MNSIM 2.0: A Behavior-Level Modeling Tool for Memristor-based Neuromorphic Computing Systems

Zhenhua Zhu[1*], Hanbo Sun[1*], Kaizhong Qiu[1*], Lixue Xia[2], Gokul Krishnan[3], Guohao Dai[1], Dimin Niu[2], Xiaoming Chen[4], X. Sharon Hu[5], Yu Cao[3], Yuan Xie[6], Yu Wang[1], Huazhong Yang[1]

[1]Dept of EE, BNRist, Tsinghua University, [2]Alibaba Group, [3]Arizona State University,[4]Institute of Computing Technology, Chinese Academy of Sciences, [5]University of Notre Dame, [6]University of California, Santa Barbara

E-mail: yu-wang@tsinghua.edu.cn

GLSVLSI 2020

Outline

- Background and Introduction
- Overview of MNSIM 2.0
- Hardware Performance Modeling & Evaluation Flow
- CNN Model Optimization Flow
- Conclusion and Future Work

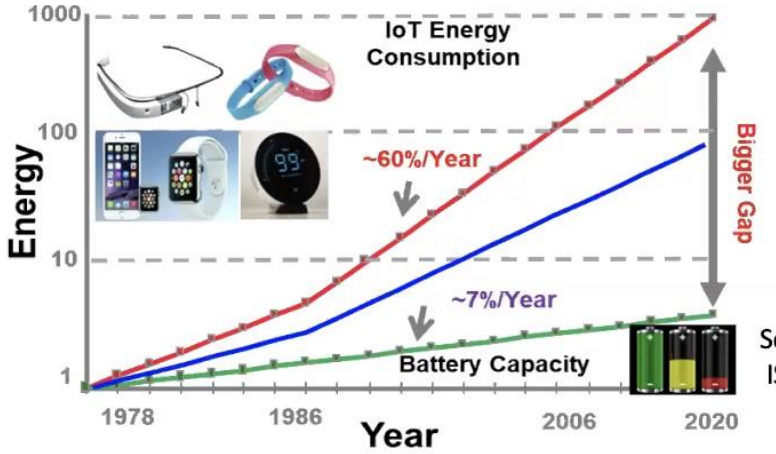
Outline

- Background and Introduction
- Overview of MNSIM 2.0
- Hardware Performance Modeling & Evaluation Flow
- CNN Model Optimization Flow
- Conclusion and Future Work

AI Era Puts Higher Demands on Energy Efficie

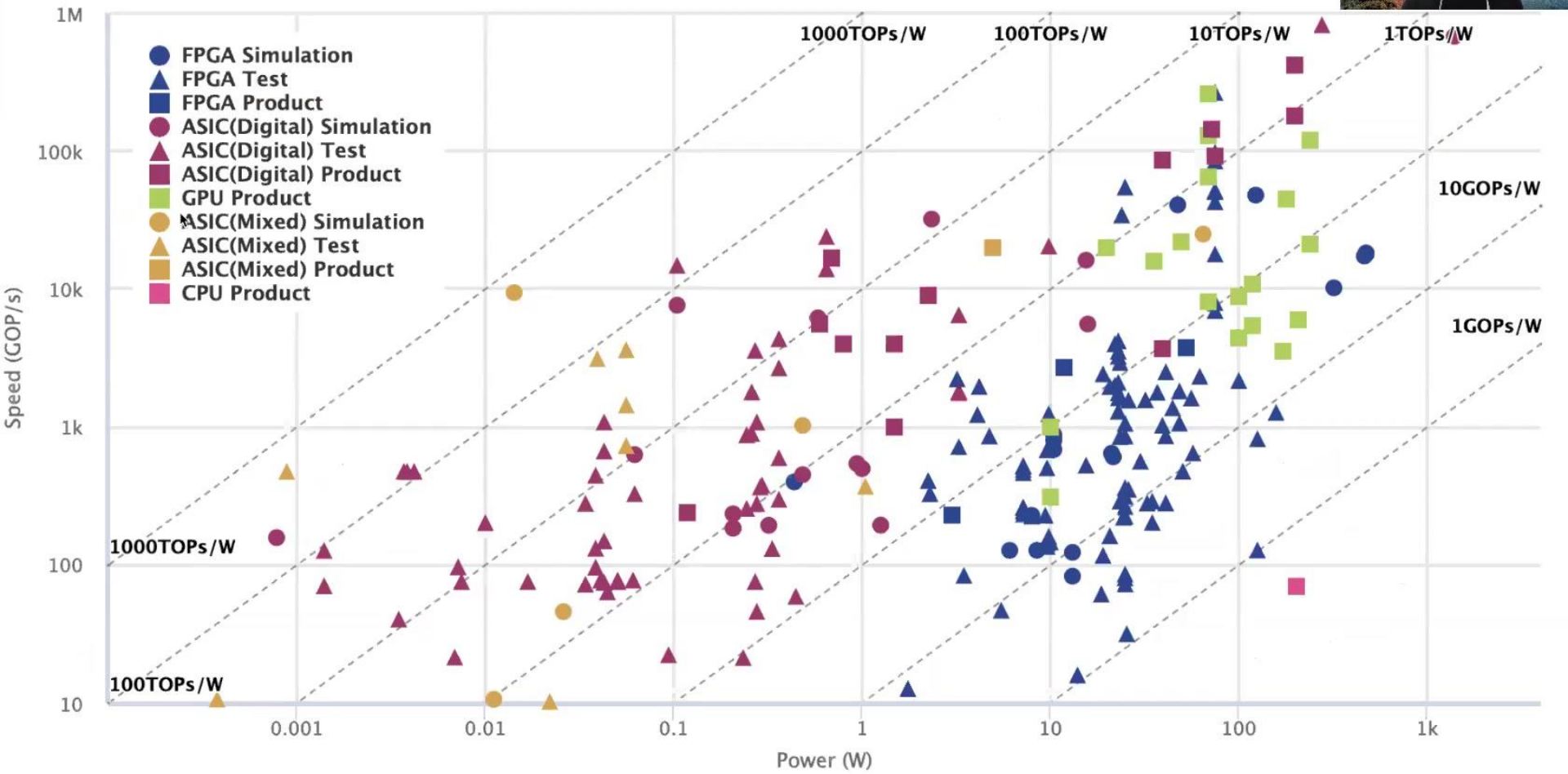
According to the White Paper on the Status of Energy Consumption in China's Data Centers, the energy consumption of data centers in 2015 will be nearly **100 billion kWh**, accounting for about **1.5%** of the total electricity consumption of the whole society.

According to data released by the US Environmental Protection Agency, by 2020, the electricity consumed by US data centers will account for **9%** of the nation's electricity use.



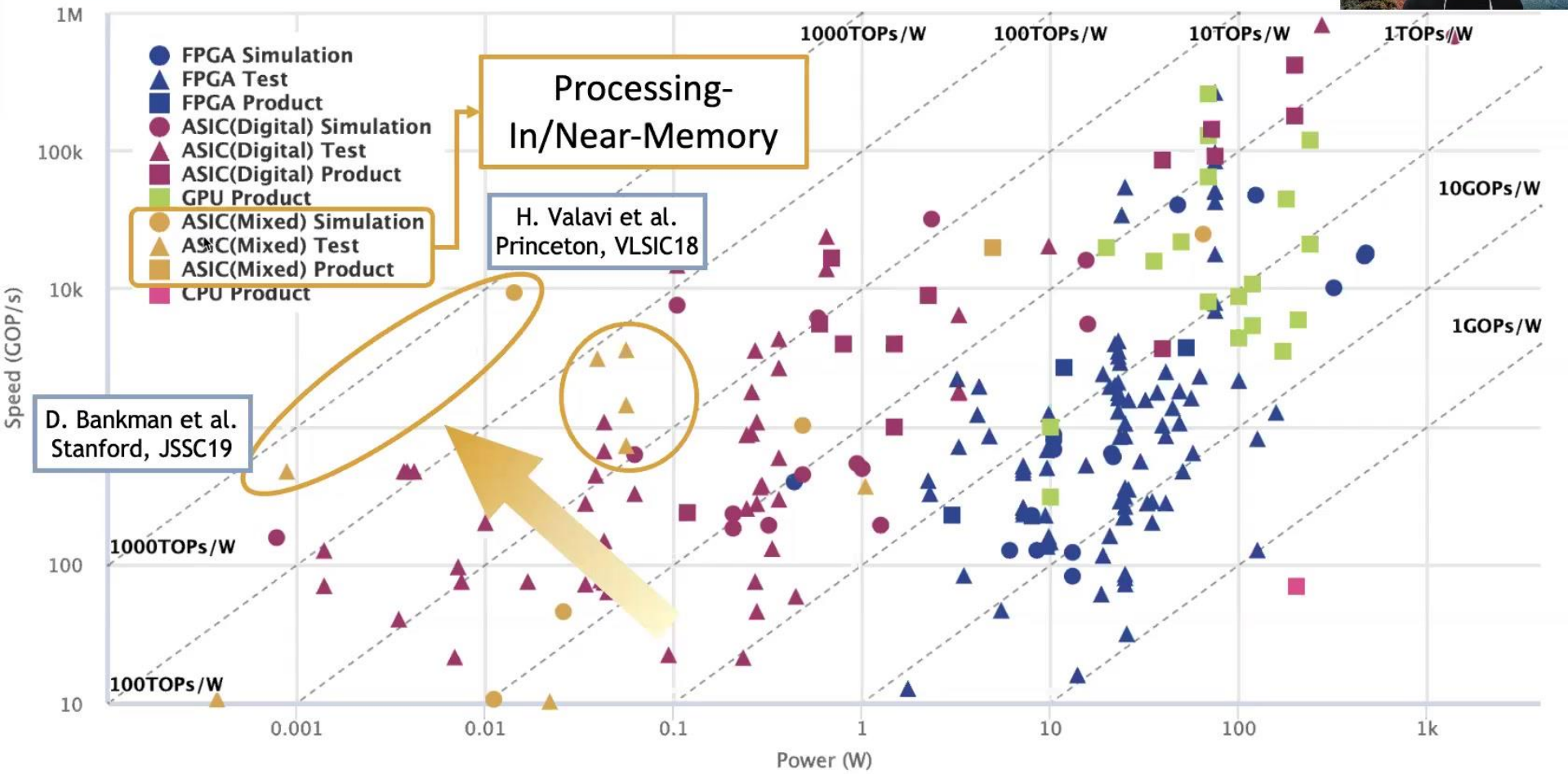
Source: Y. Liu, ISSCC 16

Comparison of NN Accelerator (~2020



https://nicsefc.ee.tsinghua.edu.cn/projects/neural-network-accelerator/

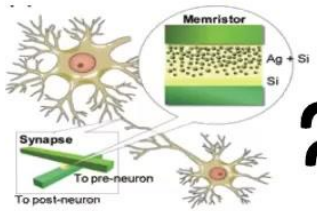
Comparison of NN Accelerator (~2020



https://nicsefc.ee.tsinghua.edu.cn/projects/neural-network-accelerator/

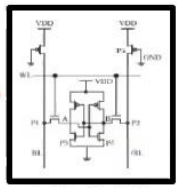
New Computation Paradigm: Processing-In-Memory (P

Entire Energy = Computation Energy + Data Movements Energy
= #Gate Number x #Gate Flips x Energy/Flip + f (#Data, Distance, Bandwidth)

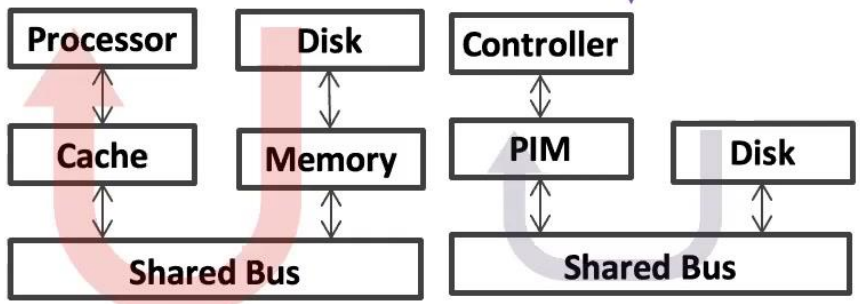


1 RRAM Cell
(0.1~0.2 nJ/Read)

≈ × + + +



1 m-bit Multiplier + 1 m-bit Adder
+
1 m-bit Reg. (SRAM)



Von Neumann Architecture

Processing-In-Memory

Advantage 1
Realize vector operations by crossbar read operation, reduce resource overhead & computation energy

Advantage2
Complete calculation in memory to reduce data transfer volume and distance

Support of Algorithms: The return of analog computation
The development of neural networks and other algorithms makes inaccurate calculation still have accurate results

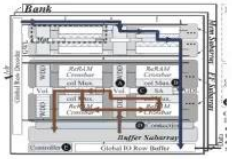
PIM-based CNN Accelerator

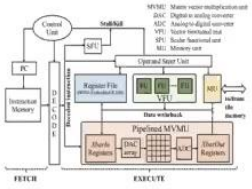
Academia

- RRAM chip, NTHU, ISSCC19
- 256 rows x 512 columns, macro level
- CNN computing: 11.75~14.6ns
- Energy Efficiency: 21.9~53.17 TOPS/W

- RRAM chip, THU, Nature
- 8 crossbars, 5-layer for MNIST, system level
- Energy Efficiency: two orders of magnitude greater than that of state-of-the-art GPU

Chip Demo





- **PRIME**, UCSB, ISCA16
- Use RRAM for NN computation
- Compared with AISC (DianNao), achieves 2360x speedup, reduces the energy consumption by 895x
- **PUMA**, HP, ASPLOS19
- Supports MLP, LSTM, CNN, etc.
- Contains an ISA and compiler design
- Achieves 2446x energy efficiency compared to GPUs

Architecture Design

Industry

MYTHIC

- Based on embedded NOR FLASH
- Highest energy efficiency: 4TOPS/W
- https://www.mythic-ai.com/technology/

IBM

- Based on PCM
- 8-bit precision device (still in research stage)
- IEDM 2018

WITINMEM 知存科技

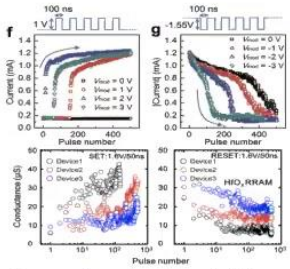
- Based on NOR Flash

SYNTIANT

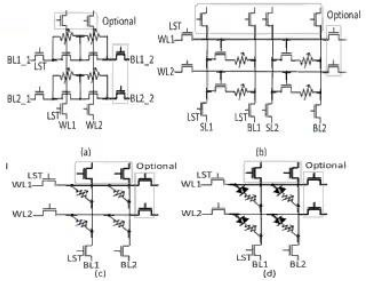
- Based on NOR Flash
- Built for voice applications

Challenge: the Design Space of PIM is Hu

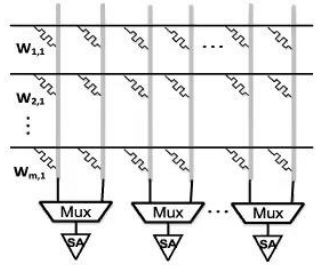
- **Huge design space** of PIM makes the circuit-level simulation (e.g. SPICE) consume **long time** (several days ~ months)



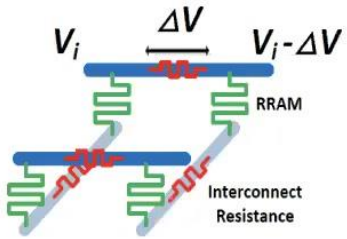
Device characteristics vary greatly under different technologies



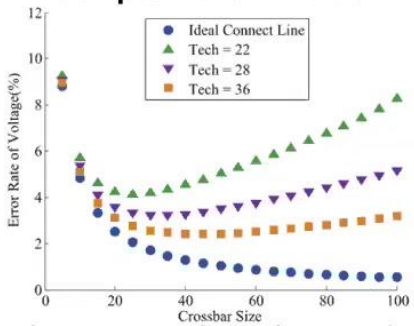
Crossbar structure design is complex and diverse



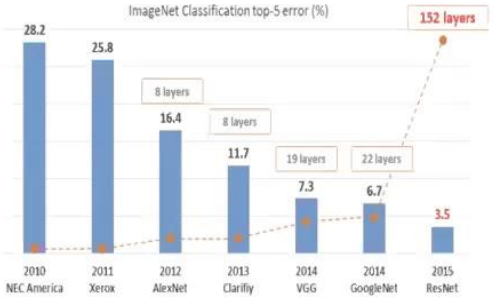
Peripheral circuits contain multiple changeable parameters



IR-drop and other non-ideal factors in computing:
Crossbar size needs optimization



Performance of single crossbar is already influenced by multiple factors

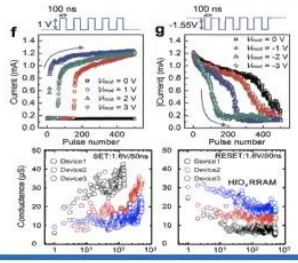


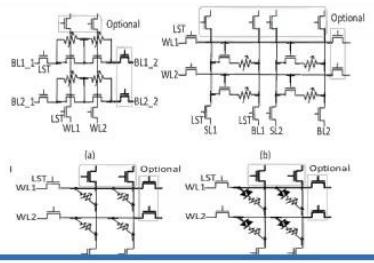
Algorithm hyperparameter space is huge

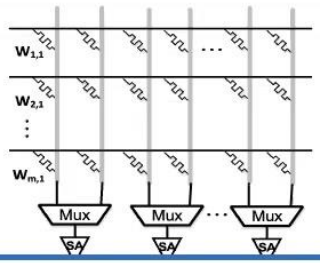
Ref: Lixue Xia, Boxun Li, Tianqi Tang, Peng Gu, Pai-yu Chen, Shimeng Yu, Yu Cao, Yu Wang, Yuan Xie, Huazhong Yang, MNSIM: Simulation Platform for Memristor-based Neuromorphic Computing System , in IEEE TCAD 2018.

Challenge: the Design Space of PIM is Hu

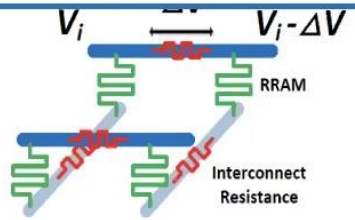
- **Huge design space** of PIM makes the circuit-level simulation (e.g. SPICE) consume **long time** (several days ~ months)



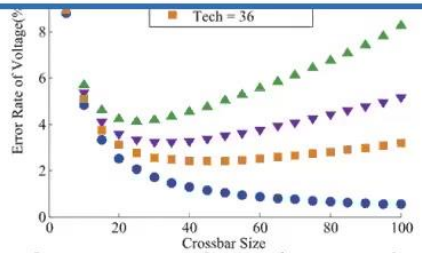




How to evaluate the computation accuracy, area, power, and latency and choose an optimal design?



IR-drop and other non-ideal factors in computing:
Crossbar size needs optimization



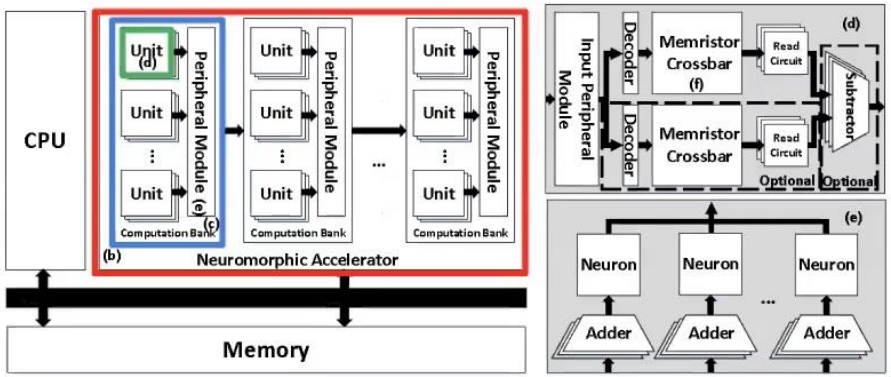
Performance of single crossbar is already influenced by multiple factors

Algorithm hyperparameter space is huge

Ref: Lixue Xia, Boxun Li, Tianqi Tang, Peng Gu, Pai-yu Chen, Shimeng Yu, Yu Cao, Yu Wang, Yuan Xie, Huazhong Yang, MNSIM: Simulation Platform for Memristor-based Neuromorphic Computing System , in IEEE TCAD 2018.

Our Solution: MNSIM 1.0

- In 2016, we proposed MNSIM 1.0 to evaluate the memristor crossbar-based computing systems [DATE'16]
  - Hierarchical structure for large network
  - Estimation model of crossbar and peripheral circuits
  - Estimation model of crossbar computing accuracy



Hierarchical Structure

Crossbar Size	16	32	64	128	256
SPICE(s)	5.35	13.76	41.62	169.12	678.2
MNSIM(s)	0.0007	0.0011	0.0030	0.0192	0.0348
Speed-Up	7642×	12509×	13873×	8088×	19489×

Crossbar Size	256	128	**64**	32	16	8
Error Rate(%)	7.71	2.07	**1.09**	1.46	2.38	3.50
Area(mm^2)	29.34	58.59	117.11	234.10	468.32	936.81
Energy(uJ)	3.74	5.94	10.35	19.21	37.09	73.38

Simulation Results

Lixue Xia, Boxun Li, Tianqi Tang, Peng Gu, Xiling Yin, Wenqin Huangfu, Pai-Yu Chen, Shimeng Yu, Yu Cao, Yu Wang, Yuan Xie and Huazhong Yang, MNSIM: Simulation Platform for Memristor-based Neuromorphic Computing System , in DATE, 2016, pp.469-474.

Our Solution: MNSIM 1.0

- In 2016, we proposed MNSIM 1.0 to evaluate the memristor crossbar-based computing systems [DATE'16]
  - Hierarchical structure for large network → Lack of support for more complex network structure of CNN
  - Estimation model of crossbar and peripheral circuits → Does not support the simulation of other circuits (e.g., buffer)
  - Estimation model of crossbar computing accuracy → Can not evaluate the accuracy of the entire network

Lixue Xia, Boxun Li, Tianqi Tang, Peng Gu, Xiling Yin, Wenqin Huangfu, Pai-Yu Chen, Shimeng Yu, Yu Cao, Yu Wang, Yuan Xie and Huazhong Yang, MNSIM: Simulation Platform for Memristor-based Neuromorphic Computing System , in DATE, 2016, pp.469-474.

Our Solution: MNSIM 2.0

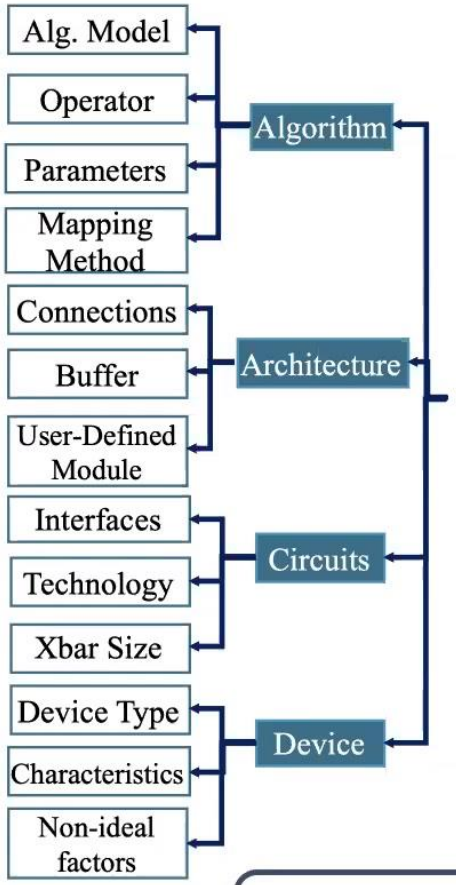
- To overcome these problems, we propose MNSIM2.0 with the following characteristics:
  - Behavior level modeling: **Fast** and suitable for design space exploration
  - **Simulates the hardware behavior** to estimate CNN computing accuracy
  - Supports **user-defined** PIM architecture and various CNN models
  - Contains performance modeling flow and CNN model optimization flow

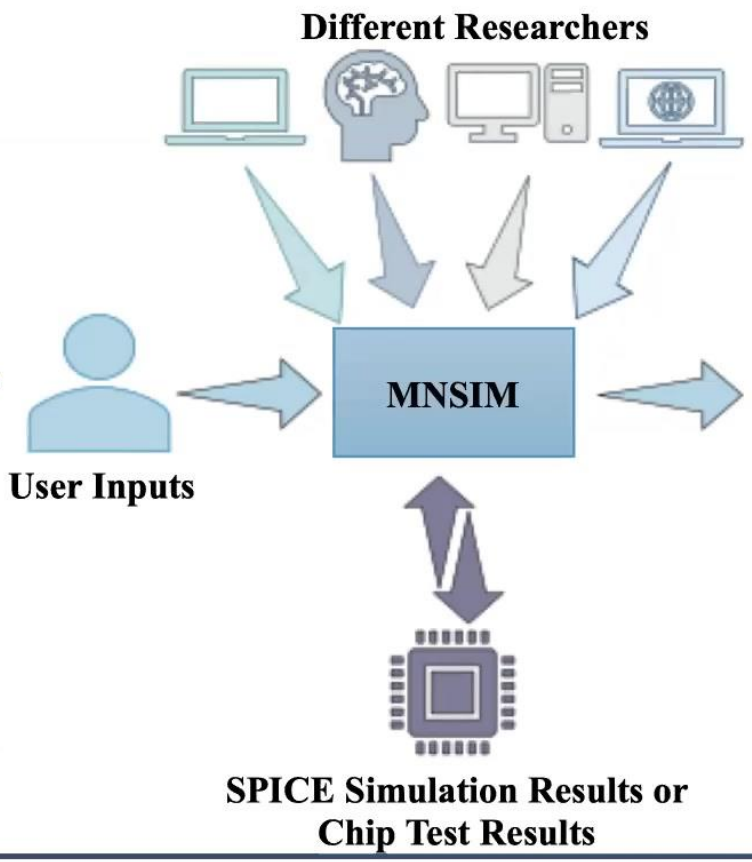
✓ Open Source: https://github.com/thu-nics/MNSIM-2.0.git

Outline

- Background and Introduction
- Overview of MNSIM 2.0
- Hardware Performance Modeling & Evaluation Flow
- CNN Model Optimization Flow
- Conclusion and Future Work

MNSIM 2.0: Behavior-Level Modeling Tool for





Output Results

- Algorithm calculation accuracy
- Hardware performance and costs:
  - ✓ Area
  - ✓ Power & Energy consumption
  - ✓ Latency
  - ✓ …
- Resource utilization
- Security assessment

The α-version code is available in:
https://github.com/thu-nics/MNSIM-2.0.git

MNSIM 2.0: Behavior-Level Modeling Tool for

- Algorithm
  - Alg. Model
  - Operator
  - Parameters
  - Mapping Method
  - Connections
- Buffer
- User-Defined Module
- Interfaces
- Technology
- Xbar Size
- Device
  - Device Type
  - Characteristics
  - Non-ideal factors

Different Researchers

Output Results

- Algorithm calculation accuracy
- Hardware performance and costs:
- Security assessment

Our Target:

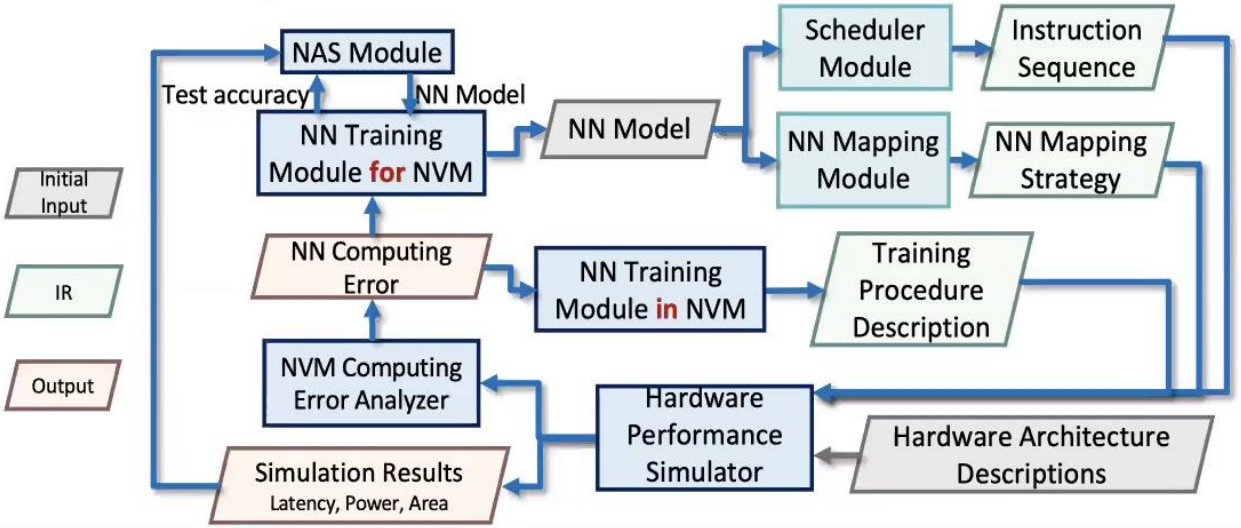
Open, General, Easy-to-Use/Extend, Efficient, and Accurate

SPICE Simulation Results or Chip Test Results

The α-version code is available in:
https://github.com/thu-nics/MNSIM-2.0.git

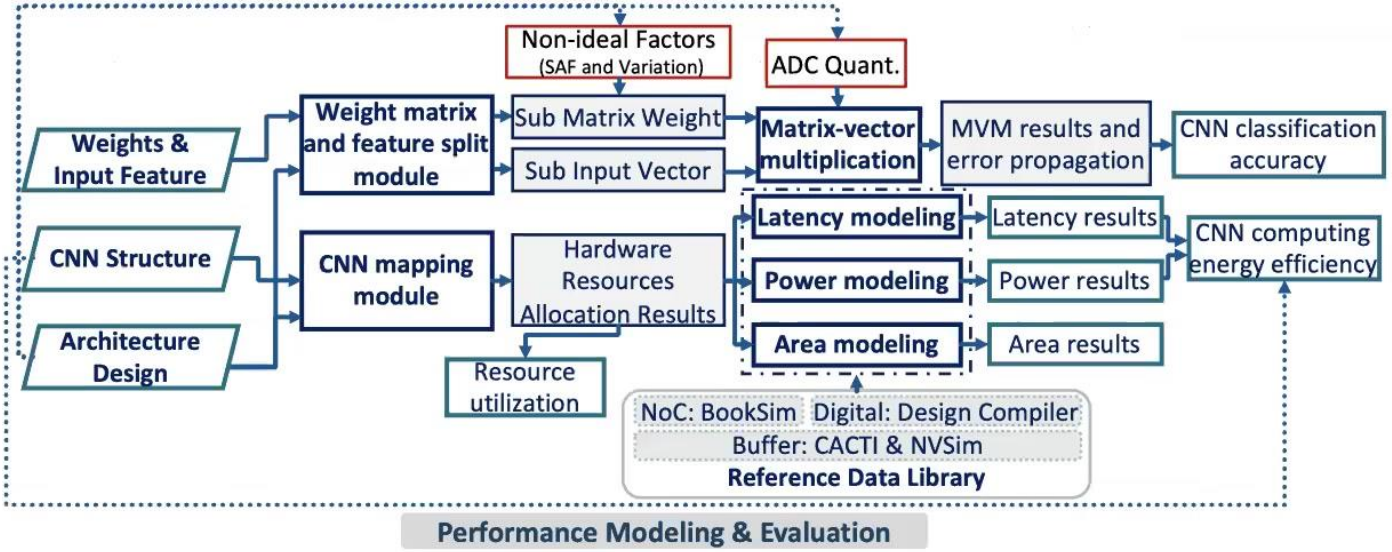
Overview of MNSIM 2.0

- We build a simulator to evaluate PIM-based NN computing performance
  - Hardware performance modeling: Simulate the **hardware performance** of PIM
  - PIM computing error analyzer: Simulate NN classification **accuracy** according to characteristics of devices and peripheral circuits
  - NN training module ***for*** PIM: In **software** level, design an NN training flow considering NVM characteristics
  - NN training modeling ***in*** PIM (In development): In **hardware** level, design an PIM-based online training architecture and corresponding data flow



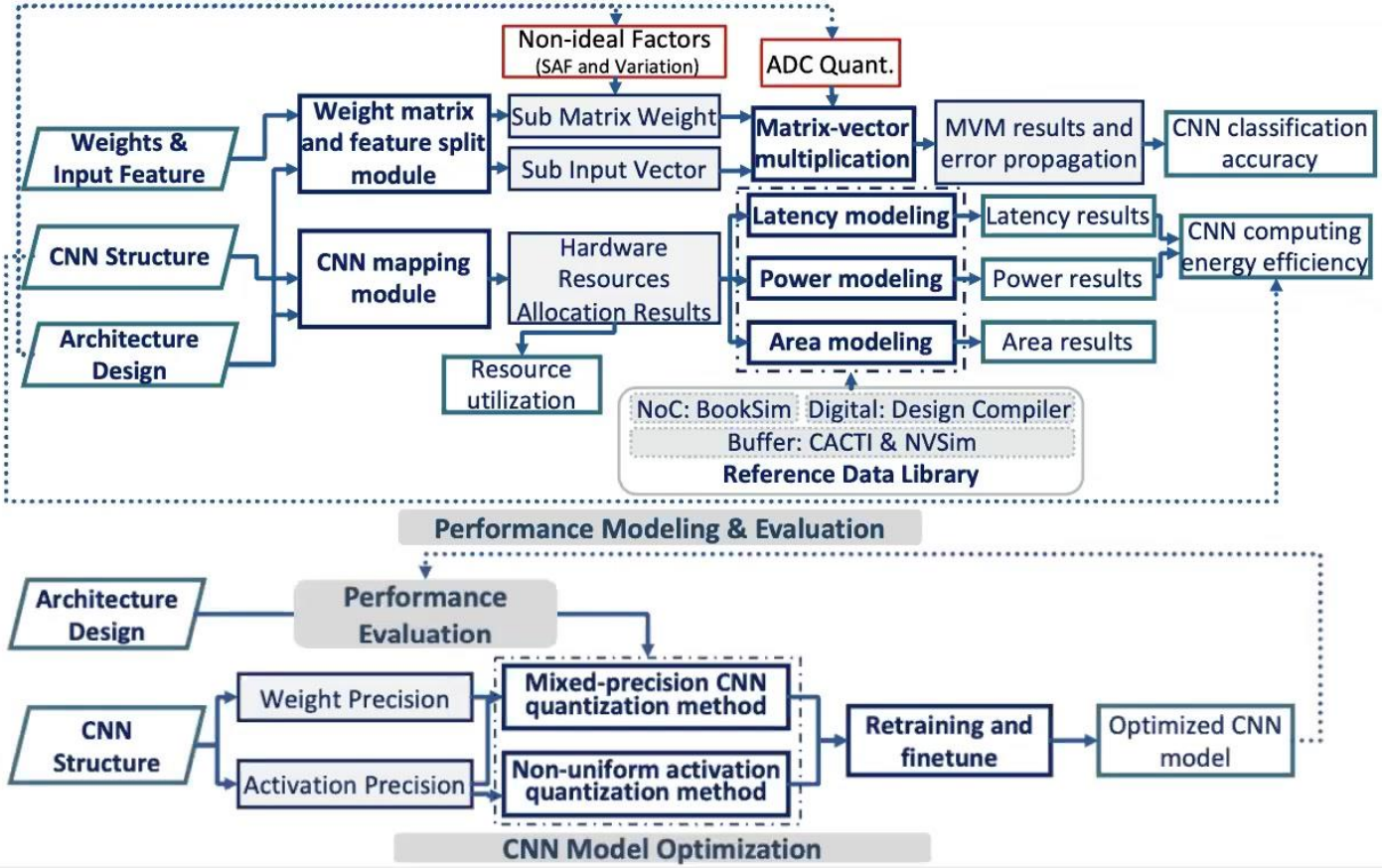
Entire Tool Flow of MNSIM 2.0

- MNSIM contains two parts: **Hardware performance modeling & evaluation flow** and CNN model optimization flow



Entire Tool Flow of MNSIM 2.0

- MNSIM contains two parts: Hardware performance modeling & evaluation flow and **CNN model optimization flow**



Outline

- Background and Introduction
- Overview of MNSIM 2.0
- Hardware Performance Modeling & Evaluation Flow
- CNN Model Optimization Flow
- Conclusion and Future Work

Hardware Performance Modeling Flow

Traditional Simulation Flow

- Accurate but **not suitable** for design space exploration and iterative optimization in the early design stage
- Not easy to use for algorithm researchers

Arch Design → HDL Code & Netlist → SPICE Simulation & Pre-Synthesis (**Time Consuming**) → Arch Update → HDL Code & Netlist → Re-simulation & Synthesis → .....

Hardware Performance Modeling Flow

Traditional Simulation Flow

Arch Design → HDL Code & Netlist → SPICE Simulation & Pre-Synthesis (**Time Consuming**) → Arch Update → HDL Code & Netlist → Re-simulation & Synthesis → .....

MNSIM Modeling Flow

- **Efficient** and **fair comparison** for design space exploration
- Designed for researchers from different areas (algorithm, arch, circuits, and device)

SPICE Simulation & Pre-Synthesis for **Basic Units** (Time Consuming) → Arch Description → Modeling → Arch Update → Modeling → Arch Update → Modeling → .....

Hardware Performance Modeling Flow

MNSIM Modeling Flow

Arch Description → Arch Update → Arch Update → .....

SPICE Simulation & Pre-Synthesis for Basic Units (Time Consuming)

Modeling → Modeling → Modeling

Basic Architecture Design

CNN Model

User Defined Arch

Algorithm Mapping

Unit Performance Look-Up-Table

Data Flow Construction

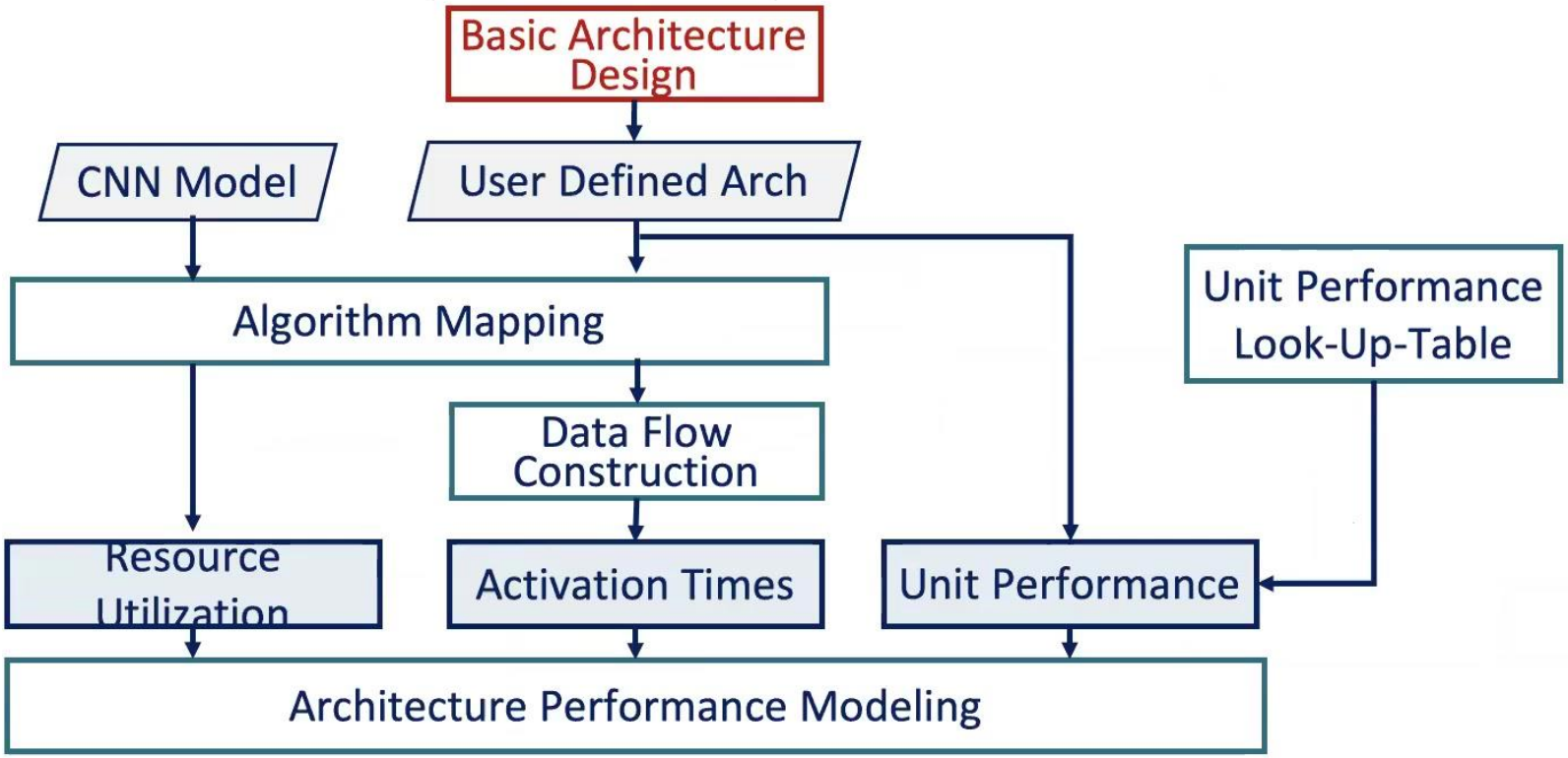
Resource Utilization

Activation Times

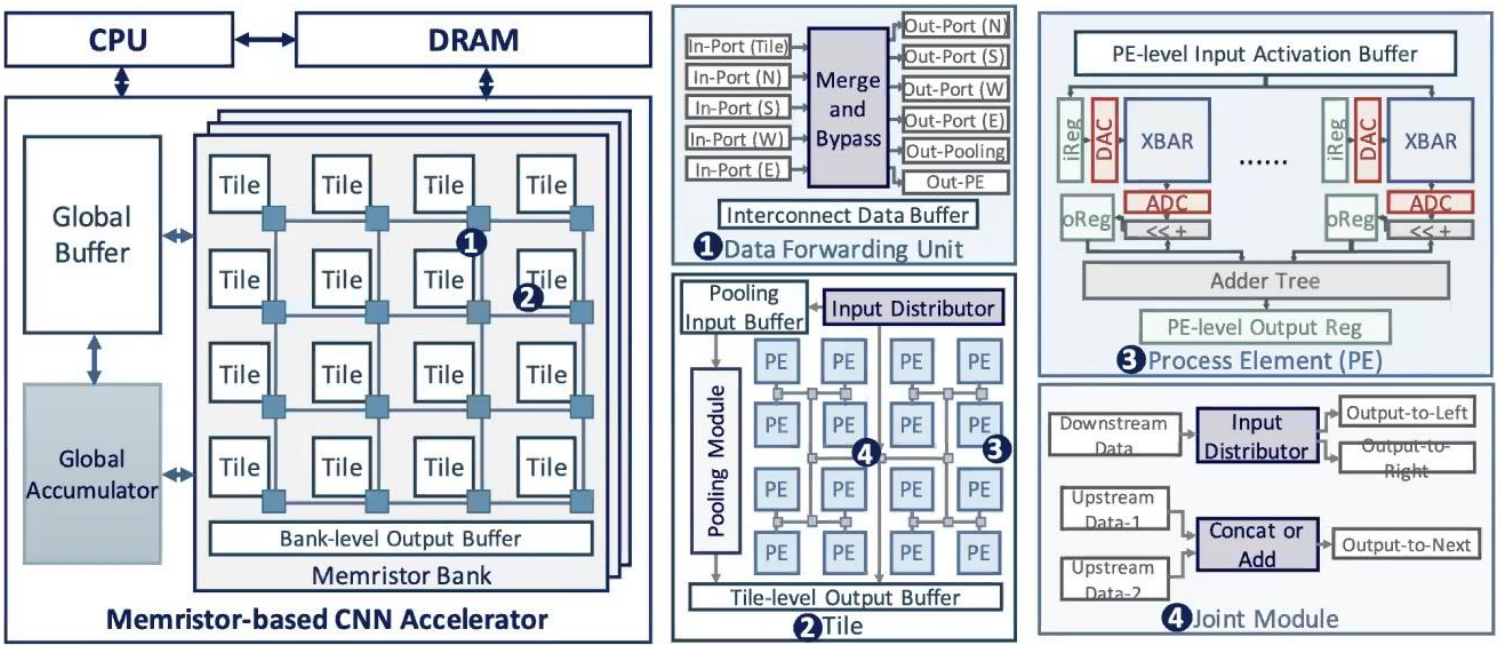
Unit Performance

Architecture Performance Modeling

Hardware Performance Modeling Flow



Basic Architecture Design in MNSIM



- Multi-Level Abstraction: be able to **describe different architecture designs** (e.g., crossbar number, size, device and interface precision)
- Support **basic NN operations**: conv, fc, pooling, element wise sum
- Support **multi-precision NN** mapping and calculation

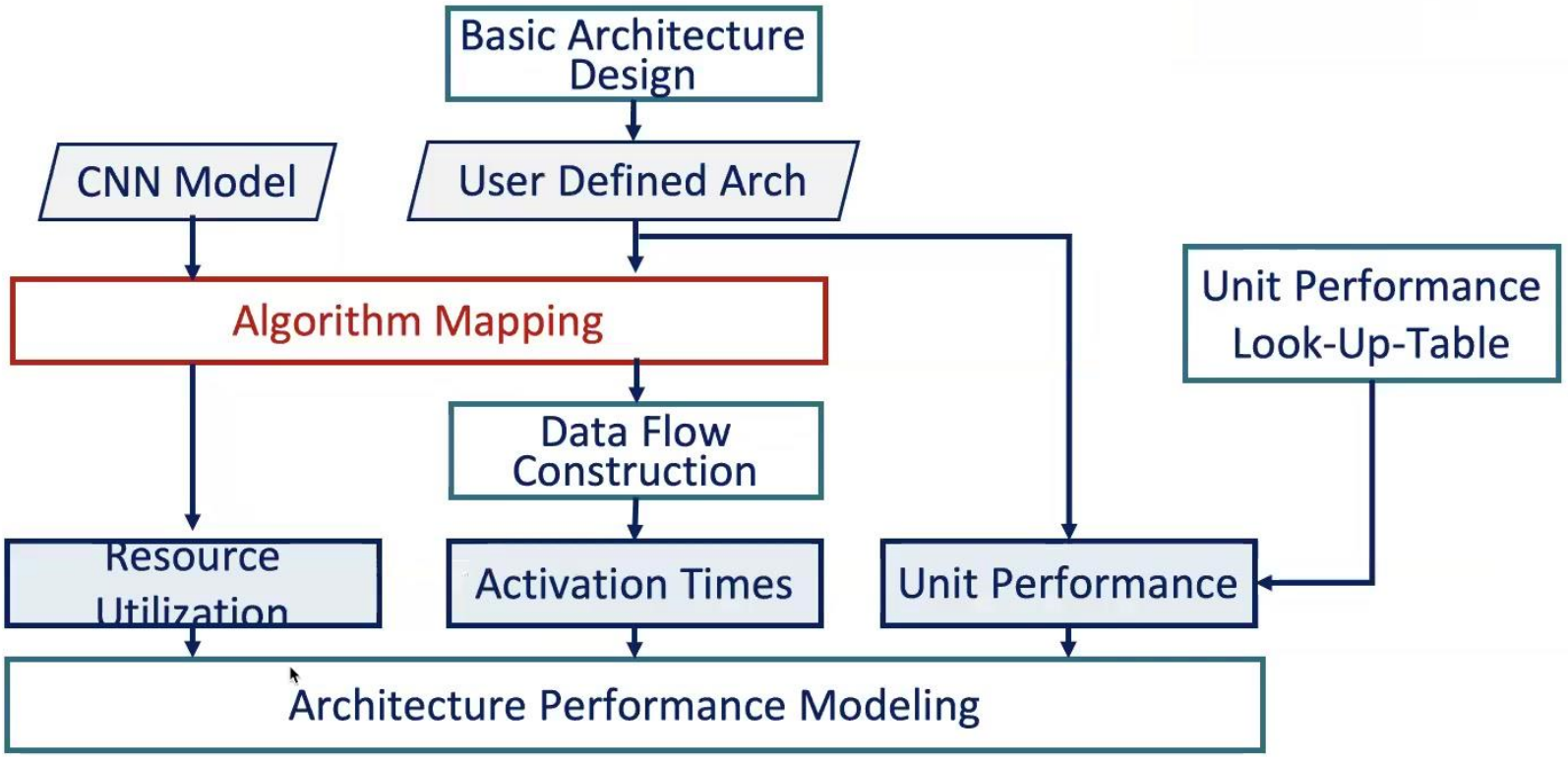
Basic Architecture Design in MNSIM

Device Level	Device technology
	Device area
	Level number of the read voltages
	Amplitudes of the read voltages
	Level number of the write voltages
	Amplitudes of the write voltages
	Read latency
	Write latency
	Device resistance level
	Resistances of each device level
	Device resistance variation
	Probability of the SAFs
Interface Level	DAC & ADC Area
	DAC & ADC Resolution
	DAC & ADC Power
	DAC & ADC Sample Rate

Xbar Level	Crossbar size
	Cell type (0T1R/1T1R)
	Transistor technology
	Wire resistance
	Wire capacity
	Load resistance
	Xbar area calculation method
PE Level	Xbar polarity (pos+neg or one xbar)
	Subtraction position (analog or digital)
	Number of crossbar groups
	DAC number
	ADC number
Tile Level	PE number in one Tile
	Adder tree structure
	Inter tile bandwidth
	Intra tile bandwidth

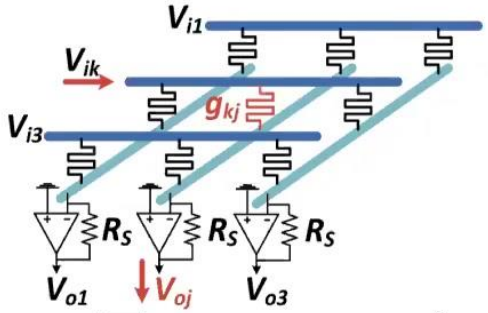
Bank Level	Tile number in one bank
	Inter bank bandwidth
	Intra bank bandwidth
	NoC structure
Digital Level	Transistor technology
	Buffer choice
	Buffer capacity
	Buffer area
	Buffer r/w bitwidth
	Buffer r/w frequency
	Buffer r/w power
	Pooling structure

Hardware Performance Modeling Flow



CNN Model Mapping and Resource Allocatio

- Resource usage calculation:

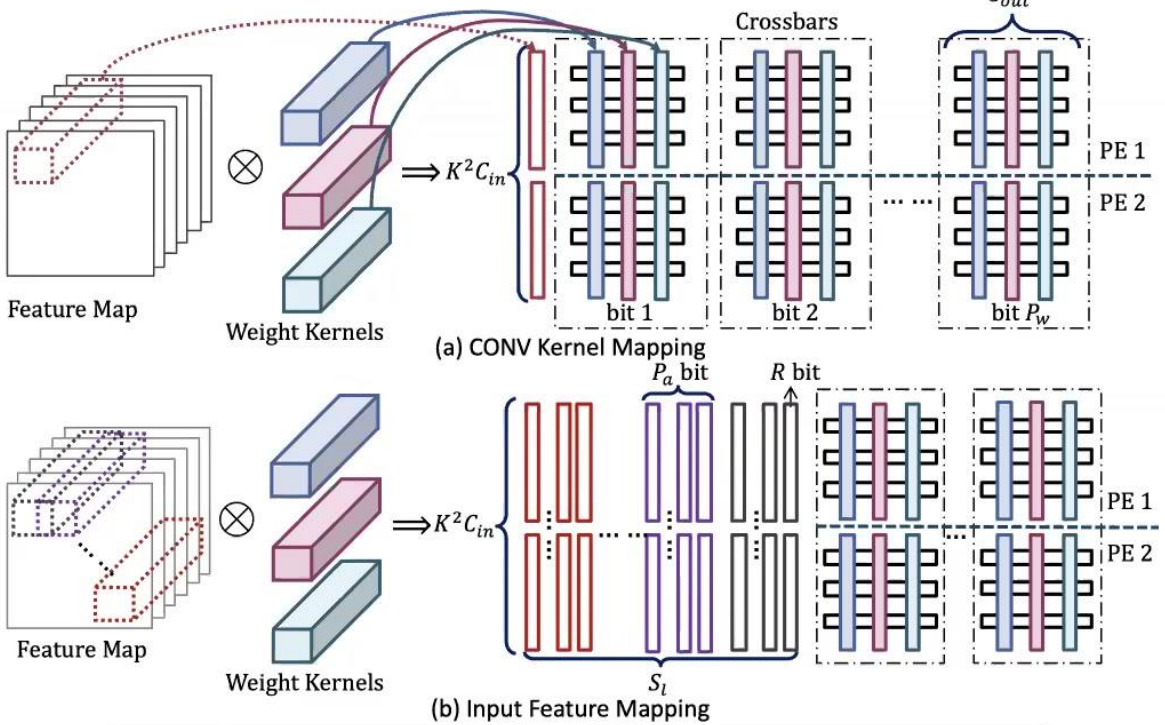


$$V_{oj} = \sum_k V_{ik} \cdot c_{kj}, c_{kj} = -\frac{g_{kj}}{g_s}$$

$$\#Tile_{CONV} = \left\lceil \frac{\left\lceil \frac{C_{in}}{\lfloor xbar_r/K^2 \rfloor} \right\rceil \left\lceil \frac{C_{out}}{xbar_c} \right\rceil \left\lceil \frac{P_w}{\#xbar/PE} \right\rceil}{\#PE/Tile} \right\rceil$$

$$\#Tile_{FC} = \left\lceil \frac{\left\lceil \frac{Length_{in}}{xbar_r} \right\rceil \left\lceil \frac{Length_{out}}{xbar_c} \right\rceil \left\lceil \frac{P_w}{\#xbar/PE} \right\rceil}{\#PE/Tile} \right\rceil$$

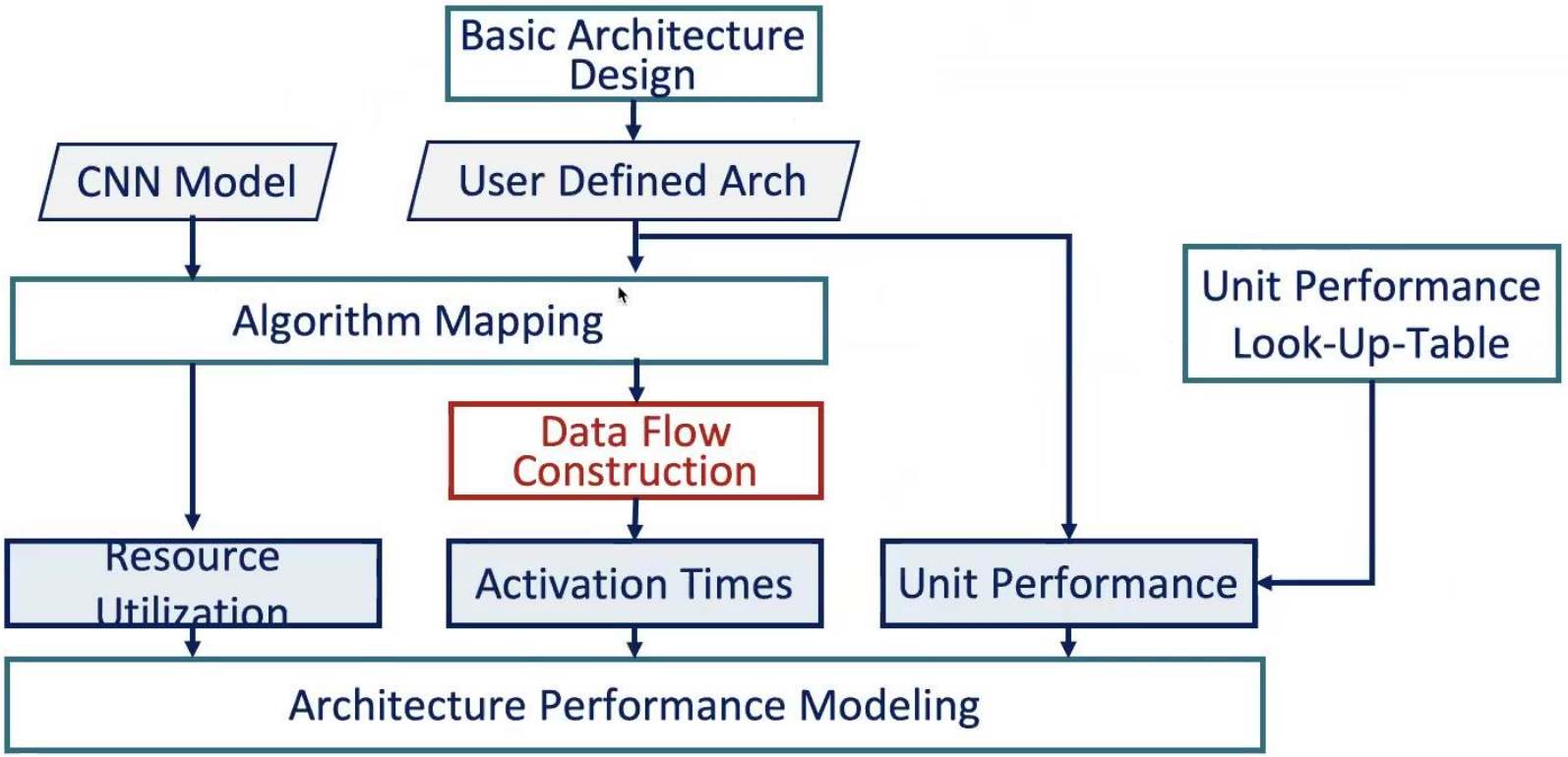
$$\#Tile_{Pooling} = \left\lceil \frac{k^2 \times C_{in}}{Pooling\ Buffer\ Size/Tile} \right\rceil$$



K: Kernel Size C_{in}: #Input Channel C_{out}: #Output Channel
S_l: Sliding Times P_w: Weight Bitwidth P_a: Input Bitwidth R: DAC resolution

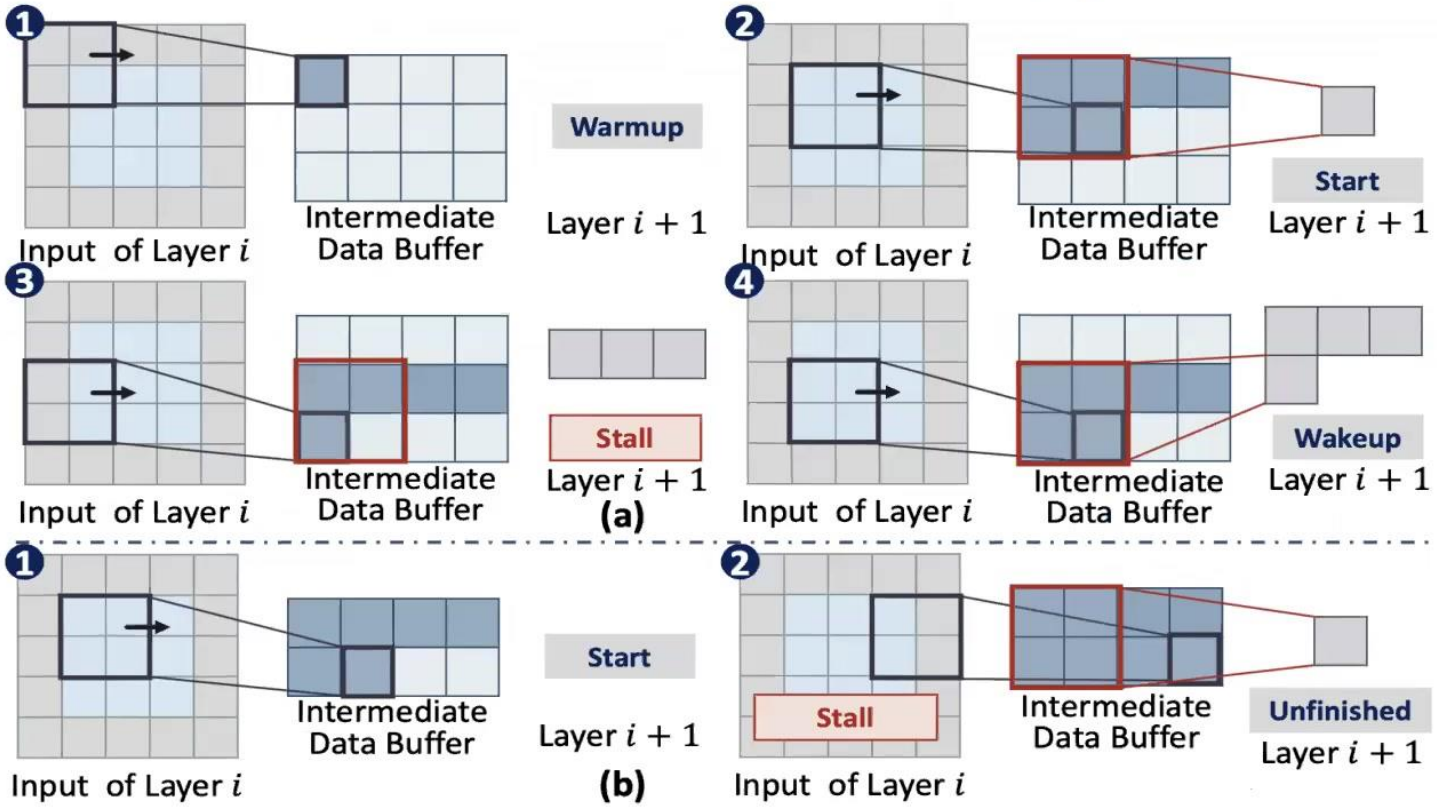
Map **high bit width** algorithms to **low precision** devices and interfaces

Hardware Performance Modeling Flow



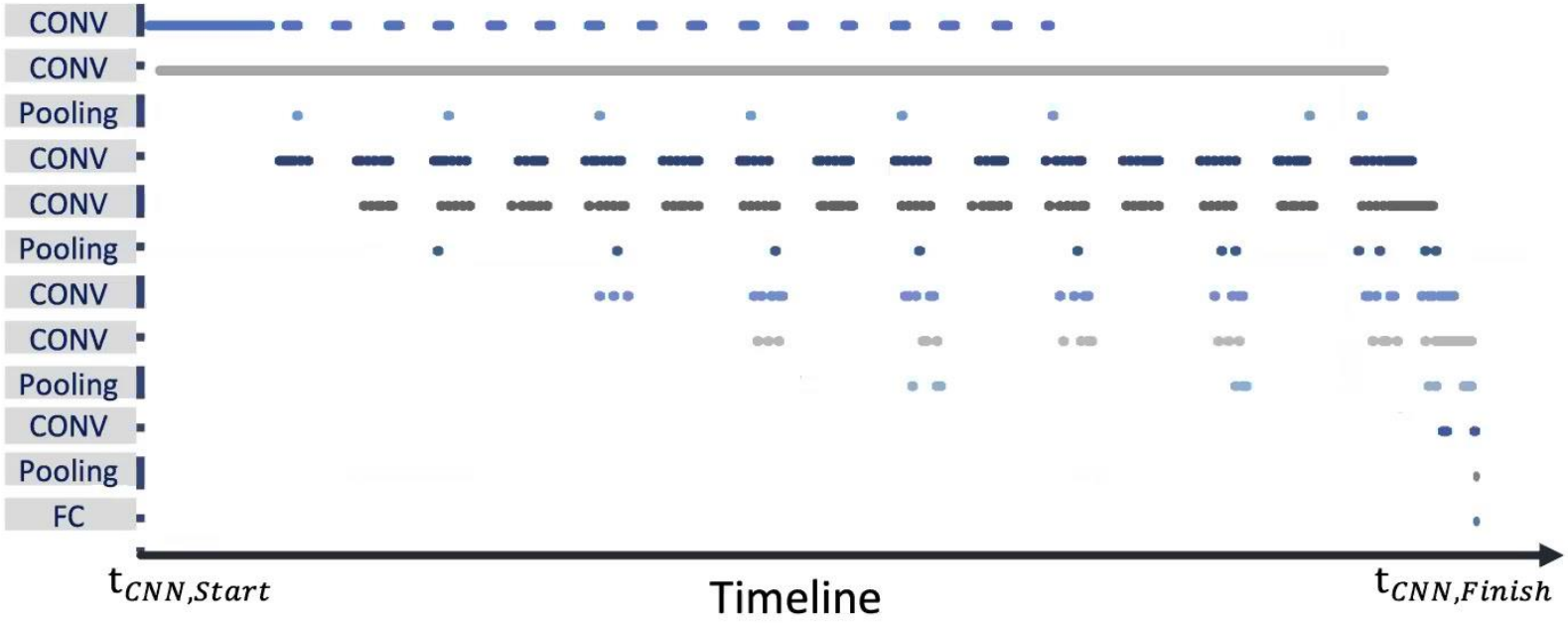
Data Flow Construction

- Data flow among different layers: inner-layer **pipeline** structure



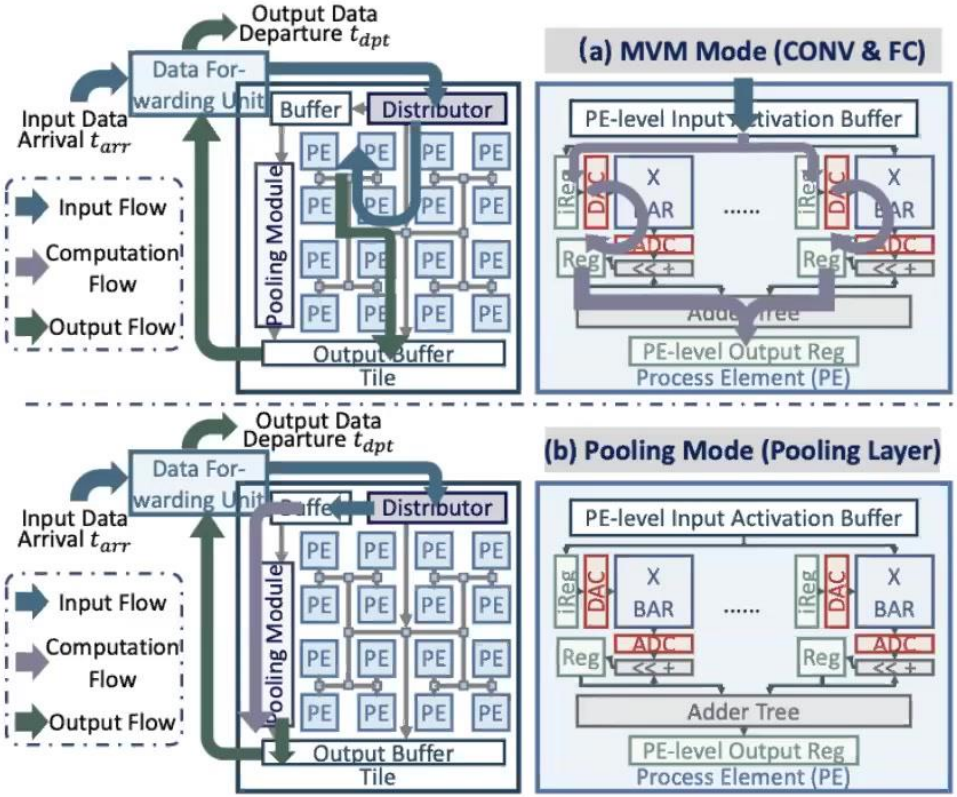
Data Flow Construction

- Data flow among different layers: inner-layer **pipeline** structure



Data Flow Construction

- Data flow in one layer

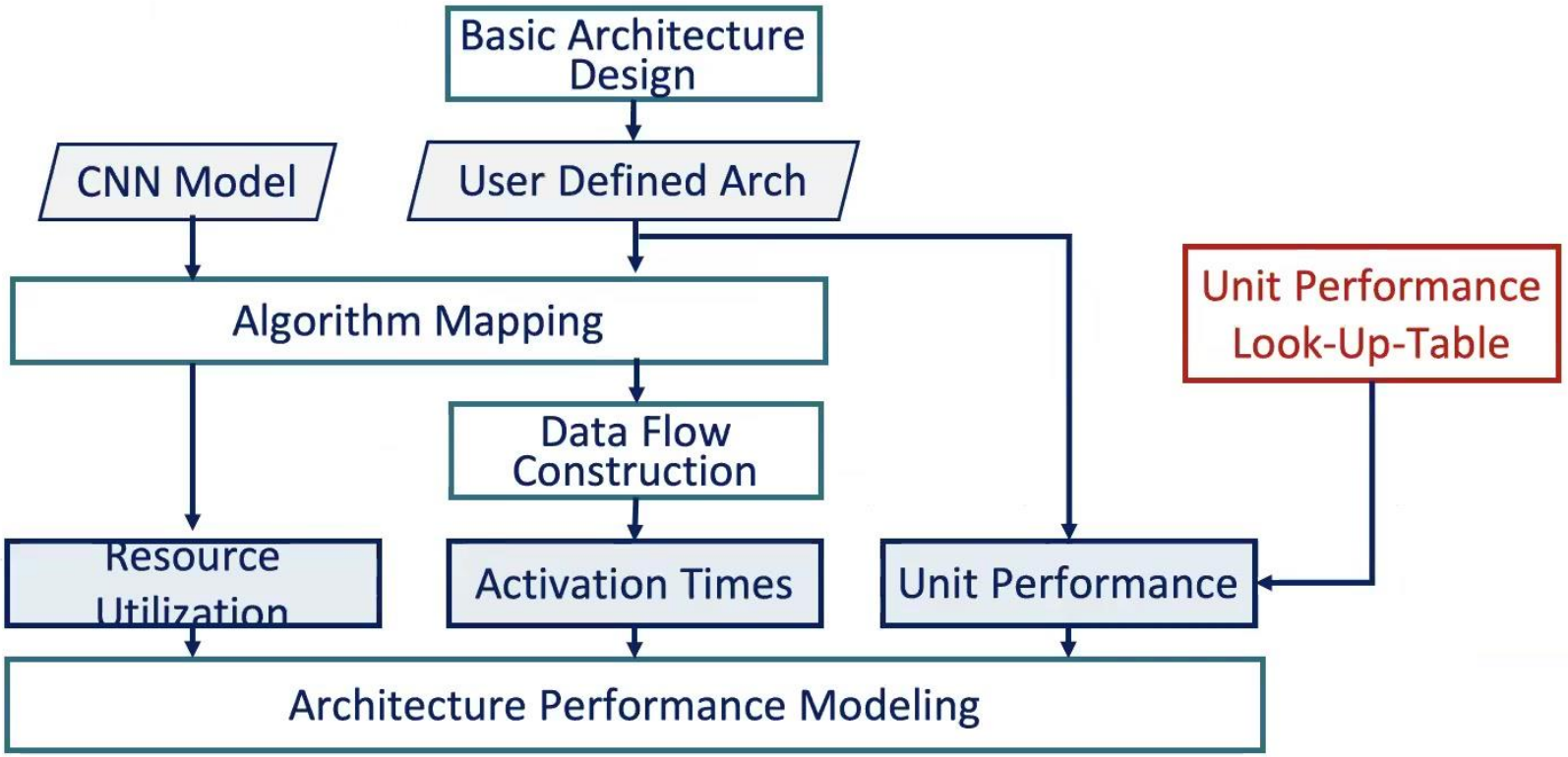


Due to the limited DAC resolution and WL/BL parallelism, multiple cycles are needed to complete the calculation of the input data stored in the iReg:

$$mul = \left\lceil \frac{P_a}{Res_{DAC}} \right\rceil \left\lceil \frac{k^2 \min(\lfloor \frac{xbar_r}{k^2} \rfloor, C_{in})}{Parallel_r} \right\rceil \left\lceil \frac{\min(C_{out}, xbar_c)}{Parallel_c} \right\rceil$$

$$T_{cf} = t_{iReg} + mul(t_{DeMUX} + t_{DAC} + t_{xbar} + t_{ADC} + t_{MUX} + t_{S\&A} + t_{oReg}) + t_{adder-tree}$$

Hardware Performance Modeling Flow



Area, Power, and Latency Models

- NVM Crossbar
  - Area calculation:

$$\text{Xbar\_Area} = \begin{cases} User's\ given\ value \\ xbar_c \times xbar_r \times Device\_Area \\ xbar_c \times xbar_r \times 3(W/L+1)F^2\ (1T1R) \\ xbar_c \times xbar_r \times 4F^2\ (0T1R) \end{cases}$$

  - Power calculation:

$$\text{Xbar\_Power} = \begin{cases} User's\ given\ value \\ xbar_c \times xbar_r \times U\% \times Device\_Power \\ xbar_c \times xbar_r \times U\% \times V_{eq}^2/R_{eq}\ (1T1R) \\ xbar_c \times xbar_r \times (U\% + (1-U\%)\beta) \times V_{eq}^2/R_{eq}\ (0T1R) \end{cases}$$

  - Latency calculation:

$$\text{Xbar\_Latency} = \begin{cases} User's\ given\ value \\ Device\_Latency + RC\_Delay \end{cases}$$

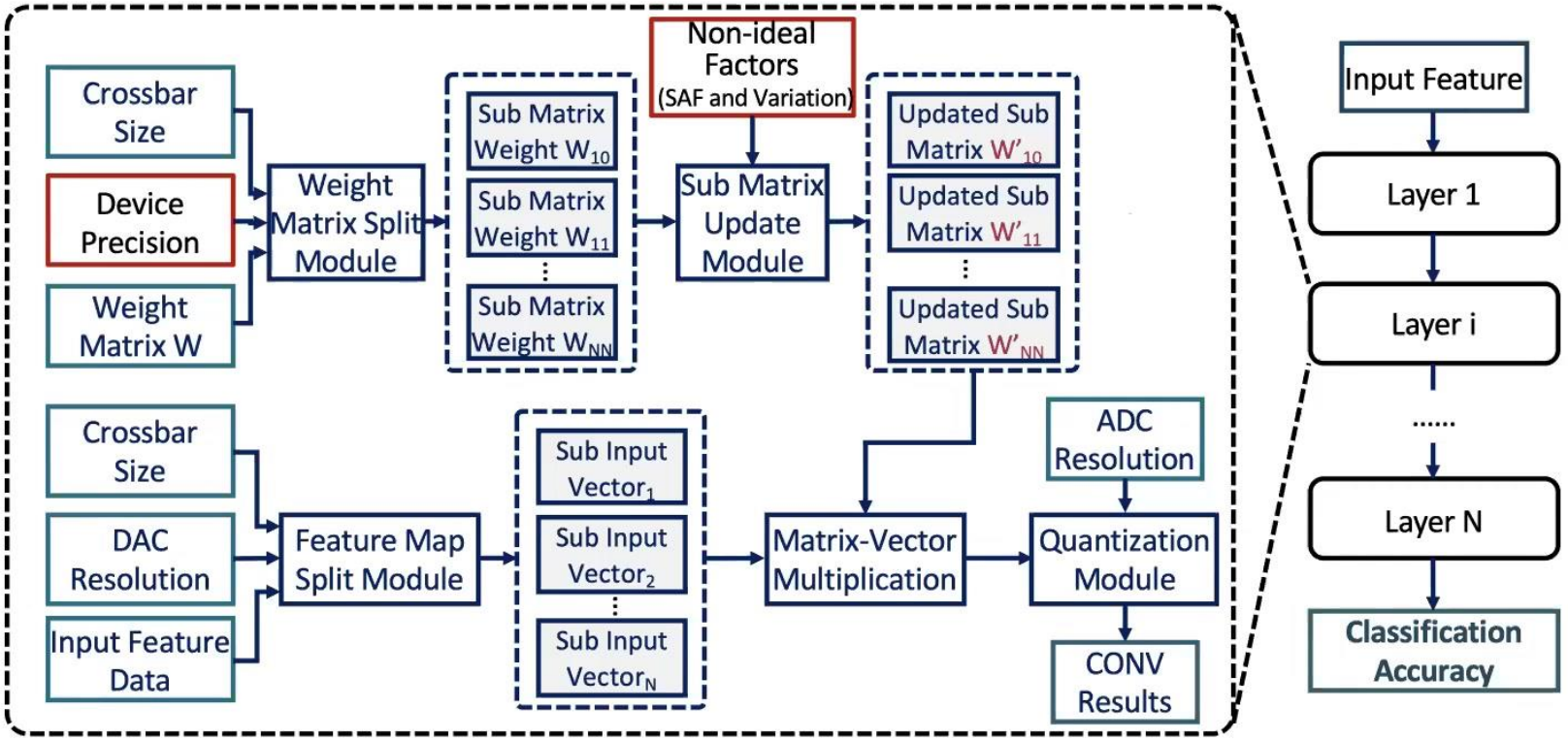
  - Use Stanford RRAM model and HSPICE for revision and verification

Area, Power, and Latency Models

- ADC/DAC:
  - Mainly customized by users
  - Include some state-of-the-art DAC/ADC designs with different resolutions as default values
- Digital Circuits:
  - Synthesize the digital circuit modules at TSMC 65nm technology node with Synopsys® Design Compiler
  - For other technology nodes, use scaling down estimation method
- Buffer Design:
  - Use CACTI and NVSIM to get the reference data
  - Use the method of fitting and looking up table through reference data to estimate the buffer read-write overhead
- NoC Simulation:
  - Use BookSim and tile-connection graph model for the inter-tile NoC simulation

Accuracy Estimation Flow

- The accuracy estimation flow simulates **the calculation behavior** of actual PIM hardware: *data splitting, quantization, non-ideal factors*
  - Current version: **PyTorch** interface, TF & Caffe will be supported in the future

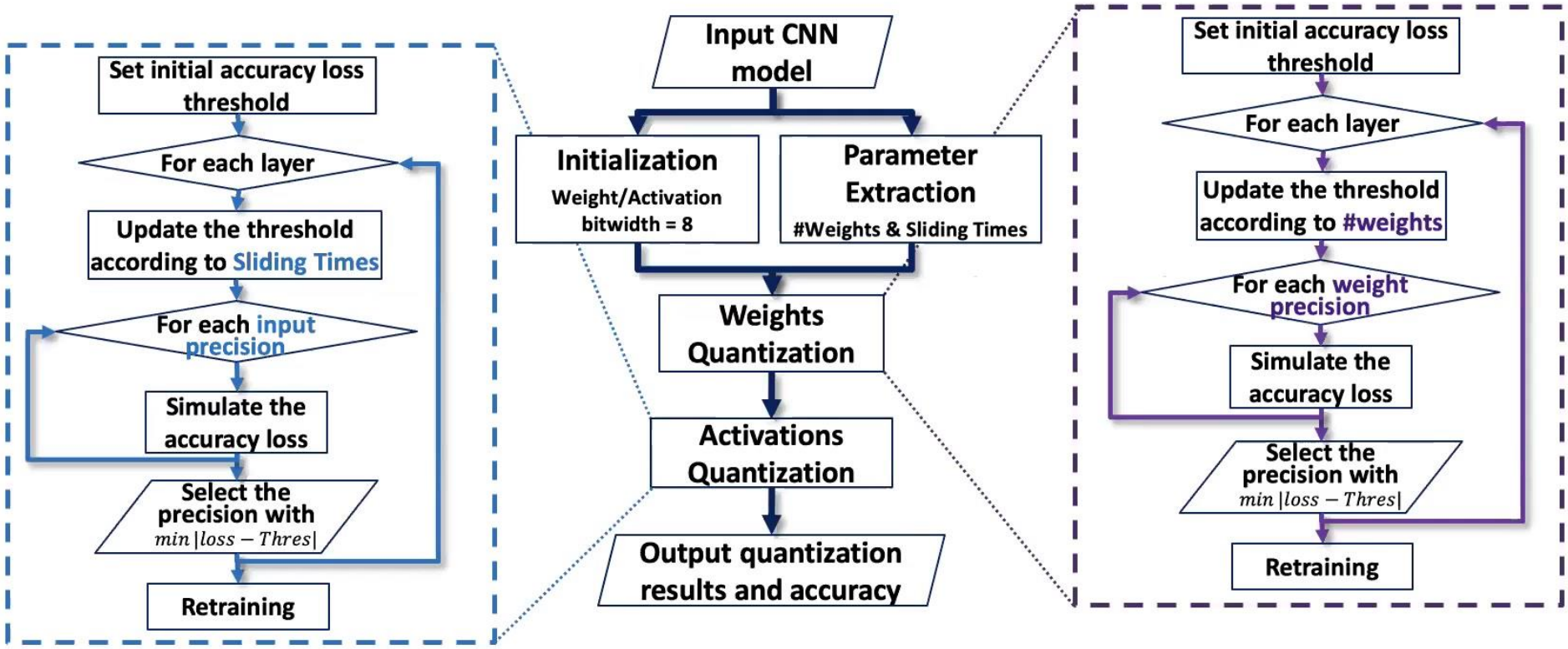


Outline

- Background and Introduction
- Overview of MNSIM 2.0
- Hardware Performance Modeling & Evaluation Flow
- CNN Model Optimization Flow
- Conclusion and Future Work

Hardware-Aware CNN Optimization

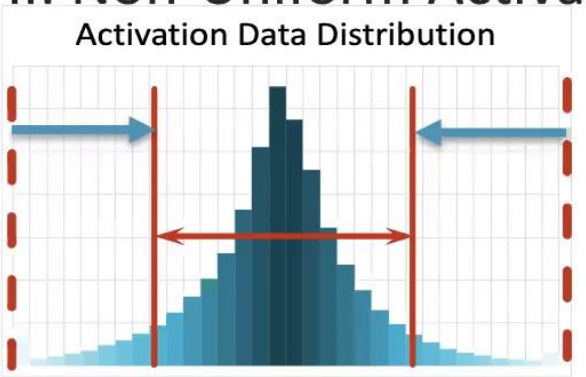
- I: Mixed Precision CNN Quantization [DAC'19]



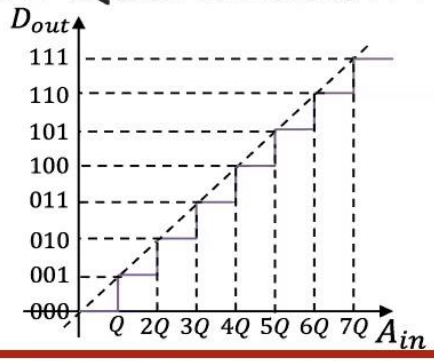
Zhenhua Zhu, Hanbo Sun, Yujun Lin, Guohao Dai, Lixue Xia, Song Han, Yu Wang, Huazhong Yang, A Configurable Multi-Precision CNN Computing Framework Based on Single Bit RRAM , in Design Automation Conference (DAC), 2019, pp.56.

Hardware-Aware CNN Optimization

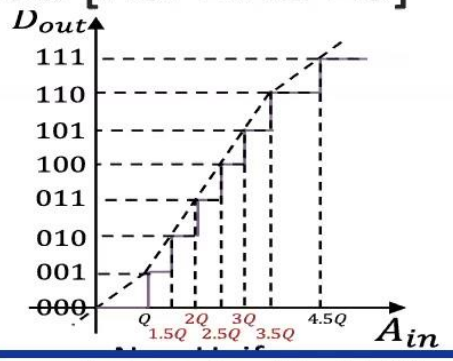
- II: Non-Uniform Activation Quantization Method [ASPDAC'20]



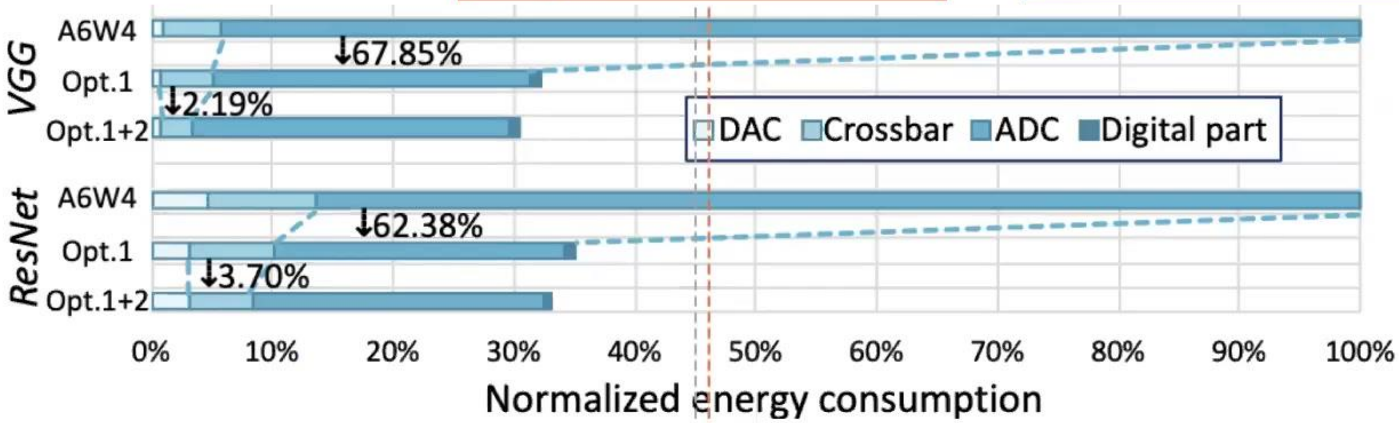
Optimized Quantization Range
$[-|\mu| - 3\sigma, |\mu| + 3\sigma]$



3 steps for [Q, 3.5Q]
@ Uniform Quantization
(**High** quantization error)



5 steps for [Q, 3.5Q]
@ Non-uniform Quantization
(**Low** quantization error)



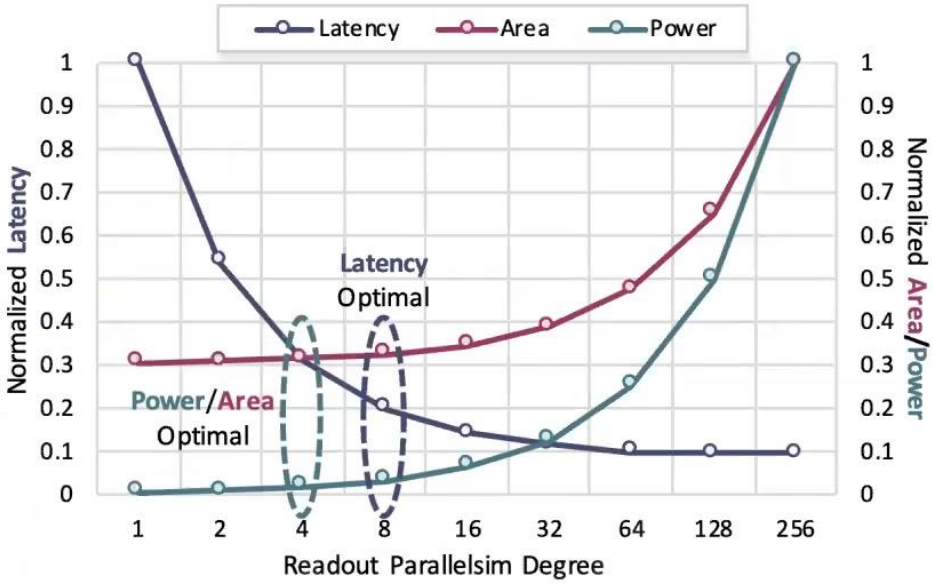
Hanbo Sun, Zhenhua Zhu, Yi Cai, Xiaoming Chen, Yu Wang, Huazhong Yang, An Energy-Efficient Quantized and Regularized Training Framework For Processing-In-Memory Accelerators , to appear in The 25th Asia and South Pacific Design Automation Conference(ASP-DAC 2020), 2020.

Case Study

- We use MNSIM to evaluate different architecture designs and NN models
  - Each simulation time < 5s
  - Three NN models are used: LeNet, VGG-8, and ResNet18 @ Cifar10
  - Different ADC resolutions and readout parallelism degrees are evaluated

	LeNet			VGG-8			ResNet18		
R	L	P	A	L	P	A	L	P	A
B	-	-	76.27	-	-	91.64	-	-	91.18
4	0.26	0.34	10.81	10.45	9.67	10.81	2.82	11.97	10.81
6	0.28	0.54	49.45	11.56	15.44	53.91	3.06	19.33	10.81
8	0.29	1.55	72.72	12.32	43.67	90.18	3.22	55.32	87.81
10	0.28	2.62	76.63	12.29	73.76	91.45	3.20	93.68	88.91

Latency (L, ms), power (P, W), and accuracy (A, %) under different ADC resolutions (R). B is algorithm baseline.



Outline

- Background and Introduction
- Overview of MNSIM 2.0
- Hardware Performance Modeling & Evaluation Flow
- CNN Model Optimization Flow
- Conclusion and Future Work

Conclusion

- Challenge in PIM systems
  - **Unacceptable long simulation time** for design space exploration
- Our solutions
  - A behavior-level modeling tool -- MNSIM 2.0
    - Hardware performance modeling & evaluation flow + CNN model optimization flow
    - Each evaluation time is less than 5 seconds → **suitable for design space exploration**
- Future work
  - We hope to build and develop MNSIM as an open simulation platform
  - We sincerely invite every Processing-In-Memory researcher to add their ideas to MNSIM to enlarge its function
  - We seek chip designer partners to perform system-level verification for MNSIM modeling results
  - https://github.com/thu-nics/MNSIM-2.0.git

Thanks for Your Attention!