CSDS 451: Designing High Performant Systems for AI

Lecture 8

9/18/2025

Sanmukh Kuppannagari

sanmukh.kuppannagari@case.edu

https://sanmukh.research.st/

Case Western Reserve University

Outline

- Systolic Arrays

Announcements

- WA1 due tonight
- WA2 will be out by tomorrow morning

Outline

- Systolic Arrays

Learning Objectives

- Learn the basics of Systolic Arrays
- Learn how to perform matrix multiplication on Systolic Arrays

Reading Materials

- Systolic Arrays:
  - Lipton, Richard J., and Daniel Lopresti. "A systolic array for rapid string comparison." *Proceedings of the Chapel Hill Conference on VLSI*. NC: Chapel Hill, 1985.
  - Kung, H. T., and Charles E. Leiserson. "Systolic arrays (for VLSI)." *Sparse Matrix Proceedings 1978*. Vol. 1. Philadelphia, PA, USA: Society for industrial and applied mathematics, 1979.

Systolic Arrays

- Homogeneous Network of Tightly Coupled Data Processing Units

- Data Processing Units
  - Triggered by the arrival of inputs
  - Produce output and send them to next nodes

Systolic Arrays

- Benefits
  - Scalability – low interconnection complexity
  - Data dependencies handled implicitly by algorithmic-architecture mapping

- Highly suitable for AI/ML workloads

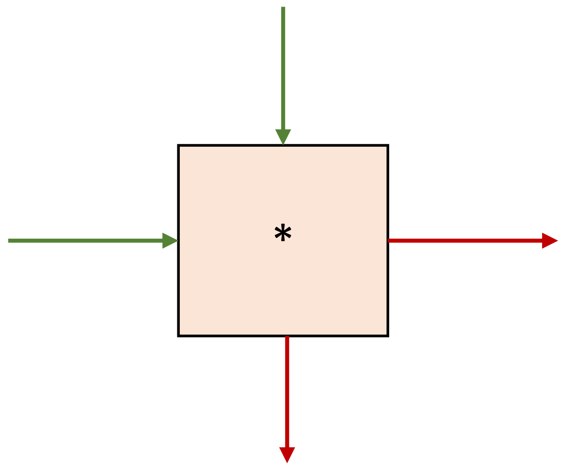
- Example Systolic Array based Devices
  - Google TPU - Google's Machine Learning Processor
  - Amazon Inferentia- Amazon's Machine Learning Processor
  - Nvidia Tensor Core – Specialized AI cores within modern Nvidia GPUs

Systolic Arrays

- Why Systolic Arrays?
- Increasingly being used in ML accelerators
- Easier to reason about performance than CPUs or GPUs

Systolic Arrays

Operation in DPU:

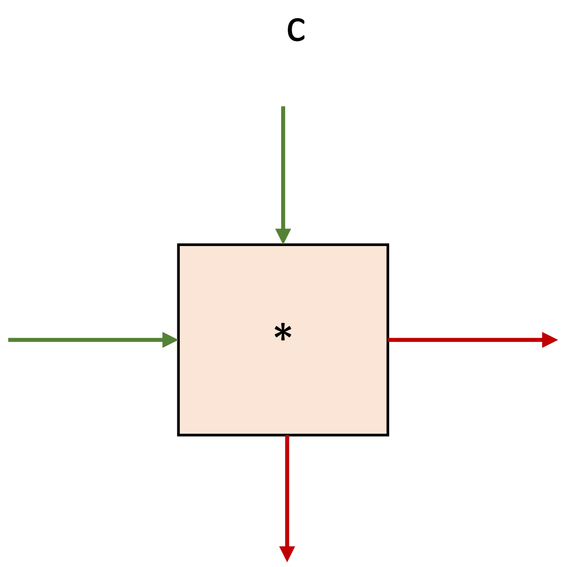


Data Processing Unit

- Compute Unit – simple computations,
- Small local memory - a few words

Systolic Arrays

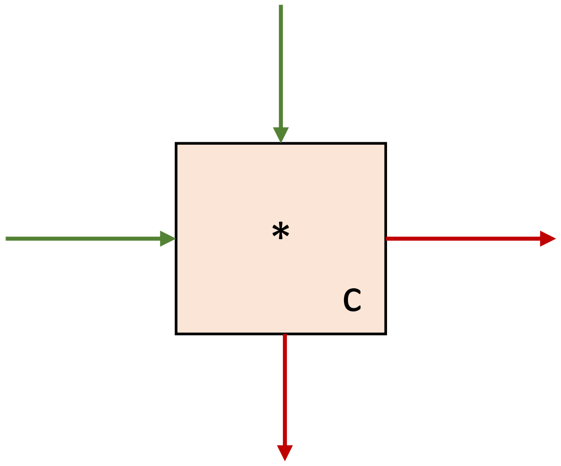
Operation in DPU:
Local Register = Top Input



Data Processing Unit
- Compute Unit – simple computations,
- Small local memory - a few words

Systolic Arrays

Operation in DPU:
Local Register = Top Input

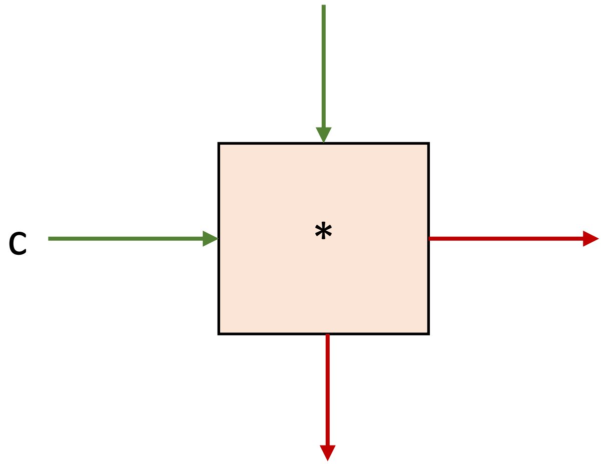


Data Processing Unit

- Compute Unit – simple computations,
- Small local memory - a few words

Systolic Arrays

Operation in DPU:
Local Register = Left Input

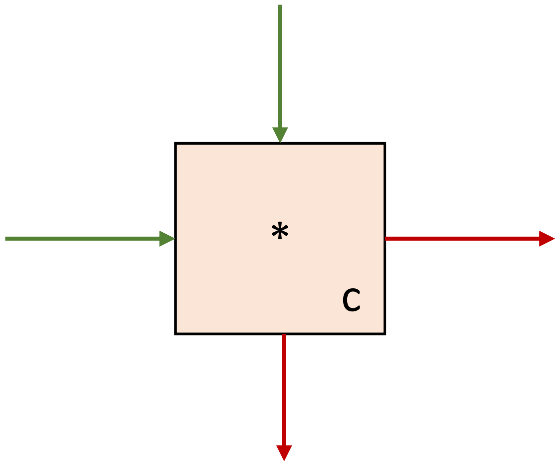


Data Processing Unit

- Compute Unit – simple computations,
- Small local memory - a few words

Systolic Arrays

Operation in DPU:
Local Register = Top Input

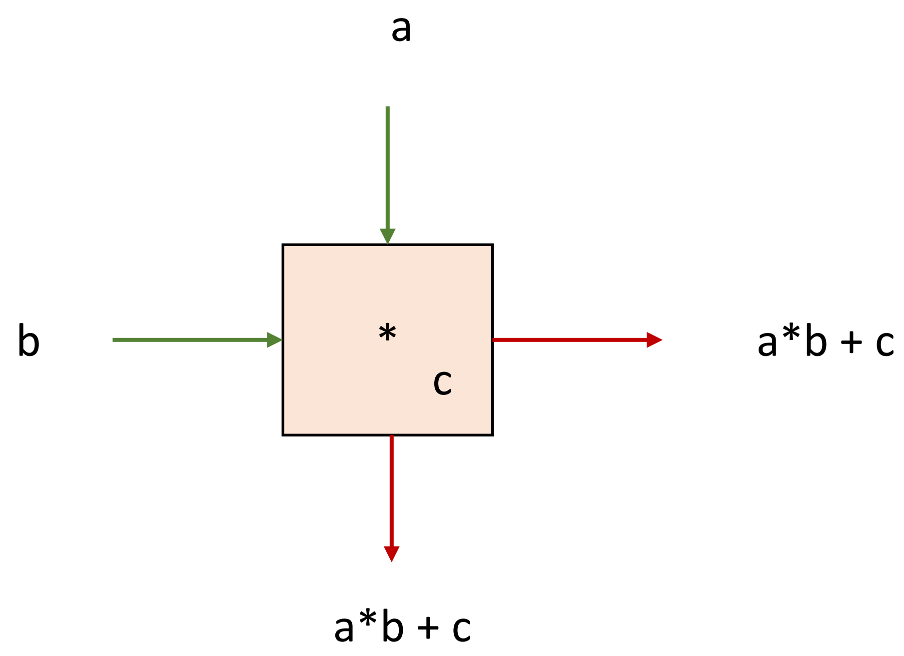


Data Processing Unit

- Compute Unit – simple computations,
- Small local memory - a few words

Systolic Arrays

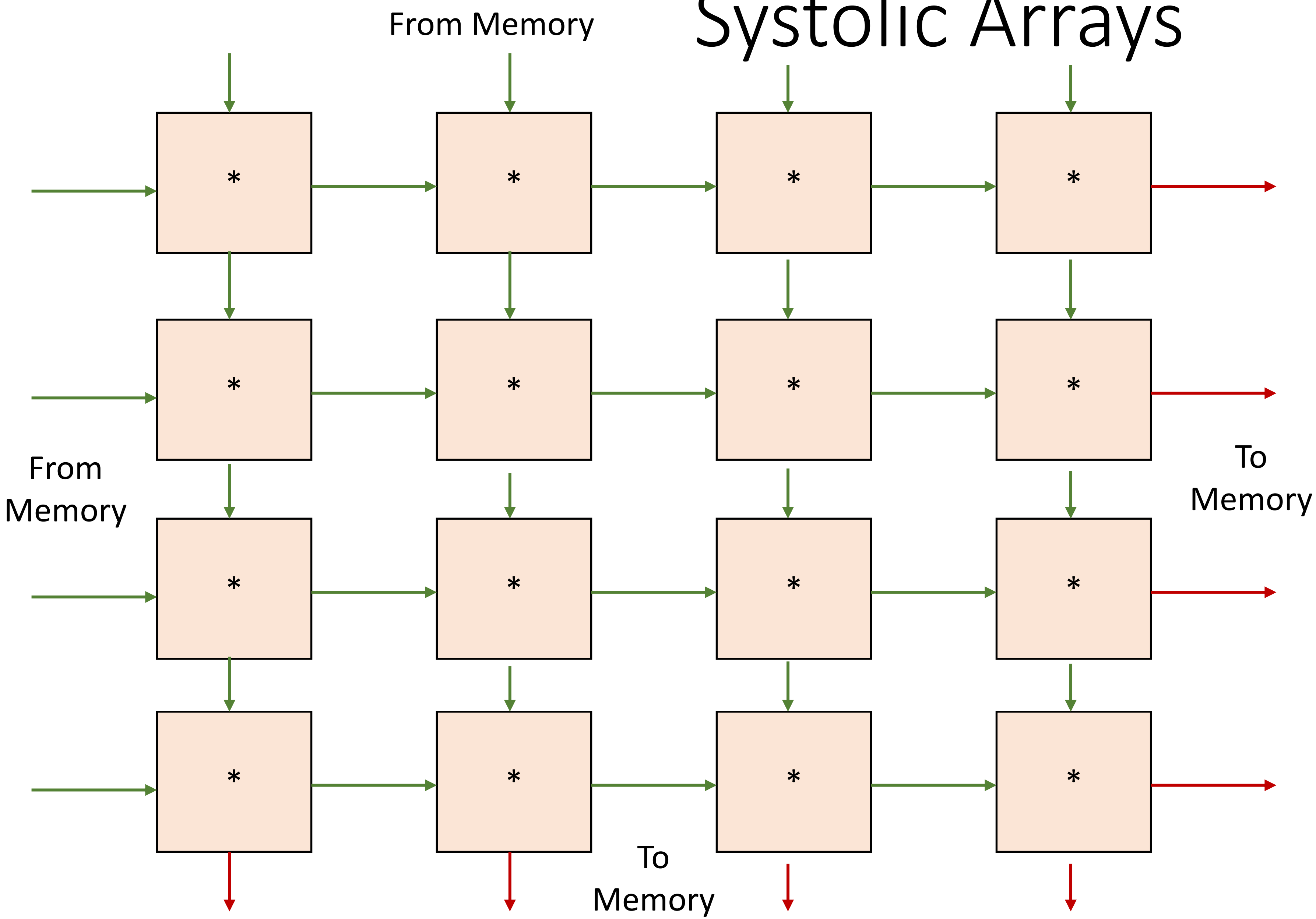
Operation in DPU:
Left-Output = Left-Input*Top-Input + Local Register
Bottom-Output = Left-Input*Top-Input + Local Register



Data Processing Unit

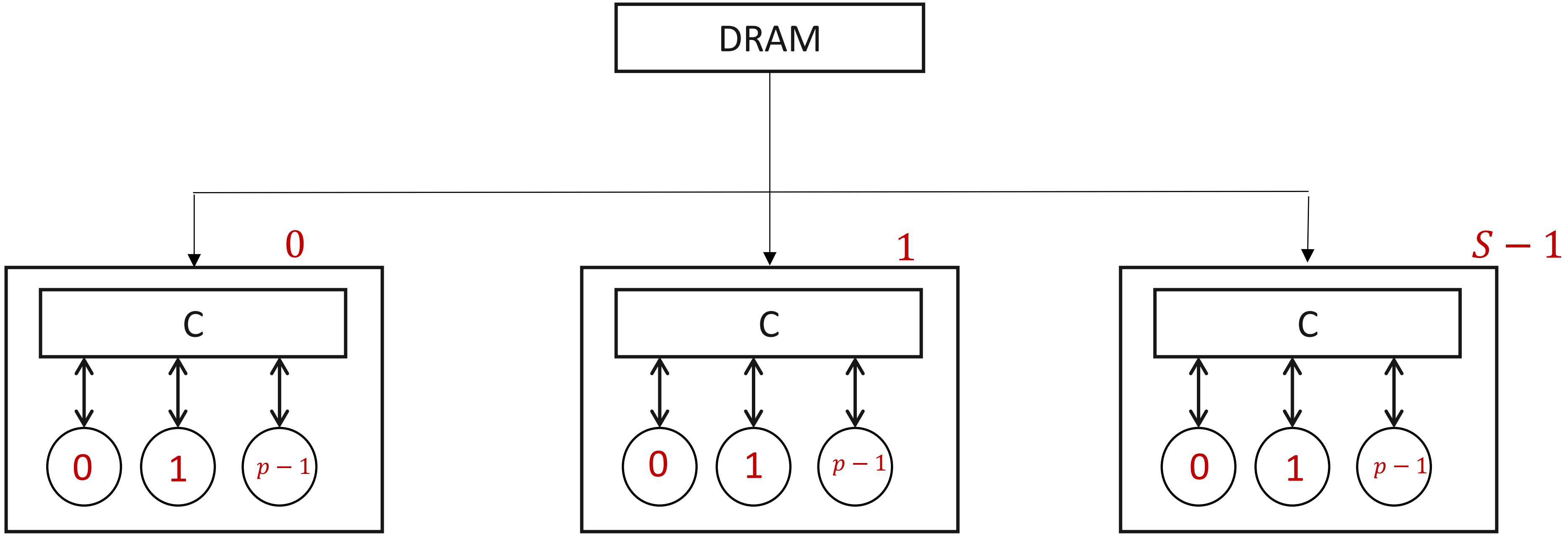
- Compute Unit – simple computations,
- Small local memory - a few words

Systolic Arrays



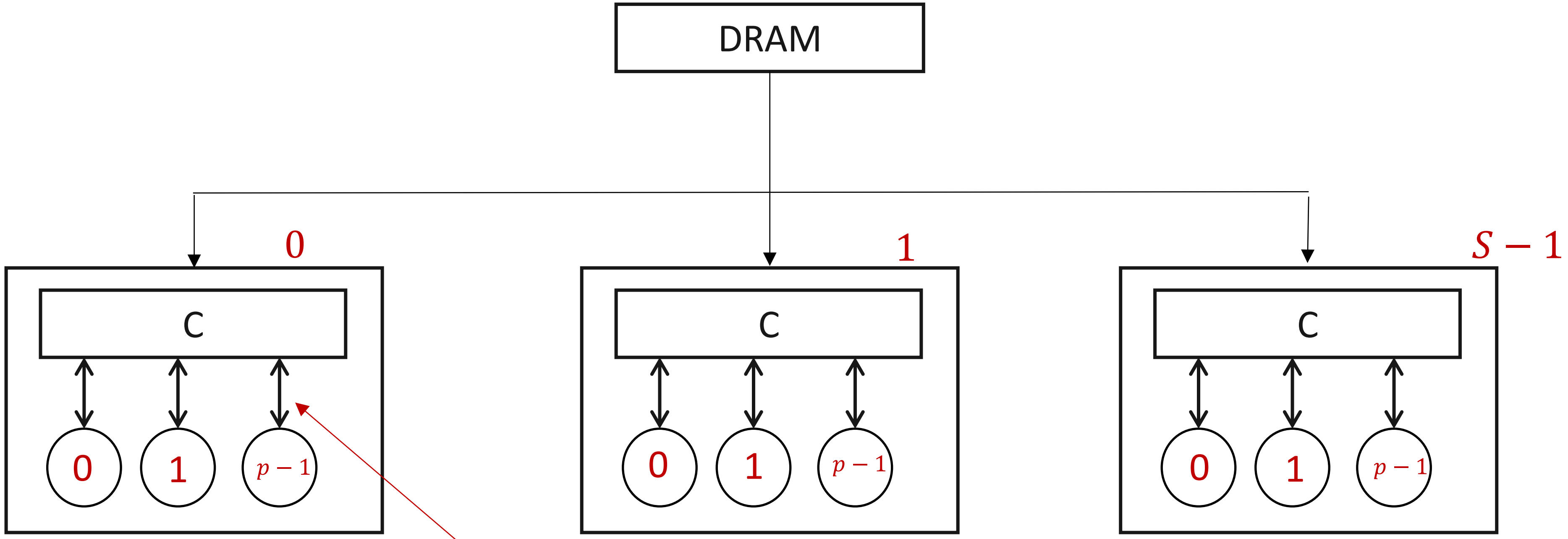
- 1, 2, or 3 dimensional array of locally connected data processing units
- **Input** received from **top** and **left**.
- **Output** from **bottom** and **right**
- Programming Systolic Arrays → determining the flow of data to achieve the desired computations

Recall: Modeling GPU Architectures



- S Blocks
- p Threads per block

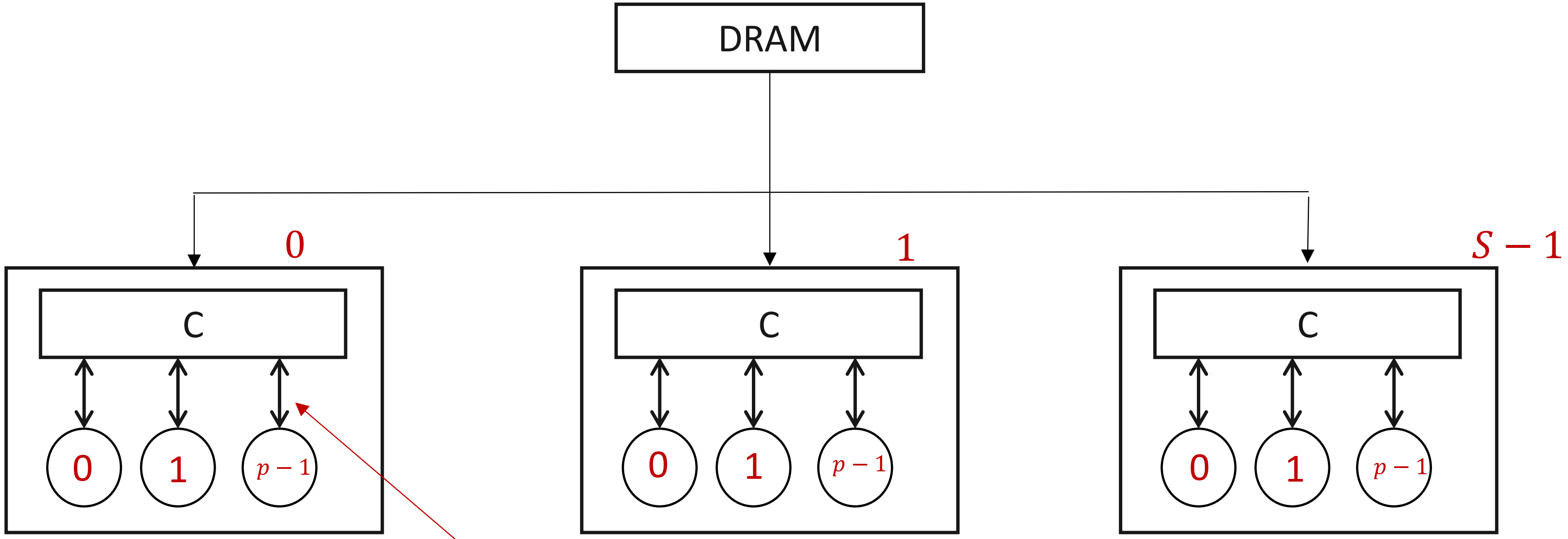
Recall: Modeling GPU Architectures



- S Blocks
- p Threads per block

For p threads, number of connections to the shared cache is p.

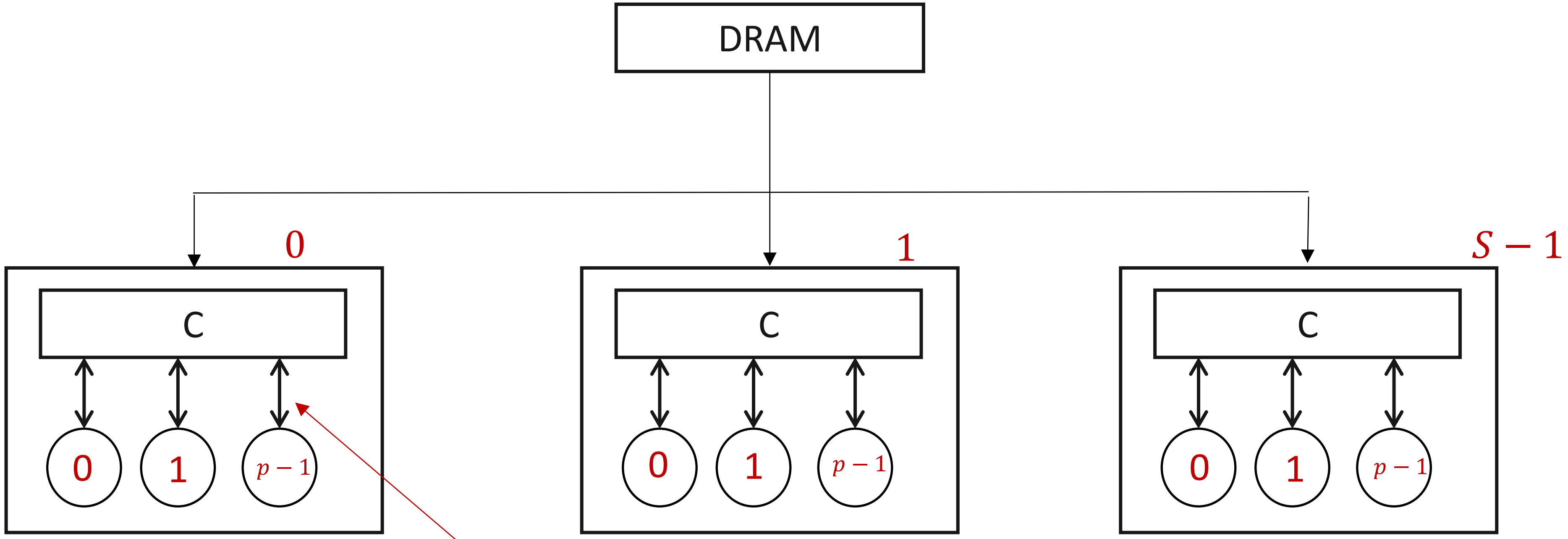
Recall: Modeling GPU Architectures



- S Blocks
- p Threads per block

For p^2 threads, the number of connections need to the shared cache is ??.

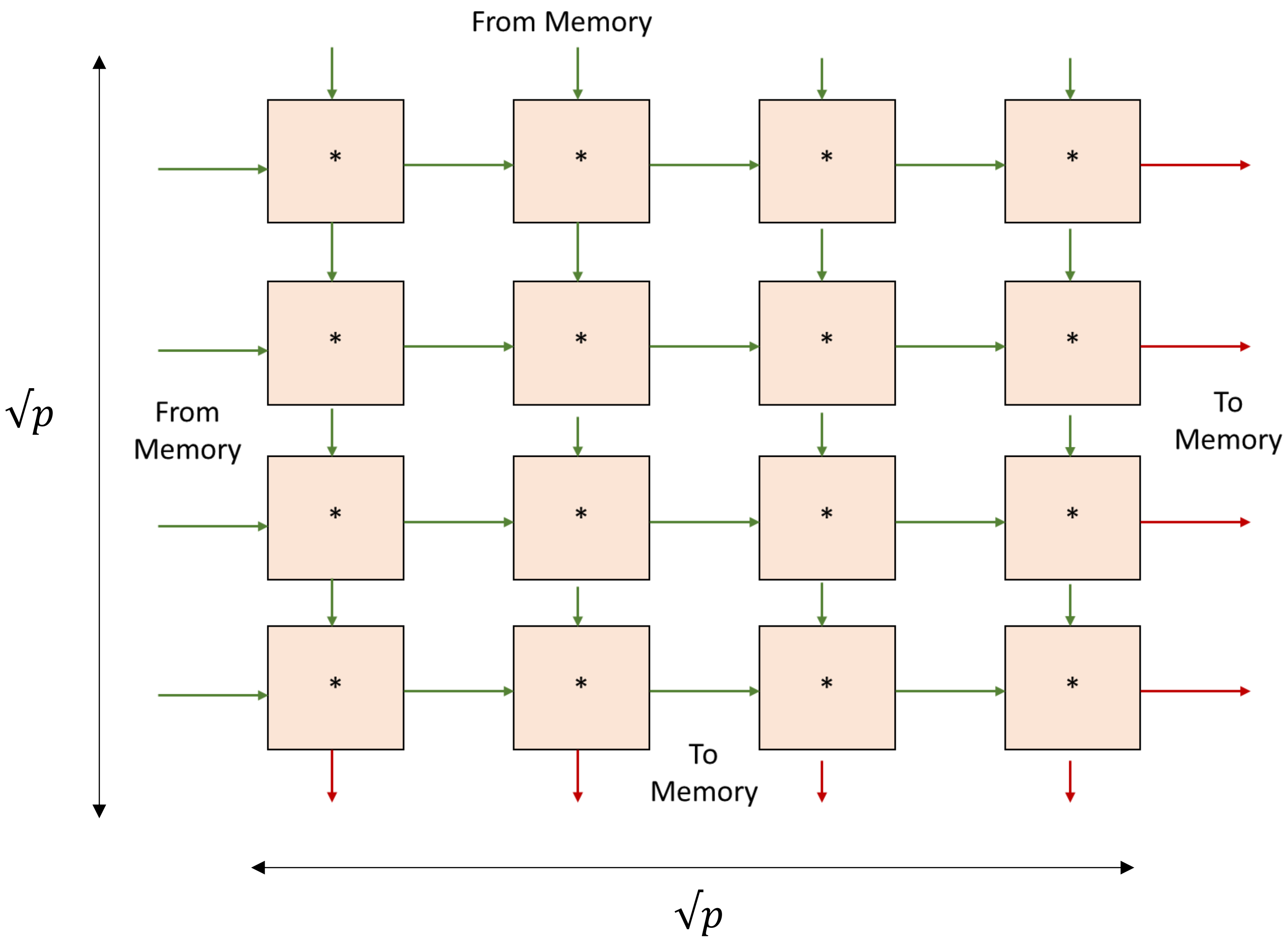
Recall: Modeling GPU Architectures



- S Blocks
- p Threads per block

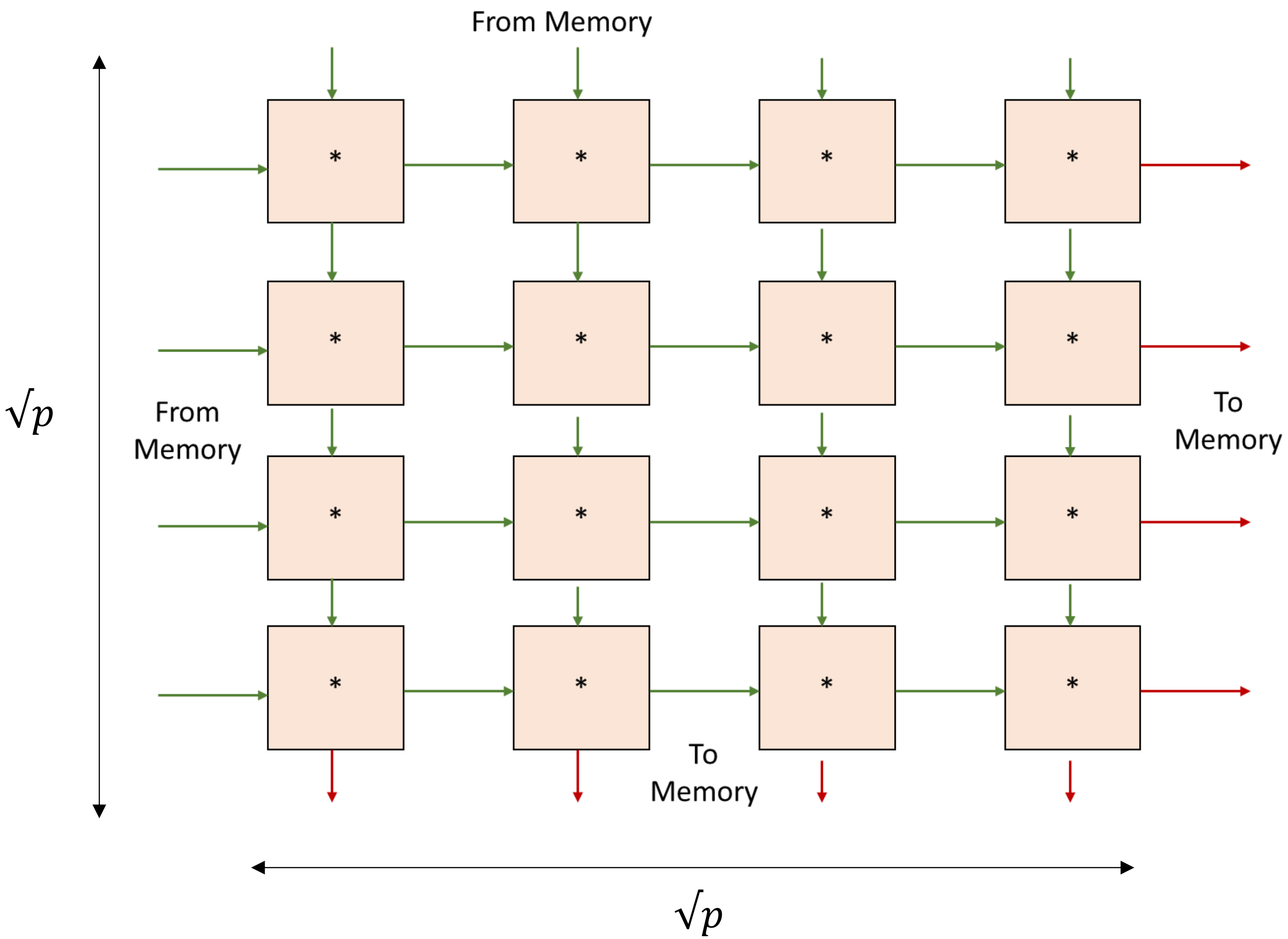
For p^2 threads, the number of connections need to the shared cache is ?? p^2

Systolic Arrays



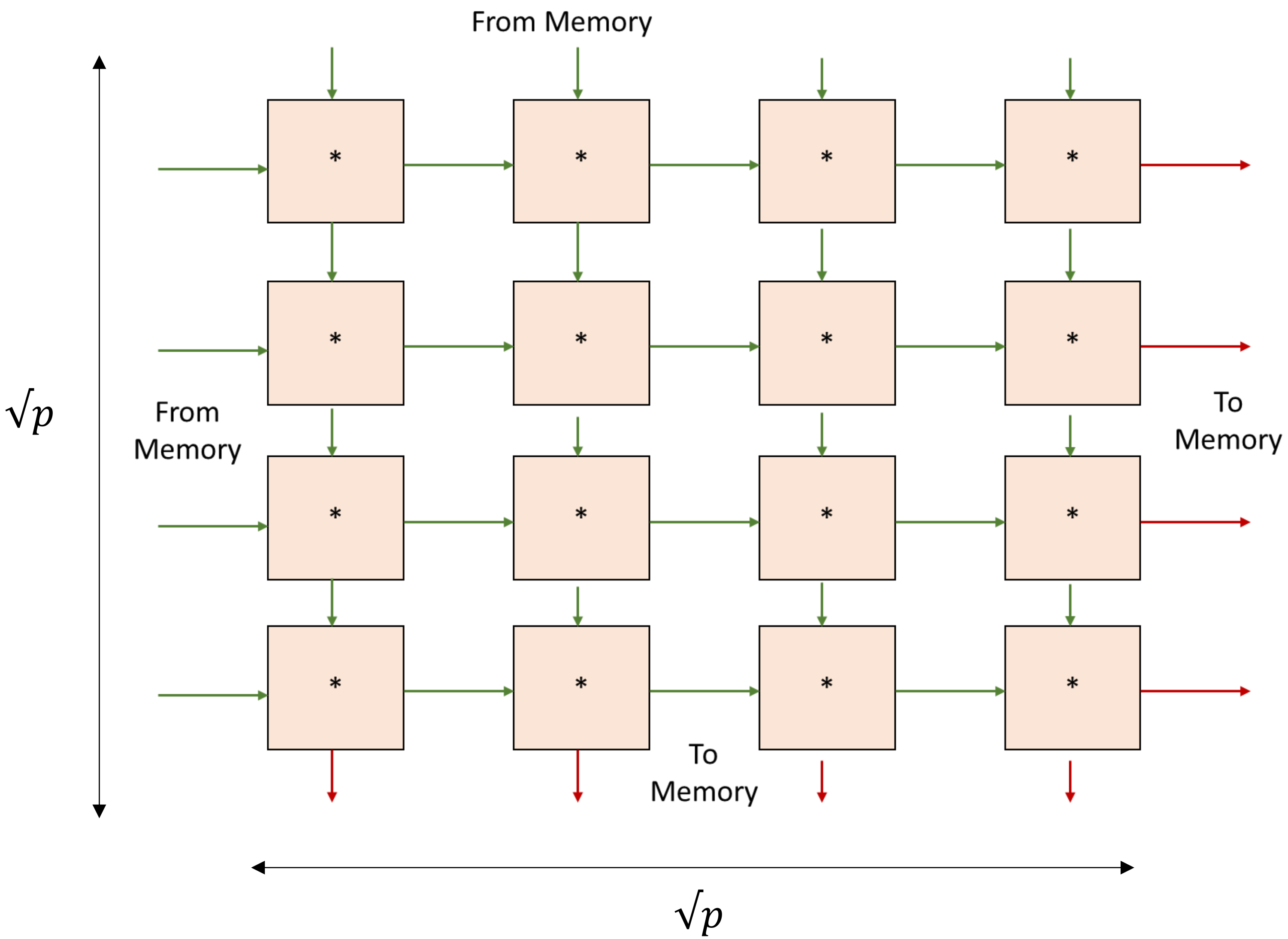
How many processor does this systolic array have?

Systolic Arrays



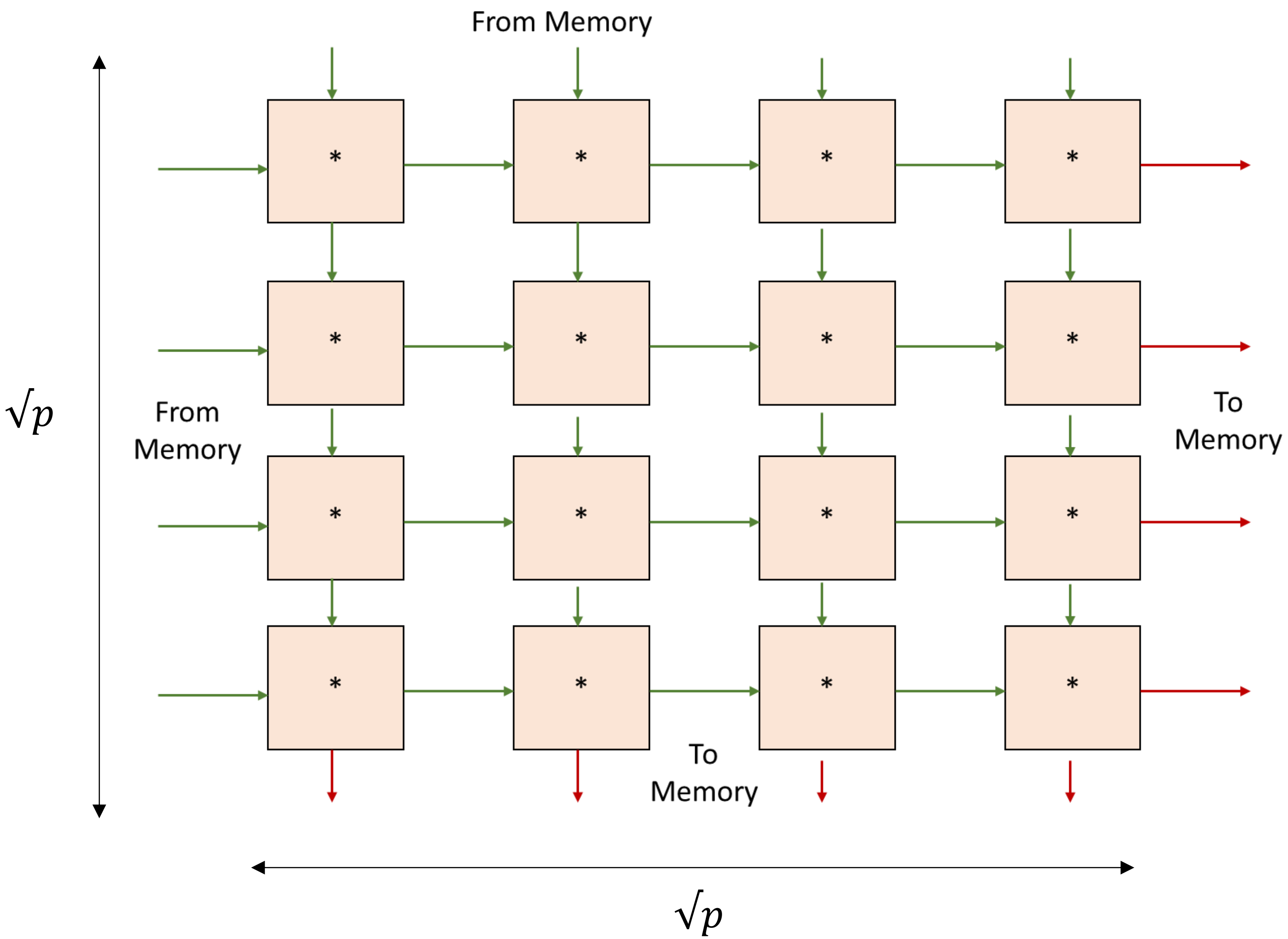
How many processor does this systolic array have? p

Systolic Arrays



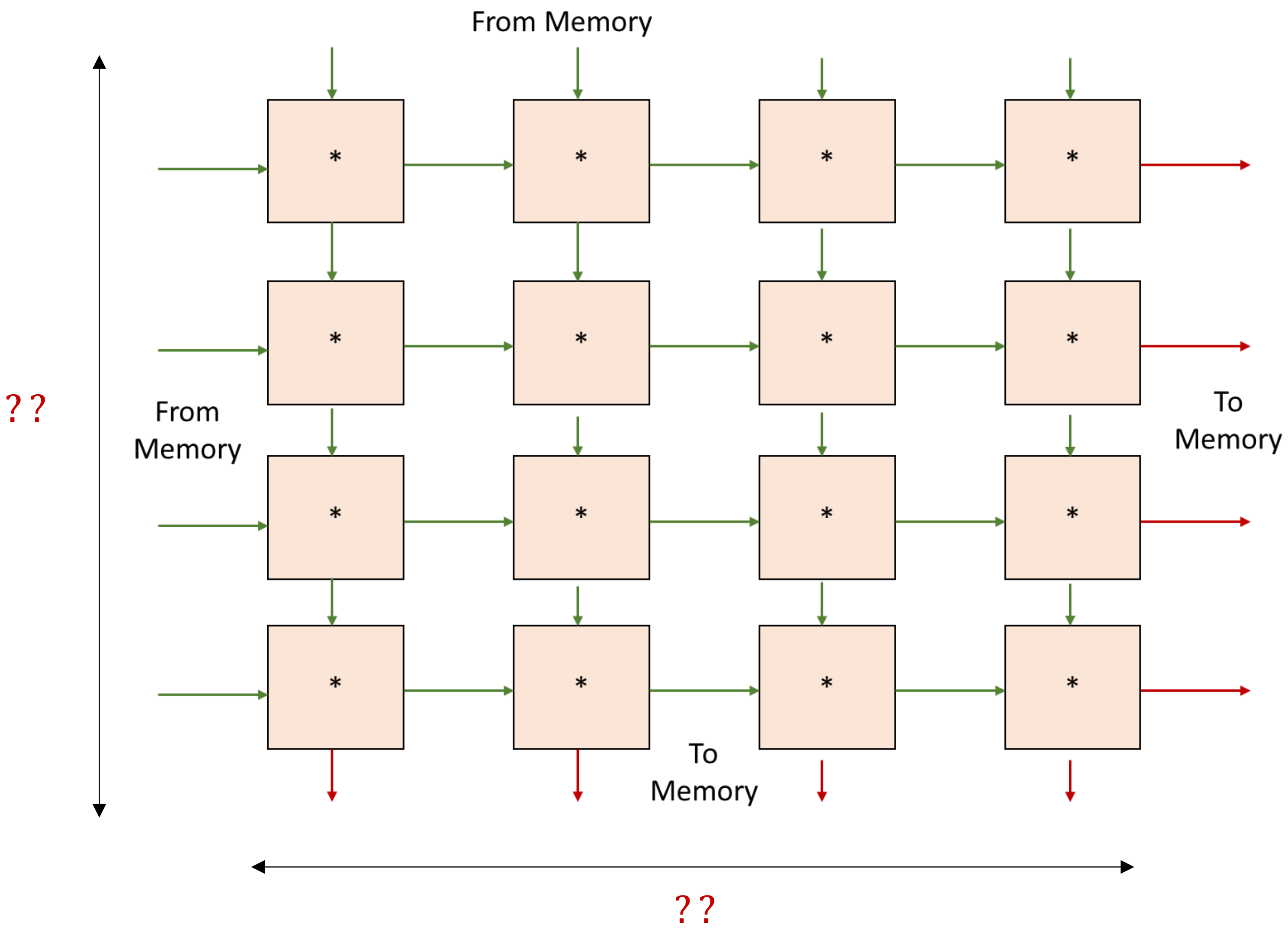
How many connections to/from memory are needed?

Systolic Arrays



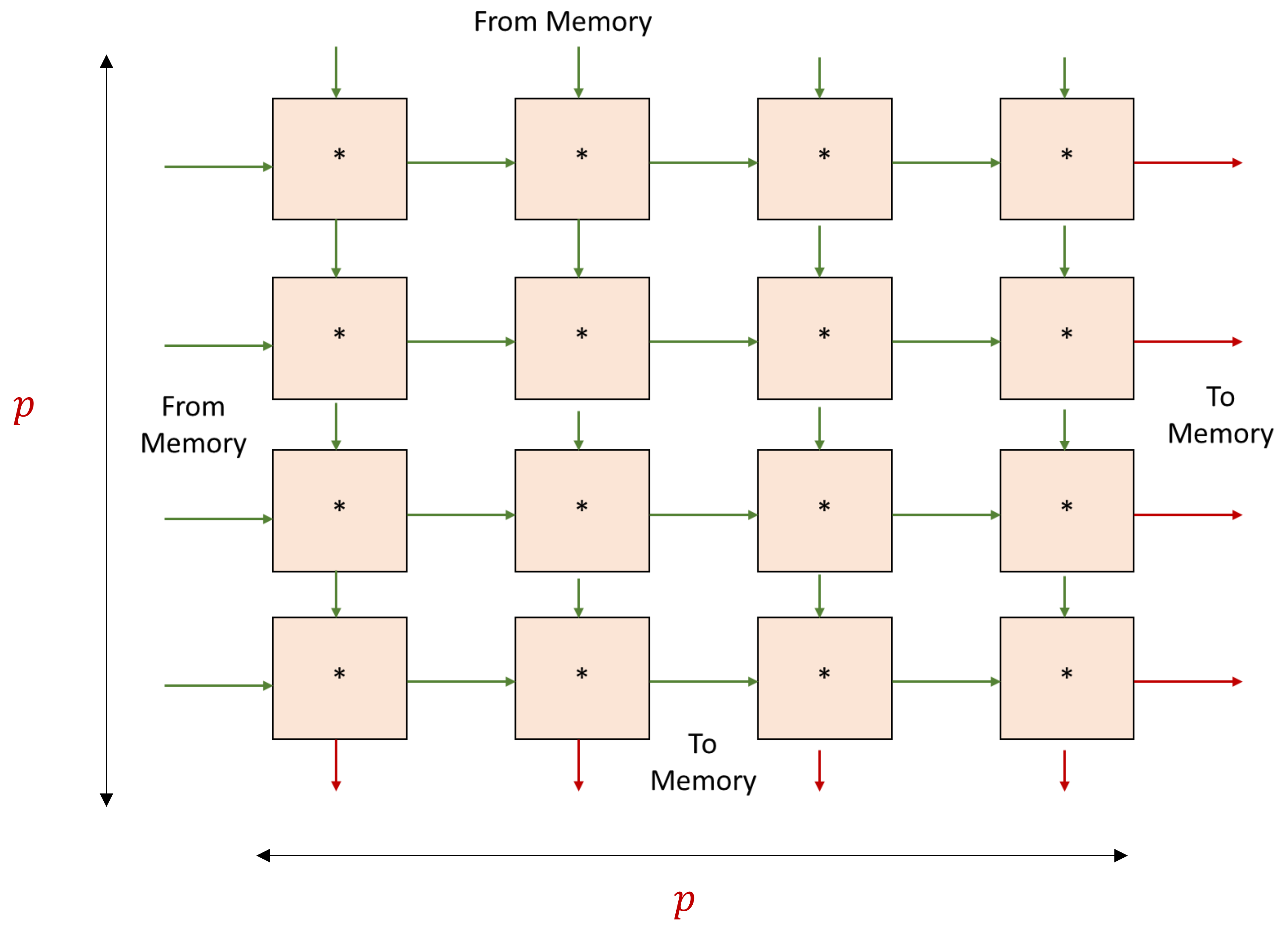
How many connections to/from memory are needed? $4\sqrt{p}$

Systolic Arrays



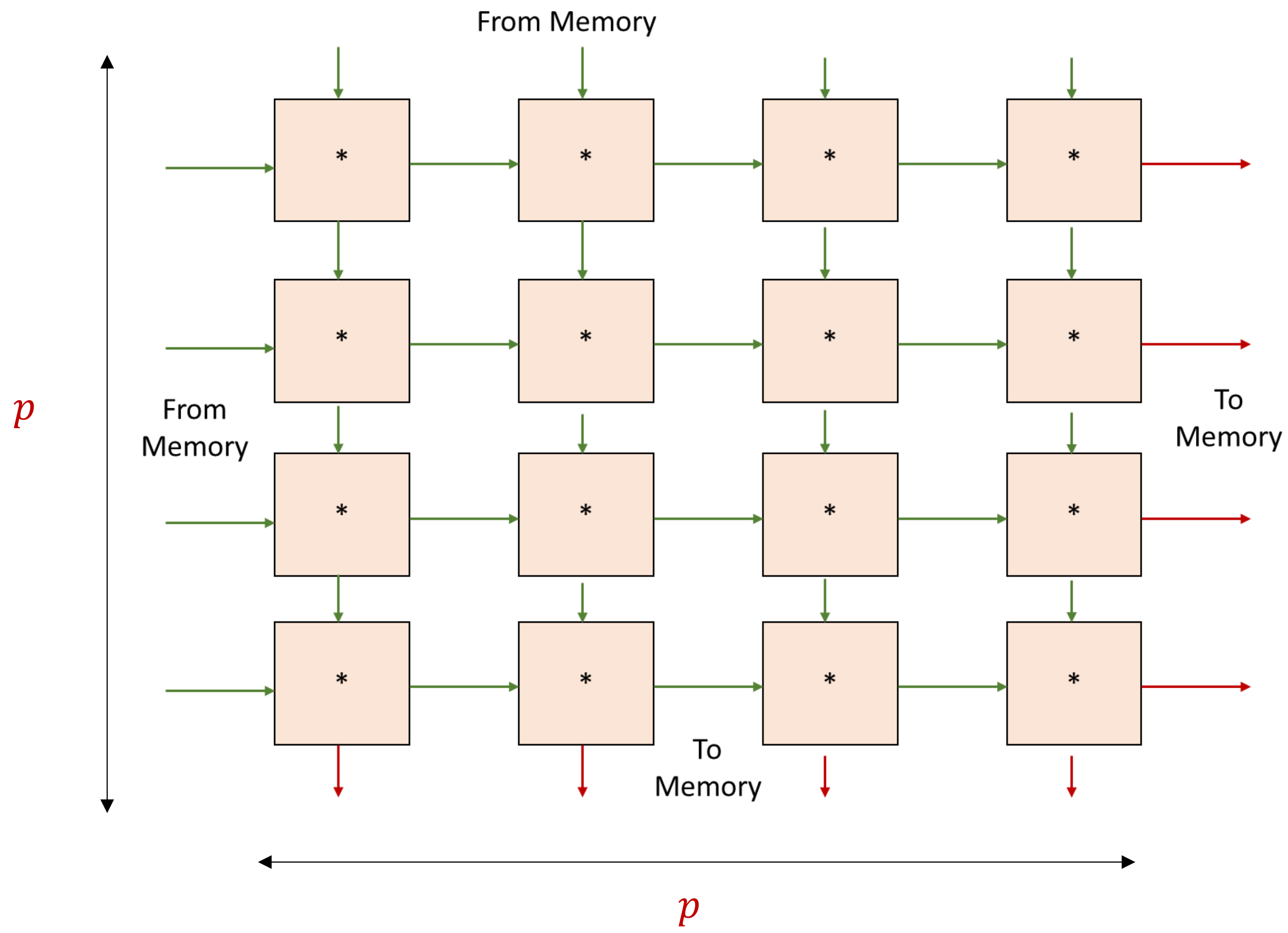
If we need p^2 processors in the systolic array, what would be the dimensions?

Systolic Arrays



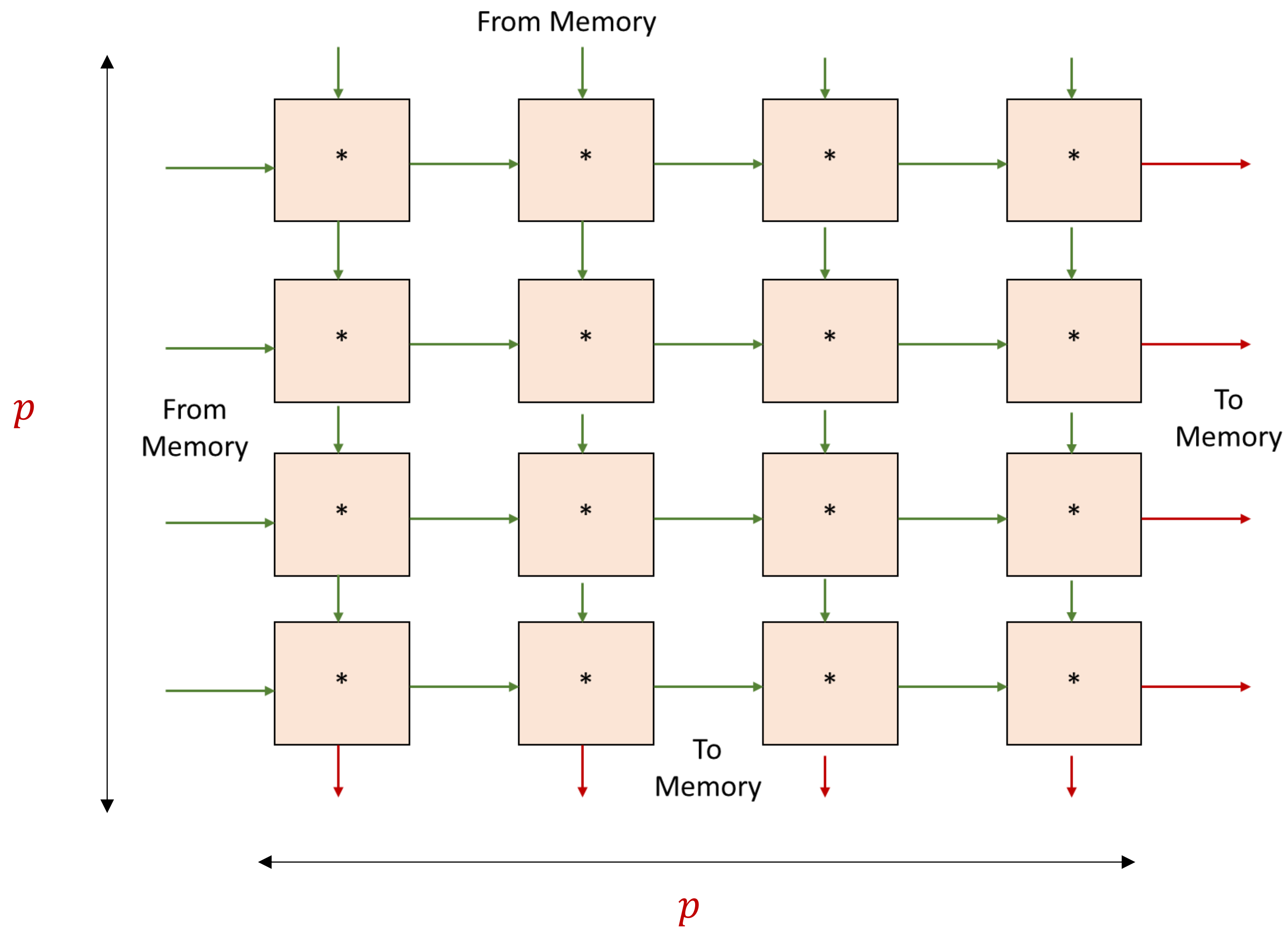
If we need p^2 processors in the systolic array, what would be the dimensions? $p \times p$

Systolic Arrays



If we need p^2 processors in the systolic array, how many connections to/from memory are needed?

Systolic Arrays



If we need p^2 processors in the systolic array, how many connections to/from memory are needed? $4p$

Systolic Arrays vs GPUs

- GPUs
  - Each processor connected to memory: memory complexity grows linearly with the number of processors
  - Each processor executes the same instructions on different data – totally independent
- Systolic Arrays
  - Only $\sqrt{p}$ processors connected to memory (where p is the number of processors): memory complexity grows linearly
  - Processors still execute the same instructions, but input of a processors is usually output of a previous processor – highly dependent

Systolic Arrays vs GPUs

- Designing algorithms for GPUs – Data parallel thinking
  - Determine output vs input vs hybrid partitioning
  - Determine the set of instructions each processor needs to execute on different data to achieve results

- Designing algorithms for systolic arrays – A bit more involved
  - Need to think what data should reach which processor at what time to achieve desired results
  - Still every processor works in lockstep, so determine the set of instructions each processor needs to execute

So Why Systolic Arrays?

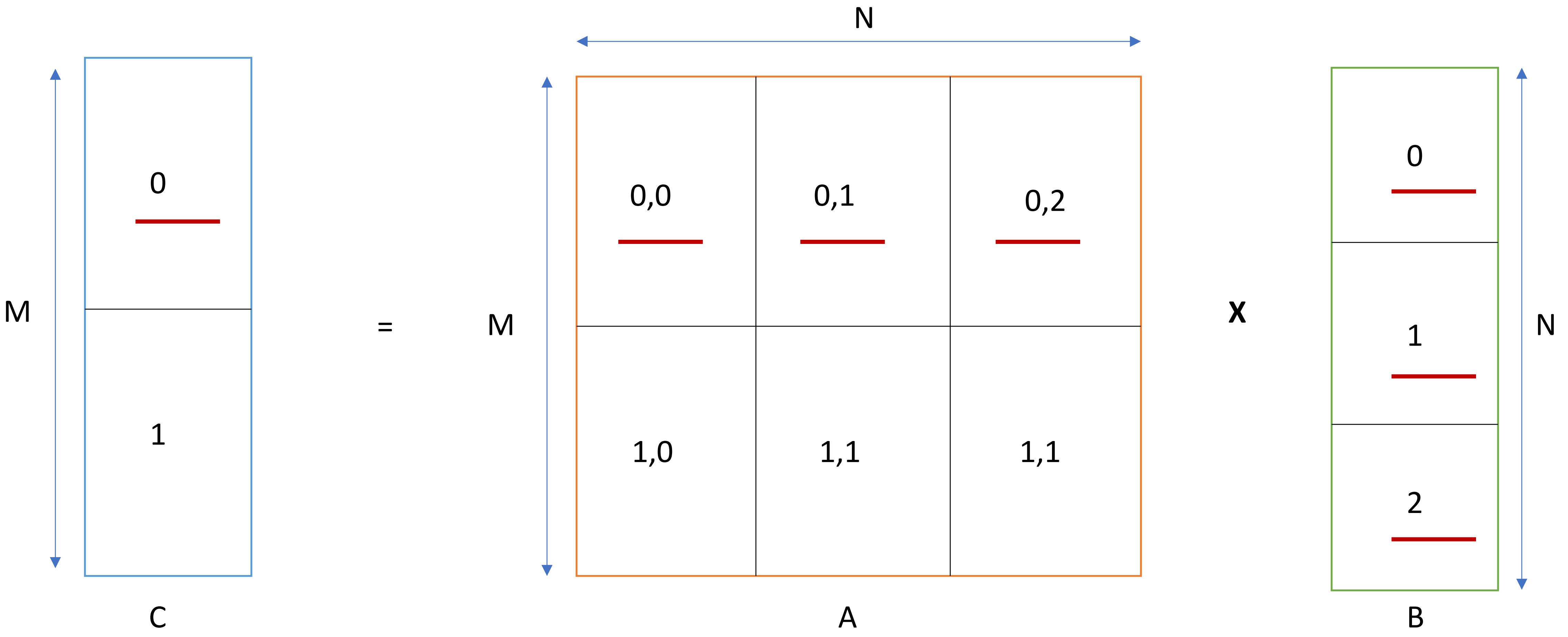
- Reduced number of connections make systolic arrays some of the best architectures for dense matrix operations – the most important kernel in AI/ML workloads
- Work Optimal algorithms already known for problems such as matrix-multiplication
  - So, we don't need to program from scratch

Matrix Vector using 1D Systolic Arrays

- Steps
  1. Load Vector into systolic array
  2. Stream in rows of matrices vertically
  3. Collect output horizontally

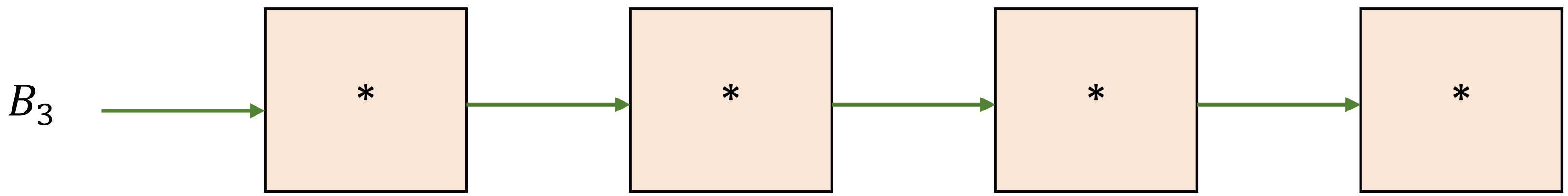
Recall: Matrix Vector Multiplication

C[i] = ∑A[i][k] * B[k], $N = 4$ in the following example



Matrix Vector using 1D Systolic Arrays

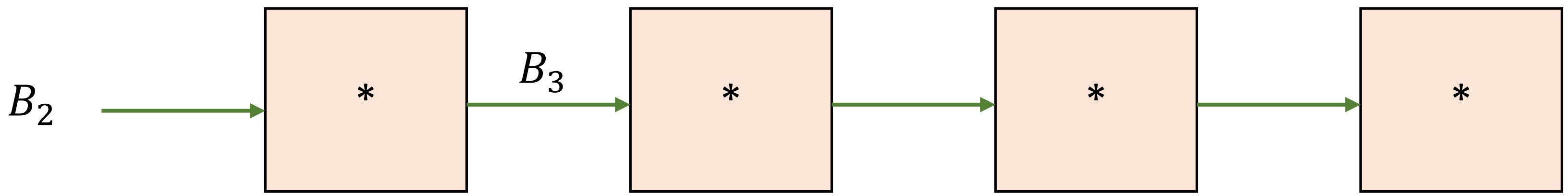
Store Vector B into the systolic array local memory



Cycle 0 – Input Available

Matrix Vector using 1D Systolic Arrays

Store Vector B into the systolic array local memory

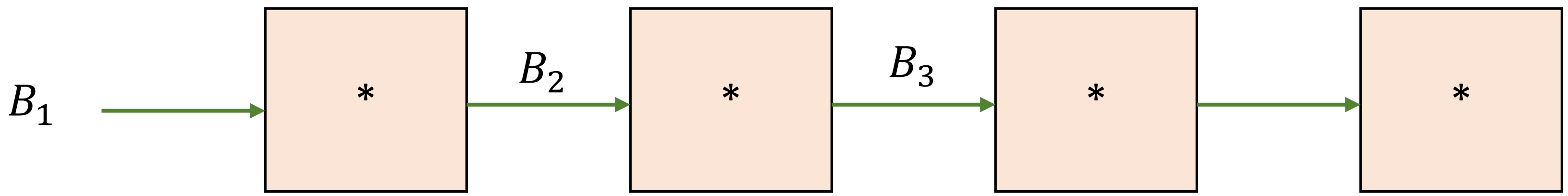


Cycle 1

Operation in DPU: Output = Input

Matrix Vector using 1D Systolic Arrays

Store Vector B into the systolic array local memory

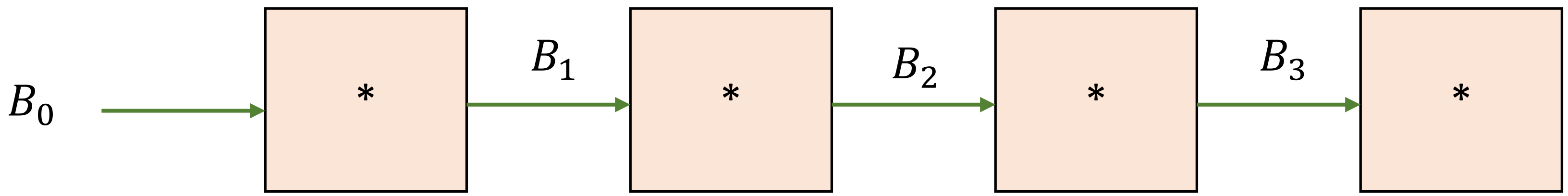


Cycle 2

Operation in DPU: Output = Input

Matrix Vector using 1D Systolic Arrays

Store Vector B into the systolic array local memory

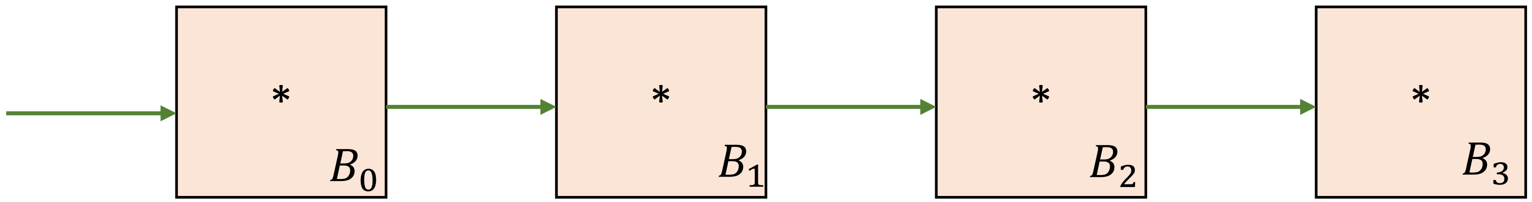


Cycle 3

Operation in DPU: Output = Input

Matrix Vector using 1D Systolic Arrays

Store Vector B into the systolic array local memory

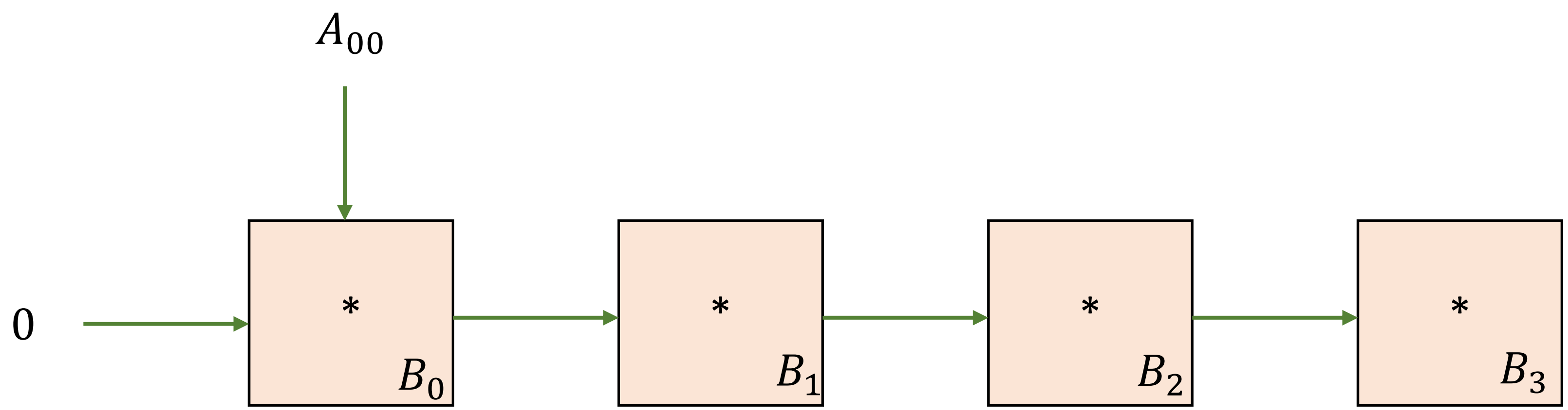


Cycle 4

Operation in DPU: Local Register = Input

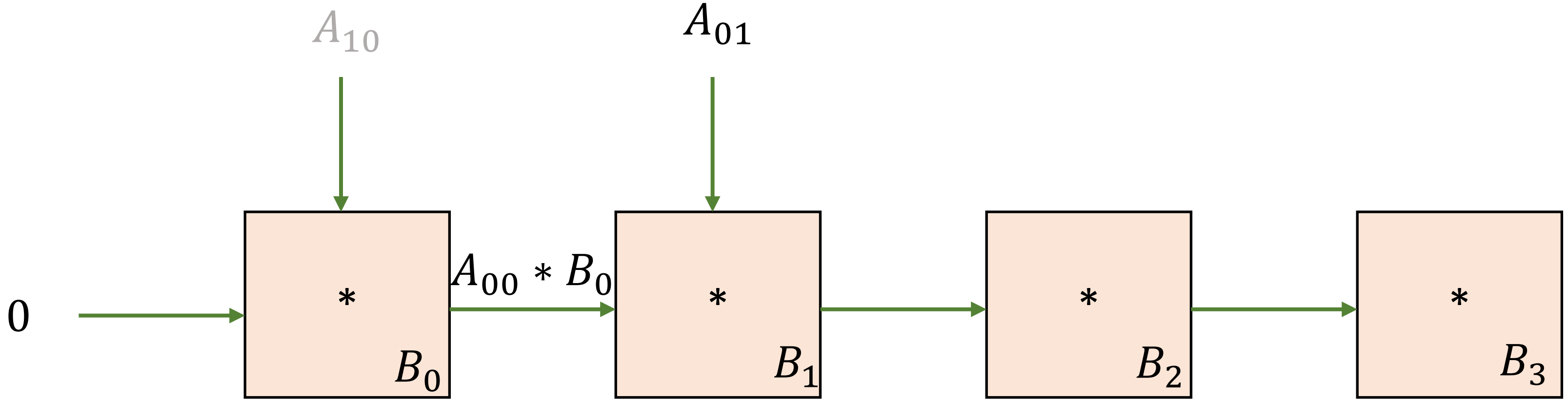
Matrix Vector using 1D Systolic Arrays

Stream Rows of A



Cycle 5

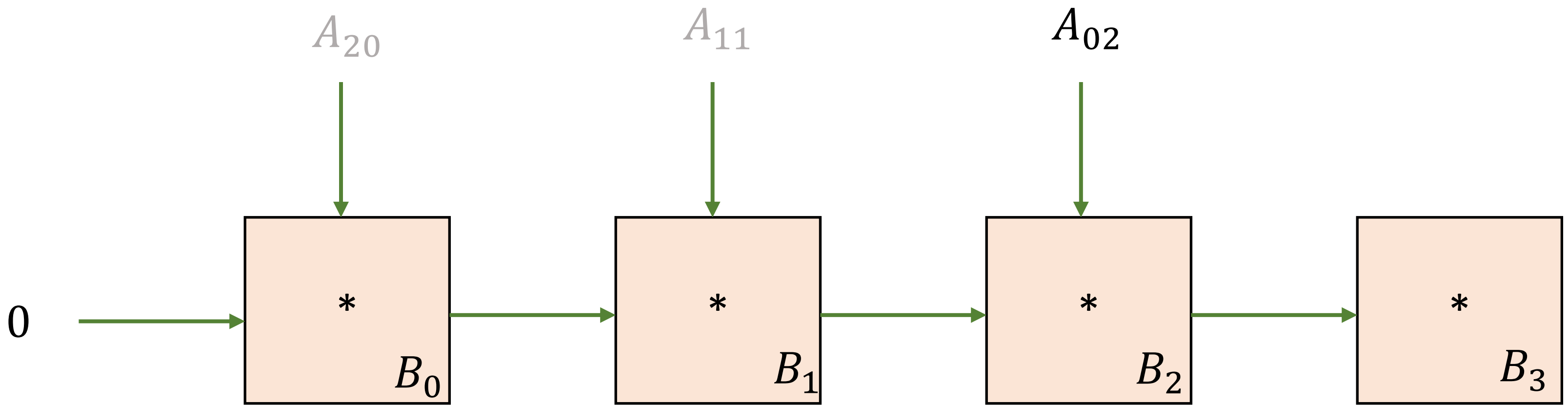
Matrix Vector using 1D Systolic Arrays



Cycle 6

Operation in DPU: Output = Top Input*Local Register + Left Input

Matrix Vector using 1D Systolic Arrays

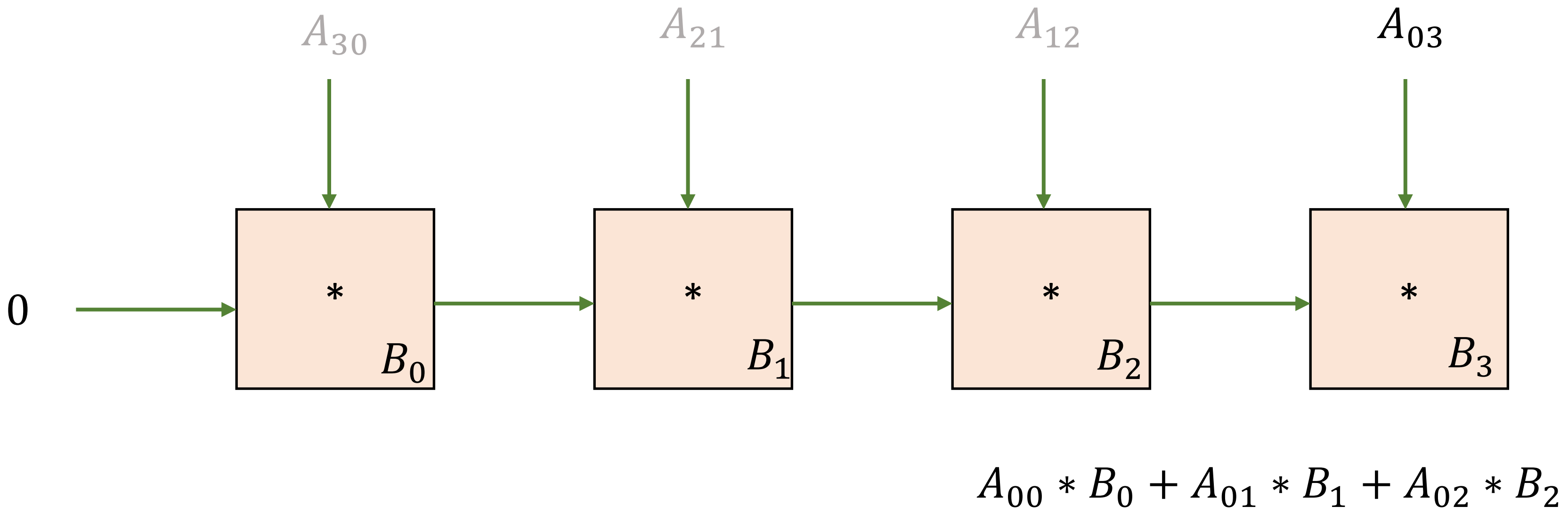


$A_{00} * B_0 + A_{01} * B_1$

Cycle 7

Operation in DPU: Output = Top Input*Local Register + Left Input

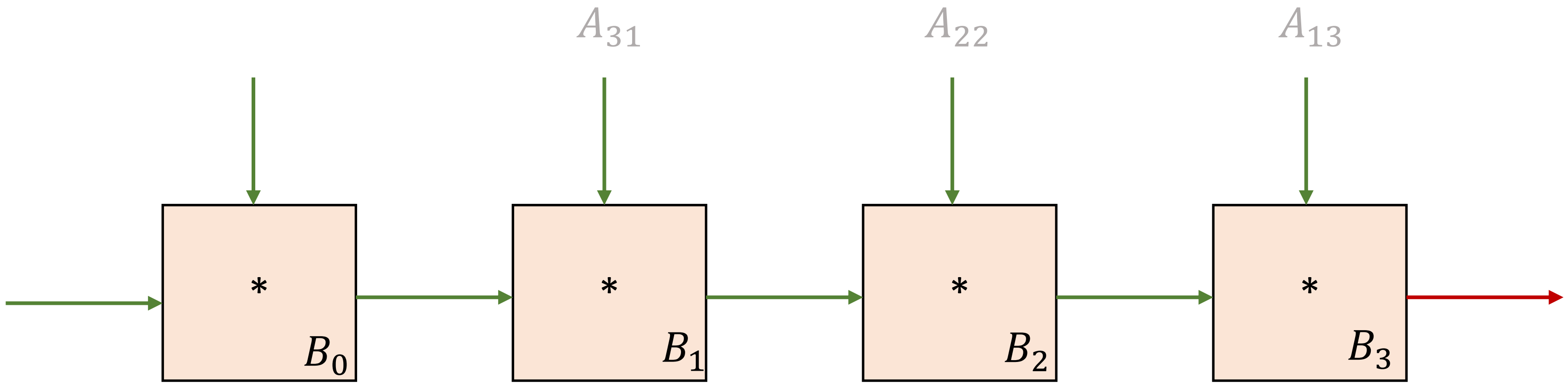
Matrix Vector using 1D Systolic Arrays



Cycle 8

Operation in DPU: Output = Top Input*Local Register + Left Input

Matrix Vector using 1D Systolic Arrays

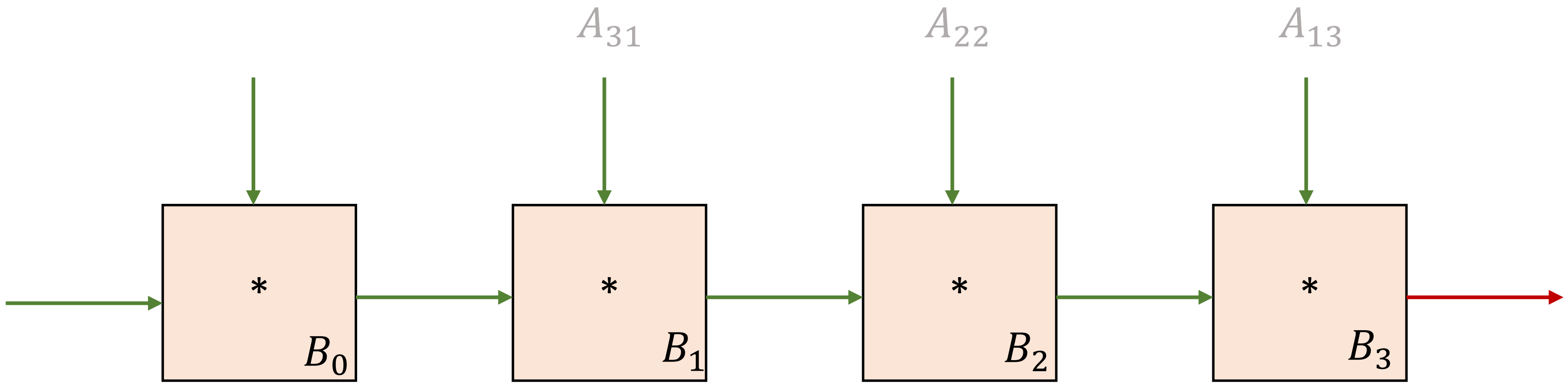


$A_{00} * B_0 + A_{01} * B_1 + A_{02} * B_2 + A_{03} * B_3$

Cycle 9

Operation in DPU: Output = Top Input*Local Register + Left Input

Matrix Vector using 1D Systolic Arrays

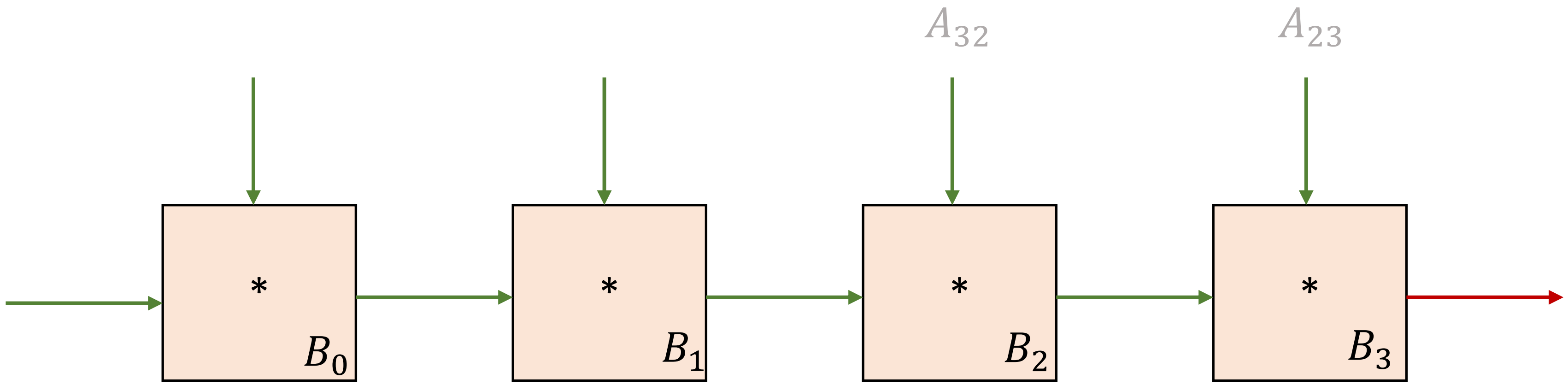


$A_{00} * B_0 + A_{01} * B_1 + A_{02} * B_2 + A_{03} * B_3 = C_0$

Cycle 9

Operation in DPU: Output = Top Input*Local Register + Left Input

Matrix Vector using 1D Systolic Arrays

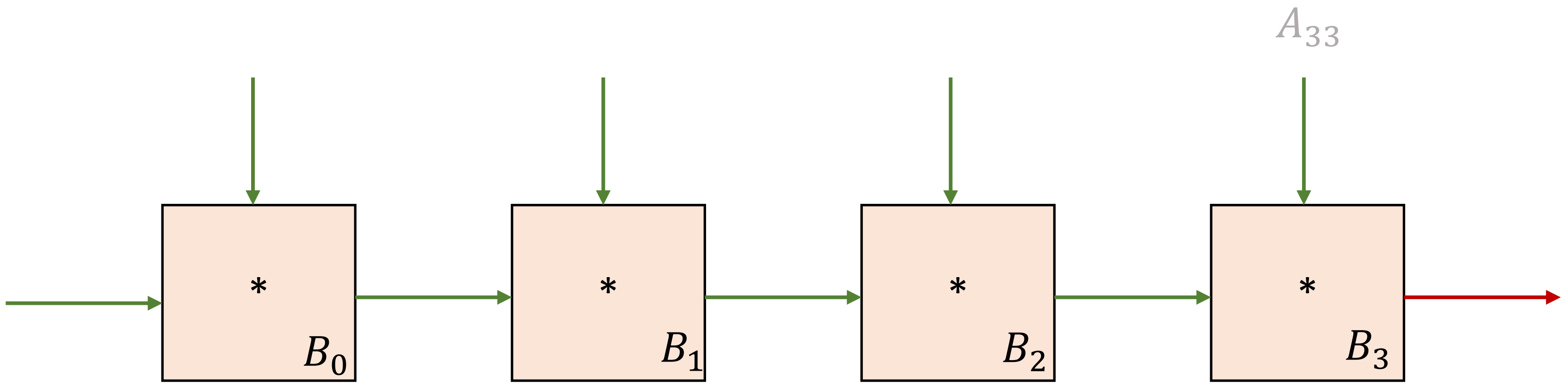


$A_{10} * B_0 + A_{11} * B_1 + A_{12} * B_2 + A_{13} * B_3 = C_1$

Cycle 10

Operation in DPU: Output = Top Input*Local Register + Left Input

Matrix Vector using 1D Systolic Arrays

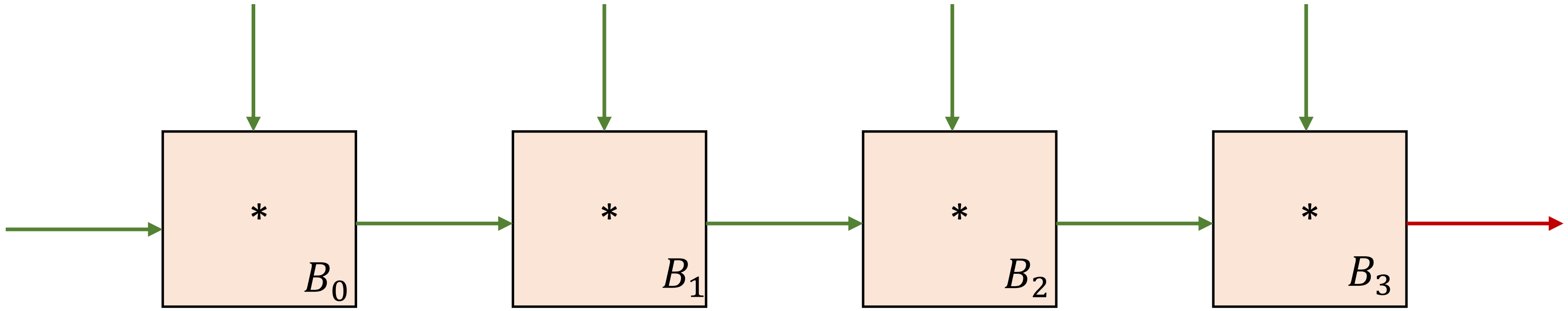


$A_{20} * B_0 + A_{21} * B_1 + A_{22} * B_2 + A_{23} * B_3 = C_2$

Cycle 11

Operation in DPU: Output = Top Input*Local Register + Left Input

Matrix Vector using 1D Systolic Arrays



$A_{30} * B_0 + A_{31} * B_1 + A_{32} * B_2 + A_{33} * B_3 = C_3$

Cycle 12

Operation in DPU: Output = Top Input*Local Register + Left Input

Matrix Vector using 1D Systolic Arrays

- Assume: Vector size n, Matrix size $n \times n$, number of processors n
- Steps
  1. Load Vector into systolic array - n cycles
  2. Stream in rows of matrices vertically
     - Load A_{xi} in cell i in cycle $(x + i + 1)$ + n
  3. Collect output horizontally
     - 1 output produced each cycle after initial latency - $2n + 1$
     - Output produced for $n \; - 1$ cycles
     - Total time: $3n$ cycles

How??

Matrix Vector using 1D Systolic Arrays

- Assume: Vector size n, Matrix size $n \times n$, number of processors n
- Steps
  1. Load Vector into systolic array - n cycles
  2. Stream in rows of matrices vertically
     - Load A_{xi} in cell i in cycle $(x + i + 1)$ + n
  3. Collect output horizontally
     - 1 output produced each cycle after initial latency - $2n + 1$
     - Output produced for $n - 1$ cycles
     - Total time: $3n$ cycles

n cycles for loading the vector, $n + 1$ to produce the first output

Matrix Vector using 1D Systolic Arrays

- Assume: Vector size n, Matrix size $n \times n$, number of processors n
- Steps
  1. Load Vector into systolic array - n cycles
  2. Stream in rows of matrices vertically
     - Load A_{xi} in cell i in cycle $(x + i + 1) + n$
  3. Collect output horizontally
     - 1 output produced each cycle after initial latency - $2n + 1$
     - Output produced for $n - 1$ cycles
     - Total time: $3n$ cycles

Why?

Matrix Vector using 1D Systolic Arrays

- Assume: Vector size n, Matrix size $n \times n$, number of processors n
- Steps
  1. Load Vector into systolic array - n cycles
  2. Stream in rows of matrices vertically
     - Load A_{xi} in cell i in cycle $(x + i + 1)$ + n
  3. Collect output horizontally
     - 1 output produced each cycle after initial latency - $2n + 1$
     - Output produced for $n \ - 1$ cycles
     - Total time: $3n$ cycles

$n - 1$ more outputs need to be produced

Matrix Vector using 1D Systolic Arrays

- MV using $p < n$ sized systolic array
- Repeat the previous steps $\lceil \frac{n}{p} \rceil$ times
- Total time - $3p \times \lceil \frac{n}{p} \rceil$

Ungraded HW Assignment: Think about how this is happening. Partial results will again need to be routed back. How do we do that? (notice the 0 that we had as the left input of leftmost DPU) Does the total time look correct?

Matrix Vector using 1D Systolic Arrays

- MV num ops = n^2
- MV Total time on systolic arrays (with n processors) = $3n$
- Number of processors = n
- Cost of the Algorithm = $n \times 3n = O(n^2)$,
- Is this cost optimal?

Matrix Vector using 1D Systolic Arrays

- MV num ops = n^2
- MV Total time on systolic arrays (with n processors) = $3n$
- Number of processors = n
- Cost of the Algorithm = $n \times 3n = O(n^2)$,
- Is this cost optimal? Yes. Serial complexity = parallel complexity

Matrix Multiplication using 2D Systolic Arrays

- $A\ - m \times k, B\ - k \times n$
- $p \times p$ systolic array
- Key Idea:
  - Each row of the systolic array perform 1D matrix vector multiplication.
  - Row i responsible for Column i of B matrix
  - What row/column of the output matrix C will Row i of the systolic array produce?

Matrix Multiplication using 2D Systolic Arrays

- $A - m \times k, B - k \times n$
- $p \times p$ systolic array
- Key Idea:
  - Each row of the systolic array perform 1D matrix vector multiplication.
  - Row i responsible for Column i of B matrix
  - What row/column of the output matrix C will Row i of the systolic array produce? Column i

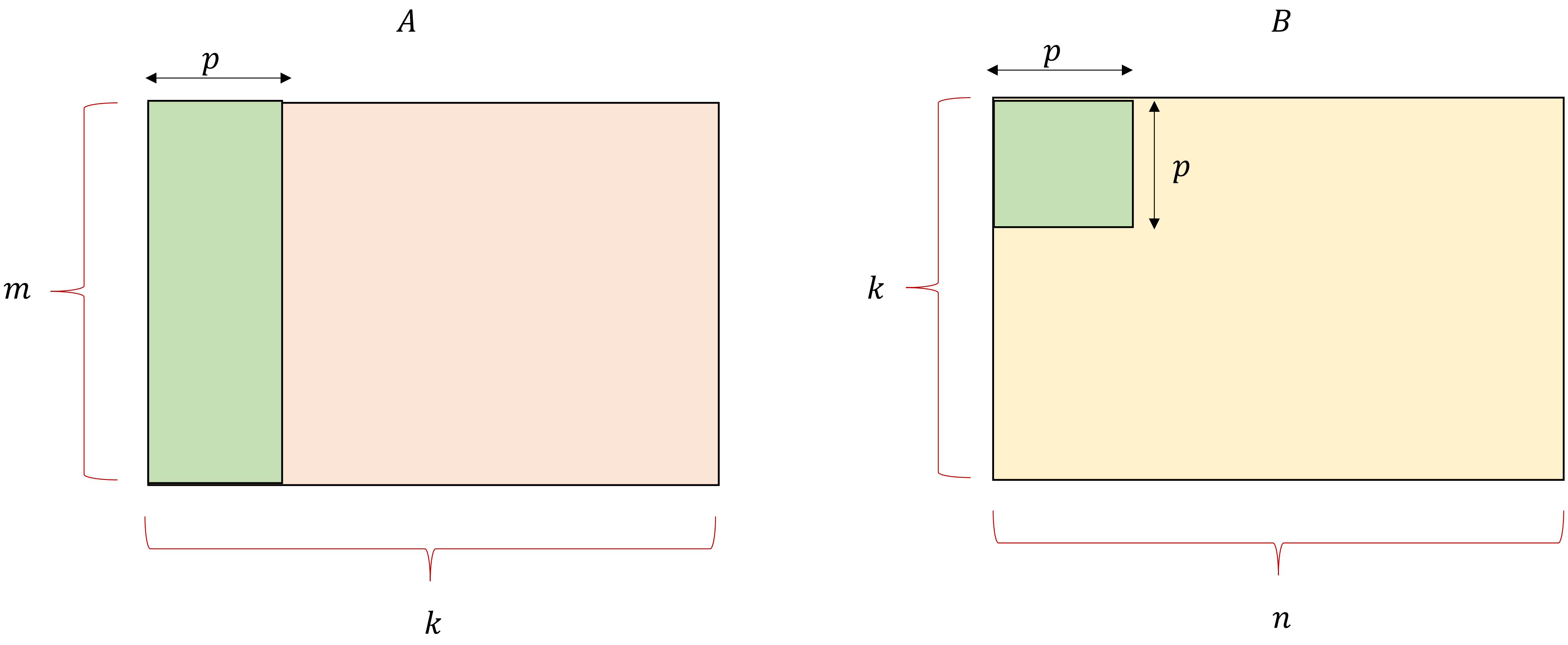
Matrix Multiplication using 2D Systolic Arrays

- $A \ - m \times k, B \ - k \times n$

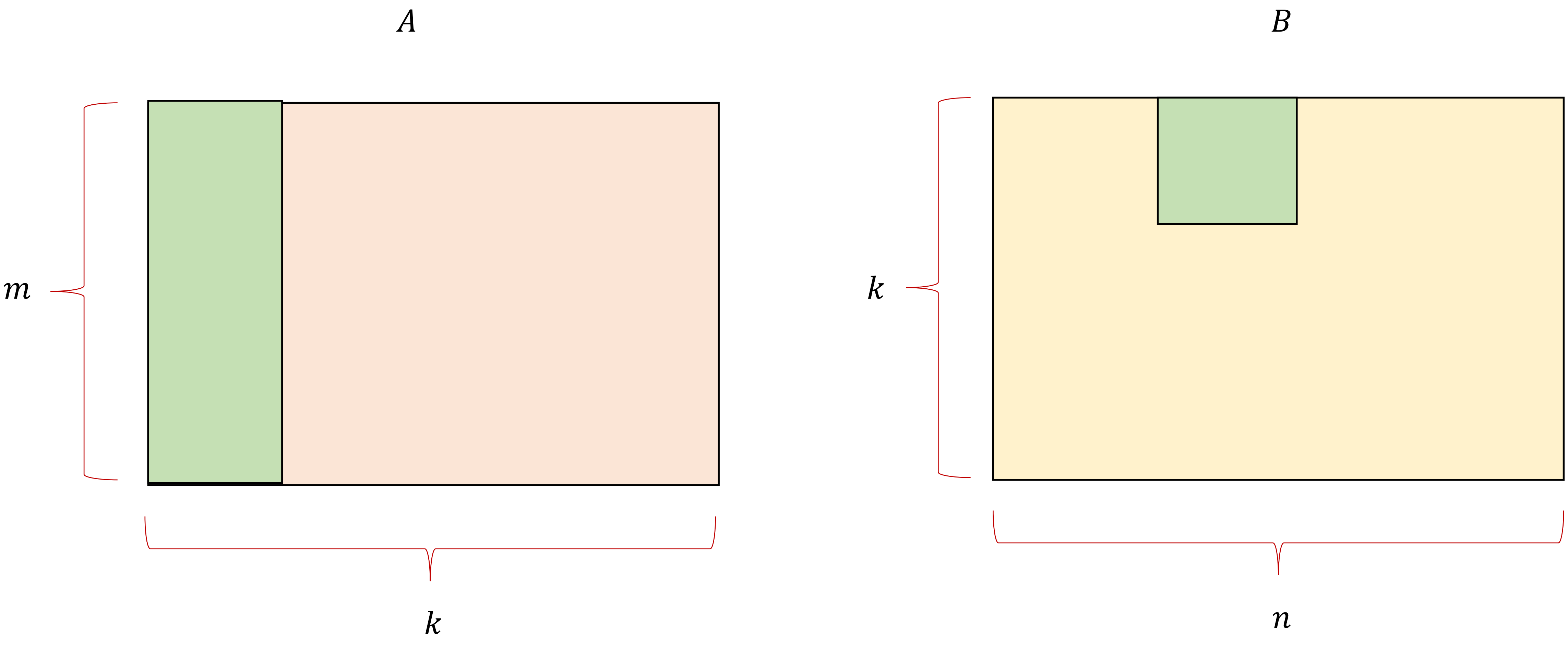
- $p \times p$ systolic array

- Steps
  1. Load a $p \times p$ blocks of B Matrix into systolic array
  2. Stream in rows of Matrix A from the corresponding columns vertically
  3. Collect output horizontally

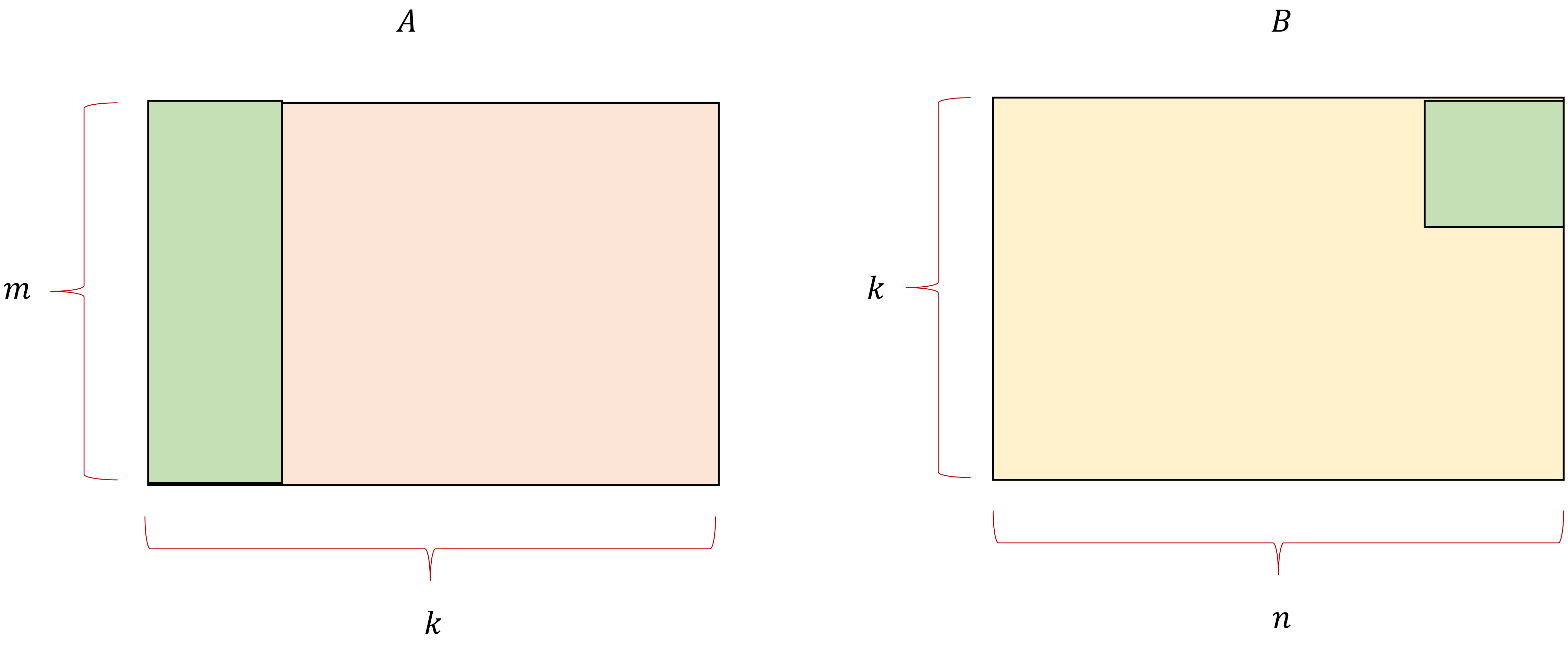
Matrix Multiplication using 2D Systolic Arrays



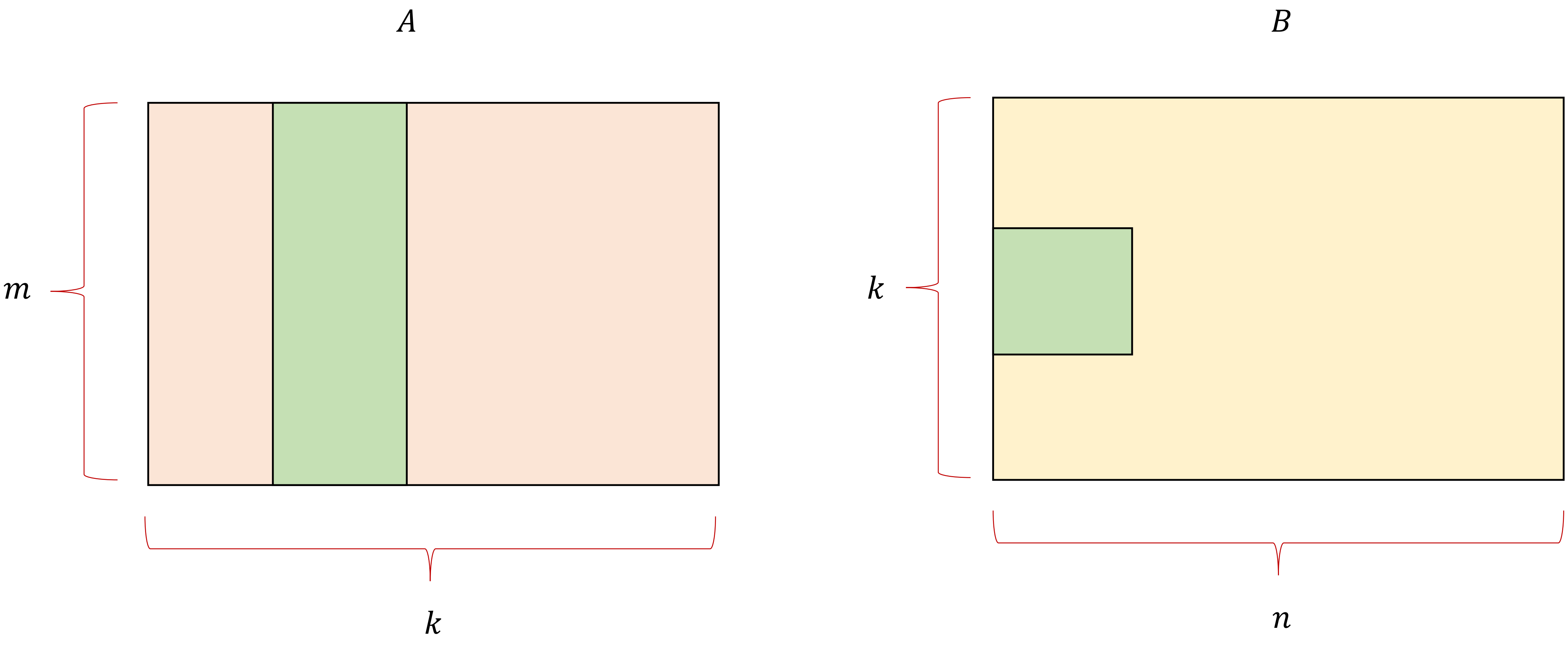
Matrix Multiplication using 2D Systolic Arrays



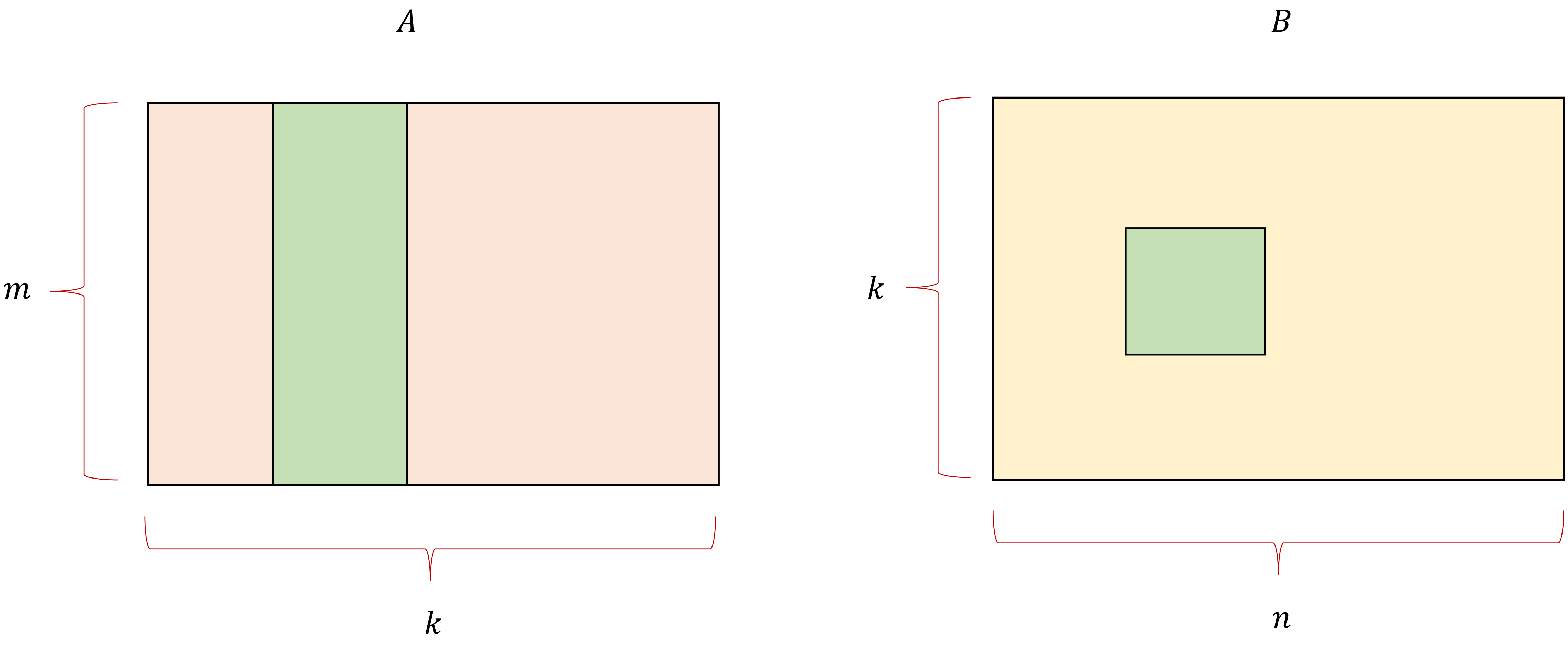
Matrix Multiplication using 2D Systolic Arrays



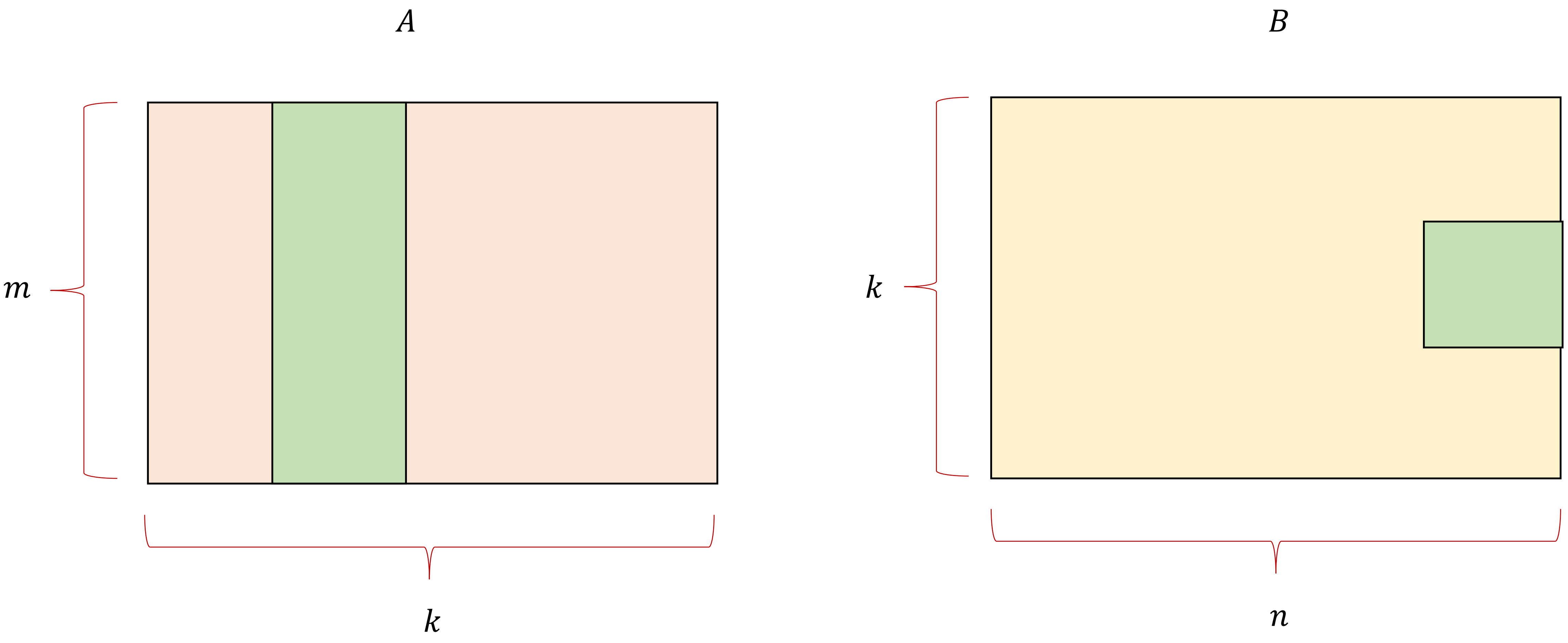
Matrix Multiplication using 2D Systolic Arrays



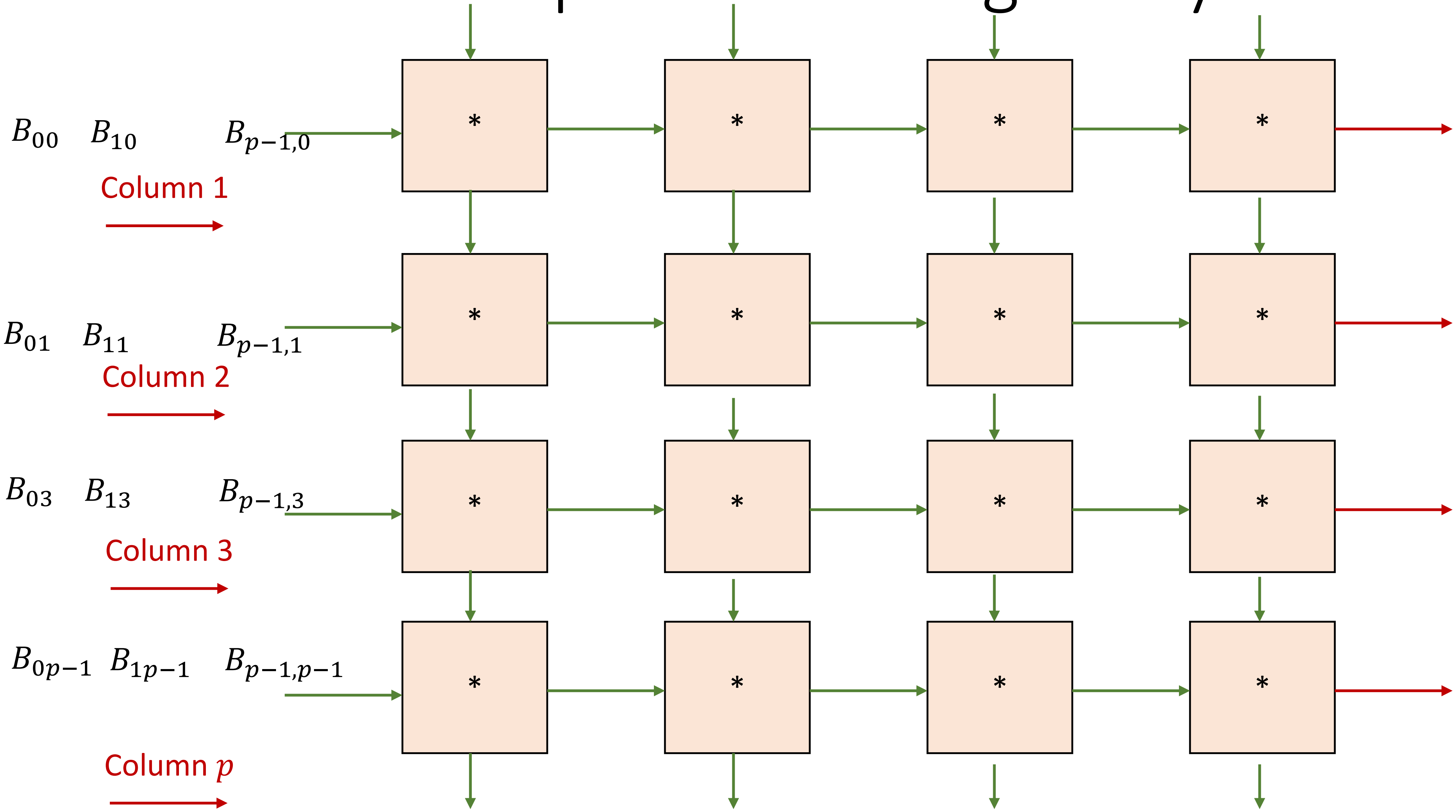
Matrix Multiplication using 2D Systolic Arrays



Matrix Multiplication using 2D Systolic Arrays



Matrix Multiplication using 2D Systolic Arrays

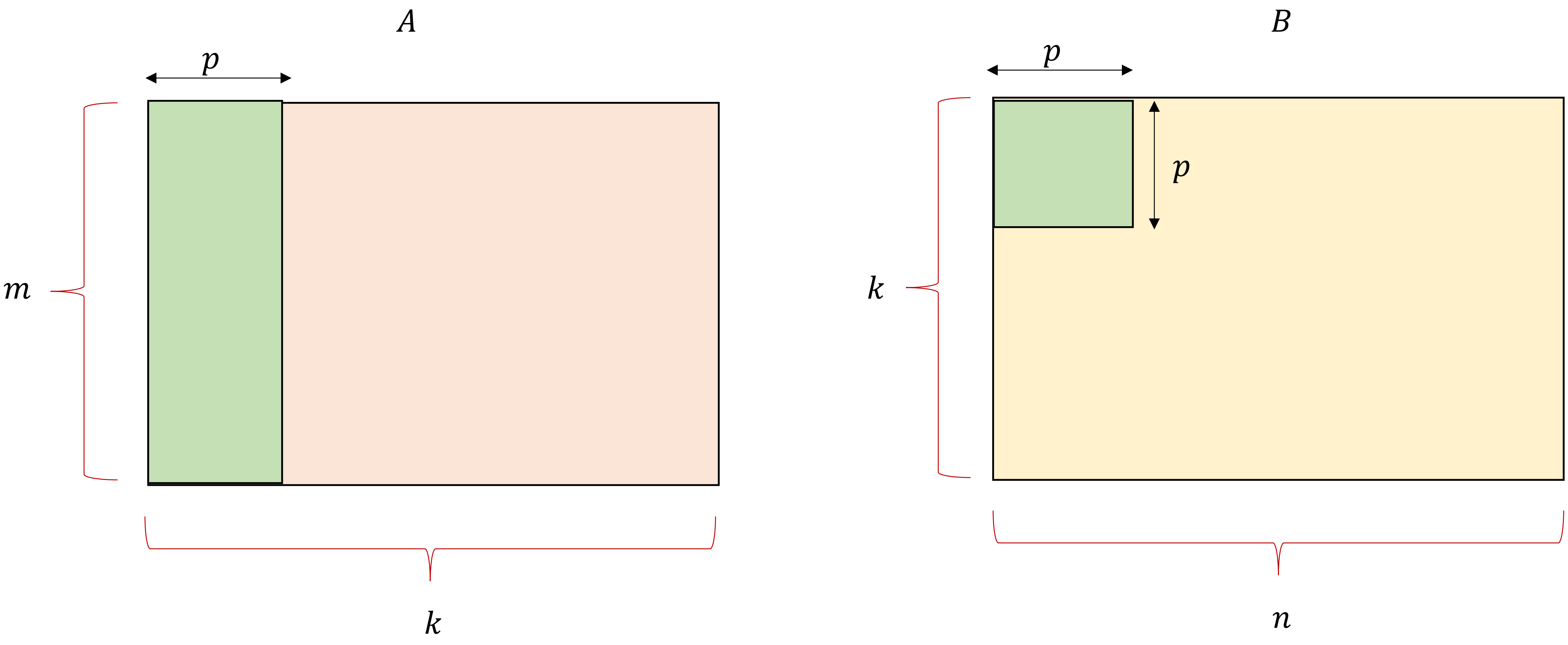


Matrix Multiplication using 2D Systolic Arrays

- Lets first calculate time for a single iteration of the steps below.

- Steps
  1. Load a $p \times p$ blocks of B Matrix into systolic array
  2. Stream in rows of Matrix A from the corresponding columns vertically
  3. Collect output horizontally

Matrix Multiplication using 2D Systolic Arrays



Matrix Multiplication using 2D Systolic Arrays

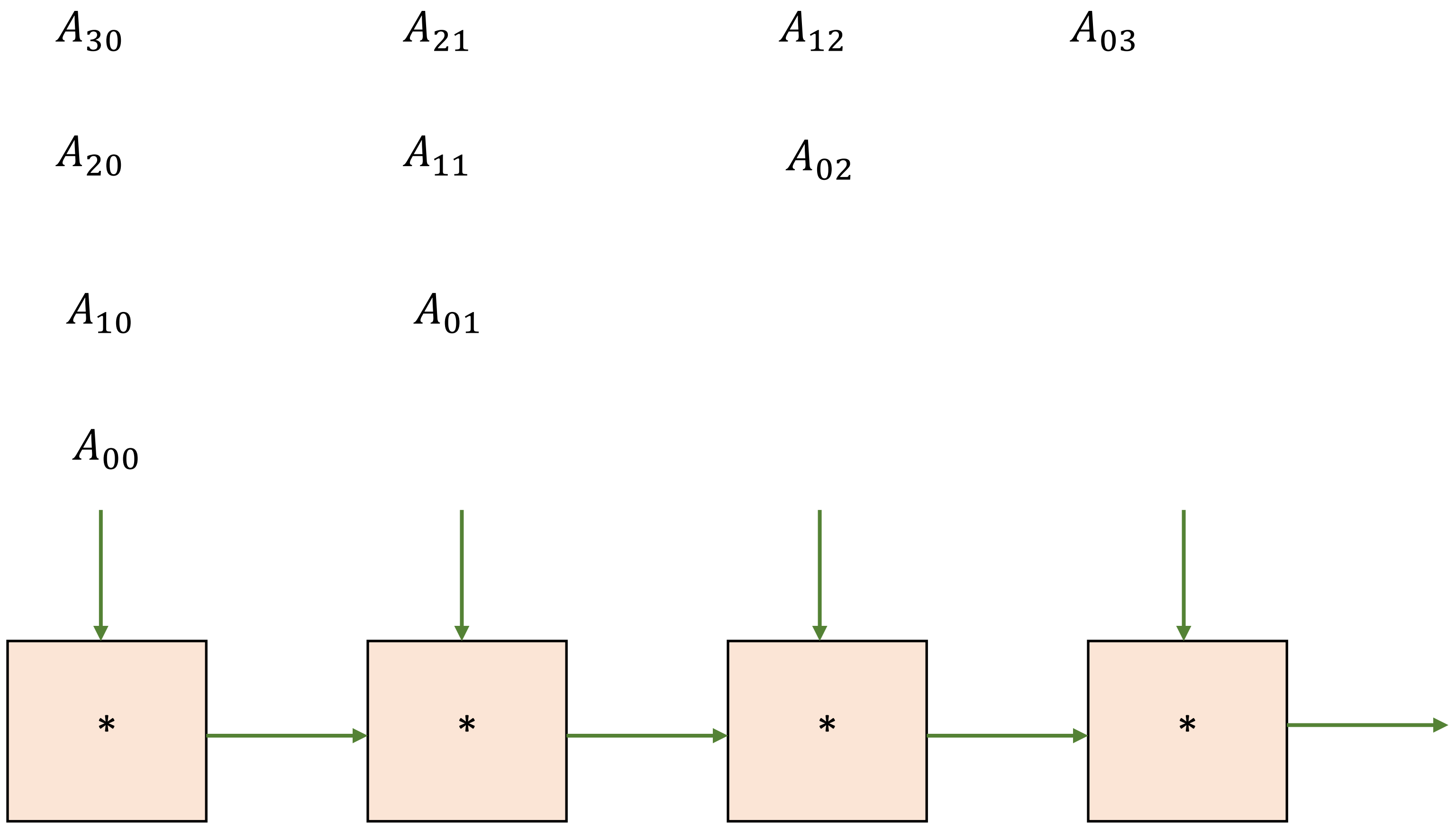
- Loading B matrix into the systolic array of size $p \times p$
- Time - ? ?

Matrix Multiplication using 2D Systolic Arrays

- Loading a block B matrix into the systolic array of size $p \times p$

- Time - p cycles

Matrix Multiplication using 2D Systolic Arrays

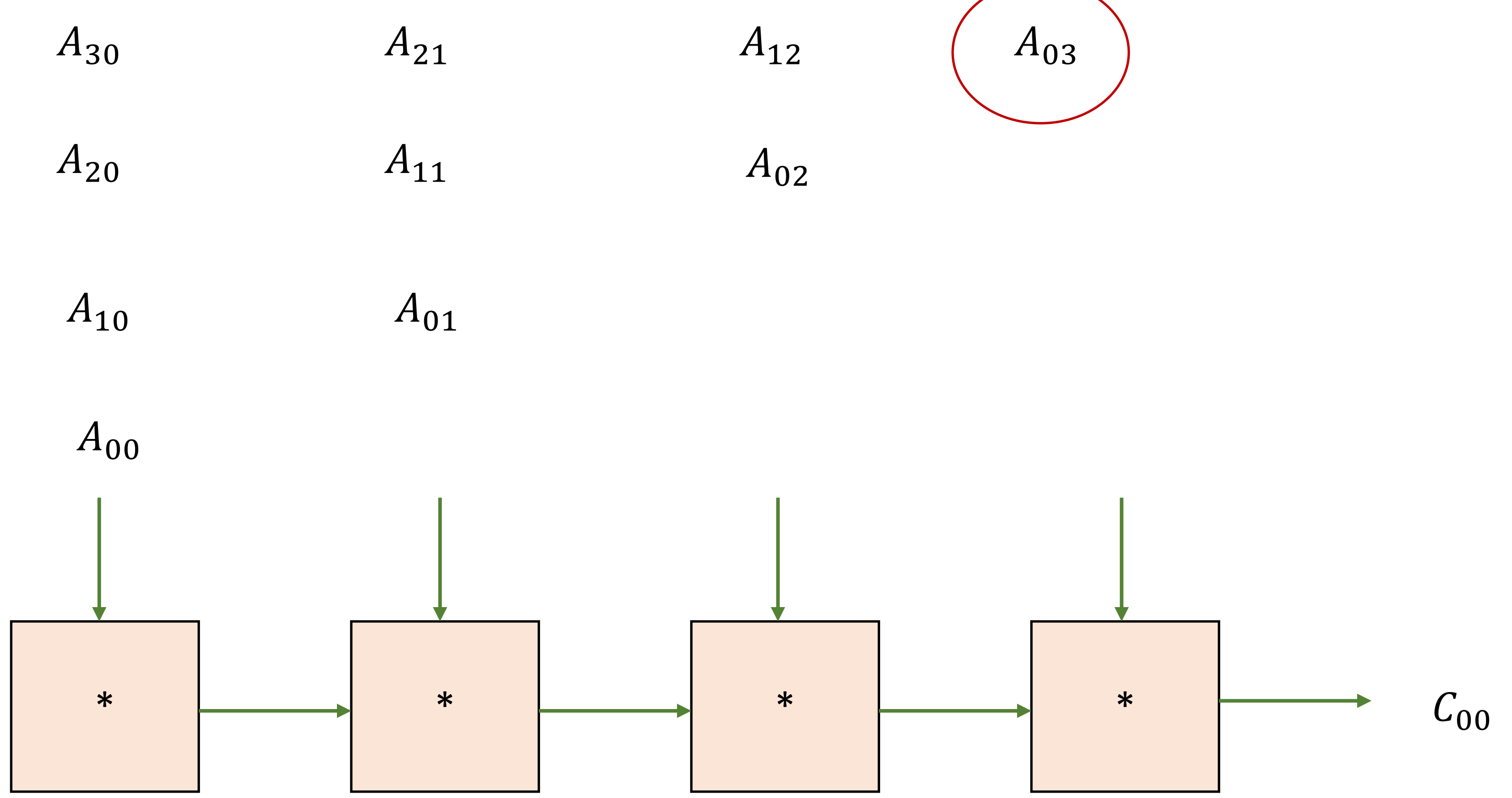


Matrix Multiplication using 2D Systolic Arrays

- Time for Steps 2 and 3

- C_{00} produced in cycle p (after loading the B matrix) as output of row 1

Matrix Multiplication using 2D Systolic Arrays



Matrix Multiplication using 2D Systolic Arrays

$$\underbrace{\left.\begin{array}{ccc} B_{00} & B_{10} & B_{p-1,0} \\ B_{01} & B_{11} & B_{p-1,1} \\ B_{03} & B_{13} & B_{p-1,3} \\ B_{0p-1} & B_{1p-1} & B_{p-1,p-1} \end{array}\right\} p}_{p}$$

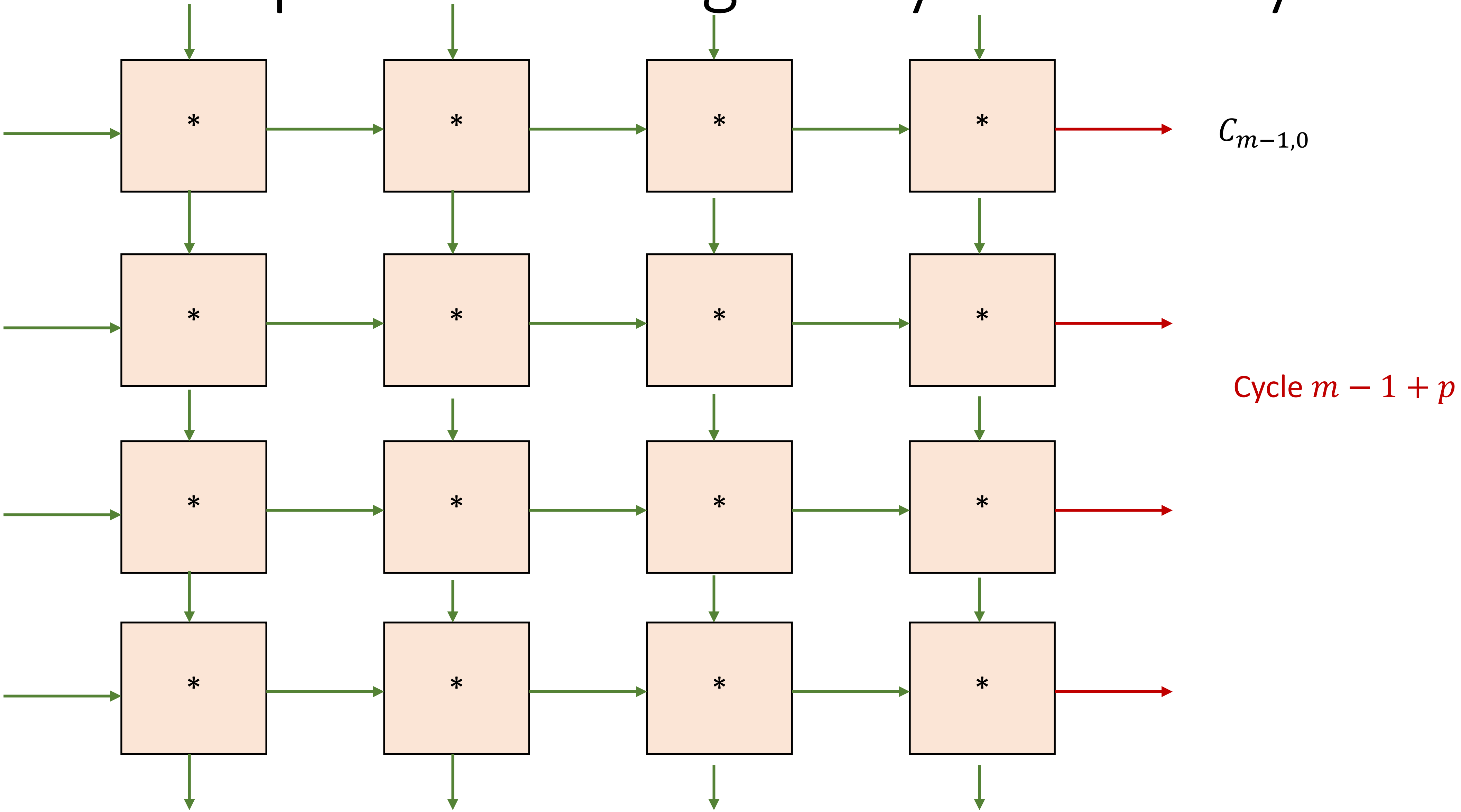
$$m \left\{ \underbrace{\begin{array}{cccc} A_{30} & A_{21} & A_{12} & A_{03} \\ A_{20} & A_{11} & A_{02} & \\ A_{10} & A_{01} & & \\ A_{00} & & & \end{array}}_{p} \right.$$

Each row of systolic array computes a 1D MV with a column of B as vector and A as matrix

Matrix Multiplication using 2D Systolic Arrays

- Time for Steps 2 and 3
- C_{00} produced in cycle p (after loading the B matrix) as output of row 1
- $C_{m-1,0}$ produced in cycle $\mathrm{p} + m - 1$ (after loading the B matrix) as output of row 1
  - As we saw in 1D MV, it takes $m - 1$ more cycles for the row to produce the last element of the output

Matrix Multiplication using 2D Systolic Arrays



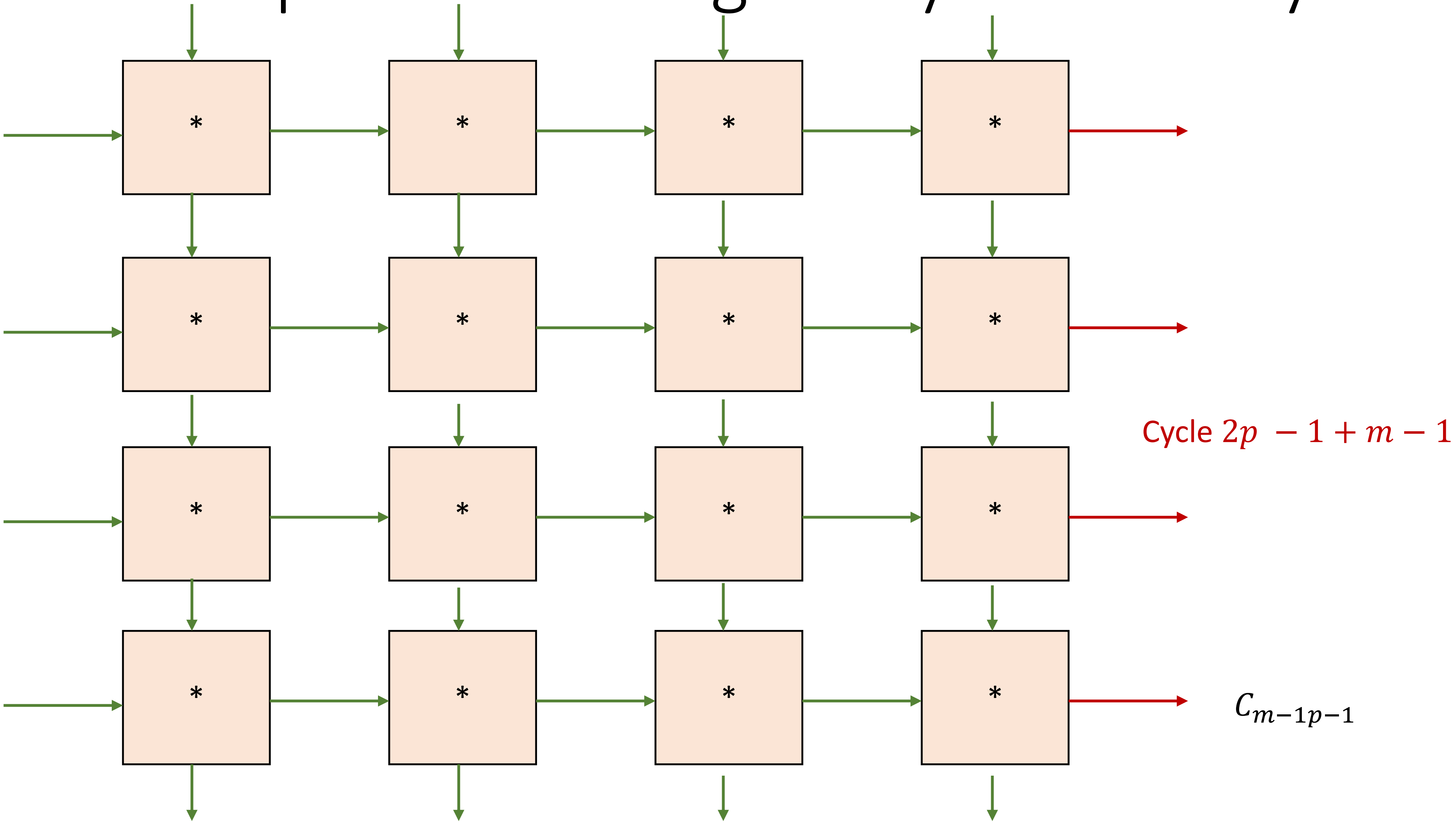
Matrix Multiplication using 2D Systolic Arrays

- Time for Steps 2 and 3

- C_{0p-1} produced in cycle $2p - 1$ (after loading the B matrix) as output of row p
  - It will take $p - 1$ more cycles for A_{0p-1} to reach the last row of the systolic array and produce output

- $C_{m-1,p-1}$ produced in cycle $2p\ - 1 + m - 1$ (after loading the B matrix) as output of row p
  - Another $m - 1$ cycles to produce the last element of Matrix Vector product output.

Matrix Multiplication using 2D Systolic Arrays



Matrix Multiplication using 2D Systolic Arrays

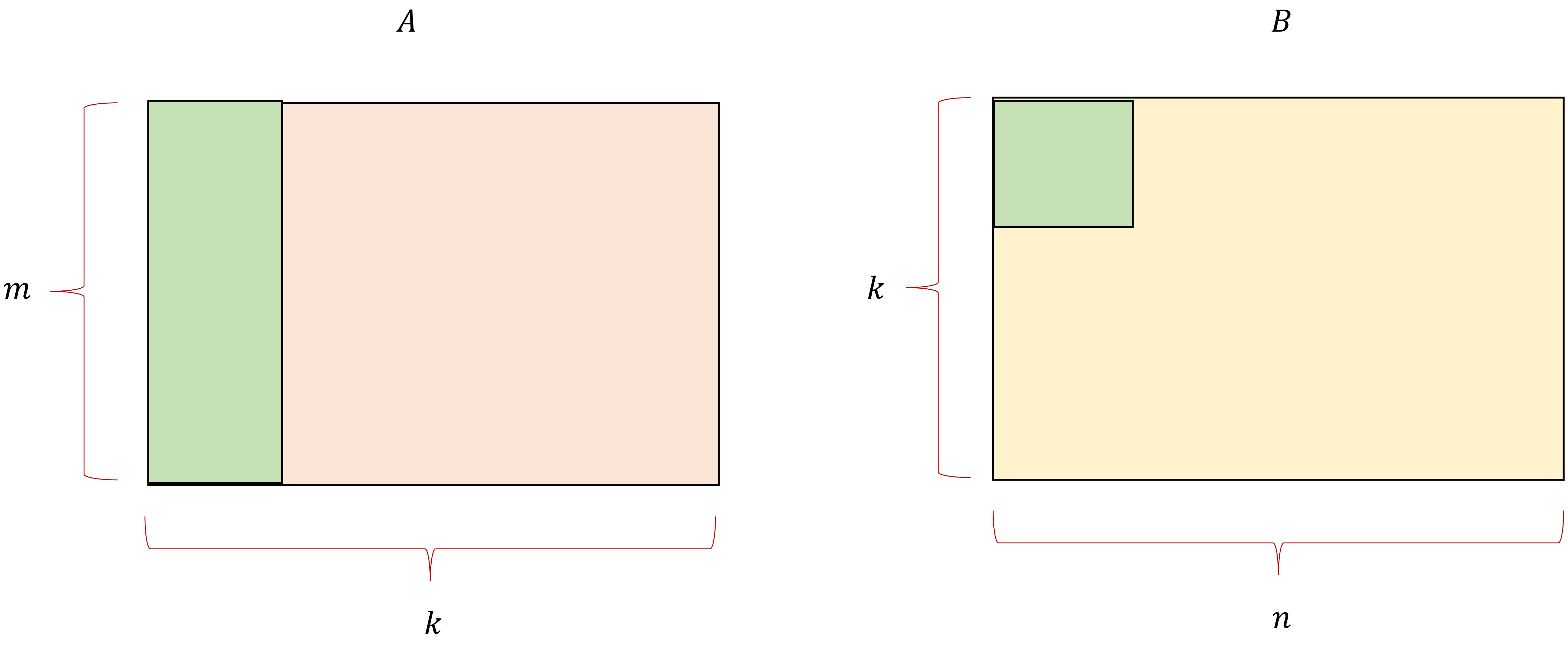
- Total Time for Matrix Multiplication
- Input Loading time - p cycles
- Output Time (after input latency) - $2p - 1 + m - 1$
- Total time for one round - $3p + m \ - 2$

Matrix Multiplication using 2D Systolic Arrays

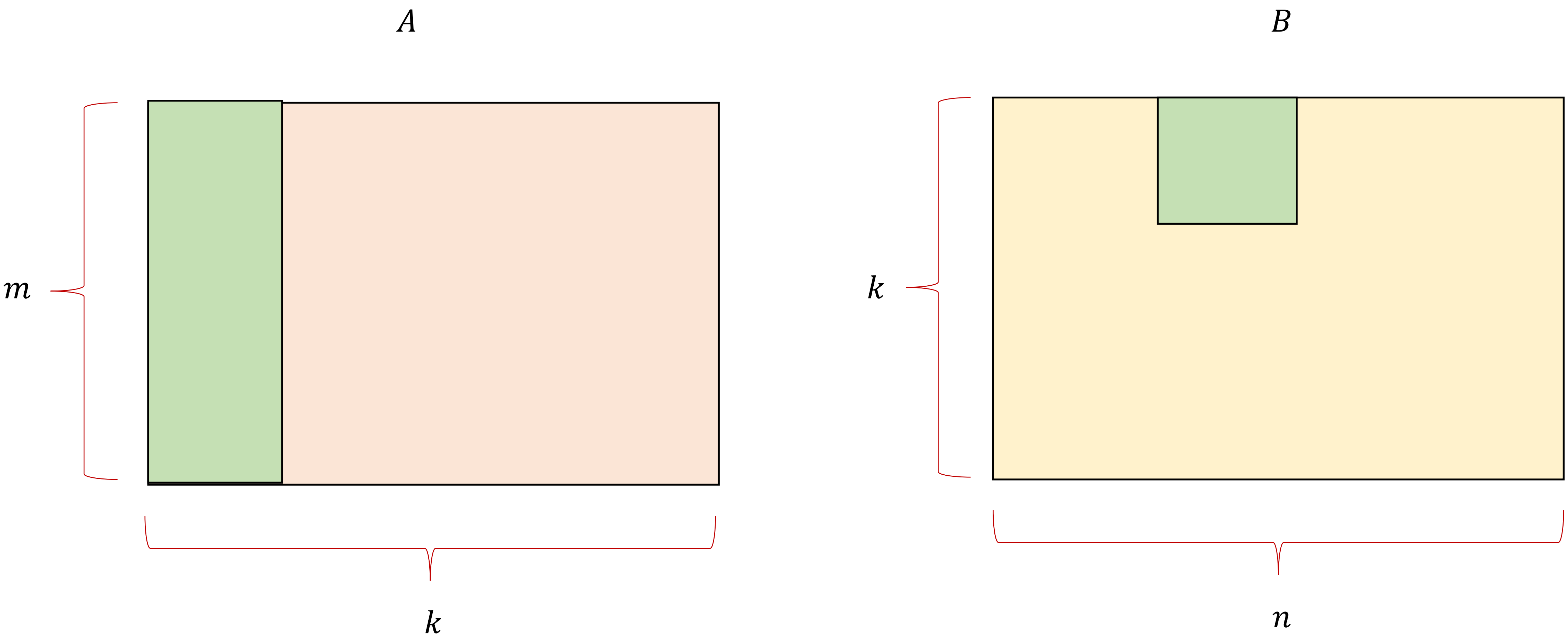
- Total time for Matrix Multiplication
- $A \; - m \times k, B \; - k \times n$

- How may rounds (iterations) do we need?

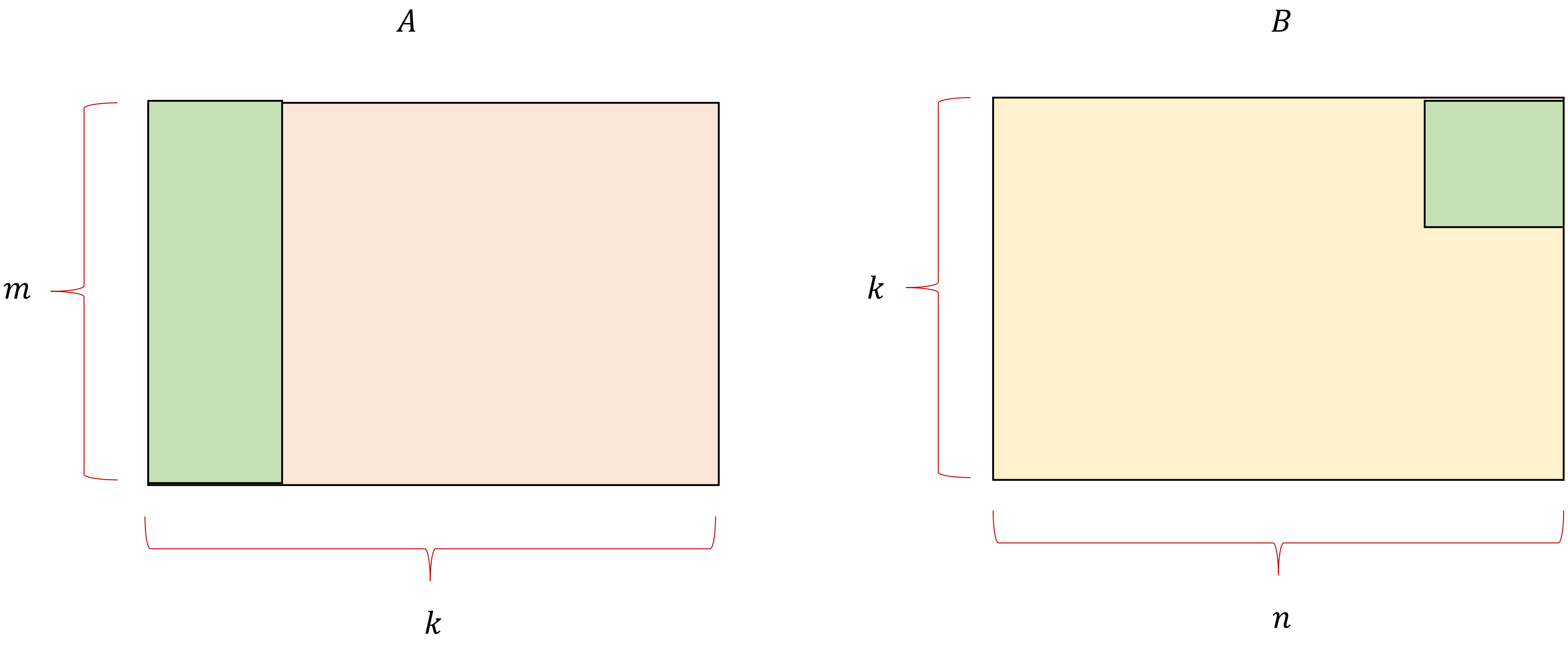
Matrix Multiplication using 2D Systolic Arrays



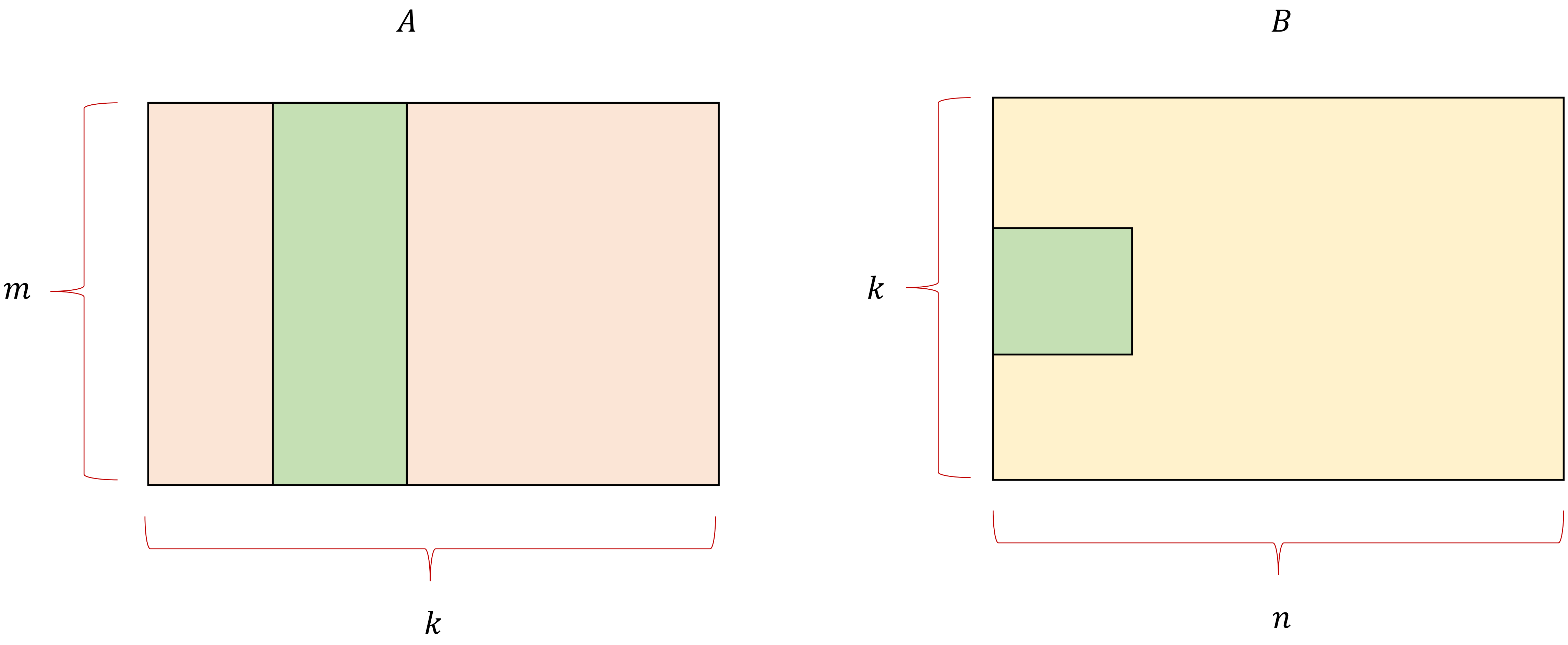
Matrix Multiplication using 2D Systolic Arrays



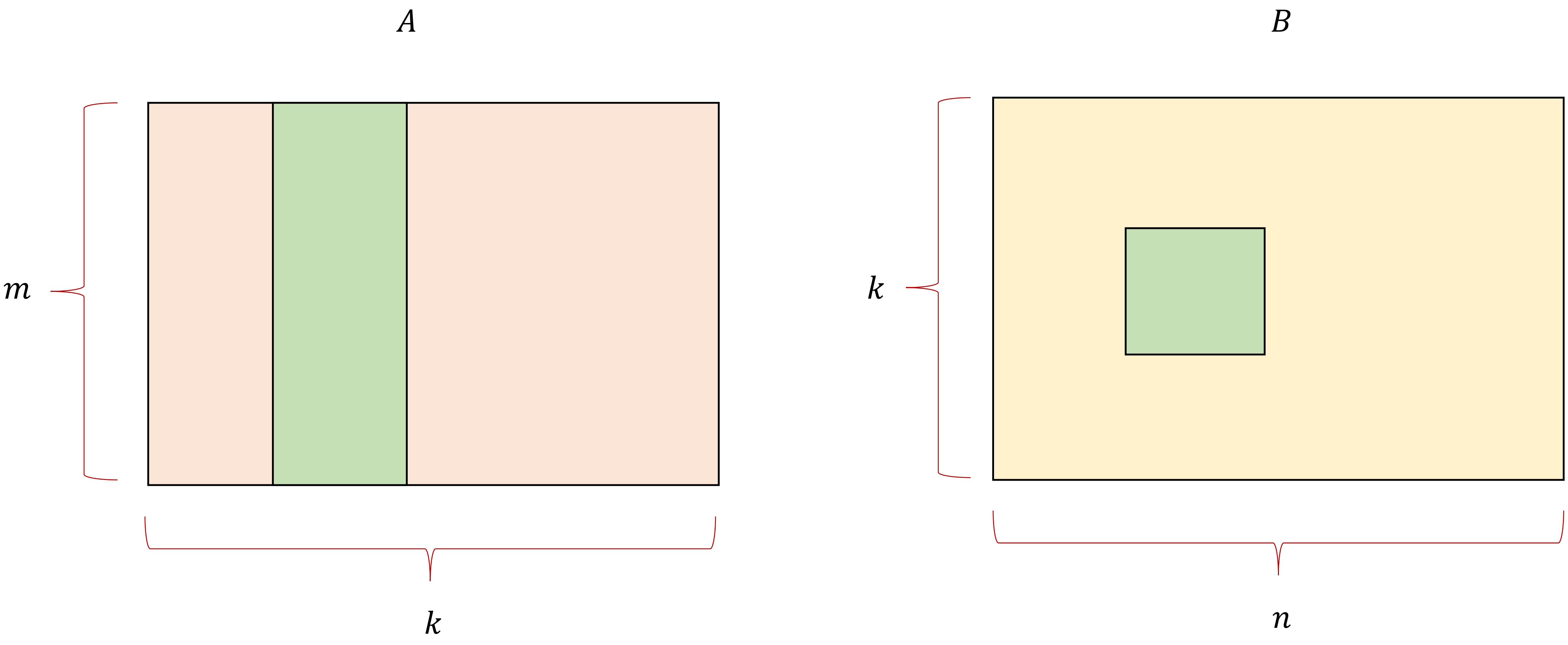
Matrix Multiplication using 2D Systolic Arrays



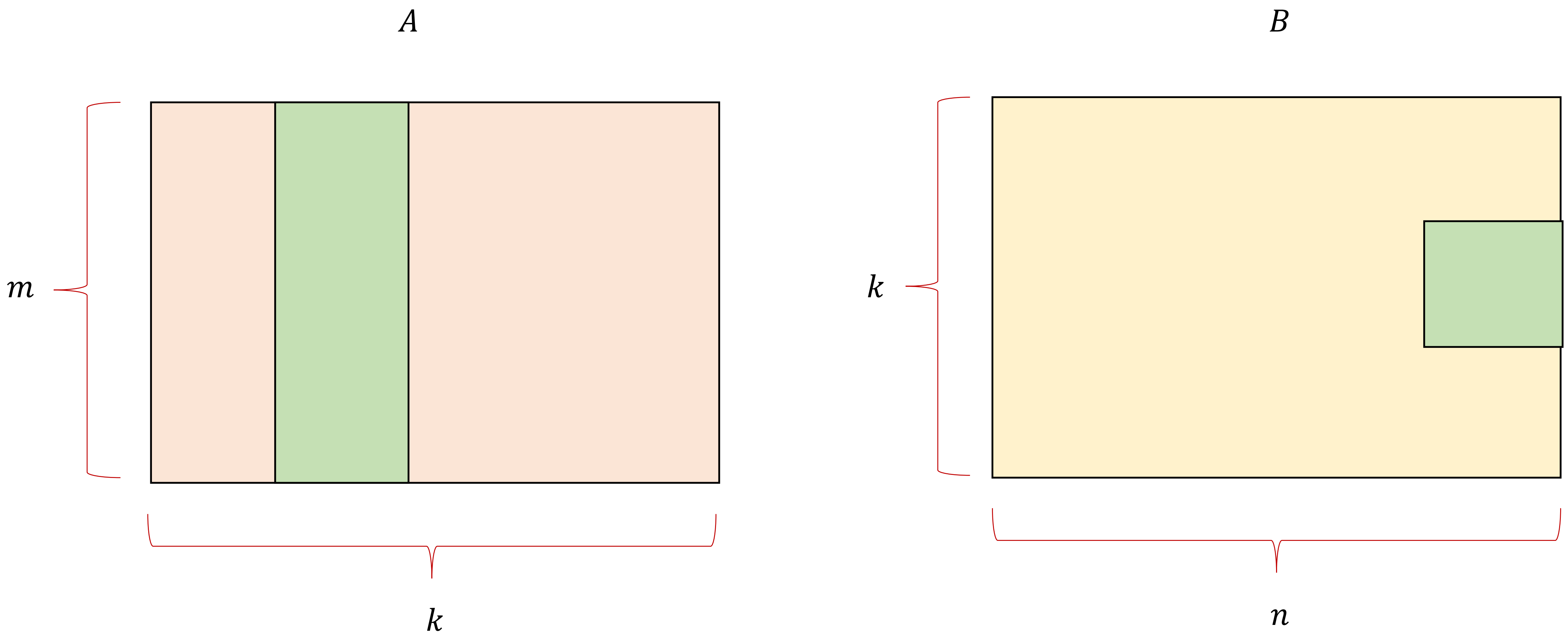
Matrix Multiplication using 2D Systolic Arrays



Matrix Multiplication using 2D Systolic Arrays



Matrix Multiplication using 2D Systolic Arrays



CASE SCHOOL OF ENGINEERING
CASE WESTERN RESERVE UNIVERSITY

Matrix Multiplication using 2D Systolic Arrays

- Total time for Matrix Multiplication
- $A\ - m \times k, B\ - k \times n$

- Number of Rounds = $\left\lceil \frac{n}{p} \right\rceil \times \lceil \frac{k}{p} \rceil$

- Total Time - $(3p + m\ - 2) \times \left\lceil \frac{n}{p} \right\rceil \times \lceil \frac{k}{p} \rceil$

Matrix Multiplication using 2D Systolic Arrays

- Total Time - $(3p + m\ - 2) \times \left\lceil \frac{n}{p} \right\rceil \times \lceil \frac{k}{p} \rceil$

- Orientation of matrices may impact performance (although not asymptotically)
  - Here we say B is stationary while A is streamed

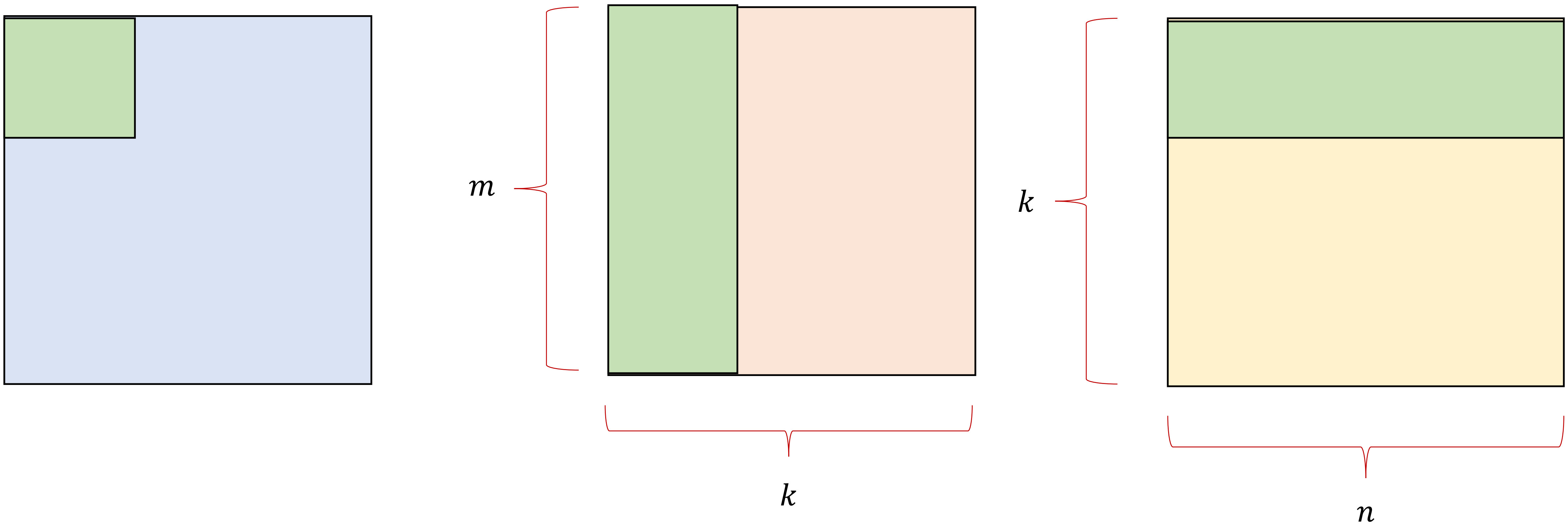
- Alternatively, we can have A as stationary and B streamed

Matrix Multiplication using 2D Systolic Arrays

- Third option is Non Stationary/Output Stationary
  - Stream both input matrices
  - Similar to Blocked Matrix Multiplication

- Ungraded Homework Assignment: Find out how to implement blocked matrix multiplication along the lines of the previous algorithms
  - Maybe a question in Exam 1

Blocked Matrix Multiplication



Next Class

- 9/23 Lecture 9
  - Accelerating Convolutional Neural Networks: Basics

Thank You

- Questions?

- Email: sanmukh.kuppannagari@case.edu

CASE SCHOOL OF ENGINEERING
CASE WESTERN RESERVE UNIVERSITY