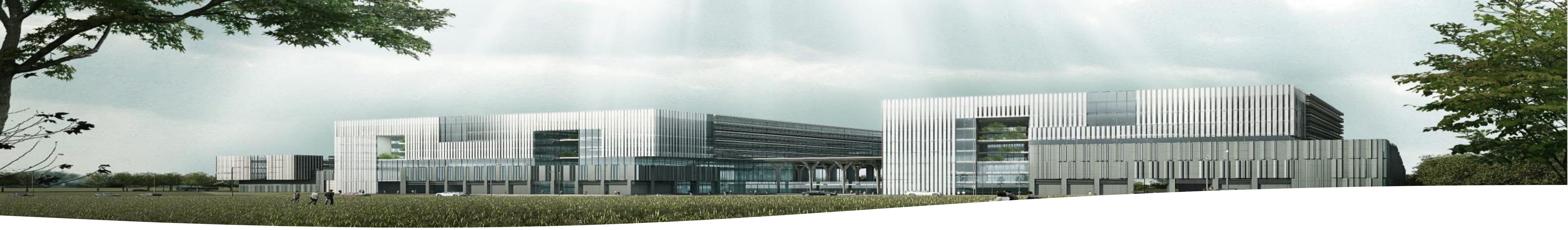


Energy-efficient DNN Hardware Accelerator with On-chip Learning – A Deep Belief Network Processor Case

Jiajun Wu

2021/07/07

華中科技大學 HUAZHONG UNIVERSITY OF SCIENCE AND TECHNOLOGY | 光学与电子信息学院 SCHOOL OF OPTICAL AND ELECTRONIC INFORMATION

Outline

- Introduction
- Overview of DNN Accelerators
- Case Study - Deep Belief Network Processor
- Broaden Vision - Development Trend of DNN Accelerators
- Conclusion
- Q&A

About Me

Jiajun Wu (Student Member of IEEE)

Education
- BSc in IC Design and Integrated Systems, HUST, 2017 – 2021
- Will join the University of Hong Kong for Ph.D. program, in Sept. 2021

Interests
- Hardware Accelerators
- Neuromorphic Computing
- Hardware-software Co-Design

About Me

- Jiajun Wu joined the Research Lab of Ultra Low-Power and Intelligent Integrated Circuits in 2019, supervised by **Prof. Chao Wang**.

- Thanks to Prof. Wang and other lab members, he has published one paper in **IEEE TCAS-I** journal and one paper in **IEEE A-SSCC** conference. He has also submitted one paper which is now under review in **IEEE JETCAS** journal.

Outline

- **Introduction**
- Overview of DNN Accelerators
- Case Study - Deep Belief Network Processor
- Broaden Vision - Development Trend of DNN Accelerators
- Conclusion
- Q&A

Introduction

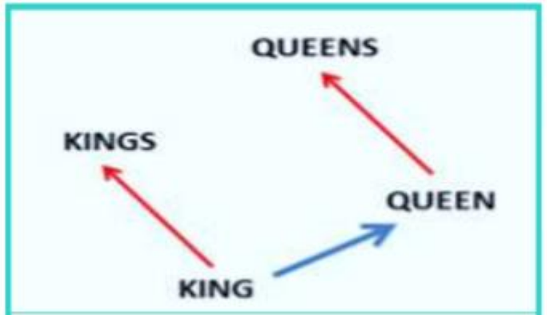
Typical AI Applications

Vision

- Search & information extraction
- Security/Video surveillance
- Self-driving cars
- Medical imaging
- Robotics

Speech

- Interactive voice response (IVR) systems
- Voice interfaces (Mobile, Cars, Gaming, Home)
- Security (speaker identification)
- Health care
- Simultaneous interpretation



Text

- Search and ranking
- Sentiment analysis
- Machine translation
- Question answering

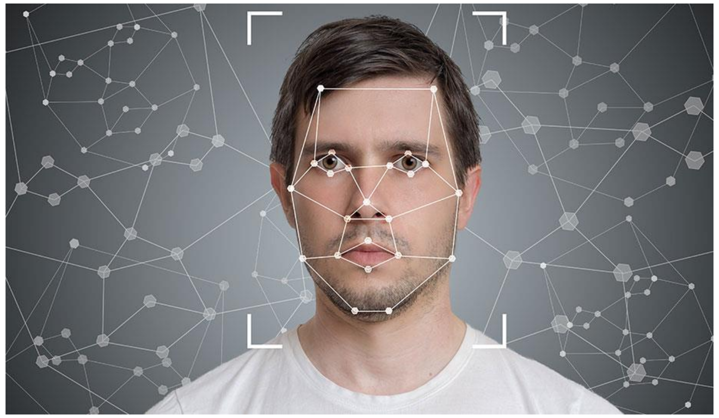
Other

- Recommendation engines
- Advertising
- Fraud detection
- AI challenges
- Drug discovery
- Sensor data analysis
- Diagnostic support

Source: https://www.hpcuserforum.com/presentations/Wisconsin2017/HPDLCookbook4HPCUserForum.pdf

Introduction

Deep Neural Network (DNN) is a critical component in the AI revolution



Face Recognition

Self-driving

Wearable Medical Devices

- **However, it comes at the cost of high computational complexity and energy consumption, limited in Internet-of-thing (IoT) applications.**

- **Besides, on-chip learning is essential for data privacy and reducing communication cost.**

Introduction

Name	**Type**	**Model size** (# params)	**Model size** (MB)	**GFLOPs** (forward pass)
AlexNet	CNN	60,965,224	233 MB	0.7
GoogleNet	CNN	6,998,552	27 MB	1.6
VGG-16	CNN	138,357,544	528 MB	15.5
VGG-19	CNN	143,667,240	548 MB	19.6
ResNet50	CNN	25,610,269	98 MB	3.9
ResNet101	CNN	44,654,608	170 MB	7.6
ResNet152	CNN	60,344,387	230 MB	11.3
Eng Acoustic Model	RNN	34,678,784	132 MB	0.035
TextCNN	CNN	151,690	0.6 MB	0.009

High computational complexity with large model size

Source: https://www.hpcuserforum.com/presentations/Wisconsin2017/HPDLCookbook4HPCUserForum.pdf

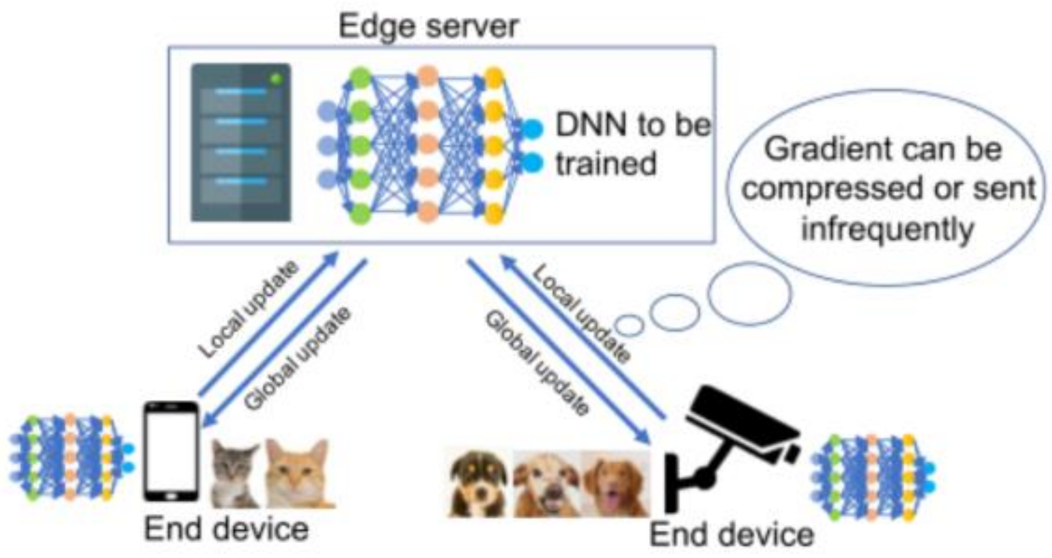
Introduction

Alpha Go
1202 CPUs, 176 GPUs,
~1 MW Power

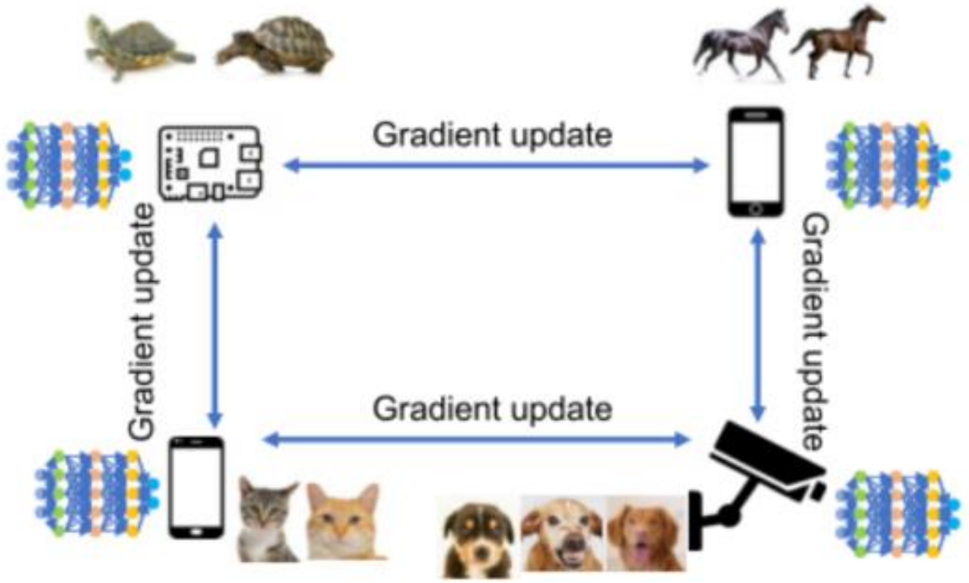
Lee Se-dol
1 Human Brain
~20W Power

Extremely huge energy consumption of DNN model

Introduction



Conventional cloud/server-based learning



On-chip edge learning

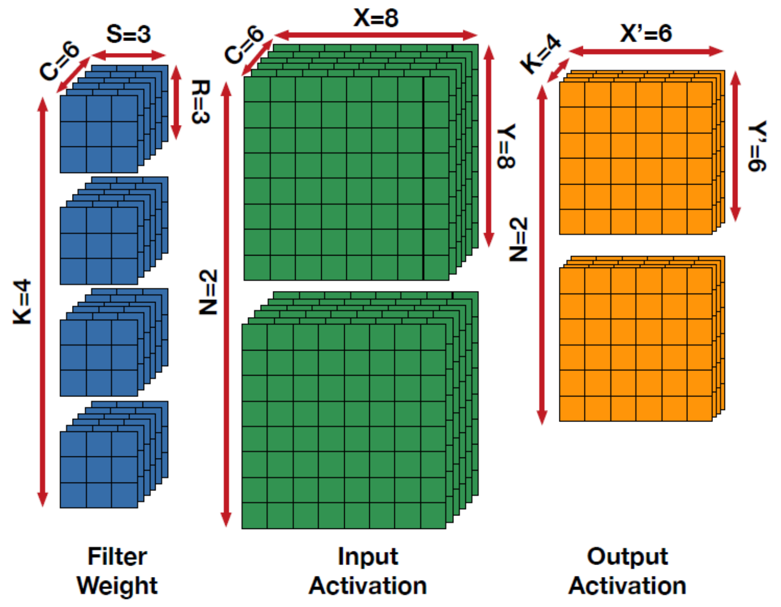
- Protect data privacy
- Reduce energy consumption of communication

Chen, Jiasi, and Xukan Ran. "Deep learning with edge computing: A review." Proceedings of the IEEE 2019: 1655-1674.

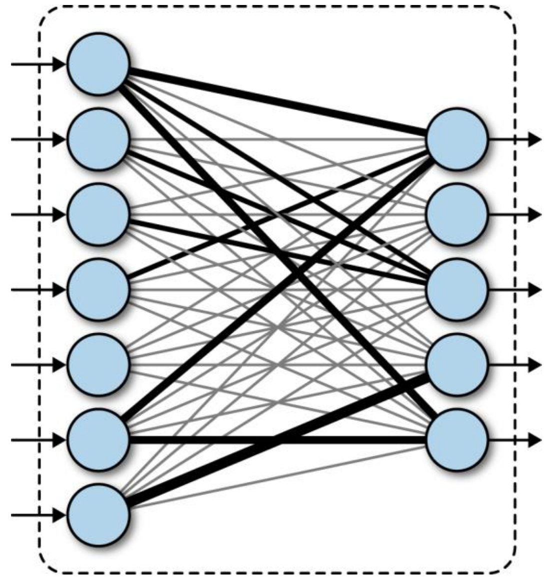
Outline

- Introduction
- **Overview of DNN Accelerators**
- Case Study - Deep Belief Network Processor
- Broaden Vision - Development Trend of DNN Accelerators
- Conclusion
- Q&A

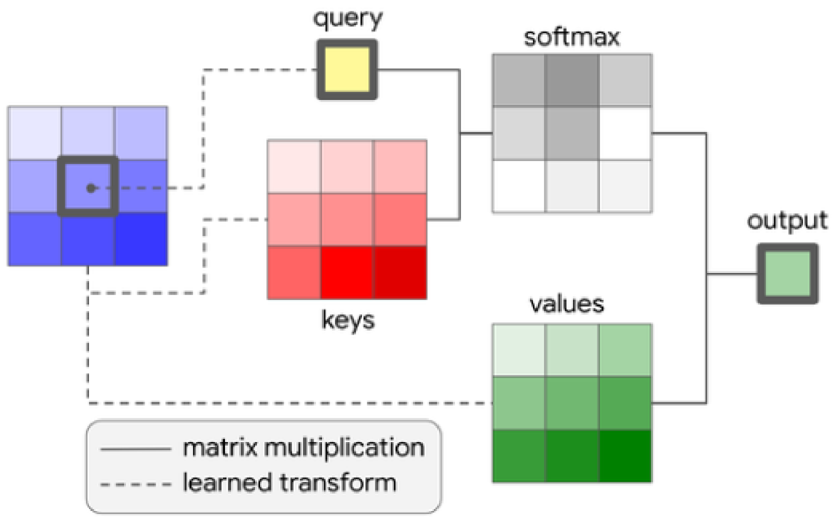
Overview



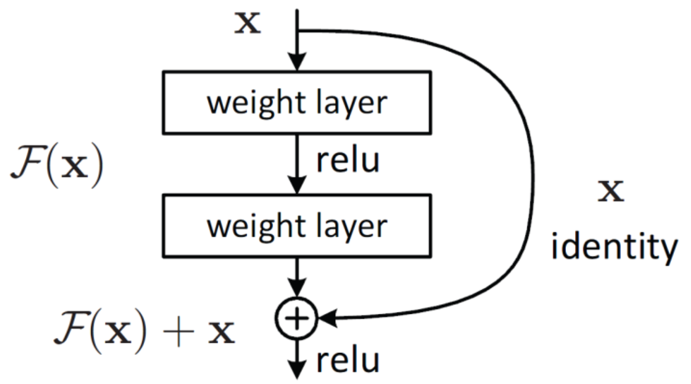
Convolutional Layer



Fully-connected Layer



Self-attention

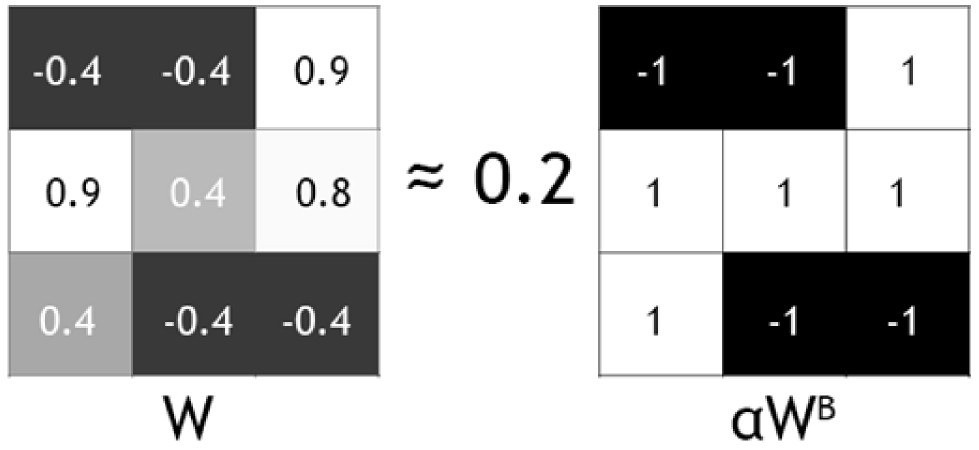


Residual Connection

…

- There are many types of layers in DNN.
- In a DNN, convolutional layers dominates the most computation cost.
- There is a trade-off between general and specific.

Overview

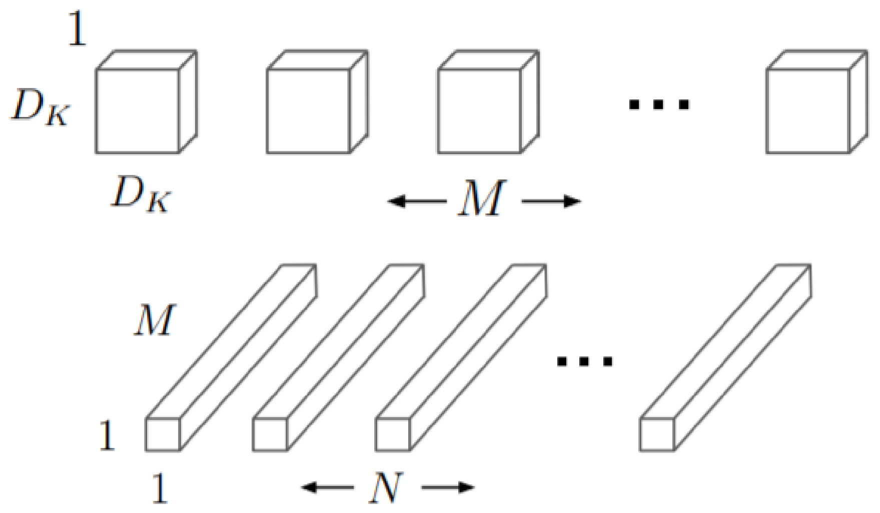


$W^B = \text{sign}(W)$

$\alpha = \frac{1}{n}\|W\|_{l1}$

Binarized Neural Network (BNN), *NIPS 2016*

- Binary weights and activations
- All the multiplication can be replaced with XNOR
- Learning process is hard to be implemented on hardware



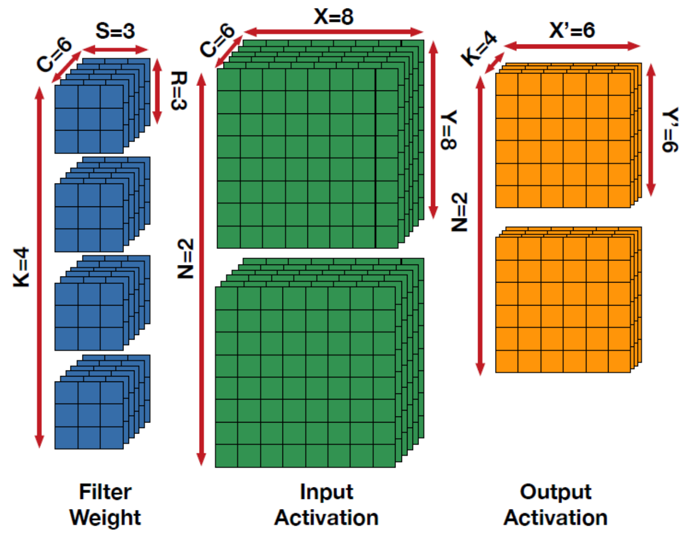
Depthwise Convolution

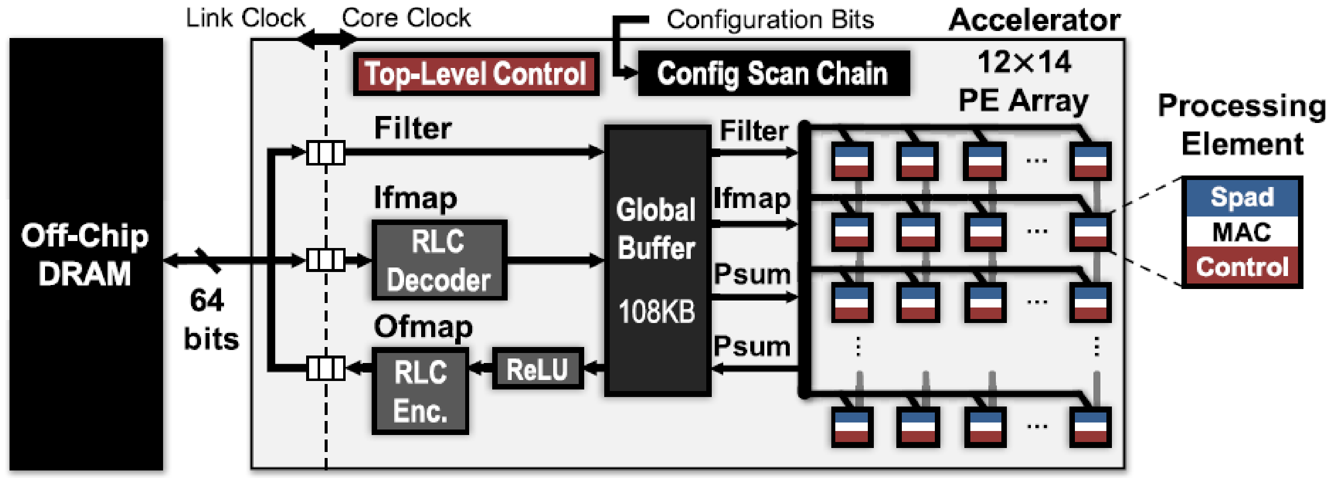
Pointwise Convolution

MobileNet, *NIPS 2016*

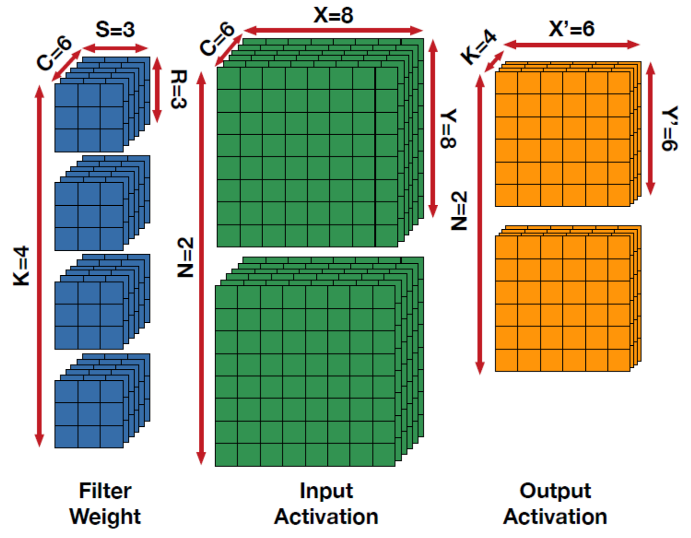
- Efficient networks for mobile IoT devices
- Reduce the channels of weight kernels
- 96.9% parameters reduced compared with VGG-16

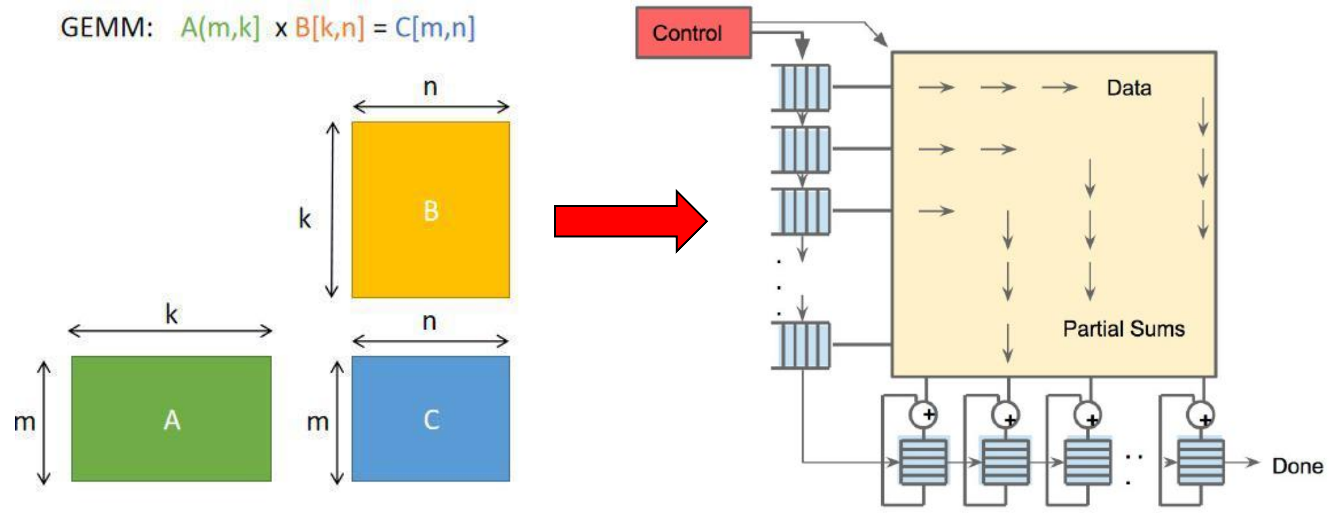
Overview





Eyeriss, *JSSC 2017*

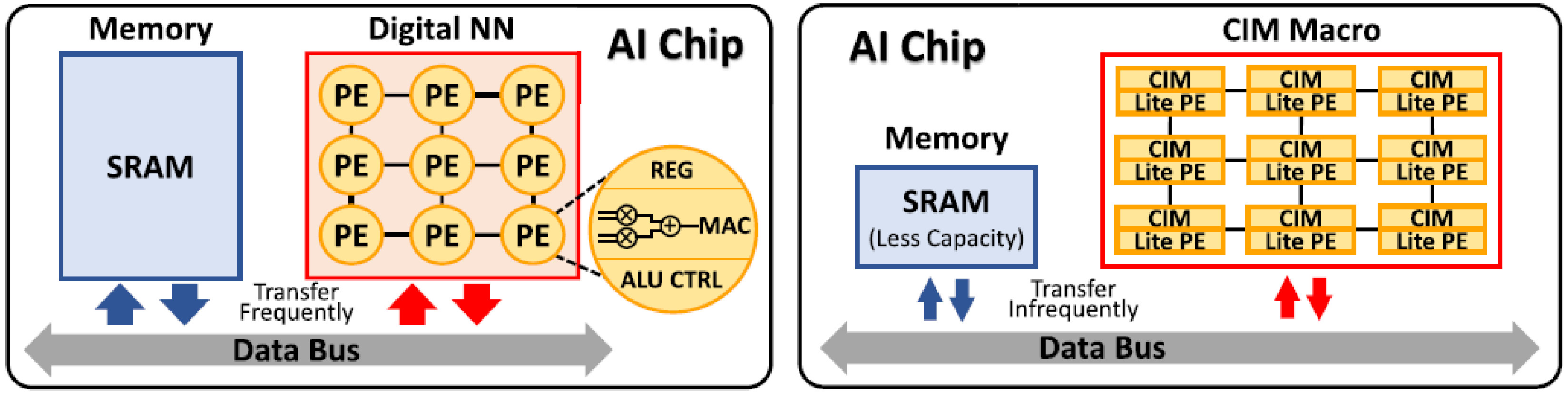




Convolutional Layer

Google TPU, *ISCA 2017*

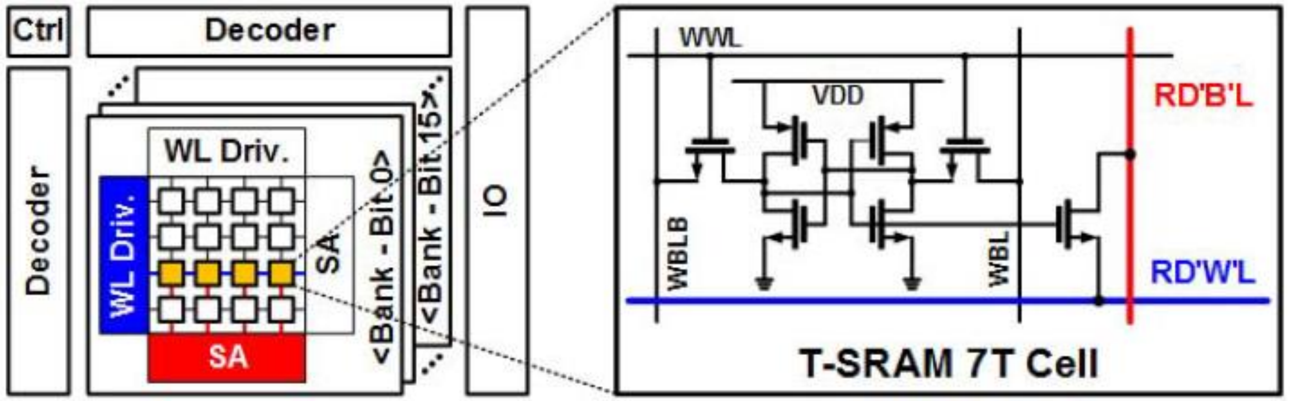
Overview



Chuan-Jia Jhang et al., *IEEE TCAS-I 2021*

Computing-in-memory

Reducing global buffer size and transfer frequency



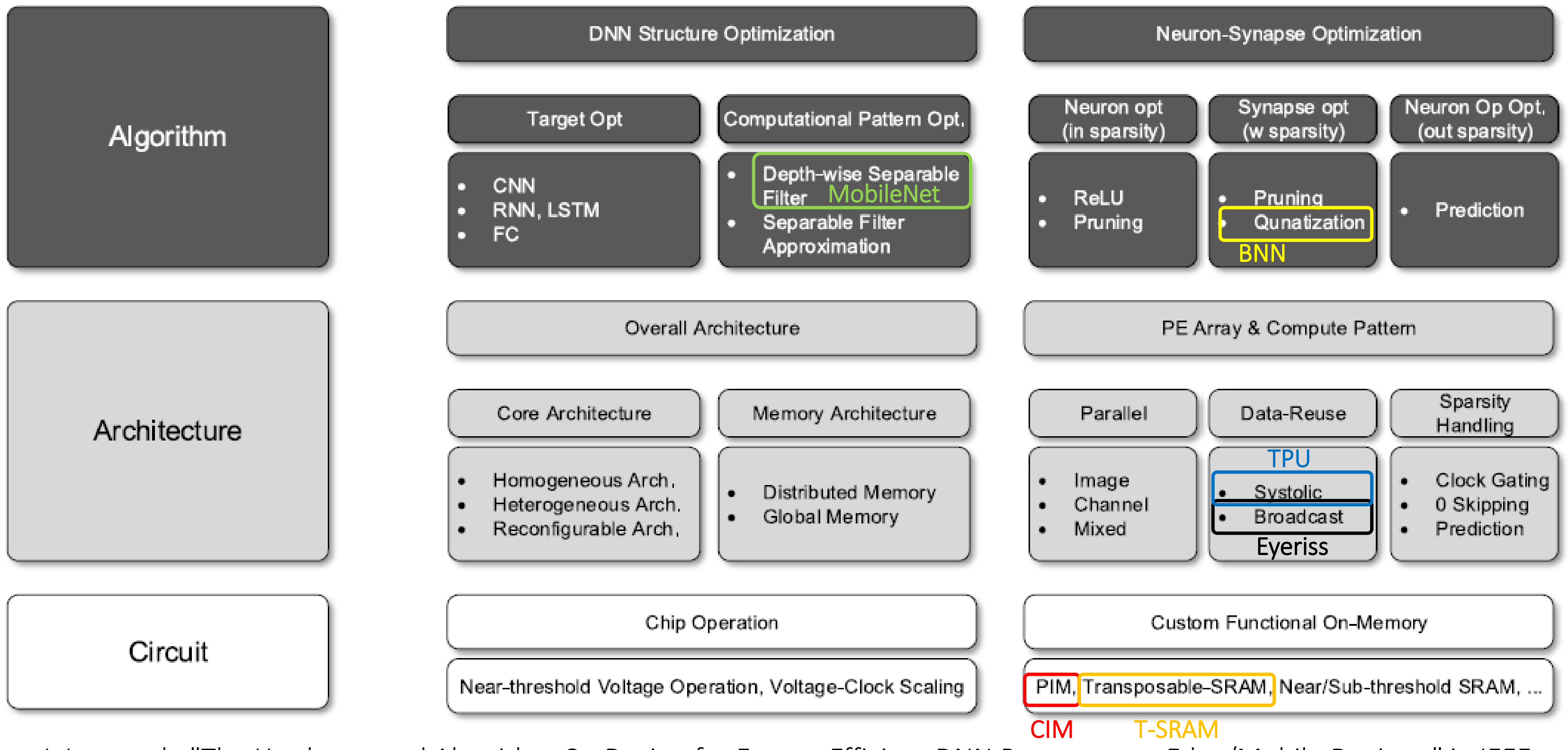
Kyeongryeol Bong et al., *IEEE JSSC 2018*

Transposable Memory (T-SRAM)

Weight matrices are transposed during forward and backward propagation.

Overview

Optimizations of DNN Accelerators



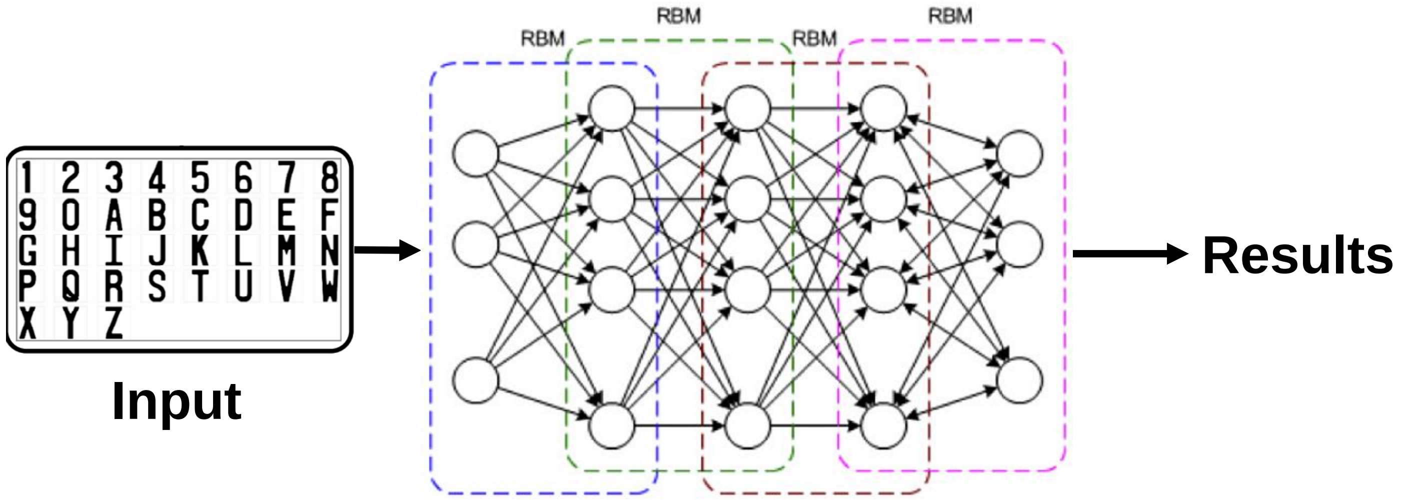
J. Lee et al., "The Hardware and Algorithm Co-Design for Energy-Efficient DNN Processor on Edge/Mobile Devices," in IEEE Transactions on Circuits and Systems I: Regular Papers, vol. 67, no. 10, pp. 3458-3470, Oct. 2020

Outline

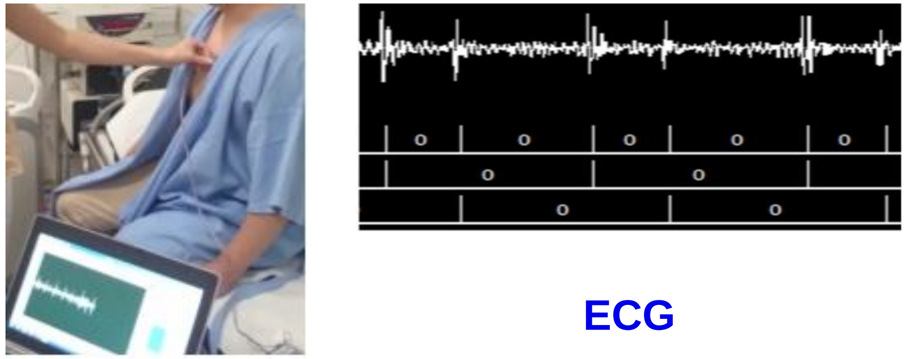
- Introduction
- Overview of DNN Accelerators
- **Case Study - Deep Belief Network Processor**
- Broaden Vision - Development Trend of DNN Accelerators
- Conclusion
- Q&A

Case Study

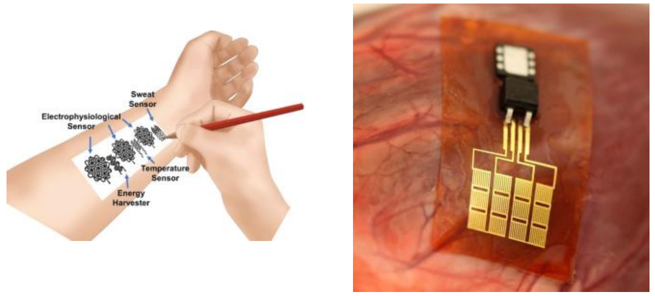
Deep Belief Network (DBN)



- Binary activations
- Unsupervised learning dominates
- Local update of weights

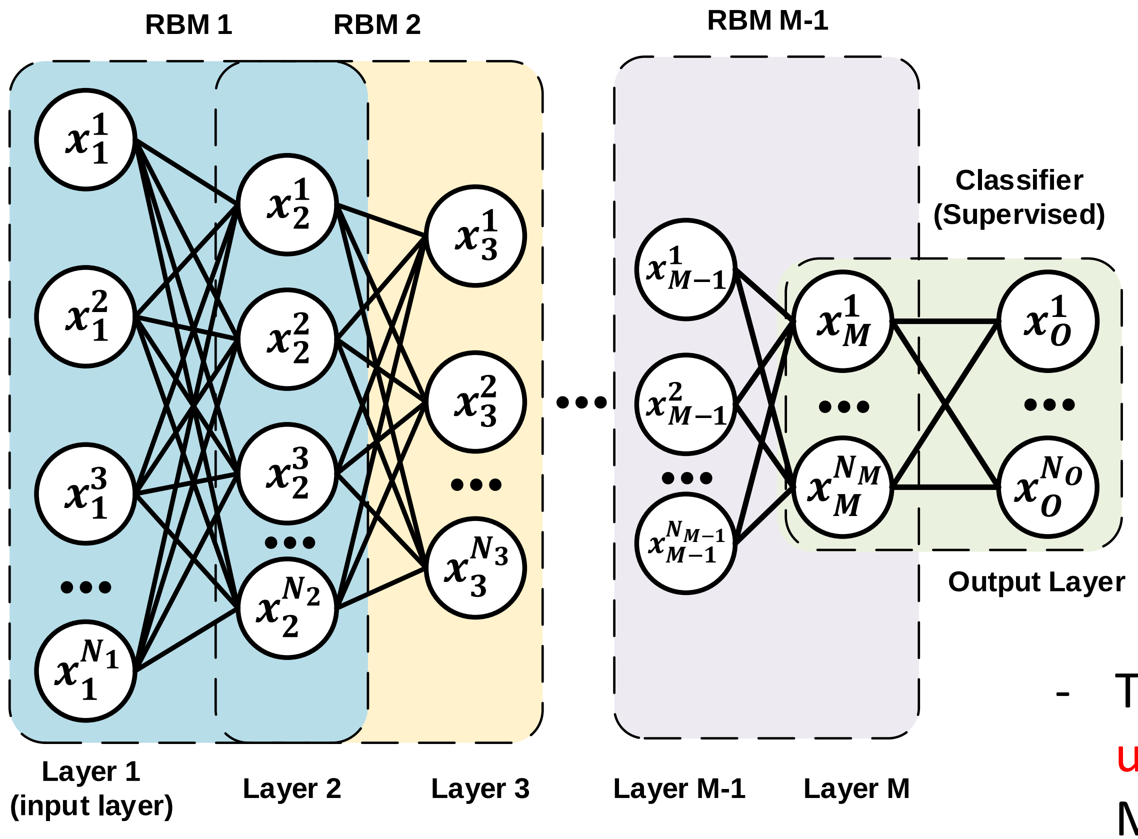

ECG

Robots

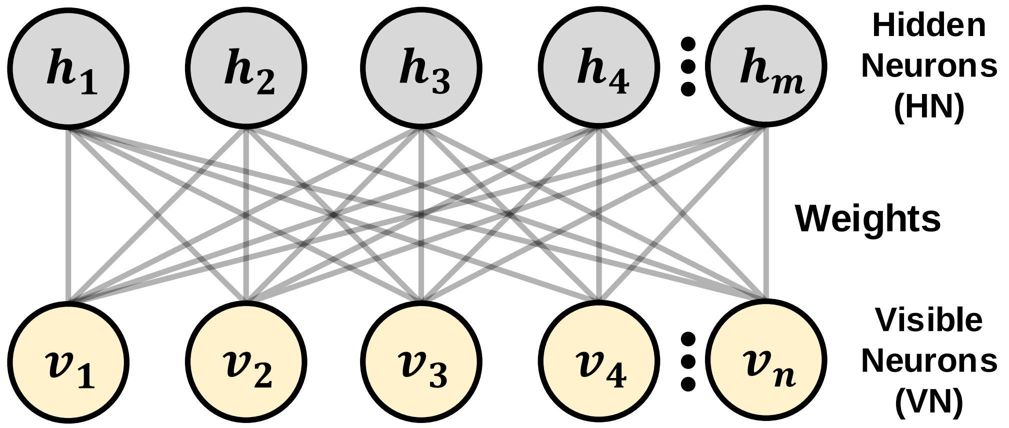


Wearable devices

Case Study



DBN Model



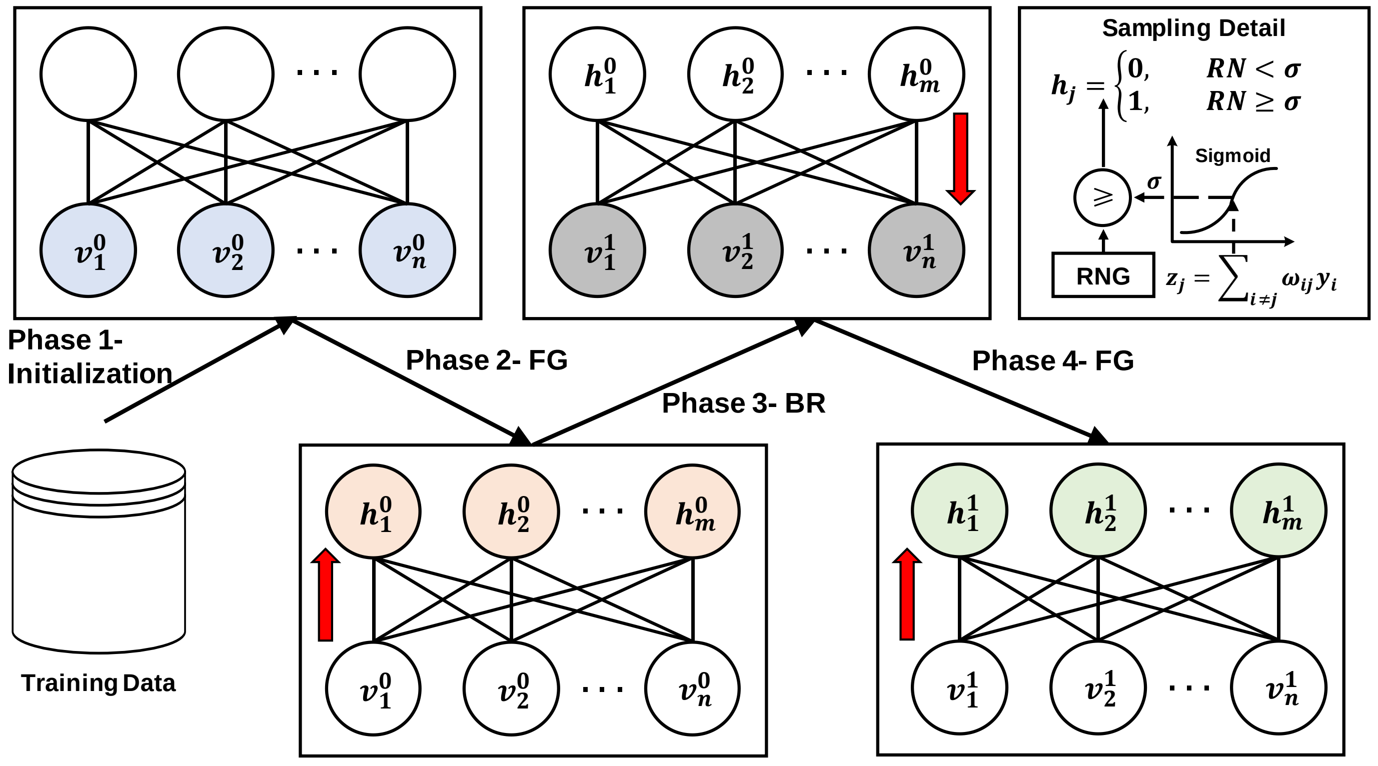
RBM Model

- The learning process of DBN is divided into unsupervised learning of several Restricted Boltzmann Machine (RBM).

- RBM is a probabilistic model serves as encoder-decoder

Jiajun Wu et al., "An Energy-efficient Deep Belief Network Processor Based on Heterogeneous Multi-core Architecture with Transposable Memory and On-chip Learning," *IEEE Journal on Emerging and Selected Topics in Circuits and Systems*, 2021, under review.

Case Study



RBM Unsupervised Learning – CD Algorithm

- Two Forward Generation (FG) and one Backward Reconstruction (BR) phases before updating weights.
- Weighted Sum and Gibbs Sampling are processed under each phase.
- For instance,

$$\boxed{z_j^{h^0} = \sum_{i=1}^{n} v_i^0 \times w_{i,j}} \quad h_j^0 = \begin{cases} 1, & \sigma(z_j^{h^0}) \geq RN \\ 0, & \sigma(z_j^{h^0}) < RN \end{cases}$$

Multiply-Accumulation, MAC **Gibbs Sampling**

- **Update,**

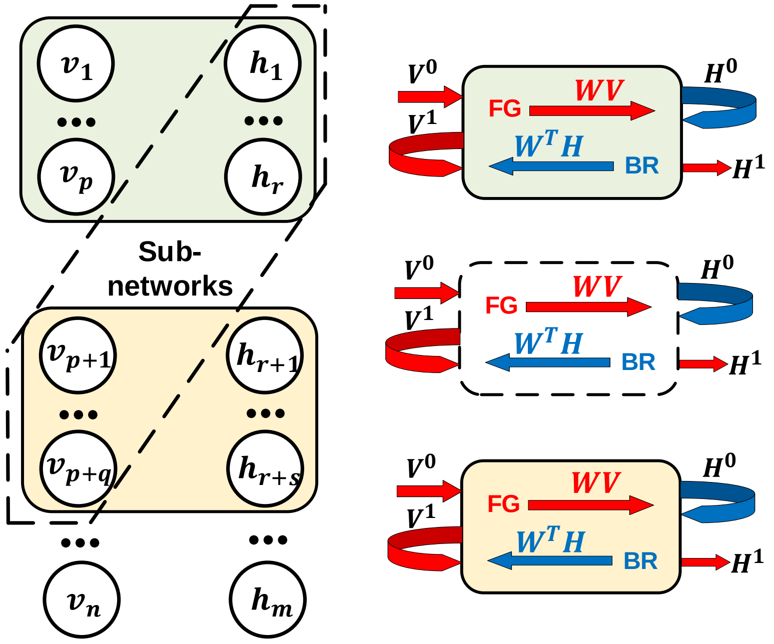
$$\Delta W = \delta(\boldsymbol{V^0 H^0} - \boldsymbol{V^1 H^1})$$ **Local!**

Jiajun Wu et al., "An Energy-efficient Multi-core Restricted Boltzmann Machine Processor with On-chip Bio-plausible Learning and Reconfigurable Sparsity," in *2020 IEEE Asian Solid-State Circuits Conference (A-SSCC)*, 2020.

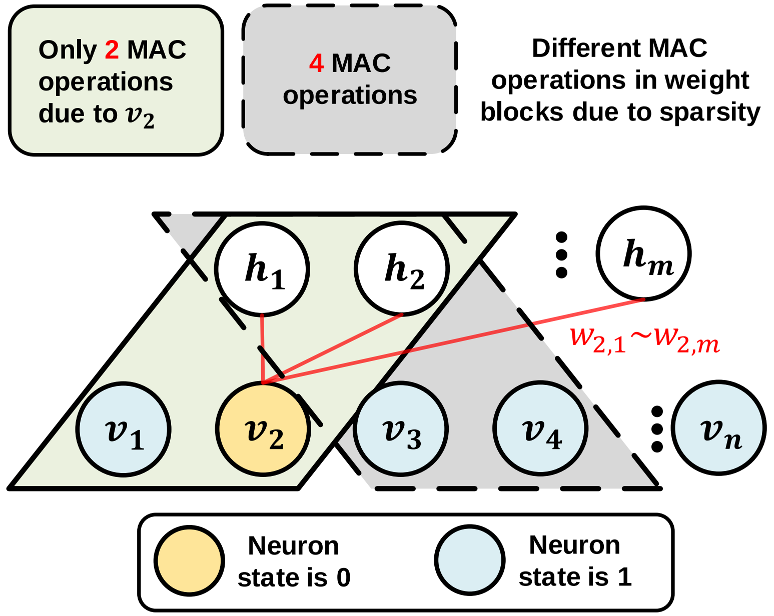
Case Study

Parallel $z_1^{h^0} = \sum_{j=1}^{n} v_j^0 \times w_{1,j} = \sum_{j=1}^{p} v_j^0 \times w_{1,j} + \sum_{j=p+1}^{p+q} v_j^0 \times w_{1,j} + \cdots + v_n^0 \times w_{m,n}$ **Sub-network**

Sparsity $v_j^0 \times w_{1,j}$ **If $v_j^0 = 0$, this multiplication will not contribute to results**



Sub-network & Reuse weights

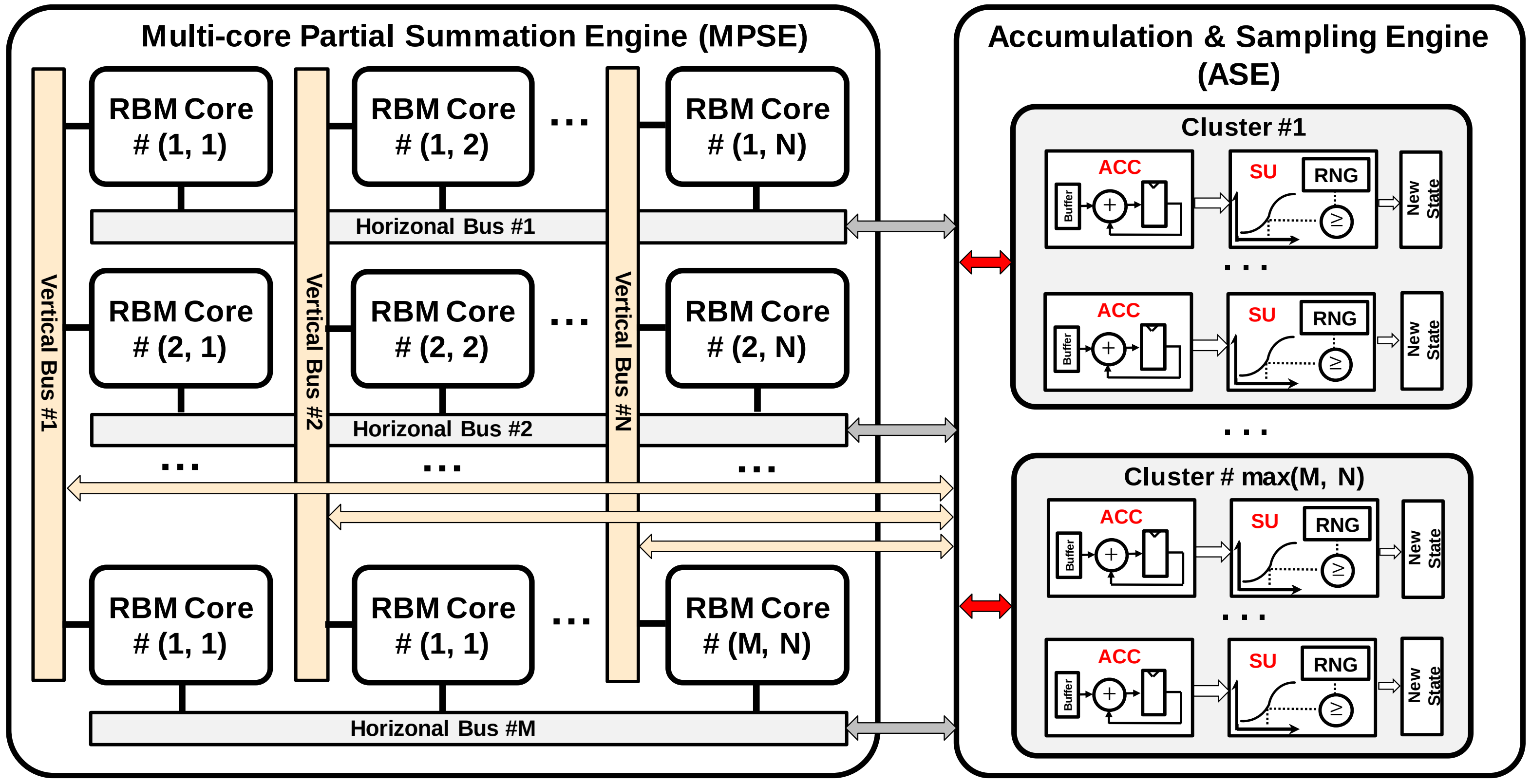


Sparsity of neuron states

Jiajun Wu et al., "An Energy-efficient Deep Belief Network Processor Based on Heterogeneous Multi-core Architecture with Transposable Memory and On-chip Learning," *IEEE Journal on Emerging and Selected Topics in Circuits and Systems*, 2021, under review.

Case Study

Architecture Design



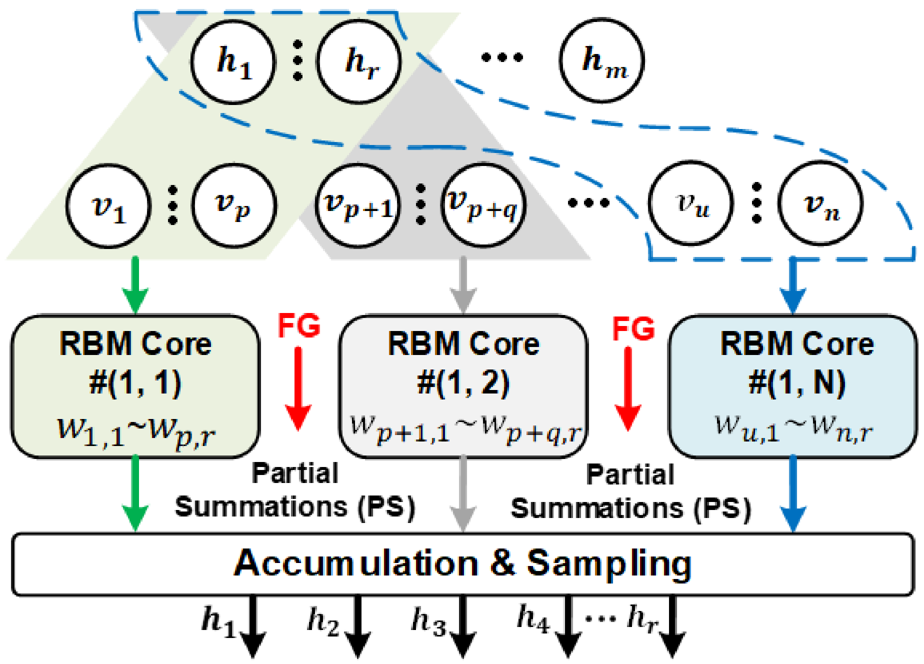
Proposed Heterogeneous Multi-core Architecture

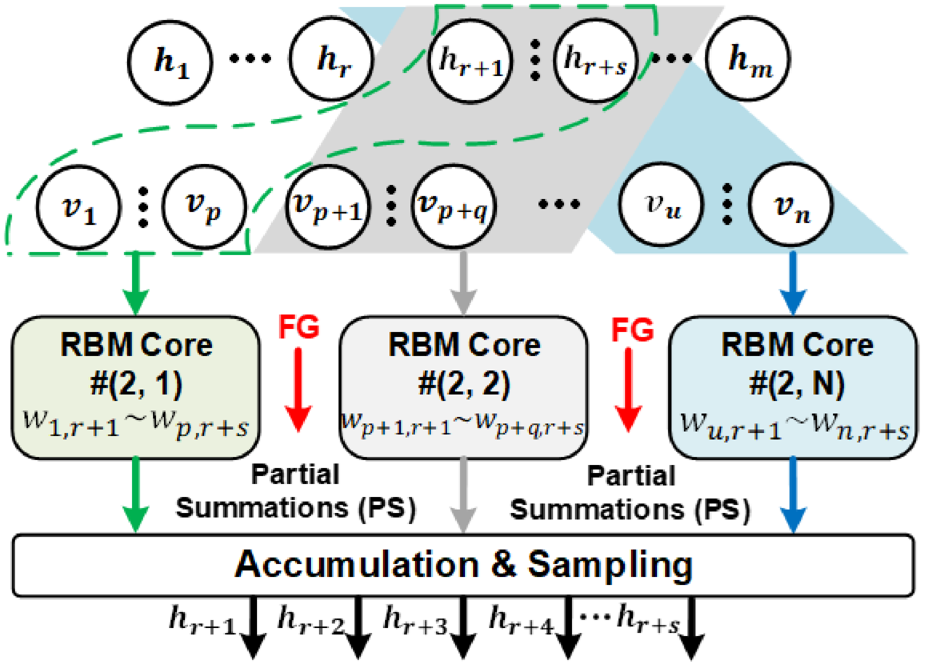
- MPSE is for MAC operations of each sub-network.
- ASE is for accumulating partial sums of sub-networks, and sampling new states by Gibbs Sampling.
- Local weights and local learning.
- Limit the data transition into local bus, reducing communication costs.

Jiajun Wu et al., "An Energy-efficient Deep Belief Network Processor Based on Heterogeneous Multi-core Architecture with Transposable Memory and On-chip Learning," *IEEE Journal on Emerging and Selected Topics in Circuits and Systems*, 2021, under review.

Case Study

Data Mapping


h_1
h_r
h_m
v_1
v_p
v_{p+1}
v_{p+q}
v_u
v_n
RBM Core #(1, 1) $w_{1,1}$~$w_{p,r}$
FG
RBM Core #(1, 2) $w_{p+1,1}$~$w_{p+q,r}$
FG
RBM Core #(1, N) $w_{u,1}$~$w_{n,r}$
Partial Summations (PS)
Partial Summations (PS)
Accumulation & Sampling
h_1 h_2 h_3 h_4 … h_r


h_1
h_r
h_{r+1}
h_{r+s}
h_m
v_1
v_p
v_{p+1}
v_{p+q}
v_u
v_n
RBM Core #(2, 1) $w_{1,r+1}$~$w_{p,r+s}$
FG
RBM Core #(2, 2) $w_{p+1,r+1}$~$w_{p+q,r+s}$
FG
RBM Core #(2, N) $w_{u,r+1}$~$w_{n,r+s}$
Partial Summations (PS)
Partial Summations (PS)
Accumulation & Sampling
h_{r+1} h_{r+2} h_{r+3} h_{r+4} … h_{r+s}

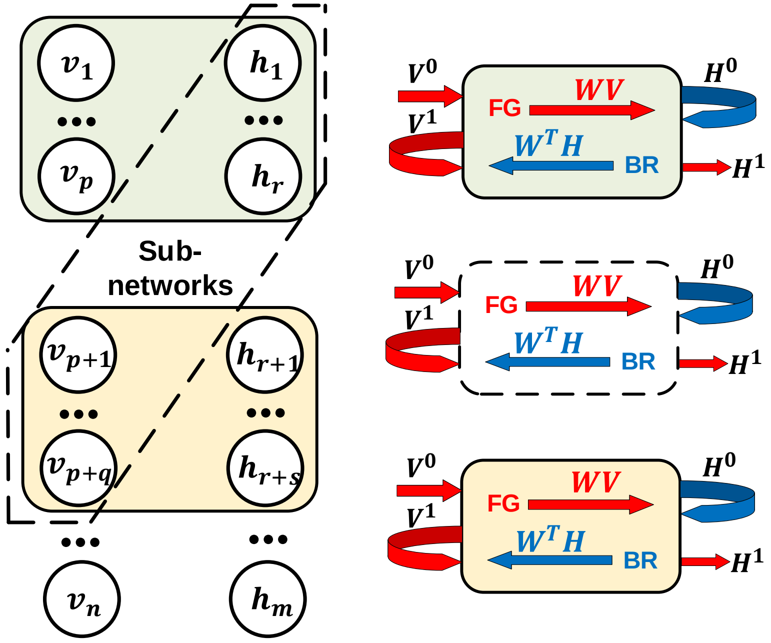
- Each RBM core is responsible for one sub-network.

- $\boldsymbol{h_1}$~$\boldsymbol{h_r}$ are mapped into the first row of RBM cores in MPSE. $\boldsymbol{h_{r+1}}$~$\boldsymbol{h_{r+s}}$ are mapped into the second row of RBM cores.

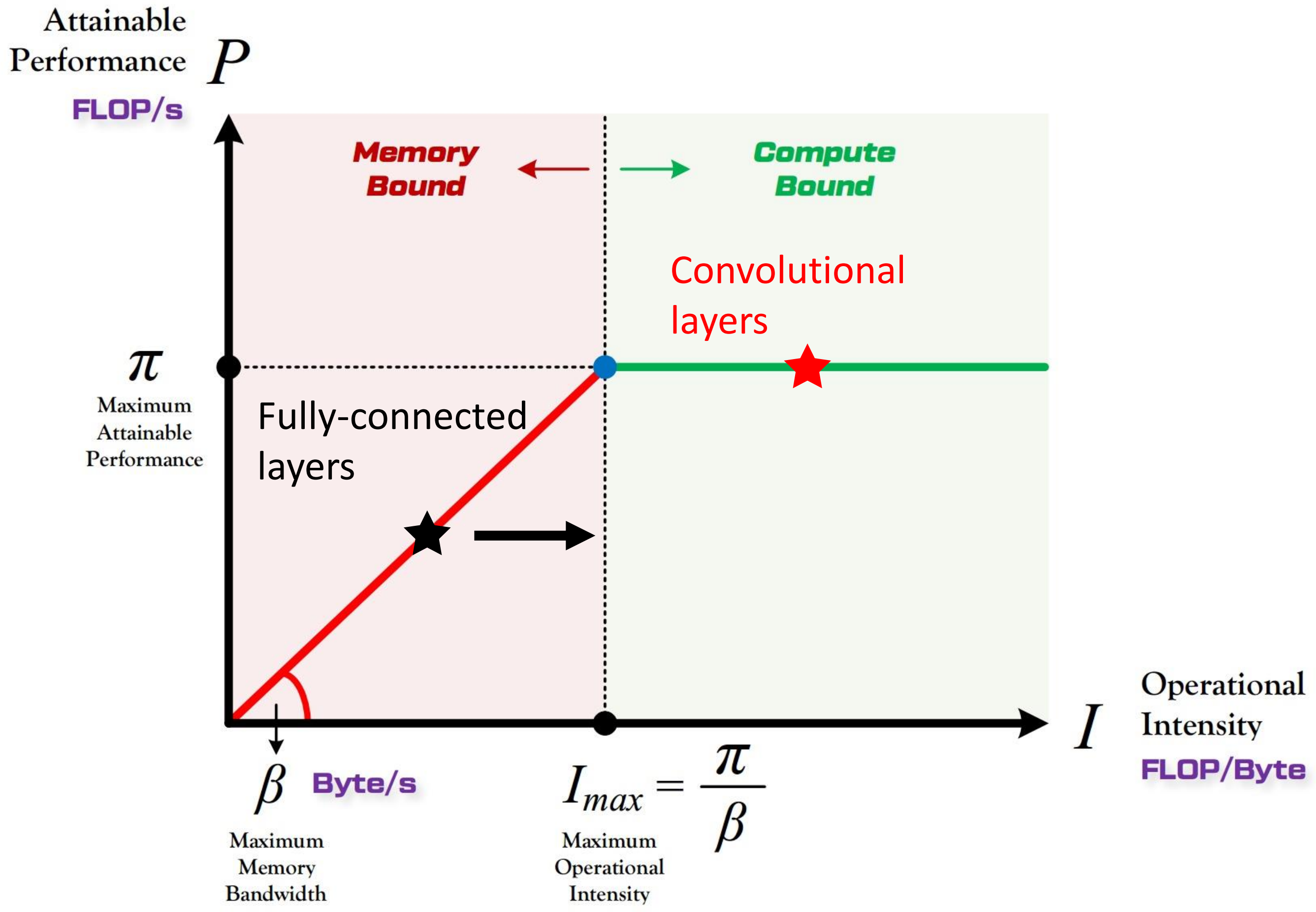
……

Case Study

Why the weights are all mapped into on-chip local memory?



Sub-network & Reuse weights

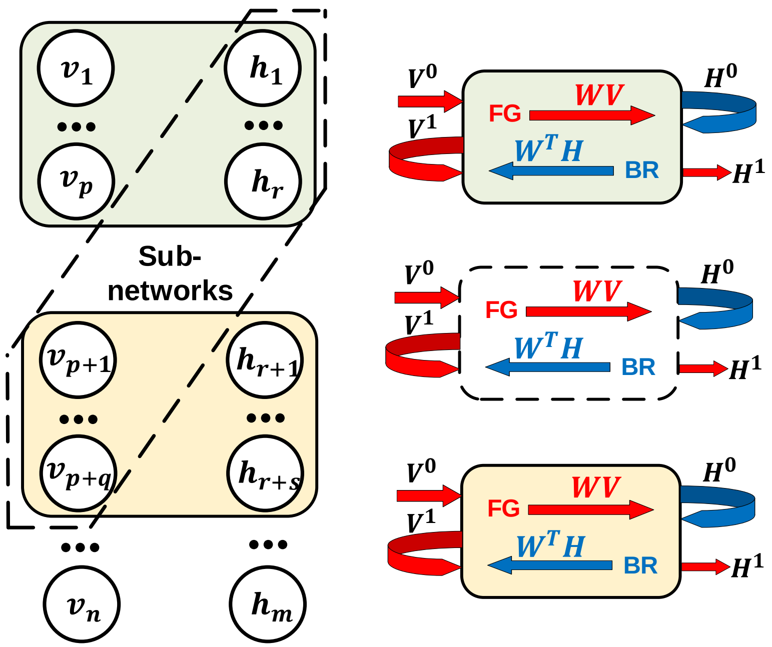


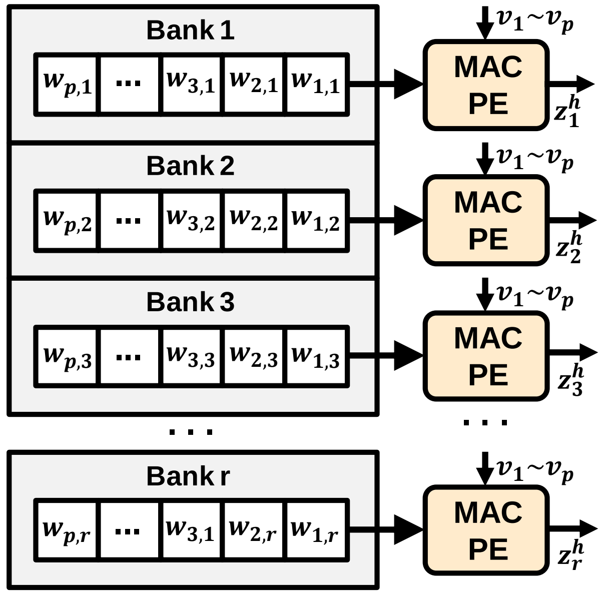
Roofline Model

Source: https://blog.csdn.net/dcrmg/article/details/79652587

Case Study

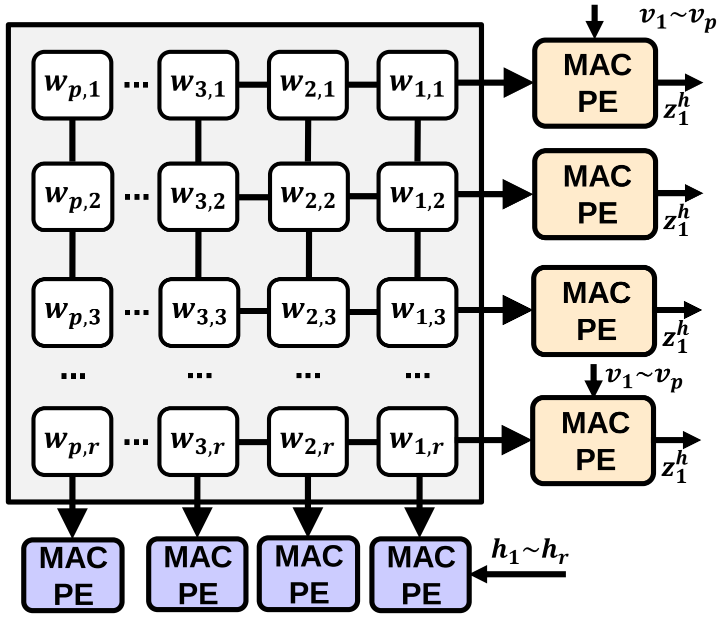
Circuit Design





Bank 1
$w_{p,1}$ … $w_{3,1}$ $w_{2,1}$ $w_{1,1}$
Bank 2
$w_{p,2}$ … $w_{3,2}$ $w_{2,2}$ $w_{1,2}$
Bank 3
$w_{p,3}$ … $w_{3,3}$ $w_{2,3}$ $w_{1,3}$
Bank r
$w_{p,r}$ … $w_{3,1}$ $w_{2,r}$ $w_{1,r}$
MAC PE
$v_1 \sim v_p$
z_1^h
z_2^h
z_3^h
z_r^h

Conventional memory: on-chip multi-bank SRAM for parallel MAC



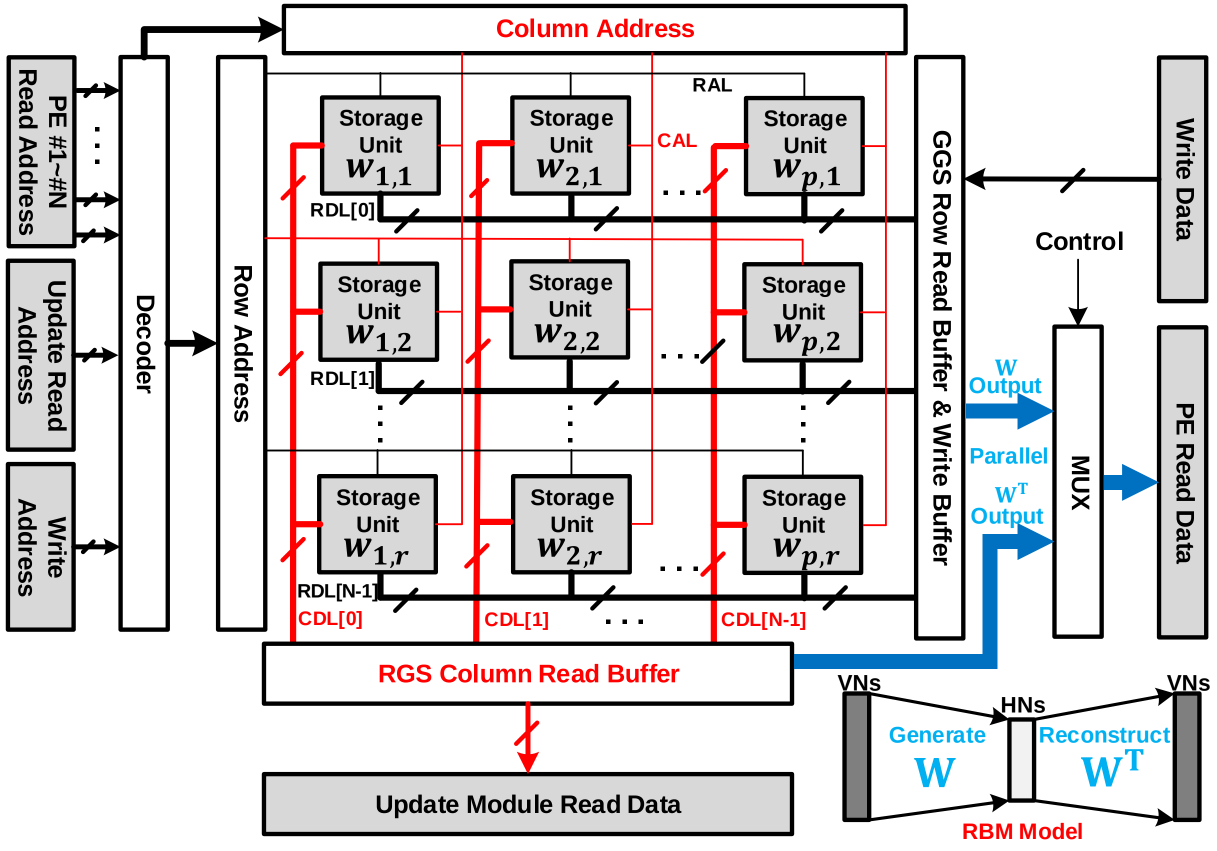
$w_{p,1}$ … $w_{3,1}$ $w_{2,1}$ $w_{1,1}$
$w_{p,2}$ … $w_{3,2}$ $w_{2,2}$ $w_{1,2}$
$w_{p,3}$ … $w_{3,3}$ $w_{2,3}$ $w_{1,3}$
$w_{p,r}$ … $w_{3,r}$ $w_{2,r}$ $w_{1,r}$
MAC PE
$v_1 \sim v_p$
z_1^h
$h_1 \sim h_r$

Transposable memory: bidirectional multi-port parallel readout (row and column) for parallel MAC.

- How to efficiently compute MACs in one sub-network?
- Utilize the transposed reuse of local weights -> Transposable Memory to improve energy efficiency

Case Study

Circuit Design



Proposed Transposable Memory

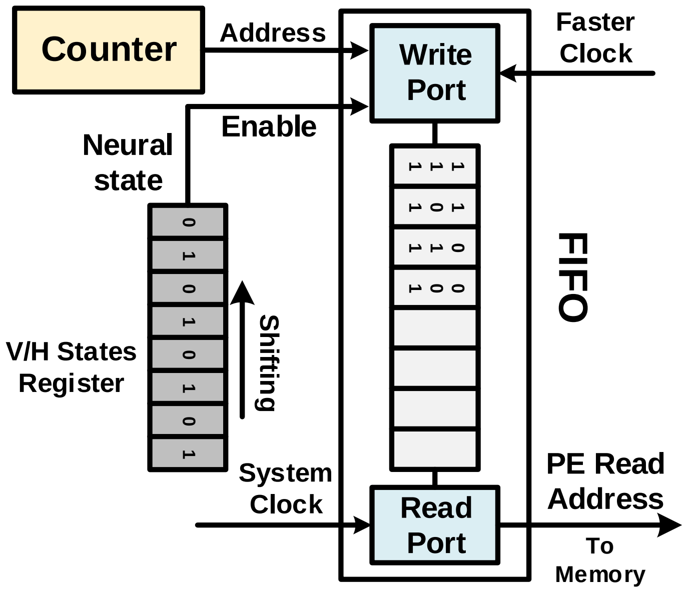
- Compared with conventional SRAM, we added a new set of address lines and data lines

- Register-based memory

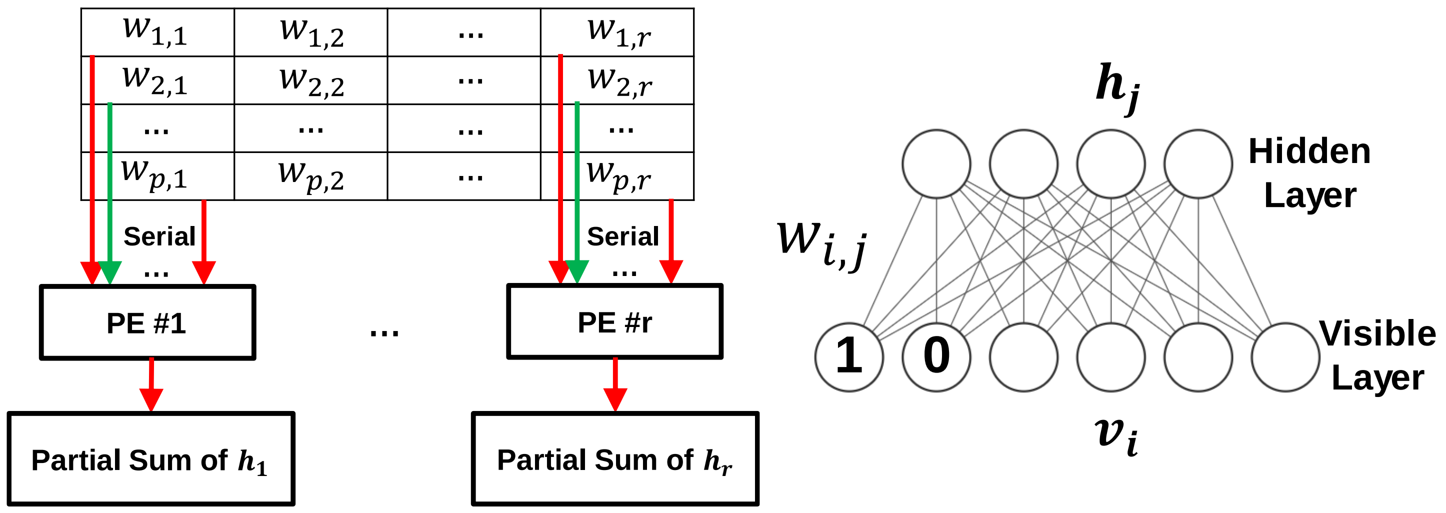
Jiajun Wu et al., "An Energy-efficient Multi-core Restricted Boltzmann Machine Processor with On-chip Bio-plausible Learning and Reconfigurable Sparsity," in *2020 IEEE Asian Solid-State Circuits Conference (A-SSCC)*, 2020.

Case Study

Circuit Design



Sparse Address Generator

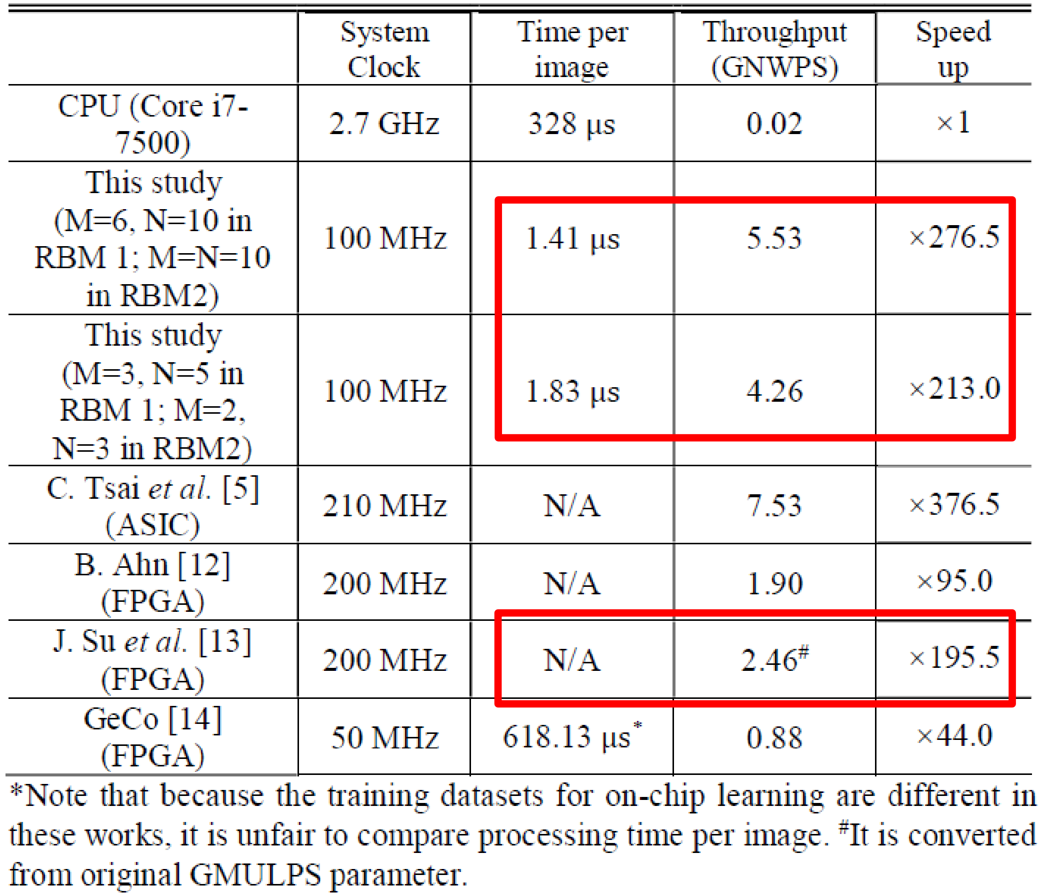


Skip Zero

- The skipping zero method reduces energy consumption of invalid reading and computations
- The sparse addresses are generated BEFORE FG and BR phases

Case Study

Performance - Throughput



	System Clock	Time per image	Throughput (GNWPS)	Speed up
CPU (Core i7-7500)	2.7 GHz	328 μs	0.02	×1
This study (M=6, N=10 in RBM 1; M=N=10 in RBM2)	100 MHz	1.41 μs	5.53	×276.5
This study (M=3, N=5 in RBM 1; M=2, N=3 in RBM2)	100 MHz	1.83 μs	4.26	×213.0
C. Tsai *et al.* [5] (ASIC)	210 MHz	N/A	7.53	×376.5
B. Ahn [12] (FPGA)	200 MHz	N/A	1.90	×95.0
J. Su *et al.* [13] (FPGA)	200 MHz	N/A	2.46#	×195.5
GeCo [14] (FPGA)	50 MHz	618.13 μs*	0.88	×44.0

*Note that because the training datasets for on-chip learning are different in these works, it is unfair to compare processing time per image. #It is converted from original GMULPS parameter.

- GNWPS： G Neural Weights Per Second
- ×2.25 faster than the state-of-art FPGA work
- The architecture with more RBM cores has higher throughput
- We can also use "FLOPS/s" to evaluate the throughput. But it does not suit this case.

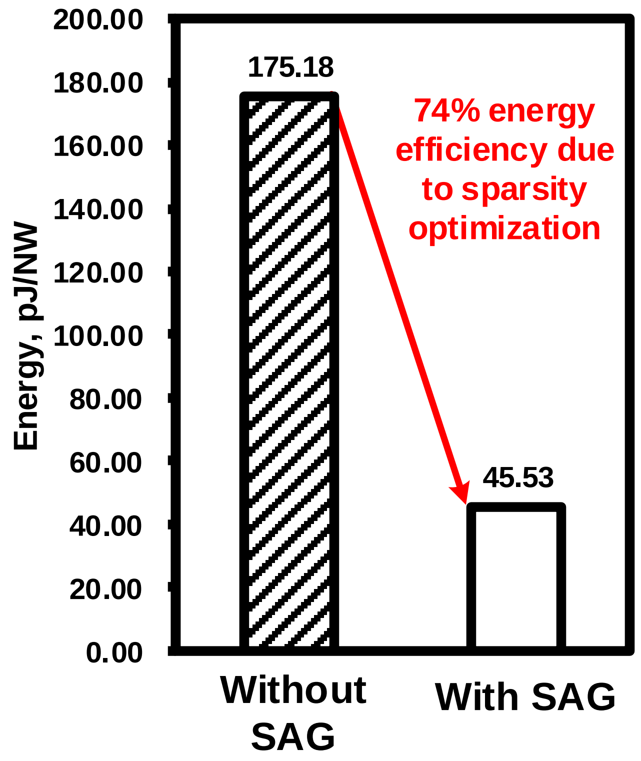
Jiajun Wu et al., "An Energy-efficient Deep Belief Network Processor Based on Heterogeneous Multi-core Architecture with Transposable Memory and On-chip Learning," *IEEE Journal on Emerging and Selected Topics in Circuits and Systems*, 2021, under review.

Case Study

Performance - Energy Efficiency

-	RBM size	Energy per NW
This study in FPGA	100-60-30	45.0 pJ
C. Tsai *et al.* [5] in 65-nm ASIC	4096-4096	41.31 pJ (energy by external memory access is not included)

- Evaluate energy efficiency: energy per Neural Weights update
- Improve 74% energy efficiency due to sparse address generator



Energy efficiency improvement due to data sparsity optimization

Case Study

From this case, what is the steps we design a hardware accelerator

Three level-based optimization

Algorithm level
- Find domain-specific models and algorithms.
- Exploit data characteristics such as reuse, sparsity etc.

Architecture level
- According to data characteristics, design a specific architecture
- Schedule and data mapping.

Circuit level
- Design efficient computation circuits.
- Optimize circuits based on data and computation characteristics.

Constraints

ASIC: Technology, Maximum Area, Bandwidth and Power etc.
FPGA: Available resource (e.g., LUT, DSP), Maximum Frequency etc.

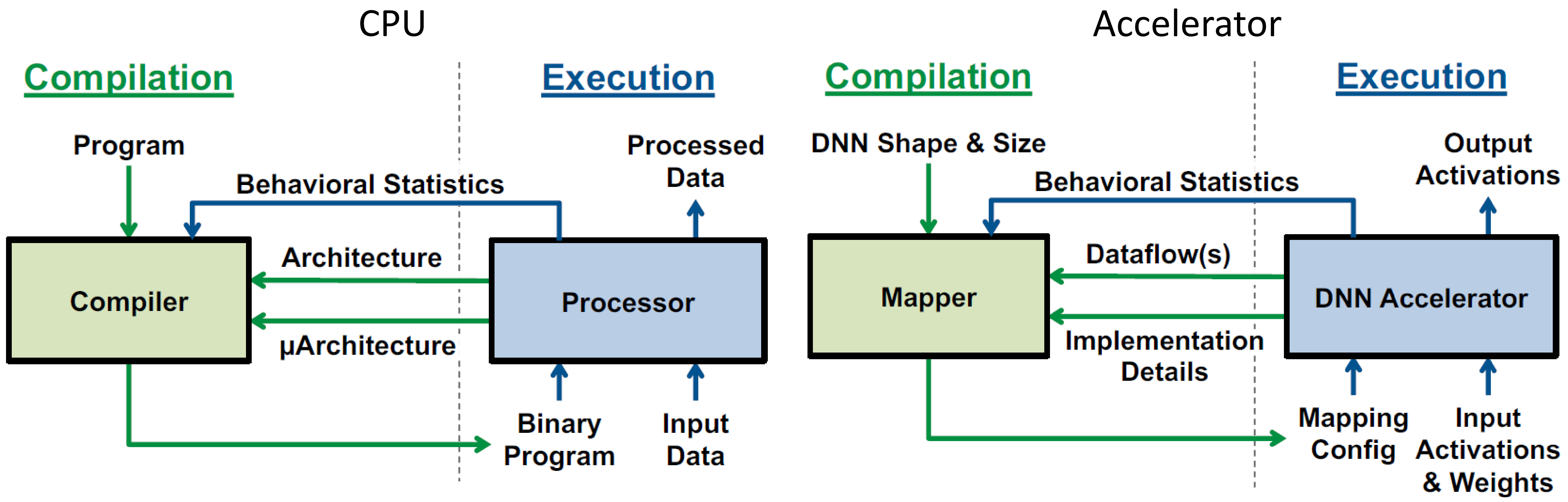
Utilization rate of PEs, Bus…

Performance
- Task performance: accuracy based on specific dataset.
- Acceleration performance: throughput, frequency etc.
- Energy efficiency: Energy during training / inference per batch / image / update etc.

Outline

Development Trend of DNN Accelerators

Part I - Compiler



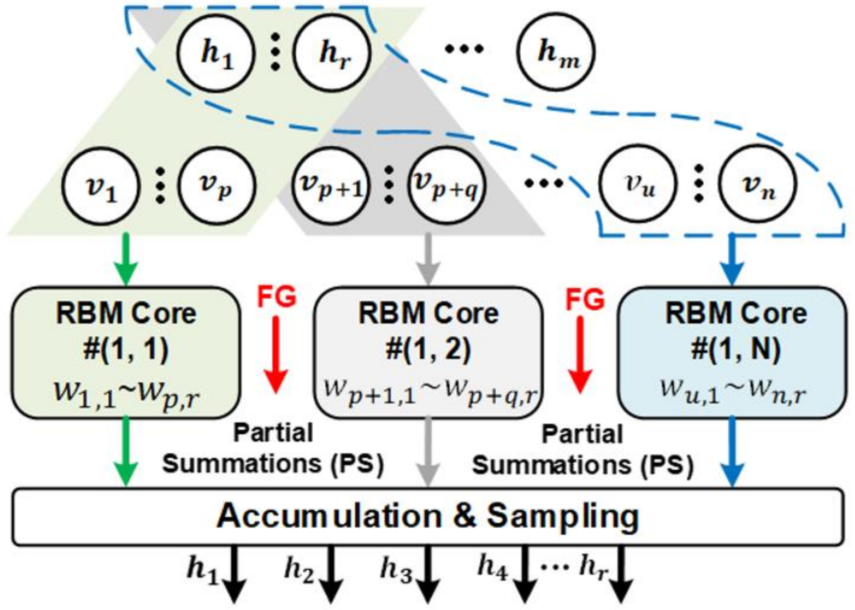
Without compiler/mapper, accelerators are only “evaluation hardware”, and cannot work in applications.

Source: http://eyeriss.mit.edu/tutorial.html

Development Trend of DNN Accelerators

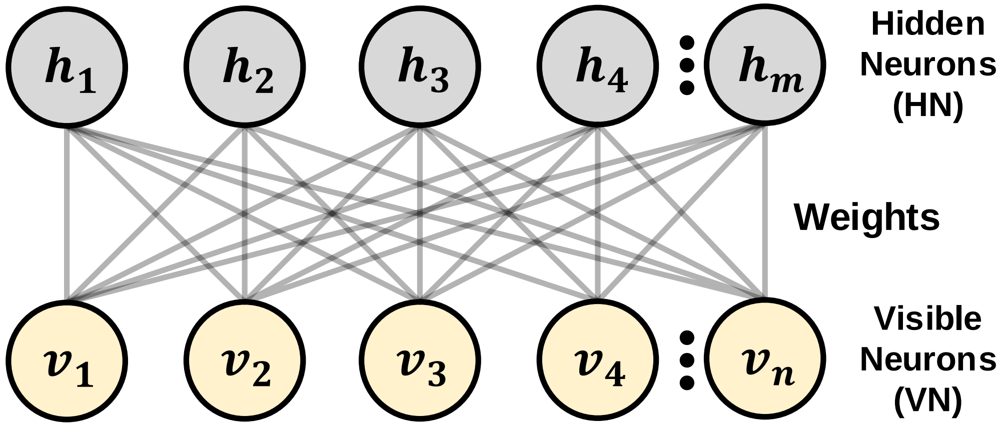
Part I - Compiler

The DBN Processor work has a compiler



h_1
h_r
h_m
v_1
v_p
v_{p+1}
v_{p+q}
v_u
v_n
RBM Core #(1, 1) $w_{1,1}$~$w_{p,r}$
FG
RBM Core #(1, 2) $w_{p+1,1}$~$w_{p+q,r}$
FG
RBM Core #(1, N) $w_{u,1}$~$w_{n,r}$
Partial Summations (PS)
Partial Summations (PS)
Accumulation & Sampling
h_1 h_2 h_3 h_4 … h_r

Map the corresponding neuron states and weights into different RBM Cores.



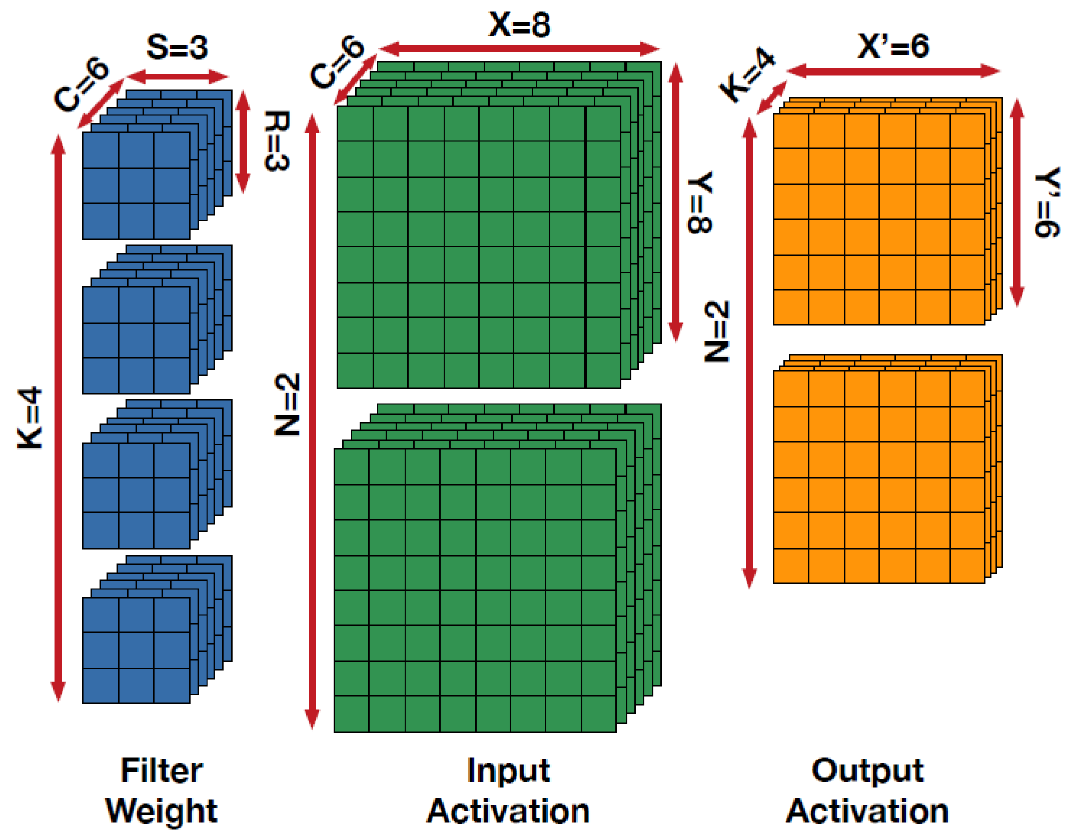
RBM Model

But the RBM model is a simple fully-connected layer, which limits the mapping space.

Development Trend of DNN Accelerators

Part I - Compiler

However, when it comes to Convolutional Layers …



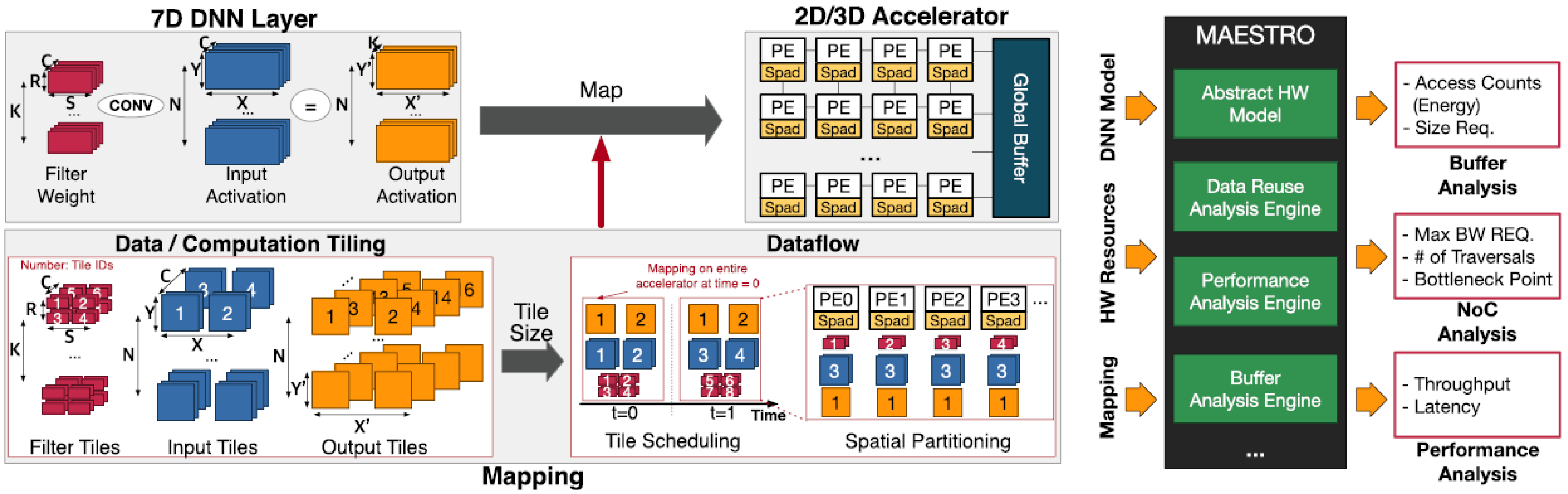
```
for(n=0; n<2; n++)
 for(k=0; k<4; k++)
  for(c=0; c<6; c++)
   for(y=0; y<8; y++)
    for(x=0; x<8; x++)
     for(r=0; r<3; r++)
      for(s=0; s<3; s++)
        O[k][y-r][x-s] += W[k][c][r][s] * I[c][y][x];
```

7D for-loop representation

Exist a large number of mapping / scheduling opportunities

Development Trend of DNN Accelerators

Part I - Compiler



Due to the large mapping space of Convolutional Layers, the compiler should be able to evaluate and judge whether the selected mapping is optimized.

Hyoukjun Kwon et al., “Understanding Reuse, Performance, and Hardware Cost of DNN Dataflows: A Data-Centric Approach,” *in IEEE MICRO 2019, October 12–16, 2019, Columbus, OH, USA*.

Development Trend of DNN Accelerators

Part I - Compiler

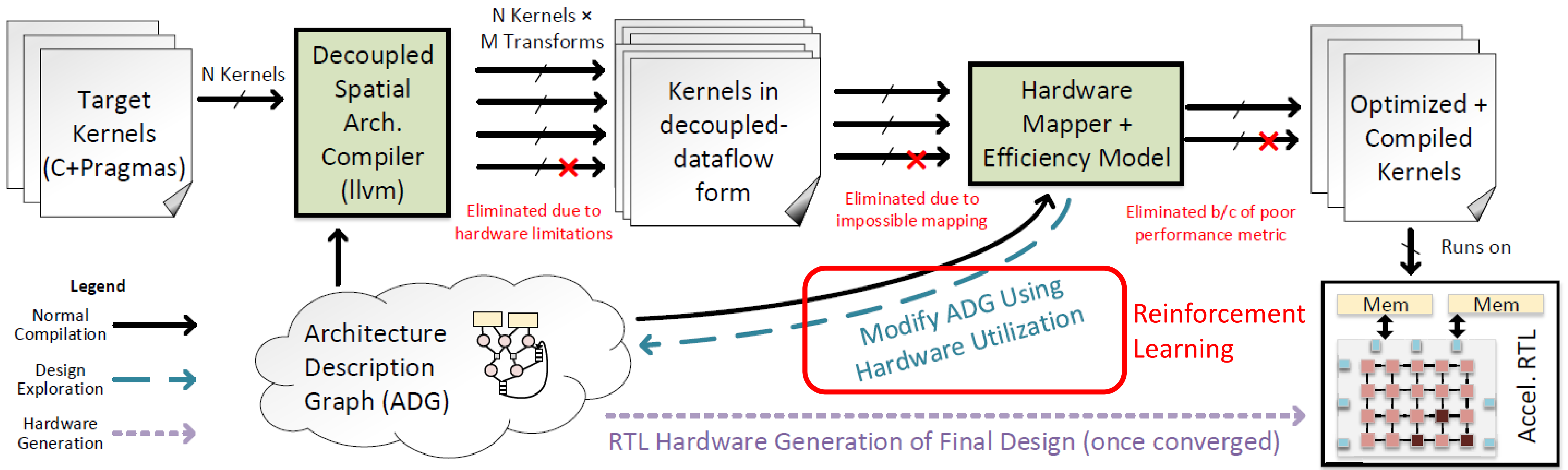
Typical works in recent years

- Tianqi Chen et al., TVM: An Automated End-to-End Optimizing Compiler for Deep Learning, OSDI 2018.
- A. Parashar et al., Timeloop: A Systematic Approach to DNN Accelerator Evaluation, ISPASS 2019.
- H. Kwon et al., Understanding Reuse, Performance, and Hardware Cost of DNN Dataflows: A Data-Centric Approach, MICRO 2019.
- Xuan Yang et al., Interstellar: Using Halide’s Scheduling Language to Analyze DNN Accelerators, ASLPOS 2020.

Development Trend of DNN Accelerators

Part II - High Level Synthesis & Design Space Exploration (DSE)

When we get a DNN model, why not give it to the machine and make itself design the best optimized architecture for us?



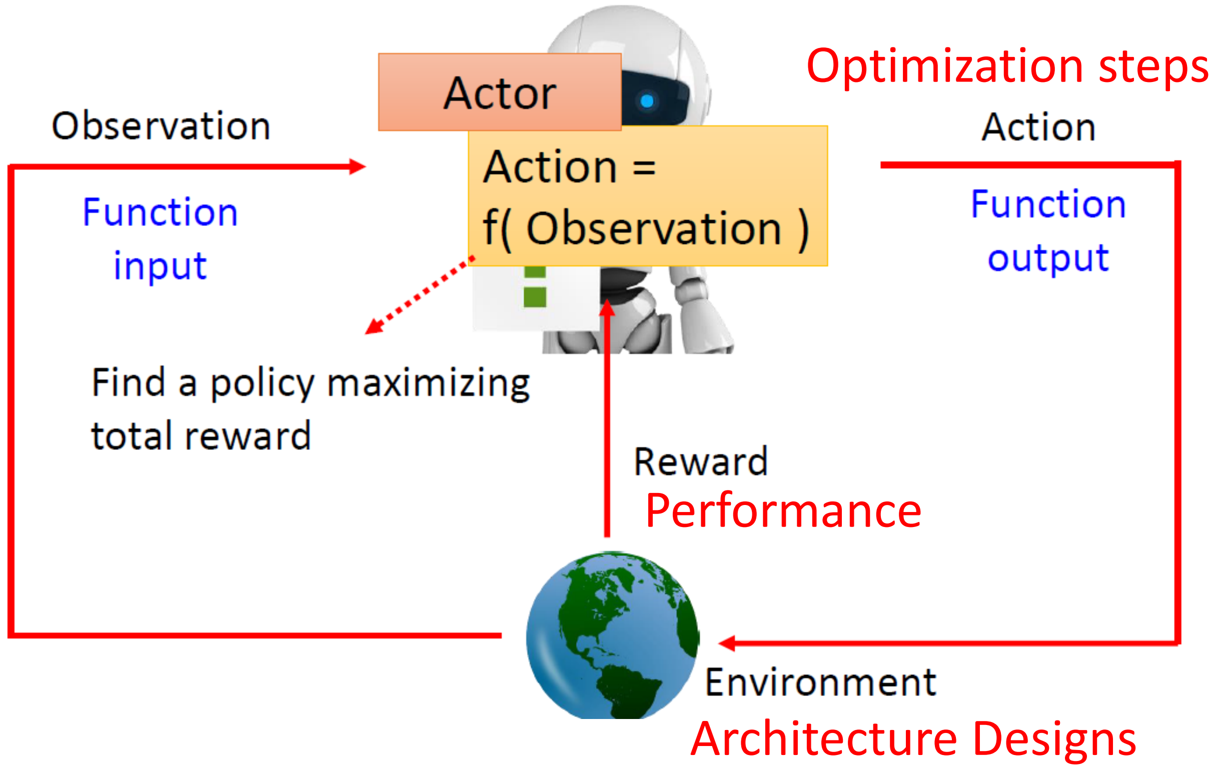
Target Kernels (C+Pragmas)
N Kernels
Decoupled Spatial Arch. Compiler (llvm)
N Kernels × M Transforms
Eliminated due to hardware limitations
Kernels in decoupled-dataflow form
Eliminated due to impossible mapping
Hardware Mapper + Efficiency Model
Eliminated b/c of poor performance metric
Optimized + Compiled Kernels
Runs on
Mem
Mem
Accel. RTL
Legend
Normal Compilation
Design Exploration
Hardware Generation
Architecture Description Graph (ADG)
Modify ADG Using Hardware Utilization
Reinforcement Learning
RTL Hardware Generation of Final Design (once converged)

Jian Weng et al., "DSAGEN: Synthesizing Programmable Spatial Accelerators," *in ACM/IEEE ISCA 2020*.

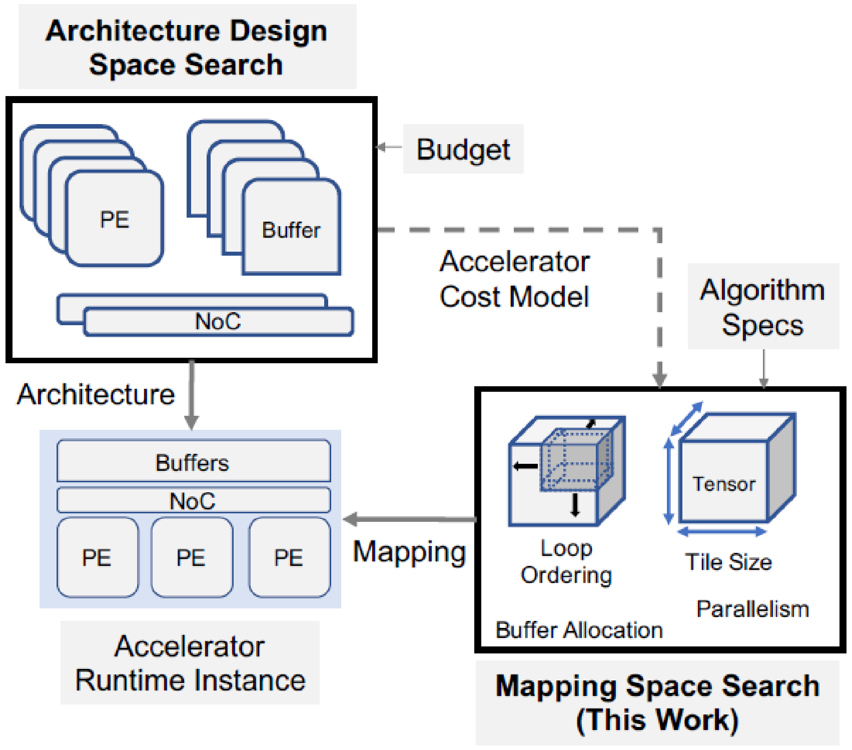
Development Trend of DNN Accelerators

Part II - High Level Synthesis & Design Space Exploration (DSE)

What is Reinforcement Learning (RL)



We can further combine Compiler and DSE Architecture-Compiler co-design



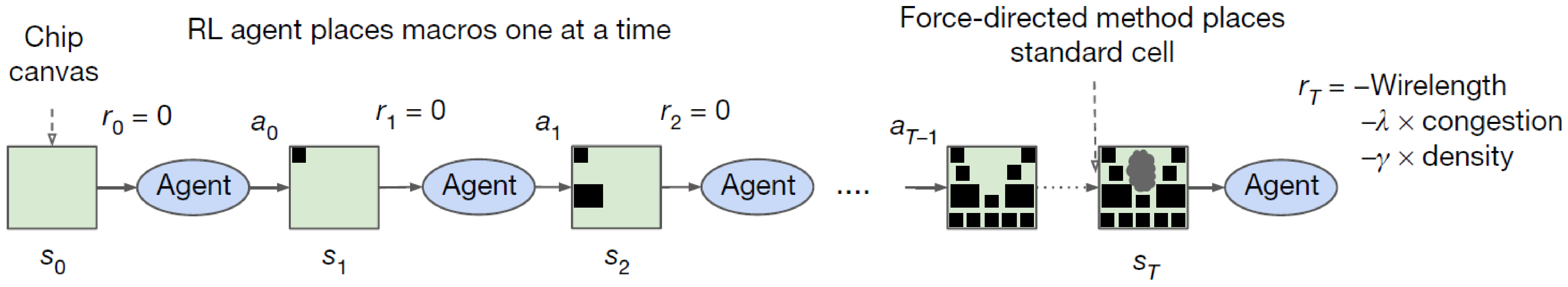
K. Hegde et al., "Mind Mappings: Enabling Efficient Algorithm-Accelerator Mapping Space Search," *in ASPLOS 2021*.

Development Trend of DNN Accelerators

Just a digression - a work published in Nature

Article

A graph placement methodology for fast chip design


Chip canvas
RL agent places macros one at a time
Force-directed method places standard cell
$r_0 = 0$
a_0
$r_1 = 0$
a_1
$r_2 = 0$
a_{T-1}
$r_T = -\text{Wirelength}$
$-\lambda \times \text{congestion}$
$-\gamma \times \text{density}$
Agent
Agent
Agent
Agent
....
s_0
s_1
s_2
s_T

Proposed by Google, and it will be implemented in the next version of TPU

A. Mirhoseini et al., “A graph placement methodology for fast chip design,” *in Nature, June 2021*.

Development Trend of DNN Accelerators

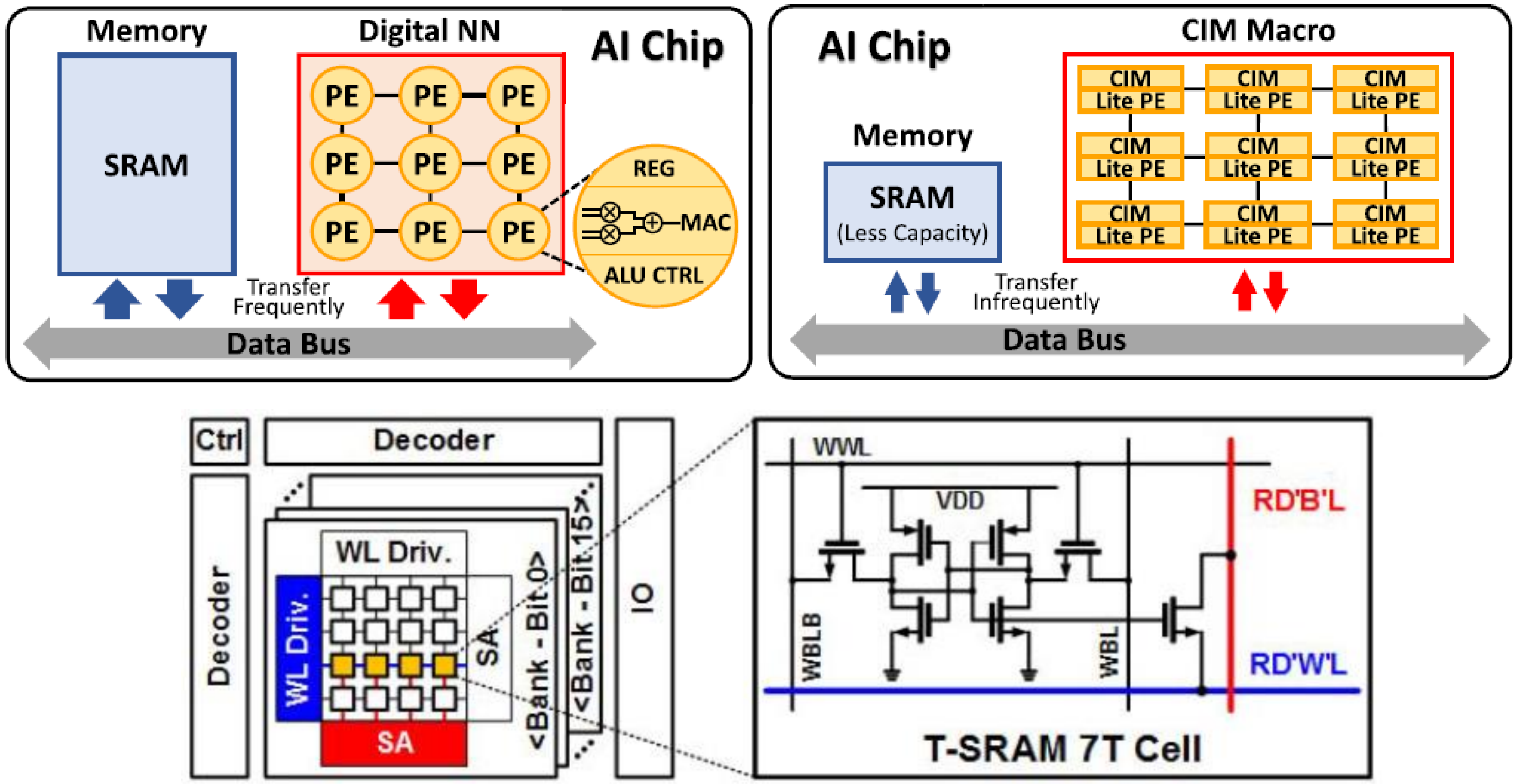
Part II - High Level Synthesis & Design Space Exploration (DSE)

Typical works in recent years

- Jian Weng et al., DSAGEN: Synthesizing Programmable Spatial Accelerators, ISCA 2020.
- Pengfei Xu et al., AutoDNNchip: An Automated DNN Chip Predictor and Builder for Both FPGAs and ASICs, FPGA 2020.
- C. Zhang et al., Caffeine: Towards uniformed representation and acceleration for deep convolutional neural networks, IEEE TCAD, 2018.
- R. Venkatesan et al., MAGNet: A Modular Accelerator Generator for Neural Networks, ICCAD 2019.

Development Trend of DNN Accelerators

Part III - Circuit / Device-level optimization



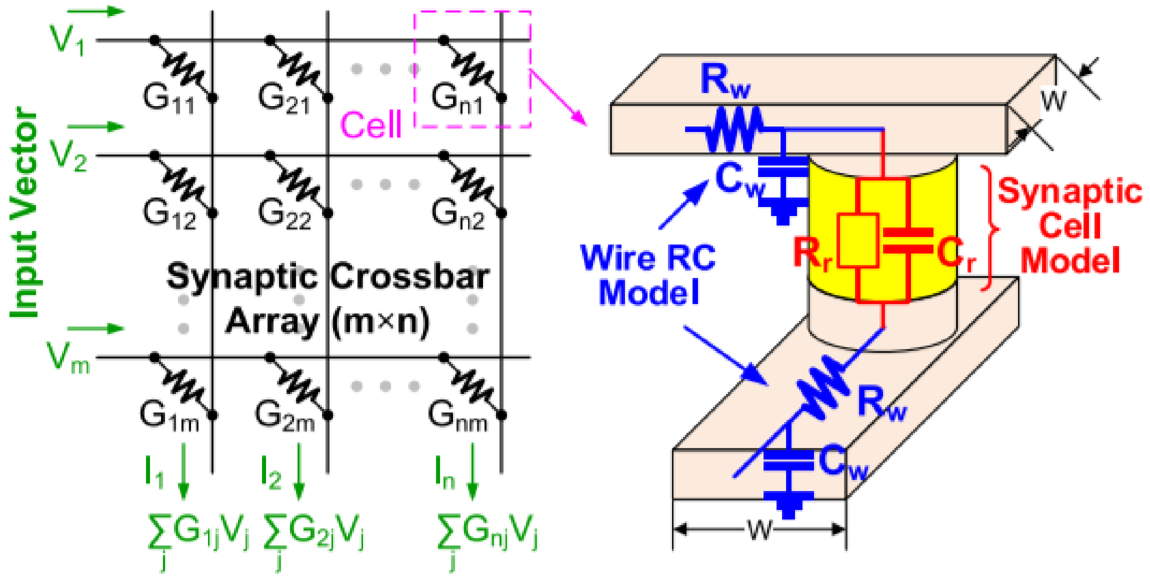
- There are many researchers who proposed circuit-level optimization (e.g., Analog Computing, Computing-in-memory and transposable memory)

- However, it still lacks a general framework for simulation and evaluation of these optimization from bottom up, when targeting a specific model.

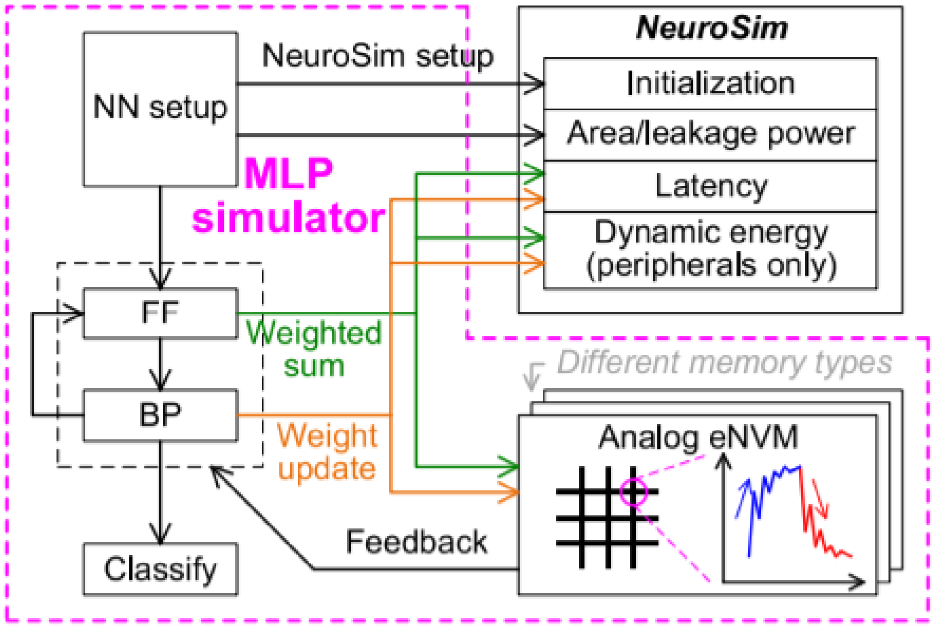
K. Bong et al., "A Low-Power Convolutional Neural Network Face Recognition Processor and a CIS Integrated With Always-on Face Detector," *in IEEE JSSC, vol. 53, no. 1, pp. 115~124, 2021.*

Development Trend of DNN Accelerators

Part III - Circuit / Device-level optimization



Weighted sum operation in NeuroSim

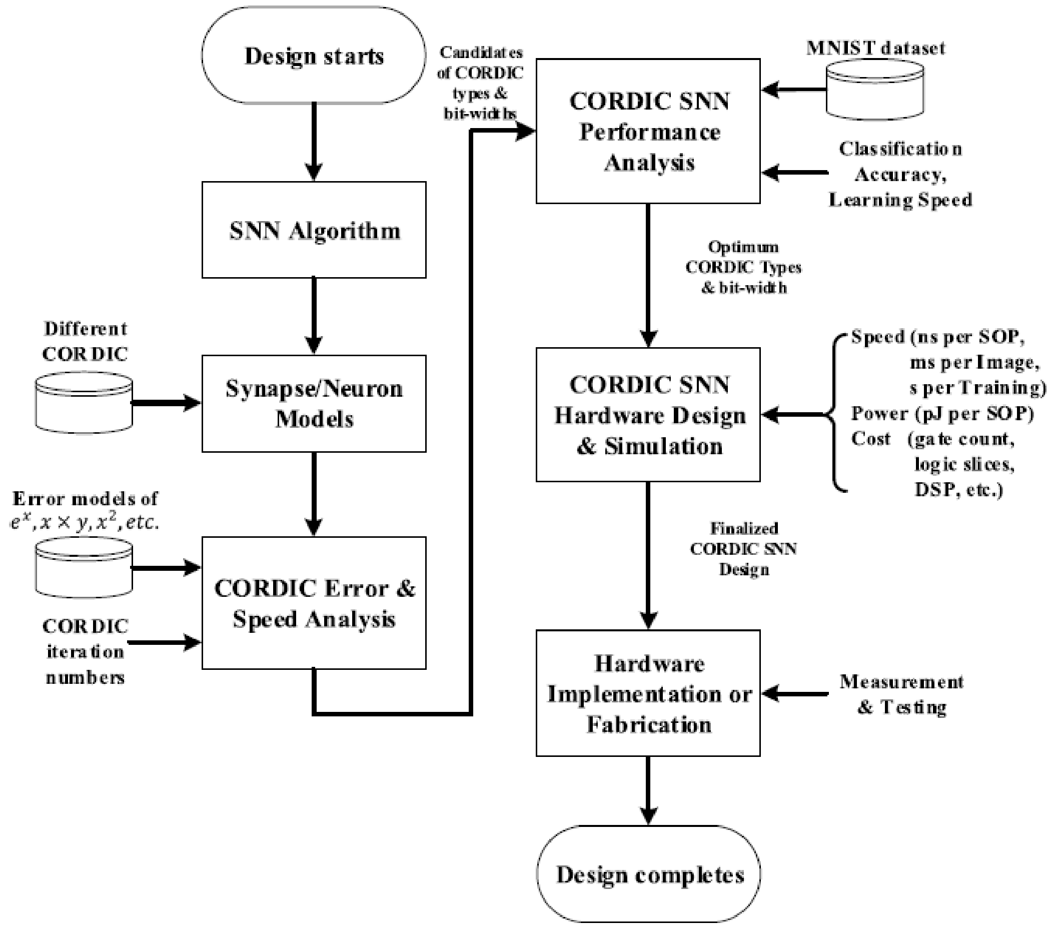


Systematic framework for simulation and evaluation

K. Bong et al., "A Low-Power Convolutional Neural Network Face Recognition Processor and a CIS Integrated With Always-on Face Detector," *in IEEE JSSC, vol. 53, no. 1, pp. 115~124, 2021.*

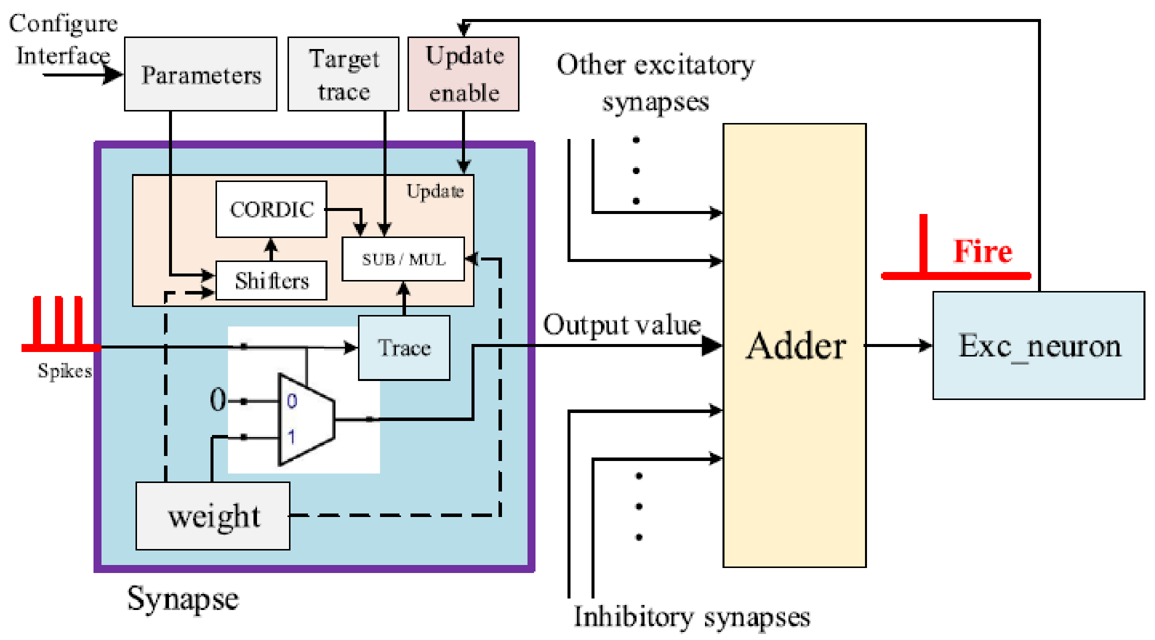
Development Trend of DNN Accelerators

Just a digression - Simulation framework



Design starts
Candidates of CORDIC types & bit-widths
CORDIC SNN Performance Analysis
MNIST dataset
Classification Accuracy, Learning Speed
SNN Algorithm
Optimum CORDIC Types & bit-width
Different CORDIC
Synapse/Neuron Models
CORDIC SNN Hardware Design & Simulation
Speed (ns per SOP, ms per Image, s per Training)
Power (pJ per SOP)
Cost (gate count, logic slices, DSP, etc.)
Error models of $e^x, x \times y, x^2, etc.$
CORDIC Error & Speed Analysis
CORDIC iteration numbers
Finalized CORDIC SNN Design
Hardware Implementation or Fabrication
Measurement & Testing
Design completes

Design and Evaluation Framework of SNN



The proposed CORDIC-based synapse guided by hardware-software co-simulation

Jiajun Wu et al., "Efficient Design of Spiking Neural Network With STDP Learning Based on Fast CORDIC," *in IEEE TCAS-I, vol. 68, no. 6, pp. 2522-2534, June 2021*.

Outline

- Introduction
- Overview of DNN Accelerators
- Case Study - Deep Belief Network Processor
- Broaden Vision - Development Trend of DNN Accelerators
- **Conclusion**
- Q&A

Conclusion

- With the development of Deep Neural Network, it is important to design energy efficient hardware accelerator with on-chip learning due to computation complexity, huge energy consumption, data privacy etc.

- To design an accelerator, researchers should exploit optimizations in algorithm, architecture and circuit.

- In the future, hardware-software co-design methodology is a development trend for accelerators, for instance, architecture-compiler co-design, systematic framework for hardware evaluation.

Acknowledgements

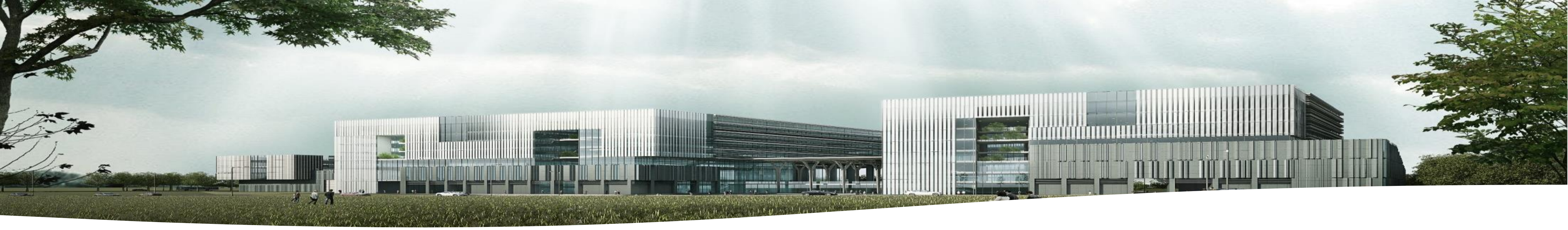
Prof. Chao Wang
Professor, HUST
IEEE Senior Member

Prof. Guoyi Yu
Associate Professor, HUST
IEEE Member

All Members in the Research Lab of Ultra Low-Power and Intelligent Integrated Circuits

My research was supported by:

- Study of Hardware Design of an Emerging Spiking Neural Network based on Statistical Learning, **National Natural Science Foundation of China**.
- ASIC and Software-Hardware Module for Robot Mapping, Location and Navigation, **National Key Research and Development Plan of China.**



Thanks!

Q&A

华中科技大学 HUAZHONG UNIVERSITY OF SCIENCE AND TECHNOLOGY | 光学与电子信息学院 SCHOOL OF OPTICAL AND ELECTRONIC INFORMATION