CoSA: Scheduling by Constrained Optimization for Spatial Accelerators

Qijing Jenny Huang*, Minwoo Kang, Grace Dinh, Thomas Norell, Aravind Kalaiah†, James Demmel, John Wawrzynek, Yakun Sophia Shao

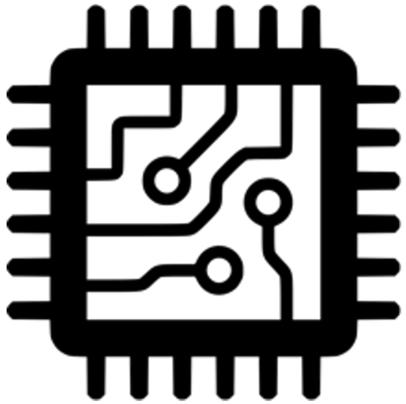
*NVIDIA, UC Berkeley, †Meta

jennyhuang@nvidia.com

Berkeley UNIVERSITY OF CALIFORNIA

Scheduling is required everywhere

Scheduling



- Algorithm

algorithmic states
to be run

- Hardware

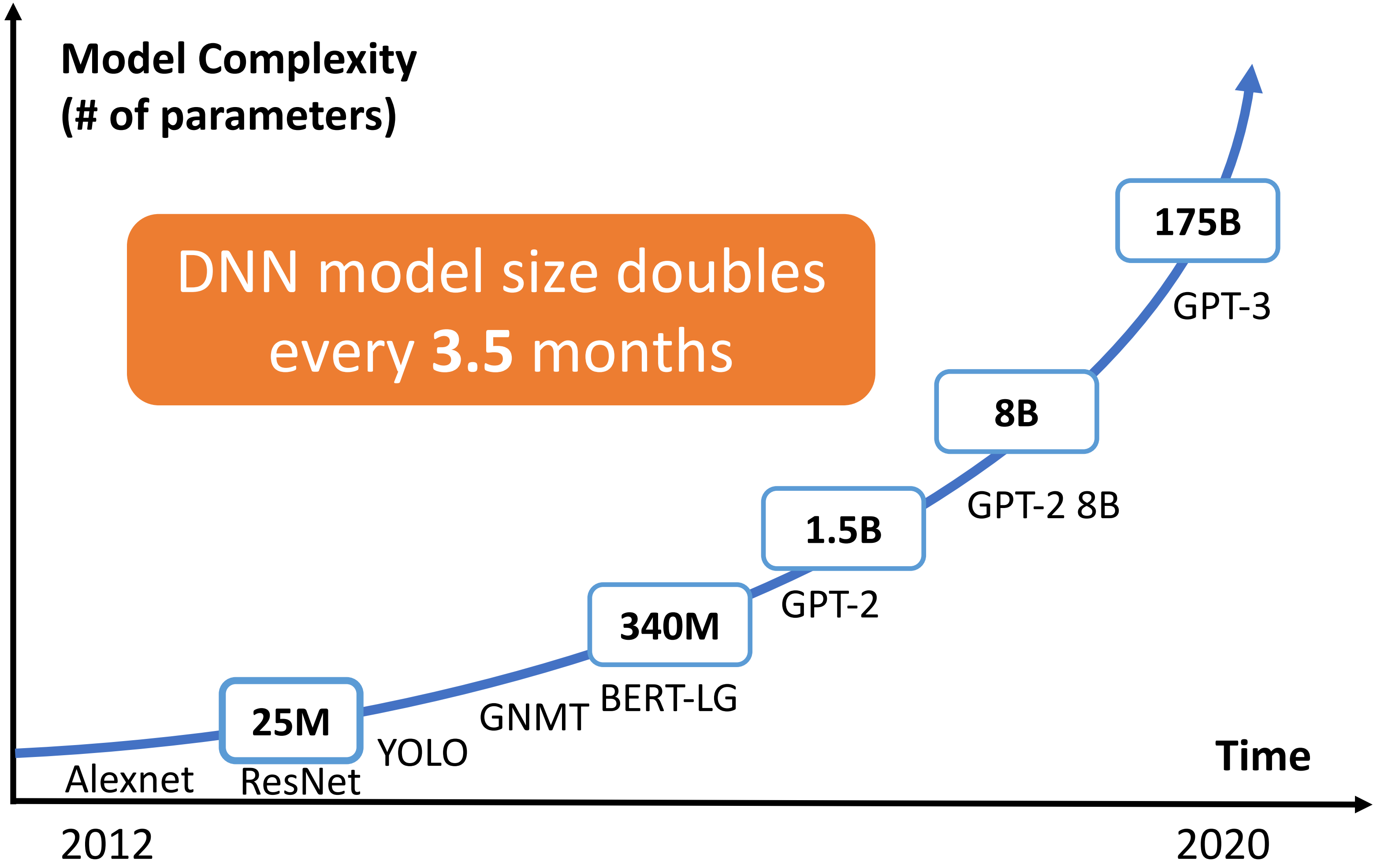
hardware resources
to be allocated

Scheduling is a big challenge

- Algorithm

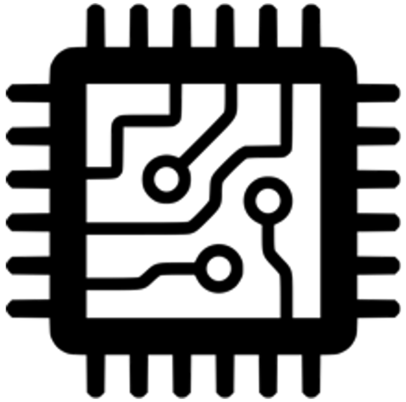
1. Exponentially growing algorithm complexity

Exponentially growing algorithm complexity



* source from Intel AI

Scheduling is a big challenge

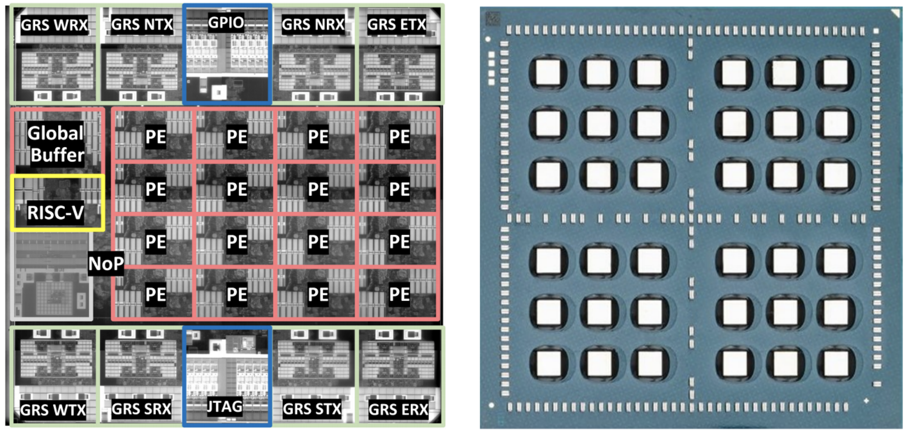


- Algorithm
- Hardware

1. Exponentially growing algorithm complexity
2. Rapidly increasing hardware capacity

Rapidly increasing hardware capacity

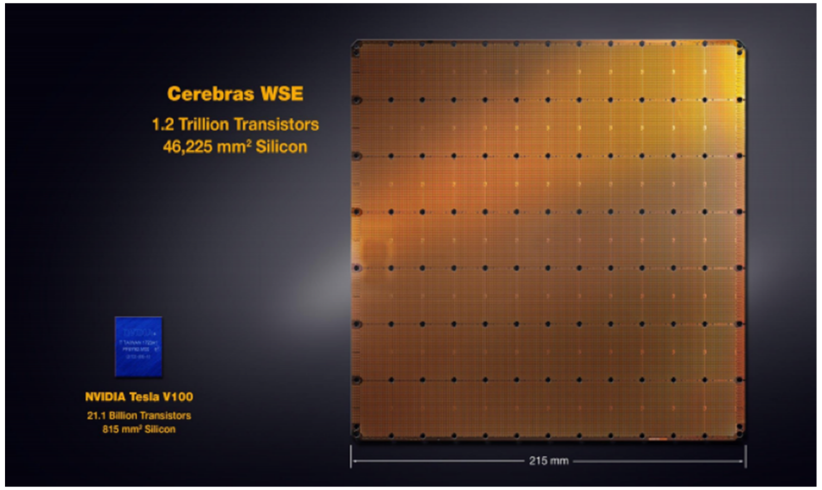
NoC/NoP Chip



(a) Simba chiplet (b) Simba package

Simba[1]
16PEs x 36 Chiplets

Wafer-scale Chip



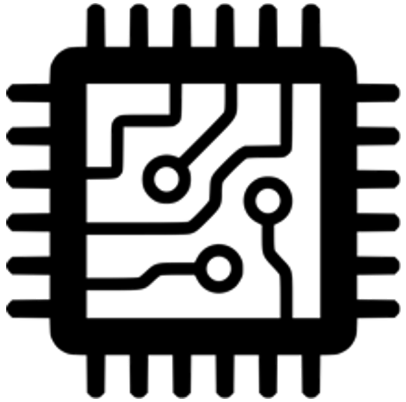
Cerebras[2]
84 Interconnected Chips

[1] Shao, Yakun Sophia, and et al. "Simba: Scaling Deep-Learning Inference with Multi-Chip-Module-Based Architecture." 2019 MICRO.
[2] "Wafer-Scale Deep Learning", https://cerebras.net/blog/wafer-scale-deep-learning-hot-chips-2019-presentation/

Scheduling is a big challenge

Intractable scheduling space

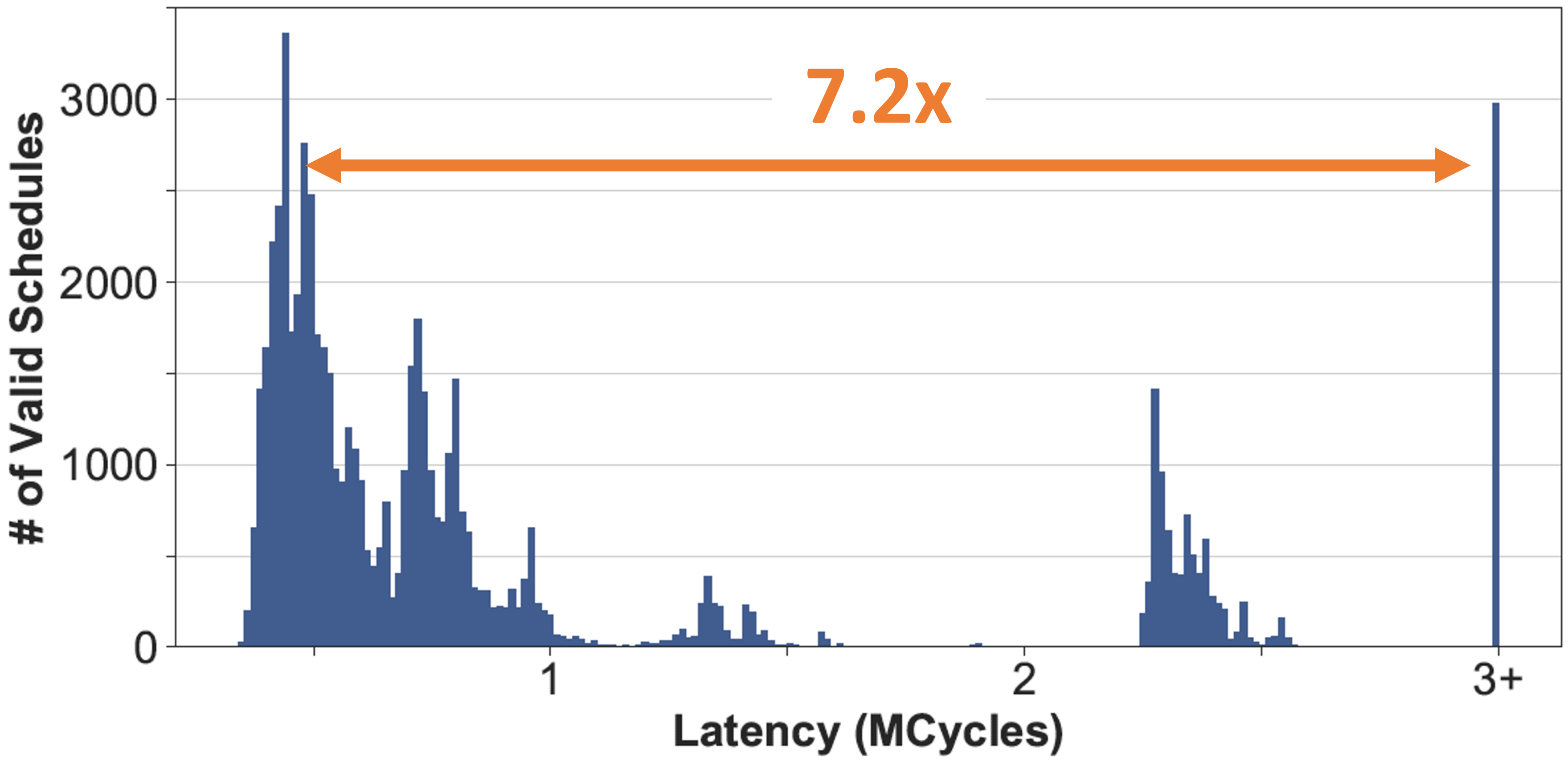
Scheduling



- Algorithm
- Hardware

1. Exponentially growing algorithm complexity
2. Rapidly increasing hardware capacity

Scheduling significantly affects performance



State-of-the-art DNN accelerator schedulers

Brute-force
Timeloop
dMazeRunner Triton
Interstellar Marvel

Feedback-based
AutoTVM Halide
FlexFlow Gamma
MindMapping

Constrained Optimization
Polly+Pluto TC
Tiramisu
CoSA

- Costly
- Sample invalid space
- Hard to generalize

One-shot solution

- Unable to determine tiling factor sizes

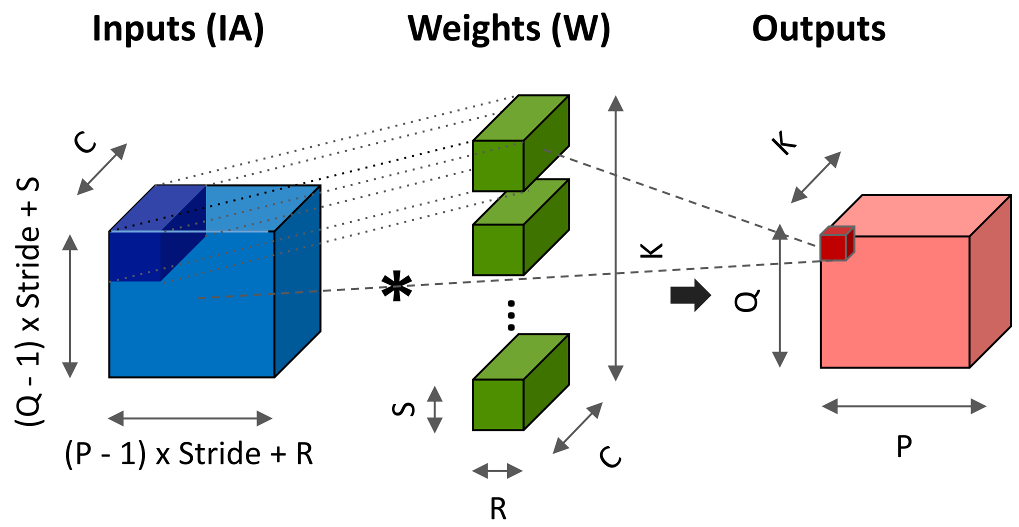
Opportunities

Workload Regularity

Hardware Regularity

Explicit Data Movement

Target Workload



R, S: weight width and height
P, Q: output width and height
C: input channel size
K: output channel size
N: batch size

DNN Layer :

```
for n in [0:N)
    for k in [0:K)
        for c in [0:C)
            for p in [0:P)
                for q in [0:Q)
                    for r in [0:R)
                        for s in [0:S)
OA[n,p,q,k] +=
    IA[n,p+r-(R-1)/2,q+s-(S-1)/2,c]
          × W[r,s,c,k]
```

Target Architecture

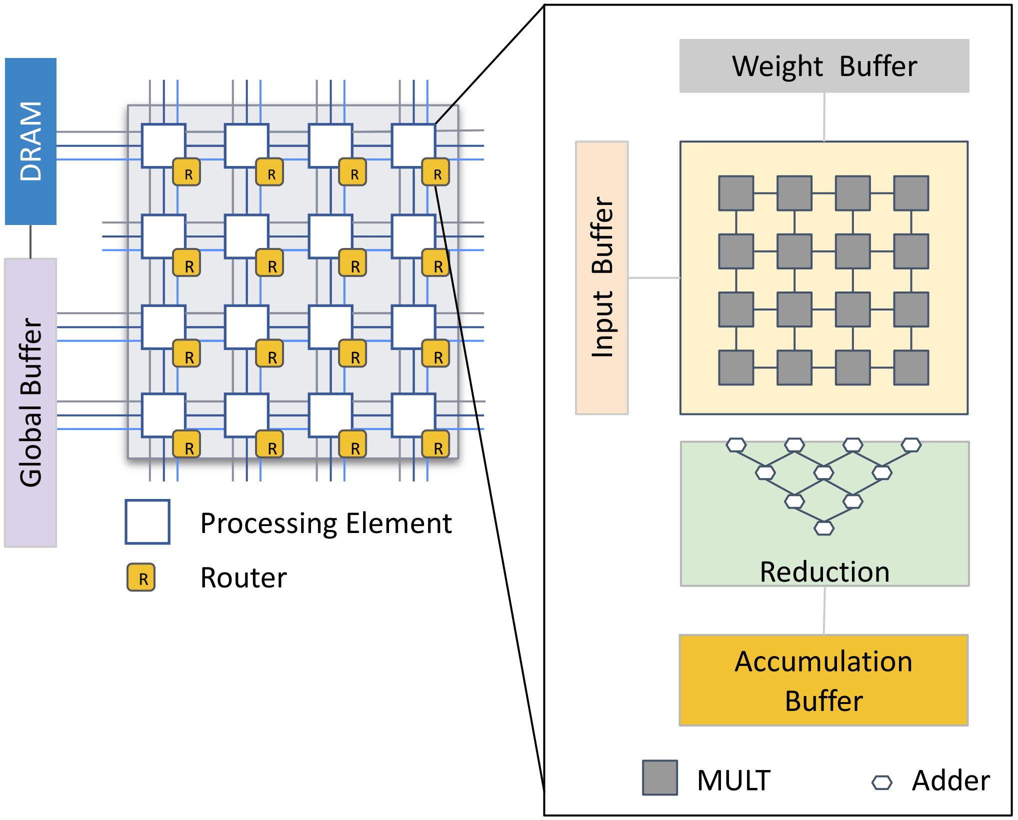
- Spatial PEs
- Multi-level Memory Hierarchy

64 MAC

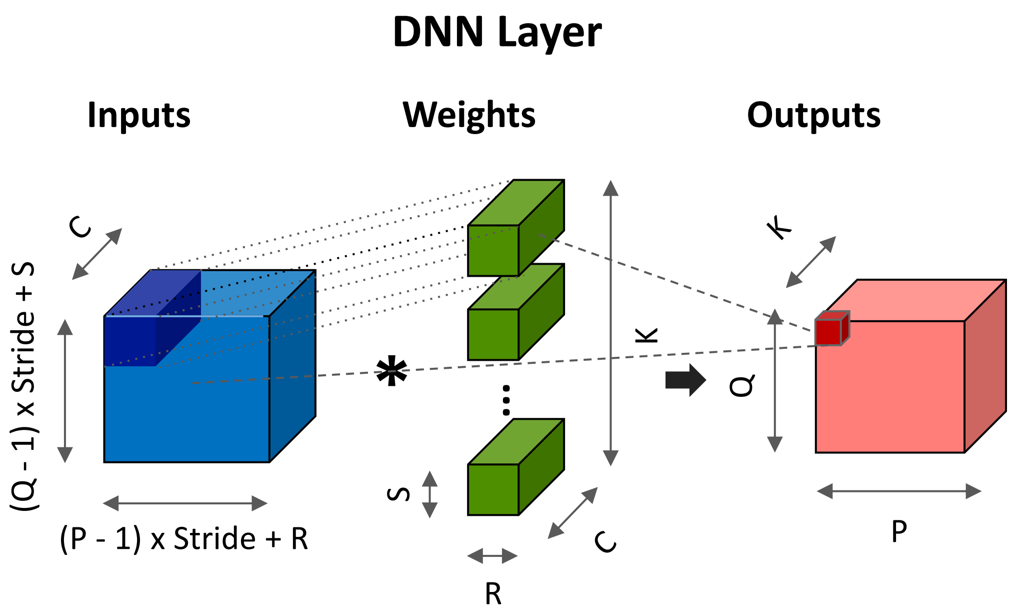
Registers	AccumBuffer
8x8	1x8
entries: 1	entries: 128
size: 64B	size: 384B

WeightBuffer	InputBuffer
1x8	1x1
entries: 4096	entries: 8192
size: 4KB	size: 8KB

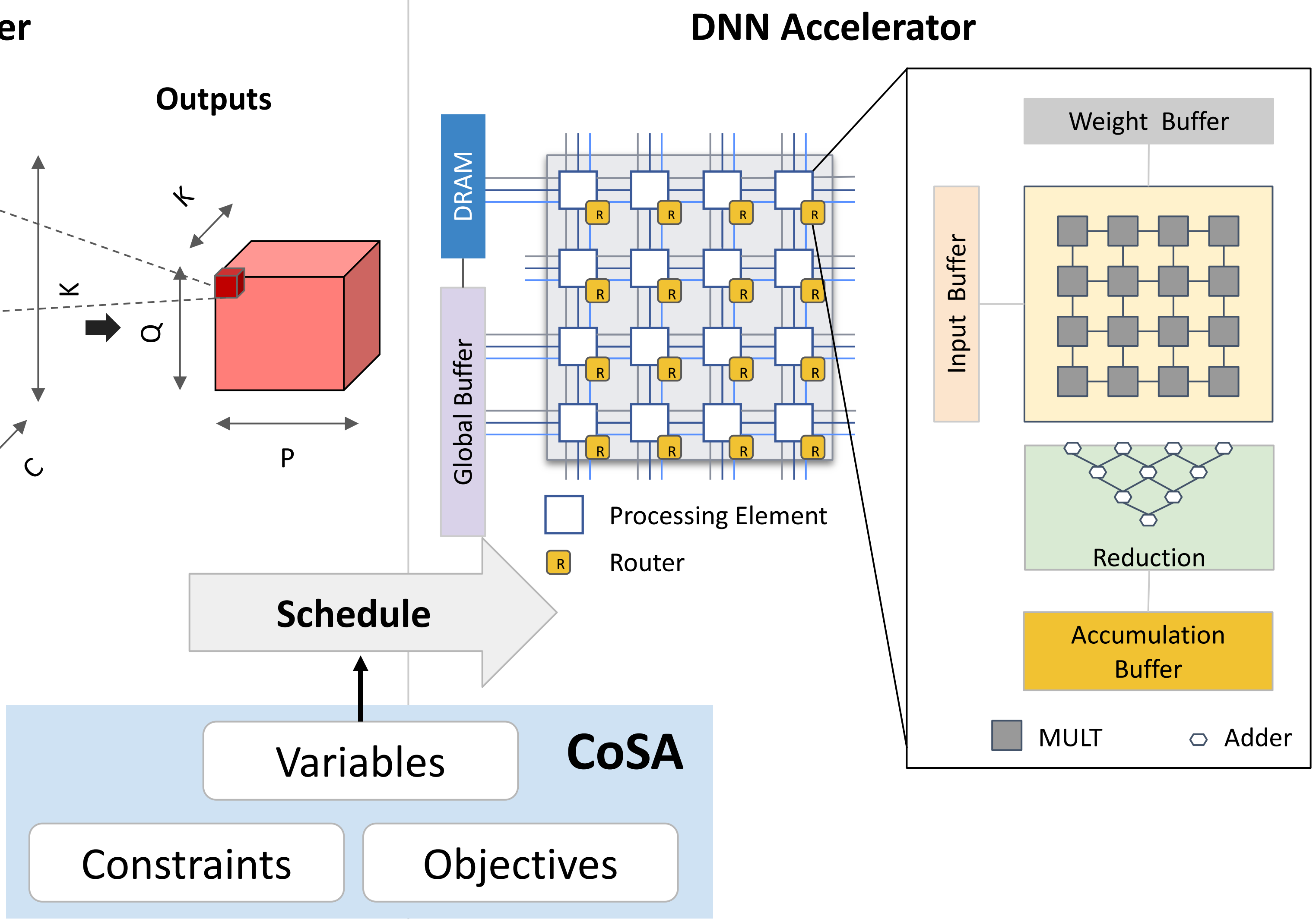
DNN Accelerator



DNN scheduling problem formulation with CoSA



R, S: weight width and height
P, Q: output width and height
C: input channel size
K: output channel size
N: batch size



Three scheduling decisions

DRAM level

for q2 = [0 : 2) :

Global Buffer level

 for q1 = [0 : 7) :

  for n0 = [0 : 3) :

   spatial_for r0 = [0 : 3) :

    spatial_for k1 = [0 : 2) :

Input Buffer level

     for c1 = [0 : 2) :

      for p1 = [0 : 2) :

Weight Buffer level

       for p0 = [0 : 2) :

        spatial_for k0 = [0 : 2) :

…

1. Tiling Factors

2. Spatial / Temporal

3. Loop Permutation

Key idea: prime factor allocation problem

Matrix-vector mult:

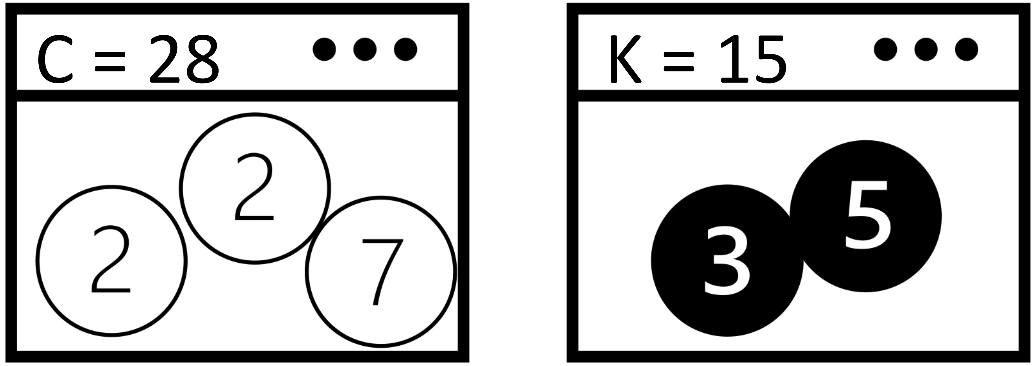
```
for c in [0:C)  // C = 28
    for k in [0:K)   // K = 15
        OA[k] += IA[c] × W[c,k]
```

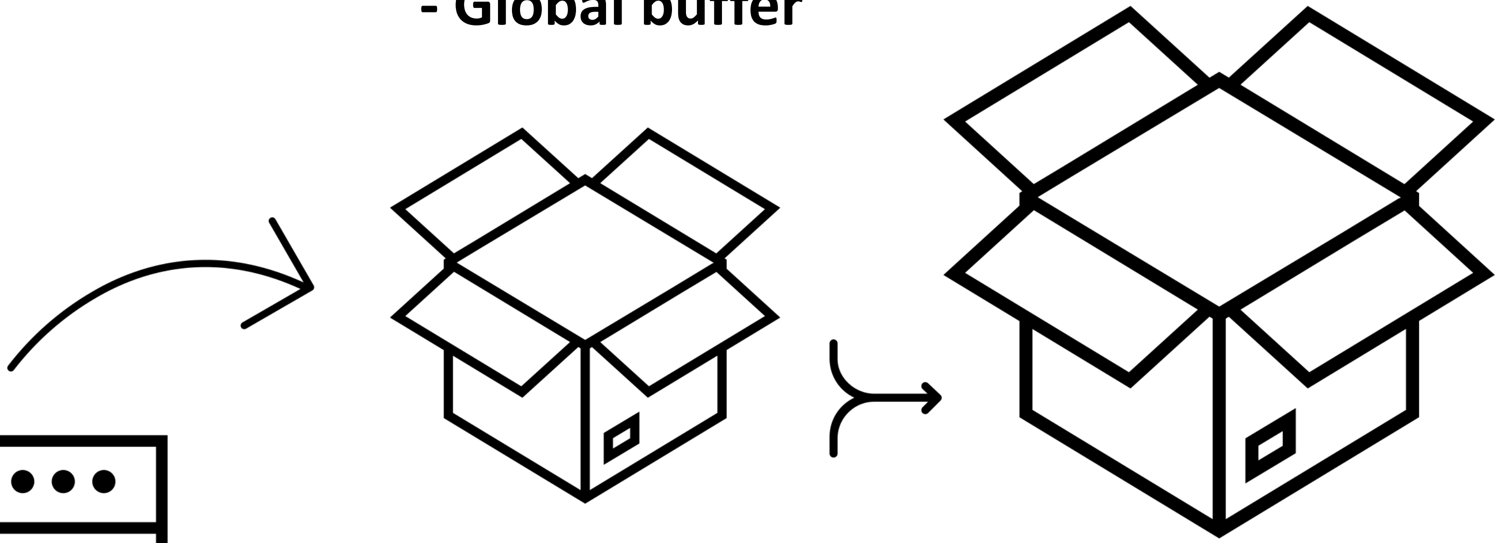
Local buffers:

- Weight buffer

- Global buffer

Prime factor items:



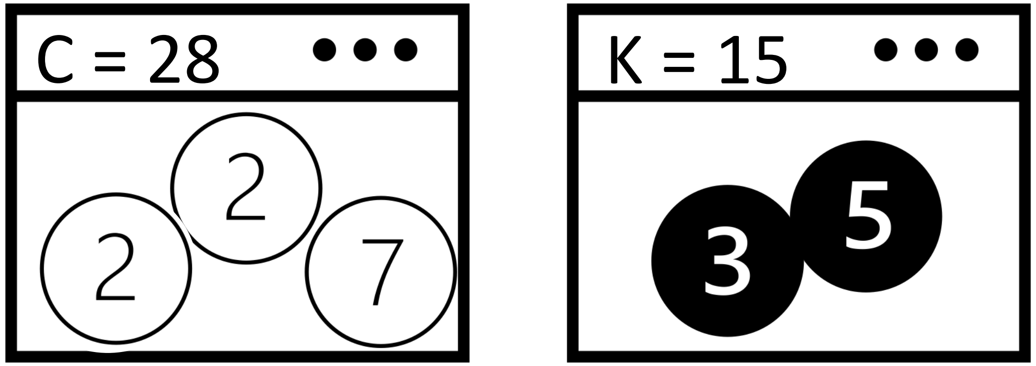


Weight Buffer
(Size = 4)

Global Buffer
(Size = 20)

CoSA Variable X – Tiling Factors

Prime factor items :



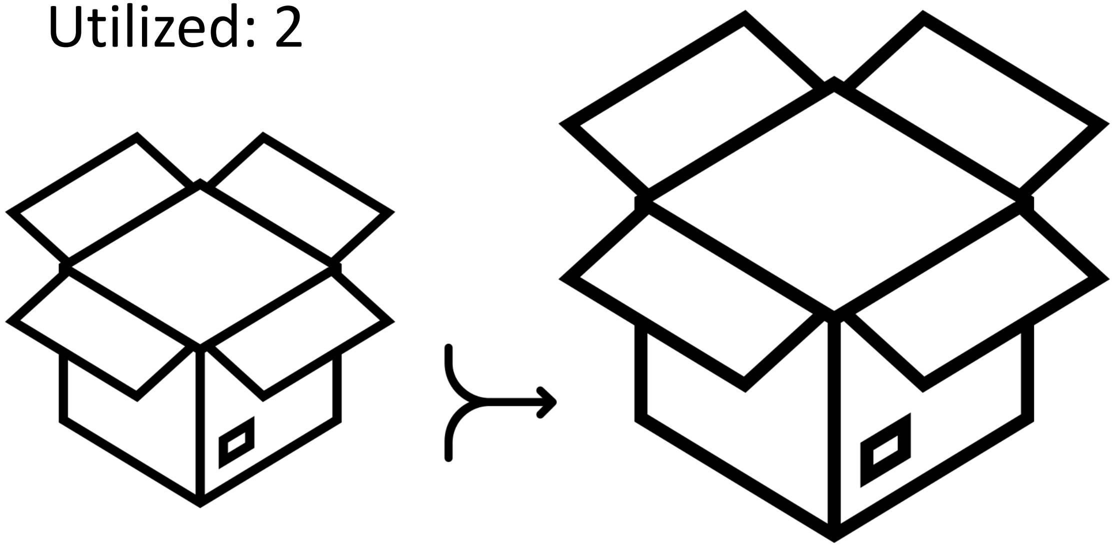
Binary allocation var X:

	C=28			K=15	
Prime Factors	2	2	7	3	5
WeightBuf	✓				
GlobalBuf		✓		✓	✓
DRAM			✓		

Local buffers:

Utilized: 2

Utilized: (2x3x5)x(2)=60

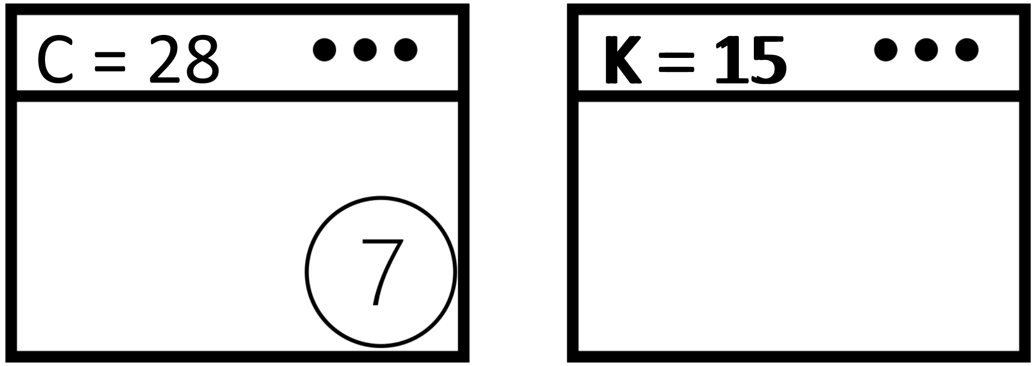


Weight Buffer (Size = 4)

Global Buffer (Size = 80)

CoSA Variable X – Spatial/Temporal Mapping

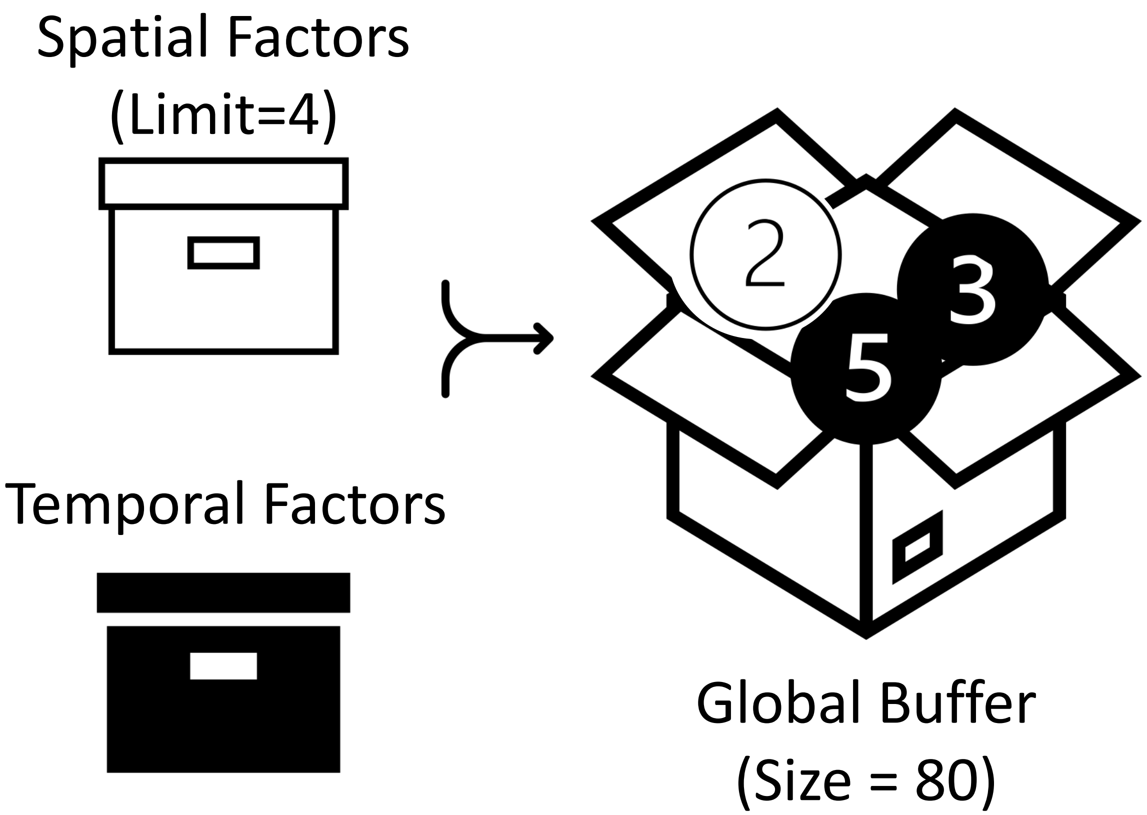
Prime factor items :



Binary allocation var X:

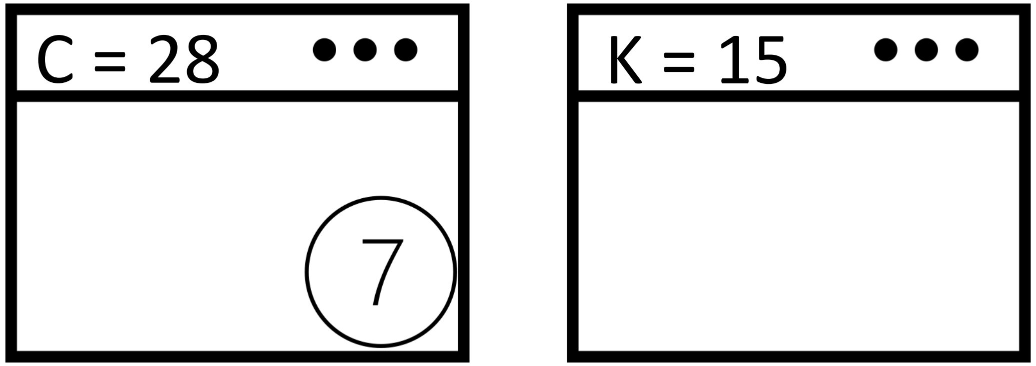
GlobalBuf		C=28			K=15	
	Prime Factors	2	2	7	3	5
	Spatial				✓	
	Temporal	✓				✓

4 PEs in the accelerator:



CoSA Variable X – Loop Permutation

Prime factor items :

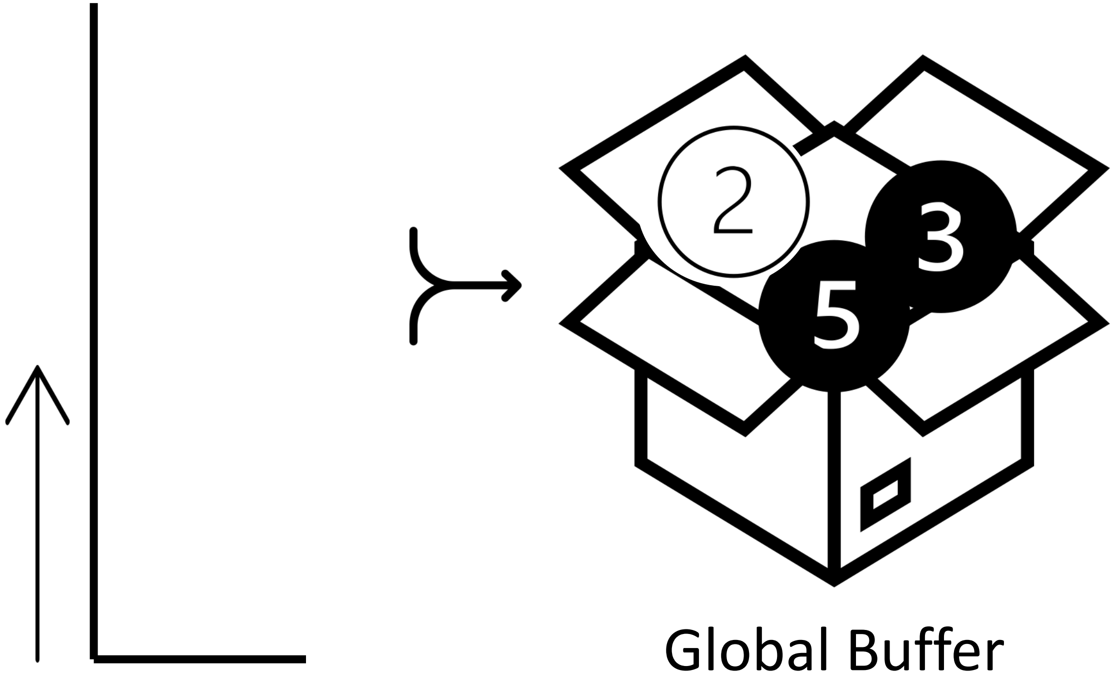


Binary allocation var X:

GlobalBuf

	C=28			K=15	
Prime Factors	2	2	7	3	5
rank0	✓				
rank1					✓
rank2					
rank3					
rank4					

Rank in global buf:



Global Buffer
(Size = 80)

CoSA Variable X – Putting it altogether

<table>
<tr><th></th><th>Memory</th><th>Perm</th><th colspan="3">C=28</th><th colspan="2">K=15</th></tr>
<tr><td>Prime Factors</td><td></td><td></td><td>2</td><td>2</td><td>7</td><td>3</td><td>5</td></tr>
<tr><td rowspan="7"></td><td>WeightBuf</td><td>...</td><td>t</td><td></td><td></td><td></td><td></td></tr>
<tr><td rowspan="5">GlobalBuf</td><td>rank0</td><td></td><td>t</td><td></td><td rowspan="5">s</td><td></td></tr>
<tr><td>rank1</td><td></td><td></td><td></td><td>t</td></tr>
<tr><td>rank2</td><td></td><td></td><td></td><td></td></tr>
<tr><td>rank3</td><td></td><td></td><td></td><td></td></tr>
<tr><td>rank4</td><td></td><td></td><td></td><td></td></tr>
<tr><td>DRAM</td><td>...</td><td></td><td></td><td>t</td><td></td><td></td></tr>
</table>

s - Spatial, t - Temporal

DRAM level

for c2 = [0 : 7) :

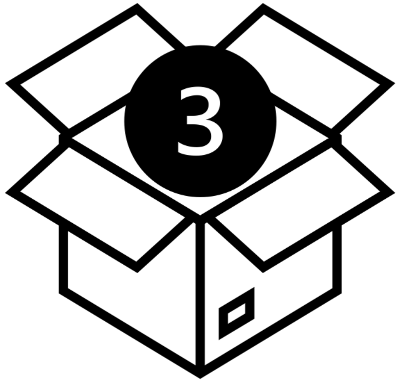
Global Buffer level

for k1 = [0 : 5) :
for c1 = [0 : 2) :
spatial_for k0 = [0 : 3) :

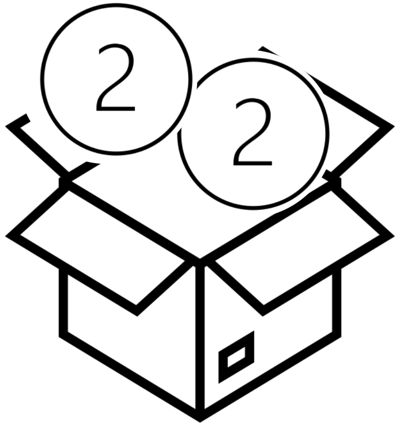
Weight Buffer level

for c0 = [0 : 2) :

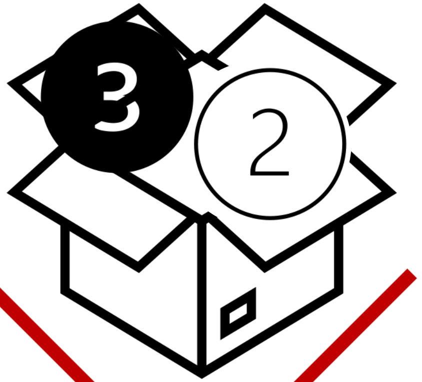
CoSA Constraints: Buffer Utilization



Weight Buffer
(Size = 4)

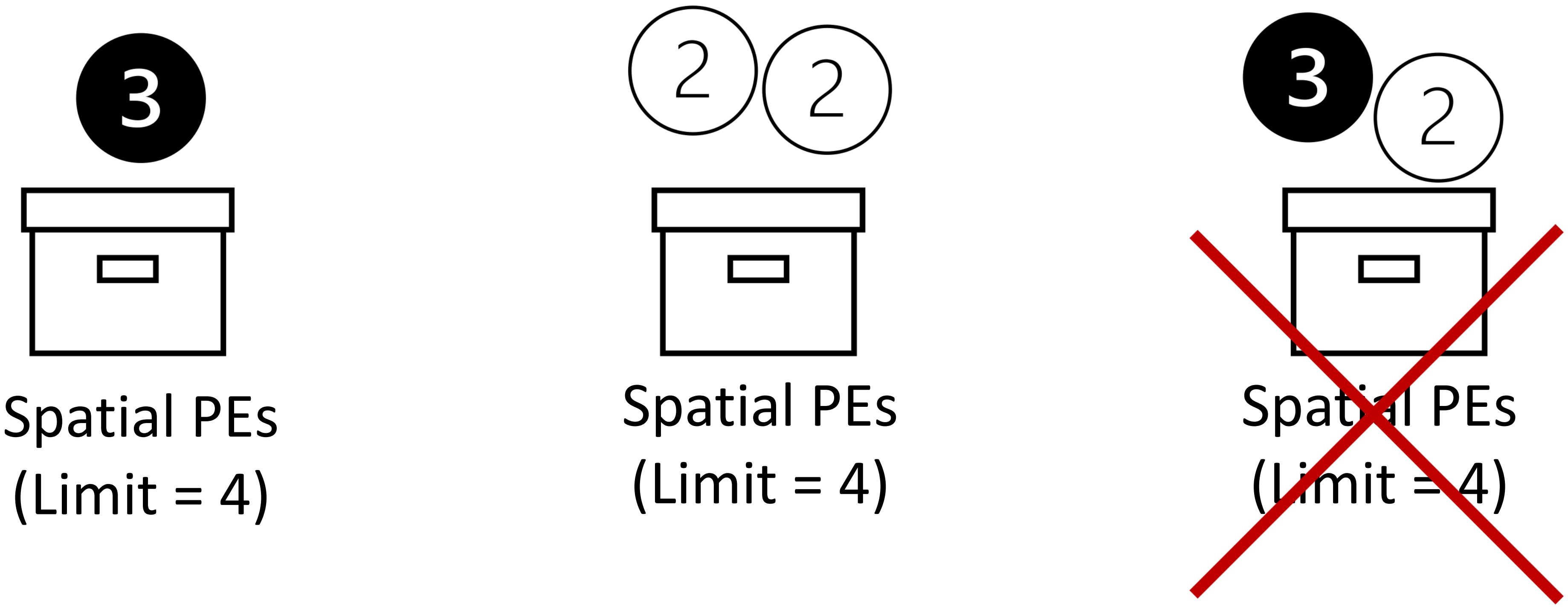


Weight Buffer
(Size = 4)

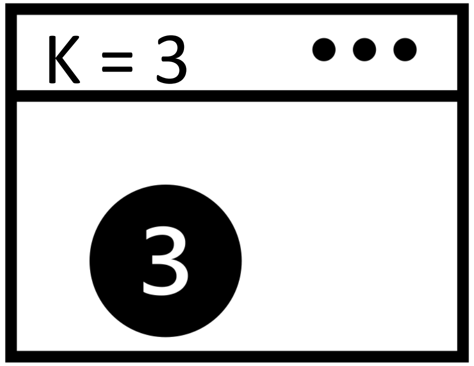


Weight Buffer
(Size = 4)

CoSA Constraints: Spatial Resources



CoSA Binary Constants A and B

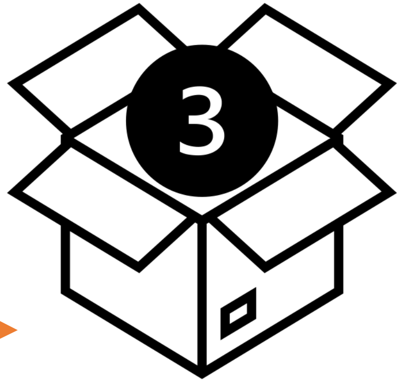


	Related			idx
	W	IA	OA	v
R	✓	-		
S	✓	-		
P		✓	✓	
Q		✓	✓	j
C	✓	✓		
K	✓		✓	
N		✓	✓	

Constant **A**

	Related			idx
	W	IA	OA	v
Register	✓	✓	✓	
AccBuf			✓	
WBuf	✓			i
InputBuf		✓		
GlobalBuf	✓	✓		
DRAM	✓	✓	✓	

Constant **B**



Weight Buffer
(Utilization = 3)

CoSA Constraints

A. Buffer Utilization

$$U_{I,v} = \prod_{i=0}^{I-1}\prod_{j=0}^{6}\prod_{k=0}^{1}\begin{cases} prime\_factor_{j,n}, & X_{(j,n),i,k}A_{v,j}B_{v,I} = 1 \\ 1, & \text{otherwise} \end{cases} \quad (1)$$

Not a linear function of X

Log trick: turning products into linear sums

- Non-linear formula:
  - 3 2 2

    $$\begin{cases} 3, & if\ X_{3\_0} = 1 \\ 1, & otherwise \end{cases} \times \begin{cases} 2, & if\ X_{2\_0} = 1 \\ 1, & otherwise \end{cases} \times \begin{cases} 2, & if\ X_{2\_1} = 1 \\ 1, & otherwise \end{cases} \leq 4$$
  - X_{p_i} represents the i th prime factor with value p

$\Big\downarrow$ Taking log

- Linear formula:
  - $log(3)X_{3_0} + \log(2)\, X_{2\_0} + \log(2)\, X_{2\_1} \leq log(4)$

CoSA Constraints

A. **Buffer Utilization**

$$U_{I,v} = \prod_{i=0}^{I-1}\prod_{j=0}^{6}\prod_{k=0}^{1}\begin{cases} prime\_factor_{j,n}, & X_{(j,n),i,k}A_{v,j}B_{v,I} = 1 \\ 1, & \text{otherwise} \end{cases} \tag{1}$$

Not a linear function of X

Taking `log`

Linear function of X

$$U_{I,v} = \sum_{i=0}^{I-1}\sum_{j=0,n=0}^{6,N}\sum_{k=0}^{1}\log(f_{j,n})A_{v,j}B_{v,I}X_{(j,n),i,k} \leq \log(M_{I,v}), \forall I \tag{2}$$

CoSA Constraints

B. Spatial Resources

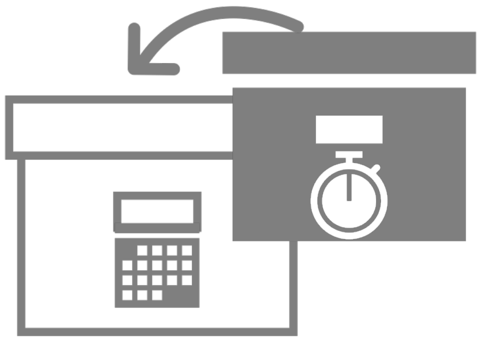
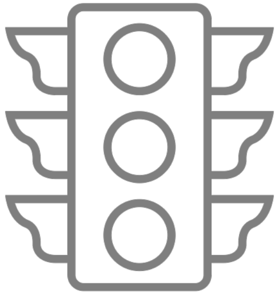
1) each problem factor can only be mapped to either spatial or temporal execution

$$\sum_{k=0}^{1} X_{(j,n),i,k} == 1, \forall (j,n), i \quad (3)$$

2) Spatially-mapped factors do not exceed exceed the resource limit

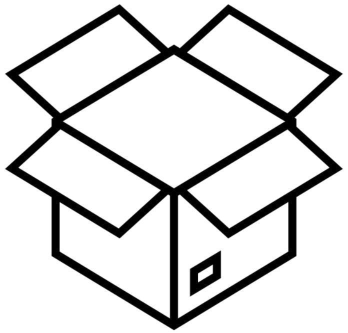
$$\sum_{j=0,n=0}^{6,N} \log(prime\_factor_{j,n}) X_{(j,n),I,0} \leq \log(S_I), \forall I \quad (4)$$

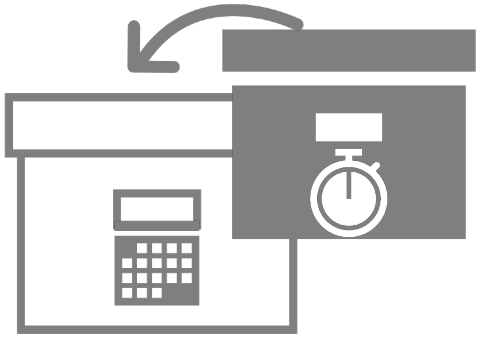
CoSA Objectives

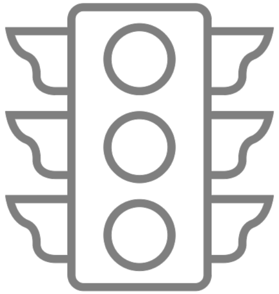




- Utilization-driven
- Compute-driven
- Traffic-driven

CoSA Objectives

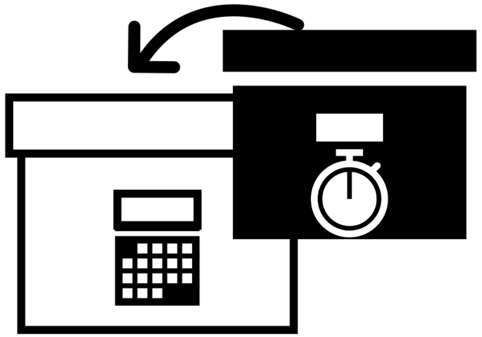


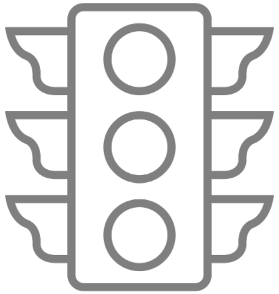




- Utilization-driven
- Compute-driven
- Traffic-driven

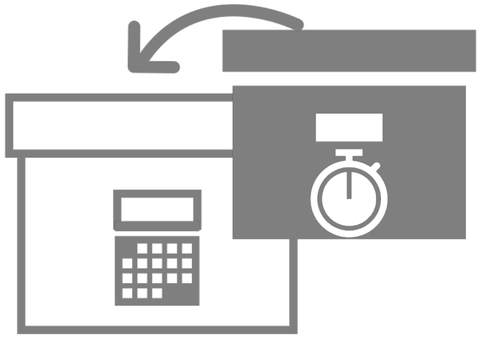
CoSA Objectives

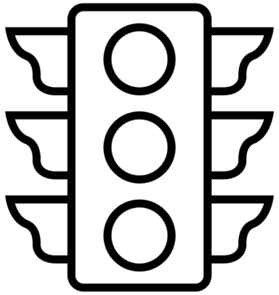




- Utilization-driven
- Compute-driven
- Traffic-driven

CoSA Objectives





- Utilization-driven
- Compute-driven
- Traffic-driven

CoSA Traffic-driven Objective

DRAM level

for c2 = [0 : 7) :

Global Buffer level

for k1 = [0 : 5) :

for c1 = [0 : 2) :

spatial_for k0 = [0 : 3) :

Weight Buffer level

for c0 = [0 : 2) :

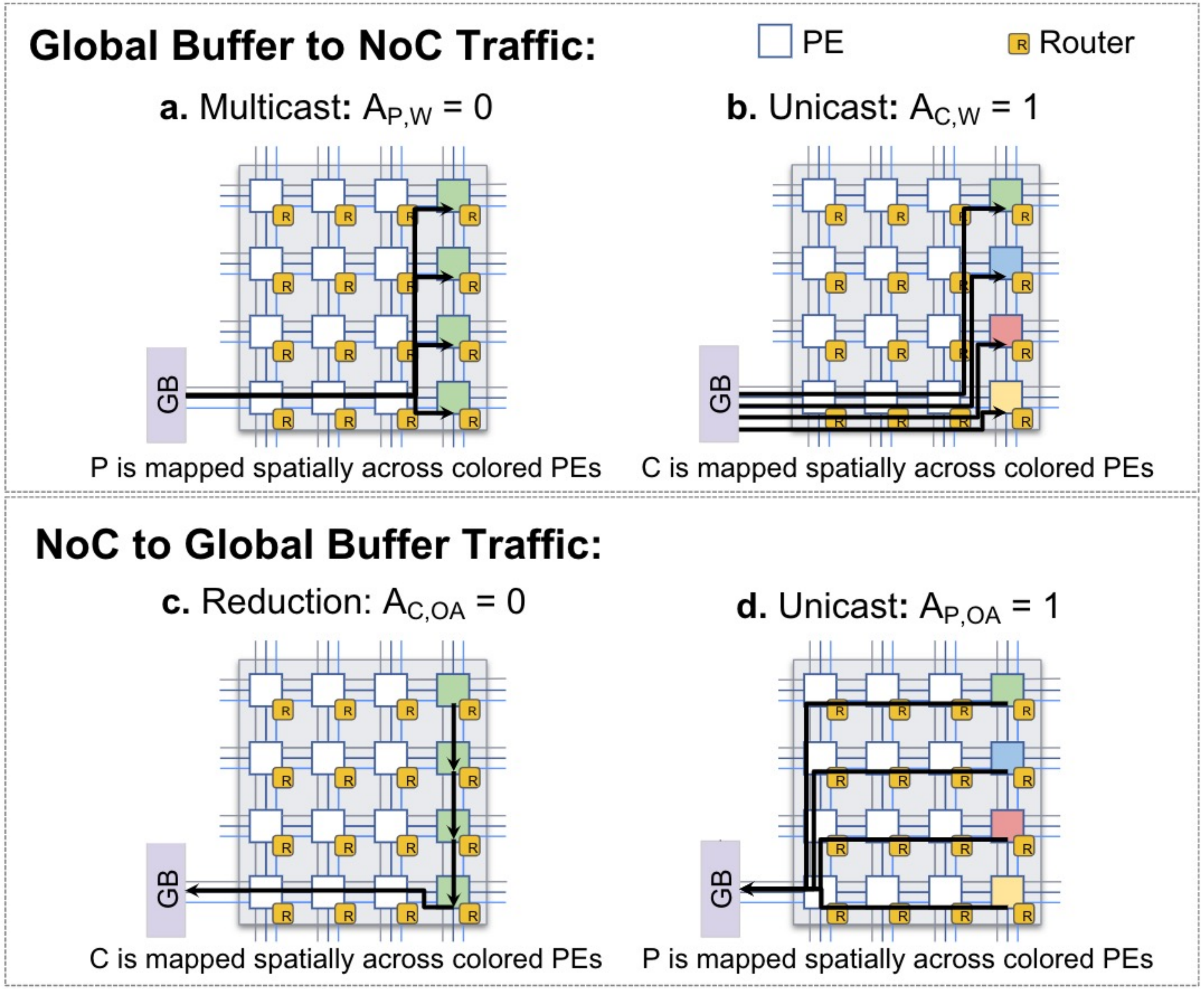
S – Temporal iteration

L – Unicast/multicast traffic

D – Data transfer size

Overall Traffic = $S \times L \times D$

Constant **A** Implied NoC Traffic Patterns



The variable **X** and constant **A** determine the traffic types of different data tensors from global buffer to PEs:

- Multicast
- Unicast
- Reduction

CoSA Objective Functions

1. Utilization-Driven Objective

$$\hat{U} = \sum_{i=0}^{I} \sum_{v=0}^{2} U_{i,v} \tag{5}$$

CoSA Objective Functions

2. Compute-Driven Objective

$$\hat{C} = \sum_{i=0}^{I} \sum_{j=0,n=0}^{6,N} \log(prime\_factor_{j,n}) X_{(j,n),i,\boxed{1)}} \qquad (11)$$

Temporal Mapping

CoSA Objective Functions

3. Traffic-Driven Objective
   a. Data size for each NoC transfer

$$D_v = \sum_{i=0}^{I-1} \sum_{j=0,n=0}^{6,N} \sum_{k=0}^{1} \log(prime\_factor_{j,n}) A_{v,j} X_{(j,n),i,k} \quad (6)$$

   b. Spatial factors to indicate the NoC traffic patterns

$$L_v = \sum_{j=0,n=0}^{6,N} \log(prime\_factor_{j,n}) X_{(j,n),I,0} A_{v,j} \quad (7)$$

   c. Temporal iteration count for different tensors to indicate data reuse

$$R_v = \sum_{p=0}^{P-1} \sum_{j=0,n=0}^{6,N} \log((prime\_factor_{j,n}) Y_{p_v} X(j,n), p, 1) \quad (9)$$

   d. Overall

$$\hat{T} = \sum_{v=0}^{2} D_v + L_v + R_v \quad (10)$$

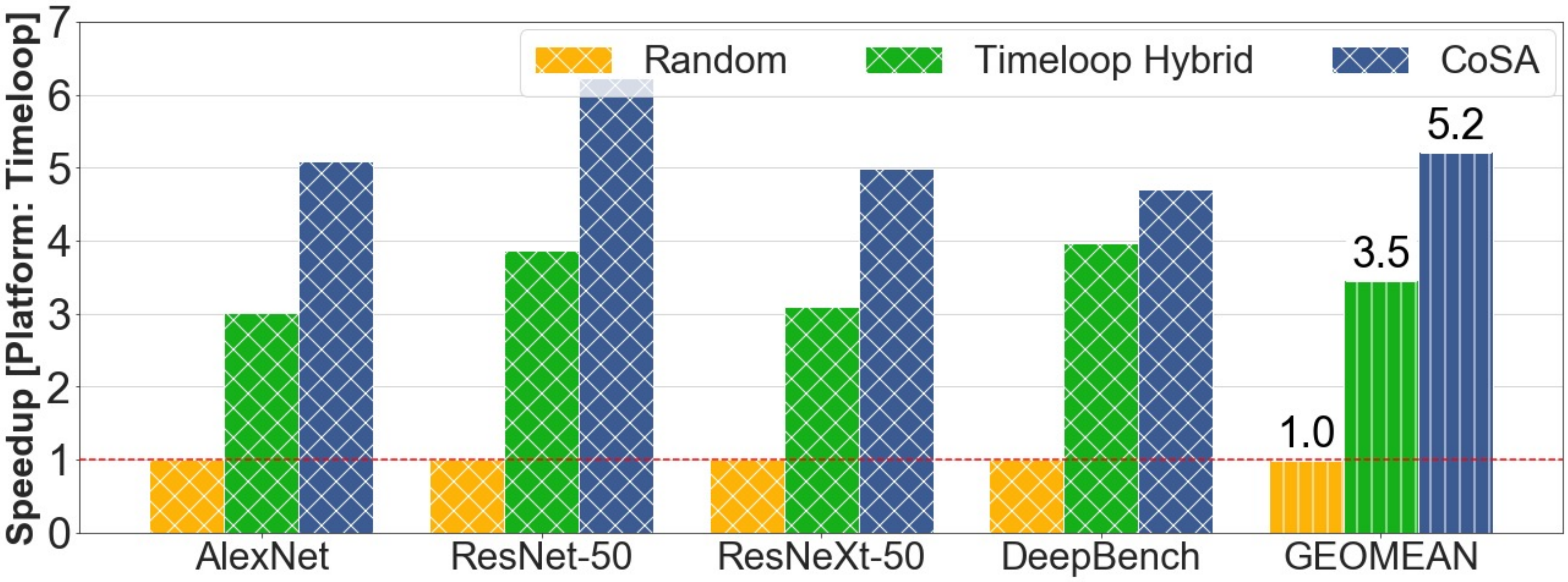
CoSA Evaluation

- Baselines:
  - Random (best out of 5 valid schedules)
  - Timeloop Hybrid (best out of 16K valid schedules)

- DNN workloads:
  - AlexNet, ResNet-50, ResNext-50, DeepBench

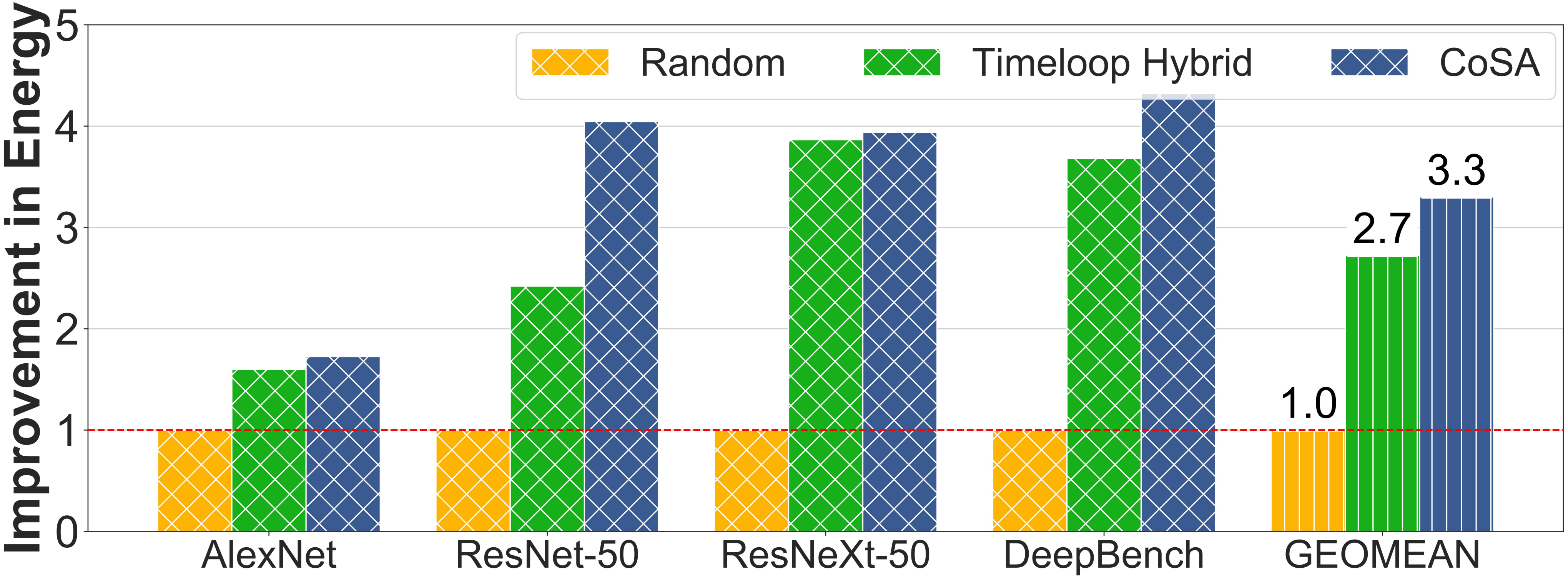
- Platforms:
  - Timeloop Simulator
  - SystemC NoC Simulator
  - GPU

1.5x latency speedup



- 5.2x better than Random
- 1.5x better than Timeloop Hybrid

1.2x better energy efficiency



- 3.3x better than Random
- 1.2x better than Timeloop Hybrid

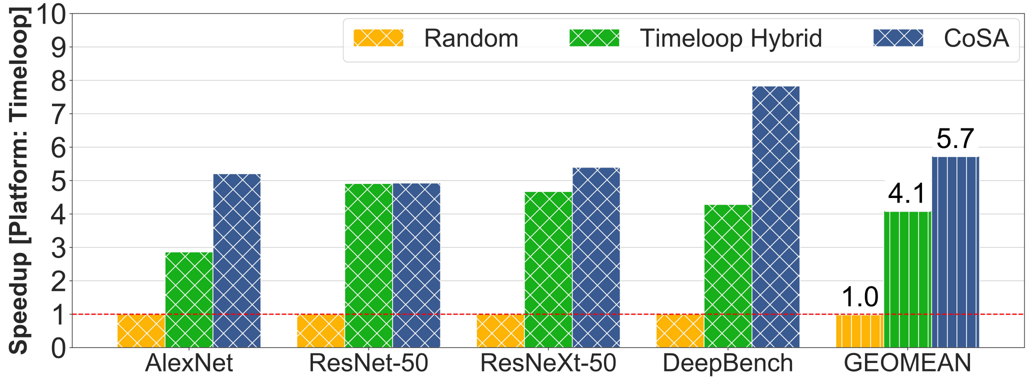
90x faster time-to-solution with CoSA

	CoSA	Random	Timeloop Hybrid
Runtime / Layer	4.2s	4.6s (1.1x)	379.9s (90.5x)
Samples / Layer	1	20K	67M
Evaluations/ Layer	1	5	16K

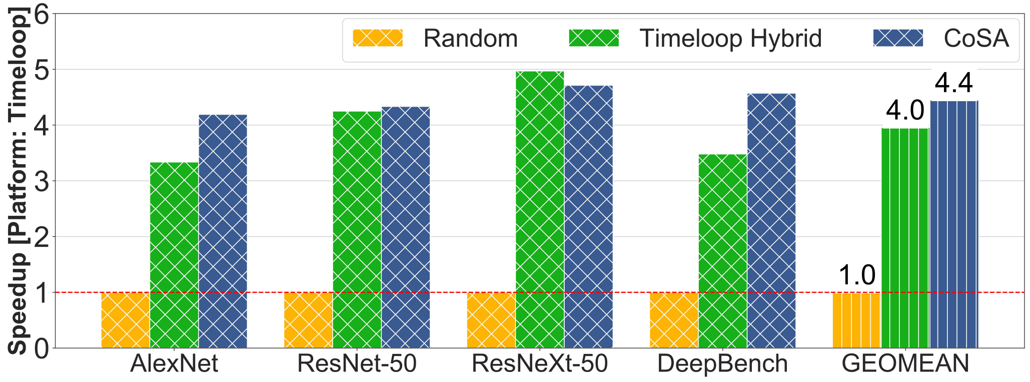
- Generates schedules within seconds
- Significantly reduces the number of samples and evaluations

CoSA generalizes to different architectures

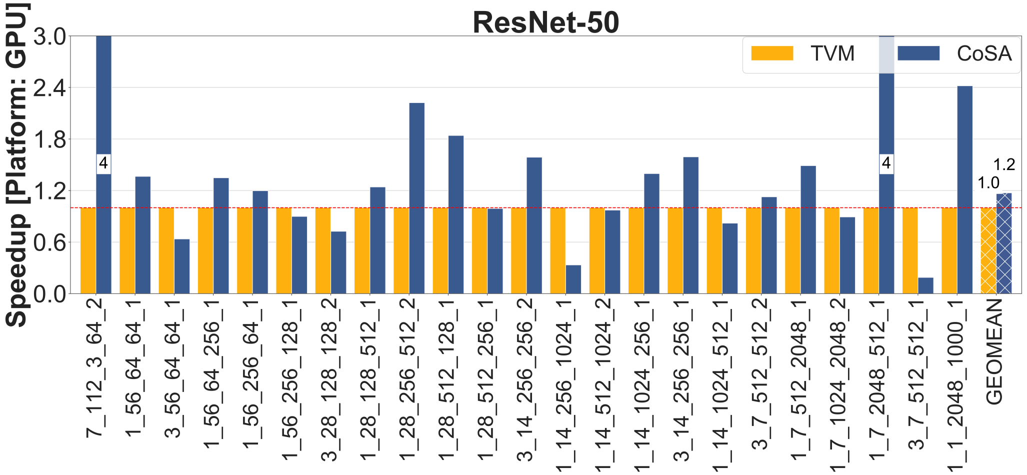
- Larger Buffers – 1.4x speedup



- 8x8 PEs – 1.1x speedup



- GPU – 1.2x speedup, 2500x faster time-to-solution over TVM (50 samples)



Conclusion

- We formulate DNN accelerator scheduling as a constrained optimization that can be solved in ***one shot***.
- We take *a* ***communication-oriented*** approach in the formulation and exposes the cost through clearly-defined objective functions.
- We demonstrate that CoSA can ***quickly*** generate ***high-performance*** schedules outperforming state-of-the-art approaches.

Github: https://github.com/ucb-bar/cosa

Questions?

jennyhuang@nvidia.com