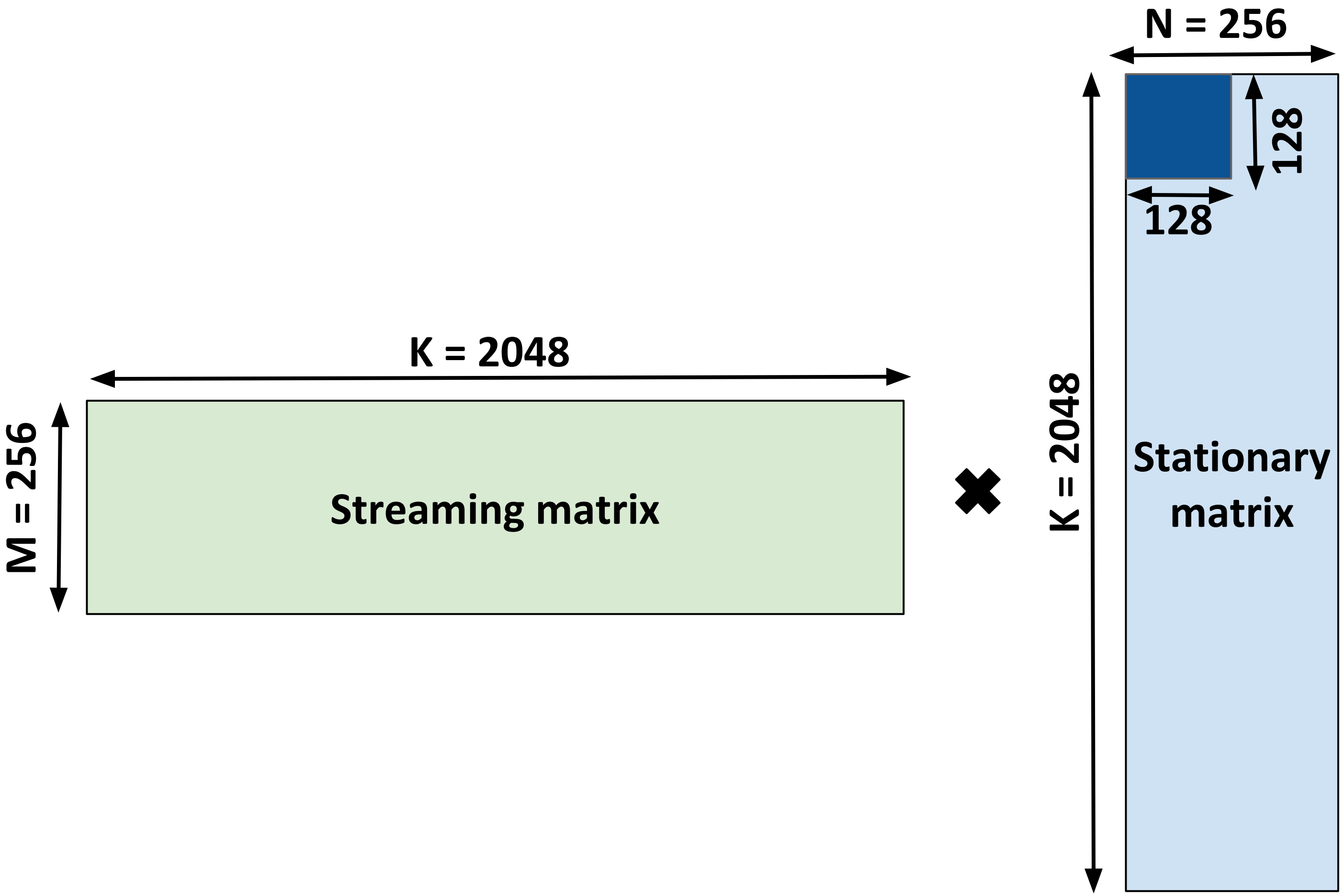
Mapping GEMMs onto TPUs



*** Assuming MK matrix is streaming and KN matrix is stationary. (aka weight stationary)*

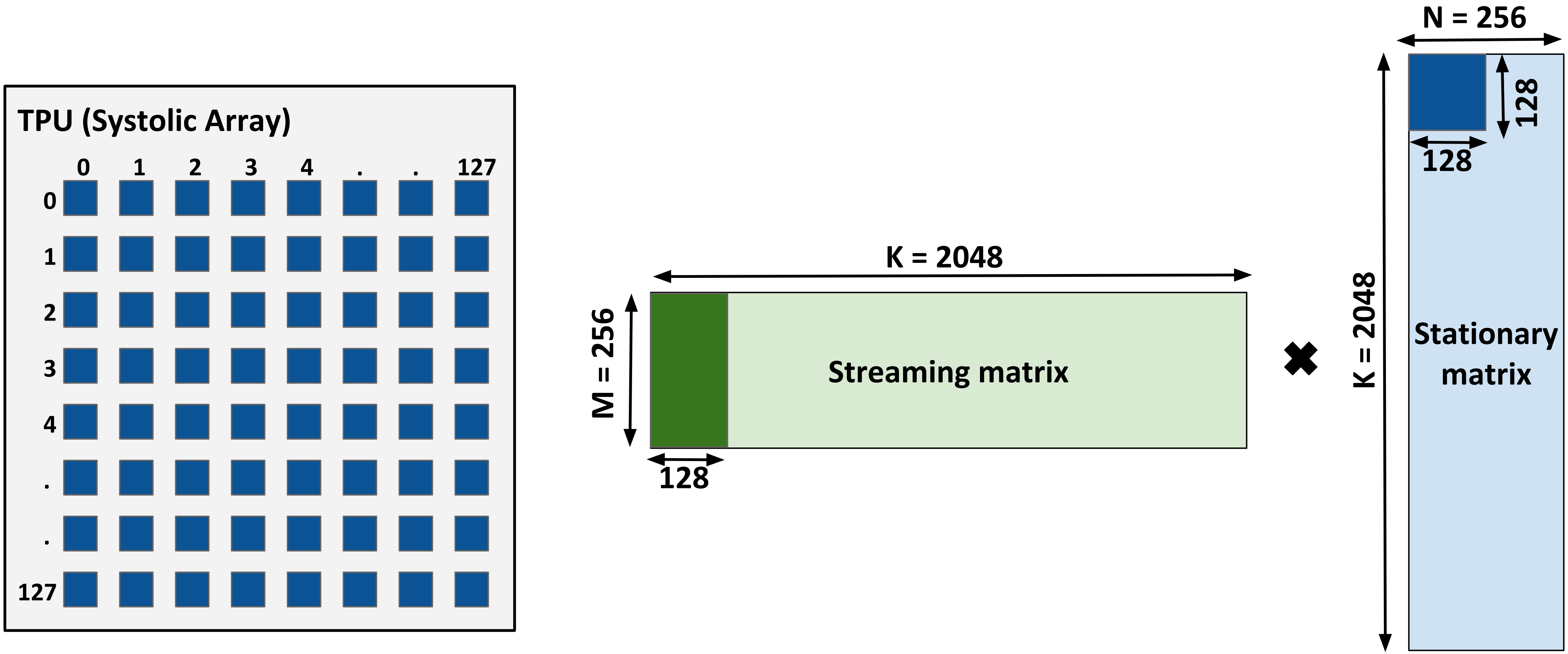
Mapping GEMMs onto TPUs





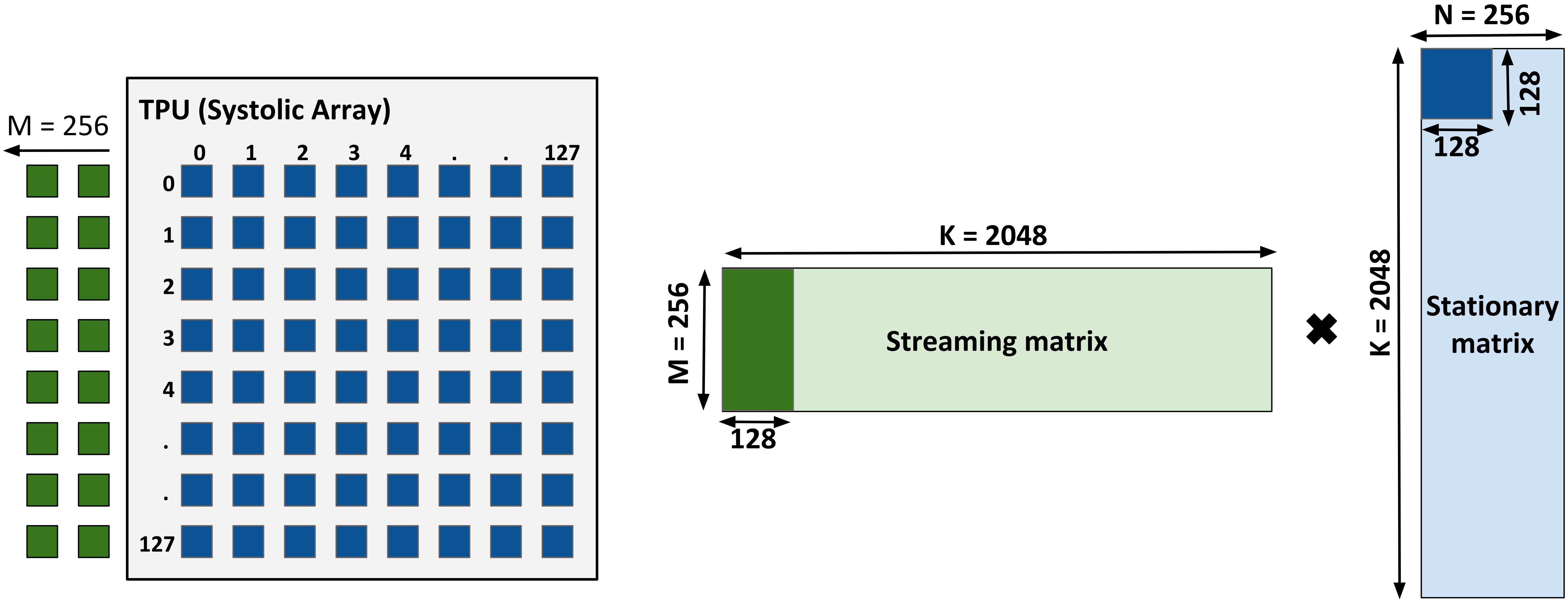
*** Assuming MK matrix is streaming and KN matrix is stationary. (aka weight stationary)*

Mapping GEMMs onto TPUs



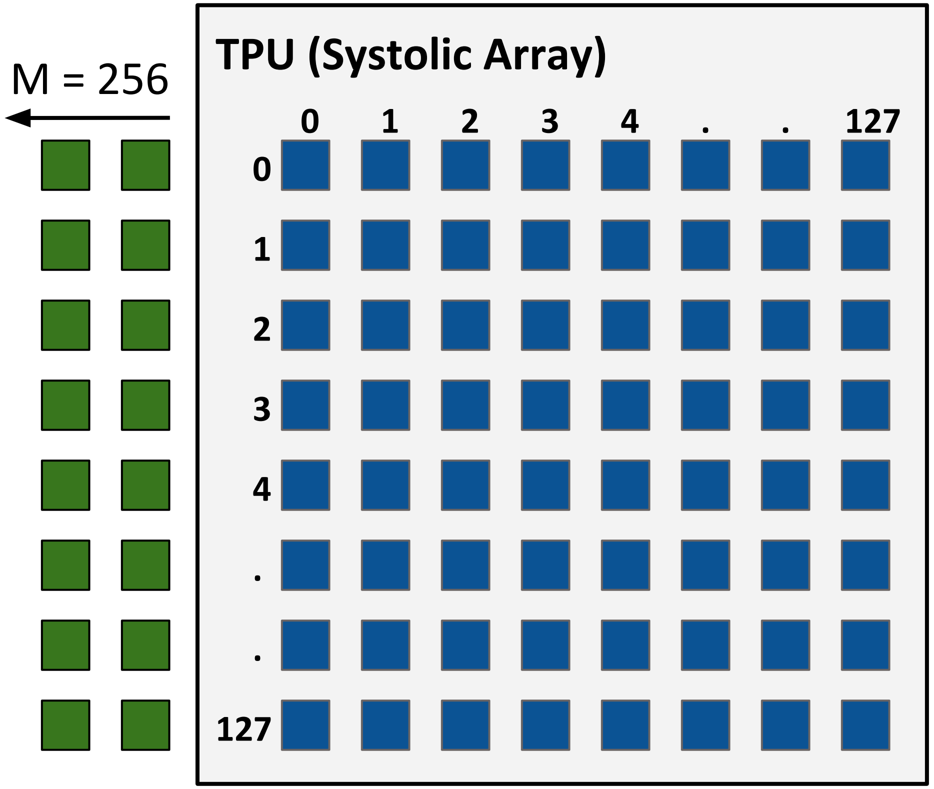
*** Assuming MK matrix is streaming and KN matrix is stationary. (aka weight stationary)*

Mapping GEMMs onto TPUs

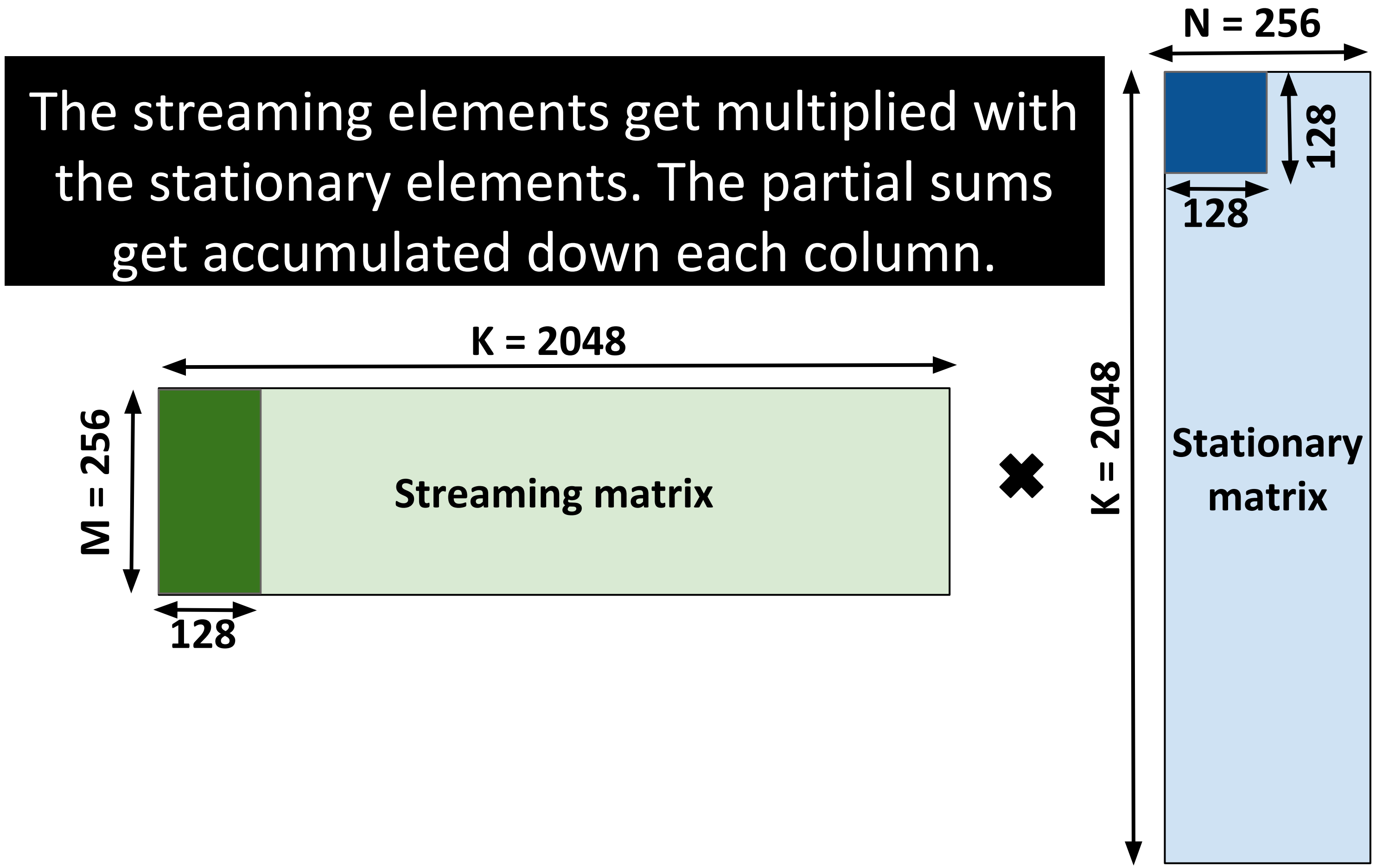


** *Assuming MK matrix is streaming and KN matrix is stationary. (aka weight stationary)*

Mapping GEMMs onto TPUs



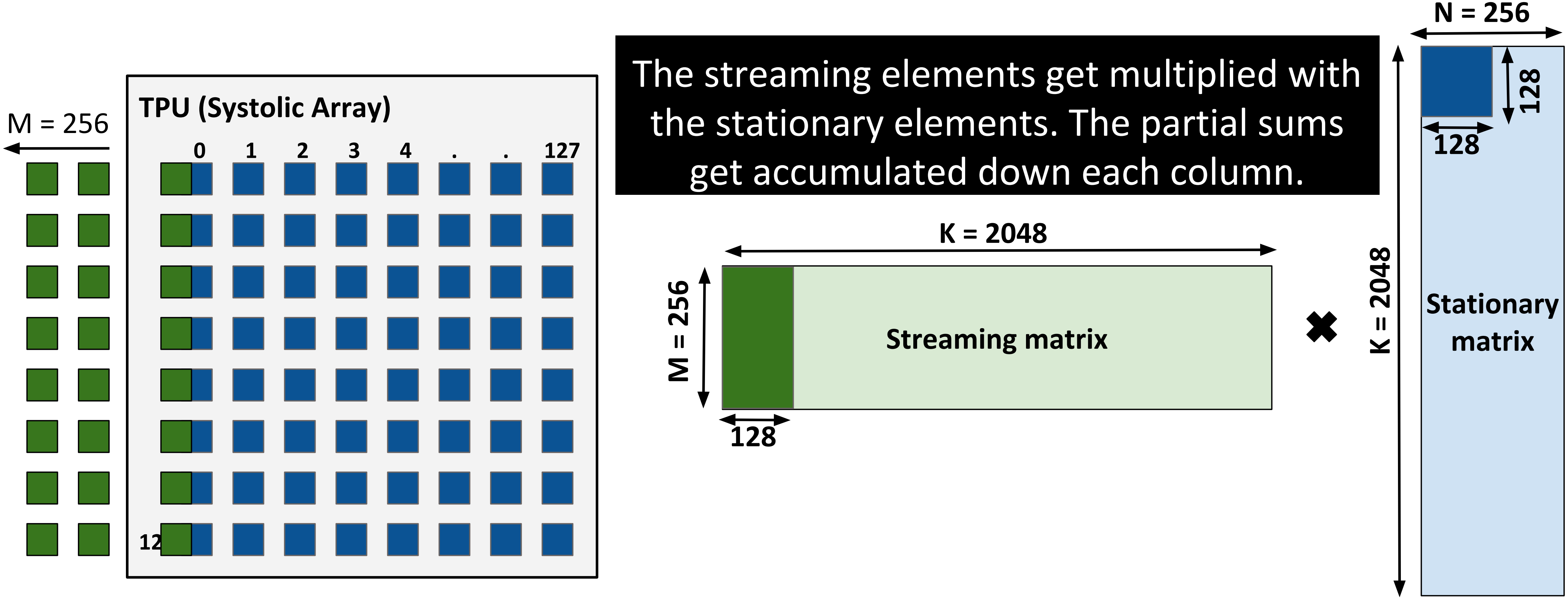
The streaming elements get multiplied with the stationary elements. The partial sums get accumulated down each column.



***** Assuming MK matrix is streaming and KN matrix is stationary. (aka weight stationary)***

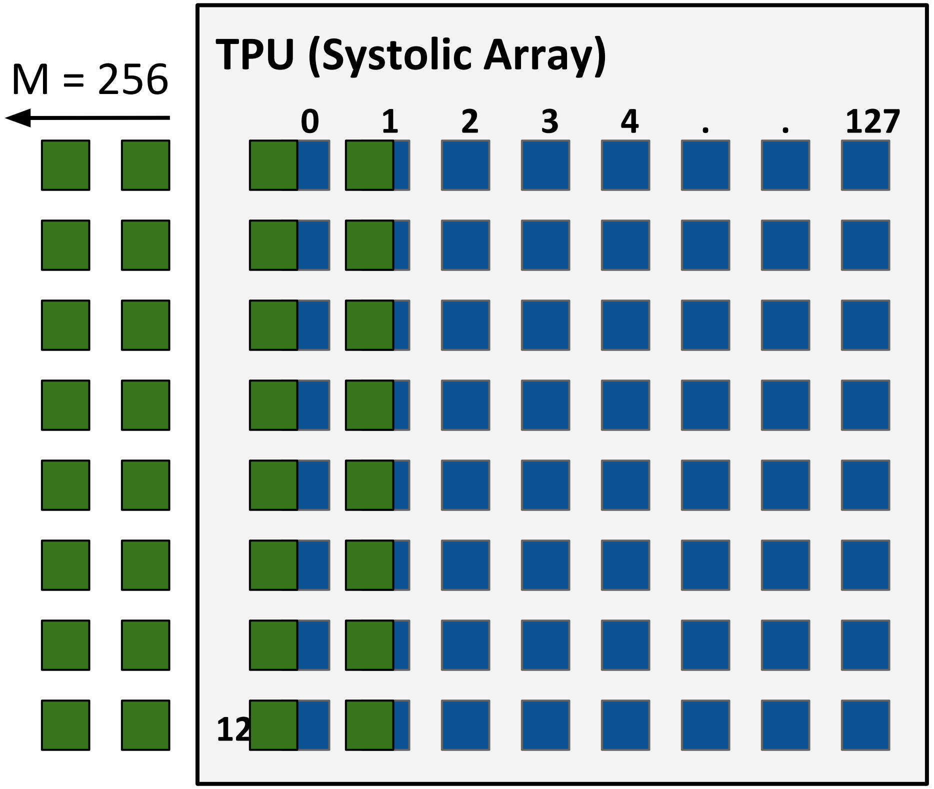
Mapping GEMMs onto TPUs

The streaming elements get multiplied with the stationary elements. The partial sums get accumulated down each column.

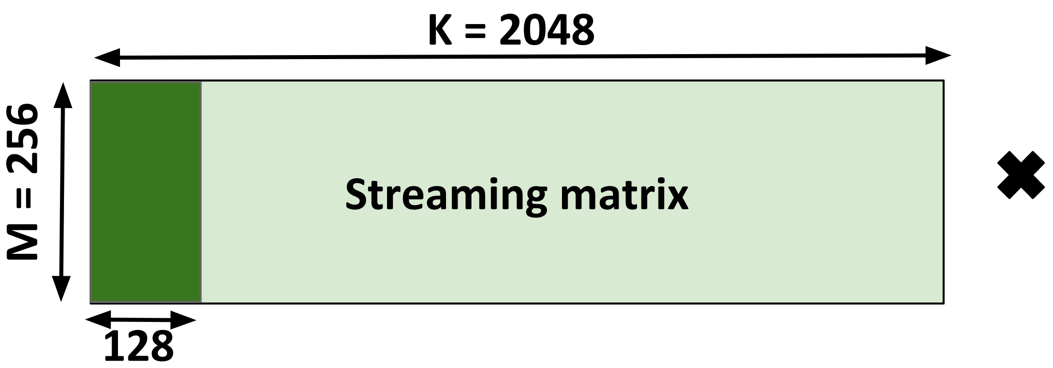


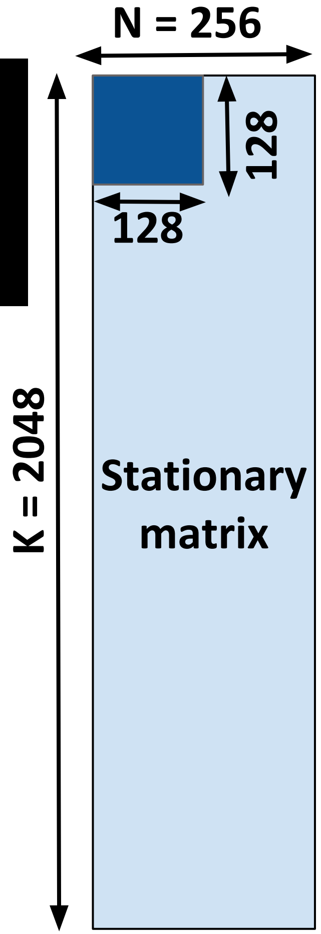
** *Assuming MK matrix is streaming and KN matrix is stationary. (aka weight stationary)*

Mapping GEMMs onto TPUs



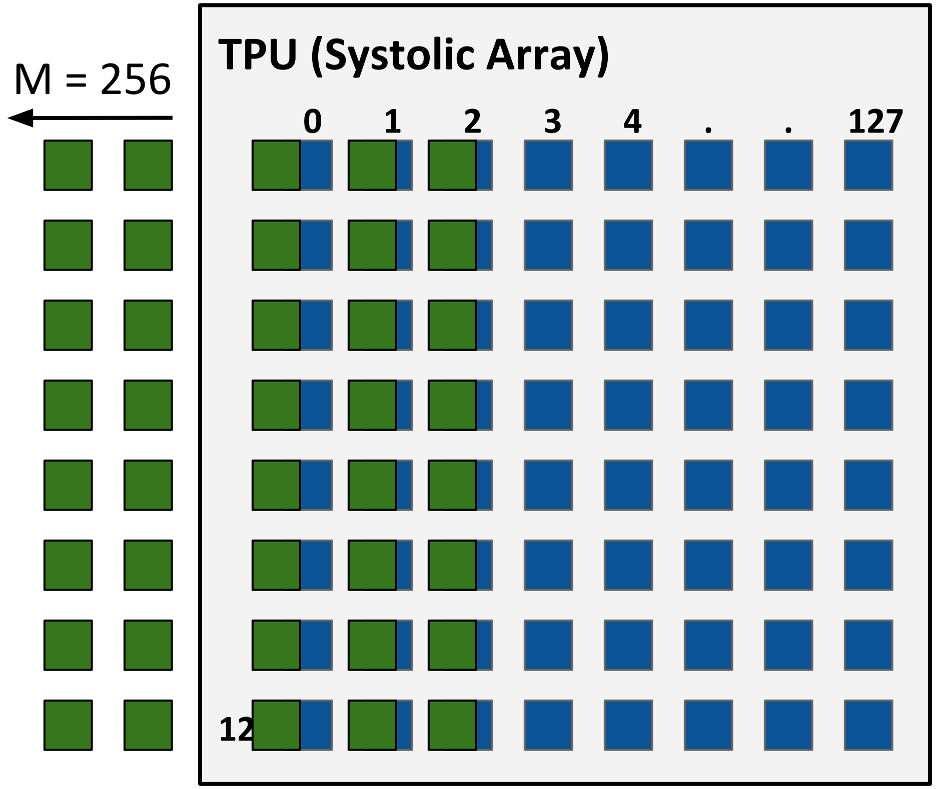
The streaming elements get multiplied with the stationary elements. The partial sums get accumulated down each column.



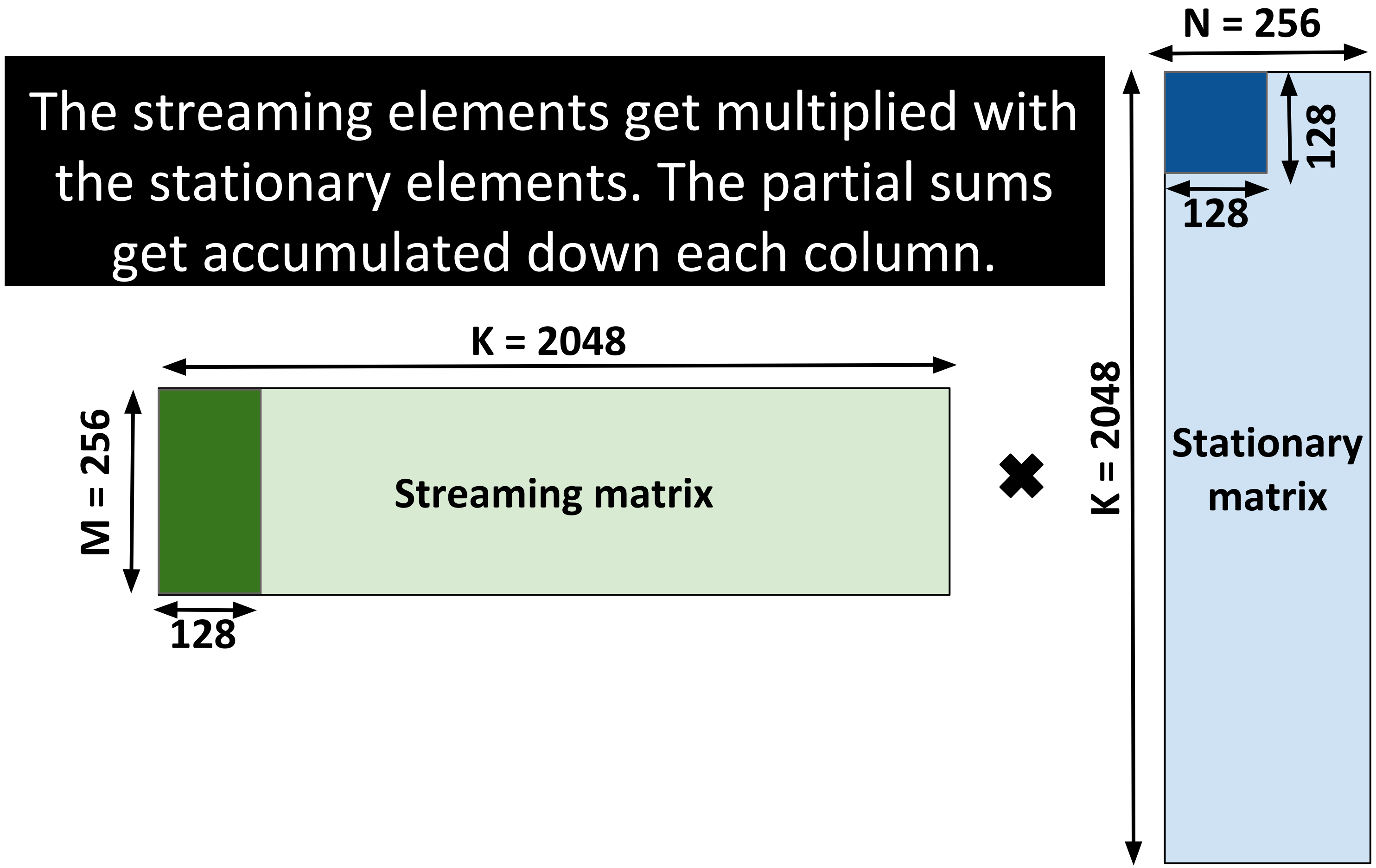


*** Assuming MK matrix is streaming and KN matrix is stationary. (aka weight stationary)*

Mapping GEMMs onto TPUs

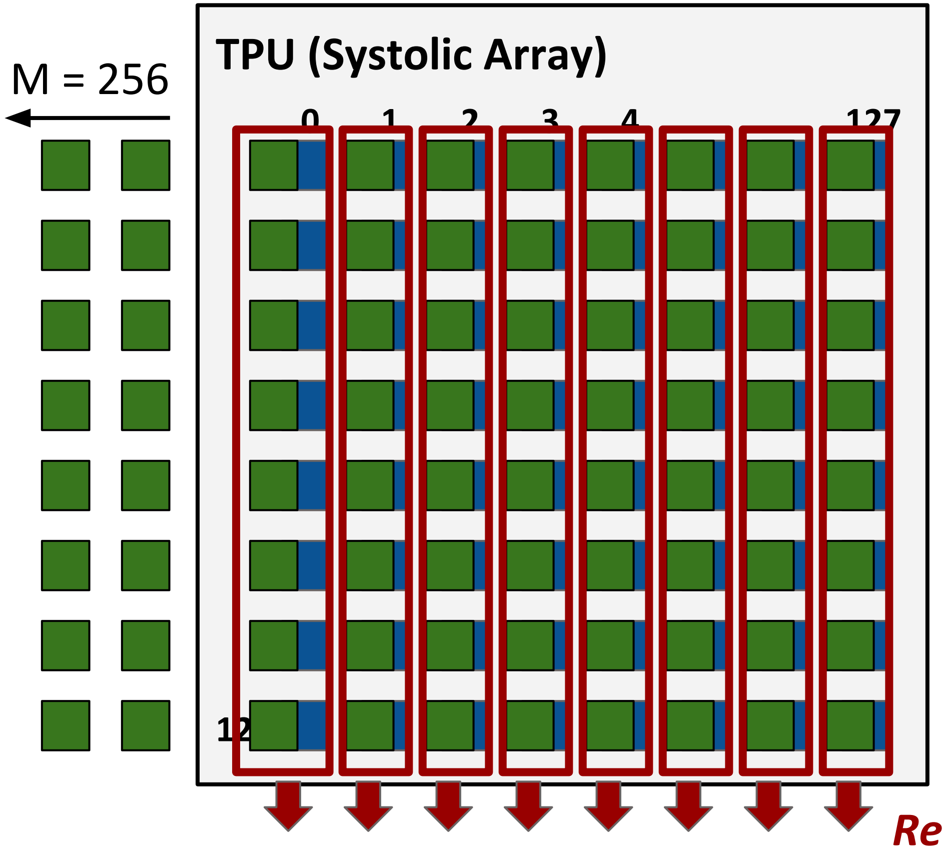


The streaming elements get multiplied with the stationary elements. The partial sums get accumulated down each column.

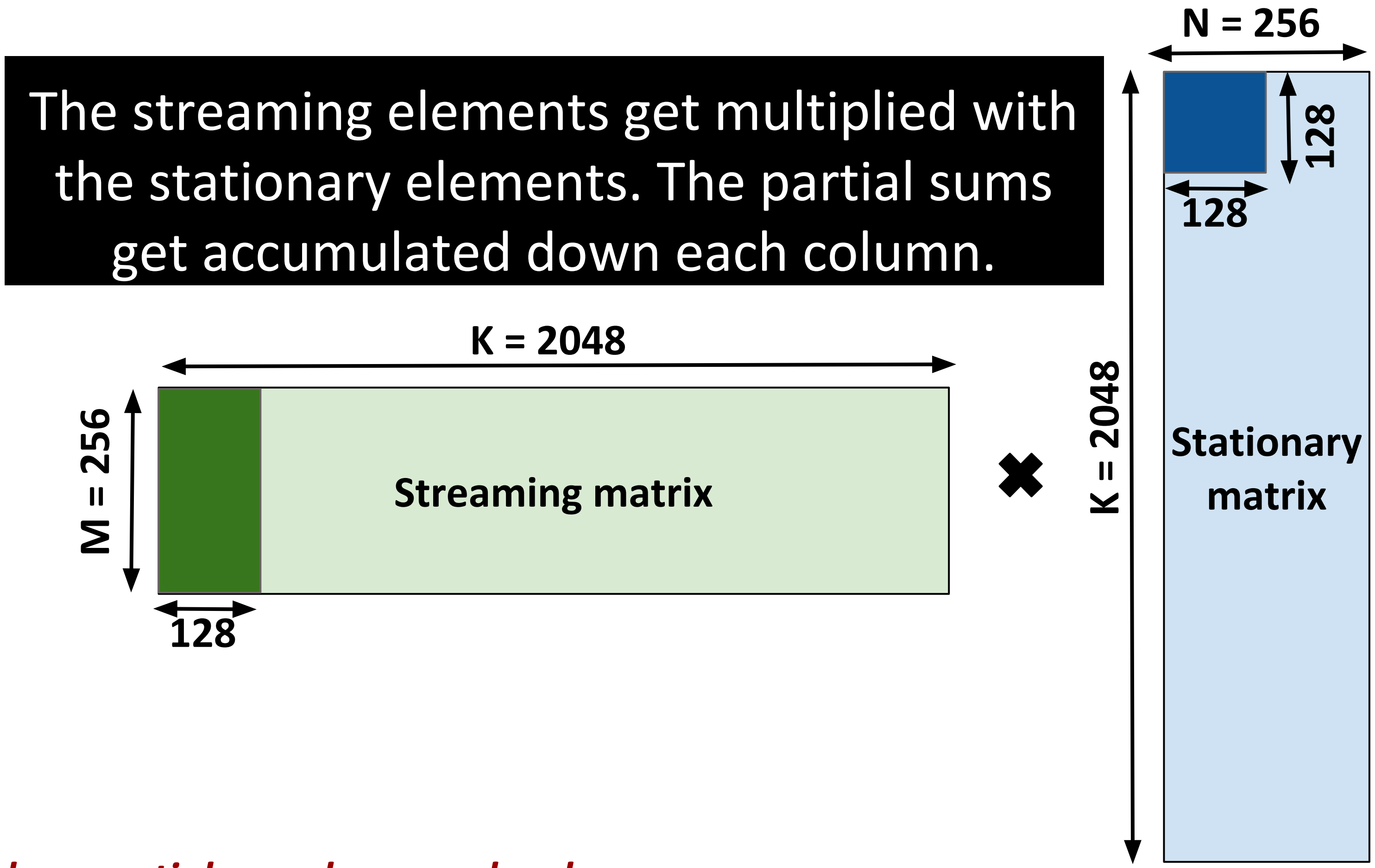


***** Assuming MK matrix is streaming and KN matrix is stationary. (aka weight stationary)***

Mapping GEMMs onto TPUs



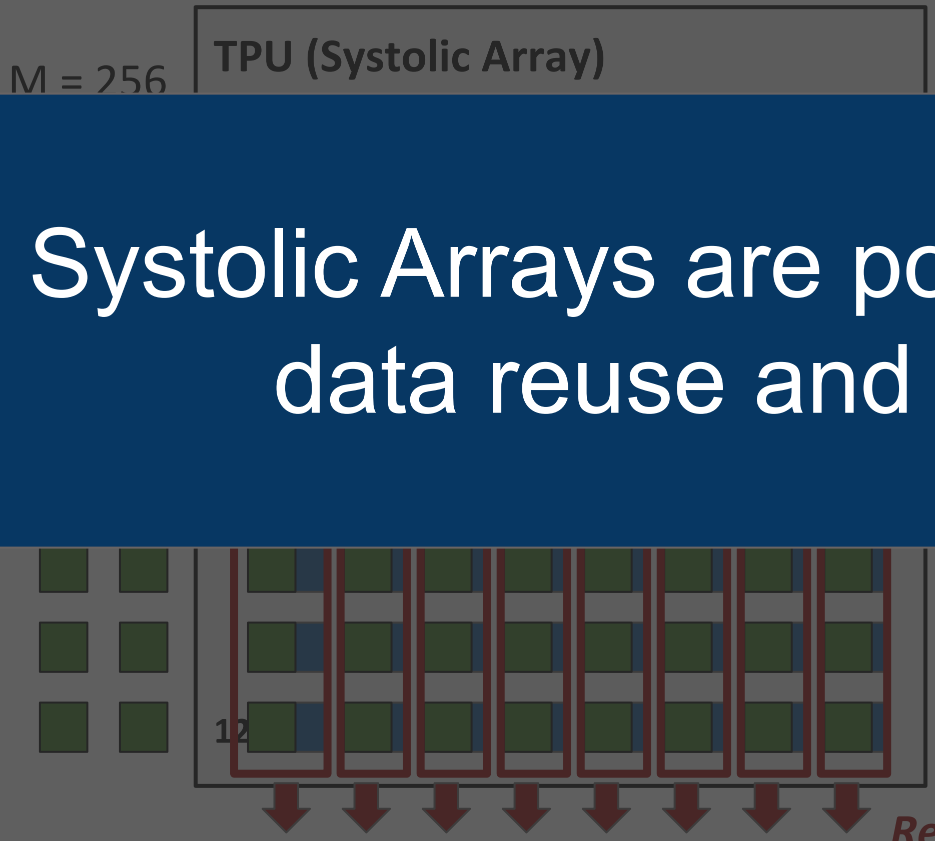
The streaming elements get multiplied with the stationary elements. The partial sums get accumulated down each column.



Reduce partial sum down each column.

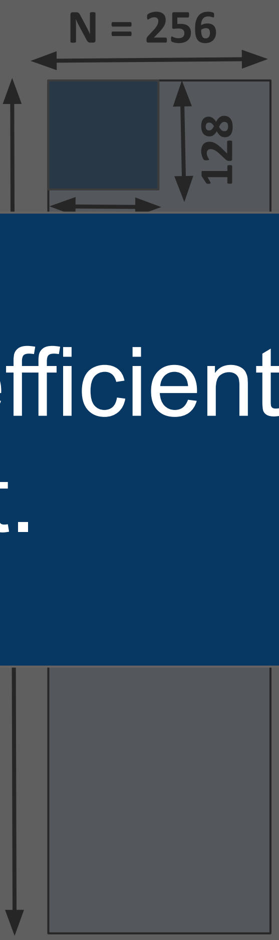
*** Assuming MK matrix is streaming and KN matrix is stationary. (aka weight stationary)*

Mapping GEMMs onto TPUs



The streaming elements get multiplied with the stationary elements. The partial sums

Systolic Arrays are popular because they enable efficient data reuse and are very simple to implement.

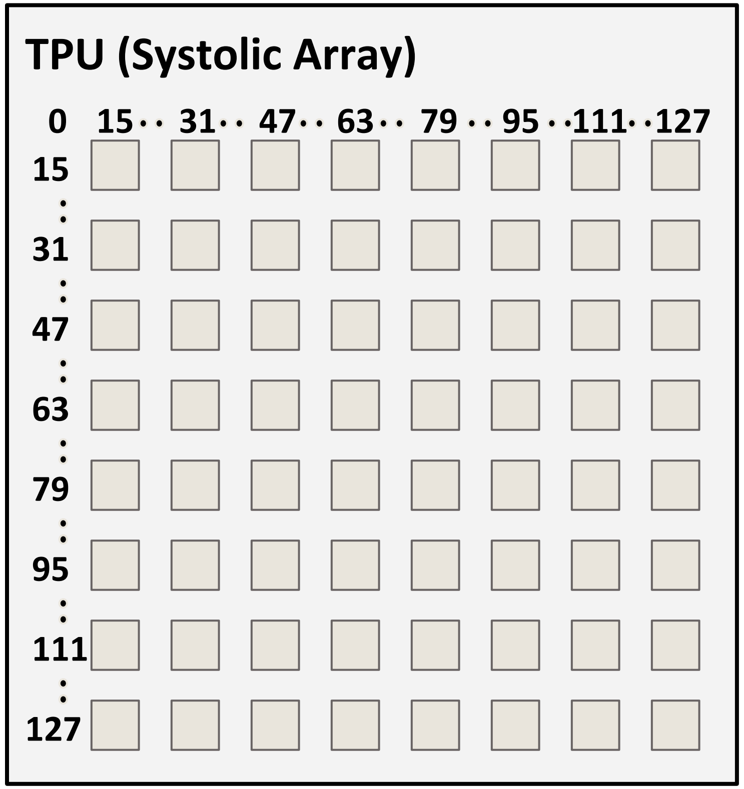


128

Reduce partial sum down each column.

*** Assuming MK matrix is streaming and KN matrix is stationary. (aka weight stationary)*

Mapping GEMMs onto TPUs - Irregularity



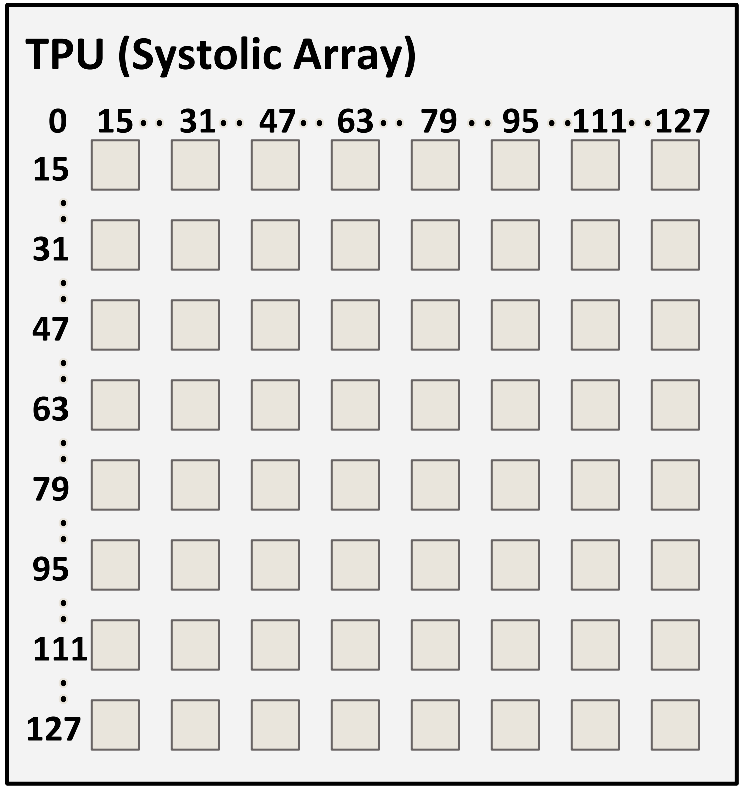
Workload	Application	Example Dimensions		
		M	N	K
GNMT	Machine Translation	128	2048	4096
		320	3072	4096
		1632	36548	1024
		2048	4096	32
DeepBench	General Workload	1024	16	500000
		35	8457	2560
Transformer	Language Understanding	31999	1024	84
		84	1024	4096
NCF	Collaborative Filtering	2048	1	128
		256	256	2048

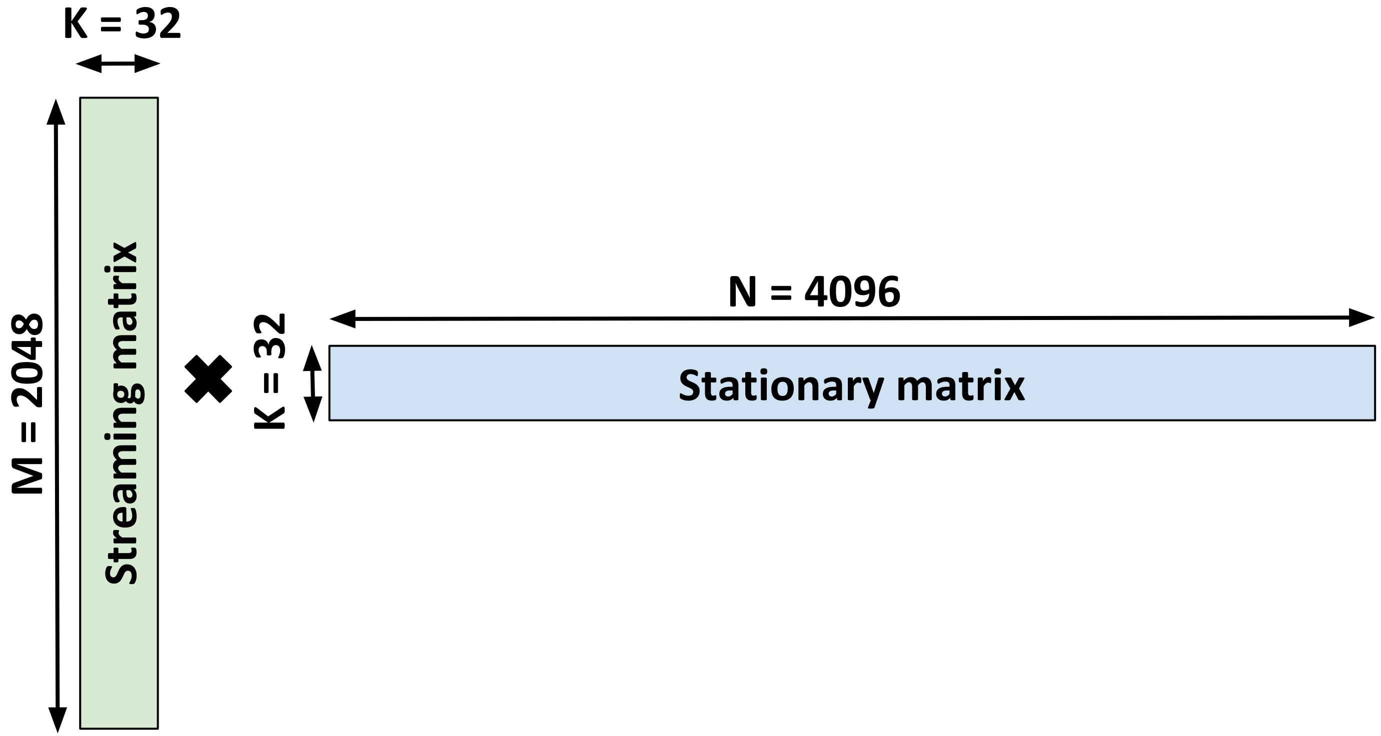
GEMMs used for evaluation.

Let's map another GEMM!

***** Assuming MK matrix is streaming and KN matrix is stationary. (aka weight stationary)***

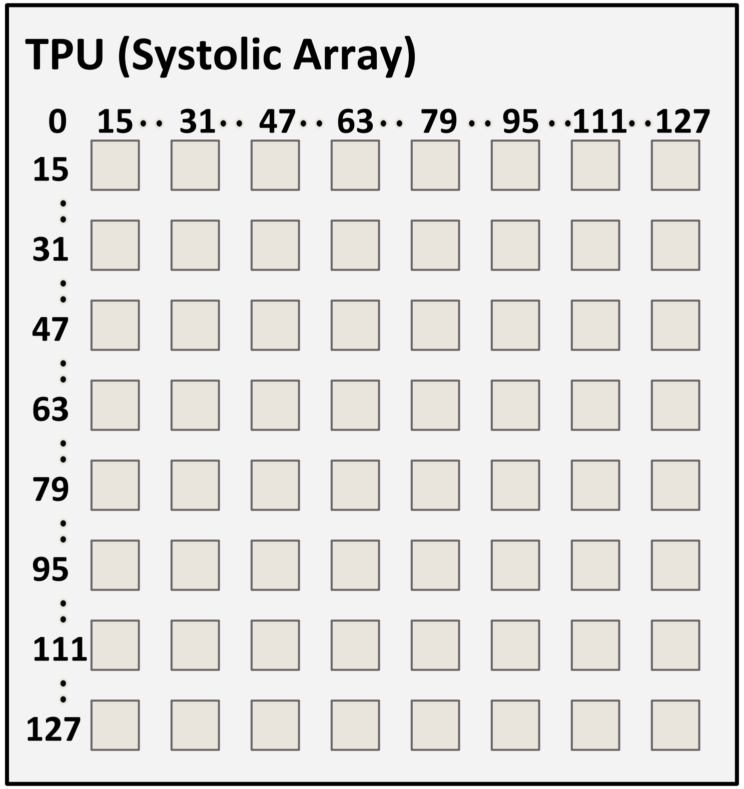
Mapping GEMMs onto TPUs - Irregularity

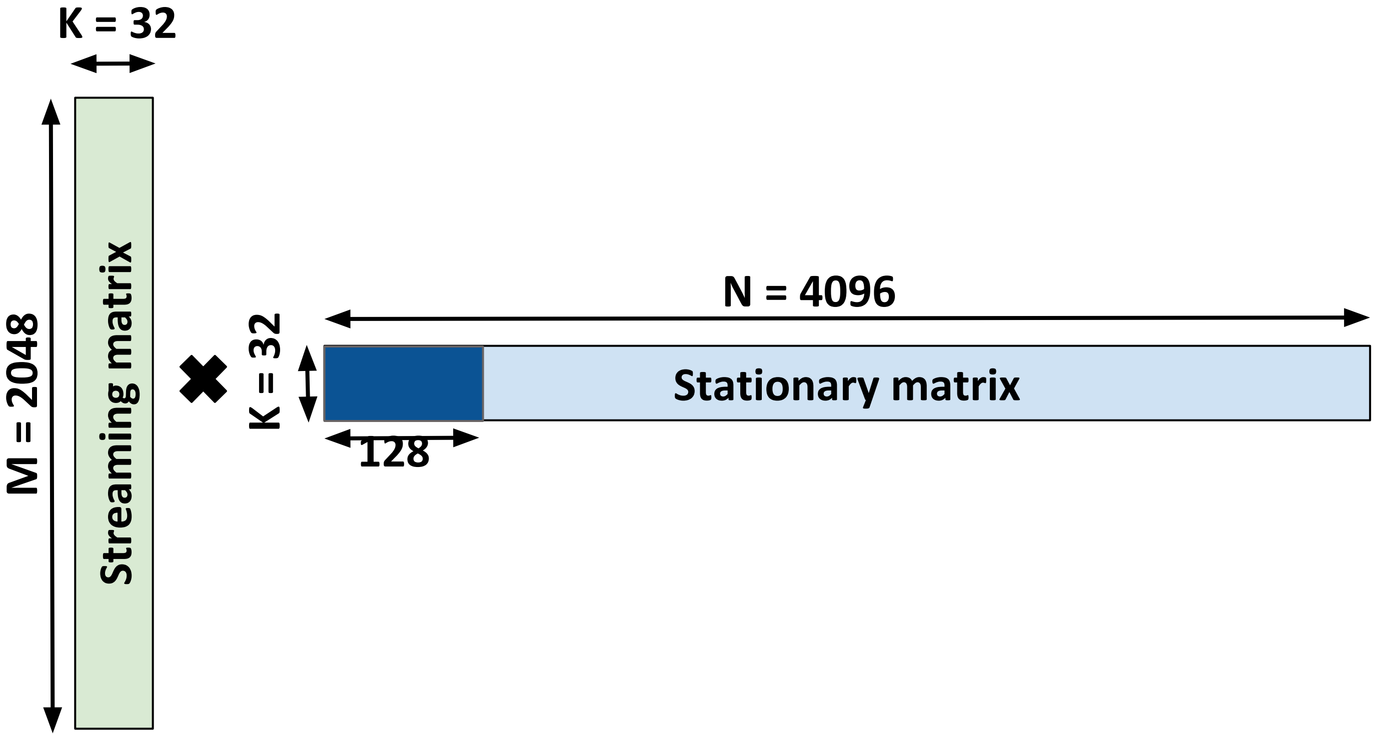




*** Assuming MK matrix is streaming and KN matrix is stationary. (aka weight stationary)*

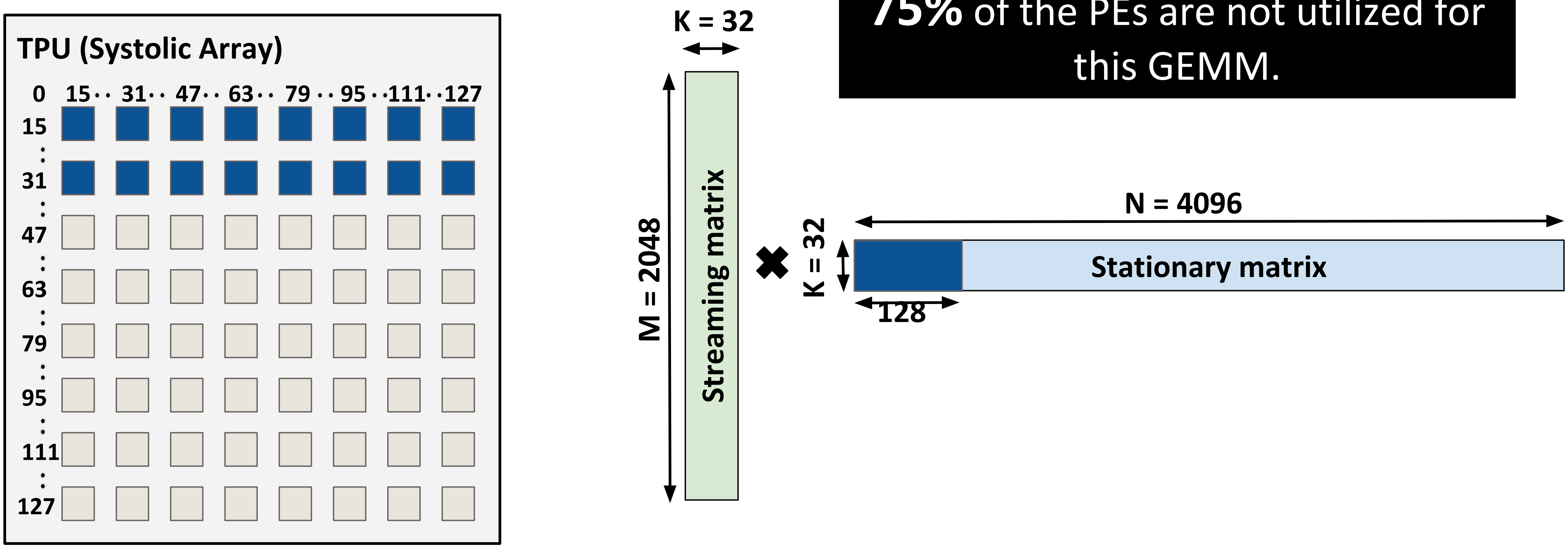
Mapping GEMMs onto TPUs - Irregularity





*** Assuming MK matrix is streaming and KN matrix is stationary. (aka weight stationary)*

Mapping GEMMs onto TPUs - Irregularity

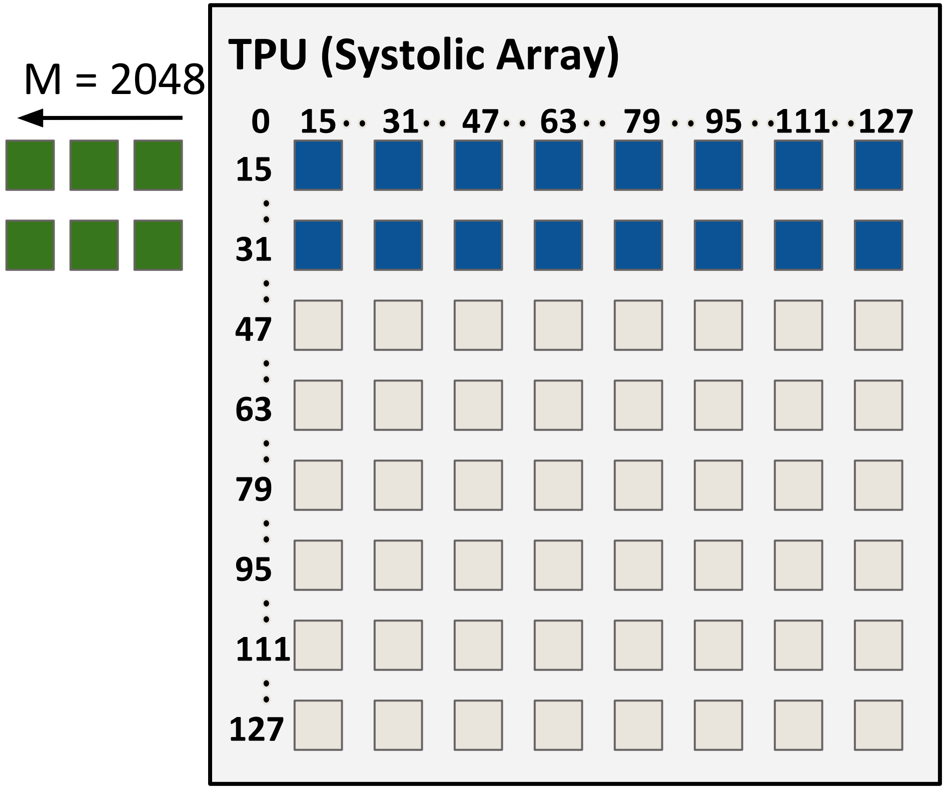


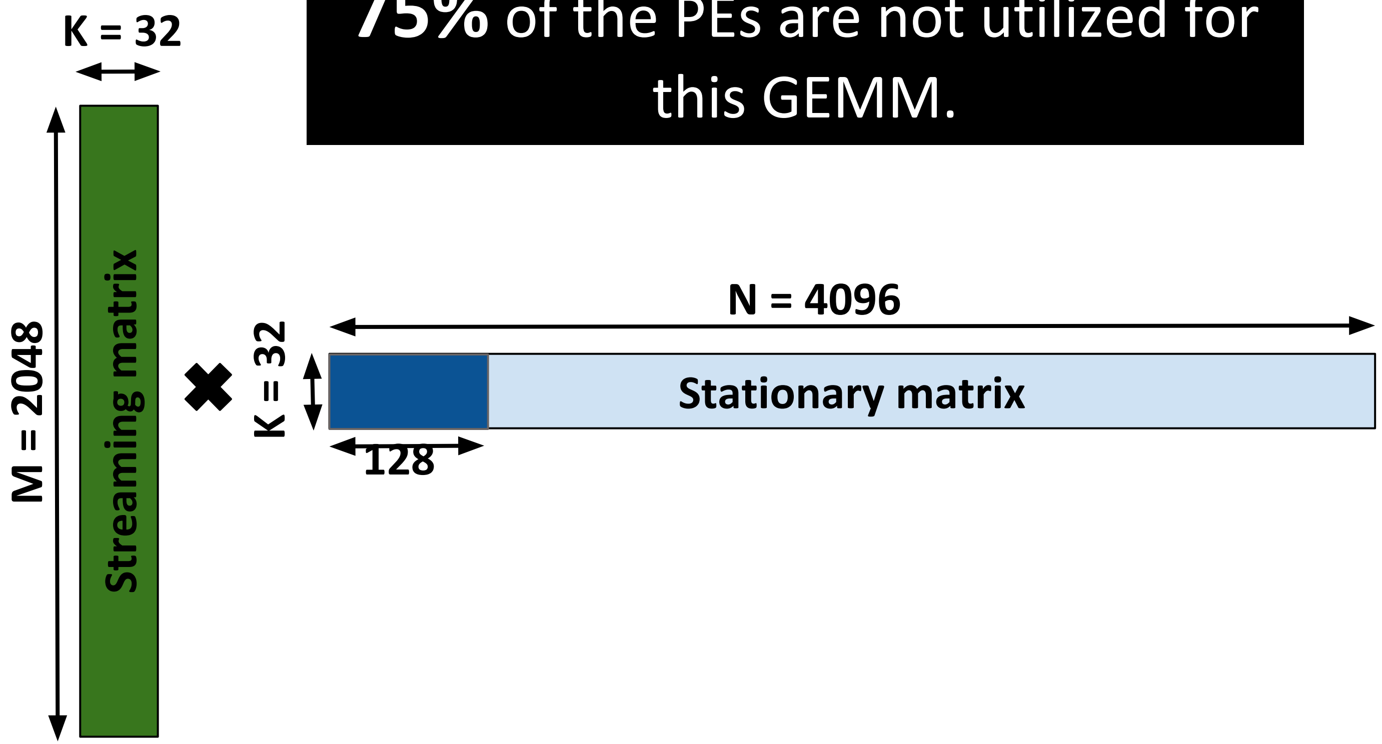
75% of the PEs are not utilized for this GEMM.

*** Assuming MK matrix is streaming and KN matrix is stationary. (aka weight stationary)*

Mapping GEMMs onto TPUs - Irregularity

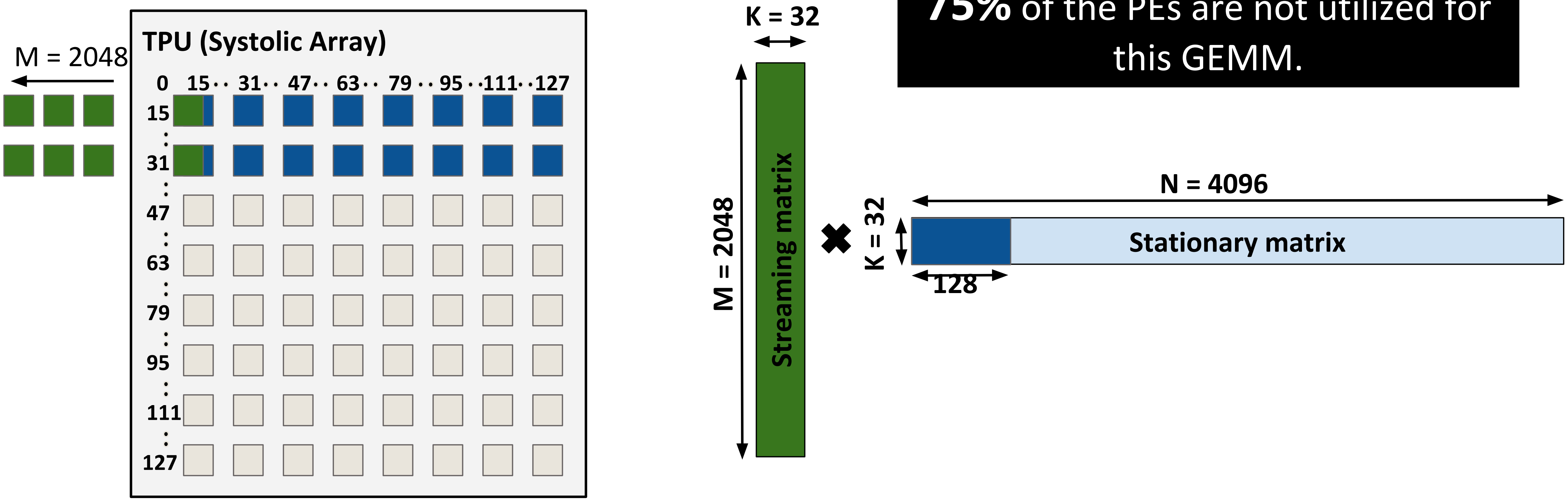
75% of the PEs are not utilized for this GEMM.





***** Assuming MK matrix is streaming and KN matrix is stationary. (aka weight stationary)***

Mapping GEMMs onto TPUs - Irregularity

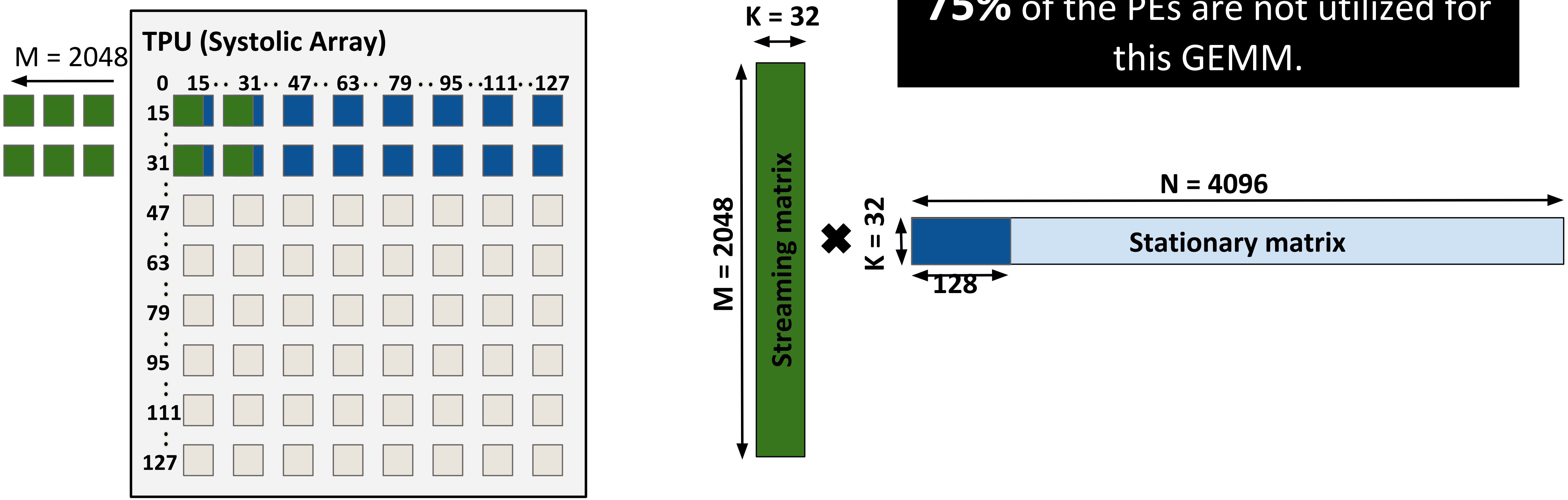


75% of the PEs are not utilized for this GEMM.

***** Assuming MK matrix is streaming and KN matrix is stationary. (aka weight stationary)***

Mapping GEMMs onto TPUs - Irregularity

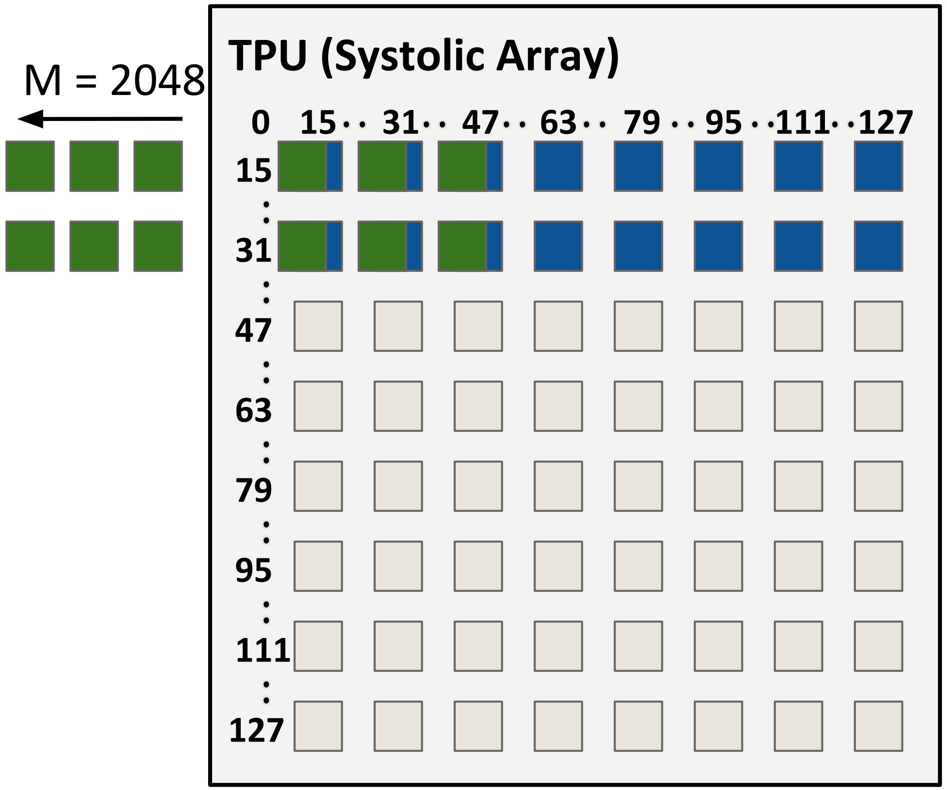
75% of the PEs are not utilized for this GEMM.

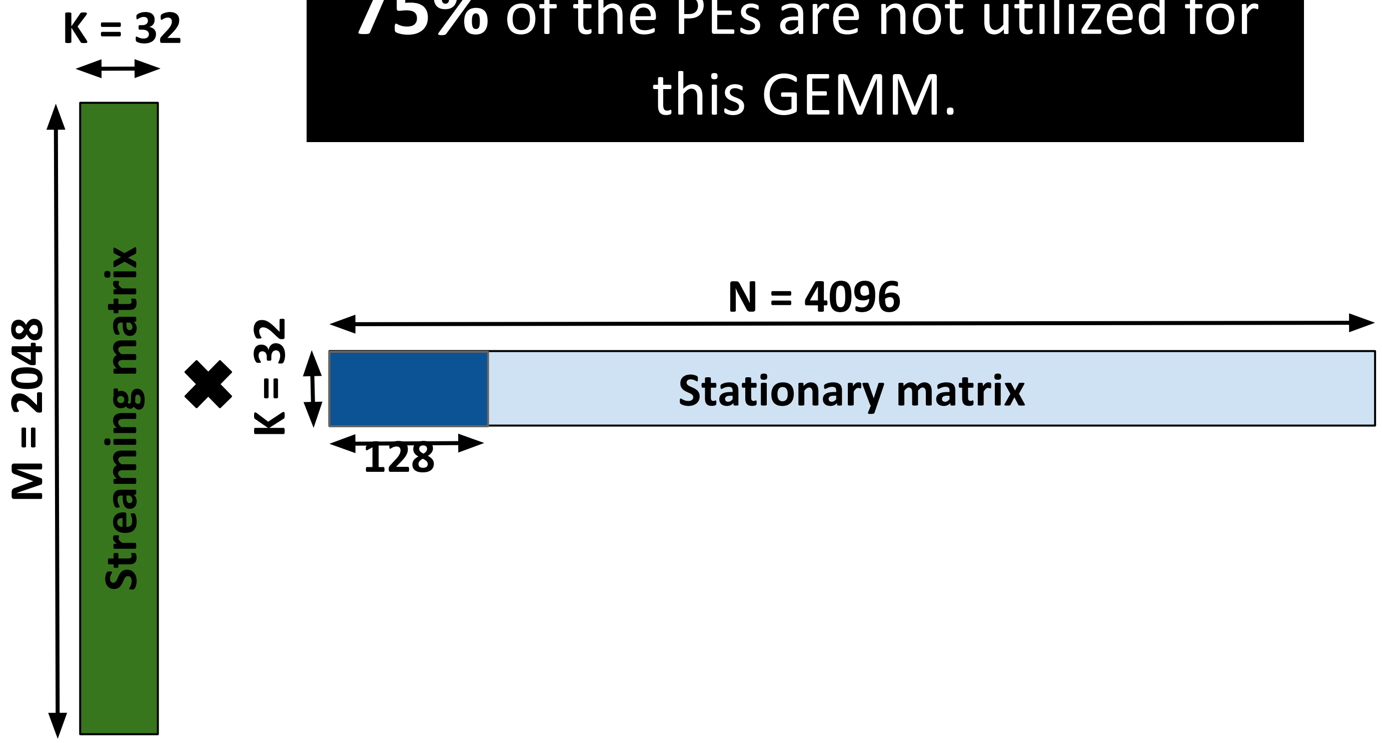


*** Assuming MK matrix is streaming and KN matrix is stationary. (aka weight stationary)*

Mapping GEMMs onto TPUs - Irregularity

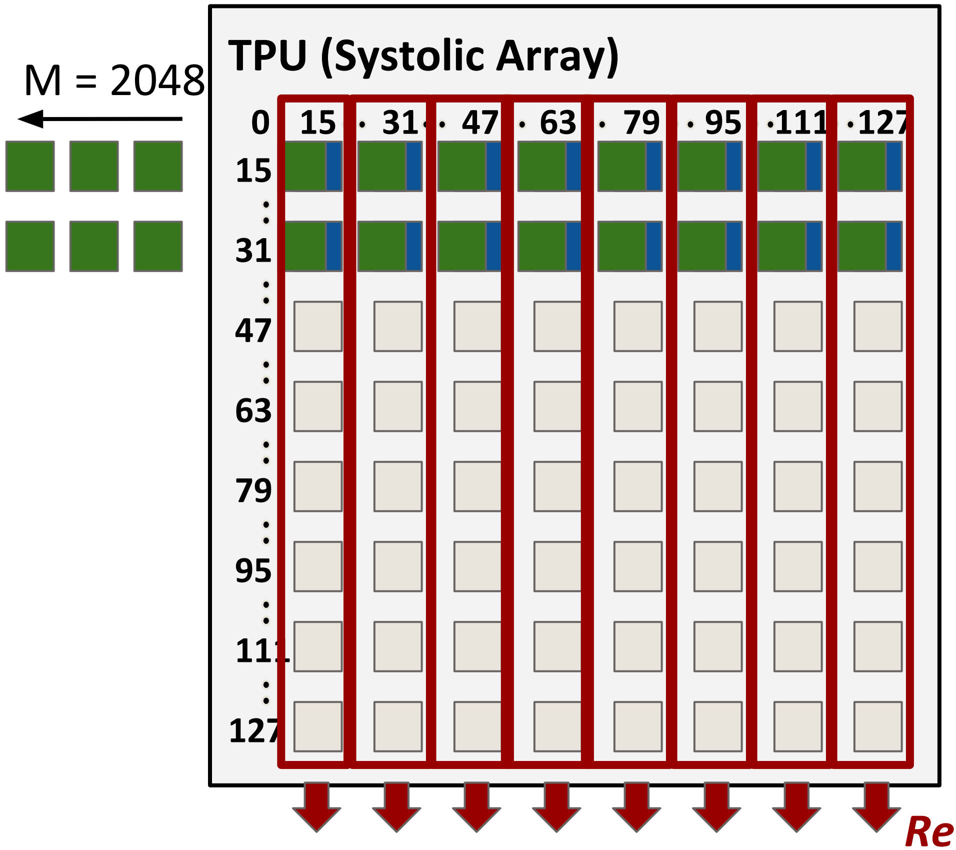
75% of the PEs are not utilized for this GEMM.



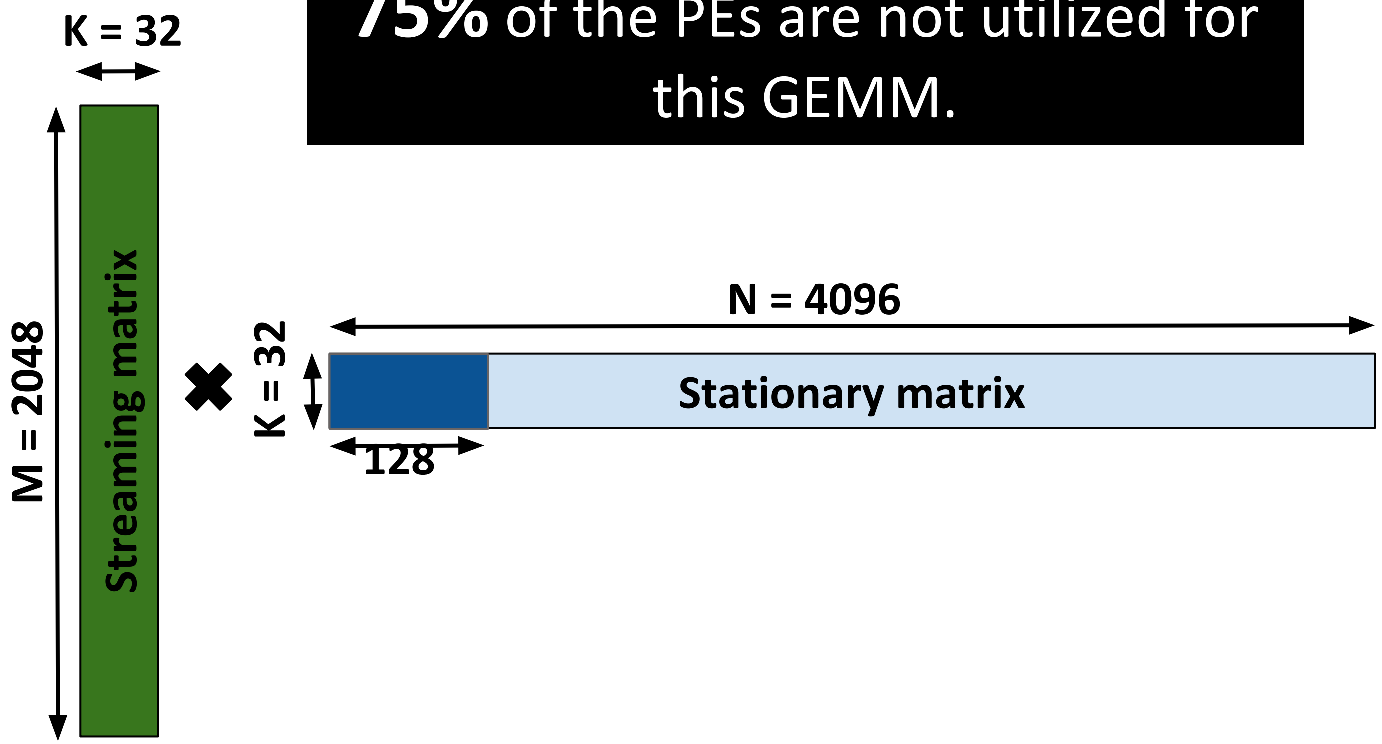


*** Assuming MK matrix is streaming and KN matrix is stationary. (aka weight stationary)*

Mapping GEMMs onto TPUs - Irregularity



75% of the PEs are not utilized for this GEMM.



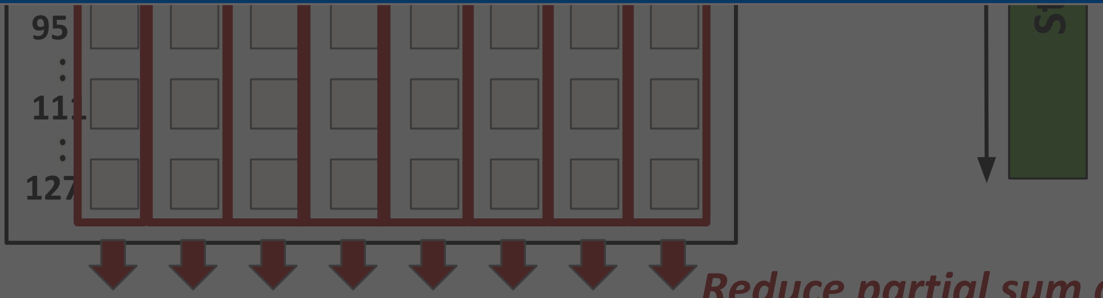
Reduce partial sum down each column.

***** Assuming MK matrix is streaming and KN matrix is stationary. (aka weight stationary)***

Mapping GEMMs onto TPUs - Irregularity

75% of the PEs are not utilized for this GEMM

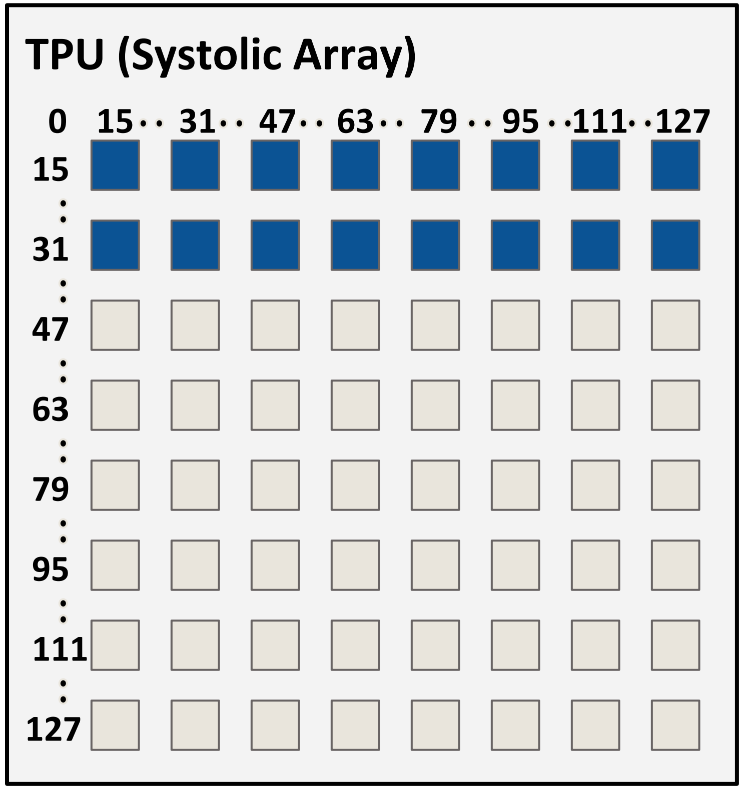
The rigid structure of Systolic Arrays cause PE underutilization. How can we enable the remaining PEs?

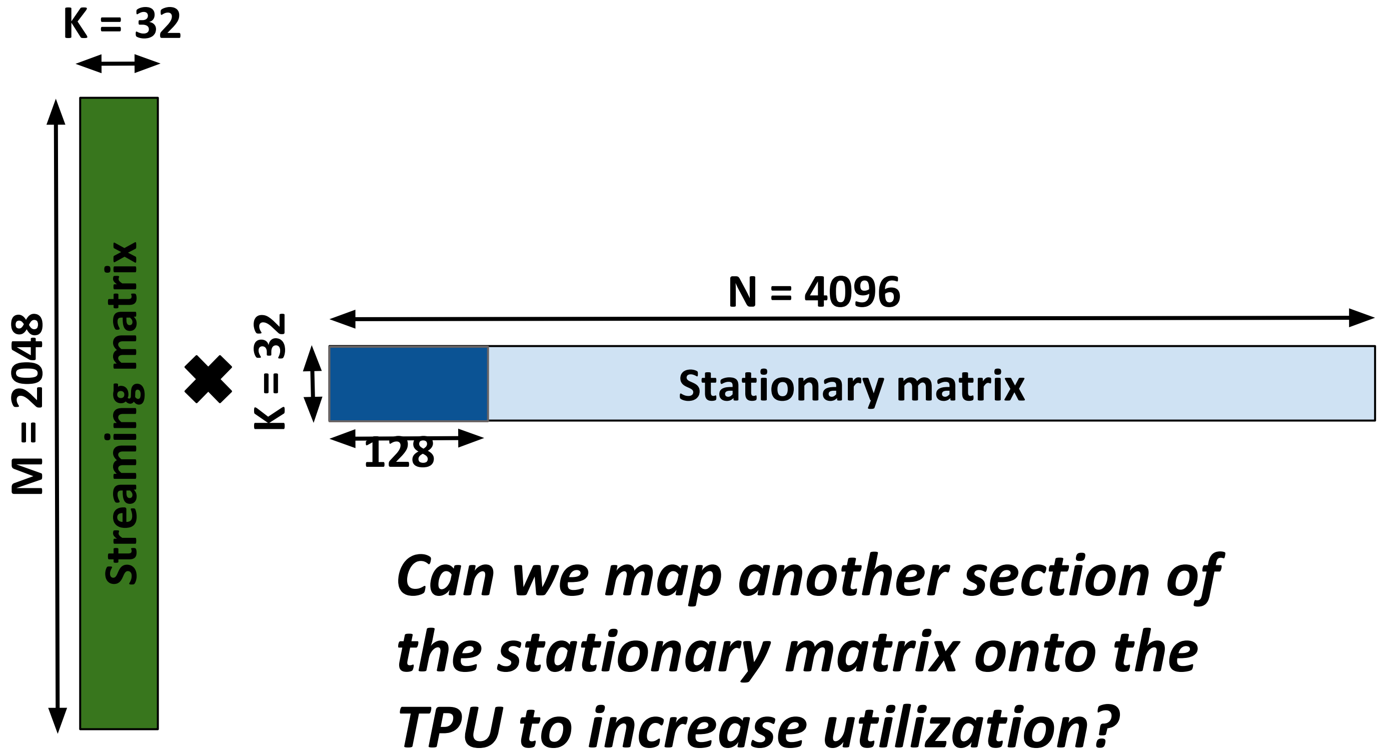


Reduce partial sum down each column.

*** Assuming MK matrix is streaming and KN matrix is stationary. (aka weight stationary)*

Mapping GEMMs onto TPUs - Irregularity

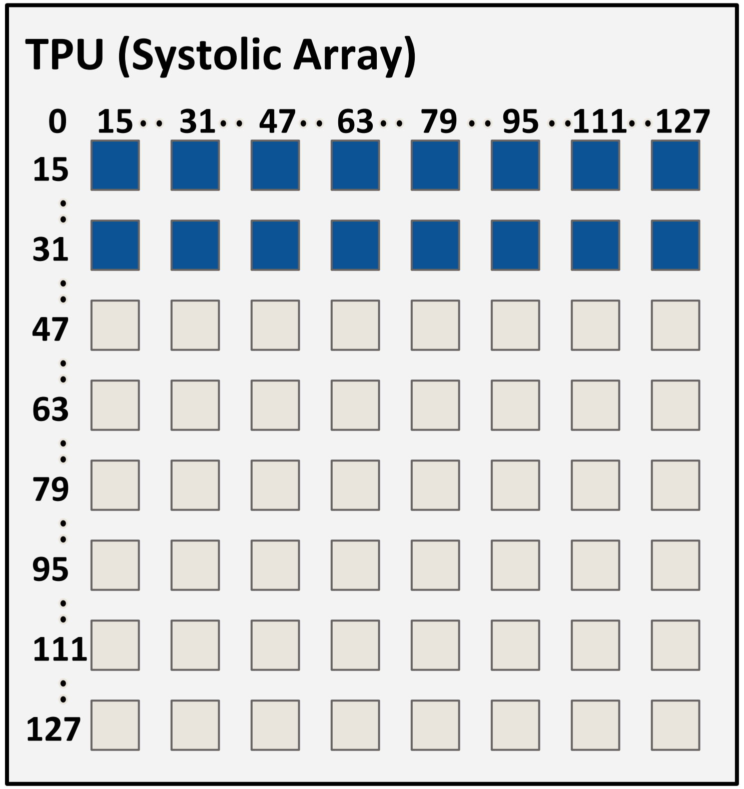


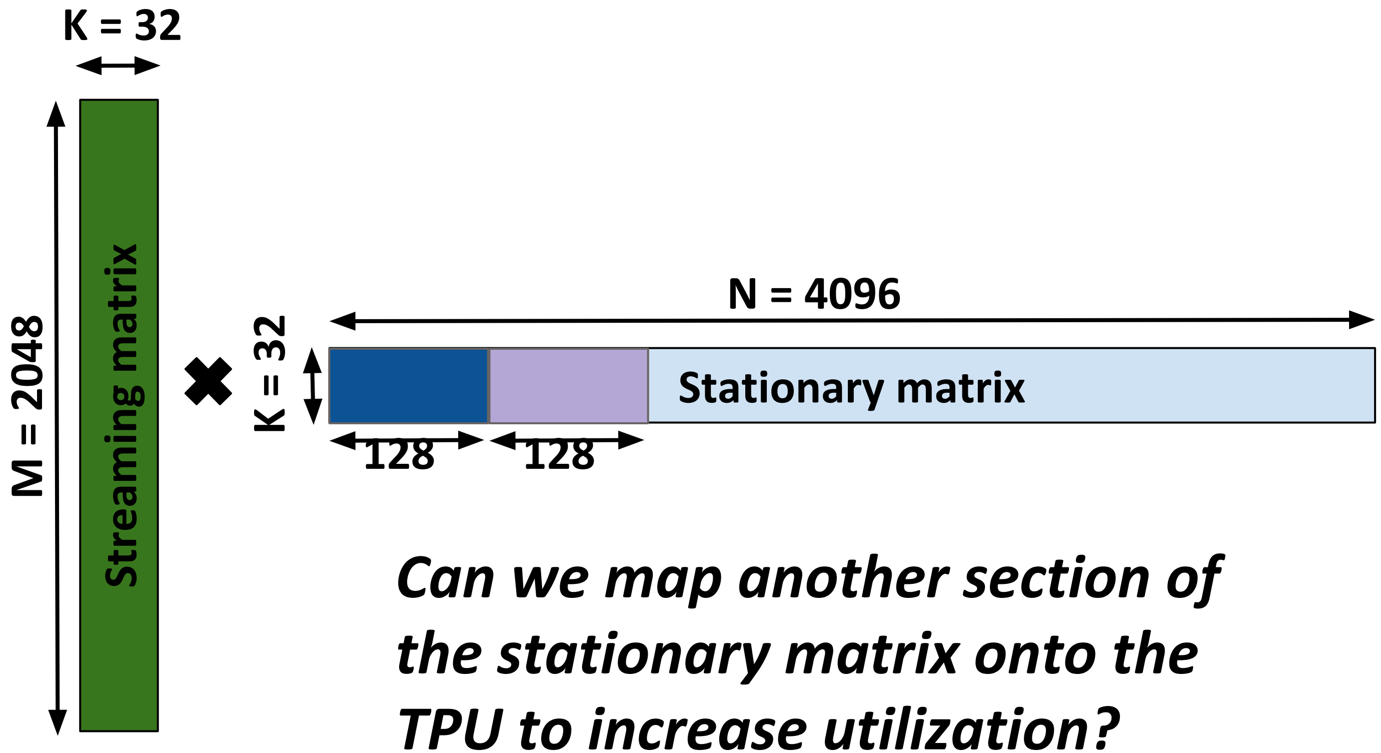


Can we map another section of the stationary matrix onto the TPU to increase utilization?

*** Assuming MK matrix is streaming and KN matrix is stationary. (aka weight stationary)*

Mapping GEMMs onto TPUs - Irregularity

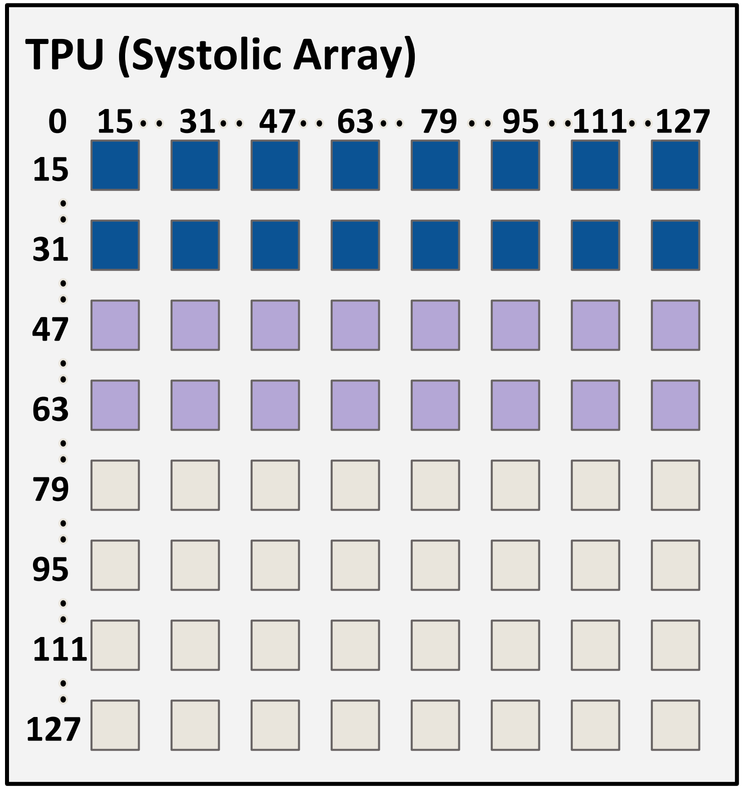


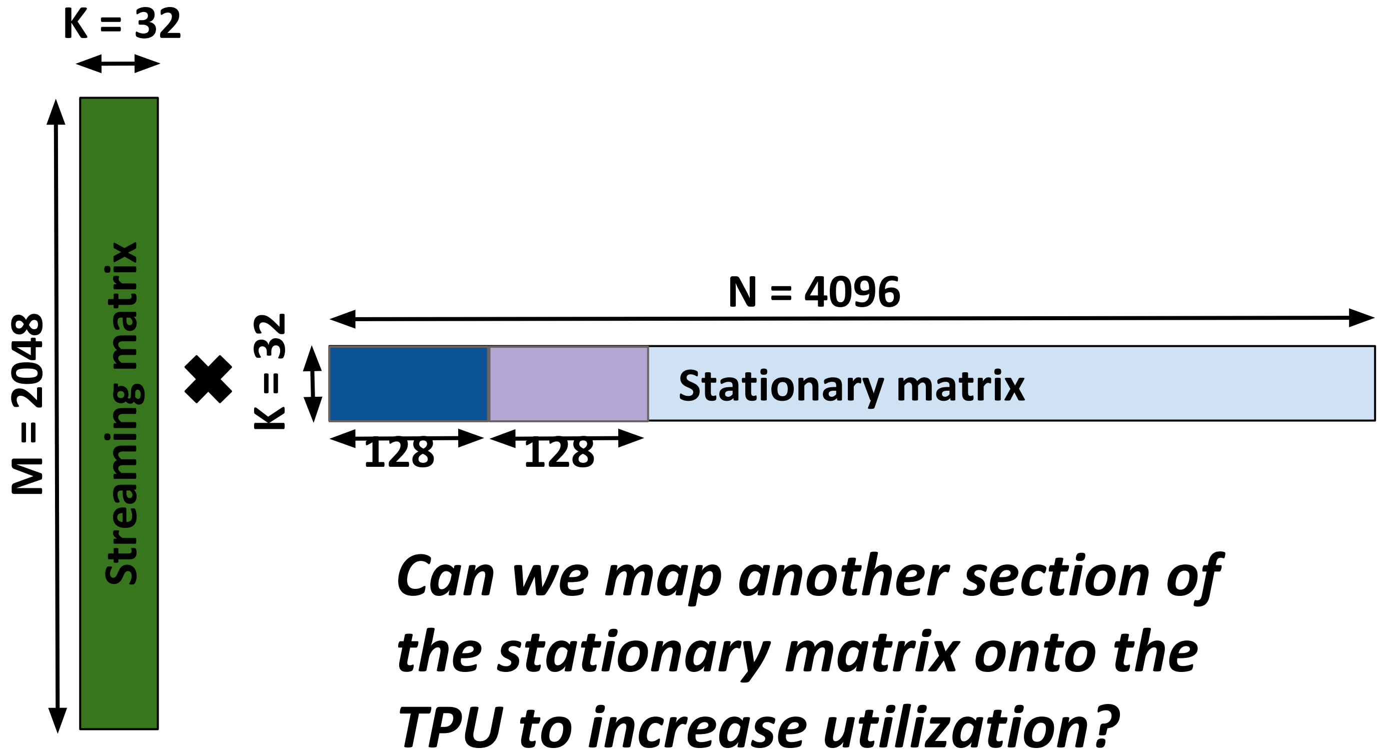


Can we map another section of the stationary matrix onto the TPU to increase utilization?

***** Assuming MK matrix is streaming and KN matrix is stationary. (aka weight stationary)***

Mapping GEMMs onto TPUs - Irregularity

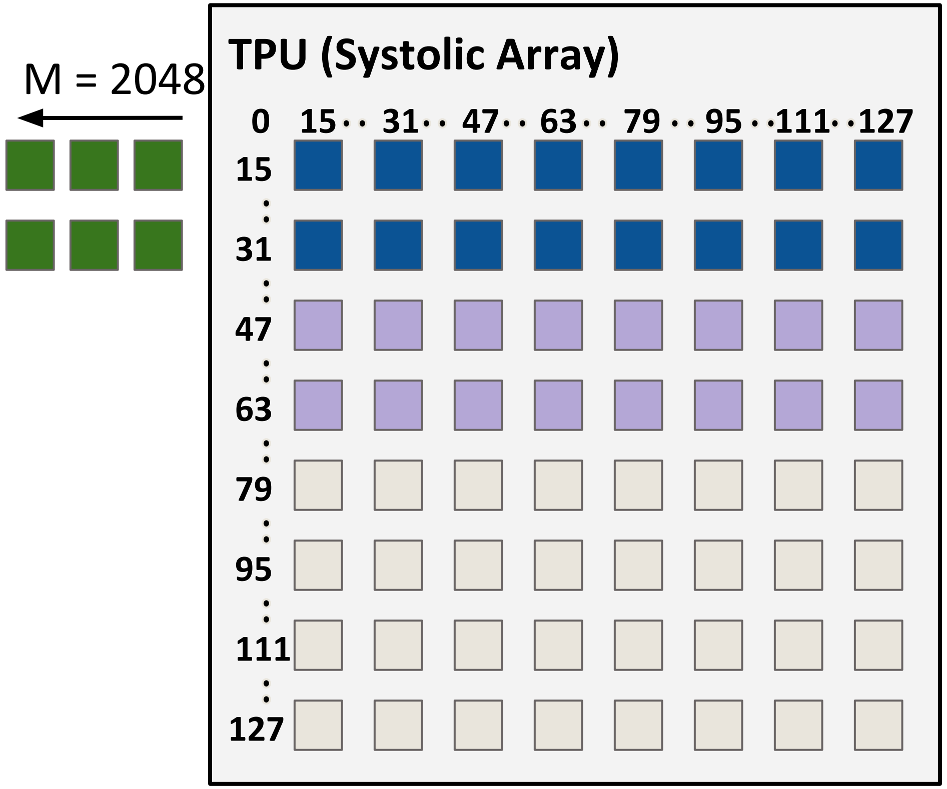




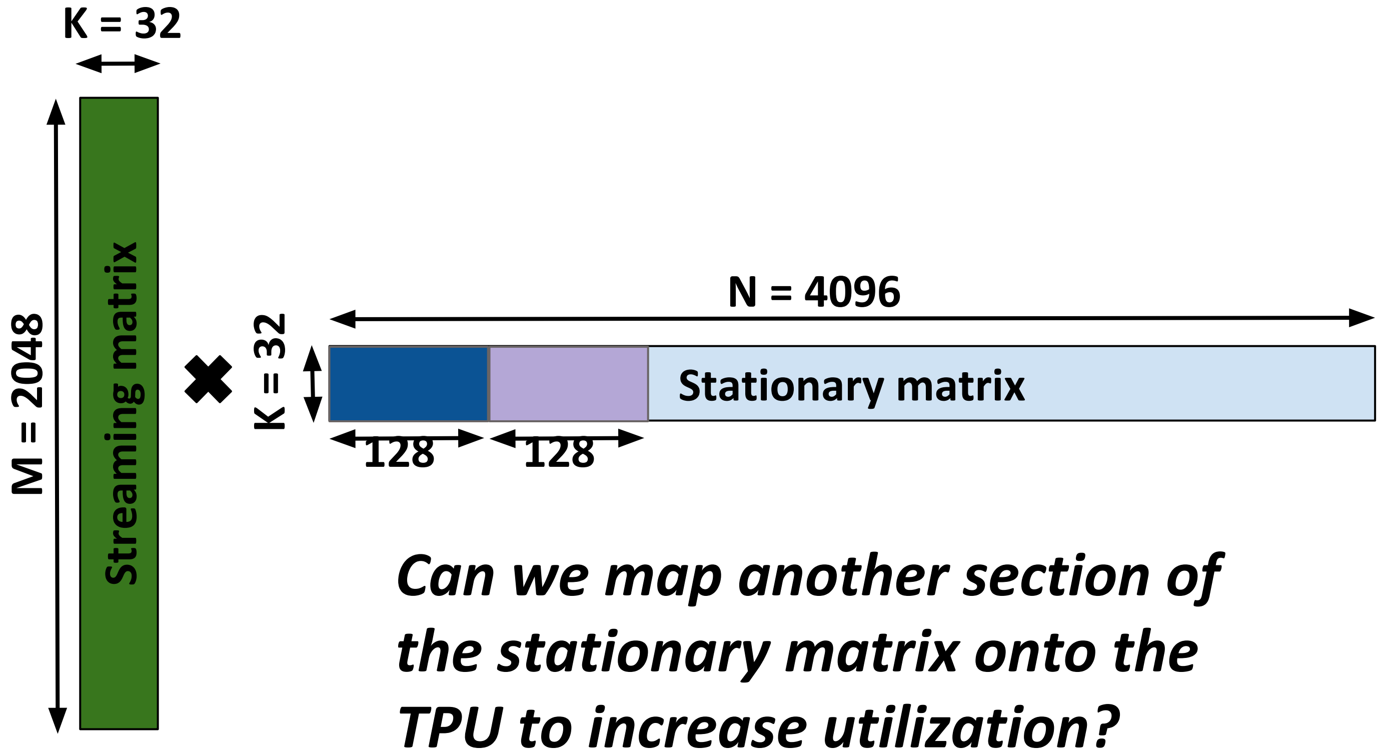
Can we map another section of the stationary matrix onto the TPU to increase utilization?

***** Assuming MK matrix is streaming and KN matrix is stationary. (aka weight stationary)***

Mapping GEMMs onto TPUs - Irregularity



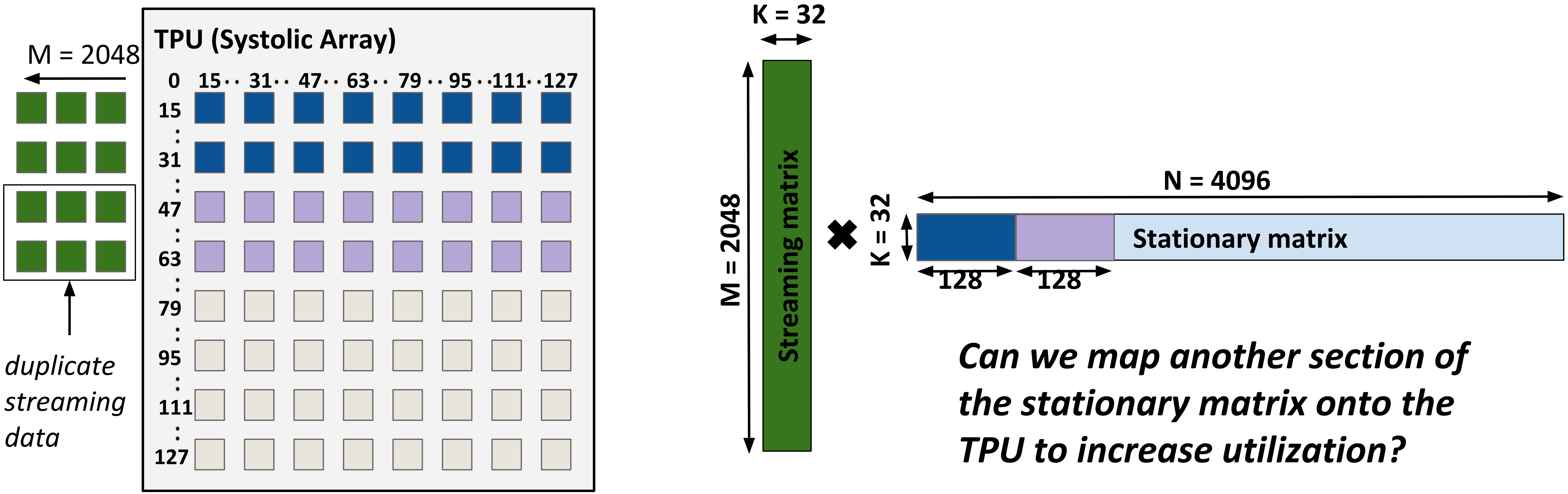
duplicate streaming data



Can we map another section of the stationary matrix onto the TPU to increase utilization?

***** Assuming MK matrix is streaming and KN matrix is stationary. (aka weight stationary)***

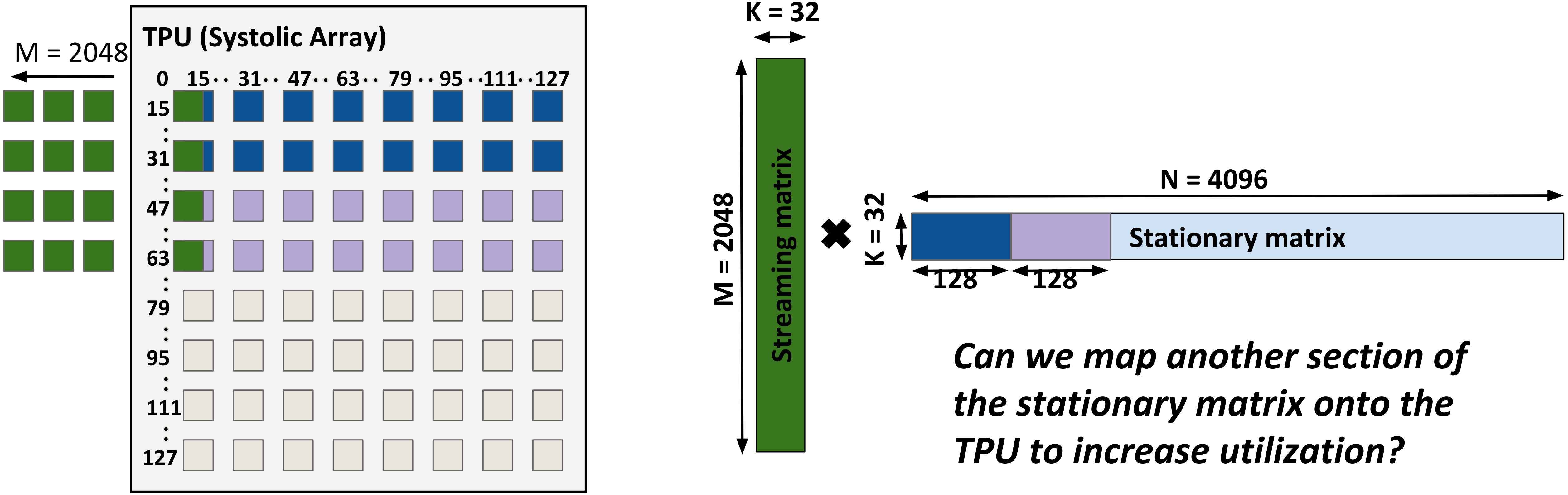
Mapping GEMMs onto TPUs - Irregularity



Can we map another section of the stationary matrix onto the TPU to increase utilization?

***** Assuming MK matrix is streaming and KN matrix is stationary. (aka weight stationary)***

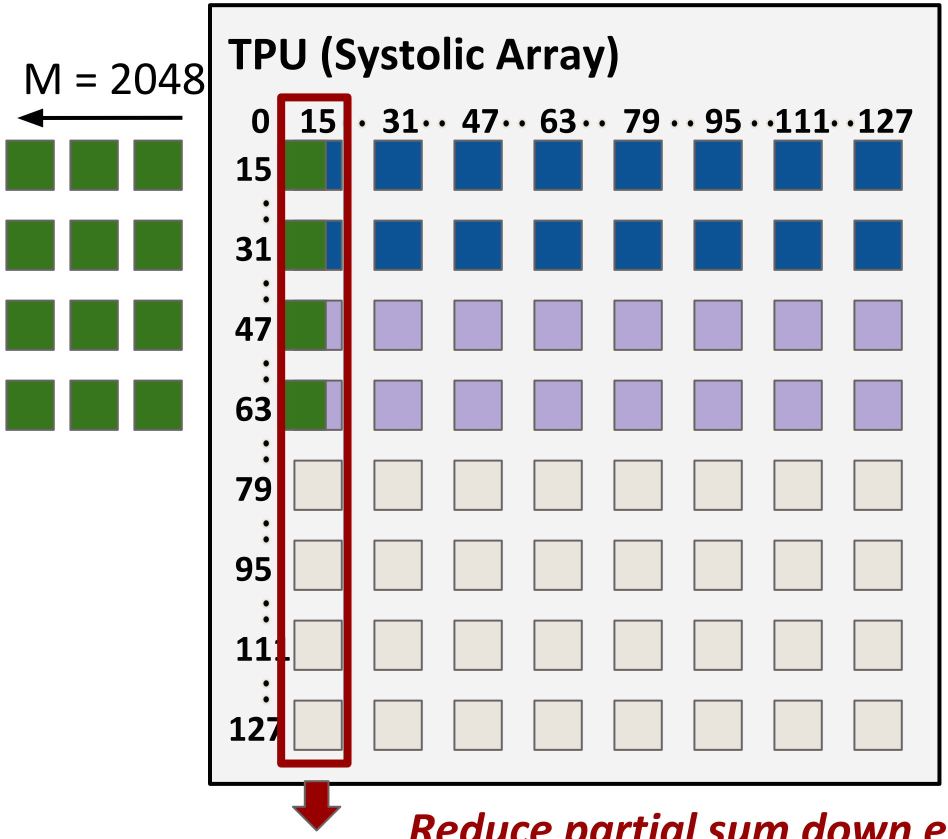
Mapping GEMMs onto TPUs - Irregularity

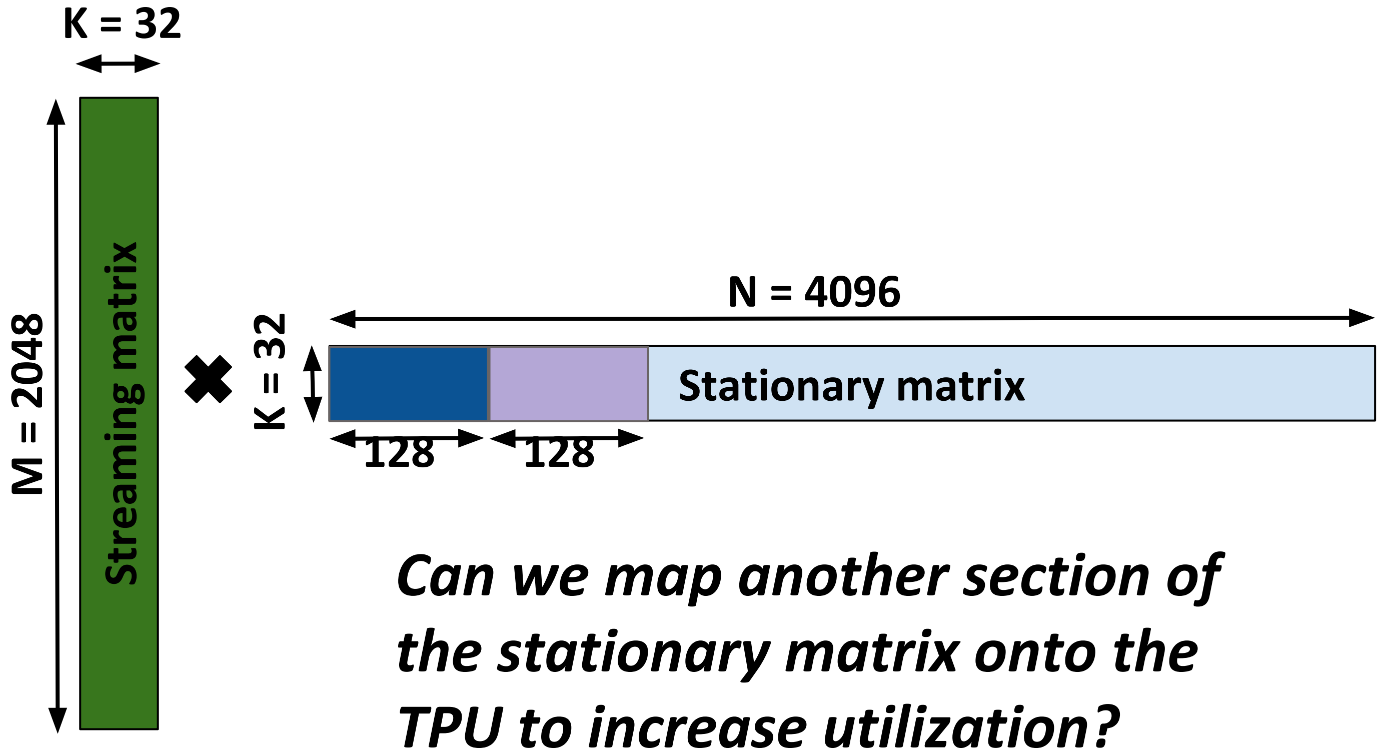


Can we map another section of the stationary matrix onto the TPU to increase utilization?

***** Assuming MK matrix is streaming and KN matrix is stationary. (aka weight stationary)***

Mapping GEMMs onto TPUs - Irregularity



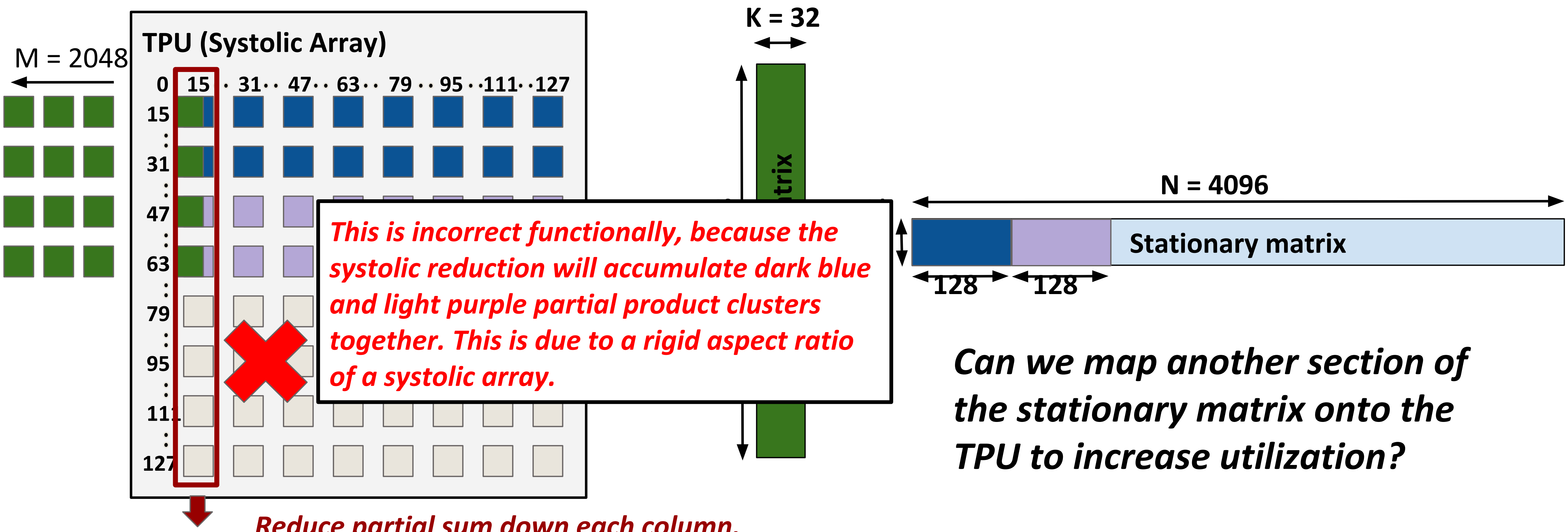


Can we map another section of the stationary matrix onto the TPU to increase utilization?

Reduce partial sum down each column.

***** Assuming MK matrix is streaming and KN matrix is stationary. (aka weight stationary)***

Mapping GEMMs onto TPUs - Irregularity



This is incorrect functionally, because the systolic reduction will accumulate dark blue and light purple partial product clusters together. This is due to a rigid aspect ratio of a systolic array.

Reduce partial sum down each column.

Can we map another section of the stationary matrix onto the TPU to increase utilization?

*** Assuming MK matrix is streaming and KN matrix is stationary. (aka weight stationary)*

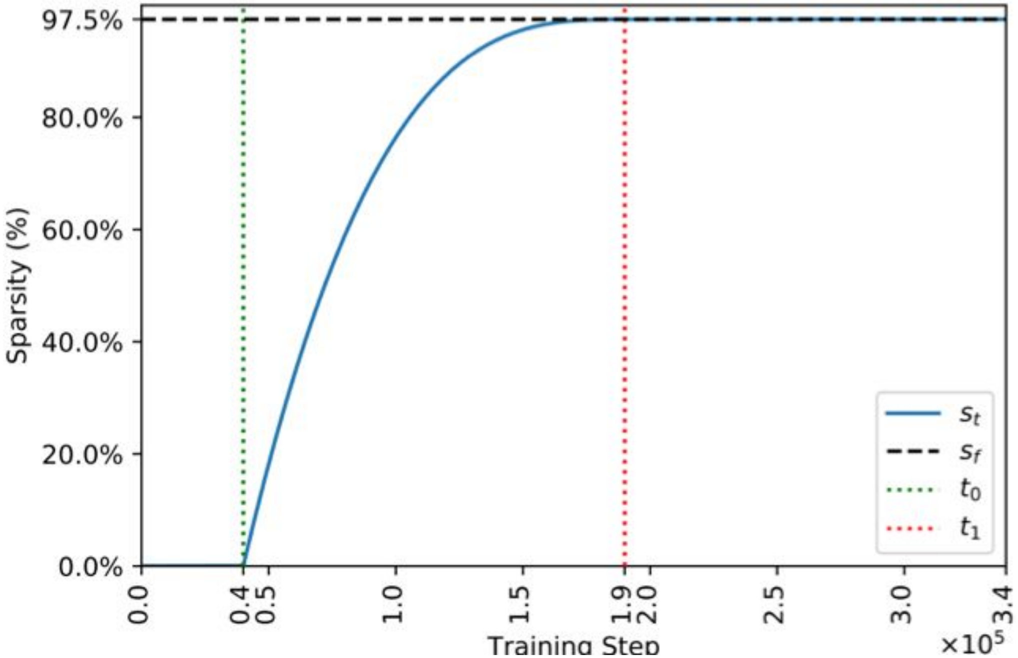
Mapping GEMMs onto TPUs - Irregularity

Observation 1: GEMMs are irregular and may not align to the aspect ratio of the systolic array.

Reduce partial sum down each column.

*** Assuming MK matrix is streaming and KN matrix is stationary. (aka weight stationary)*

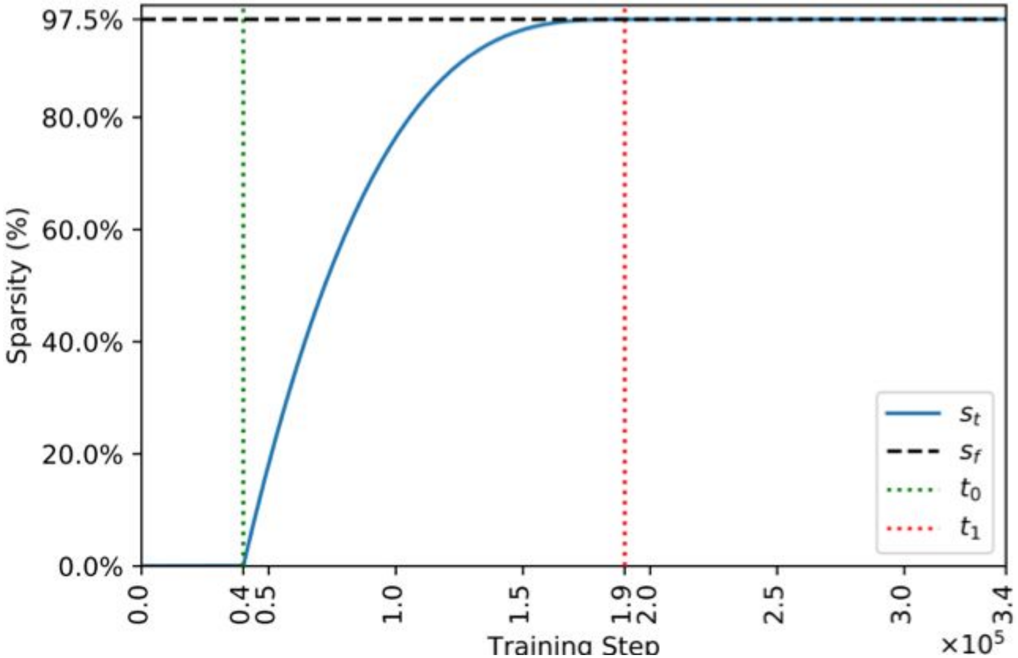
Sparsity in DNN Models



GNMT Pruning - Temporal Sparsity

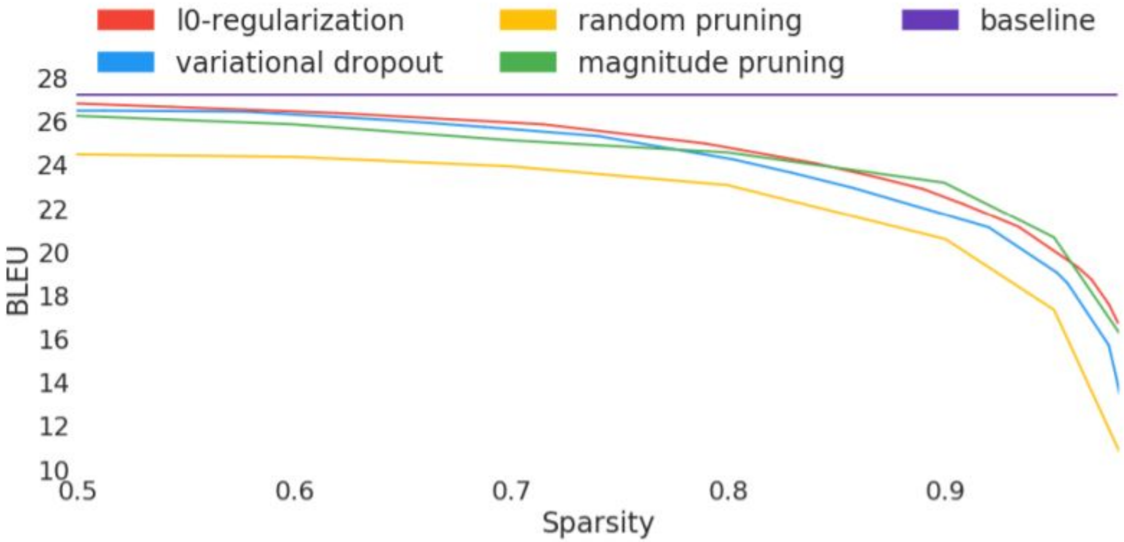
(https://www.intel.ai/compressing-gnmt-models)

Sparsity in DNN Models



GNMT Pruning - Temporal Sparsity

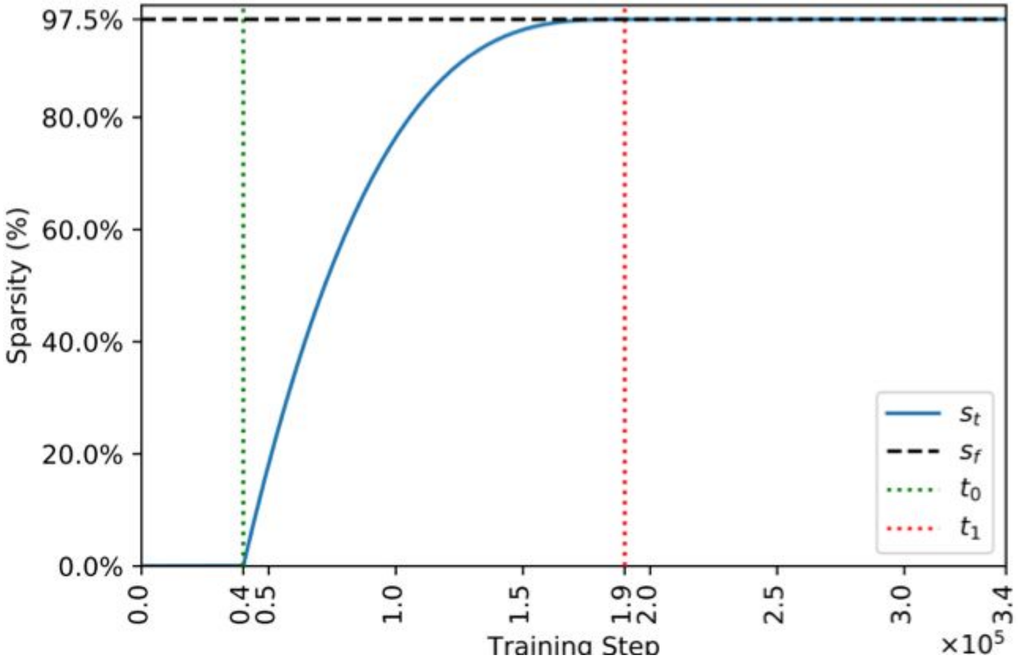
(https://www.intel.ai/compressing-gnmt-models)



Transformer Sparsity - Impact on BLEU

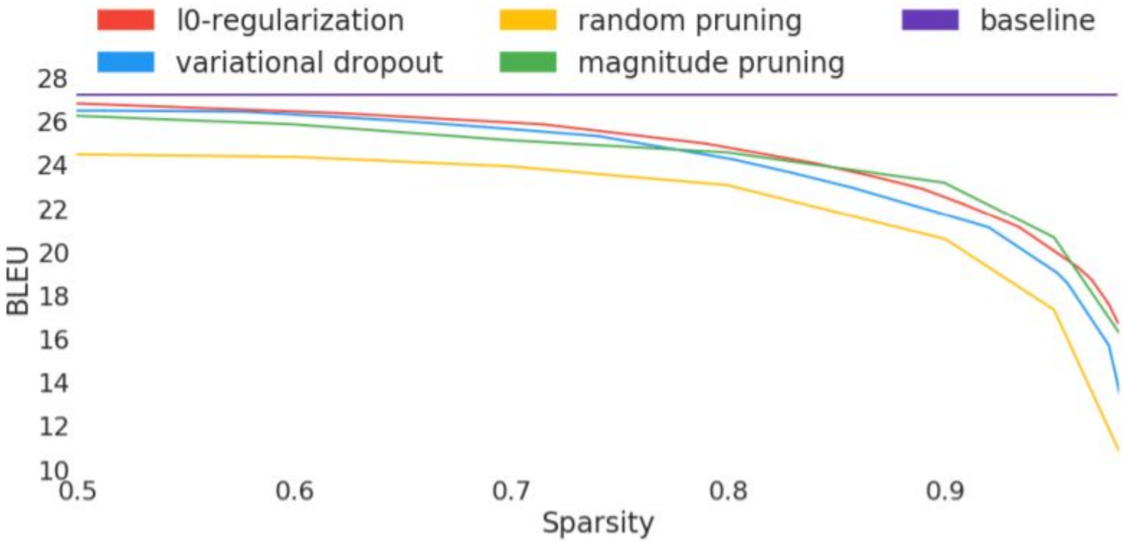
(The State of Sparsity in Deep Neural Networks, Gale et al., arXiv)

Sparsity in DNN Models



GNMT Pruning - Temporal Sparsity

(https://www.intel.ai/compressing-gnmt-models)

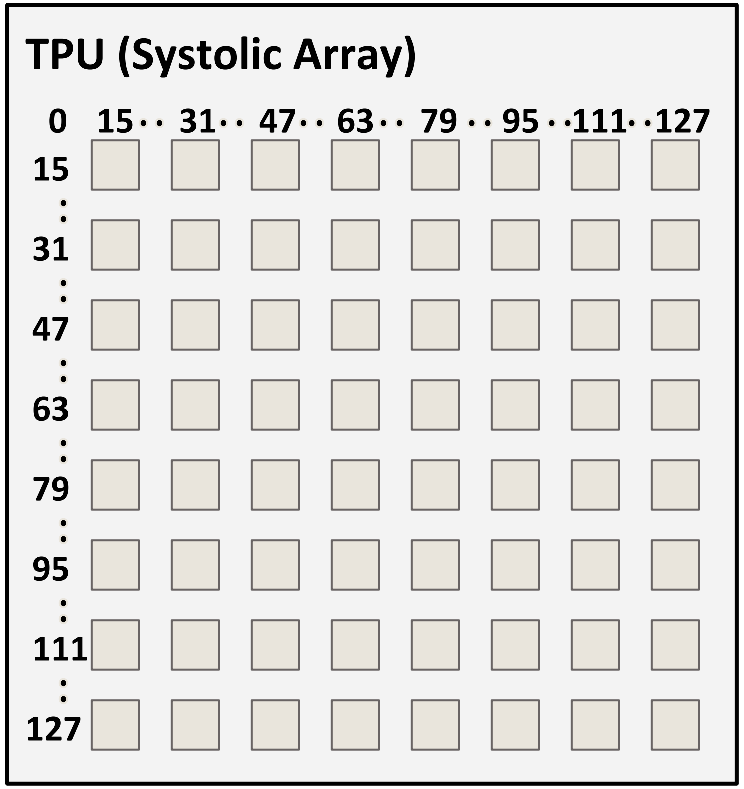


Transformer Sparsity - Impact on BLEU

(The State of Sparsity in Deep Neural Networks, Gale et al., arXiv)

Weight sparsity ranges from **40%** to **90%**. Activation sparsity is approximately **30%** to **70%** from ReLU, dropout, etc.

Mapping GEMMs onto TPUs - Sparsity



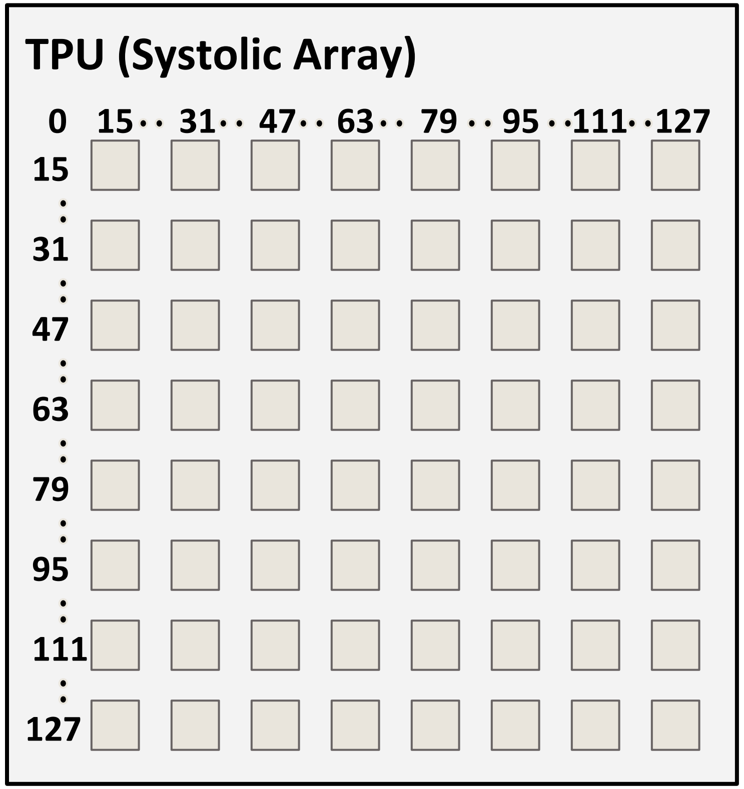
Workload	Application	Example Dimensions		
		M	N	K
GNMT	Machine Translation	128	2048	4096
		320	3072	4096
		1632	36548	1024
		2048	4096	32
DeepBench	General Workload	1024	16	500000
		35	8457	2560
Transformer	Language Understanding	31999	1024	84
		84	1024	4096
NCF	Collaborative Filtering	2048	1	128
		256	256	2048

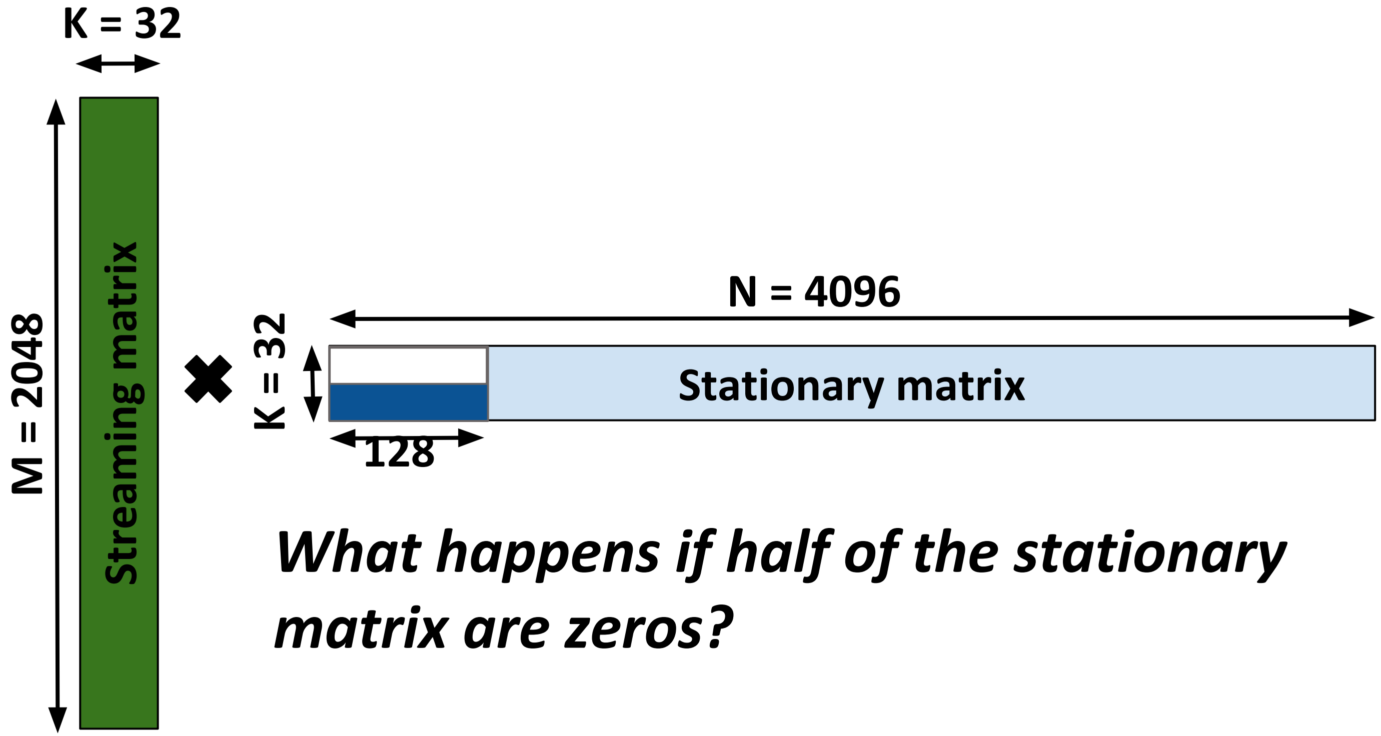
GEMMs used for evaluation.

Usually these GEMMs are sparse!

***** Assuming MK matrix is streaming and KN matrix is stationary. (aka weight stationary)***

Mapping GEMMs onto TPUs - Sparsity

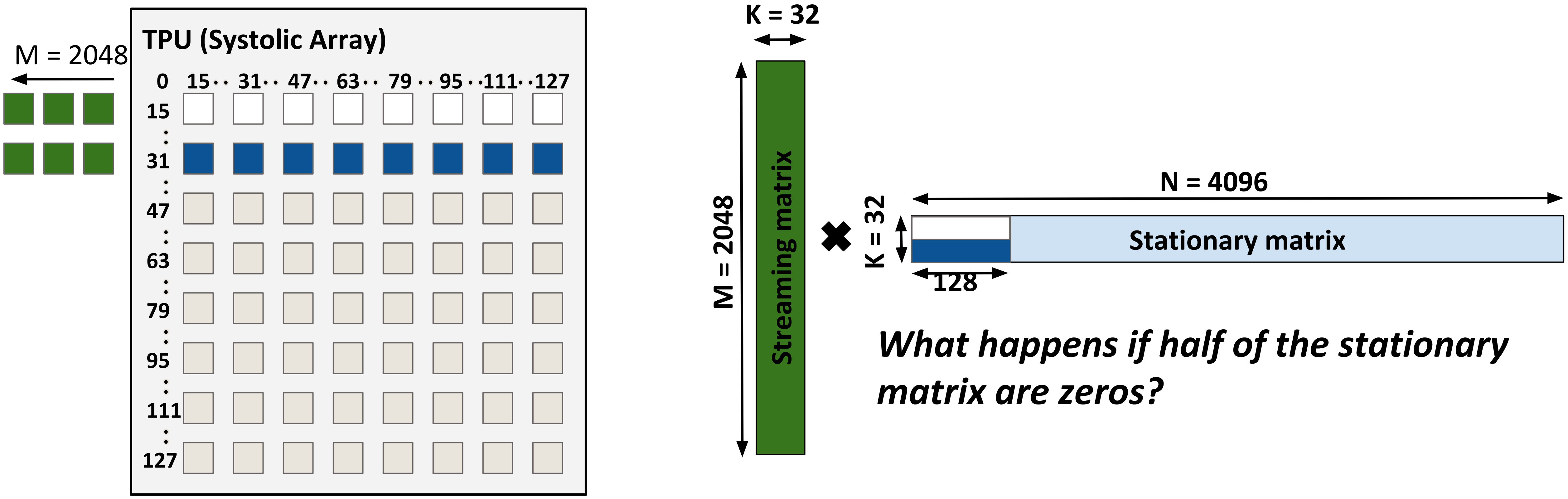




What happens if half of the stationary matrix are zeros?

***** Assuming MK matrix is streaming and KN matrix is stationary. (aka weight stationary)***

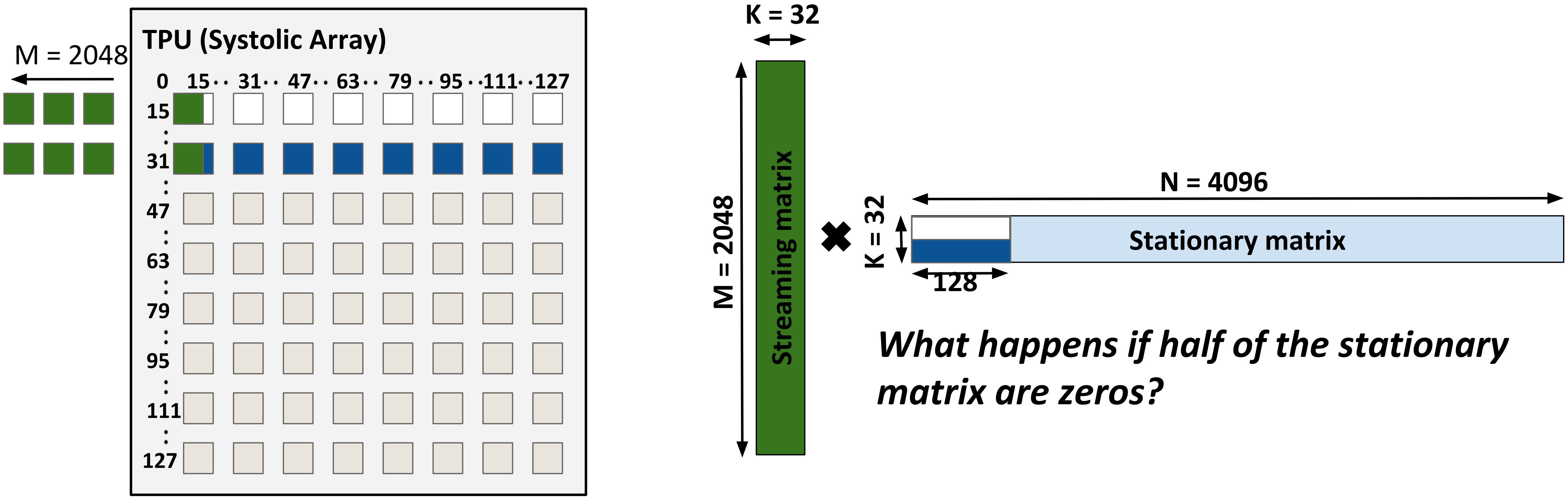
Mapping GEMMs onto TPUs - Sparsity



What happens if half of the stationary matrix are zeros?

***** Assuming MK matrix is streaming and KN matrix is stationary. (aka weight stationary)***

Mapping GEMMs onto TPUs - Sparsity

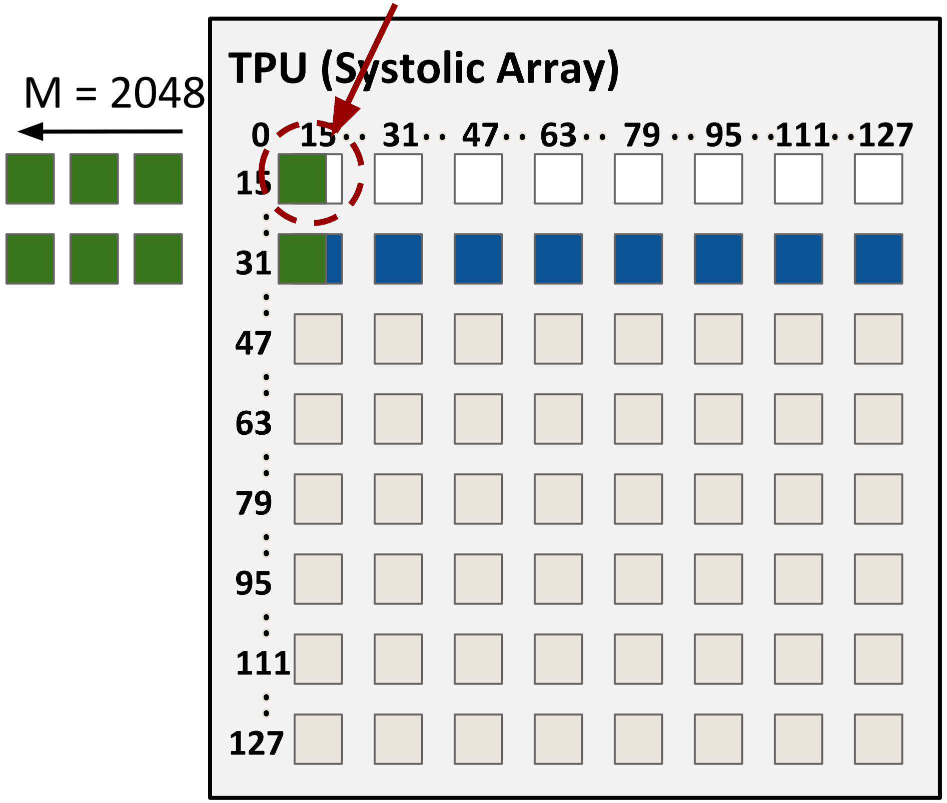


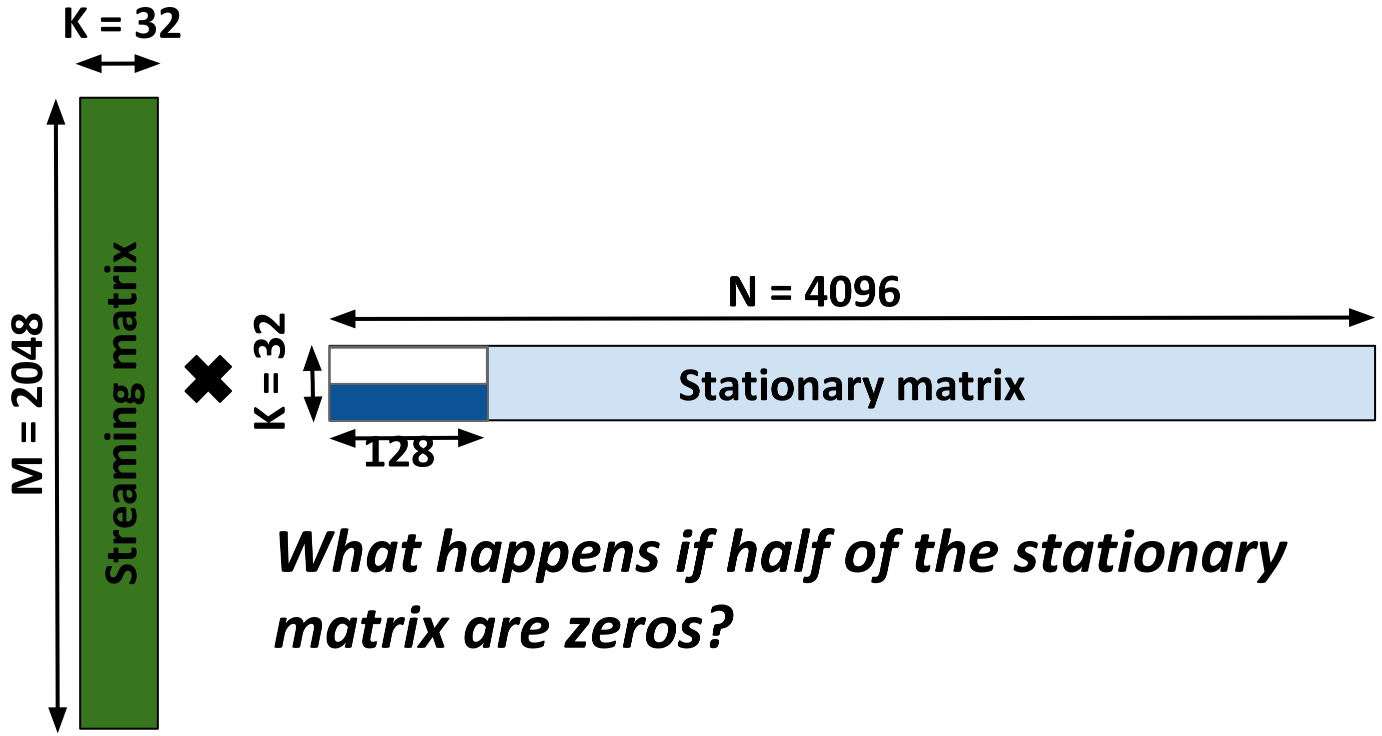
What happens if half of the stationary matrix are zeros?

*** Assuming MK matrix is streaming and KN matrix is stationary. (aka weight stationary)*

Mapping GEMMs onto TPUs - Sparsity

Multiplication with an operand that is zero is considered underutilized.





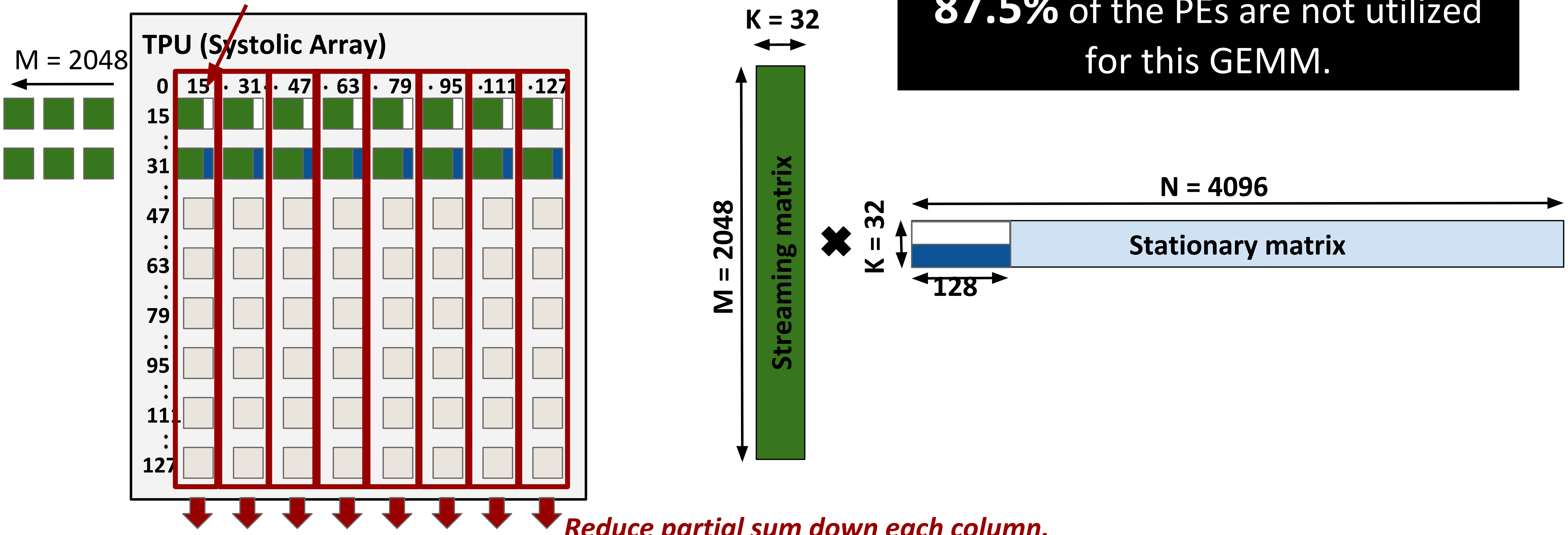
What happens if half of the stationary matrix are zeros?

*** Assuming MK matrix is streaming and KN matrix is stationary. (aka weight stationary)*

Mapping GEMMs onto TPUs - Sparsity

Multiplication with an operand that is zero is considered underutilized.

87.5% of the PEs are not utilized for this GEMM.



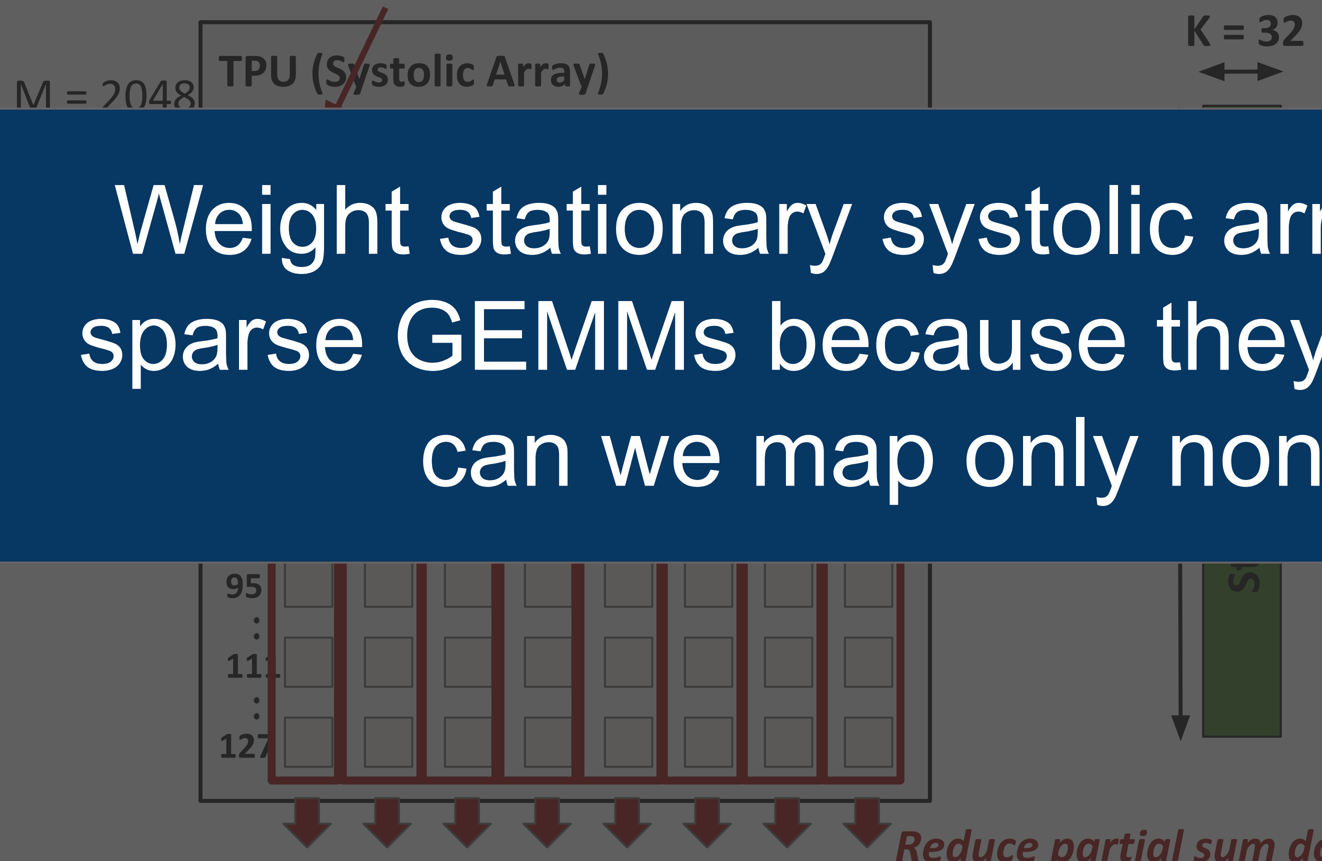
Reduce partial sum down each column.

*** Assuming MK matrix is streaming and KN matrix is stationary. (aka weight stationary)*

Mapping GEMMs onto TPUs - Sparsity

Multiplication with an operand that is zero is considered underutilized.

87.5% of the PEs are not utilized for this GEMM

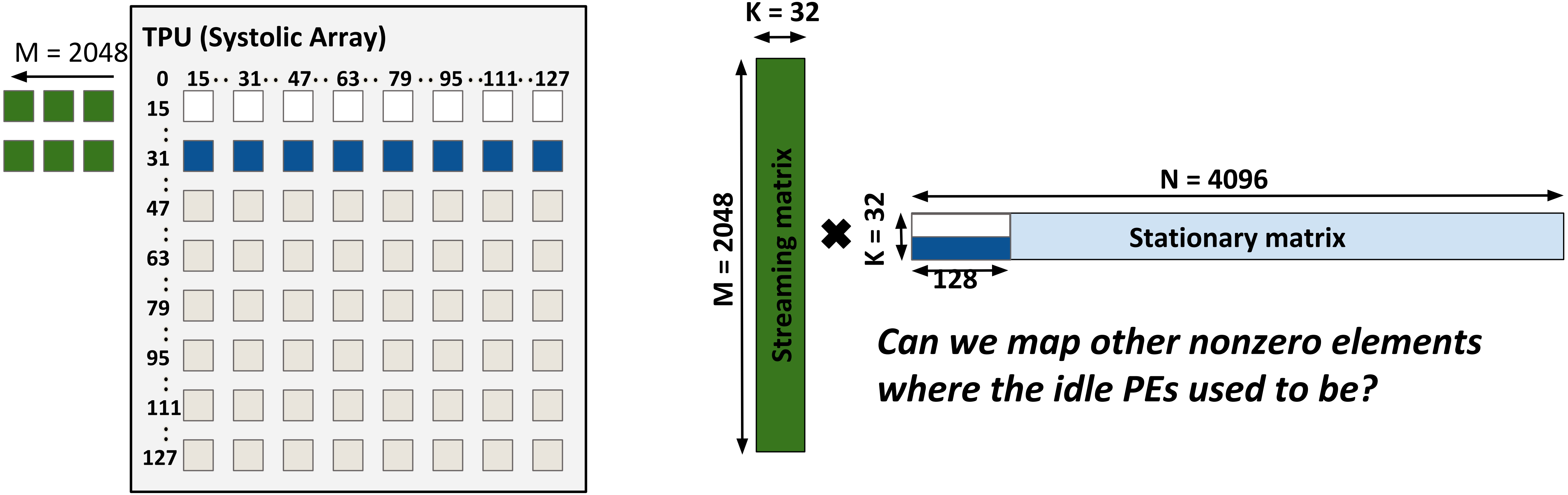


Weight stationary systolic arrays are underutilized for sparse GEMMs because they have to map zeros. How can we map only nonzeros stationary?

Reduce partial sum down each column.

*** Assuming MK matrix is streaming and KN matrix is stationary. (aka weight stationary)*

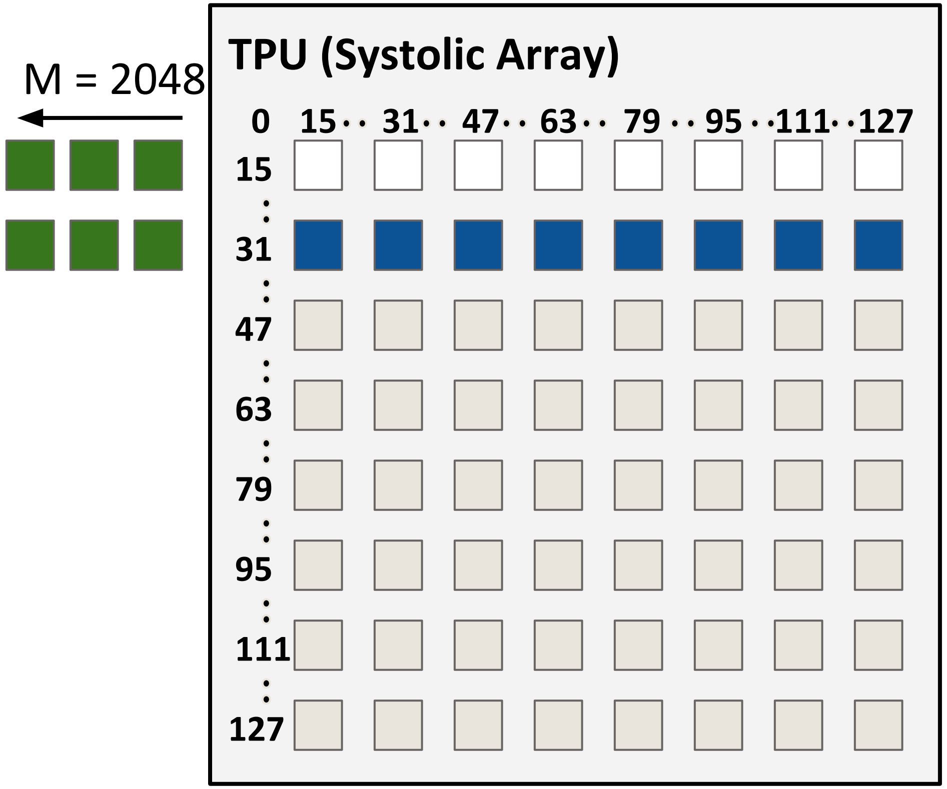
Mapping GEMMs onto TPUs - Sparsity

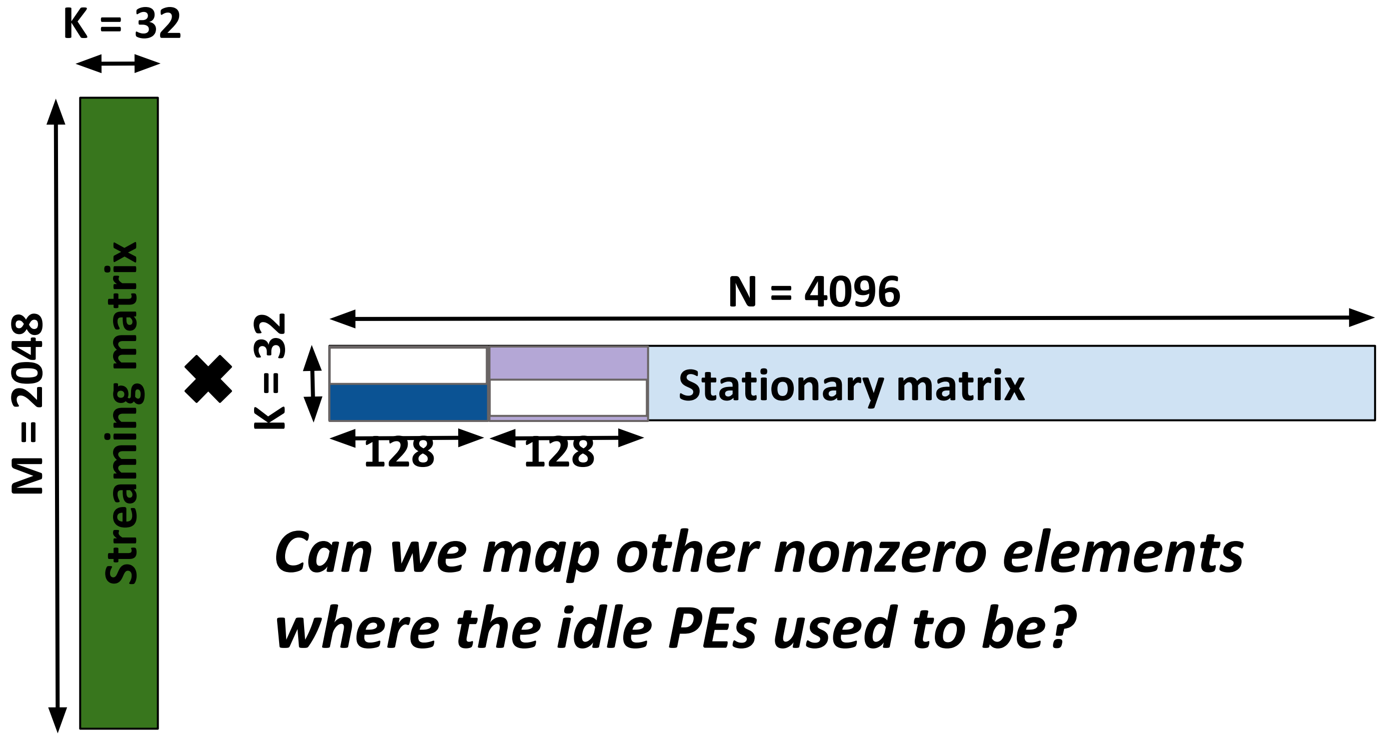


Can we map other nonzero elements where the idle PEs used to be?

***** Assuming MK matrix is streaming and KN matrix is stationary. (aka weight stationary)***

Mapping GEMMs onto TPUs - Sparsity

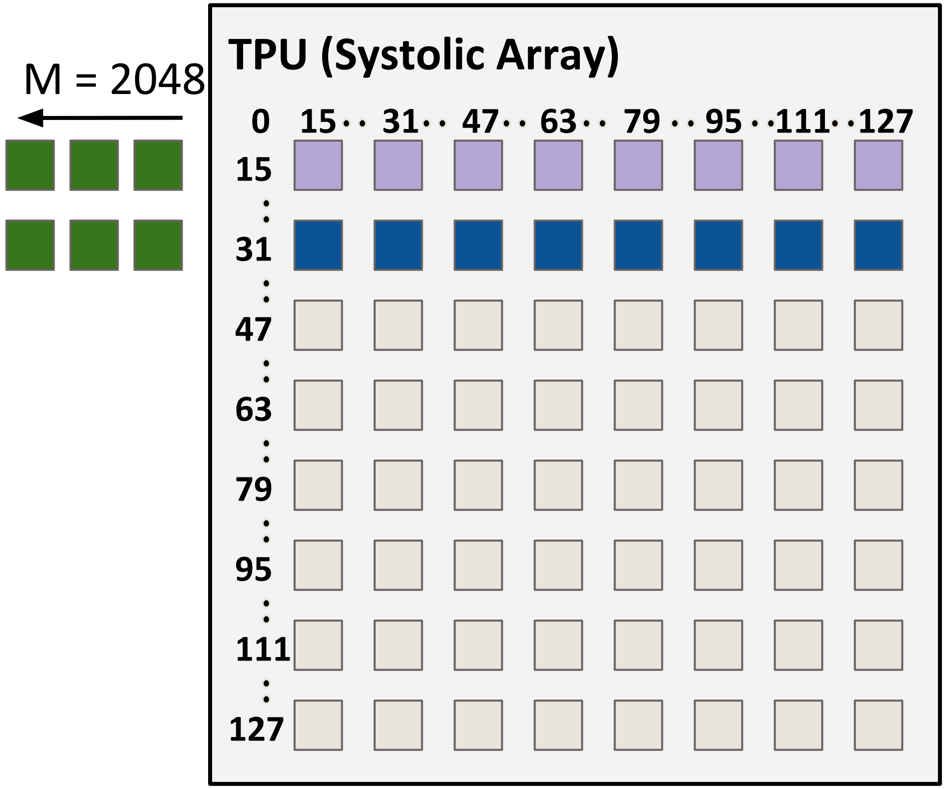


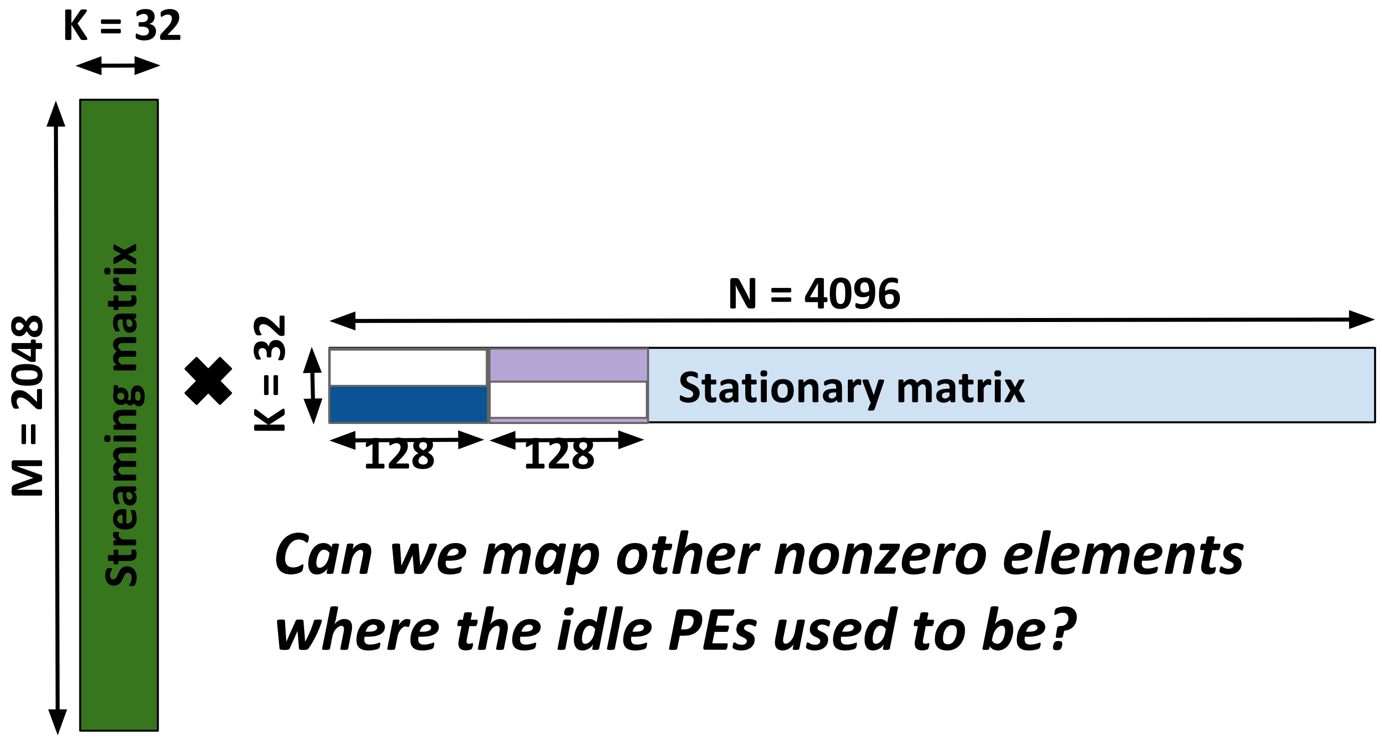


Can we map other nonzero elements where the idle PEs used to be?

***** Assuming MK matrix is streaming and KN matrix is stationary. (aka weight stationary)***

Mapping GEMMs onto TPUs - Sparsity

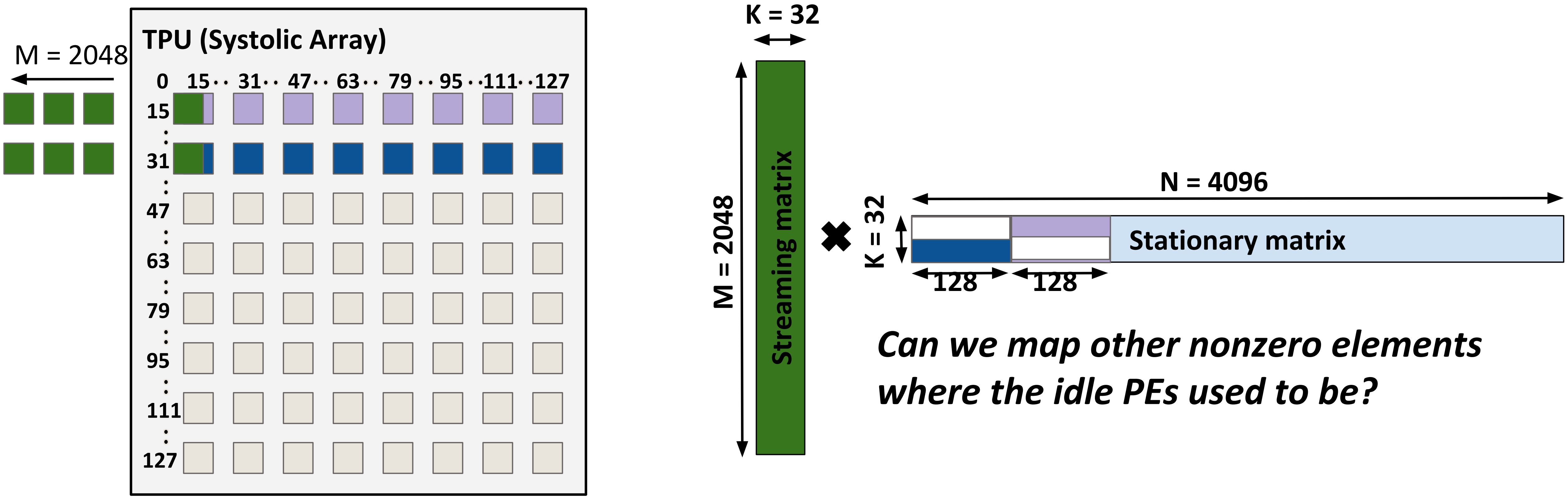




Can we map other nonzero elements where the idle PEs used to be?

***** Assuming MK matrix is streaming and KN matrix is stationary. (aka weight stationary)***

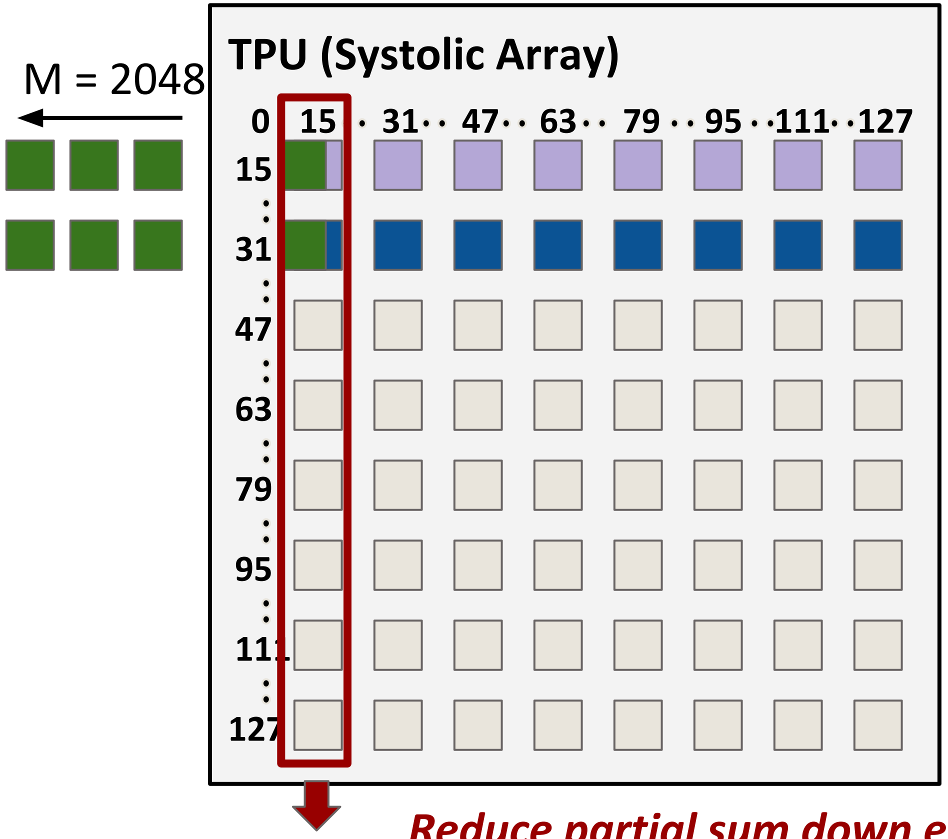
Mapping GEMMs onto TPUs - Sparsity



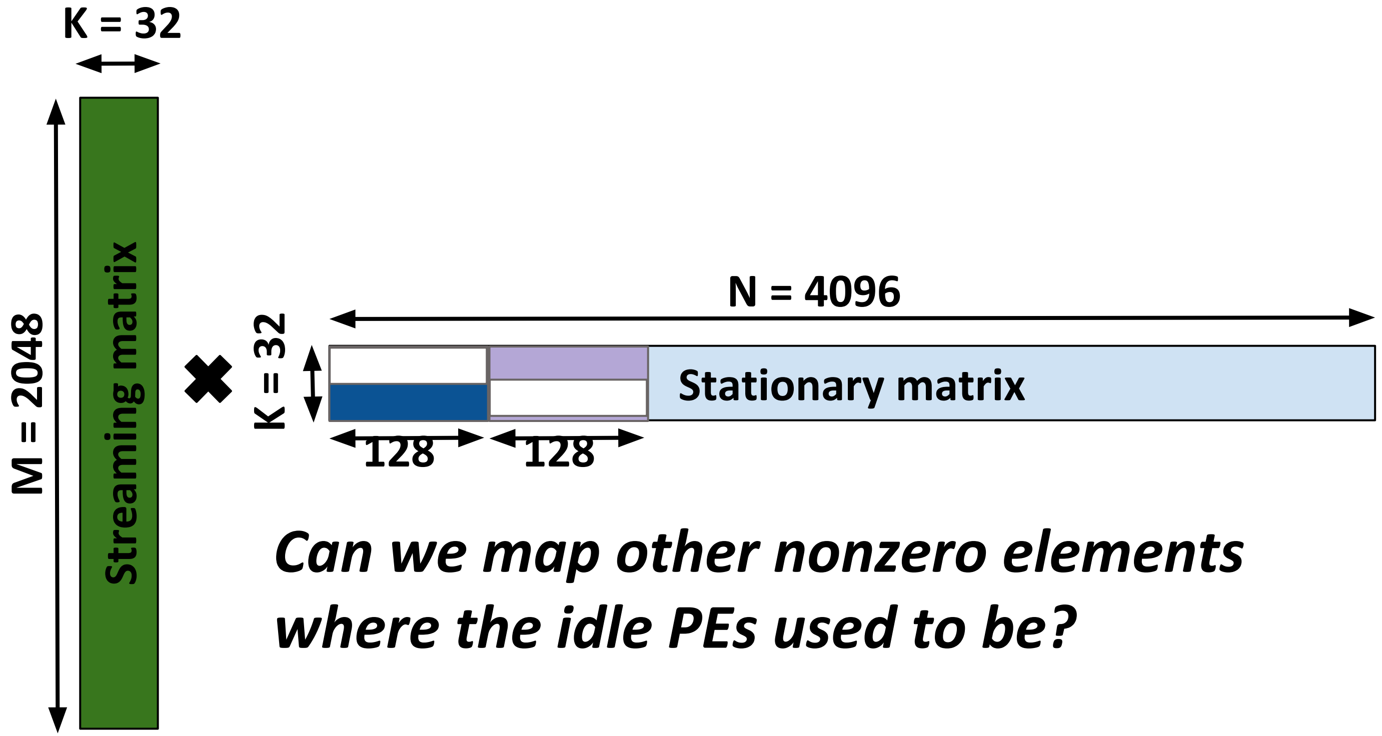
Can we map other nonzero elements where the idle PEs used to be?

***** Assuming MK matrix is streaming and KN matrix is stationary. (aka weight stationary)***

Mapping GEMMs onto TPUs - Sparsity



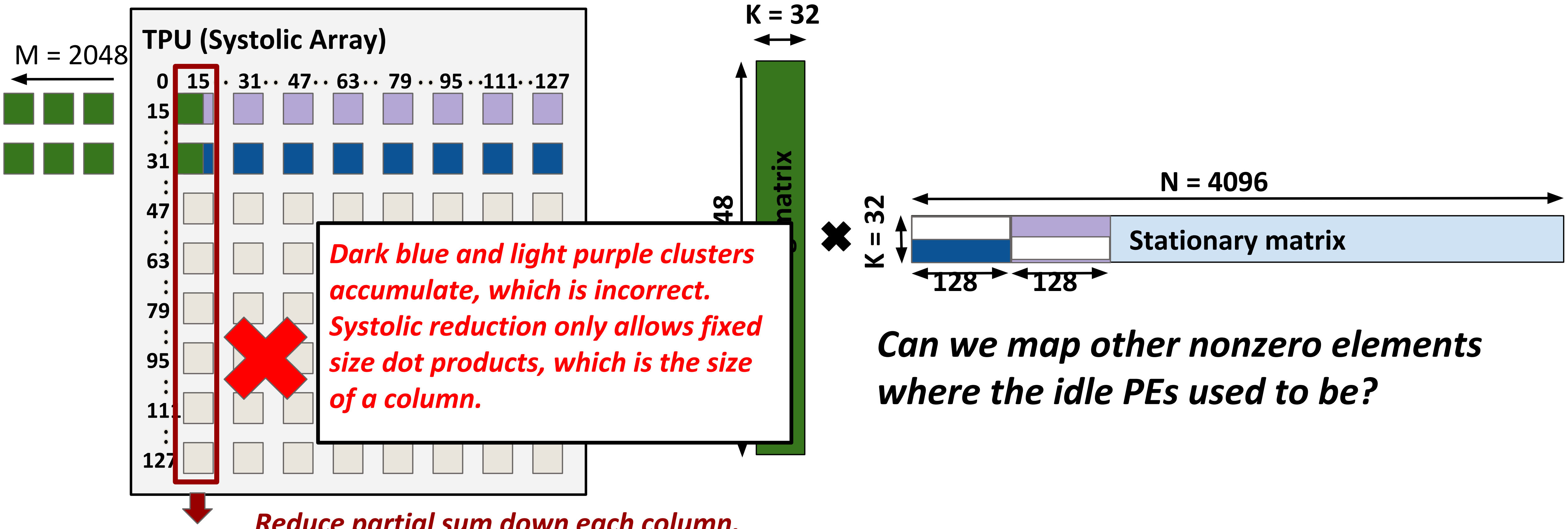
Reduce partial sum down each column.



Can we map other nonzero elements where the idle PEs used to be?

***** Assuming MK matrix is streaming and KN matrix is stationary. (aka weight stationary)***

Mapping GEMMs onto TPUs - Sparsity



Dark blue and light purple clusters accumulate, which is incorrect. Systolic reduction only allows fixed size dot products, which is the size of a column.

Reduce partial sum down each column.

Can we map other nonzero elements where the idle PEs used to be?

***** Assuming MK matrix is streaming and KN matrix is stationary. (aka weight stationary)***

Mapping GEMMs onto TPUs - Sparsity

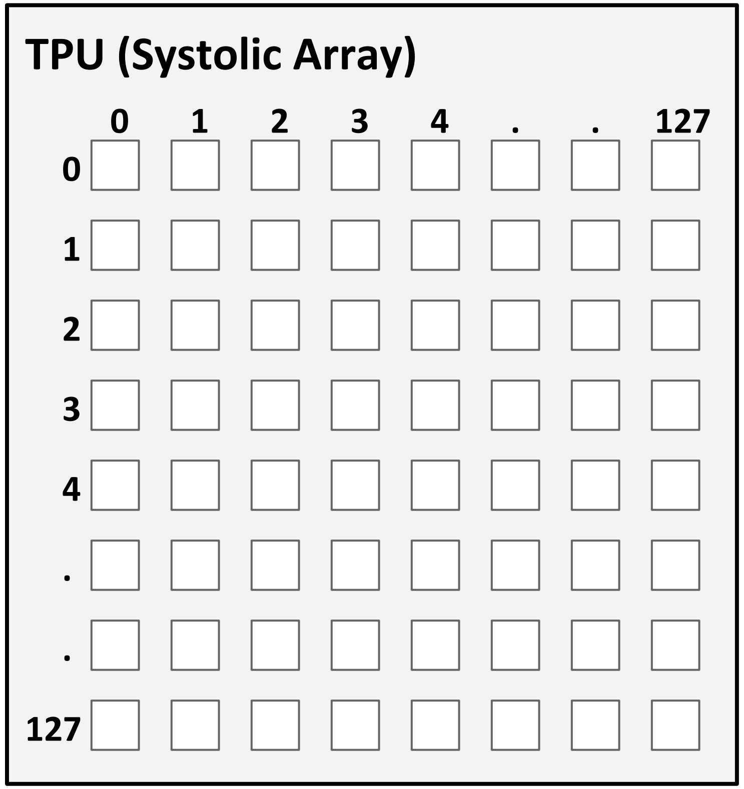
Observation 1: GEMMs are irregular and may not align to the aspect ratio of the systolic array.

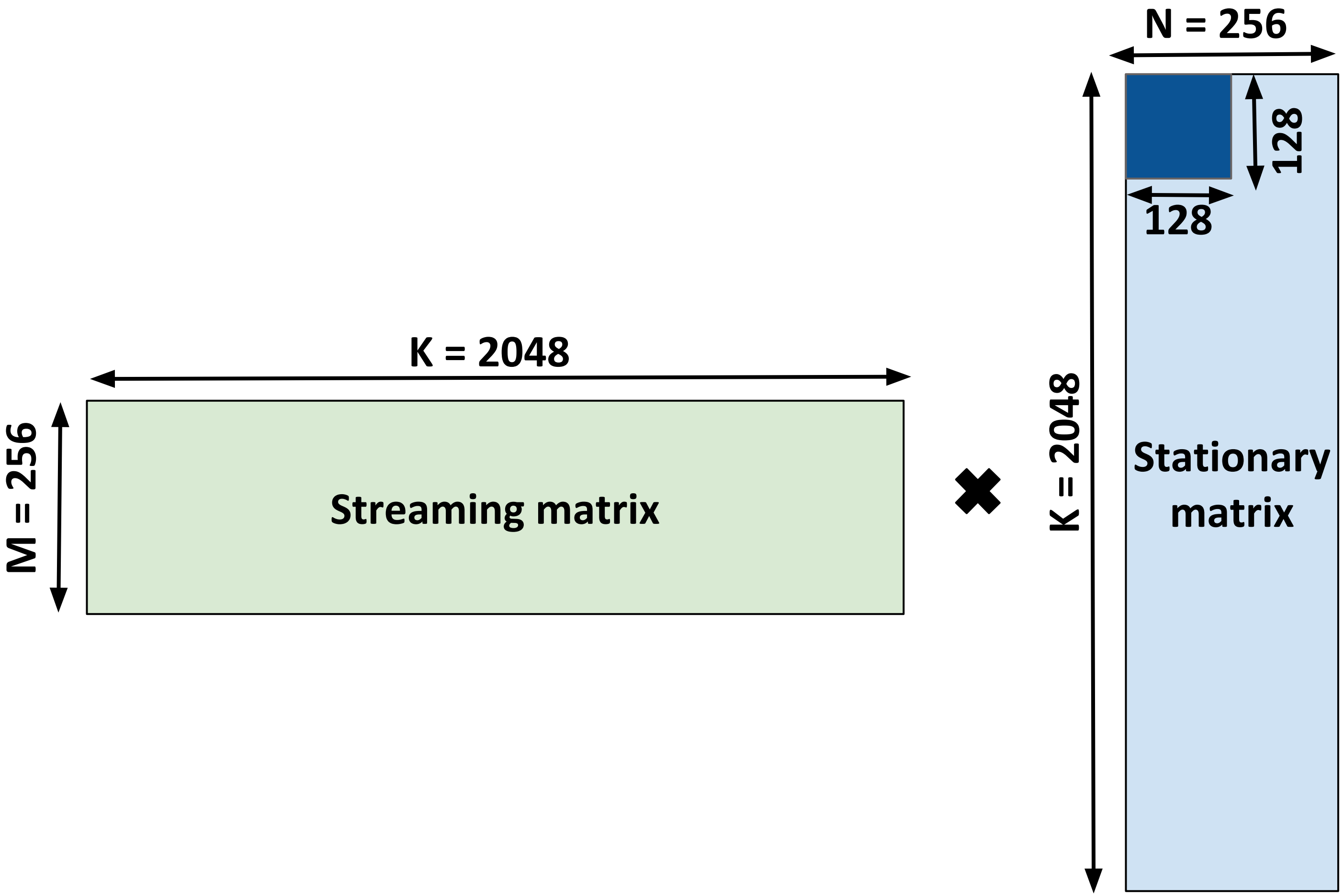
Observation 2: Sparse weights cause underutilization of the PEs and require variable sized dot product accumulation.

Reduce partial sum down each column.

*** Assuming MK matrix is streaming and KN matrix is stationary. (aka weight stationary)*

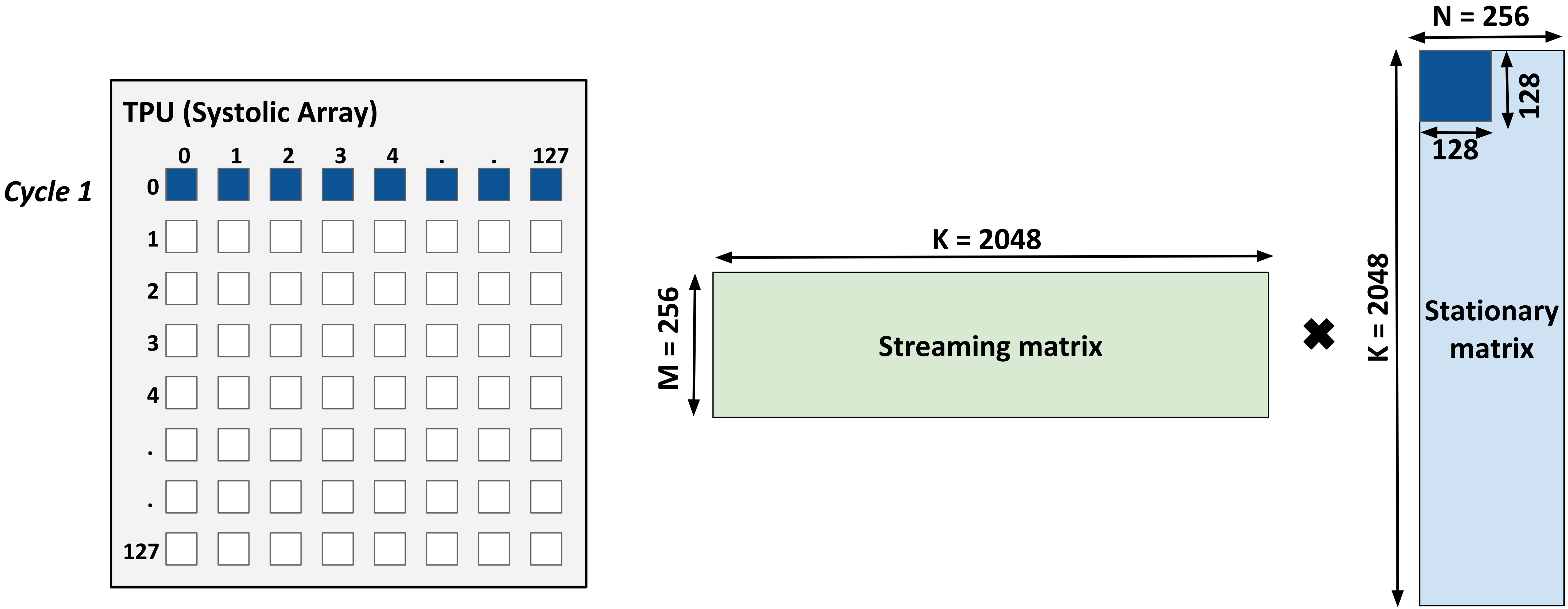
Mapping GEMMs onto TPUs - Scalability





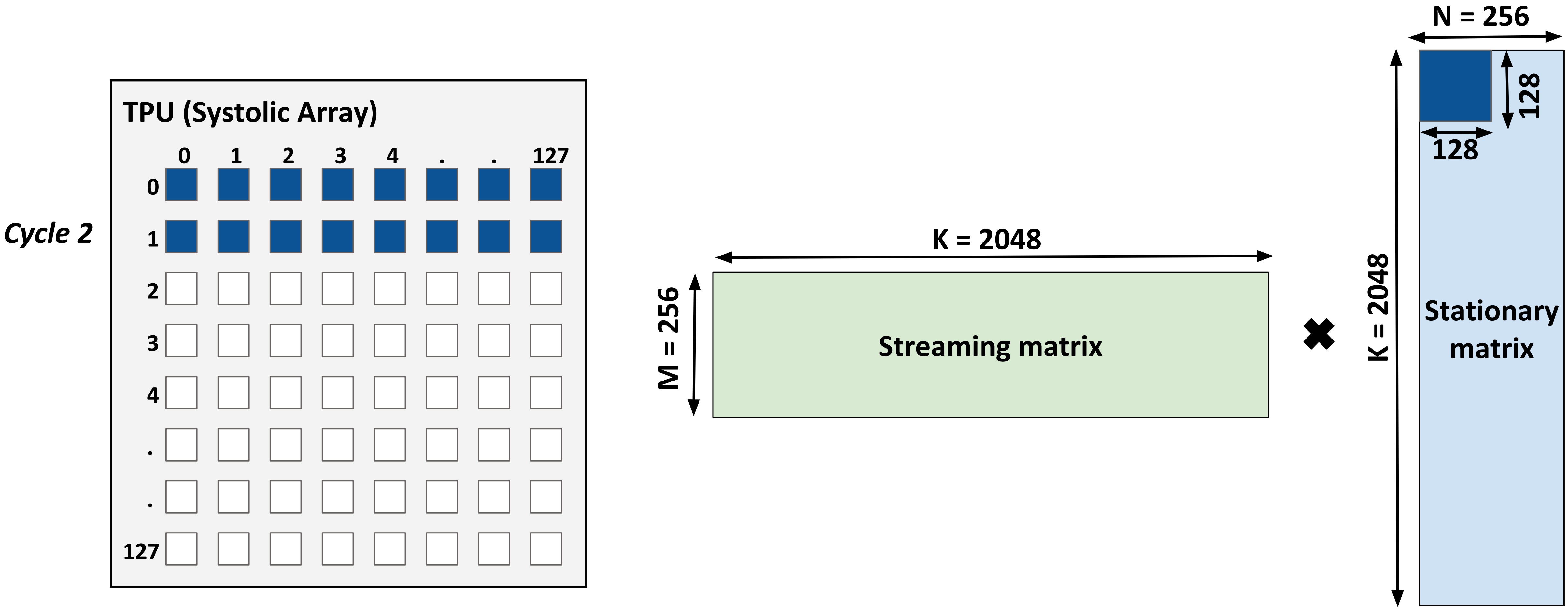
*** Assuming MK matrix is streaming and KN matrix is stationary. (aka weight stationary)*

Mapping GEMMs onto TPUs - Scalability



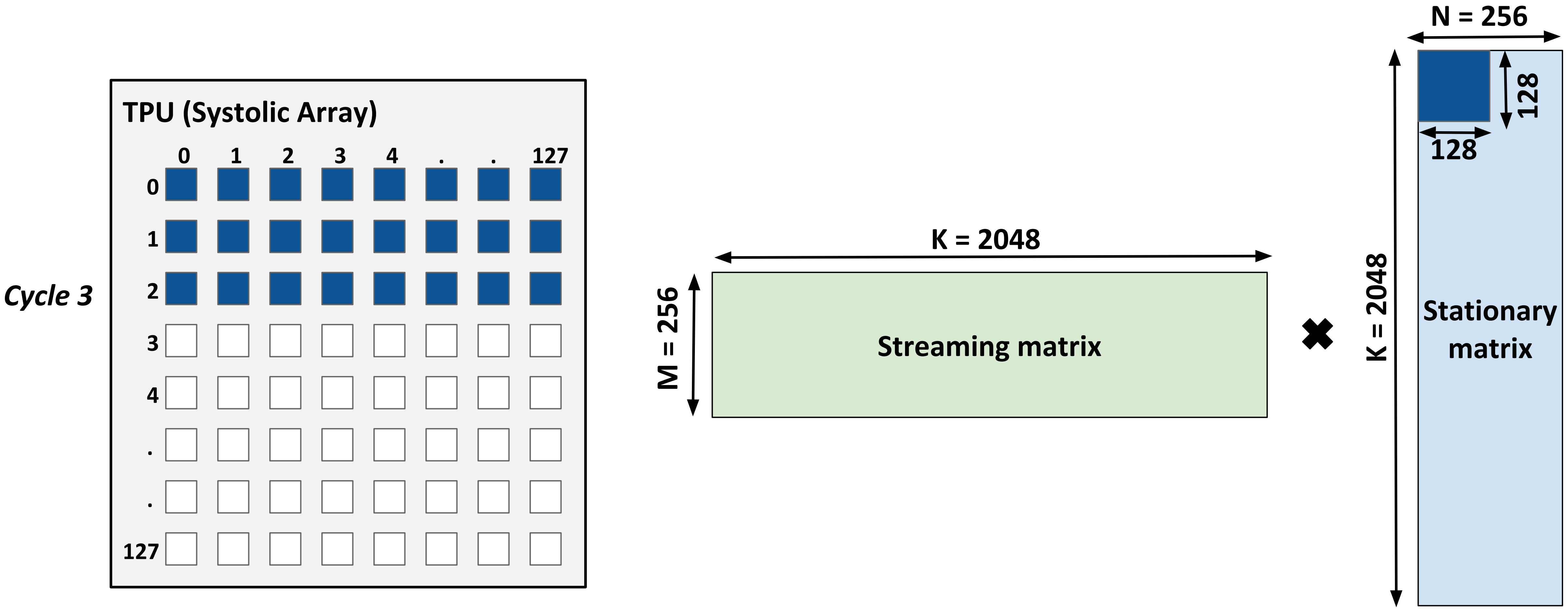
*** Assuming MK matrix is streaming and KN matrix is stationary. (aka weight stationary)*

Mapping GEMMs onto TPUs - Scalability



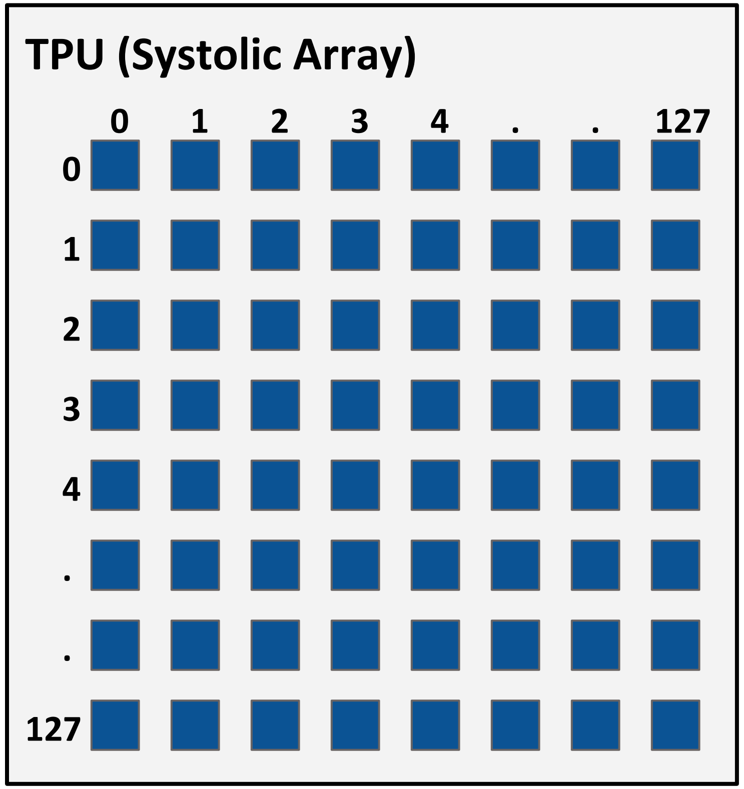
** *Assuming MK matrix is streaming and KN matrix is stationary. (aka weight stationary)*

Mapping GEMMs onto TPUs - Scalability

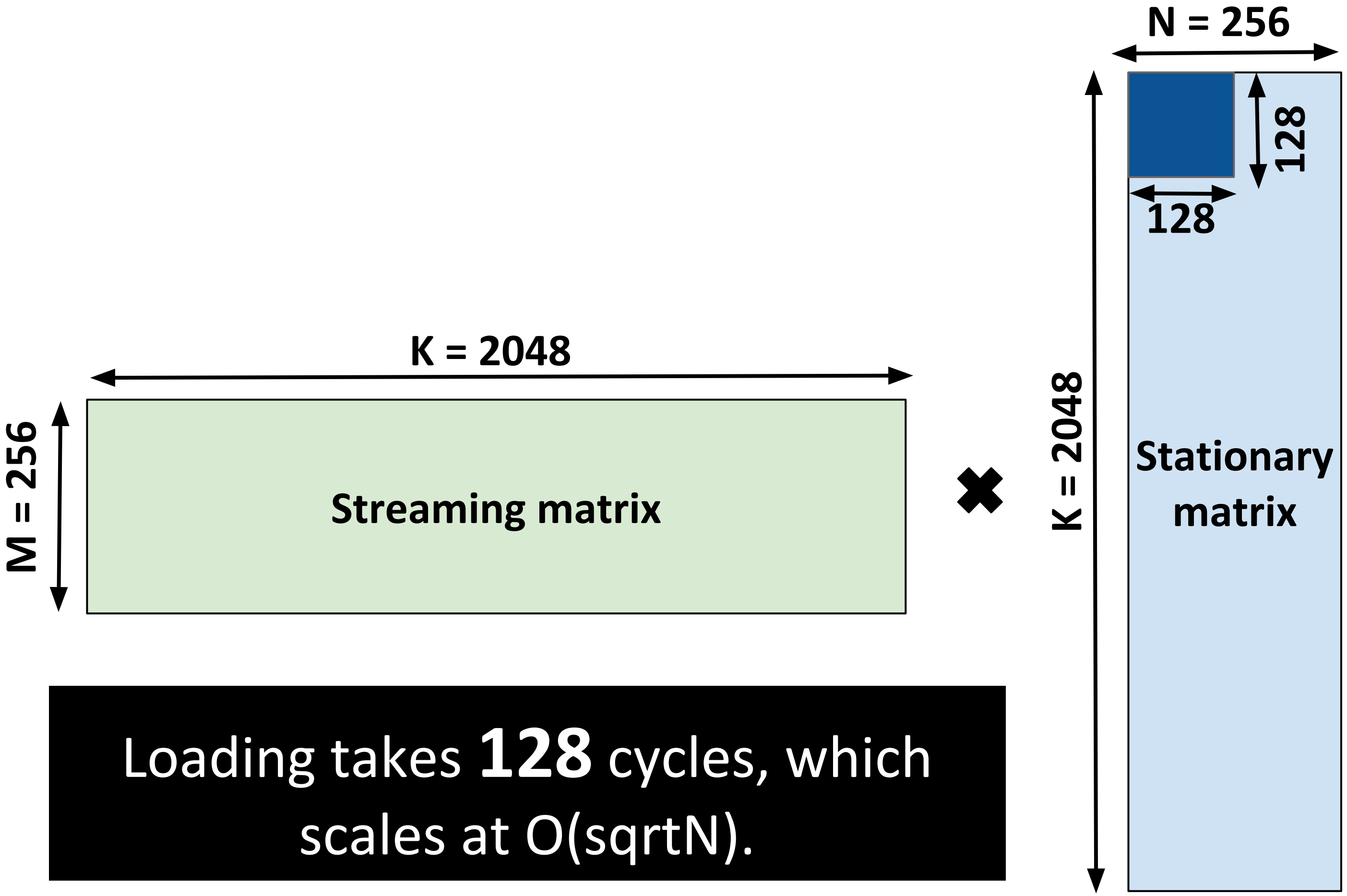


*** Assuming MK matrix is streaming and KN matrix is stationary. (aka weight stationary)*

Mapping GEMMs onto TPUs - Scalability



Cycle 128

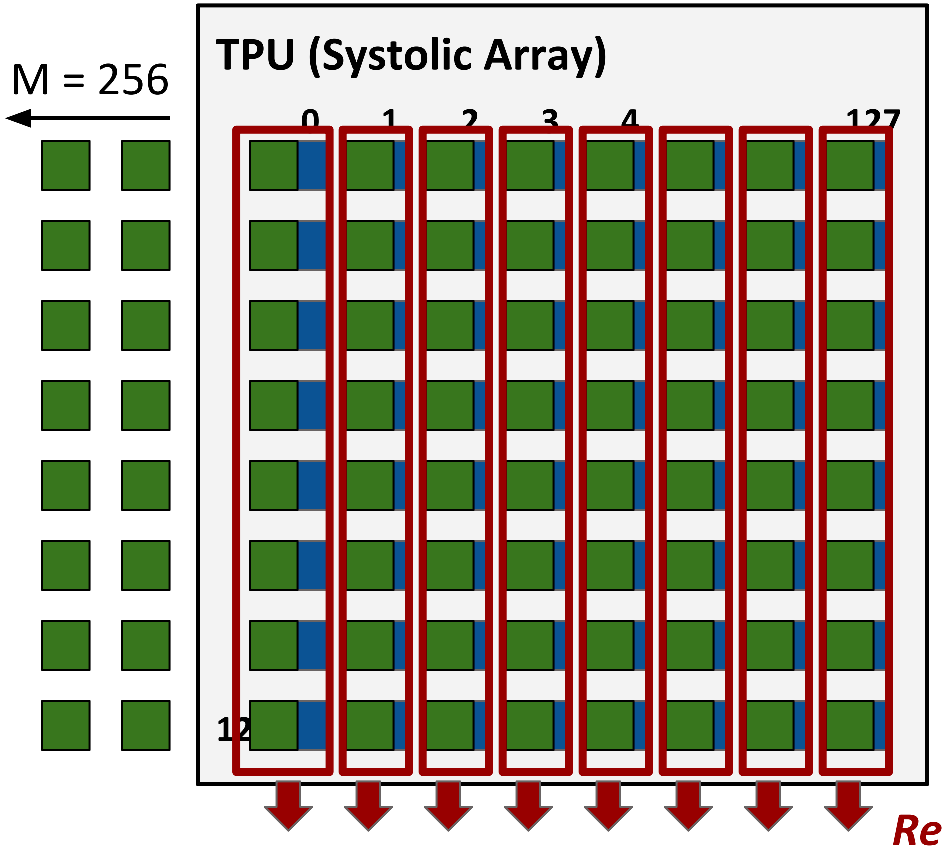


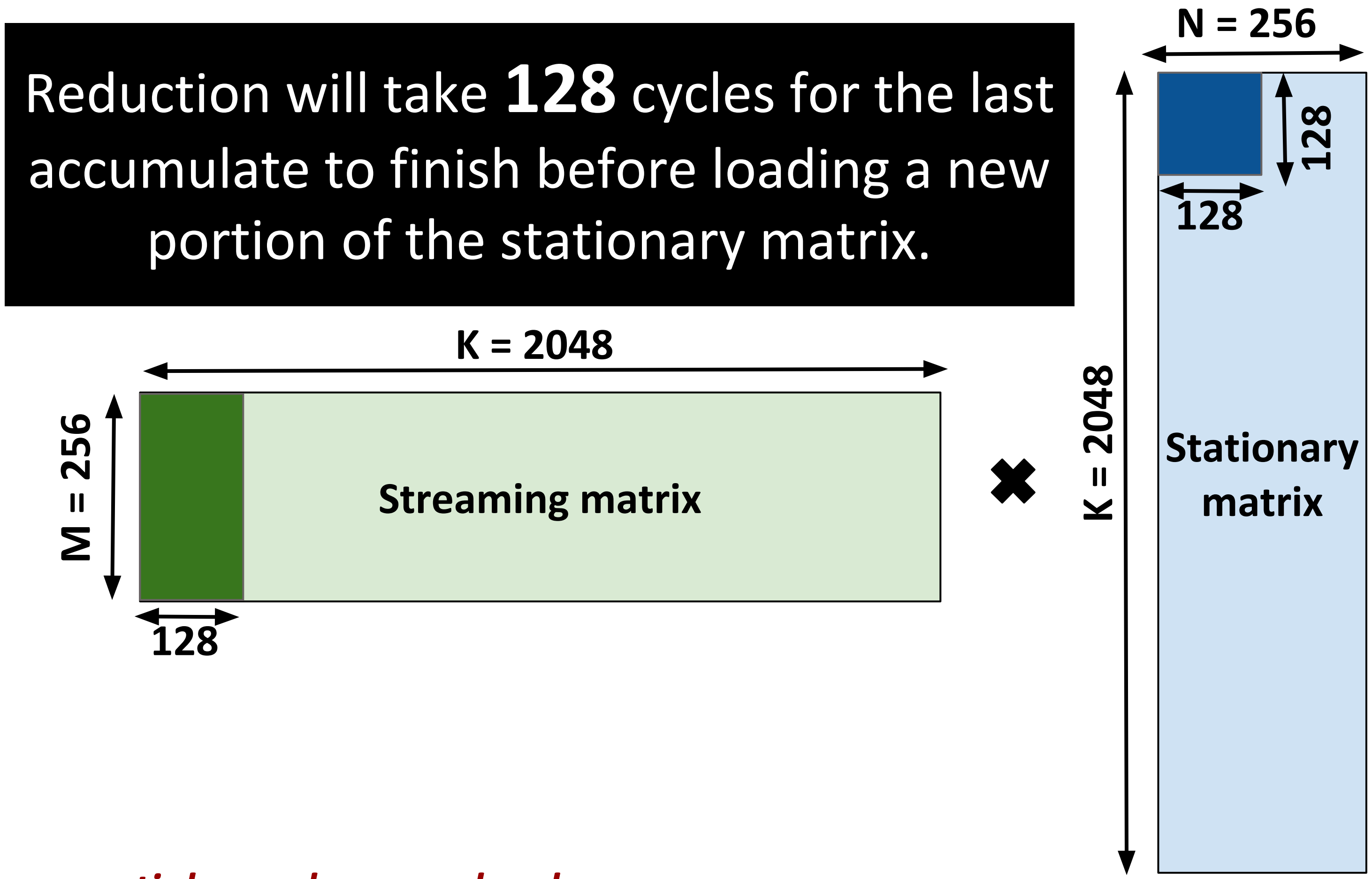
Loading takes **128** cycles, which scales at O(sqrtN).

***** Assuming MK matrix is streaming and KN matrix is stationary. (aka weight stationary)***

Mapping GEMMs onto TPUs - Scalability

Reduction will take **128** cycles for the last accumulate to finish before loading a new portion of the stationary matrix.





Reduce partial sum down each column.

***** Assuming MK matrix is streaming and KN matrix is stationary. (aka weight stationary)***

Mapping GEMMs onto TPUs - Scalability

N = 256

Observation 1: GEMMs are irregular and may not align to the aspect ratio of the systolic array.

Observation 2: Sparse weights cause underutilization of the PEs and require variable sized dot product accumulation.

Observation 3: Large systolic arrays have significant load and reduction latency.

Reduce partial sum down each column.

***** Assuming MK matrix is streaming and KN matrix is stationary. (aka weight stationary)***

Mapping GEMMs onto TPUs - Scalability

N = 256

Observation 1: GEMMs are irregular and may not align to the aspect ratio of the systolic array.

Observation 2: Sparse weights cause underutilization of the PEs and require variable sized dot product accumulation.

Observation 3: Large systolic arrays have significant load and reduction latency.

Takeaway: Systolic Arrays are underutilized on emerging GEMM workloads that are both sparse and irregular.

Reduce partial sum down each column.

*** Assuming MK matrix is streaming and KN matrix is stationary. (aka weight stationary)*

Outline

- Motivation
  - GEMMs in Deep Learning
  - Utilization on TPU (Systolic Array)
- **Accelerator Requirements**
- SIGMA
  - Interconnects Implementations
  - Full System Design
- Evaluation
- Conclusion

Key Accelerator Requirements

Requirement	Systolic Array Limitation	SIGMA Desired Traits
Flexibility	● rigid aspect ratio	● ability to mimic any 2D aspect ratio

Key Accelerator Requirements

Requirement	Systolic Array Limitation	SIGMA Desired Traits
Flexibility	● rigid aspect ratio	● ability to mimic any 2D aspect ratio
Sparsity Support	● data forwarding every cycle requires systolic array to map zeros	● sparsity support by mapping only nonzeros stationary ● ability to create simultaneous variable sized dot product

Key Accelerator Requirements

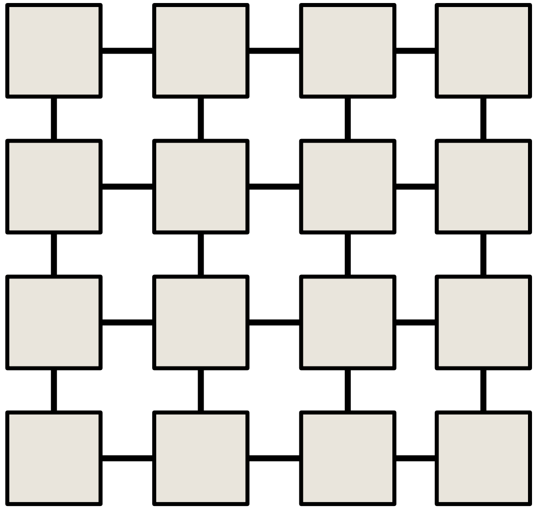
Requirement	Systolic Array Limitation	SIGMA Desired Traits
Flexibility	● rigid aspect ratio	● ability to mimic any 2D aspect ratio
Sparsity Support	● data forwarding every cycle requires systolic array to map zeros	● sparsity support by mapping only nonzeros stationary ● ability to create simultaneous variable sized dot product
Scalability	● O(sqrtN) distribution ● O(sqrtN) reduction	● O(1) distribution ● O(logN) reduction

Key Accelerator Requirements

Requirement	Systolic Array Limitation	SIGMA Desired Traits
Scalability	● O(sqrtN) distribution ● O(sqrtN) reduction	● O(1) distribution ● O(logN) reduction

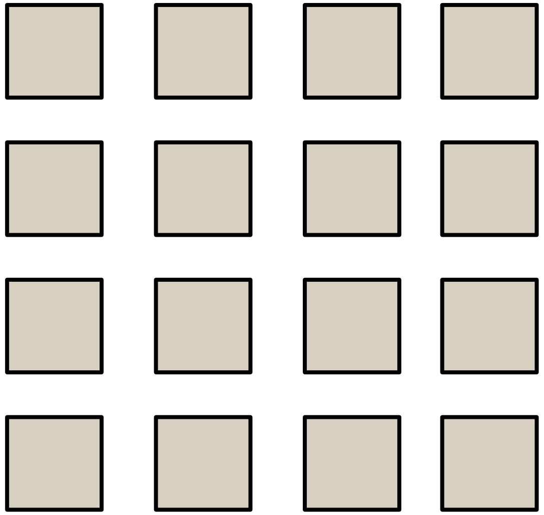
With flexible and scalable interconnects between all PEs, SIGMA can solve the three requirements.

Systolic vs SIGMA High Level Interconnects



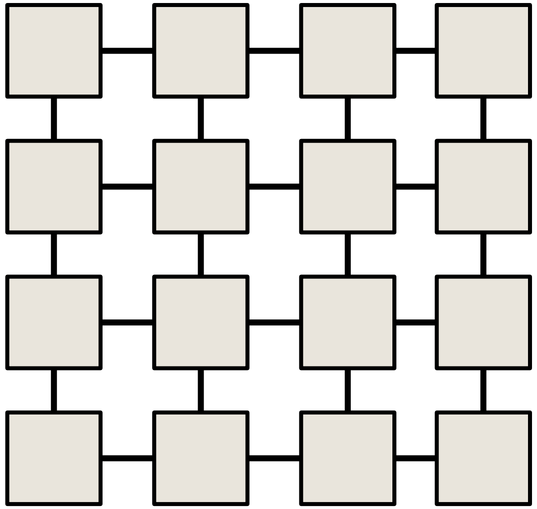
4 x 4 Systolic Array

- rigid aspect ratio
- fixed size dot product
- O(sqrtN) distribution and reduction



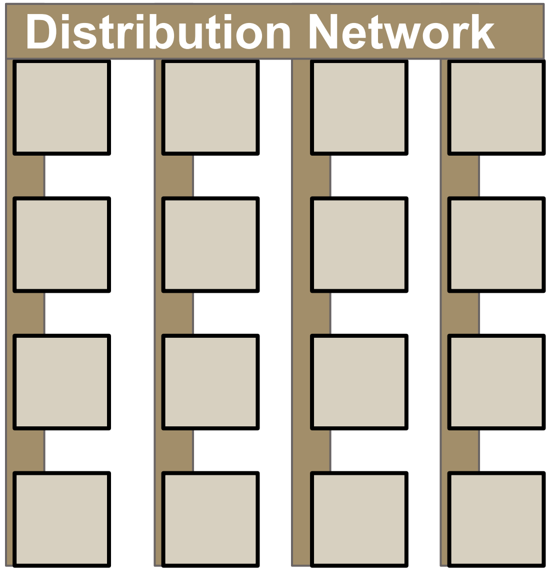
16 PE SIGMA

Systolic vs SIGMA High Level Interconnects



4 x 4 Systolic Array

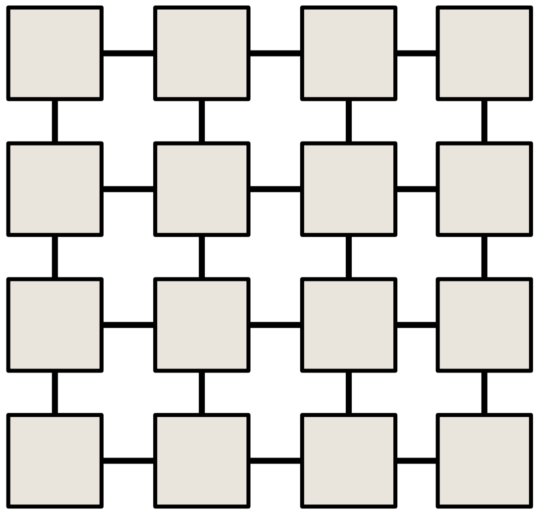
- rigid aspect ratio
- fixed size dot product
- O(sqrtN) distribution and reduction



16 PE SIGMA

***** Microarchitecture details on the networks will be discussed later***

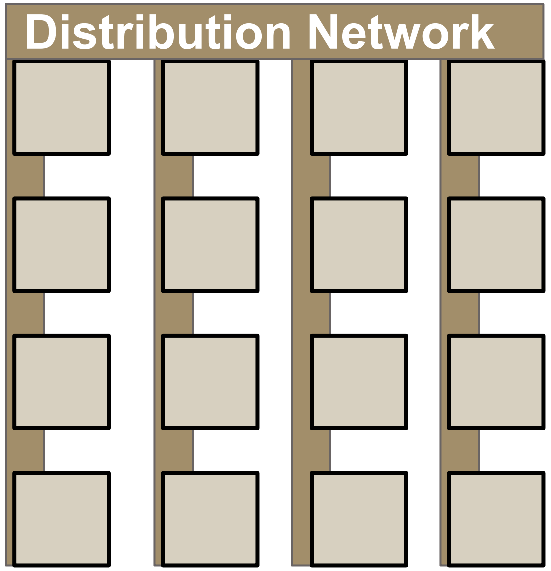
Systolic vs SIGMA High Level Interconnects



4 x 4 Systolic Array

- rigid aspect ratio
- fixed size dot product
- O(sqrtN) distribution and reduction

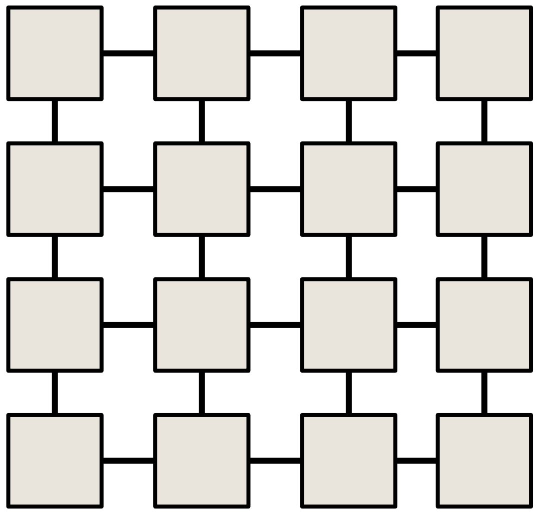
- **Distribution network allows SIGMA to mimic any aspect ratio to address irregular GEMMs**
- **Ability to send any streaming element to any PE**
- **O(1) distribution**



16 PE SIGMA

***** Microarchitecture details on the networks will be discussed later***

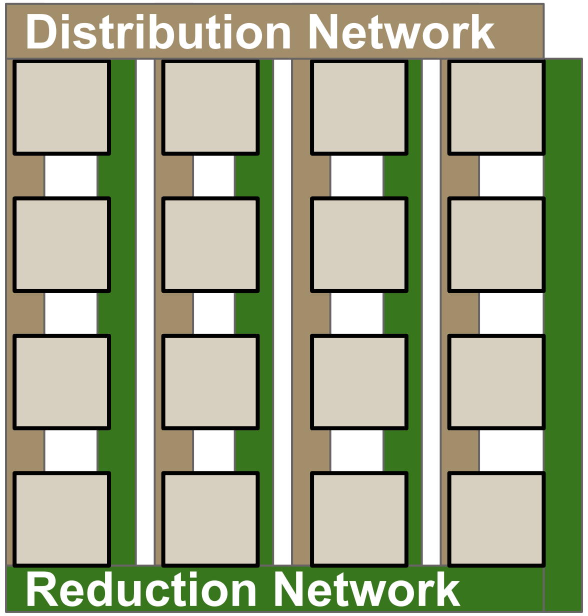
Systolic vs SIGMA High Level Interconnects



4 x 4 Systolic Array

- rigid aspect ratio
- fixed size dot product
- O(sqrtN) distribution and reduction

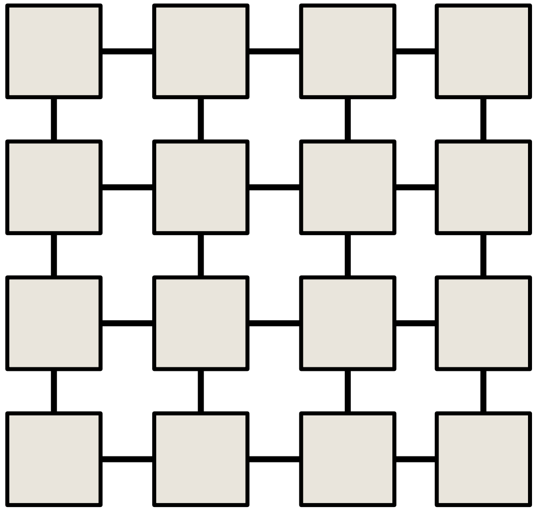
- **Distribution network allows SIGMA to mimic any aspect ratio to address irregular GEMMs**
- **Ability to send any streaming element to any PE**
- **O(1) distribution**



16 PE SIGMA

***** Microarchitecture details on the networks will be discussed later***

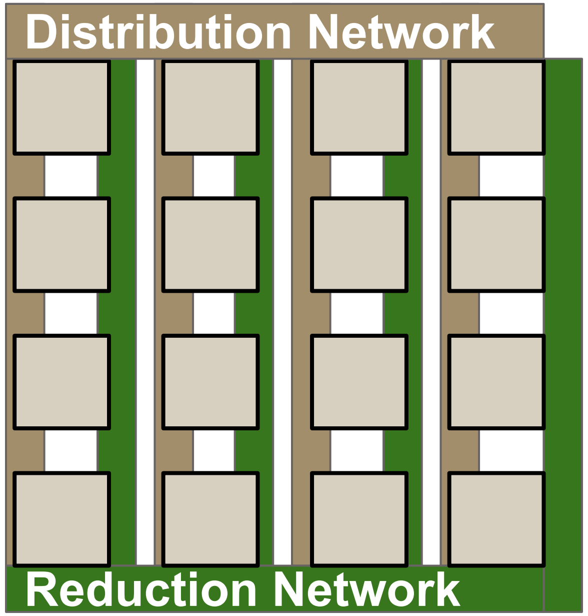
Systolic vs SIGMA High Level Interconnects



4 x 4 Systolic Array

- rigid aspect ratio
- fixed size dot product
- O(sqrtN) distribution and reduction

- **Distribution network allows SIGMA to mimic any aspect ratio to address irregular GEMMs**
- **Ability to send any streaming element to any PE**
- **O(1) distribution**

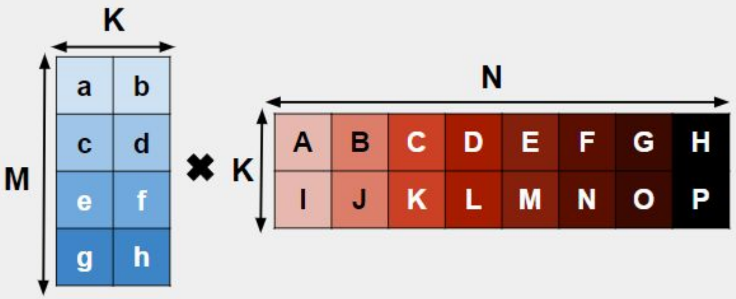


16 PE SIGMA

***** Microarchitecture details on the networks will be discussed later***

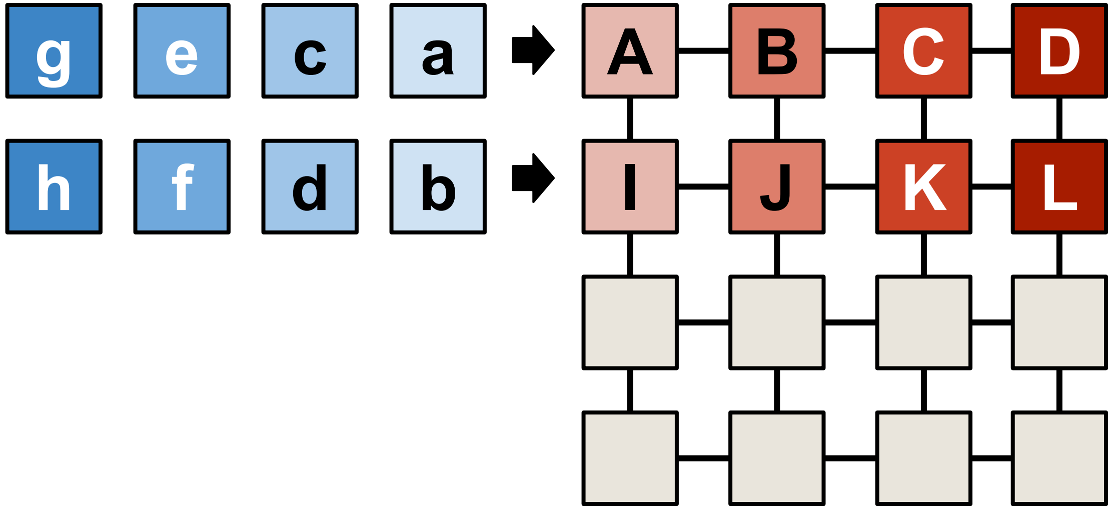
- **Reduction network allows SIGMA to reduce variable sized dot products**
- **Addresses sparsity and irregularity**
- **O(logN) reduction**

Irregular GEMMs on SIGMA



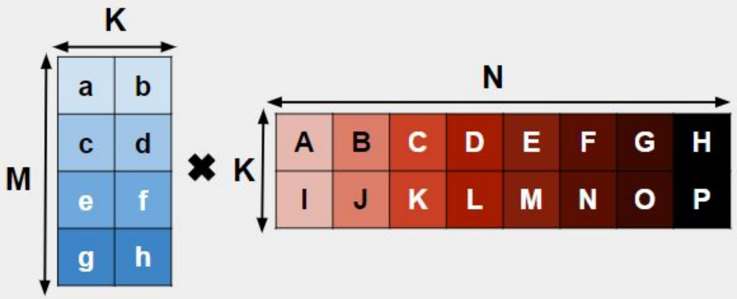
Set up the stationary values.

***** Assuming MK matrix is streaming and KN matrix is stationary. (aka weight stationary)***



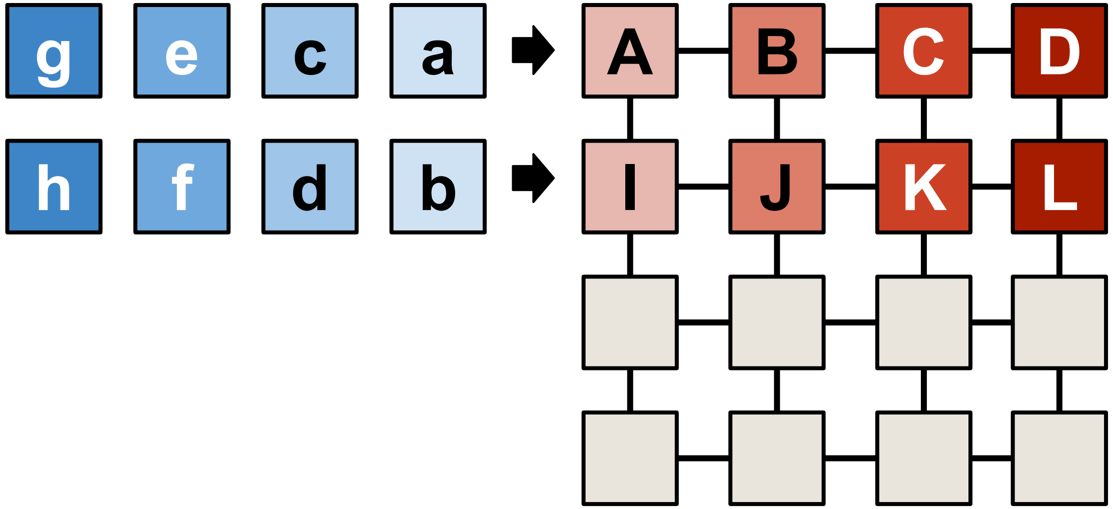
4 x 4 Systolic Array

Irregular GEMMs on SIGMA

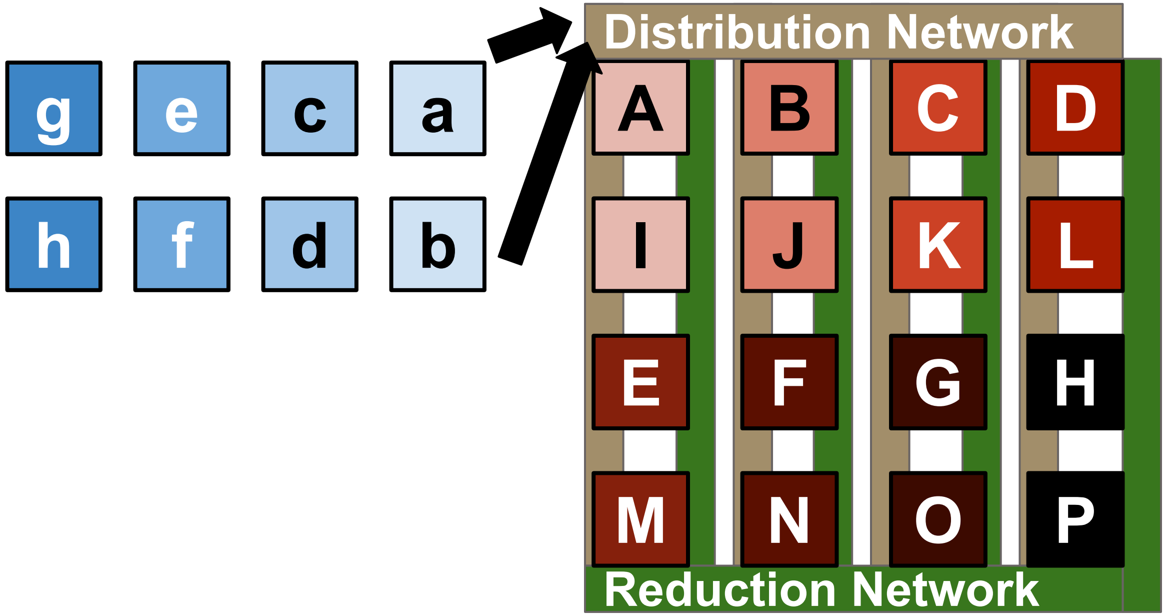


Set up the stationary values.

***** Assuming MK matrix is streaming and KN matrix is stationary. (aka weight stationary)***

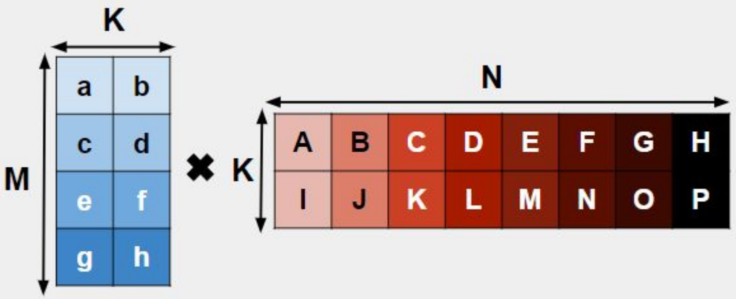


4 x 4 Systolic Array



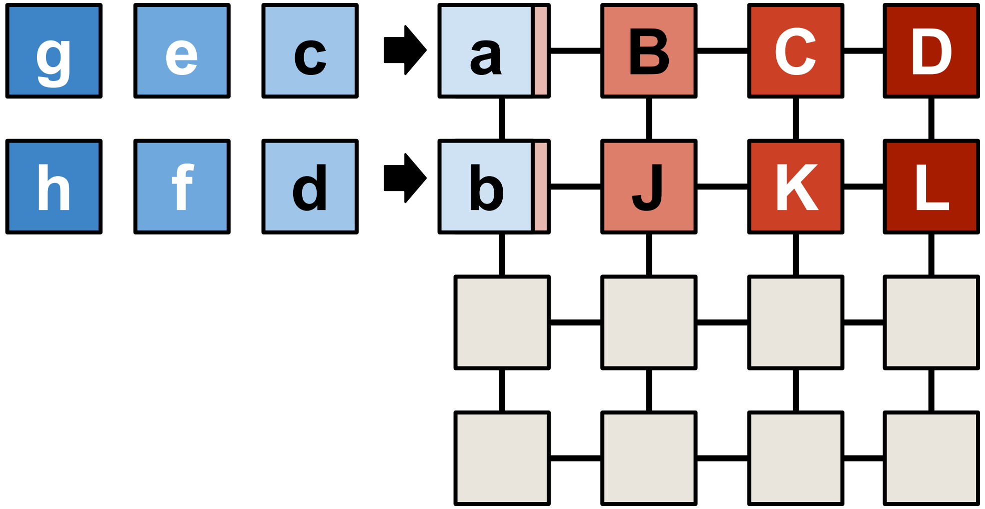
16 PE SIGMA

Irregular GEMMs on SIGMA

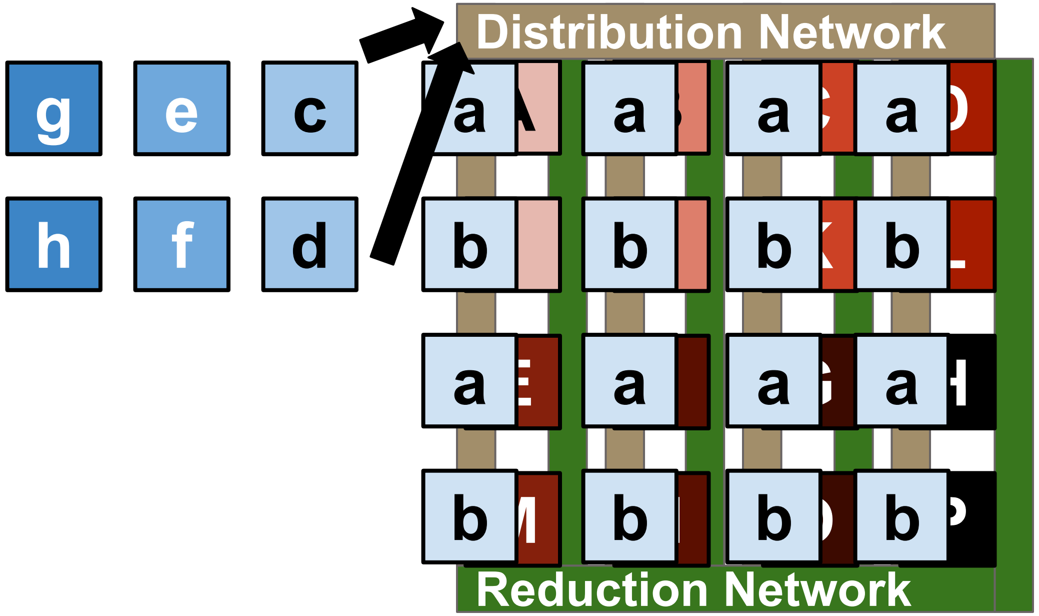


Next cycle: Multicast first row of MK to the corresponding stationary elements.

***** Assuming MK matrix is streaming and KN matrix is stationary. (aka weight stationary)***

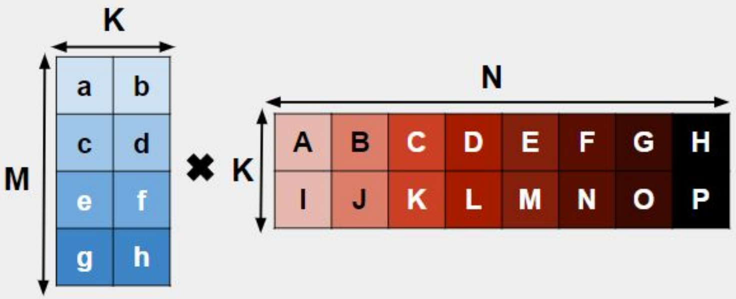


4 x 4 Systolic Array



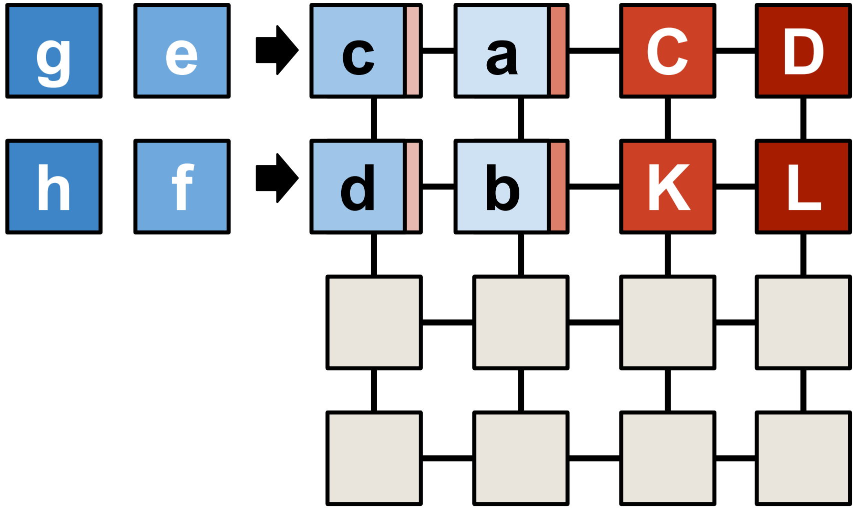
16 PE SIGMA

Irregular GEMMs on SIGMA

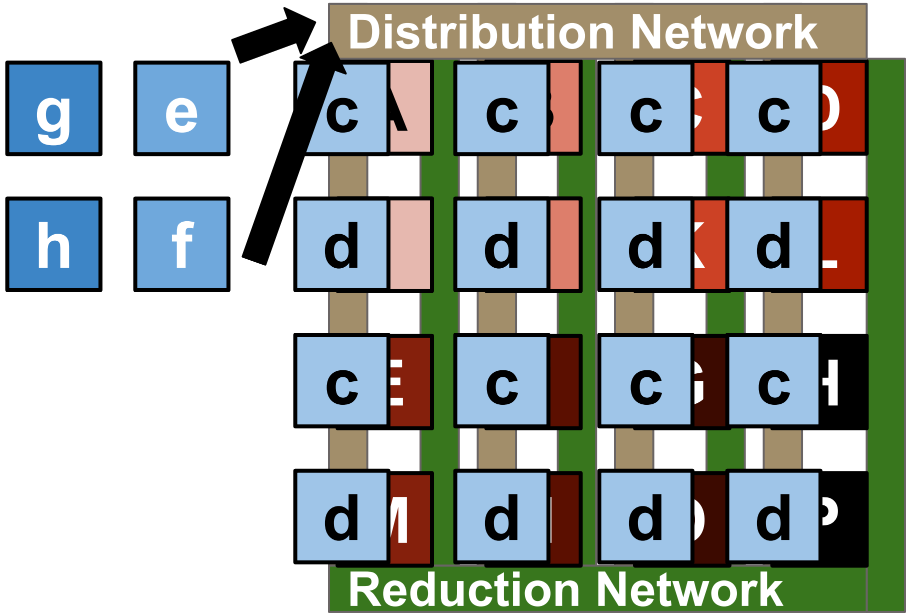


Next cycle: Multicast second row of MK to the corresponding stationary elements.

*** Assuming MK matrix is streaming and KN matrix is stationary. (aka weight stationary)*

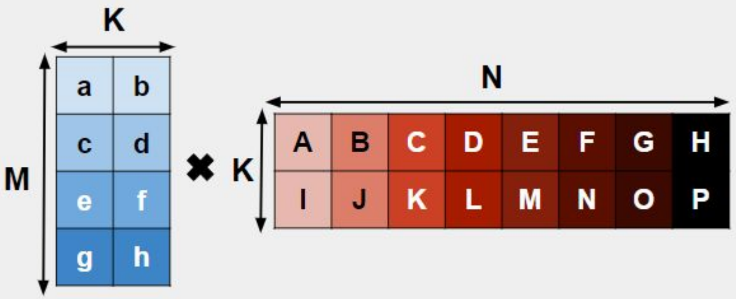


4 x 4 Systolic Array



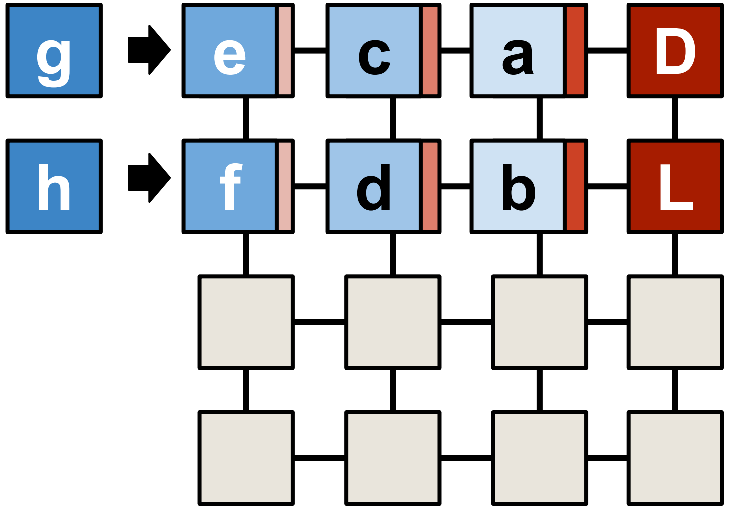
16 PE SIGMA

Irregular GEMMs on SIGMA

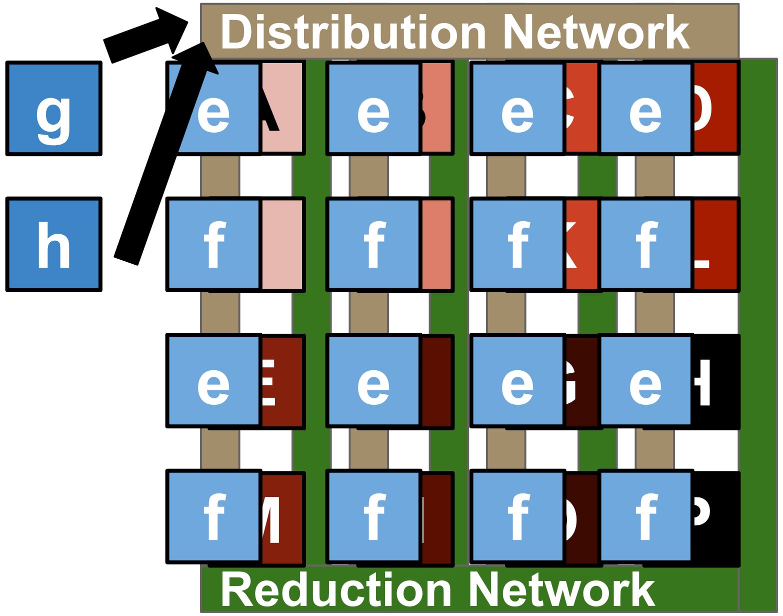


Next cycle: Multicast third row of MK to the corresponding stationary elements.

***** Assuming MK matrix is streaming and KN matrix is stationary. (aka weight stationary)***

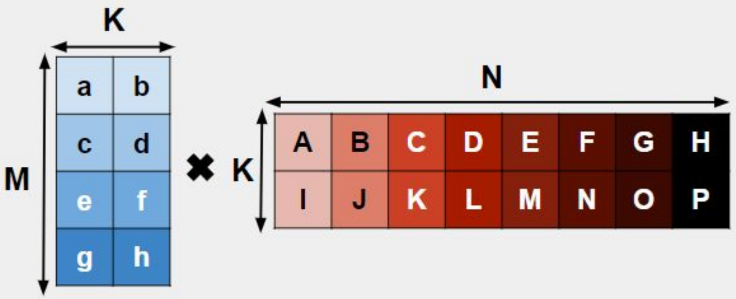


4 x 4 Systolic Array



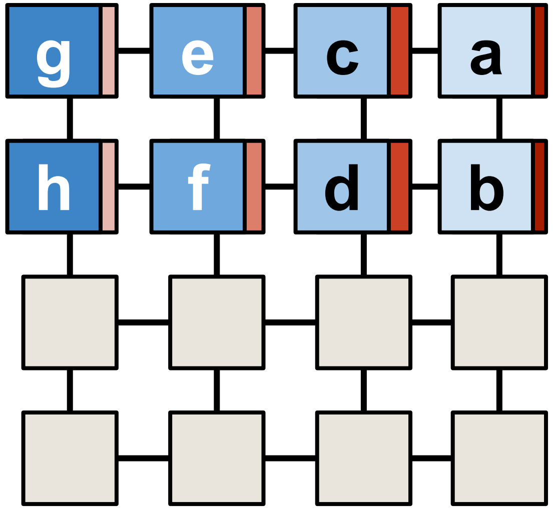
16 PE SIGMA

Irregular GEMMs on SIGMA

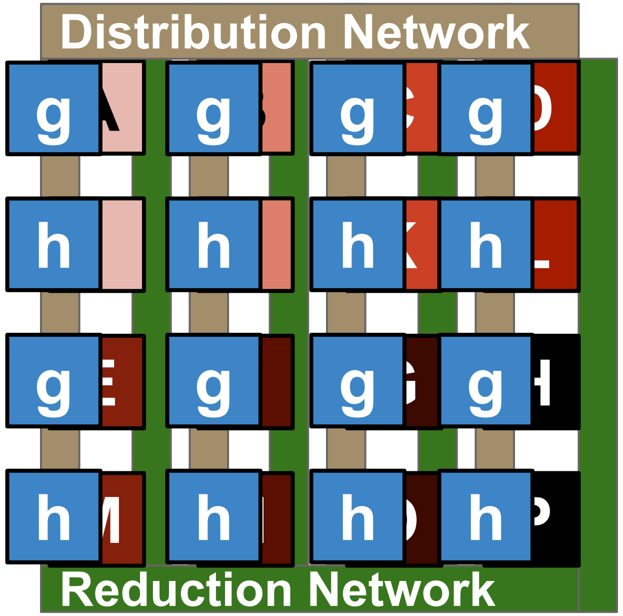


Next cycle: Multicast fourth row of MK to the corresponding stationary elements.

***** Assuming MK matrix is streaming and KN matrix is stationary. (aka weight stationary)***

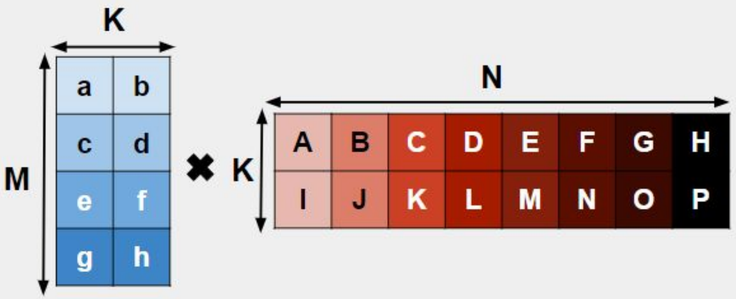


4 x 4 Systolic Array



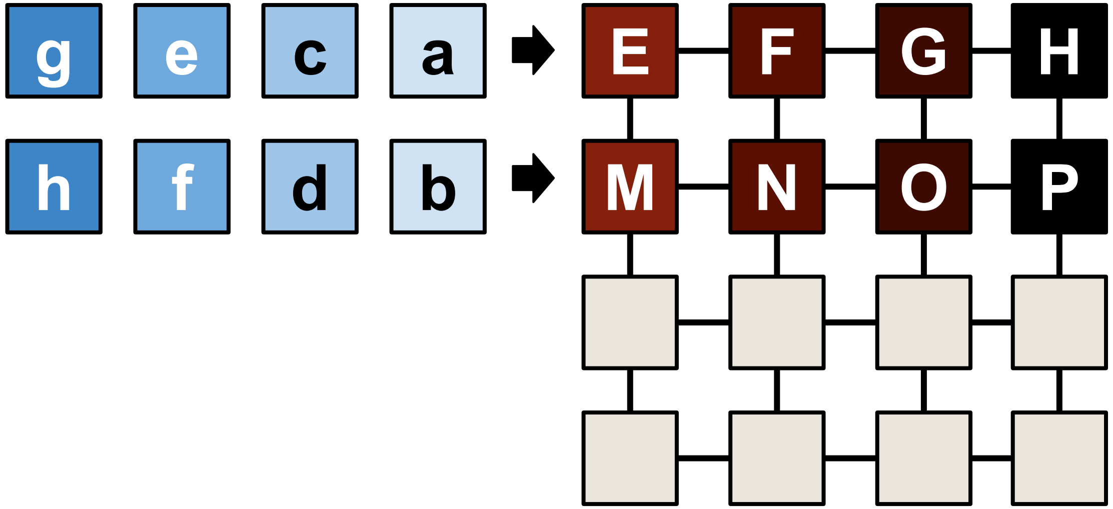
16 PE SIGMA

Irregular GEMMs on SIGMA

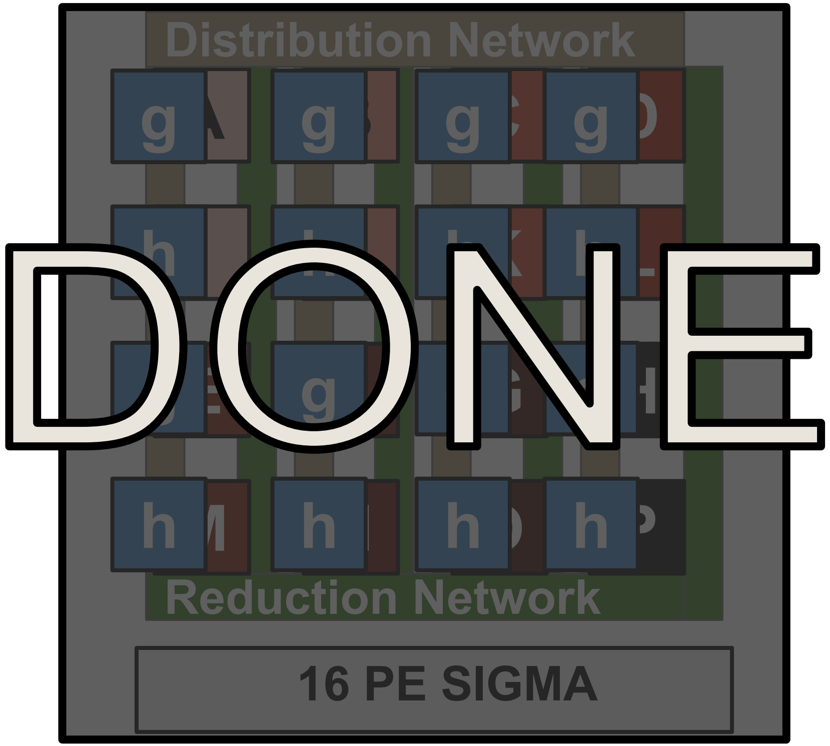


After accumulation, SIGMA is done. However, the systolic array has to map the other side of the stationary matrix and stream in the MK matrix again (referred to as folding).

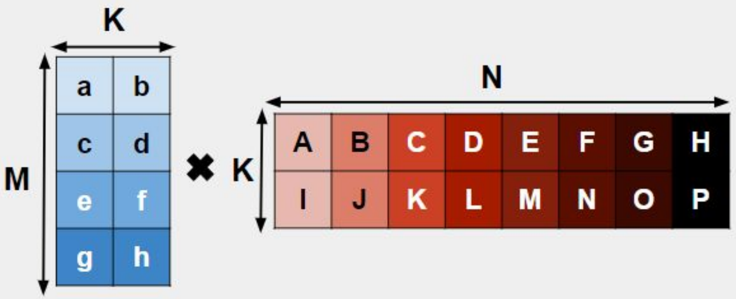
***** Assuming MK matrix is streaming and KN matrix is stationary. (aka weight stationary)***



4 x 4 Systolic Array



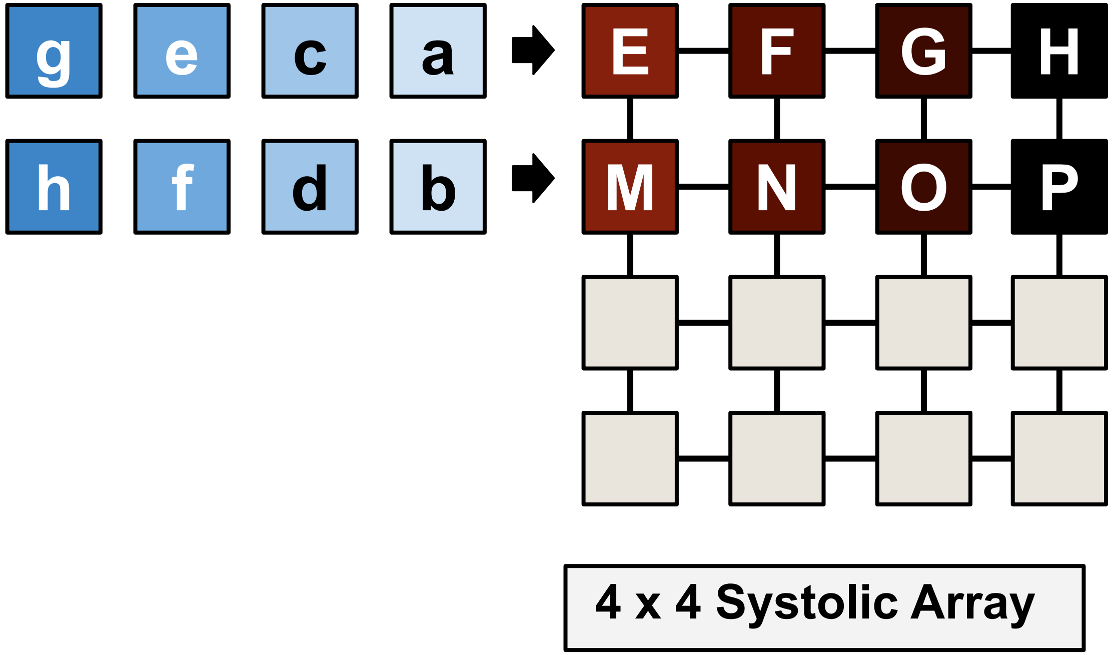
Irregular GEMMs on SIGMA

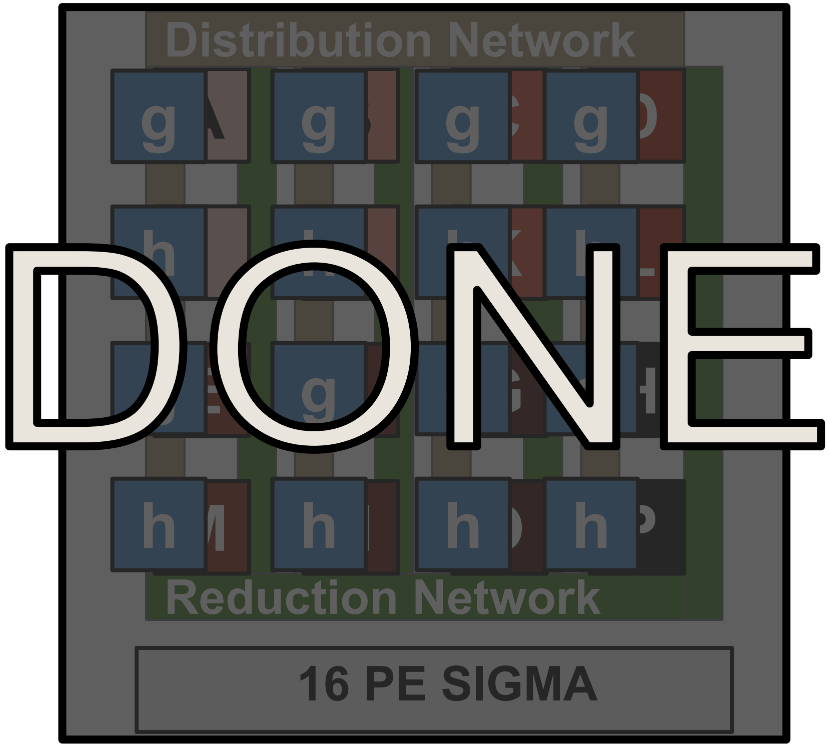


SIGMA reduces the number of folds, which then reduces the number of memory references on the streaming matrix.

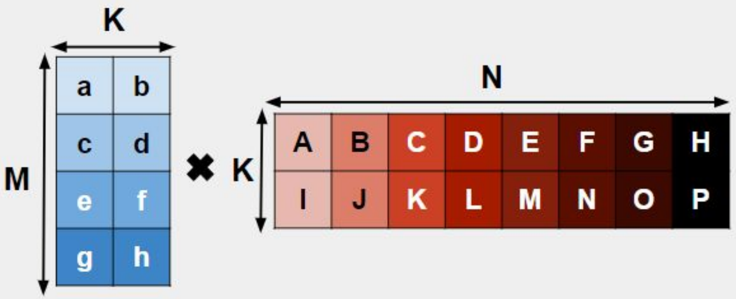
***** Assuming MK***

stationary)



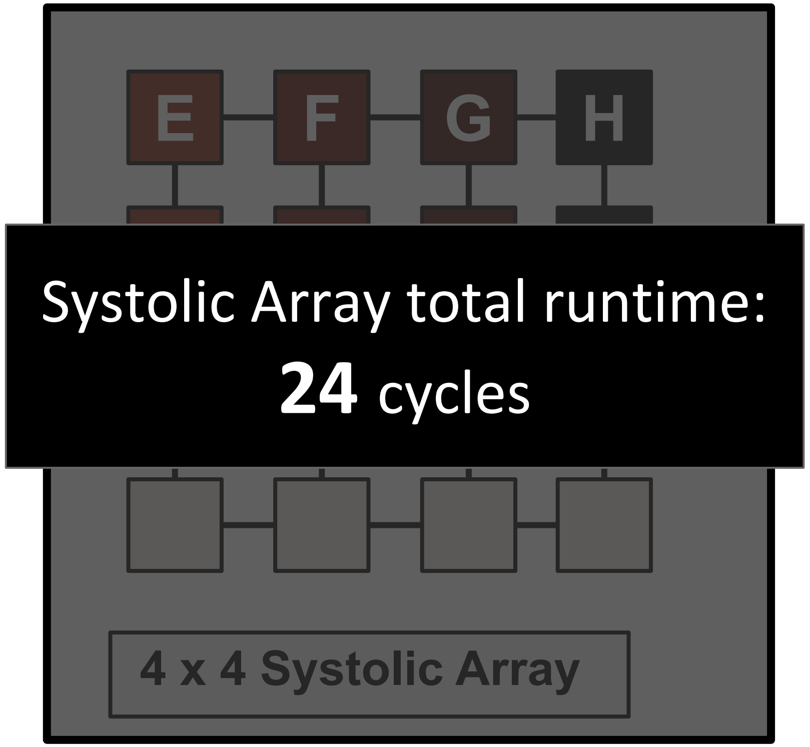


Irregular GEMMs on SIGMA

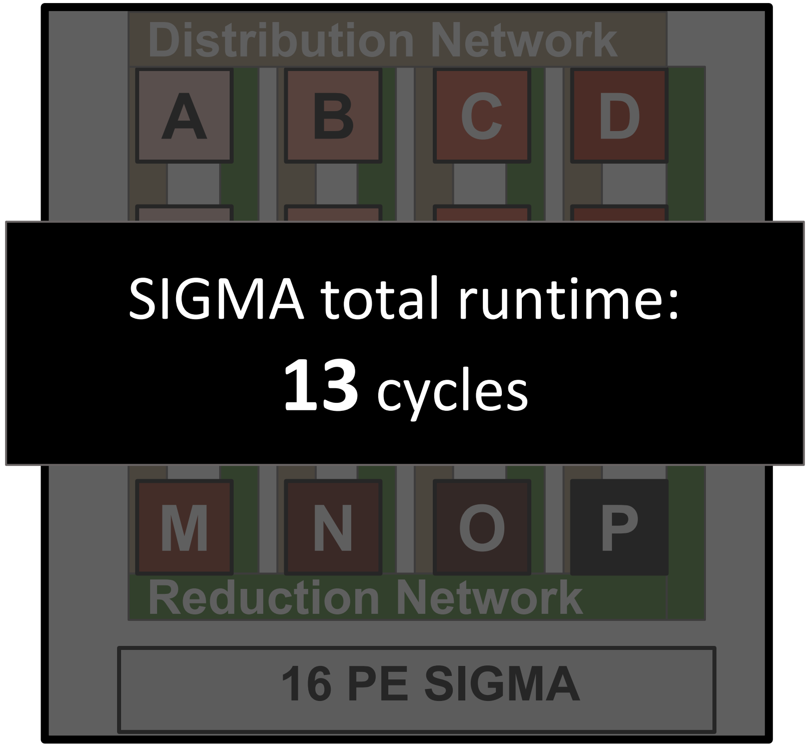


Final cycle count.

***** Assuming MK matrix is streaming and KN matrix is stationary. (aka weight stationary)***

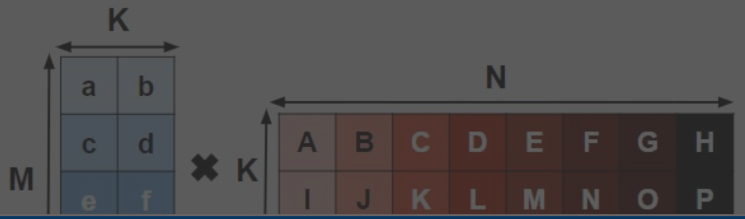


Systolic Array total runtime:
24 cycles



SIGMA total runtime:
13 cycles

Irregular GEMMs on SIGMA

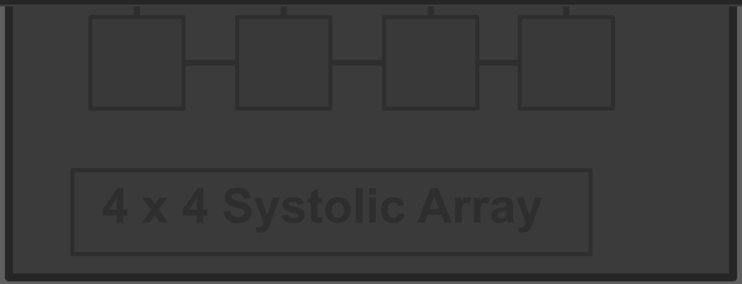


Final cycle count.

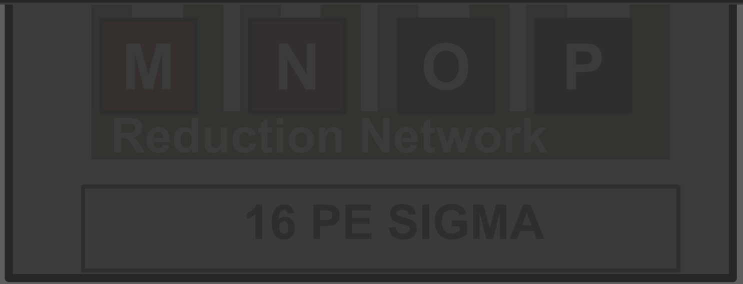
***** Assuming MK matrix is streaming and KN matrix is stationary. (aka***

SIGMA maximizes PE utilization with its flexible interconnects for irregular GEMMs.

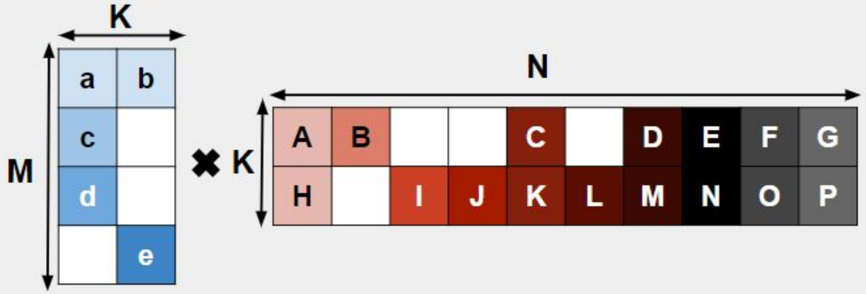
24 cycles



13 cycles

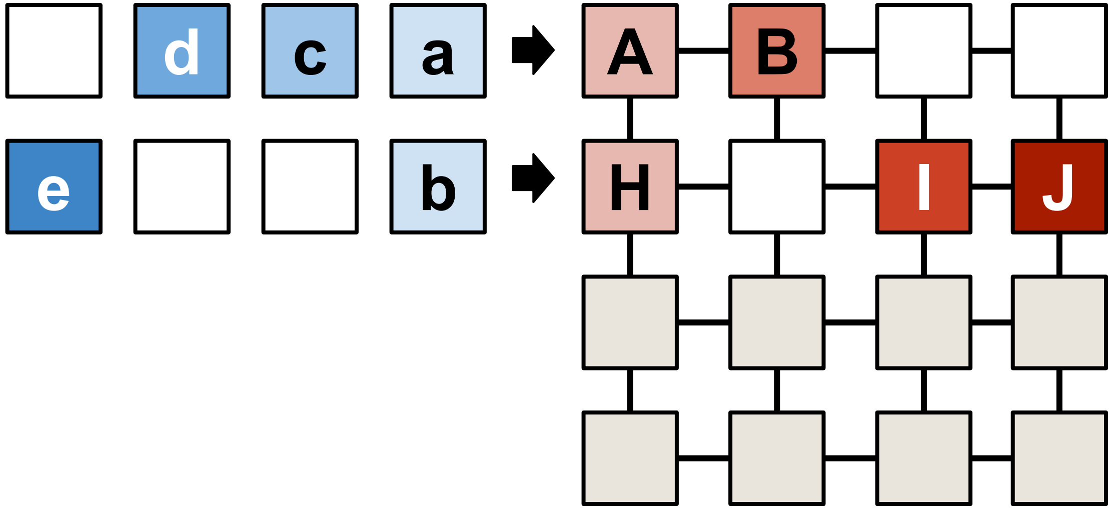


Sparse Irregular GEMMs on SIGMA



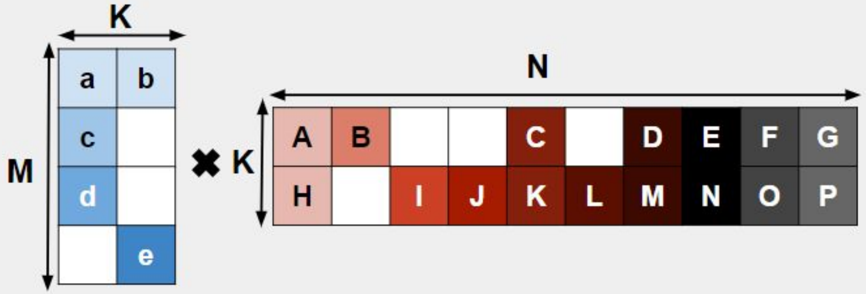
Set up the stationary values.

***** Assuming MK matrix is streaming and KN matrix is stationary. (aka weight stationary)***



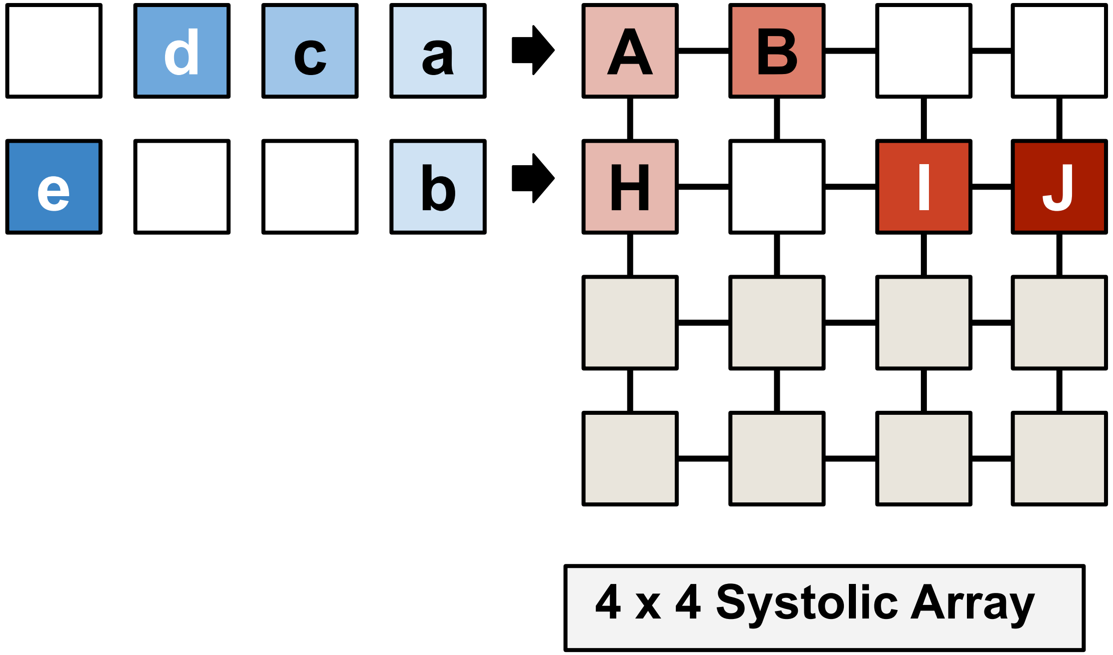
4 x 4 Systolic Array

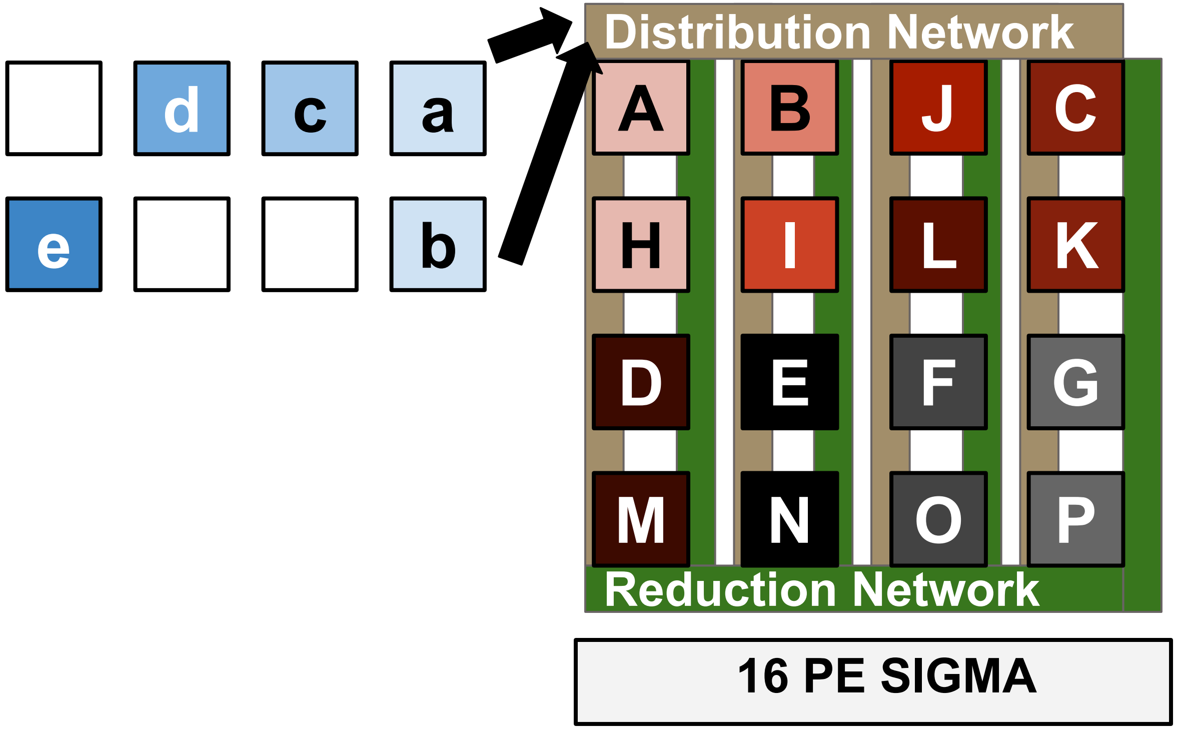
Sparse Irregular GEMMs on SIGMA



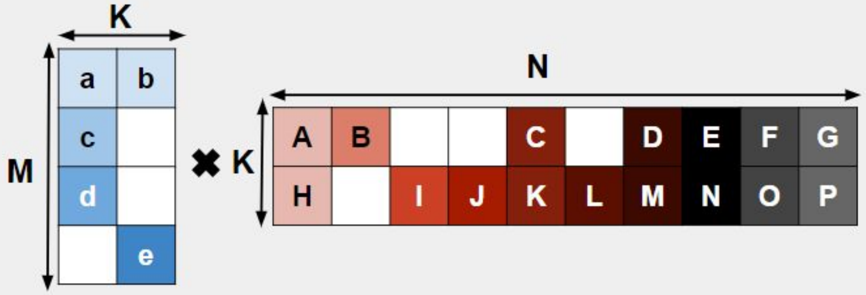
Set up the stationary values.

***** Assuming MK matrix is streaming and KN matrix is stationary. (aka weight stationary)***



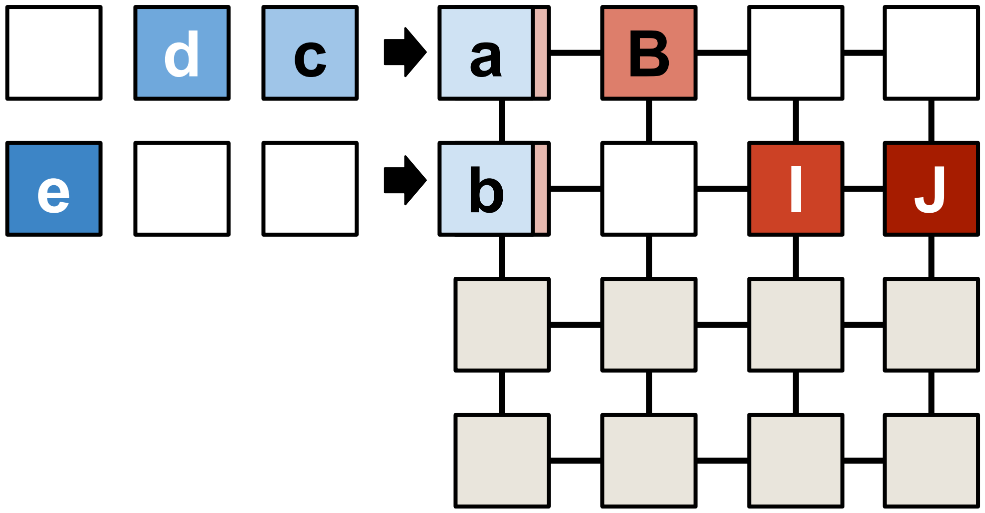


Sparse Irregular GEMMs on SIGMA

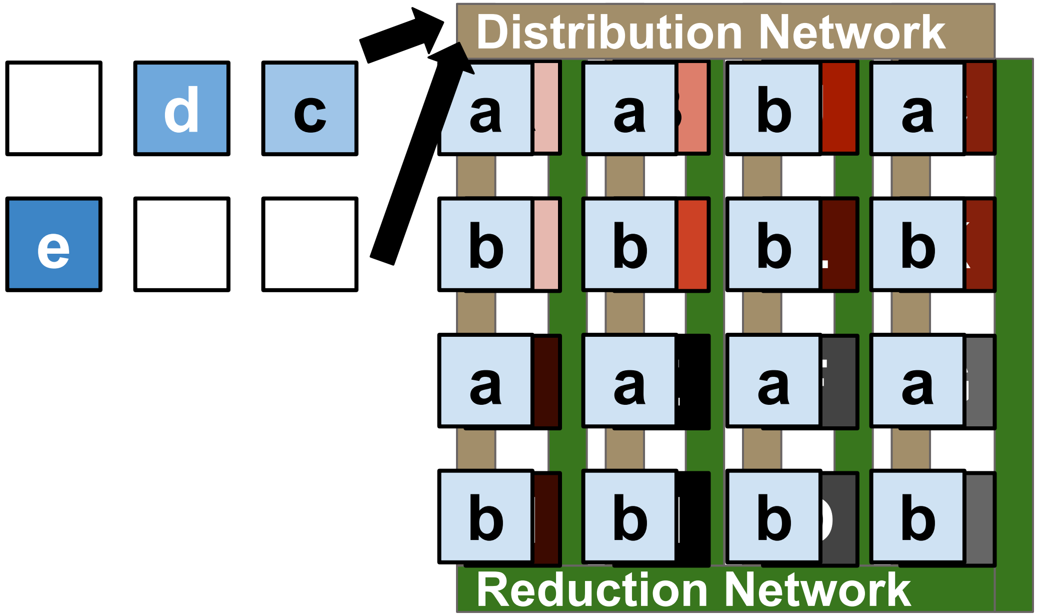


Next cycle: Multicast first row of MK to the corresponding stationary elements.

***** Assuming MK matrix is streaming and KN matrix is stationary. (aka weight stationary)***

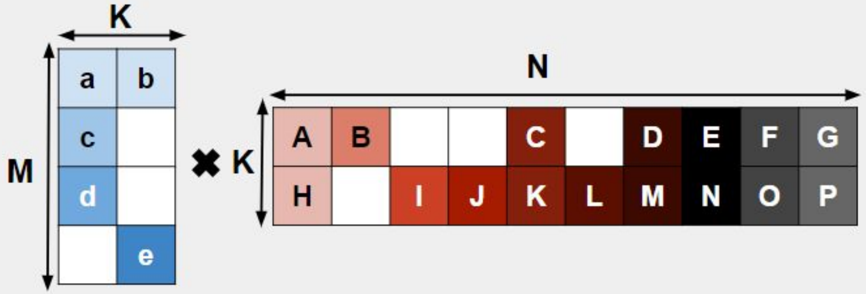


4 x 4 Systolic Array



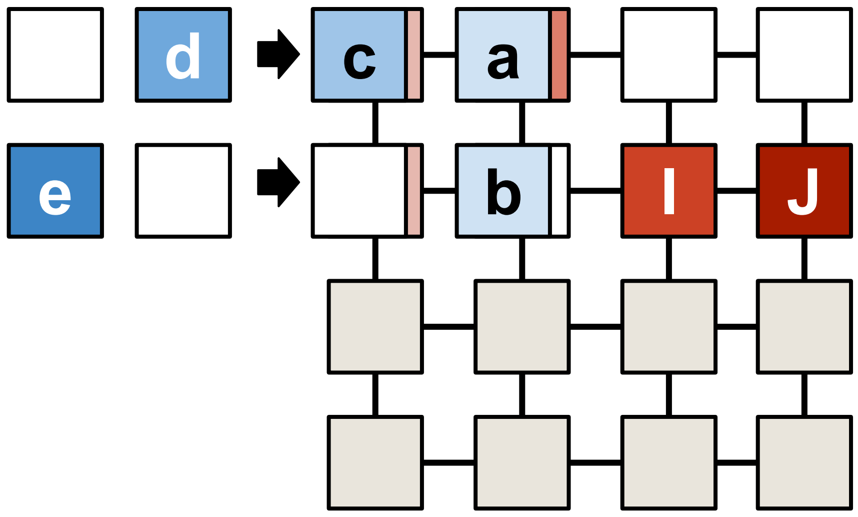
16 PE SIGMA

Sparse Irregular GEMMs on SIGMA

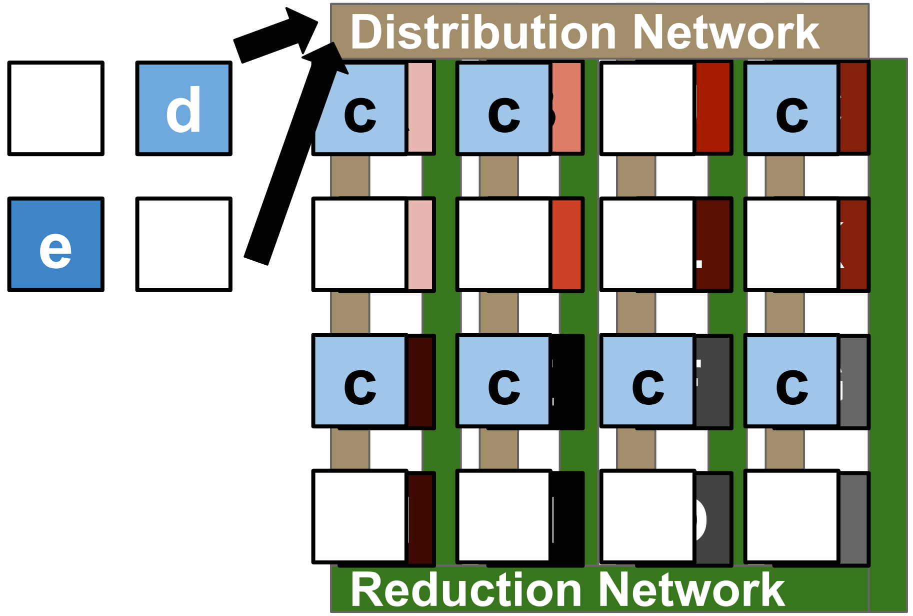


Next cycle: Multicast second row of MK to the corresponding stationary elements.

***** Assuming MK matrix is streaming and KN matrix is stationary. (aka weight stationary)***

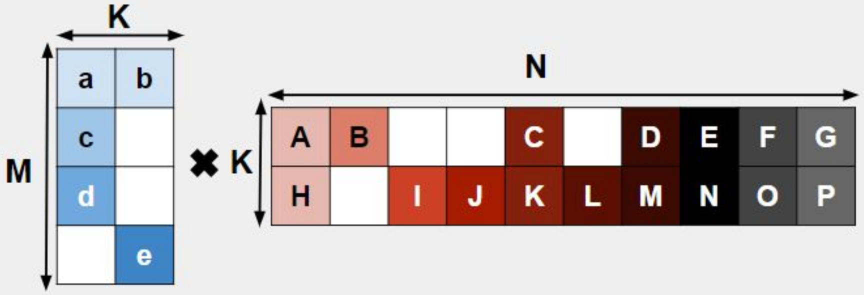


4 x 4 Systolic Array



16 PE SIGMA

Sparse Irregular GEMMs on SIGMA

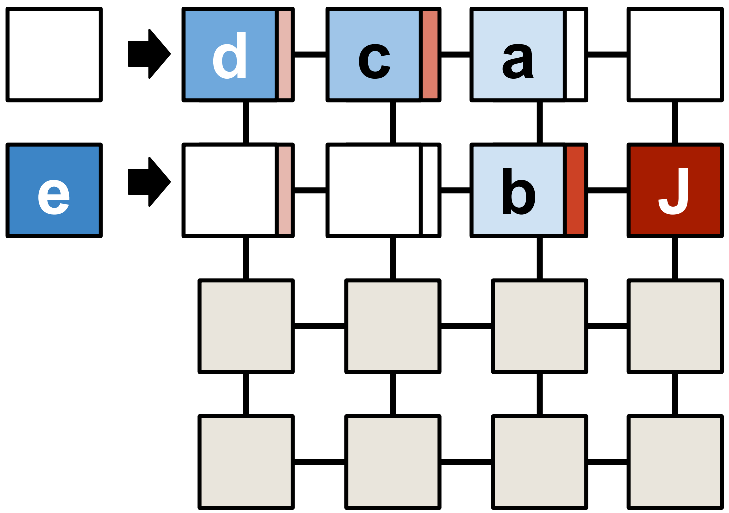


a	b
c	
d	
	e

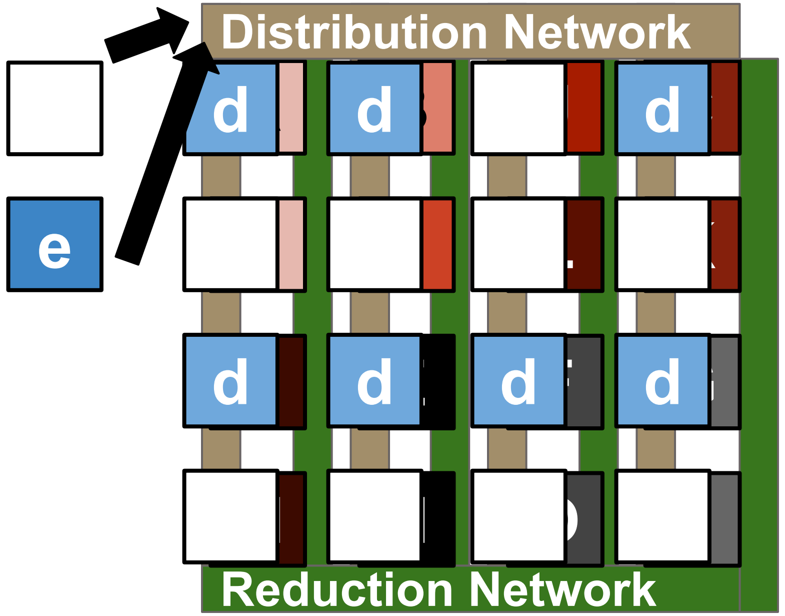
A	B			C		D	E	F	G
H		I	J	K	L	M	N	O	P

Next cycle: Multicast third row of MK to the corresponding stationary elements.

***** Assuming MK matrix is streaming and KN matrix is stationary. (aka weight stationary)***

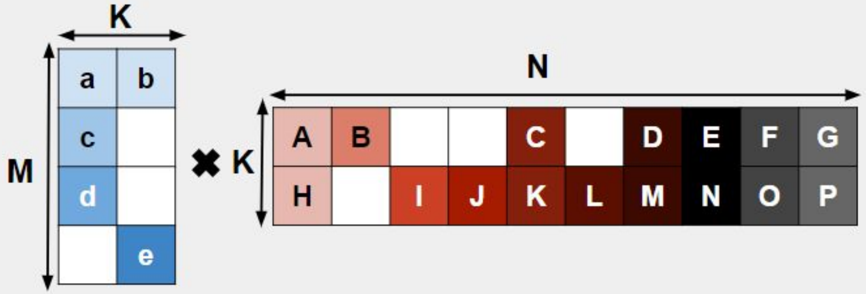


4 x 4 Systolic Array



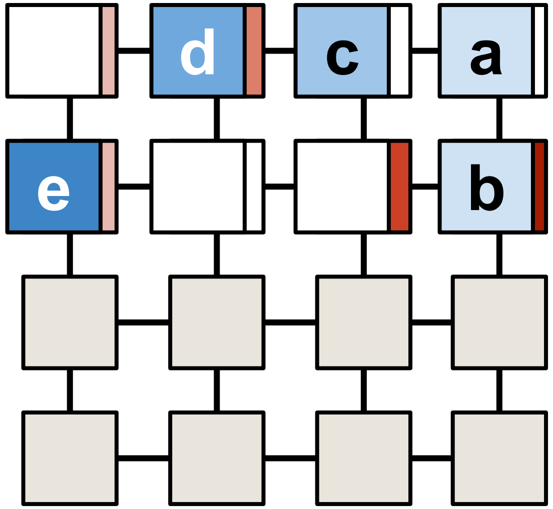
16 PE SIGMA

Sparse Irregular GEMMs on SIGMA

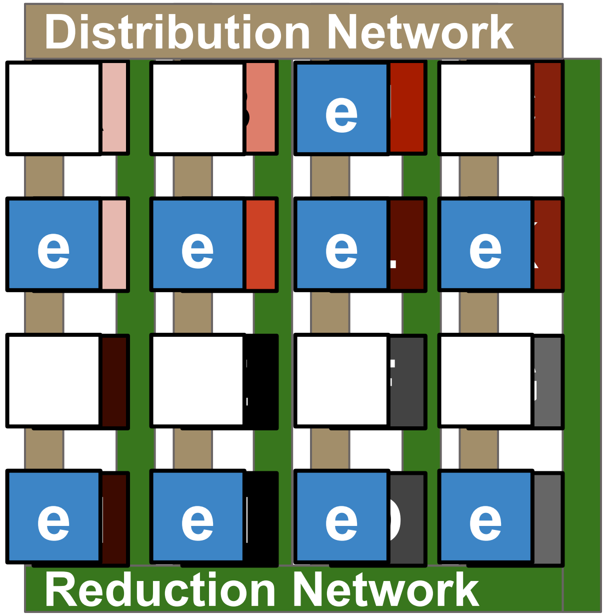


Next cycle: Multicast fourth row of MK to the corresponding stationary elements.

***** Assuming MK matrix is streaming and KN matrix is stationary. (aka weight stationary)***

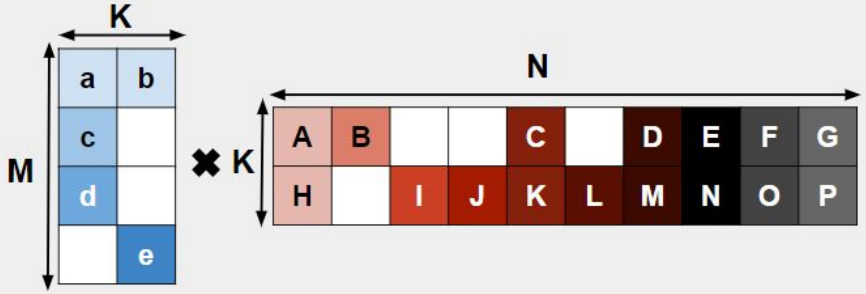


4 x 4 Systolic Array



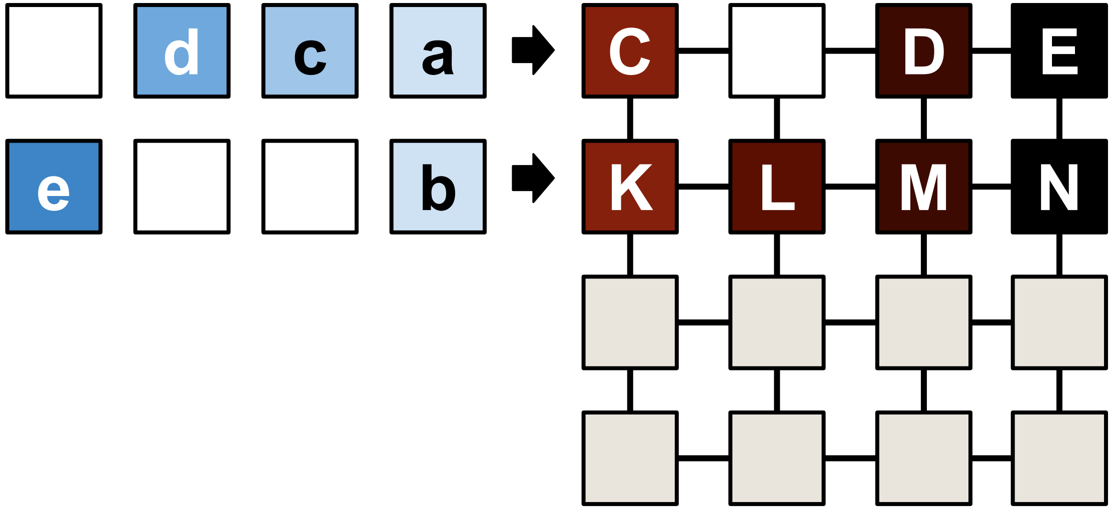
16 PE SIGMA

Sparse Irregular GEMMs on SIGMA

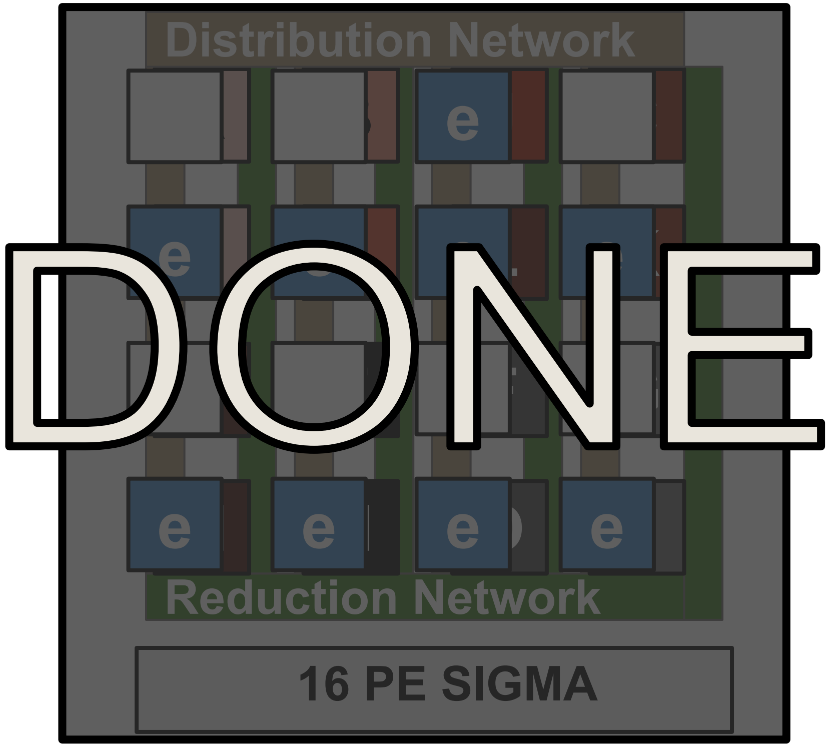


After accumulation, SIGMA is done. However, the systolic array has to map the other part of the stationary matrix and stream in the MK matrix again (referred to as folding).

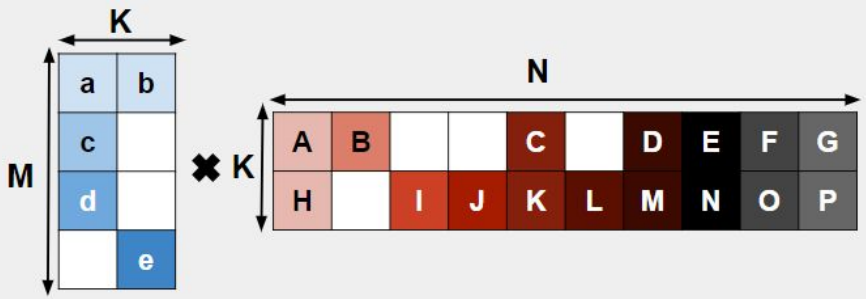
*** Assuming MK matrix is streaming and KN matrix is stationary. (aka weight stationary)*



4 x 4 Systolic Array

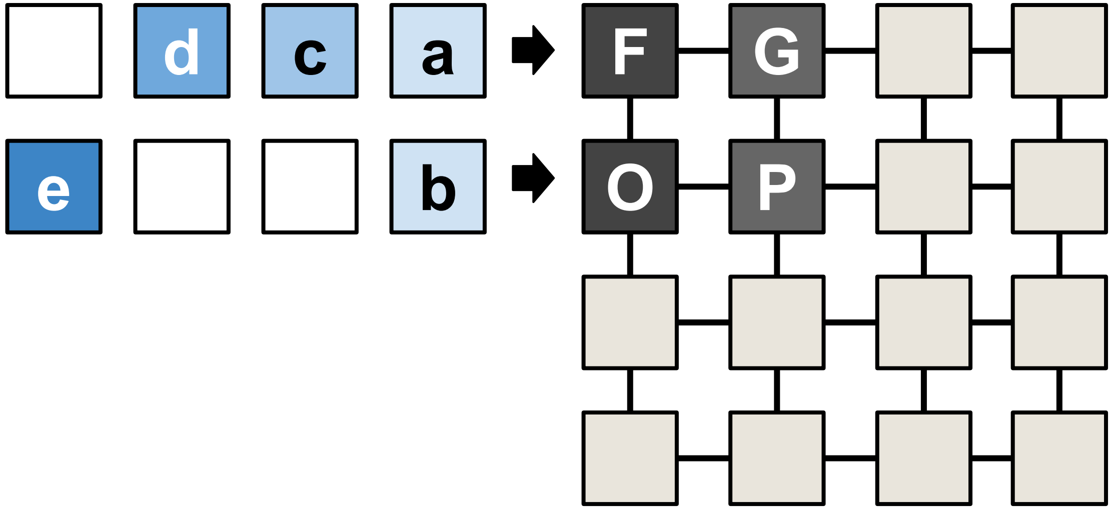


Sparse Irregular GEMMs on SIGMA

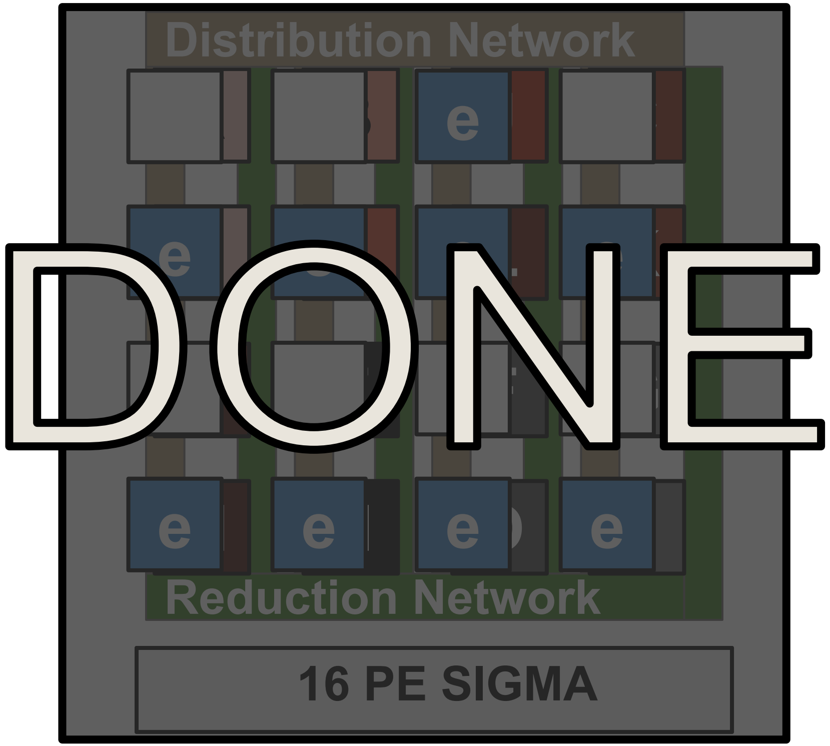


Again, the systolic array has to map another part of the stationary matrix and stream MK again.

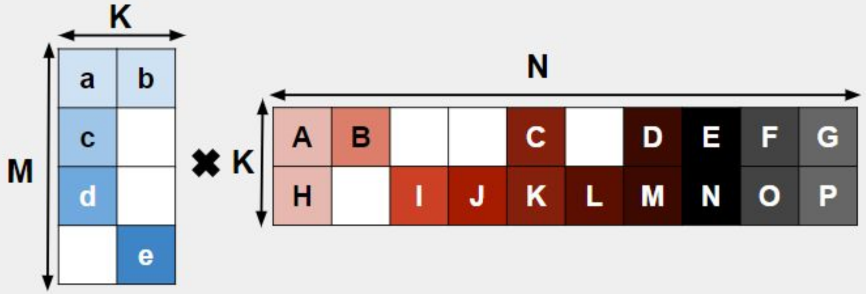
*** Assuming MK matrix is streaming and KN matrix is stationary. (aka weight stationary)*



4 x 4 Systolic Array

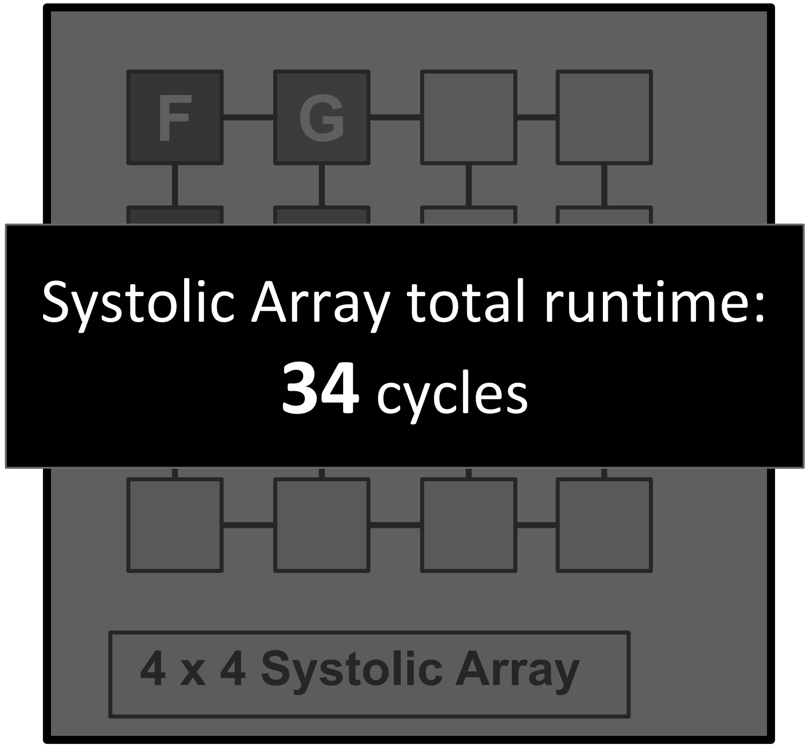


Sparse Irregular GEMMs on SIGMA



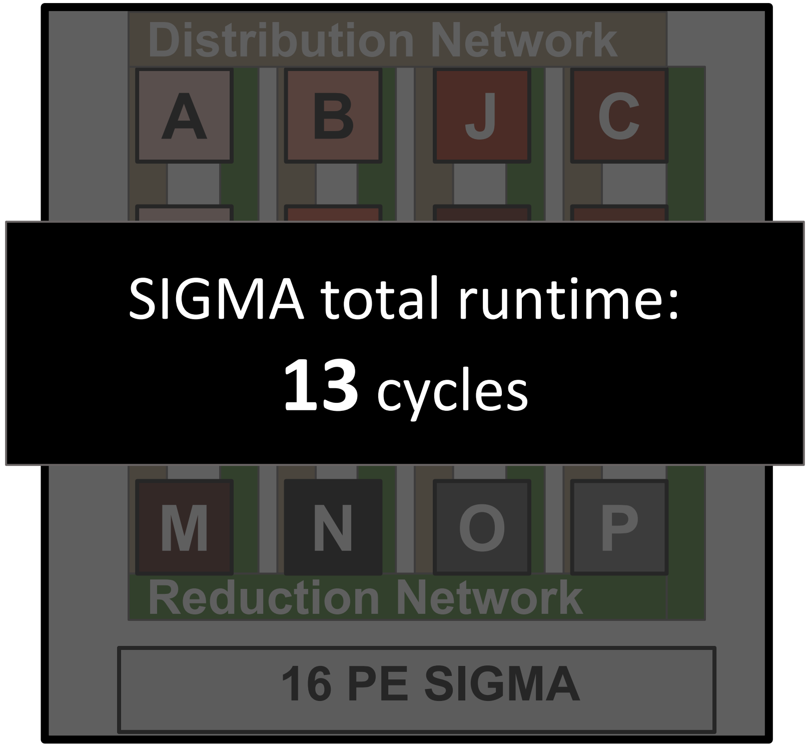
Final cycle count.

***** Assuming MK matrix is streaming and KN matrix is stationary. (aka weight stationary)***



Systolic Array total runtime:

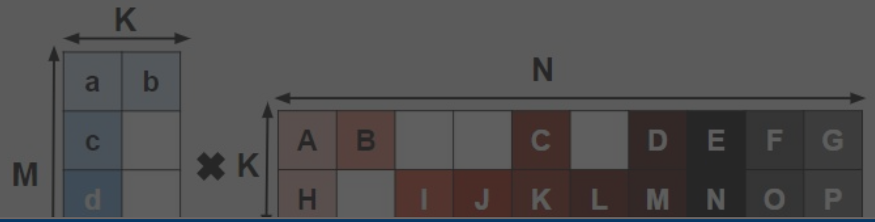
34 cycles



SIGMA total runtime:

13 cycles

Sparse Irregular GEMMs on SIGMA

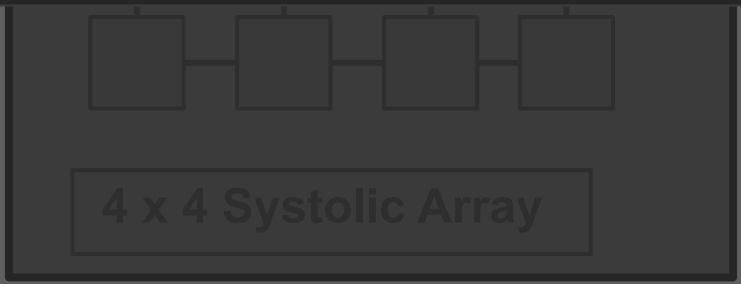


Final cycle count.

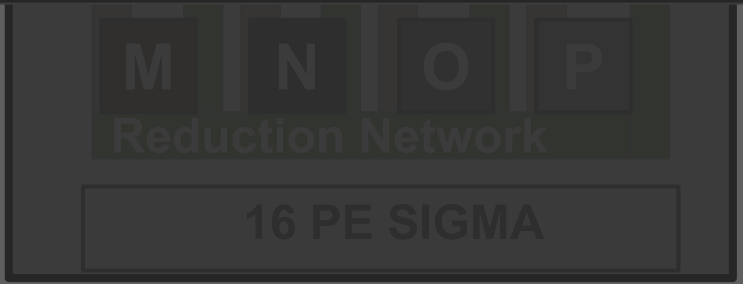
**** Assuming MK matrix is streaming and KN matrix is stationary. (aka*

SIGMA maps only nonzeros stationary; therefore, reduces the number of folds needed.

34 cycles



13 cycles



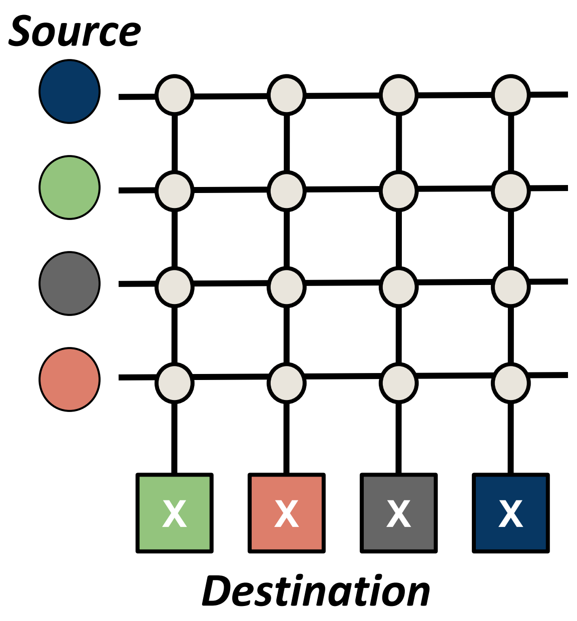
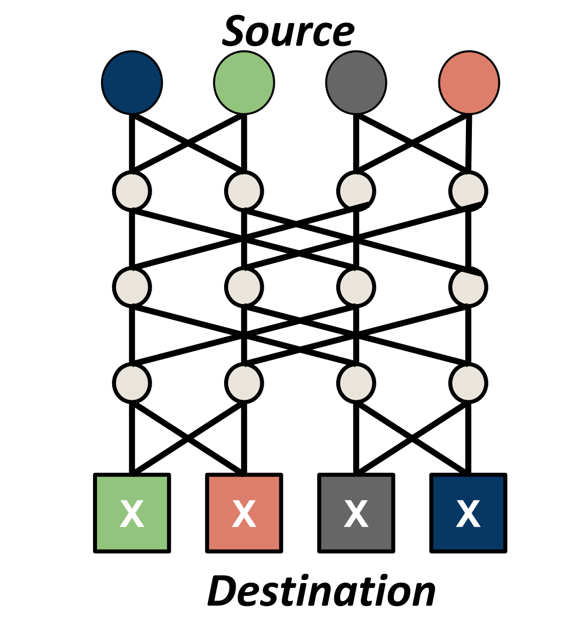
Outline

- Motivation
  - GEMMs in Deep Learning
  - Utilization on TPU (Systolic Array)
- Accelerator Requirements
- **SIGMA**
  - **Interconnects Implementations**
  - Full System Design
- Evaluation
- Conclusion

O(1) Distribution Topology

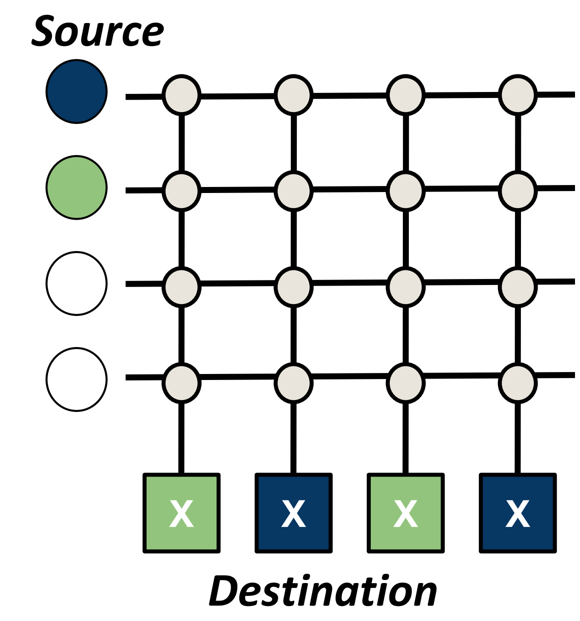
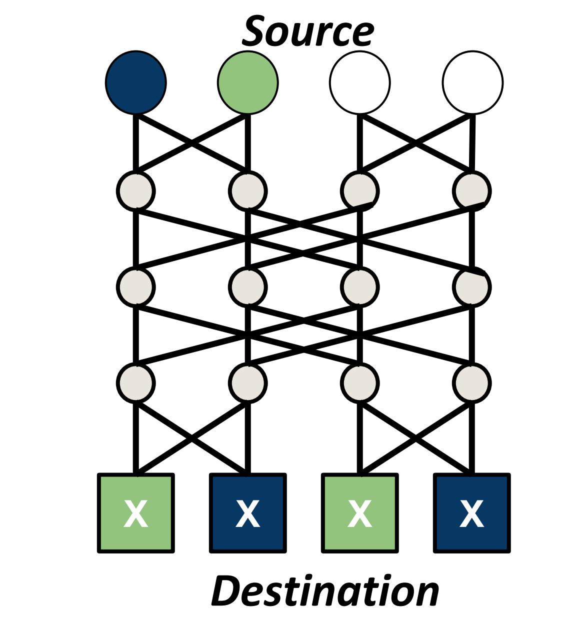
Unicast

(Loading Stationary Matrix)

Crossbar	Benes
	

Multicast

(Sending Streaming Matrix)

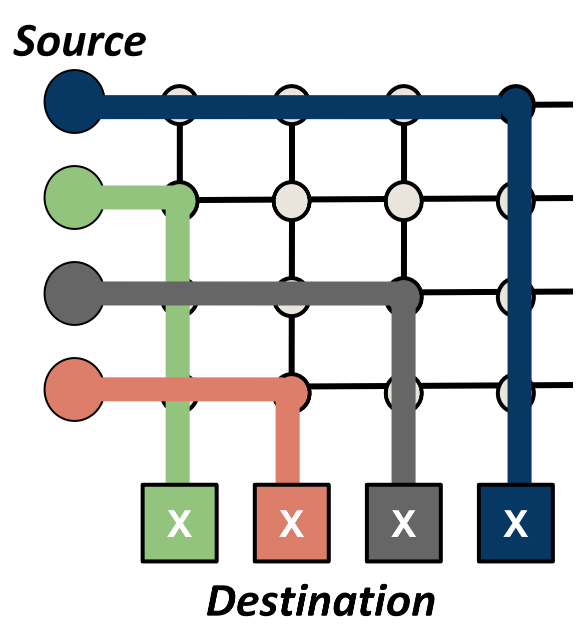
Crossbar	Benes
	

O(1) Distribution Topology

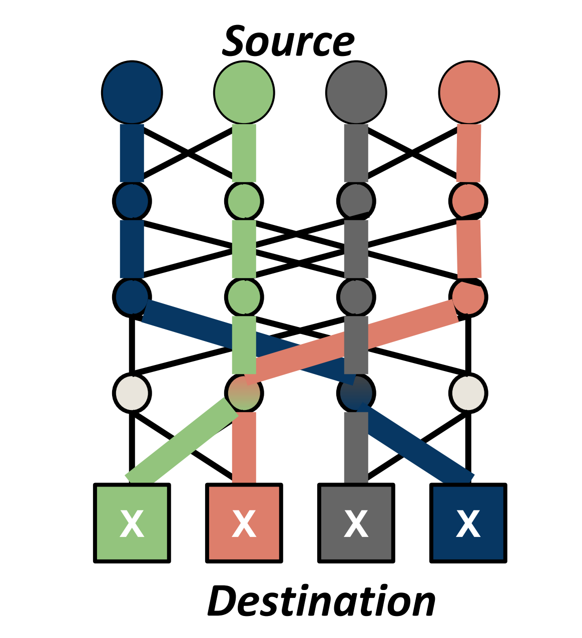
Unicast

(Loading Stationary Matrix)

Crossbar



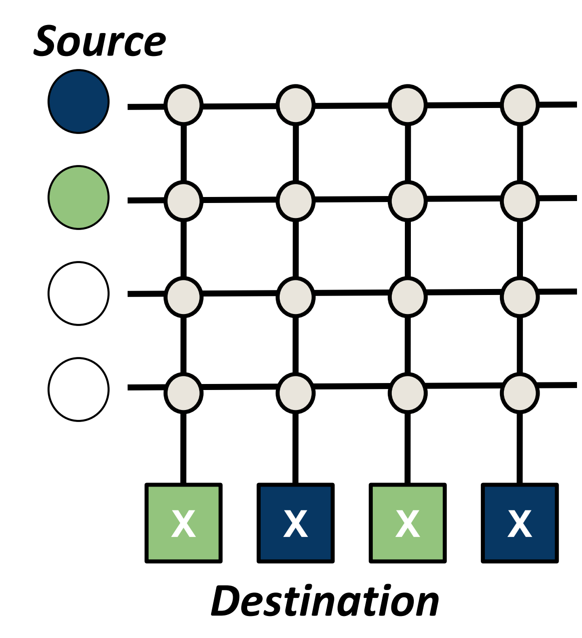
Benes



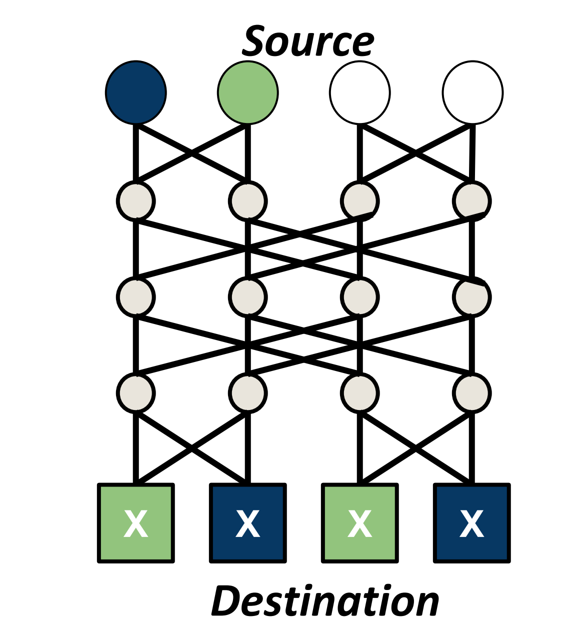
Multicast

(Sending Streaming Matrix)

Crossbar



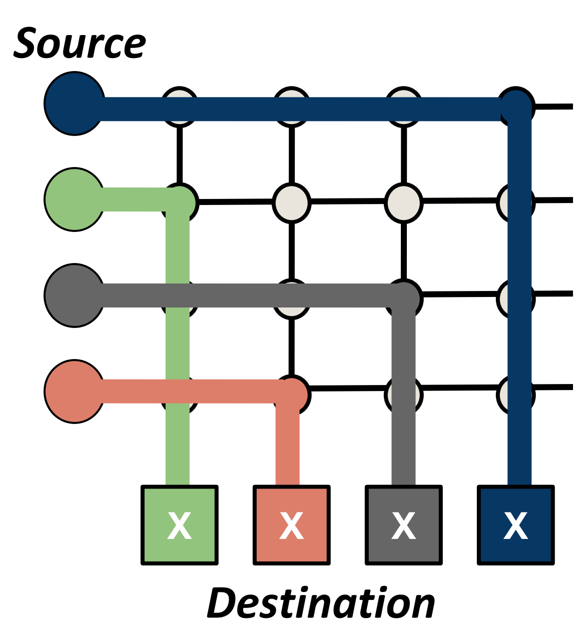
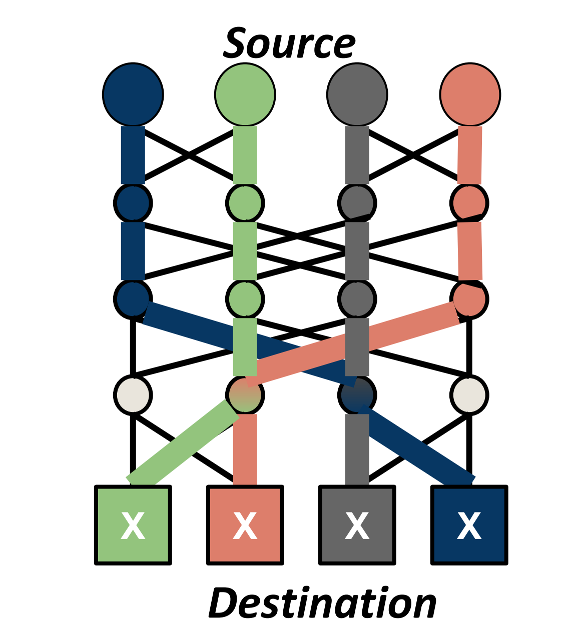
Benes



O(1) Distribution Topology

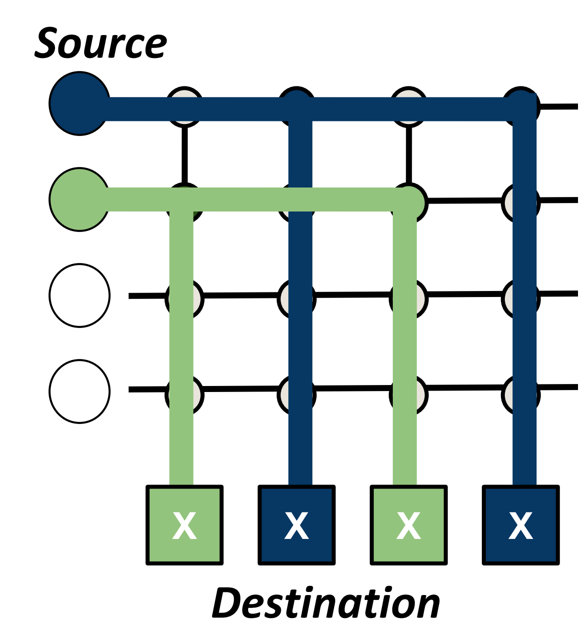
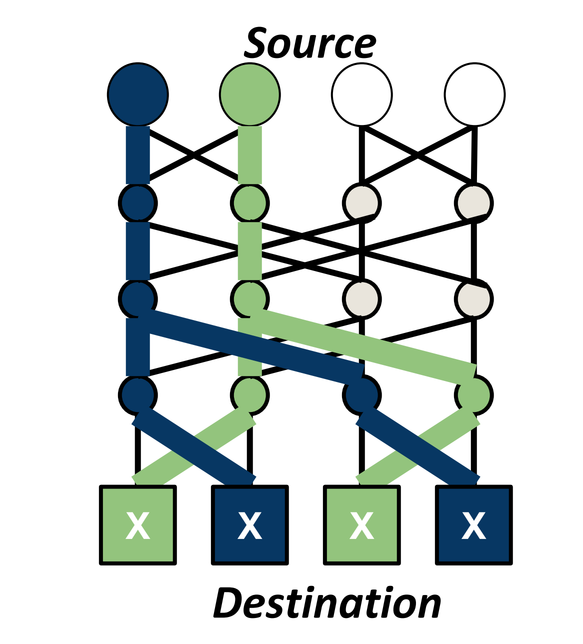
Unicast

(Loading Stationary Matrix)

Crossbar	Benes
	

Multicast

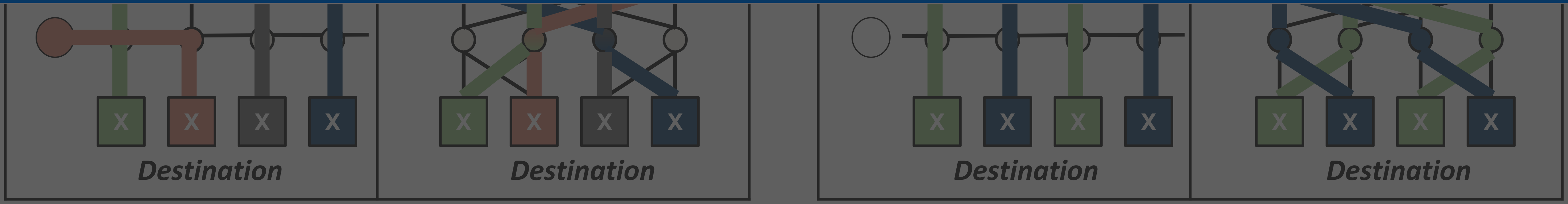
(Sending Streaming Matrix)

Crossbar	Benes
	

O(1) Distribution Topology

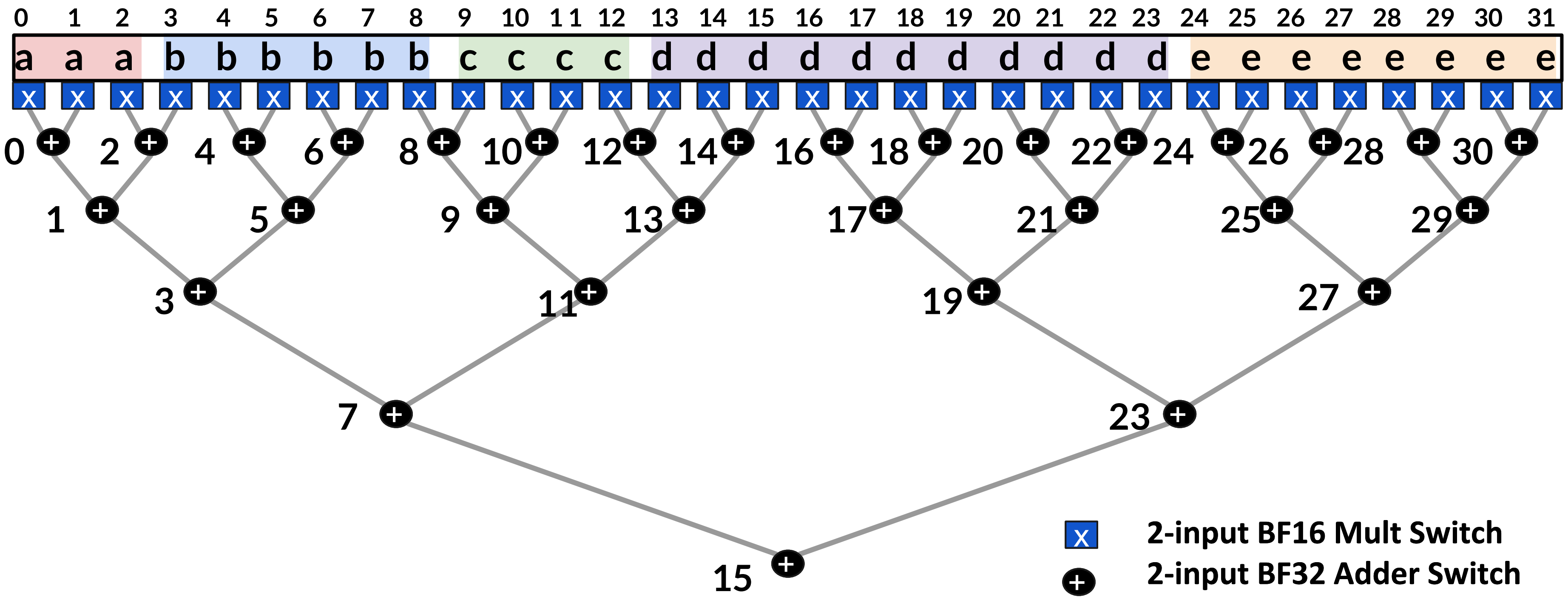
Unicast **Multicast**

SIGMA's distribution can be either a Crossbar or Benes network. We chose Benes because the number of switches scale by O(N logN).



O(logN) Reduction (Limitation of Adder Tree)

Different clusters of partial sums. ➡

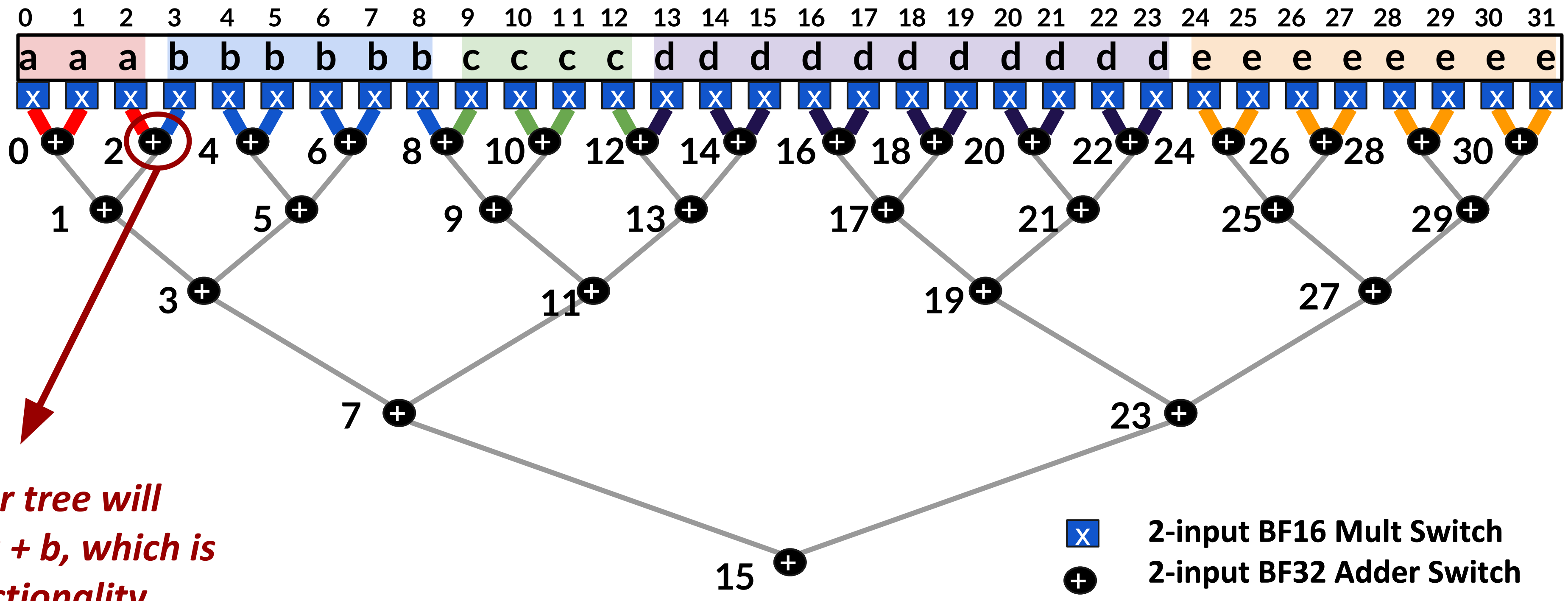


☒ **2-input BF16 Mult Switch**

⊕ **2-input BF32 Adder Switch**

O(logN) Reduction (Limitation of Adder Tree)

Different clusters of partial sums. ➡



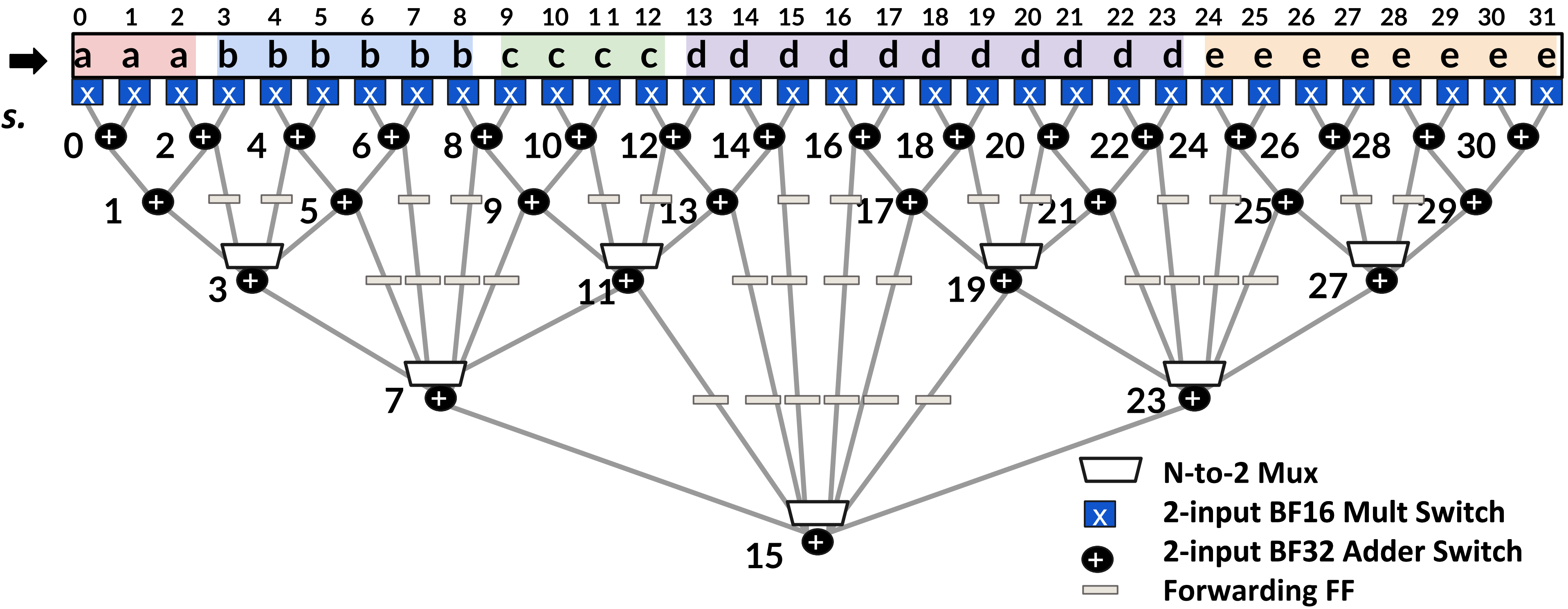
Regular adder tree will accumulate a + b, which is incorrect functionality.

X **2-input BF16 Mult Switch**

⊕ **2-input BF32 Adder Switch**

Forwarding Adder Network (FAN)

Different clusters of partial sums. ➡



N-to-2 Mux

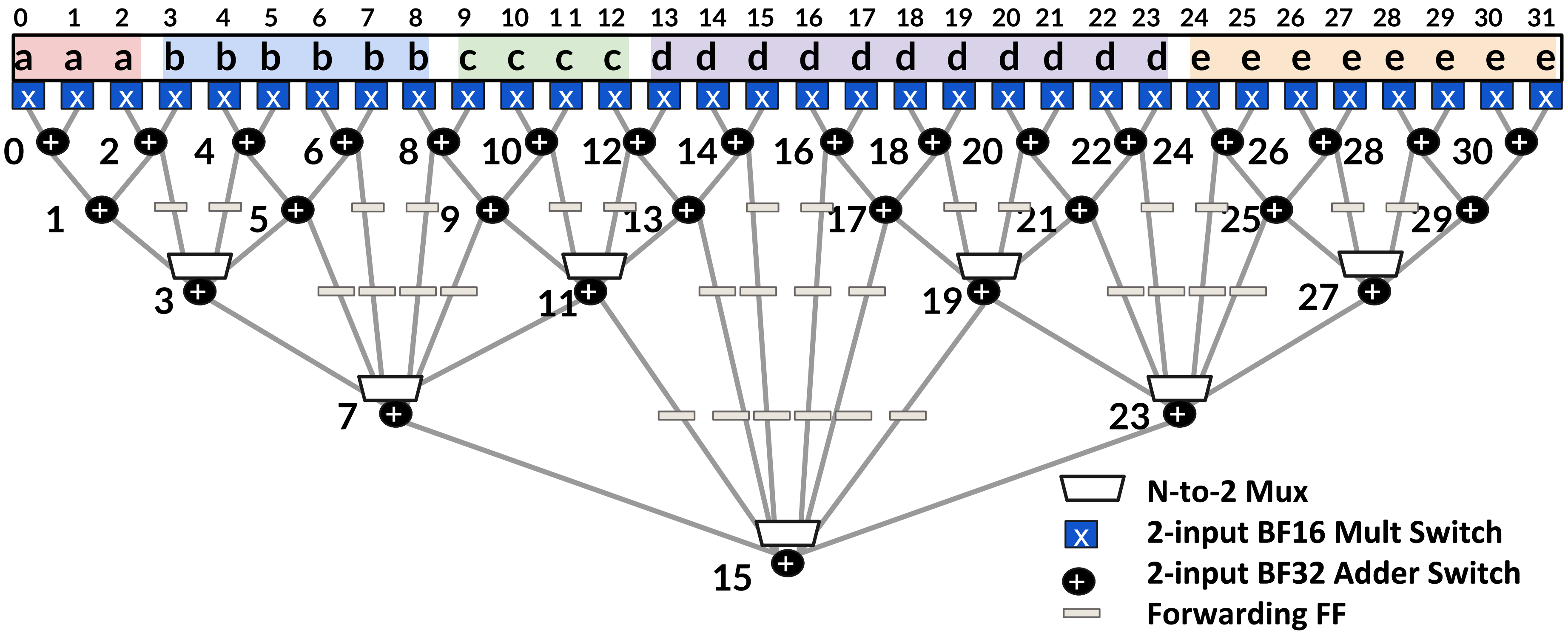
☒ 2-input BF16 Mult Switch

⊕ 2-input BF32 Adder Switch

▭ Forwarding FF

Forwarding Adder Network (FAN)

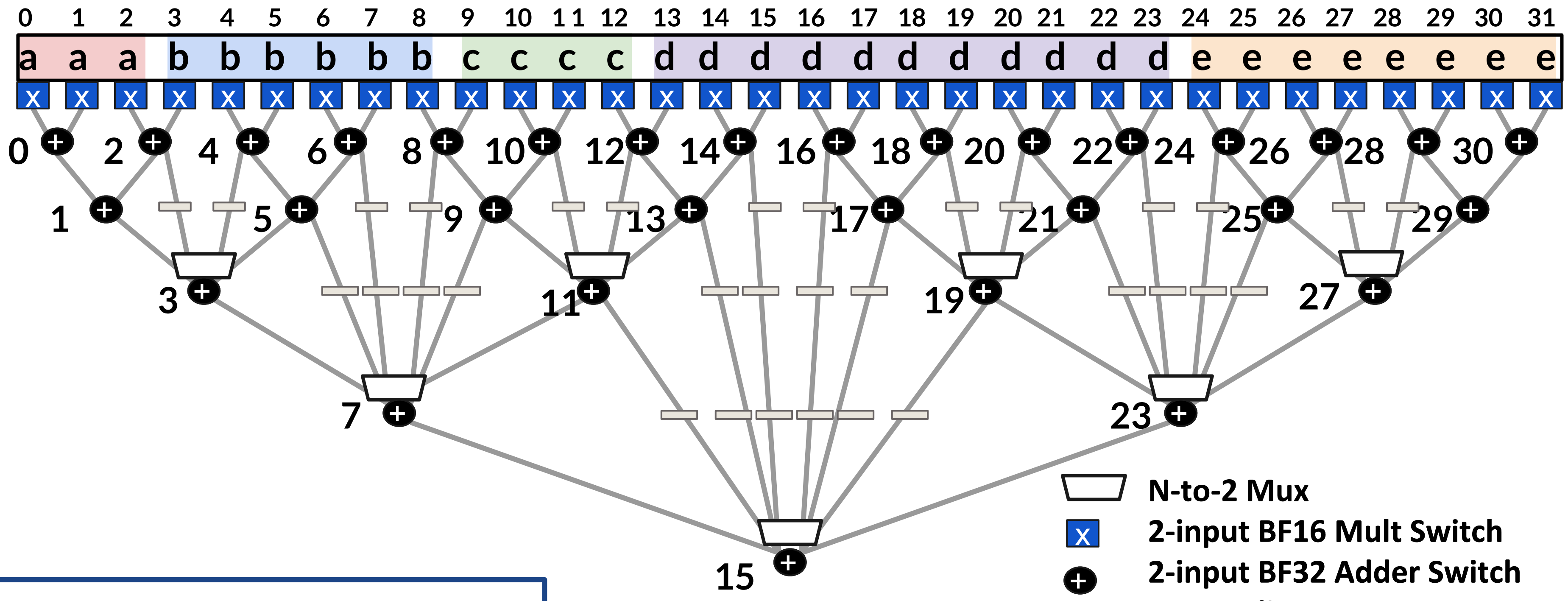
Different clusters of partial sums. ➡



FAN is optimized for floating point reductions, commonly used during DNN training.

Forwarding Adder Network (FAN)

Different clusters of partial sums. ➡

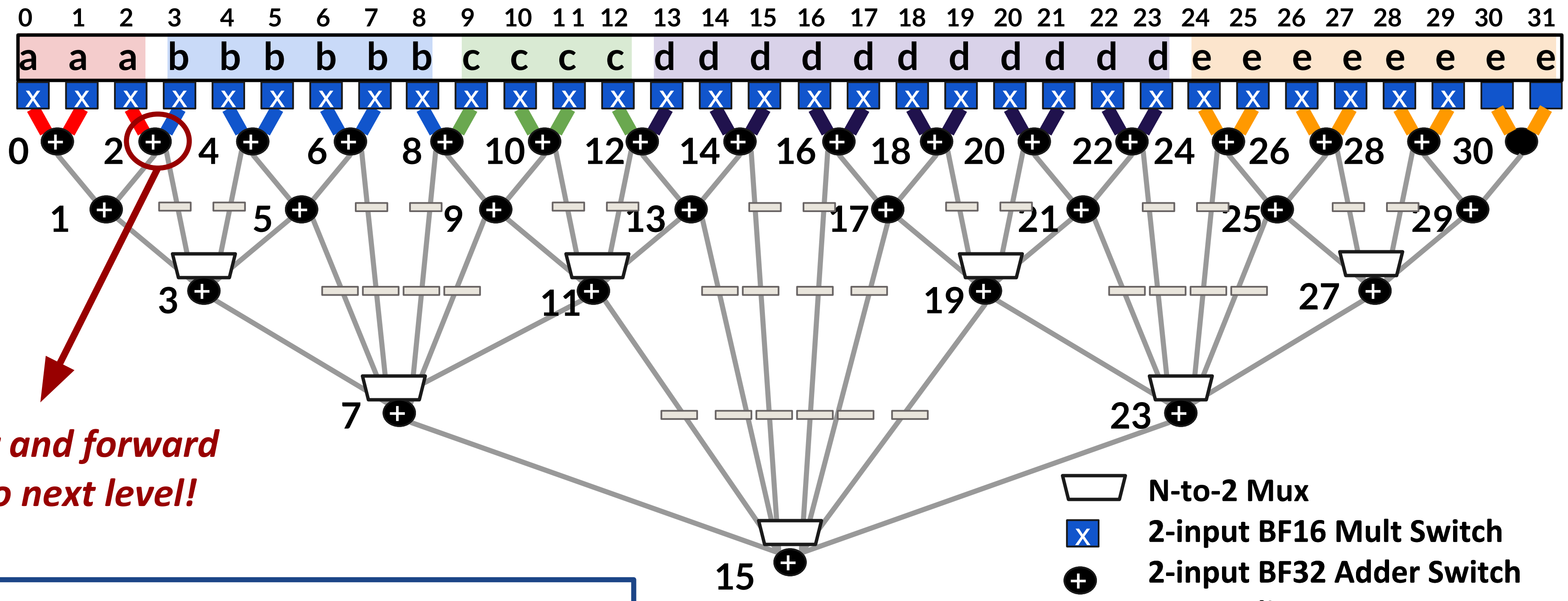


Pipelined at 1 cycle per adder level.

FAN is optimized for floating point reductions, commonly used during DNN training.

Forwarding Adder Network (FAN)

Different clusters of partial sums. ➡



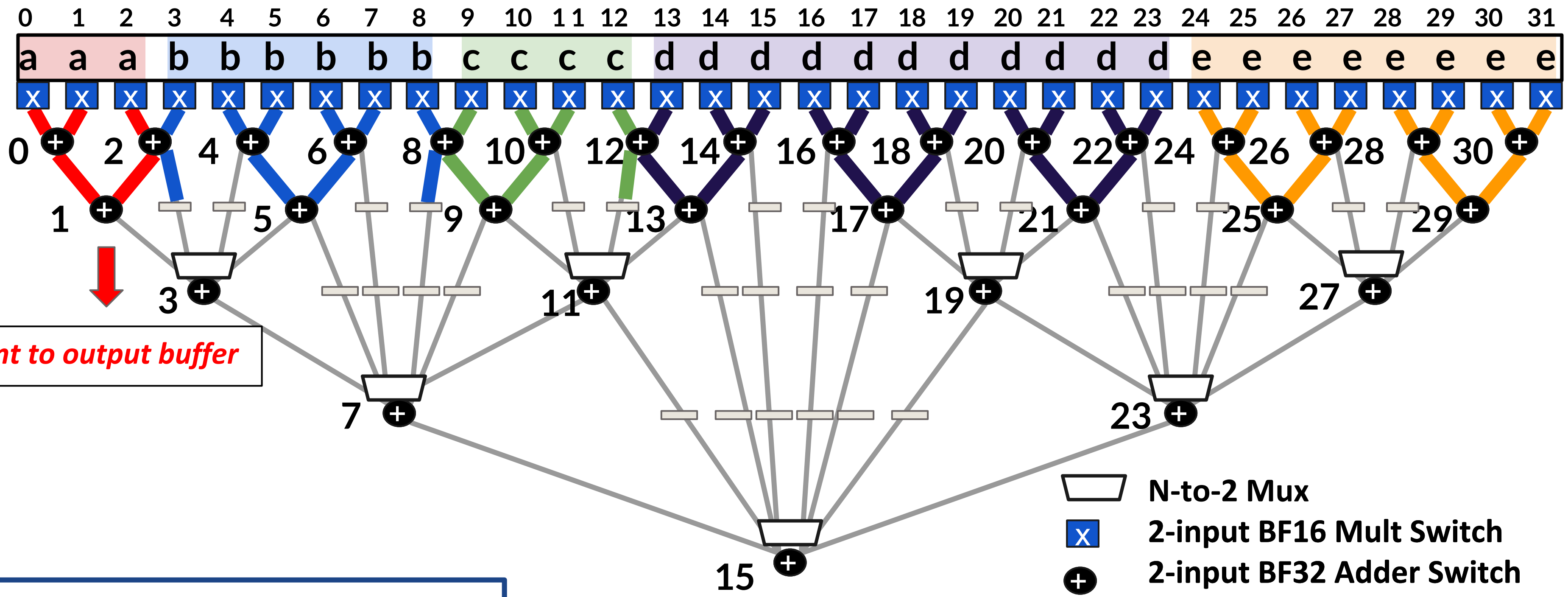
Bypass adder and forward partial sum to next level!

Cycle 1: Bypass conflicting partial sums

FAN is optimized for floating point reductions, commonly used during DNN training.

Forwarding Adder Network (FAN)

Different clusters of partial sums. ➡



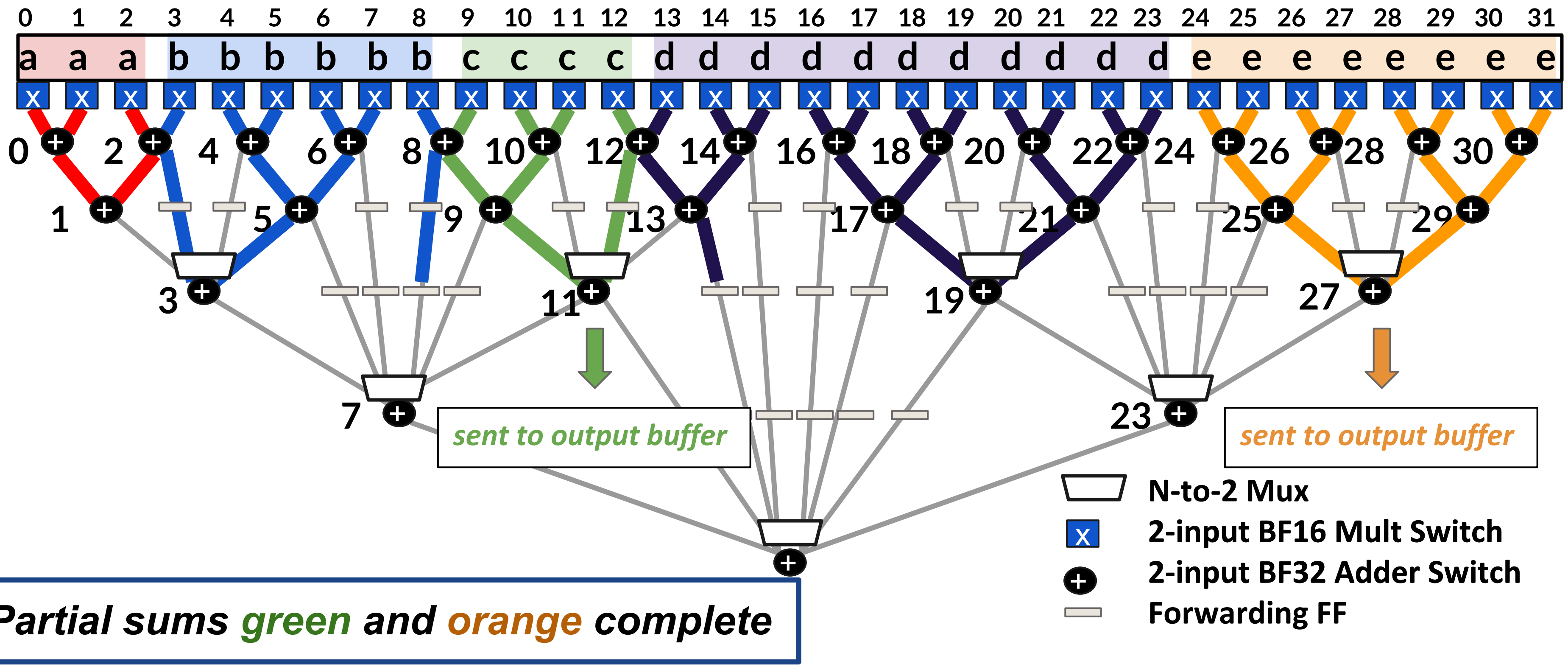
sent to output buffer

Cycle 2: Partial sum red complete

FAN is optimized for floating point reductions, commonly used during DNN training.

Forwarding Adder Network (FAN)

Different clusters of partial sums. ➡

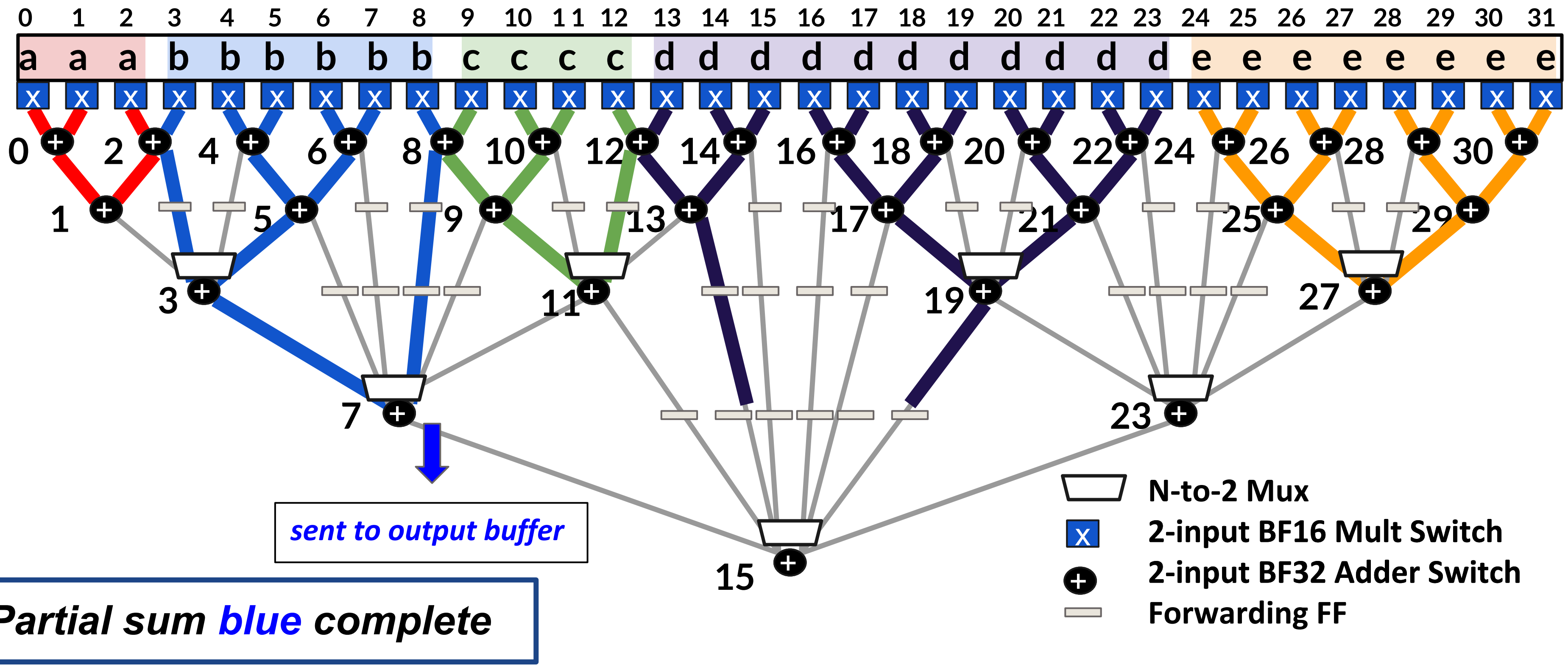


Cycle 3: Partial sums green and orange complete

FAN is optimized for floating point reductions, commonly used during DNN training.

Forwarding Adder Network (FAN)

Different clusters of partial sums. ➡



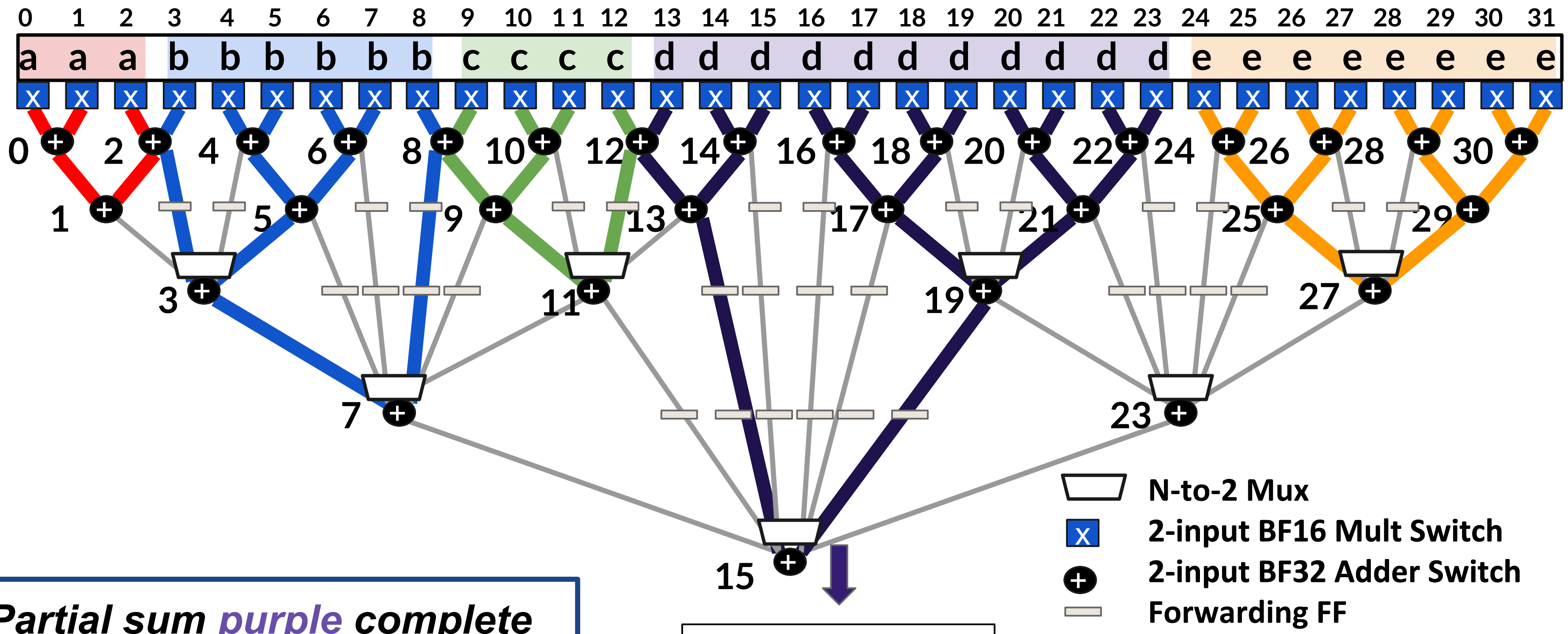
sent to output buffer

Cycle 4: Partial sum blue complete

FAN is optimized for floating point reductions, commonly used during DNN training.

Forwarding Adder Network (FAN)

Different clusters of partial sums. ➡



N-to-2 Mux

X 2-input BF16 Mult Switch

+ 2-input BF32 Adder Switch

▭ Forwarding FF

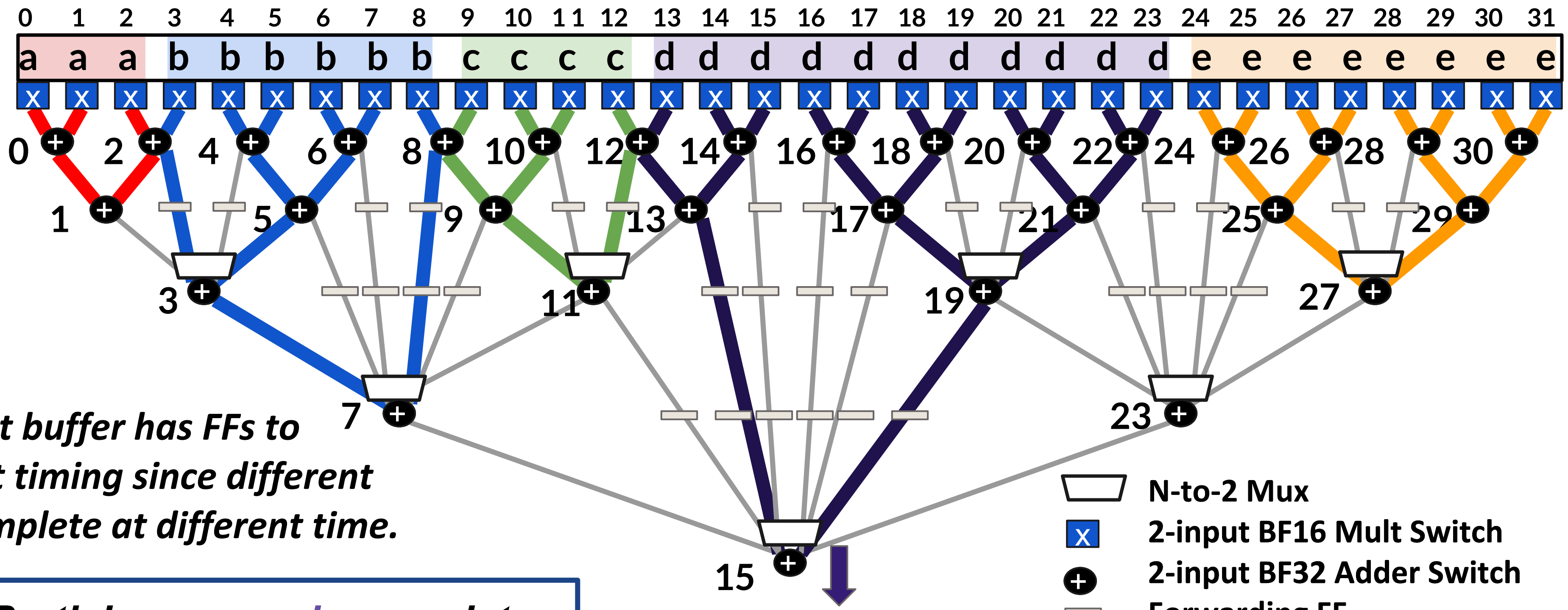
Cycle 5: Partial sum purple complete

sent to output buffer

FAN is optimized for floating point reductions, commonly used during DNN training.

Forwarding Adder Network (FAN)

Different clusters of partial sums. ➡



Note: The output buffer has FFs to maintain correct timing since different clusters may complete at different time.

Cycle 5: Partial sum purple complete

send to output buffer

FAN is optimized for floating point reductions, commonly used during DNN training.

Forwarding Adder Network (FAN)

Our novel FAN topology is both lightweight and flexible.

FAN is optimized for floating point reductions, commonly used during DNN training.

Forwarding Adder Network (FAN)

Our novel FAN topology is both lightweight and flexible.

It can replace regular adder trees in other hardware accelerators.

FAN is optimized for floating point reductions, commonly used during DNN training.

Forwarding Adder Network (FAN)

Our novel FAN topology is both lightweight and flexible.

It can replace regular adder trees in other hardware accelerators.

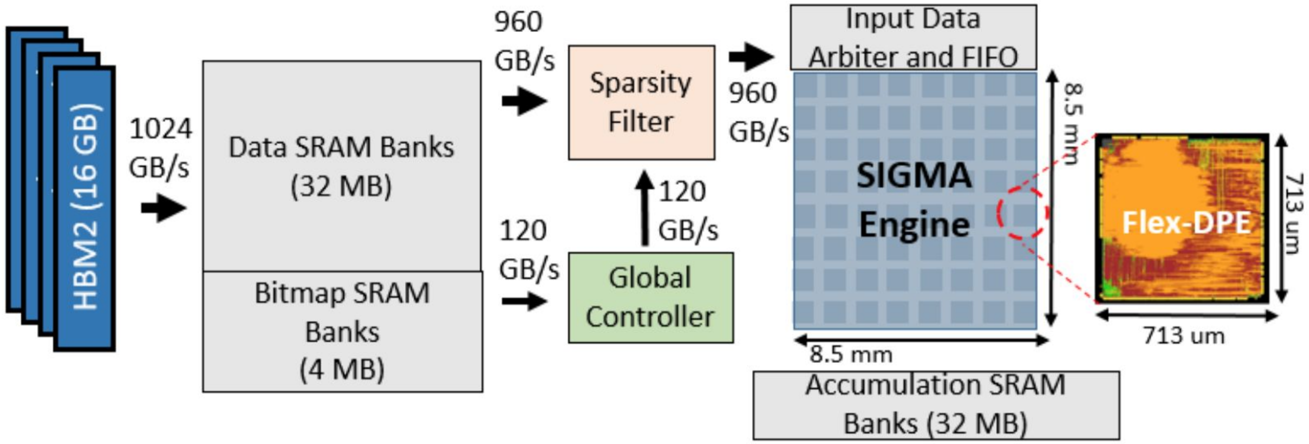
More details such as the routing algorithm and overhead analysis can be found in the paper.

FAN is optimized for floating point reductions, commonly used during DNN training.

Outline

- Motivation
    - GEMMs in Deep Learning
    - Utilization on TPU
- Accelerator Requirements
- **SIGMA**
    - Interconnects Implementations
    - **Full System Design**
- Evaluation
- Conclusion

SIGMA High Level Diagram

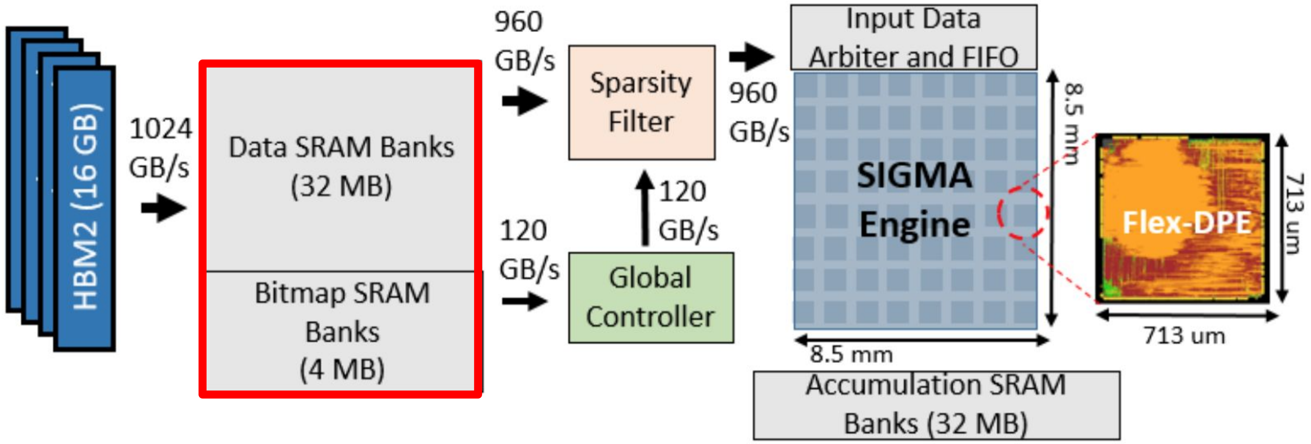


Note: SIGMA Engine contains multiple SIGMA units called Flex-DPEs.

SIGMA High Level Diagram

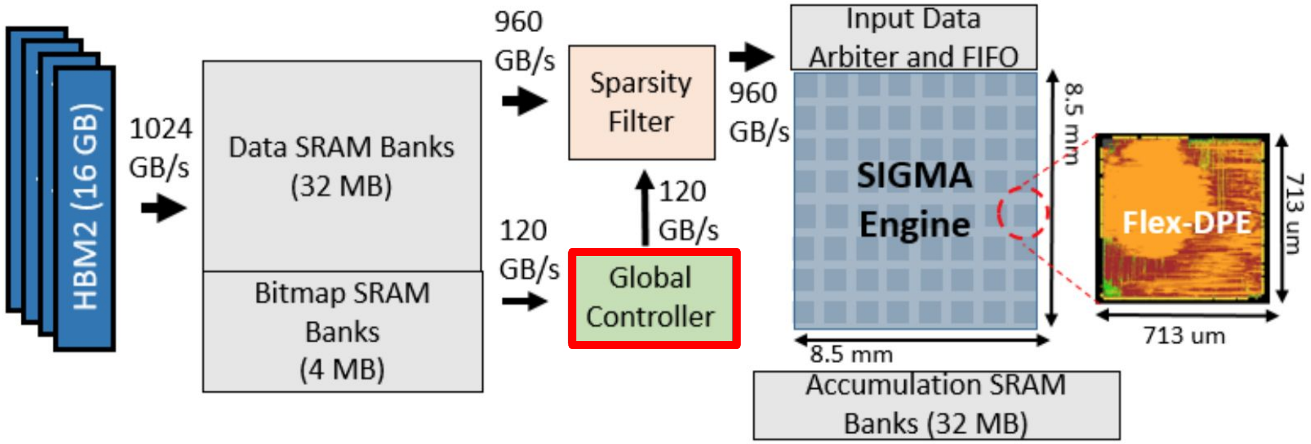
Data and Bitmap SRAM Banks

- ***Contains bitmap compression format of GEMM matrices.***



Note: SIGMA Engine contains multiple SIGMA units called Flex-DPEs.

SIGMA High Level Diagram



Note: SIGMA Engine contains multiple SIGMA units called Flex-DPEs.

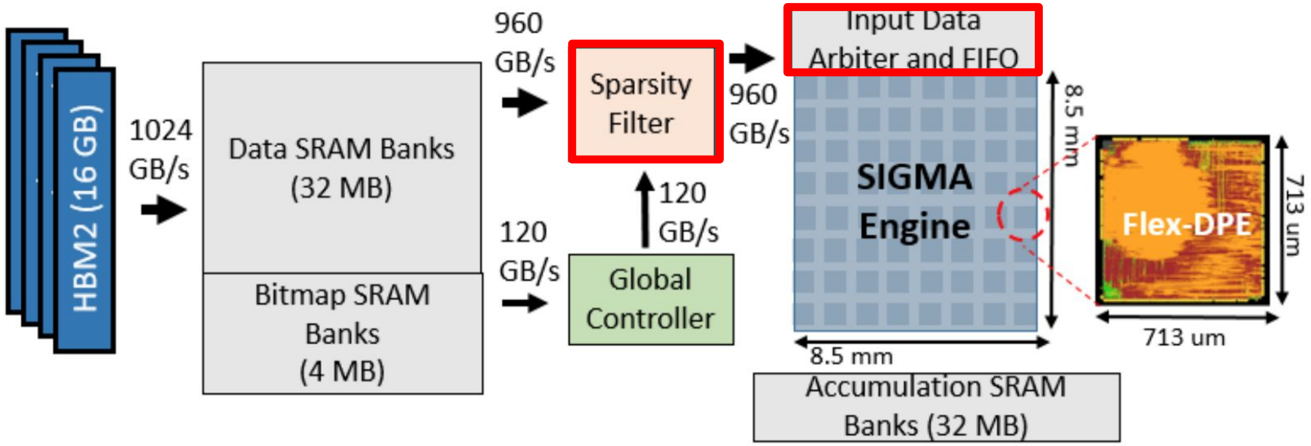
Data and Bitmap SRAM Banks

- ***Contains bitmap compression format of GEMM matrices.***

Global Controller

- ***Logic comparisons on bitmaps to determine what nonzero stationary elements are required***

SIGMA High Level Diagram



Note: SIGMA Engine contains multiple SIGMA units called Flex-DPEs.

Data and Bitmap SRAM Banks

- *Contains bitmap compression format of GEMM matrices.*

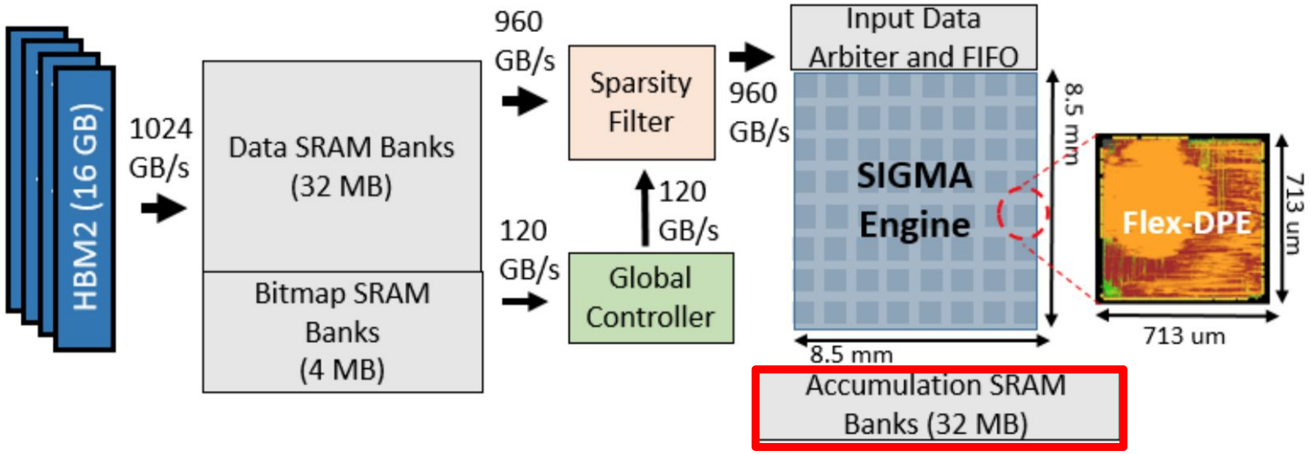
Global Controller

- *Logic comparisons on bitmaps to determine what nonzero stationary elements are required*

Sparsity Filter && Input Data Arbiter

- ***Reorganizes data for loading stationary elements and sending streaming elements***

SIGMA High Level Diagram



Note: SIGMA Engine contains multiple SIGMA units called Flex-DPEs.

Data and Bitmap SRAM Banks

- *Contains bitmap compression format of GEMM matrices.*

Global Controller

- *Logic comparisons on bitmaps to determine what nonzero stationary elements are required*

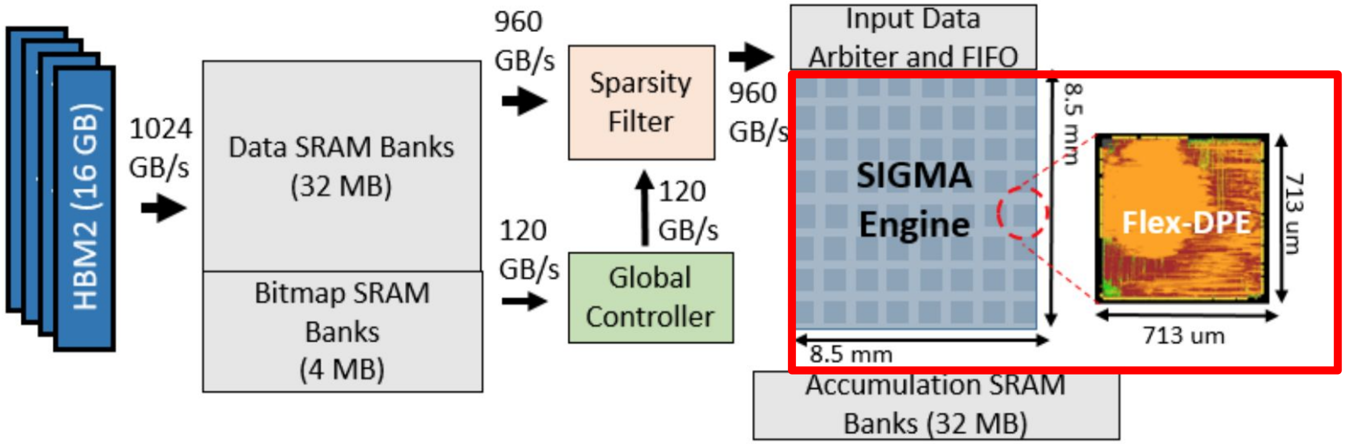
Sparsity Filter && Input Data Arbiter

- *Reorganizes data for loading stationary elements and sending streaming elements*

Accumulation SRAM

- ***Buffer for partial sum accumulations***

SIGMA High Level Diagram



Note: SIGMA Engine contains multiple SIGMA units called Flex-DPEs.

Data and Bitmap SRAM Banks

- *Contains bitmap compression format of GEMM matrices.*

Global Controller

- *Logic comparisons on bitmaps to determine what nonzero stationary elements are required*

Sparsity Filter && Input Data Arbiter

- *Reorganizes data for loading stationary elements and sending streaming elements*

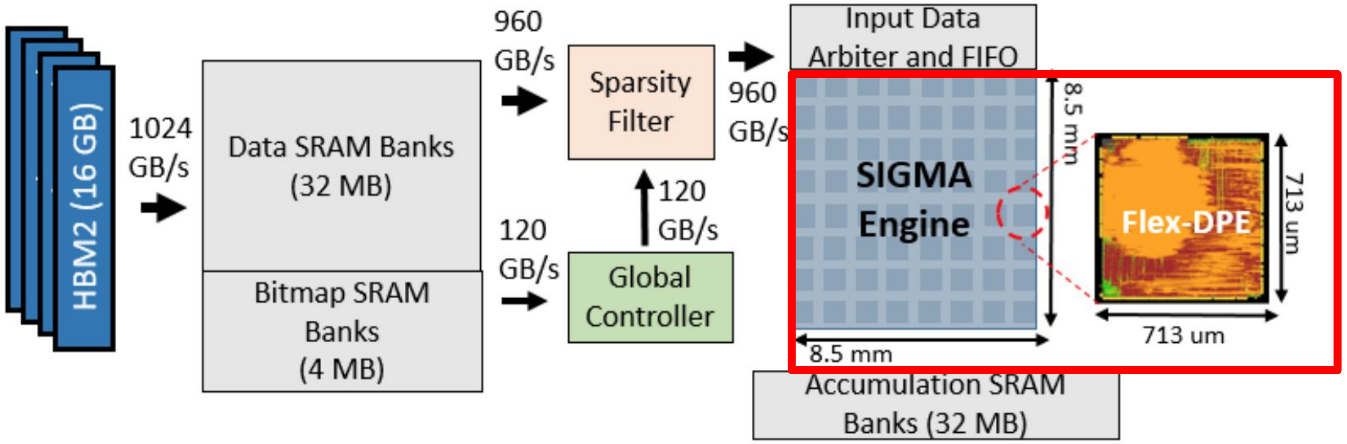
Accumulation SRAM

- *Buffer for partial sum accumulations*

SIGMA Engine

- ***Compute engine***

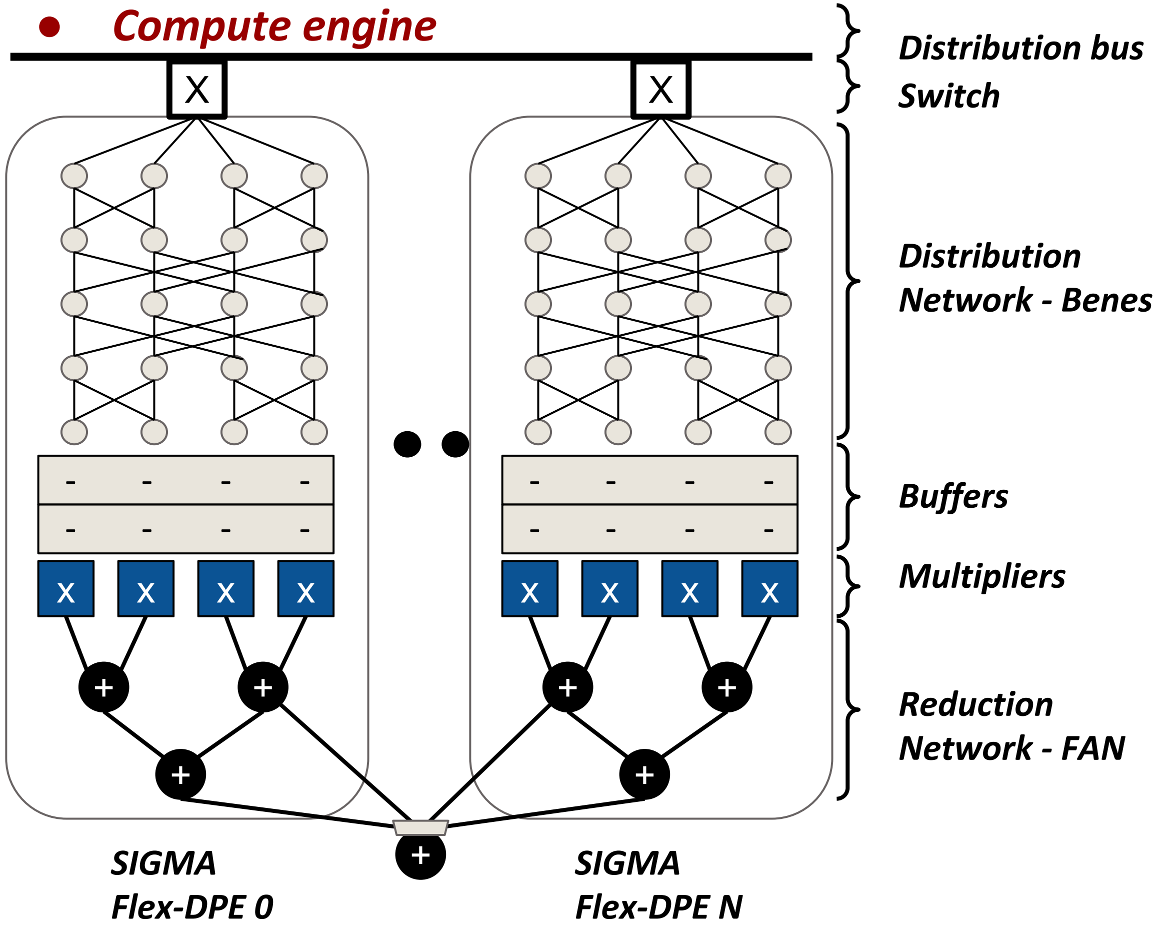
SIGMA High Level Diagram



Note: SIGMA Engine contains multiple SIGMA units called Flex-DPEs.

SIGMA Engine

- ***Compute engine***



Outline

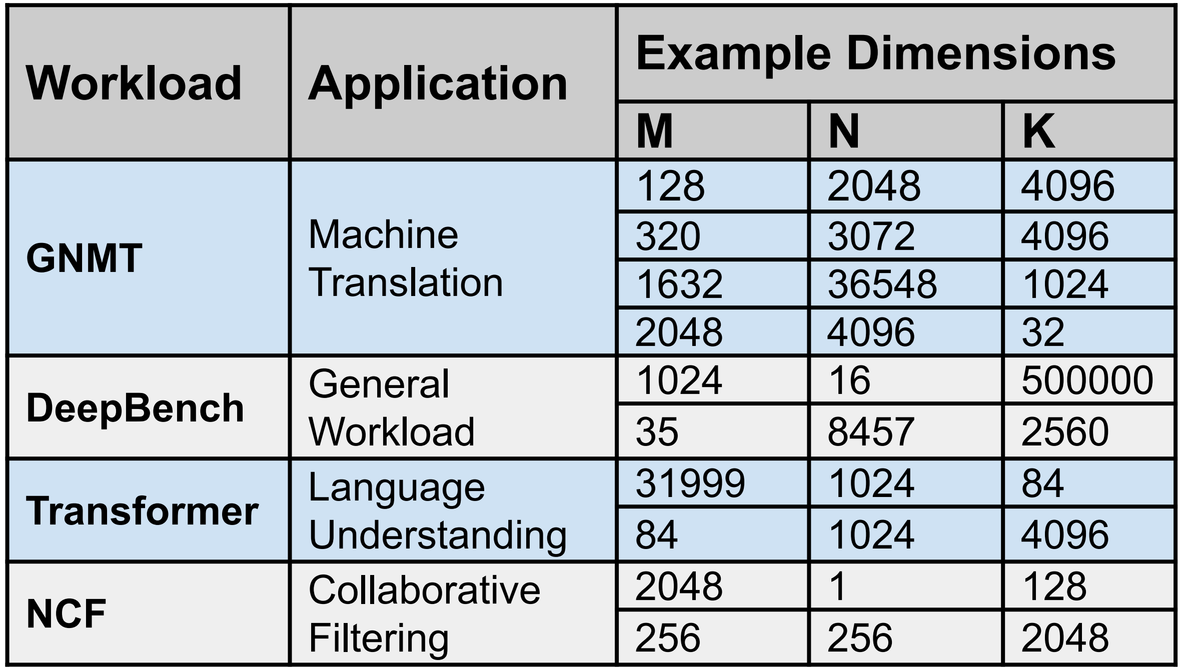
- Motivation
  - GEMMs in Deep Learning
  - Utilization on TPU
- Accelerator Requirements
- SIGMA
  - Interconnects Implementations
  - Full System Design
- **Evaluation**
- Conclusion

Methodology

- **Hardware components are written in Verilog**

- **Post layout area and power numbers are on a 28nm process**

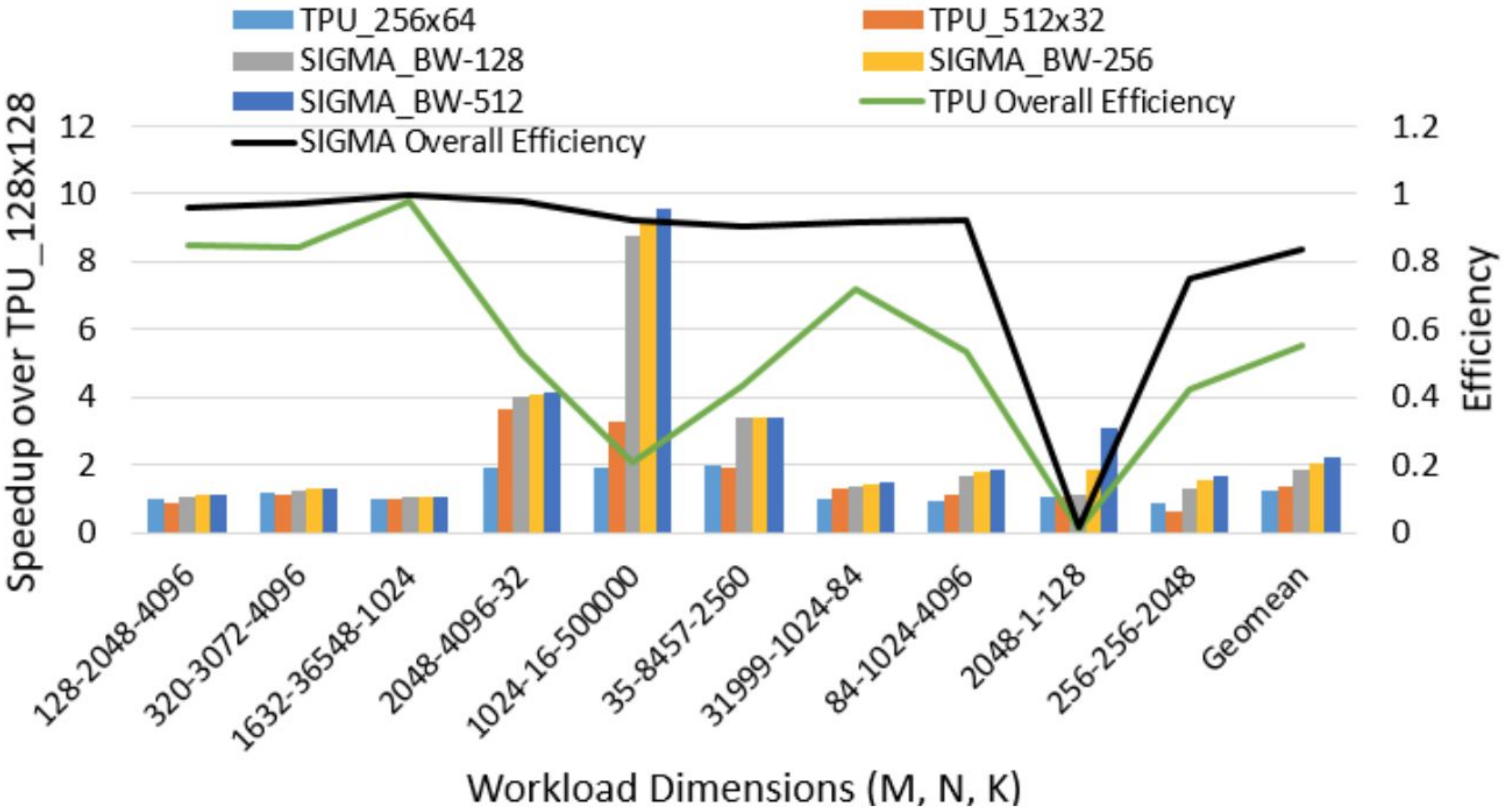
- **Analytical model for cycle counts assumes uniform random sparsity**



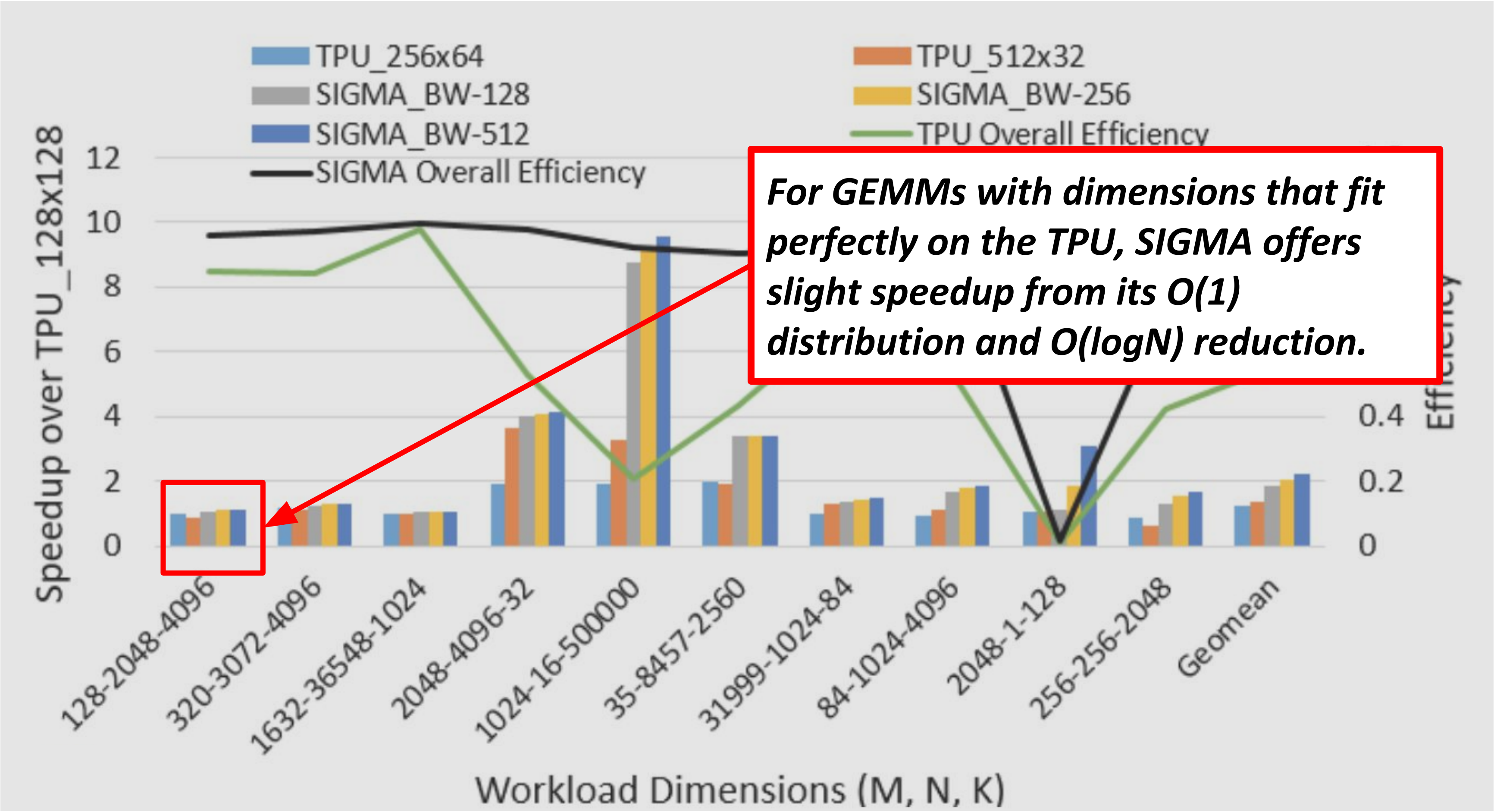
Workload	**Application**	**Example Dimensions**		
		M	**N**	**K**
GNMT	Machine Translation	128	2048	4096
		320	3072	4096
		1632	36548	1024
		2048	4096	32
DeepBench	General Workload	1024	16	500000
		35	8457	2560
Transformer	Language Understanding	31999	1024	84
		84	1024	4096
NCF	Collaborative Filtering	2048	1	128
		256	256	2048

GEMMs used for evaluation.

SIGMA vs TPU - Dense GEMMs

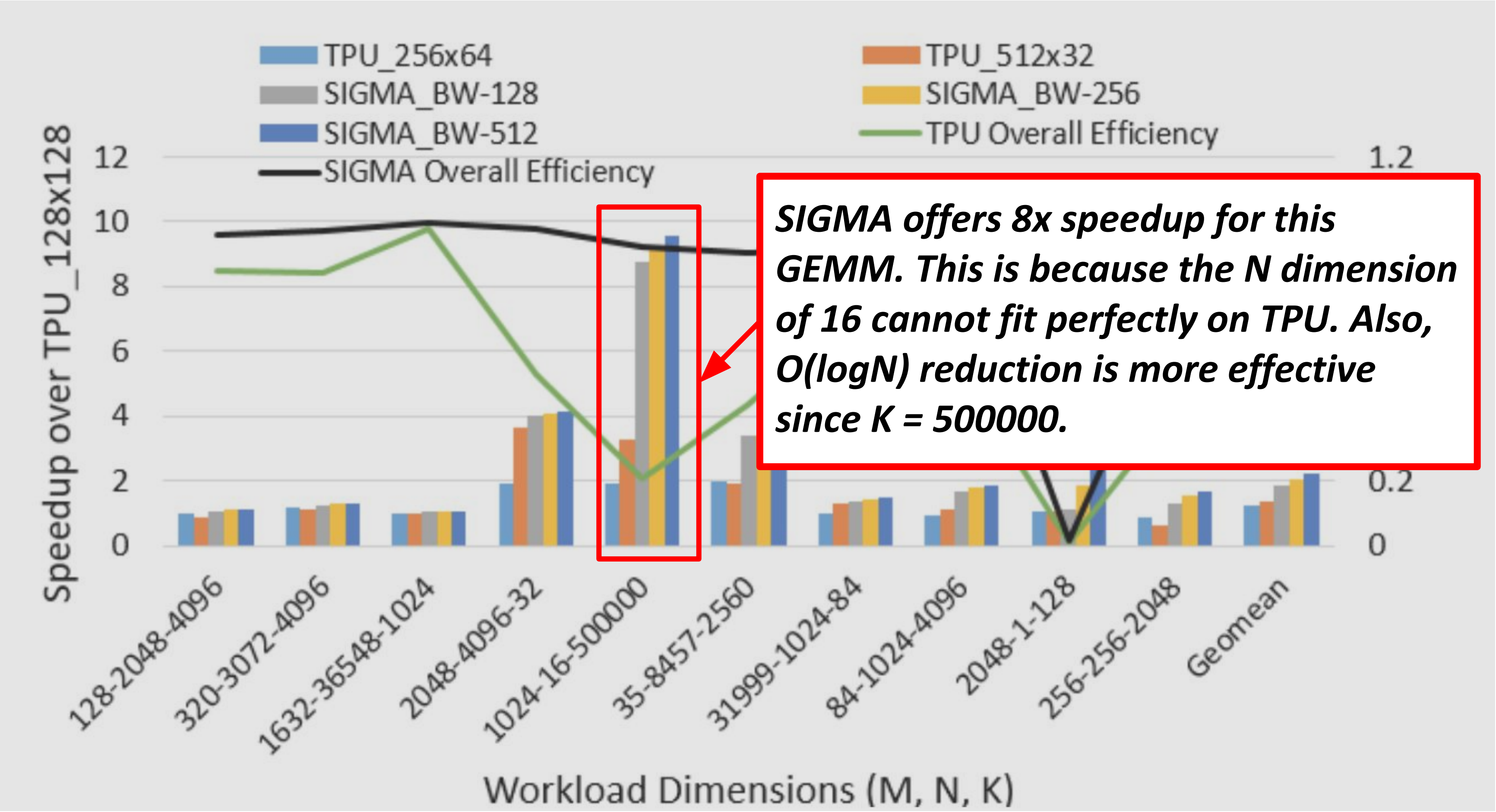


SIGMA vs TPU - Dense GEMMs



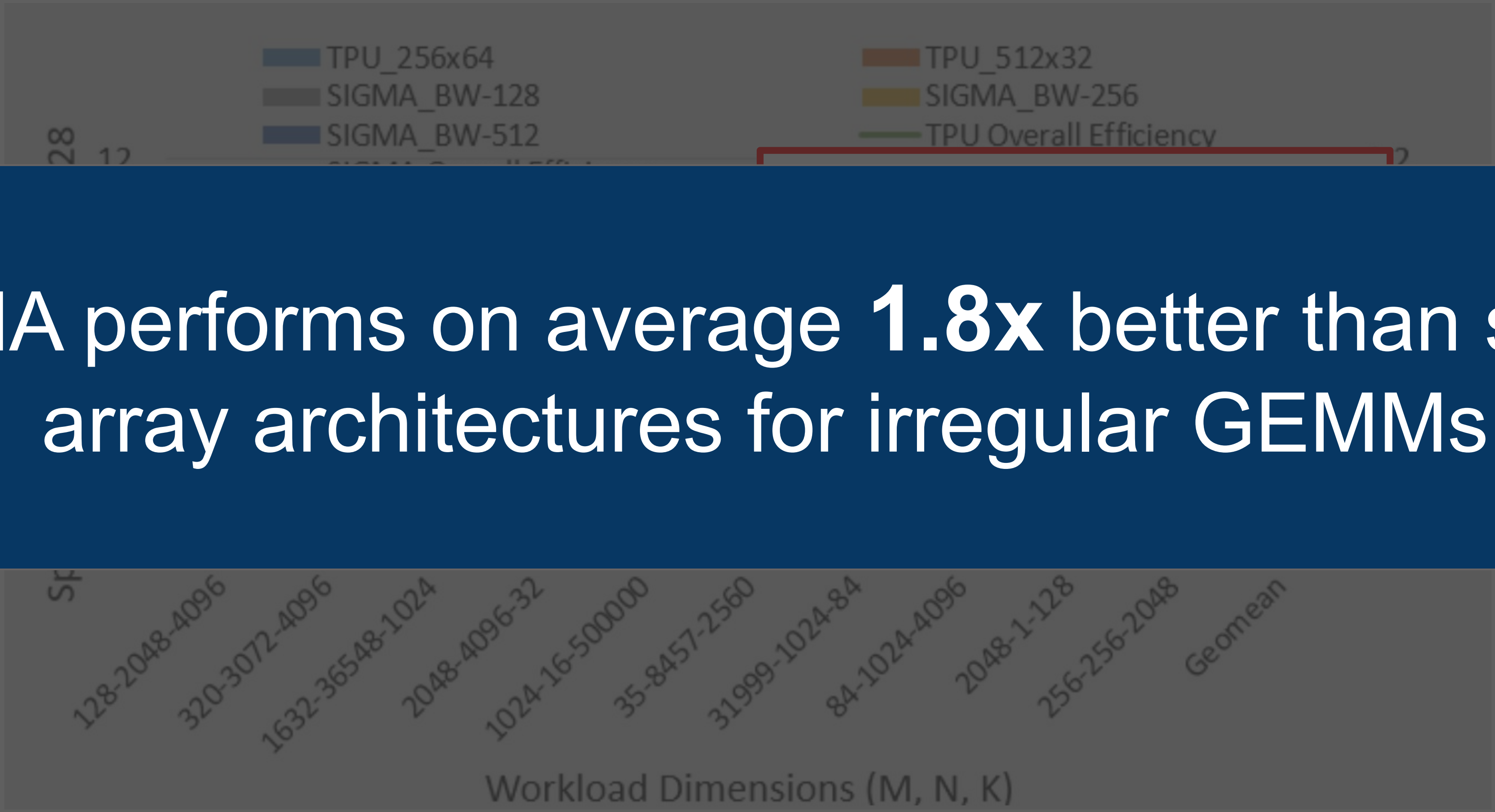
For GEMMs with dimensions that fit perfectly on the TPU, SIGMA offers slight speedup from its O(1) distribution and O(logN) reduction.

SIGMA vs TPU - Dense GEMMs



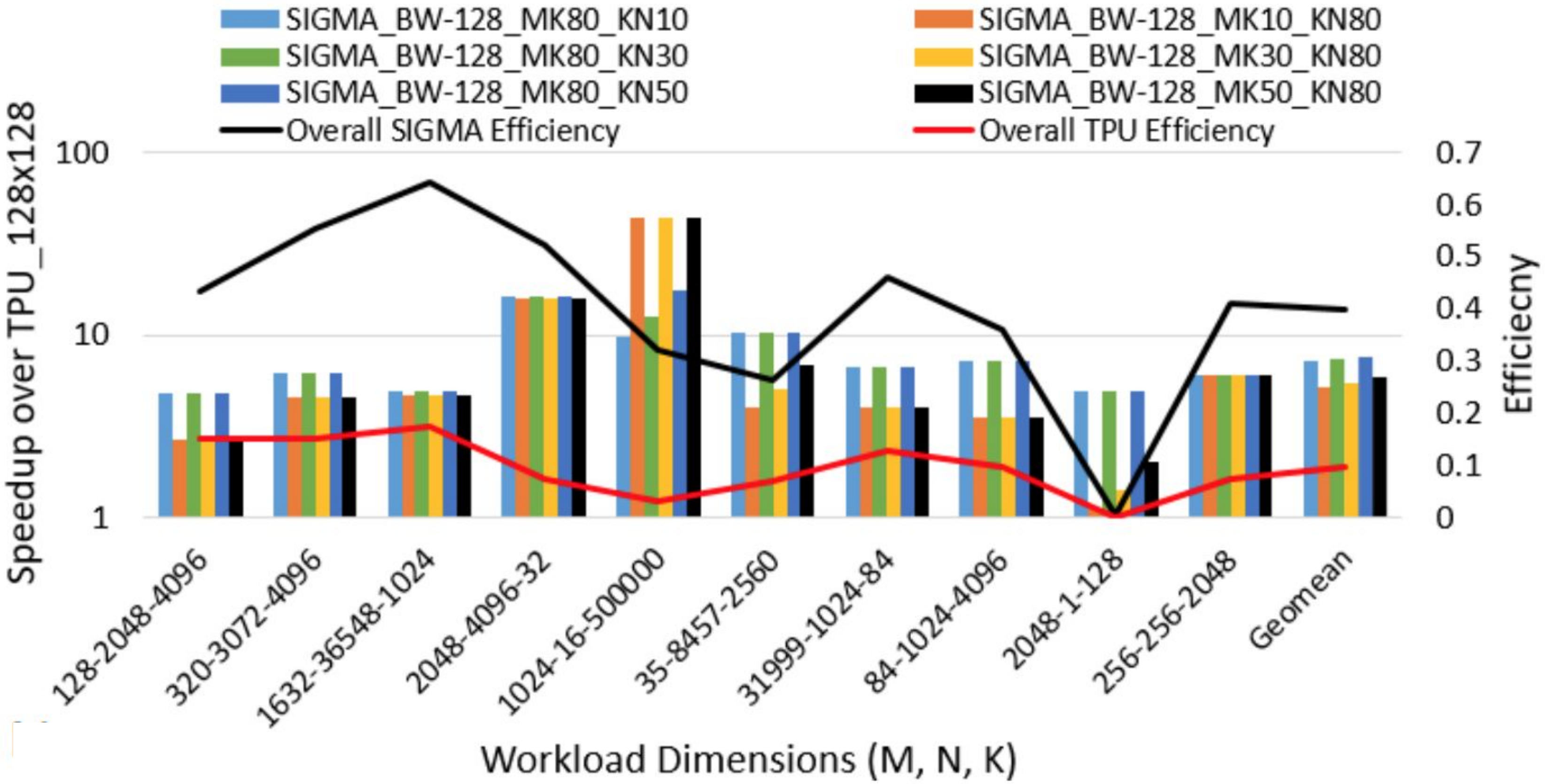
SIGMA offers 8x speedup for this GEMM. This is because the N dimension of 16 cannot fit perfectly on TPU. Also, O(logN) reduction is more effective since K = 500000.

SIGMA vs TPU - Dense GEMMs

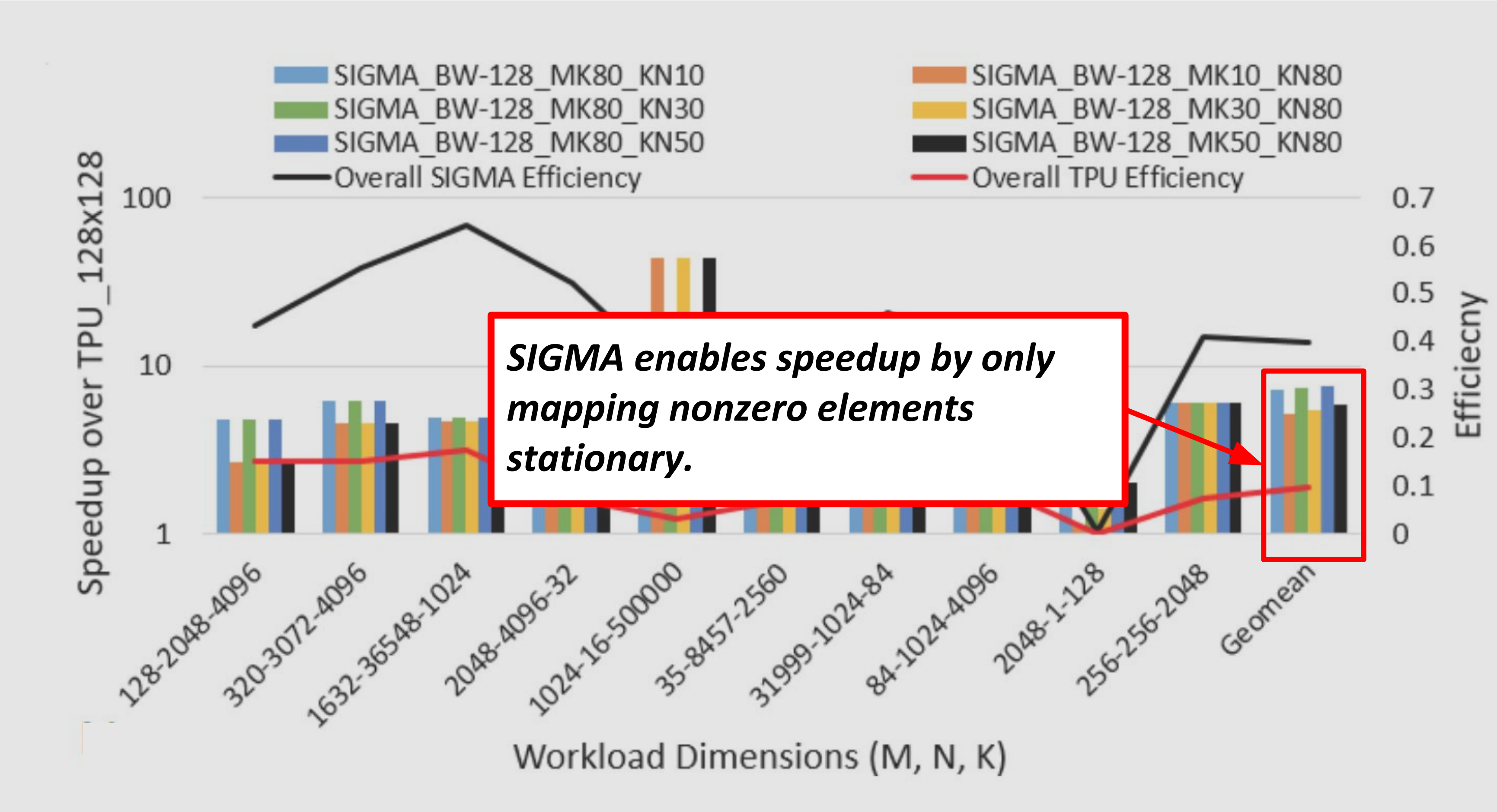


SIGMA performs on average **1.8x** better than systolic array architectures for irregular GEMMs.

SIGMA vs TPU - Sparse GEMMs

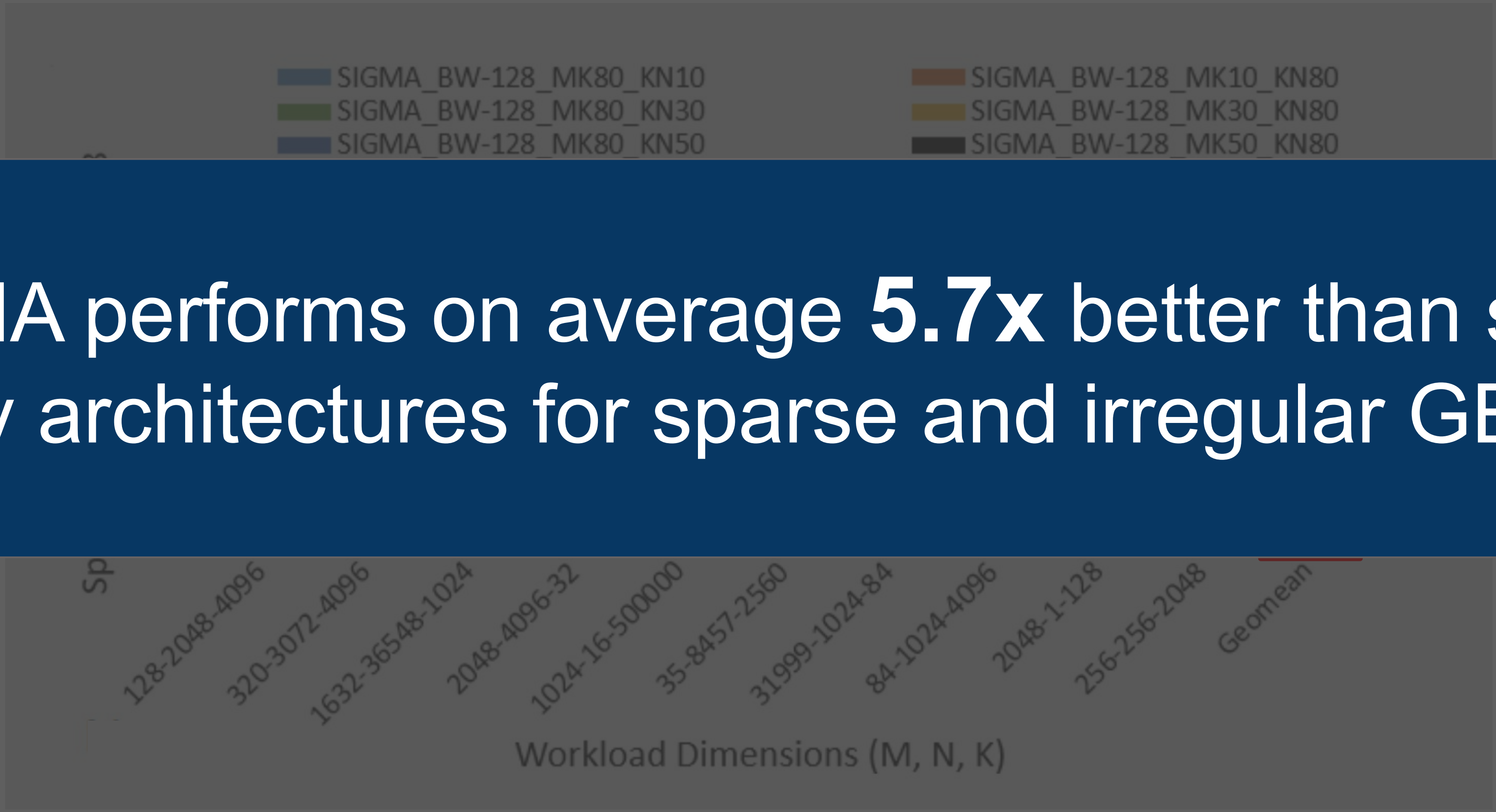


SIGMA vs TPU - Sparse GEMMs



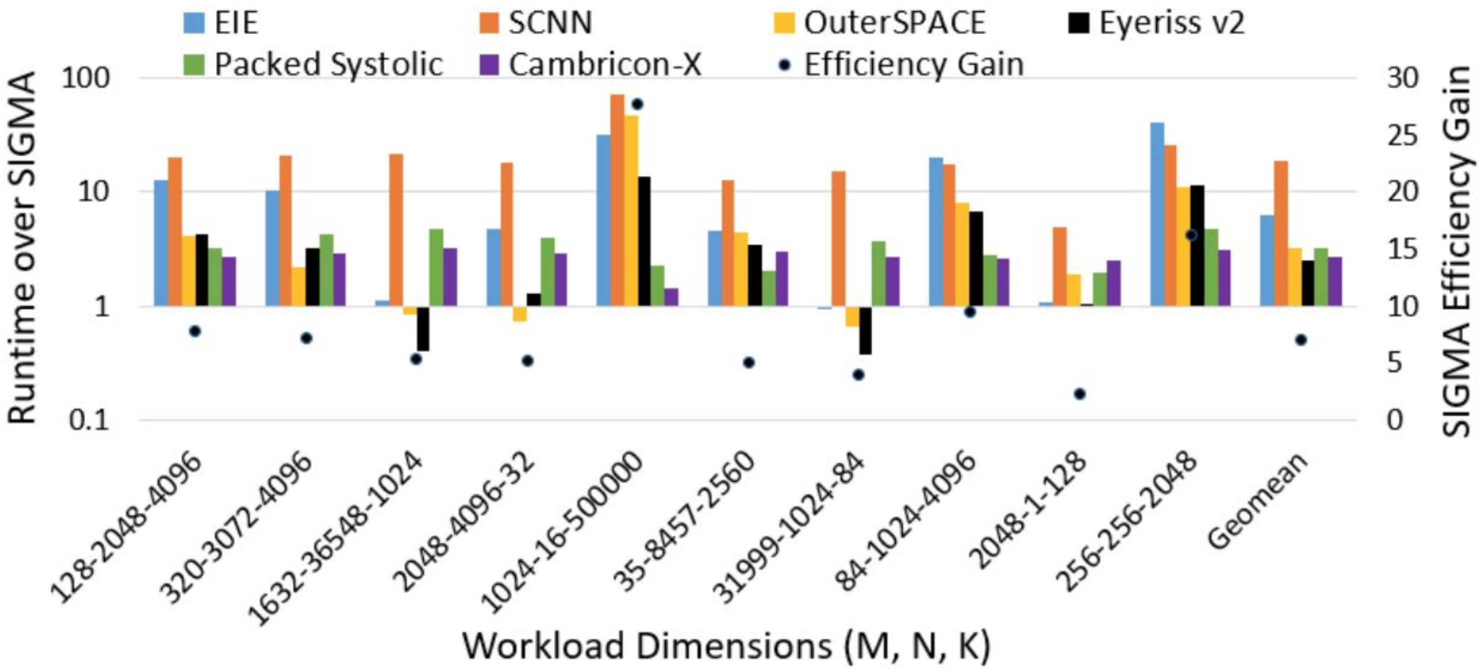
SIGMA enables speedup by only mapping nonzero elements stationary.

SIGMA vs TPU - Sparse GEMMs

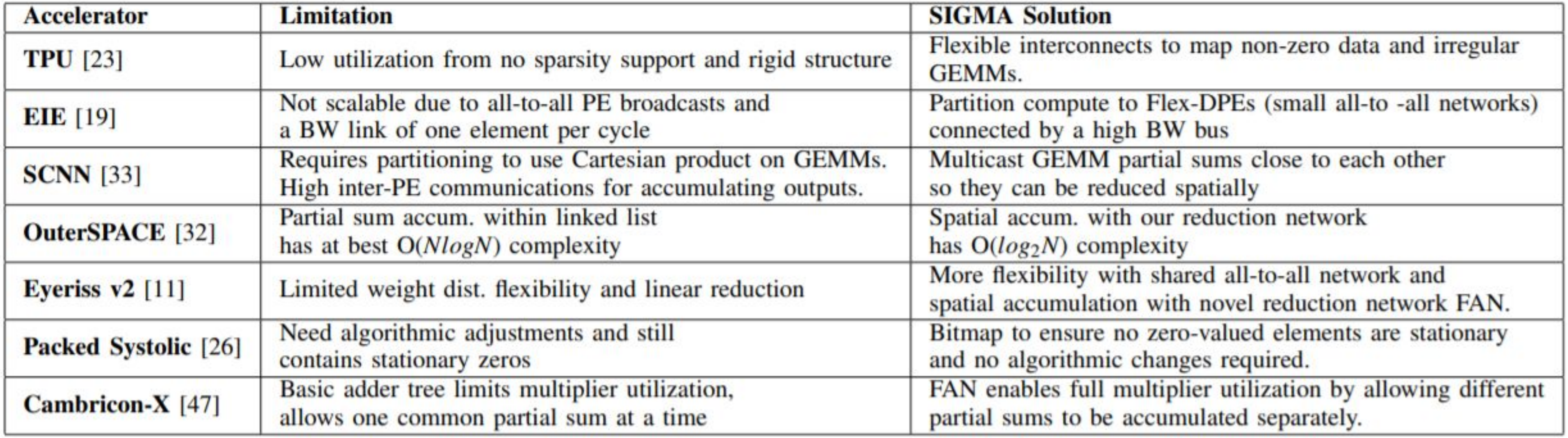


SIGMA performs on average **5.7x** better than systolic array architectures for sparse and irregular GEMMs.

SIGMA vs Sparse Accelerators



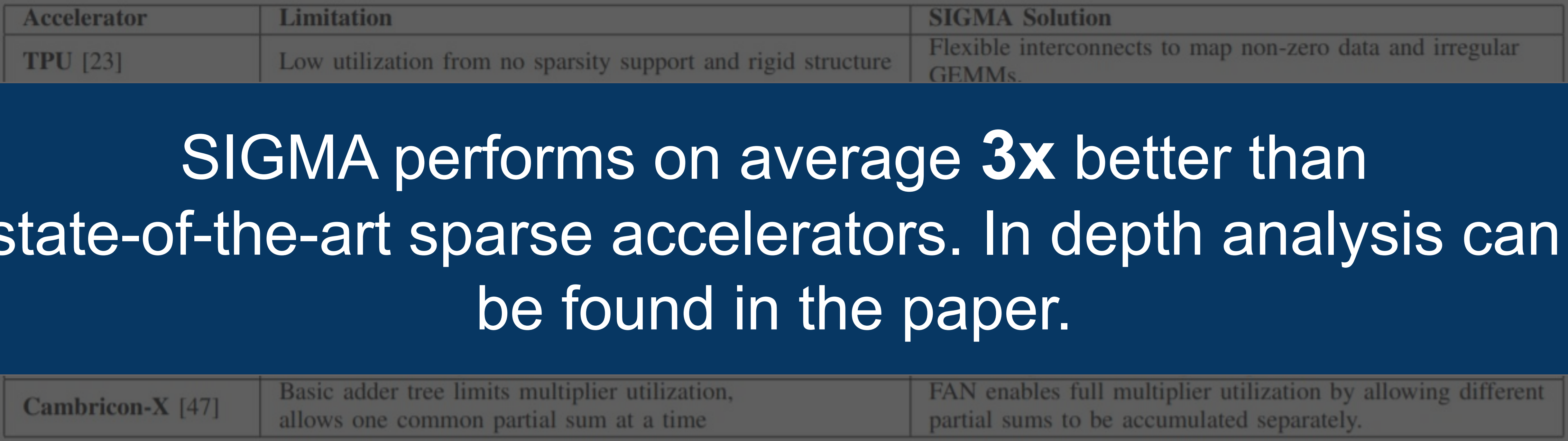
SIGMA Qualitative Analysis



Accelerator	Limitation	SIGMA Solution
TPU [23]	Low utilization from no sparsity support and rigid structure	Flexible interconnects to map non-zero data and irregular GEMMs.
EIE [19]	Not scalable due to all-to-all PE broadcasts and a BW link of one element per cycle	Partition compute to Flex-DPEs (small all-to -all networks) connected by a high BW bus
SCNN [33]	Requires partitioning to use Cartesian product on GEMMs. High inter-PE communications for accumulating outputs.	Multicast GEMM partial sums close to each other so they can be reduced spatially
OuterSPACE [32]	Partial sum accum. within linked list has at best O($NlogN$) complexity	Spatial accum. with our reduction network has O(log_2N) complexity
Eyeriss v2 [11]	Limited weight dist. flexibility and linear reduction	More flexibility with shared all-to-all network and spatial accumulation with novel reduction network FAN.
Packed Systolic [26]	Need algorithmic adjustments and still contains stationary zeros	Bitmap to ensure no zero-valued elements are stationary and no algorithmic changes required.
Cambricon-X [47]	Basic adder tree limits multiplier utilization, allows one common partial sum at a time	FAN enables full multiplier utilization by allowing different partial sums to be accumulated separately.

Table III: **Qualitative Comparision of SIGMA against state-of-the-art accelerators.**

SIGMA Qualitative Analysis

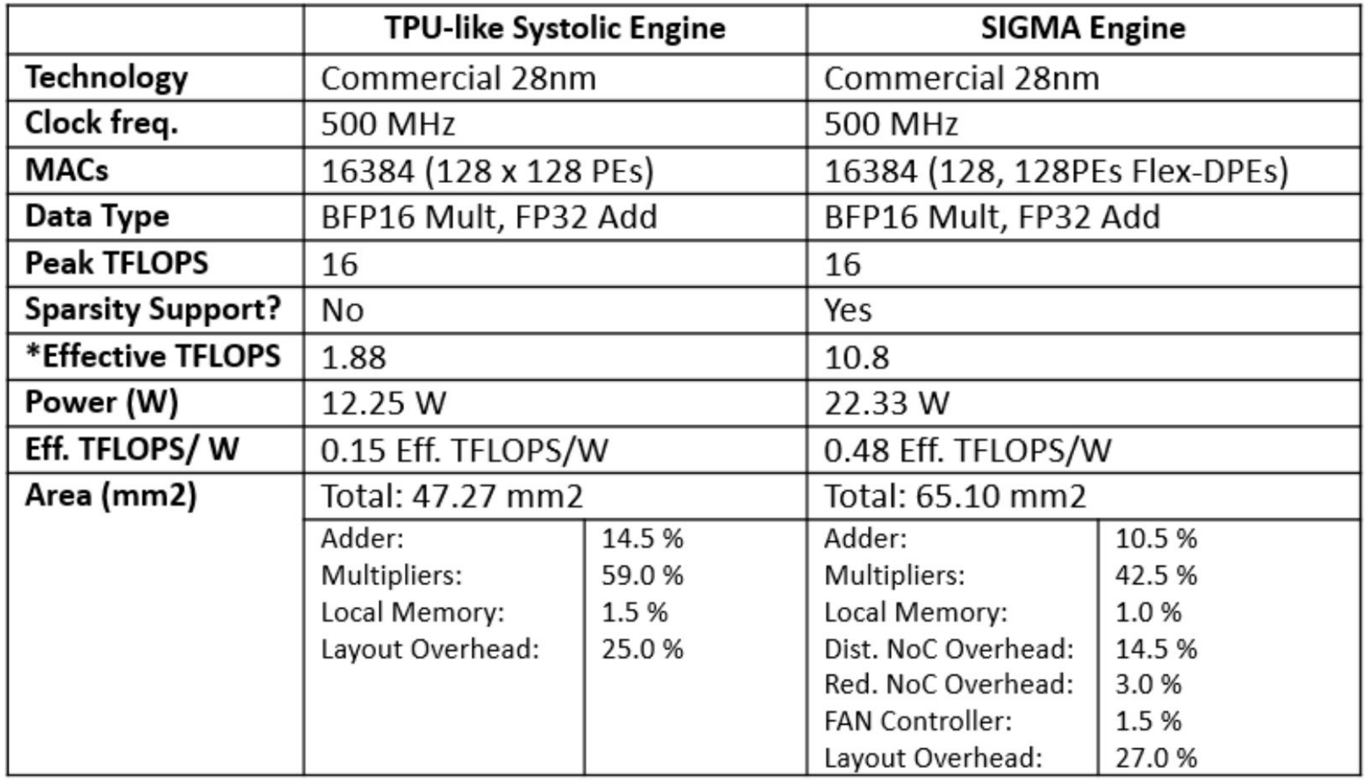


Accelerator	Limitation	SIGMA Solution
TPU [23]	Low utilization from no sparsity support and rigid structure	Flexible interconnects to map non-zero data and irregular GEMMs.
Cambricon-X [47]	Basic adder tree limits multiplier utilization, allows one common partial sum at a time	FAN enables full multiplier utilization by allowing different partial sums to be accumulated separately.

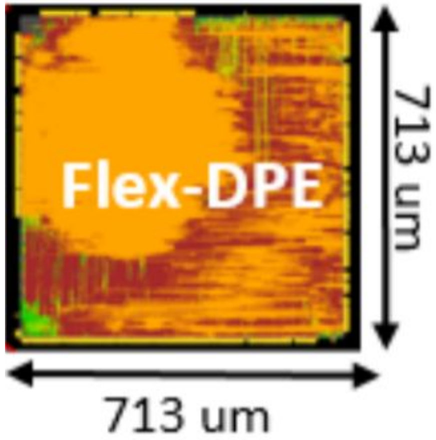
Table III: **Qualitative Comparision of SIGMA against state-of-the-art accelerators.**

SIGMA performs on average **3x** better than state-of-the-art sparse accelerators. In depth analysis can be found in the paper.

Systolic Array vs SIGMA Comparison

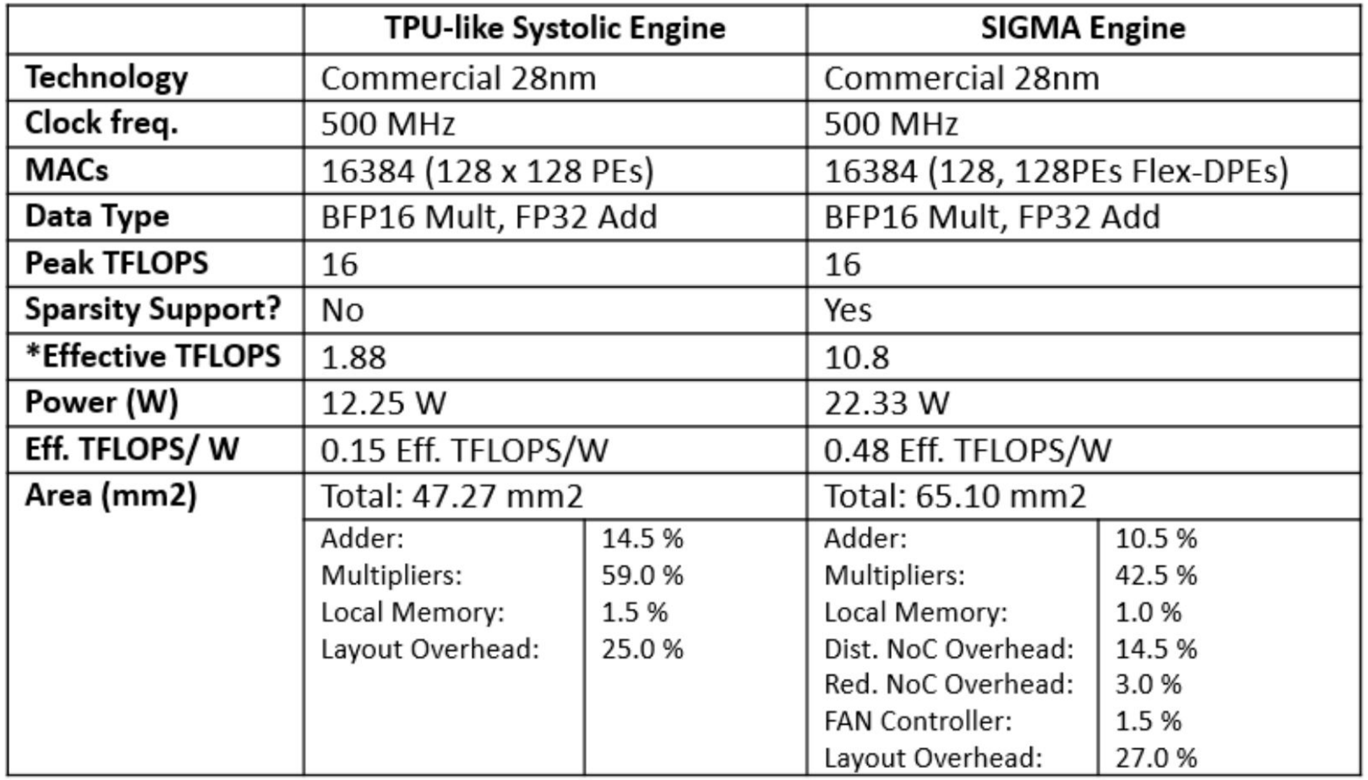


	TPU-like Systolic Engine	SIGMA Engine
Technology	Commercial 28nm	Commercial 28nm
Clock freq.	500 MHz	500 MHz
MACs	16384 (128 x 128 PEs)	16384 (128, 128PEs Flex-DPEs)
Data Type	BFP16 Mult, FP32 Add	BFP16 Mult, FP32 Add
Peak TFLOPS	16	16
Sparsity Support?	No	Yes
*Effective TFLOPS	1.88	10.8
Power (W)	12.25 W	22.33 W
Eff. TFLOPS/ W	0.15 Eff. TFLOPS/W	0.48 Eff. TFLOPS/W
Area (mm2)	Total: 47.27 mm2	Total: 65.10 mm2
	Adder: 14.5 % Multipliers: 59.0 % Local Memory: 1.5 % Layout Overhead: 25.0 %	Adder: 10.5 % Multipliers: 42.5 % Local Memory: 1.0 % Dist. NoC Overhead: 14.5 % Red. NoC Overhead: 3.0 % FAN Controller: 1.5 % Layout Overhead: 27.0 %

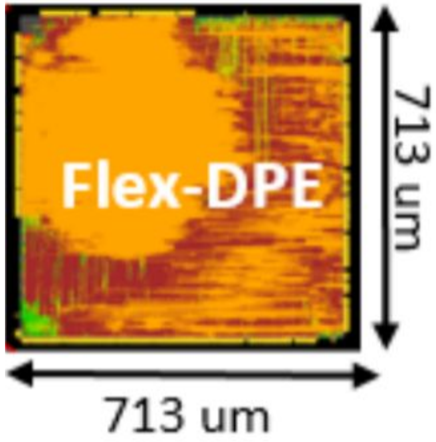


***** Effective TFLOPs is calculated by multiplying the base dense TFLOPs with the average efficiency computed across GEMMs.***

Systolic Array vs SIGMA Comparison



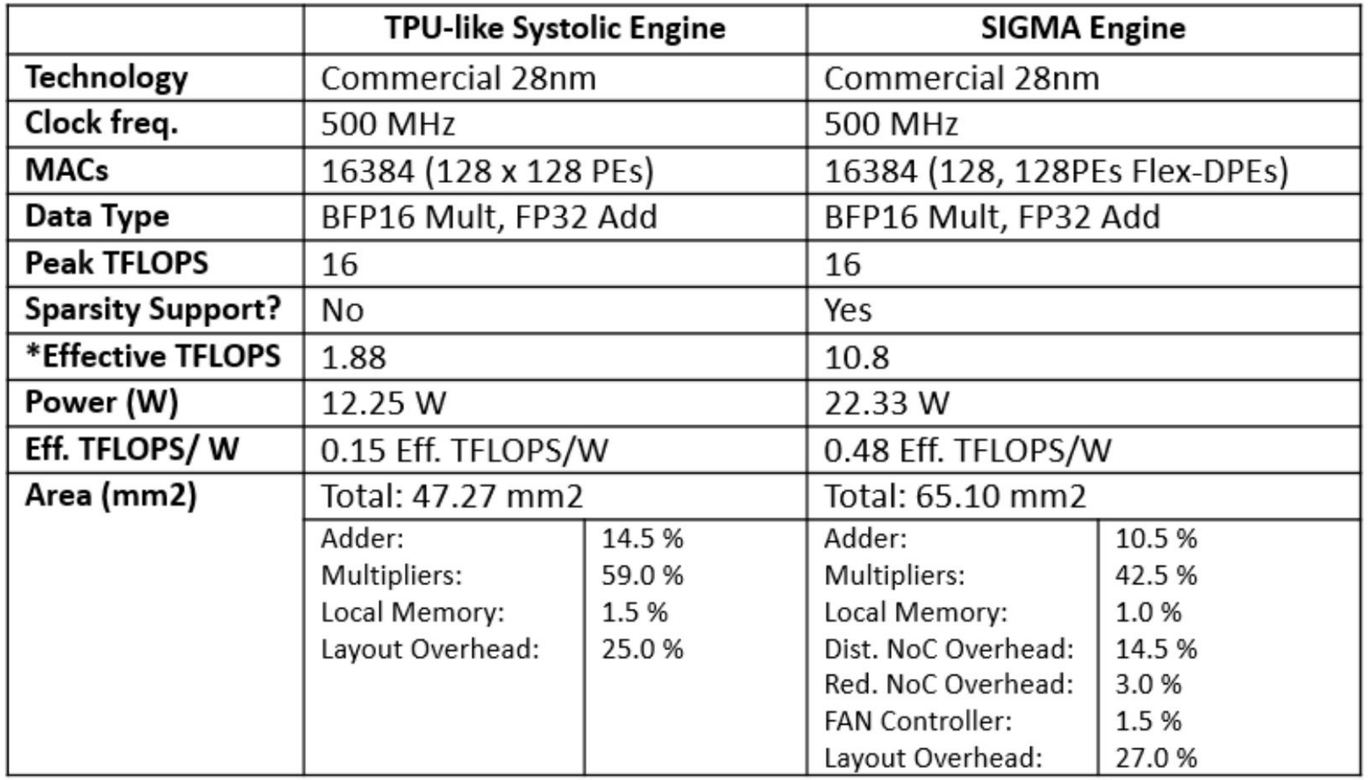
	TPU-like Systolic Engine		SIGMA Engine	
Technology	Commercial 28nm		Commercial 28nm	
Clock freq.	500 MHz		500 MHz	
MACs	16384 (128 x 128 PEs)		16384 (128, 128PEs Flex-DPEs)	
Data Type	BFP16 Mult, FP32 Add		BFP16 Mult, FP32 Add	
Peak TFLOPS	16		16	
Sparsity Support?	No		Yes	
*Effective TFLOPS	1.88		10.8	
Power (W)	12.25 W		22.33 W	
Eff. TFLOPS/ W	0.15 Eff. TFLOPS/W		0.48 Eff. TFLOPS/W	
Area (mm2)	Total: 47.27 mm2		Total: 65.10 mm2	
	Adder: Multipliers: Local Memory: Layout Overhead:	14.5 % 59.0 % 1.5 % 25.0 %	Adder: Multipliers: Local Memory: Dist. NoC Overhead: Red. NoC Overhead: FAN Controller: Layout Overhead:	10.5 % 42.5 % 1.0 % 14.5 % 3.0 % 1.5 % 27.0 %



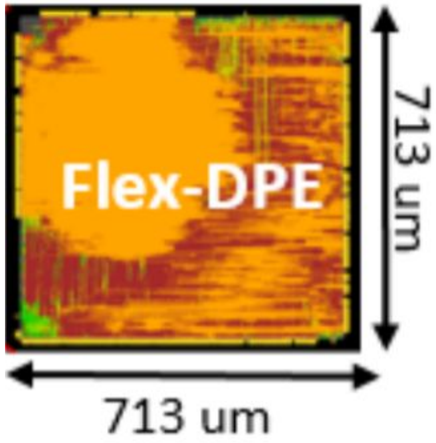
***** Effective TFLOPs is calculated by multiplying the base dense TFLOPs with the average efficiency computed across GEMMs.***

SIGMA consumes 38% more area and 82% more power than Systolic Array.

Systolic Array vs SIGMA Comparison



	TPU-like Systolic Engine	SIGMA Engine
Technology	Commercial 28nm	Commercial 28nm
Clock freq.	500 MHz	500 MHz
MACs	16384 (128 x 128 PEs)	16384 (128, 128PEs Flex-DPEs)
Data Type	BFP16 Mult, FP32 Add	BFP16 Mult, FP32 Add
Peak TFLOPS	16	16
Sparsity Support?	No	Yes
*Effective TFLOPS	1.88	10.8
Power (W)	12.25 W	22.33 W
Eff. TFLOPS/ W	0.15 Eff. TFLOPS/W	0.48 Eff. TFLOPS/W
Area (mm2)	Total: 47.27 mm2 Adder: 14.5 % Multipliers: 59.0 % Local Memory: 1.5 % Layout Overhead: 25.0 %	Total: 65.10 mm2 Adder: 10.5 % Multipliers: 42.5 % Local Memory: 1.0 % Dist. NoC Overhead: 14.5 % Red. NoC Overhead: 3.0 % FAN Controller: 1.5 % Layout Overhead: 27.0 %

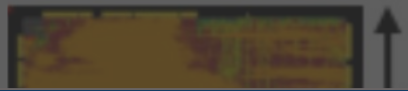


***** Effective TFLOPs is calculated by multiplying the base dense TFLOPs with the average efficiency computed across GEMMs.***

SIGMA achieves 5.7x higher effective TFLOPS for a 3.2x higher effective TFLOPS/W.

Systolic Array vs SIGMA Comparison

	TPU-like Systolic Engine		SIGMA Engine	
Technology	Commercial 28nm		Commercial 28nm	
	Multipliers:	59.0 %	Multipliers:	42.5 %
	Local Memory:	1.5 %	Local Memory:	1.0 %
	Layout Overhead:	25.0 %	Dist. NoC Overhead:	14.5 %
			Red. NoC Overhead:	3.0 %
			FAN Controller:	1.5 %
			Layout Overhead:	27.0 %



SIGMA consumes more resources but achieves higher effective TFLOPS/W.

multiplying the base dense TFLOPs with the average efficiency computed across GEMMs.

SIGMA achieves 5.7x higher effective TFLOPS for a 3.2x higher effective TFLOPS/W.

Outline

- Motivation
  - GEMMs in Deep Learning
  - Utilization on TPU
- Accelerator Requirements
- SIGMA
  - Interconnects Implementations
  - Full System Design
- Evaluation
- **Conclusion**

Conclusion

- **GEMM is a key component of Deep Learning workloads, but they are often irregular and sparse.**

Conclusion

- **GEMM is a key component of Deep Learning workloads, but they are often irregular and sparse.**
- **High utilization from systolic arrays is challenging because of their rigid structure.**

Conclusion

- **GEMM is a key component of Deep Learning workloads, but they are often irregular and sparse.**
- **High utilization from systolic arrays is challenging because of their rigid structure.**
- **SIGMA enables high compute utilization on sparse irregular GEMMs.**

Conclusion

- **GEMM is a key component of Deep Learning workloads, but they are often irregular and sparse.**
- **High utilization from systolic arrays is challenging because of their rigid structure.**
- **SIGMA enables high compute utilization on sparse irregular GEMMs.**
- **SIGMA performs 5.7x better than systolic arrays and 3x better than other state-of-the-art sparse accelerators at the cost of extra hardware, specifically for the O(1) distribution and the novel FAN reduction interconnects.**

Conclusion

- **GEMM is a key component of Deep Learning workloads, but they are often irregular and sparse.**
- **High utilization from systolic arrays is challenging because of their rigid structure.**
- **SIGMA enables high compute utilization on sparse irregular GEMMs.**
- **SIGMA performs 5.7x better than systolic arrays and 3x better than other state-of-the-art sparse accelerators at the cost of extra hardware, specifically for the O(1) distribution and the novel FAN reduction interconnects.**
- **SIGMA achieves 3.2x higher Effective TFLOPS/W than Systolic Arrays.**

Conclusion

- **GEMM is a key component of Deep Learning workloads, but they are often irregular and sparse.**
- **High utilization from systolic arrays is challenging because of their rigid structure.**
- **SIGMA enables high compute utilization on sparse irregular GEMMs.**
- **SIGMA performs 5.7x better than systolic arrays and 3x better than other state-of-the-art sparse accelerators at the cost of extra hardware, specifically for the O(1) distribution and the novel FAN reduction interconnects.**
- **SIGMA achieves 3.2x higher Effective TFLOPS/W than Systolic Arrays.**
- **Future work: Optimizations such as power gating and software stack design.**

Thank you! 😊

Material Backup

Walkthrough

Compressed bitmap equivalent

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

Stationary Bitmap

A	B	C	D	E	F	G	H	I

Stationary Nonzero data

a	b	c	d	e	f	g

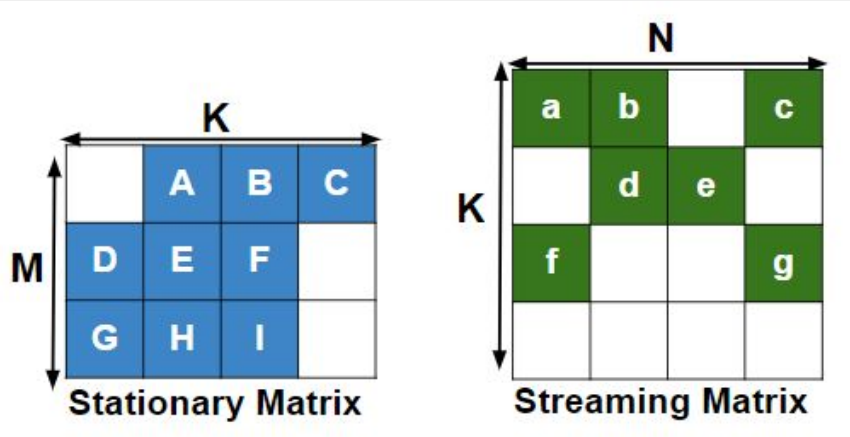
Streaming Nonzero data

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

Streaming Bitmap

Dim. Registers	
M-dim	3
N-dim	4
K-dim	4

Example GEMM



	A	B	C
D	E	F	
G	H	I	

Stationary Matrix

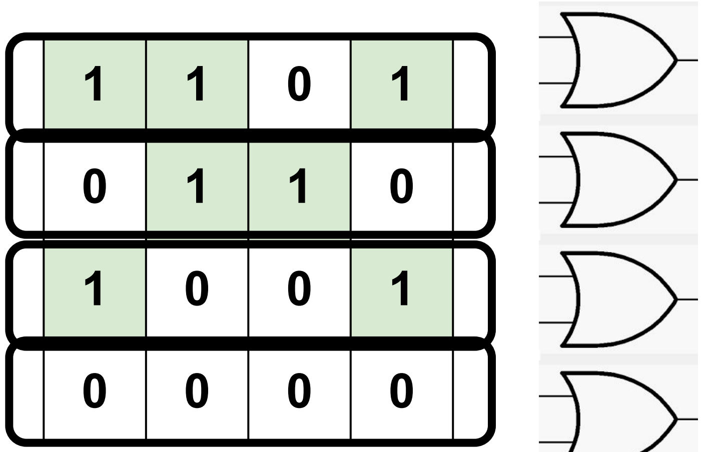
a	b		c
	d	e	
f			g

Streaming Matrix

Walkthrough Section

i) **Get bitmap from memory**
ii) Row wise OR on streaming matrix
iii) Element wise AND on stationary matrix
iv) Generate stationary’ bitmap
vi) Unicast stationary values
vii) Get source - destination pairs
viii) Multicast streaming values and reduce

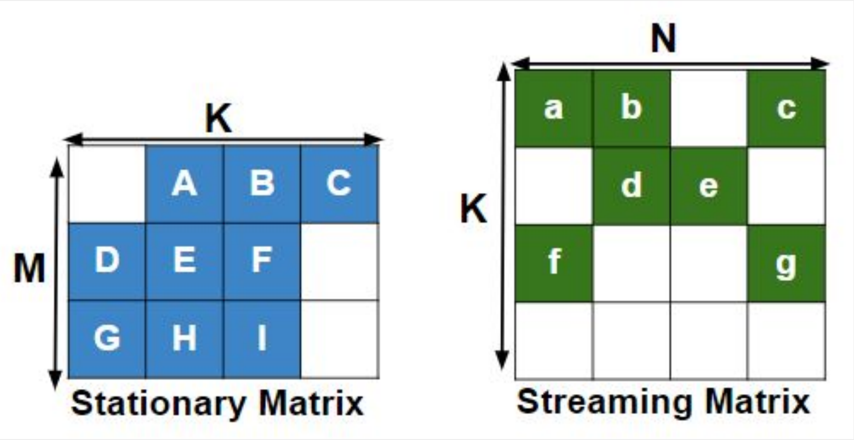
Walkthrough



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

Streaming Bitmap

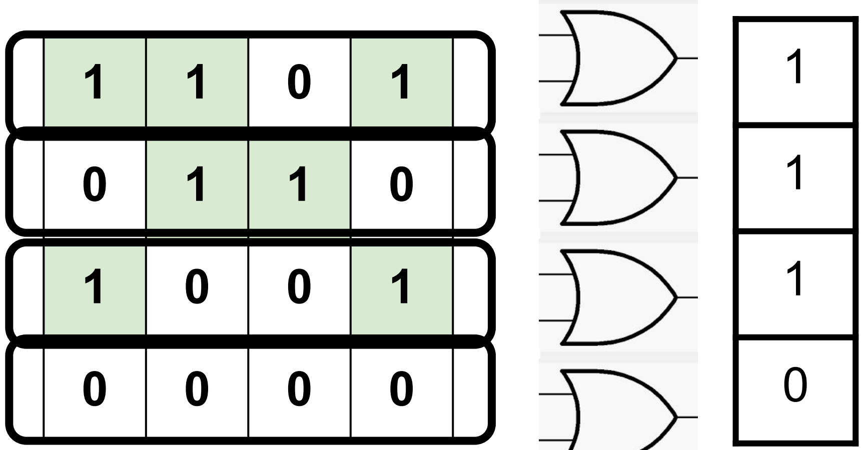
Example GEMM



Walkthrough Section

i) Get bitmap from memory
ii) **Row wise OR on streaming matrix**
iii) Element wise AND on stationary matrix
iv) Generate stationary’ bitmap
vi) Unicast stationary values
vii) Get source - destination pairs
viii) Multicast streaming values and reduce

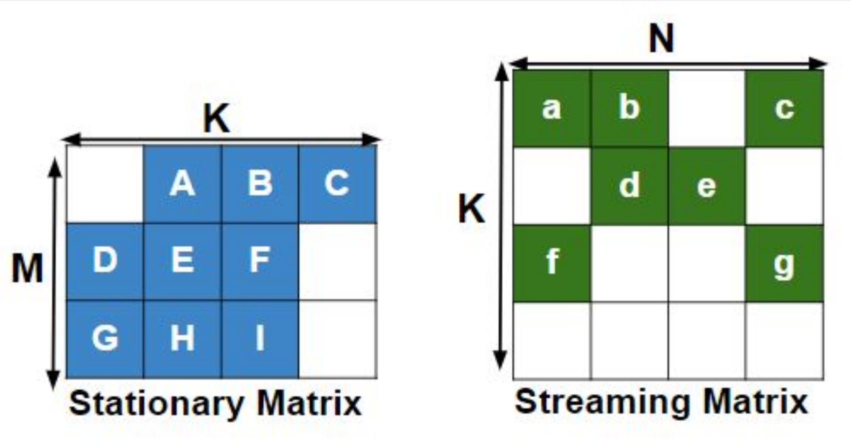
Walkthrough



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

Streaming Bitmap

Example GEMM



	A	B	C
D	E	F	
G	H	I	

Stationary Matrix

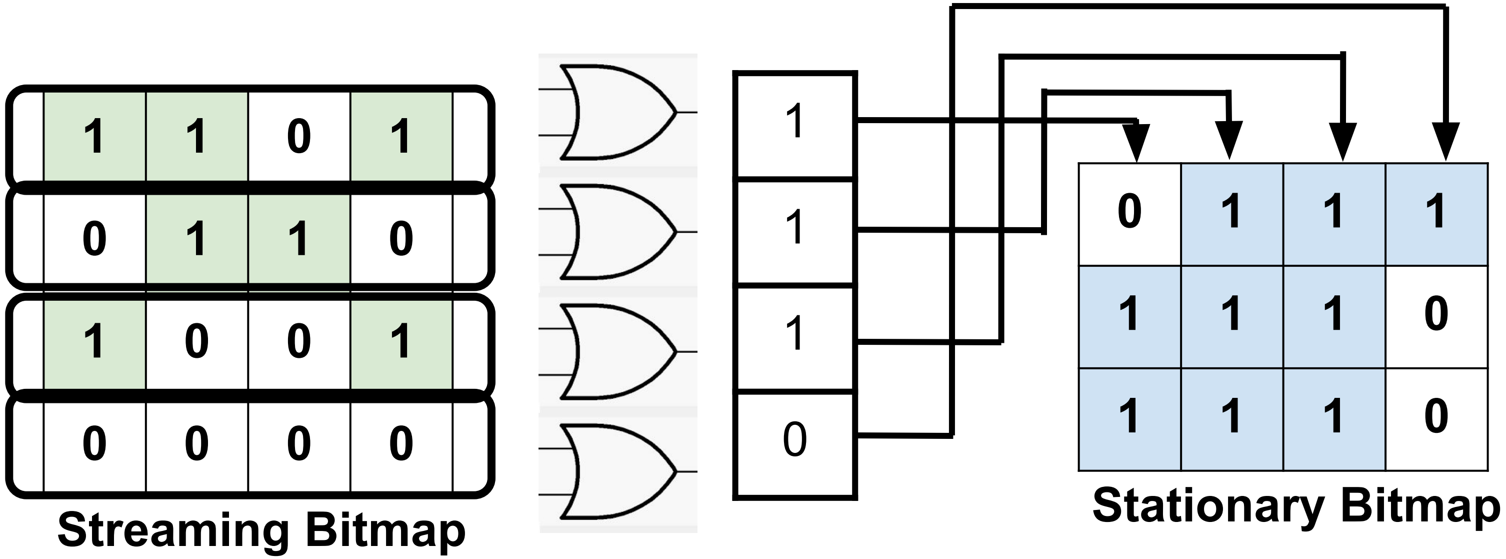
a	b		c
	d	e	
f			g

Streaming Matrix

Walkthrough Section

i) Get bitmap from memory
ii) Row wise OR on streaming matrix
iii) Element wise AND on stationary matrix
iv) Generate stationary’ bitmap
vi) Unicast stationary values
vii) Get source - destination pairs
viii) Multicast streaming values and reduce

Walkthrough



Streaming Bitmap

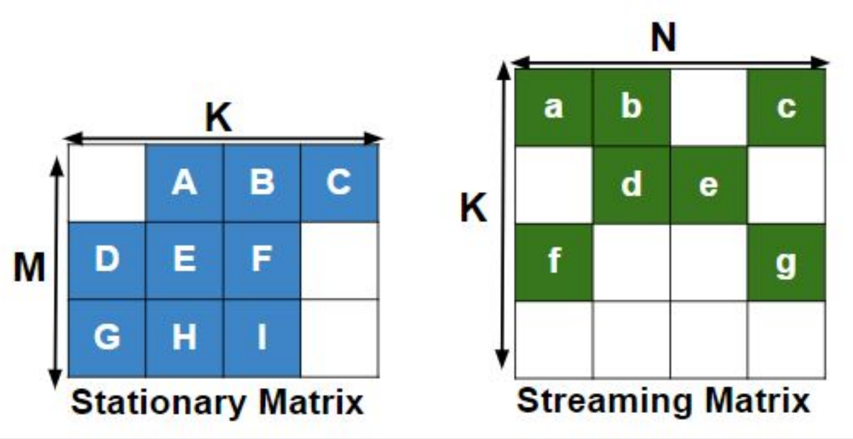
1	1	0	1
0	1	1	0
1	0	0	1
0	0	0	0

1
1
1
0

Stationary Bitmap

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

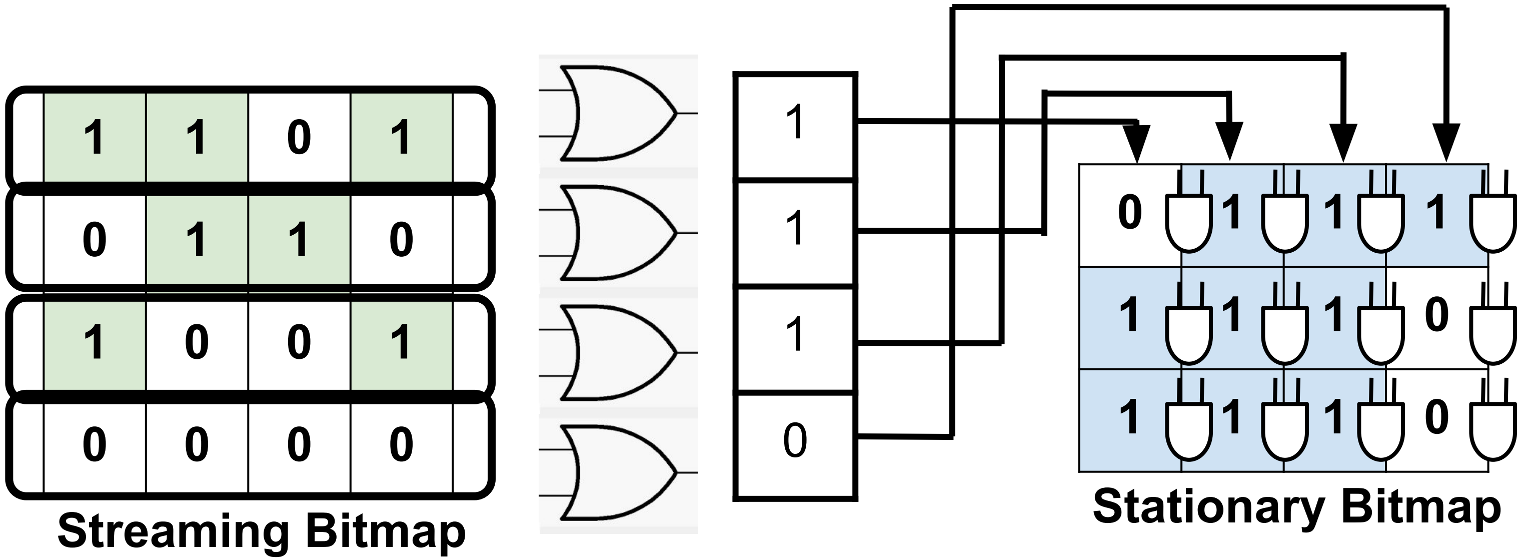
Example GEMM



Walkthrough Section

i) Get bitmap from memory
ii) Row wise OR on streaming matrix
iii) **Element wise AND on stationary matrix**
iv) Generate stationary’ bitmap
vi) Unicast stationary values
vii) Get source - destination pairs
viii) Multicast streaming values and reduce

Walkthrough



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

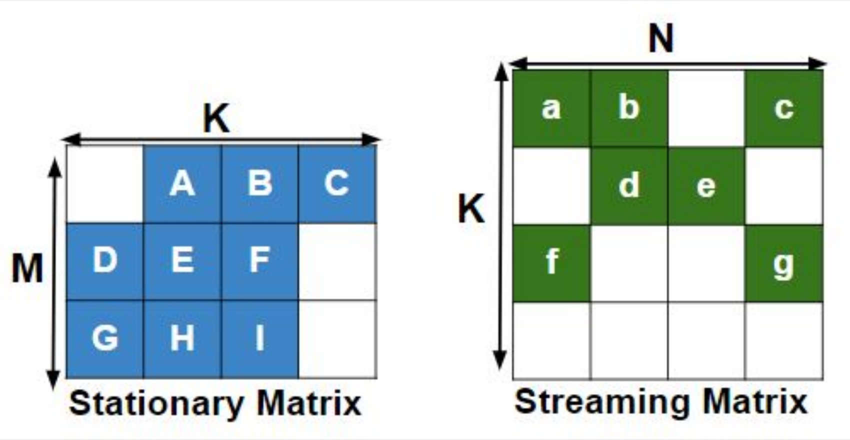
Streaming Bitmap

1
1
1
0

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

Stationary Bitmap

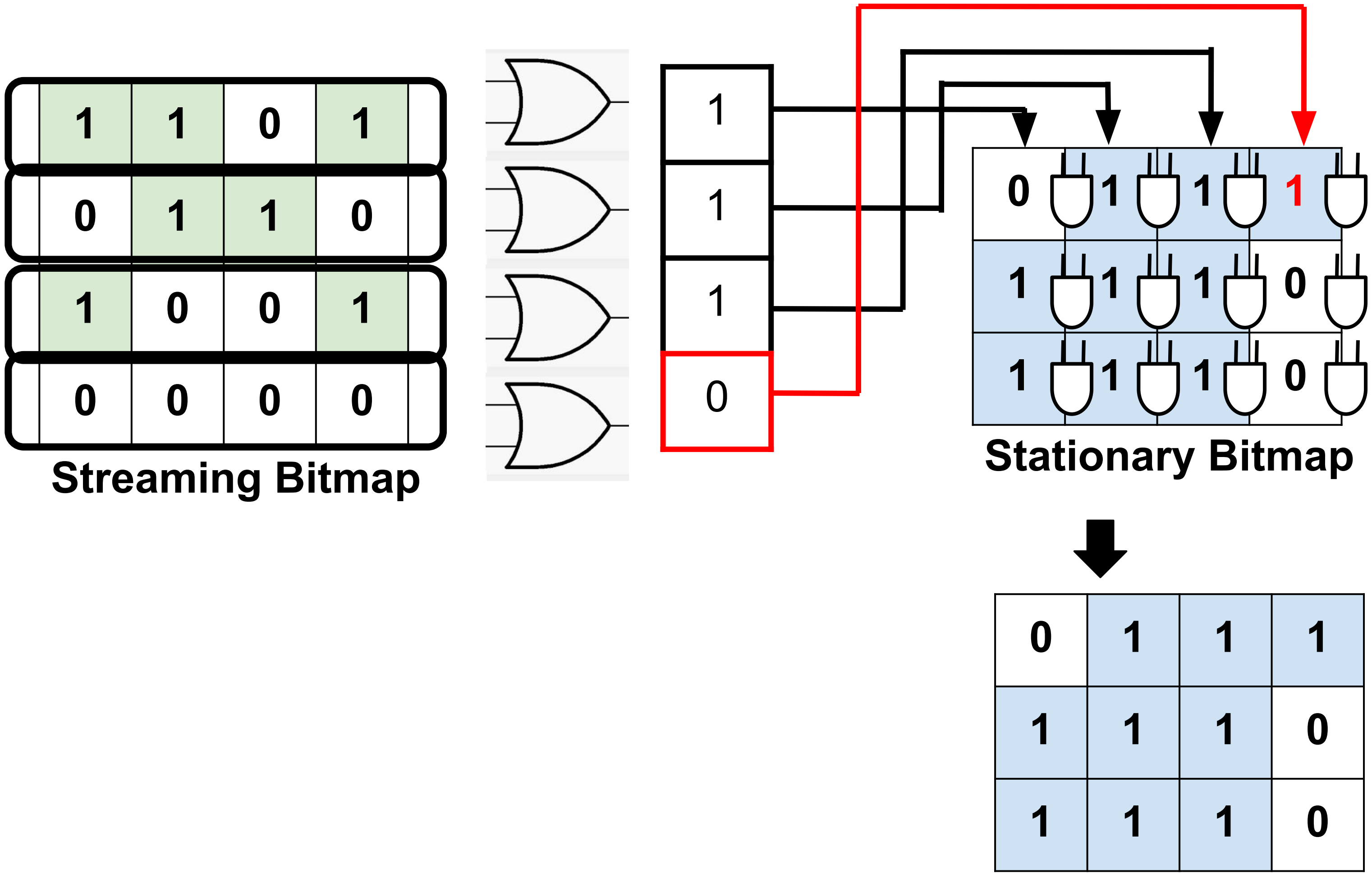
Example GEMM



Walkthrough Section

i) Get bitmap from memory
ii) Row wise OR on streaming matrix
iii) **Element wise AND on stationary matrix**
iv) Generate stationary’ bitmap
vi) Unicast stationary values
vii) Get source - destination pairs
viii) Multicast streaming values and reduce

Walkthrough



Streaming Bitmap

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

1
1
1
0

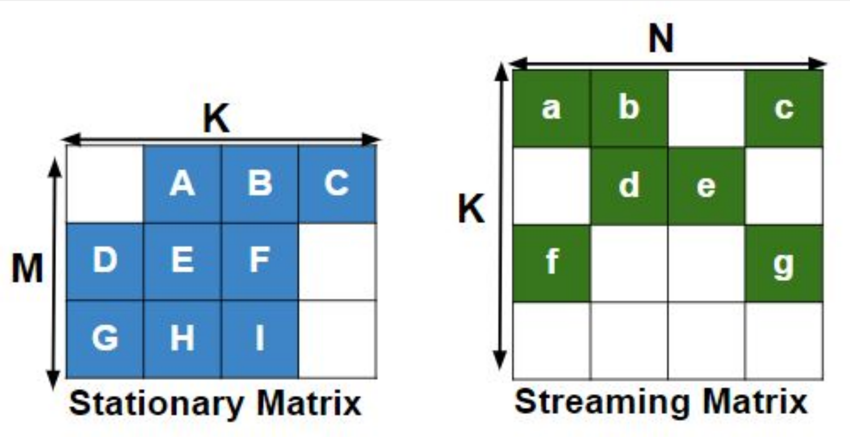
Stationary Bitmap

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

Stationary’ Bitmap

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

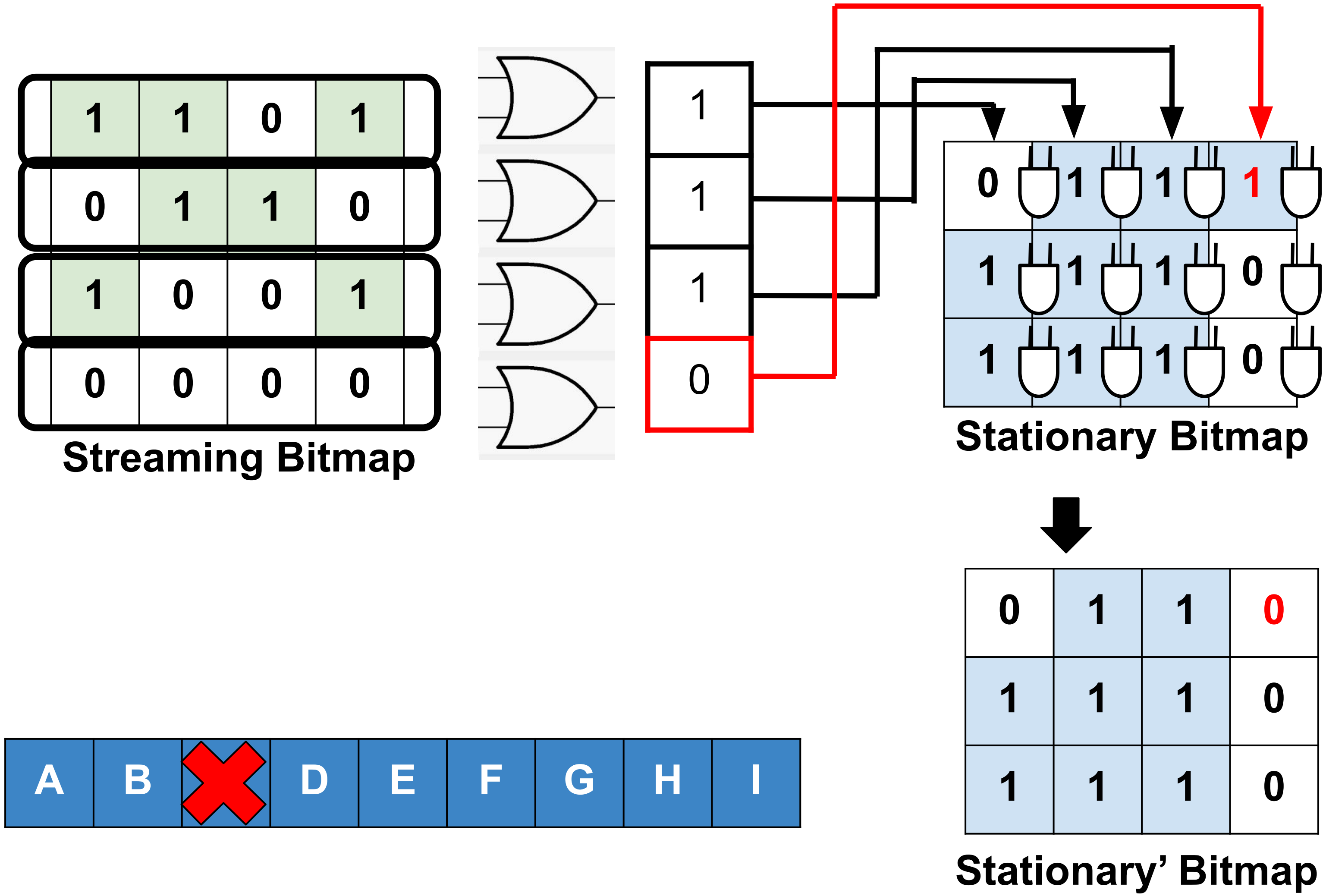
Example GEMM



Walkthrough Section

i) Get bitmap from memory
ii) Row wise OR on streaming matrix
iii) Element wise AND on stationary matrix
iv) **Generate stationary’ bitmap**
vi) Unicast stationary values
vii) Get source - destination pairs
viii) Multicast streaming values and reduce

Walkthrough



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

Streaming Bitmap

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

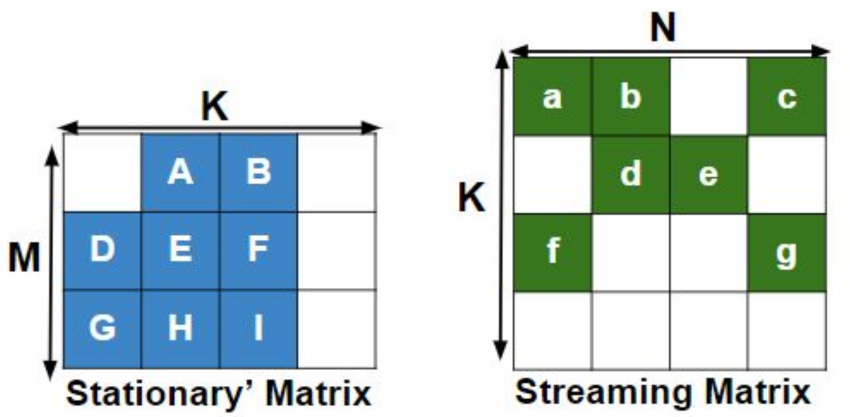
Stationary Bitmap

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

Stationary’ Bitmap

A	B	❌	D	E	F	G	H	I

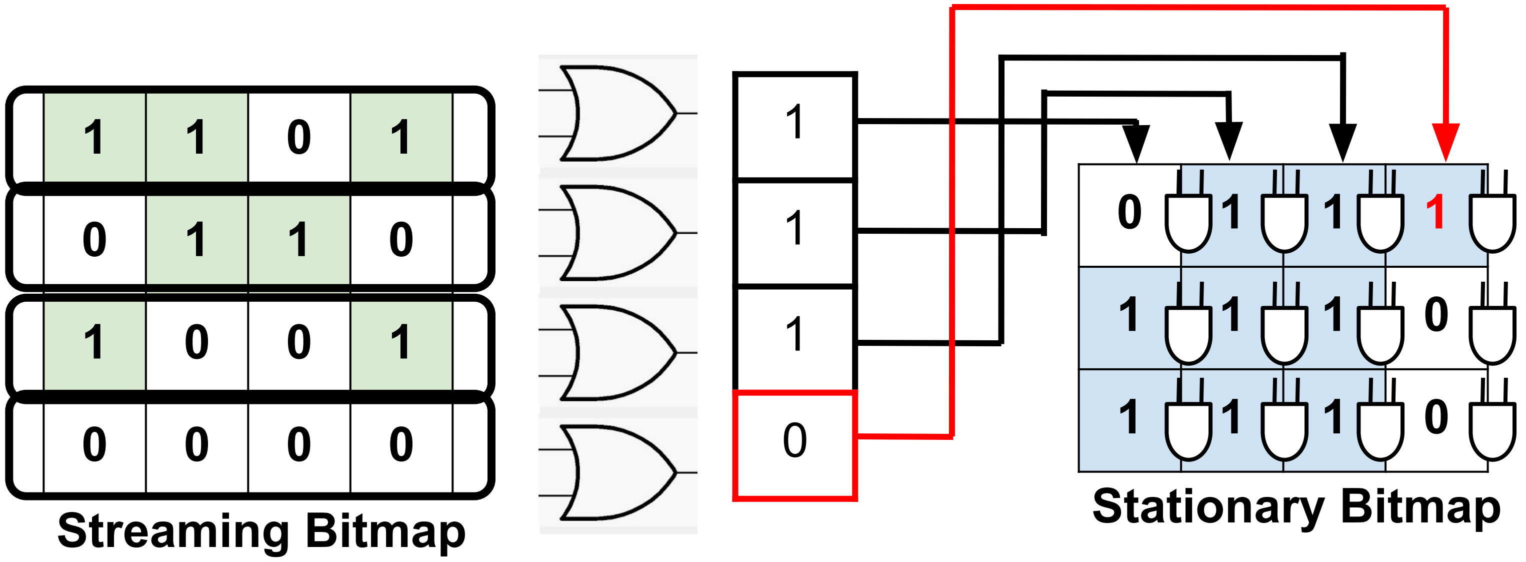
Example GEMM



Walkthrough Section

i) Get bitmap from memory
ii) Row wise OR on streaming matrix
iii) Element wise AND on stationary matrix
iv) Generate stationary’ bitmap
vi) Unicast stationary values
vii) Get source - destination pairs
viii) Multicast streaming values and reduce

Walkthrough



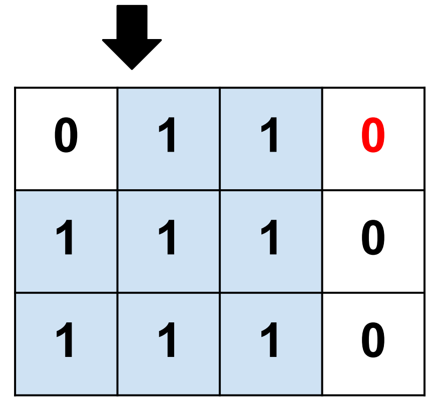
1	1	0	1
0	1	1	0
1	0	0	1
0	0	0	0

Streaming Bitmap

1
1
1
0

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

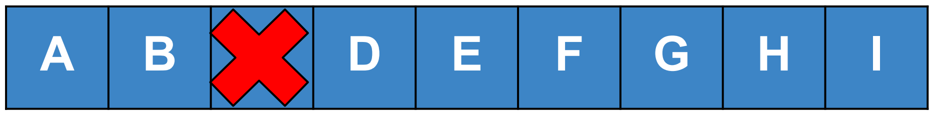
Stationary Bitmap



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

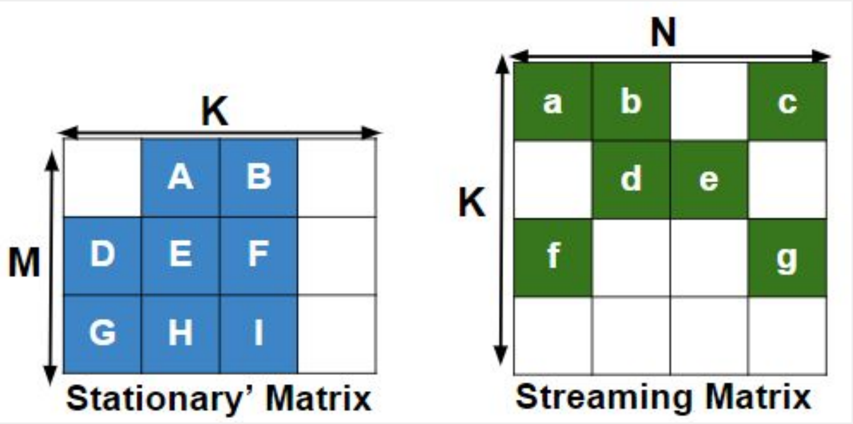
Stationary’ Bitmap

The main idea is to find the necessary nonzero elements to map stationary.



A	B	☒	D	E	F	G	H	I

Example GEMM



	A	B	
D	E	F	
G	H	I	

Stationary’ Matrix

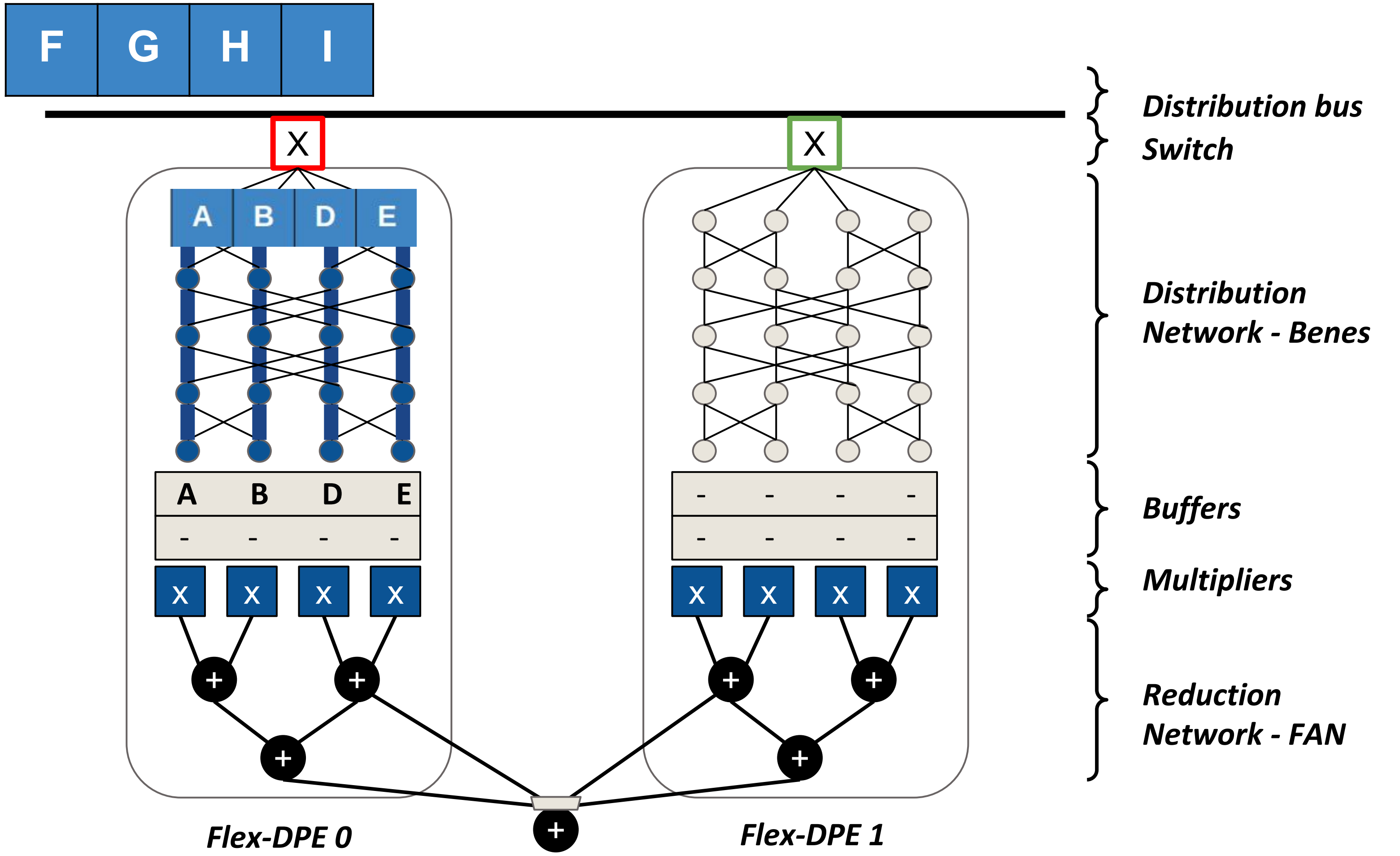
a	b		c
	d	e	
f			g

Streaming Matrix

Walkthrough Section

i) Get bitmap from memory
ii) Row wise OR on streaming matrix
iii) Element wise AND on stationary matrix
iv) Generate stationary’ bitmap
vi) Unicast stationary values
vii) Get source - destination pairs
viii) Multicast streaming values and reduce

Walkthrough



Cycle 1: unicast elements stationary to Flex-DPE 0

Example GEMM

Stationary' Matrix (M × K)

	A	B	
D	E	F	
G	H	I	

Streaming Matrix (K × N)

a	b		c
	d	e	
f			g

Walkthrough Section

i) Get bitmap from memory

ii) Row wise OR on streaming matrix

iii) Element wise AND on stationary matrix

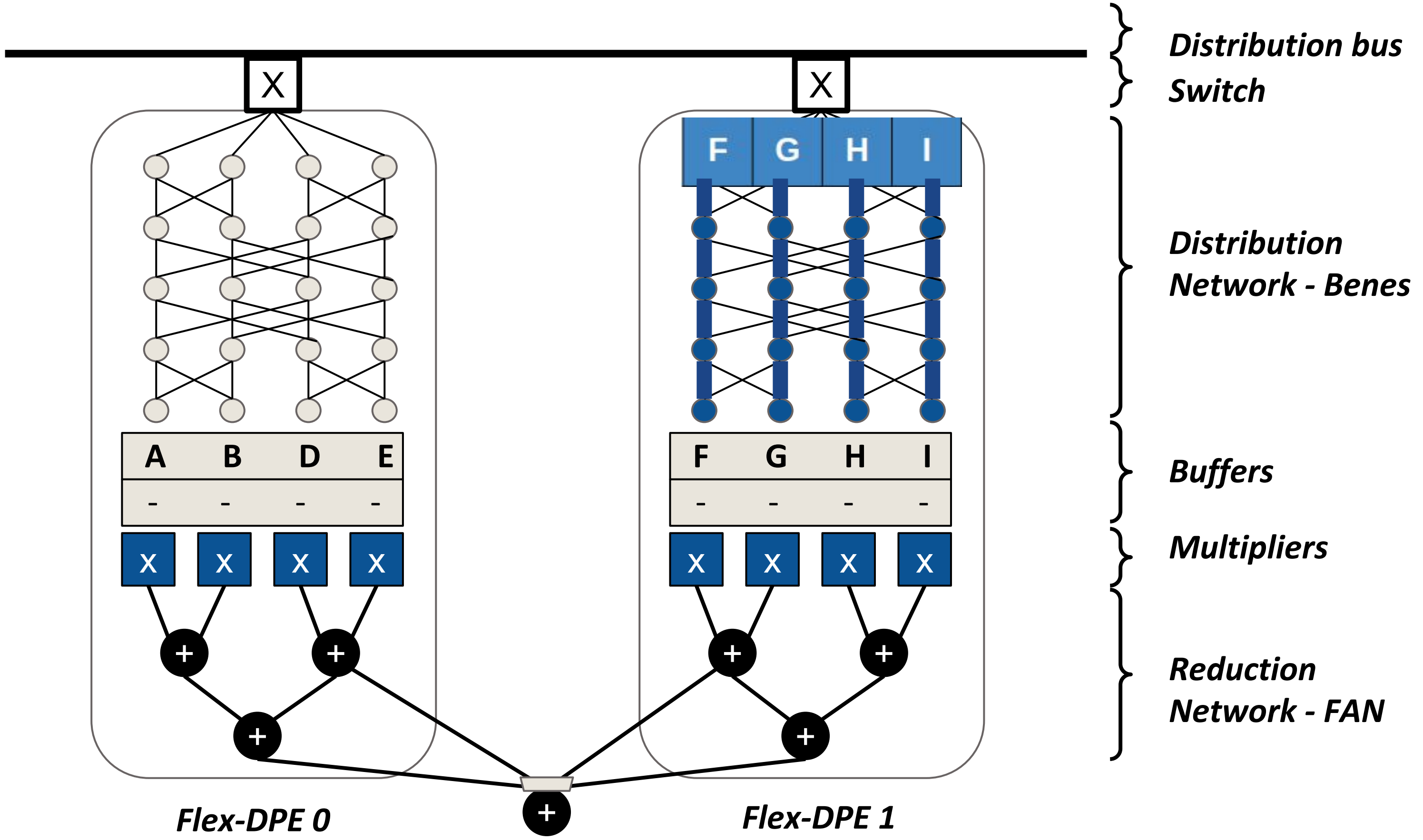
iv) Generate stationary' bitmap

vi) Unicast stationary values

vii) Get source - destination pairs

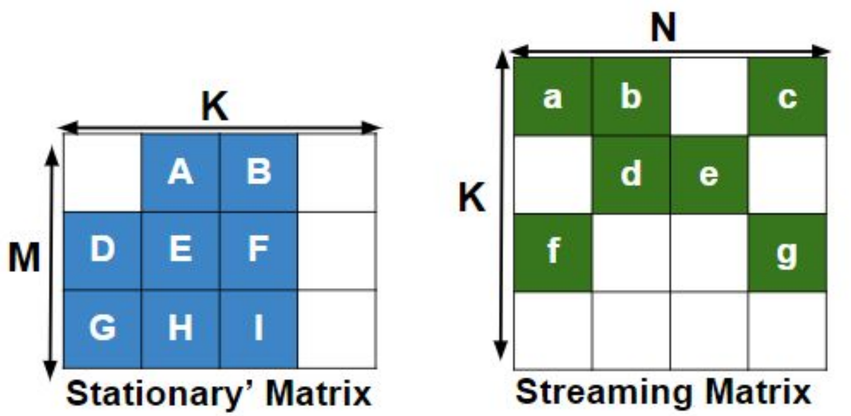
viii) Multicast streaming values and reduce

Walkthrough



Cycle 2: unicast elements stationary to Flex-DPE 1

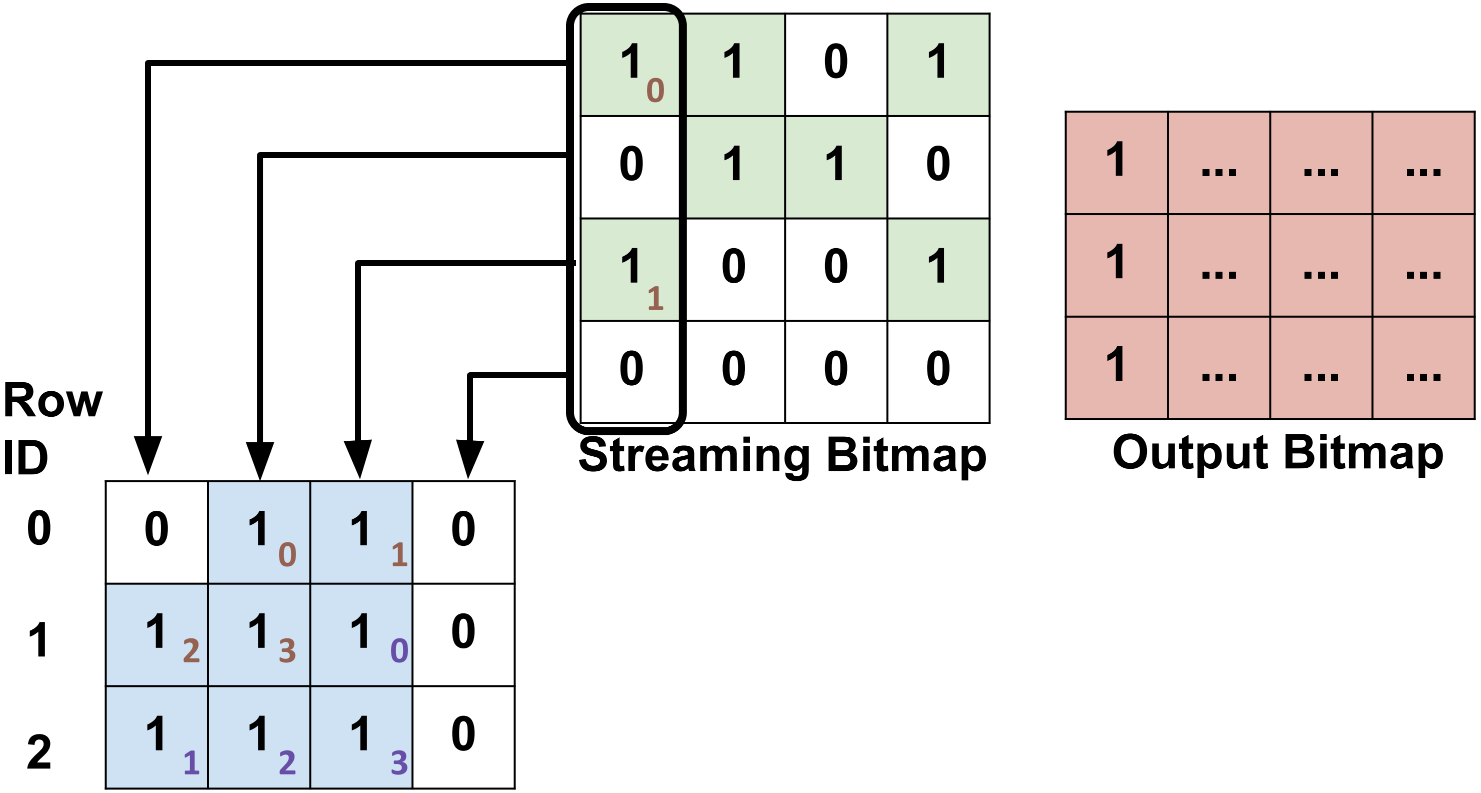
Example GEMM



Walkthrough Section

i) Get bitmap from memory
ii) Row wise OR on streaming matrix
iii) Element wise AND on stationary matrix
iv) Generate stationary' bitmap
vi) Unicast stationary values
vii) Get source - destination pairs
viii) Multicast streaming values and reduce

Walkthrough



1_0	1	0	1
0	1	1	0
1_1	0	0	1
0	0	0	0

Streaming Bitmap

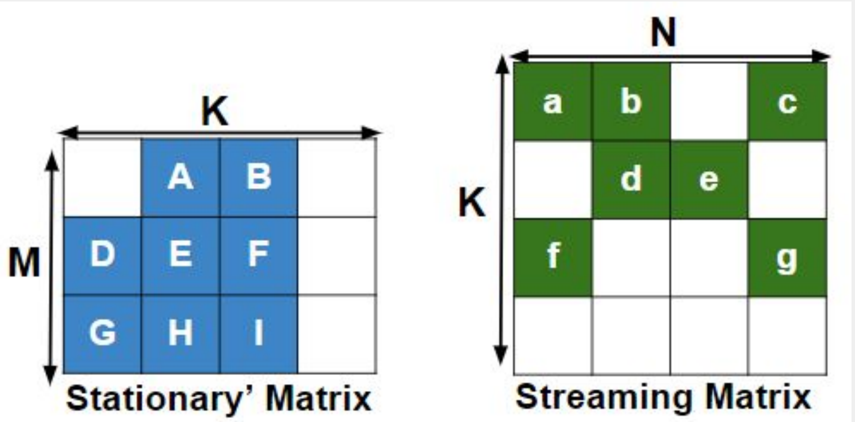
1	...	...	...
1	...	...	...
1	...	...	...

Output Bitmap

Row ID				
0	0	1_0	1_1	0
1	1_2	1_3	1_0	0
2	1_1	1_2	1_3	0

Stationary’ Bitmap

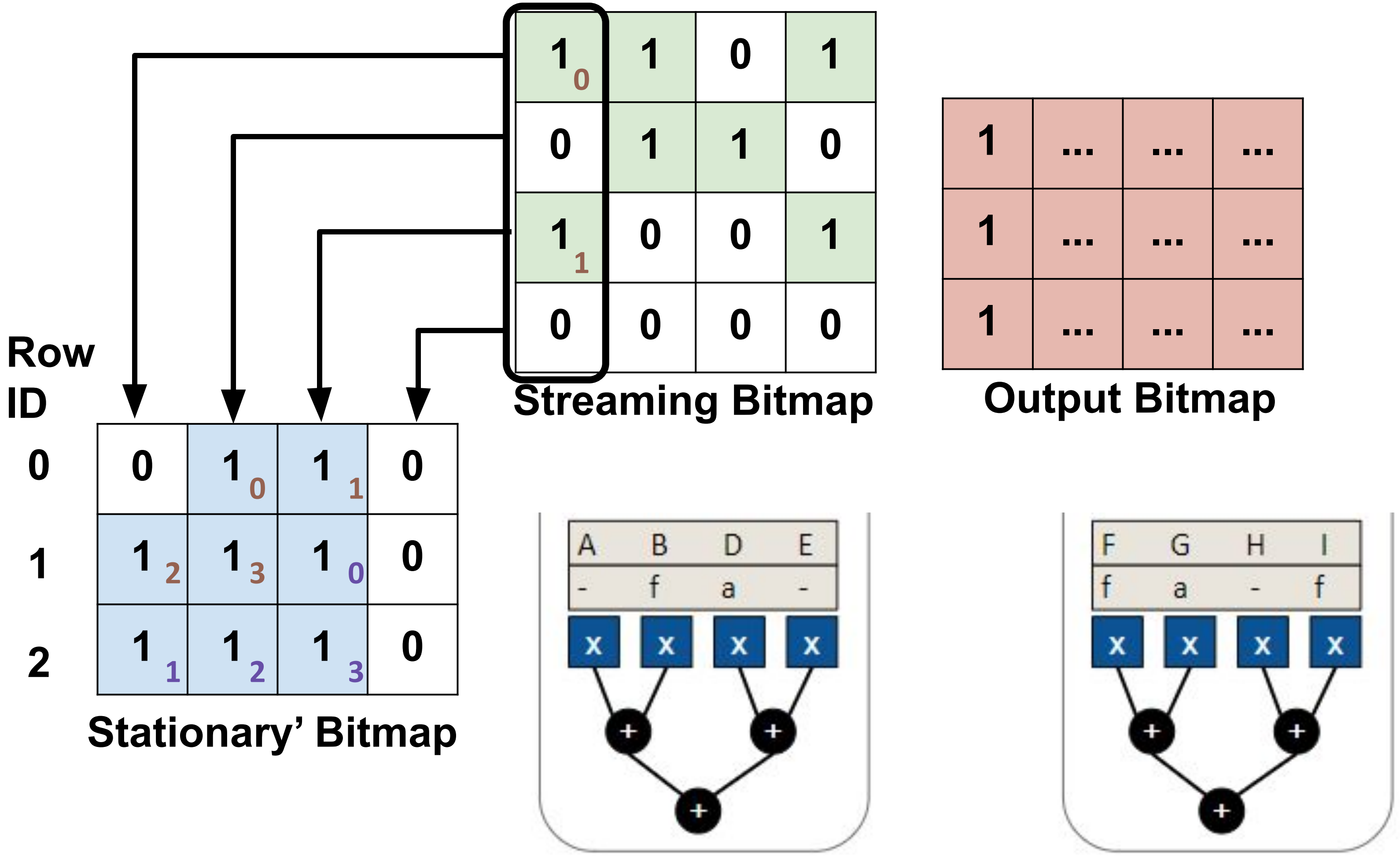
Example GEMM



Walkthrough Section

i) Get bitmap from memory
ii) Row wise OR on streaming matrix
iii) Element wise AND on stationary matrix
iv) Generate stationary’ bitmap
vi) Unicast stationary values
vii) Get source - destination pairs
viii) Multicast streaming values and reduce

Walkthrough



1_0	1	0	1
0	1	1	0
1_1	0	0	1
0	0	0	0

Streaming Bitmap

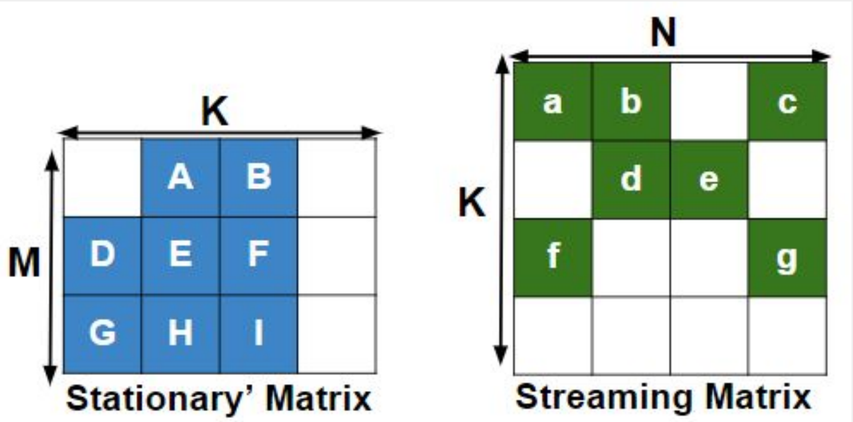
1	...	...	...
1	...	...	...
1	...	...	...

Output Bitmap

Row ID				
0	0	1_0	1_1	0
1	1_2	1_3	1_0	0
2	1_1	1_2	1_3	0

Stationary’ Bitmap

Example GEMM



	A	B	
D	E	F	
G	H	I	

Stationary’ Matrix

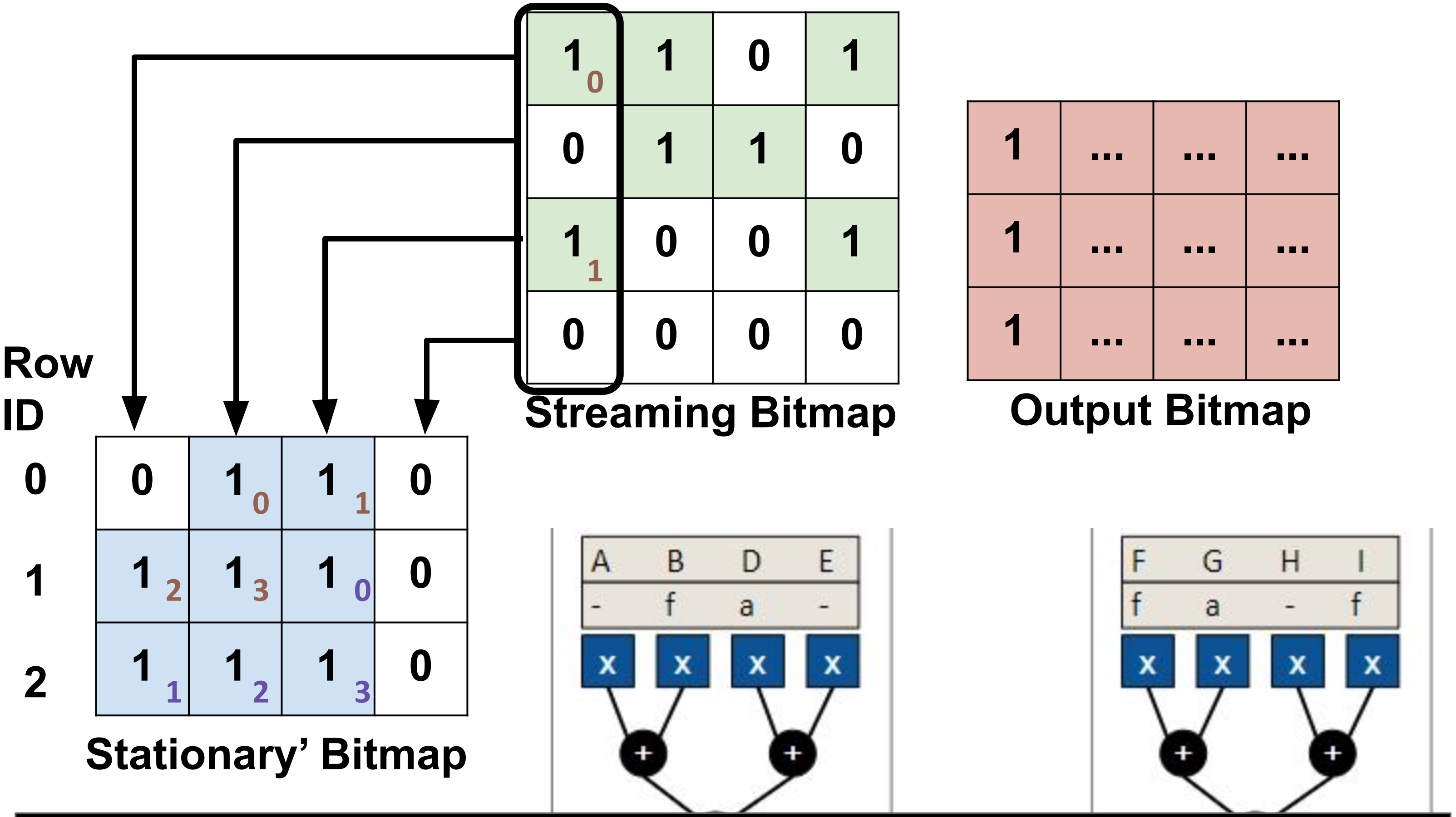
a	b		c
	d	e	
f			g

Streaming Matrix

Walkthrough Section

i) Get bitmap from memory
ii) Row wise OR on streaming matrix
iii) Element wise AND on stationary matrix
iv) Generate stationary’ bitmap
vi) Unicast stationary values
vii) Get source - destination pairs
viii) Multicast streaming values and reduce

Walkthrough



Streaming Bitmap

1_0	1	0	1
0	1	1	0
1_1	0	0	1
0	0	0	0

Output Bitmap

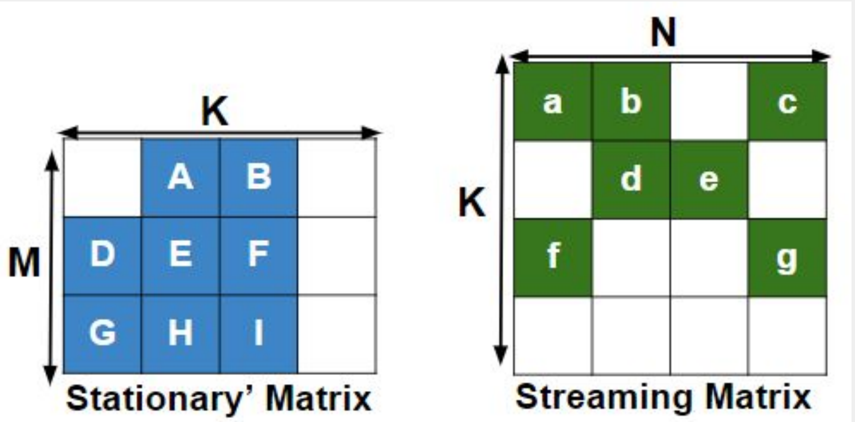
1	...	...	...
1	...	...	...
1	...	...	...

Stationary' Bitmap

Row ID				
0	0	1_0	1_1	0
1	1_2	1_3	1_0	0
2	1_1	1_2	1_3	0

The main idea is to find the matching source and destination indices.

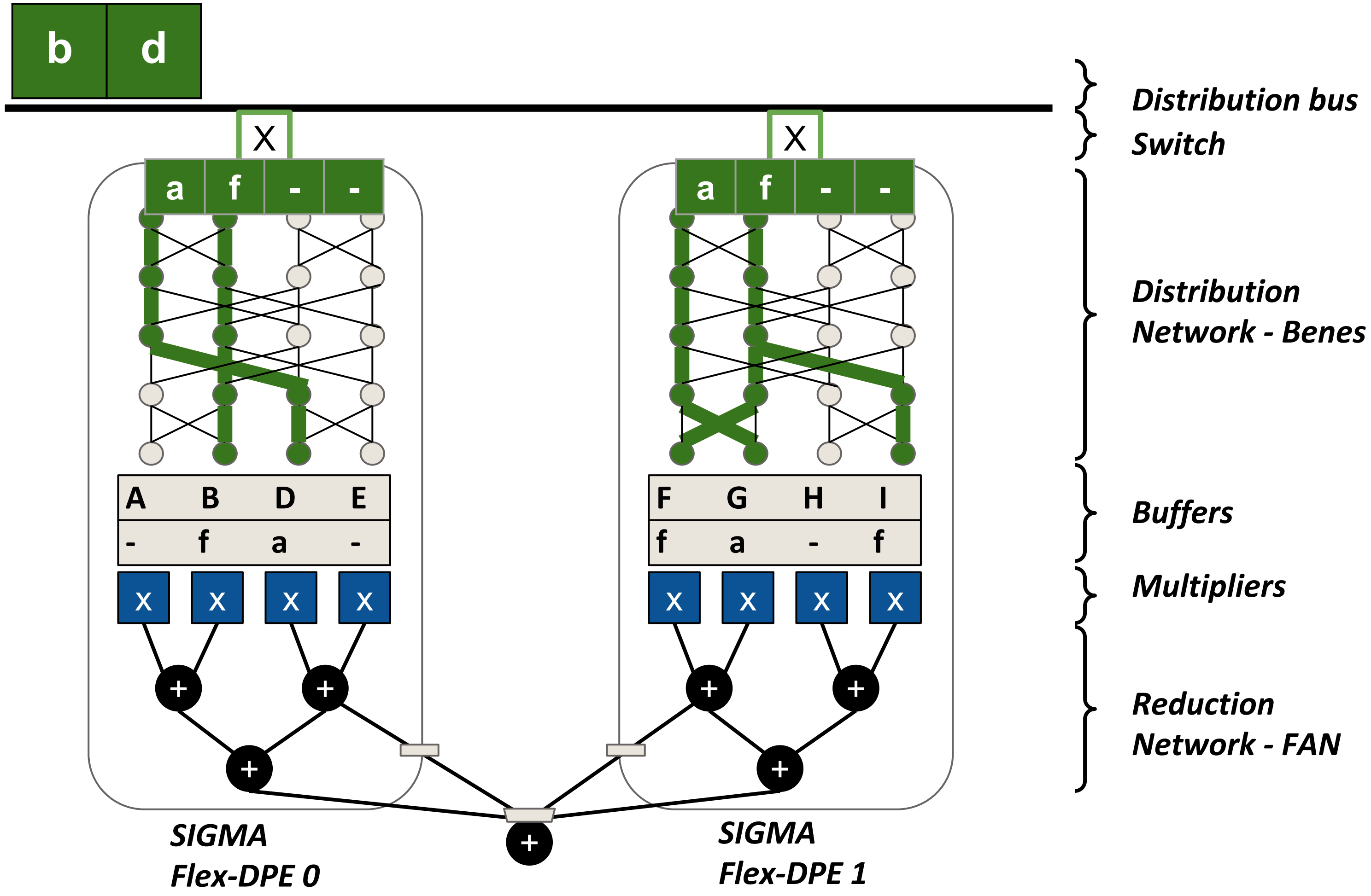
Example GEMM



Walkthrough Section

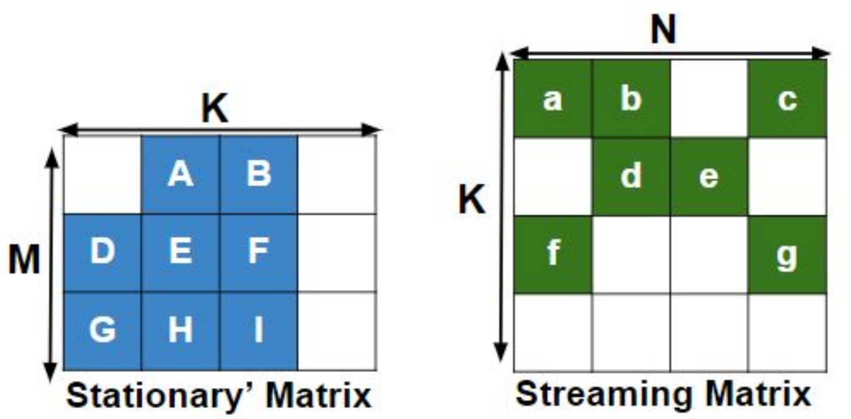
i) Get bitmap from memory
ii) Row wise OR on streaming matrix
iii) Element wise AND on stationary matrix
iv) Generate stationary' bitmap
vi) Unicast stationary values
vii) Get source - destination pairs
viii) Multicast streaming values and reduce

Walkthrough



Cycle 3: multicast 1st column of streaming matrix and reduce

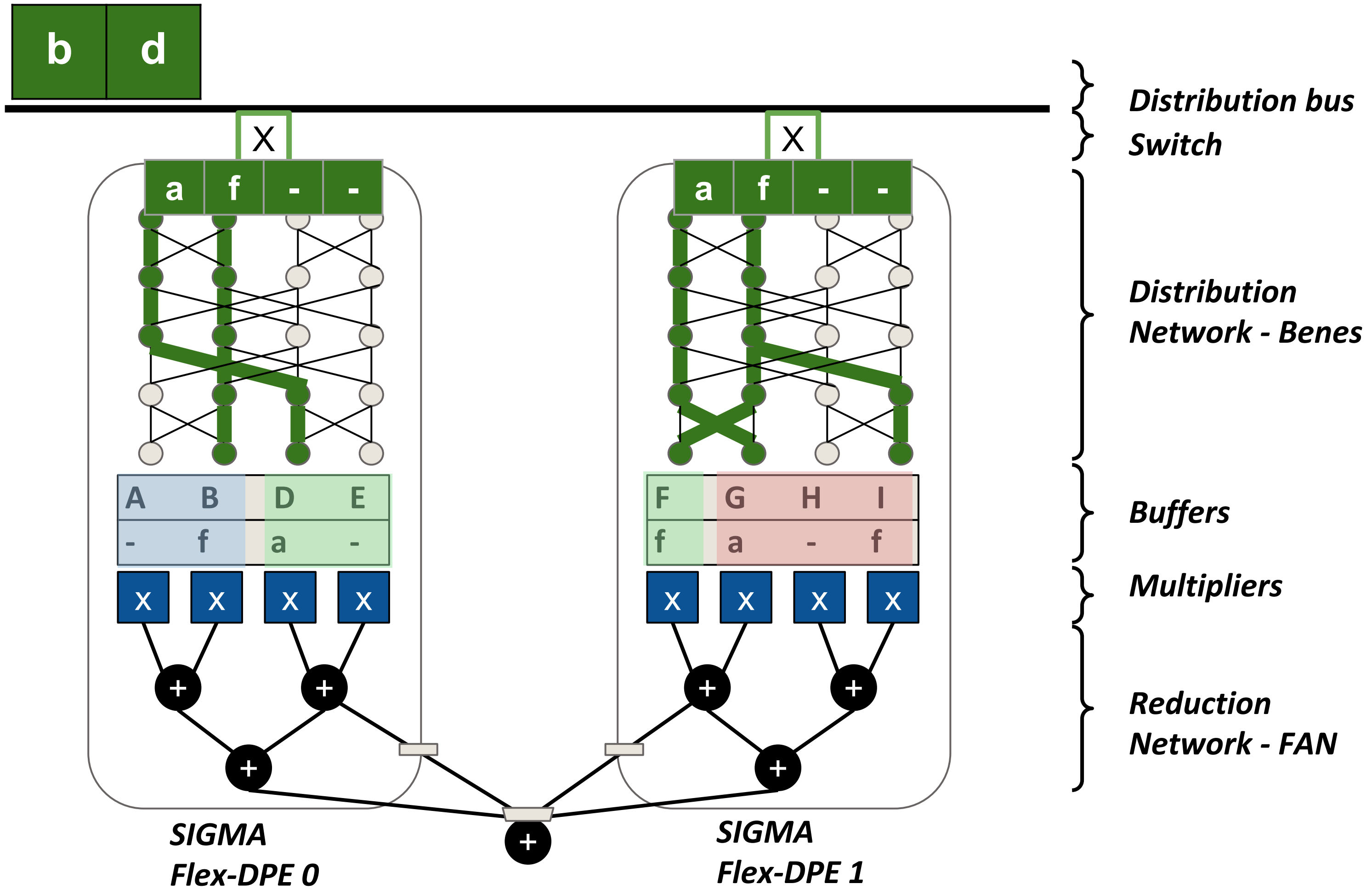
Example GEMM



Walkthrough Section

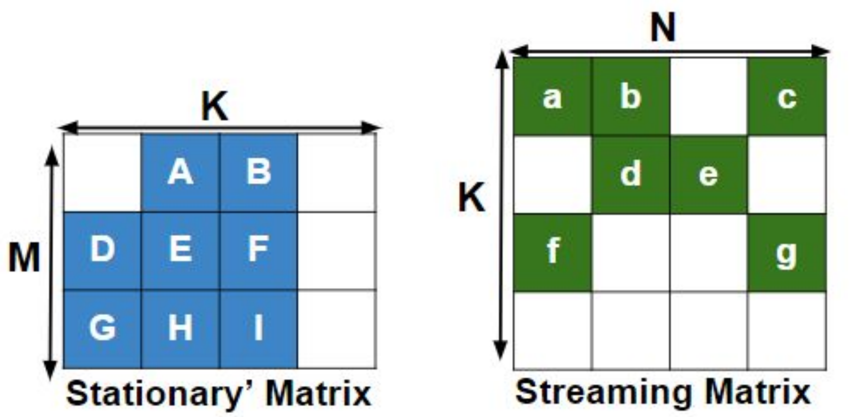
i) Get bitmap from memory
ii) Row wise OR on streaming matrix
iii) Element wise AND on stationary matrix
iv) Generate stationary' bitmap
vi) Unicast stationary values
vii) Get source - destination pairs
viii) Multicast streaming values and reduce

Walkthrough



Cycle 3: multicast 1st column of streaming matrix and reduce

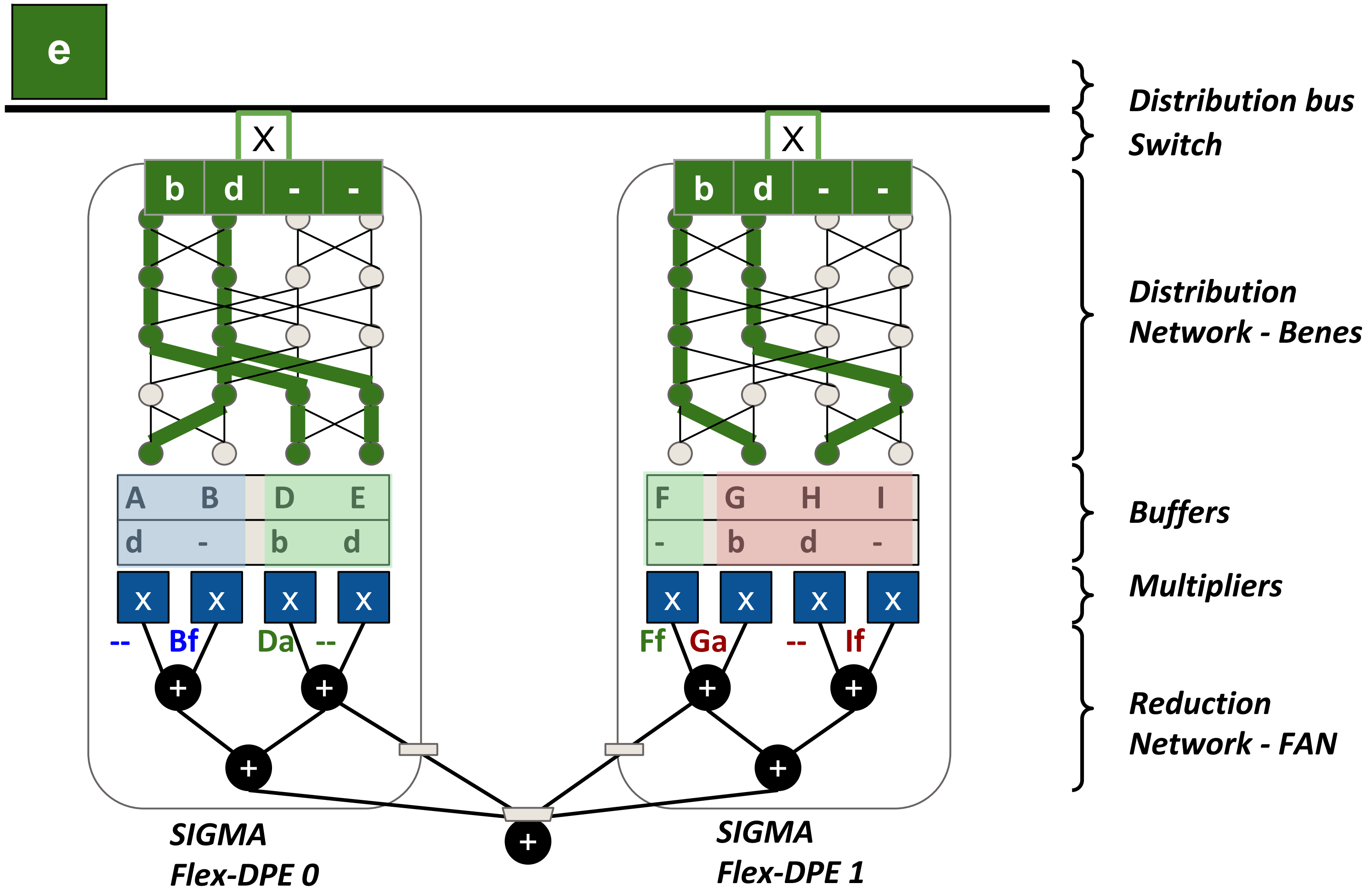
Example GEMM



Walkthrough Section

i) Get bitmap from memory
ii) Row wise OR on streaming matrix
iii) Element wise AND on stationary matrix
iv) Generate stationary' bitmap
vi) Unicast stationary values
vii) Get source - destination pairs
viii) Multicast streaming values and reduce

Walkthrough



Distribution bus

Switch

Distribution Network - Benes

Buffers

Multipliers

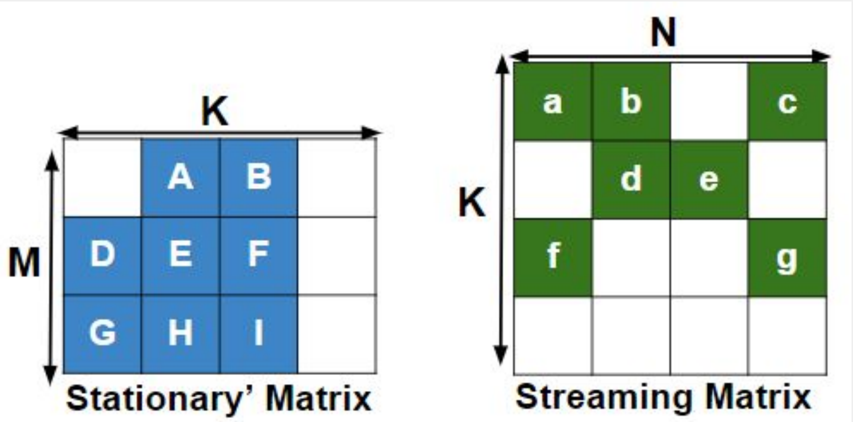
Reduction Network - FAN

SIGMA Flex-DPE 0

SIGMA Flex-DPE 1

Cycle 4: multicast 2nd column of streaming matrix and reduce

Example GEMM



	A	B	
D	E	F	
G	H	I	

Stationary' Matrix

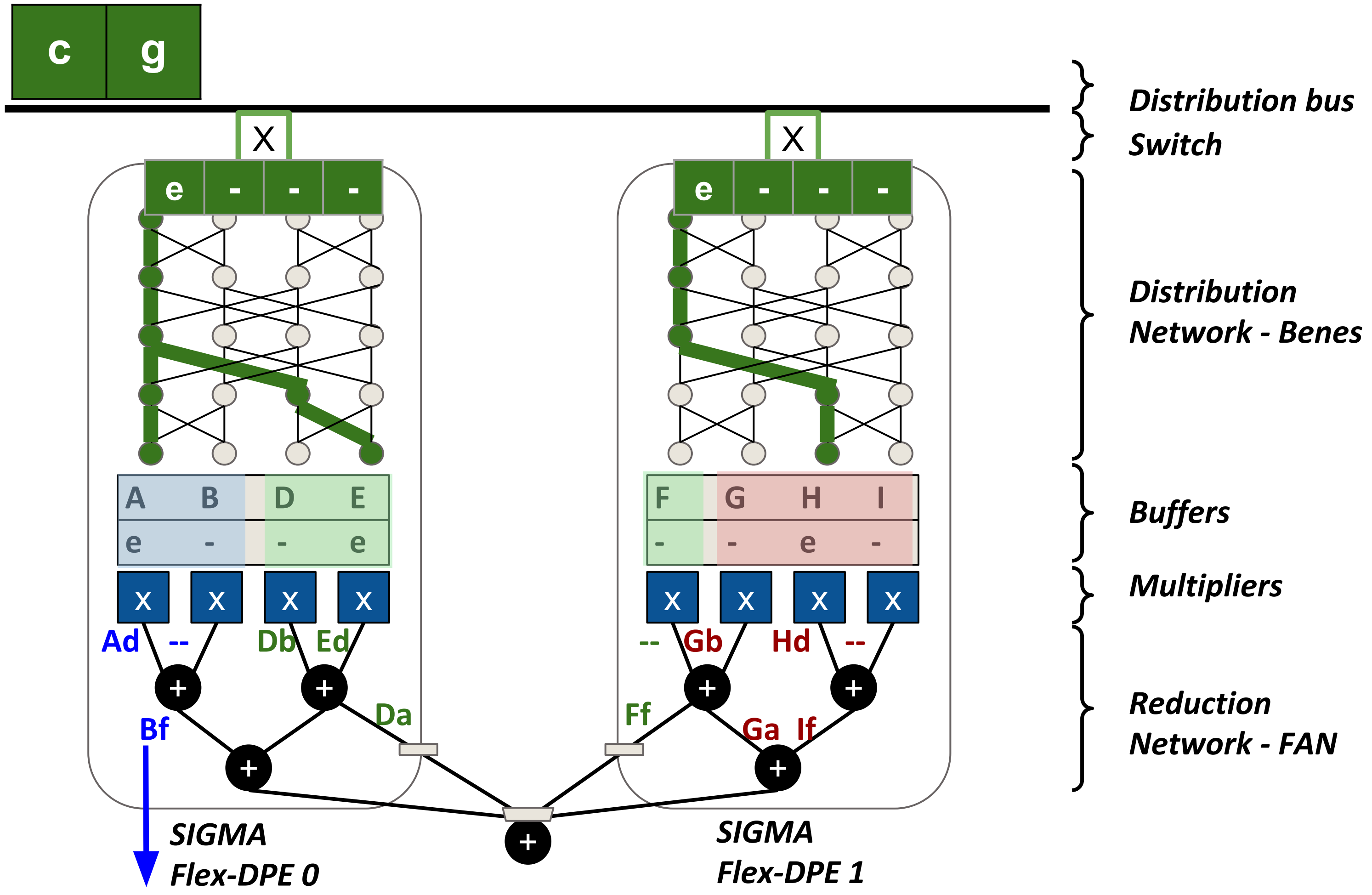
a	b		c
	d	e	
f			g

Streaming Matrix

Walkthrough Section

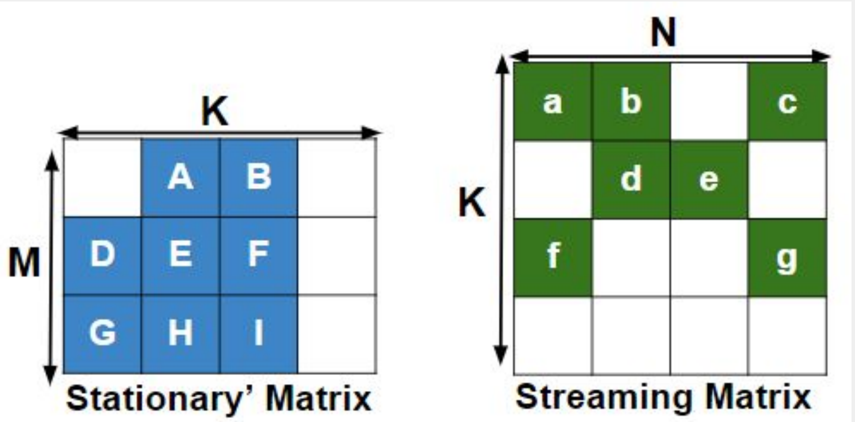
i) Get bitmap from memory
ii) Row wise OR on streaming matrix
iii) Element wise AND on stationary matrix
iv) Generate stationary' bitmap
vi) Unicast stationary values
vii) Get source - destination pairs
viii) Multicast streaming values and reduce

Walkthrough



Cycle 5: multicast 3rd column of streaming matrix and reduce

Example GEMM



	A	B	
D	E	F	
G	H	I	

Stationary' Matrix

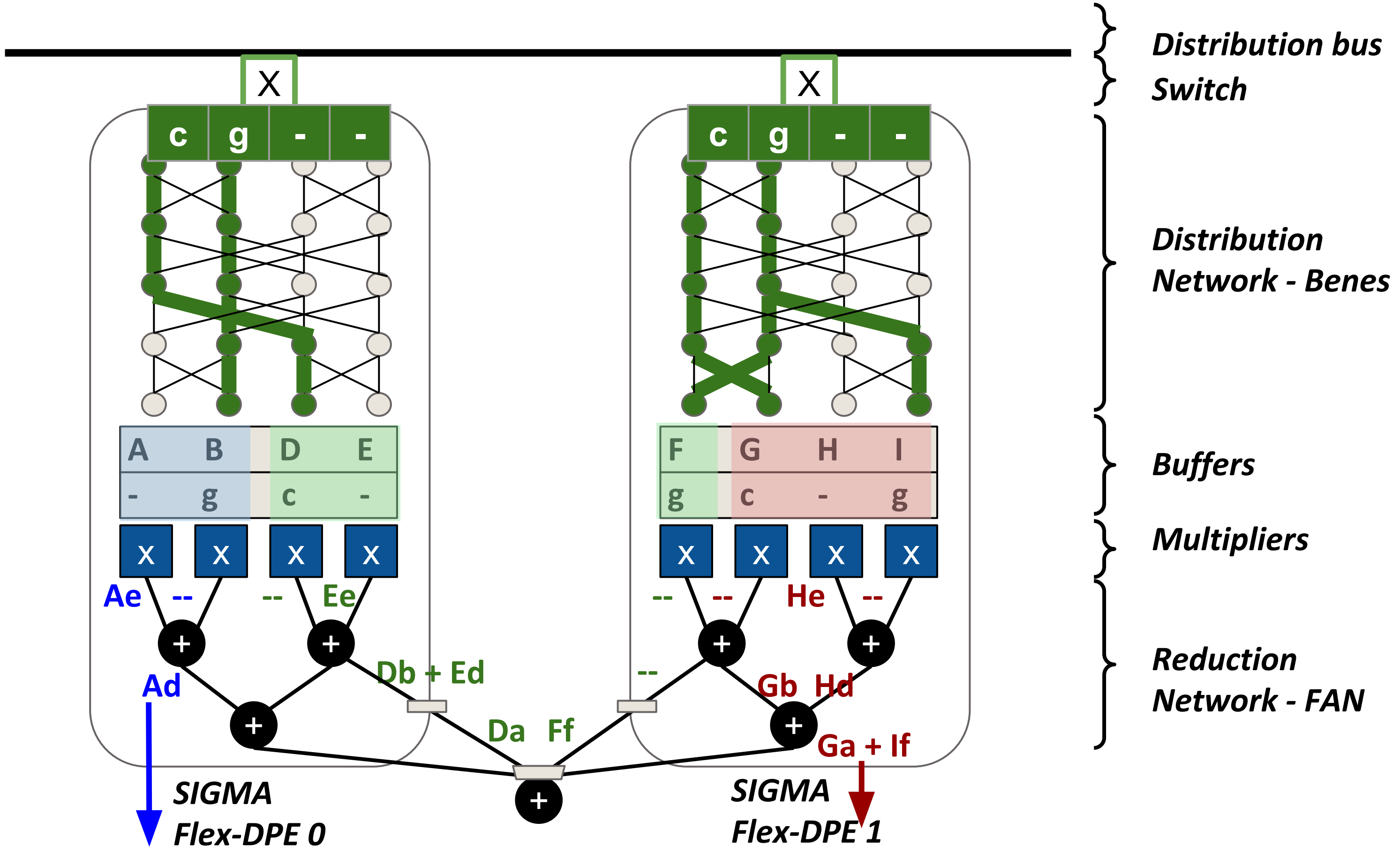
a	b		c
	d	e	
f			g

Streaming Matrix

Walkthrough Section

i) Get bitmap from memory
ii) Row wise OR on streaming matrix
iii) Element wise AND on stationary matrix
iv) Generate stationary' bitmap
vi) Unicast stationary values
vii) Get source - destination pairs
viii) Multicast streaming values and reduce

Walkthrough



Distribution bus

Switch

Distribution Network - Benes

Buffers

Multipliers

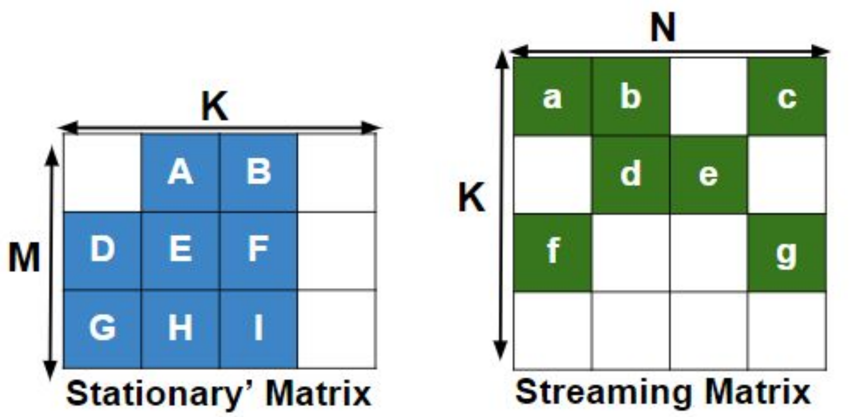
Reduction Network - FAN

SIGMA Flex-DPE 0

SIGMA Flex-DPE 1

Cycle 6: multicast last column of streaming matrix and reduce

Example GEMM



Stationary' Matrix

	A	B	
D	E	F	
G	H	I	

Streaming Matrix

a	b		c
	d	e	
f			g

Walkthrough Section

i) Get bitmap from memory

ii) Row wise OR on streaming matrix

iii) Element wise AND on stationary matrix

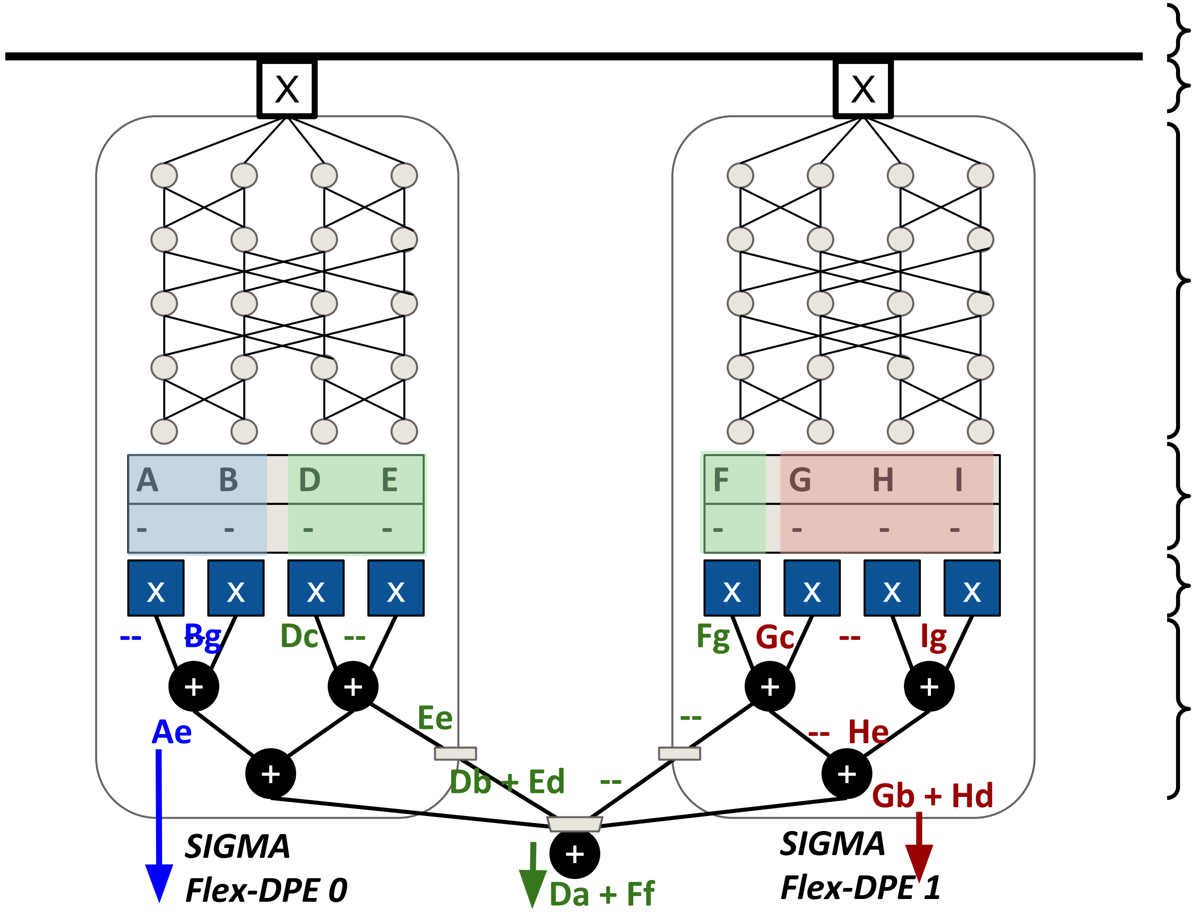
iv) Generate stationary' bitmap

vi) Unicast stationary values

vii) Get source - destination pairs

viii) Multicast streaming values and reduce

Walkthrough



Distribution bus

Switch

Distribution Network - Benes

Buffers

Multipliers

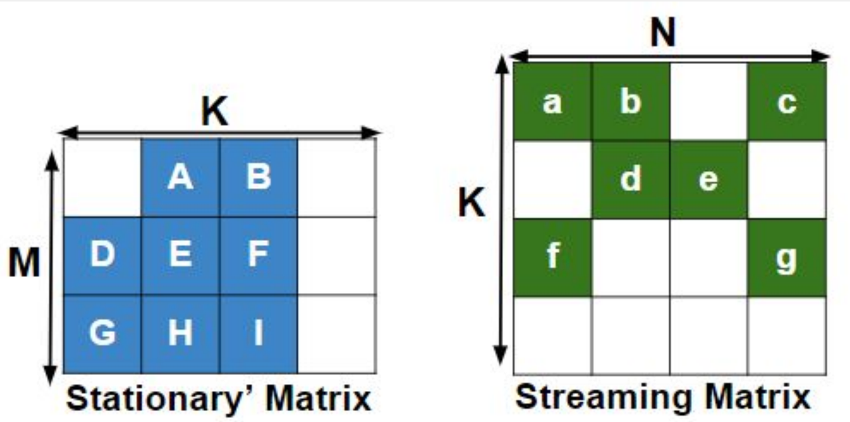
Reduction Network - FAN

SIGMA Flex-DPE 0

SIGMA Flex-DPE 1

Cycle 7: FAN Reduction

Example GEMM



Walkthrough Section

i) Get bitmap from memory

ii) Row wise OR on streaming matrix

iii) Element wise AND on stationary matrix

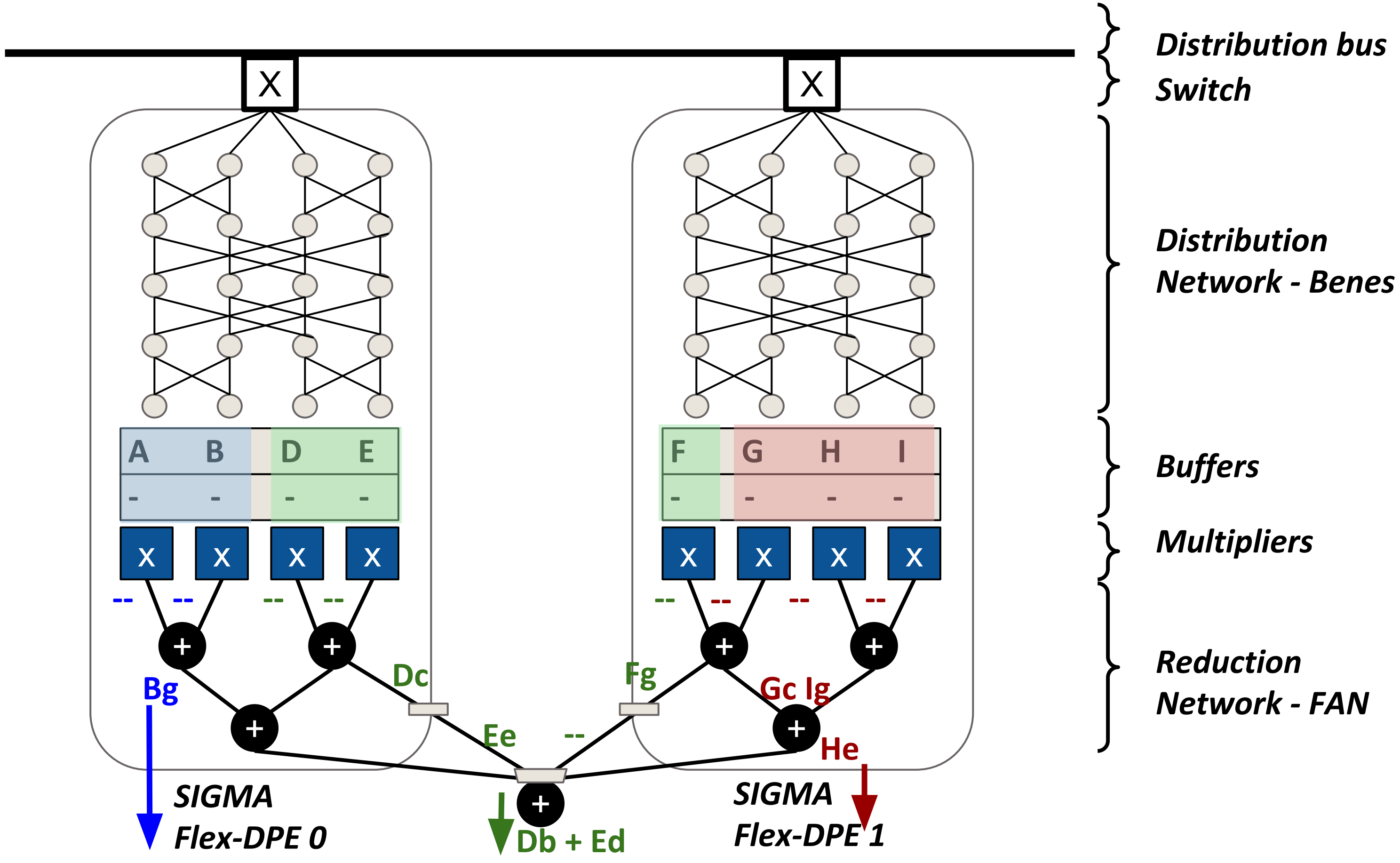
iv) Generate stationary' bitmap

vi) Unicast stationary values

vii) Get source - destination pairs

viii) Multicast streaming values and reduce

Walkthrough



Distribution bus

Switch

Distribution Network - Benes

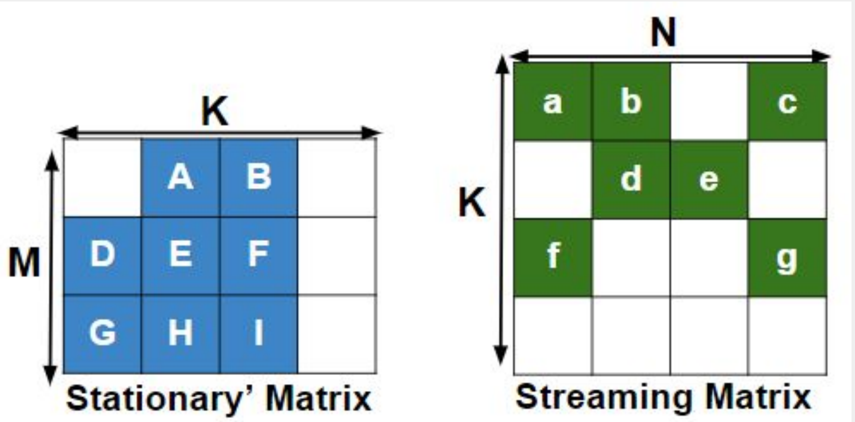
Buffers

Multipliers

Reduction Network - FAN

Cycle 8: FAN Reduction

Example GEMM



Stationary' Matrix

	A	B	
D	E	F	
G	H	I	

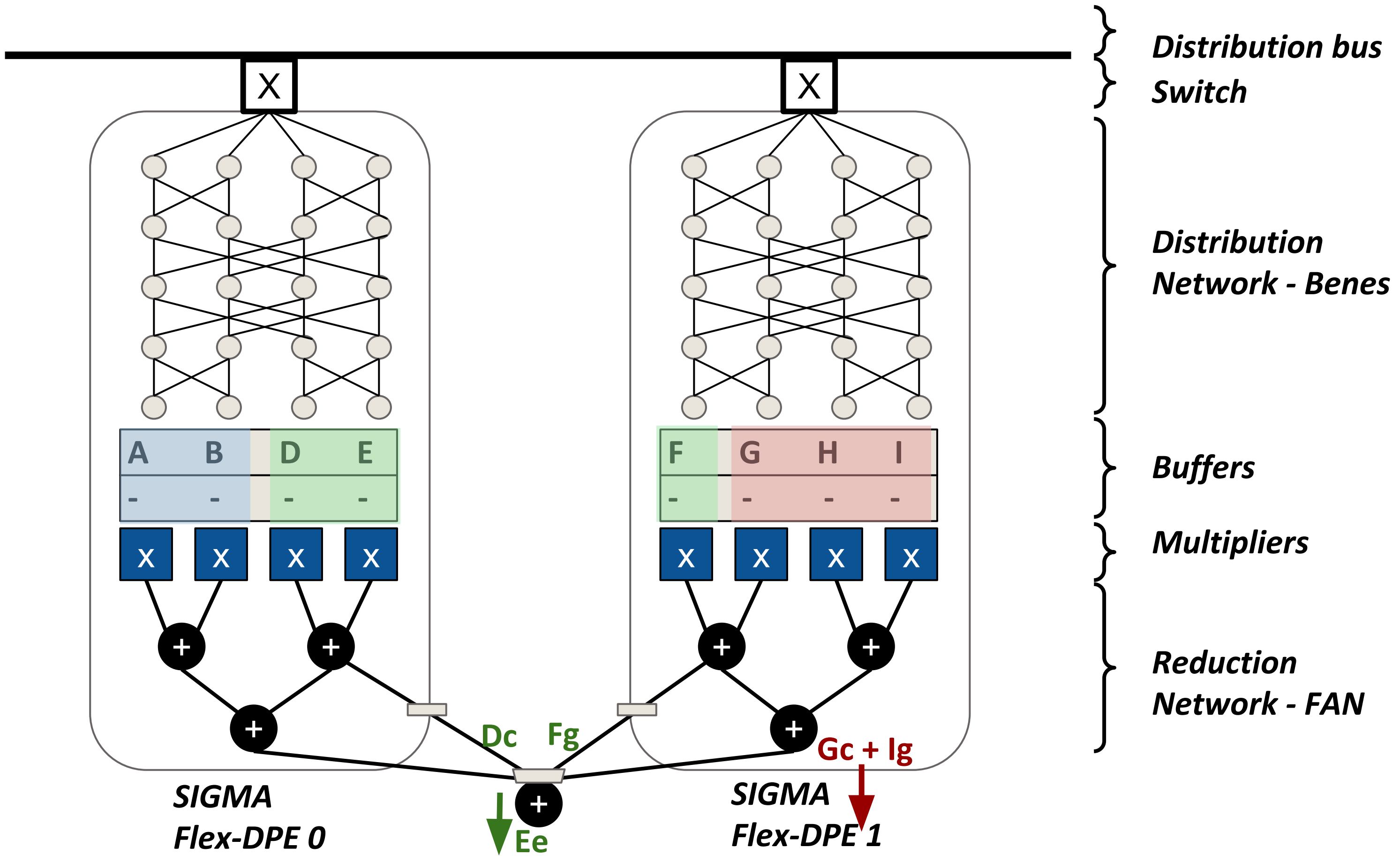
Streaming Matrix

a	b		c
	d	e	
f			g

Walkthrough Section

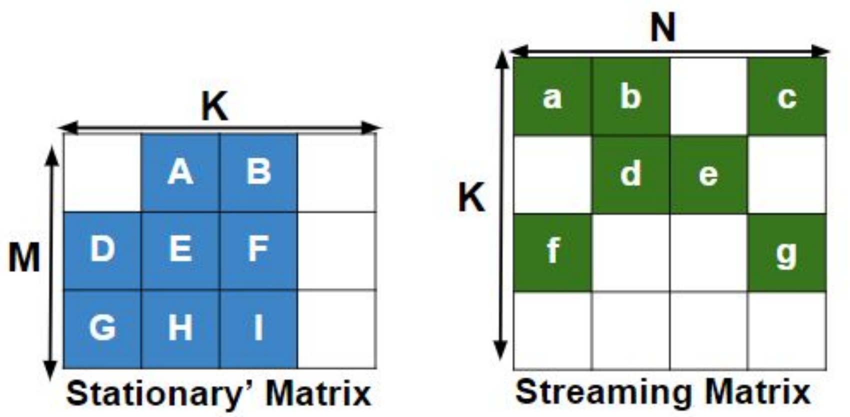
i) Get bitmap from memory
ii) Row wise OR on streaming matrix
iii) Element wise AND on stationary matrix
iv) Generate stationary' bitmap
vi) Unicast stationary values
vii) Get source - destination pairs
viii) Multicast streaming values and reduce

Walkthrough



Cycle 9: FAN Reduction

Example GEMM



	A	B	
D	E	F	
G	H	I	

Stationary’ Matrix

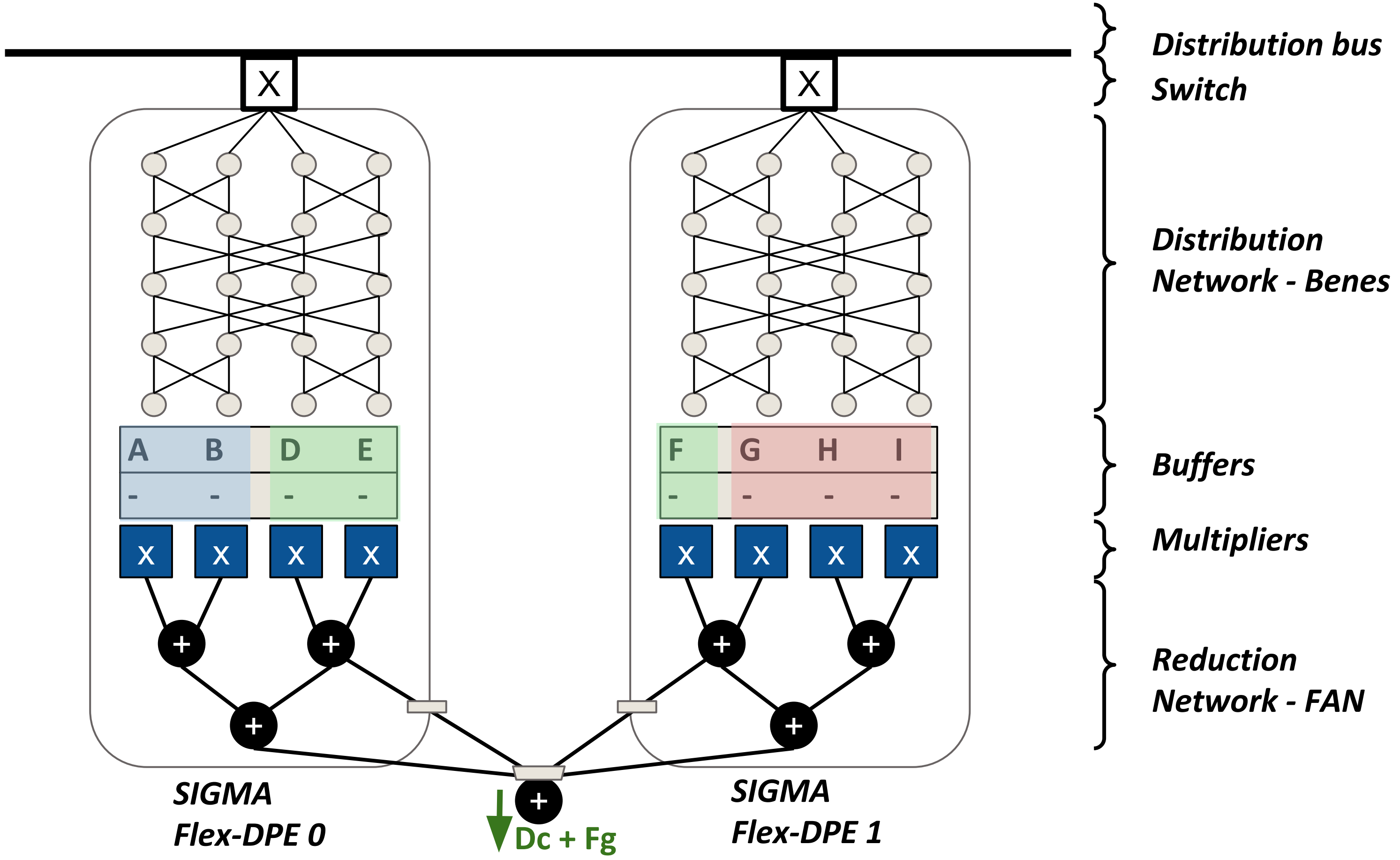
a	b		c
	d	e	
f			g

Streaming Matrix

Walkthrough Section

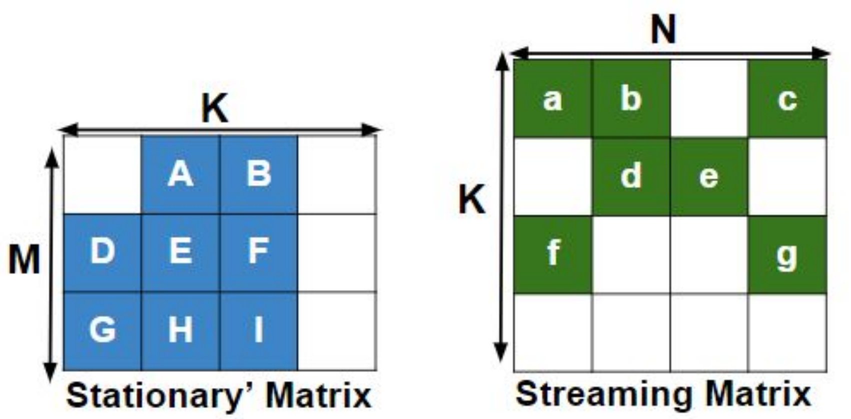
i) Get bitmap from memory
ii) Row wise OR on streaming matrix
iii) Element wise AND on stationary matrix
iv) Generate stationary’ bitmap
vi) Unicast stationary values
vii) Get source - destination pairs
viii) Multicast streaming values and reduce

Walkthrough



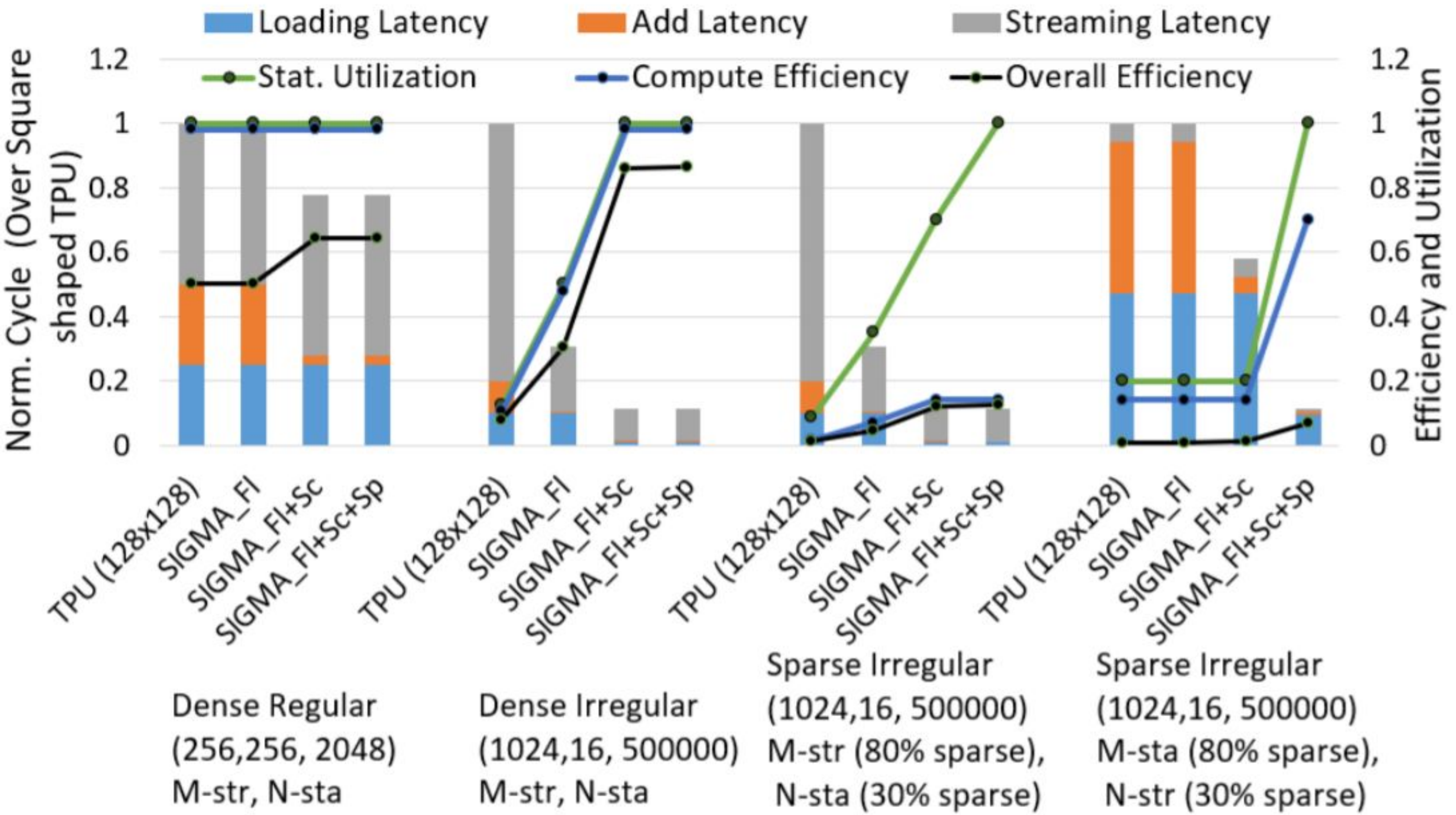
Cycle 10: FAN Reduction

Example GEMM



Walkthrough Section

- i) Get bitmap from memory
- ii) Row wise OR on streaming matrix
- iii) Element wise AND on stationary matrix
- iv) Generate stationary' bitmap
- vi) Unicast stationary values
- vii) Get source - destination pairs
- viii) Multicast streaming values and reduce


Loading Latency
Add Latency
Streaming Latency
Stat. Utilization
Compute Efficiency
Overall Efficiency
Norm. Cycle (Over Square shaped TPU)
Efficiency and Utilization
1.2
1
0.8
0.6
0.4
0.2
0
TPU (128x128)
SIGMA_Fl
SIGMA_Fl+Sc
SIGMA_Fl+Sc+Sp
Dense Regular (256,256, 2048) M-str, N-sta
Dense Irregular (1024,16, 500000) M-str, N-sta
Sparse Irregular (1024,16, 500000) M-str (80% sparse), N-sta (30% sparse)
Sparse Irregular (1024,16, 500000) M-sta (80% sparse), N-str (30% sparse)