A Communication-centric Approach for Designing *Flexible* Accelerators

STONNE Tutorial

Tushar Krishna

Associate Professor
School of ECE
Georgia Institute of Technology

Email: tushar@ece.gatech.edu

Tutorial at ASPLOS

March 1, 2022

Organizers

Tushar Krishna
Associate Professor, School of ECE
Georgia Institute of Technology
tushar@ece.gatech.edu

José L. Abellán
Associate Professor
Catholic University of Murcia
jlabellan@ucam.edu

Manuel E. Acacio
Full Professor
University of Murcia
meacacio@um.es

Raveesh Garg
Ph.D Student
Georgia Institute of Technology
raveesh.g@gatech.edu

Francisco Munoz-Martinez
Ph.D Student
University of Murcia
francisco.munoz2@um.es

Agenda

Attention: Tutorial is being recorded

Time (CET)	Time (ET)	Topic	Presenter
14:00 – 14:40	8:00 – 8:40	**Flexible Accelerators**	Tushar Krishna
14:40 – 15:10	8:40 – 9:10	**Cycle accurate simulation and Overview of STONNE**	José Luis Abellán
15:10 – 16:10	9:10 – 10:10	**(Hands-on) STONNE Deep-Dive**	Francisco Muñoz-Martínez
16:10 – 16:40	10:10 – 10:40	**Coffee Break**	
16:40 – 17:10	10:40 – 11:10	**(Hands-on) STONNE Deep-Dive**	Francisco Muñoz-Martínez
17:10 – 17:40	11:10 – 11:40	**Dataflow exploration for Graph Neural Networks**	Raveesh Garg
17:50 – 18:00	11:50 – 12:00	**Roadmap for Future Development**	Manuel Acacio

Tutorial Website https://stonne-simulator.github.io/ASPLOSTUT.html
includes agenda and STONNE/OMEGA installation instructions

Agenda

Attention: Tutorial is being recorded

Time (CET)	Time (ET)	Topic	Presenter
14:00 – 14:40	8:00 – 8:40	**Flexible Accelerators**	Tushar Krishna
14:40 – 15:10	8:40 – 9:10	**Cycle accurate simulation and Overview of STONNE**	José Luis Abellán
15:10 – 16:10	9:10 – 10:10	**(Hands-on) STONNE Deep-Dive**	Francisco Muñoz-Martínez
16:10 – 16:40	10:10 – 10:40	**Coffee Break**	
16:40 – 17:10	10:40 – 11:10	**(Hands-on) STONNE Deep-Dive**	Francisco Muñoz-Martínez
17:10 – 17:40	11:10 – 11:40	**Dataflow exploration for Graph Neural Networks**	Raveesh Garg
17:50 – 18:00	11:50 – 12:00	**Roadmap for Future Development**	Manuel Acacio

Tutorial Website https://stonne-simulator.github.io/ASPLOSTUT.html
includes agenda and STONNE/OMEGA installation instructions

Deep Learning Applications

Object Detection

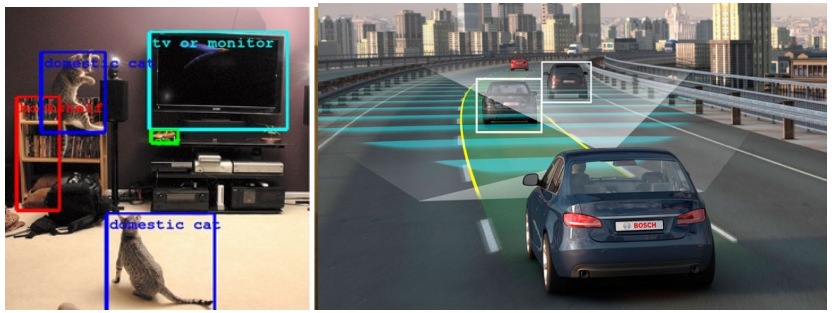
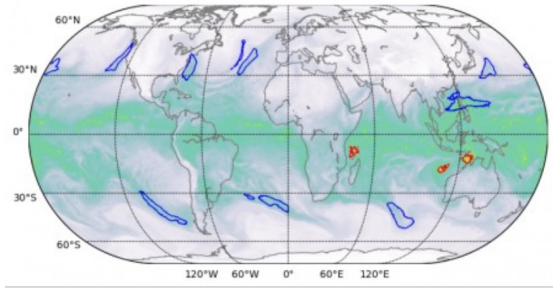
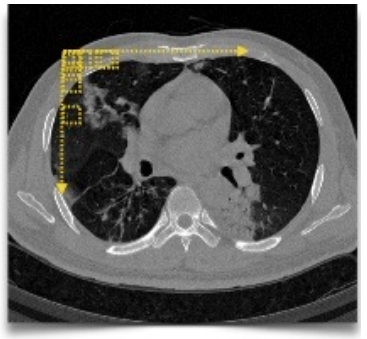


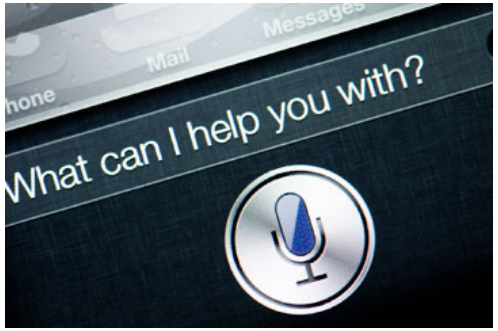
Image Segmentation



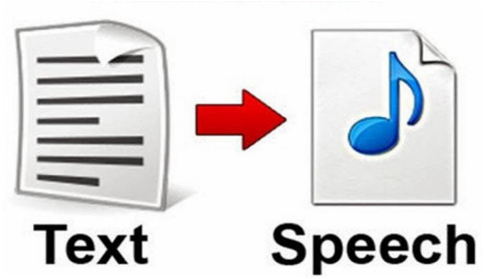
Medical Imaging



Speech Recognition



Text to Speech

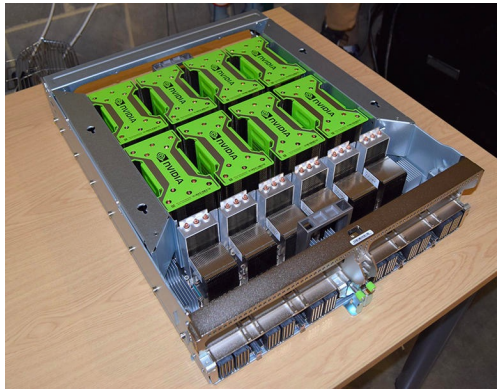


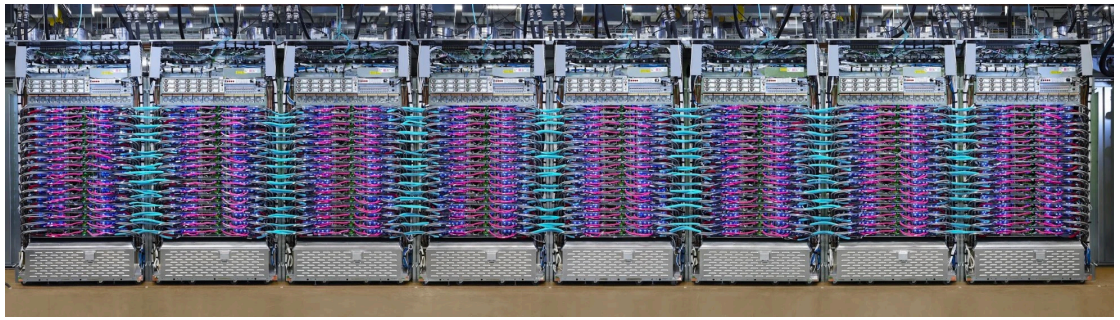
Recommendations

Games

Computation Platforms for Deep Learning

Training → Inference





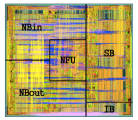
CPU / GPU / TPU clusters

ARM Trillum

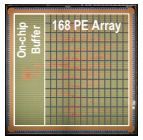
NVDLA

Apple Neural Engine

CambriconX



ShiDianNao



Eyeriss

Inference Accelerators

Outline

- DNN Accelerators
- Benefits of Mapping Flexibility
- Communication-centric Accelerators
- Case Study
- Conclusion

Why do we need DNN accelerators?

- **Millions of Parameters (i.e., weights)**
  - Billions of computations

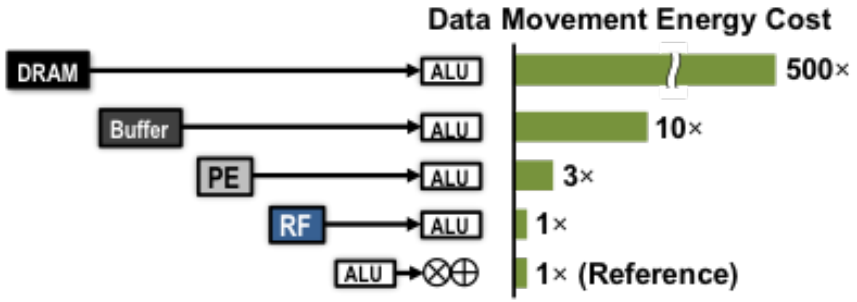
Need lots of parallel compute

This makes CPUs inefficient

DNN Topology	Number of Weights
AlexNet (2012)	3.98M
VGGnet-16 (2014)	28.25M
GoogleNet (2015)	6.77M
Resnet-50 (2016)	23M
DLRM (2019)	540M
Megatron (2019)	8.3B

  - Heavy data movement

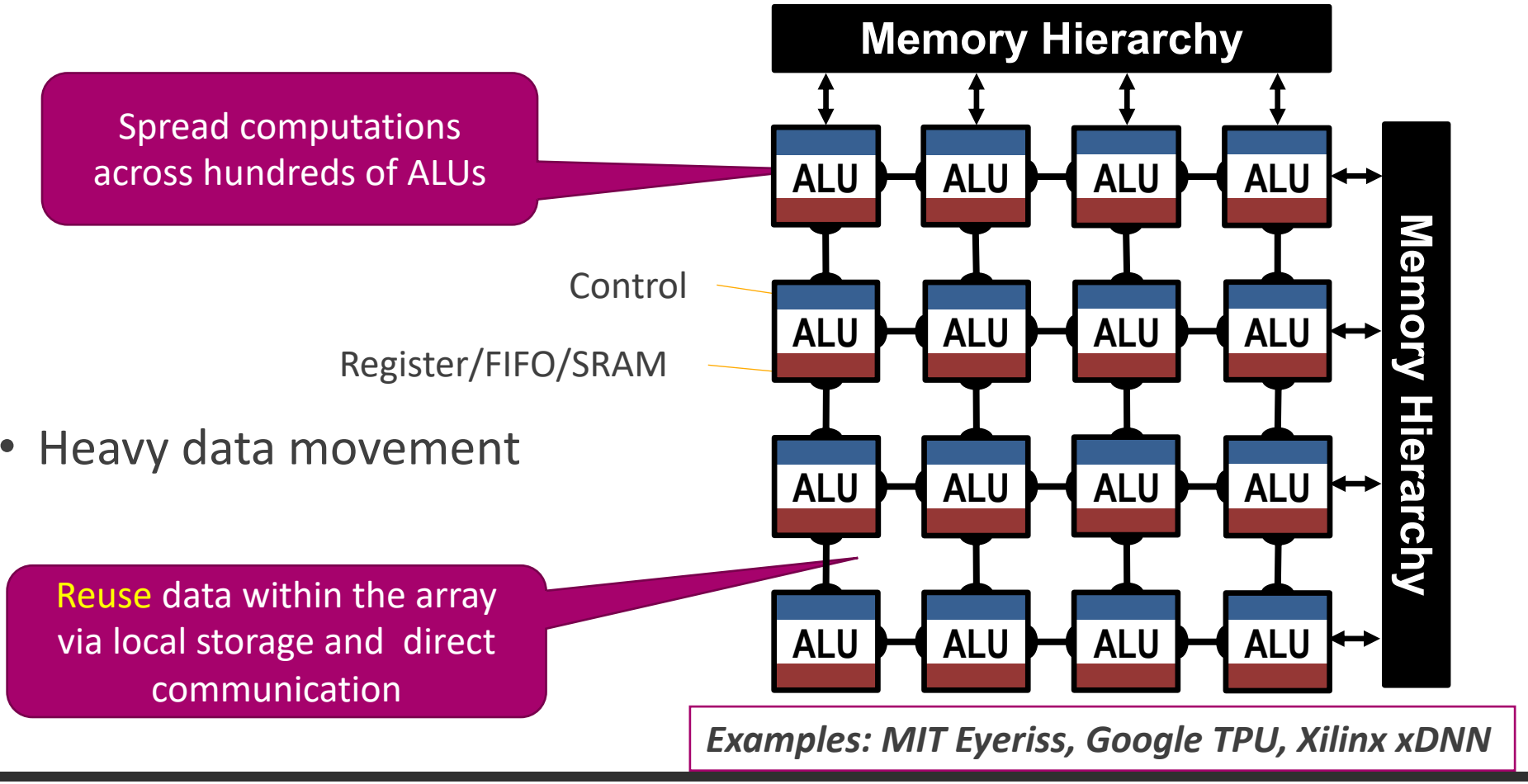
Need to reduce energy



This makes GPUs inefficient

Spatial (or Dataflow) Accelerators

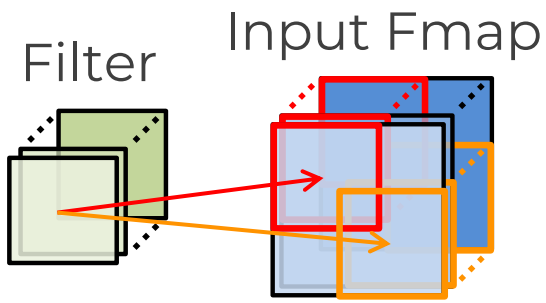
- **Millions of Parameters (i.e., weights)**
  - Billions of computations
  - Heavy data movement



Types of Algorithmic Data Reuse in DNNs

Convolutional Reuse

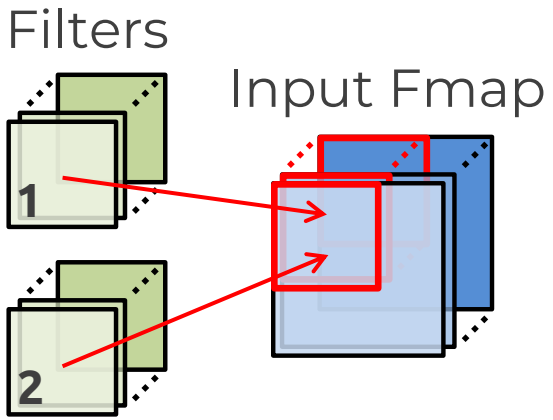
CONV layers only
(sliding window)



Reuse: Activations
Filter weights

Fmap Reuse

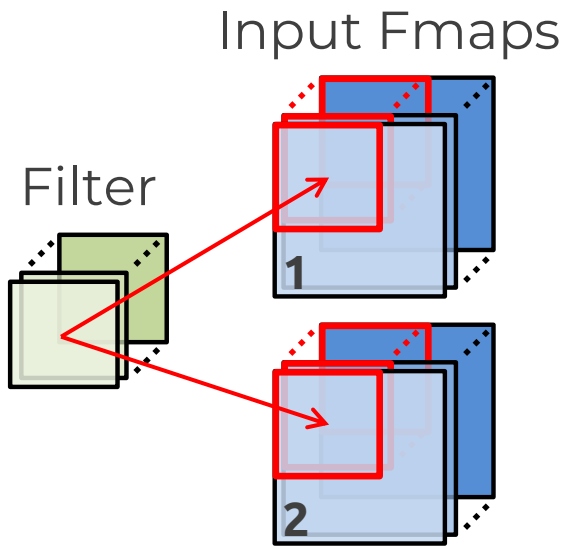
CONV and FC layers



Reuse: Activations

Filter Reuse

CONV and FC layers
(batch size > 1)

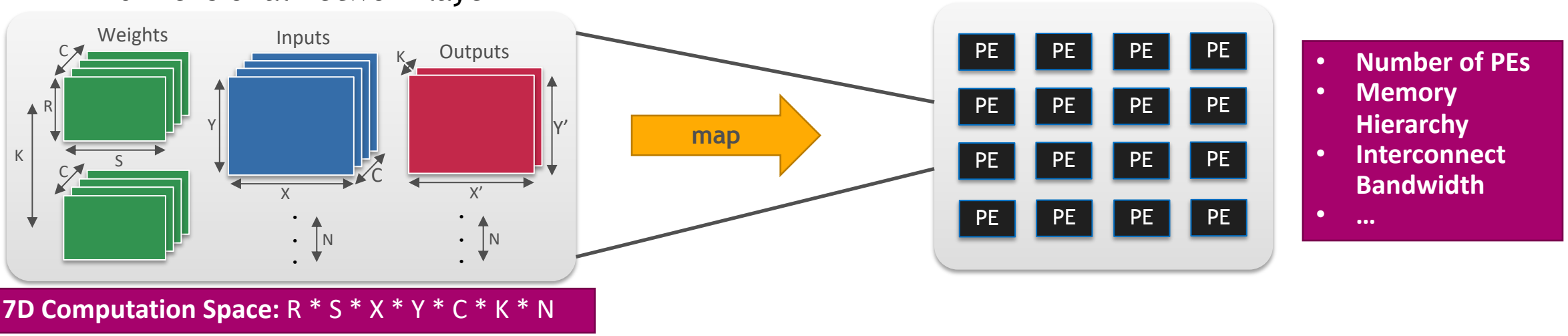


Reuse: Filter weights

Slide Acknowledgment: Yu-Hsin Chen, Vivenne Sze, Joel Emer (MIT)

Mapping and Dataflow

7-dimensional network layer



7D Computation Space: R * S * X * Y * C * K * N

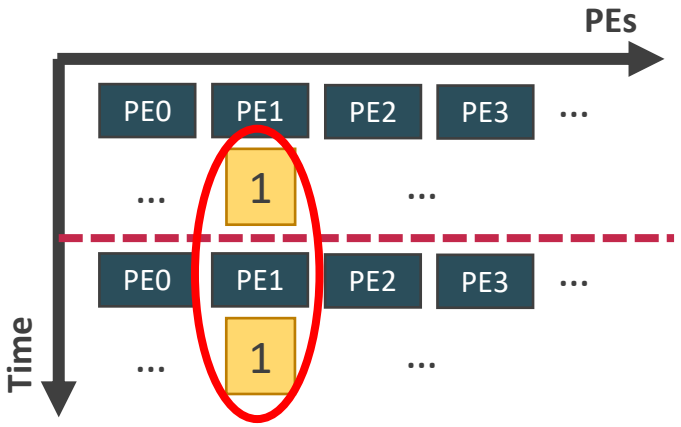
- **Precise Definition of Mapping:** Fine-grained schedule of computations within DNN accelerators
  - **Computation Order** *(slowest tensor dimension often called "stationary")*
  - **Parallelization Strategy** *(which loops to unroll spatially)*
  - **Tiling Strategy** *(number of levels of memory hierarchy)*
  - **Tile Sizes**

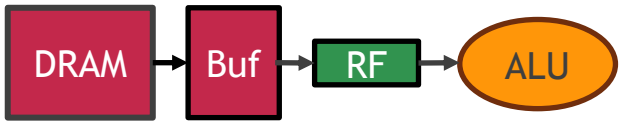
Dataflow

- **Note:** *Implication of mapping/dataflow: translate algorithmic data reuse to HW data reuse*

Hardware Structures to Exploit Reuse

Temporal Reuse

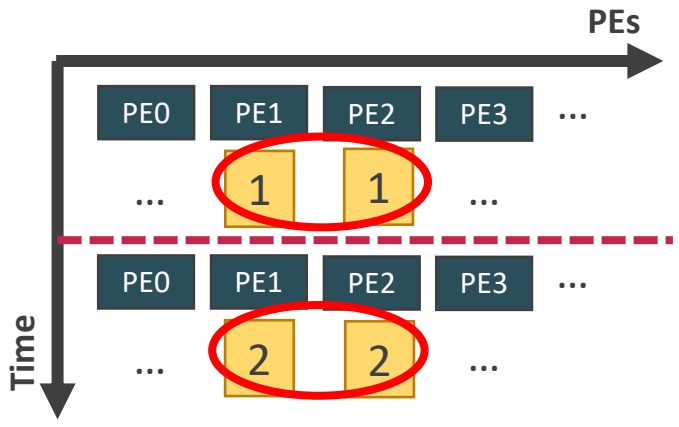


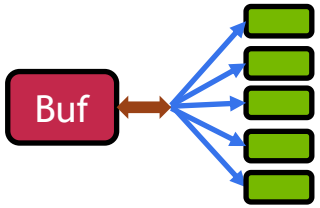


Memory Hierarchy / Staging Buffers

E.g., Custom memory hierarchies in accelerators.

Spatial Reuse

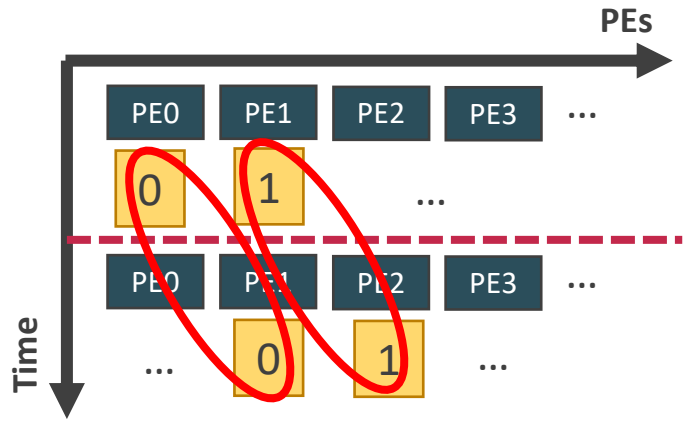


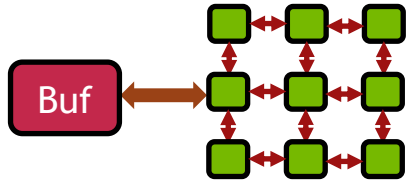


Multicasting-support NoCs

E.g., Hierarchical Bus in Eyeriss (ISCA 2016), Tree in MAERI (ASPLOS 2018)

Spatio-Temporal Reuse





Neighbor-to-Neighbor Connections

E.g., TPU (ISCA 2017), local network in Eyeriss (ISCA 2016)

The Accelerator's "Dataflow" determines which structure are needed and their size

Outline

- DNN Accelerators
- Benefits of Mapping Flexibility
- Communication-centric Accelerators
- Case Study
- Conclusion

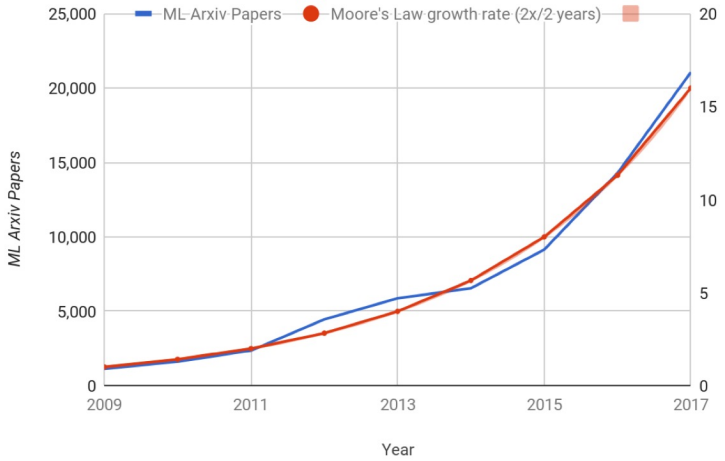
Why do we need *flexible* DNN accelerators?

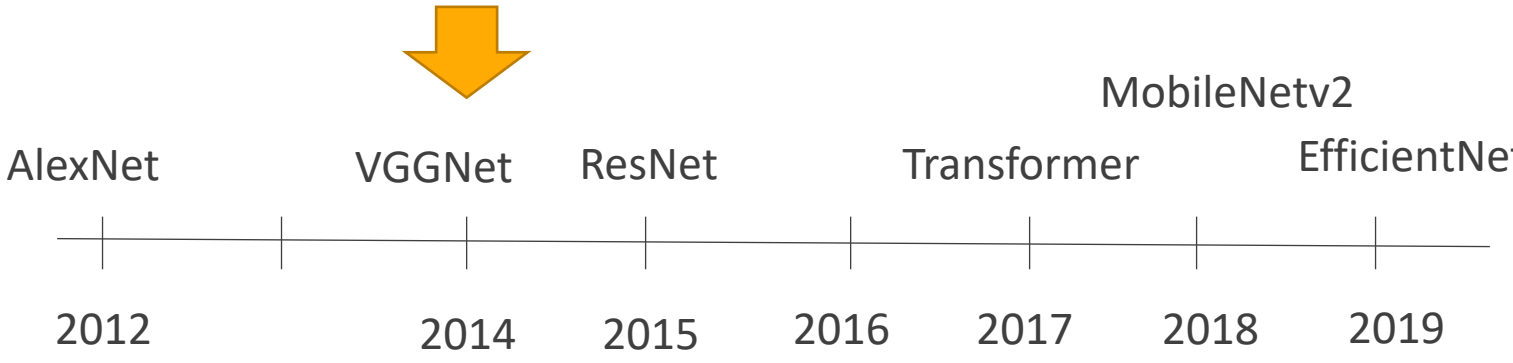
- **Trend 1: Diversity in DNN Models**
  - Layer Sizes
  - Layer Shapes
  - Layer Types

Image Processing | Speech Recognition | Translation | ...

Evolution of DNN Applications

<Number of new ML papers in Arxiv>

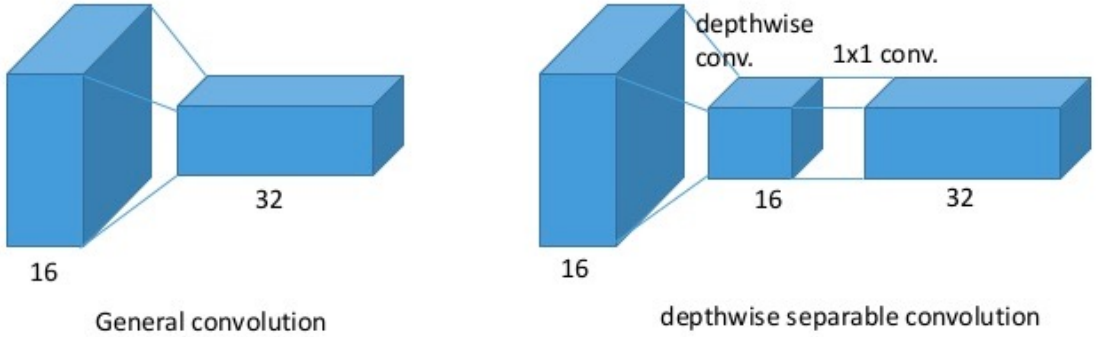




Evolution of DNN Models

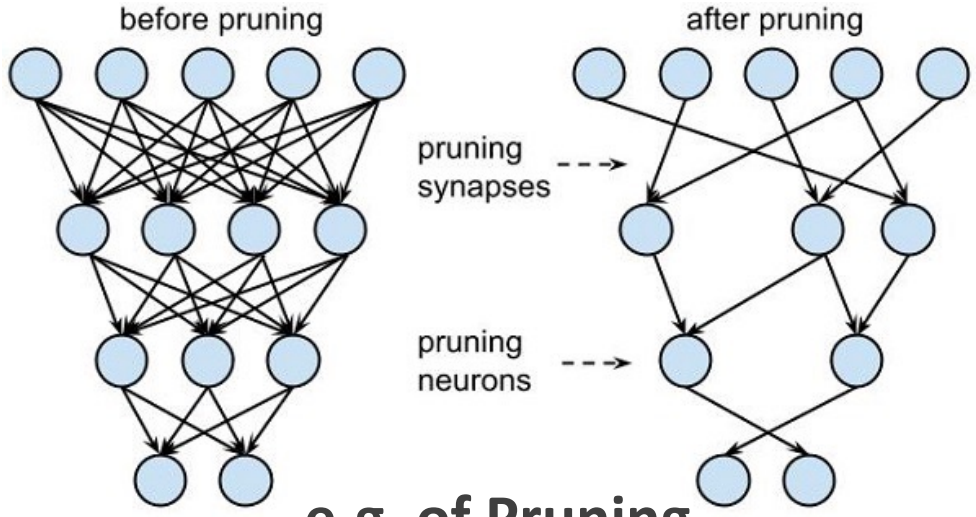
Why do we need *flexible* DNN accelerators?

- **Trend 1: Diversity in DNN Models**
  - Layer Sizes
  - Layer Shapes
  - Layer Types



e.g. of Depth-wise Separable CONV

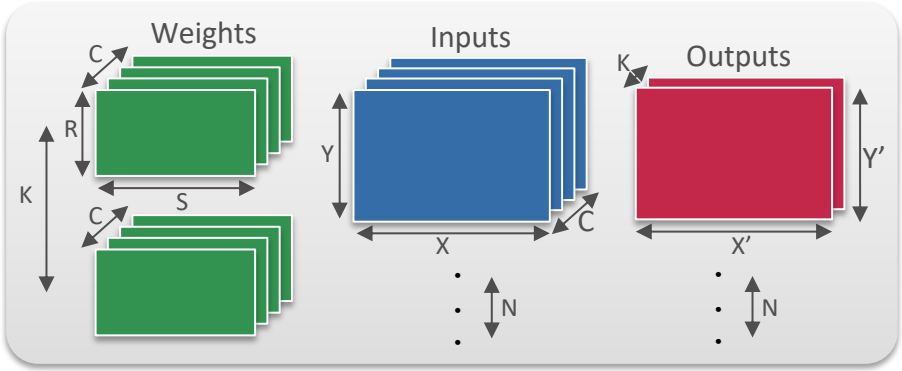
- **Trend 2: Diversity in Implementations**
  - Depth-wise/Point-wise Convolutions
  - Pruning → Sparsity



e.g. of Pruning

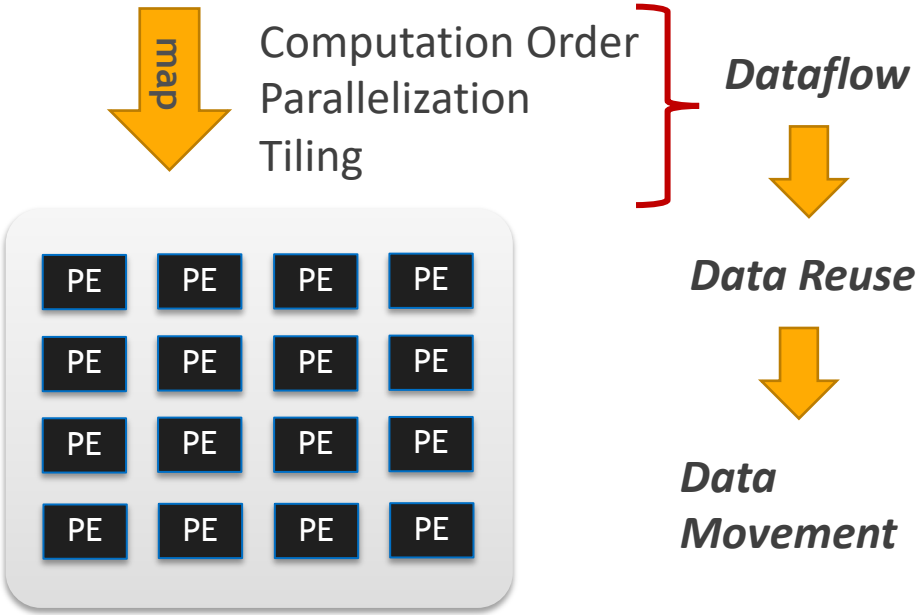
Why do we need *flexible* DNN accelerators?

- **Trend 1: Diversity in DNN Models**
  - Layer Sizes
  - Layer Shapes
  - Layer Types



- **Trend 2: Diversity in Implementations**
  - Depth-wise/Point-wise Convolutions
  - Pruning → Sparsity

- **Trend 3: Diversity in Mapping/Dataflow**
  - Loop Transformations ("Dataflow")
    - Order, Parallelization, Tiling
    - "Weight Stationary", "Row Stationary"
  - Partitioning Strategies – Per Layer, Cross Layer, ..



Why do we need *flexible* DNN accelerators?

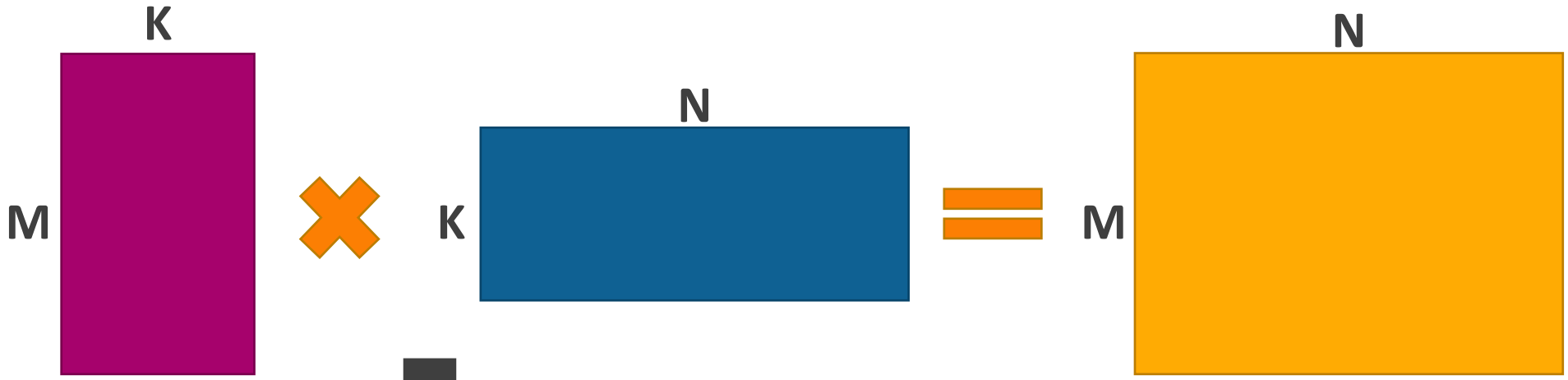
- **Trend 1: Diversity in DNN Models**
  - Layer Sizes
  - Layer Shapes
  - Layer Types
- **Trend 2: Diversity in Implementations**
  - Depth-wise/Point-wise Convolutions
  - Pruning → Sparsity
- **Trend 3: Diversity in Mapping/Dataflow**
  - Loop Transformations ("Dataflow")
    - Order, Parallelization, Tiling
    - "Weight Stationary", "Row Stationary"
  - Partitioning Strategies – Per Layer, Cross Layer, ..

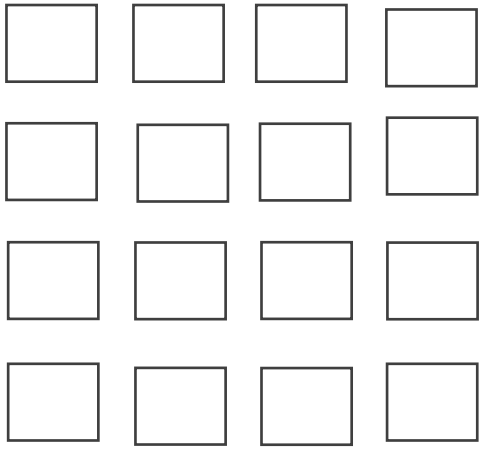
Myriad "irregular" shapes, sizes, accesses

Challenge:
Getting high-utilization from accelerator for all cases.

Why?
Aren't DNNs essentially Matrix-Matrix multiplications?

Example of GEMM Operation





e.g., TPU

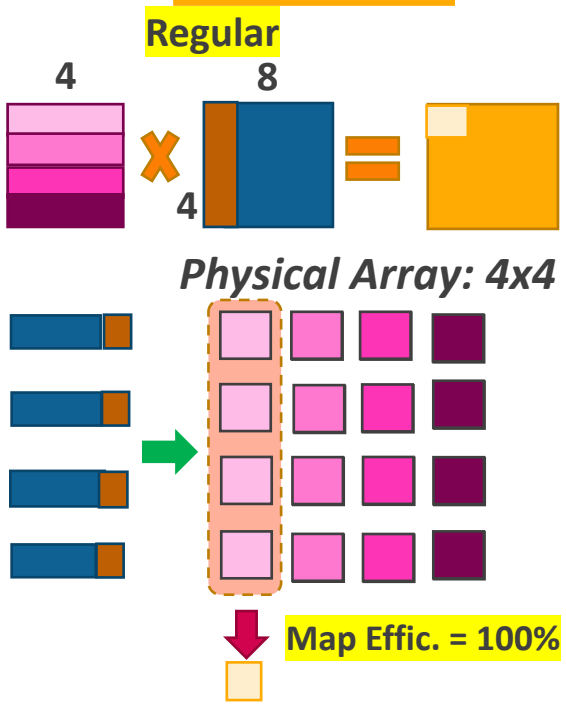
Metric 1: Mapping Efficiency

Percentage of PEs with useful computations mapped over them

Metric 2: Utilization

Mapping Efficiency x Activity

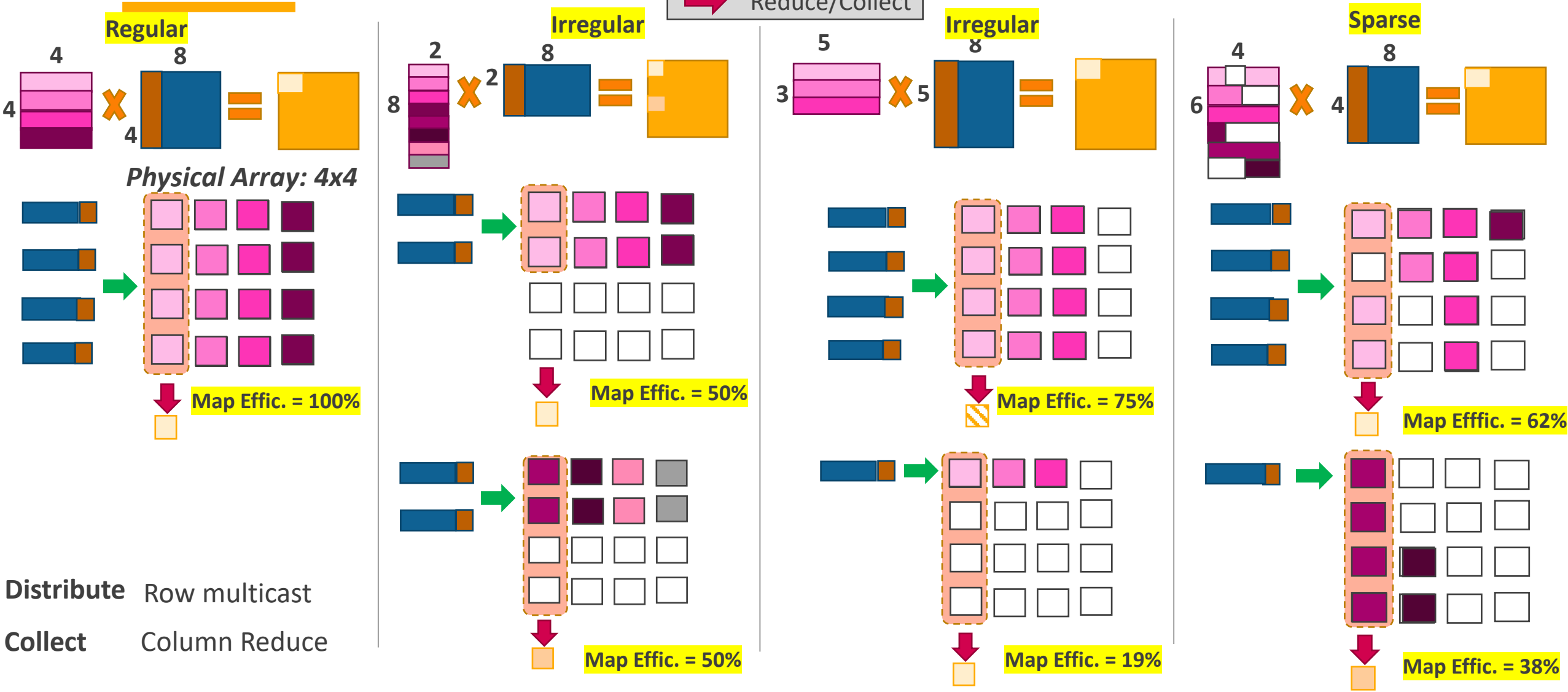
Mapping Examples



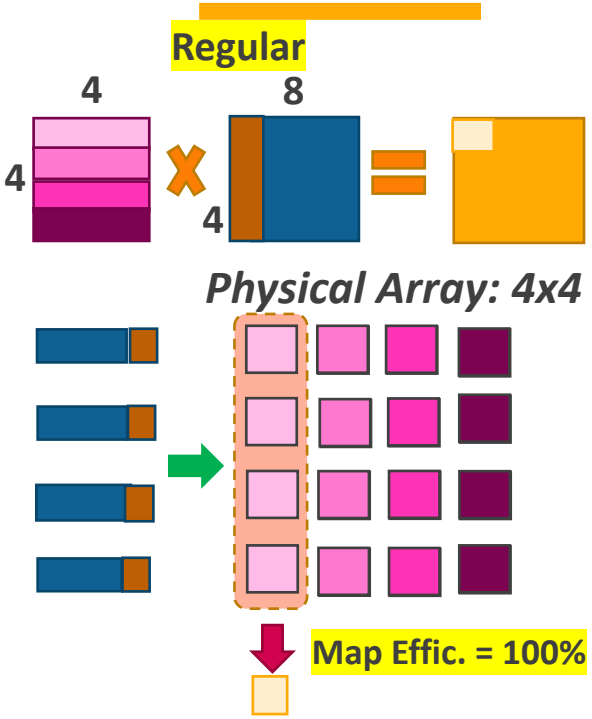
Distribute Row multicast

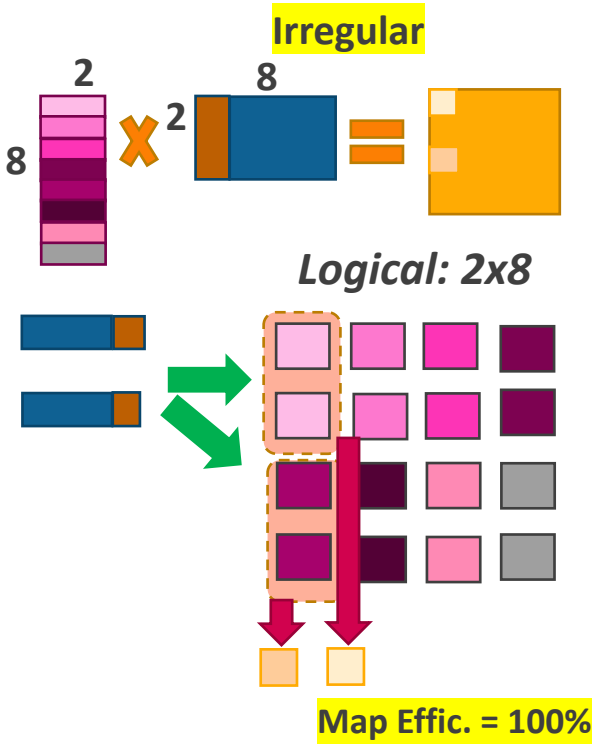
Collect Column Reduce

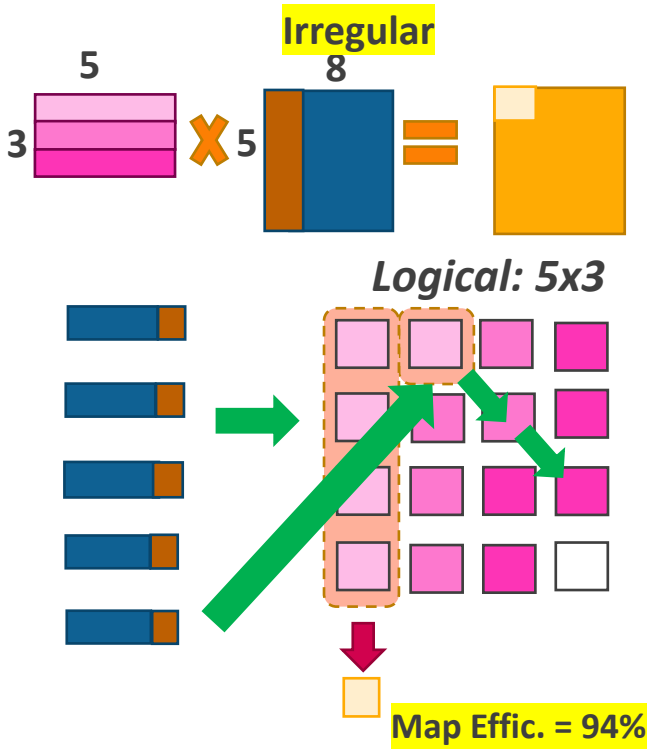
Mapping Examples

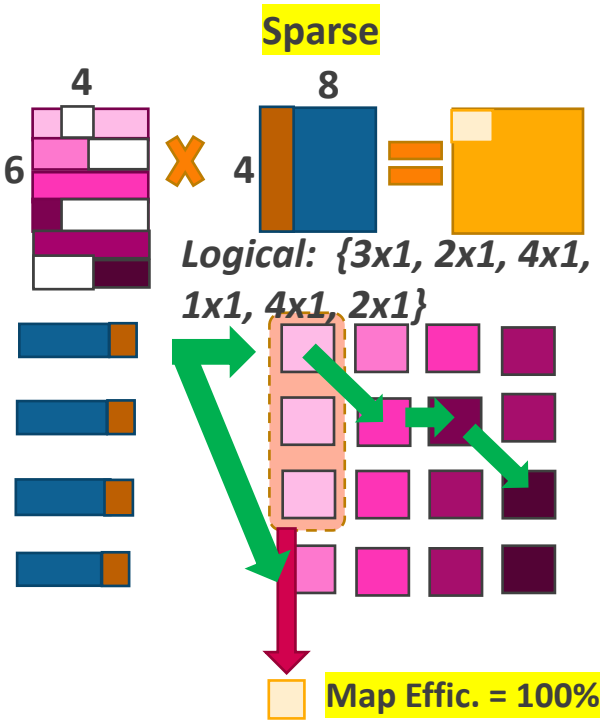


Mapping Efficiency needs Mapping Flexibility







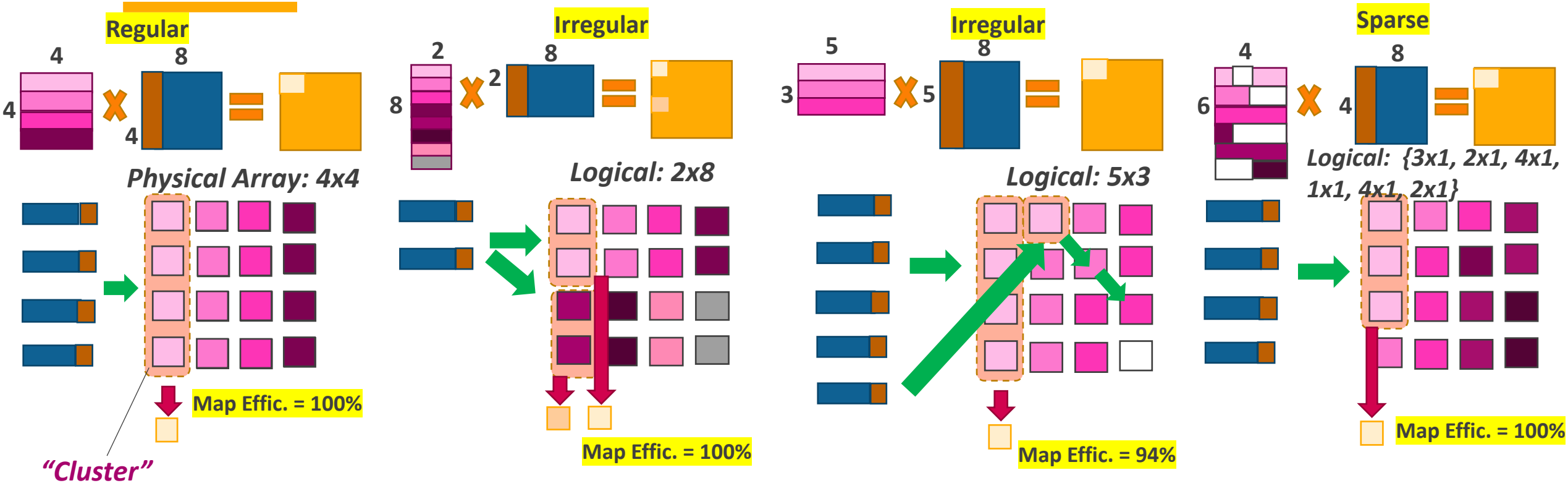


How to support Mapping Flexibility?

Distribute	Row multicast	Spatial Multicast	Multicast to non-neighbors	Only send non-zeros
Collect	Column Reduce	Multiple Parallel	Variable Length	Variable Non-Uniform Length

Flexible data distribution and reduction

Levels of Flexibility



Fixed Homogeneous Clusters
(i.e., fixed cluster size => fixed aspect ratio)

Partially-Flexible Homogeneous Clusters
(configurable (limited choices) number of PEs per cluster)

Fully-Flexible Homogeneous Clusters
(configurable (any choice) number of PEs per cluster)

Fully-Flexible Heterogeneous Clusters
(configurable (any choice) unequal sized clusters)

Summary of Mapping Challenge

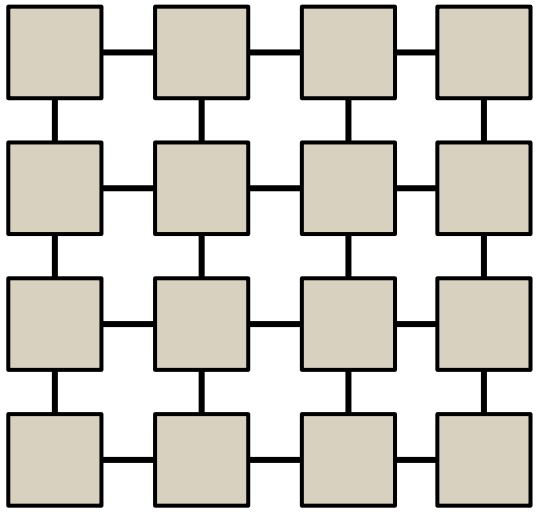
- Irregular and Sparse GEMMs introduce under-utilization

- Desired Features for 100% Utilization
  - **Distribution**
    - Variable Sized Multicast
    - Multiple in Parallel
    - [For Sparsity: Only send non-zero weights/activations]
  - **Collection**
    - Variable Sized Reduction
    - Multiple in Parallel

Outline

- DNN Accelerators
- Benefits of Mapping Flexibility
- Communication-centric Accelerators
- Case Study
- Conclusion

Communication-centric Accelerator Design



4x4 Rigid Array

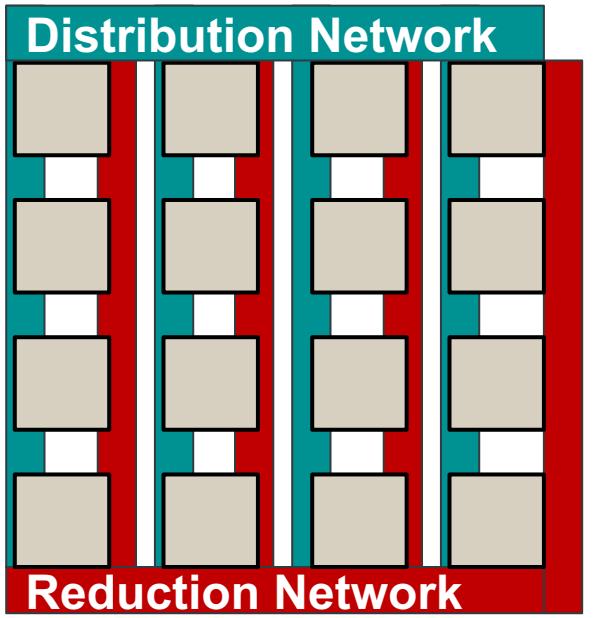
✓ ***Fixed Homogeneous Clusters***

Communication-centric Accelerator Design

Functionality:
Distribute weight and input operands to any PE
(pattern depends on Mapping)

Optimizations:
- Bandwidth *amplification* via in-network *multicast*
- Local forwarding links

Implementation:
Topology: Bus/Store-and-Fwd/Tree/..
Latency: O(1) to O(N)
No-stall Bandwidth: O(sqrt(N)-O(N)



Functionality:
Collection final (or partial) outputs from the PEs
(pattern depends on Mapping)

Optimizations:
- Bandwidth *reduction* via in-network *reduction*

Implementation:
Area: Bus/Reduce-and-Fwd/Tree/…
Latency: O(1)/O(log(N))/O(sqrt(N)
No-stall Bandwidth: O(sqrt(N)-O(N)

- ✓ ***Part-Flexible Homogenous Clusters***
- ✓ ***Full-Flexible Homogenous Clusters***
- ✓ ***Full-Flexible Hetero. Clusters***

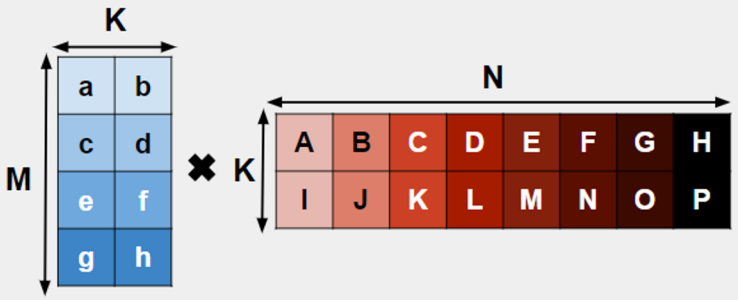
Comm-centric Classification of DNN Accelerators

Accelerator	Distribution Network	Reduction Network	Level of Flexibility
Eyeriss (ISSCC 2016)	(multiple) Bus O(1), param	Bus O(1), 1	Partially-Flexible Homogeneous Clusters
TPU (ISCA 2017)	Store-and-Fwd O(sqrtN), sqrt(N)	Reduce-and-Fwd O(sqrt(N), sqrt(N)	Fixed Homogeneous Clusters
NVDLA (2017)	Bus O(1), 1	Tree O(logN), 1	Fixed Homogeneous Clusters
MAERI (ASPLOS 2018)	Chubby-Tree O(1), param	Augmented Reduction Tree (ART) O(logN), param	Fully-Flexible Homogeneous Clusters
Eyeriss_v2 (JETCAS 2019)	Hierarchical Mesh O(sqrt(N)), param	Hierarchical Mesh O(sqrt(N)), param	Fully-Flexible Homogeneous Clusters *(with Sparsity Controller)*
SIGMA (HPCA 2020)	Benes O(1), N	Forwarding Adder Network (FAN) O(log(N)), N/2	Fully-Flexible Heterogeneous Clusters *(with Sparsity Controller)*

Latency, Injection Bandwidth

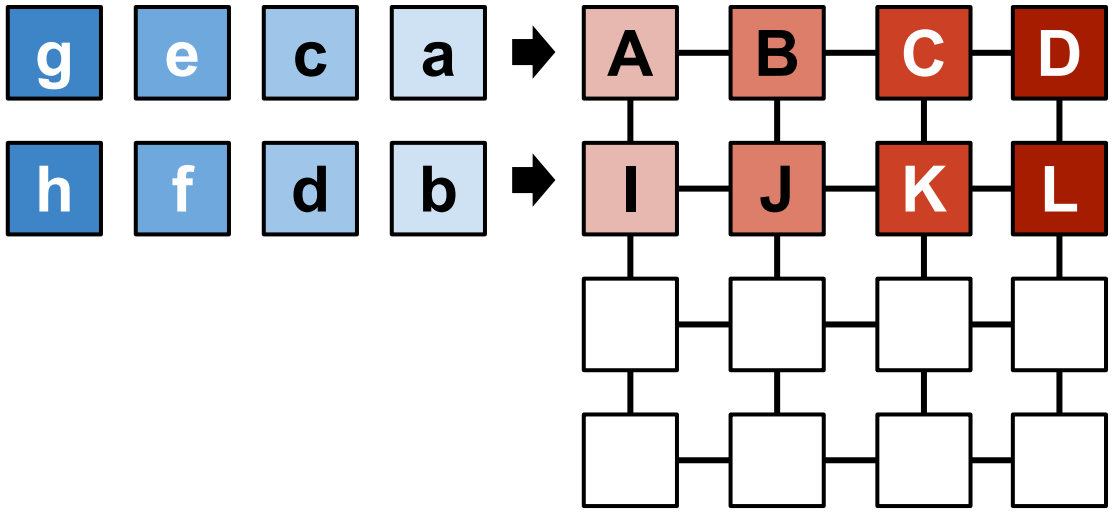
Latency, Ejection Bandwidth

Example - Irregular GEMMs on SIGMA



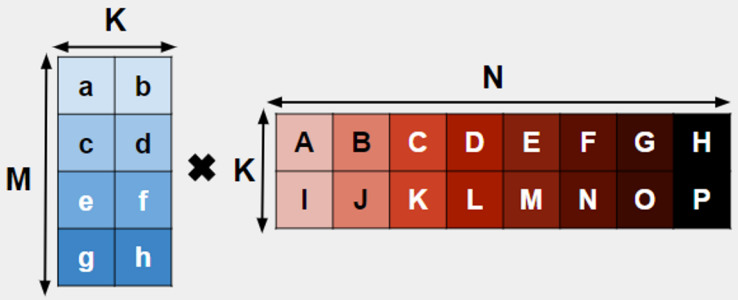
Set up the stationary values.

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



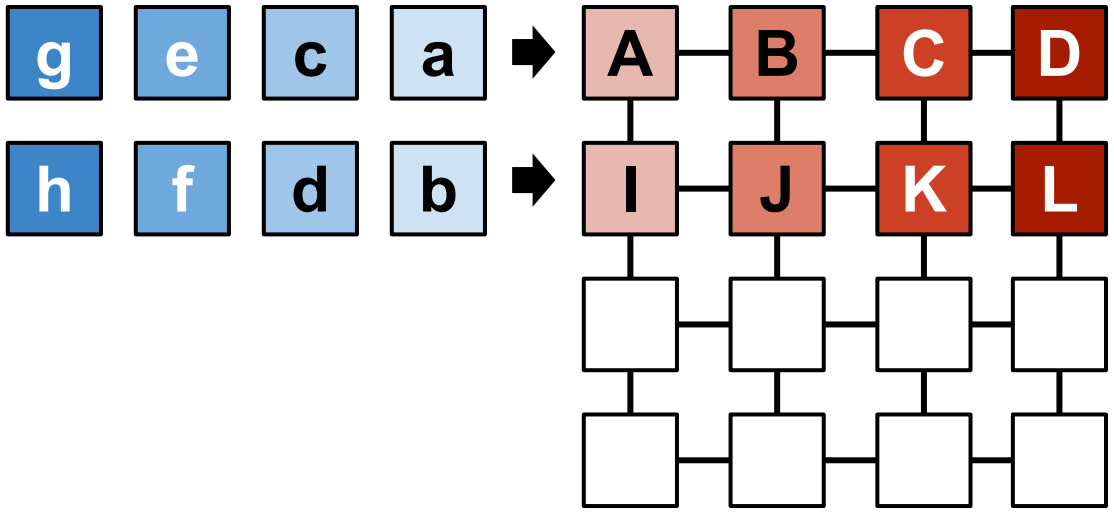
4 x 4 Systolic Array

Example - Irregular GEMMs on SIGMA

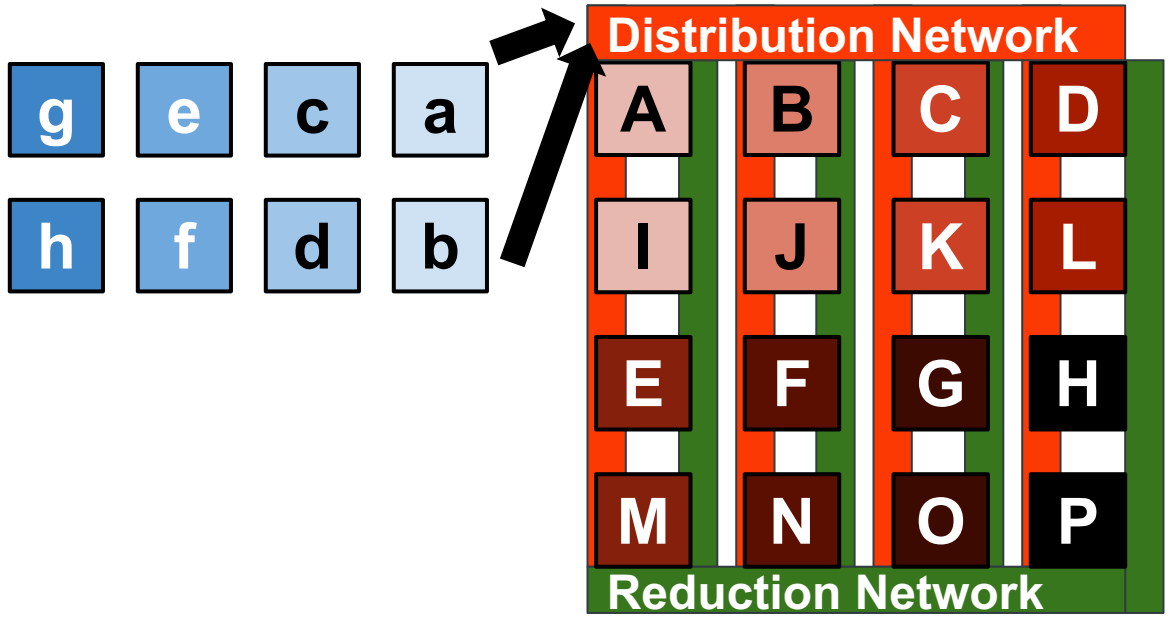


Set up the stationary values.

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

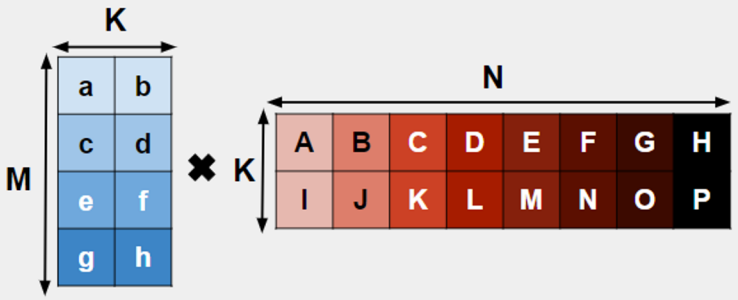


4 x 4 Systolic Array



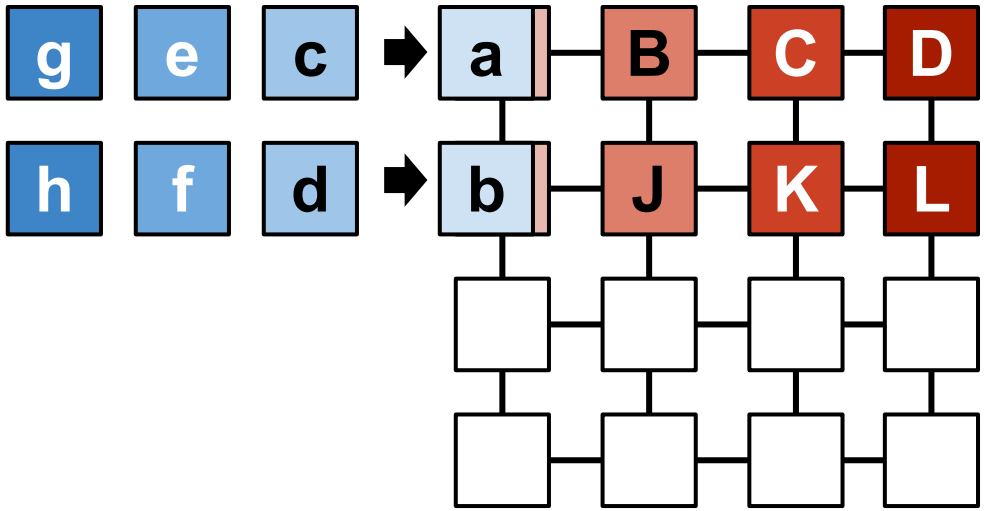
16 PE SIGMA

Example - Irregular GEMMs on SIGMA

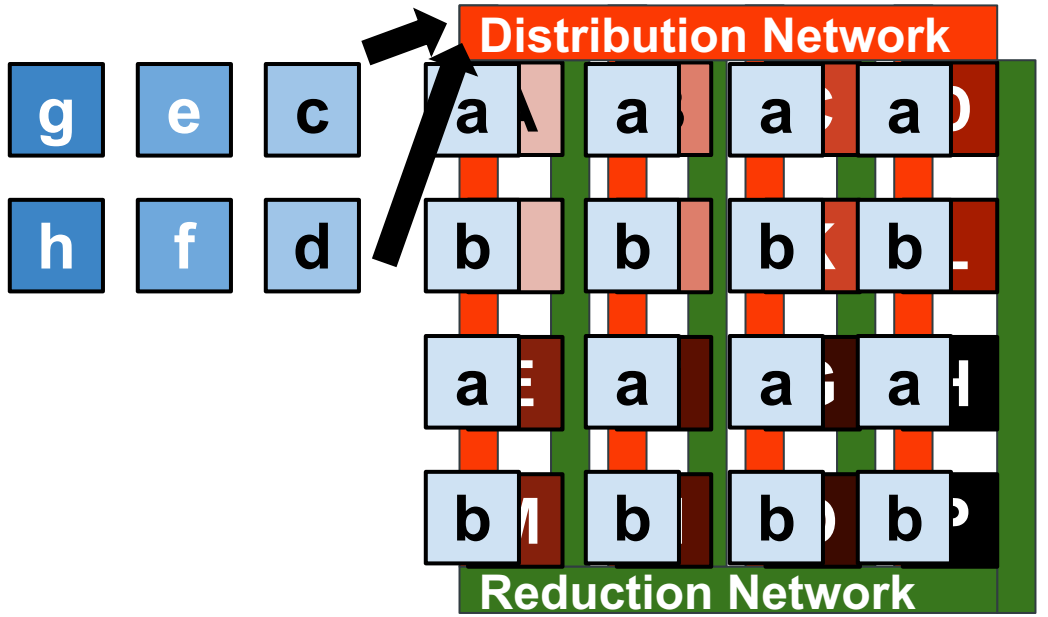


Next cycle: Multicast first row of MK to the corresponding stationary elements.

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

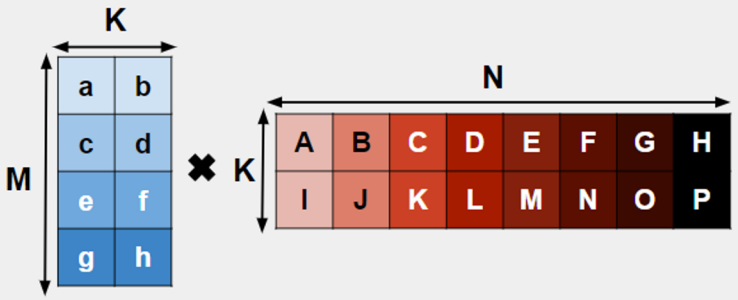


4 x 4 Systolic Array



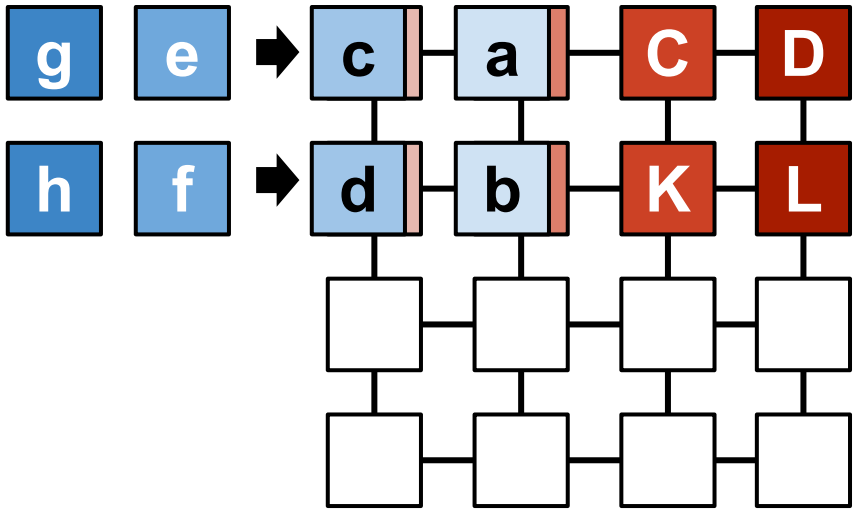
16 PE SIGMA

Example - Irregular GEMMs on SIGMA

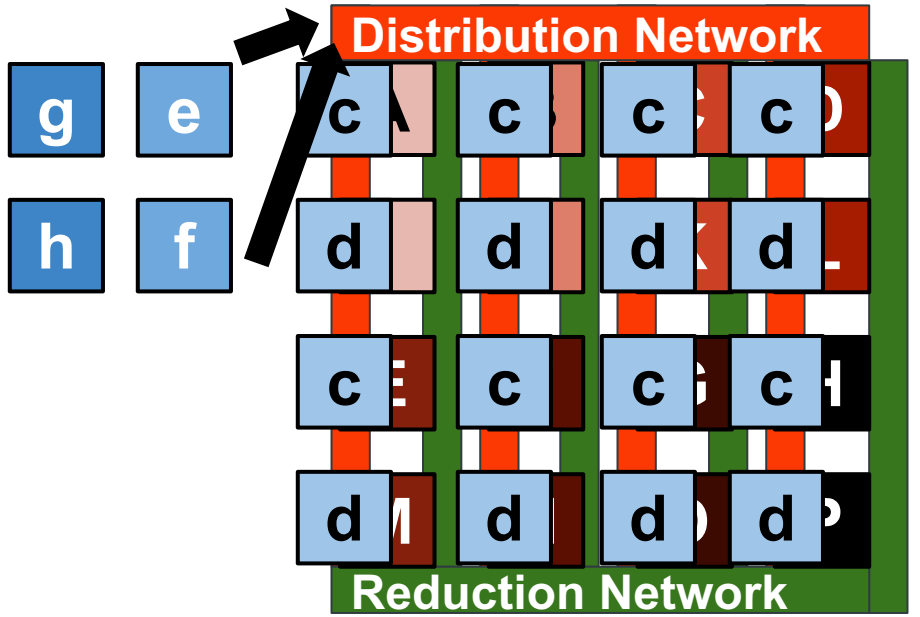


Next cycle: Multicast second row of MK to the corresponding stationary elements.

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

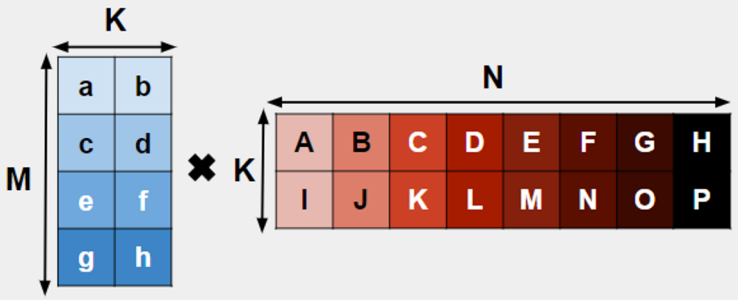


4 x 4 Systolic Array



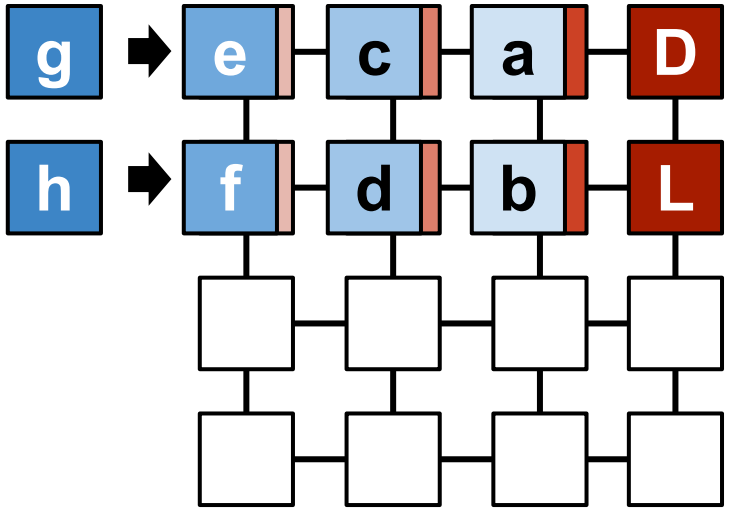
16 PE SIGMA

Example - Irregular GEMMs on SIGMA

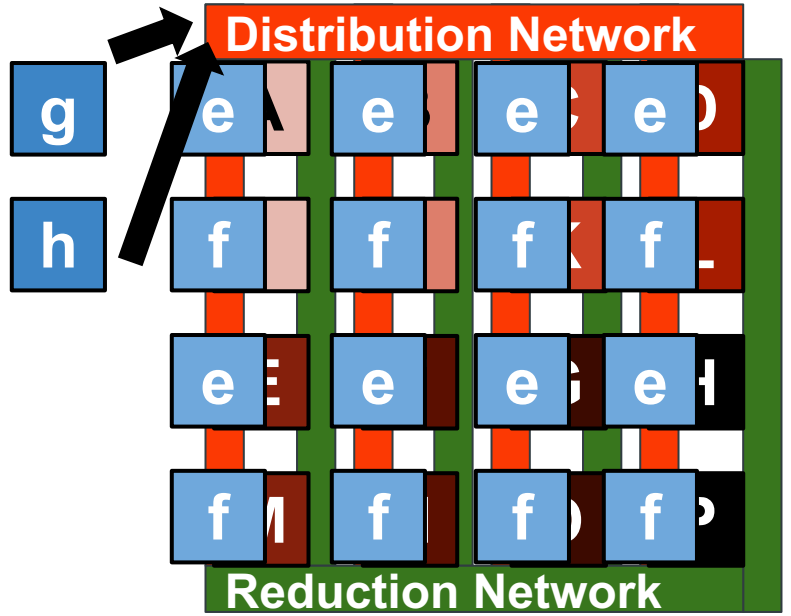


Next cycle: Multicast third row of MK to the corresponding stationary elements.

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

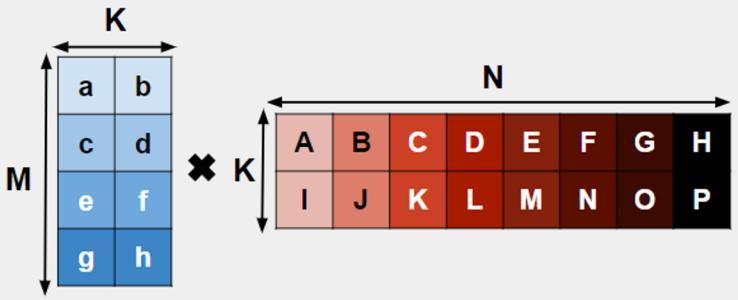


4 x 4 Systolic Array



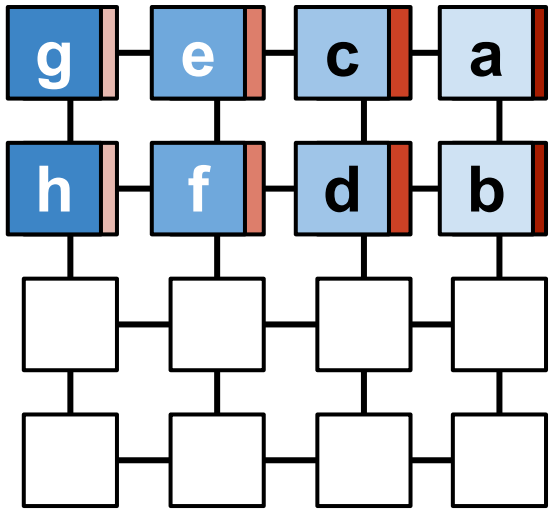
16 PE SIGMA

Example - Irregular GEMMs on SIGMA

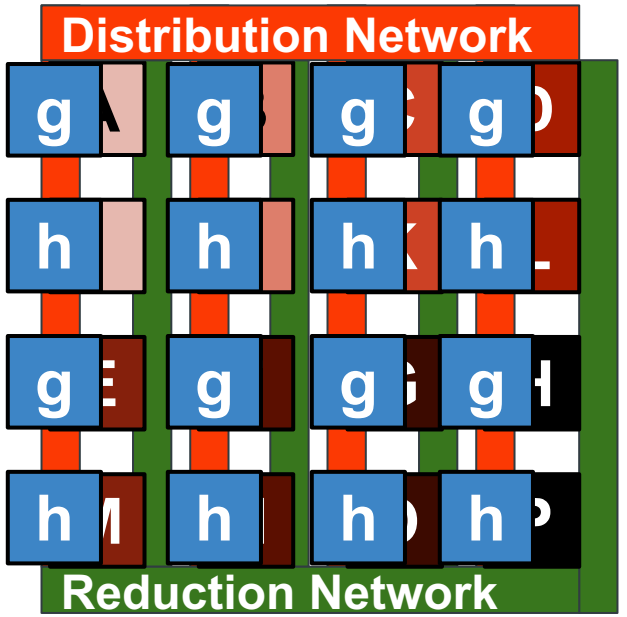


Next cycle: Multicast fourth row of MK to the corresponding stationary elements.

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

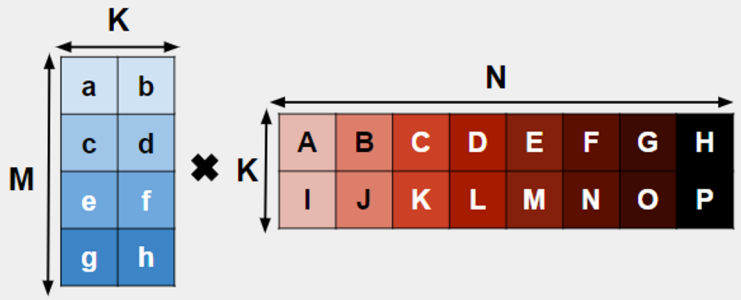


4 x 4 Systolic Array



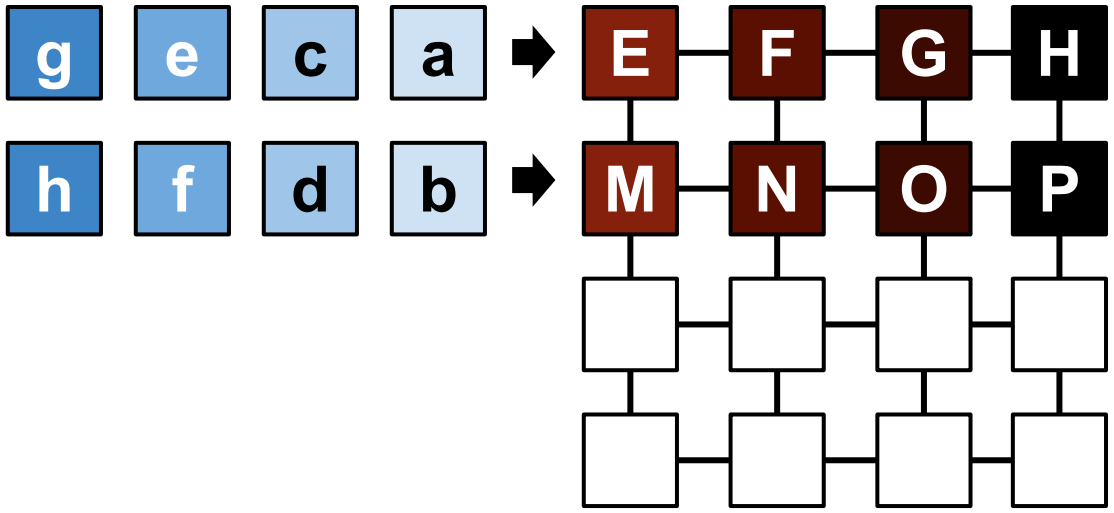
16 PE SIGMA

Example - Irregular GEMMs on SIGMA

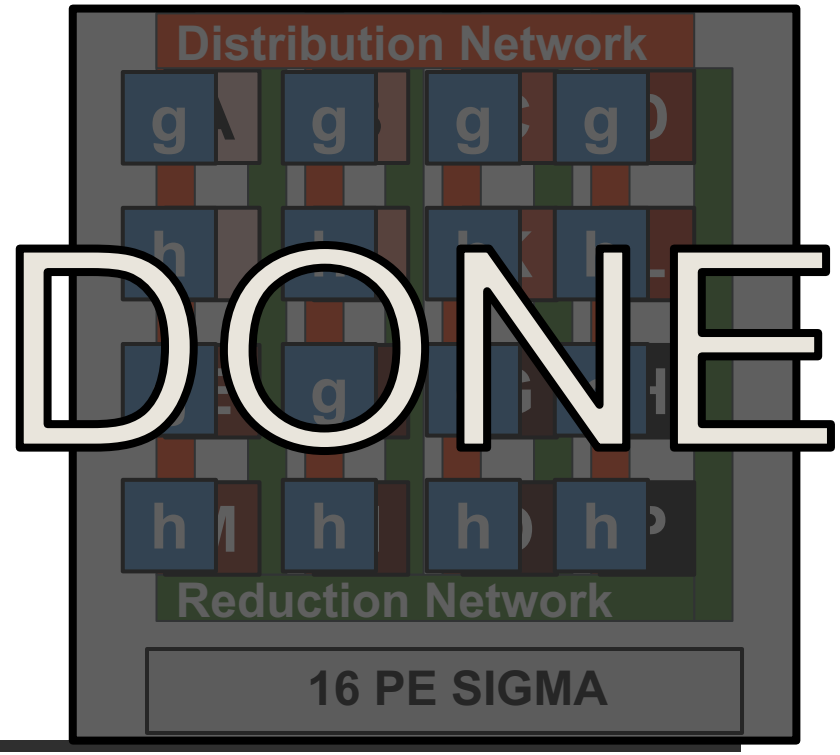


After accumulation, SIGMA is done. However, the systolic array has to map the other side of the stationary matrix and stream in the MK matrix again (referred to as folding).

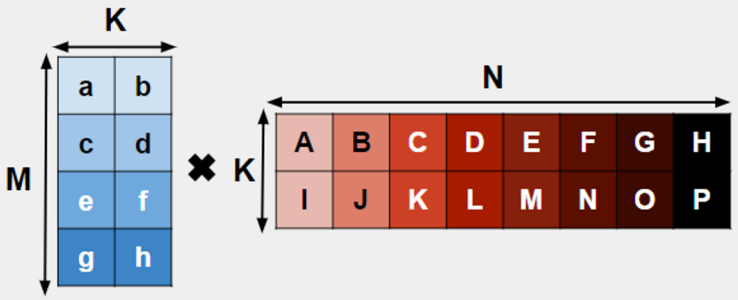
***** Assuming MK matrix is streaming and KN matrix is stationary. (aka weight stationary)***



4 x 4 Systolic Array

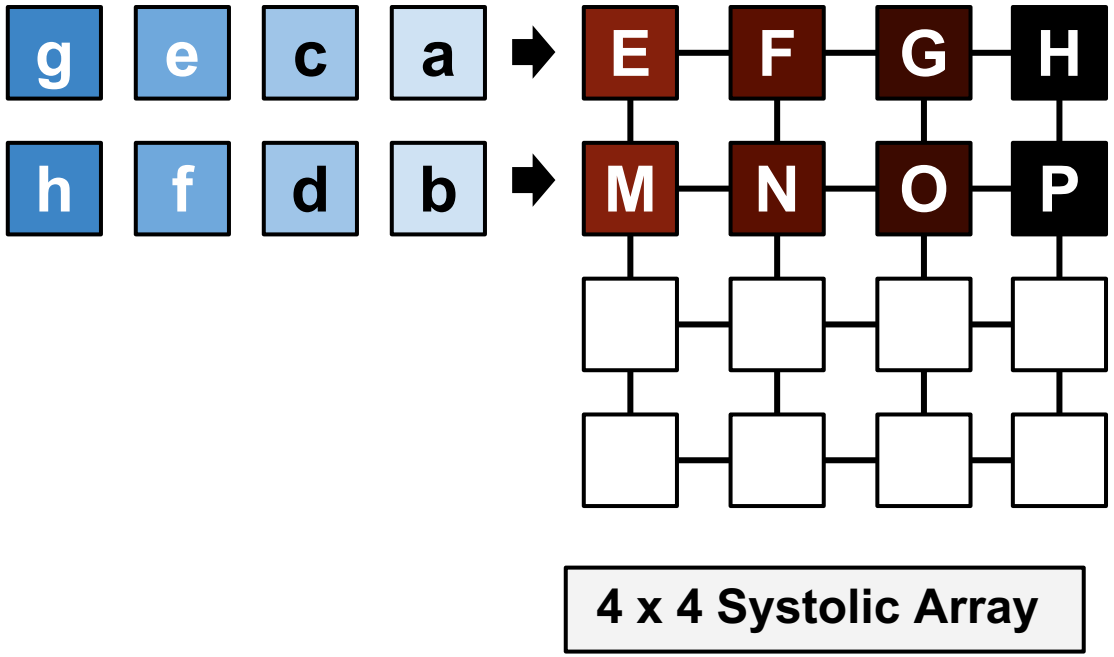


Example - Irregular GEMMs on SIGMA

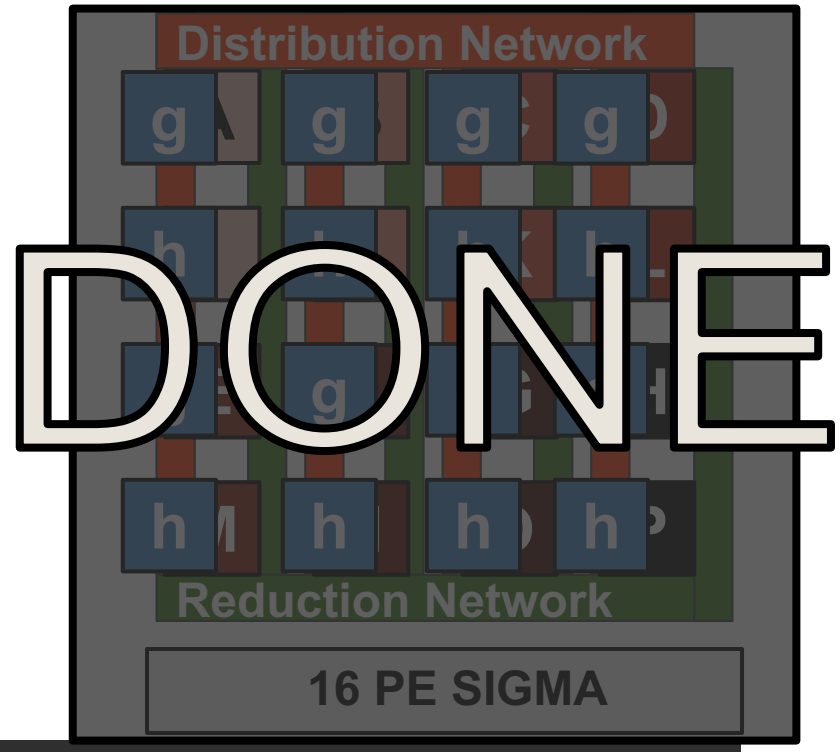


SIGMA reduces the number of folds, which then reduces the number of memory references on the streaming matrix.

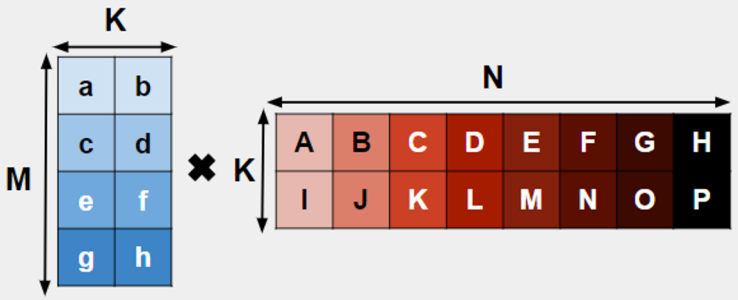
*** Assuming** stationary)



4 x 4 Systolic Array

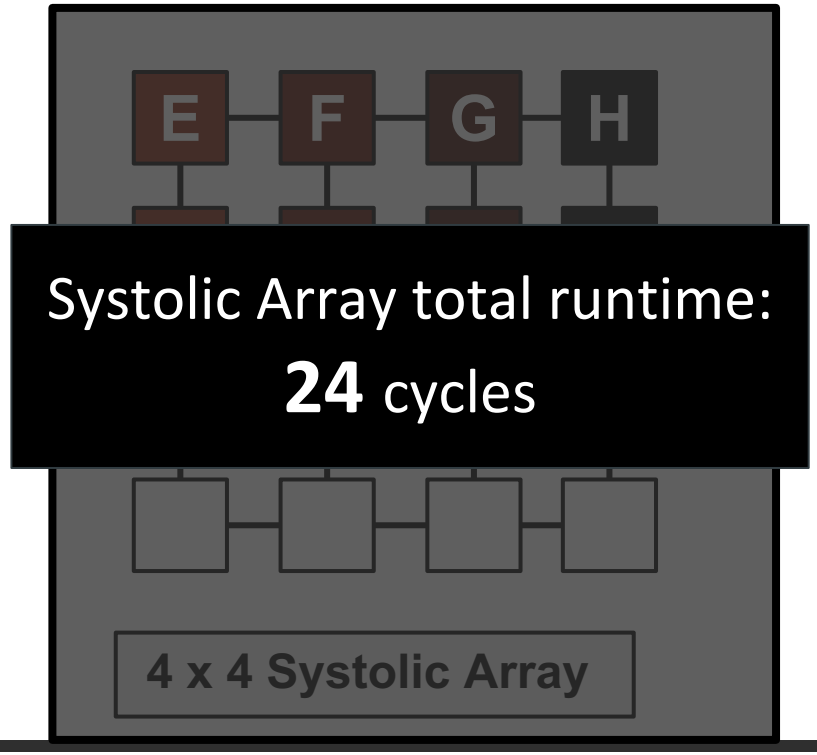


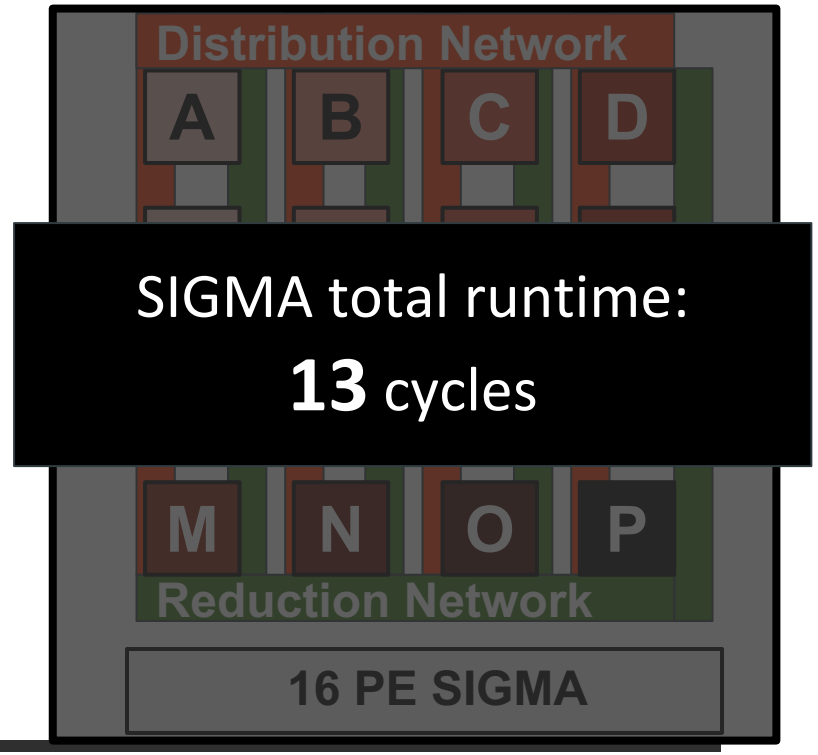
Example - Irregular GEMMs on SIGMA



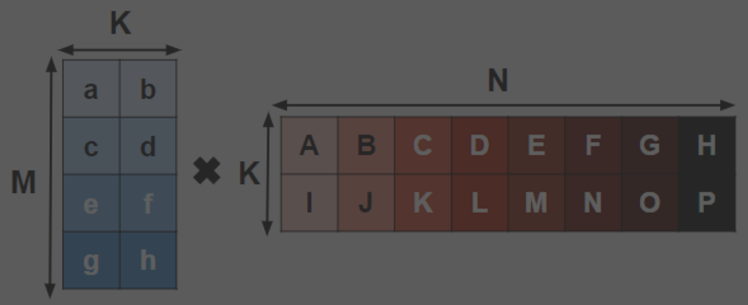
Final cycle count.

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





Example - Irregular GEMMs on SIGMA



Final cycle count.

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

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

Systolic Array total runtime:

24 cycles

4 x 4 Systolic Array

SIGMA total runtime:

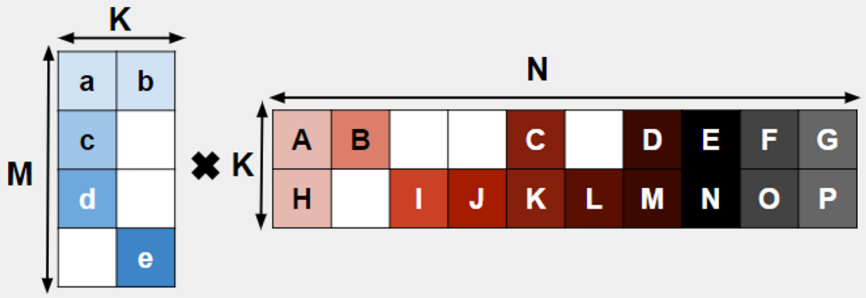
13 cycles

M N O P

Reduction Network

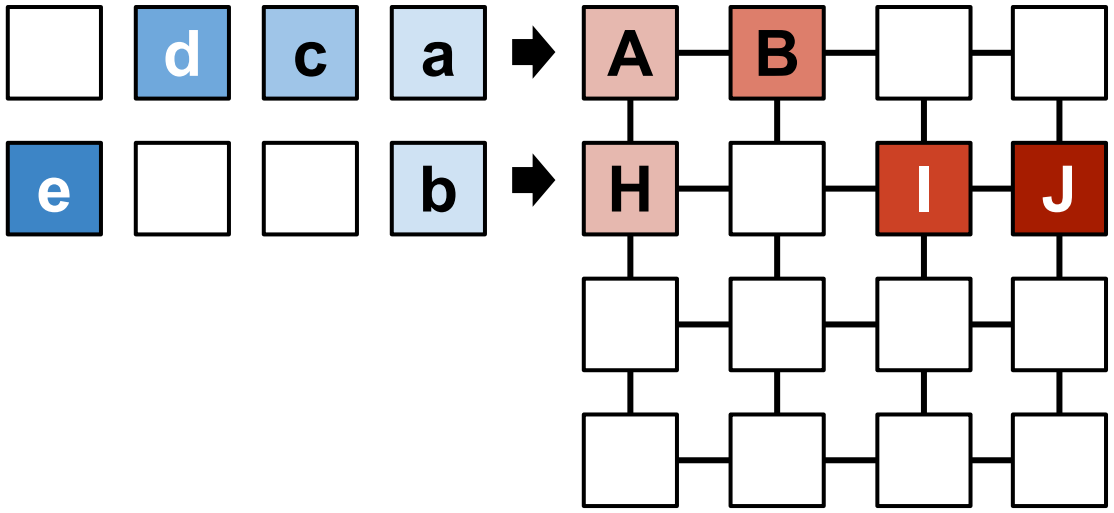
16 PE SIGMA

Example - Sparse Irregular GEMMs on SIGMA



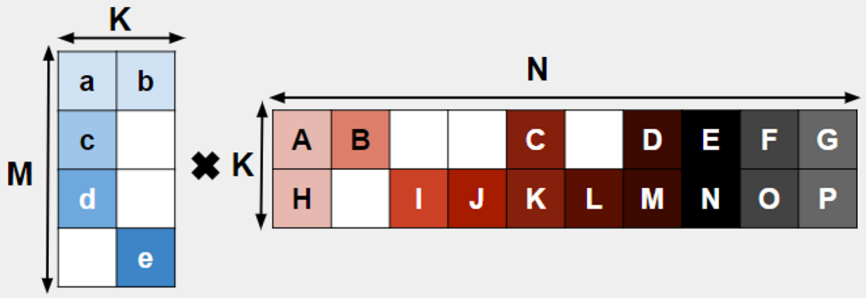
Set up the stationary values.

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



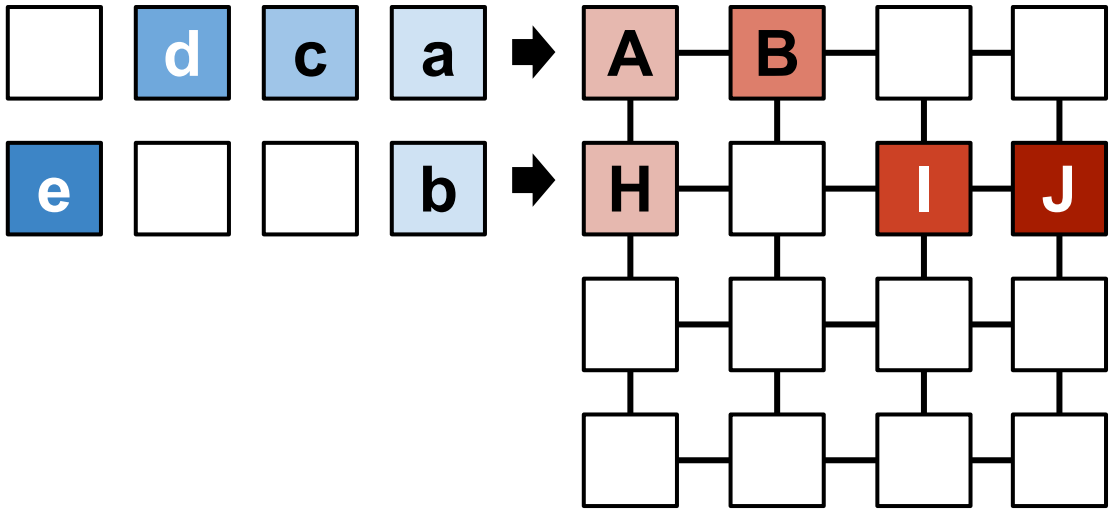
4 x 4 Systolic Array

Example - Sparse Irregular GEMMs on SIGMA

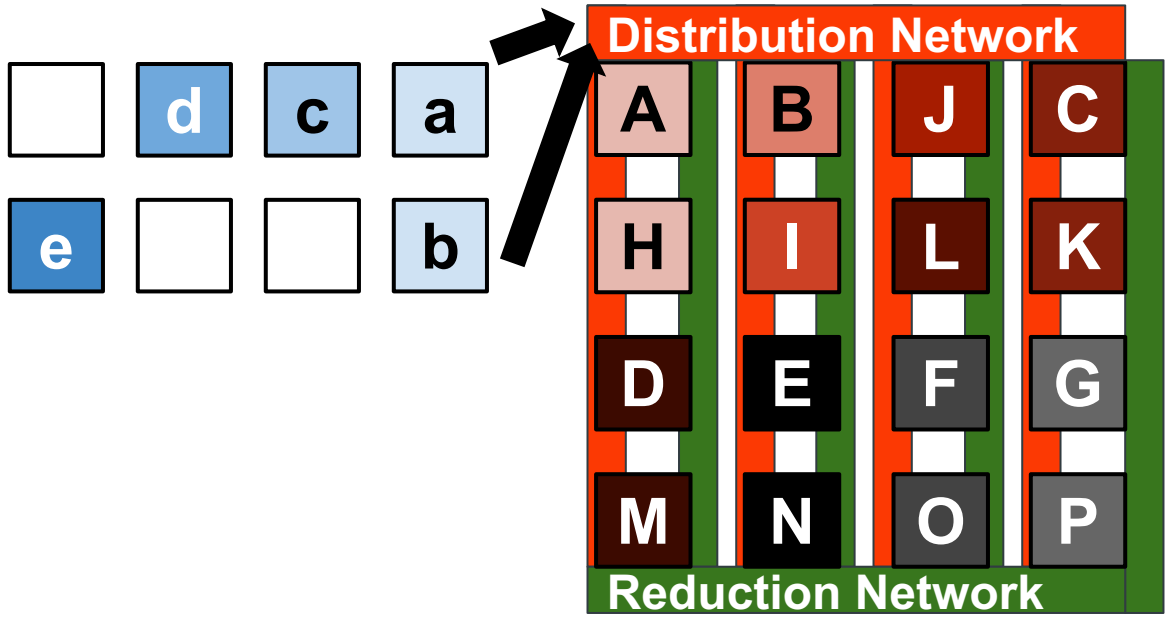


Set up the stationary values.

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

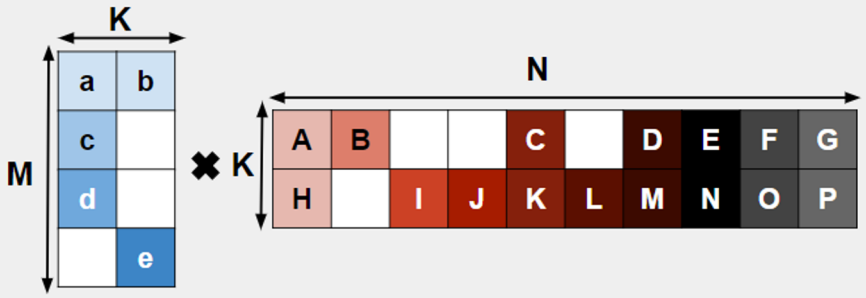


4 x 4 Systolic Array



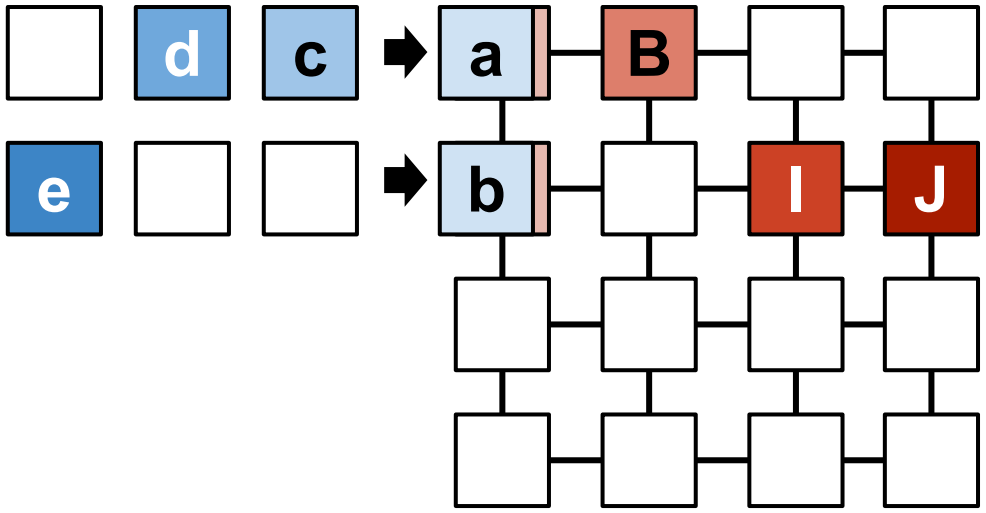
16 PE SIGMA

Example - Sparse Irregular GEMMs on SIGMA

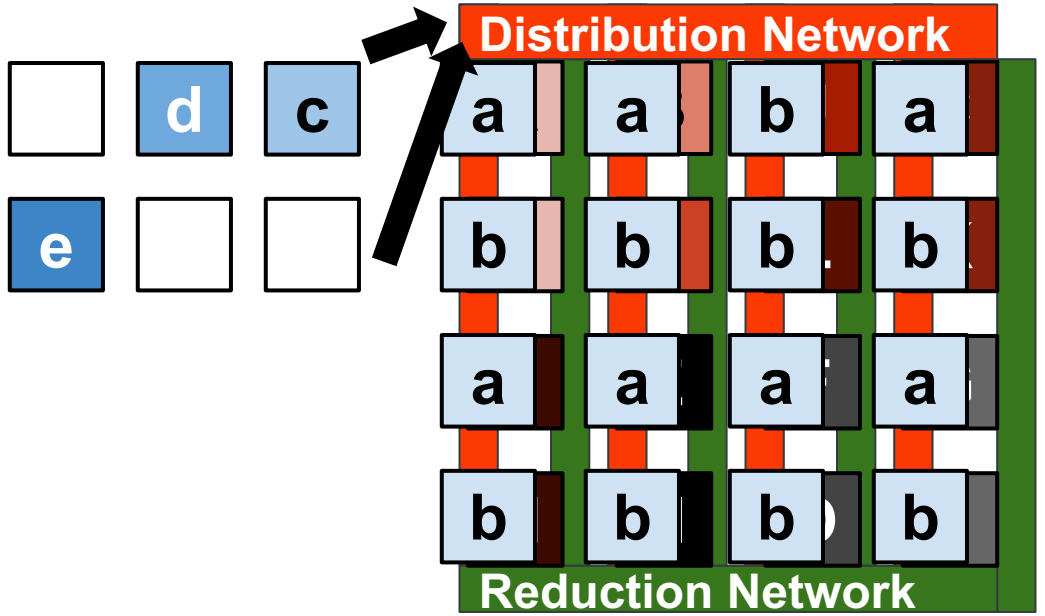


Next cycle: Multicast first row of MK to the corresponding stationary elements.

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

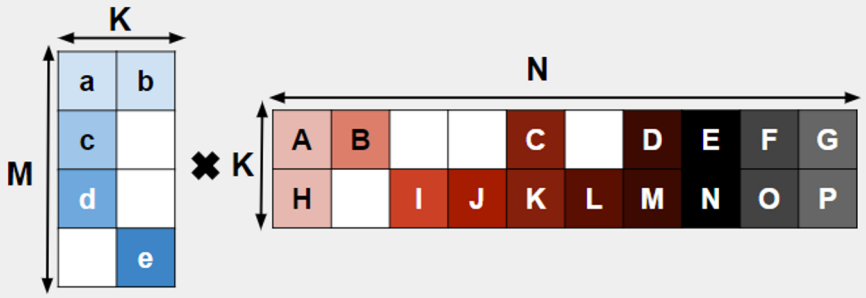


4 x 4 Systolic Array



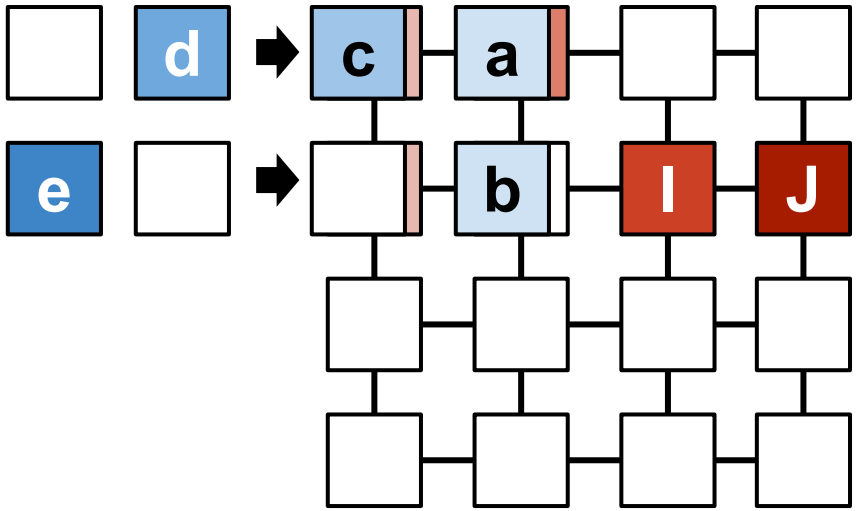
16 PE SIGMA

Example - Sparse Irregular GEMMs on SIGMA

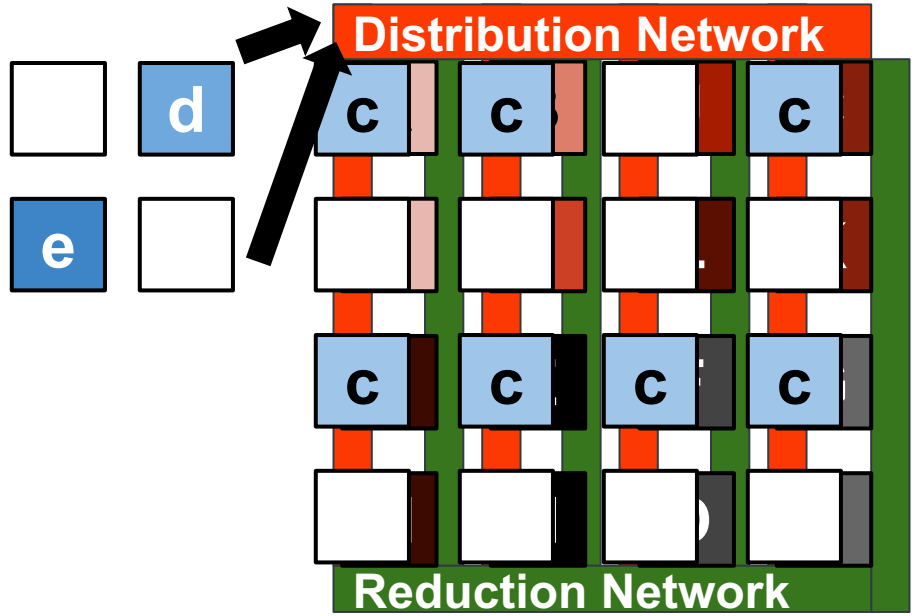


Next cycle: Multicast second row of MK to the corresponding stationary elements.

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

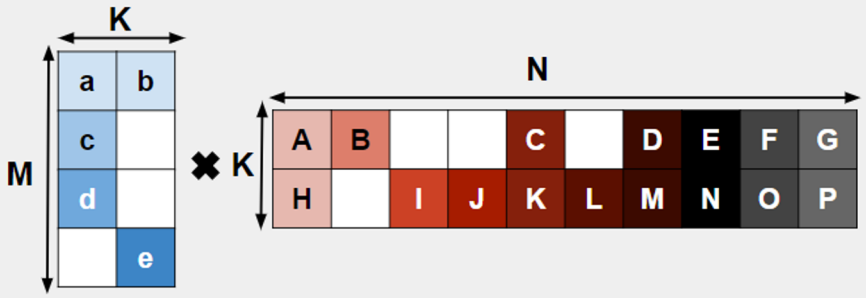


4 x 4 Systolic Array



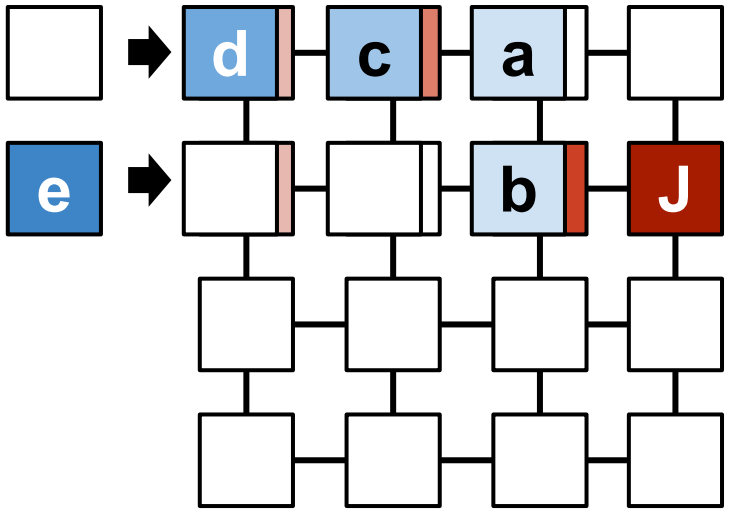
16 PE SIGMA

Example - Sparse Irregular GEMMs on SIGMA

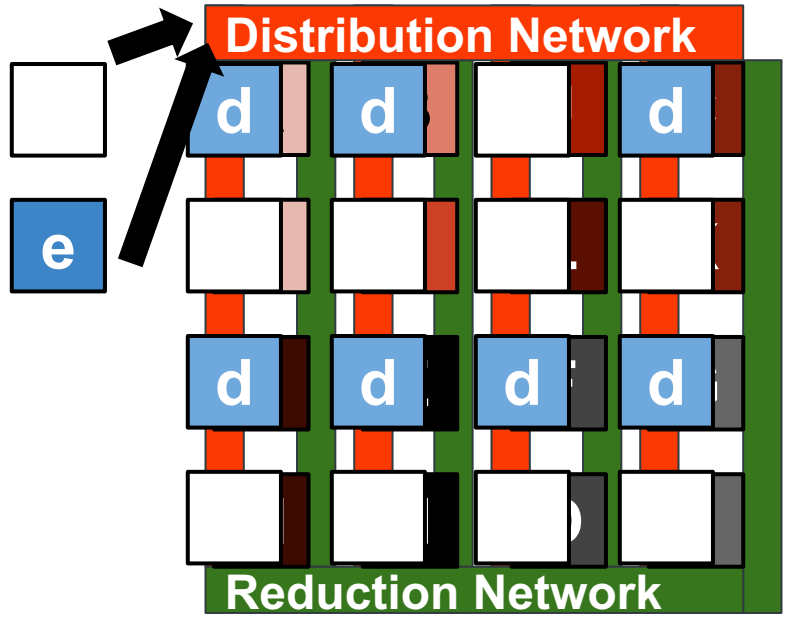


Next cycle: Multicast third row of MK to the corresponding stationary elements.

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

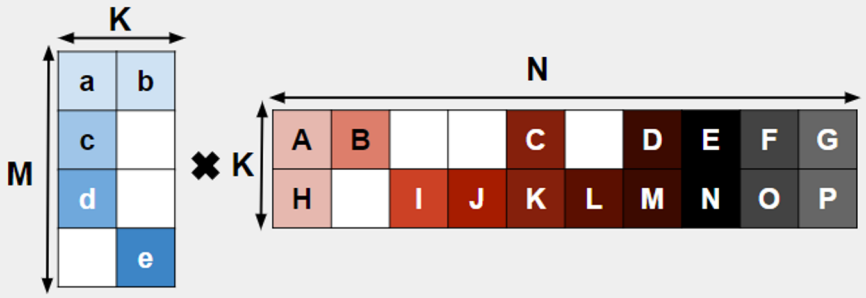


4 x 4 Systolic Array



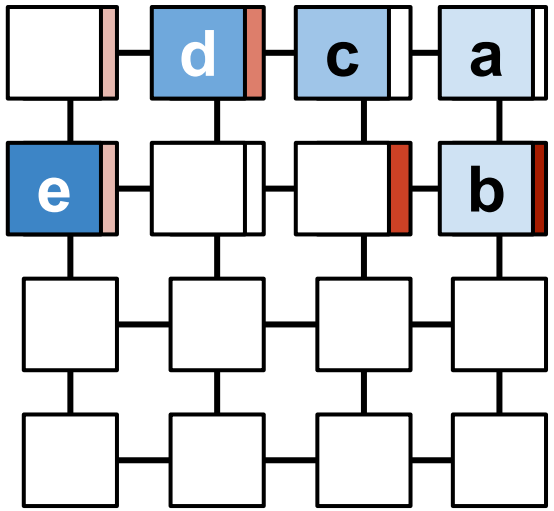
16 PE SIGMA

Example - Sparse Irregular GEMMs on SIGMA

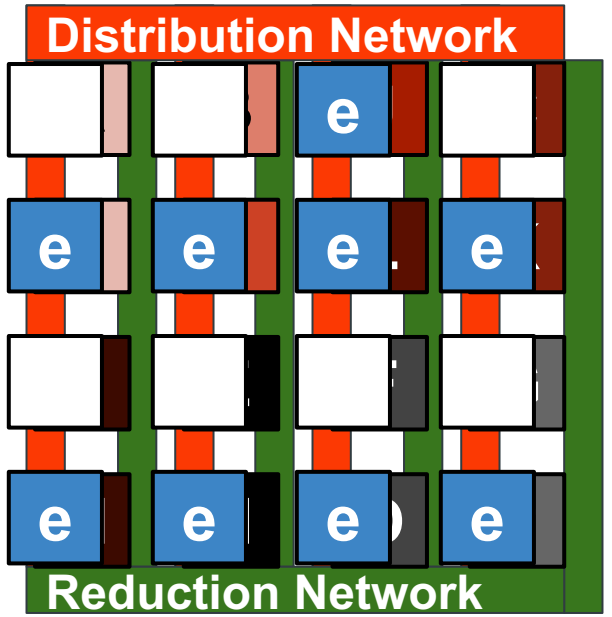


Next cycle: Multicast fourth row of MK to the corresponding stationary elements.

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

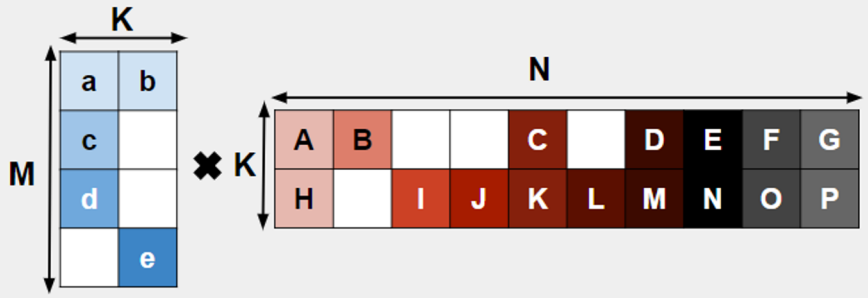


4 x 4 Systolic Array



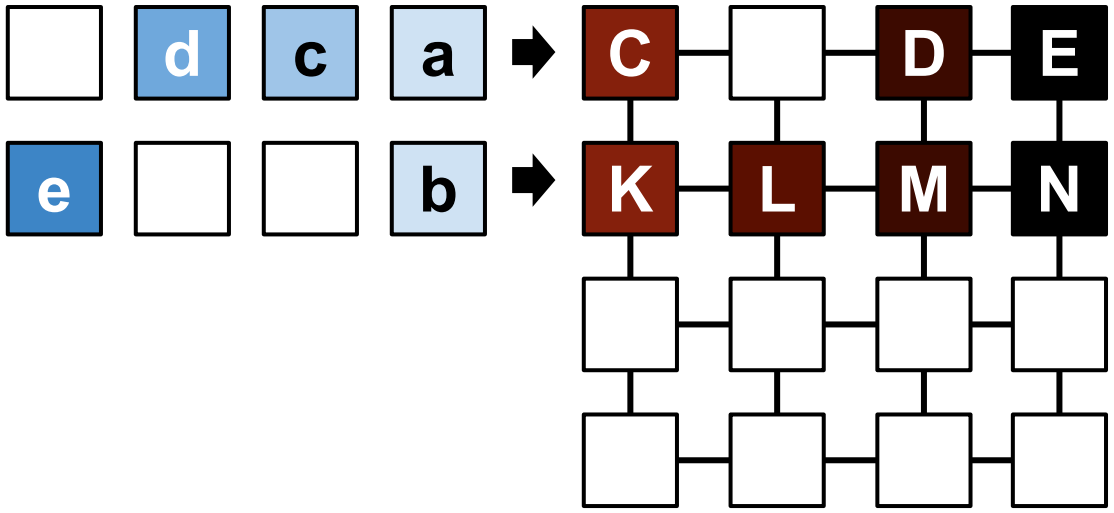
16 PE SIGMA

Example - Sparse Irregular GEMMs on SIGMA

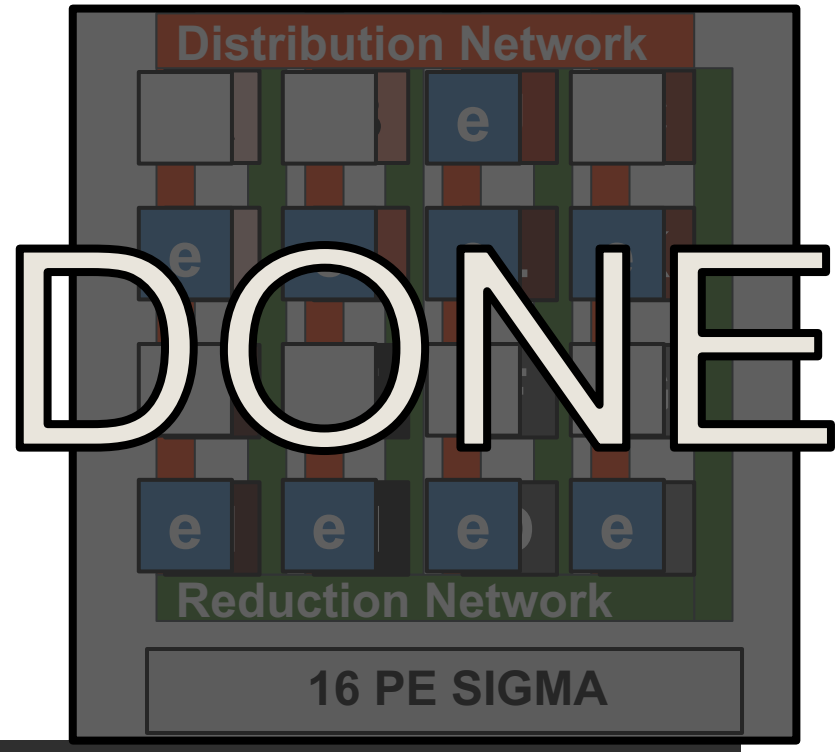


After accumulation, SIGMA is done. However, the systolic array has to map the other part of the stationary matrix and stream in the MK matrix again (referred to as folding).

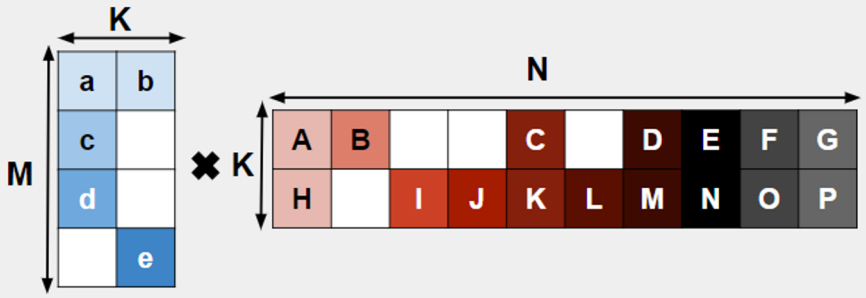
***** Assuming MK matrix is streaming and KN matrix is stationary. (aka weight stationary)***



4 x 4 Systolic Array

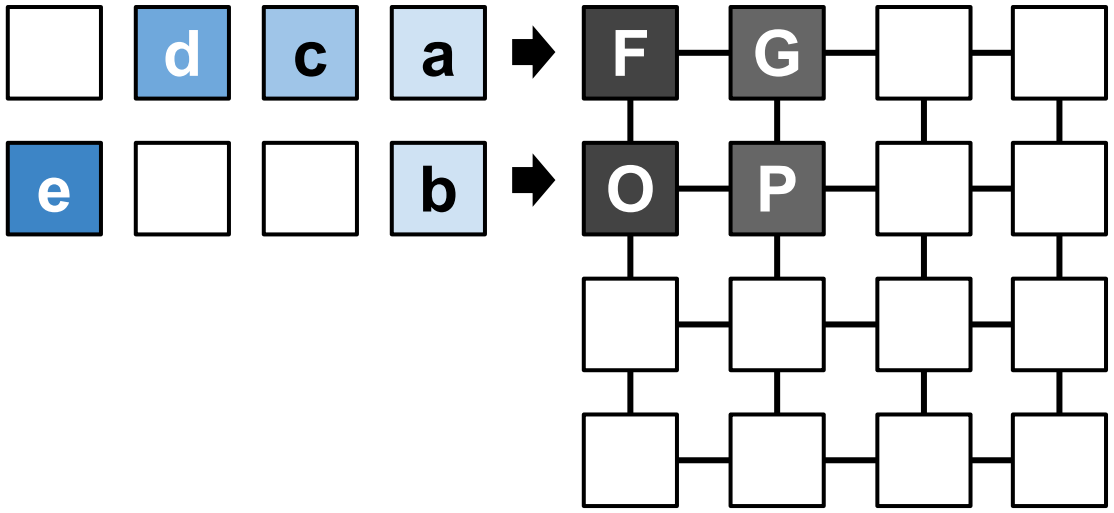


Example - Sparse Irregular GEMMs on SIGMA

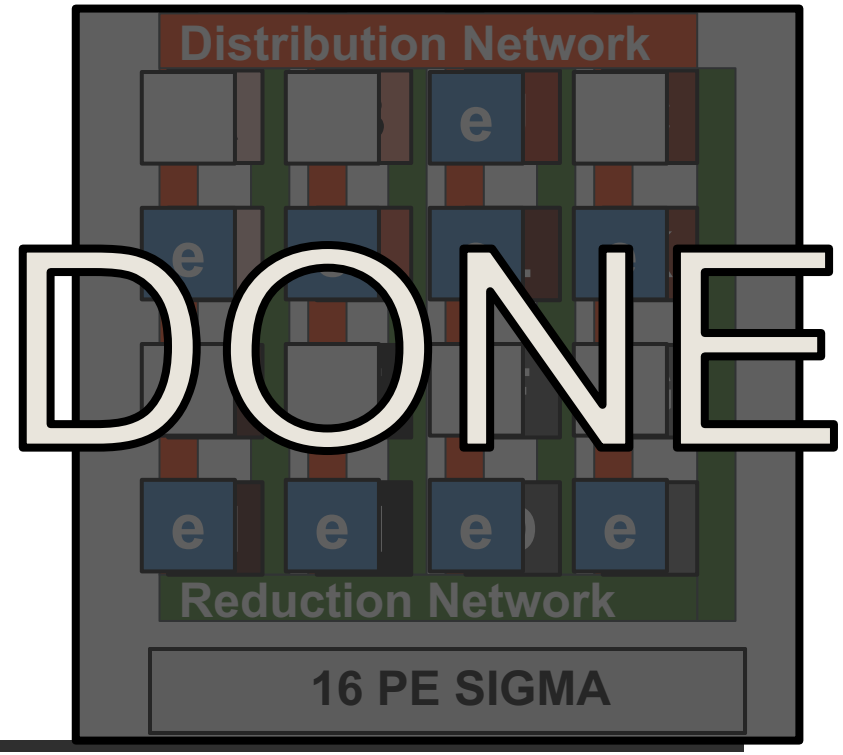


Again, the systolic array has to map another part of the stationary matrix and stream MK again.

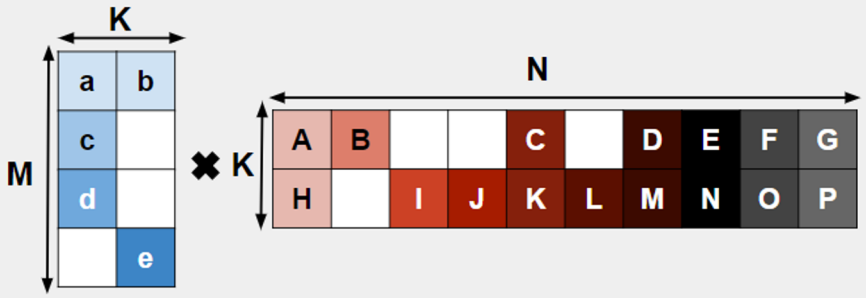
***** Assuming MK matrix is streaming and KN matrix is stationary. (aka weight stationary)***



4 x 4 Systolic Array

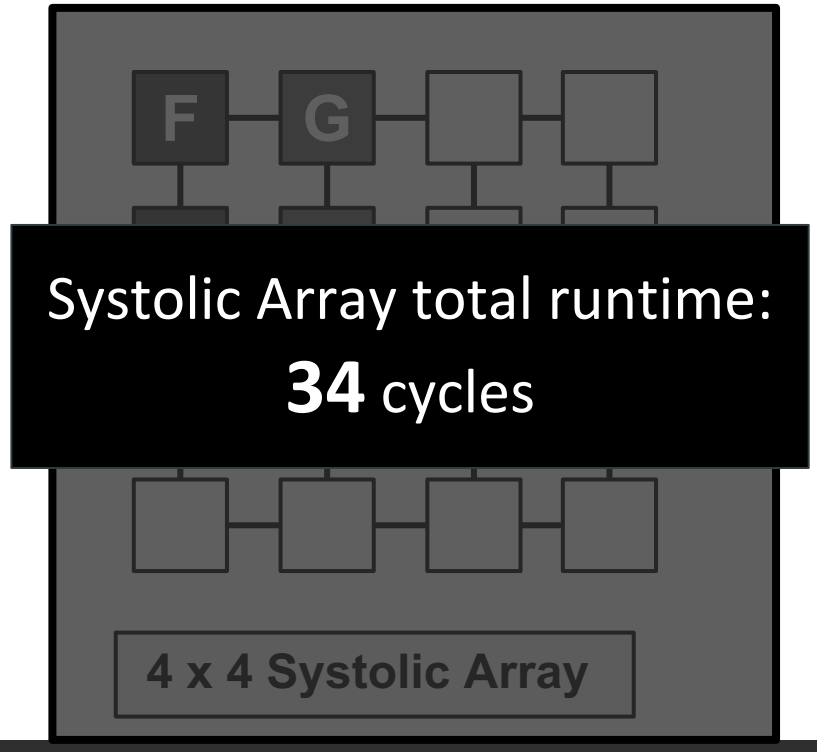


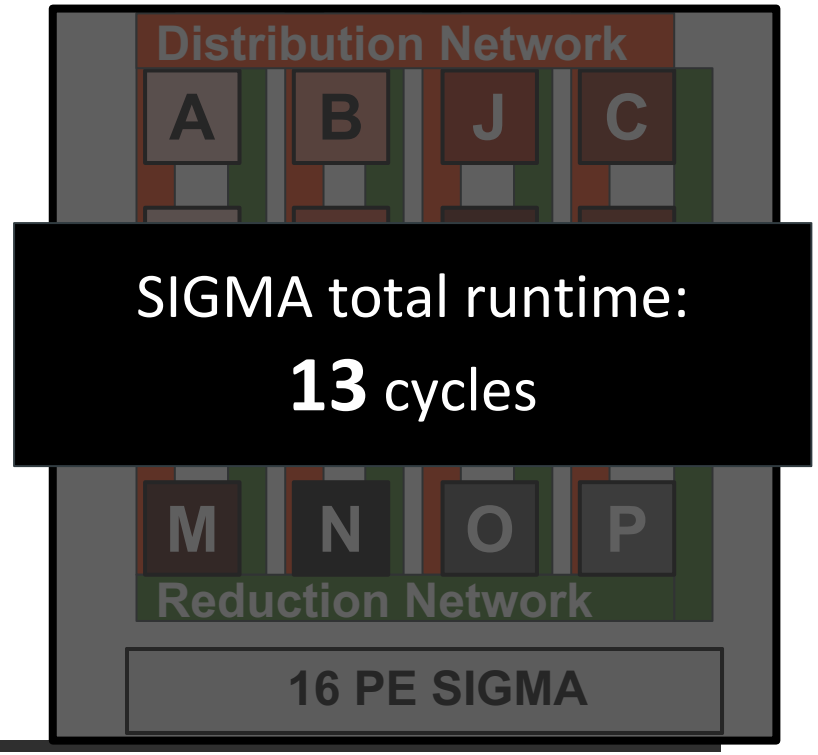
Example - Sparse Irregular GEMMs on SIGMA



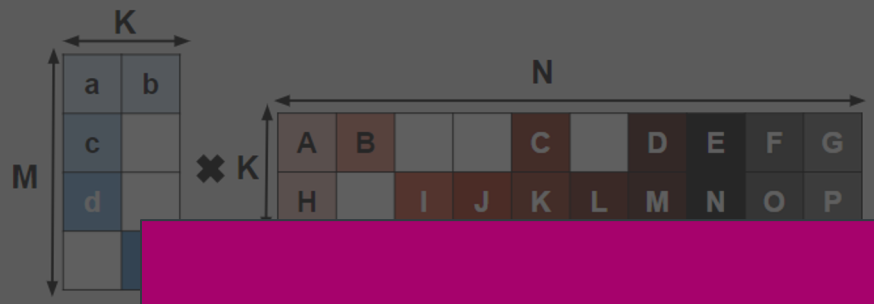
Final cycle count.

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





Example - Sparse Irregular GEMMs on SIGMA



Final cycle count.

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

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

34 cycles

4 x 4 Systolic Array

13 cycles

Reduction Network

16 PE SIGMA

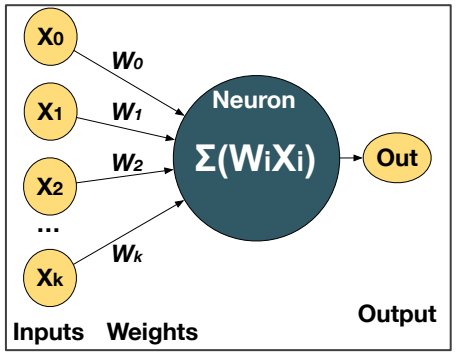
Outline

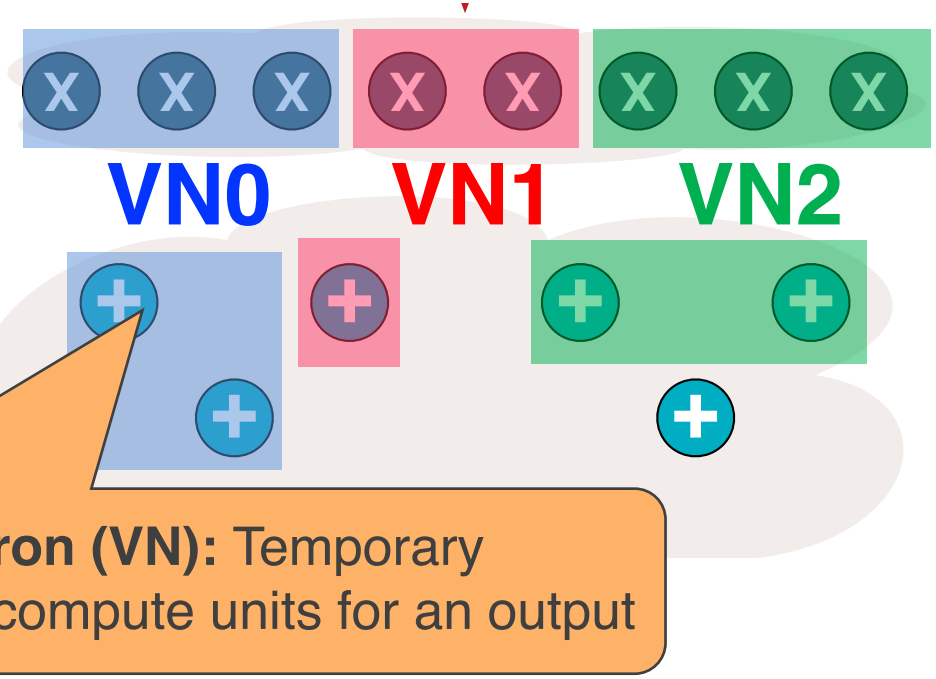
- DNN Accelerators
- Benefits of Mapping Flexibility
- Communication-centric Accelerators
- Case Study
  - MAERI
    - Abstraction
    - Implementation
    - Mapping
  - SIGMA
- Conclusion

Hyoukjun Kwon, Ananda Samajdar, and Tushar Krishna
***MAERI**: Enabling Flexible Dataflow Mapping over DNN Accelerators via Reconfigurable Interconnects:*
ASPLOS 2018**, **IEEE Micro Top Picks 2019 Honorable Mention

The MAERI Abstraction

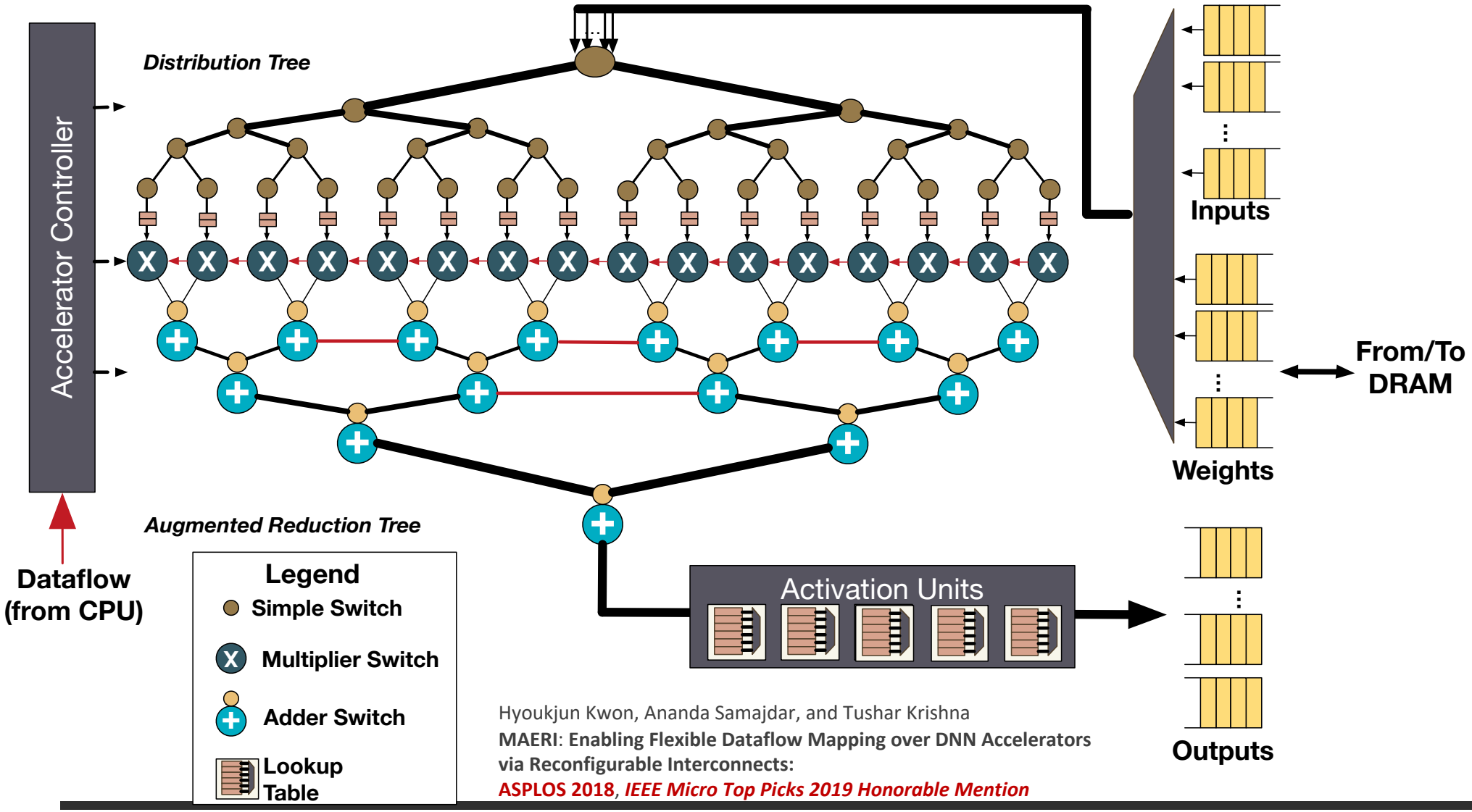
MAERI





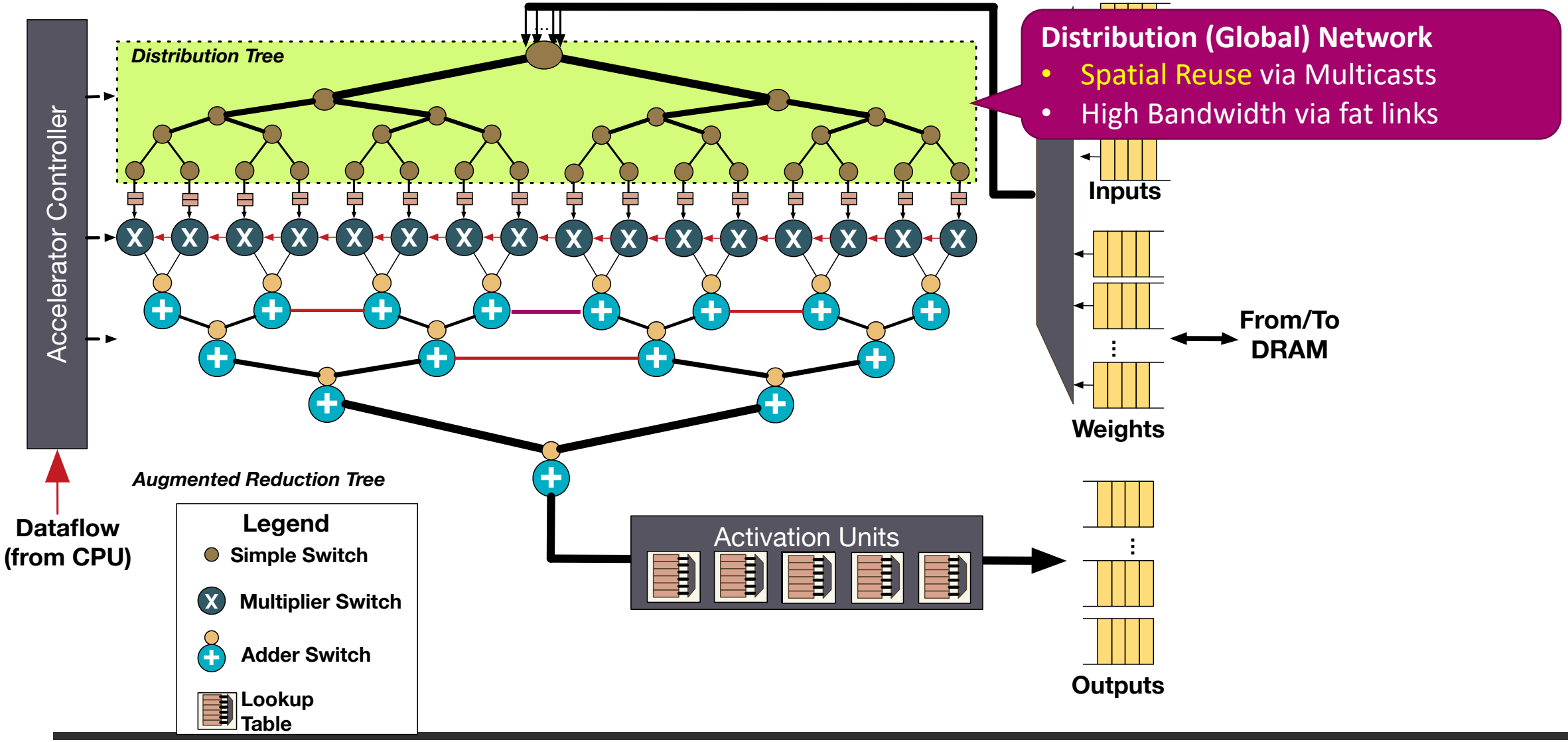
Each Virtual Neuron (aka cluster) calculates a dot-product

The MAERI Implementation

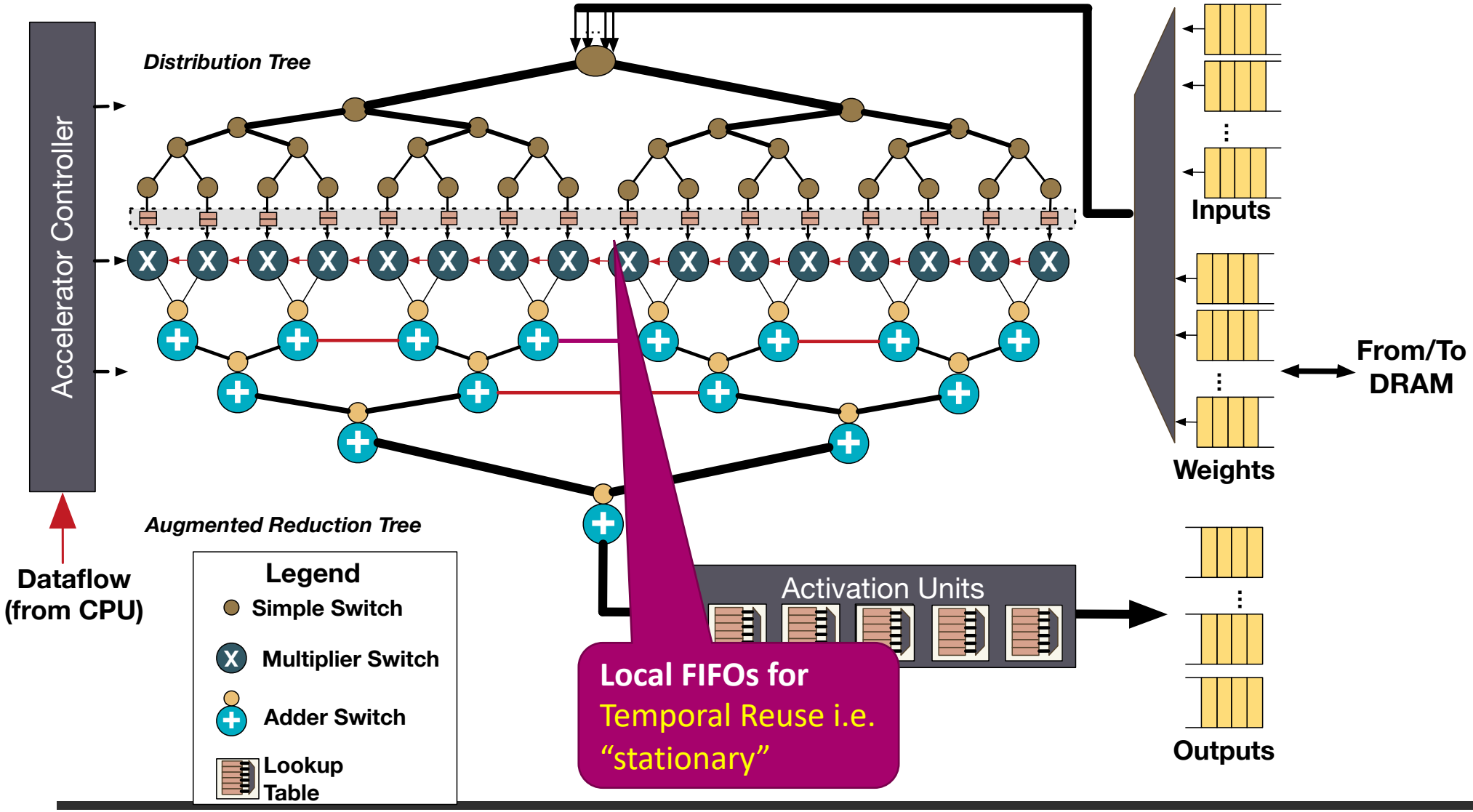


Hyoukjun Kwon, Ananda Samajdar, and Tushar Krishna
MAERI: Enabling Flexible Dataflow Mapping over DNN Accelerators via Reconfigurable Interconnects:
ASPLOS 2018, ***IEEE Micro Top Picks 2019 Honorable Mention***

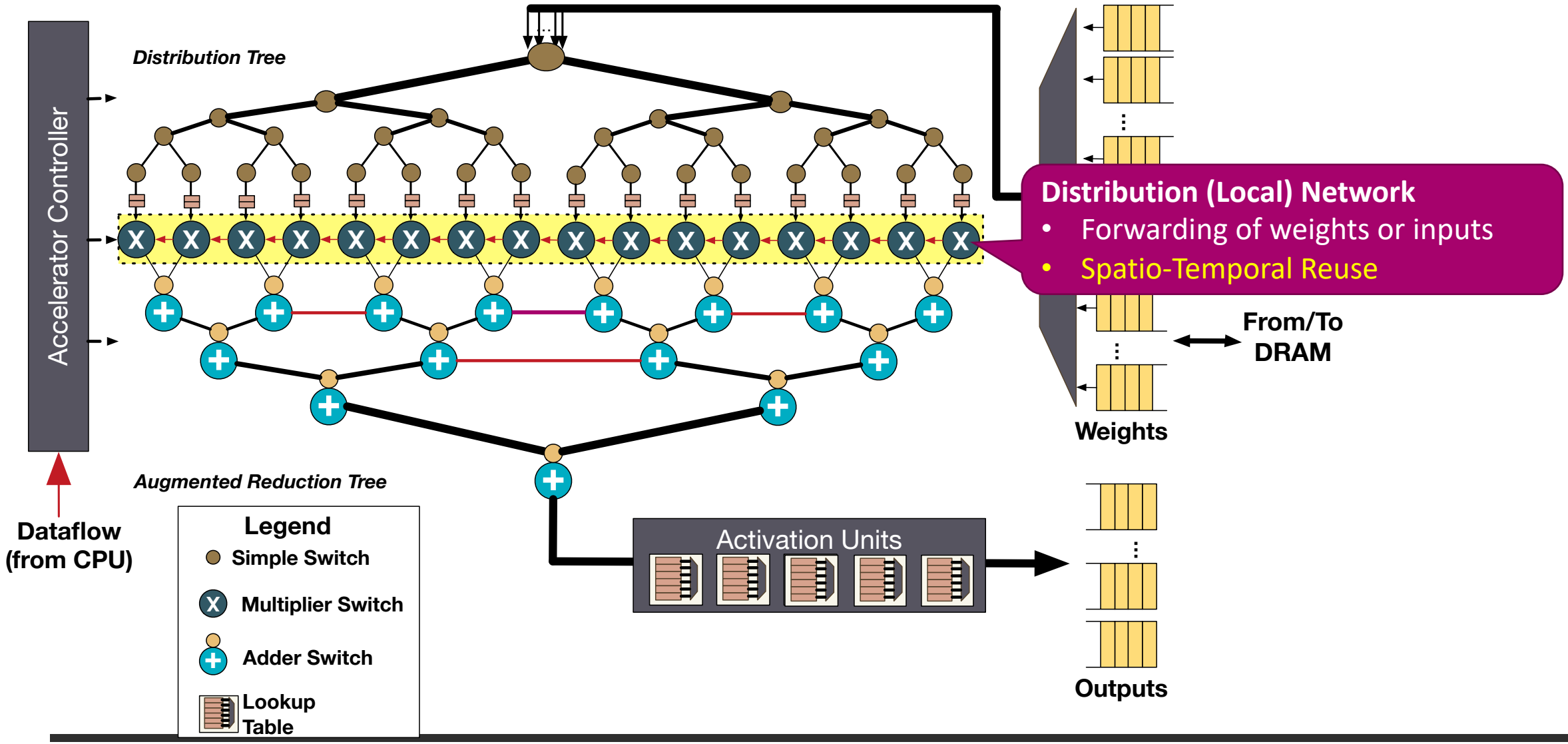
The MAERI Implementation



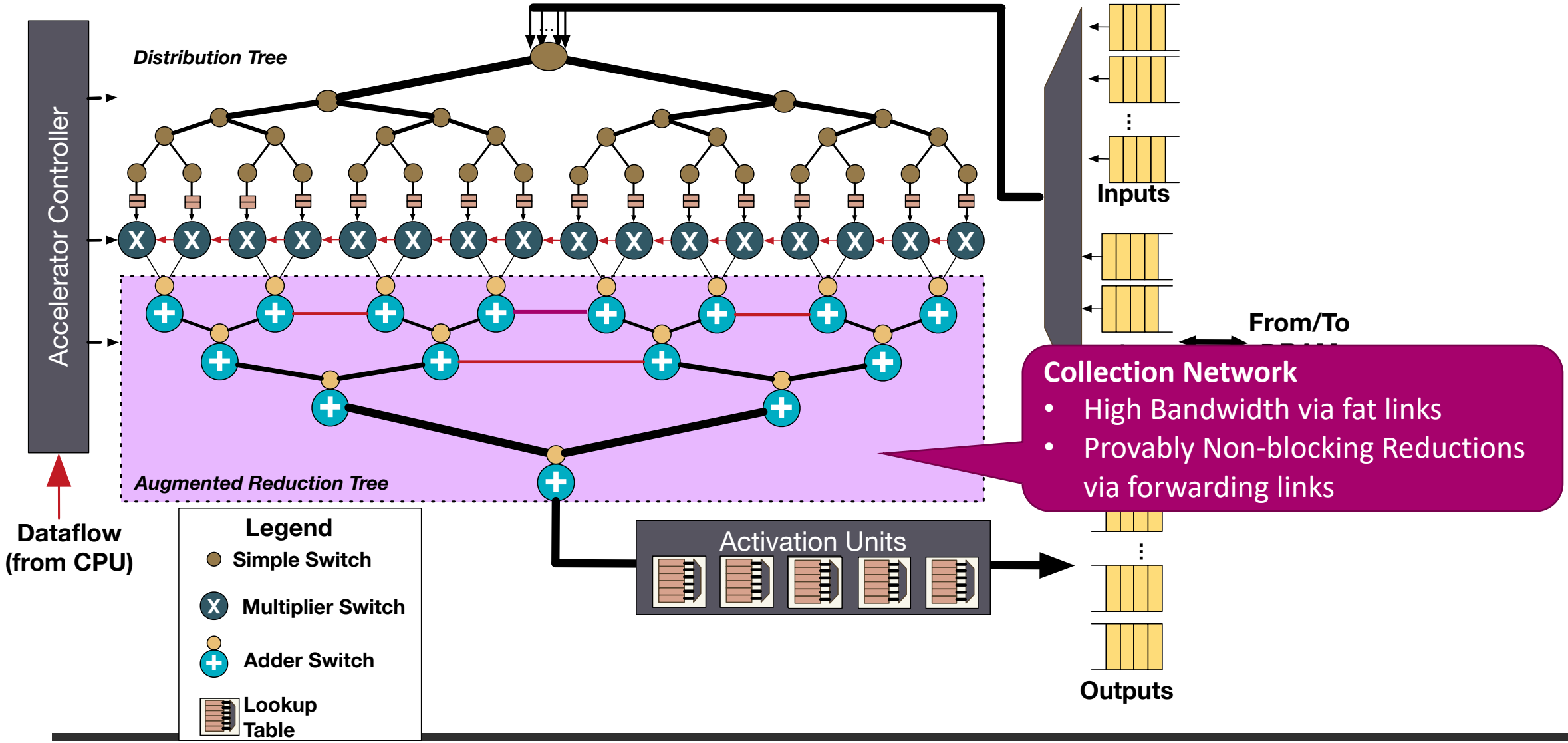
The MAERI Implementation



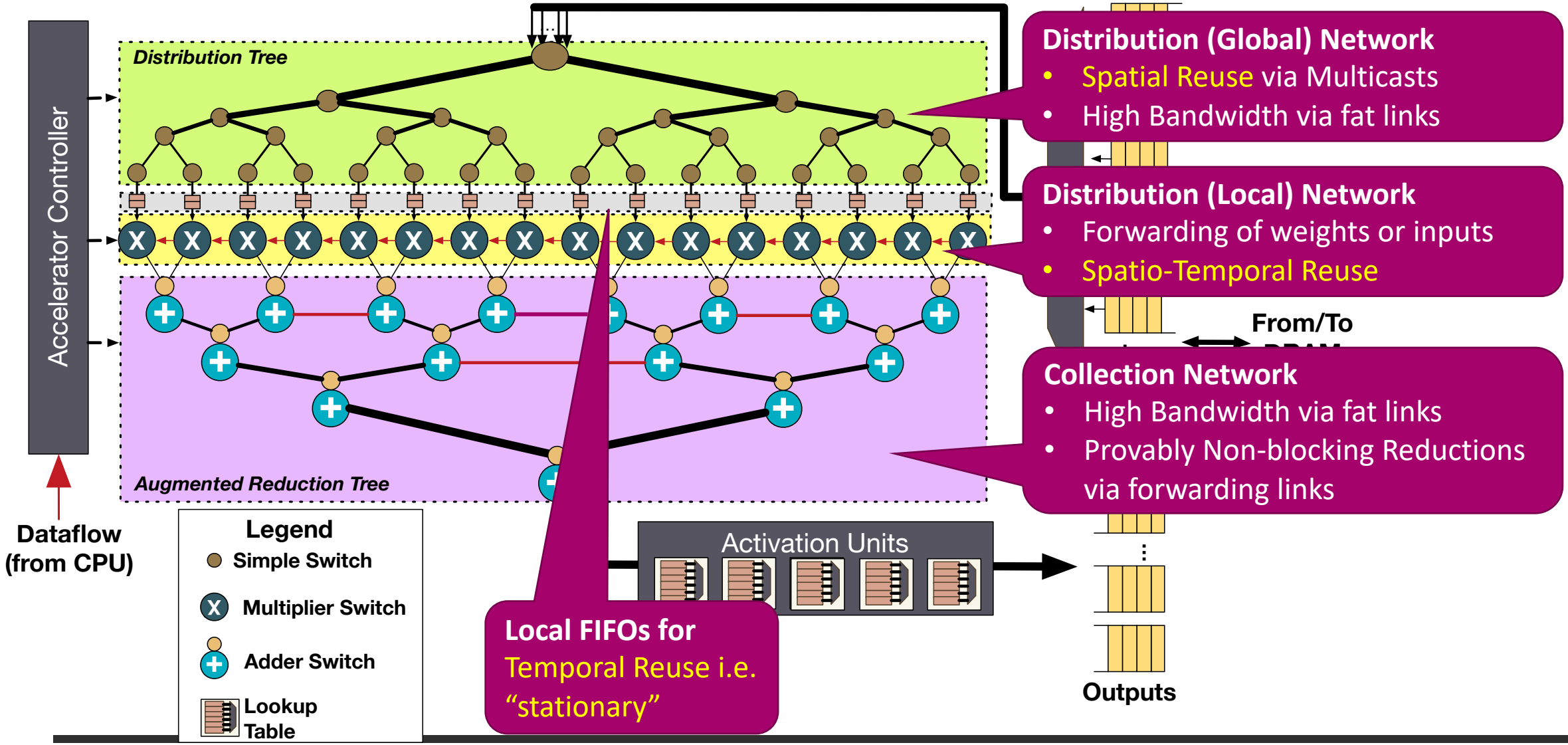
The MAERI Implementation

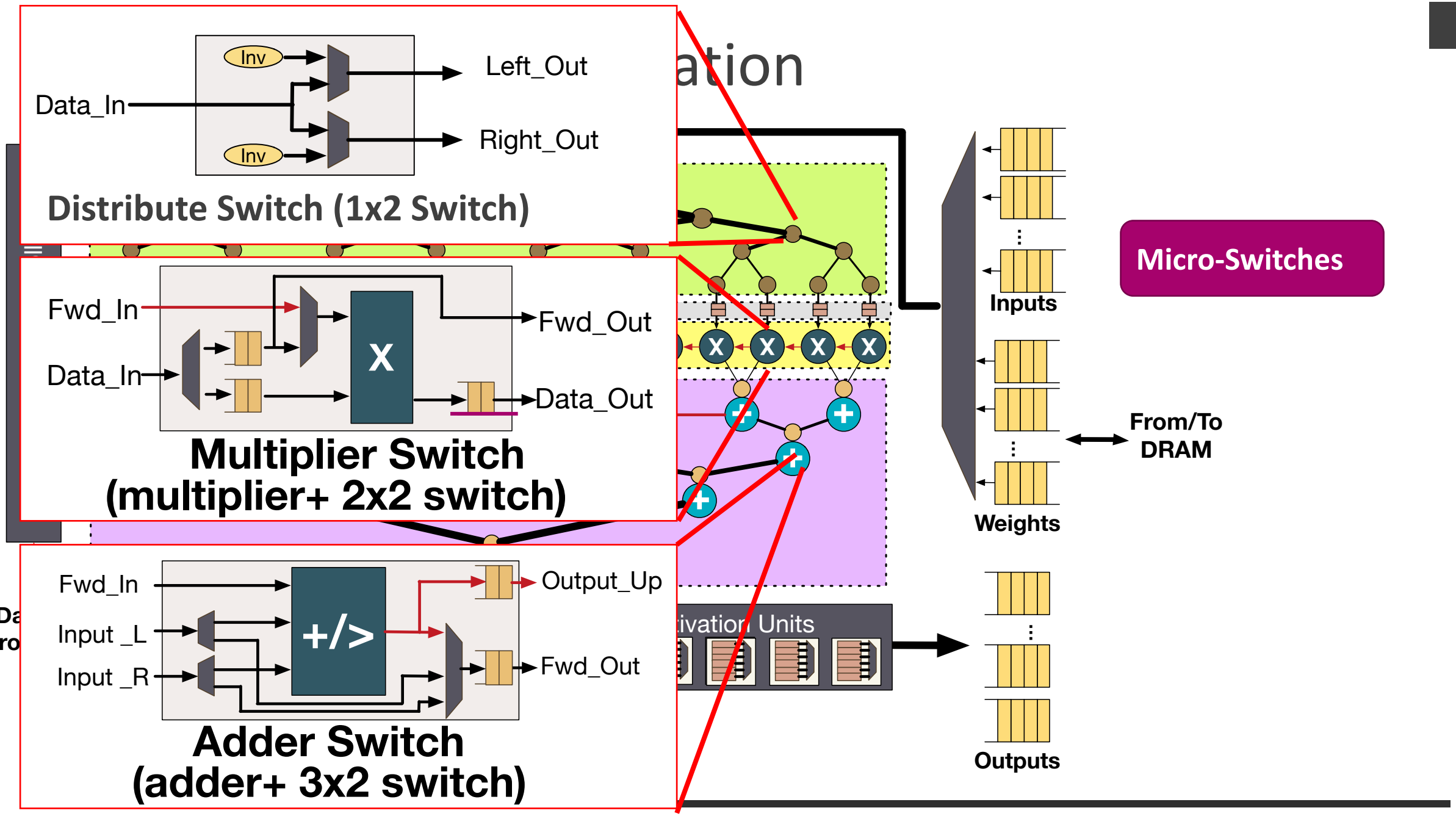


The MAERI Implementation



The MAERI Implementation




Inv
Data_In
Inv
Left_Out
Right_Out
Distribute Switch (1x2 Switch)
Fwd_In
Data_In
X
Fwd_Out
Data_Out
Multiplier Switch
(multiplier+ 2x2 switch)
Fwd_In
Input _L
Input _R
+/>
Output_Up
Fwd_Out
Adder Switch
(adder+ 3x2 switch)
Micro-Switches
Inputs
Weights
From/To
DRAM
Outputs

Summary of MAERI's Features

- **A Communication-Centric Approach to DNN Accelerator Design**
- **Specialized Topology**
  - **Tree based distribution**
    - ✓ Multiple simultaneous ***multicasts*** of variable sizes
  - **Novel tree topology for collection**
    - ✓ Multiple simultaneous ***reductions*** (dot-products) of variable sizes (i.e., virtual neuron)
- **Supports all kinds of data reuse (that a mapper can leverage)**
  - ✓**Temporal:** enabled via local buffers in mult. switches
  - ✓**Spatial (Multicasting):** enabled via distribution tree
  - ✓**Spatio-Temporal (St-and-Fwd):** enabled via forwarding links between multipliers
- **Reconfigurable light-weight micro-switches**
  - Exposed to Mapper to support multiple DNNs and Dataflow strategies

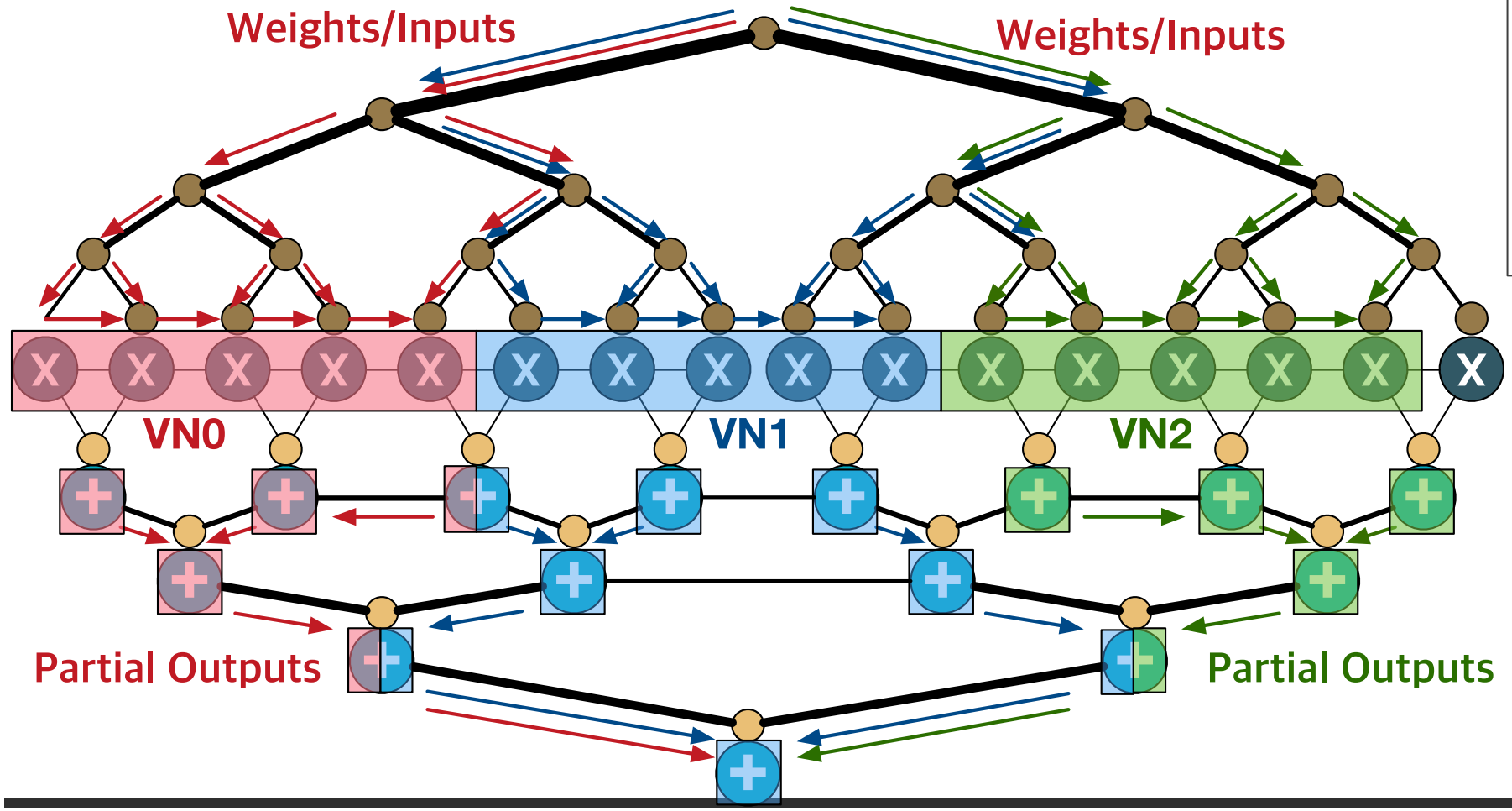
Outline

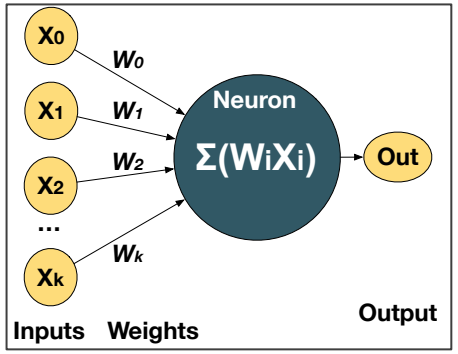
- DNN Accelerators
- Benefits of Mapping Flexibility
- Communication-centric Accelerators
- Case Study
  - MAERI
    - Abstraction
    - Implementation
    - Mapping
  - SIGMA
- Conclusion

Hyoukjun Kwon, Ananda Samajdar, and Tushar Krishna
***MAERI**: Enabling Flexible Dataflow Mapping over DNN Accelerators via Reconfigurable Interconnects:*
ASPLOS 2018**, **IEEE Micro Top Picks 2019 Honorable Mention

Example Mapping – Dense CNN

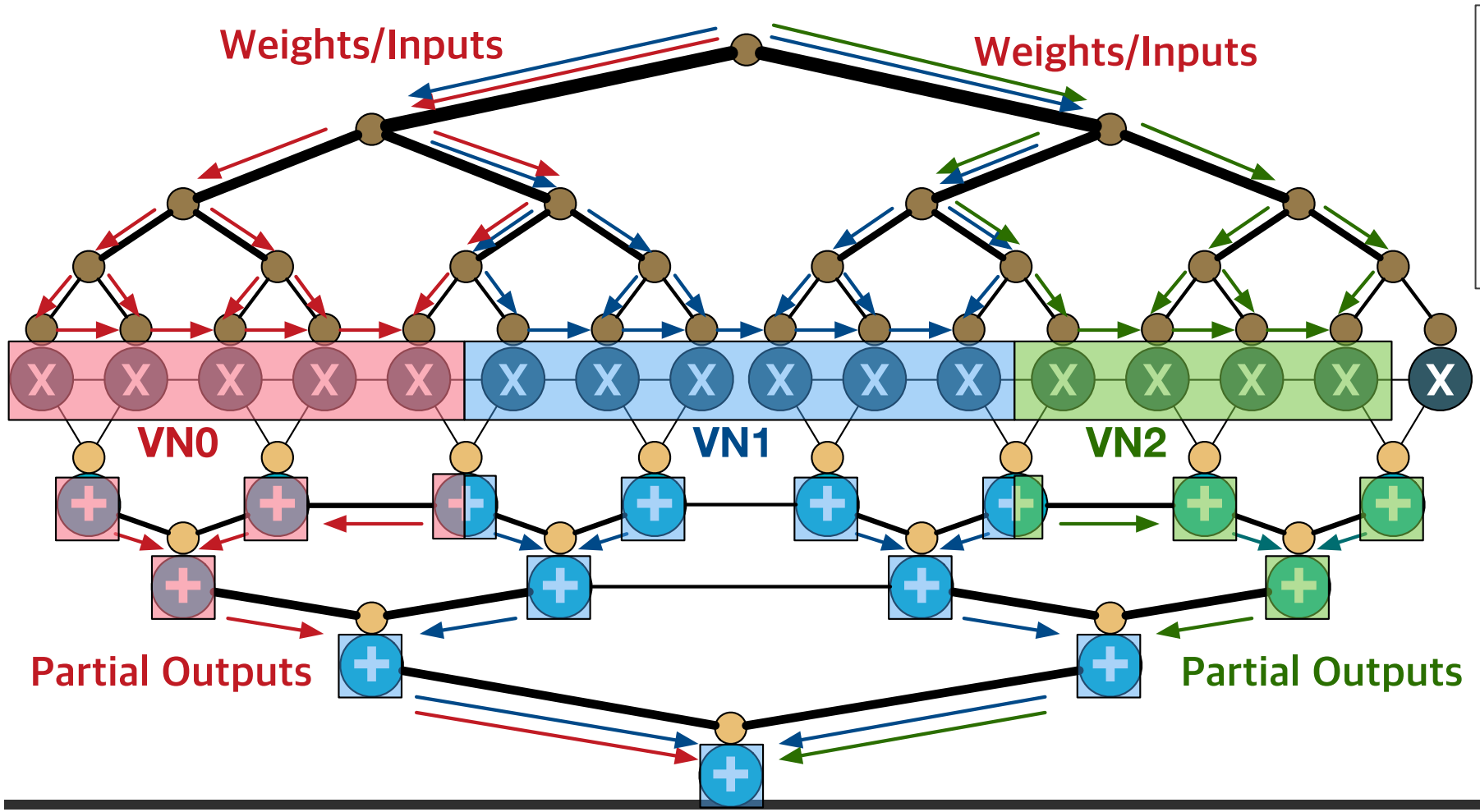
Our Key insight: Each mapping translates into neurons of different sizes

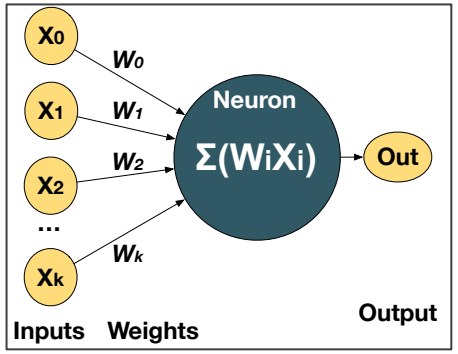




Example Mapping – Sparse DNN

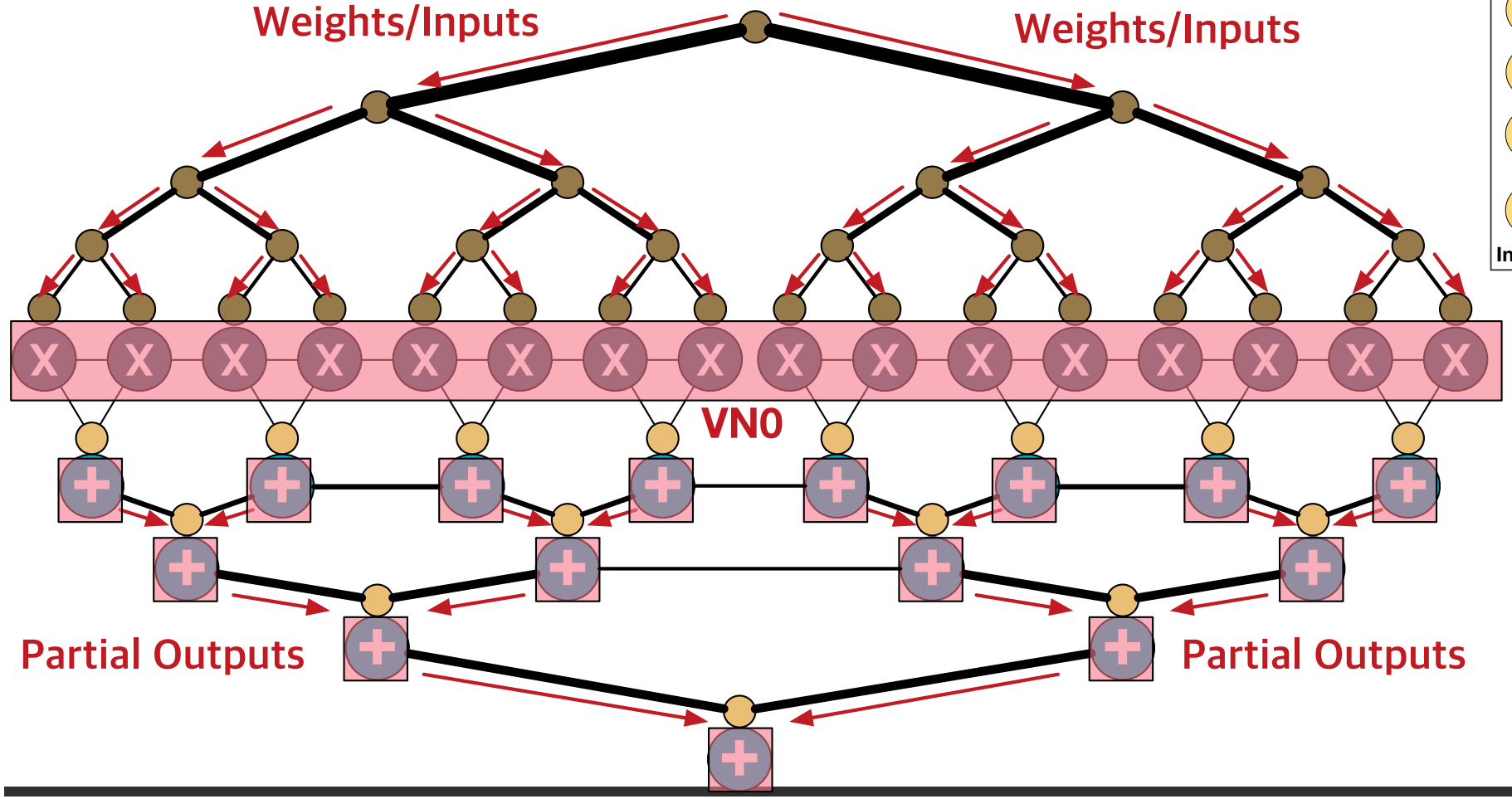
Our Key insight: Each mapping translates into neurons of different sizes

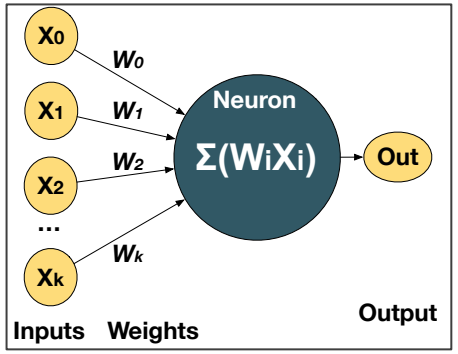




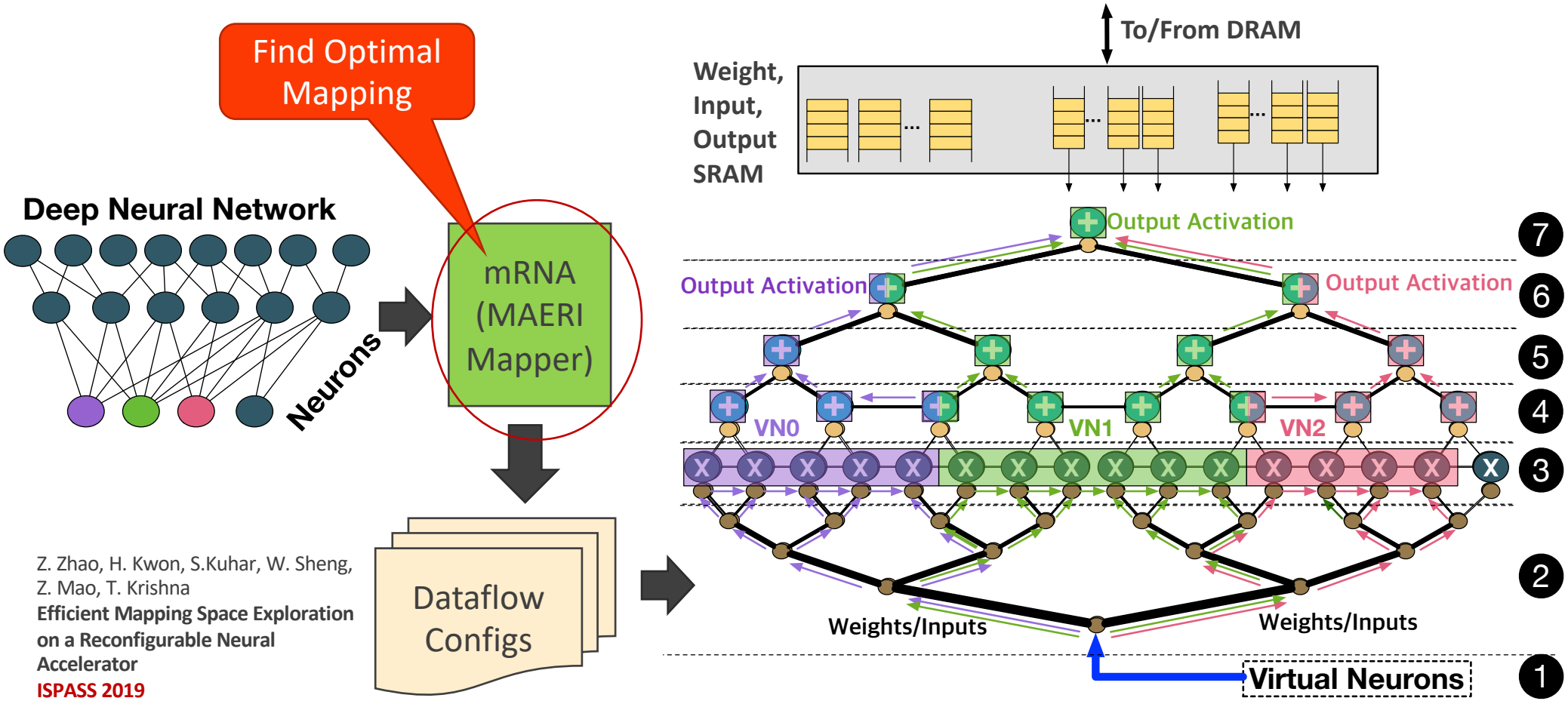
Example Mapping – GEMM/FC

Our Key insight: Each mapping translates into neurons of different sizes





Searching Optimal Mappings for MAERI



Z. Zhao, H. Kwon, S.Kuhar, W. Sheng, Z. Mao, T. Krishna
Efficient Mapping Space Exploration on a Reconfigurable Neural Accelerator
ISPASS 2019

~100% Utilization

MAERI

Hyoukjun Kwon, Ananda Samajdar, and Tushar Krishna
MAERI: Enabling Flexible Dataflow Mapping over DNN Accelerators via Reconfigurable Interconnects:
ASPLOS 2018, ***IEEE Micro Top Picks 2019 Honorable Mention***

Loop optimizations for CNN

```
for(n=0; n<N; n++) {
 for(k=0; k<K; k++) {
  for(y=0; x<X'; y++) {
   for(x=0; y<Y'; x++) {
    O[n][k][x][y] = ReLU(O[n][k][x][y] );
    for(c=0; c<C; c++) {
     for(j=0; j<R; j++) {
      for(i=0; i<S; i++) {
       O[n][k][x][y] += W[k][c][i][j] *
                        I[n][c][x+i][y+j];
}}}}}}}
```

Loop fission

Loop tiling/blocking

Loop unrolling

Loop interchanging

```
for(n=0; n<N; n=n+T_N) {
 for(k=0; k<K; k=k+T_K) {
  for(x=0; x<X'; x=x+T_X') {
   for(y=0; y<Y'; y=y+T_Y') {
    for(c=0; c<C; c=c+T_C) {
     for(j=0; j<R; j=j+T_X) {
      for(i=0; i<S; i=i+T_Y) {
       for(tn=n; tn<T_N; tn++)
        for(tk=k; tk<T_K; tk++) {
         for(tx=x; tx<T_X'; tx++) {
          for(ty=y; ty<T_Y'; ty++) {
           for(tc=c; tc<T_C; tc++) {
            for(tj=j; tj<T_X; tj++) {
             for(ti=i; ti<T_Y; ti++) {
              O[tn][tk][tx][ty] +=
              W[tk][tc][ti][tj] *
              I[tn][tc][tx+ti][ty+tj];
}}}}}}}}}
```

tile

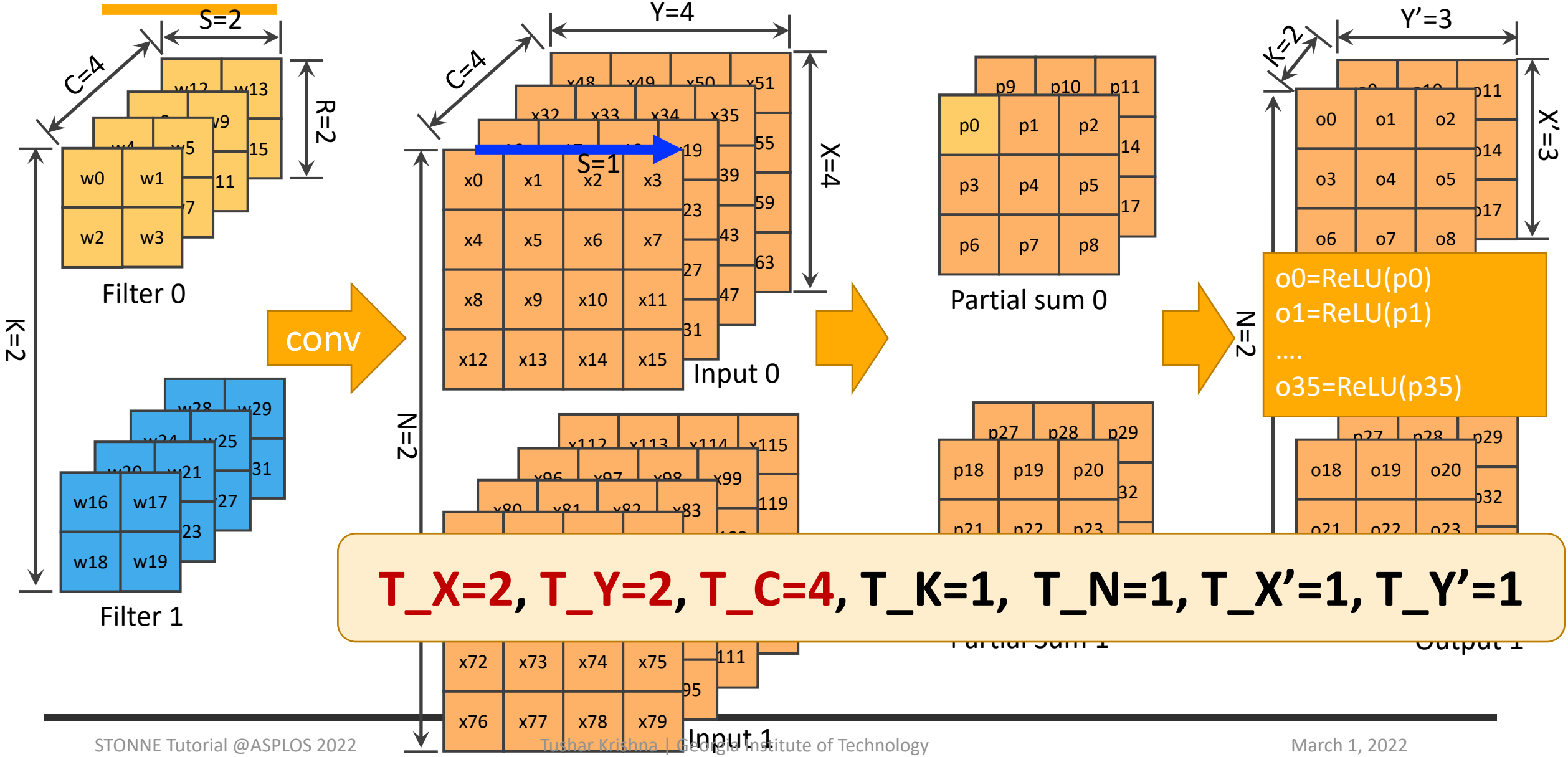
- Loop fission: decide the reconfiguration format of MAERI
- Loop tiling: decide the tile shape that is mapped onto MAERI
- Loop unrolling: unrolling all the operations inside the kernel onto MAERI
- Loop interchanging: decide the execution order of the tiles.

Tile parameters and Virtual Neuron

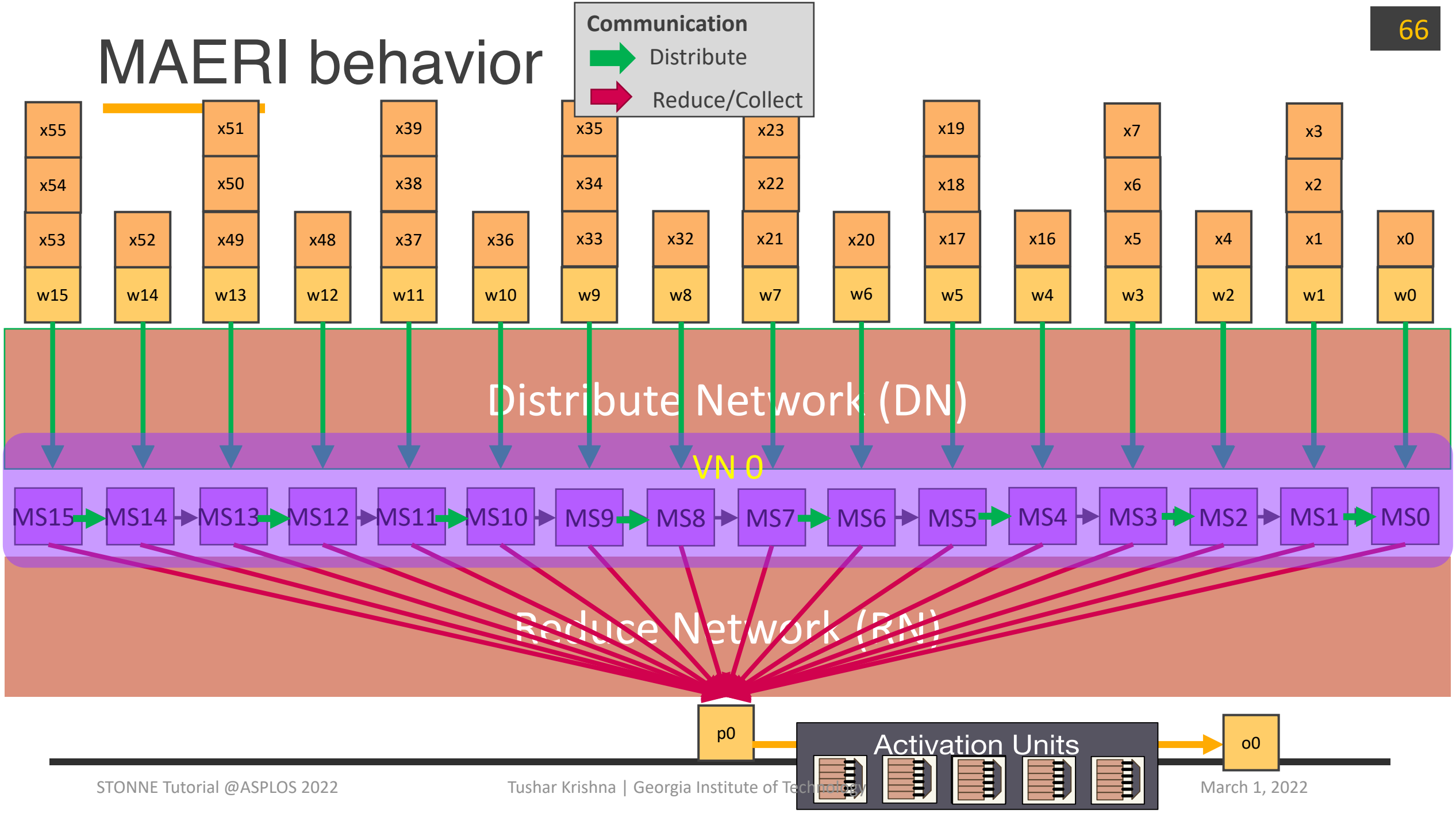
Symbol	Description
T_X	Number of mapped rows of inputs and weights
T_Y	Number of mapped columns of inputs and weights
T_K	Number of mapped filters
T_N	Number of mapped input feature maps
T_C	Number of mapped input and weight channel
T_X’	Number of mapped rows of output
T_Y’	Number of mapped columns of output

Virtual Neuron	Calculation
VN Size	T_X*T_Y*T_C
Number of VNs	T_K*T_N*T_X’*T_Y’

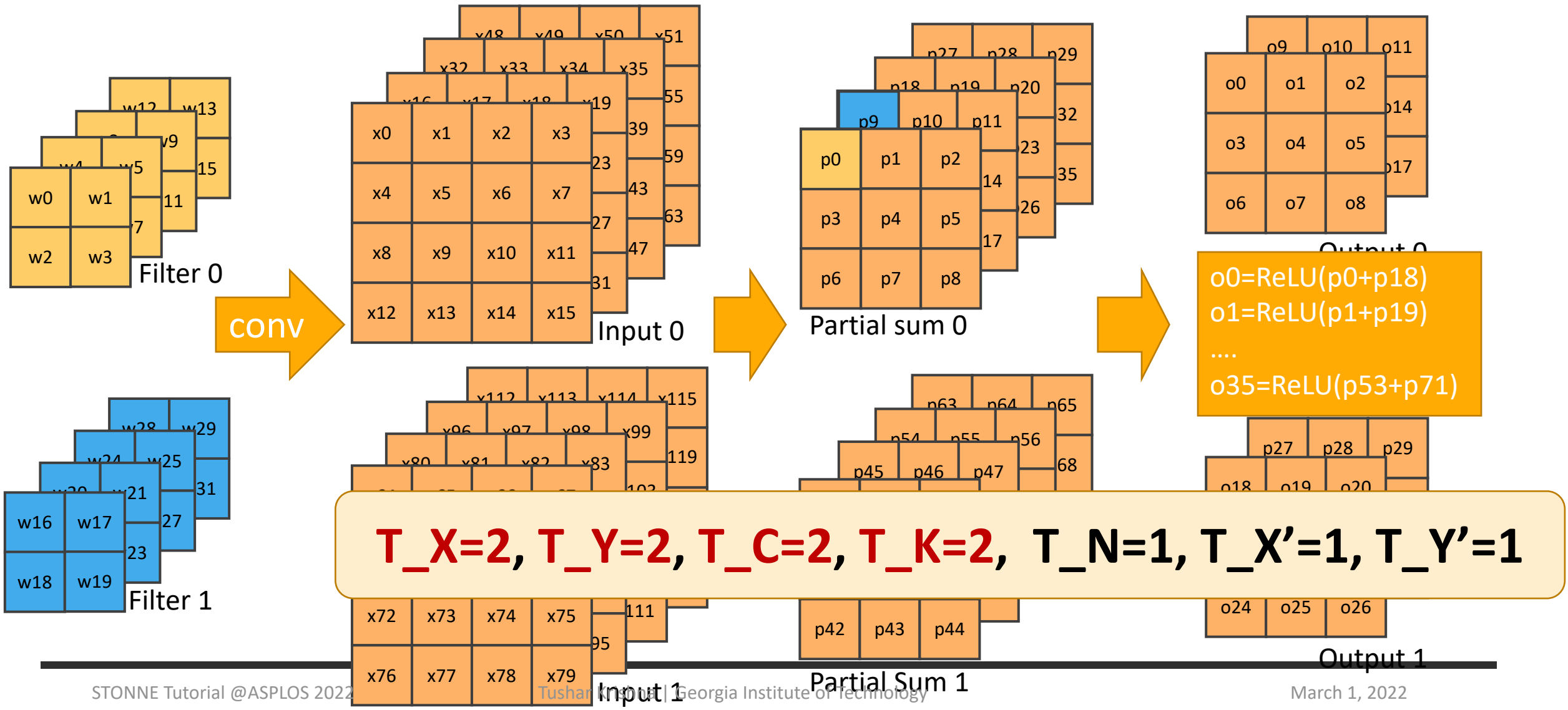
Mapping strategy 1 (16 MSes)



MAERI behavior

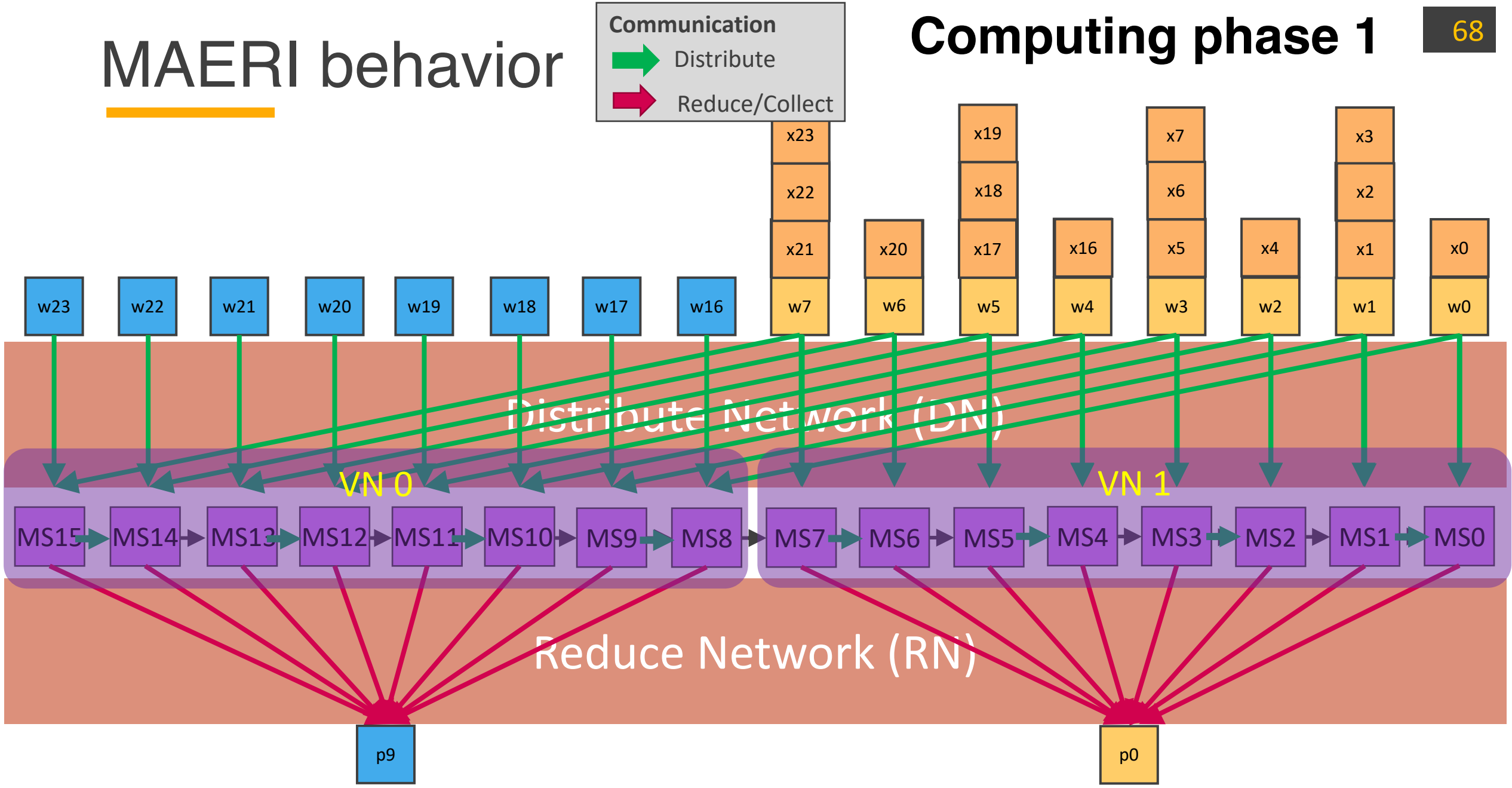


Mapping strategy 2



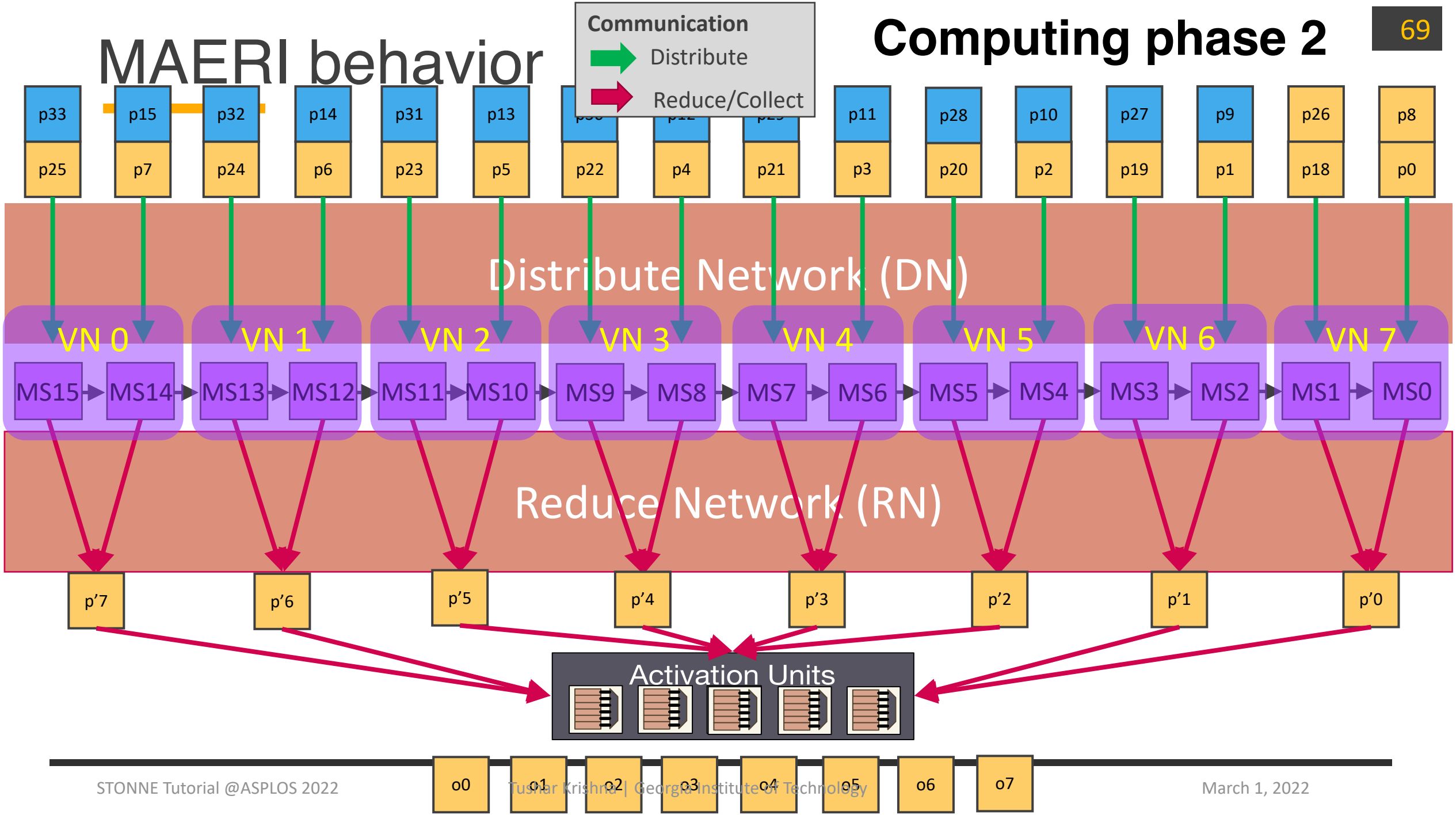
MAERI behavior

Computing phase 1



MAERI behavior

Computing phase 2



Outline

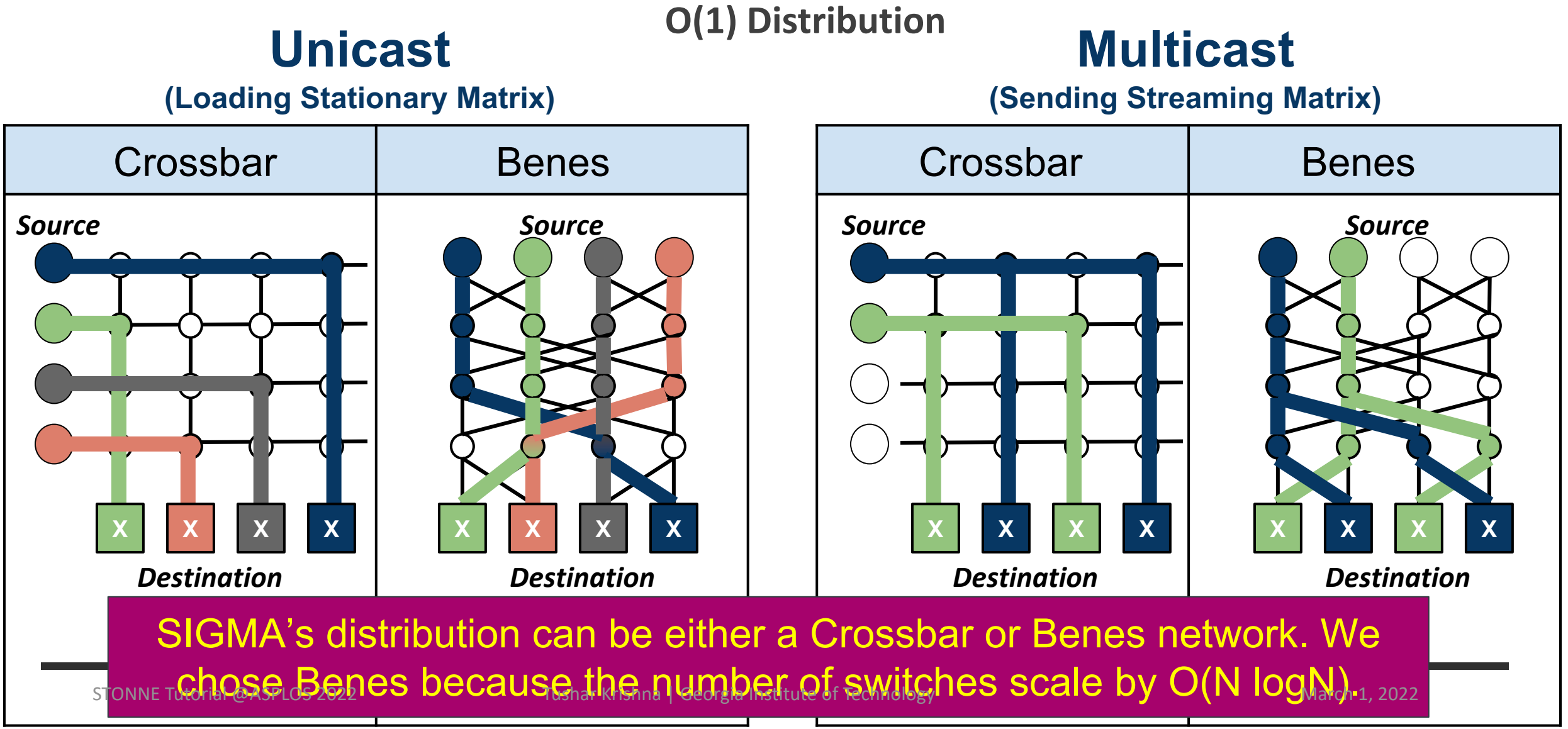
- DNN Accelerators
- Benefits of Mapping Flexibility
- Communication-centric Accelerators
- Case Study
  - MAERI
  - SIGMA
- Conclusion

Eric Qin, Ananda Samajdar, Hyoukjun Kwon, Vineet Nadella,,Dipankar Das, Sudarshan Srinivasan, Bharat Kaul, and Tushar Krishna

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

HPCA 2020, Best Paper Award

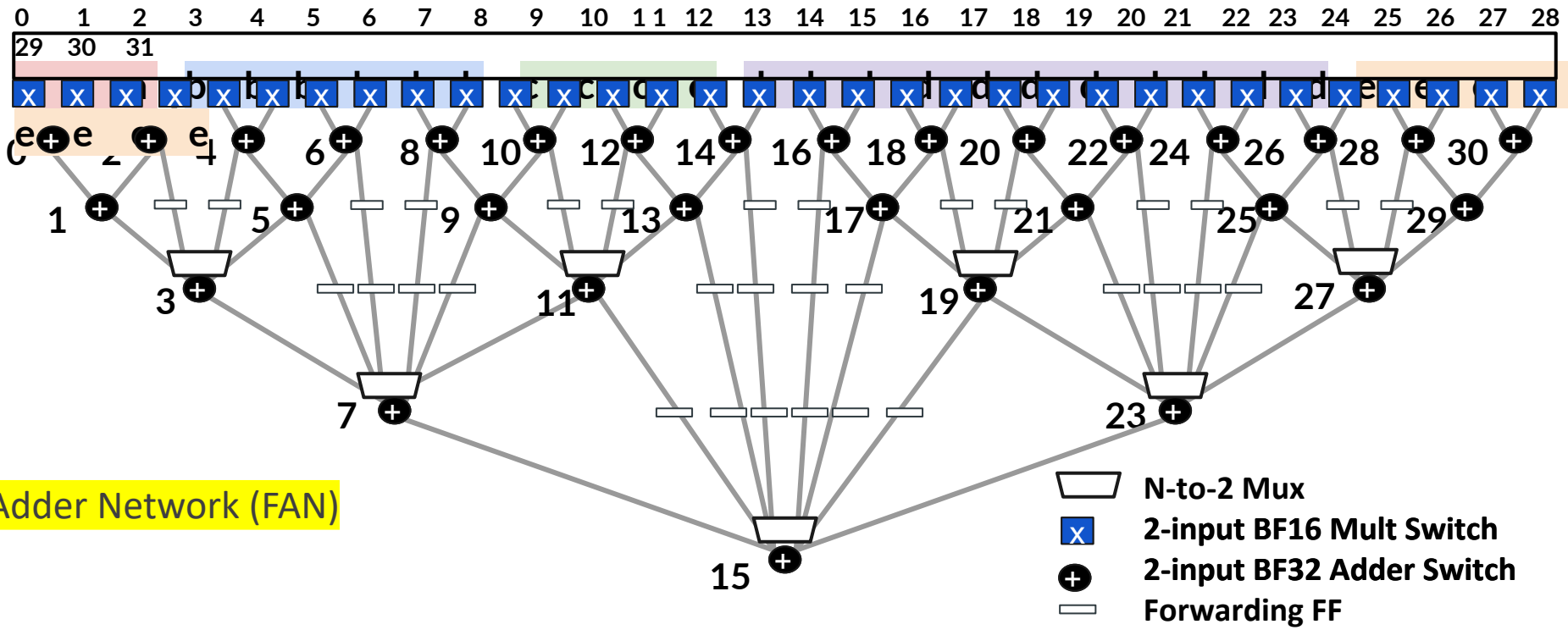
SIGMA O(1) Distribution Network



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

SIGMA O(logN) Reduction Network

Different clusters of partial sums.



Forwarding Adder Network (FAN)

Guarantees non-blocking reductions of arbitrary sizes

Outline

- DNN Accelerators
- Benefits of Mapping Flexibility
- Communication-centric Accelerators
- Case Study
- Conclusion

Conclusion

- **DNN models evolving rapidly**
  - Flexible Mapping support needed to future-proof DNN accelerators
- **Communication-centric Accelerator Design**
  - Rigid Accelerators (e.g., Systolic Arrays) limit mapping efficiency
  - Flexible distribution and collection networks within accelerators provide mapping flexibility
    - Ability to create dot-products of arbitrary and unequal sizes
      - Example: MAERI (ASPLOS 2018), SIGMA (HPCA 2020)
- **Acknowledgments**
  - Hyoukjun Kwon (now at Meta) – Developer of MAERI
  - Zhongyuan Zhao (now at Cornell) – Developer of mRNA
  - Eric Qin (Georgia Tech) – Developer of SIGMA

Agenda

Attention: Tutorial is being recorded

Time (CET)	Time (ET)	Topic	Presenter
14:00 – 14:40	8:00 – 8:40	**Flexible Accelerators**	Tushar Krishna
14:40 – 15:10	8:40 – 9:10	**Cycle accurate simulation and Overview of STONNE**	José Luis Abellán
15:10 – 16:10	9:10 – 10:10	**(Hands-on) STONNE Deep-Dive**	Francisco Muñoz-Martínez
16:10 – 16:40	10:10 – 10:40	**Coffee Break**	
16:40 – 17:10	10:40 – 11:10	**(Hands-on) STONNE Deep-Dive**	Francisco Muñoz-Martínez
17:10 – 17:40	11:10 – 11:40	**Dataflow exploration for Graph Neural Networks**	Raveesh Garg
17:50 – 18:00	11:50 – 12:00	**Roadmap for Future Development**	Manuel Acacio

Tutorial Website https://stonne-simulator.github.io/ASPLOSTUT.html
includes agenda and STONNE/OMEGA installation instructions