# Edge-AI Accelerator in 2-nm Technology with Data Dependent Multicore Processing

**June 23rd 2026**

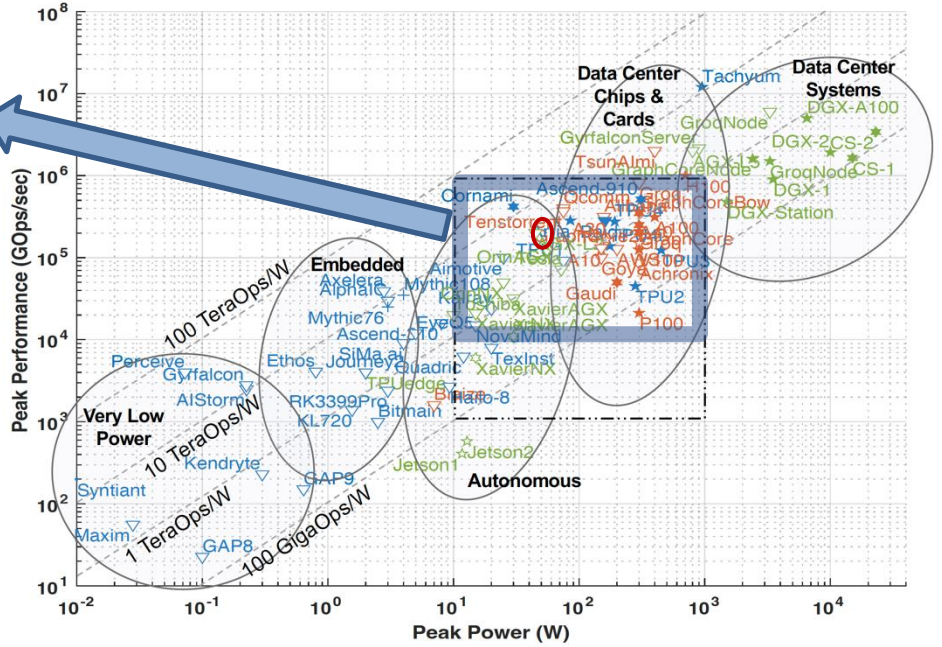
**Fumio Arakawa**
d.lab, The University of Tokyo

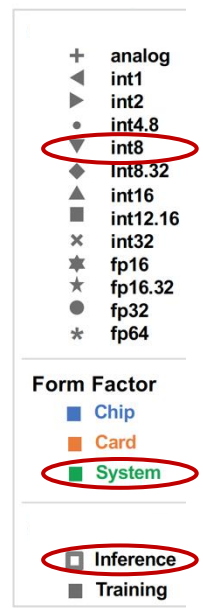
# Outline

- Introduction: Various AI Accelerators and Target
- Project Overview: Arch. definition, Chip Developments, SW development kits
- Important Application: Large Language Model (LLM) – Large Matrix Processing
- Our Approach: **Data-dependent partition**, **broadcast of load data** & **remote store**
  - Processing order definition by **valid** bit of each register & **keep** option (introduced last year)
- Compute-Node Array Overview
  - **Large nodes for efficient matrix processing**
- **Memory access & data transfer efficiency of various processing partitions**
- Summary

# Various AI Accelerators and Target

- Many Accelerators have been announced by NVIDIA, Google, Tenstorrent, etc.
- NVIDIA's **Jetson AGX Orin:** 138 (275) TOPS, 60 W, **2.3 (4.6) TOPS/W**, Int8 (w/ sparse accel.)
  **DRIVE AGX L2:** 200 TOPS, 45 W, **4.4 TOPS/W**, Int8
- Our PJ target is to realize much higher efficiency (Upper left direction).

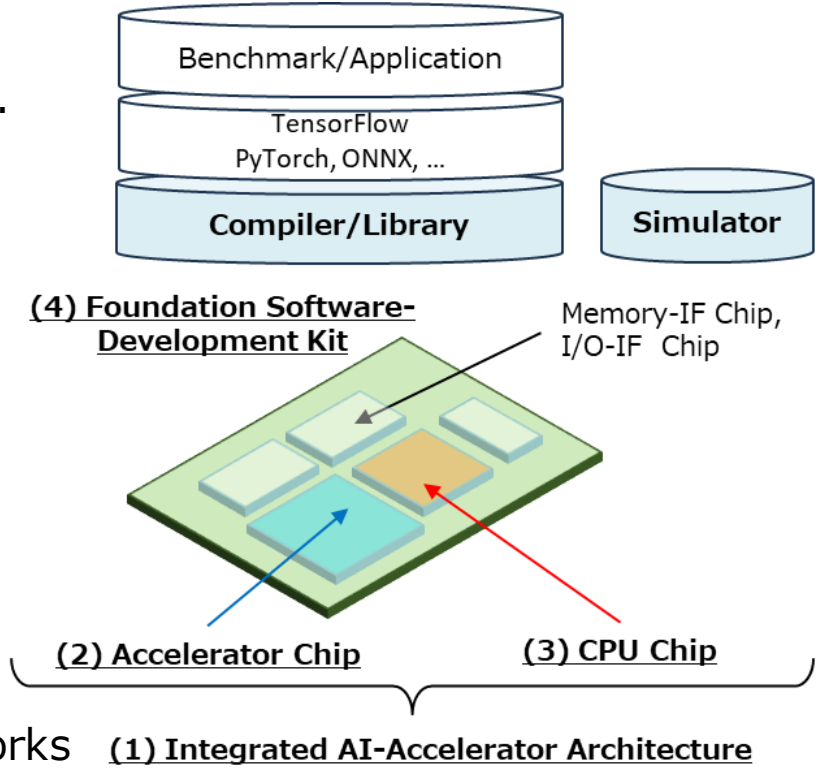




source: Albert Reuther, et al., "AI and ML Accelerator Survey and Trends," 2022 IEEE High Performance Extreme Computing (HPEC) Conference (Oct 2022)

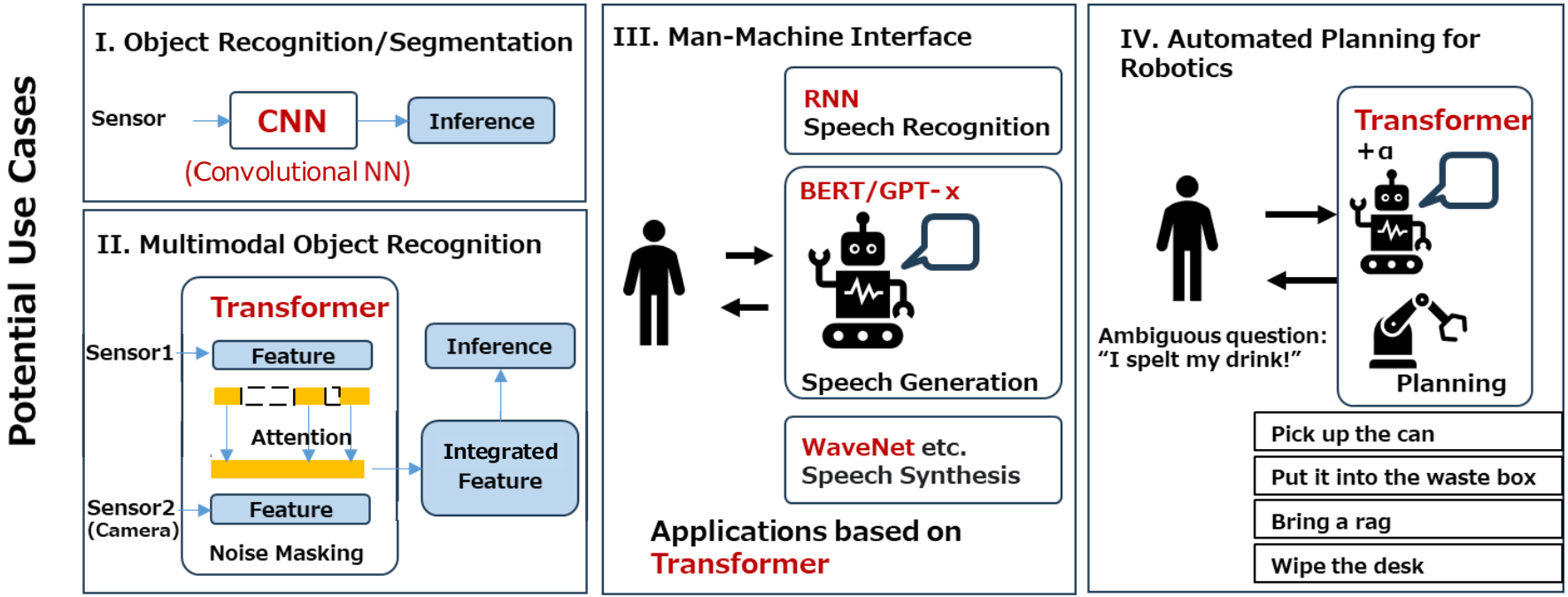
# Project Overview

- Leading-edge Semiconductor Technology Center (LSTC)
  - member: AIST, Rapidus, The Univ. of Tokyo
- International collaboration with Tenstorrent inc.
- Development items
  - ① **AI-accelerator architecture**
    - integrating accelerator and CPU chips
    - HW-SW codesign
  - ② **Accelerator chip**
    - New highly-efficient Architecture
  - ③ **CPU chip**
    - conforming to RISC-V ISA
  - ④ **Software development kit**
    - conforming to de-fact-standard AI flameworks



# Potential Use Cases of edge-AI accelerator

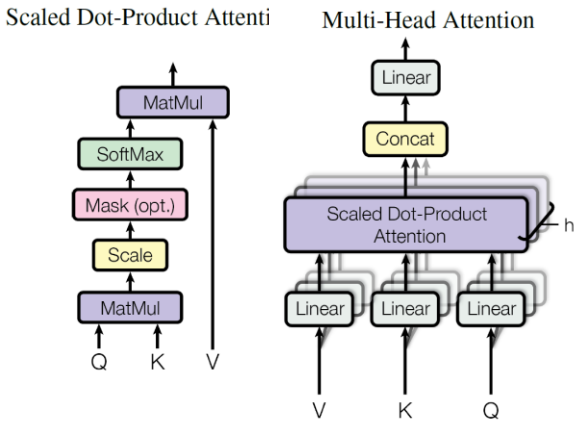
- Factory automation, Automotive, Retailing, Healthcare, Robot assistant, etc.

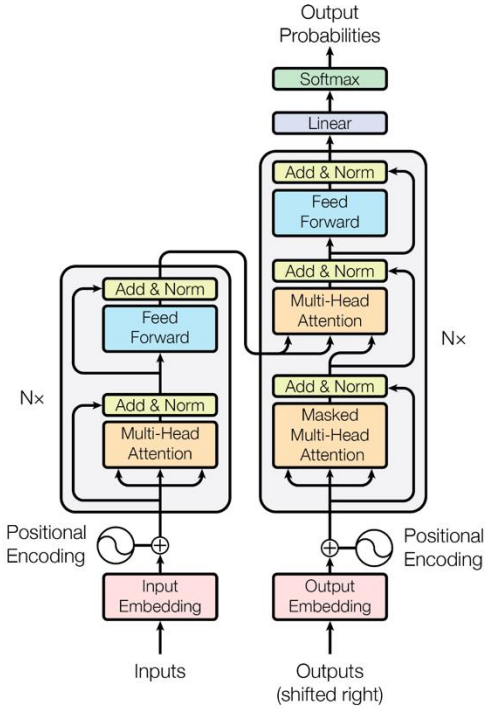


# Background

- Large Language Model (**LLM**), an important **AI** application
  - multiplication of **large** matrices (ex. 4,096-by-4,096) with **many** arithmetic units
- Transfer the data **timely to registers**
  - Cannot to keep all large-matrix data near the arithmetic units
- Cost and Power in the advanced process (We use **2nm**)
  - calculation : decreased
  - transfer: **relatively increased**

- **Efficient data transfer: more and more important !!**

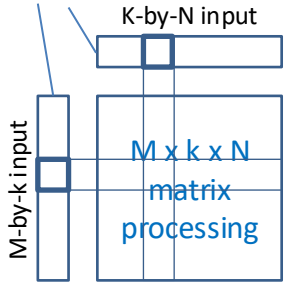


# Merit of matrix processing and our matrix block (MB)

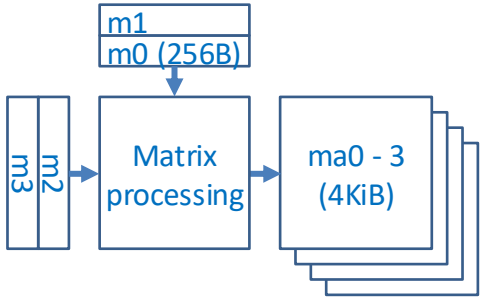
- **Less data transfer** per calculation ➔ Many AI accelerators have matrix units
  - Difference is particularly noticeable in **large matrices**.
  - **1/64** [1] for 64 x k x 64 **matrix** processing (vs. scalar processing)

Each data is used N or M times.

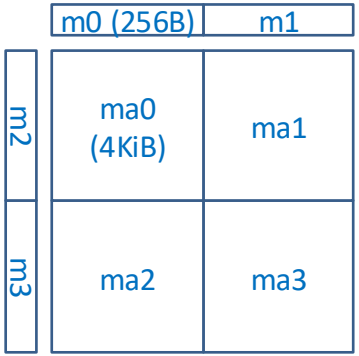



1) M x k x N: **MN** sums of **k** products, ie. **MNk** calc. w/ **(M+N)k** operands
(M+N)k/MNk = **1/N + 1/M** operands/calc. (**Scalar**: **2** operands/calc.)
matrix vs. scalar ratio: **(1/N + 1/M)/2** = (1/64+1/64)/2 = 1/64, for n=m=64

- Matrix block (MB) of each node
  - **32** x **8** Byte (B) x **32** matrix processing
  - 8B = **4**x2B (bf16) or **8**x1B (fp8 or int8)
  - input regs.: **4** x 256B (32 x 8B), m0–m3
  - acc. regs.: **4** x 4KiB (32 x 32 x 4B), ma0–ma3
  - **64** x **8**B x **64** processing per **4** cycles per node




Matrix block




64 x 8B x 64 processing

# Our Approach

- **Data-dependent partition** & **multicore** assignment of processing
  - Conventionally, only data-**in**dependent partition is allowed. ← Independent processing & Sync.
- Load & **broadcast** data to **multiple** compute nodes (CN).
  - as **various shape of matrices**: 1/4/8/32 rows of 256/64/32/4 Bytes
  - Broadcasting to Max. **8 & 4 nodes** (64 x 4 x 64 matrix processing/per node)

    → 512 x 4 x 256 matrix processing
  - # of transfers: (1/256 + 1/512)/2 = **3/1024** (vs. Single core with a Scalar unit)
- **Store** data **remotely** to shared memory (SM) of **another node**.
  - Each SM block can directly store the data from 4 input ports of NoC router.
  - Remote stores reduce # of loads/stores for inter-node data exchange.

# Data-dependent partition of processing

- Significant effect of **reducing data loads** and **memory footprint**

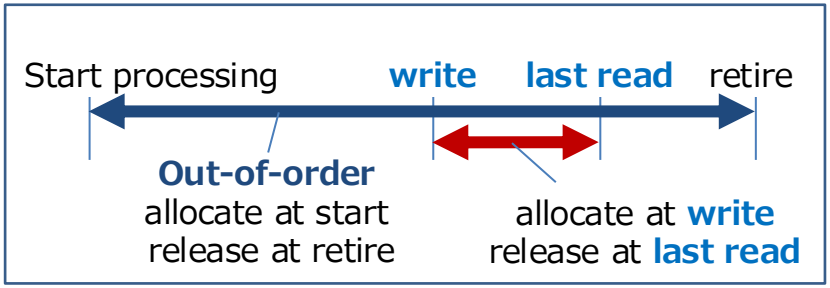
- Order of data definition and use
  - Single core w/ Single flow: Naturally defined ➡ **collapsed** by Data-dependent partition

- Order definition by **Valid bit** of each register
  - **Setting**/**Clearing** valid bit when data is **written**/**lastly read**, respectively
  - Wait **write**/**read** until register becomes **invalid**/**valid**, respectively

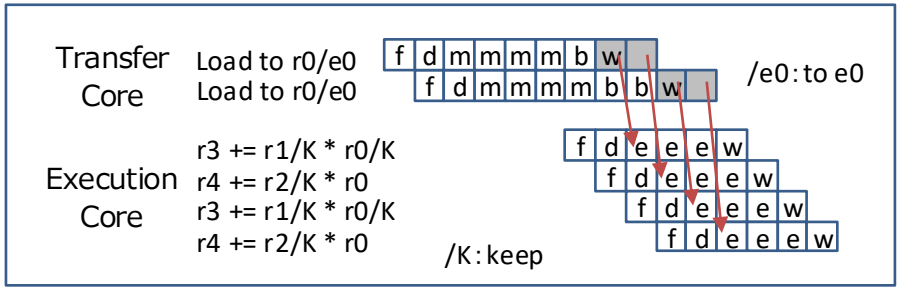
- **This ensures correct processing w/ data-dependent partition.**

# Short Register Lifetime

- Allocate a register only while register value is necessary (**write** to **last read**).
- **Overrun buffer** ensures correct operations even if register allocation is failed.
- Perfect load timing control by software with hardware assistance also ensures it.
- Hardware can identify "**last read**" with "**valid**" bit and "**keep**" option.




Short register lifetime of **valid** & **keep** method

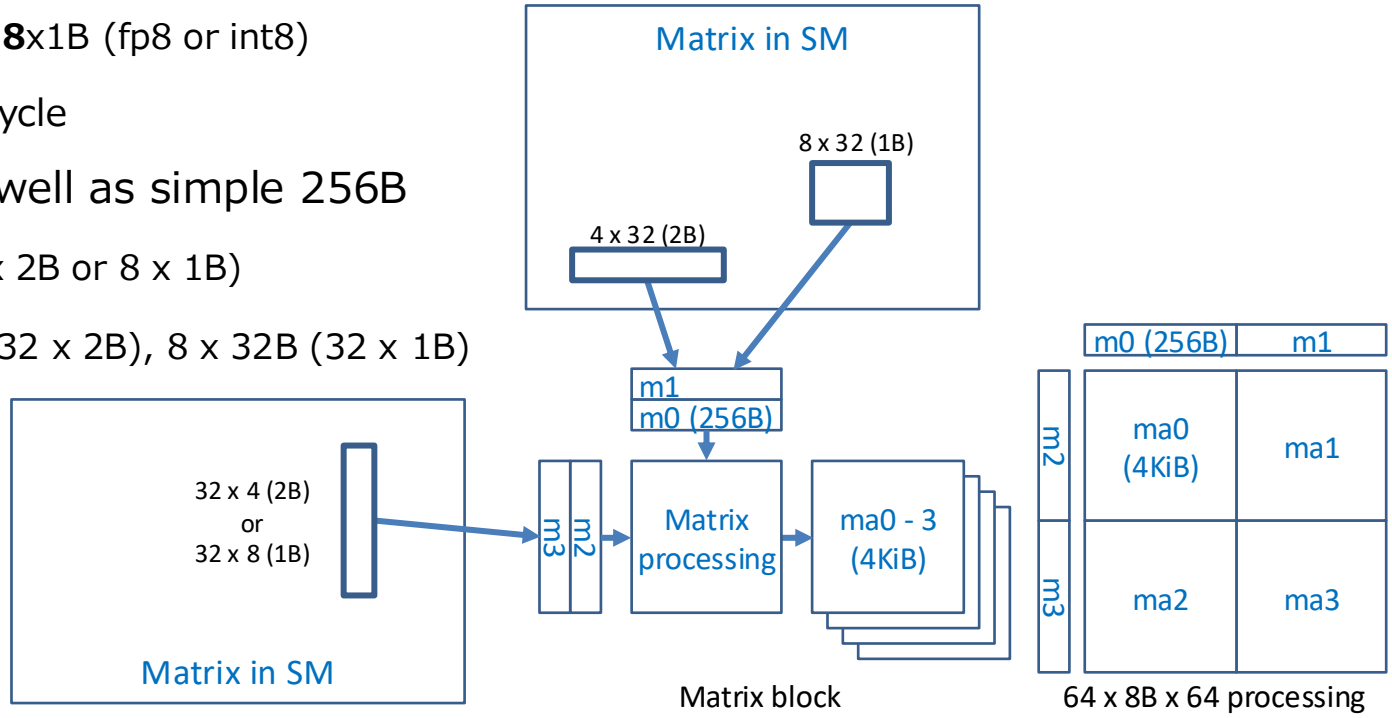



Example pipeline flow of short register lifetime

- **Drastically reduce # of registers required, compared to out-of-order method.**

# Various-shape loads of matrices

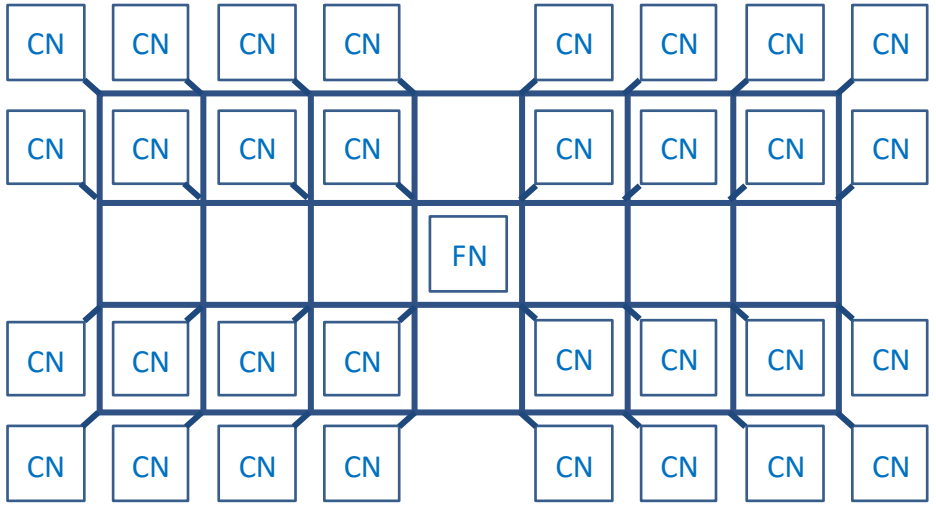
- **Multiple bank & data alignment of SM** for various-shape loads of matrices
- Input for **64 x 8B x 64** matrix processing
  - 8B = **4**x2B (bf16) or **8**x1B (fp8 or int8)
  - **32 x 8B x 32** per cycle
- **3-shape** loads as well as simple 256B
  - Vertical: 32 x 8B (4 x 2B or 8 x 1B)
  - Horizontal: 4 x 64B (32 x 2B), 8 x 32B (32 x 1B)



# Compute-Node Array Overview

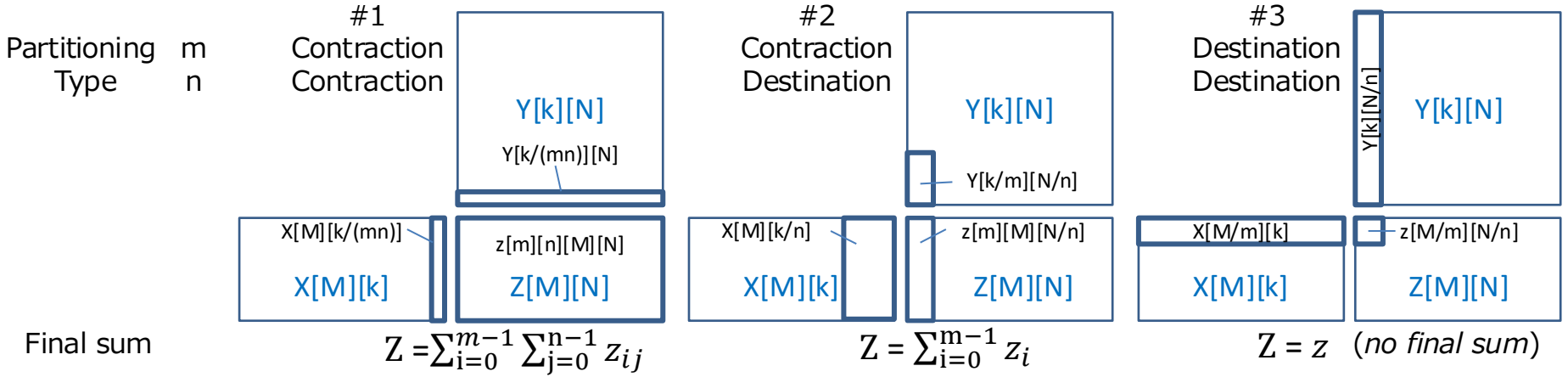
- Compute node (CN): EC + TC + SM (Shared memory)
  - **Large node for efficient matrix processing**
  - Execution Core (EC): **64x4x64** matrix contraction per 4 cycles
  - Transfer Core (TC): EC-sustainable load & Broadcast, inter-CN remote store
- Flow control node (FN): FCC + IM
  - Flow control core (FCC):

    Load and dispatch instructions to all CNs.
  - IM (Instruction memory)
- NoC Connects 32 CNs.
  - 4-row & 8-column shape



# Partitioning & Assignments of Large Matrix Multiply

Matrix multiply **Z = XY** to compute node **CN[m][n]** array

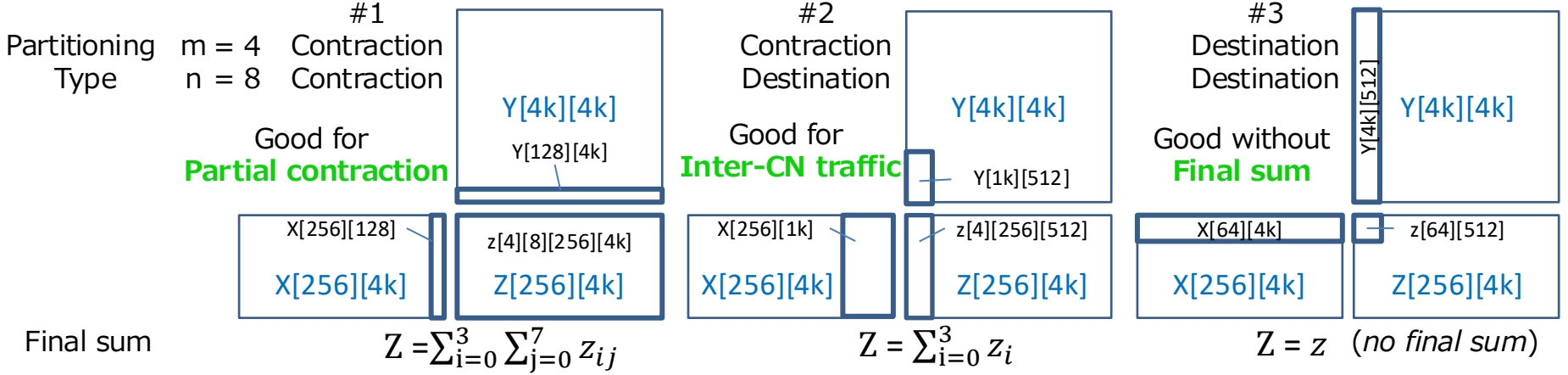



| | | | | | | |
|---|---|---|---|---|---|---|
| Setup & load: X, Y | Mk(1+N/64), | kN(1+M/64) | **Mk(1+N/64n) (n)**, | kN(1+M/64) | **Mk(1+N/64n) (n)**, | **kN(1+M/64m) (m)** |
| Store: remote, local | MN(mn-1) **(—)**, | MN **(mn)** | MN(mn-1) **(—)**, | MN **(m)** | **—**, | **MN** |
| Final : load, store | MNmn **(2-1/mn)**, | MN **(mn)** | MNm (2-1/m)), | MN **(m)** | **—**, | **—** |
| Mem. footprint: X, Y, Z | **Mk**, **kN**, MN(mn+1) **(2MNmn)** | | **Mk**, | **kN**, MN(m+1) **(2MNm)** | **Mk (n)**, | **kN (m)**, **MN** |
| Inter-CN traffic: X, Y, Z | **Mk**, **kN**, **MN(mn-1)** | | Mk(1+N/64n) **(Mkn)**, | **kN**, MN(m-1) | Mk(1+N/64n) **(Mkn)**, | kN(1+M/64m) **(kNm)**, **—** |

Assuming **64**x**4**x**64** Contraction Per load **(ratio)**, **(value)**: w/o broadcast/remote-store if different from w/ them

# Partitioning & Assignments of Large Matrix Multiply

Matrix multiply **Z = XY** to compute node **CN[4][8]** array




| | | | |
|---|---|---|---|
| Setup & load: X, Y | 65M, 80M | **9M** (**72M**), 80M | **9M** (**72M**), **32M** (**128M**) |
| Store: remote, local | 31M (—), **1M** (**32M**) | 3M (—), **1M** (4M) | —, **1M** |
| Final : load, store | 32M (**63M**), 1M (**32M**) | 4M (7M), 1M (4M) | —, — |
| Mem. footprint: X, Y, Z | **1M**, **16M**, 33M (**64M**) | 1M (8M), 16M, 5M (8M) | **1M** (**8M**), **16M** (**64M**), **1M** |
| Inter-CN traffic: X, Y, Z | **1M**, **16M**, **31M** | **9M** (8M), **16M**, **3M** | **9M** (8M), **32M** (**64M**), — |

Assuming **64**x**4**x**64** Contraction Per load

( ): w/o broadcast or remote store if different from w/ them

# Partitioning & Assignments of Large Matrix Multiply

Matrix multiply **Z = XY** to compute node **CN[4][8]** array

| Partitioning Type (m = 4, n = 8) | #1 Contraction Contraction | #2 Contraction Destination | #3 Destination Destination |
|---|---|---|---|
| Pros | **Local & efficient partial contraction** | **Least inter-CN traffic** Less mem. size & access w/ broadcast & remote store | **No final sum** **Least mem. size & access w/ broadcast** |
| Cons | **Most mem. size & access, inter-CN traffic, & calc. for final sum** | More mem. size & access & calc. than #3 for final sum | More inter-CN traffic than #2 for more broadcast |
| Final sum | $Z = \sum_{i=0}^{3} \sum_{j=0}^{7} z_{ij}$ | $Z = \sum_{i=0}^{3} z_i$ | $Z = z$ (*no final sum*) |
| Setup & load: X, Y | 65M, 80M | **9M** (**72M**), 80M | **9M** (**72M**), **32M** (**128M**) |
| Store: remote, local | 31M (—), **1M** (**32M**) | 3M (—), **1M** (4M) | —, **1M** |
| Final : load, store | 32M (**63M**), 1M (**32M**) | 4M (7M), 1M (4M) | —, — |
| Mem. footprint: X, Y, Z | **1M, 16M,** 33M (**64M**) | 1M (8M), 16M, 5M (8M) | **1M** (**8M**), **16M** (**64M**), **1M** |
| Inter-CN traffic: X, Y, Z | **1M, 16M, 31M** | **9M** (8M), **16M**, **3M** | **9M** (8M), **32M** (**64M**), — |

Assuming **64**x**4**x**64** Contraction Per load

( ): w/o broadcast or remote store if different from w/ them

# Summary

- Large Language Model (LLM) Support
- **Efficient data transfer is more and more important.**
- **Data-dependent division** & **multicore** assignment of processing
  - **Short Register Lifetime ➔ Drastically reduce # of registers required**
- Transfer core **broadcasts** memory load data to **multiple execution** cores.
  - Significant effect of **reducing data loads** and **memory footprint**
- Connect 32 Compute nodes (CNs) and a Flow control node (FN) by NoC
  - **Large nodes for efficient matrix processing, Less data transfer** per processing ➔ **1/64**
  - 64 x 8B x 64 **Matrix processing** per 4 cycles per CN
- **Memory footprint & inter-CN traffic of partitioned large-matrix multiply**

# Acknowledgement


- This presentation is based on results obtained from “Research and Development Project of the Enhanced Infrastructures for Post 5G Information and Communication Systems” JPNP20017), commissioned by the New Energy and Industrial Technology Development Organization (NEDO).


**Thank you !!**