ETLTC & THE ACM CHAPTER ON ELEARNING & TECHNICAL COMMUNICATION PRESENTS

ETLTC 2022

Robust Cognitive Brain-inspired Computing System: Architectures and Algorithms

Khanh N. Dang, Ph.D.

VNU-key Laboratory for Smart Integrated Systems

Vietnam National University, Hanoi (VNU)

Website: https://khanhdang.github.io/

Email: khanh.n.dang@vnu.edu.vn

Aizu-Wakamatsu, 2022

Content

- Introduction
- Our neuromorphic architecture
  - Neuron
  - Processing core
  - 3D Network-on-Chip Integration
- Algorithm and Application
  - Initial mapping solution with Genetic Algorithm
  - Fault-tolerant mapping with Genetic Algorithm
  - Training SNN with ternary weights
  - Application: Multi security-cores control with SNN
- Conclusion

Content

- Introduction
- Our neuromorphic architecture
  - Neuron
  - Processing core
  - 3D Network-on-Chip Integration
- Algorithm and Application
  - Initial mapping solution with Genetic Algorithm
  - Fault-tolerant mapping with Genetic Algorithm
  - Training SNN with ternary weights
  - Application: Multi security-cores control with SNN
- Conclusion

Brains remain unrivaled computing device

Parrot

Brain
Power: 50mW
Weight: 2.2 grams

Navigates and learns unknown environments at 35km/h

Can learn to speak words

Can learn to manipulate cups to drink

Autonomous Drone

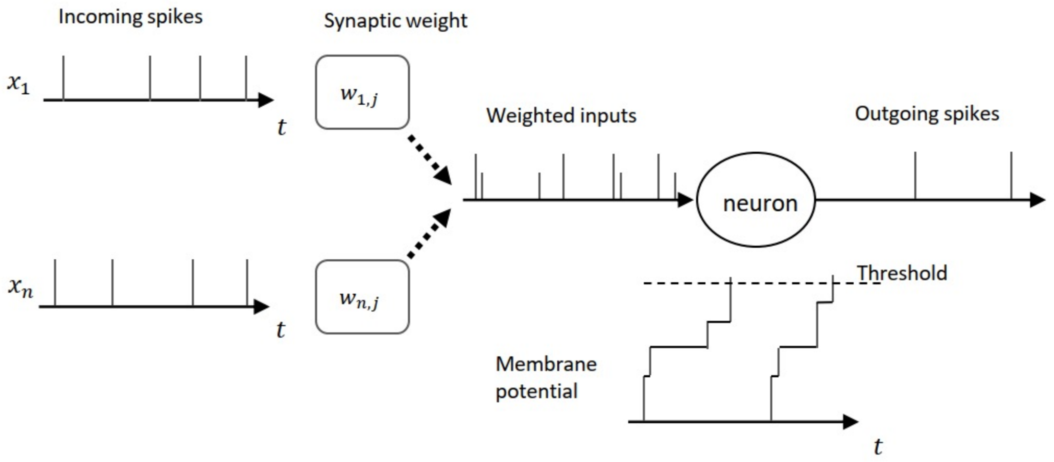
CPU/GPU
Power: 18,000mW
Weight: 40 grams

Pretrained to flight between known gates at walking pace

Cannot learn anything online

Neuromorphic Computing

- Considered as the third generation of neural networks.
- Mimic the operation of biological brain:
  - Neuron communicates via action potentials (spikes),
  - Neuron integrates the action potentials into its membrane potential,
  - Once the membrane potential crosses the threshold, neuron fires (issues spikes)
  - Action potentials travel from upstream neuron to downstream neuron via axon-dendrite-synapse
  - The connections between neurons have different strengths (weights)



Neuromorphic Computing: Benefits

- Near Data/Memory Computing
- Sparse connections (spatial and temporal)
- On-chip learning without weight movement and data storage
- Power and area efficient:
  - No floating-point unit
  - No multiplication (discuss later)
  - No off-chip DRAM

Notable existing works

- Neurogrid by Stanford University:
  - Mixed signal neuron design: capacitor as neuron to integrate the incoming spikes (current).
  - Firing by voltage comparator
- BrainscaleS (European Union project):
  - Also mixed signal neuron design with leaky function (leak current)
  - Hierarchical routing structure
- SpikNNaker by University of Manchester:
  - One million ARM968 cores system
  - Each core simulation 1000 neurons
  - Supercomputer-like structure
- TrueNorth by IBM:
  - 2D-Network-on-Chip based system
  - Integrate and Fire neuron (digital)
  - 256 neuron per node, 256 input, 64k synapses/node
  - 1-bit weight, offline learning
- Loihi by Intel:
  - 2D-Network-on-Chip based system
  - Programmable neuron model (digital)
  - Variable bit width for weight

A comparison between human brain and largescale system

Platform:	Human brain	Neurogrid	BrainScaleS	TrueNorth	SpiNNaker
Technology:	Biology	Analogue, sub-threshold	Analogue, over threshold	Digital, fixed	Digital, programmable
Microchip:		Neurocore	HiCANN		18 ARM cores
Feature size:	10 μm[a]	180 nm	180 nm	28 nm	130 nm
# transistors:		23 M	15 M	5.4 B	100 M
die size:		1.7 cm^2	0.5 cm^2	4.3 cm^2	1 cm^2
# neurons:		65 k	512	1 M	16 k
# synapses:		~100 M	100 k	256 M	16 M
power:		150 mW	1.3 W	72 mW	1 W
Board/unit:		PCB	20 cm wafer	PCB	PCB
# chips:		16	352	16	48
# neurons:		1 M	200 k	16 M	768 k
# synapses:		4 B	40 M	4B	768 M
power:		3 W	500 W	1 W	80 W
Reference system:	1.4 kg		20 wafers in 7 × 19″ racks		600 PCBs in 6 × 19″ racks
# neurons:	100 B		4 M		460 M
# synapses:	10^{15}		1 B		460 B
power:	20 W		10 kW		50 kW
Energy/connection:	10 fJ	100 pJ	100 pJ	25 pJ	10 nJ
Speed versus biology:	1×	1×	10 000×	1×	1×
Interconnect:	3D direct signalling	Tree-multicast	Hierarchical	2D mesh-unicast	2D mesh-multicast
Neuron model:	Diverse, fixed	Adaptive quadratic IF	Adaptive exponential IF	LIF	Programmable[b]
Synapse model:	Diverse	Shared dendrite	4-bit digital	Binary, 4 modulators	Programmable[c]
Run-time plasticity:	Yes!	No	STDP	No	Programmable[d]

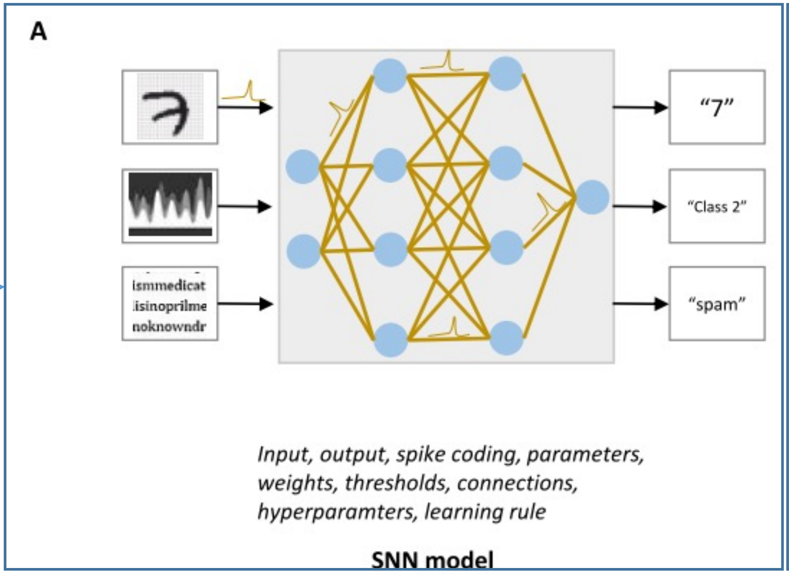
Content

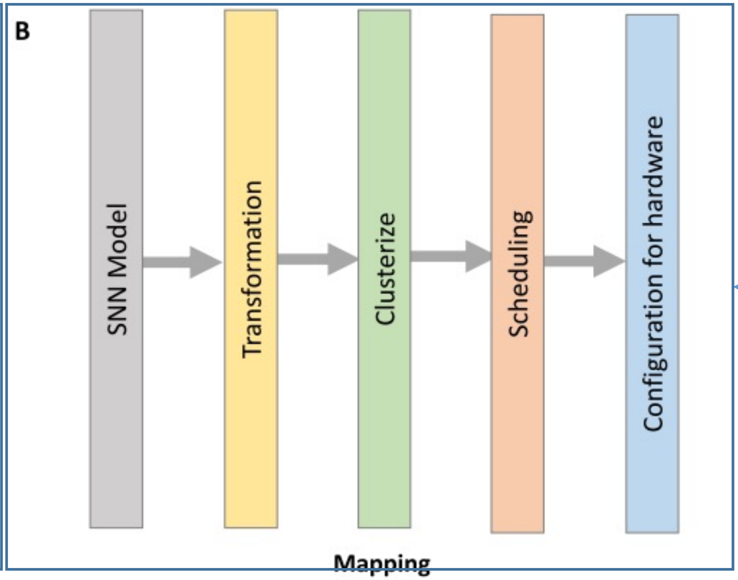
- Introduction
- Our neuromorphic architecture
  - Neuron
  - Processing core
  - 3D Network-on-Chip Integration
- Algorithm and Application
  - Initial mapping solution with Genetic Algorithm
  - Fault-tolerant mapping with Genetic Algorithm
  - Training SNN with ternary weights
  - Application: Multi security-cores control with SNN
- Conclusion

Our neuromorphic platform

- Hardware:
  - A completed neuromorphic system
  - Neuron: Leaky-Integrate-and-Fire (LIF) model in fully parallel mode
  - Synapses: parallel SRAMs
  - Axon: 3D-Network-on-Chip communication
  - Learning: off-chip (ANN-SNN conversion), on-chip (bio-plausible STDP)
- Software:
  - Mapping neuromorphic to hardware using Genetic Algorithm
  - Tolerating faults in neurons using Genetic Algorithm
  - Ternary weight training for MLP and CNN
  - Application: SNN controller for multi AES-cores system

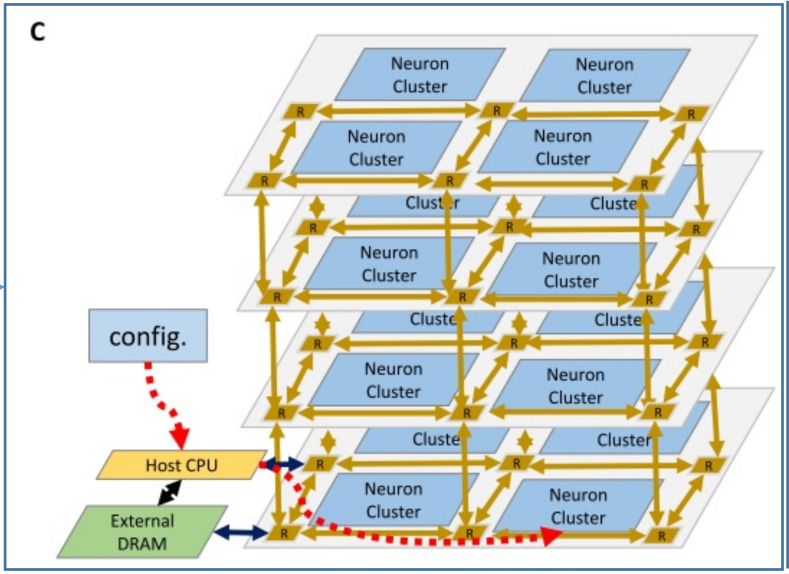
Our platform



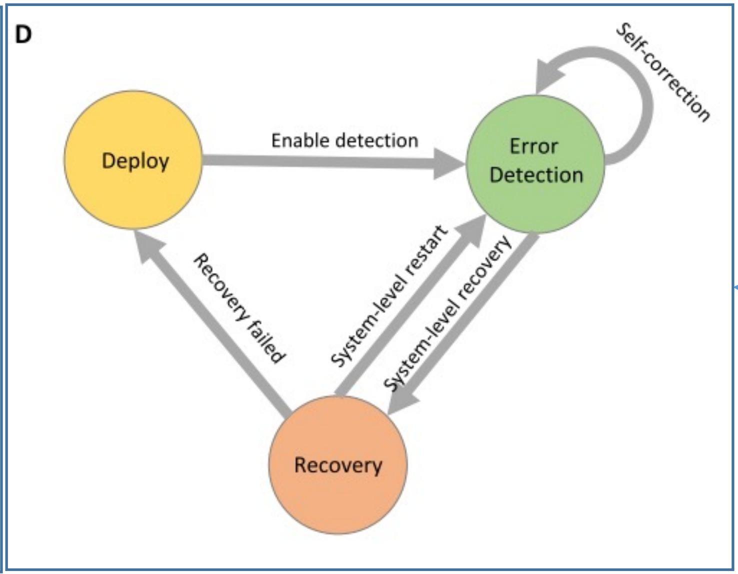


Start with building the SNN model

Map the SNN model into hardware



Neuromorphic Hardware

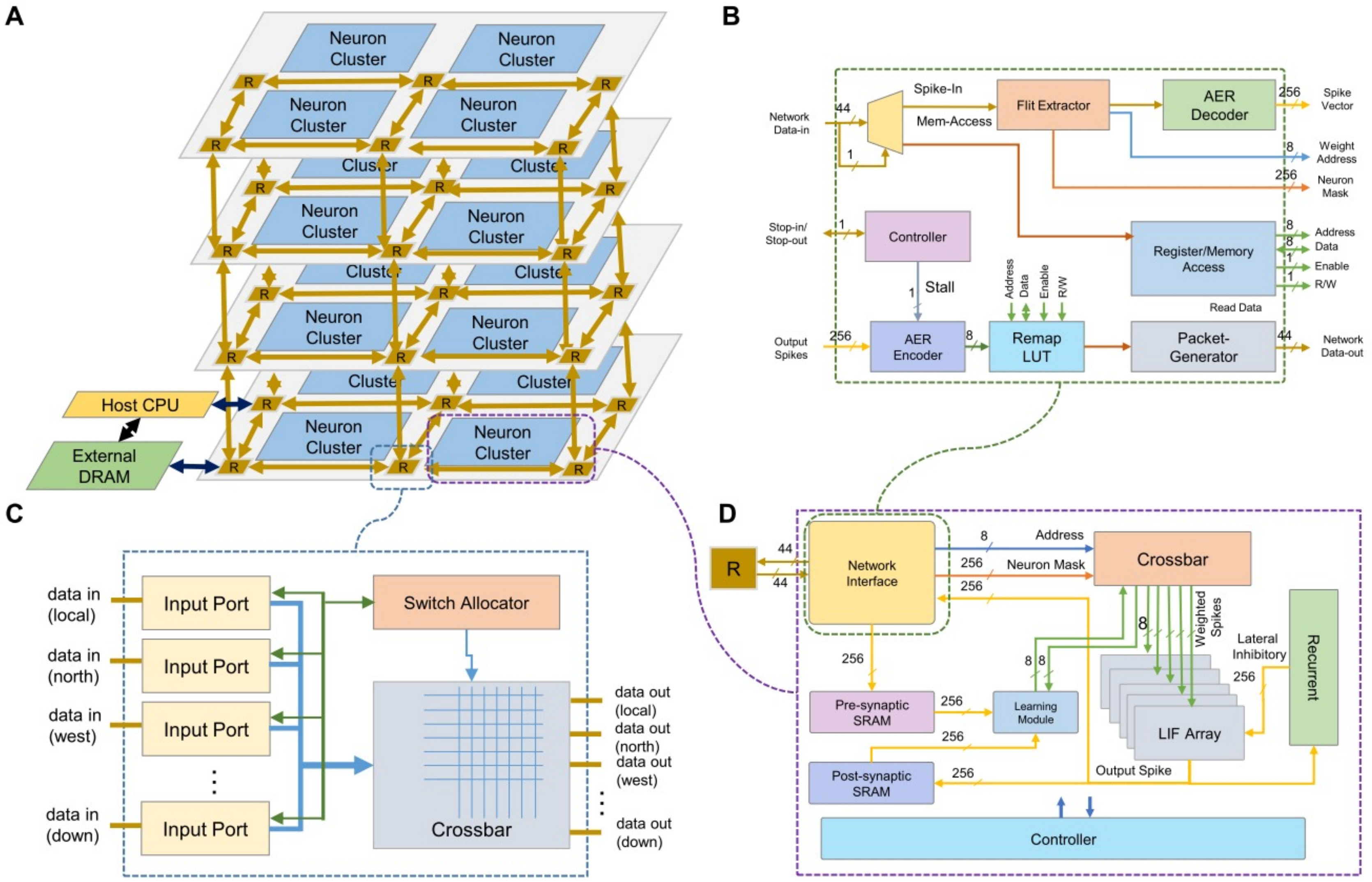


Runtime Maintenance

Deploy into hardware

Maintain the hardware platform

Architecture of the neuromorphic system



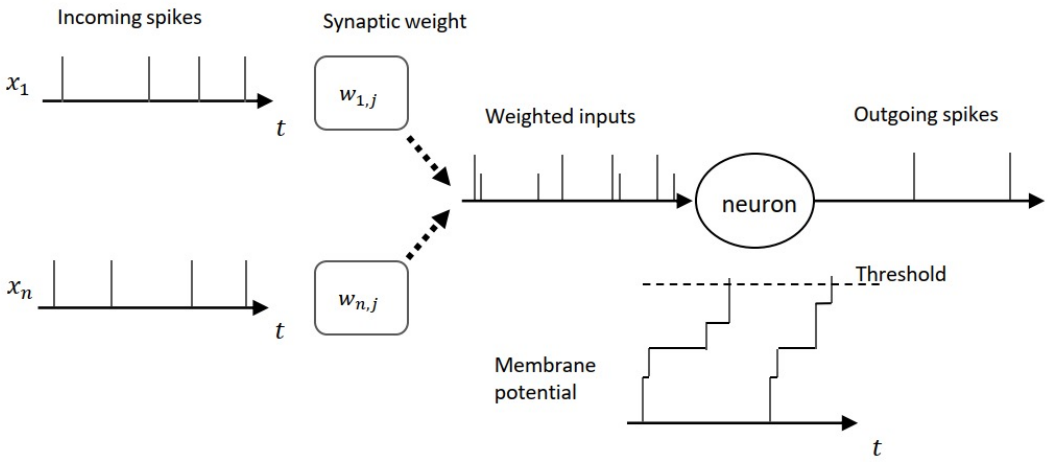
Leaky Integrated and Fire Neuron

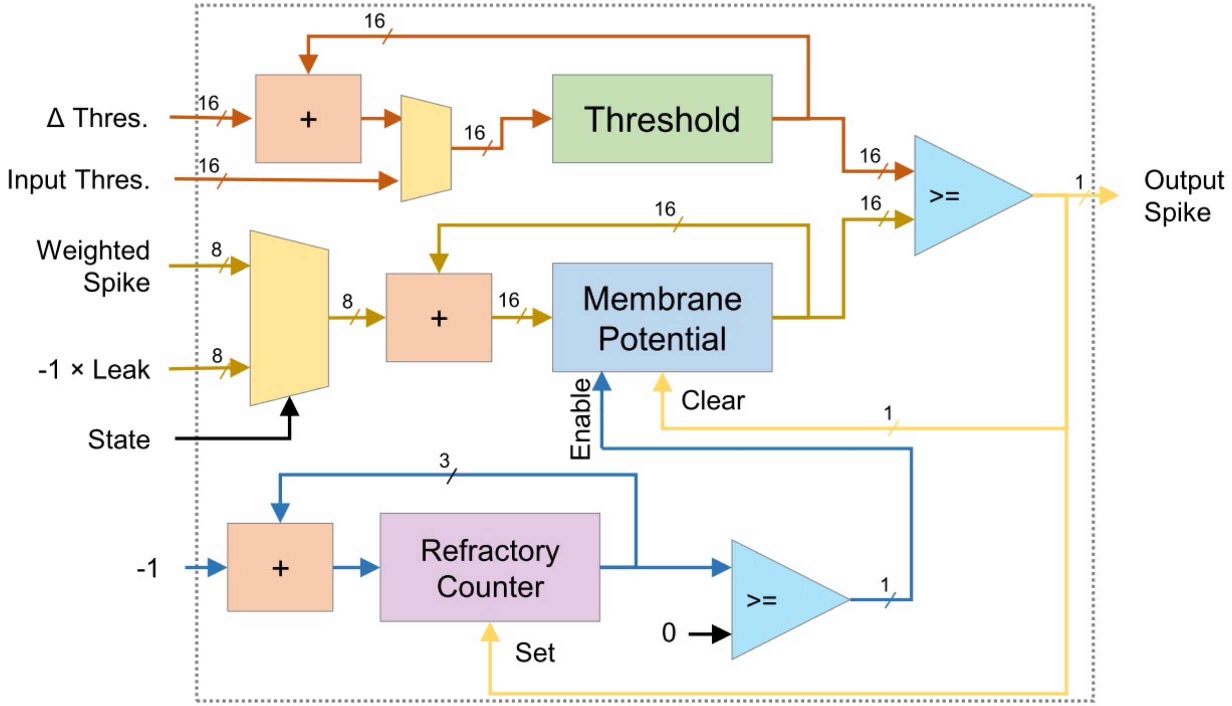
- Computation:

$$V_j(t) = V_j(t-1) + \sum_i w_{i,j} x_i(t-1) - \lambda$$

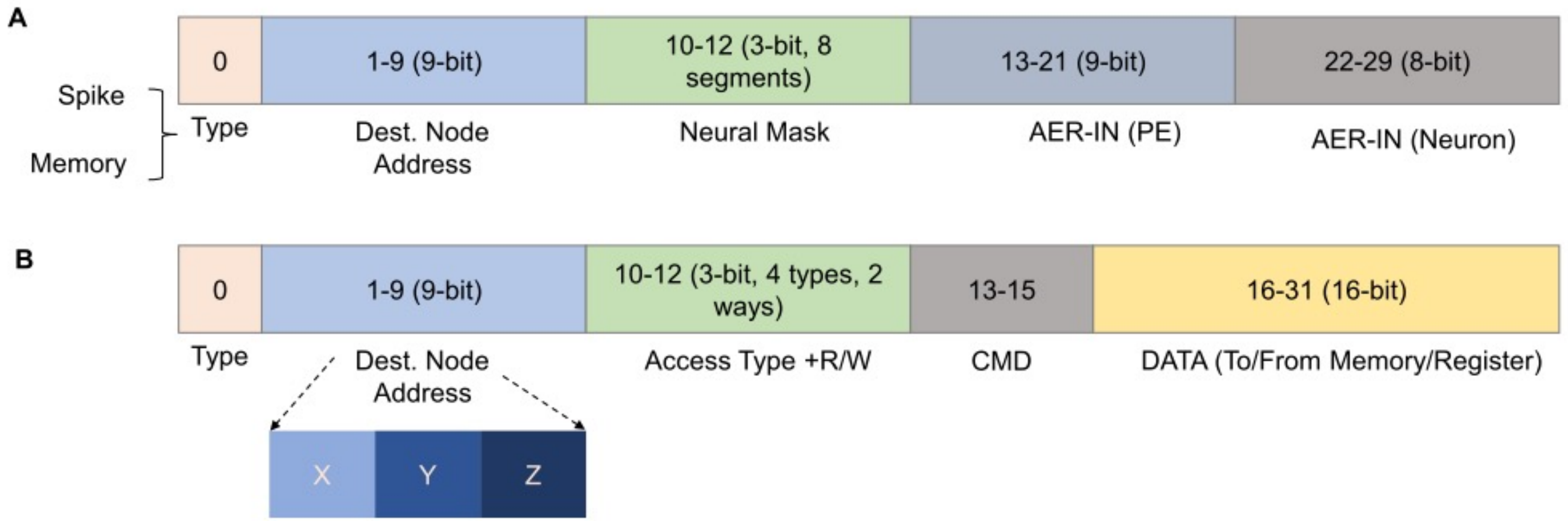
  - $V_j(t)$: the membrane potential of neuron j at time step t
  - $w_{i,j}$: the synapse weight between neuron i and neuron j
  - $x_i(t-1)$: the output of the presynaptic neuron i
  - λ: leaky value.
- The output of a neuron is:

$$x_j(t) = \begin{cases} 1 \; if \; V_j(t) > V_{thres} \\ 0, \qquad otherwise. \end{cases}$$





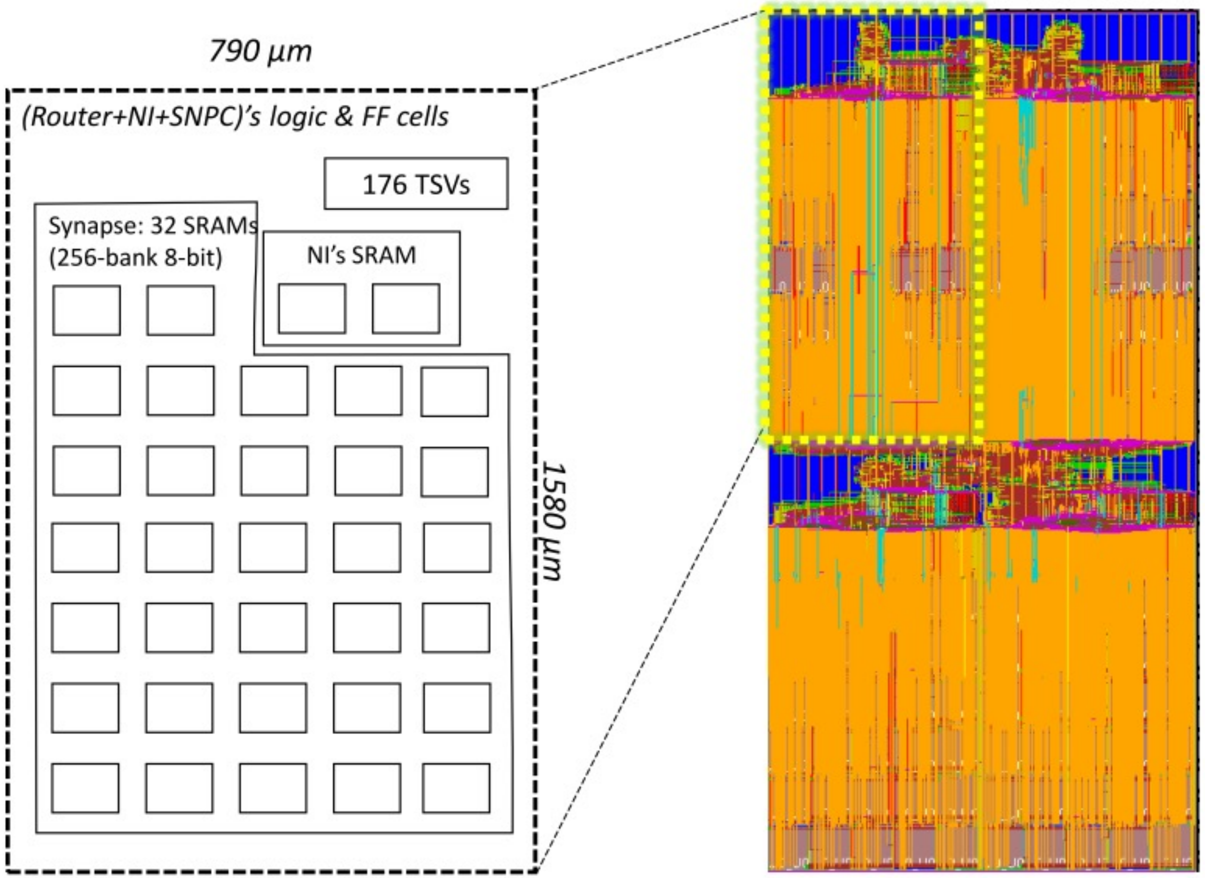
On-chip interconnect



- Use our 3D Network-on-Chip platform: stacking 3D-IC, 3D-Mesh topology, inter-layer using through-silicon-vias (TSV)
- Two types of packets:
  - Spike packet: include the AER for PE (SNPC) and neuron.
    - Spike packet also includes mask for sparsity
  - Memory access: to read and write memory.

Hardware Implementation

- Hardware Architecture:
  - Designed in Verilog HDL
  - Designed with commercial CAD tools with NANGATE45nm
  - TSV integration using FreePDK3D45
  - OpenRAM for SRAM generation



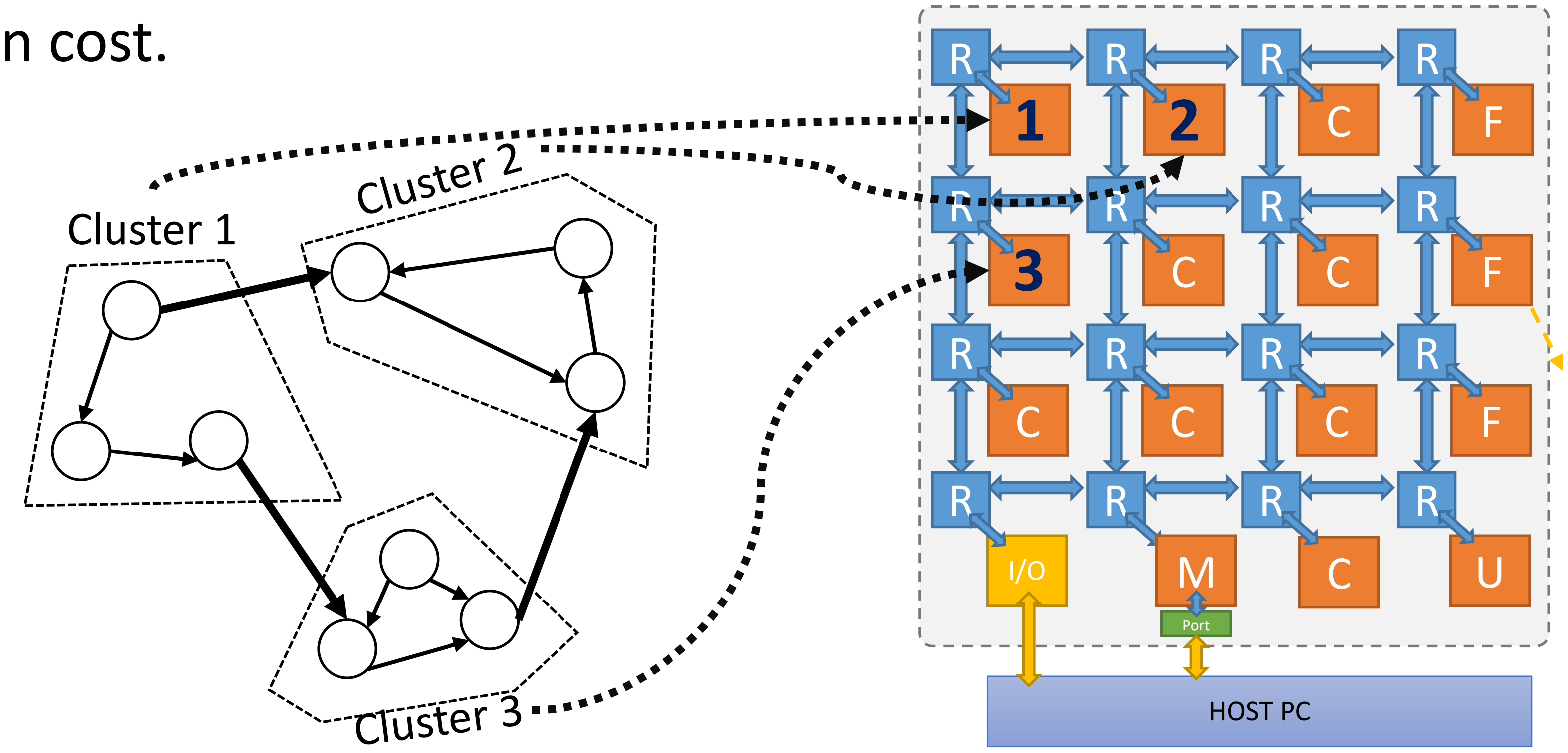
Layout of a 2x2 NoC-based SNN layer with migration support. tile's size is 790µm × 1580µm.

Content

- Introduction
- Our neuromorphic architecture
  - Neuron
  - Processing core
  - 3D Network-on-Chip Integration
- Algorithm and Application
  - Initial mapping solution with Genetic Algorithm
  - Fault-tolerant mapping with Genetic Algorithm
  - Training SNN with ternary weights
  - Application: Multi security-cores control with SNN
- Conclusion

[1] Initial mapping

- When deploying, initial mapping (placing neurons and connecting them) is one of the major problems
  - NoC mapping is NP-completed problem
- We provide a Genetic-Algorithm based solution:
  - Optimize for communication cost.
  - Could be easily extended.



[1] Initial mapping for SNN using Genetic Algorithm

Algorithm 1: Proposed Genetic Algorithm for Neurons Mapping

```
// initialize phase
S1: load the system configuration;
S2: randomize the K mapping solutions;
// evolve phase
for (generation g_i in 1 to G) do
    S3: remove the wrong mapping solutions;
    S4: calculate cost function (communication cost) fo
    S5: select the B best out of K solutions based on t
    S6: mutate the B best solutions to have new K sol
    S6: crossover the new K solutions to have new pop
     S7: check if it satisfies the communication cost or                    ions;
// finalize phase
S7: calculate cost function for each solution of the population;
S8: select the B = 1 best out of K solutions based on the cost function;
```

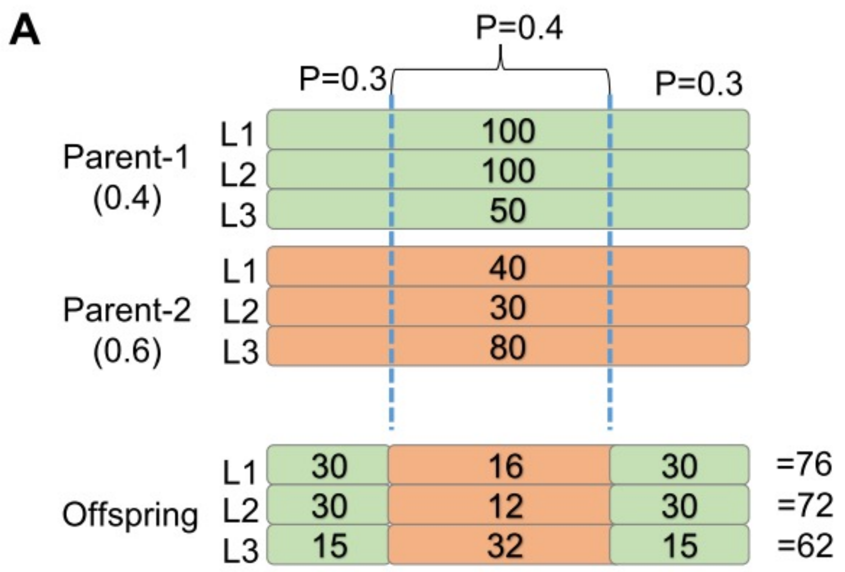
$$F_{cost} = \sum_{i=0,j=0}^{W} d_{ij} \times c_{ij}$$

d_{ij}: distance between neuron i and neuron j

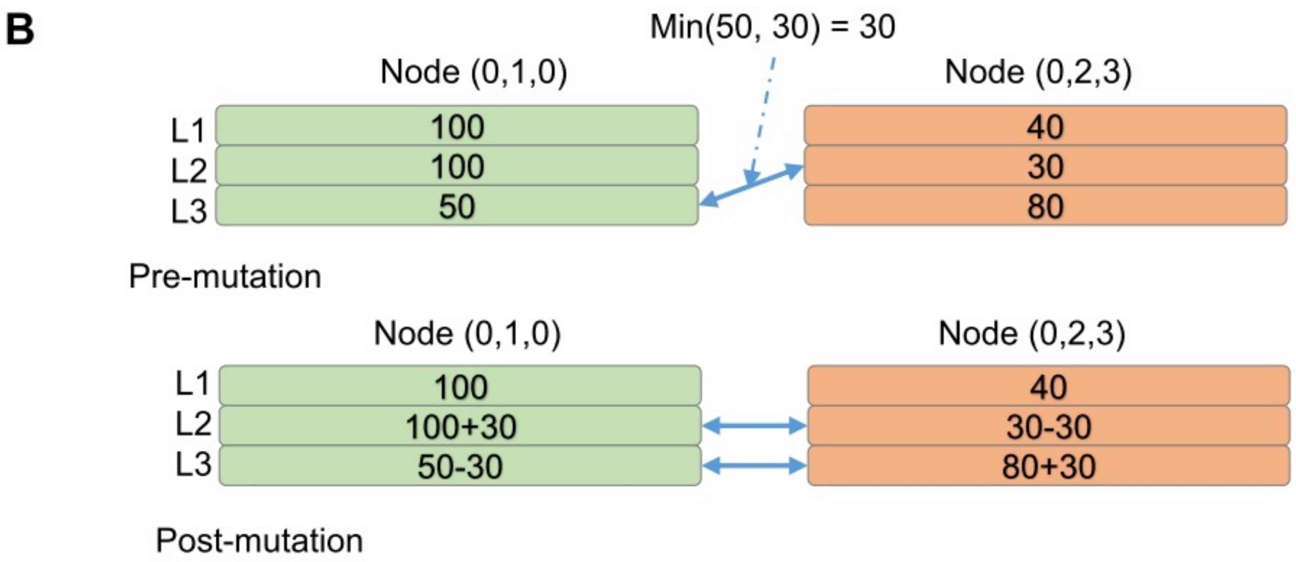
c_{ij}: connection between neuron i and neuron j

Abderazek Ben Abdallah, **Khanh N. Dang**, *“Towards Robust Cognitive 3D Brain-inspired Cross-paradigm System”*, **Frontiers in Neuroscience**, Frontiers, Volume 15, pp. 795, 2021. [DOI: 10.3389/fnins.2021.690208]

Crossover and Mutation

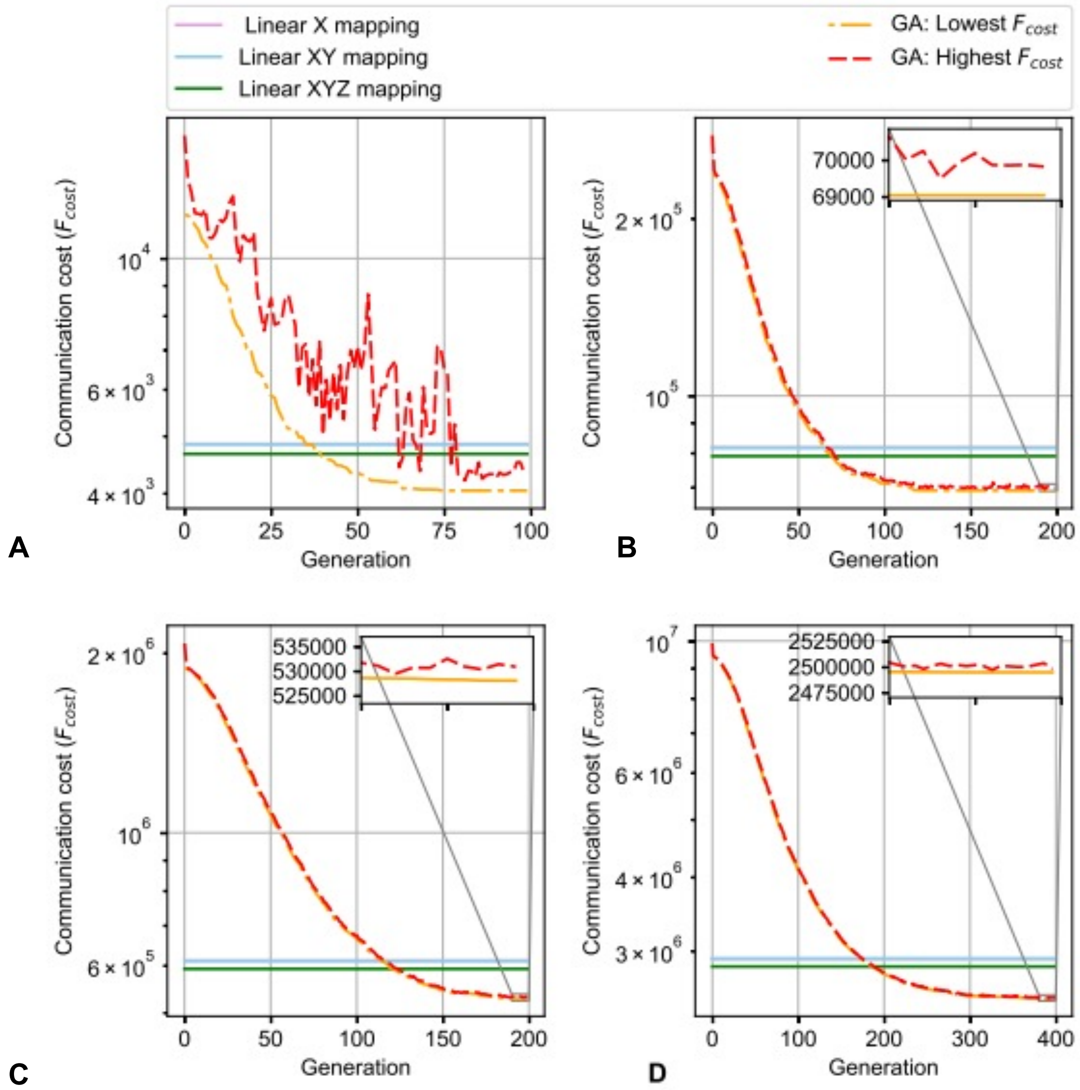


Crossover



Mutation

Initial mapping with GA

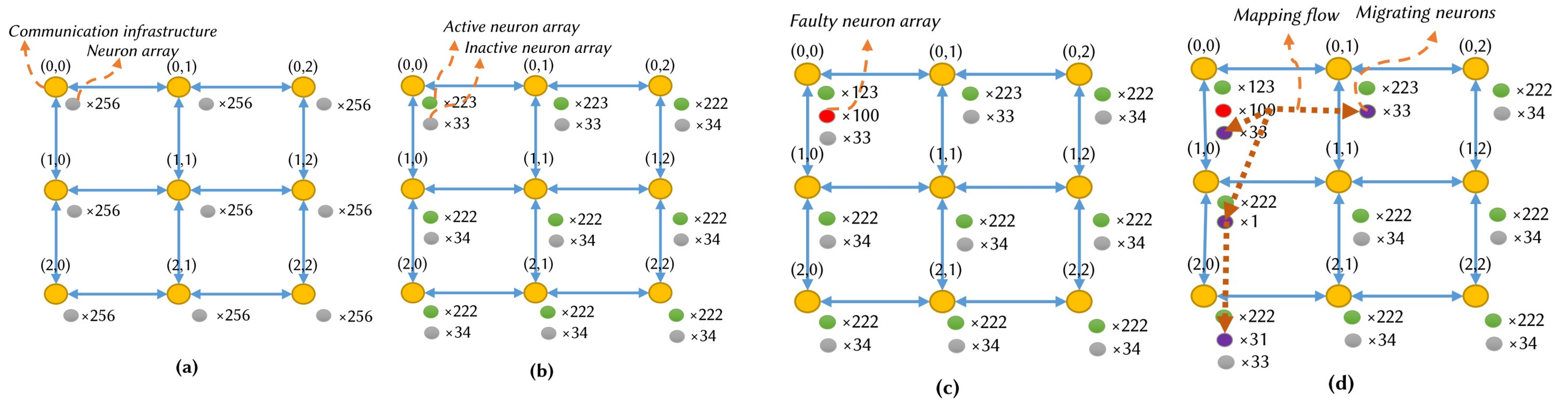


We compare to the linear mapping method from SpiNNaker and implement it for the 3D topology to get Linear X, Linear XY, and Linear XYZ

- Genetic Algorithm Result for initial mapping. (A) 4 × 4 × 4 NoC-based, 256 neurons/node. (B) 6 × 6 × 6 NoC-based, 256 neurons/node. (C) 8 × 8 × 8 NoC-based, 256 neurons/node.(D) 10 × 10 × 10 NoC-based, 256 neurons/node.

[2] Fault-tolerant mapping

- After deploying, there is a probability of neuron failure (faults in memory, neuron or controller).

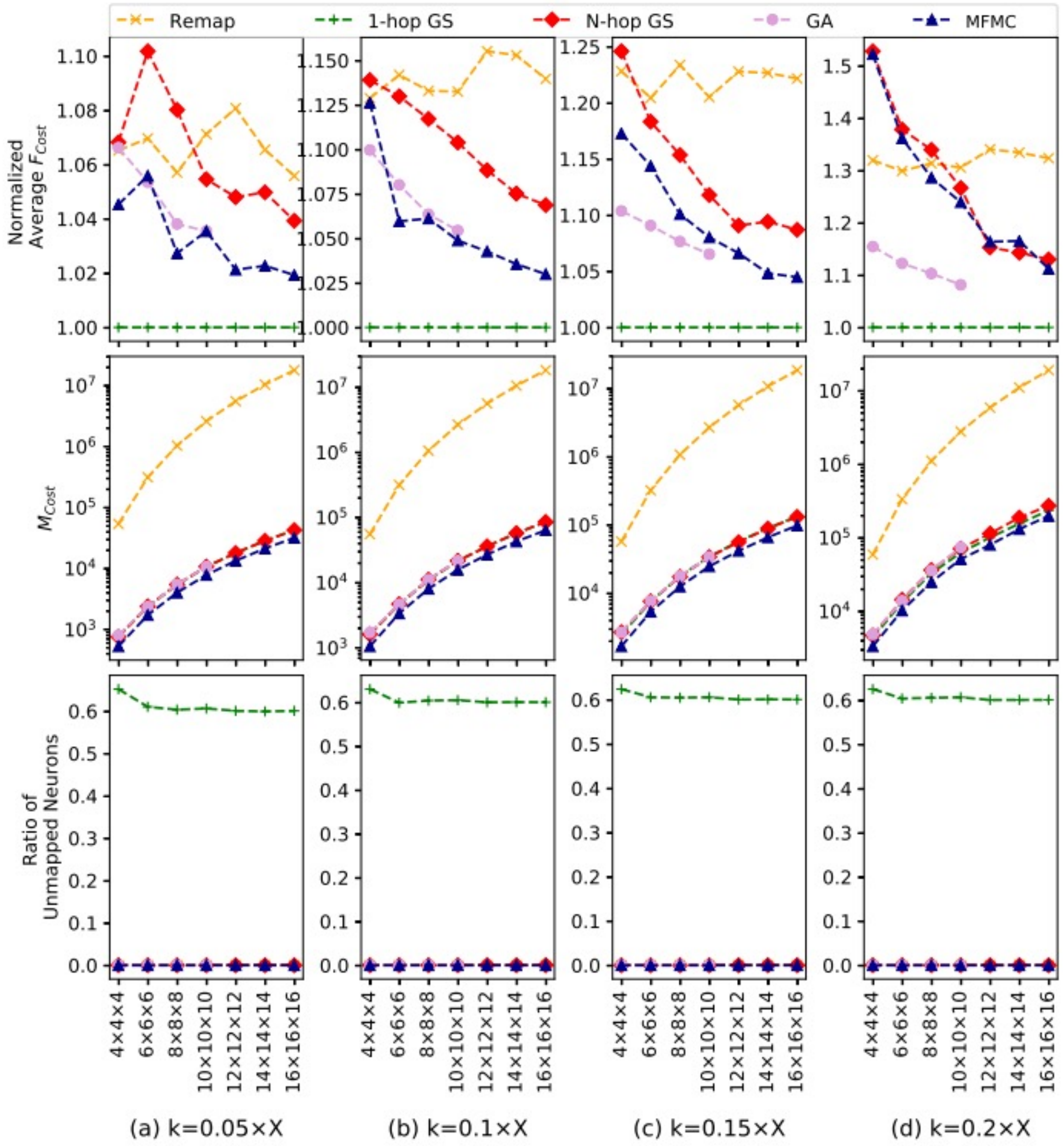


Khanh N. Dang, Nguyen Anh Vu Doan, Abderazek Ben Abdallah *"MigSpike: A Migration Based Algorithm and Architecture for Scalable Robust Neuromorphic Systems"*, **IEEE Transactions on Emerging Topics in Computing (TETC)**, [in-press]() [[DOI: 10.1109/TETC.2021.3136028]()]

[2] Fault-tolerant mapping with GA

- Similar to initial mapping, GA is also used to solve the fault-tolerant mapping.
  - Two cost function: (1) migration cost: movement of neurons during the correction phase and (2) communication cost: travelling distance of spikes
- Crossover is performed by mixing two parents with adjustments to make sure the right amount of neuron being mapped.
- Mutation by reducing the migration cost

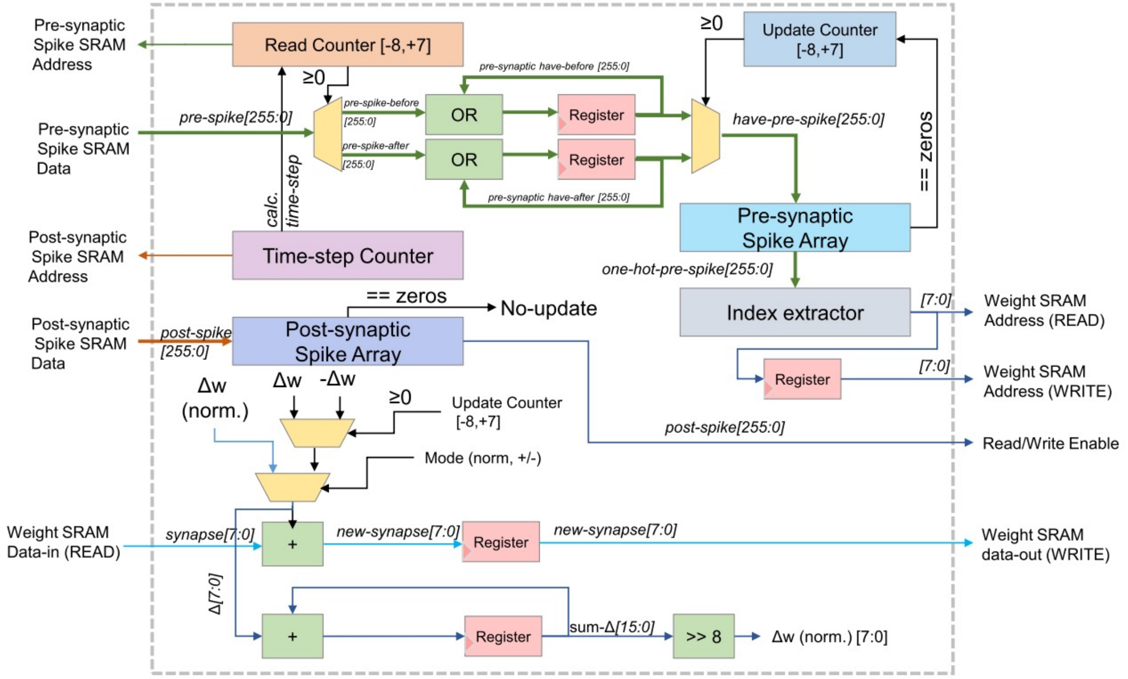
Fault-tolerant Mapping Result



- Output mapping for migrated neurons with random fault patterns in 3D-NoCs. The system has 256 neurons per node; 20% of neurons are spare with 1 redundant node without any allocated neuron at 0% fault rate.

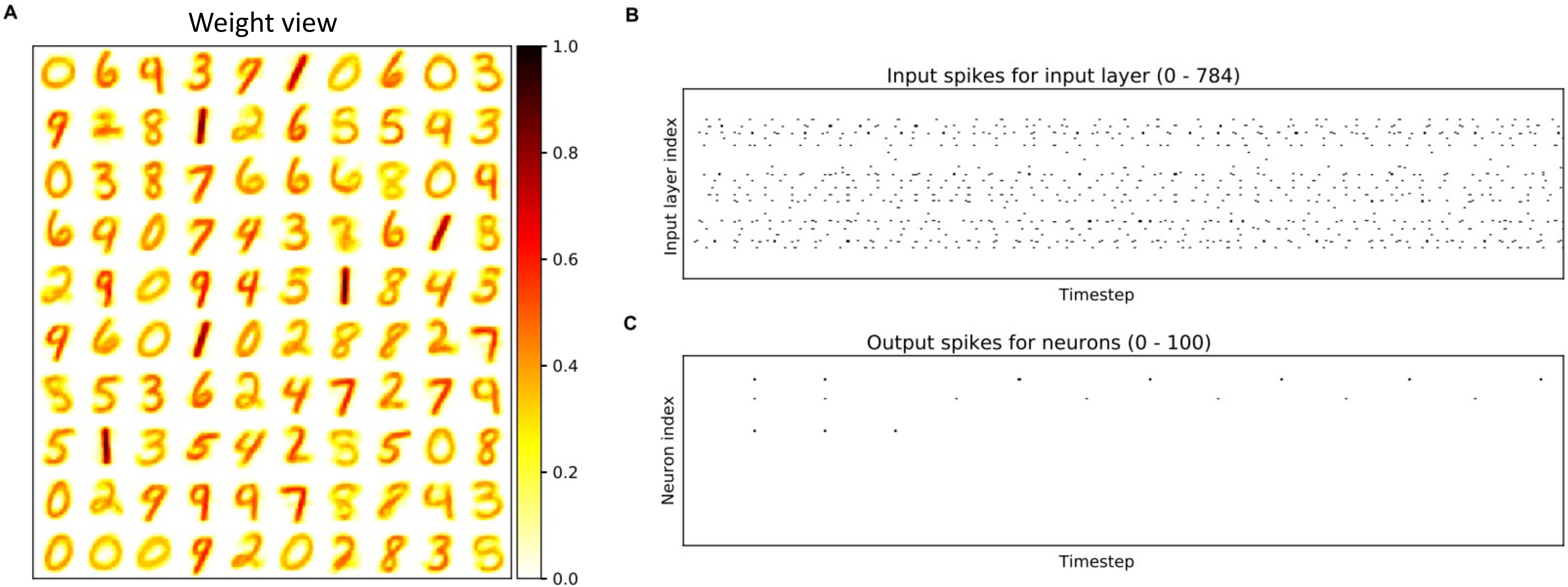
[3] Training for SNN

- Training for Neuromorphic Systems:
  - Spike-timing-dependent plasticity (STDP): a bio-plausible learning method
    - Support online learning with hardware STDP block.
    - Limitation: single-layer (local) training and low accuracy for classification tasks
  - ANN-SNN conversion:
    - Train with ANN first.
    - Convert to SNN by shifting the computation domain.
    - Can be trained with deep neural networks and maintain high accuracy and low power features



Spike-timing-dependent plasticity (STDP) Learning Module

STDP for MNIST



Network: 784:N with lateral inhibitory connections in the second layer.

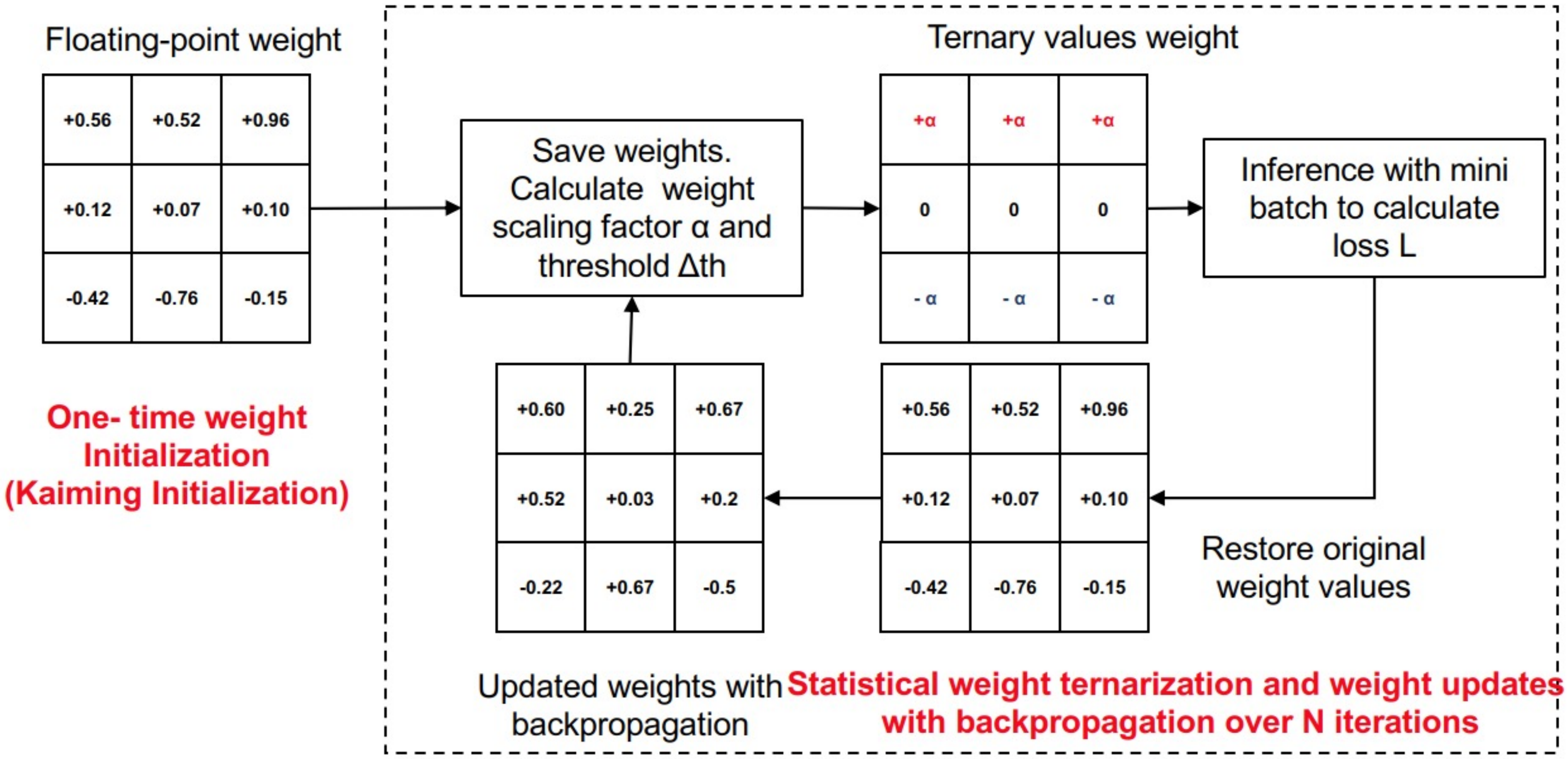
N=100 → accuracy = 71.32%, N=400 → accuracy = 84.05%

[3] Training SNN with ternary weight in ANN-SNN conversion

- Neural networks system usually trained with float values (weight, bias) that requires 32 bit to represent.
- Quantization process can covert these values into fixed point format (8/16 bit) to reduce the storage and computation complexity.
- We can quantize it into ternary value (-α, 0, α) (α is fixed) that require 2-bit to represent. ➔ This ternary process has been done with DNN. We proposed a training method for SNN.

Duy-Anh Nguyen, Xuan-Tu Tran, **Khanh N. Dang**, and Francesca Iacopi, *"A Low-Power, High-Accuracy with Fully On-Chip Ternary Weight Hardware Architecture for Deep Spiking Neural Networks"*, **Microprocessors and Microsystems,** 2022 (in-press).

Training process with ternary weights



Ternary process

- Ternary weight:

$$w_L^{tern} = \begin{cases} \alpha \times Sign\left(w_L^{fp}\right) if \left|w_L^{fp}\right| \geq \Delta_{th} \\ 0 \qquad\qquad\qquad otherwise. \end{cases}$$

- α is the Mean of absolute values:

$$\alpha = E\left(\left|w_L^{fp}\right|\right) \forall\left\{\left|w_L^{fp}\right| \geq \Delta_{th}\right\}$$

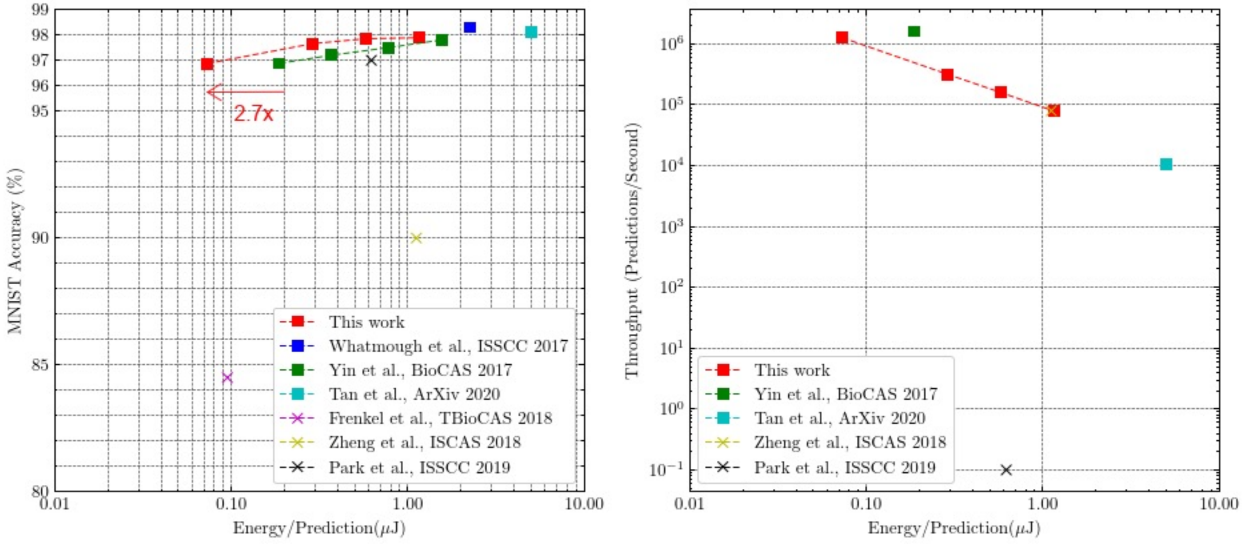
- Threshold:

$$\Delta_{th} = \beta \times \max\left(\left|w_L^{fp}\right|\right)$$

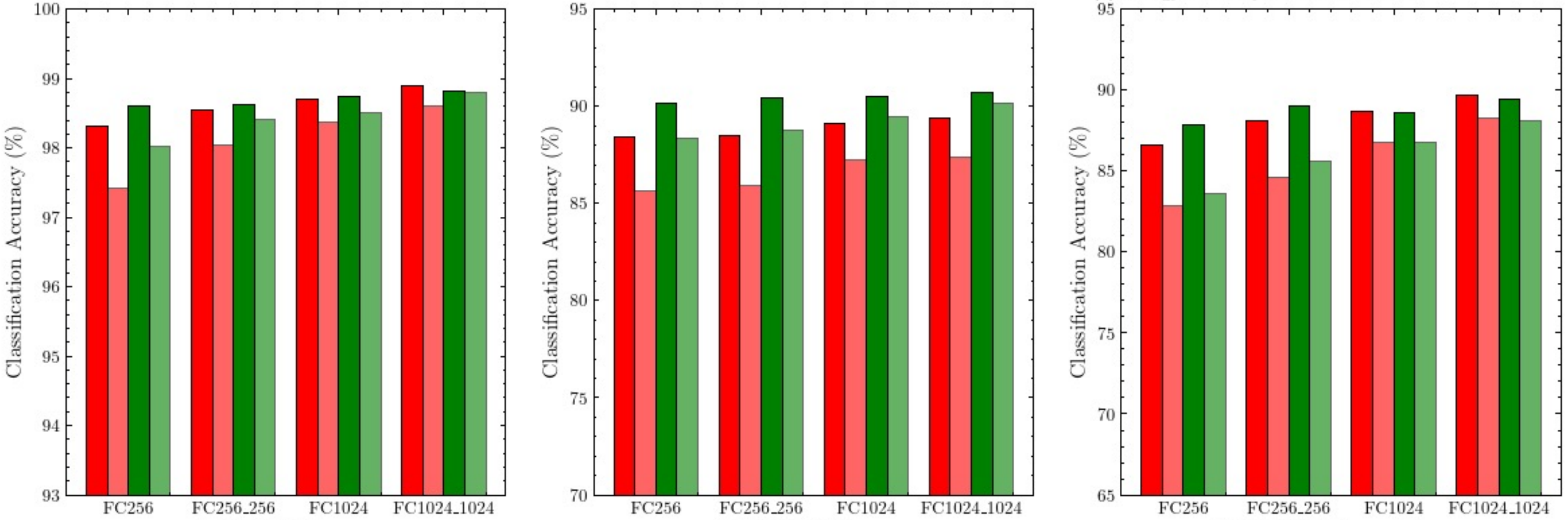
Note: $\beta = 0.01$ in our experience.

Fully Connected Network Models

Network name	Network type	Network configuration	Weight Precision
FC256	Fully Connected	784-256-10	32b
FC256_TW	Fully Connected	784-256-10	2b
FC256_256	Fully Connected	784-256-256-10	32b
FC256_256_TW	Fully Connected	784-256-256-10	2b
FC1024	Fully Connected	784-1024-10	32b
FC1024_TW	Fully Connected	784-1024-10	2b
FC1024_1024	Fully Connected	784-1024-1024-10	32b
FC1024_1024_TW	Fully Connected	784-1024-1024-10	2b



(a) Energy efficiency for MNIST dataset
(b) Energy Efficiency vs Throughput



(a) MNIST dataset (b) FMNIST dataset (c) EMNIST (letters) dataset

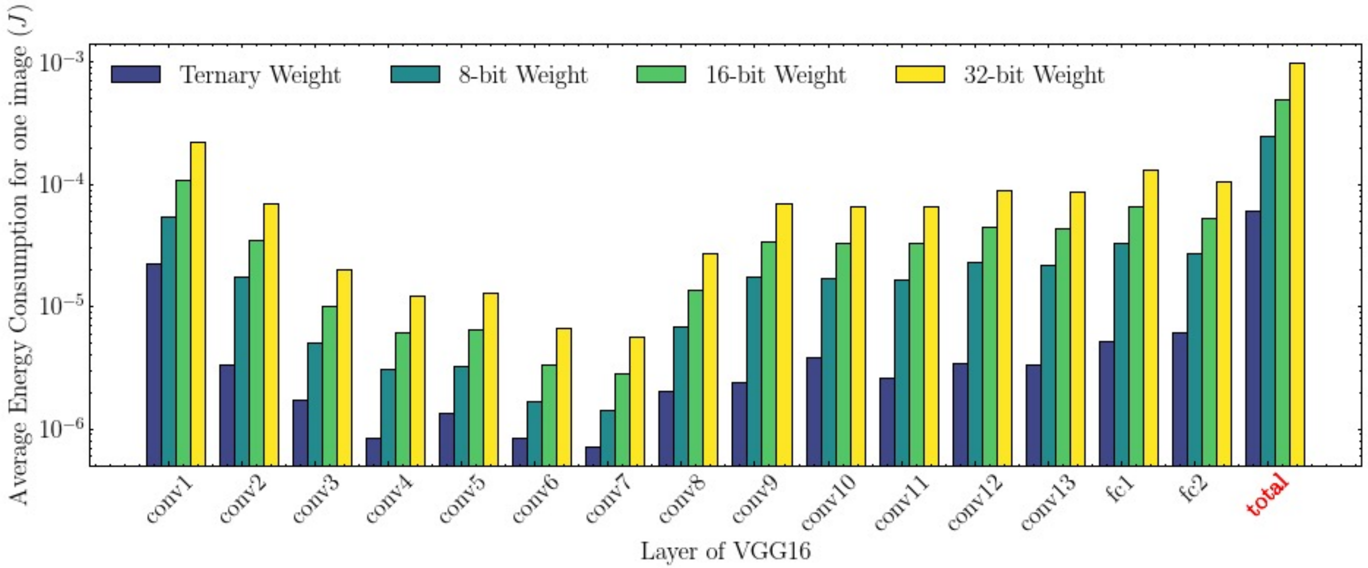
Convolutional Neural Network Models

Architecture	DNN	DNN-SNN (converted, 32b FP)	TW-SNN (2b)
VGG5	88.61%	87.21% (T=250)	85.10% (T=250)
VGG9	90.26%	89.58% (T=250)	87.58% (T=250)
VGG13	91.38%	90.00% (T=250)	89.47% (T=250)
VGG16	91.63%	90.34% (T=250)	89.71% (T=250)

CIFAR-10 dataset

Model	Training method	Architecture	Accuracy	Timesteps
Hunsberger et al. (2015)[56]	DNN-SNN Conversion	2 Conv, 2 Linear	82.95%	6000
Cao et al. (2015)[28]	DNN-SNN Conversion	3 Conv, 2 Linear	77.43%	400
Sengupta et al. (2019)[23]	DNN-SNN Conversion	VGG16	91.55%	2500
Lee et al. (2020)[57]	Spiking Backpropagation	VGG9	90.45%	100
Park et al. (2019)[58]	DNN-SNN Conversion	VGG16	91.41%	793
This work	TW-SNN	VGG16	89.71%	250

Comparison using CIFAR-10 dataset



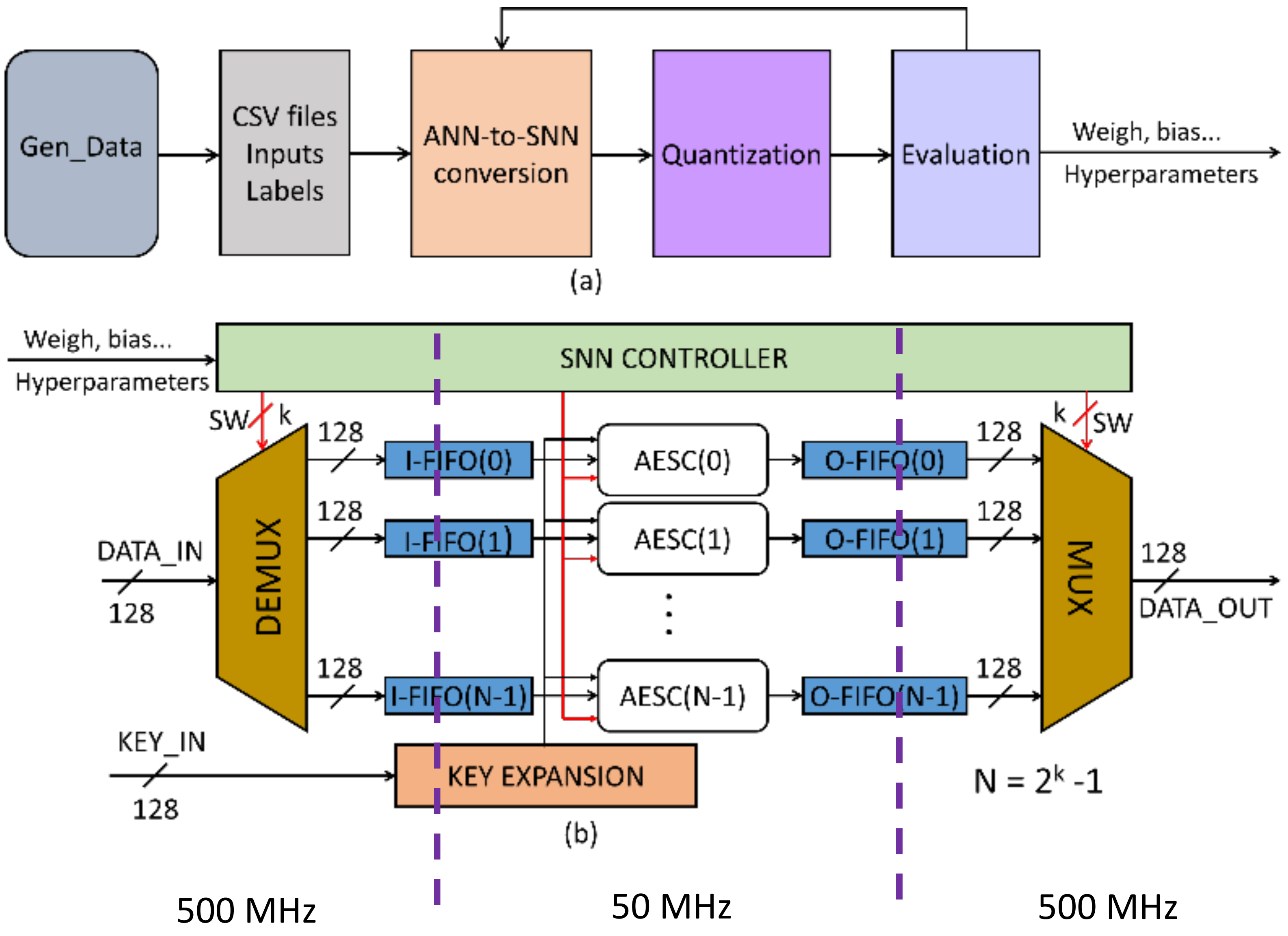
Energy breakdown for VGG16

[4] Application: Multi security-cores control with SNN

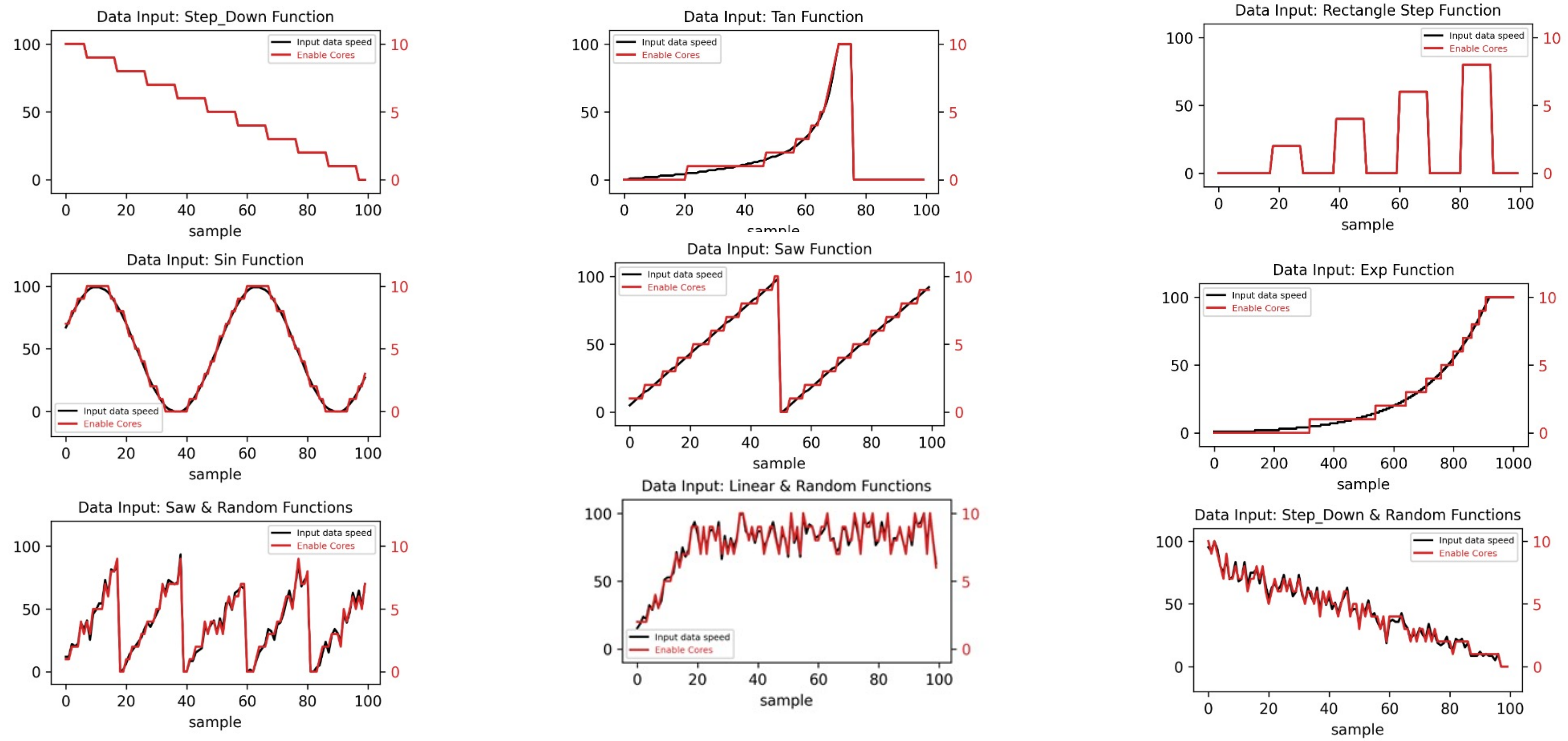
- The baseline slide system consists of 10 AES cores which allow extremely high bandwidth and adaption.
- To reduce the power consumption, we apply clock-gating and power-gating to each core.
- However, the decision process to turn on/turn off a core must be considered:
  - Turn on too many core → waste power consumption.
  - Turn off too many core → cannot satisfy the demanded throughput.
- We design our own dataset for the throughput adaptation and train SNN to control:
  - SNN has low complexity → extremely small controller.
  - Control the system (change the number of cores) in every T cycles
  - Can be turn off to save power.

Dong-Khoi Pham *et al. "A Low-power multicore AES system with neuromorphic controller", (submitted)*

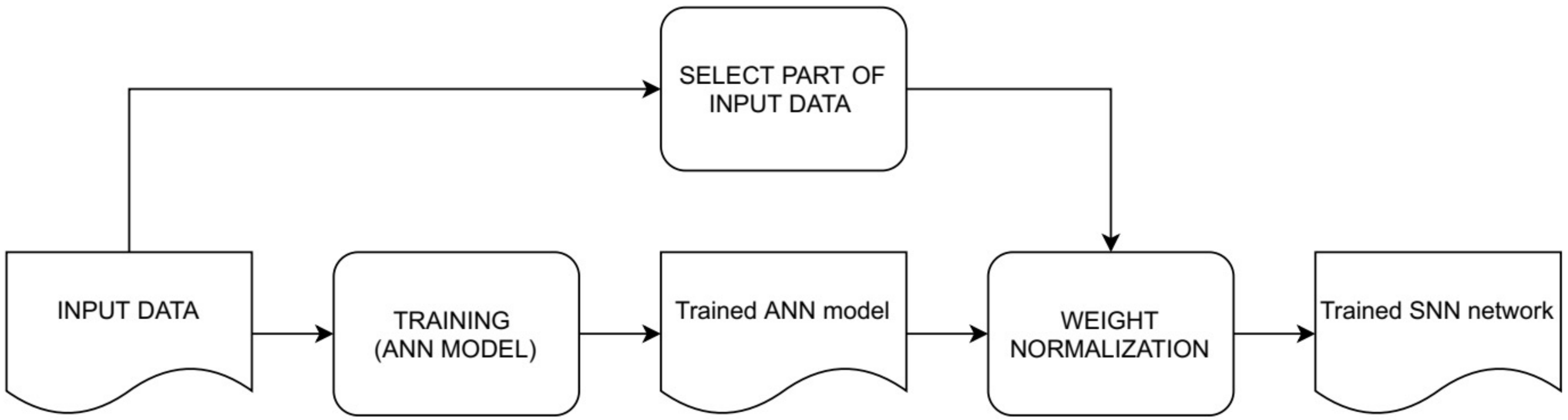
The software/hardware flows



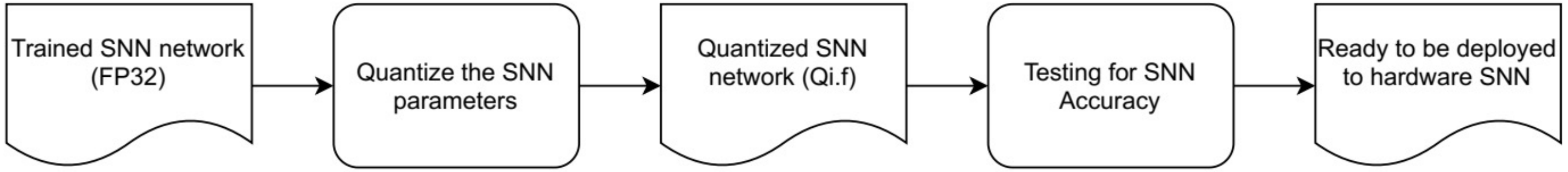
Example of training data (9 out of 27)



SNN training and architecture

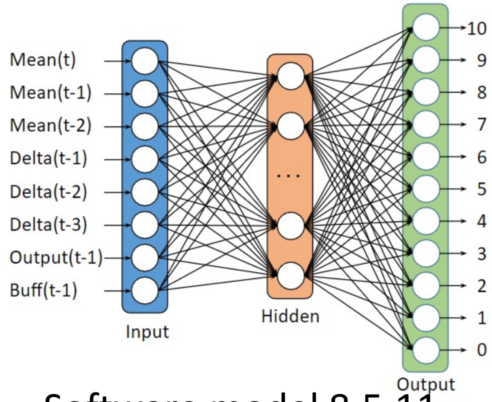


(a)



(b)

SNN training: (a)The ANN-to-SNN conversion flow and (b)The quantization flow (8-bit)



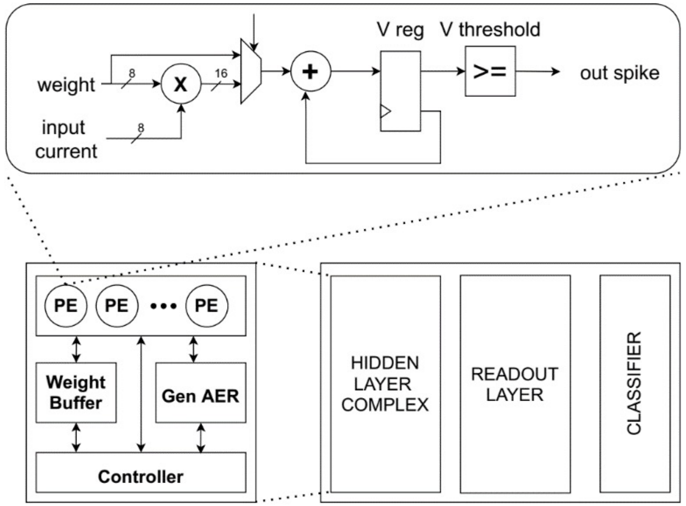
Software model 8:5:11

Mean: average throughput in T cycles

Delta: difference between mean values

Buff: buffer status

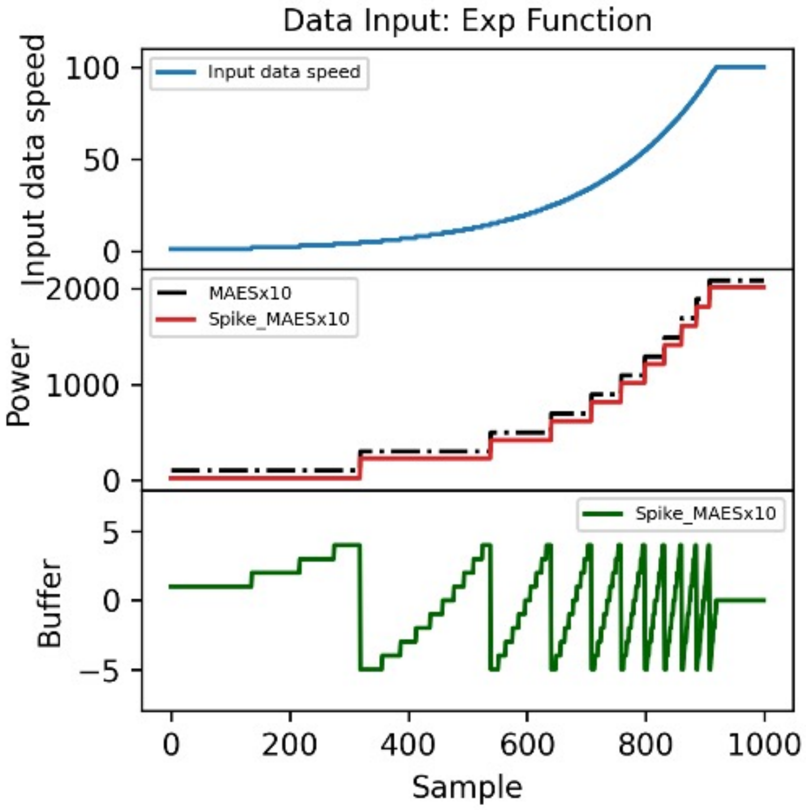
Output (t-1): previous number of core

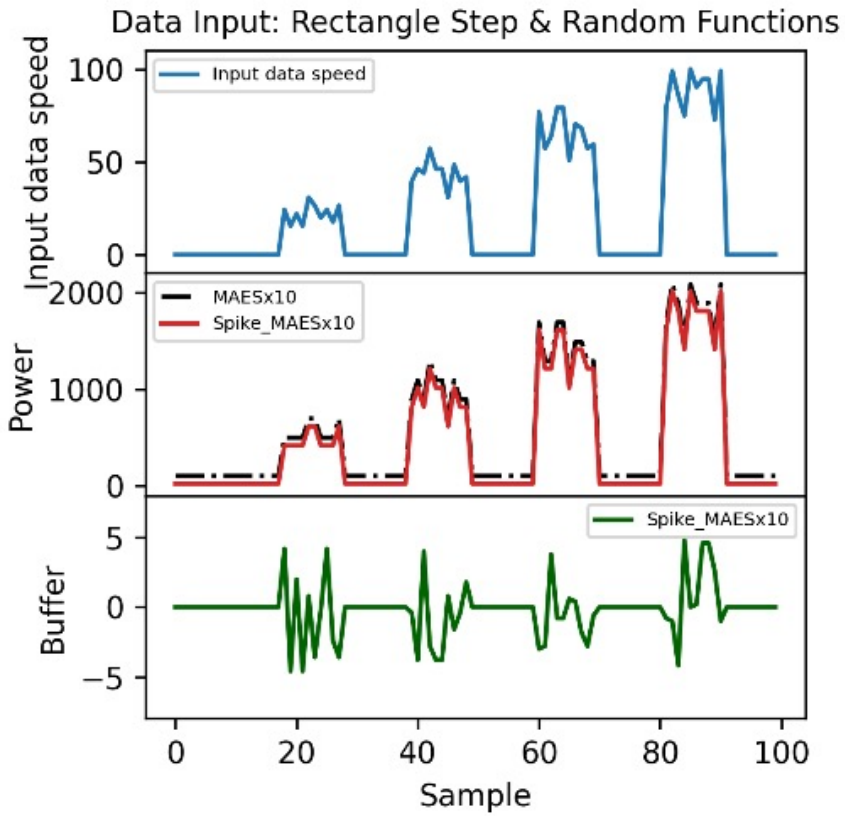


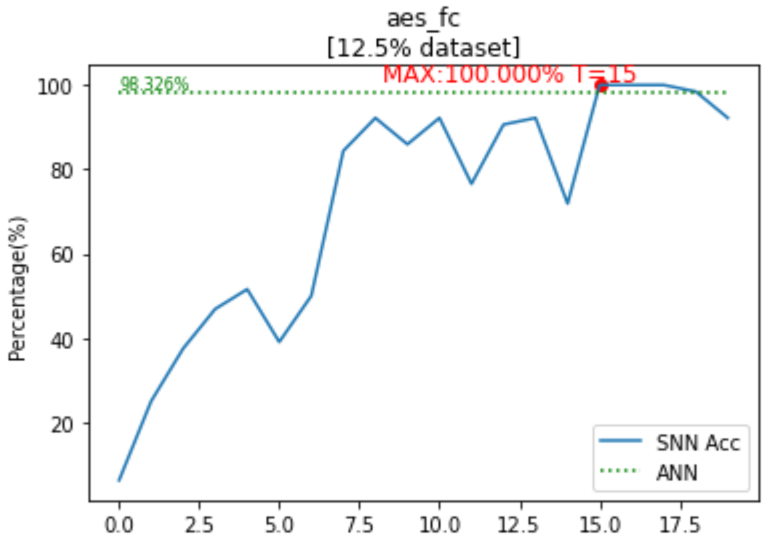
SNN hardware architecture:

SNN control results

- Compare between with and without SNN controller







Floating point training per timestep

Area Cost Full System		
Modules	Area (μm^2)	Percent
System	991,987.4234	100
MAESx10	897903.757	
MAESx10/DEMUX1	35733.641	
MAESx10/MUX	35811.313	
SNN-CONTROLLER	**22,537.3824**	**2.27**

Area cost

8-bit fixed point accuracy: 95%

Error cases are still acceptable with ± 1 from the idea number of core

Content

- Introduction
- Our neuromorphic architecture
  - Neuron
  - Processing core
  - 3D Network-on-Chip Integration
- Algorithm and Application
  - Initial mapping solution with Genetic Algorithm
  - Fault-tolerant mapping with Genetic Algorithm
  - Training SNN with ternary weights
  - Application: Multi security-cores control with SNN
- Conclusion

Conclusion

- We proposed a hardware architecture for neuromorphic computing:
    - Support LIF neuron
    - Clusterize neurons into node
    - Connected via 3D-Network-on-Chip
- We support offline and online training for the neuromorphic system.
- Augmented algorithms:
    - Initial mapping with Genetic Algorithm
    - Fault-tolerant mapping Genetic Algorithm.
- Future works:
    - Advanced memory technology: RRAM, STT-RAM, PWM.
    - Advanced 3D-IC: monolithic 3D.

Reference

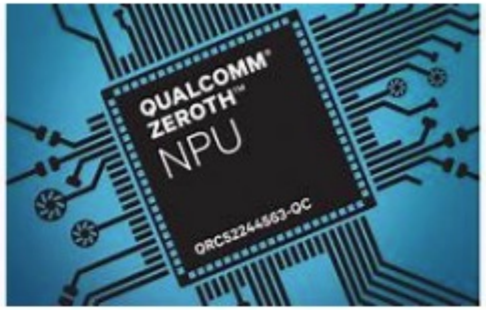
- H. Hazan et al., “BindsNET: A machine learning-oriented spiking neural networks library in Python,” Frontiers in Neuroinformatics, vol. 12, p. 89, 2018.
- F. Akopyan et al., “TrueNorth: Design and tool flow of a 65 mW 1 million neuron programmable neurosynaptic chip,” IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems, vol. 34, no. 10, pp. 1537–1557, Oct 2015.
- M. Davies et al., “Loihi: A neuromorphic manycore processor with on-chip learning,” IEEE Micro, vol. 38, no. 1, pp. 82–99, January 2018.
- S. B. Furber et al., “The SpiNNaker project,” Proceedings of the IEEE, vol. 102, no. 5, pp. 652–665, May 2014
- B. V. Benjamin et al., “Neurogrid: A mixed-analog-digital multichip system for large-scale neural simulations,” Proceedings of the IEEE, vol. 102, no. 5, pp. 699–716, May 2014.
- K. Banerjee et al., “3-D ICs: A novel chip design for improving deep-submicrometer interconnect performance and systems-on-chip integration,” Proc. IEEE, vol. 89, no. 5, pp. 602–633, 2001.
- P. U. Diehl and M. Cook, “Unsupervised learning of digit recognition using spike-timing-dependent plasticity,” Frontiers in computational neuroscience, vol. 9, p. 99, 2015.
- P. U. Diehl et al., “Fast-classifying, high-accuracy spiking deep networks through weight and threshold balancing,” in 2015 International Joint Conference on Neural Networks (IJCNN), July 2015, pp. 1–8.

Thank you for your attention!

Backup Slides

Just in case ;)

Notable existing works

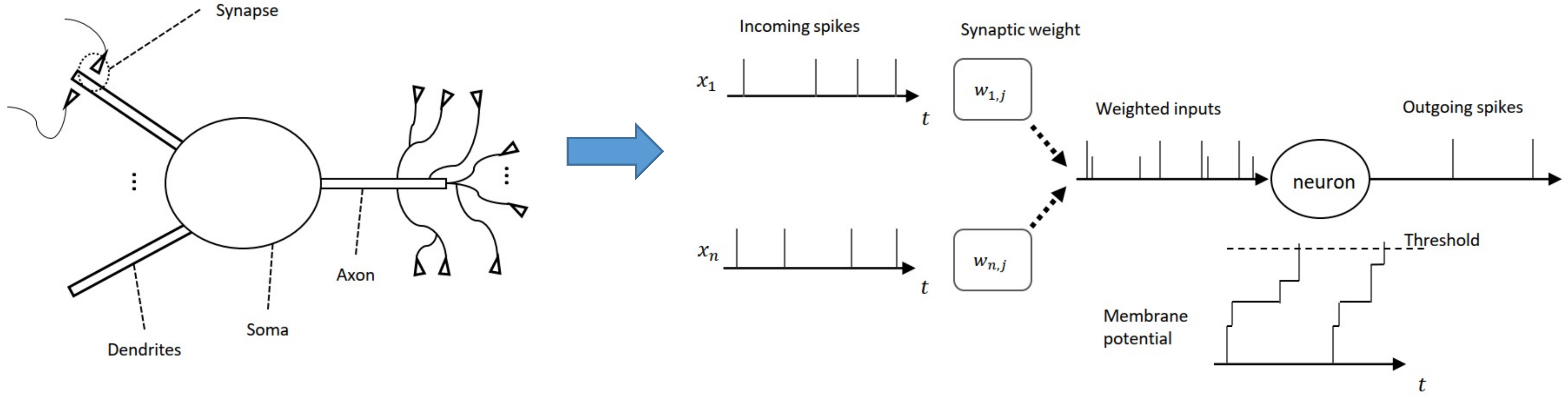


Qualcomm Zeroth (2013)

IBM TrueNorth (2014)

Intel Loihi (2017)

BrainChip (2019)



Learning for neuromorphic system

- Offline learning with backpropagation trained ANN-to-SNN conversion:
  - Feed-forward network is trained with back-propagation.
  - Weight normalization and conversion is performed.
  - Weights and parameters are exported to fixed bit format.
  - Deploy by downloading the weights and parameters.
- Online learning with STDP (Spike-Timing-Dependent Plasticity):
  - Follow the Hebbian learning rule and divide into Long-Term-Depression (LTD) and Long-Term-Potentiation (LTP)
  - LTP: if the spike on the synapse occurs before firing (cause the firing), the strength of synapse increase.
  - LTD: if the spike on the synapse occurs after firing, the strength of synapse increase.
  - Supported by hardware STDP block.

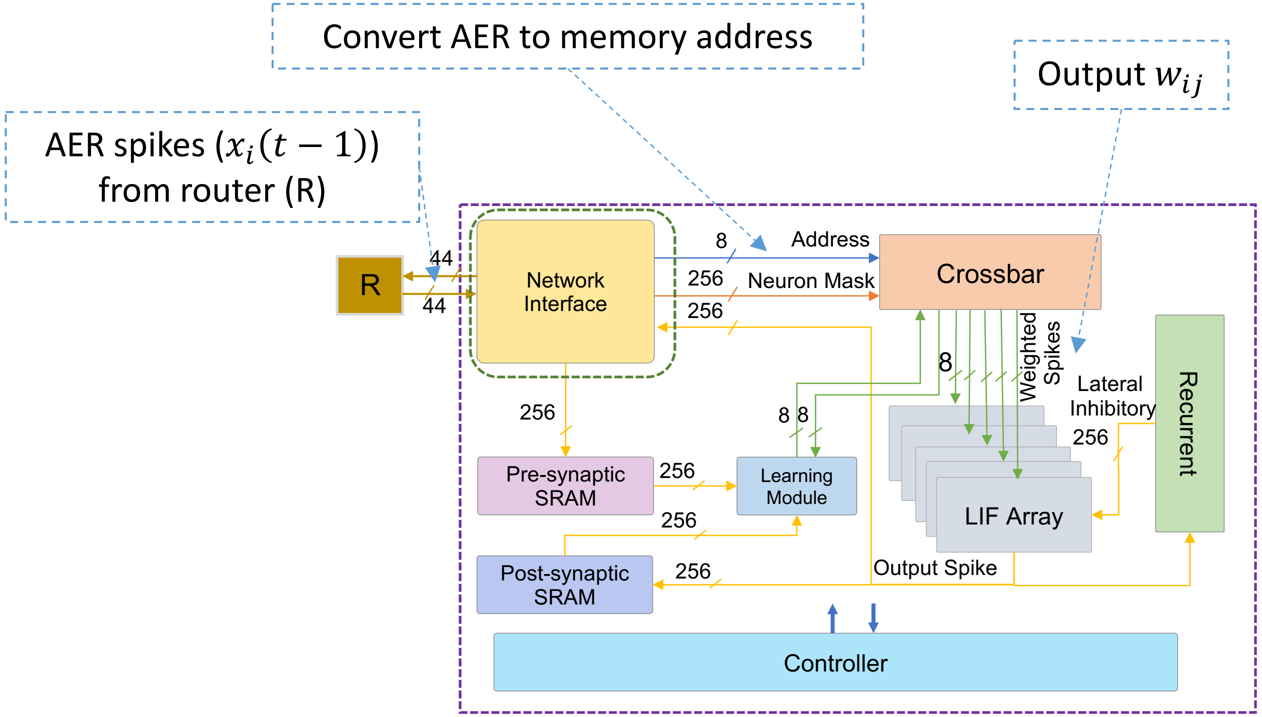
Near Data Processing Approach

$$V_j(t) = \underbrace{V_j(t-1)}_{\substack{\text{Membrane}\\ \text{potential}\\ \text{(on register)}}} + \underbrace{\sum_i w_{i,j} x_i(t-1)}_{\substack{\text{Weighted spike}\\ \text{(input)}}} - \underbrace{\lambda}_{\text{Leaky (fixed)}}$$

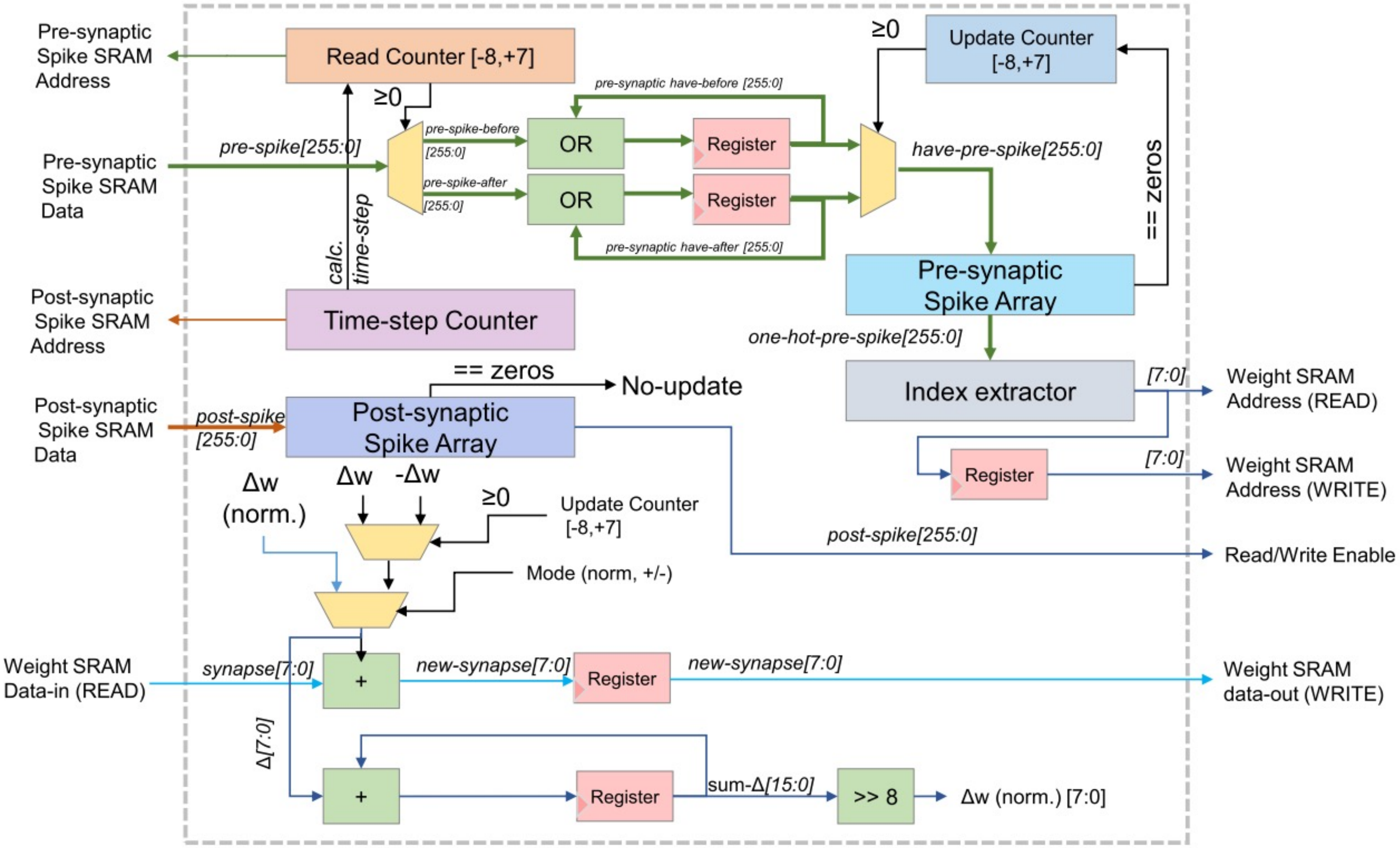
- The “integration” and “leaky” of LIF neuron is performed by an adder
- The most challenging part is the weighted input (product of input spikes and weight)
- Conventional ANN uses multipliers as a part of its MAC (multiplication and accumulation)

Near Data Processing Approach (cnt.)

- In our design, multiplication is converted to memory accessing
- $x_i(t-1)$, as binary input, is converted to memory access.
- If $x_i(t-1)$ = 0, no access (return 0)
- If $x_i(t-1)$ = 1, By accessing the address of $w_{i,j}$, we can extract the value of $w_{i,j}x_i(t-1)$

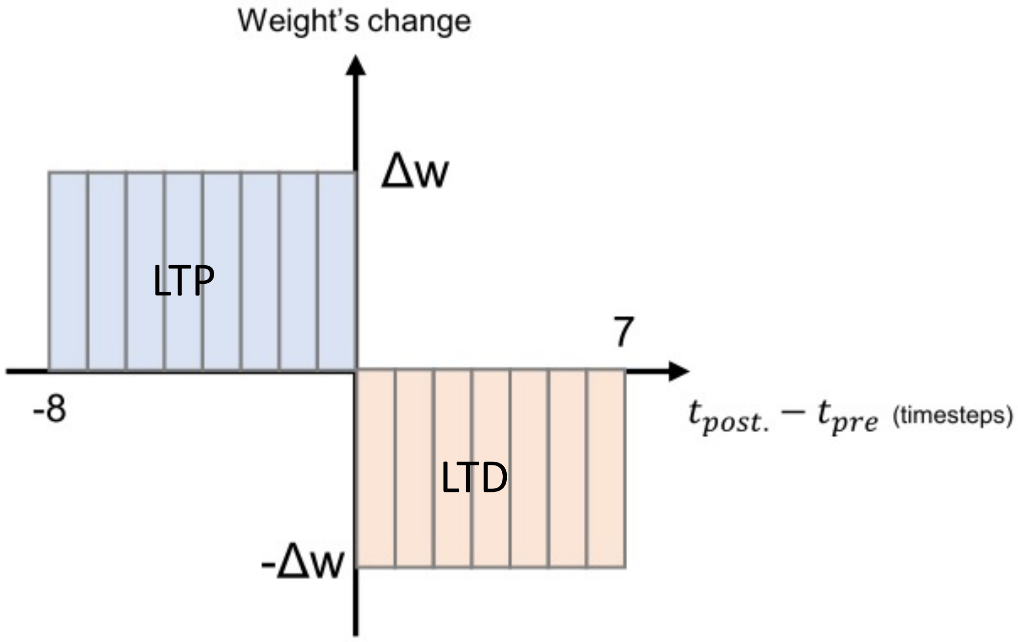


STDP



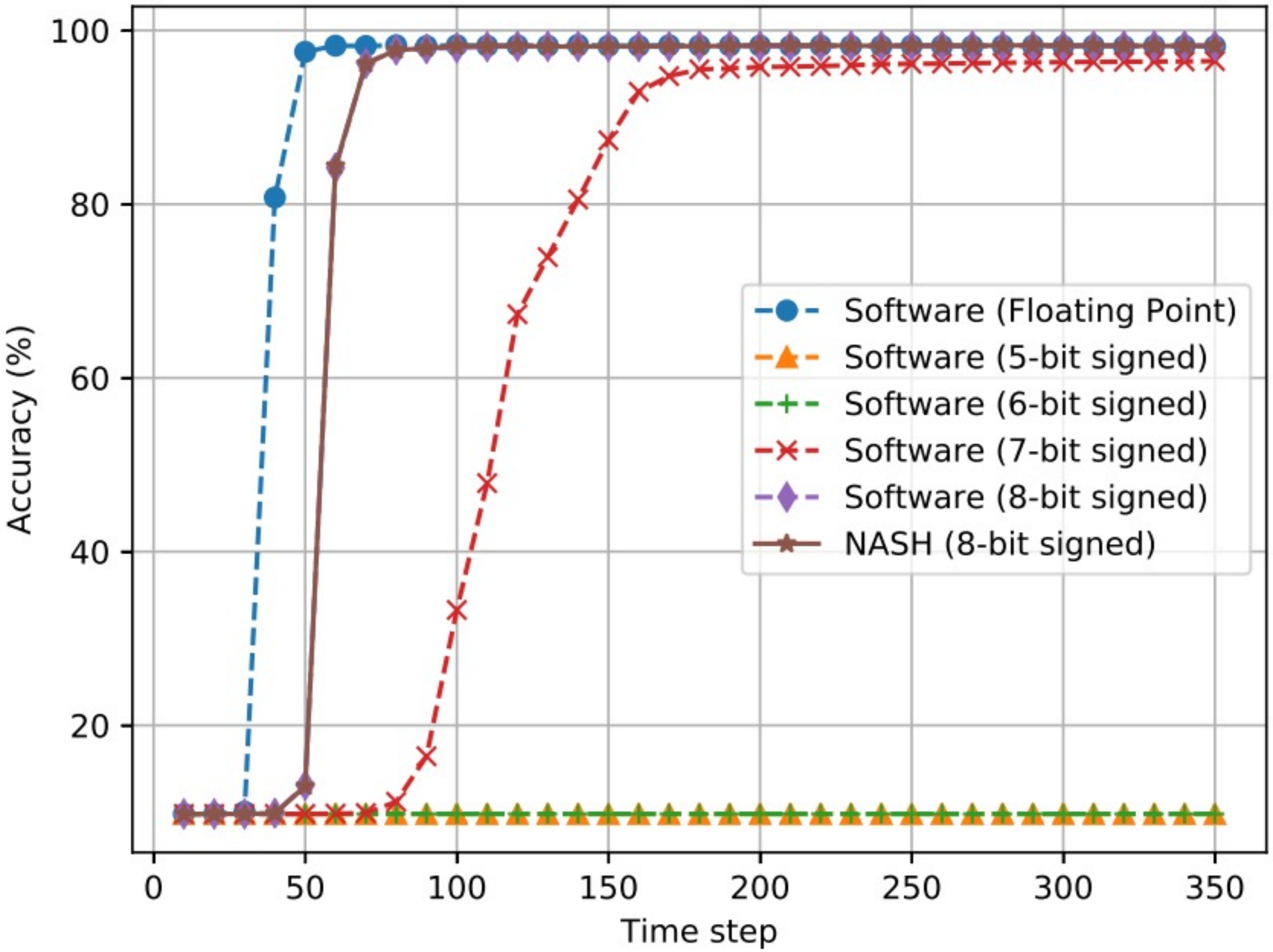
Pre-synaptic Spike SRAM Address
Read Counter [-8,+7]
≥0
Update Counter [-8,+7]
≥0
pre-synaptic have-before [255:0]
pre-spike-before [255:0]
OR
Register
pre-spike[255:0]
Pre-synaptic Spike SRAM Data
pre-spike-after [255:0]
OR
Register
have-pre-spike[255:0]
== zeros
calc. time-step
pre-synaptic have-after [255:0]
Pre-synaptic Spike Array
Post-synaptic Spike SRAM Address
Time-step Counter
one-hot-pre-spike[255:0]
== zeros
No-update
Index extractor
[7:0]
Weight SRAM Address (READ)
Post-synaptic Spike SRAM Data
post-spike [255:0]
Post-synaptic Spike Array
Register
[7:0]
Weight SRAM Address (WRITE)
Δw (norm.)
Δw
-Δw
≥0
Update Counter [-8,+7]
post-spike[255:0]
Read/Write Enable
Mode (norm, +/-)
Weight SRAM Data-in (READ)
synapse[7:0]
+
new-synapse[7:0]
Register
new-synapse[7:0]
Weight SRAM data-out (WRITE)
Δ[7:0]
+
Register
sum-Δ[15:0]
>> 8
Δw (norm.) [7:0]

STDP: update rule



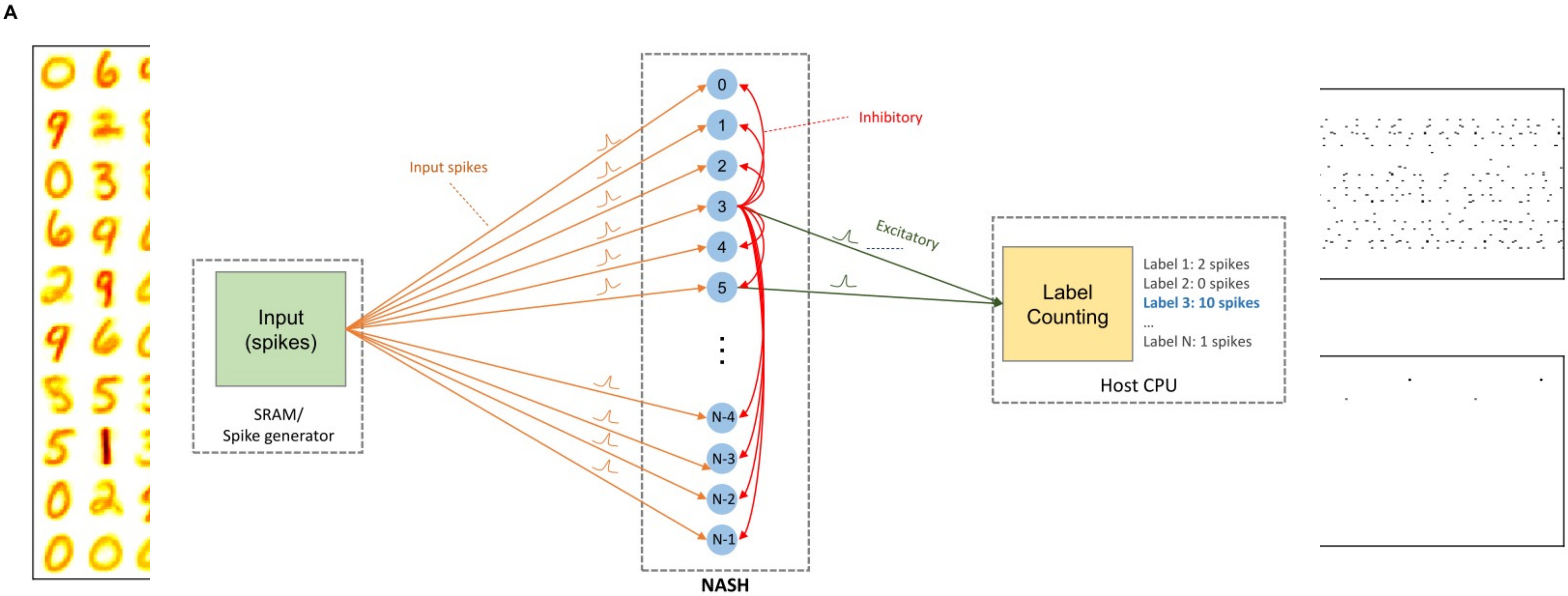
- The adjustment window is 16 timesteps.
- If neuron j fires, STDP update initiates.
- Presynaptic neuron i fires within 1 to 8 timesteps before the firing time of neuron j ➔ $w_{i,j}$ increases
- Presynaptic neuron i fires within 0 to 7 timesteps after the firing time of neuron j ➔ $w_{i,j}$ decrease

ANN-to-SNN conversion for MNIST



Network: 784:1024:1024:10

STDP for MNIST



Network: 784:N with lateral inhibitory connections in the second layer.

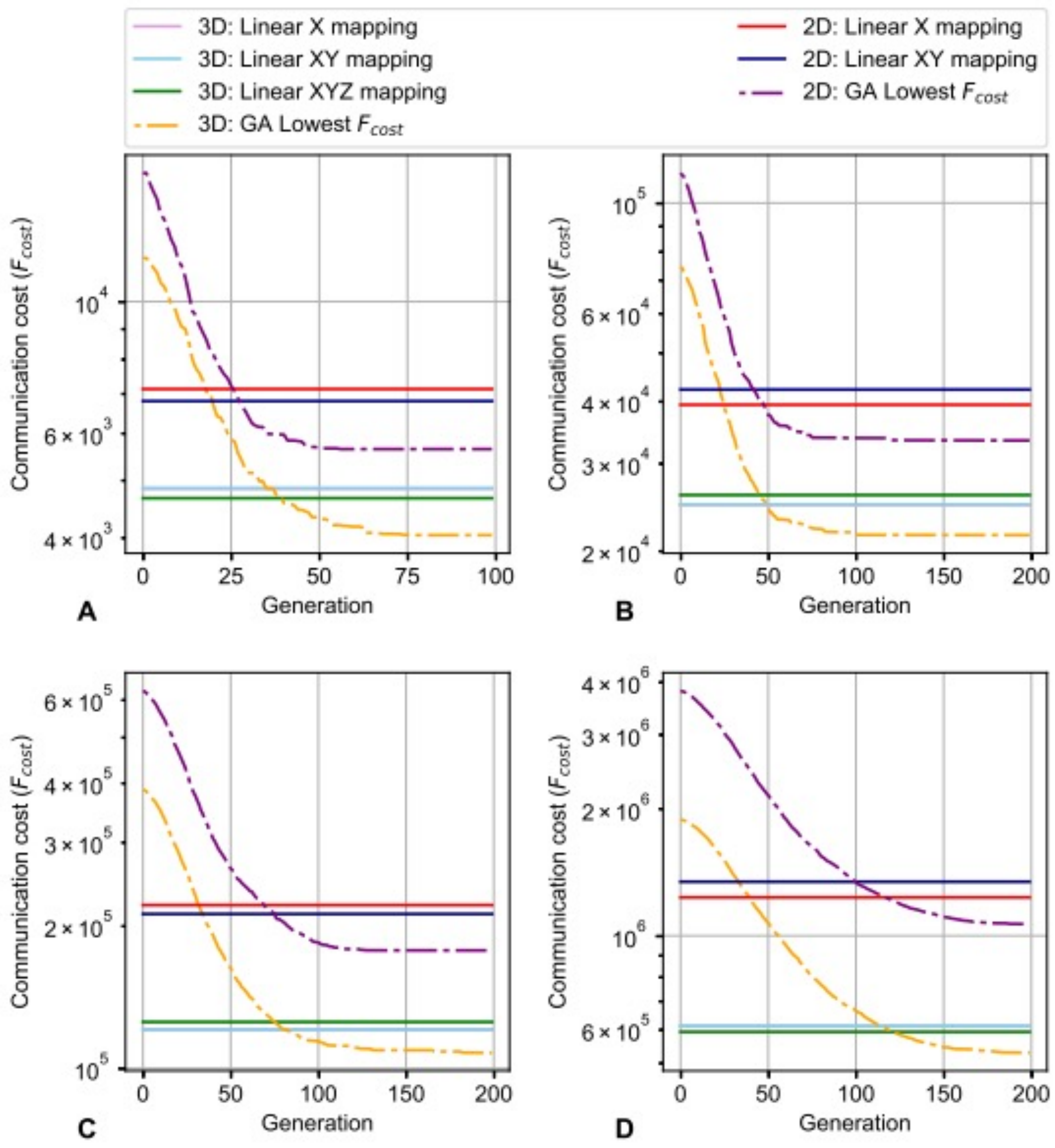
N=100 → accuracy = 71.32%, N=400 → accuracy = 84.05%

Fault-tolerant mapping: Execution time

	Size	Remap	1-hop GS	N-hop GS	GA[2]	MFMC	Load Balancing [41]	Kernighan-Lin [41]
2D NoCs	[4,4]	20.1 μs	90.3 μs	113 μs	5.916 s	351.4 μs	-	-
	[6,6]	38.2 μs	378.2 μs	421.3 μs	53.826 s	672.9 μs	-	-
	[8,8]	74.3 μs	0.640 ms	1.155 ms	140.392 s	1.202 ms	-	-
	[10,10]	216.5 μs	1.526 ms	2.076 ms	327.070 s	2.640 ms	-	-
	[12,12]	369 μs	3.413 ms	3.499 ms	640.914 s	5.032 ms	-	-
	[14,14]	608 μs	5.438 ms	6.150 ms	1220.911 s	7.428 ms	-	-
	[16,16]	932.5 μs	7.698 ms	8.713 ms	1932.054 s	11.097 ms	-	-
3D NoCs	[4,4,4]	39.2 μs	780.100 μs	942.8 μs	141.509 s	1.276 ms	-	-
	[6,6,6]	116.3 μs	7.349 ms	8.387 ms	1498.355 s	8.771ms	-	-
	[8,8,8]	394.5 μs	13.487 ms	14.076 ms	3.178 h	24.604 ms	-	-
	[10,10,10]	1.200 ms	32.151 ms	44.706 ms	15.237 h	63.545 ms	-	-
	[12,12,12]	3.365 ms	101.105 ms	106.223 ms	N/A [3]	116.658 ms	-	-
	[14,14,14]	8.333 ms	184.655 ms	204.551 ms	N/A [3]	208.800 ms	-	-
	[16,16,16]	19.938 ms	236.766 ms	291.288 ms	N/A [3]	442.986 ms	-	-
Any	W=1000, E=128	14.3 μs	44.6 μs	65.400 μs	3.856 s	310.1 μs	4.570 s	2030.1 s

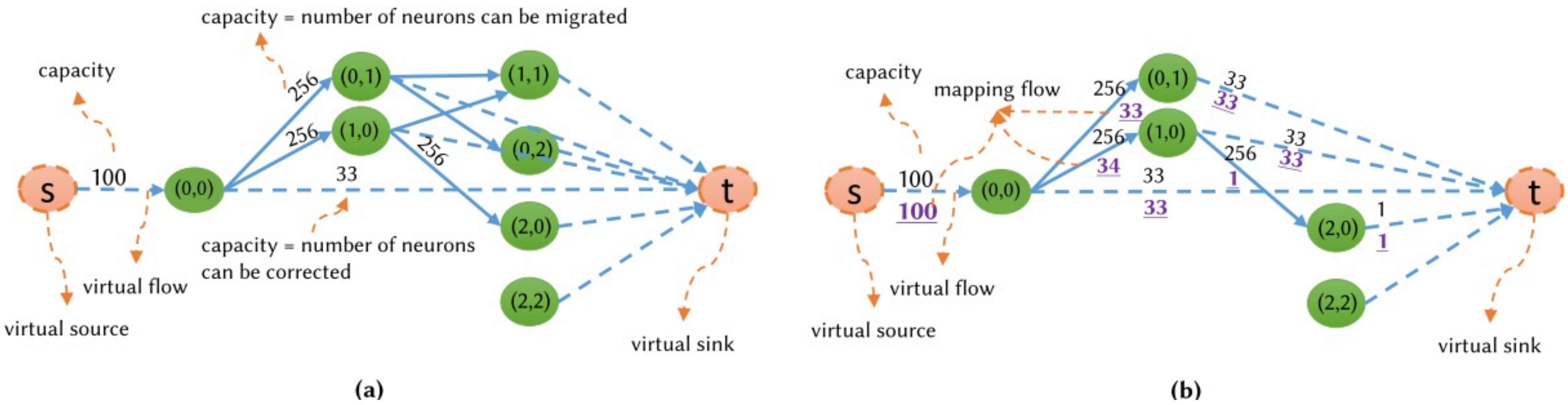
[0] The execution only takes into account the computation time for finding the new mapping. Other calculations such as setting up or configuration generation are not counted.
[1] System configuration: E=256, W = 0.8*X, k = 0.2*X.
[2] Genetic Algorithm configuration: 100 parents, 20 bests, 40 crossover and 40 mutations per generation, 200 generations.
[3] GA in 3D-NoC with the size from [12,12,12] is infeasible to perform due to extremely long execution time and large allocated memory.
[4] Linear mapping: the neurons with lower indexes are mapped to the nodes with lower indexes.

Initial mapping with GA: 2D vs 3D



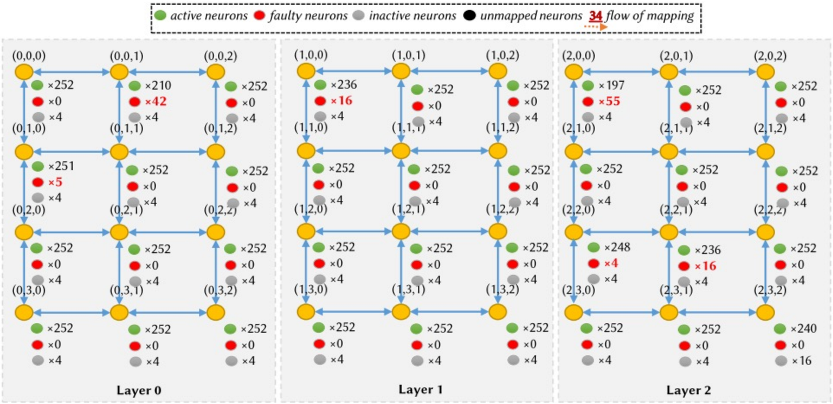
- Comparison between 3D and 2D mapping. (A) 64 nodes (4 × 4 × 4 and 8 × 8) NoC-based, 256 neurons/node. (B) 128 nodes (4 × 4 × 8 and 8 × 16) NoC-based, 256 neurons/node. (C) 256 nodes (4 × 8 × 8 and 16 × 16) NoC-based, 256 neurons/node.(D) 512 nodes (8 × 8 × 8 and 16 × 32) NoC-based, 256 neurons/node.

Mapping solution: Max-Flow Min-Cut

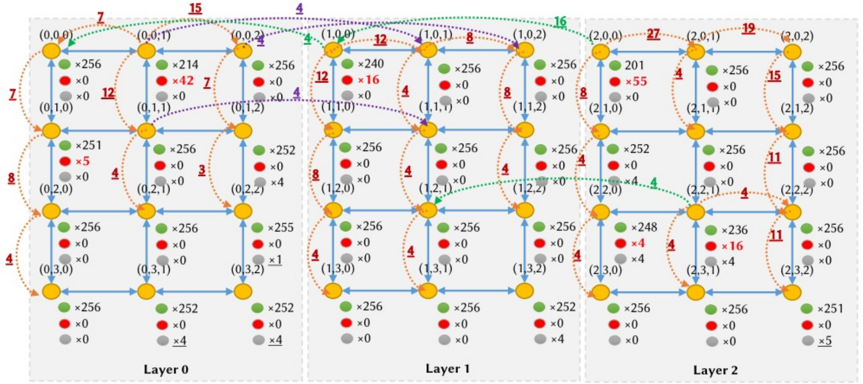


- For all nodes, create a flow of migration with the capacity is the maximum migrating neurons they can perform.
- All neurons can travel d_{max} hops → connect each node to all nodes in d_{max} hops.
- Flow from virtual source to each node: number of faulty neurons
- Flow from each node to virtual sink: number of spares
- Solve the max-flow problem using Ford-Fulkerson

Example: input



Example: output



Mutation for GA in fault-tolerant mapping