# Logic Circuit Design Lab (ICE2005)

Final Project

Instructor: Il Yong Chun, EEE & AI

(Revised by Il Yong Chun, 22-11-07)

# Agenda

- Computations in deep neural networks

- Project assignment

# Computations in Deep Neural Networks

# AI, ML, & DL…

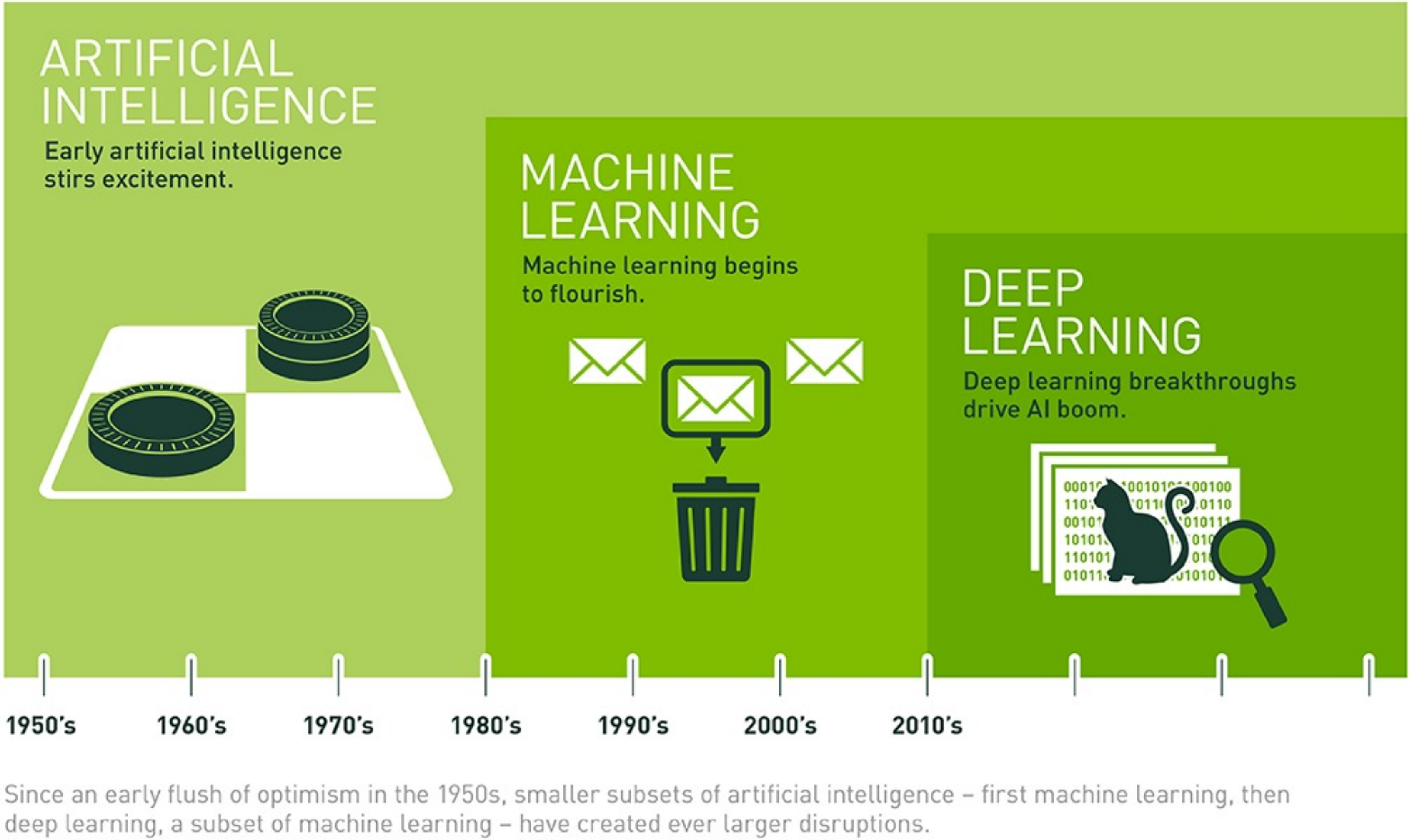



*Figure from NVIDIA (https://developer.nvidia.com/deep-learning)

# Deep Neural Networks (DNNs)

- Fully-connected networks (FCN)
- Convolutional neural network (CNN)
- Recurrent Neural Network
- Generative Adversarial Network
- Iterative Neural Network
- and more....

# DNN Example for Image Classification

- CNN+FCN

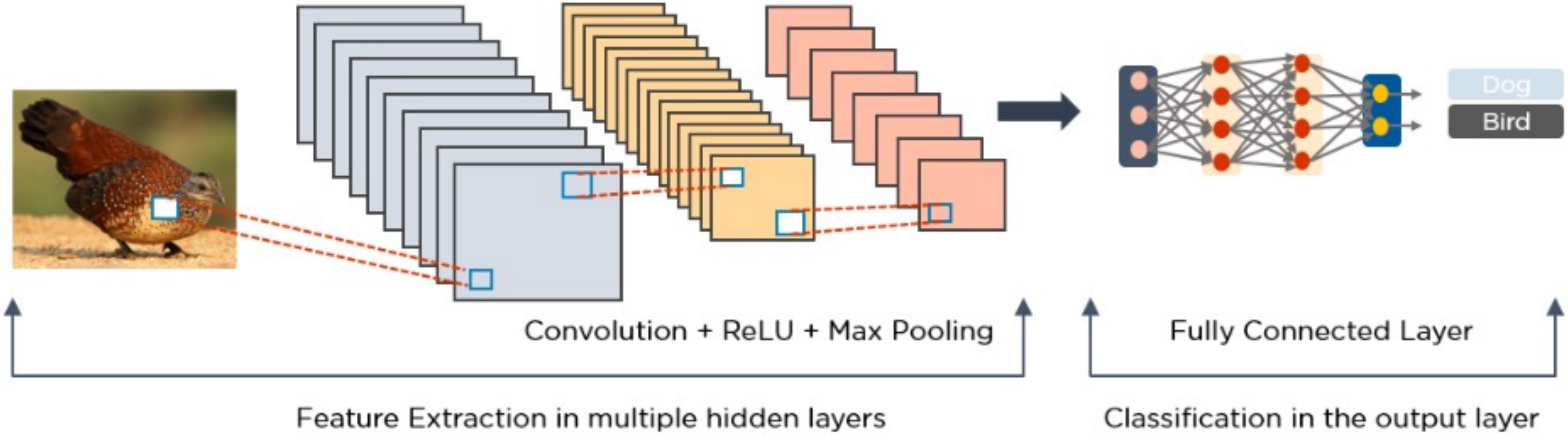
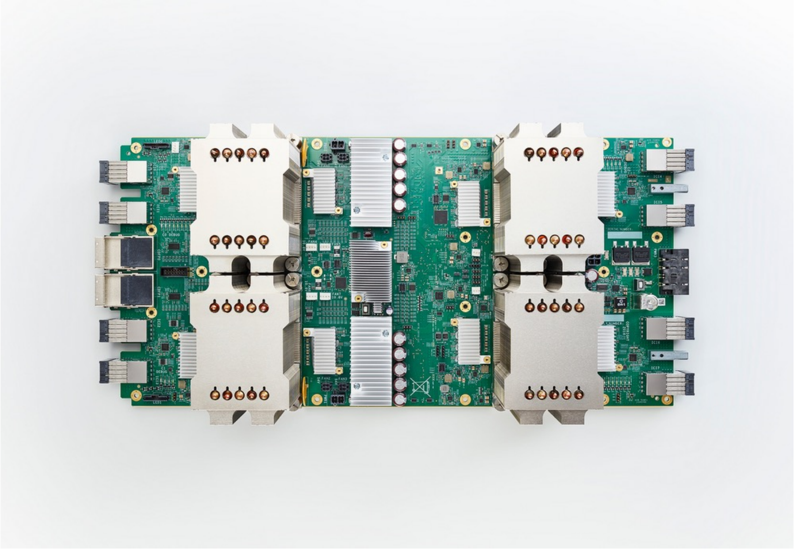
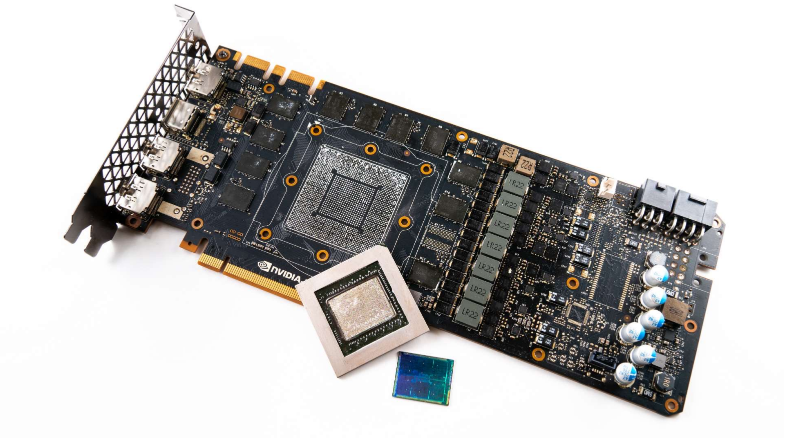



*Figure from Simplilearn
(https://www.simplilearn.com/tutorials/deep-learning-tutorial/deep-learning-algorithm)

# DNN Accelerators



Google TPU



GPUs (including Tensor cores)

**What do they accelerate?**

# Major Computations in DNNs

- 2D convolution: Aggregating sums of partial matrix multiplications

| | | | |
|---|---|---|---|
| $A_{11}$ | $A_{12}$ | $A_{13}$ | $A_{14}$ |
| $A_{21}$ | $A_{22}$ | $A_{23}$ | $A_{24}$ |
| $A_{31}$ | $A_{32}$ | $A_{33}$ | $A_{34}$ |
| $A_{41}$ | $A_{42}$ | $A_{43}$ | $A_{44}$ |

Input

| | | |
|---|---|---|
| $B_{11}$ | $B_{12}$ | $B_{13}$ |
| $B_{21}$ | $B_{22}$ | $B_{23}$ |
| $B_{31}$ | $B_{32}$ | $B_{33}$ |

Filter

| | |
|---|---|
| $C_{11}$ | $C_{12}$ |
| $C_{21}$ | $C_{22}$ |

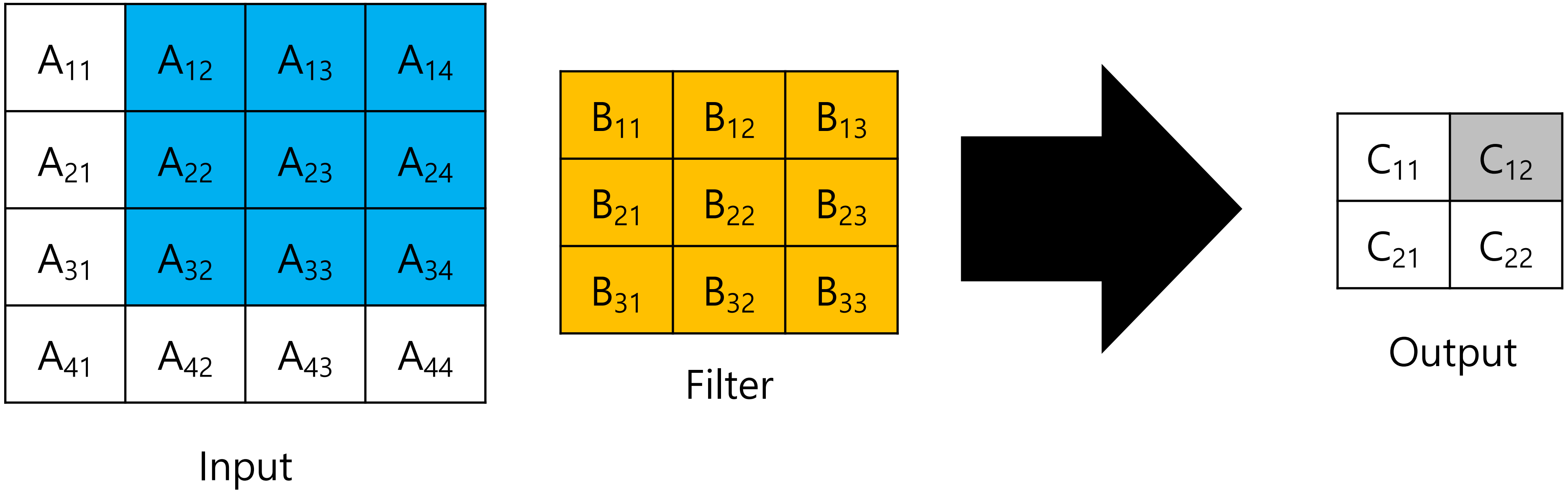
Output

Rotated filter by 180 degree!

| | | |
|---|---|---|
| $B_{33}$ | $B_{32}$ | $B_{31}$ |
| $B_{23}$ | $B_{22}$ | $B_{21}$ |
| $B_{13}$ | $B_{12}$ | $B_{11}$ |

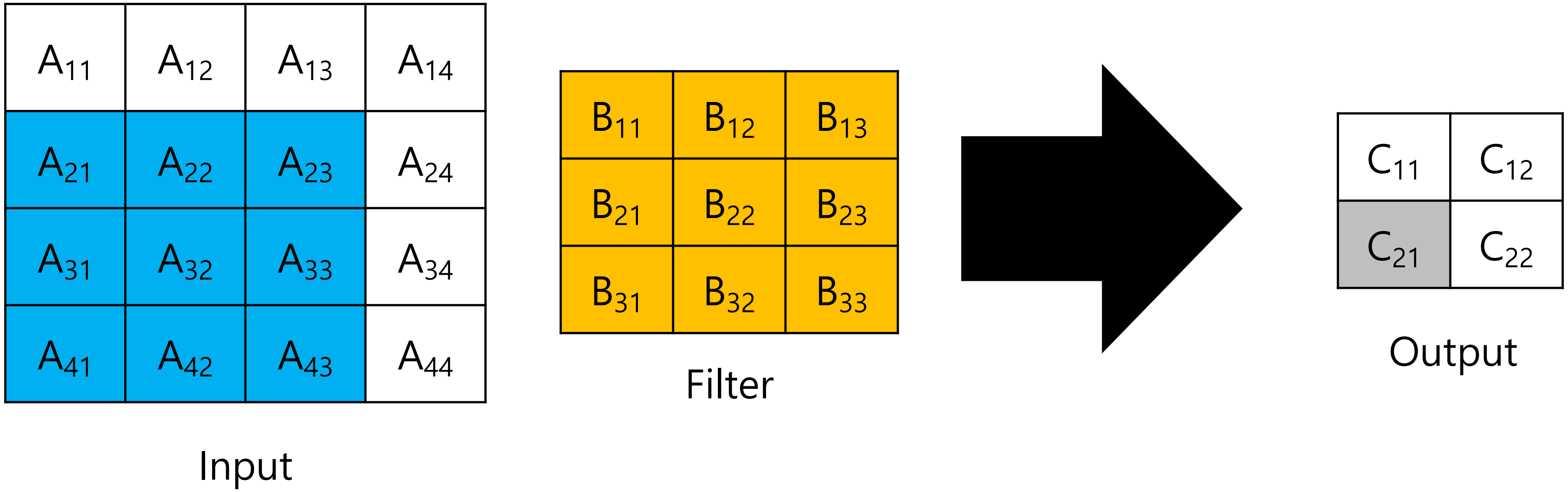
$$
\begin{aligned}
C_{11} = {} & A_{11} * B_{33} + A_{12} * B_{32} + A_{13} * B_{31} + \\
& A_{21} * B_{23} + A_{22} * B_{22} + A_{23} * B_{21} + \\
& A_{31} * B_{13} + A_{32} * B_{12} + A_{33} * B_{11}
\end{aligned}
$$

# Striding



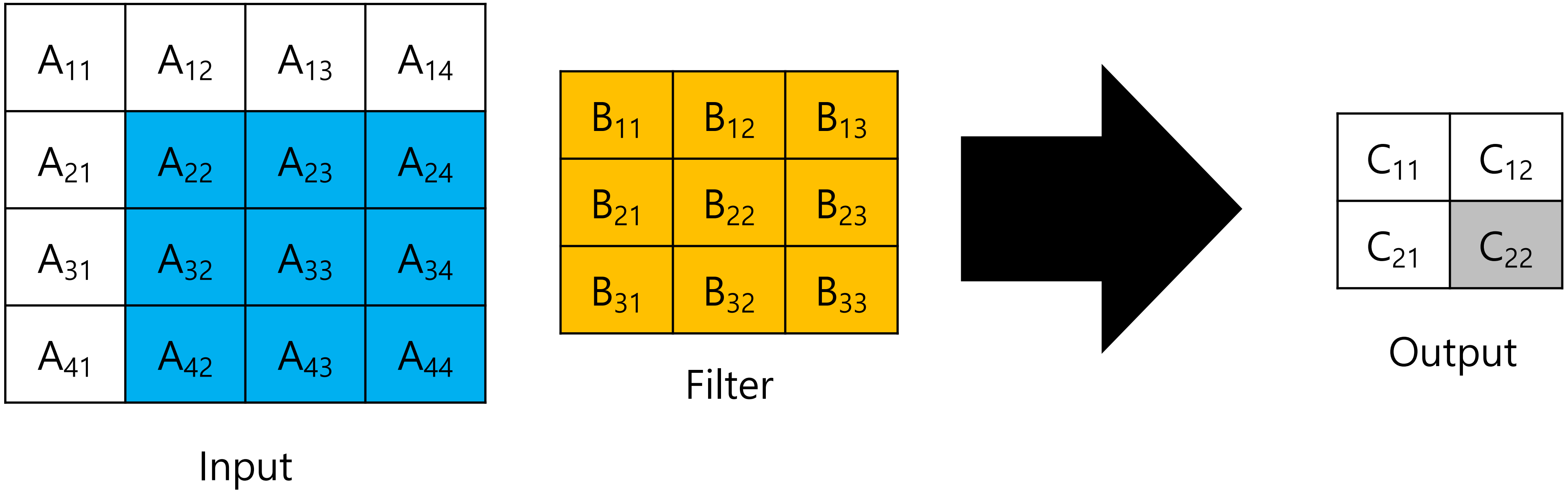

What would the corresponding equation look like?

# Next Step




What would the corresponding equation look like?

# Last Step




What would the corresponding equation look like?

# Term Project

# Your Project

- A set of logic that computes a 2x2 output using
  - A single processing element (PE)
  - A 3x3 systolic array
  - A 2x2 systolic array

- **Do the following.**
  - **Simulation study**
  - **Verifying your design by displaying results.**

| | | | |
|---|---|---|---|
| $A_{11}$ | $A_{12}$ | $A_{13}$ | $A_{14}$ |
| $A_{21}$ | $A_{22}$ | $A_{23}$ | $A_{24}$ |
| $A_{31}$ | $A_{32}$ | $A_{33}$ | $A_{34}$ |
| $A_{41}$ | $A_{42}$ | $A_{43}$ | $A_{44}$ |

Input

| | | |
|---|---|---|
| $B_{11}$ | $B_{12}$ | $B_{13}$ |
| $B_{21}$ | $B_{22}$ | $B_{23}$ |
| $B_{31}$ | $B_{32}$ | $B_{33}$ |

Filter

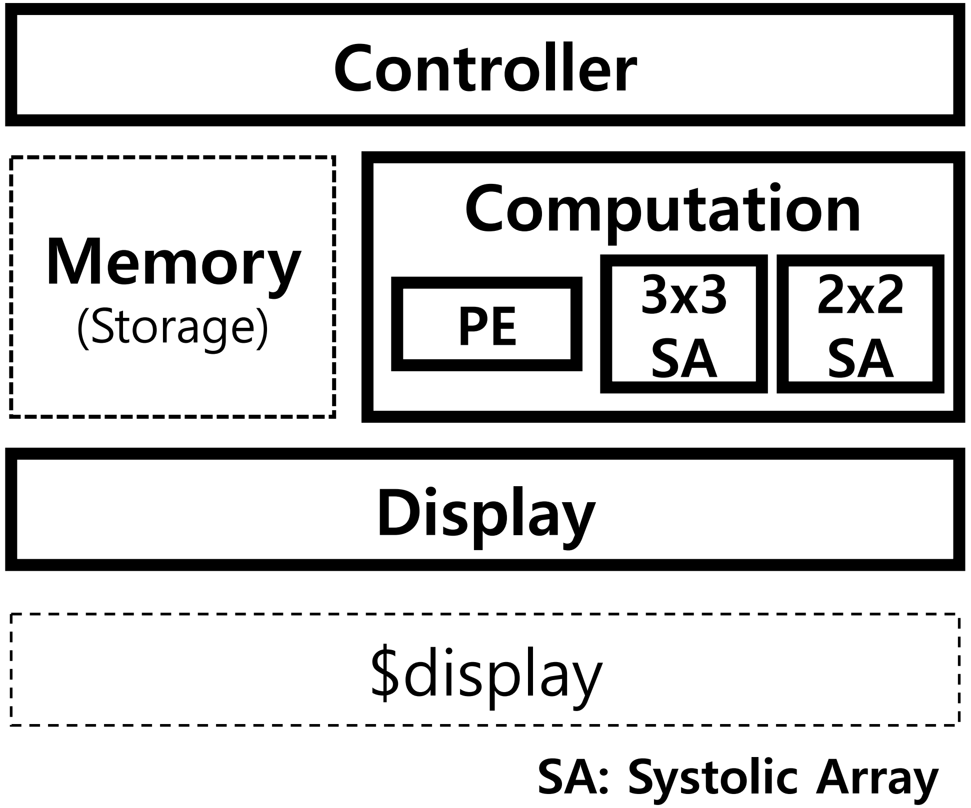
2D convolution

| | |
|---|---|
| $C_{11}$ | $C_{12}$ |
| $C_{21}$ | $C_{22}$ |

Output

# Design Overview (1/2)

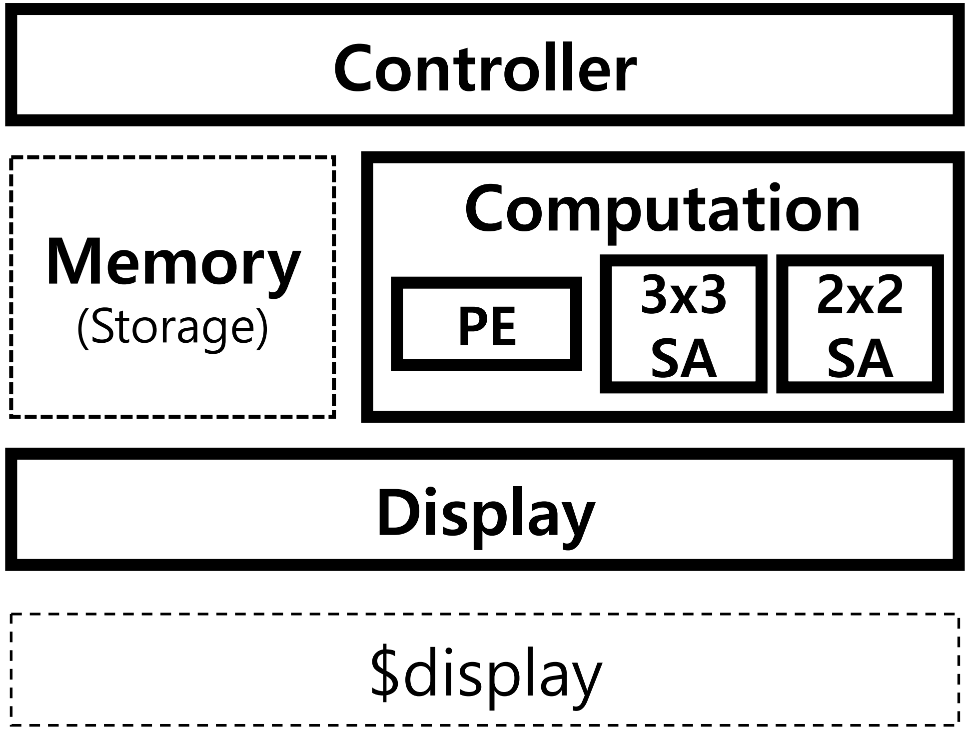
- **Controller**
  - Initializes and of memory, computation module, and display module.
  - Takes controls of all the modules.
  - For each module, if you need a micro-control inside it, you can implement that control flow in the corresponding module.




- **Memory** keeps and outputs the numbers of
  - An input matrix (16 numbers)
  - A filter (9 numbers)

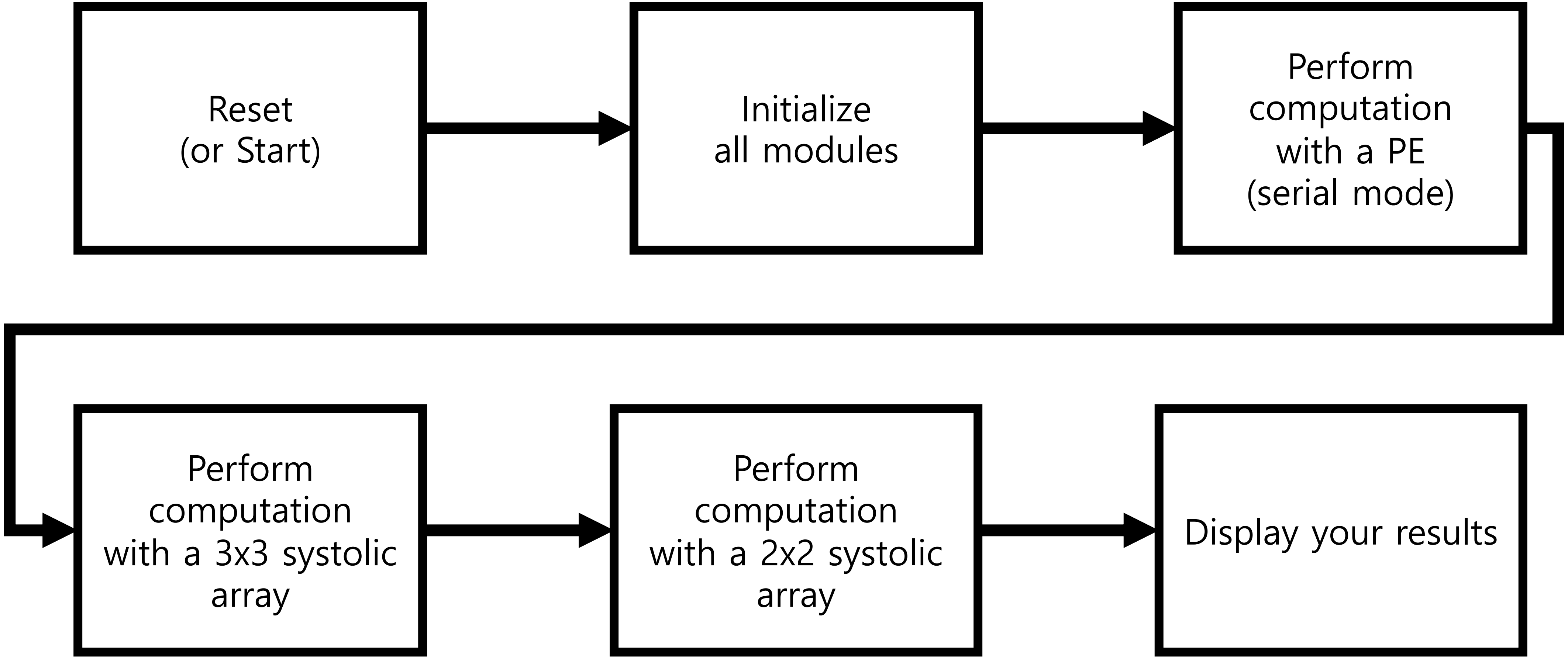
# Design Overview (2/2)

- **Computation module** performs computations in two modes.
  - **Serial mode**: Compute with a single PE.
  - **Parallel mode 1**: Compute with 3x3 systolic array.
  - **Parallel mode 2**: Compute with 2x2 systolic array.
- **Display module** shows computation results.




SA: Systolic Array

# Execution Flow

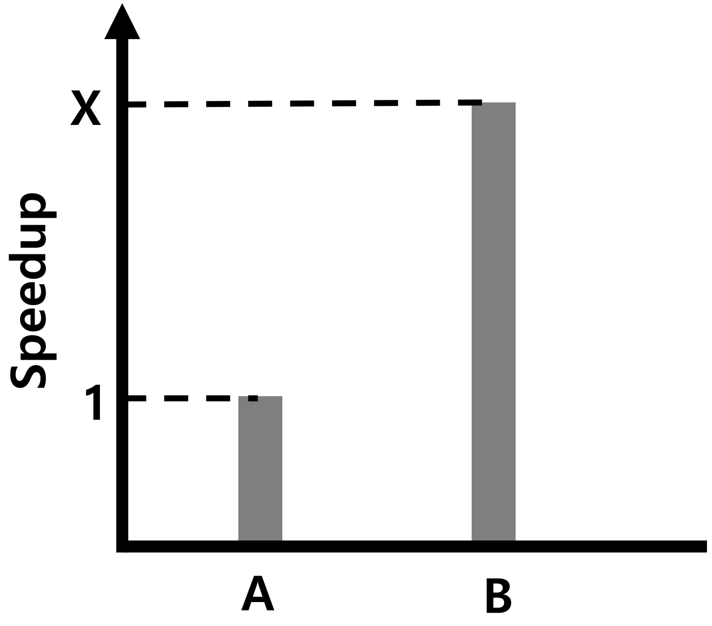


# Simulation Study (1/3)

- **Set a test plan for each module (7 testbench files)**
  - Controller
  - Memory
  - PE
  - Two systolic arrays
  - Display module
  - Top module

# Simulation Study (2/3)

- **Do performance comparisons**
  - A single PE vs a systolic array




$$X = \text{Speedup}_{BA} = \frac{\text{Execution Time}_A}{\text{Execution Time}_B}$$

# Simulation Study (3/3)

- Report the following
  - Speedup of 3x3 systolic array over a single PE
  - Speedup of 2x2 systolic array over a single PE

# Verification

- **<u>Show all the computation results.</u>**
  - 4 values in 2x2 output matrix
    - Computed by a 3x3 systolic array
    - Computed by a 2x2 systolic array
    - More details in Requirement 5
  - $display or any other visualization method is fine.
    - Make sure to show each value of output and its corresponding index (e.g., c_{1,1}, c_{1,2}).

- Don’t need to display the results with a single PE.

# Requirement 0: General

- Follow the design overview.
- Follow the execution flow.
- Satisfy all the following requirements.
- Refer to the following implementation styles.
- Everything except the execution flow and the guidelines are free to choose/design/implement.

**<u>Remind that you may follow the honor code.</u>**

# Requirement 0: General (2/2)

- All the numbers in the input matrix, the filter, and the output (output matrix) are **8-bit unsigned integers**.

- If a calculation result incurs overflow, simply use the lower 8 bits only.

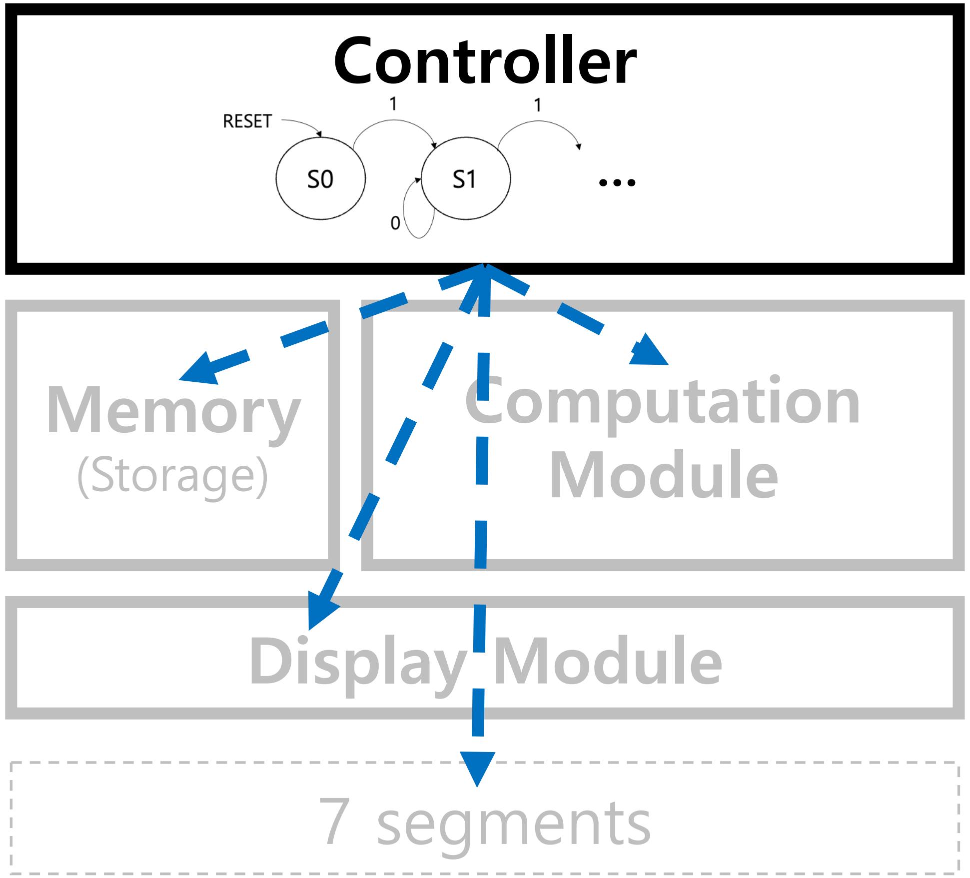
# Requirement 1: Controller

- Design your controller with state machine(s).
(**Week 10**)

- Design a complete flow
for initializing and taking controls of all the modules.
(Memory, computation modules, and display modules)

- Moore machine or Mealy machine?
→ Use what you prefer.

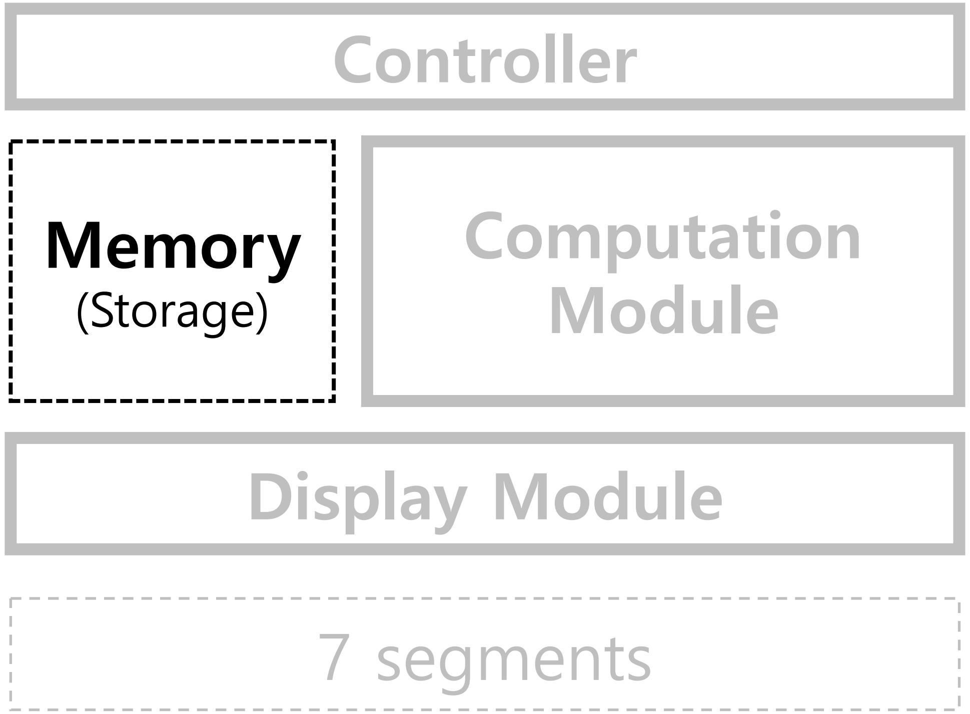
- Reset or Start: Design it as you want
  - You may use any buttons.
  - You may set an automatic reset or start.



# Requirement 2: Memory

- Implement it by yourself.

- **Memory** keeps and outputs the numbers of
  - An input matrix (16 numbers)
  - A filter (9 numbers)

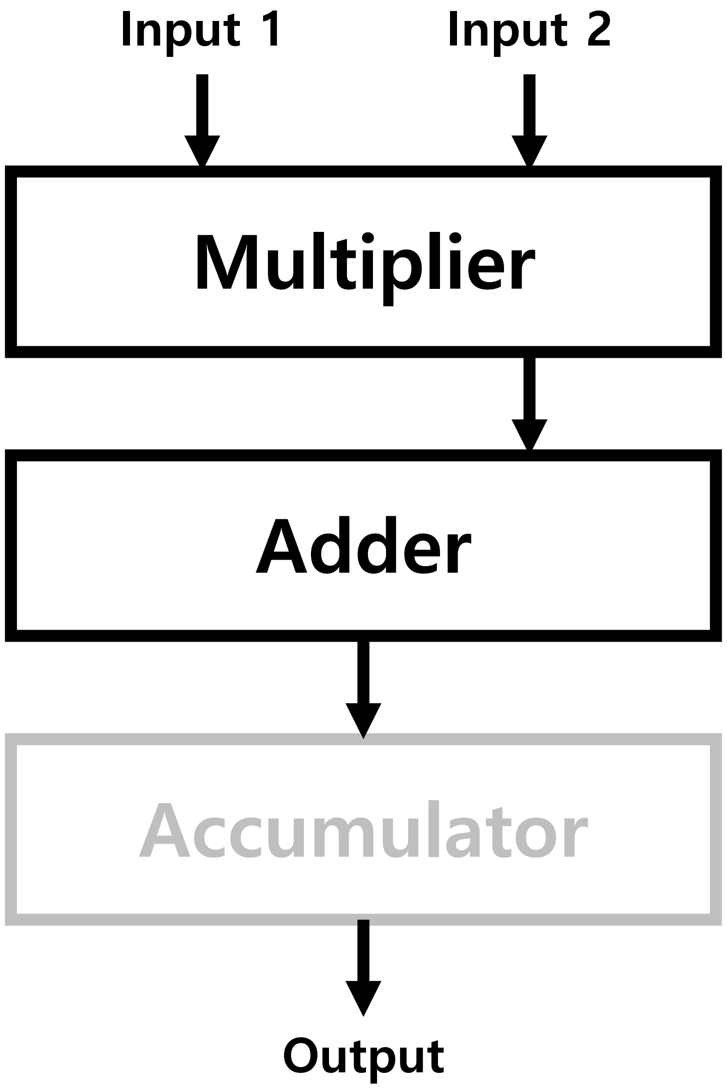


# Requirement 3: PE

- Three basic modules
  - Multiplier: You can implement it with adders.
  - Adder
  - Accumulator
    - It keeps an intermediate calculation result.
    - Decide the case that we need it.
    - If you don't need it at all, it's fine not to use it.
  - Any other resources?

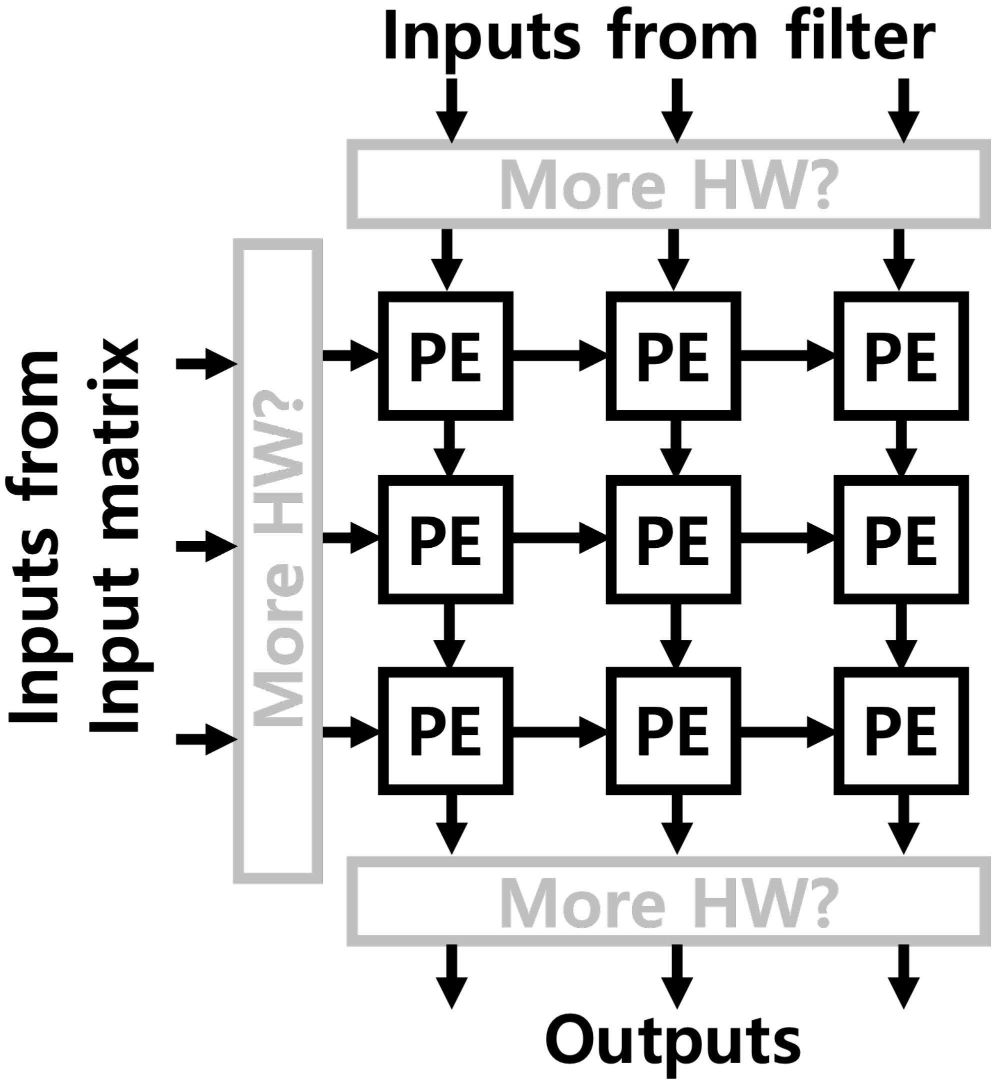
- Implement the modules by reminding
  - Three modeling methodologies (**Week 2**)
  - Structural modeling (**Week 6**)
  - Adders with what? (**Week 3**)
  - Multipliers with what? (**Week 3**)
  - Registers with what? (**Week 8**)

- **You may want to reuse your source codes implemented in previous labs.**



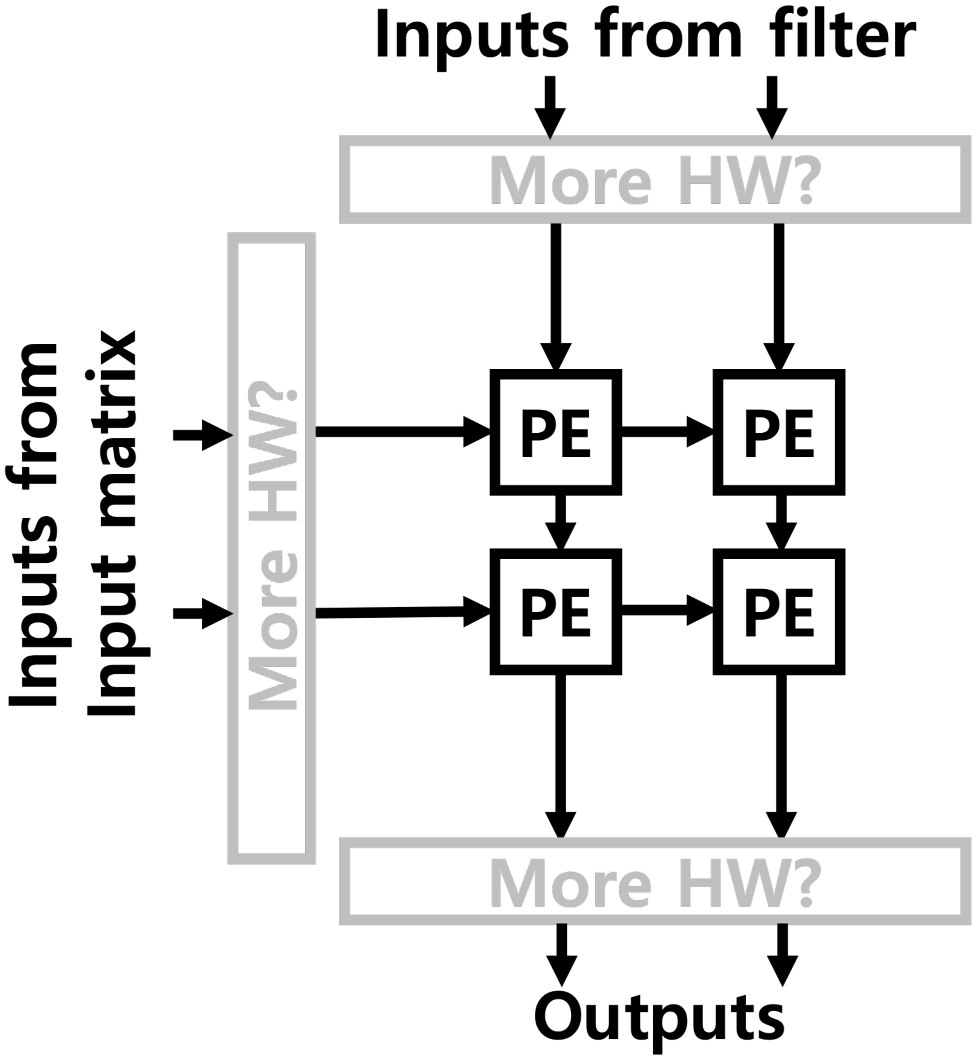
# Requirement 4: Systolic Array (1/2)

- Implement
  a 3x3 systolic array.
- Implement the modules by structural modeling (**Week 6**).
- You may add any resources if needed.
  - Just, be clear about your design.
- Refer to an example. (Systolic_Array_Example.pdf)



# Requirement 4: Systolic Array (2/2)

- Implement a 2x2 systolic array.
- Implement the modules by structural modeling (**<u>Week 6)</u>**.
- You may add any resources if needed.
  - Just, be clear about your design.
- Change the dataflow of the input matrix and the filter accordingly.
  - You should implement a different dataflow from the dataflow shown in "Systolic_Array_Example.pdf."



# Requirement 5: Display Flow

| | |
|---|---|
| 12 | 21 |
| 1 | 255 |

Output

Values calculated by a 3x3 systolic array

12

21

1

255

A reasonable time gap (e.g., 1 sec)

| | |
|---|---|
| 12 | 21 |
| 0 | 255 |

Output

Values calculated by a 2x2 systolic array

12

21

0

255

**Wrong. Debug it.**

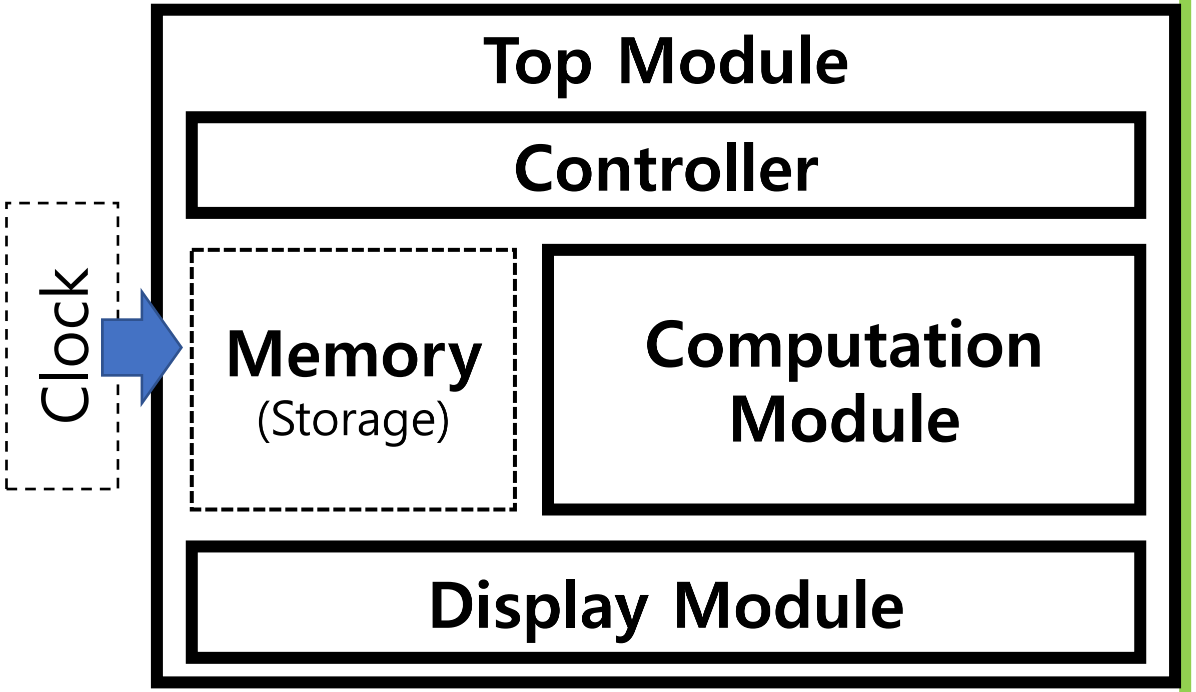
# Requirement 6: Demonstration

- Once you are done with implementing Requirements 1--5, let me or TAs know.
- We will give you a random 4x4 input matrix and a random 3x3 filter.
- Apply them to your source codes and show your calculation results following Requirement 5.
- If everything is ok, your demonstration is succesful!
- Results from the final demo will be evaluated.

# Requirement 7: Top Module

- You may want to implement a top module.

- Top module
  - Includes all the modules.
  - does not include a test pattern.
  - Triggers initialization and makes your design work.
  - Interfaces with external components of your modules. (e.g., clock)




- Top module just activates all the modules and make them work.

# Requirement 8: Testbench

- Create a test bench file for each module.

- Report the simulation results,
  which you have been doing throughout this semester.
  - You may use ModelSim.

# Requirement 9: Collaboration

- **<u>Each member should implement at least one module and one testbench.</u>**

- All members review source code each other and make all the codes best.

- **<u>Have deep discussions regarding interfaces between the modules.</u>**

- Clearly write down what I emphasized above in your final report.

# Implementation Styles

- You would need structural modeling (**Week 6**) everywhere.

- Along with the structural modeling methodology, use either
  - Behavioral modeling: Easiest except for what?
  - Gate-level modeling: Hardest except for what?
  - If you use a more difficult methodology, I'll give a higher score.
  - Try to reuse the source code that you implemented during this semester.

- If you employ gate-level modeling, use
  - AND, OR, NOT gates that the instructor provided
  - XNOR gate

- For the controller, just use "behavioral modeling."

# Available Questions

**Please find out what to do by yourselves.**

- Remind that this is a project.
- Questions regarding the following will **never** be answered.
  - Specifications of the modules in this project
    ex) PE and systolic array structures, memory structure, etc.
  - Abstract questions
    ex) I can't compile. How can I fix it?

# Overall evaluation criterion

- **Simulation demonstration**
  - Will be evaluated real-time in each class

- **Report**
  - 1 report per group
  - Page limit: 25 pages
  - What to include
    - Brief description about the project's objective
    - Theoretical design (I believe that you all figure out what is needed.)
    - Simulation results and discussions (waveform, analysis, etc.)
    - Contributions from each team member.

- **Source code files**

# Share Your Questions

- Ask questions via
  - Q&A board in i-Campus (responses may be delayed)
  - Offline Q&A sessions

- **<u>Questions asked via any private channels, including</u>**
  - e-mails
  - i-Campus messages
  - phone calls

  **<u>will *not* be answered.</u>**

# Your Source Codes

- Upload to i-Campus
  - Copy all the files into the project folder.
  - Submit the zip file with your report to i-Campus.

# Deadline

- Refer to i-Campus.
- I STRONGLY recommend avoiding late submission.
- Each delay in submission of a second after the deadline will result in penalty in the attendance score.
- Each delay in submission of a day and more after the deadline will result in….
  → 0 score for the corresponding report.
  → **Regarded as an absence.**