



A Systematic Methodology for Characterizing Scalability of DNN Accelerators using SCALE-Sim

Ananda Samajdar¹, Jan Moritz Joseph^{1,2}, Yuhao Zhu³, Paul Whatmough⁴, Matthew Mattina⁴, and Tushar Krishna¹

¹Georgia Tech
Atlanta, GA

²Otto-Von-Guericke Univ.
Magdeburg, Germany

³Univ. of Rochester
Rochester, NY

⁴ARM ML Research Lab
Boston, MA

Email: anandsamajdar@gatech.edu

Acknowledgments



Yuhao Zhu
Assistant Professor
Univ. of Rochester



Jan Moritz Joseph
Post Doc Researcher
Georgia Tech



Paul Whatmough
Principal Research Engineer
ARM Research, Boston

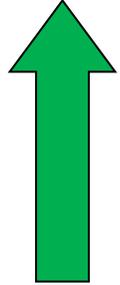


Matthew Mattina
Senior Director Research
ARM Research, Boston



Tushar Krishna
Assistant Professor
Georgia Tech

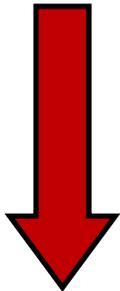
Feeding the computational beast



Number of applications using DNNs

Sizes of the models

- Megatron LM
- GPT2



Tolerance on latency

Energy efficiency

Benefits from device scaling

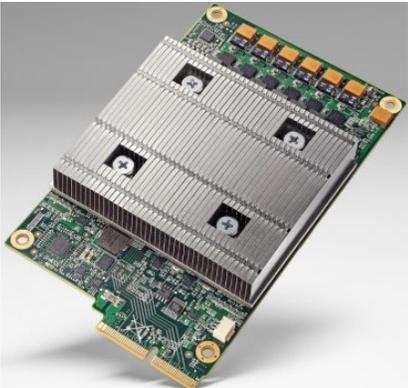
Need accelerators with higher performance and power efficiency



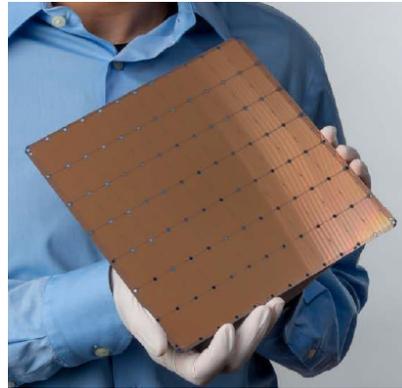
Designing the next accelerator

Easy part *Adding more compute power*

Scale UP: Make one big chip

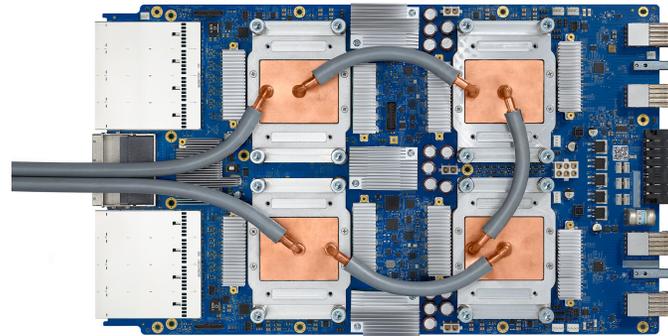


TPUv1

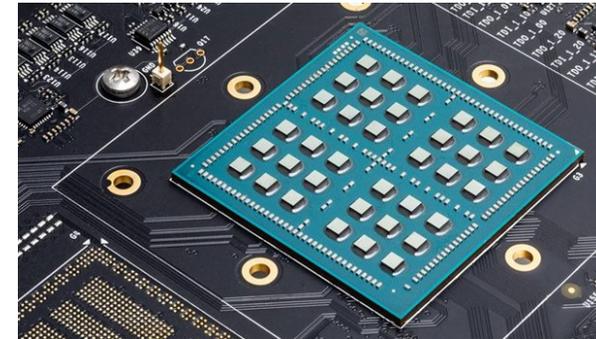


Cerebras WSP

Scale Out: Make a big collection of chips



TPU v3



Simba

Not so easy part

Attaining high utilization

Data orchestration

Energy efficiency

Which design choice is more beneficial ?

Outline

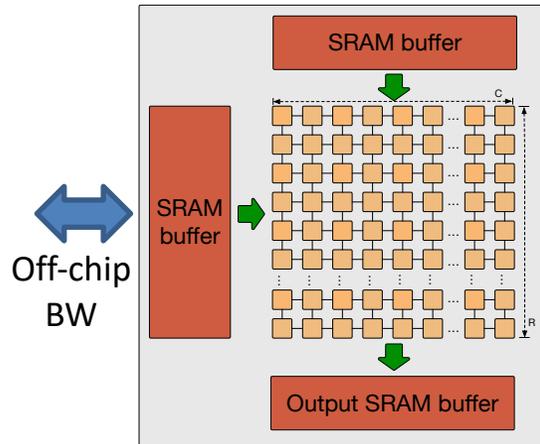
- Introduction
- Scale-up vs Scale-out
 - Target Systems
 - Analytical Modeling
 - Simulation
- Results
 - Performance
 - Energy

Outline

- Introduction
- Scale-up vs Scale-out
 - Target Systems
 - Analytical Modeling
 - Simulation
- Results
 - Performance
 - Energy

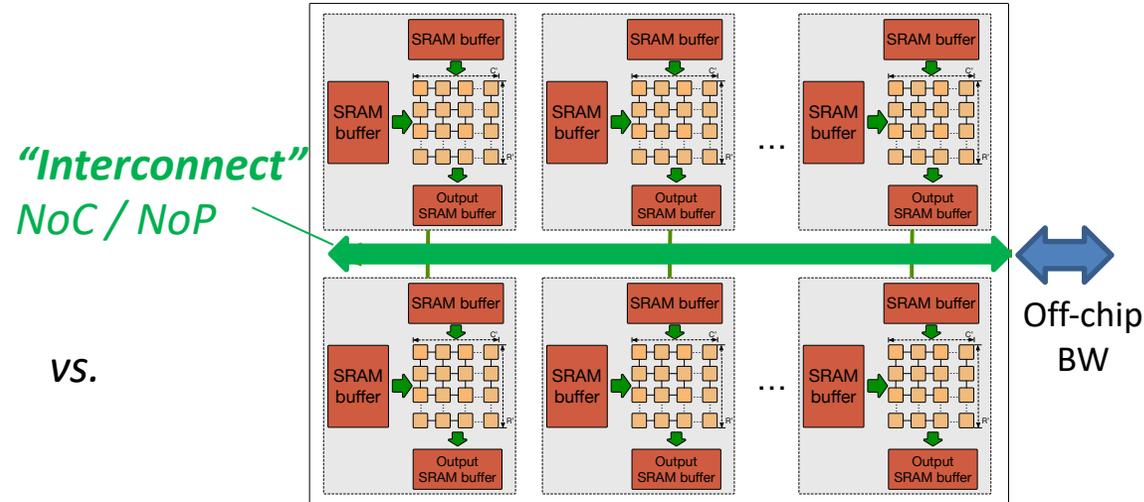
Target Systems

Monolithic array for Scale-up



variable row and cols to model different sized scale-up systems

Distributed arrays for Scale-out



variable number of homogeneous partitions + variable rows and cols within partitions, to model different sized scale-out systems

Both monolithic and distributed systems have equal

- Number of MAC units
- Total size of SRAM buffers

Questions of Interest

- Runtime
- Utilization
- Bandwidth Requirements

For analysis we developed,

- An *analytical model* for first order insights
- *SCALE-Sim** for cycle-accurate simulations

* <https://github.com/ARM-software/SCALE-Sim>

Outline

- Introduction
- Scale-up vs Scale-out
 - Target Systems
 - Analytical Modeling
 - Simulation
- Results
 - Performance
 - Energy

Analytical Modeling: Assumptions

- **Stall-free Compute Array**

- How?

- Systolic Array: Simple structures with deterministic behavior

- **Stall-free Interconnect**

- How?

- Non-blocking interconnect with sufficient bandwidth

- **Stall-free On-Chip Memory-system**

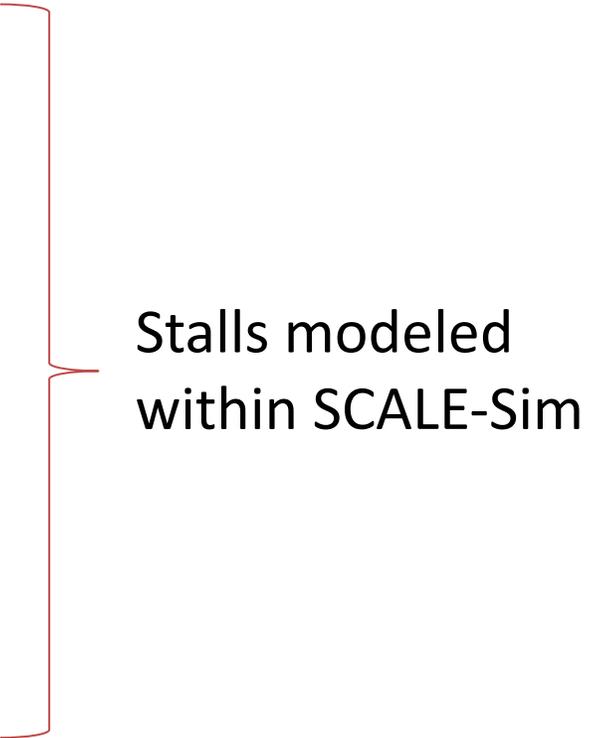
- How?

- Perfect Prefetch (hidden behind compute) via double-buffering
- No bank conflicts

- **Stall-free Off-chip Memory System**

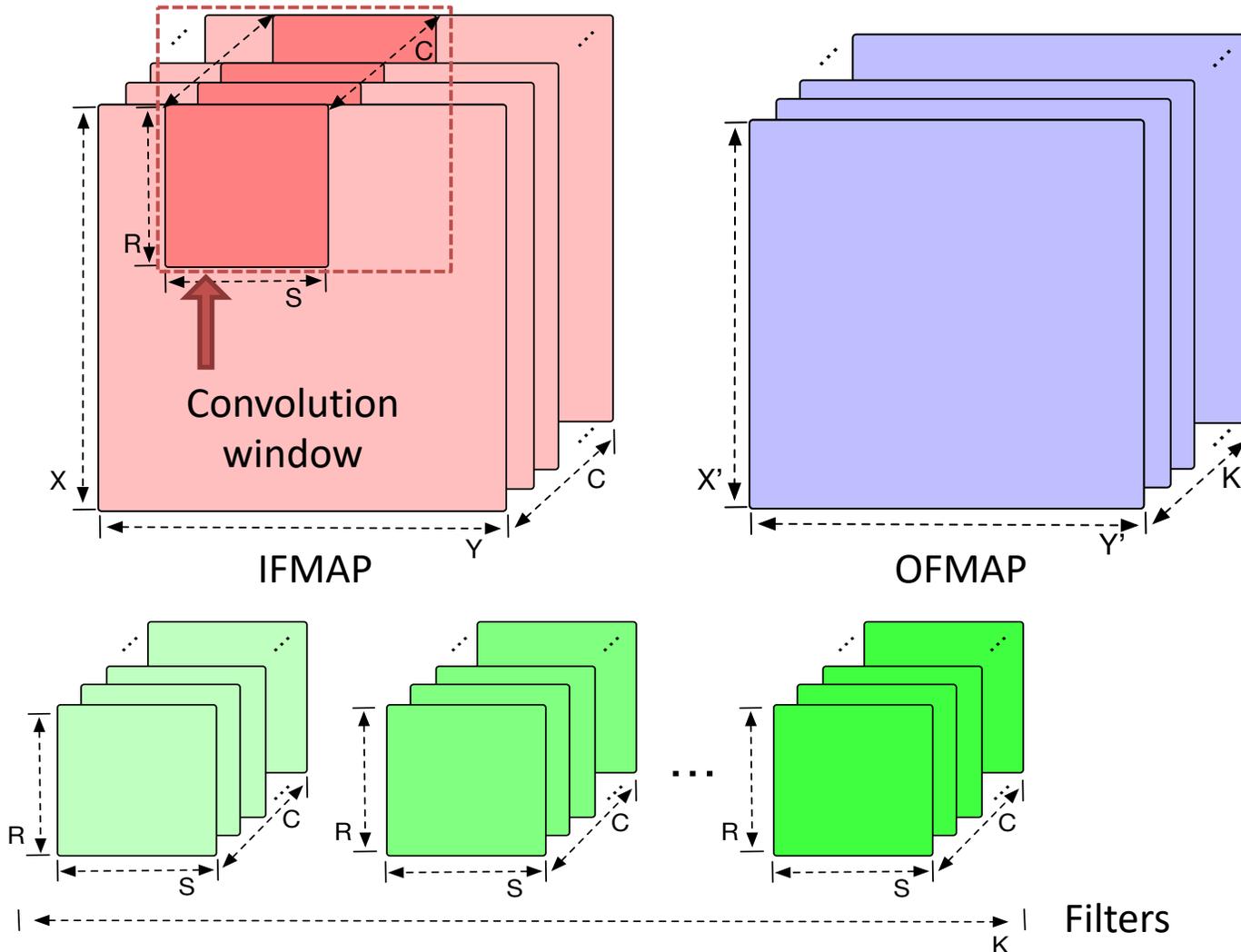
- How?

- Sufficient Memory Bandwidth for target DNN



Stalls modeled
within SCALE-Sim

Analytical Modeling: Terminology



Convolution Window:

- IFMAP pixels used to generate one output pixel ($=R*S*C$)
- **Number of convolution window =** Number of OFMAP pixel generated by one filter ($=X'*Y'$)

OFMAP channels:

- Number of 2D output matrices created
- Equal to number of filters ($= K$)

OFMAP Pixel per channel:

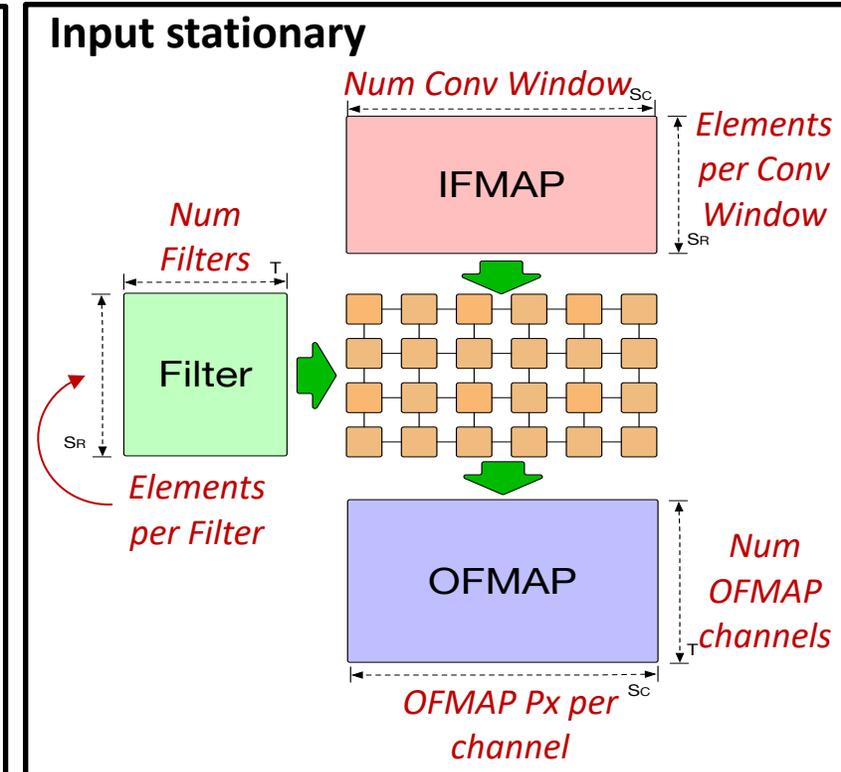
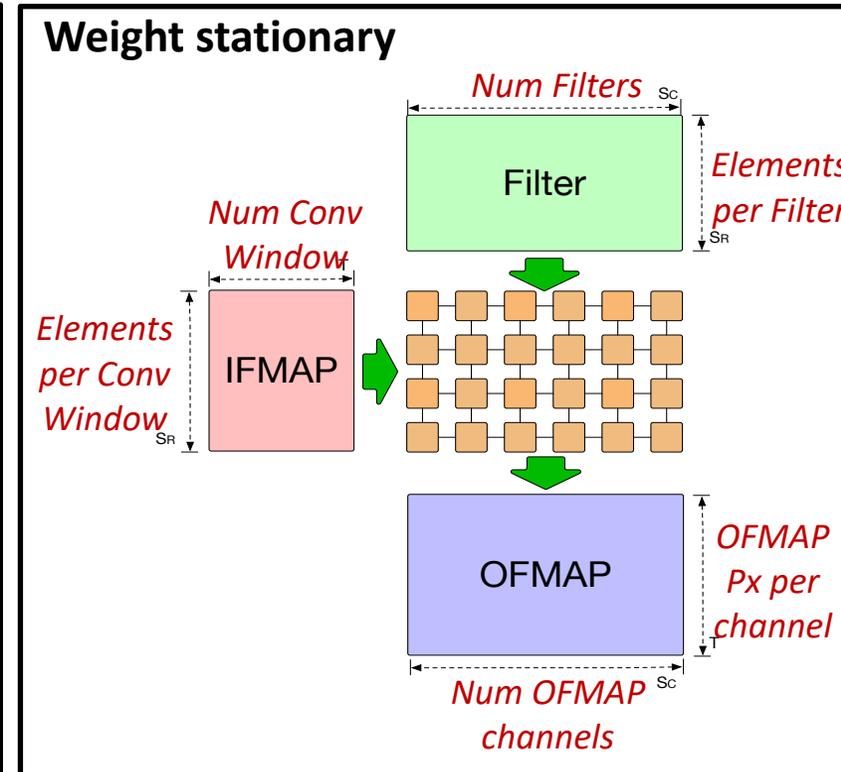
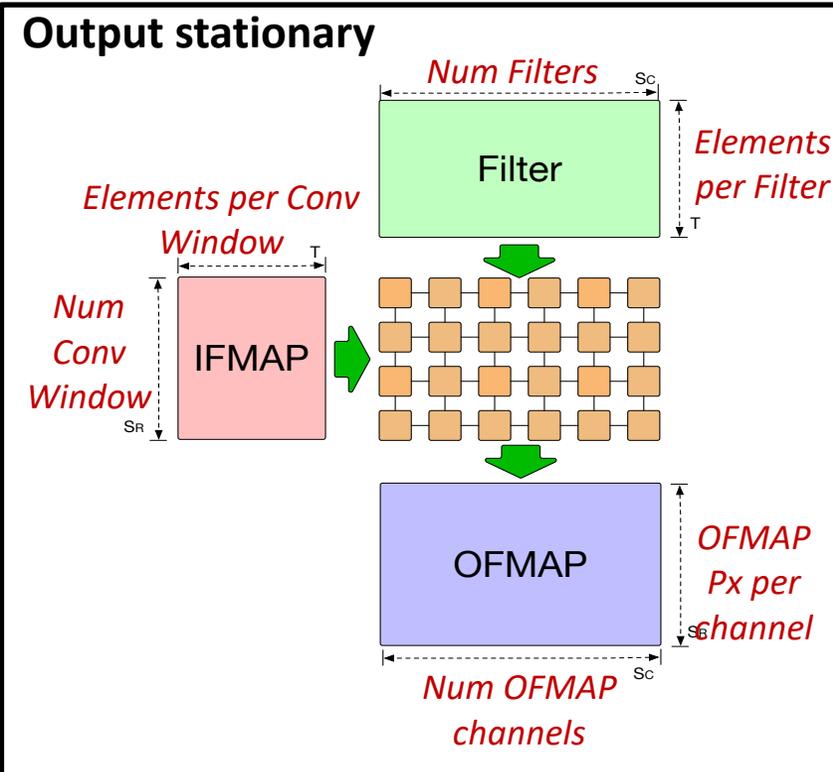
- Number of pixel generated by one filter ($=X'*Y'$)

Modeling Systolic Arrays

S_R : Spatial mapping dimension along rows

S_C : Spatial mapping dimension along cols

T : Temporal mapping dimension



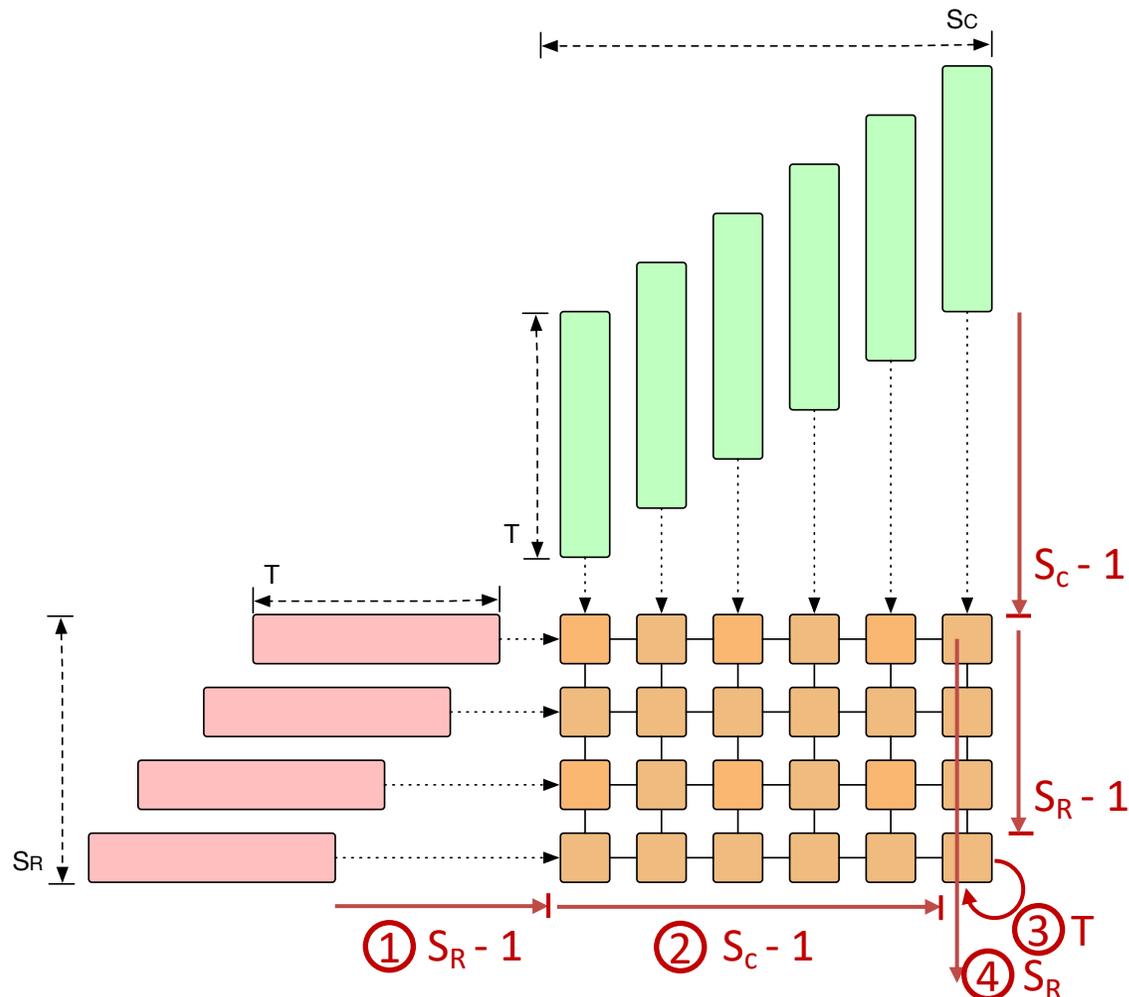
	Spatial (S_R)	Spatial (S_C)	Temporal (T)
OS	OFMAP Px per channel	Num Filter	Elem per Conv Window
WS	Elem per Conv Window	Num Filter	OFMAP Px per channel
IS	Elem per Conv Window	OFMAP Px per channel	Num Filter

Modeling Output Stationary

S_R : Spatial mapping dimension along rows

S_C : Spatial mapping dimension along cols

T : Temporal mapping dimension



Considering ideal case: Entire computation fits in the array

MAC at the right edge corner finishes last

- ① Time to account for skew
- ② Time to fill all the cols before first computation starts
- ③ Computation time
- ④ Time taken to eject the last output

Runtime

$$\tau = S_R - 1 + S_C - 1 + T + S_R$$

$$\text{Or, } \tau = 2S_R + S_C + T - 2$$

	Spatial (S_R)	Spatial (S_C)	Temporal (T)
OS	OFMAP Px per channel	Num Filter	Elem per Conv Window

Modeling Weight Stationary

S_R : Spatial mapping dimension along rows

S_C : Spatial mapping dimension along cols

T : Temporal mapping dimension

Considering ideal case: Entire computation fits in the array

Right most row is the critical path

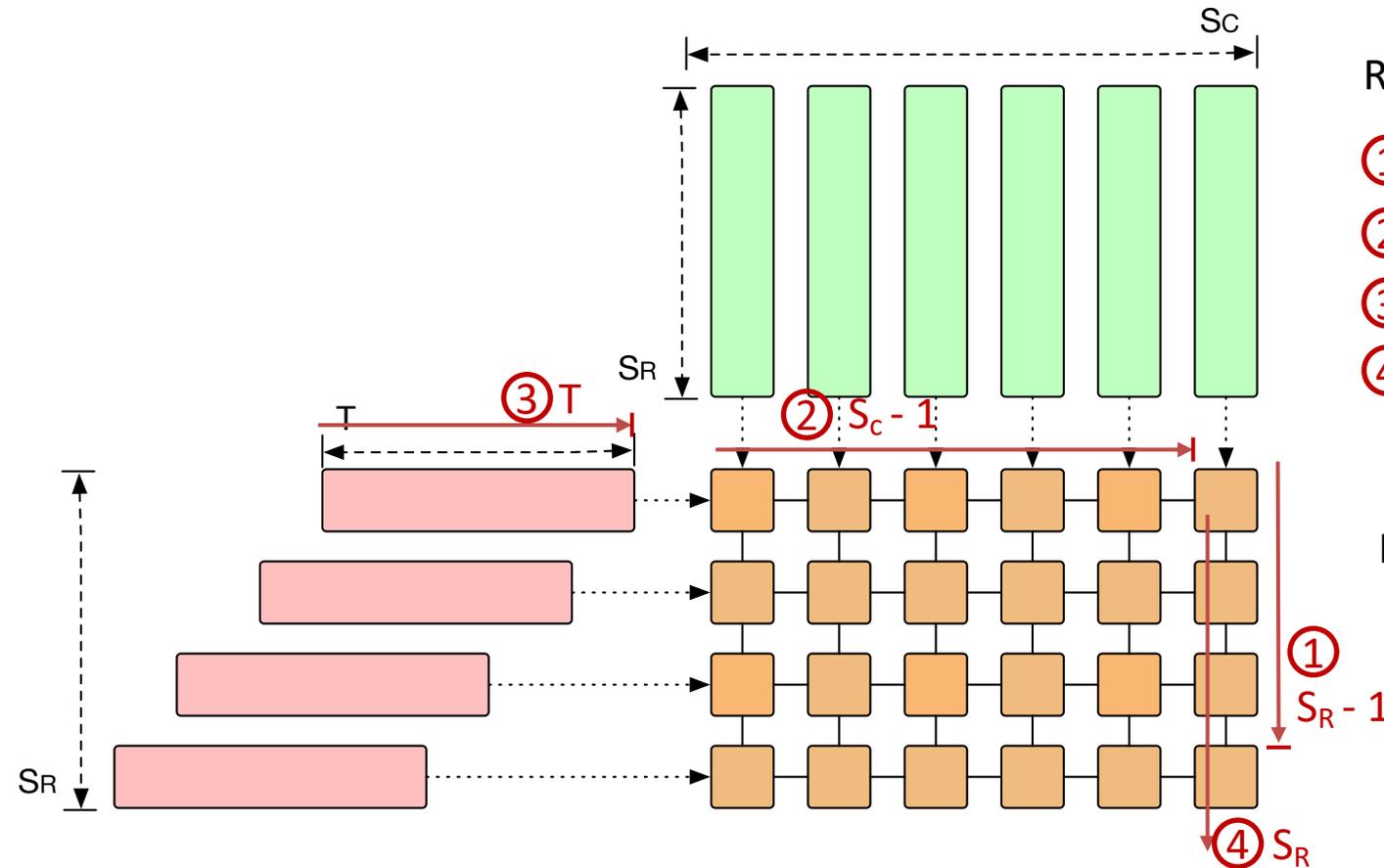
- ① Prefill time
- ② Time for IFMAP operand to reach the row
- ③ Time taken to stream inputs for all conv windows
- ④ Time taken for last reduction and ejection

Runtime

$$\mathbf{\tau} = S_R - 1 + S_C - 1 + T + S_R$$

Or, $\mathbf{\tau} = 2S_R + S_C + T - 2$

	Spatial (S_R)	Spatial (S_C)	Temporal (T)
WS	Elem per Conv Window	Num Filter	OFMAP Px per channel



Modeling Performance for Serialization

S_R : Spatial mapping dimension along rows

S_C : Spatial mapping dimension along cols

T : Temporal mapping dimension

Relaxing the ideal assumption

In reality, mapping mismatch will lead to serialization

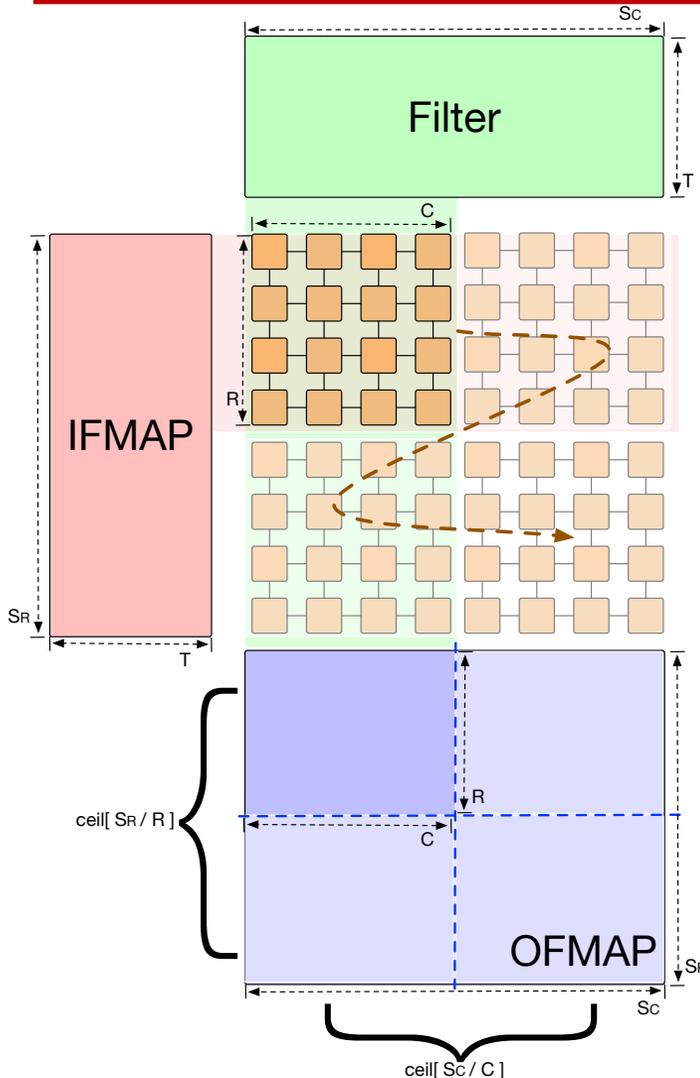
Number of serial steps (*fold*s)

= Folds in row dimension x Fold in col dimension

$$= \lceil S_R / R \rceil \times \lceil S_C / C \rceil$$

Runtime, τ = Runtime for one-fold * Number of folds

$$= (2R + C + T - 2) \times \lceil S_R / R \rceil \times \lceil S_C / C \rceil$$

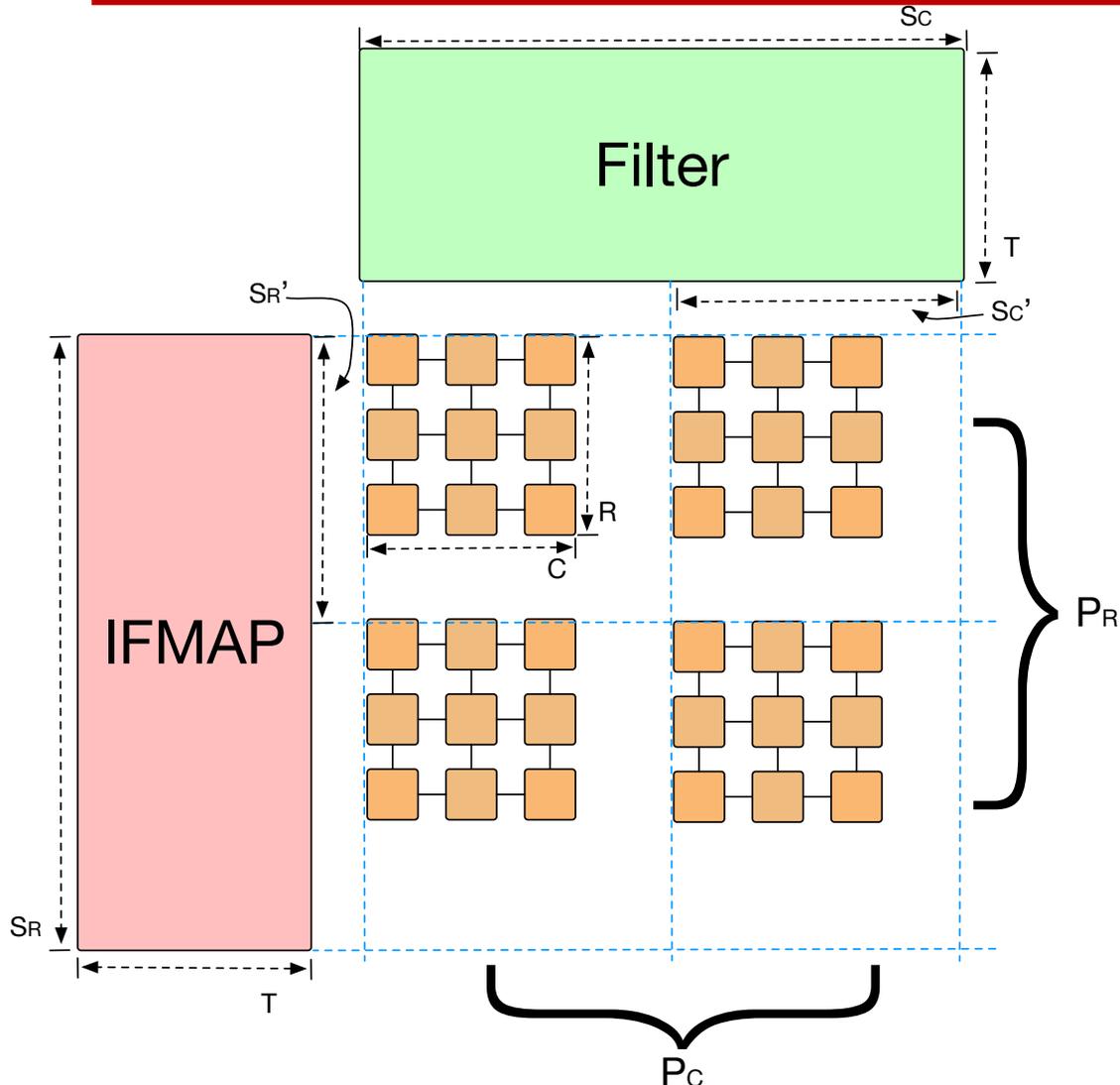


Performance in a Distributed Setting

S_R : Spatial mapping dimension along rows

S_C : Spatial mapping dimension along cols

T : Temporal mapping dimension



The distributed (scale-out) accelerator unit is arranged as a $P_R \times P_C$ spatial grid

Each unit works on a part of the problem in parallel, and therefore need only parts of the operand

Thus,

$$\text{Runtime, } \boldsymbol{\tau} = \text{Runtime of one unit} \\ = (2R + C + T - 2) \times \lceil S_R' / R \rceil \times \lceil S_C' / C \rceil$$

Where,

$$S_R' = \lceil S_R / P_R \rceil \quad \text{and} \quad S_C' = \lceil S_C / P_C \rceil$$

Outline

- Introduction
- **Scale-up vs Scale-out**
 - Target Systems
 - Analytical Modelling
 - **Simulation**
- Results
 - Performance
 - Energy

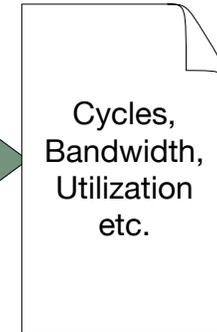
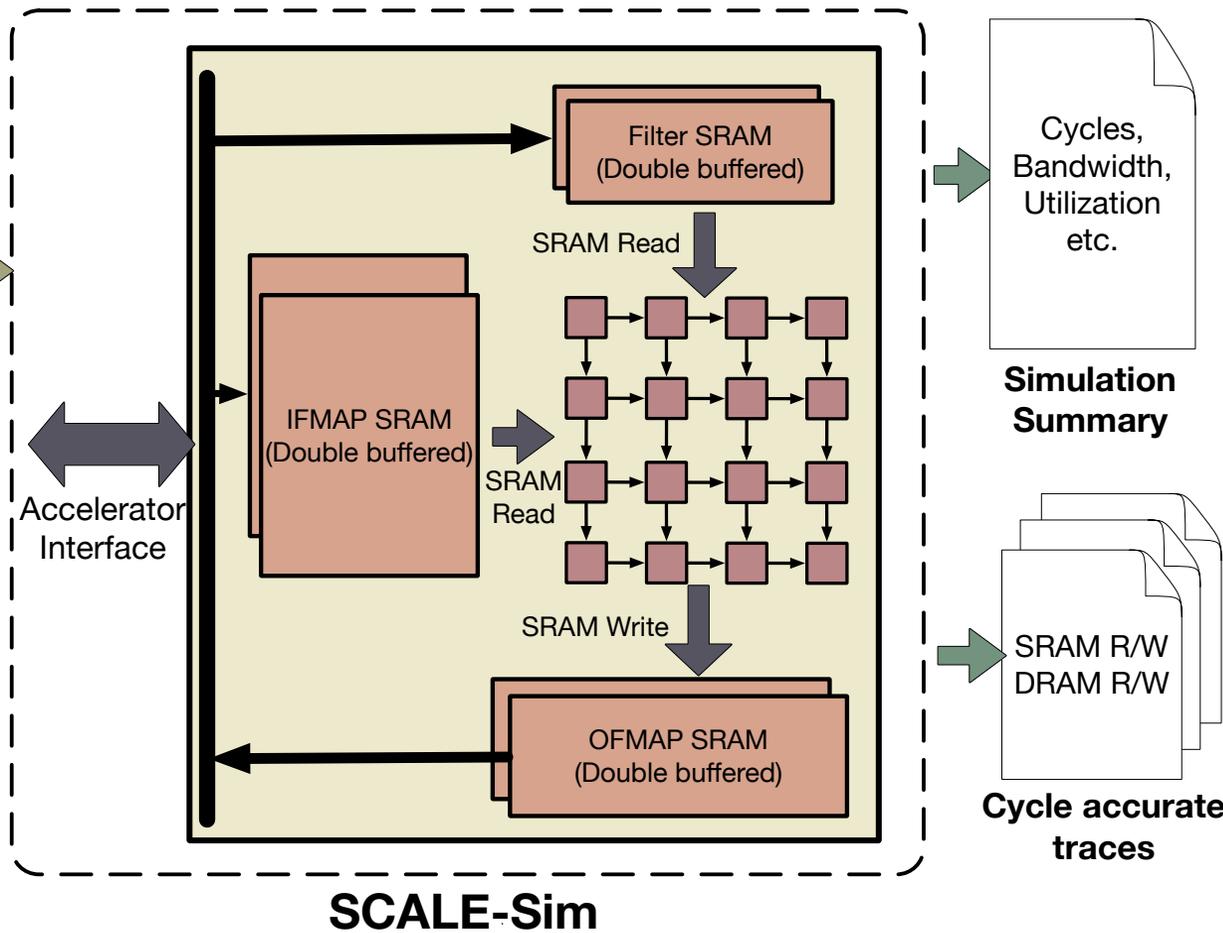
Simulations using SCALE-Sim

Parameter	Value
Array Height	32
Array Width	32
IFMAP SRAM	1024
Filter SRAM	1024
OFRAM SRAM	128
Dataflow	WS

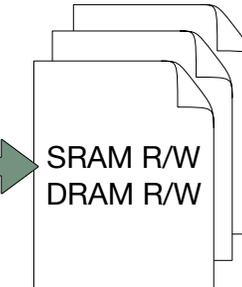
Config file



DNN Topology file



Simulation Summary



Cycle accurate traces

Cycle accurate systolic array based accelerator simulator

Inputs

1. Architecture configuration

- Array dimensions
- Buffer sizes
- Dataflow (OS, WS, IS)

2. Workload hyper parameters

SCALE sim generates,

1. Runtime in cycles
2. Cycle accurate memory traces
3. Interface bandwidth requirements

[BW required = Total reads / cycles taken]

Codebase: <https://github.com/ARM-software/SCALE-Sim>

Inputs

Config

Parameter	Description
ArrayHeight	Number of rows of the MAC systolic array
ArrayWidth	Number of columns of the MAC systolic array
IfmapSRAMsz	Size of the working set SRAM for IFMAP in KBytes
FilterSRAMsz	Size of the working set SRAM for filters in KBytes
OfmapSRAMsz	Size of the working set SRAM for OFMAP in KBytes
IfmapOffset	Offset to the generated addresses for IFMAP px
FilterOffset	Offset to the generated addresses for filter px
OfmapOffset	Offset to the generated addresses for OFMAP px
DataFlow	Dataflow for this run. Legal values are 'os', 'ws', and 'is'
Topology	Path to the topology file

Topology

Parameter	Description
Layer Name	User defined tag
IFMAP Height	Dimension of IFMAP matrix
IFMAP Width	Dimension of IFMAP matrix
Filter Height	Dimension of one Filter matrix
Filter Width	Dimension of one Filter matrix
Channels	Number of Input channels
Num Filter	Number of Filter matrices. This is also the number of OFMAP channels
Strides	Strides in convolution

Outputs

```
(mlhw) anand@airchitect:~/repos/scale_sim/outputs$ tree
.
├── isca_tutorial
│   └── layer_wise
│       ├── yolo_tiny_example_Conv1_dram_filter_read.csv
│       ├── yolo_tiny_example_Conv1_dram_ifmap_read.csv
│       ├── yolo_tiny_example_Conv1_dram_ofmap_write.csv
│       ├── yolo_tiny_example_Conv1_sram_read.csv
│       └── yolo_tiny_example_Conv1_sram_write.csv
├── yolo_tiny_example_avg_bw.csv
├── yolo_tiny_example_cycles.csv
├── yolo_tiny_example_detail.csv
└── yolo_tiny_example_max_bw.csv
2 directories, 9 files
```

Layer-wise cycle accurate traces

- Format

Cycle_num, <addr1>, <addr2>, ... , <addrN>

- SRAM traces reflect the addresses of the data read /produced at any cycle

- Assumes stall free operation

- DRAM traces are generated such that there is no SRAM miss

- Bandwidth demand is generated by parsing the DRAM trace

Summarized metrics

Codebase: <https://github.com/ARM-software/SCALE-Sim>

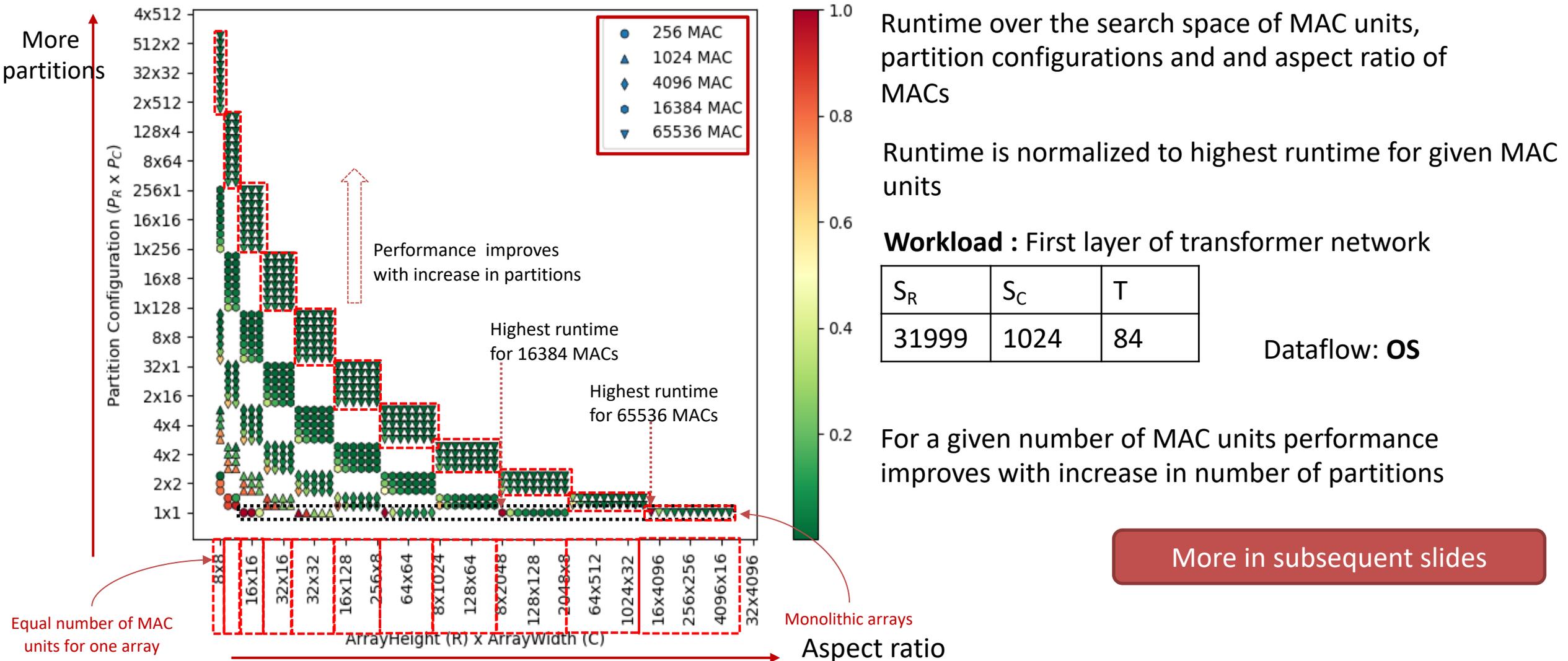
Target Workloads

Name	S_R	T	S_C
GNMT0	128	4096	2048
GNMT1	320	4096	3072
GNMT2	1632	1024	36548
GNMT3	2048	32	4096
DB0	1024	50000	16
DB1	35	2560	4096
TF0	31999	84	1024
TF1	84	4096	1024
NCF0	2048	128	1
NCF1	256	2048	256

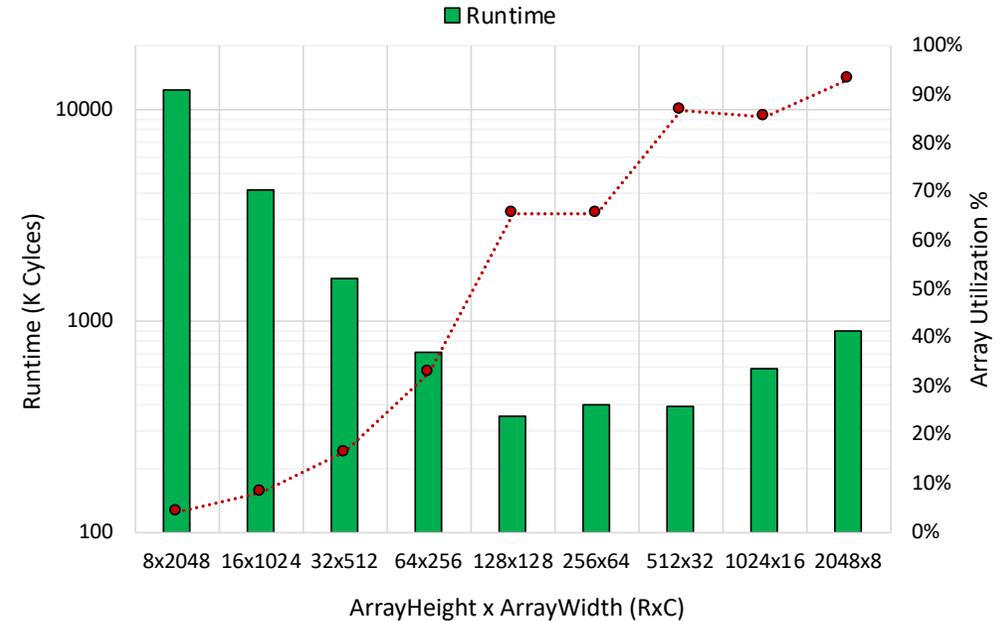
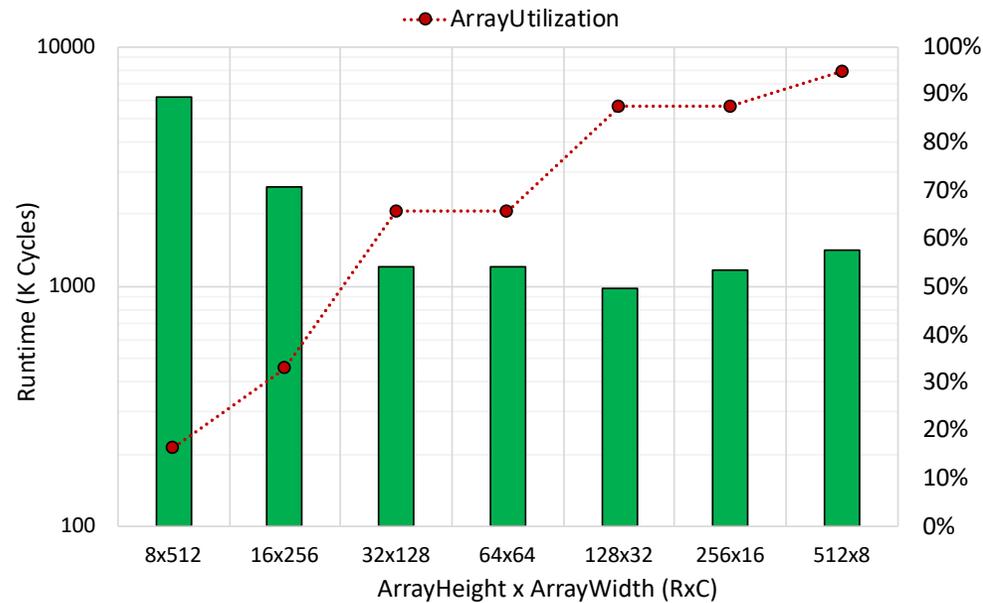
Outline

- Introduction
- Systolic arrays
 - Analytical Modelling
 - Simulation
- Results
 - Performance
 - Energy

Performance



Performance for Scale-up (Monolithic) Configuration



Workload : First layer of transformer network

Observations:

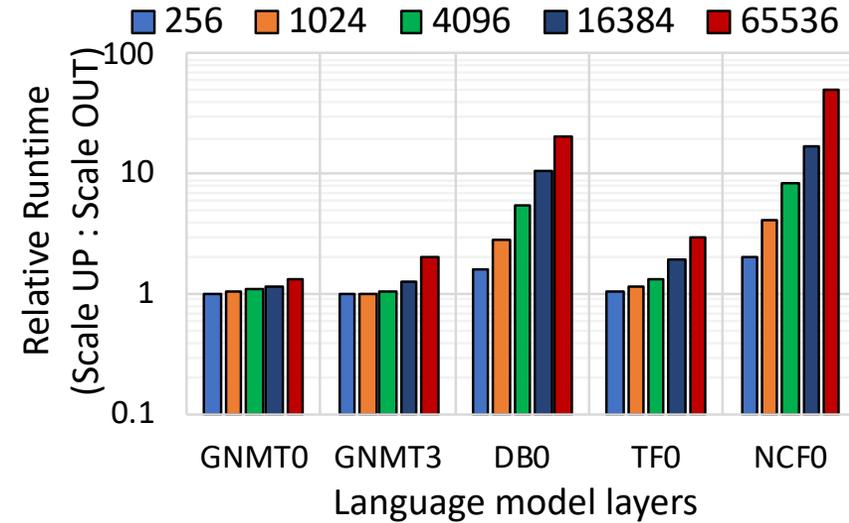
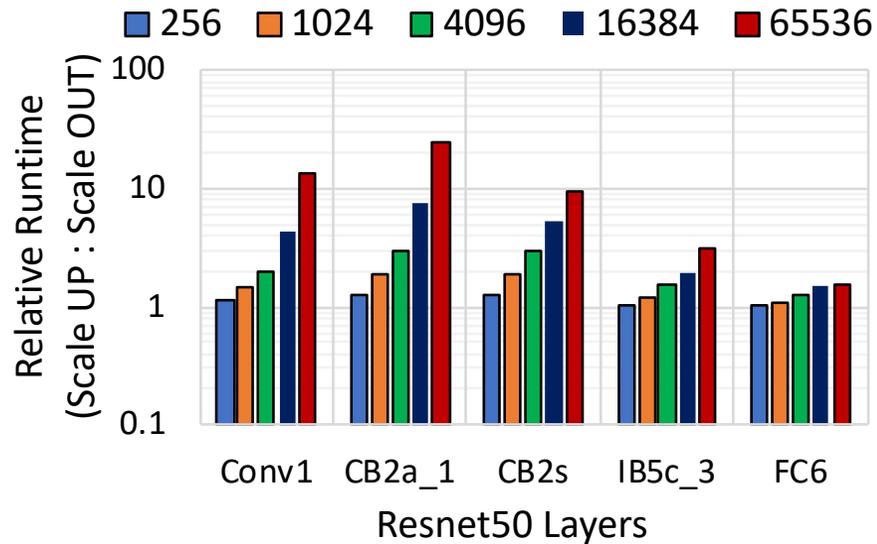
1. Utilization is not the sole contributor to performance
2. Best aspect ratio is dependent on workload and the number of MAC units

$$\tau = (2R + C + T - 2) * \lceil S_R / R \rceil * \lceil S_C / C \rceil$$

Irregular array size
increase runtime per fold

Low utilization increases
number of folds

Performance: Monolithic vs Distributed configurations



Ratio of lowest runtime obtained for Scaled-up and Scaled-out configuration

$$S_R' = [S_R / P_R]$$

$$S_C' = [S_C / P_C]$$

Workload : Selected layers from Resnet-50 and language model networks

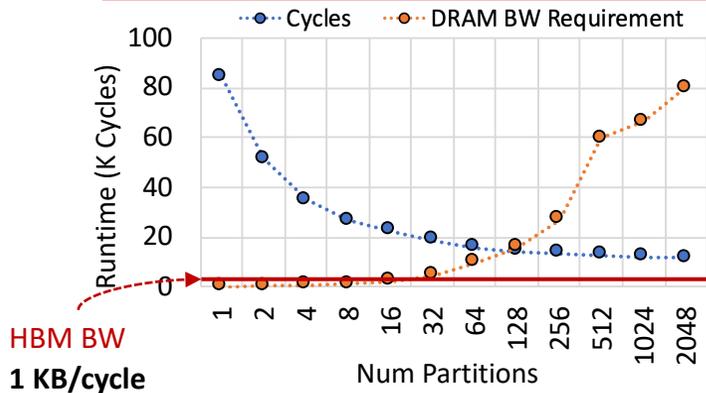
Observation: Scaling out always has the lower runtime than scaled up monolithic array

$$\tau = (2R + C + T - 2) \times [S_R' / R] * [S_C' / C]$$

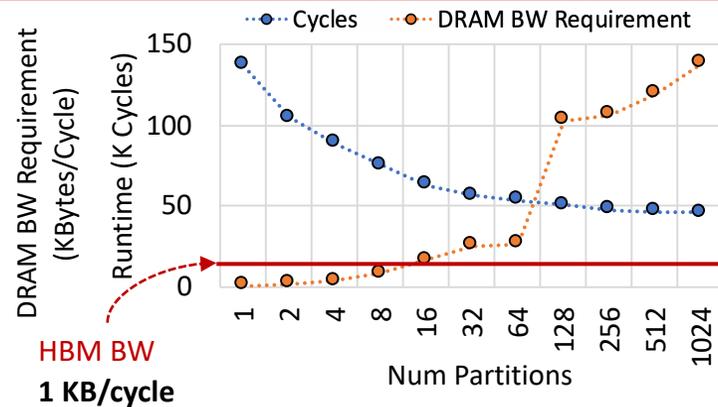
Always smaller for scaled-out

Better utilization in scaled-out configuration reduces the number of folds

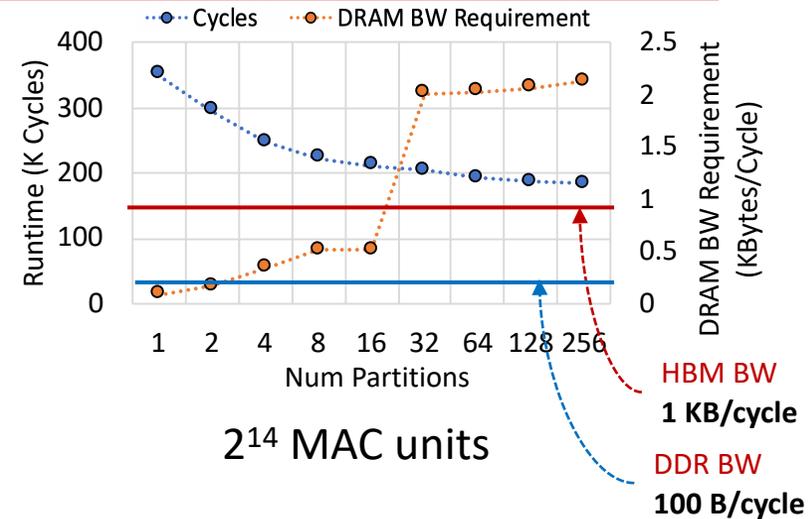
The Bandwidth Bottleneck



2^{18} MAC units



2^{16} MAC units



2^{14} MAC units

Workload: First layer in transformer network

Observation: External bandwidth requirements are prohibitive to performance

Reason: Loss of reuse

1. **Spatial reuse:** Lost due to cutting off the wires
2. **Temporal reuse:** Lost due to replication of operands

Possible mitigation

1. Allocate more on chip buffer to win back temporal reuse
2. Wait for new memory technology

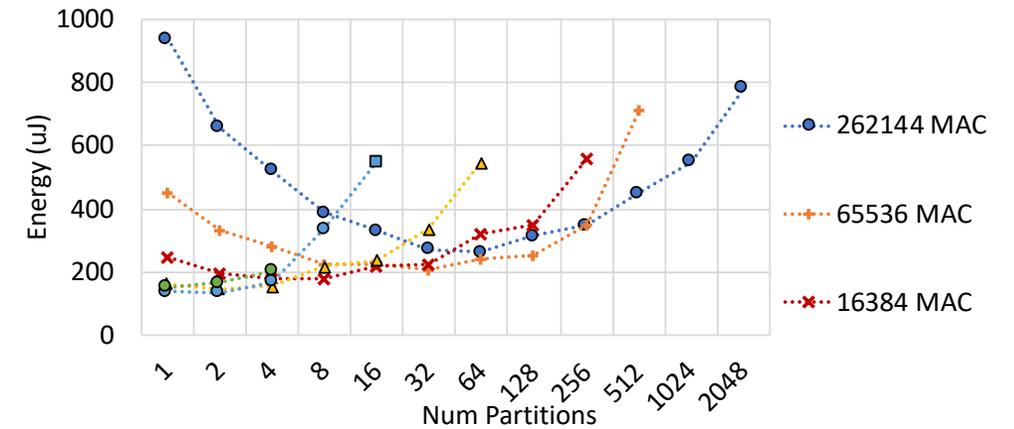
Finding the sweet spot for performance is determined by the implementation budget

Energy Efficiency

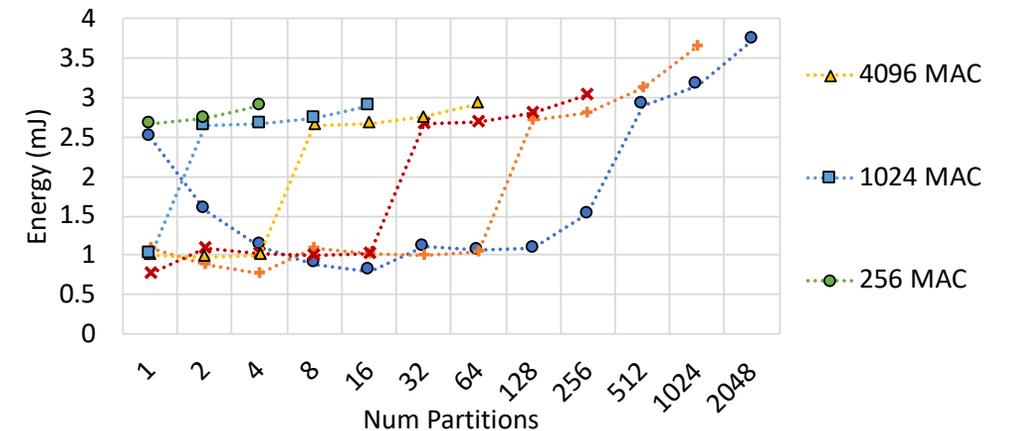
$$\begin{aligned} \text{Energy} = & \text{Energy per MAC} * \text{Num MAC units} * \text{Cycles taken} \\ & + \text{Energy per SRAM access} * \text{Num SRAM accesses} \\ & + \text{Energy per DRAM access} * \text{Num DRAM accesses} \end{aligned}$$

Observations

1. Unlike performance, there is a definitive sweet spot
2. Higher the number of MAC units, the sweet spot lies in the configuration with finer grained distribution
3. For smaller array sizes monolithic configurations are more energy efficient



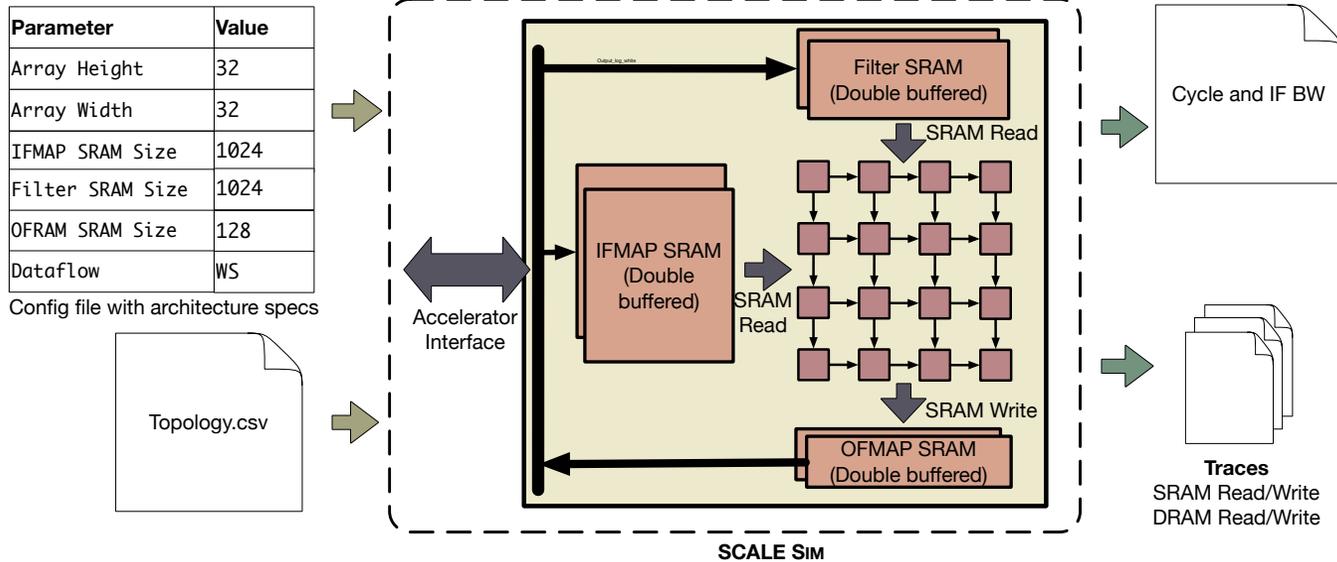
Representative layer from Resnet50



Representative layer from Transformer

Summary

Thank you!



Scaling DNN accelerators efficiently is non intuitive even for simple designs like a systolic array

We propose an **analytical model** and **simulation infrastructure (SCALE-Sim)** to help make scaling decisions

Our analysis depict that we can find out sweet spots for energy efficient and performant configurations

Feel free to try SCALE Sim:

<https://github.com/ARM-software/SCALE-Sim>

ASTRA-SIM uses SCALE-Sim internally!

- Talk in this session

Questions? Send me an email:
anandsamajdar@gatech.edu