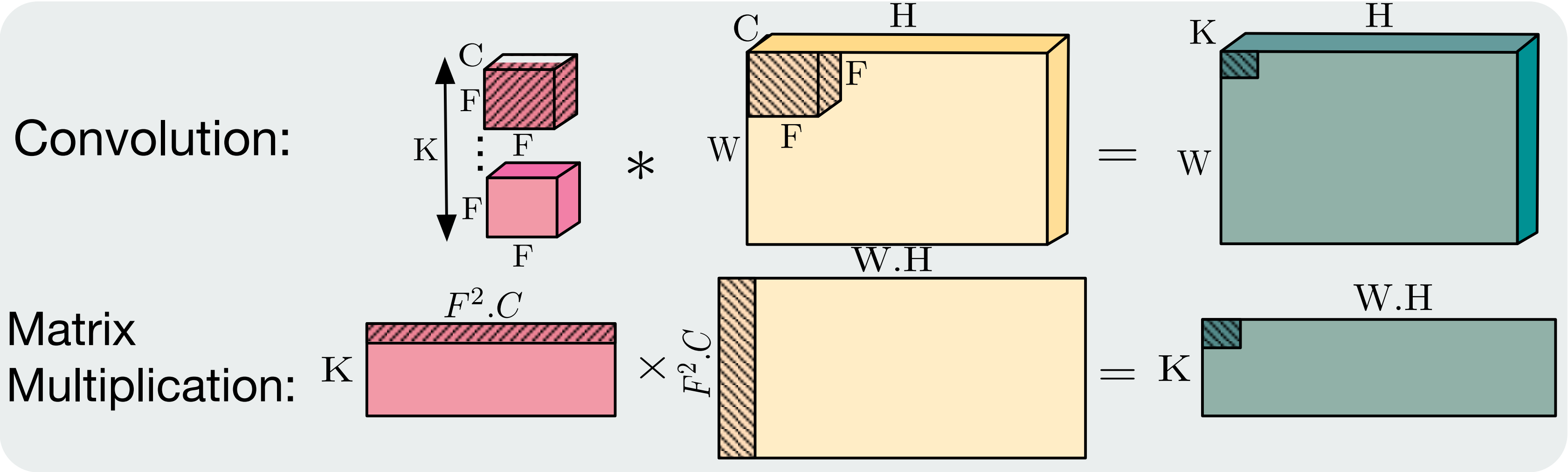
Proposing a Fast and Scalable Systolic Array for Matrix Multiplication

Bahar Asgari, Ramyad Hadidi, Hyesoon Kim

Matrix Multiplication

Matrix multiplication is the key operation in many applications

Example: convolution in neural networks



Systolic arrays perform matrix multiplication that

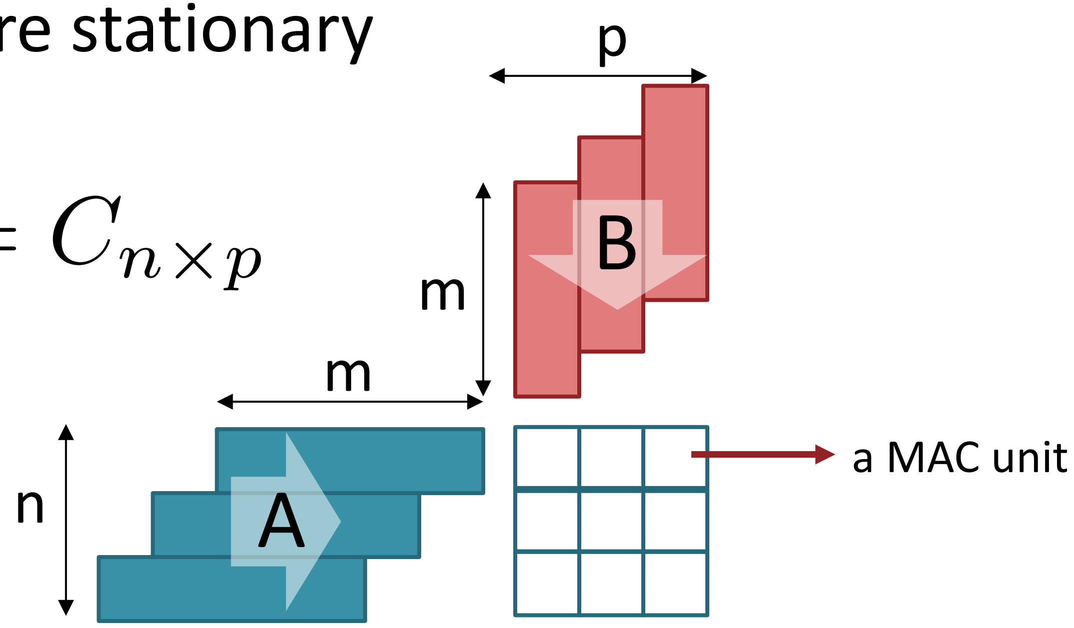
- Includes several similar operations (i.e., multiply and accumulation)
- Captures high data reuse rate

Systolic Arrays for Matrix Multiplication

Non-stationary

None of the operands are stationary

$$A_{n \times m} \times B_{m \times p} = C_{n \times p}$$



Systolic Arrays for Matrix Multiplication

Non-stationary

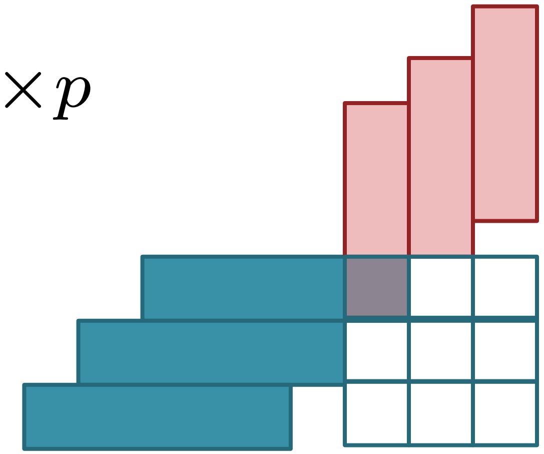
None of the operands are stationary

$$A_{n\times m} \times B_{m\times p} = C_{n\times p}$$

Phase 1:

only processing

Time steps: 1



Systolic Arrays for Matrix Multiplication

Non-stationary

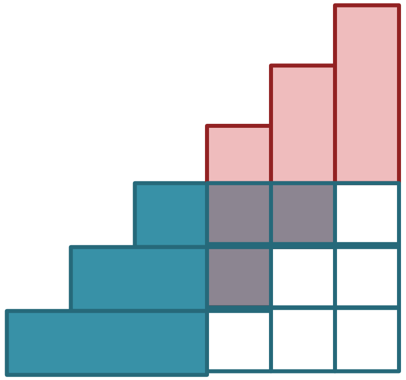
None of the operands are stationary

$$A_{n\times m} \times B_{m\times p} = C_{n\times p}$$

Phase 1

only processing

Time steps: 2



Systolic Arrays for Matrix Multiplication

Non-stationary

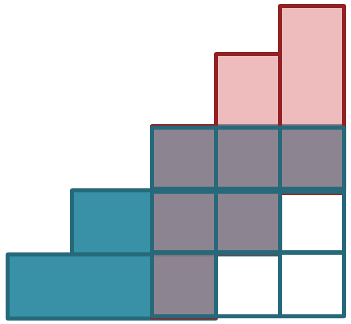
None of the operands are stationary

$$A_{n \times m} \times B_{m \times p} = C_{n \times p}$$

Phase 1:

only processing

Time steps: 3



Systolic Arrays for Matrix Multiplication

Non-stationary

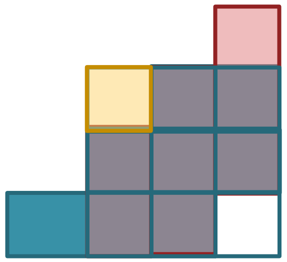
None of the operands are stationary

$$A_{n\times m} \times B_{m\times p} = C_{n\times p}$$

Phase 1:

only processing

Time steps: 4



Systolic Arrays for Matrix Multiplication

Non-stationary

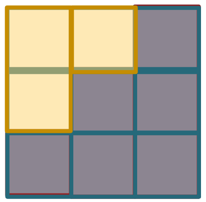
None of the operands are stationary

$$A_{n\times m} \times B_{m\times p} = C_{n\times p}$$

Phase 1:

only processing

Time steps: 5



Systolic Arrays for Matrix Multiplication

Non-stationary

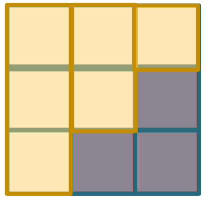
None of the operands are stationary

$$A_{n\times m} \times B_{m\times p} = C_{n\times p}$$

Phase 1:

only processing

Time steps: n + m



Systolic Arrays for Matrix Multiplication

Non-stationary

None of the operands are stationary

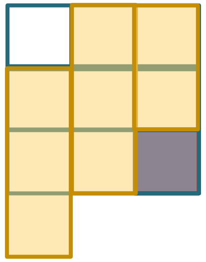
$$A_{n\times m} \times B_{m\times p} = C_{n\times p}$$

Phase 2:

processing and offloading

Time steps: n + m + **1**

Phase 1



Systolic Arrays for Matrix Multiplication

Non-stationary

None of the operands are stationary

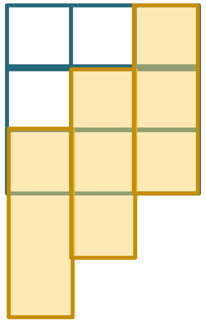
$$A_{n\times m} \times B_{m\times p} = C_{n\times p}$$

Phase 3:

only offloading

Time steps: n + m + p - 2 **+ 1**

Phase 1 (n + m)   Phase 2 (p - 2)



Systolic Arrays for Matrix Multiplication

Non-stationary

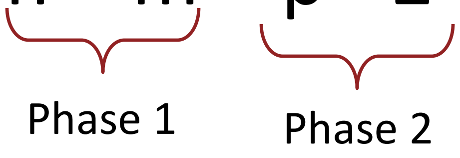
None of the operands are stationary

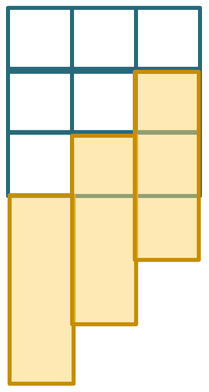
$$A_{n\times m} \times B_{m\times p} = C_{n\times p}$$

Phase 3:

only offloading

Time steps: n + m + p - 2 **+ 2**





Systolic Arrays for Matrix Multiplication

Non-stationary

None of the operands are stationary

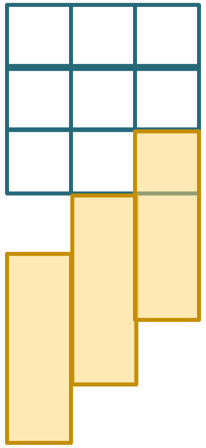
$$A_{n\times m} \times B_{m\times p} = C_{n\times p}$$

Phase 3:

only offloading

Time steps: n + m + p - 2 **+ n**

Phase 1 Phase 2



Systolic Arrays for Matrix Multiplication

Non-stationary

None of the operands are stationary

$$A_{n\times m} \times B_{m\times p} = C_{n\times p}$$

Phase 3:

only offloading

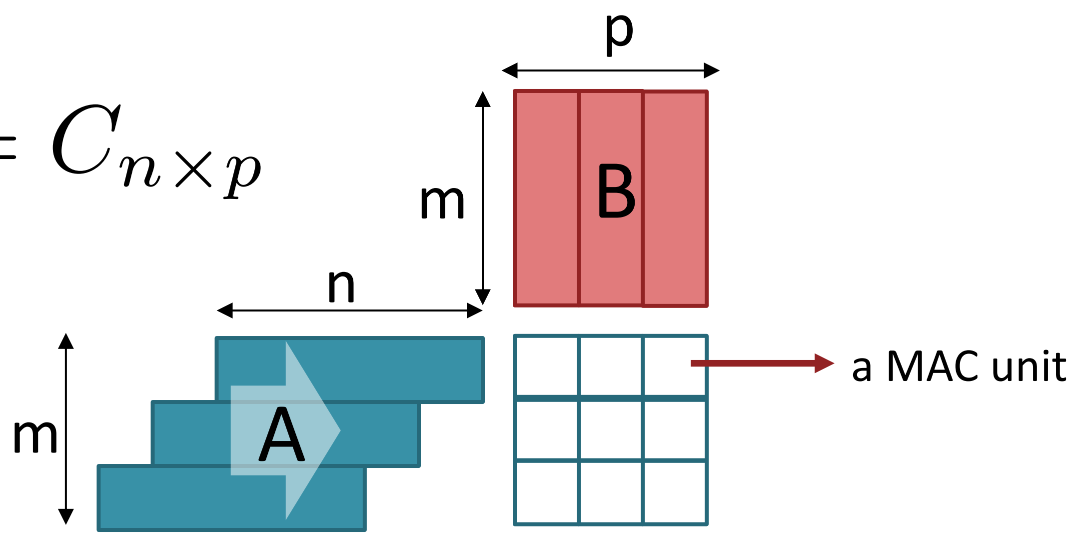
Time steps: 2n + m + p - 2

Systolic Arrays for Matrix Multiplication

Stationary

One operand (here, B) is stationary

$$A_{n\times m} \times B_{m\times p} = C_{n\times p}$$

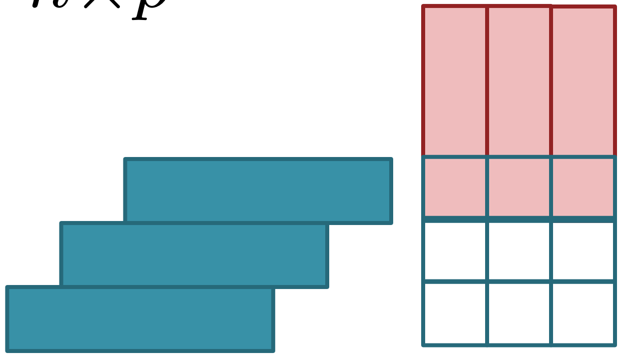


Systolic Arrays for Matrix Multiplication

Stationary

One operand (here, B) is stationary

$$A_{n \times m} \times B_{m \times p} = C_{n \times p}$$



Phase 1:

only loading B

Time steps: 1

Systolic Arrays for Matrix Multiplication

Stationary

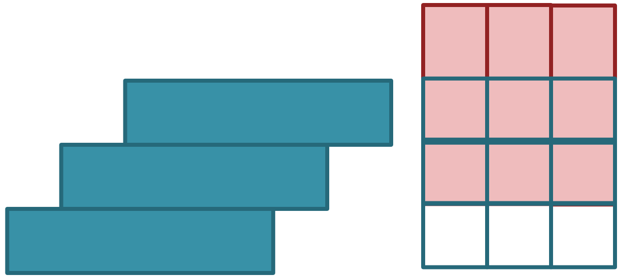
One operand (here, B) is stationary

$$A_{n \times m} \times B_{m \times p} = C_{n \times p}$$

Phase 1:

only loading B

Time steps: m - 1

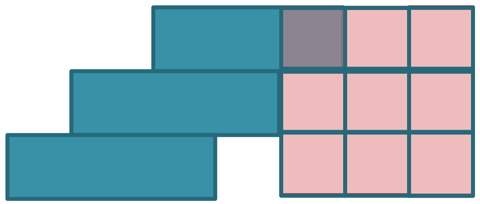


Systolic Arrays for Matrix Multiplication

Stationary

One operand (here, B) is stationary

$$A_{n \times m} \times B_{m \times p} = C_{n \times p}$$



Phase 2:

loading B and processing

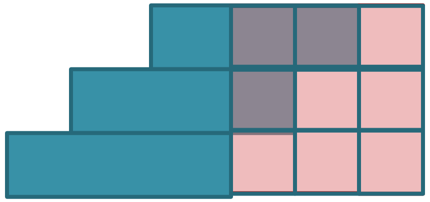
Time steps: $\underbrace{m - 1}_{\text{Phase 1}}$ **+ 1**

Systolic Arrays for Matrix Multiplication

Stationary

One operand (here, B) is stationary

$$A_{n\times m} \times B_{m\times p} = C_{n\times p}$$



Phase 3:

only processing

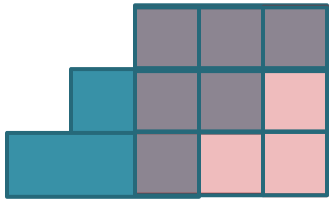
Time steps: $\underbrace{m}_{\text{Phase 1 \&2}}$ **+ 1**

Systolic Arrays for Matrix Multiplication

Stationary

One operand (here, B) is stationary

$$A_{n\times m} \times B_{m\times p} = C_{n\times p}$$



Phase 3:

only processing

Time steps: $\underbrace{m}_{\text{Phase 1 \&2}}$ **+ m - 1**

Systolic Arrays for Matrix Multiplication

Stationary

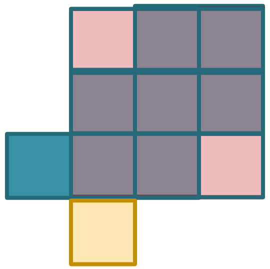
One operand (here, B) is stationary

$$A_{n\times m} \times B_{m\times p} = C_{n\times p}$$

Phase 4:

processing and offloading

Time steps: $\underbrace{2m - 1}_{\text{Phase 1 \&2\&3}}$ **+ 1**



Systolic Arrays for Matrix Multiplication

Stationary

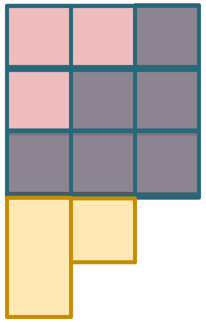
One operand (here, B) is stationary

$$A_{n \times m} \times B_{m \times p} = C_{n \times p}$$

Phase 4:

processing and offloading

Time steps: $\underbrace{2m - 1}_{\text{Phase 1 \&2\&3}}$ **+ 2**



Systolic Arrays for Matrix Multiplication

Stationary

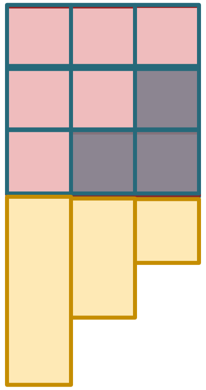
One operand (here, B) is stationary

$$A_{n \times m} \times B_{m \times p} = C_{n \times p}$$

Phase 4:

processing and offloading

Time steps: $\underbrace{2m - 1}_{\text{Phase 1 \&2\&3}}$ **+ 3**



Systolic Arrays for Matrix Multiplication

Stationary

One operand (here, B) is stationary

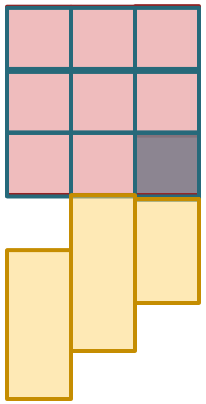
$$A_{n \times m} \times B_{m \times p} = C_{n \times p}$$

Phase 4:

processing and offloading

Time steps: 2m - 1 **+ n + p - 2**

Phase 1 &2&3



Systolic Arrays for Matrix Multiplication

Stationary

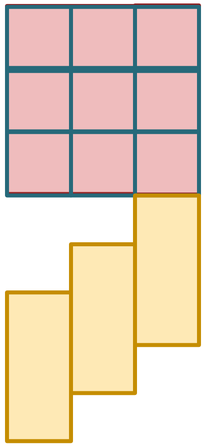
One operand (here, B) is stationary

$$A_{n \times m} \times B_{m \times p} = C_{n \times p}$$

Phase 5:

only offloading

Time steps: $\underbrace{2m - 1}_{\text{Phase 1 \&2\&3}} + \underbrace{n + p - 2}_{\text{Phase 4}}$ **+ 1**



Stationary

One operand (here, B) is stationary

$$A_{n \times m} \times B_{m \times p} = C_{n \times p}$$

Phase 5:

only offloading

Time steps: n + 2m + p - 2

Key Challenge

The systolic arrays proposed by prior work are not scalable:

- Their latency grows linearly with the size of the inputs
- Latency is the key metric for single-batch inference

$$A_{n \times m} \times B_{m \times p} = C_{n \times p}$$

Non-Stationary

Time steps: 2n + m + p - 2

Stationary

Time steps: n + 2m + p - 2

Key Insight and Proposed Systolic Array

Matrix multiplication consists of

- Multiplication
- Additions → This can be done in log(m) for m numbers

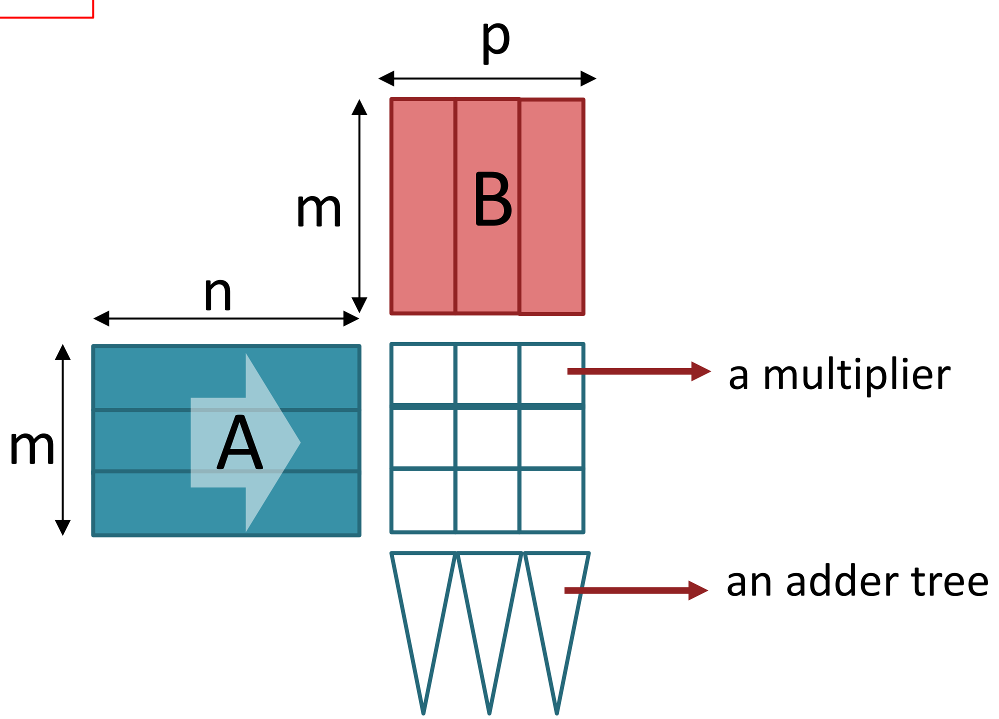
In optimized implementation

- Latency increases sublinearly with the input size

We propose a systolic array with separate

- Multiplier array
- Adder-tree array

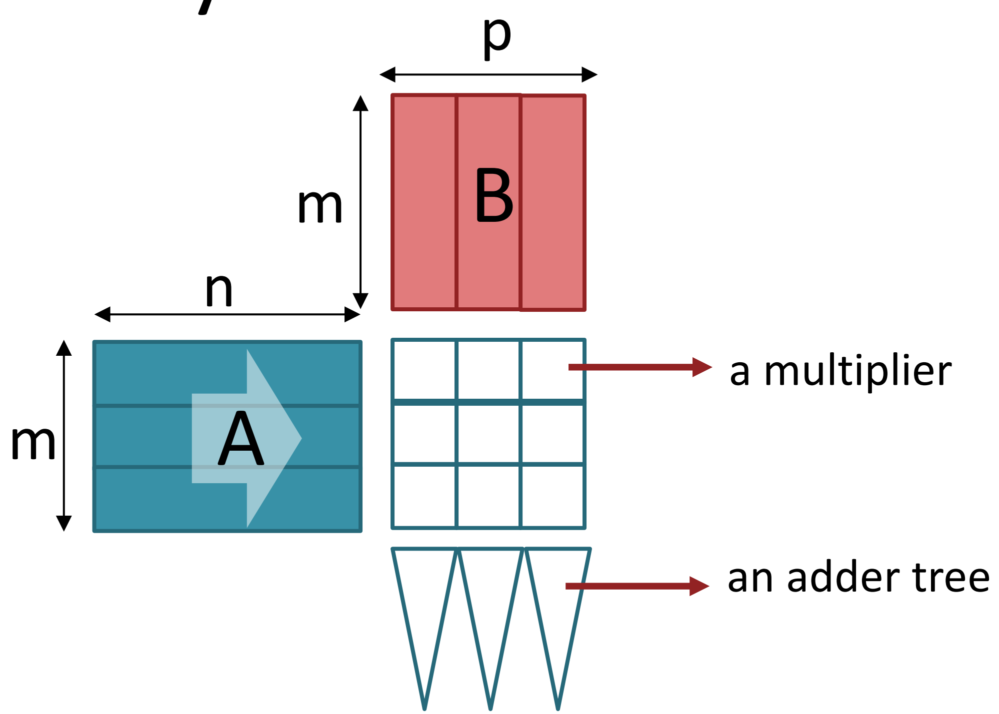
Time steps: n + ~~2m~~ + p - 2
m + log(m)



Our proposed systolic array

One operand (here, B) is stationary

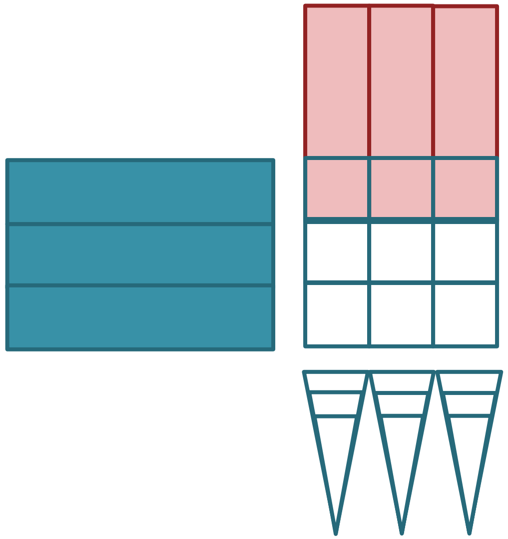
$$A_{n\times m} \times B_{m\times p} = C_{n\times p}$$



Our proposed systolic array

One operand (here, B) is stationary

$$A_{n \times m} \times B_{m \times p} = C_{n \times p}$$



Phase 1:

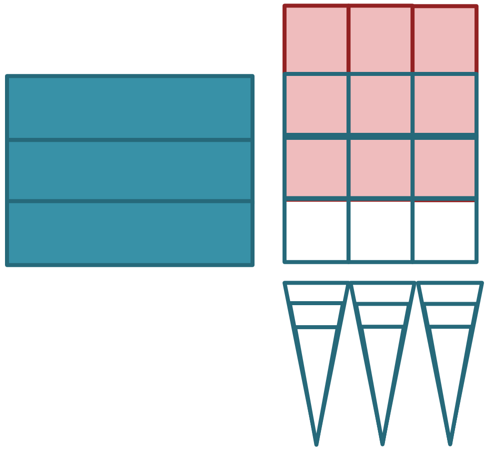
only loading B

Time steps: 1

Our proposed systolic array

One operand (here, B) is stationary

$$A_{n\times m} \times B_{m\times p} = C_{n\times p}$$



Phase 1:

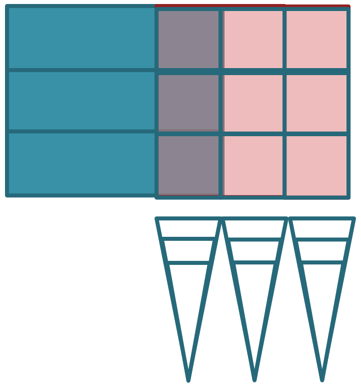
only loading B

Time steps: m-1

Our proposed systolic array

One operand (here, B) is stationary

$$A_{n \times m} \times B_{m \times p} = C_{n \times p}$$



Phase 2:

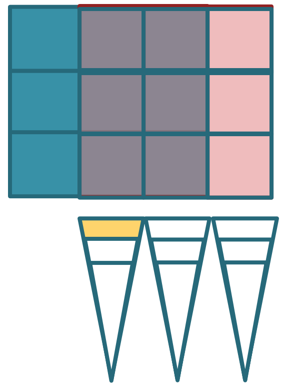
loading B and multiplication

Time steps: $\underbrace{m - 1}_{\text{Phase 1}}$ **+ 1**

Our proposed systolic array

One operand (here, B) is stationary

$$A_{n\times m} \times B_{m\times p} = C_{n\times p}$$



Phase 3:

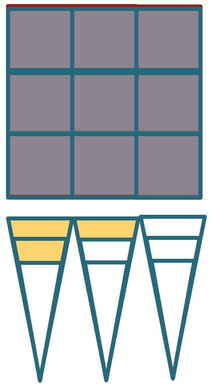
multiplication and addition

Time steps: $\underbrace{m}_{\text{Phase 1 \&2}}$ **+ 1**

Our proposed systolic array

One operand (here, B) is stationary

$$A_{n \times m} \times B_{m \times p} = C_{n \times p}$$



Phase 3:

multiplication and addition

Time steps: m **+ 2**

Phase 1 &2

Our proposed systolic array

One operand (here, B) is stationary

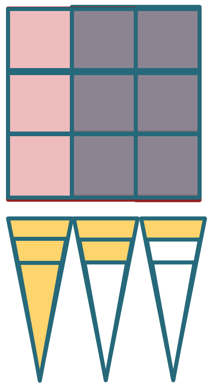
$$A_{n\times m} \times B_{m\times p} = C_{n\times p}$$

Phase 3:

multiplication and addition

Time steps: m **+ 3**

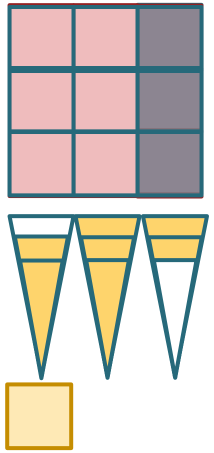
Phase 1 &2



Our proposed systolic array

One operand (here, B) is stationary

$$A_{n\times m} \times B_{m\times p} = C_{n\times p}$$



Phase 3:

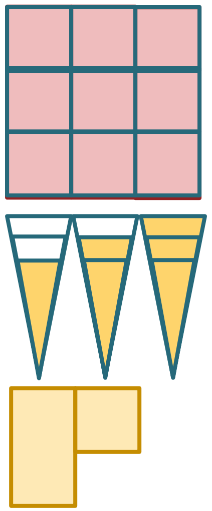
multiplication and addition

Time steps: $\underbrace{m}_{\text{Phase 1 \&2}}$ **+ 4**

Our proposed systolic array

One operand (here, B) is stationary

$$A_{n\times m} \times B_{m\times p} = C_{n\times p}$$



Phase 4:

only addition

Time steps: $\underbrace{m}_{\text{Phase 1 \&2}} + \underbrace{n + p - 2}_{\text{Phase 3}}$ **+ 1**

Our proposed systolic array

One operand (here, B) is stationary

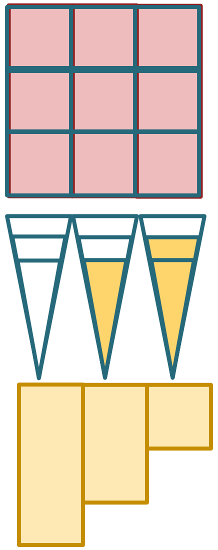
$$A_{n\times m} \times B_{m\times p} = C_{n\times p}$$

Phase 4:

only addition

Time steps: m + n + p - 2 **+ 2**

Phase 1 &2 Phase 3



Our proposed systolic array

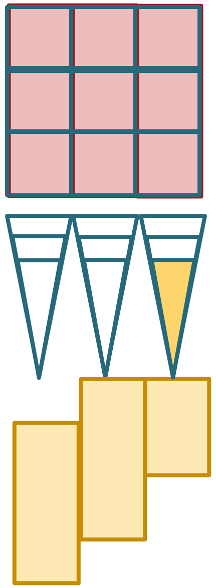
One operand (here, B) is stationary

$$A_{n\times m} \times B_{m\times p} = C_{n\times p}$$

Phase 4:

only addition

Time steps: $\underbrace{m}_{\text{Phase 1 \&2}} + \underbrace{n + p - 2}_{\text{Phase 3}}$ **+ 3**



Our proposed systolic array

One operand (here, B) is stationary

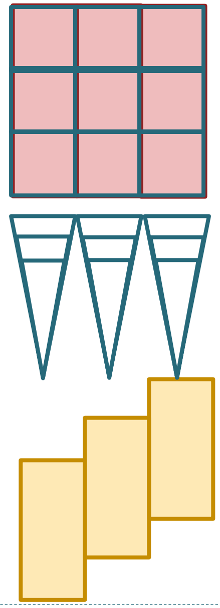
$$A_{n\times m} \times B_{m\times p} = C_{n\times p}$$

Phase 4:

only addition

Time steps: m + n + p - 2 **+ log (m)**

Phase 1 &2 Phase 3



Our proposed systolic array

One operand (here, B) is stationary

$$A_{n \times m} \times B_{m \times p} = C_{n \times p}$$

Phase 4:

only addition

Time steps: n + m + log(m) + p - 2

Implementation

Tools and Devices:

- ZYNQ XC7z020
- Vivado HLS

Benchmark:

- DNNs (VGG16, VGGS, AlexNet, CifarNet, ResNet50)

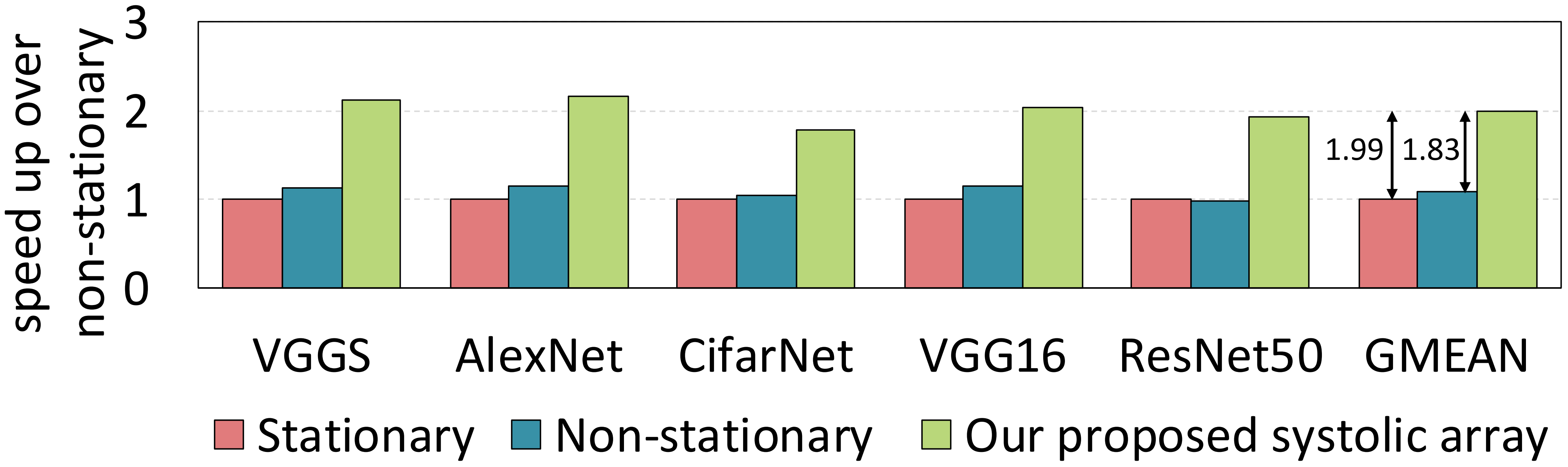
Metrics:

- Latency
- Energy consumption

Results – Speedup and Energy Consumption

Our proposed systolic array is

- 1.99x faster than non-stationary while consuming 2.12x less energy
- 1.83x faster than stationary while consuming 2.27x less energy



Conclusions

Systolic arrays have seen significant interest

- because of their unique interconnections that satisfies the unique requirement of data reuse in matrix multiplication.

Although the systolic arrays in prior work offer high throughput, their latency is not optimized

- Latency is the key factor for single-batch inference!

To optimize latency, we propose a new systolic array consisting of separate multiplier and adder-tree arrays

- It is faster than both prior proposals when the size of the operands grows