Graduate Institute of Electronics Engineering, NTU

Undergraduate Project – Compute-in-Memory (CIM) Study and DNN+NeuroSim Simulation Results

Speaker: 吳育丞 吳冠緯 陳宥辰

Mentor: Kevin Chris

Advisor: Prof. An-Yeu (Andy) Wu

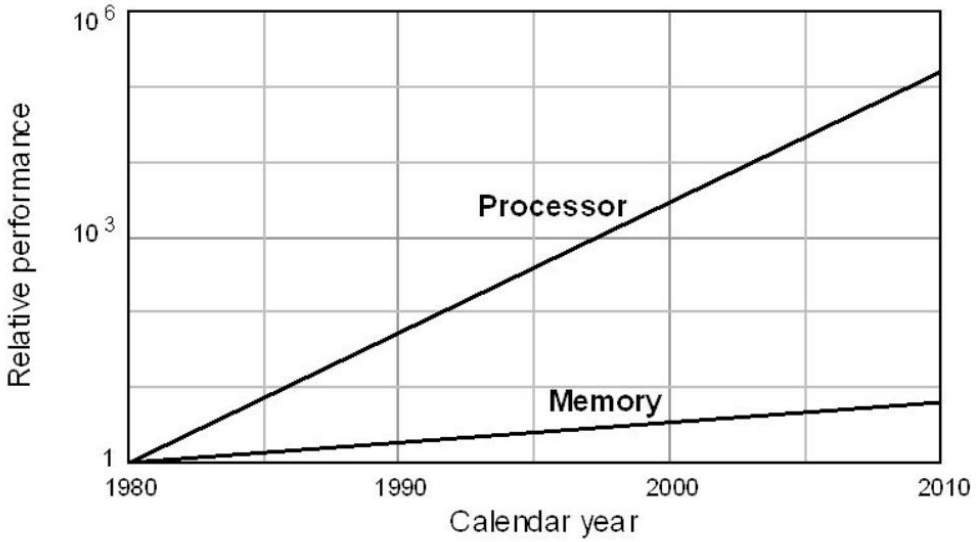
Date:12/01/2021

Outline

- CIM
- DNN+NeuroSim
  - Benchmark methodologies
  - Python wrapper
  - C++ simulation
- ResNet-20 Inference Simulation Results

CIM Basics

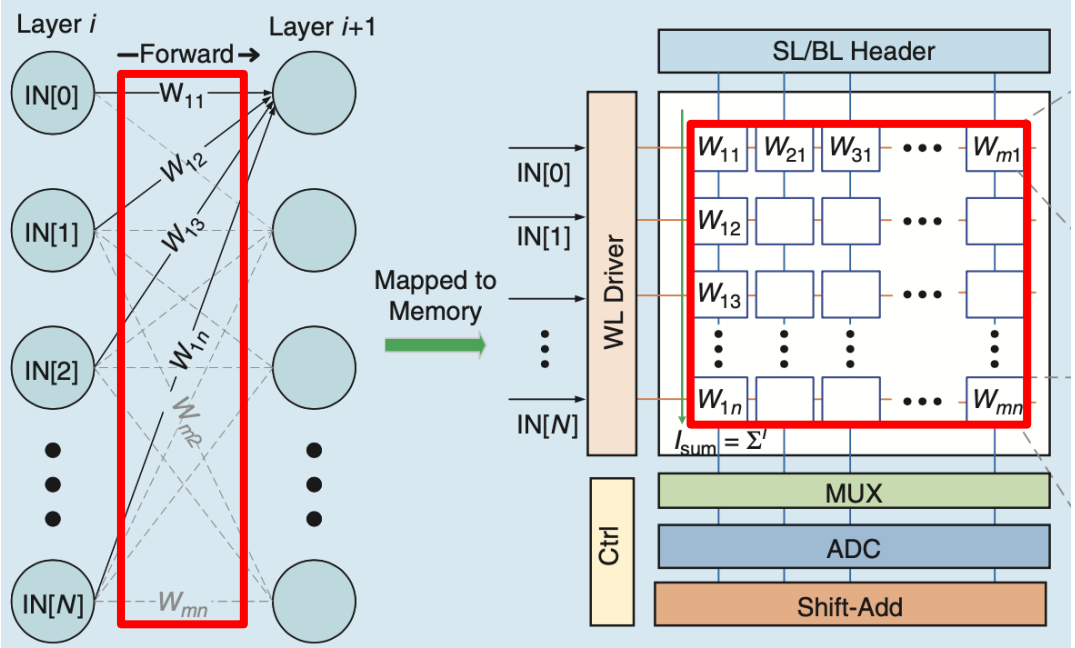
- **CIM** addresses the **memory-wall problem** in hardware accelerator design for **deep learning**.



Memory density and capacity have grown along with the CPU power and complexity, but memory speed has not kept pace.

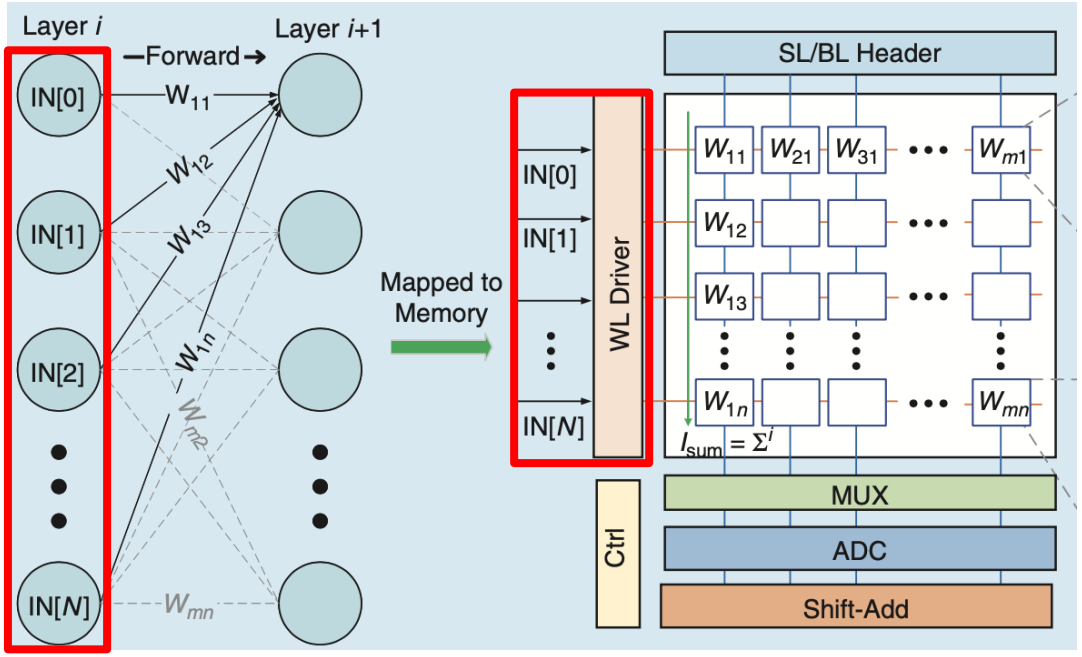
CIM Basics

- A layer of neural network is mapped to the memory sub-arrays and the **weights** could be mapped as the **conductance of memory cells**.
- **Inputs** are loaded in parallel as **voltage** to activate multiple rows, and **column currents are summed up and digitalized** (**partial sum** due to limited sub-array size) by **ADC**.



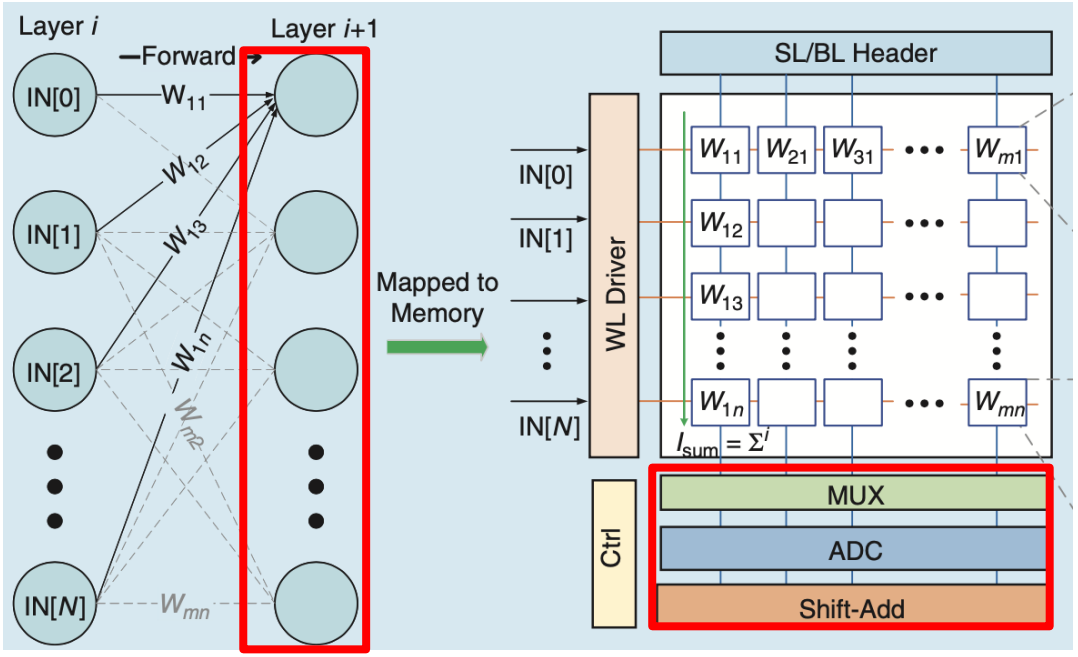
CIM Basics

- A layer of neural network is mapped to the memory sub-arrays and the **weights** could be mapped as the **conductance of memory cells**.
- **Inputs** are loaded in parallel as **voltage** to activate multiple rows, and **column currents are summed up and digitalized** (**partial sum** due to limited sub-array size) by **ADC**.



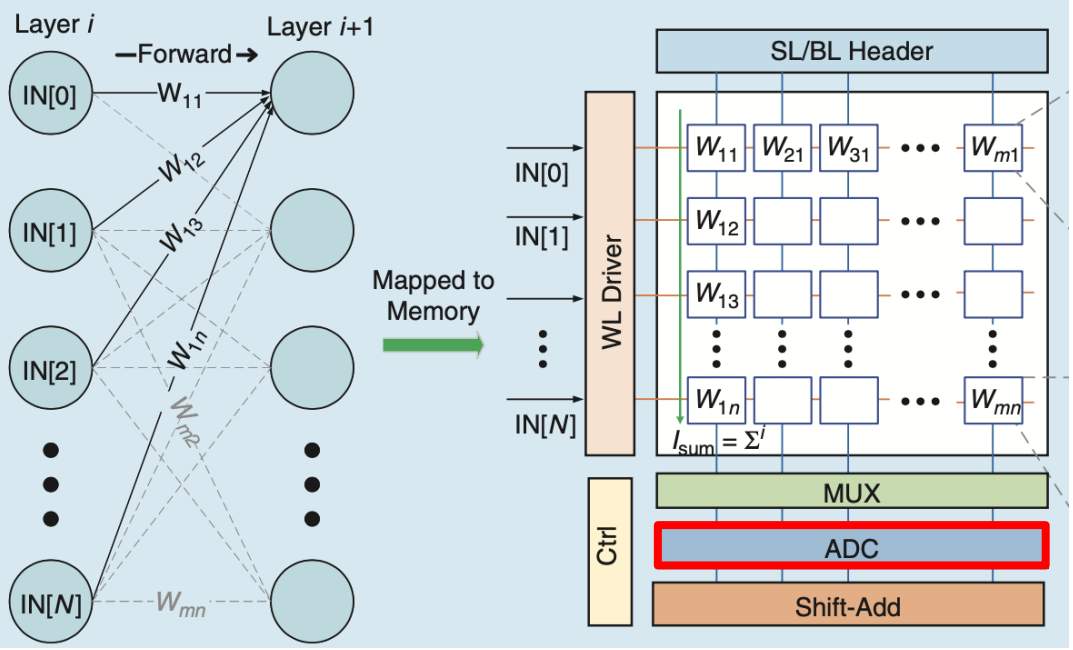
CIM Basics

- A layer of neural network is mapped to the memory sub-arrays and the **weights** could be mapped as the **conductance of memory cells**.
- **Inputs** are loaded in parallel as **voltage** to activate multiple rows, and **column currents are summed up and digitalized** (**partial sum** due to limited sub-array size) by **ADC**.



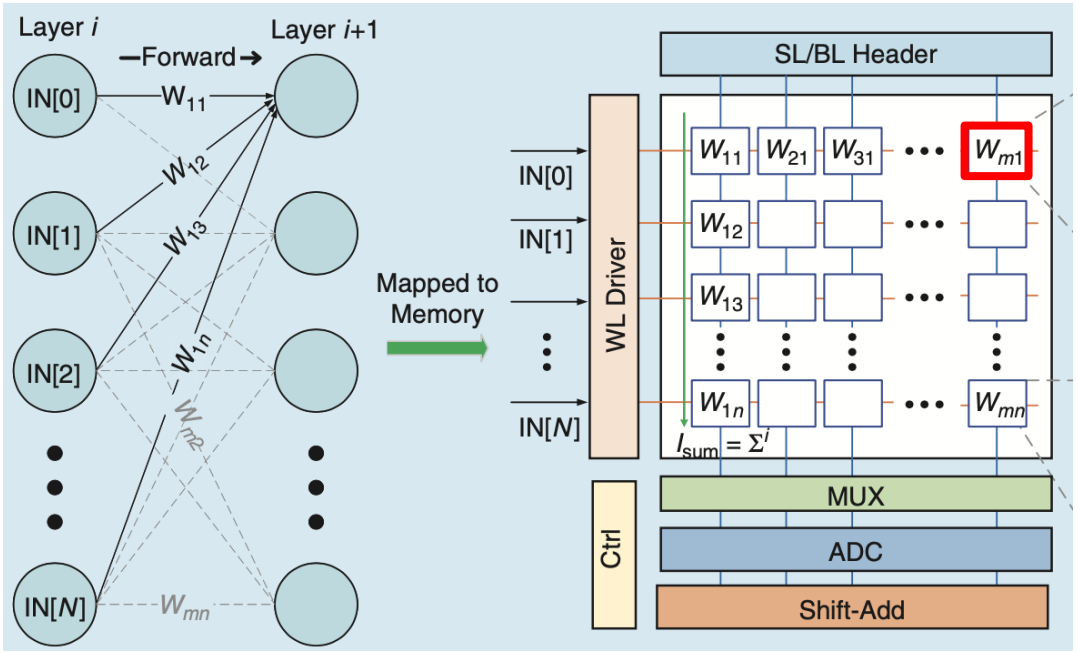
CIM current conditions & challenges

- So far, most of the reported **CIM** designs are targeted for **inference**.
- **ADC is a major bottleneck for area and power efficiency**.
- The **resolution of ADC** is determined by the **partial sum precision required from sub-array**.



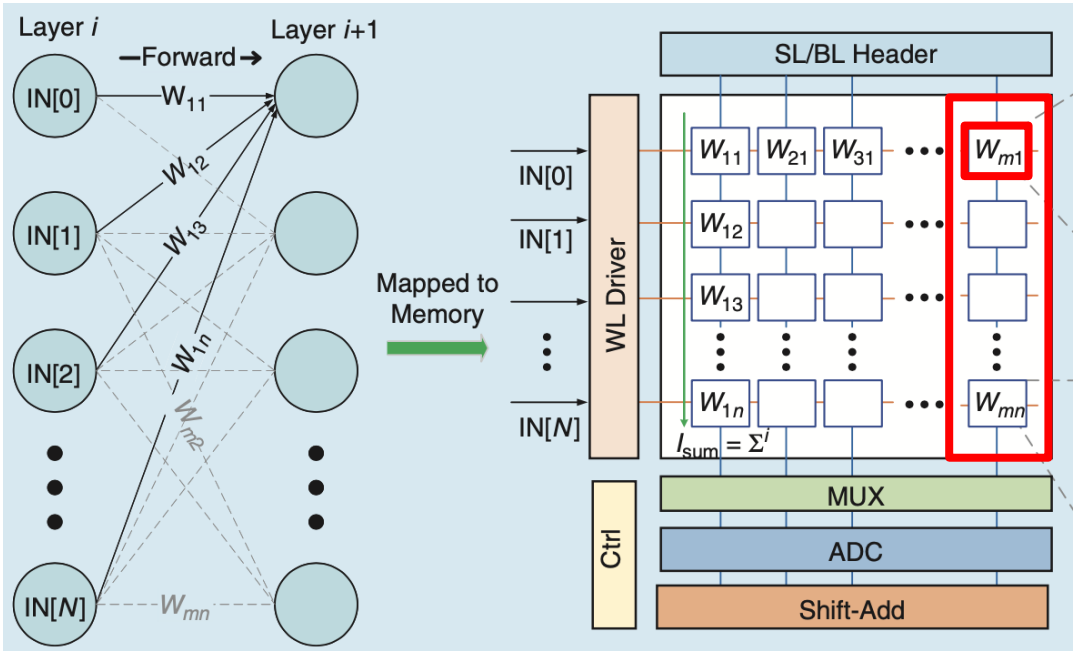
CIM current conditions & challenges

- So far, most of the reported **CIM** designs are targeted for **inference**.
- **ADC is a major bottleneck for area and power efficiency**.
- The **resolution of ADC** is determined by the **partial sum precision required from sub-array**.



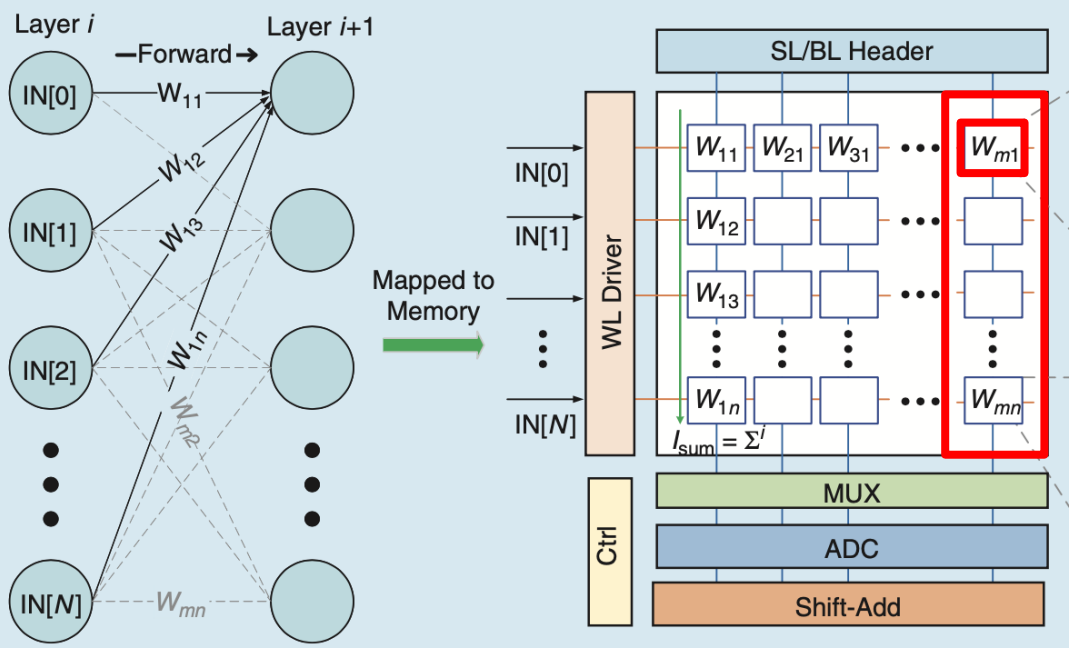
CIM current conditions & challenges

- So far, most of the reported **CIM** designs are targeted for **inference**.
- **ADC is a major bottleneck for area and power efficiency**.
- The **resolution of ADC** is determined by the **partial sum precision required from sub-array**.



CIM current conditions & challenges

- So far, most of the reported **CIM** designs are targeted for **inference**.
- **ADC is a major bottleneck for area and power efficiency**.
- The **resolution of ADC** is determined by the **partial sum precision required from sub-array**.

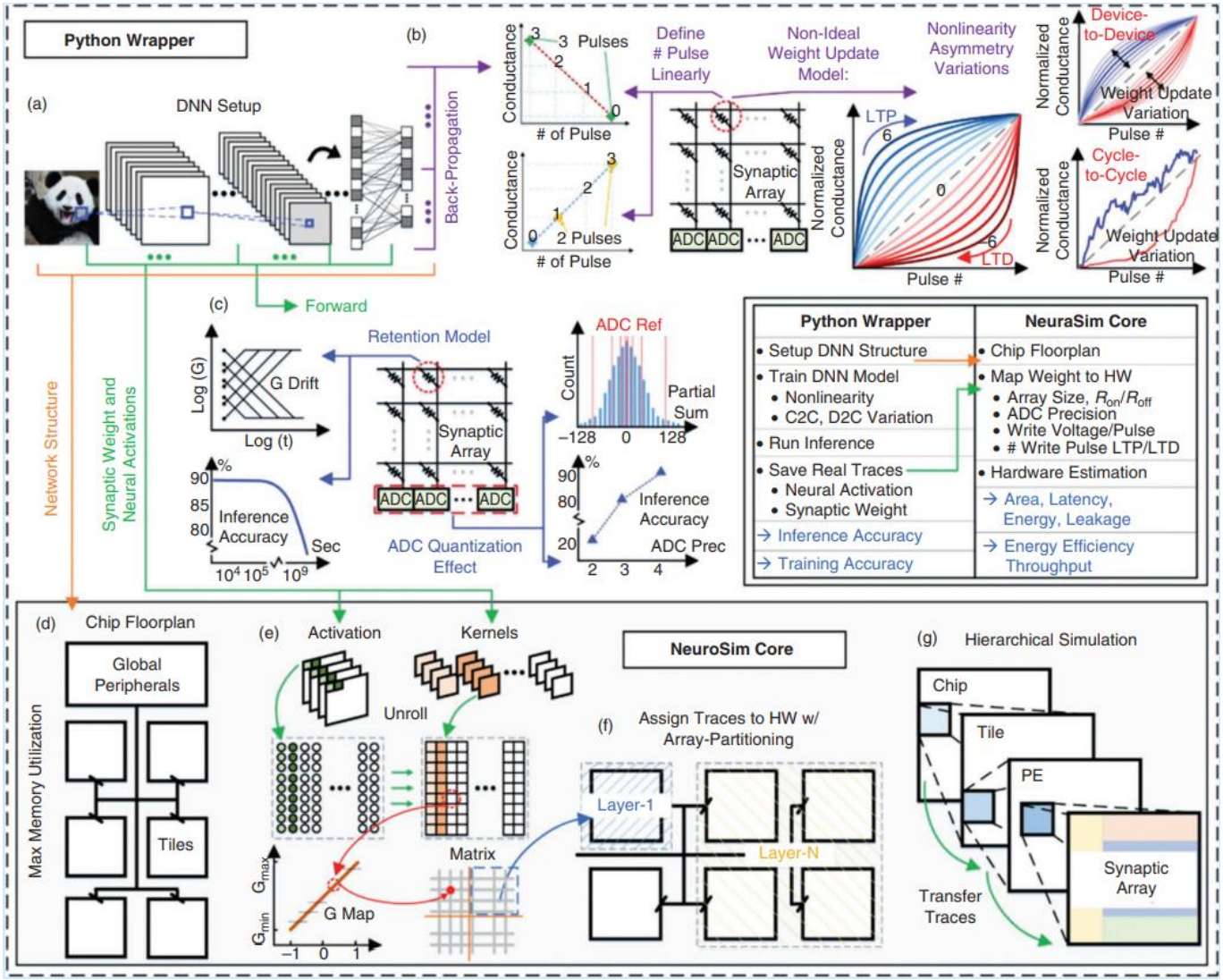


Outline

- CIM
- DNN+NeuroSim
  - Benchmark methodologies
  - Python wrapper
  - C++ simulation
- ResNet-20 Inference Simulation Results

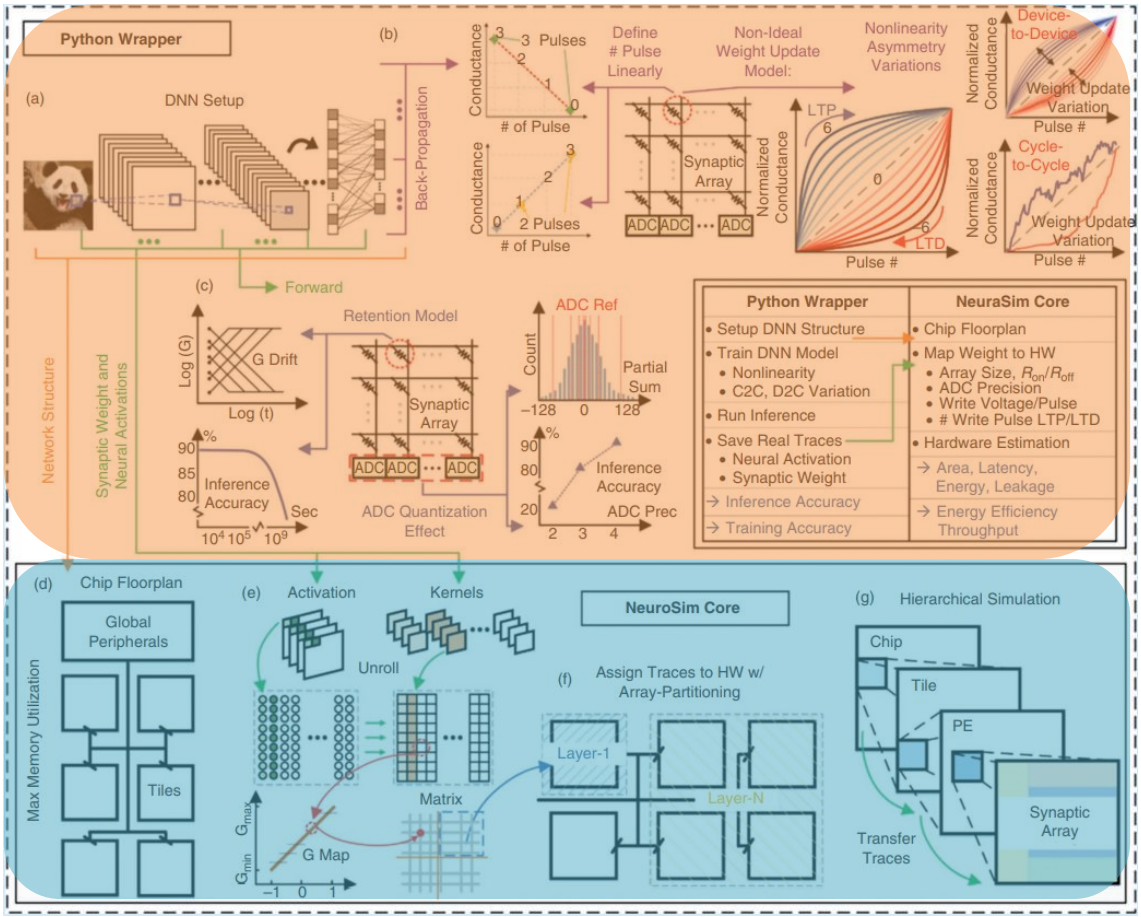
Why Benchmark ?

- Be capable of evaluating versatile device technologies for CIM inference and training performance from software / hardware co-design's perspective.


Python Wrapper
(a)
DNN Setup
(b)
Back-Propagation
Forward
Define # Pulse Linearly
Non-Ideal Weight Update Model:
Nonlinearity Asymmetry Variations
Conductance
3 Pulses
# of Pulse
2 Pulses
Synaptic Array
ADC
Normalized Conductance
LTP
LTD
Pulse #
Device-to-Device
Weight Update Variation
Cycle-to-Cycle
(c)
Retention Model
Log (G)
G Drift
Log (t)
Inference Accuracy
Sec
ADC Quantization Effect
ADC Ref
Count
Partial Sum
ADC Prec
Network Structure
Synaptic Weight and Neural Activations
Python Wrapper
NeuraSim Core
• Setup DNN Structure
• Chip Floorplan
• Train DNN Model
• Nonlinearity
• C2C, D2C Variation
• Map Weight to HW
• Array Size, R_{on}/R_{off}
• ADC Precision
• Write Voltage/Pulse
• # Write Pulse LTP/LTD
• Run Inference
• Save Real Traces
• Neural Activation
• Synaptic Weight
• Hardware Estimation
→ Inference Accuracy
→ Area, Latency, Energy, Leakage
→ Training Accuracy
→ Energy Efficiency Throughput
(d)
Chip Floorplan
Global Peripherals
Tiles
Max Memory Utilization
(e)
Activation
Kernels
Unroll
Matrix
G Map
NeuroSim Core
(f)
Assign Traces to HW w/ Array-Partitioning
Layer-1
Layer-N
(g)
Hierarchical Simulation
Chip
Tile
PE
Synaptic Array
Transfer Traces

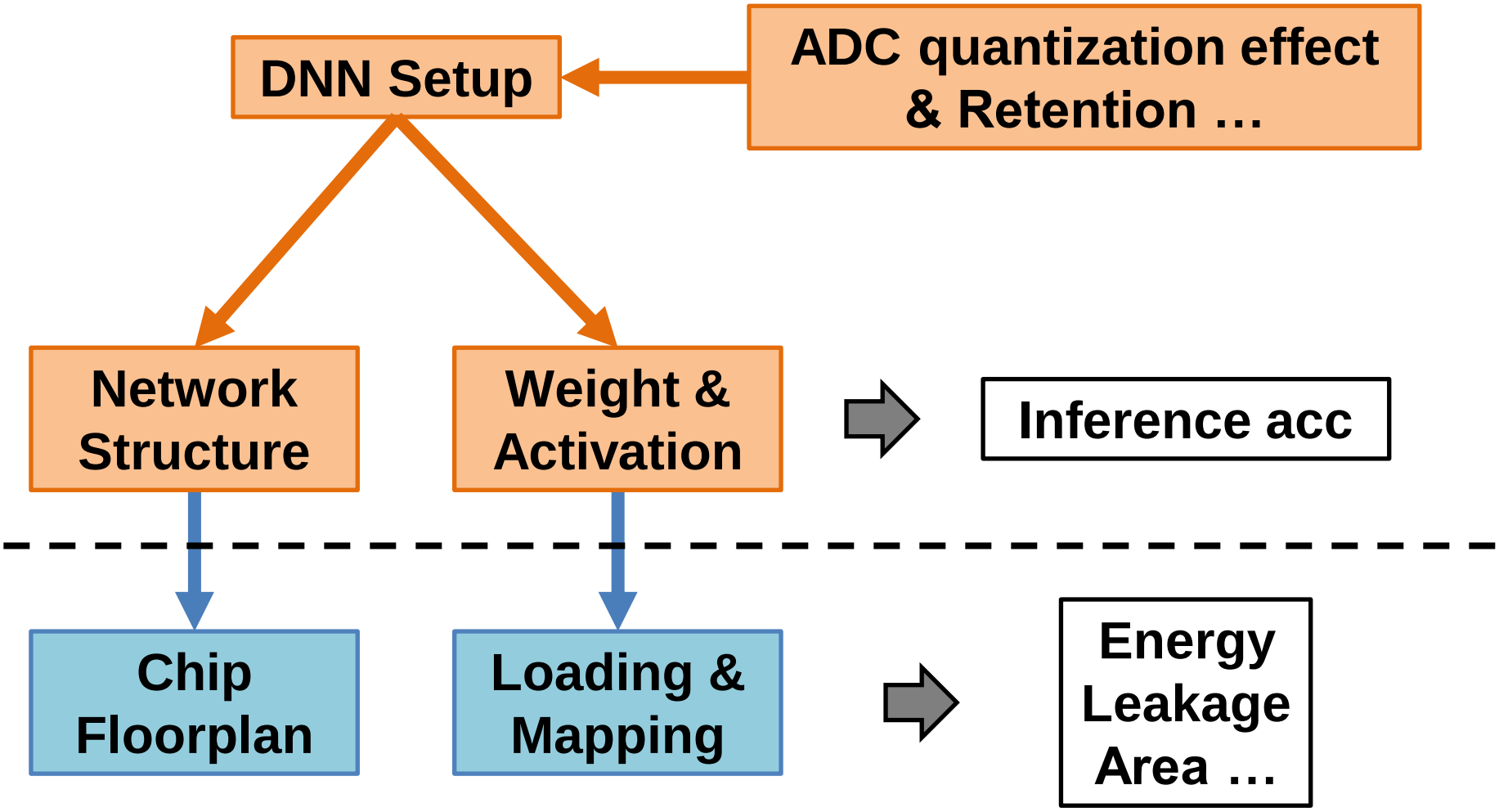
Overall Structure

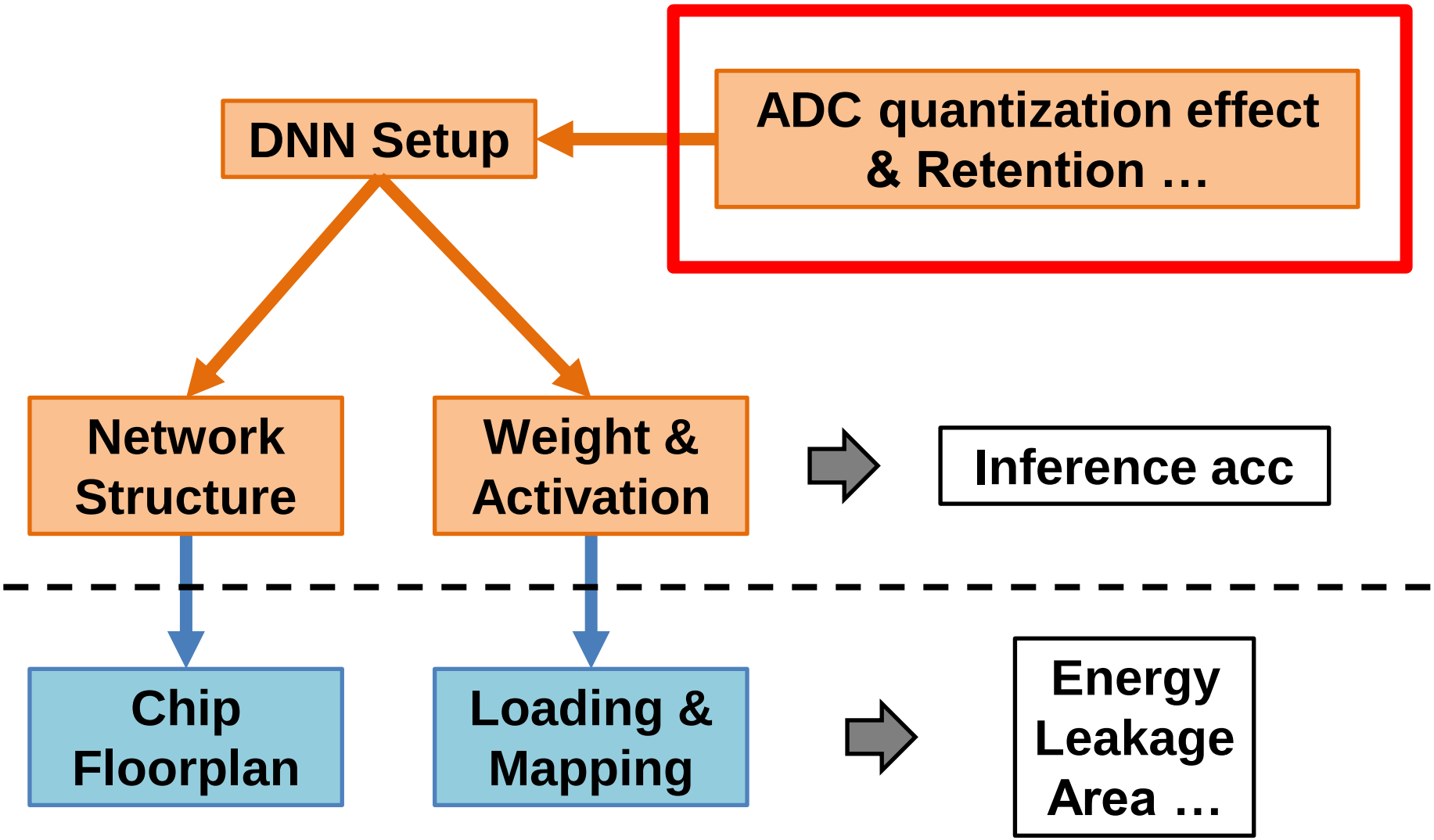
Python Wrapper



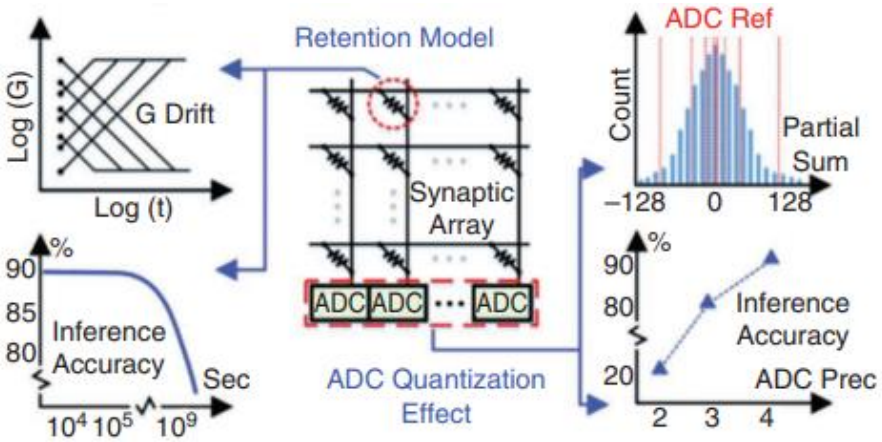
NeuroSim Core

Flow Chart


DNN Setup
ADC quantization effect & Retention …
Network Structure
Weight & Activation
Inference acc
Chip Floorplan
Loading & Mapping
Energy Leakage Area …

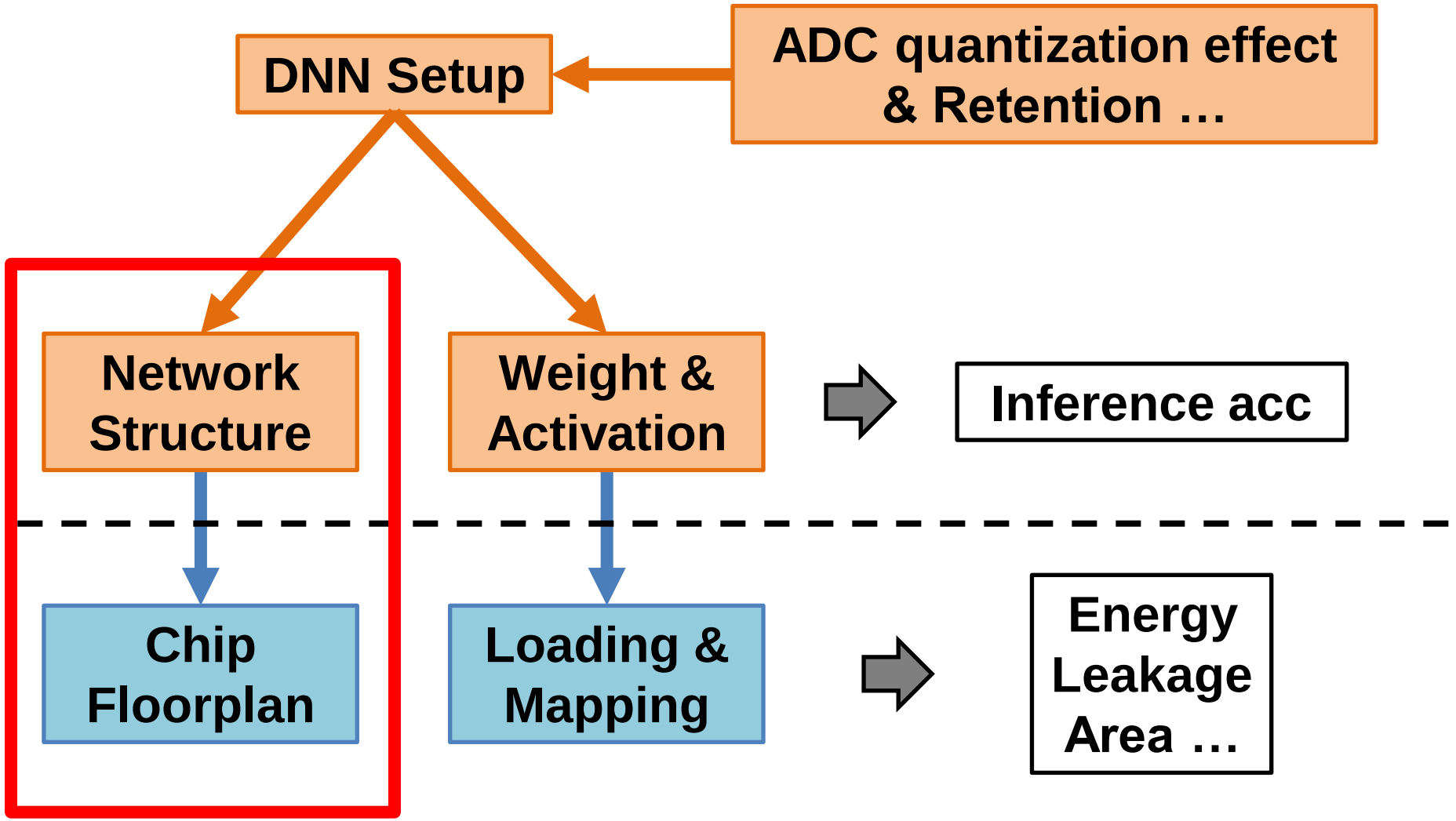

ADC quantization effect
& Retention …
DNN Setup
Network
Structure
Weight &
Activation
Inference acc
Chip
Floorplan
Loading &
Mapping
Energy
Leakage
Area …

Introducing Hardware Effects

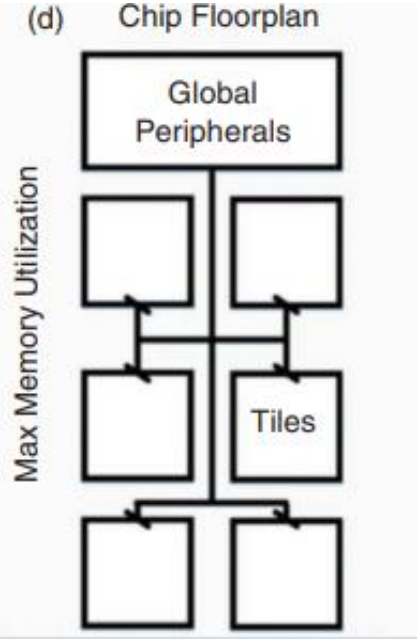


Retention model
ADC quantization effect

are introduced during inference


DNN Setup
ADC quantization effect & Retention …
Network Structure
Weight & Activation
Inference acc
Chip Floorplan
Loading & Mapping
Energy Leakage Area …

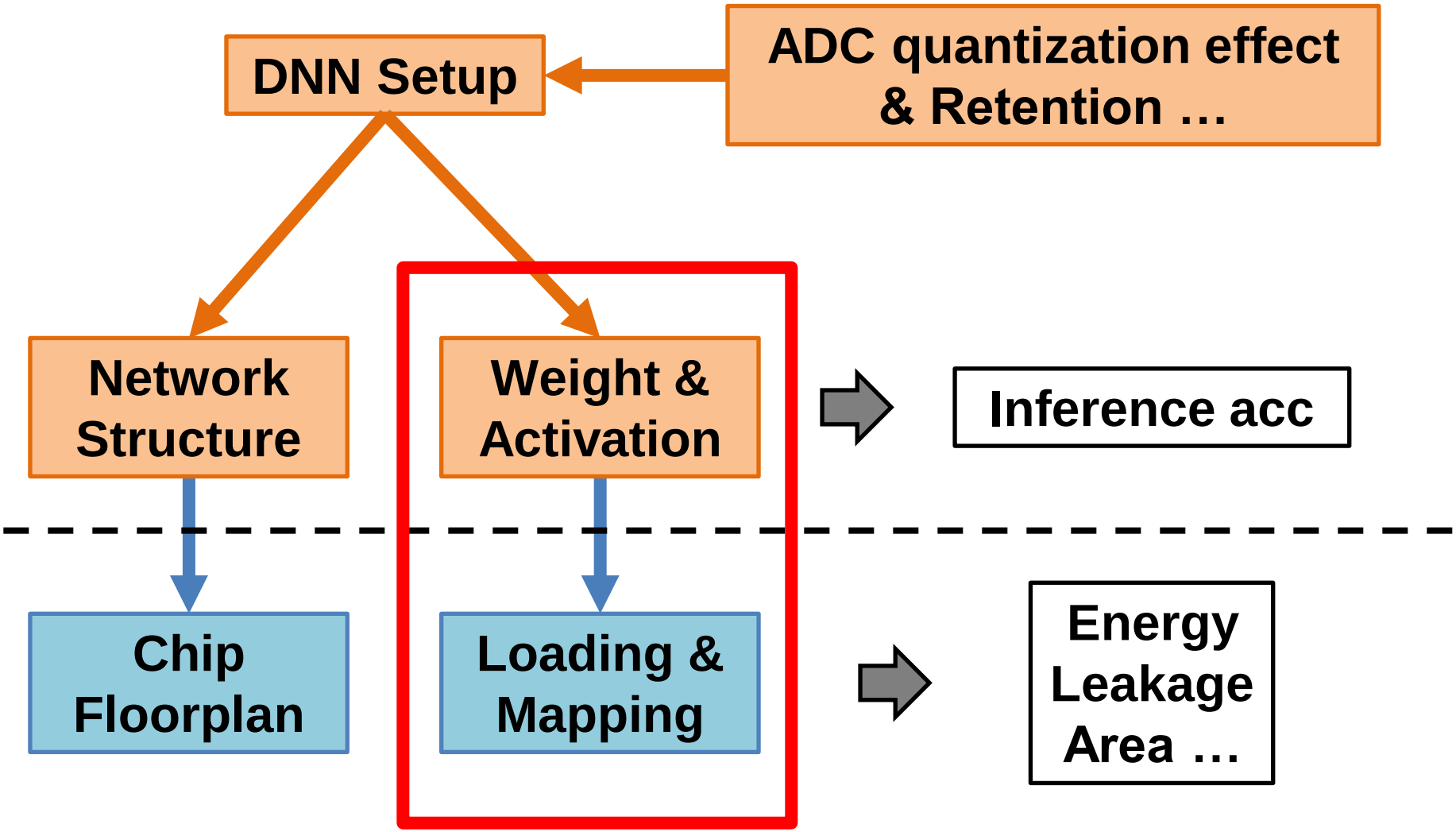
Floorplan



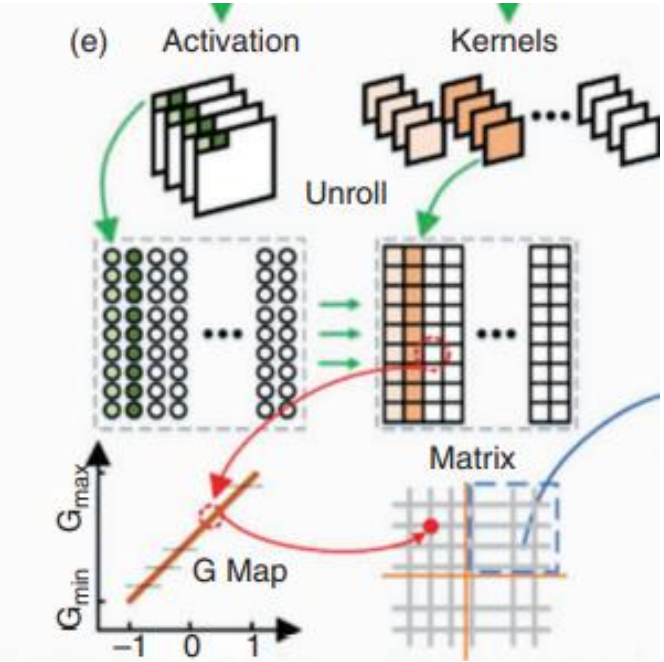
Taking the **predefined network topology** as input

Automatically define the **chip floorplan**

Maximize memory utilization


DNN Setup
ADC quantization effect & Retention …
Network Structure
Weight & Activation
Inference acc
Chip Floorplan
Loading & Mapping
Energy Leakage Area …

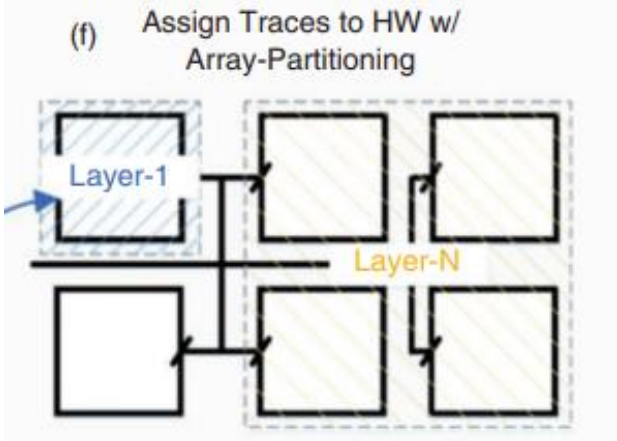
Mapping



Neural activations and updated synaptic weights are stored

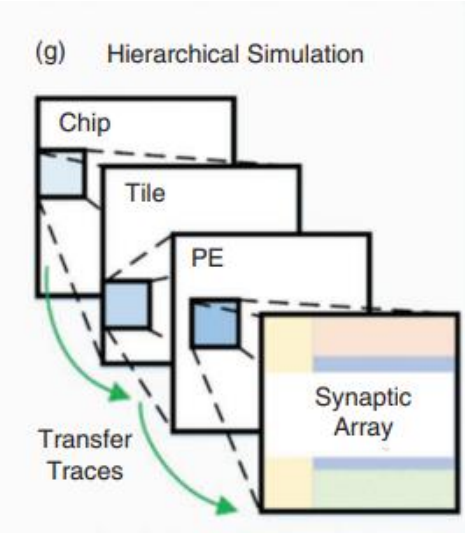
Mapping data to conductance and digital voltage input cycles

Assigning Traces



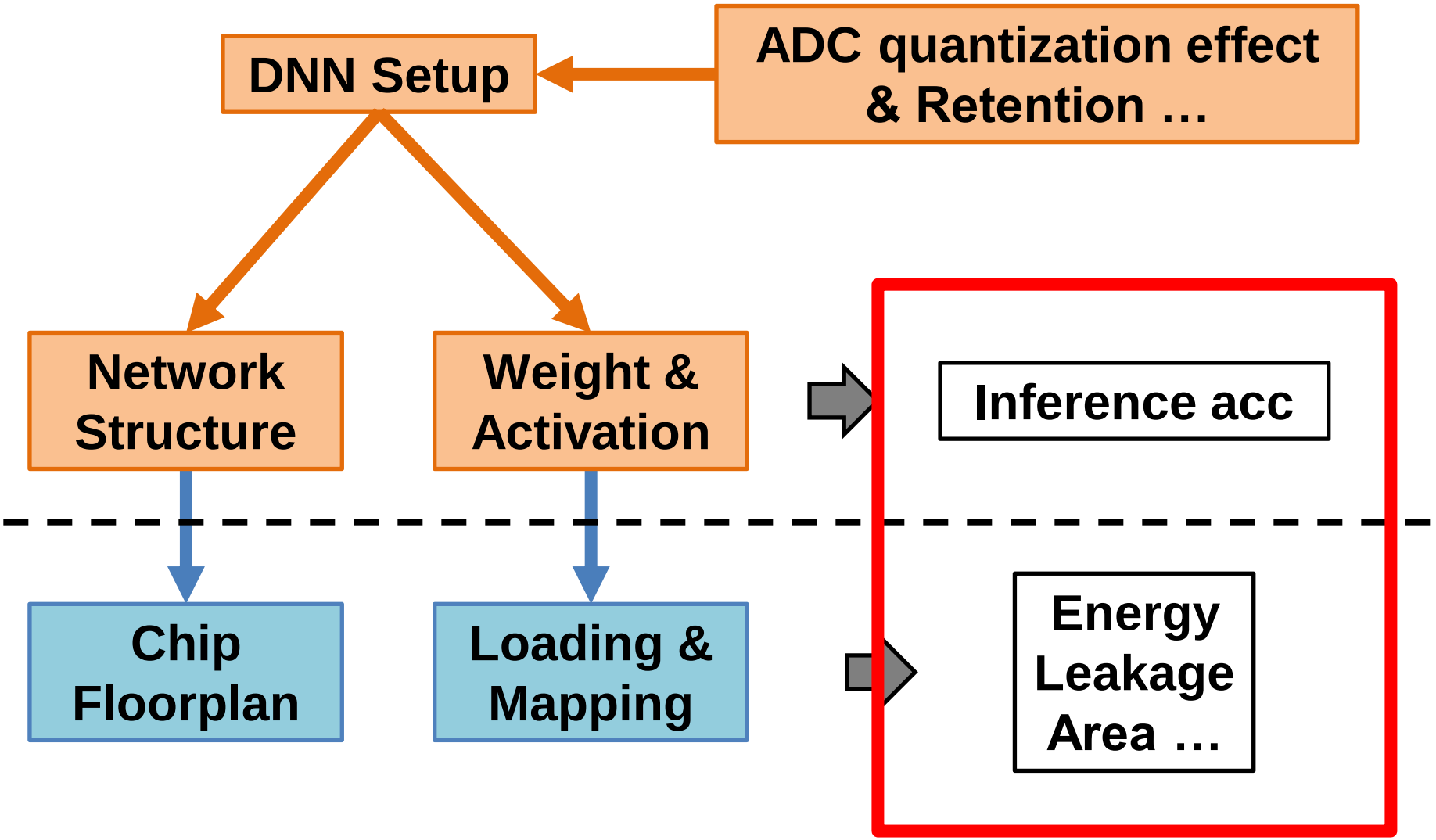
The traces are partitioned and assigned to different locations of the chip

Chip Hierarchy

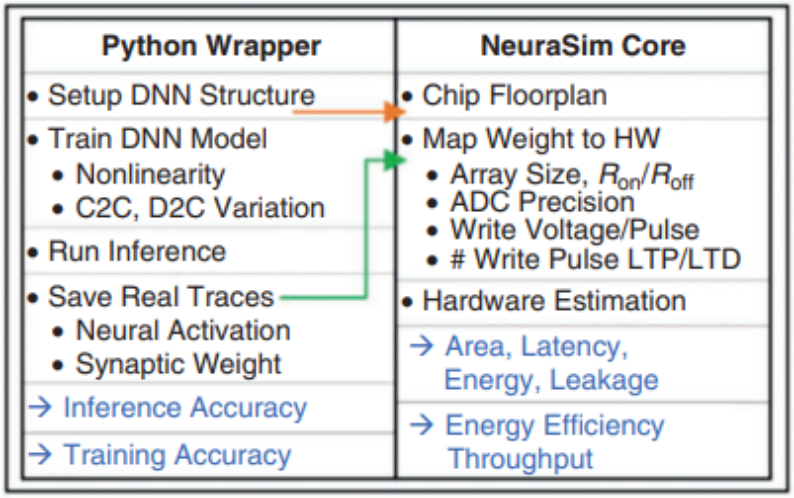


The **top-down hierarchy** of the CIM system is defined as :

Chip
Tile
Processing Element (PE)
Synaptic array


DNN Setup
ADC quantization effect & Retention …
Network Structure
Weight & Activation
Inference acc
Chip Floorplan
Loading & Mapping
Energy Leakage Area …

Simulation Results



Python Wrapper	NeuraSim Core
• Setup DNN Structure	• Chip Floorplan
• Train DNN Model • Nonlinearity • C2C, D2C Variation	• Map Weight to HW • Array Size, R_{on}/R_{off} • ADC Precision • Write Voltage/Pulse • # Write Pulse LTP/LTD
• Run Inference	• Hardware Estimation
• Save Real Traces • Neural Activation • Synaptic Weight	→ Area, Latency, Energy, Leakage
→ Inference Accuracy	→ Energy Efficiency Throughput
→ Training Accuracy	

Hardware-constrained training and inference **accuracy**

Hardware metrics：
Area / Latency / Leakage …

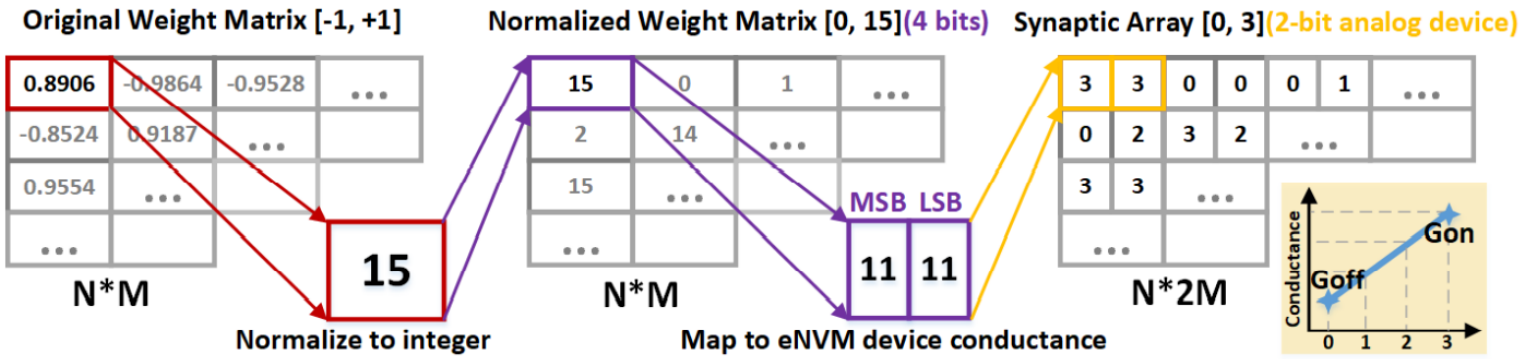
Outline

- CIM
- DNN+NeuroSim
  - Benchmark methodologies
  - Python wrapper
  - C++ simulation
- ResNet-20 Inference Simulation Results

- Weight Mapping
- Retention
- Conductance On / Off Ratio
- ADC Quantization Effects

Weight Mapping

- WAGE → constraint the weights within range [-1, +1]
- Normalized to decimal integer
  → digitalized to conductance level
- Synaptic weight precision / synaptic device bit precision



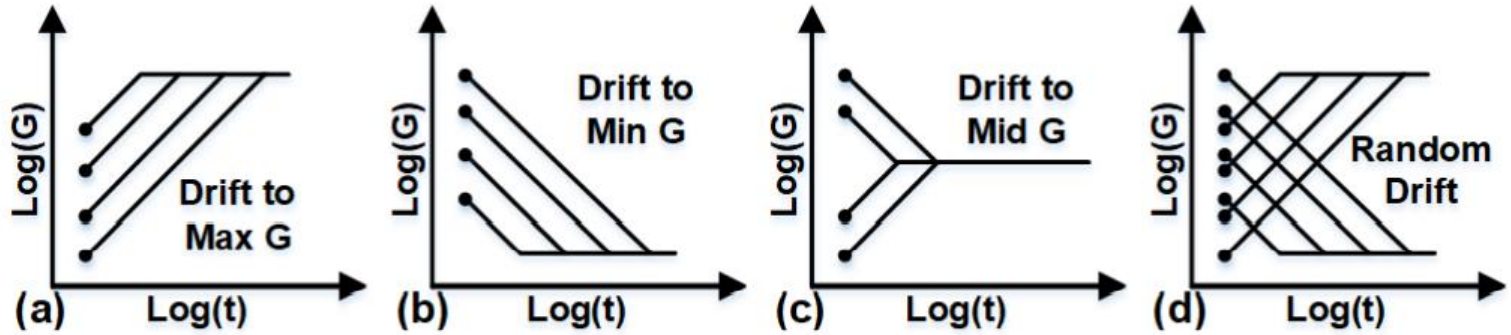
- Weight Mapping
- Retention
- Conductance On / Off Ratio
- ADC Quantization Effects

Retention

- What is retention ?
- Random Drift or not can be chosen

$$G=G_0\left(\frac{t}{t_0}\right)^v$$

G_0 : initial conductance
v : drift coefficient
t : retention time
t_0 : time constant = 1 (default)



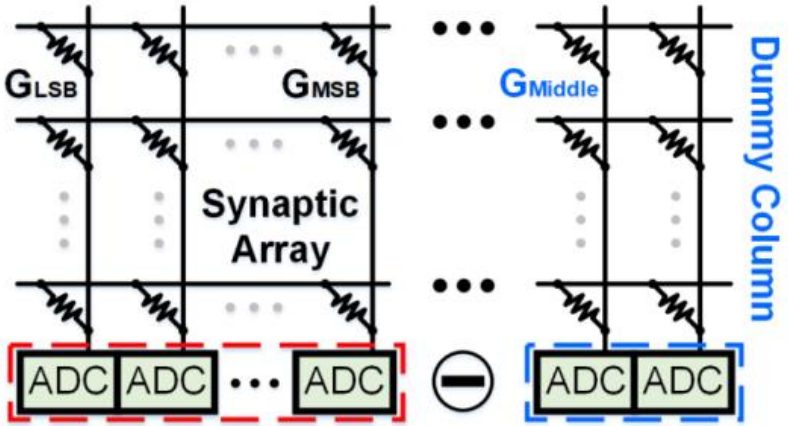
- Weight Mapping
- Retention
- Conductance On / Off Ratio
- ADC Quantization Effects

Conductance On / Off Ratio

- About on / off ratio
  - G stands for conductance
  - G can't be 0 since R can't be infinity → define on / off ratio
  - Gmiddle = (Gmin + Gmax) / 2

[Gmin, Gmax] – [Gmiddle, Gmiddle]

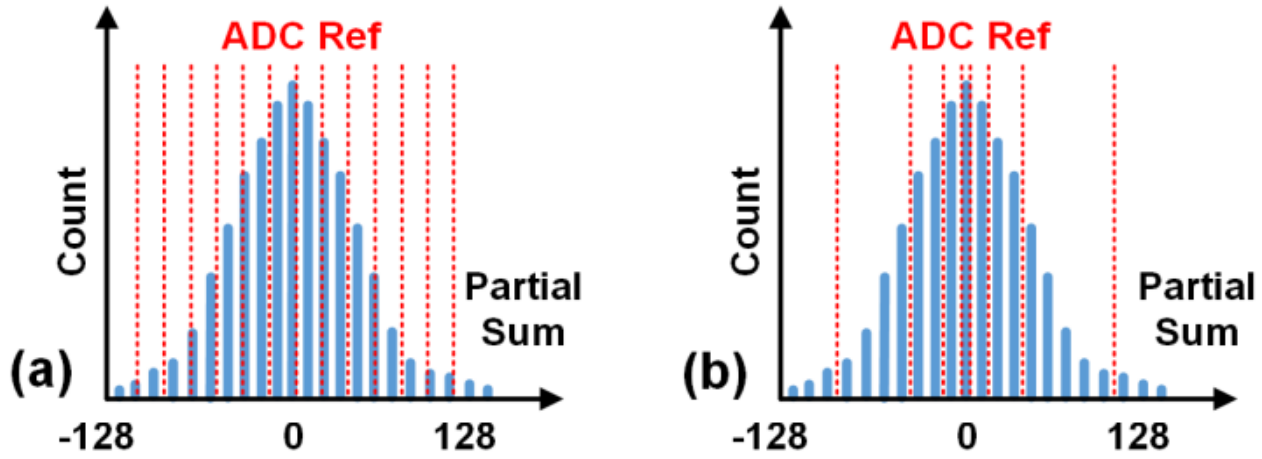
Let D = (Gmax – Gmin) / 2
→ [-D, +D]



- Weight Mapping
- Retention
- Conductance On / Off Ratio
- ADC Quantization Effects

ADC Quantization Effects

- High ADC precision cost lots of area and energy
- Partial sum distribution is centered in the middle
- Use non-linear quantization may save ~1 bit (can be truncated with less loss)
- May be more sensitive to partial sum variation

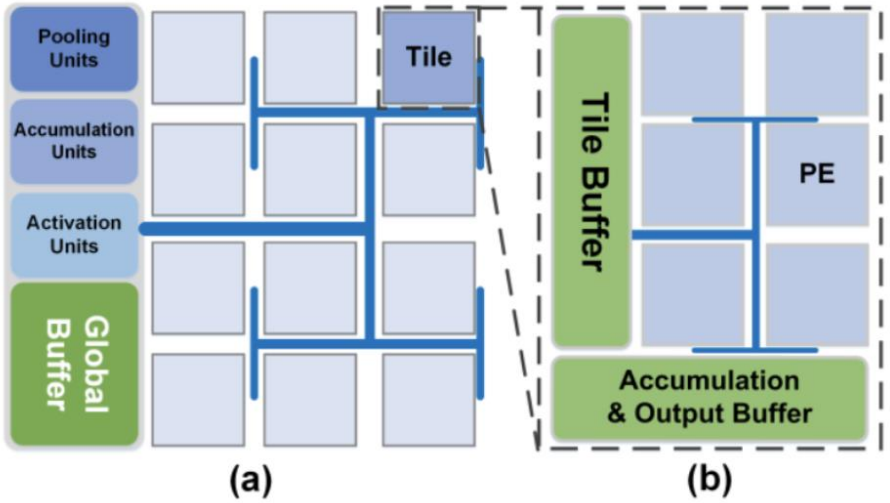


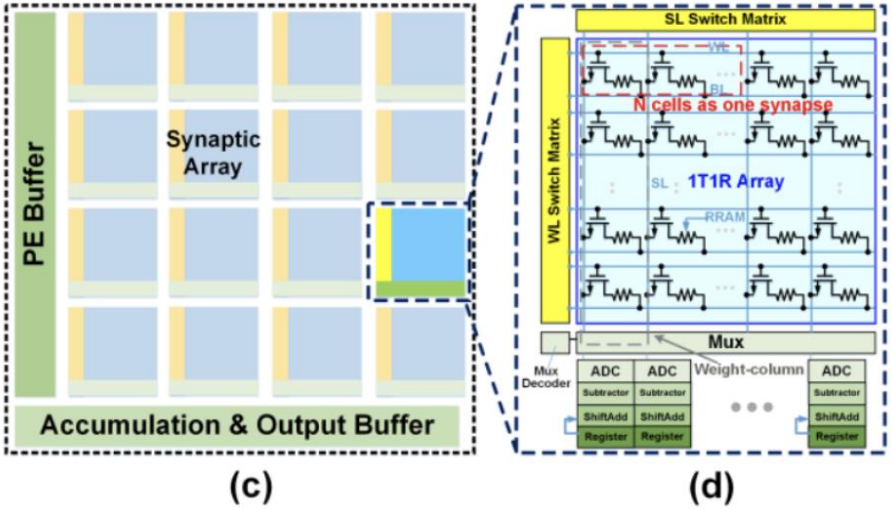
Outline

- CIM
- DNN+NeuroSim
  - Benchmark methodologies
  - Python wrapper
  - C++ simulation
- ResNet-20 Inference Simulation Results

Chip Hierarchy

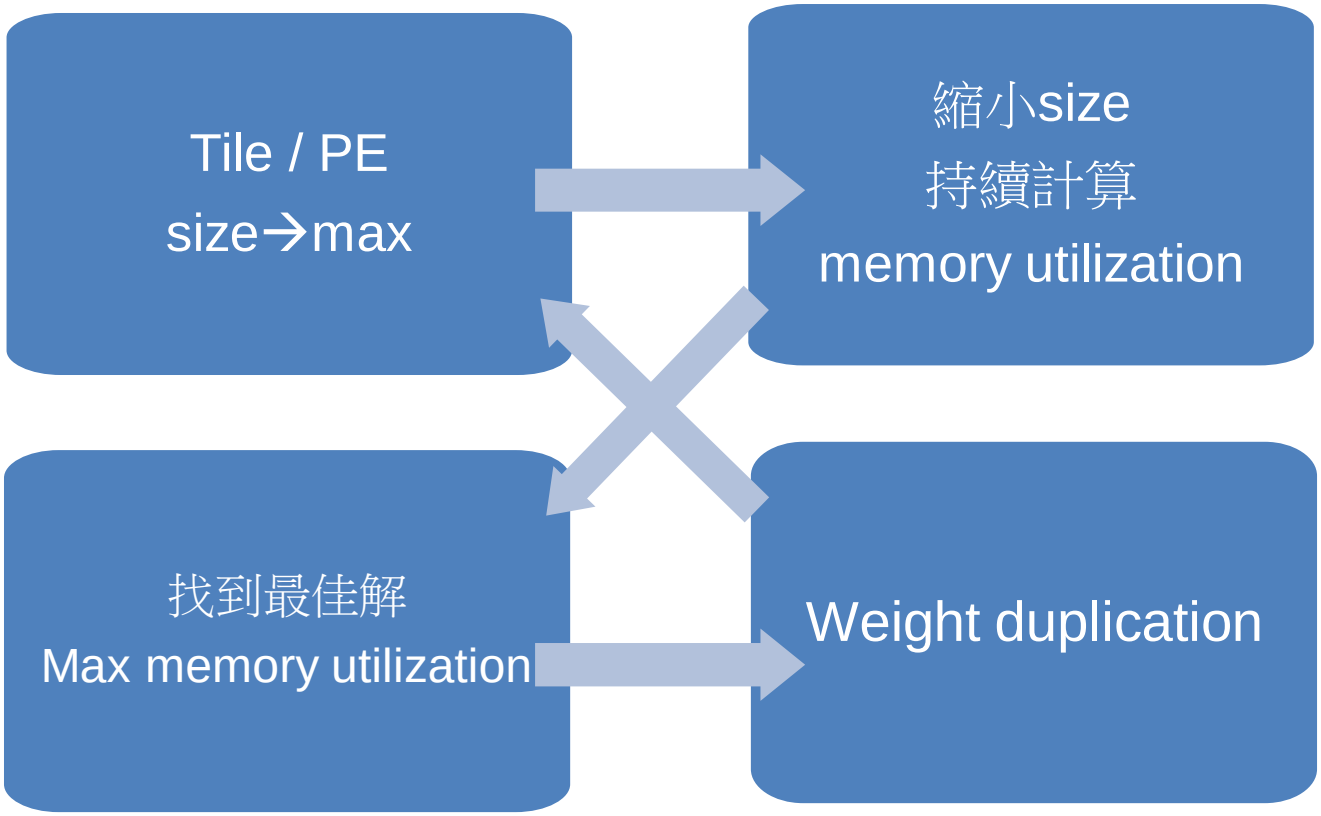
- **Chip**
  - **Tile**
  - Global buffer (input of each layer)
  - Accumulation units
  - **Activation/Pooling units**
  - **H tree**
    - **PE**
    - Tile buffer
    - Accumulation module
    - Output buffer
    - H tree
      - **Synaptic array**
      - PE buffer
      - Accumulation module
      - Output buffer





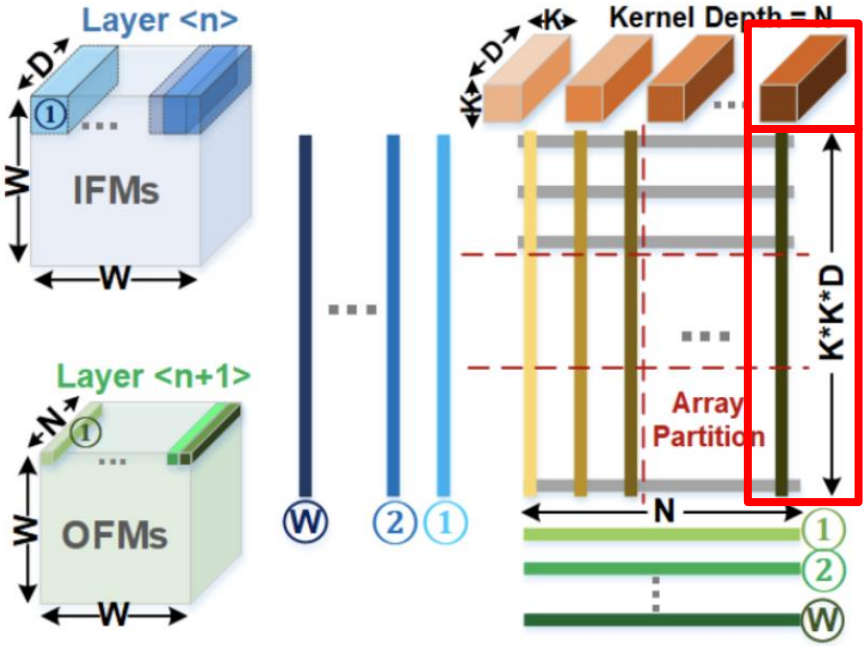
Floorplan flow chart

- **Floorplan**: Optimize the **memory utilization** based on fixed synaptic array size.
- **PE size** is at most **half of the tile size** to keep the exist of chip hierarchy.



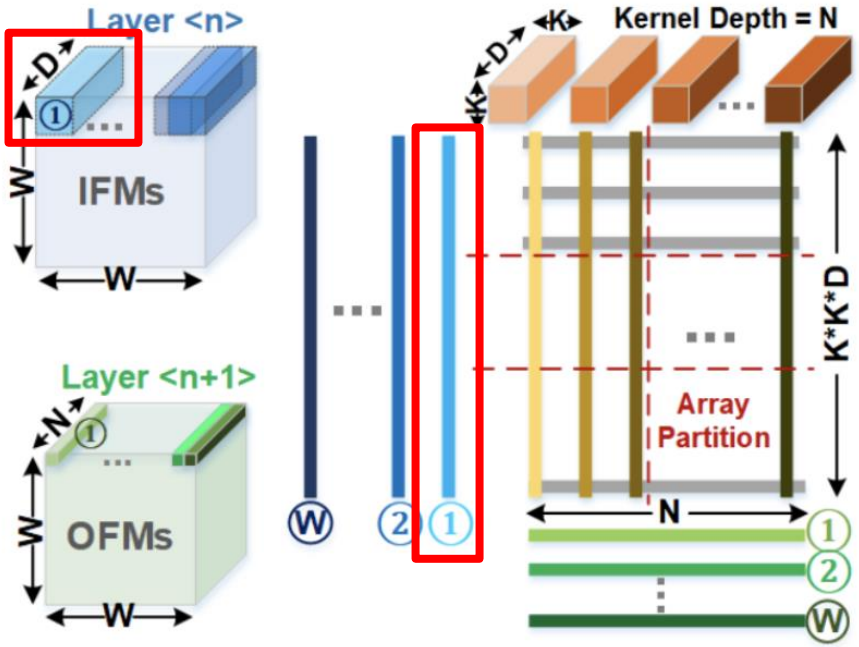
CNN Mapping Method

- Conventional mapping:
  1. Unroll N kernels(3D matrix) into a 2D matrix(N 1D vectors).
  2. Unroll each part of input(3D matrix) (e.g. ①) into 1D vector.
  3. Load each part of input sequentially to obtain the OFMs.



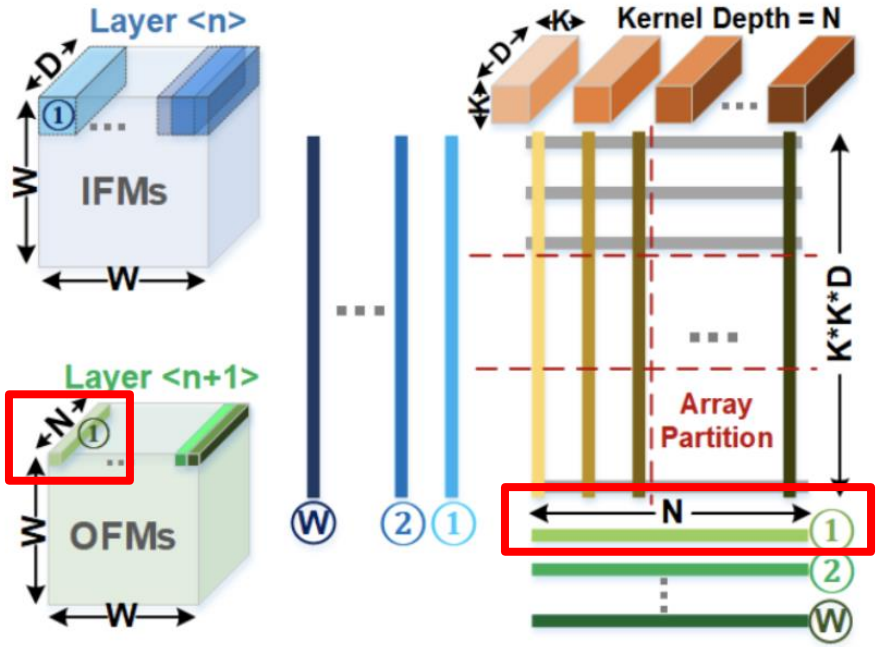
CNN Mapping Method

- Conventional mapping:
  1. Unroll N kernels(3D matrix) into a 2D matrix(N 1D vectors).
  2. Unroll each part of input(3D matrix) (e.g. ①) into 1D vector.
  3. Load each part of input sequentially to obtain the OFMs.



CNN Mapping Method

- Conventional mapping:
  1. Unroll N kernels(3D matrix) into a 2D matrix(N 1D vectors).
  2. Unroll each part of input(3D matrix) (e.g. ①) into 1D vector.
  3. Load each part of input sequentially to obtain the OFMs.



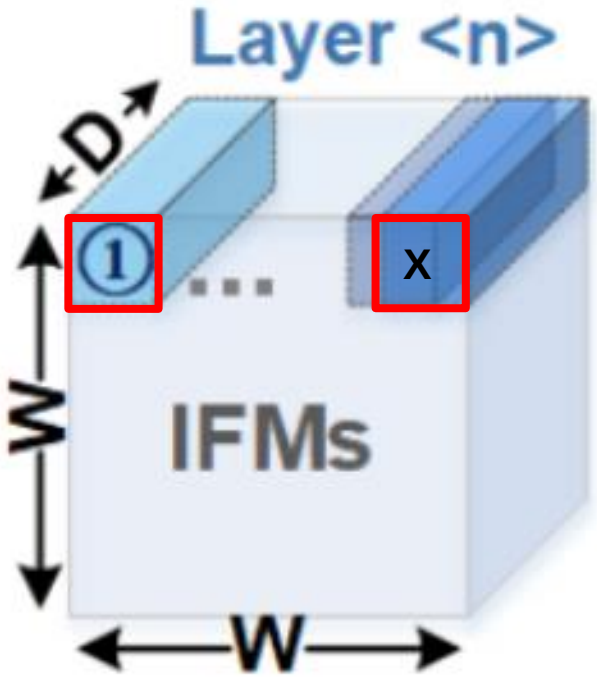
CNN Mapping Method

- Conventional mapping:

  Inputs will be reused after several cycles if stride size<kernel size.

  →revisit input data from upper level buffers

  →more latency and energy



CNN Mapping Method

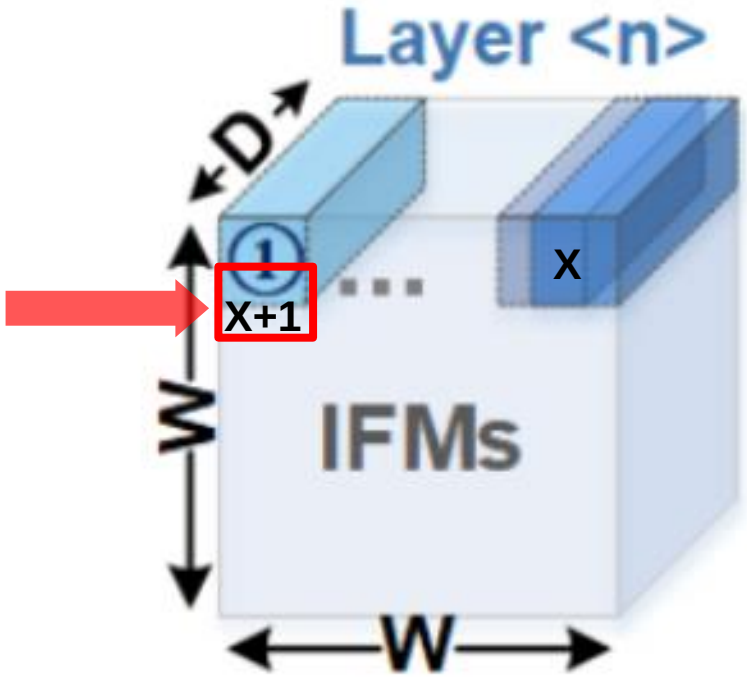
- Conventional mapping:

  Inputs will be reused after several cycles if stride size<kernel size.

  →revisit input data from upper level buffers

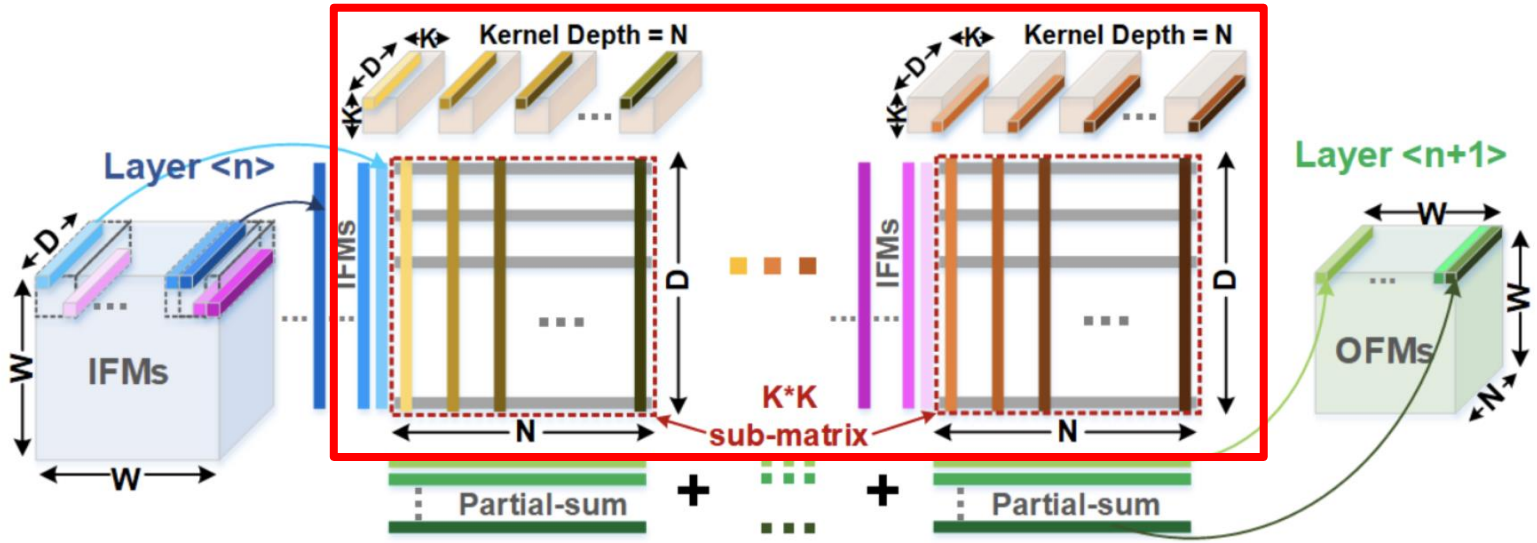
  →more latency and energy

(X+1)th input matrix reused
input data in 1st input matrix



CNN Mapping Method

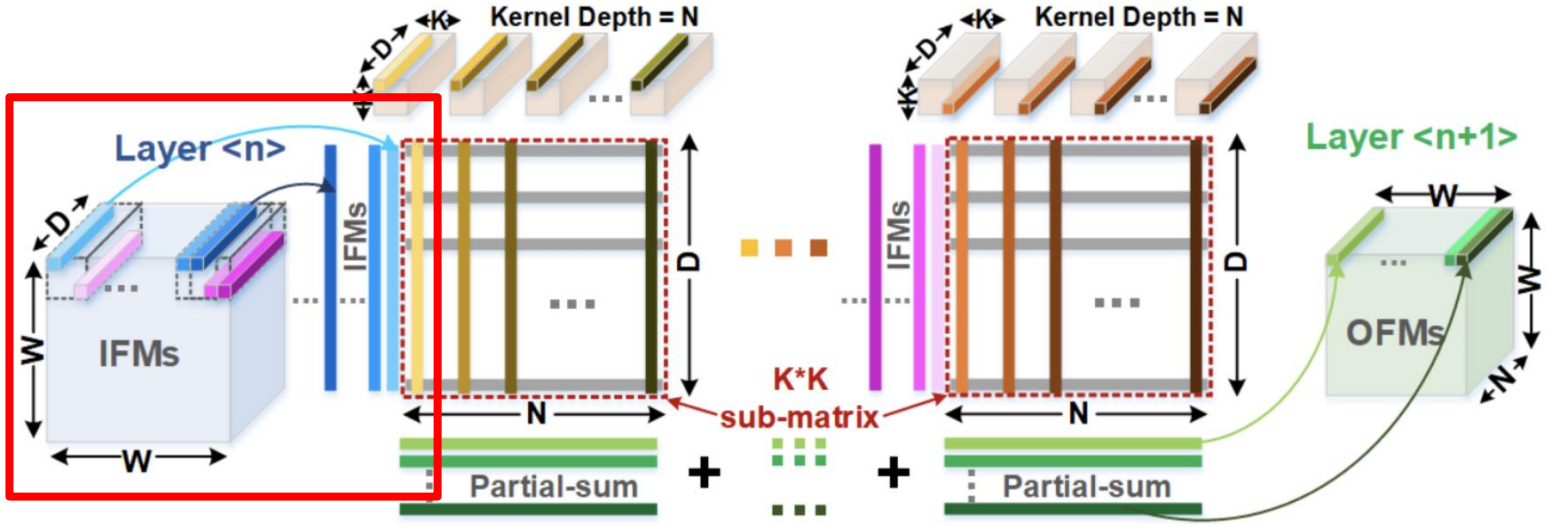
- Novel mapping:
  1. Partition each kernel into K*K 2D matrices(stored in K*K PEs).
  2. Load input of size 1*1*D sequentially into each PE for partial sums.
  3. Accumulate all partial sums to obtain the OFMs.



- IFM size = W*W*D, Kernel size = K*K*D*N, OFM size = W*W*N
- Number of sub-matrix = K*K

CNN Mapping Method

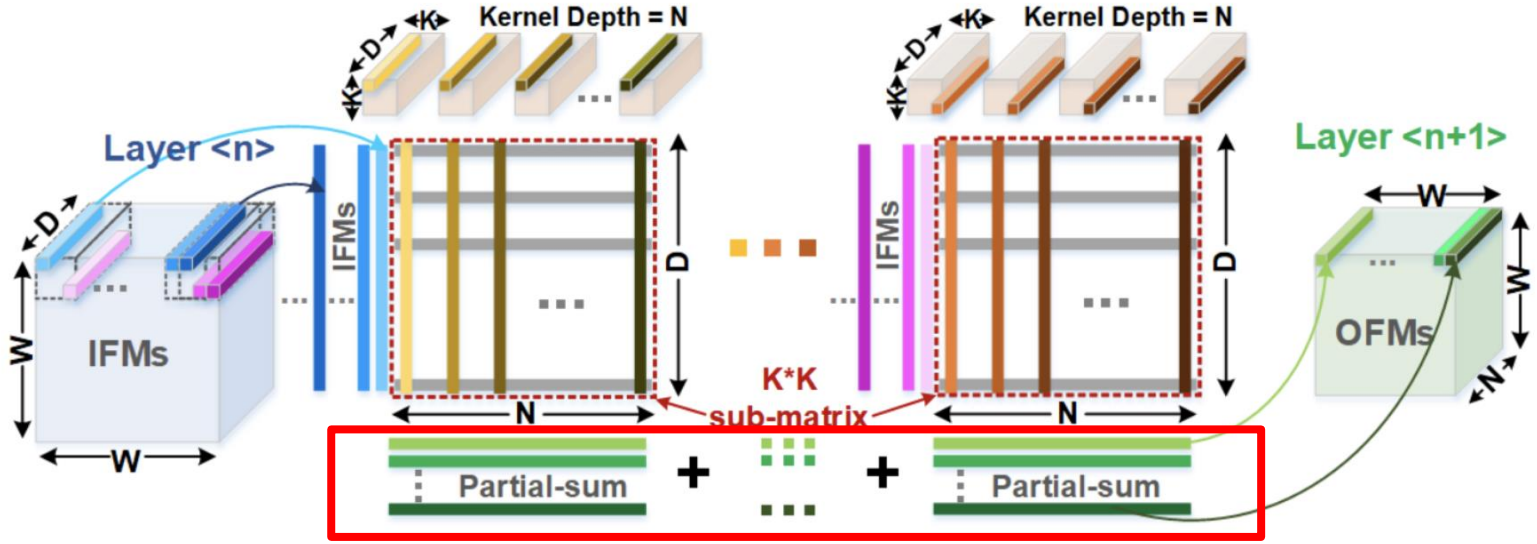
- Novel mapping:
  1. Partition each kernel into K*K 2D matrices(stored in K*K PEs).
  2. Load input of size 1*1*D sequentially into each PE for partial sums.
  3. Accumulate all partial sums to obtain the OFMs.



- IFM size = W*W*D, Kernel size = K*K*D*N, OFM size = W*W*N
- Number of sub-matrix = K*K

CNN Mapping Method

- Novel mapping:
  1. Partition each kernel into K*K 2D matrices(stored in K*K PEs).
  2. Load input of size 1*1*D sequentially into each PE for partial sums.
  3. Accumulate all partial sums to obtain the OFMs.

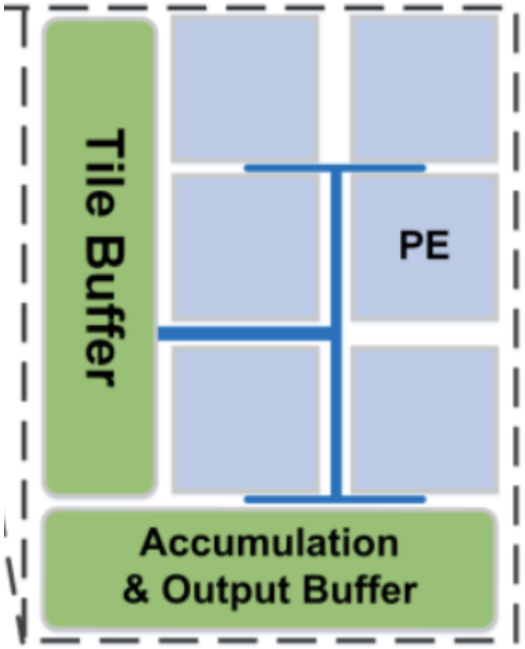


- IFM size = W*W*D, Kernel size = K*K*D*N, OFM size = W*W*N
- Number of sub-matrix = K*K

CNN Mapping Method

- Novel mapping:

  Partial sums could be calculate in parallel by transferring input data among different PEs. → reducing latency and energy



Pipeline System

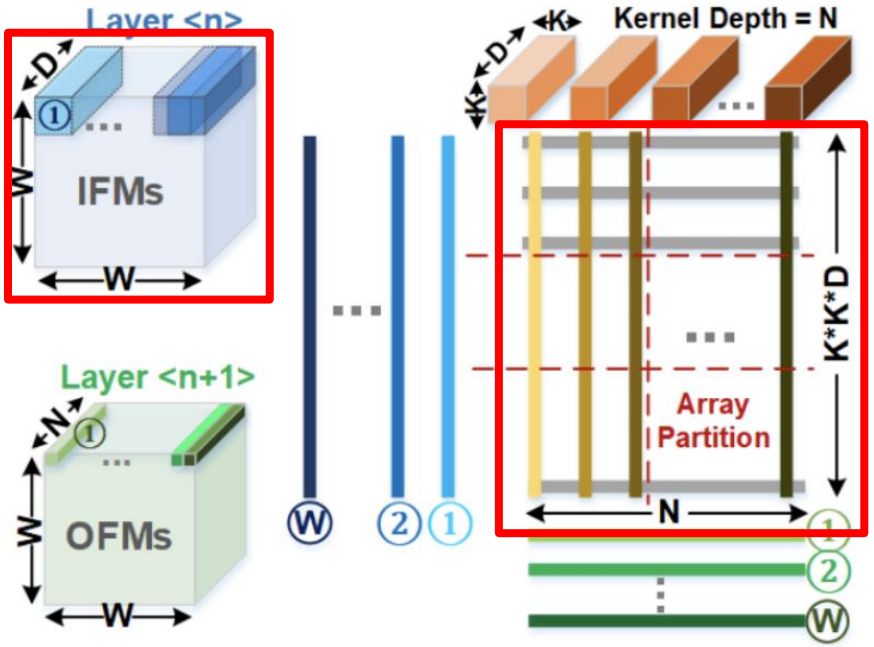
- Each layer is a stage in pipeline system.
- Calculate the IFMs of different layers simultaneously.
- IFMs tend to be deeper but smaller from shallower layers to deeper layers.

  → obtain the OFM faster and the synaptic matrix becomes bigger.

  → operate weight-duplication to the weight matrices of shallower layers.

  → reduce latency.

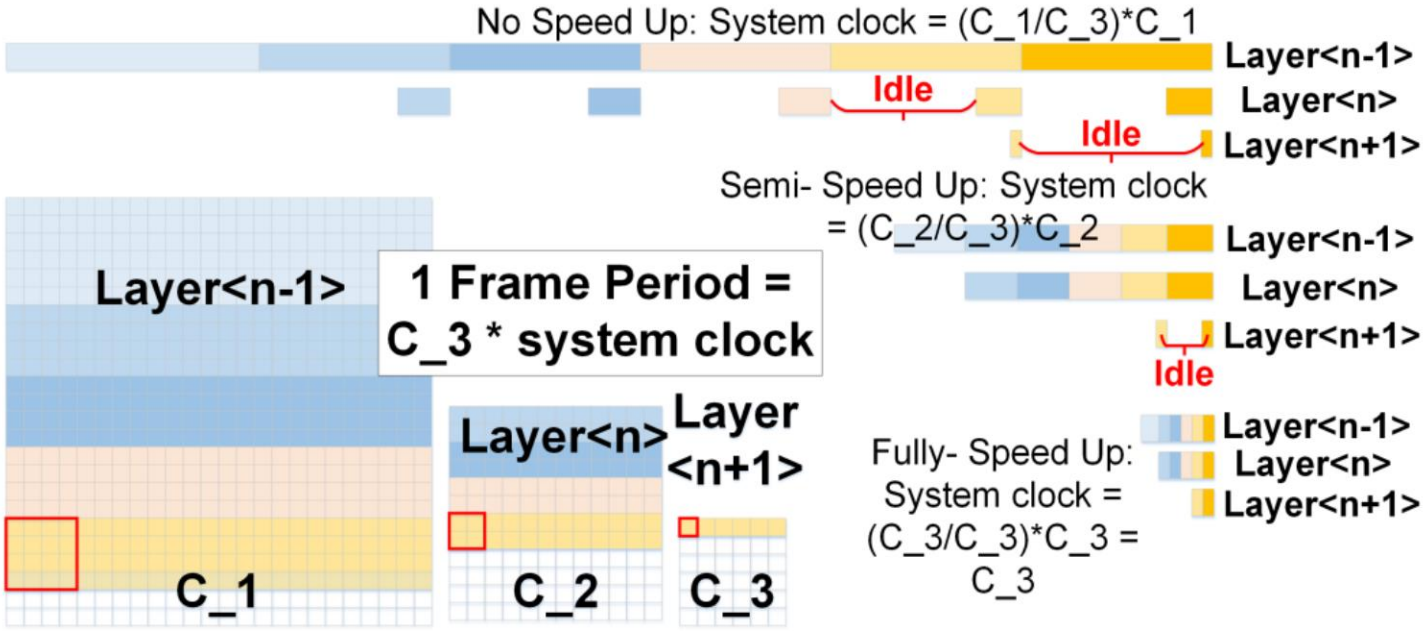
W becomes smaller
→Obtain OFM faster



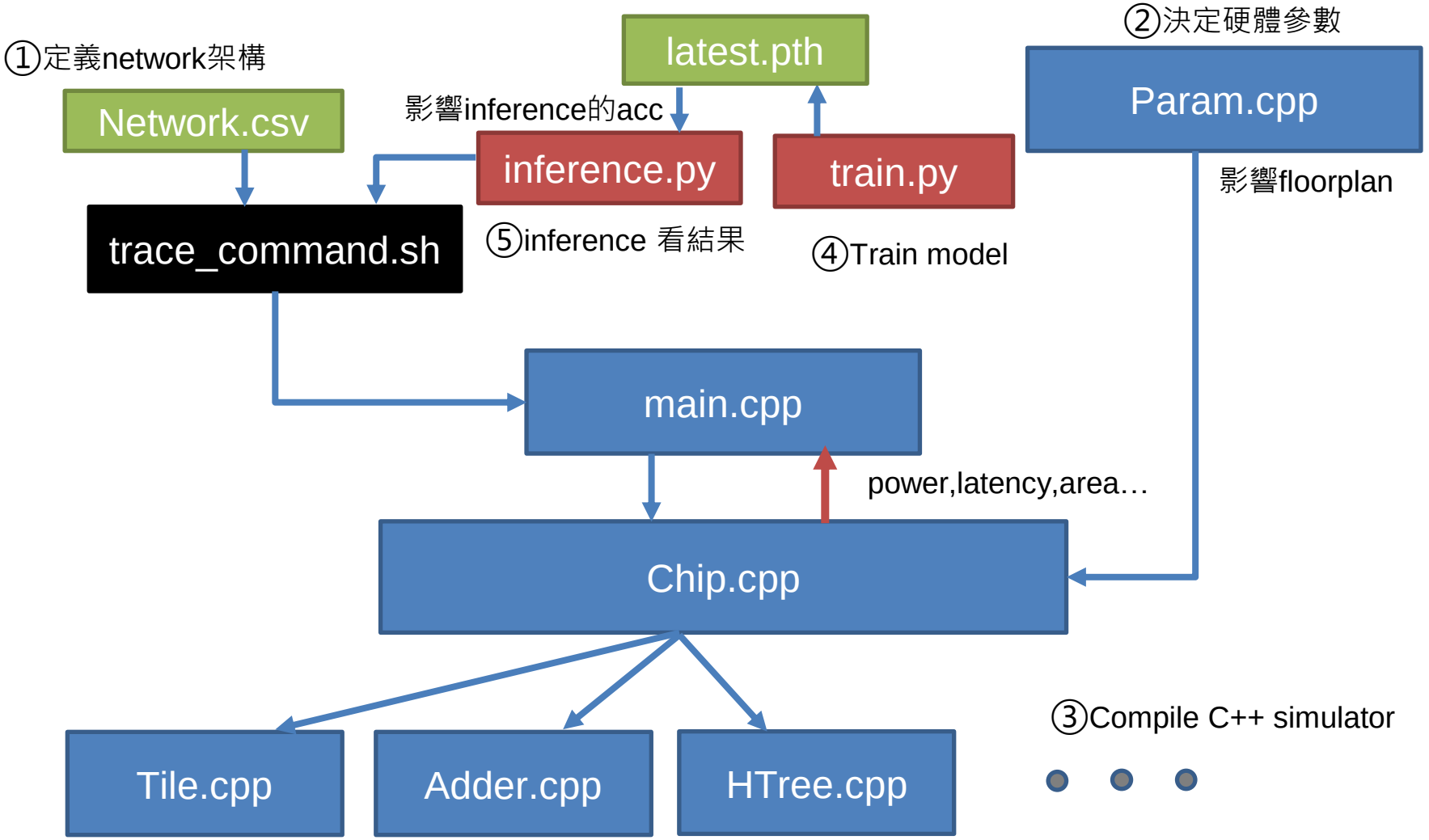
N becomes bigger

Pipeline System

- Each layer is a stage in pipeline system.
- Calculate the IFMs of different layers simultaneously.
- IFMs tend to be deeper but smaller from shallower layers to deeper layers.
  → operate weight-duplication to the weight matrices of shallower layers.
  → reduce latency



DNN+NeuroSim Structure



DNN+NeuroSim Training with ResNet20

- Details:
  - Dataset: cifar10
  - Model: ResNet-20
  - Mode: WAGE
  - Without hardware effect (due to GPU memory limit)

Outline

- CIM
- DNN+NeuroSim
  - Benchmark methodologies
  - Python wrapper
  - C++ simulation
- ResNet-20 Inference Simulation Results

ResNet20 Inference Simulation With Hardware Effects

- Inference of ResNet20 model without hardware effects: acc:87.15%

- Simulation:

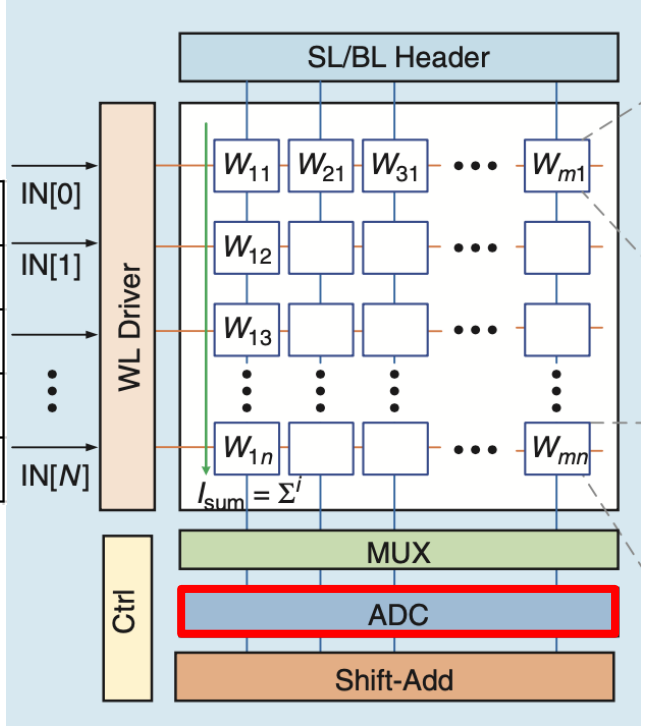
Hardware Parameters	default	simulation
ADC precision	5 bits	4/ 5/ 6/ 7
Subarray size	128*128	16/ 32/ 64/ 128
Weight bit-width	8 bits	6/ 8/ 10/ 12
Bits per cell	4 bits	

- Results
  - Accuracy
  - Hardware performance: area, latency, leakage energy, total energy

1. ADC precision

ADC precision determines precision of each synaptic array's partial sum.

Hardware Parameters	default	simulation
ADC precision	5 bits	4/ 5/ 6/ 7
Subarray size	128*128	16/ 32/ 64/ 128
Weight bit-width	8 bits	6/ 8/ 10/ 12
Bits per cell	4 bits	

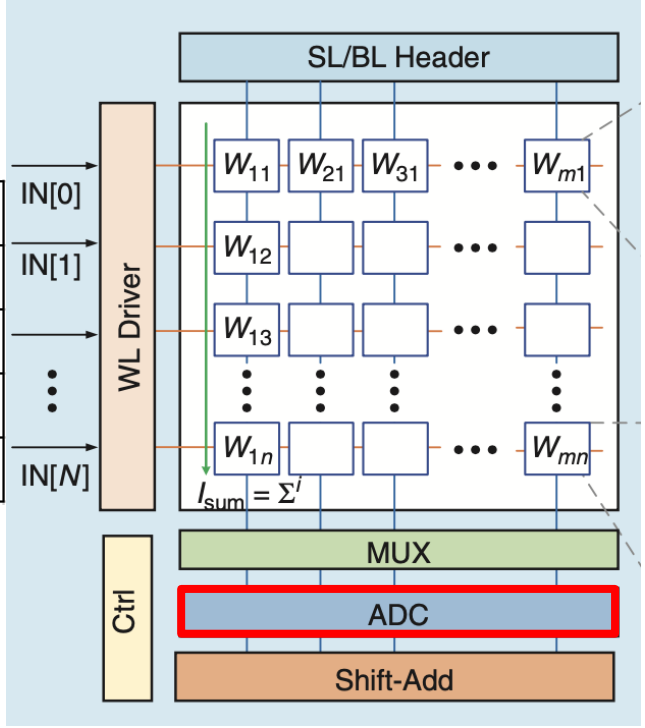


1. ADC precision

ADC precision determines precision of each synaptic array's partial sum.

Hardware Parameters	default	simulation
ADC precision	5 bits	4/ 5/ 6/ 7
Subarray size	128*128	16/ 32/ 64/ 128
Weight bit-width	8 bits	6/ 8/ 10/ 12
Bits per cell	4 bits	

9 bits



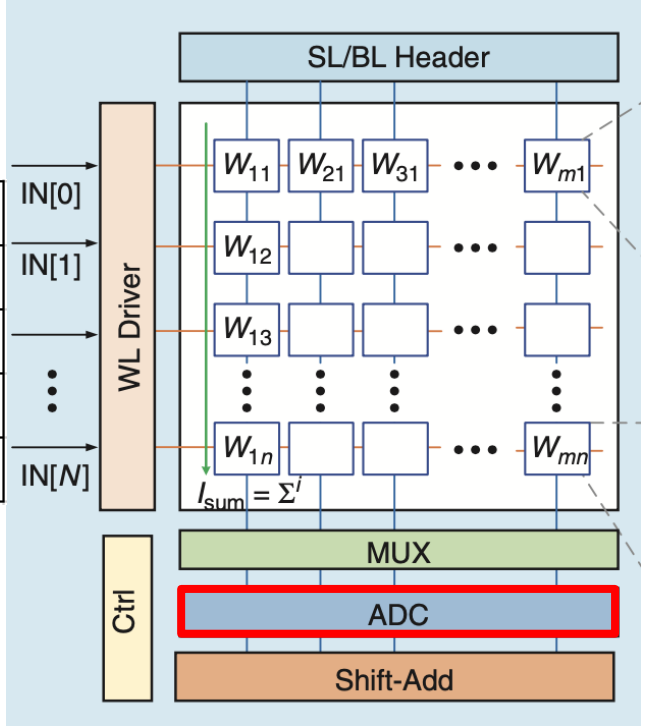
1. ADC precision

ADC precision determines precision of each synaptic array's partial sum.

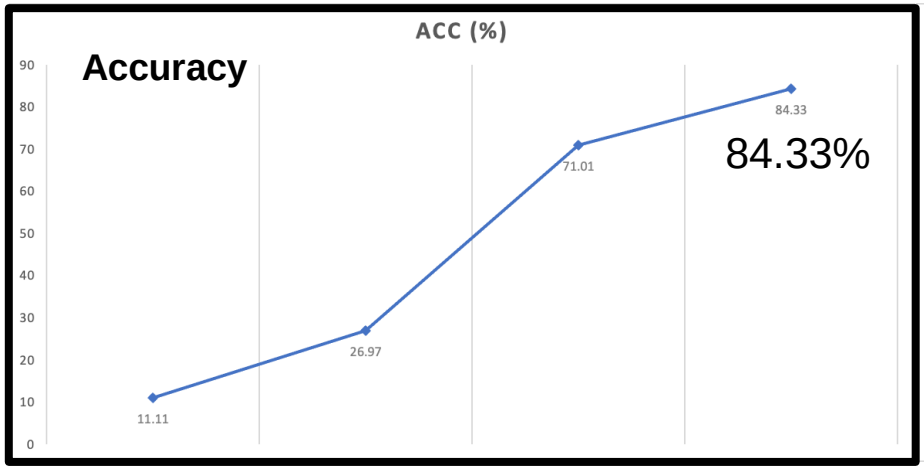
Hardware Parameters	default	simulation
ADC precision	5 bits	4/ 5/ 6/ 7
Subarray size	128*128	16/ 32/ 64/ 128
Weight bit-width	8 bits	6/ 8/ 10/ 12
Bits per cell	4 bits	

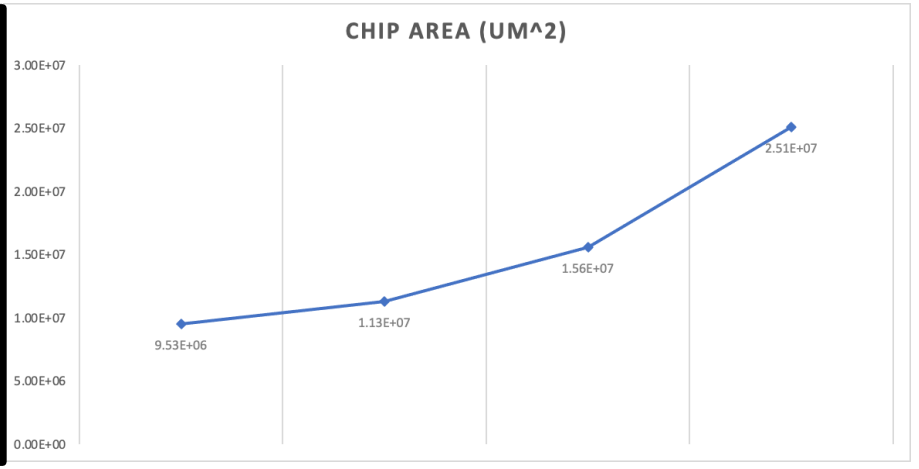
9 bits

All cases of different ADC precision cause **precision loss for partial sum**.

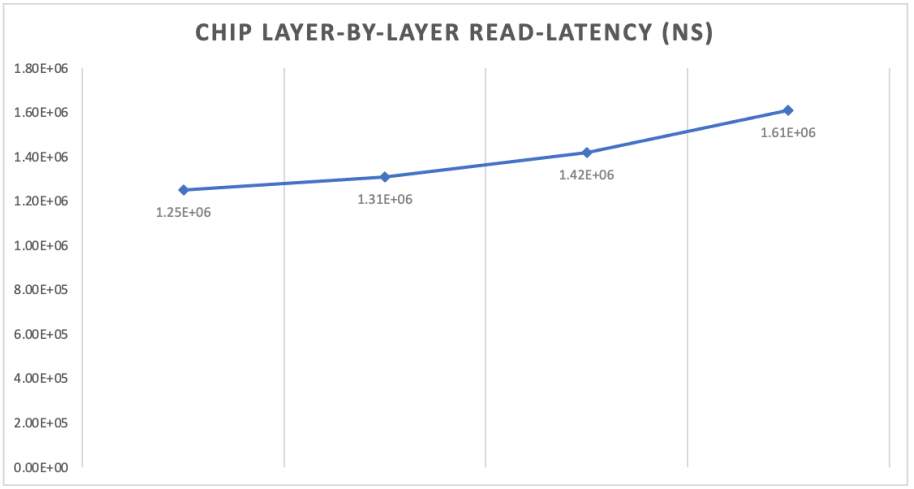


Inference Simulation Results(ADC precision=4/5/6/7)

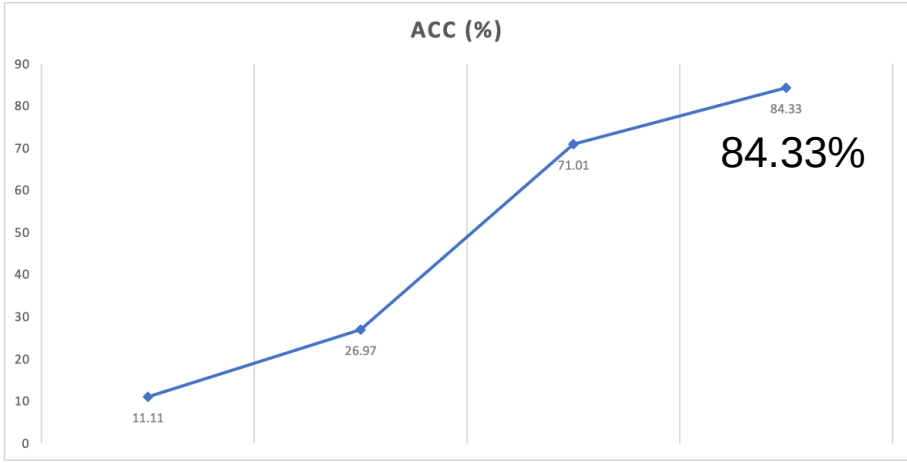


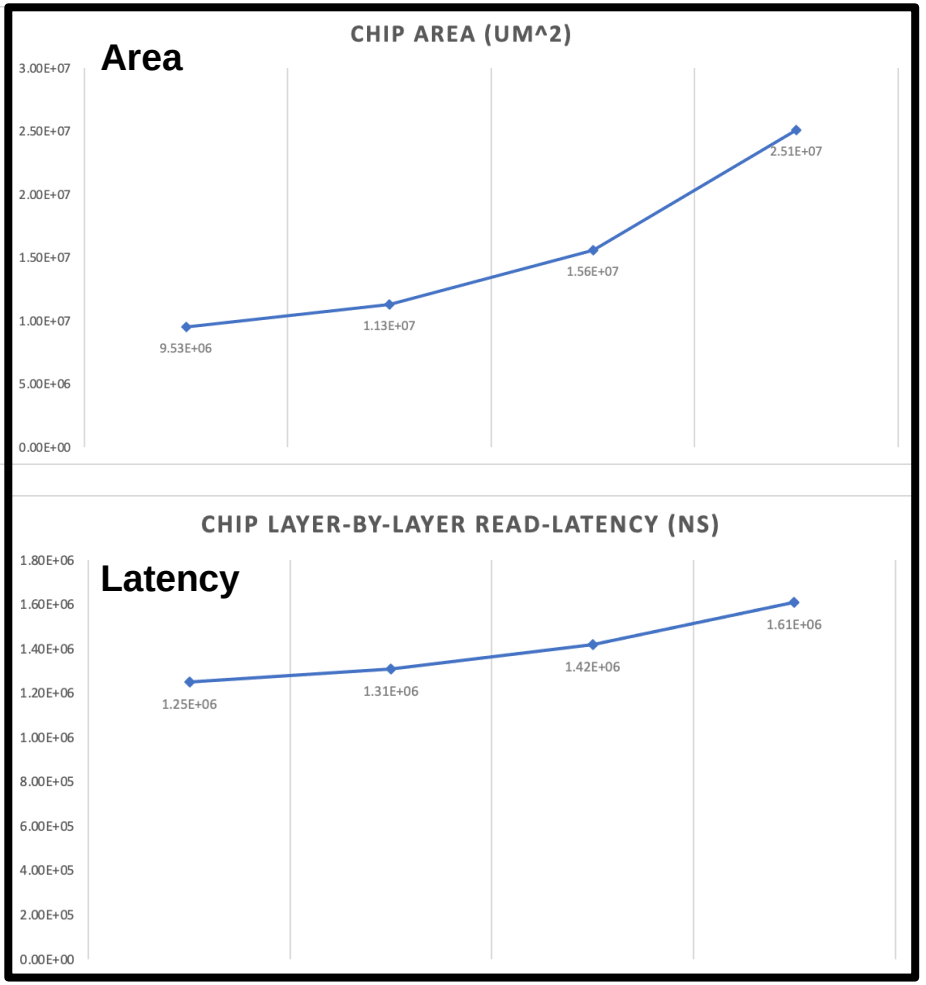


With higher ADC precision,
accuracy, **area** and **latency** increase.



Inference Simulation Results(ADC precision=4/5/6/7)



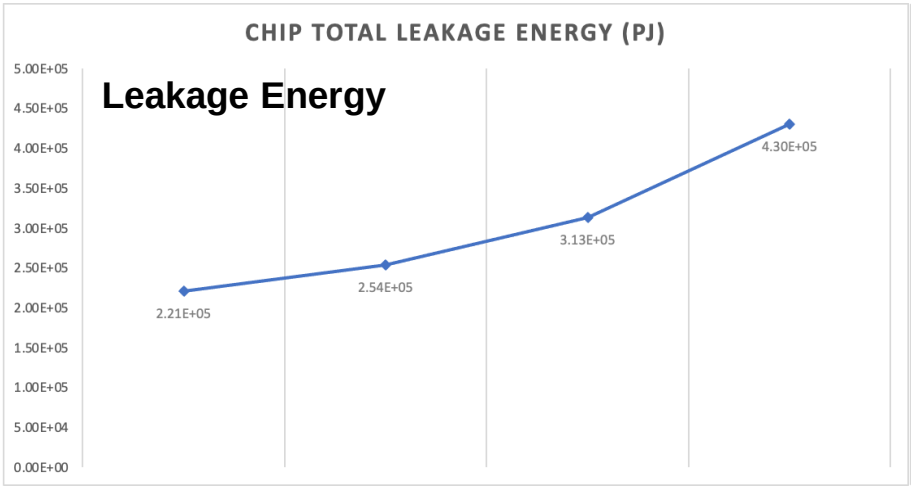


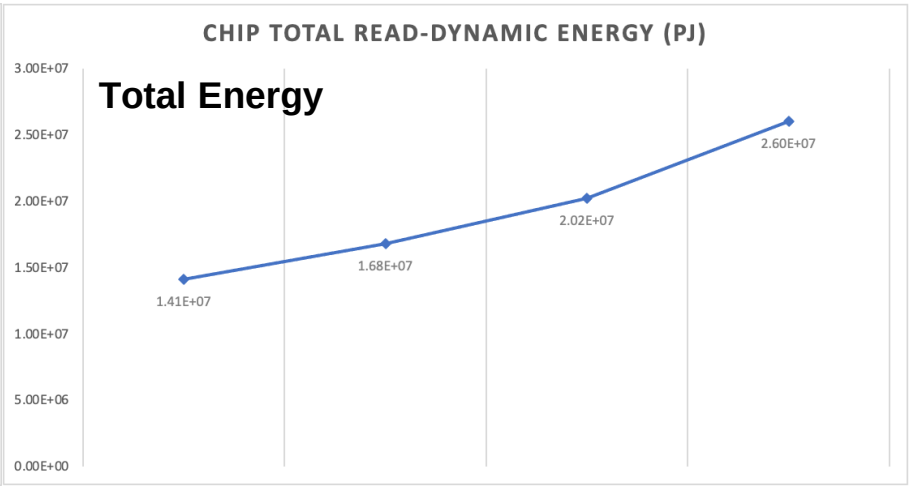
With higher ADC precision,
accuracy, **area** and **latency** increase.

ADC is a part of peripheral circuits.
→ **Don't affect floorplan.**

With ADC precision = 7,
ADC area / Total area = **58.5%.**

Inference Simulation Results(ADC precision=4/5/6/7)





With higher ADC precision,
leakage energy and **total energy** increase.

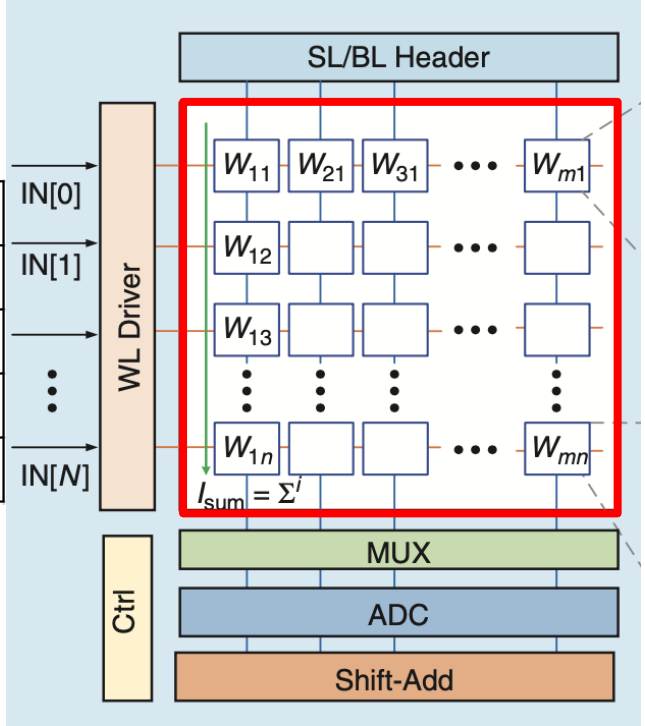
Higher leakage power causes **lower power efficiency**.

With ADC precision = 7,
ADC total energy / Total energy = **54.2%.**

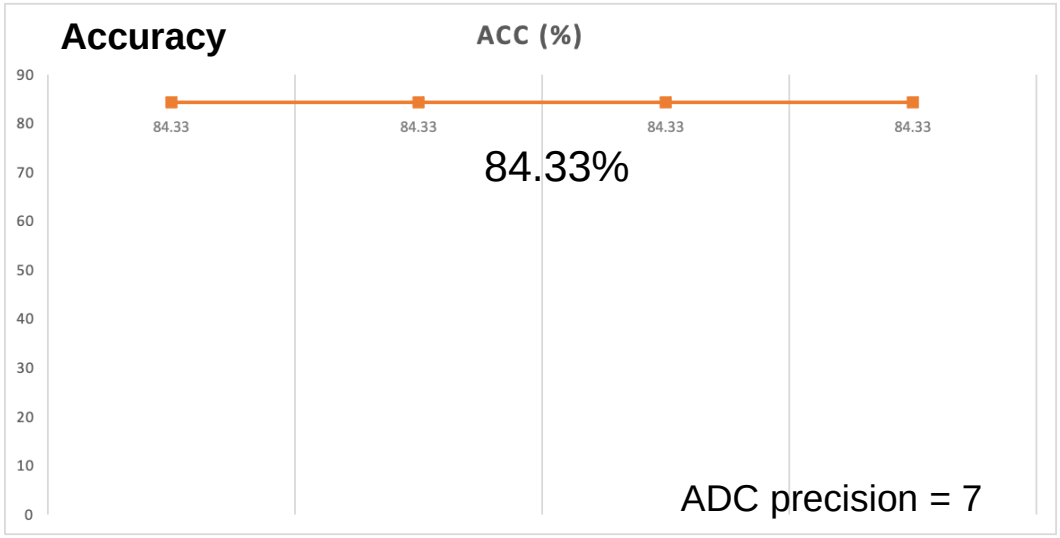
2. Subarray size

Subarray size affects chip floorplan.

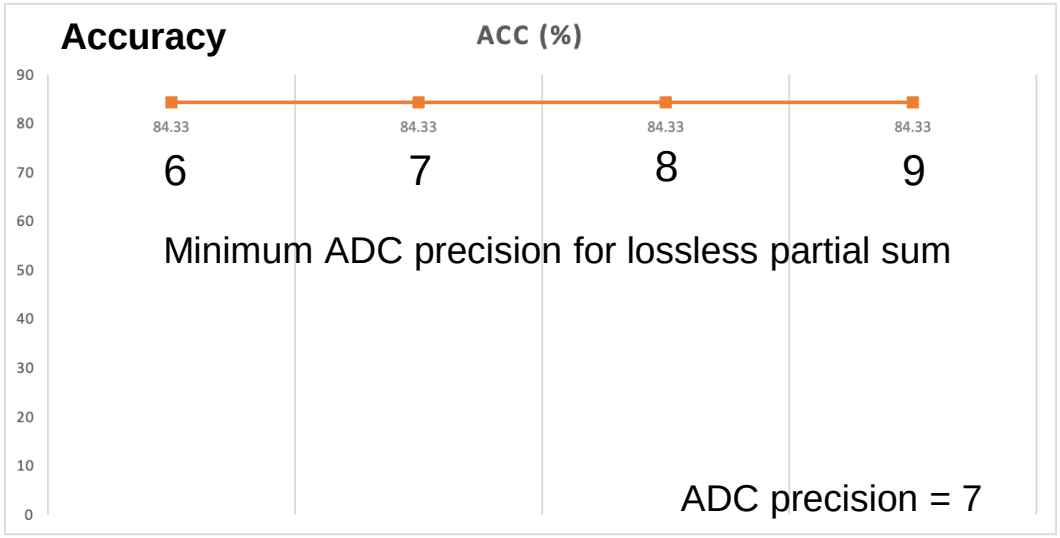
Hardware Parameters	default	simulation
ADC precision	5 bits	4/ 5/ 6/ 7
Subarray size	128*128	16/ 32/ 64/ 128
Weight bit-width	8 bits	6/ 8/ 10/ 12
Bits per cell	4 bits	



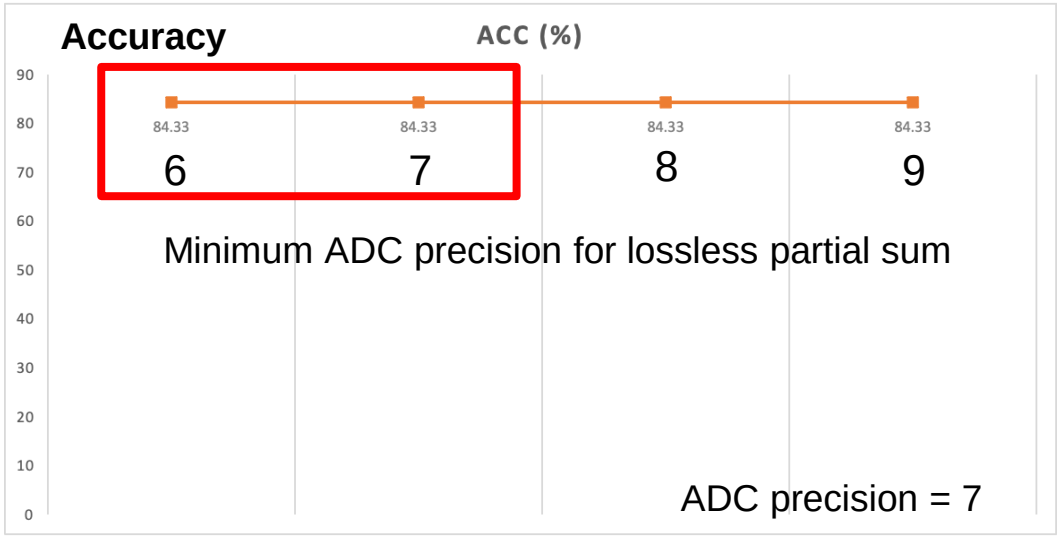
Inference Simulation Results(Subarray size=16/32/64/128)



Inference Simulation Results(Subarray size=16/32/64/128)



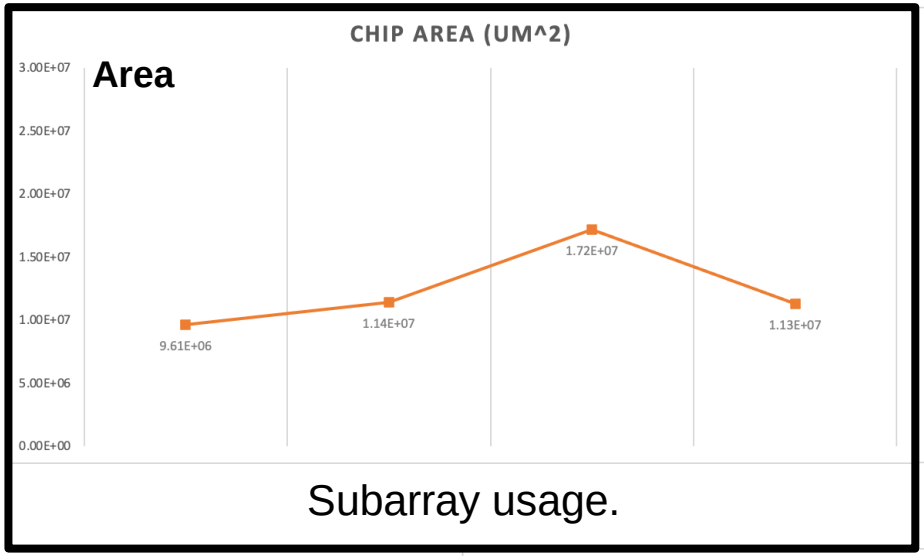
Inference Simulation Results(Subarray size=16/32/64/128)



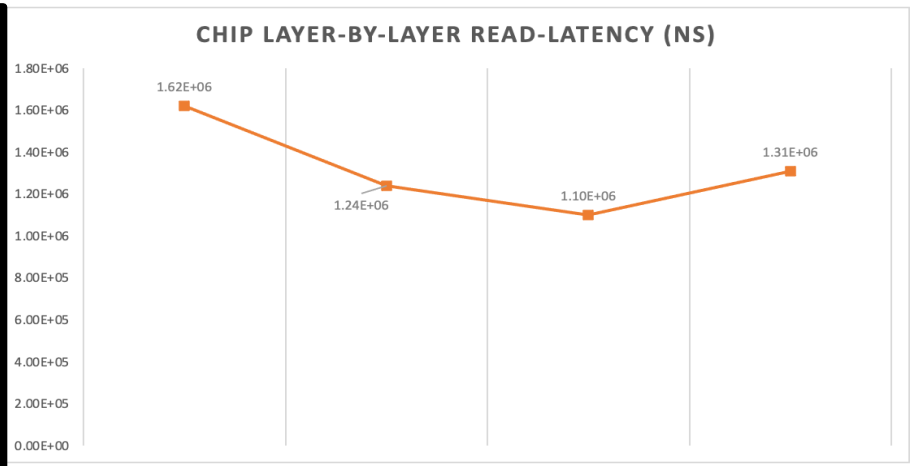
With smaller Subarray size(partial sum precision <= 7), **accuracy** is expected to be **higher**.

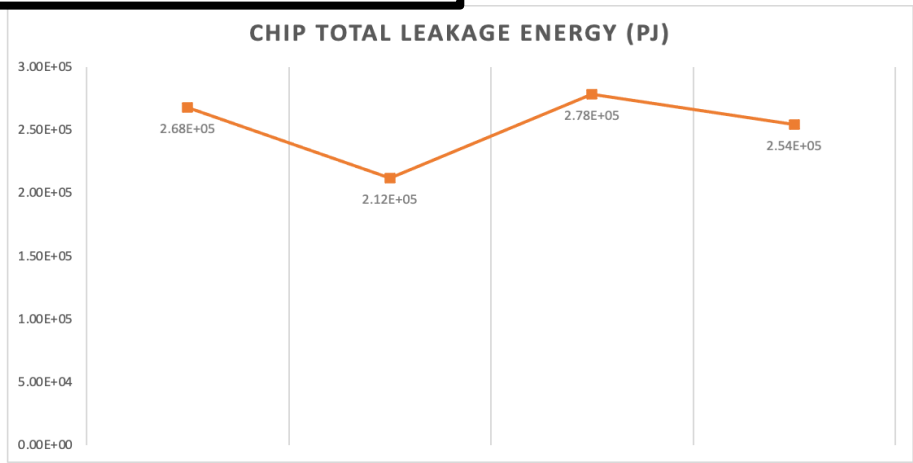
→ Accumulation modules' precision might be related to ADC precision.

Inference Simulation Results (Subarray size=16/32/64/128)

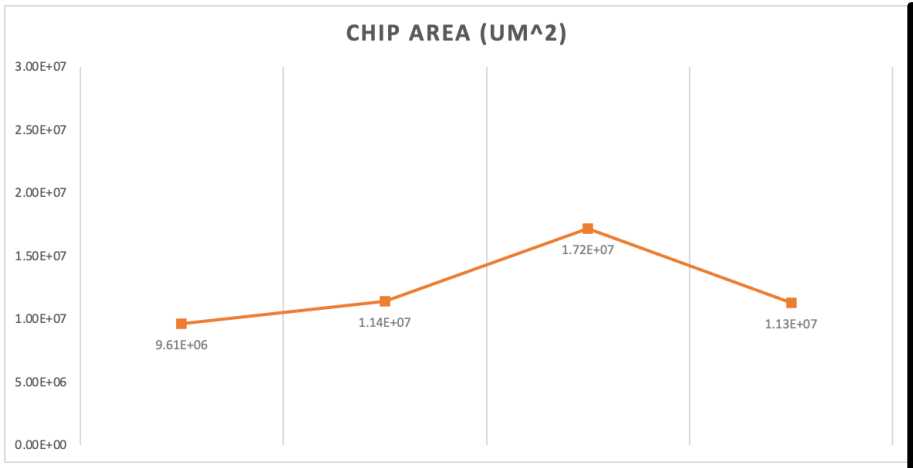


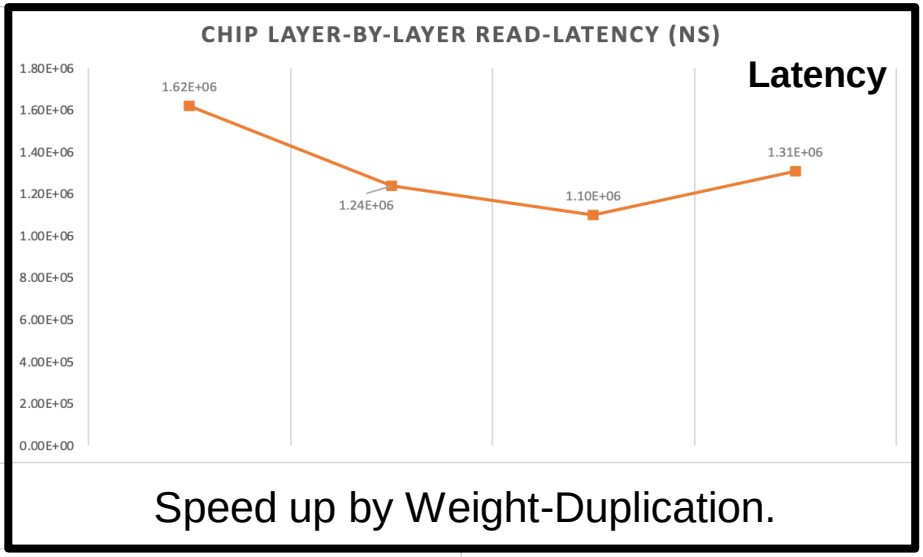
Subarray usage.



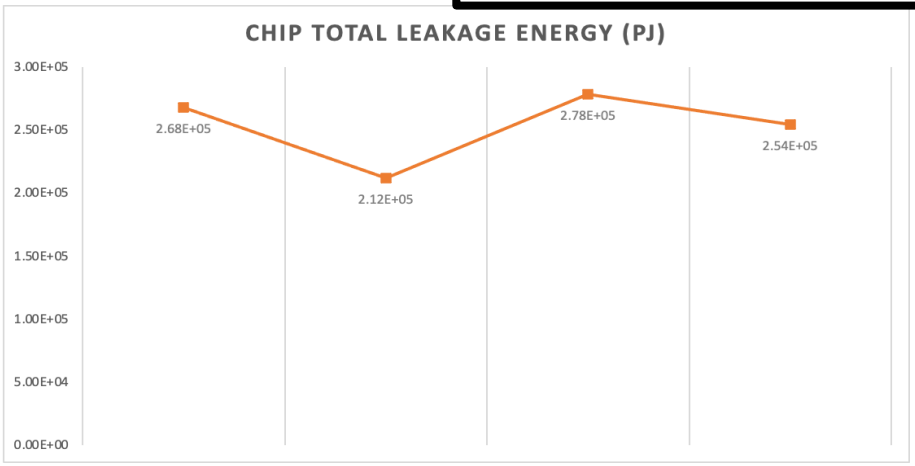


Inference Simulation Results (Subarray size=16/32/64/128)

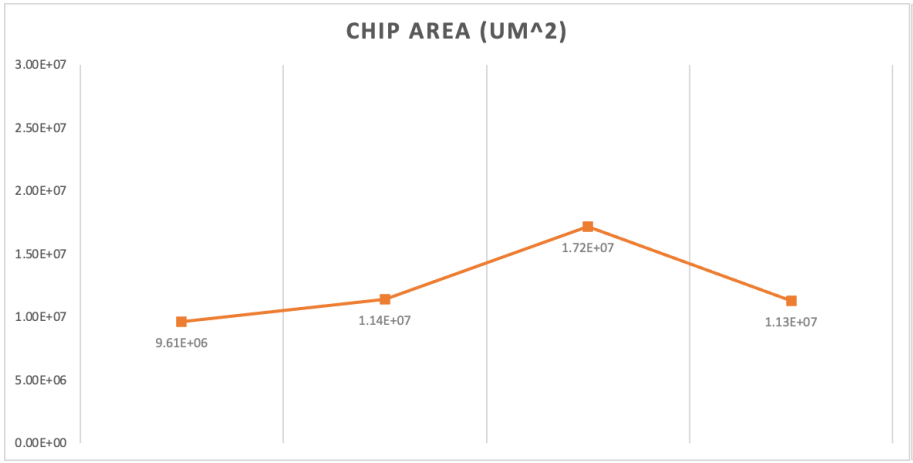


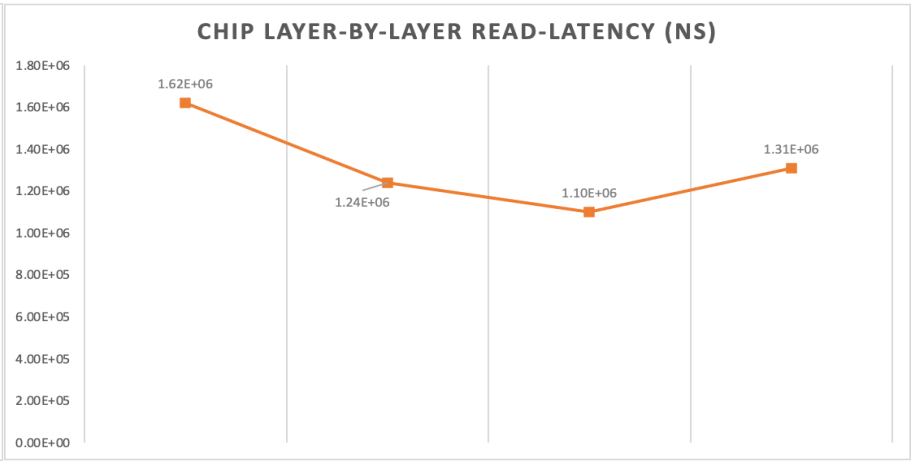


Speed up by Weight-Duplication.

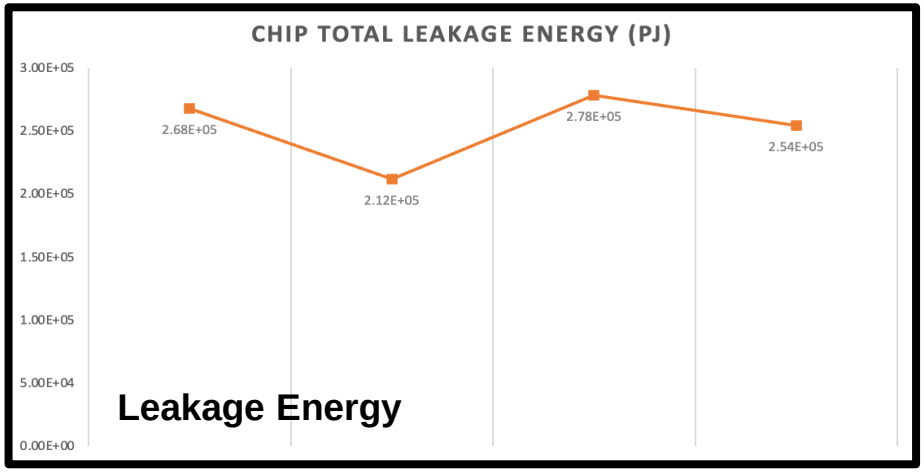


Inference Simulation Results (Subarray size=16/32/64/128)

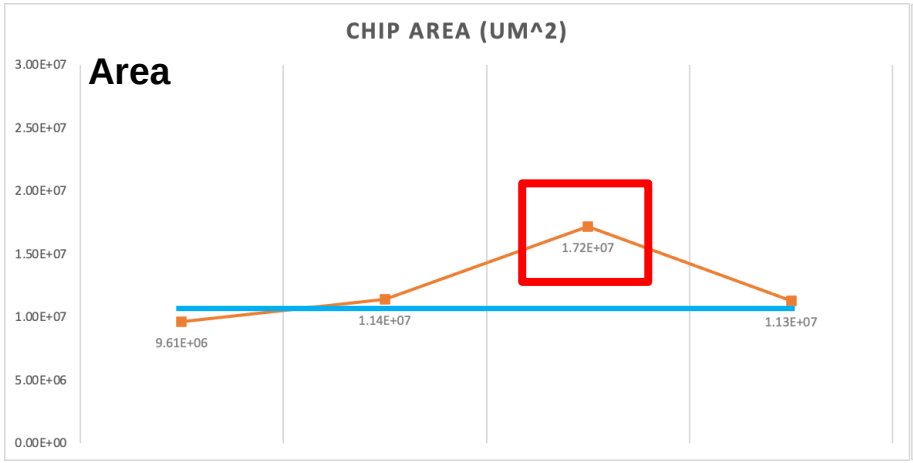


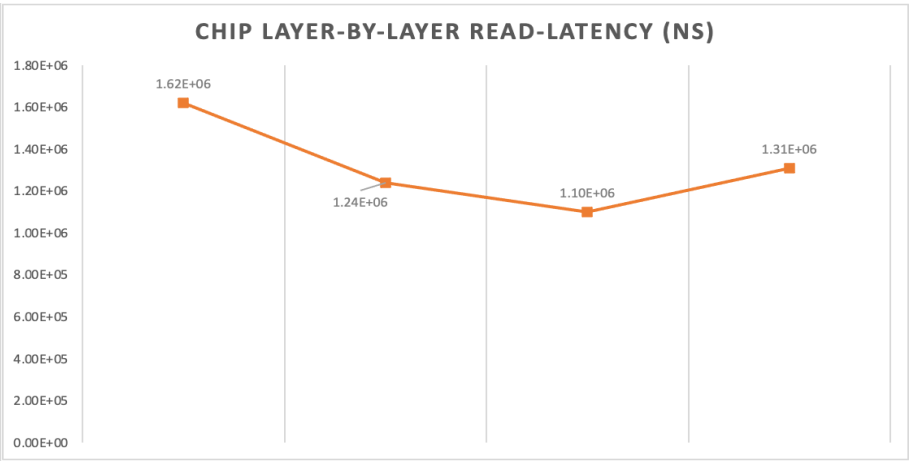


Leakage energy is related to **area** and **latency**.

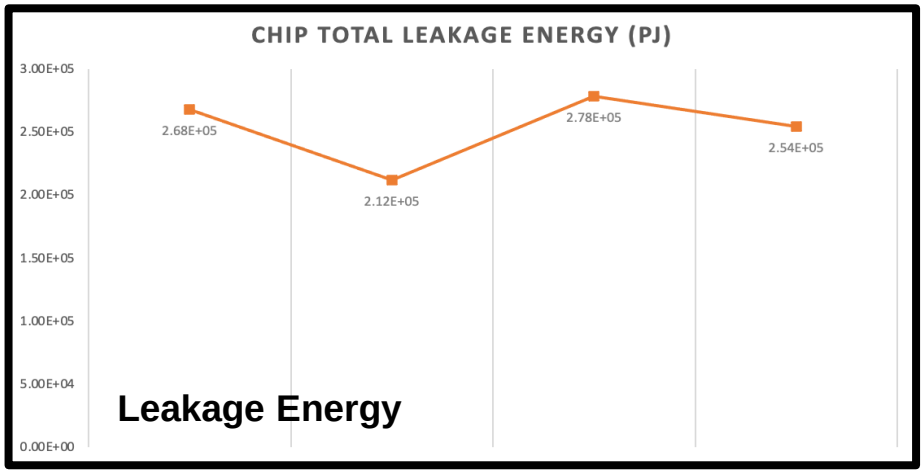


Inference Simulation Results (Subarray size=16/32/64/128)

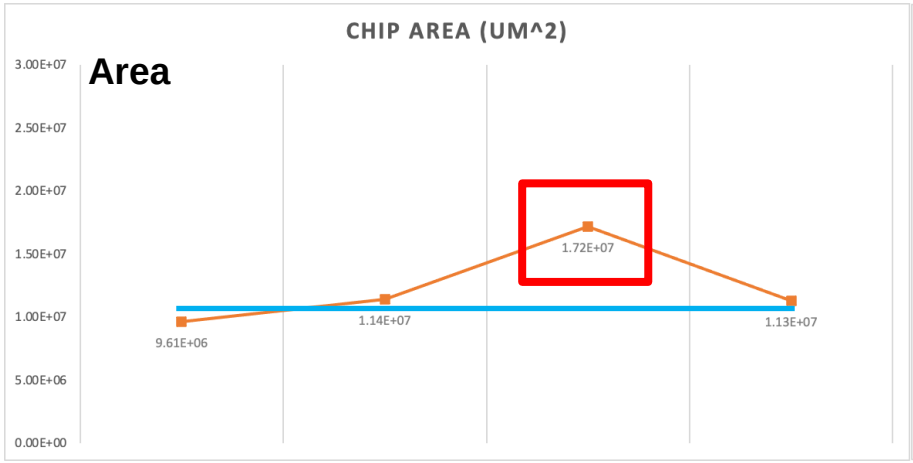


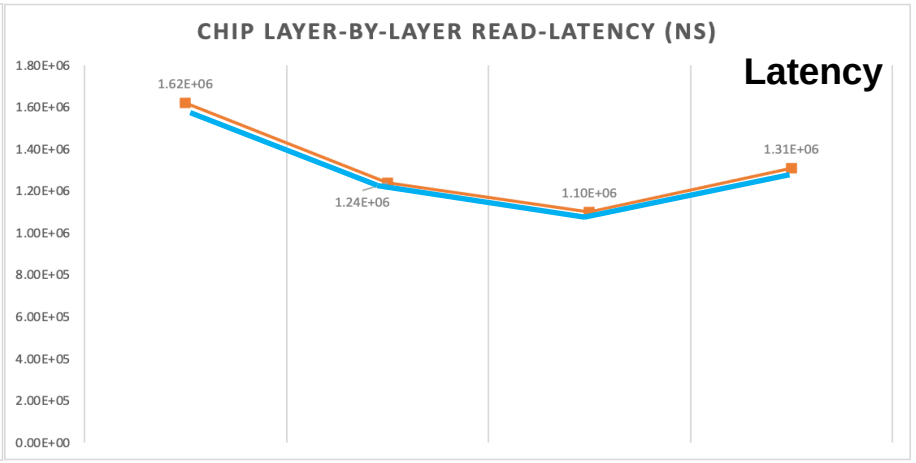


Leakage energy is related to **area** and **latency**.

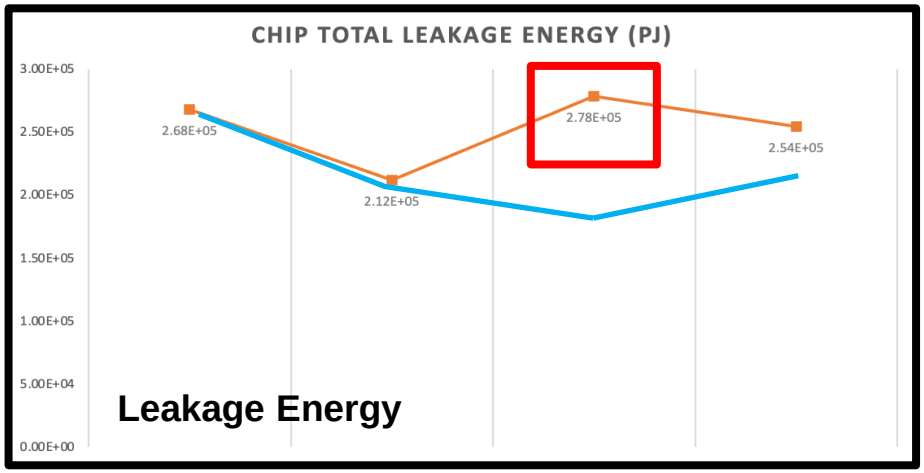


Inference Simulation Results (Subarray size=16/32/64/128)

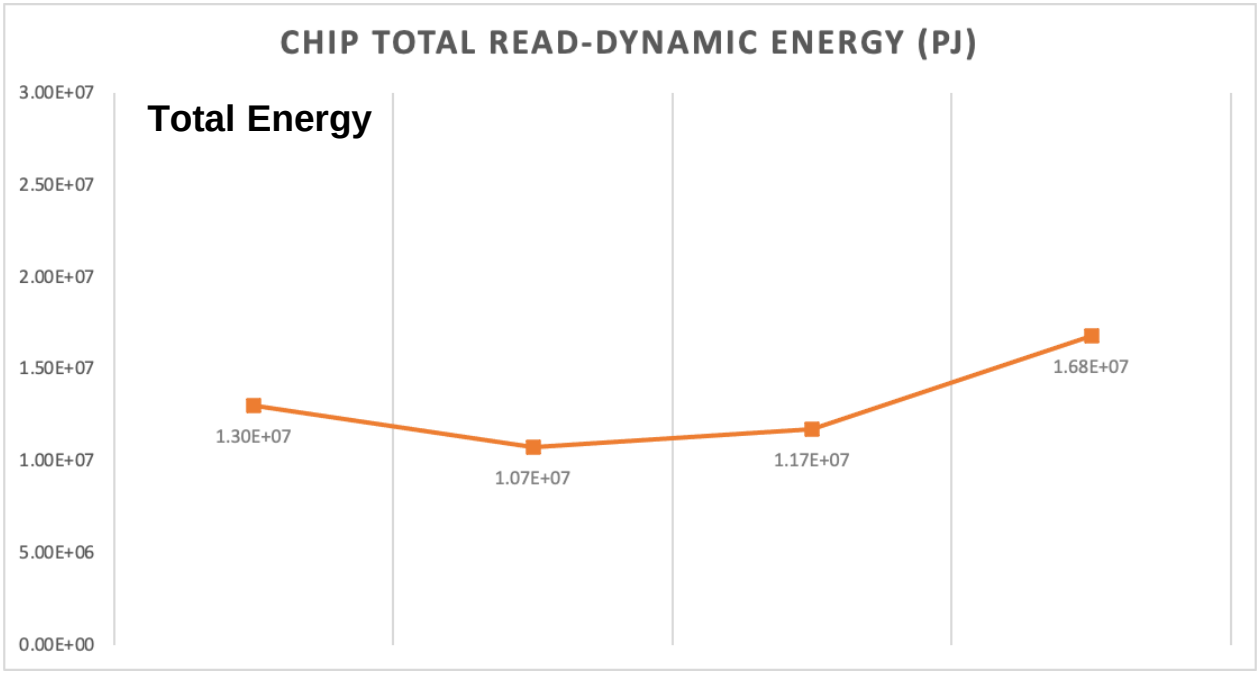




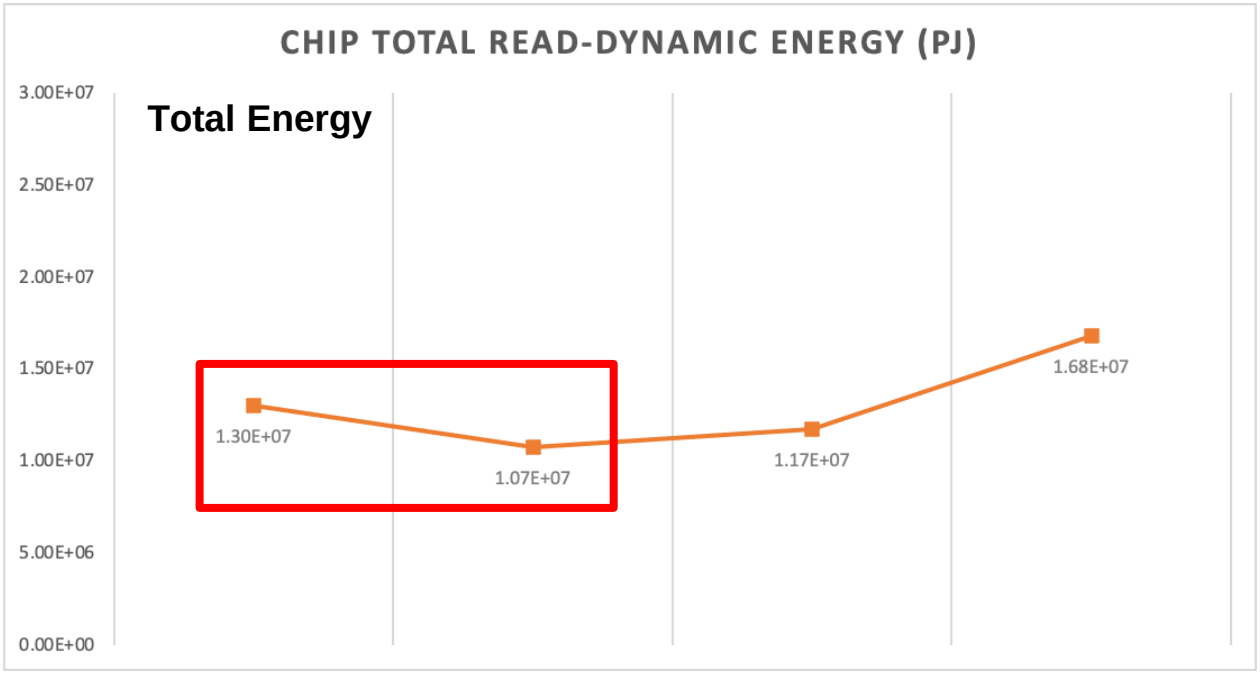
Leakage energy is related to **area** and **latency**.



Inference Simulation Results (Subarray size=16/32/64/128)

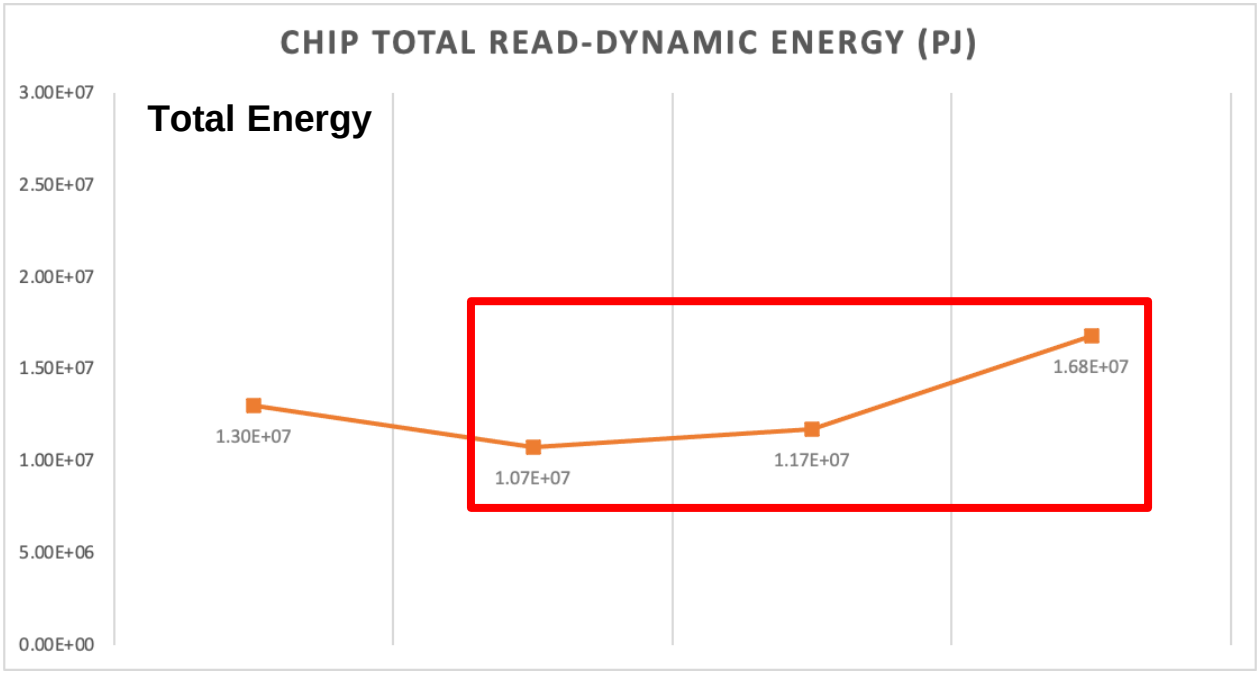


Inference Simulation Results (Subarray size=16/32/64/128)



Subarray usage and clock period are close, accumulation circuits dominates.

Inference Simulation Results (Subarray size=16/32/64/128)



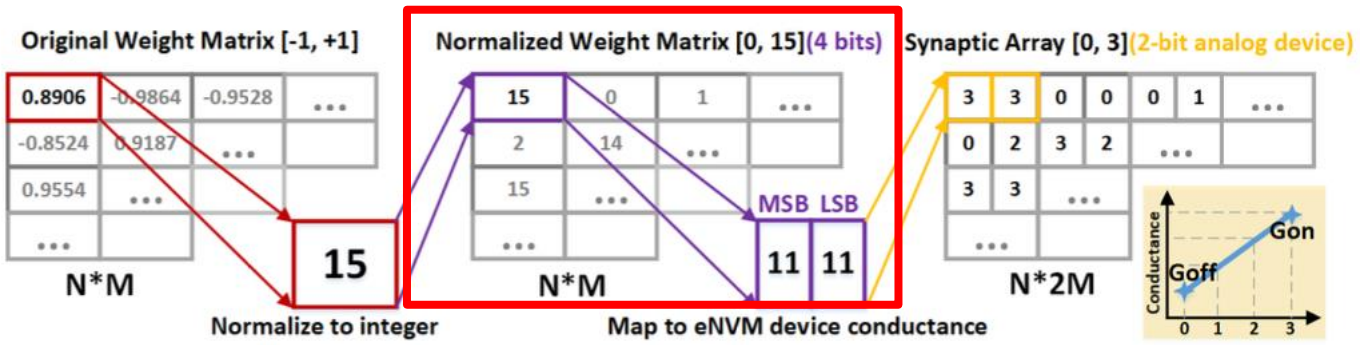
Clock period increases, ADC dominates.

3. Weight bit-width

Weight bit-width determines the number of cells to store one weight.

Hardware Parameters	default	simulation
ADC precision	5 bits	4/ 5/ 6/ 7
Subarray size	128*128	16/ 32/ 64/ 128
Weight bit-width	8 bits	6/ 8/ 10/ 12
Bits per cell	4 bits	

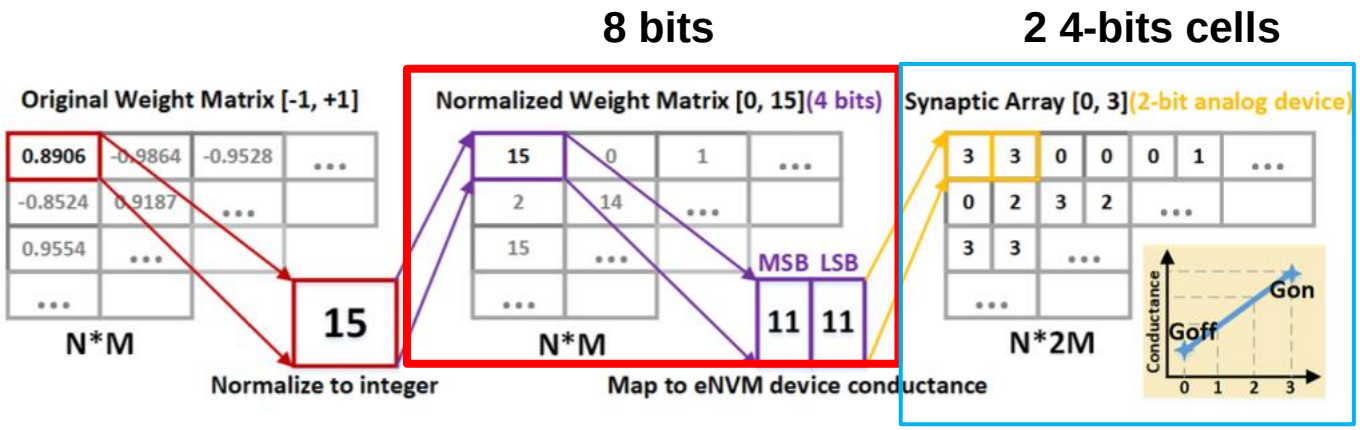
8 bits



3. Weight bit-width

Weight bit-width determines the number of cells to store one weight.

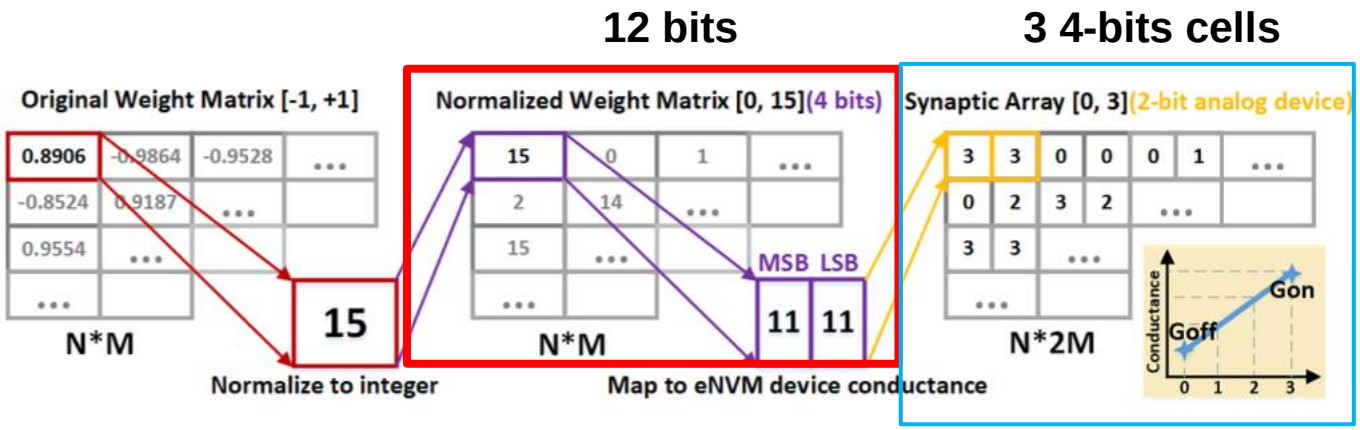
Hardware Parameters	default	simulation
ADC precision	5 bits	4/ 5/ 6/ 7
Subarray size	128*128	16/ 32/ 64/ 128
Weight bit-width	8 bits	6/ 8/ 10/ 12
Bits per cell	4 bits	



3. Weight bit-width

Weight bit-width determines the number of cells to store one weight.

Hardware Parameters	default	simulation
ADC precision	5 bits	4/ 5/ 6/ 7
Subarray size	128*128	16/ 32/ 64/ 128
Weight bit-width	8 bits	6/ 8/ 10/ 12
Bits per cell	4 bits	



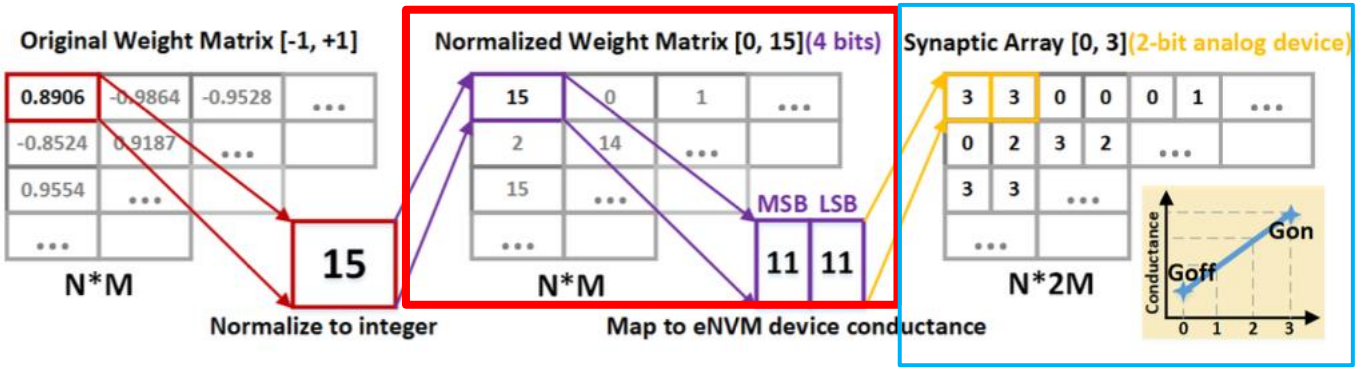
3. Weight bit-width

Weight bit-width determines the number of cells to store one weight.

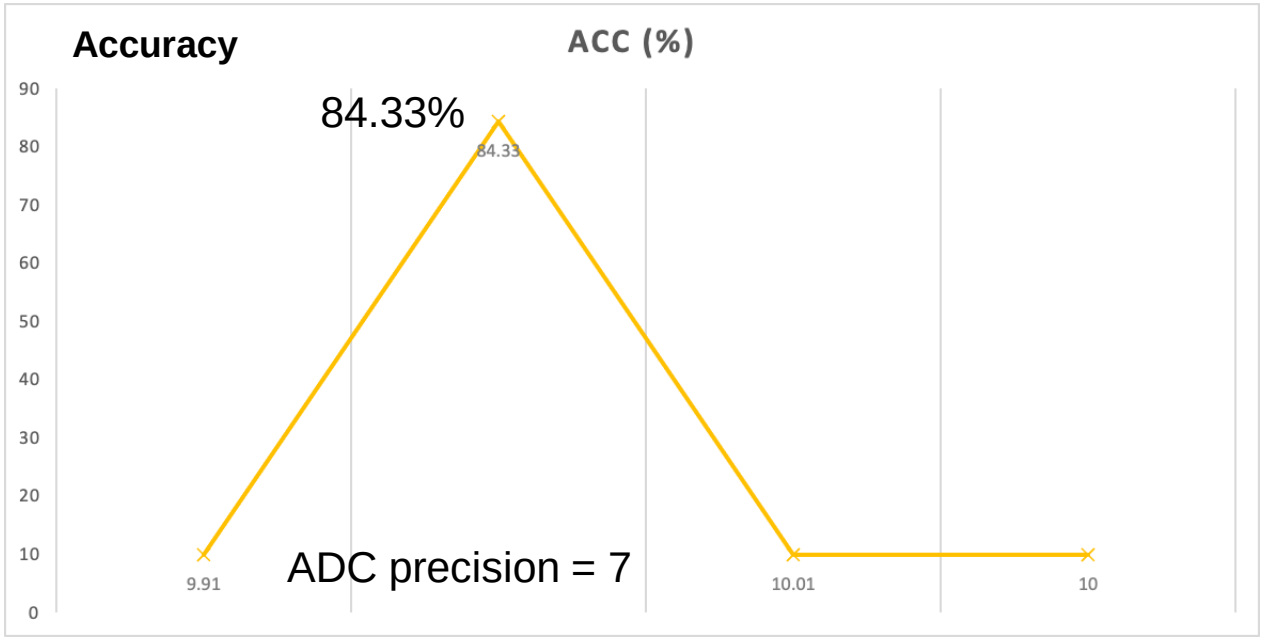
Hardware Parameters	default	simulation
ADC precision	5 bits	4/ 5/ 6/ 7
Subarray size	128*128	16/ 32/ 64/ 128
Weight bit-width	8 bits	6/ 8/ 10/ 12
Bits per cell	4 bits	

12 bits

3 4-bits cells → **Might affect floorplan**

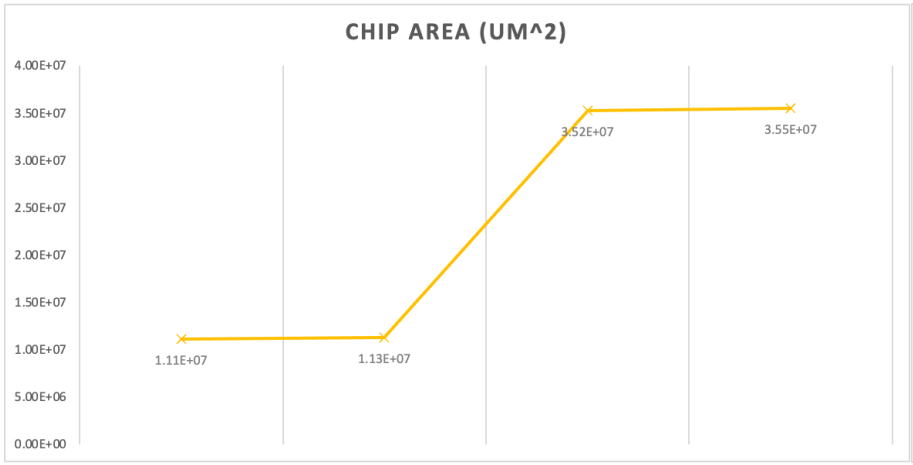


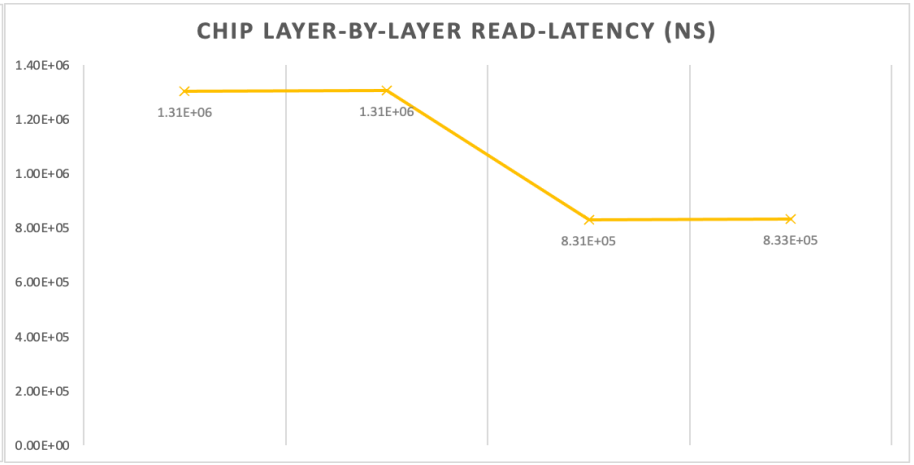
Inference Simulation Results(Weight bit-width=6/8/10/12)

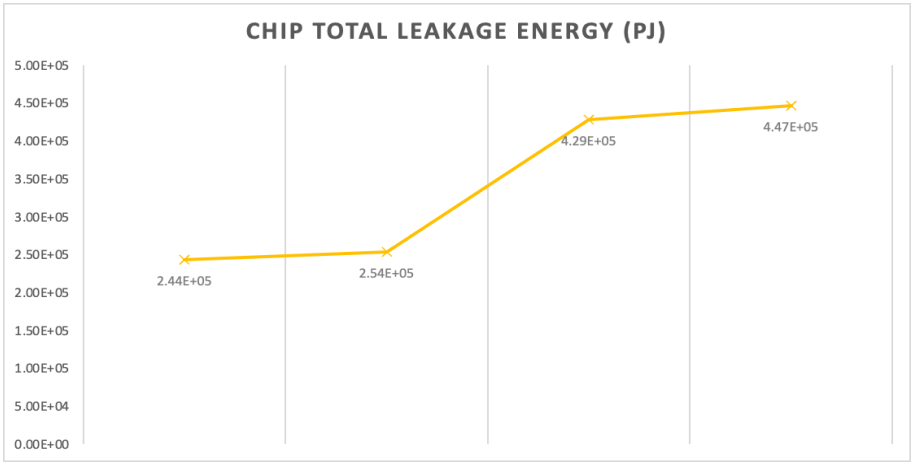


The model contains a massive number of weights and inference includes many weights multiplication.

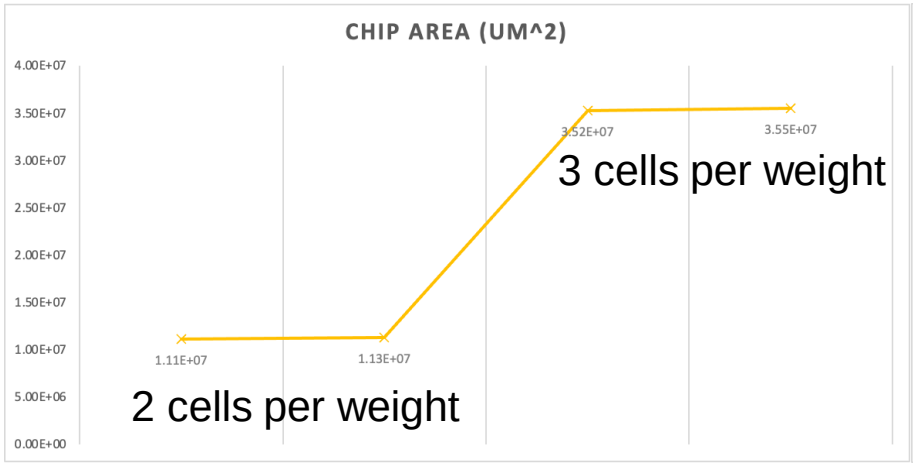
Inference Simulation Results(Weight bit-width=6/8/10/12)

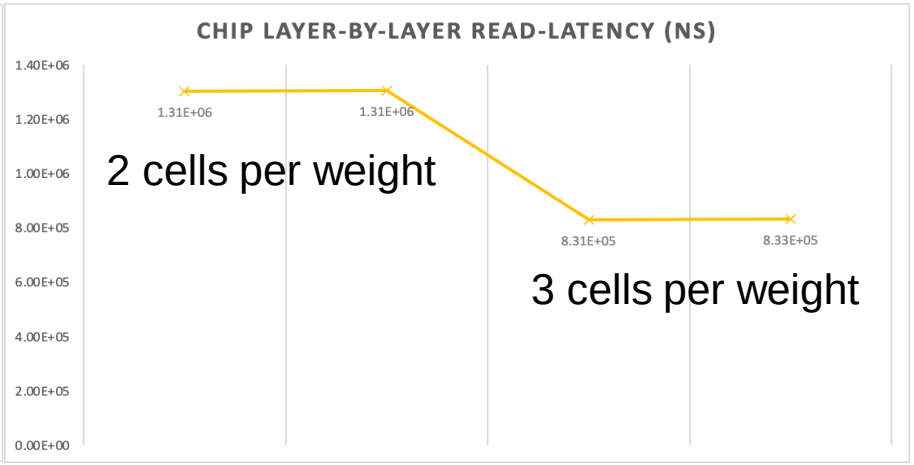


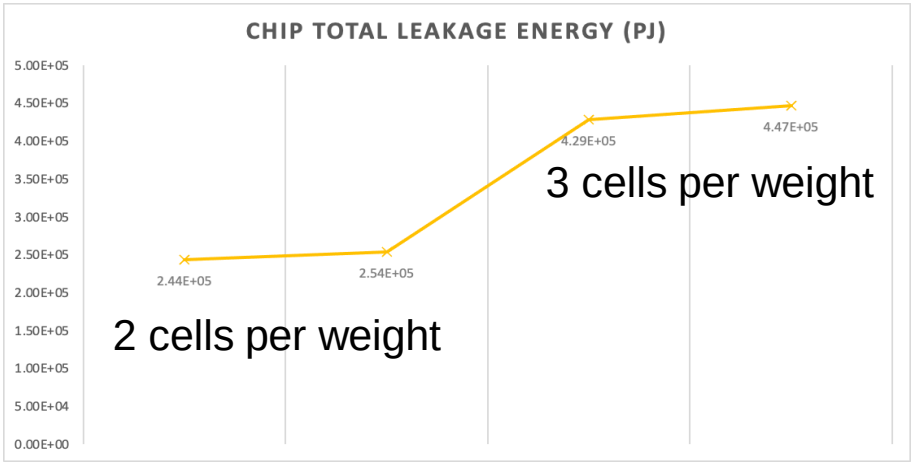




Inference Simulation Results(Weight bit-width=6/8/10/12)

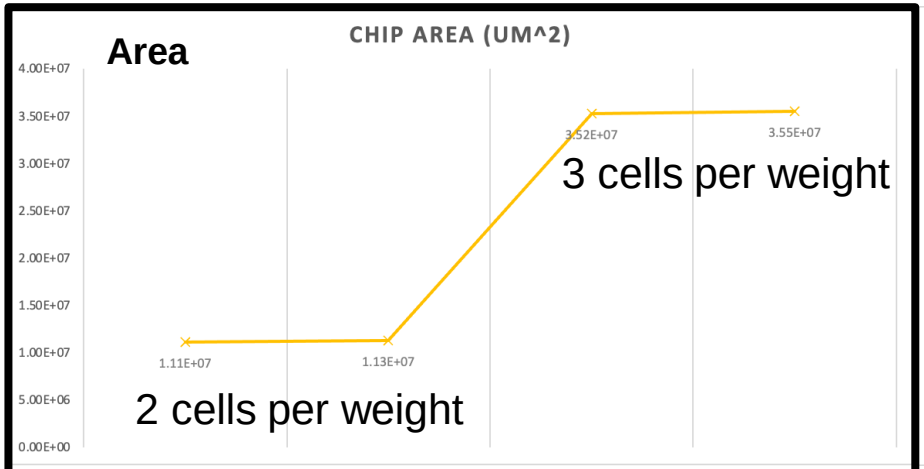






Inference Simulation Results(Weight bit-width=6/8/10/12)

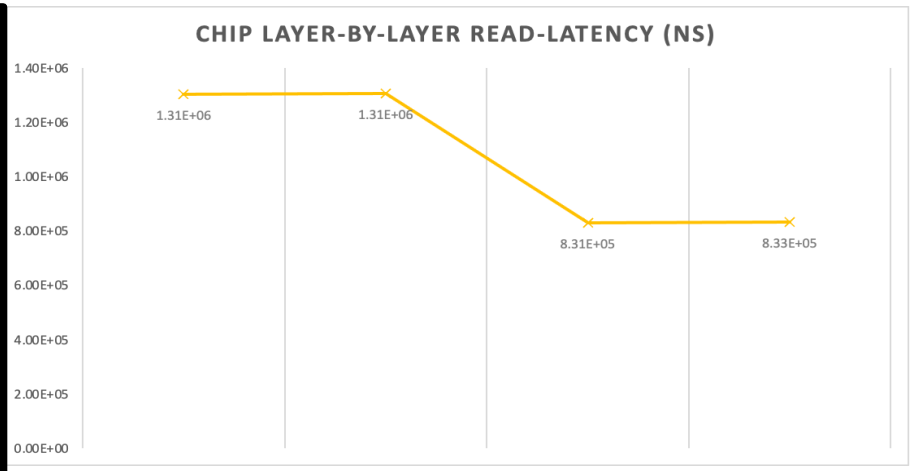
Area

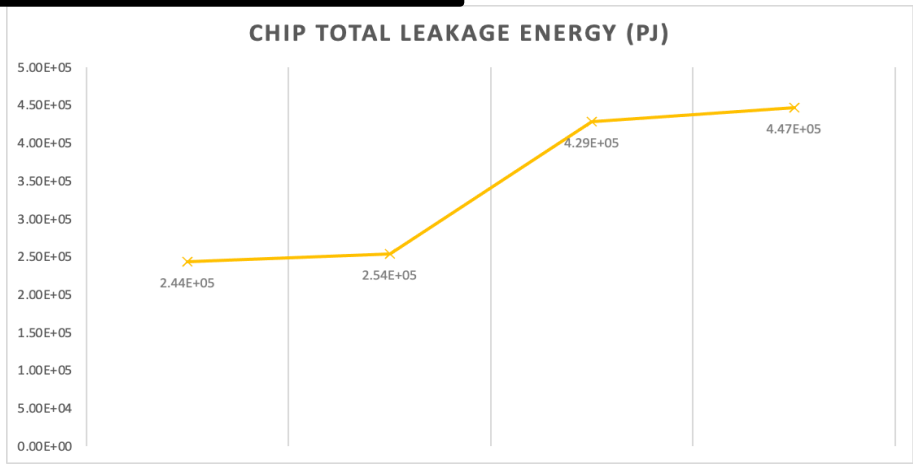


3 cells per weight

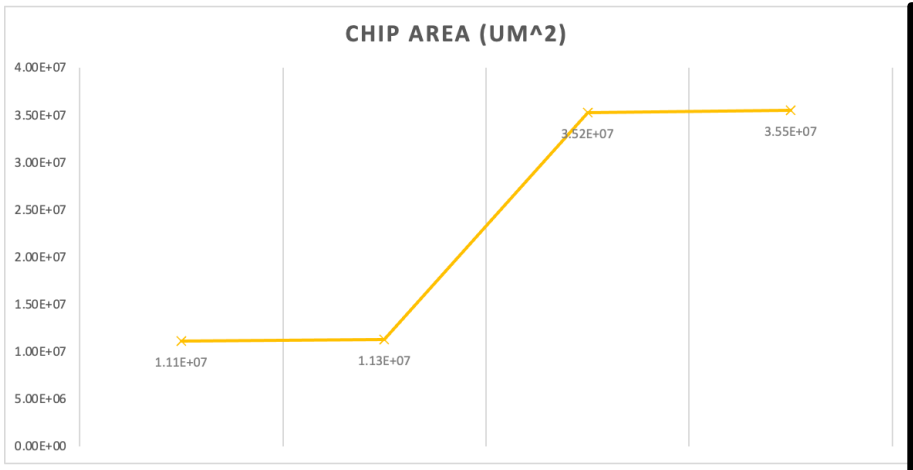
2 cells per weight

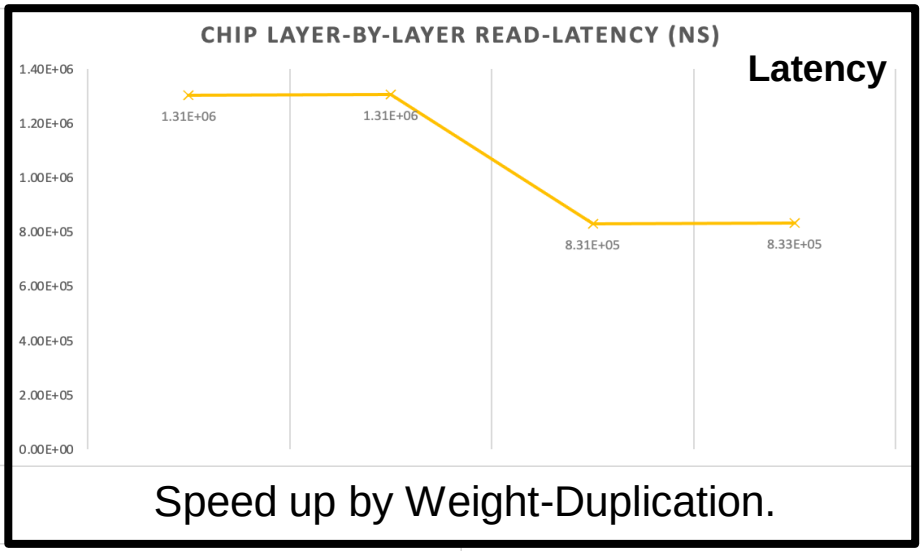
More subarrays are needed.



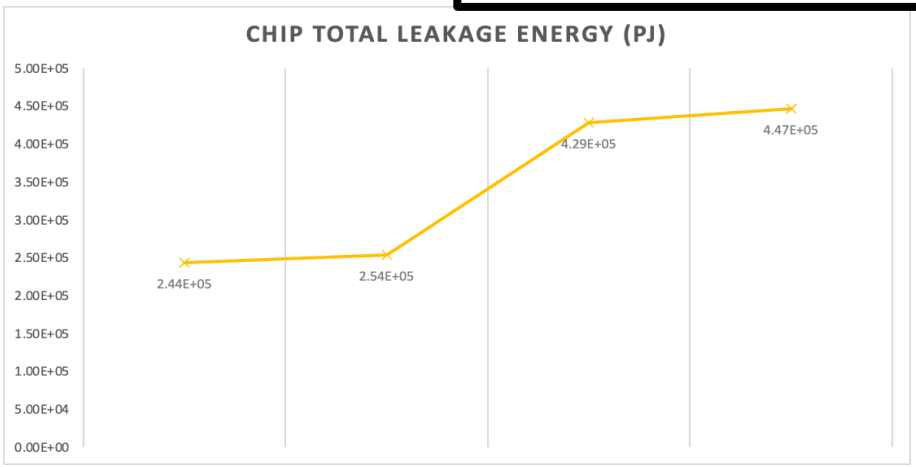


Inference Simulation Results(Weight bit-width=6/8/10/12)

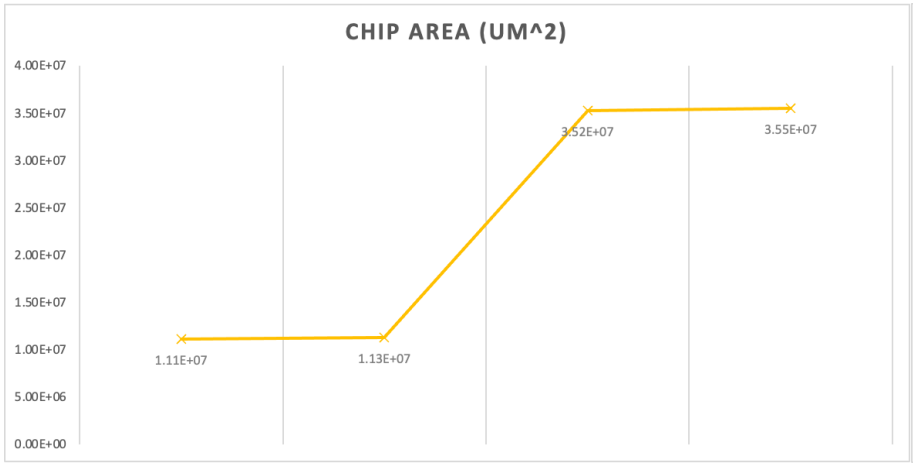


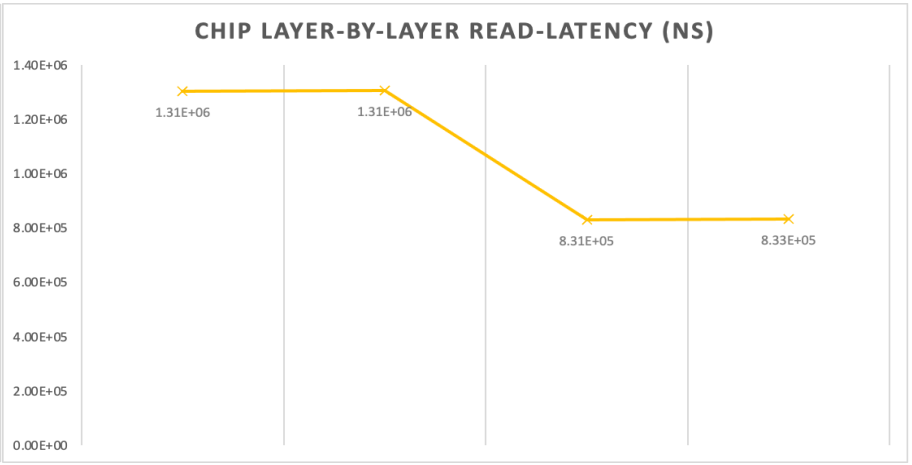


Speed up by Weight-Duplication.

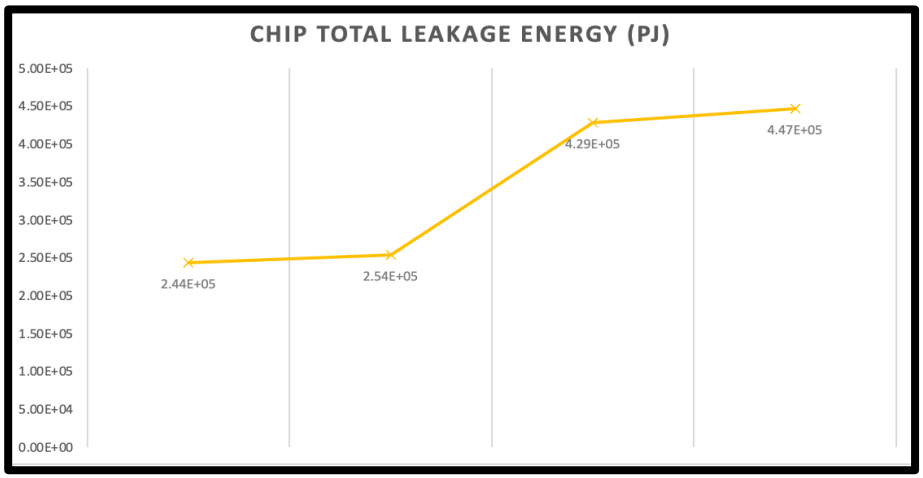


Inference Simulation Results(Weight bit-width=6/8/10/12)





Leakage energy is related to **area** and **latency**. In this case, **area dominates**.



Summary

- CIM addresses memory-wall problem by emerging VMM into memory subarrays.
- DNN+NeuroSim emulates the DNN inference performance on the CIM based DLA.
- The chip hierarchy: chip → tile → PE → synaptic array.
- Novel mapping for CNN models could reduce latency and energy.
- ADC is a major bottleneck for area and power efficiency.
- There is usually a trade-off between area and latency.

Future work

- **Change DNN+NeuroSim to bit-plane structure.**

  原本NeuroSim mapping的方式，是按照layer做partition，尤其同一weight的多個memory cells會在同一個synaptic array。而bit-plane的方法是將各個layer的weight按照significance做partition，相同significance的weight bit會被放在一起，這樣的好處在於要增減weight bit-width的時候不會影響到其他significance的weight bits的mapping，加上partial sum要做shift and add的部分也會更為簡單。

- **Multi-exit architecture.**

  相較一般model只有final output，Multi-exit architecture在中間的layer拉出多個output，產生多個子model。通常model越深，越有能力執行複雜的任務，卻也伴隨著hardware effect影響越來越大的問題，形成trade-off，所以這樣的設計能讓model提供更多的彈性。

- **Find out how other hardware parameters influence the whole performance.**

Reference & Resource

[1] **Compute-in-Memory Chips for Deep Learning: Recent Trends and Prospects**
Shimeng Yu, Hongwu Jiang, Shanshi Huang, Xiaochen Peng, and Anni Lu

[2] **DNN+NeuroSim Framework V1.3 (https://github.com/neurosim/DNN_NeuroSim_V1.3)**
Developers: Xiaochen Peng, Shanshi Huang, and Anni Lu PI: Prof. Shimeng Yu, Georgia Institute of Technology

Thanks for listening !