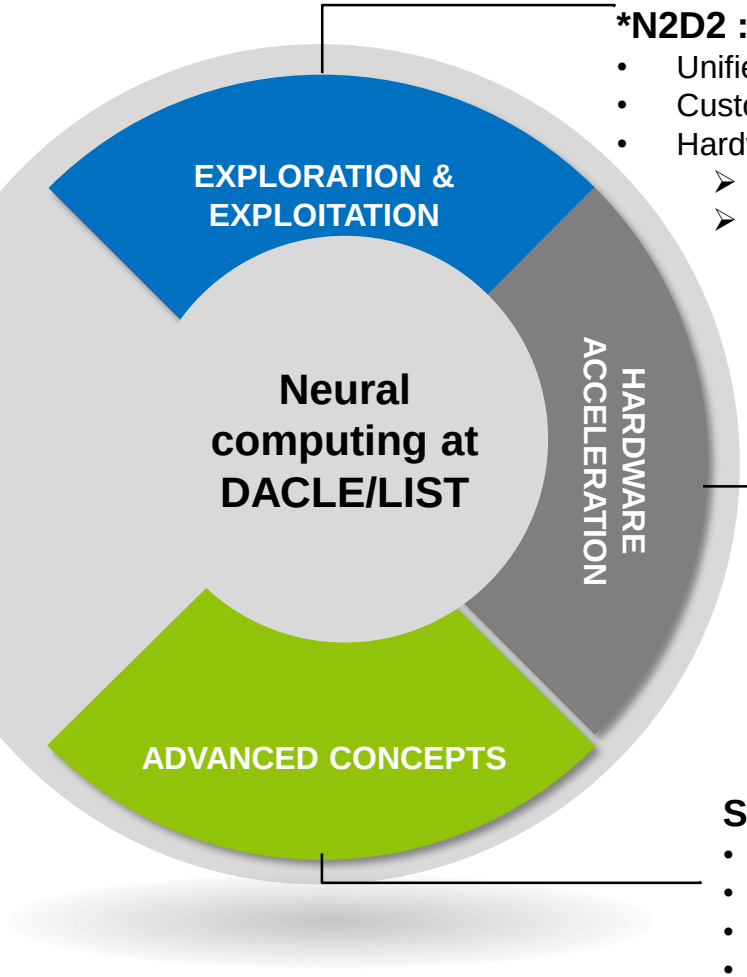
NEURAL COMPUTING DESIGN: N2D2 AND SPIKE ARCHITECTURES

Vincent Lorrain

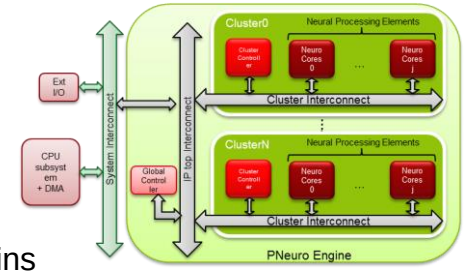
Olivier.bichler@cea.fr, Johannes.thiele@cea.fr , vincent.lorrain@cea.fr

GENERAL VIEW OF OUR THEMES



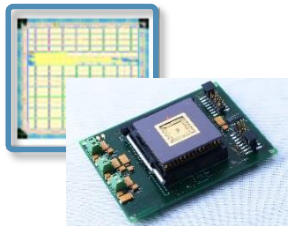
*N2D2 : DNN design framework

- Unified modeling and NN exploration tool (including spike coding)
- Custom applications building & optimization (CNN, Faster-RCNN…)
- Hardware mapping & benchmarking (CPUs, GPUs, FPGAs, ASIPs)
  - ➢ Programmable code generation: OpenMP, OpenCL, CuDA, TensorRT, PNeuro…
  - ➢ FPGA code generation: C/HLS, DNeuro



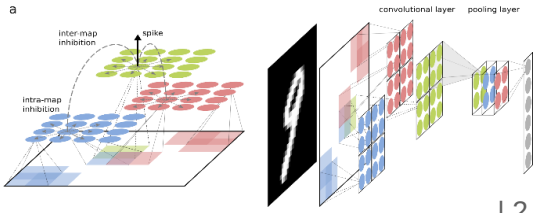
Hardware design

- Programmable processor **PNeuro**
  - ➢ Clustered 8-bit SIMD architecture
  - ➢ Designed for DNN processing chains
  - ➢ Support traditional image processing operations
- Dataflow FPGA IP **DNeuro**
  - ➢ Optimized RTL DNN layer kernels
  - ➢ Automatic RTL generation through N2D2



Spike neural networks

- Spike-coding, spike-BP, bio-inspired unsupervised learning (STDP)
- RRAM and new devices
- EU H2020 Project: NeuRAM3
- **NeuroSpike**
  - ➢ CNN spike acrhitecture



*N2D2 is available at https://github.com/CEA-LIST/N2D2/

ARTIFICIAL INTELLIGENCE

Artificial Intelligence

Machine intelligence that equals or exceeds human intelligence or efficiency at a specific task

Machine Learning

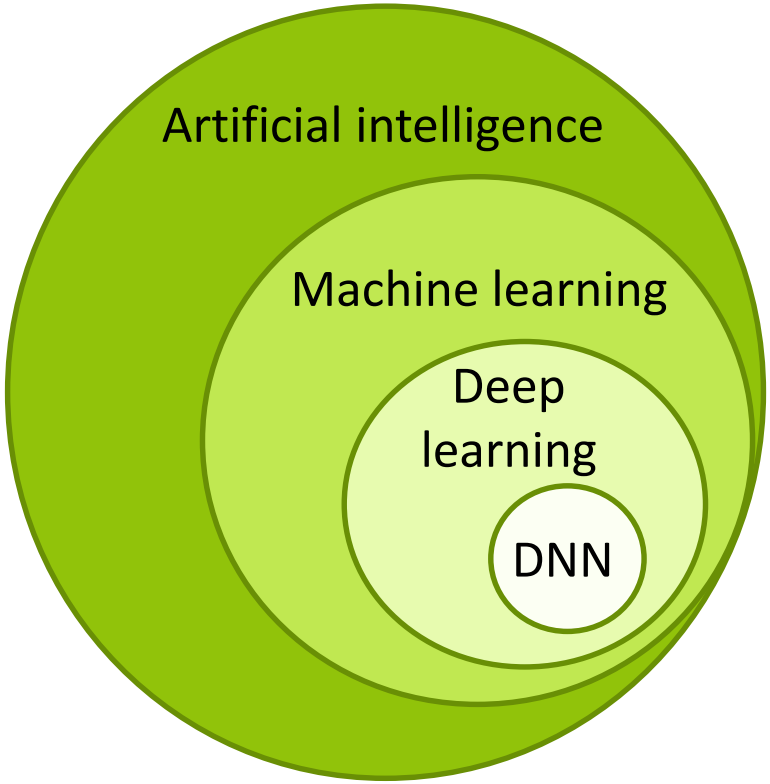
Provides computers with the ability to learn without being explicitly programmed

Deep Learning

Algorithms that permit software to train itself to perform tasks

Deep Neural Networks

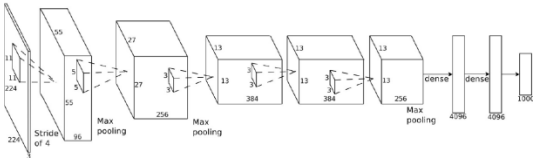
A hierarchy of multiple layers that mimic the neural networks of our brain

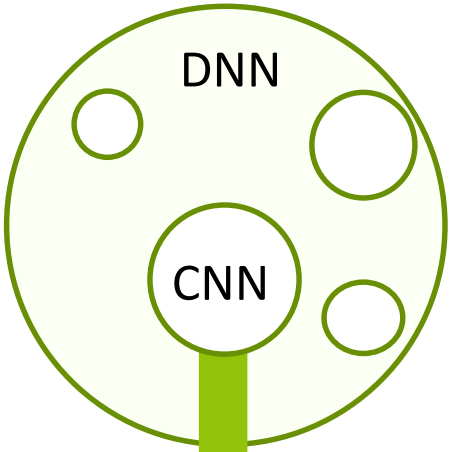


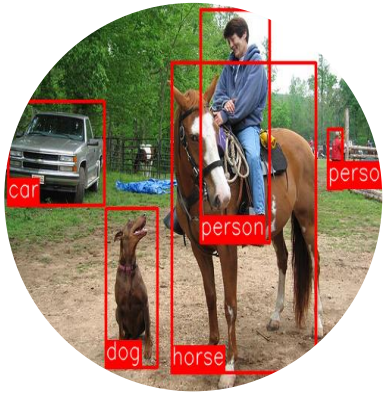
WHY THE CNN

Convolutional Neural Network

Today a fundamental building block in image recognition neural network-based applications

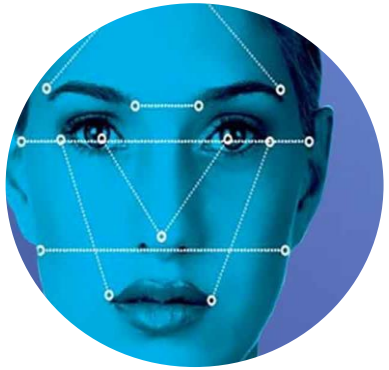






Object detection

Pedestrian detection

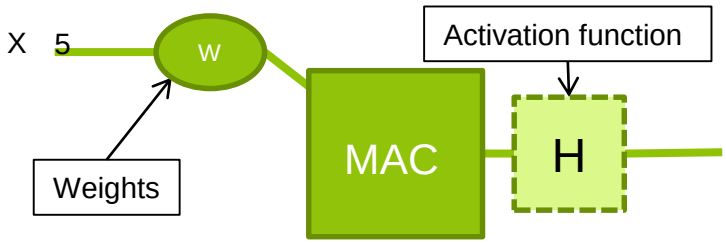


Facial recognition

WHY THE SPIKE

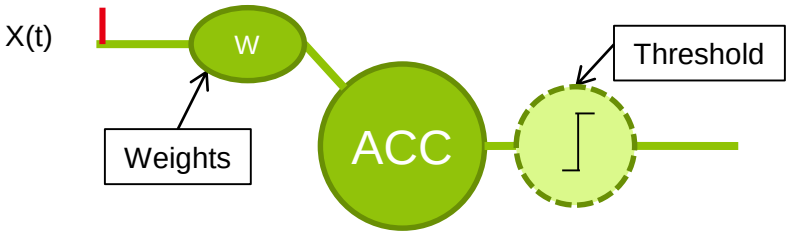
Two digital synchronous neuron instantiation

Frame ➔ vectorial operations



Out = H($\sum X_n W_n$)

Spike ➔ asynchronous data



Out = Threshold($\int X_n W_n dt$)

*MAC(16bits) 45nm 0,9v:
MULT(16bits) ~ 1,17 pJ
ACC(32bits) ~ 0,1 pJ
TOTAL ~ 1,27 pJ

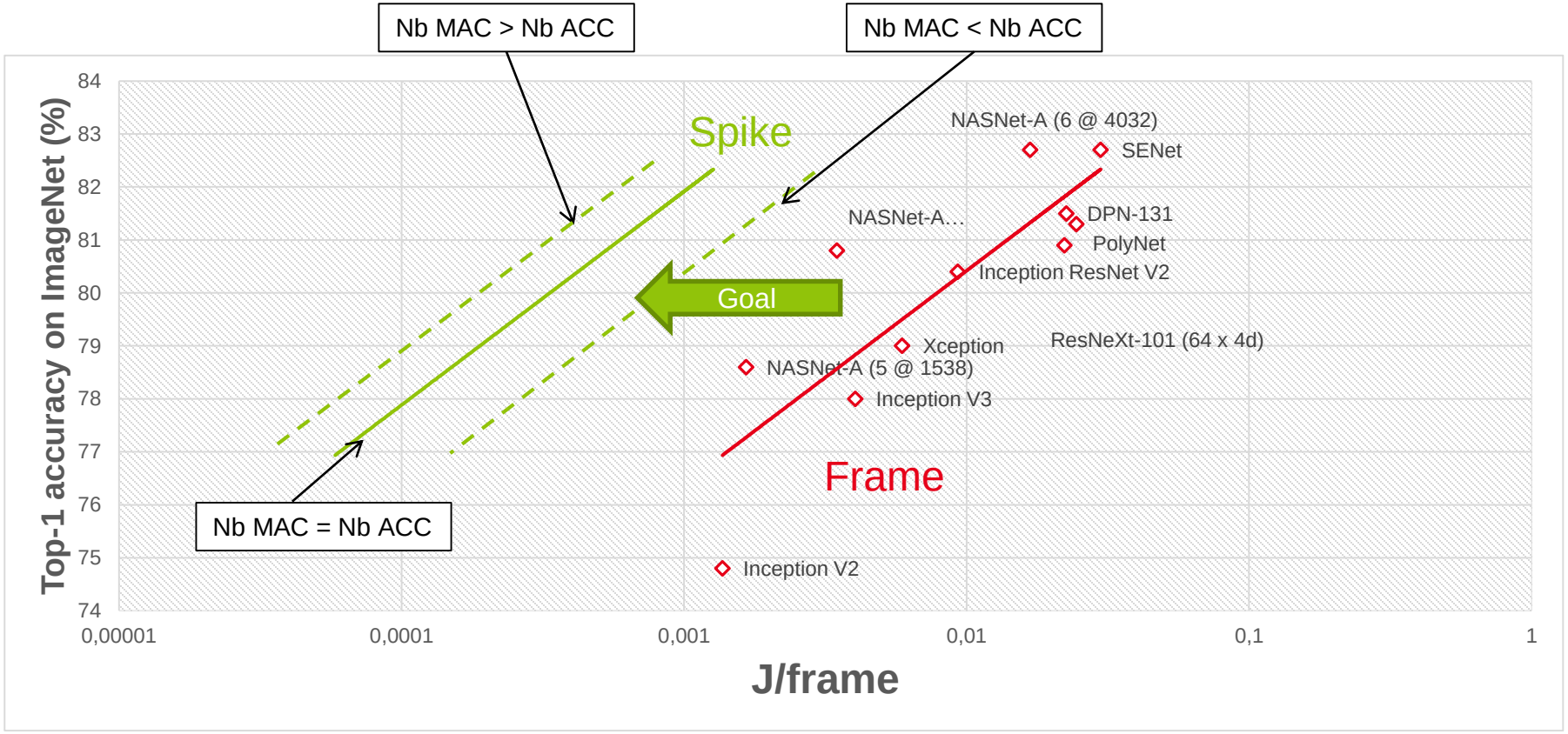
X42 energy reduction per operation

*ACC(8bits) 45nm 0,9v
ACC ~ 0,03 pJ

TOTAL ~ 0,03 pJ

Horowitz, M. computing's energy problem (and what we can do about it). In *Solid-State Circuits Conference Digest of Technical Papers (ISSCC),*

SPIKE ENERGY EFFICIENCY?



The potential of spike-based CNN:

- **Less energy per operation**
- **Sparsity ➔ less operations per pixel**

BUT ➔ requires a specific architecture

OUTLINE

1. N2D2
2. NeuroSpike
3. Perspectives for Spike
4. Other work : DNeuro

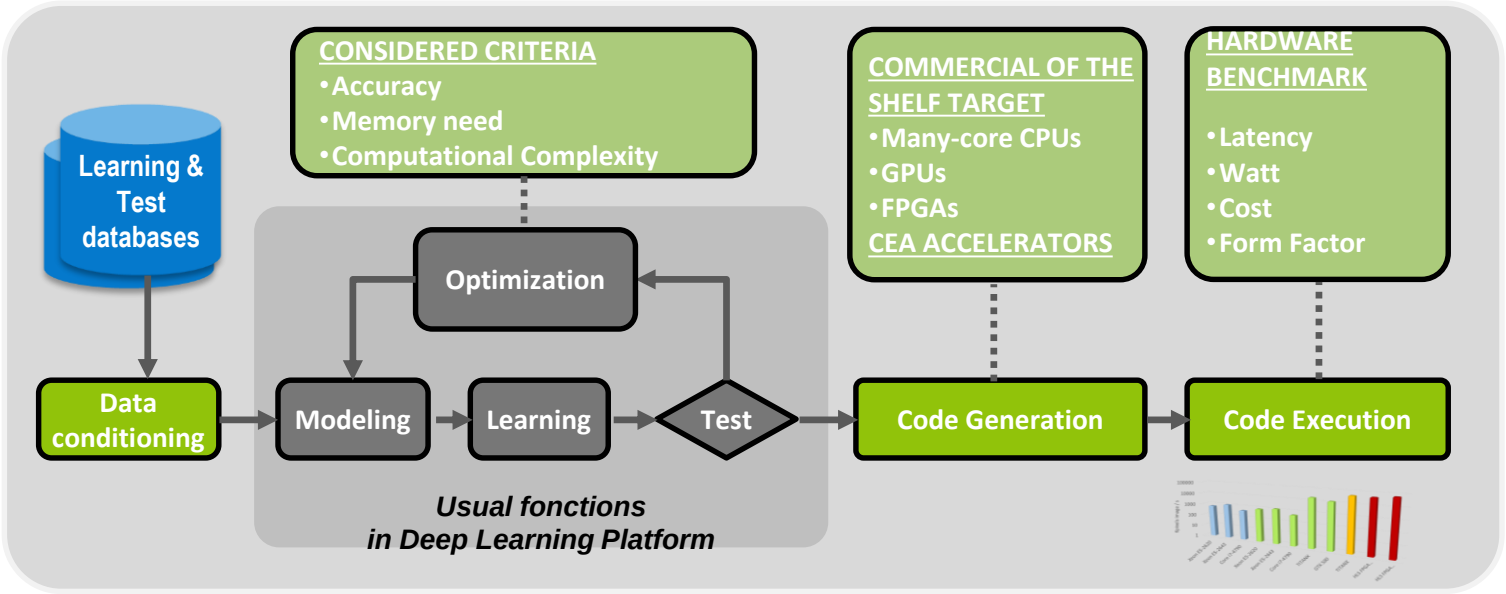
A UNIQUE PLATFORM FOR THE DESIGN AND EXPLORATION OF DNN APPLICATIONS

- Full proficiency of the framework
  - Contributions, algorithms, implementations and code (no dependency except OpenCV, no third party code) in C++
- Large flexibility
  - Open to developments and specific orientations based on industrial needs
- Unified modeling and tool flow for both formal and spike coding
- Explore Deep Neural Network (DNN) topologies with fast simulation and efficient analysis view
  - Experiment state-of-the-art learning techniques with large databases
  - Integrated benchmarking tools (number of computing cycles, memory footprint…)
- Easily integrate data conditioning by chaining pre-/post-processing transformations
  - Benefit from approximate computing to generate optimized DNN with reduced complexity
  - Data range adaptation tools (for 8 bits integer operations or less)

N2D2

N2D2: NEURAL NETWORK DESIGN & DEPLOYMENT

- **CEA's platform for the design and exploration of DNN applications:**
  - Spike coding: modeling and tool flow for both formal and spike coding
    - Advanced architectures exploration: unsupervised (STDP), NVM integration, 3D stacking…
  - Hardware exports: unified tool flow for hardware targets code generation, including generic CoTS and specific hardware
    - Precision reduction and data range adaptation (8 bits INT operations or less)
    - Benchmarking tools (number of computing cycles, memory footprint…)



N2D2 :SPIKE TOOL QUICK VIEW

Current OS version:

- Standard deep learning tool for feedforward networks
- Hardware exports for Deep Learning
- Automatic spike transcoding for inference
- Event-based spiking neural network simulation

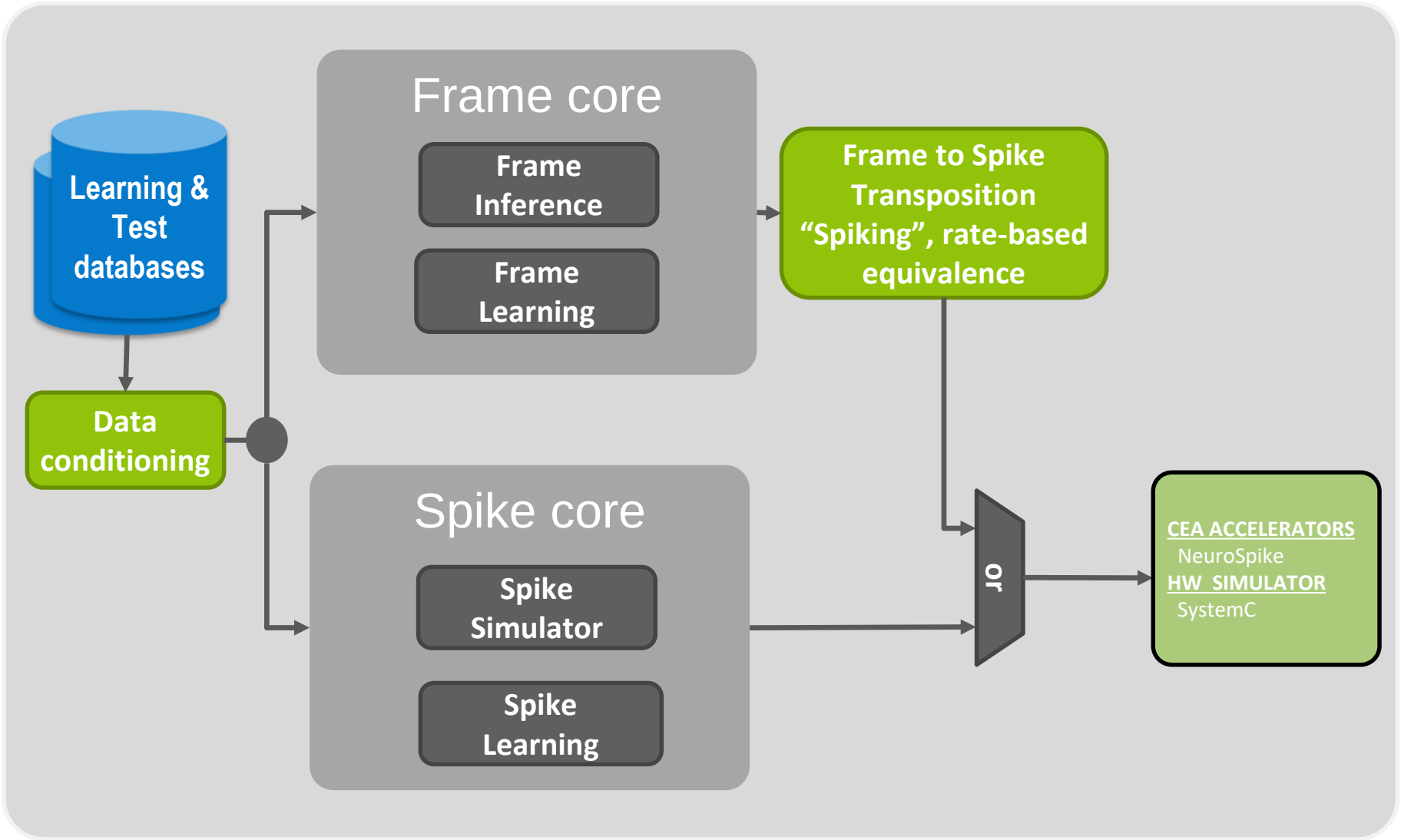
NEW: CUDA accelerated SNN:

- Clock-based (sparse binary spike matrix multiplications)
- Multi-layer learning of spiking networks with STDP and BP
- Scales well to large networks, in particular ConvNets
- Particularly suitable for high firing rates (i.e. rather dense matrices)

In process:

- Merge frame-based and spiking neural networks in one framework
- Make N2D2 framework for neuromorphic circuit optimization
- Exports for neuromorphic hardware

N2D2 :SPIKE FLOW QUICK VIEW

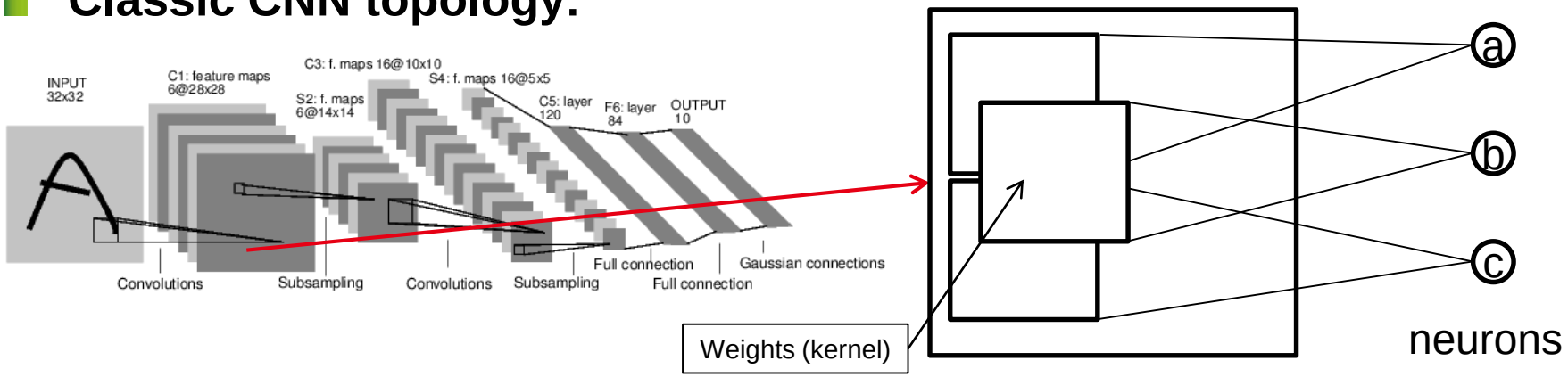


THE NEUROSPIKE

- The CNN layer and layer types
- Distribute the neurons for hardware efficiency
- Read the weights effectively
- Generalize the architecture for CNN layer
  - Generalization of layer (FC, MaxPooling)
  - Generalization for any CNN topology

THE CNN LAYER AND LAYER TYPES

- **Classic CNN topology:**



3 main types of layers :

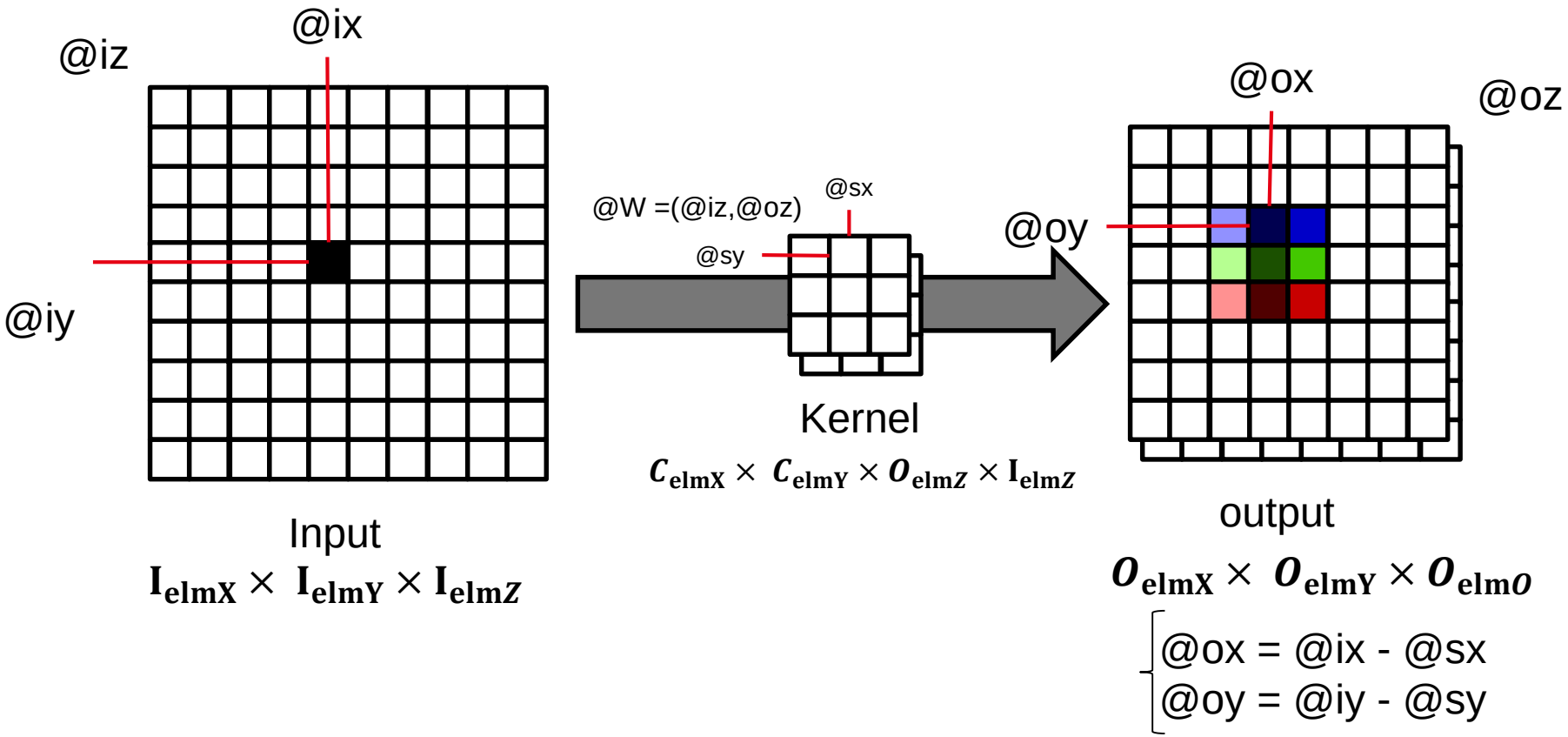
Convolution ➔ Neuron model, neurons partially connect with weights sharing

Fully connected ➔ Neuron model fully connect to the input

MaxPooling ➔ Max operation (≠ neuron model, no weight), same connectivity as convolution

CNN ➔ layered network with 3 main types of layers

DISTRIBUTE THE NEURONS FOR HARDWARE EFFICIENCY



One input spike is connected to $C_{elmX} \times C_{elmY} \times O_{elmZ}$ neurons

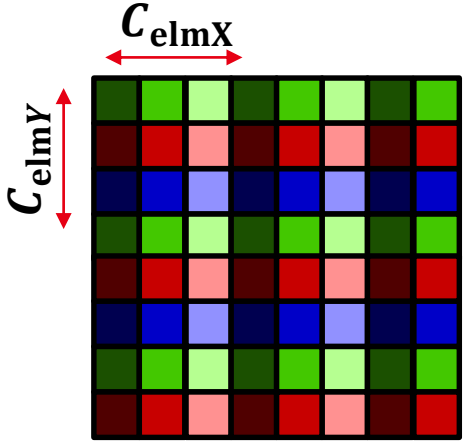
THE INDEPENDENT NEURONS

Independent neurons in convolutional layer:

= Neurons that do not share any input

With sequential (AER) input spikes:

- Independent neurons cannot be triggered at the same time
  ➔ These neurons can share the same **SPE** (Spike process elements)
- Nb. neurons that can be triggered at the same time = kernel size
- The distance between these neurons = $\boldsymbol{C_{elmX}}$ on o_x and $\boldsymbol{C_{elmY}}$ on o_y



Colors representation of independent neurons
For 3x3 kernel on one 8x8 output map
➔ Same color are independent neurons

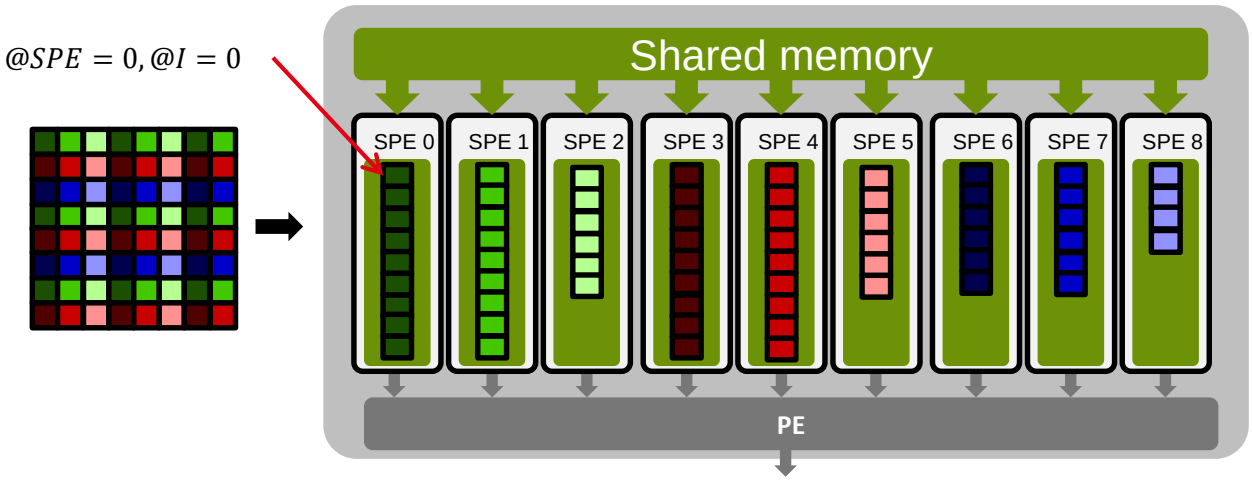
THE REPARTITION OF NEURONS

- **Hardware repartition of one output map**
  - Independent neurons are in the same SPE
  - The weights are distributed in relation of the @ of input spike
  - The outputs are serialized by the PE to be send to the AER

$$@SPE = (@o_x + O_{elmX} \times @o_y) mod\ (C_{elmX} \times C_{elmY})$$

$$@I = \frac{(@o_x + O_{elmX} \times @o_y)}{C_{elmX} \times C_{elmY}} + \frac{O_{elmX} \times O_{elmy}}{C_{elmX} \times C_{elmY}} \times @o_z$$

For more than one output map

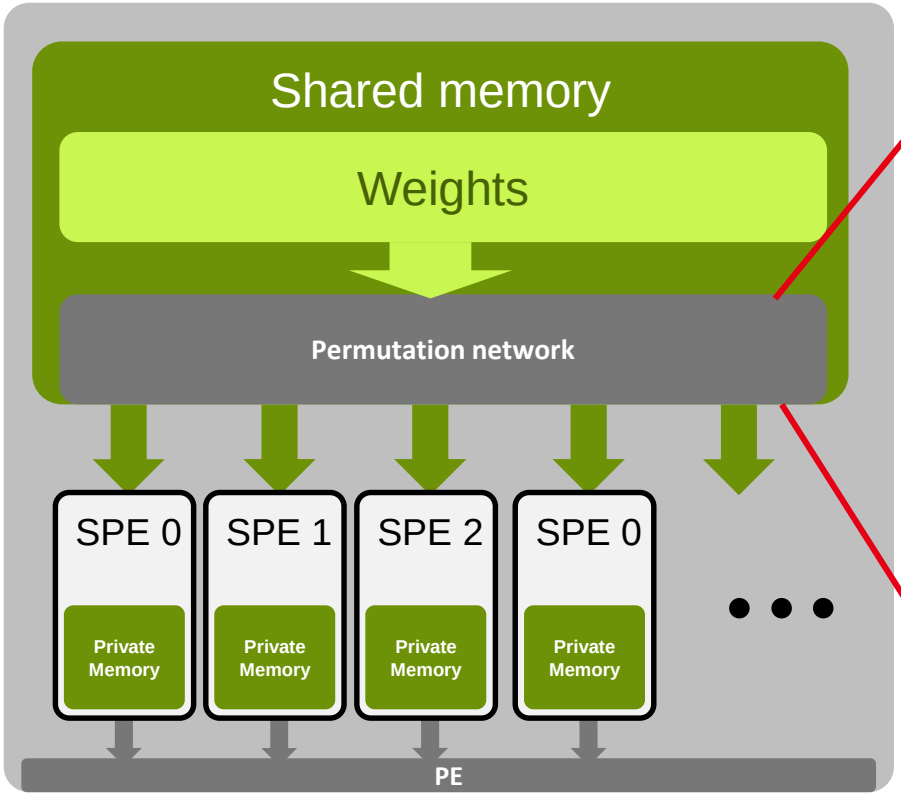


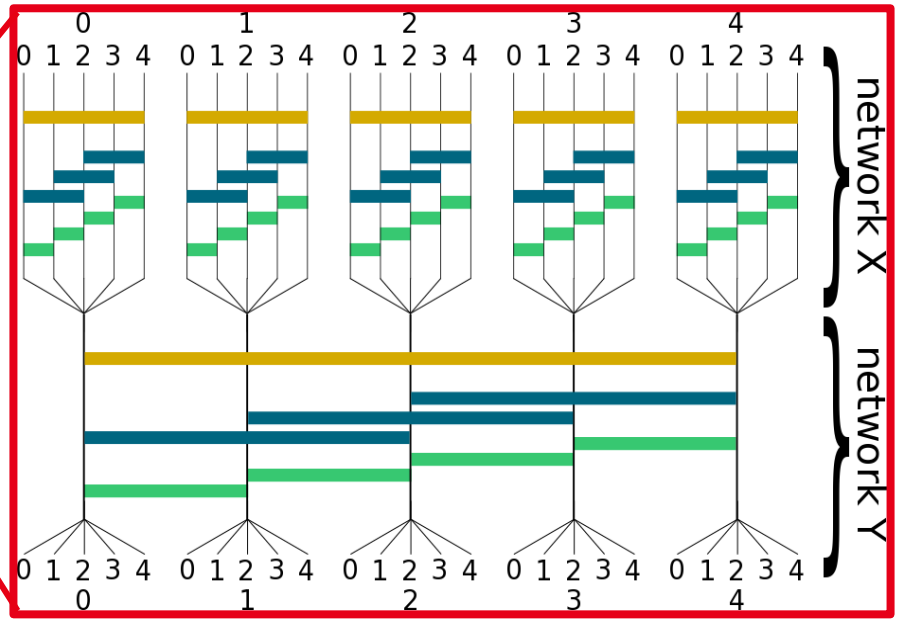
For ASIC the Nb SPE = Nb elem max kernel $(C_{elm_{MAX}})$

THE WEIGHTS DISTRIBUTION SOLUTION

- **Distribution of the weights to the SPE**
  - **Digital or analog weights allowed**
  - **Reduced and distributed hardware addressing**

- **Chosen solution ➔ permutation network**





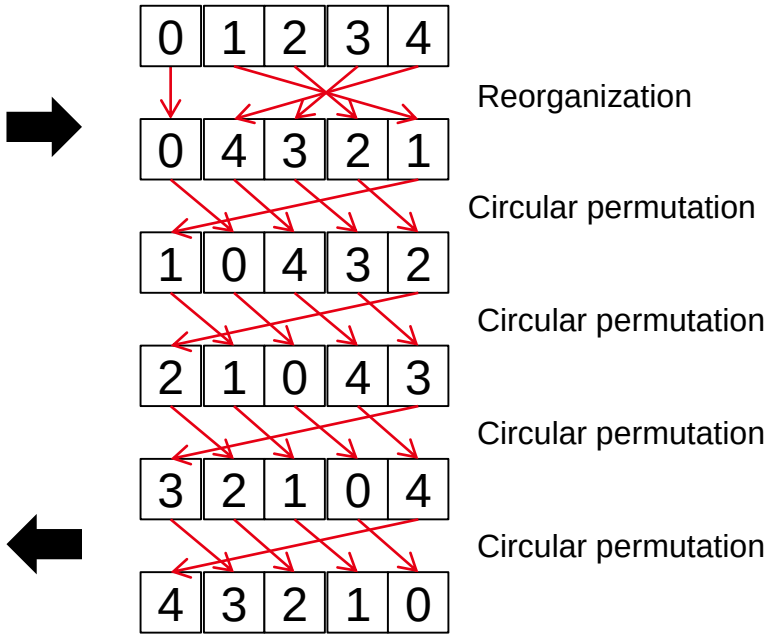
WEIGHTS DISTRIBUTION ACORDING TO THE NEURONS REPARTITION

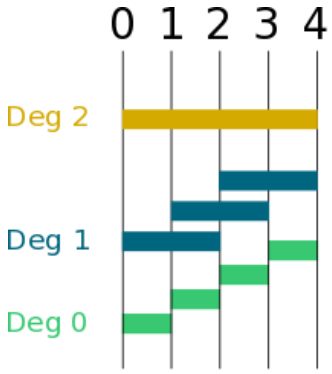
- **Actual equation of the permutation network, $s_y = 0$**

$$s_x = \begin{cases} (i_x \bmod C_{elmX}) - SPE_x & \text{if } i_x \bmod C_{elmX} \geq SPE_x \\ else\ (i_x \bmod C_{elmX}) - SPE_x + C_{elmX} \end{cases}$$

i_x	$SPE\_x = 0$	$SPE\_x = 1$	$SPE\_x = 2$	$SPE\_x = 3$	$SPE\_x = 4$
0	**0**	**4**	**3**	**2**	**1**
1	**1**	**0**	**4**	**3**	**2**
2	**2**	**1**	**0**	**4**	**3**
3	**3**	**2**	**1**	**0**	**4**
4	**4**	**3**	**2**	**1**	**0**

Ex. $f(i_x) = s_x$ for $C_x = 5$

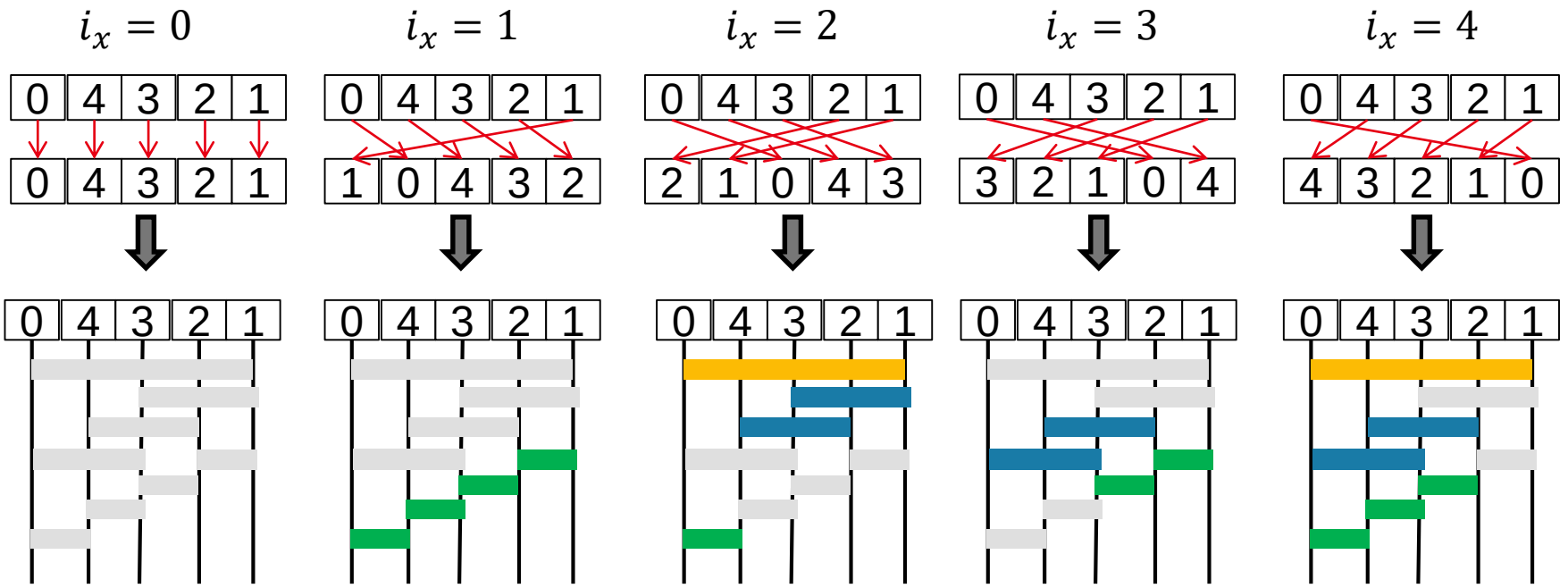




Identical logic For S_y

ONE CASE OF DISTRIBUTION

- **Network configuration examples**



The permutation can be applied on sub weight vector in case of kernel concatenation

$$C_{elmX} \times C_{elmY} < n \times C_{elmMax}$$

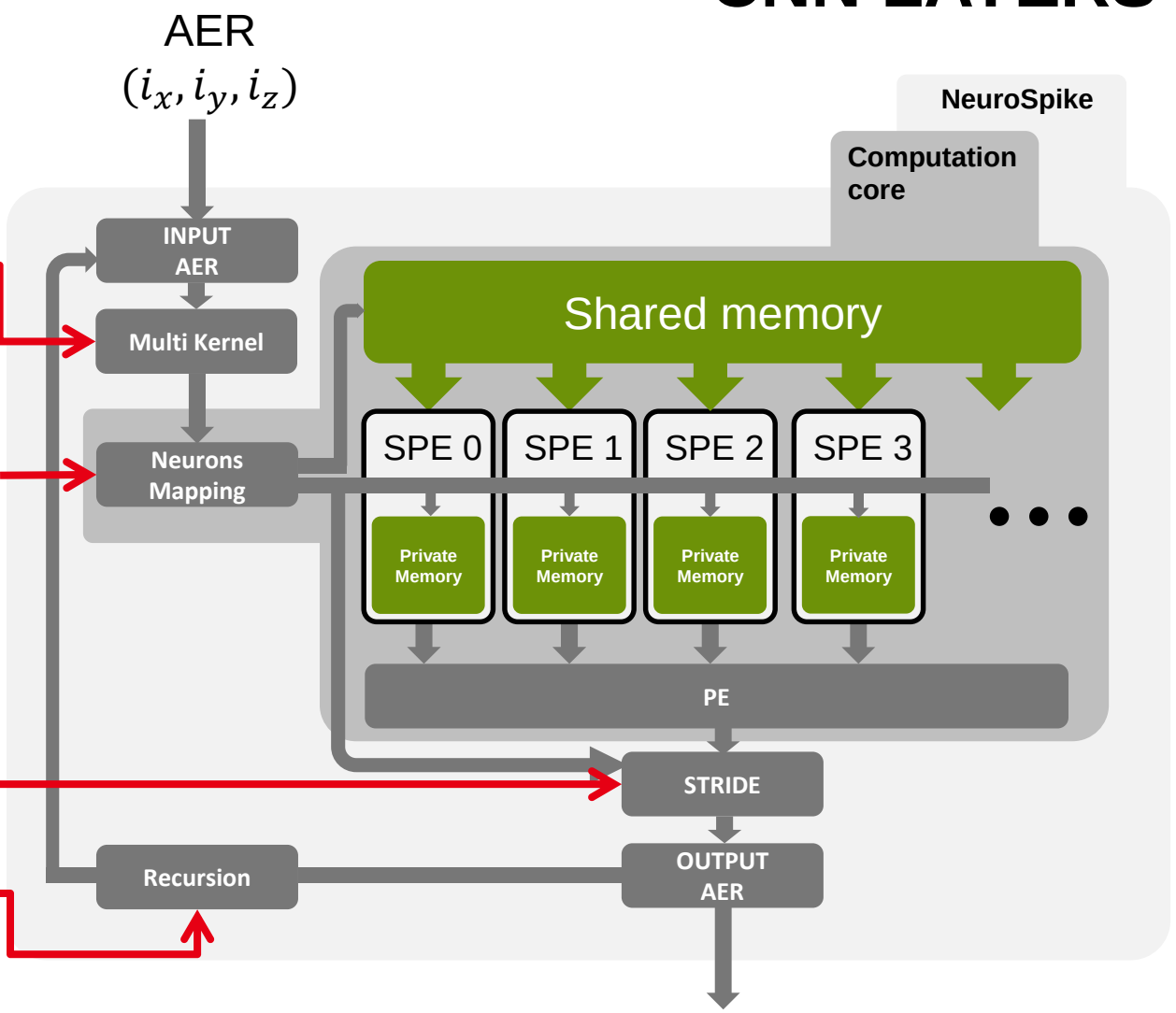
GENERALIZE THE ARCHITECTURE FOR CNN LAYERS

Allow sequential Output computation

Control The permutation network And the @I in all SPE

Allow stride /= 1

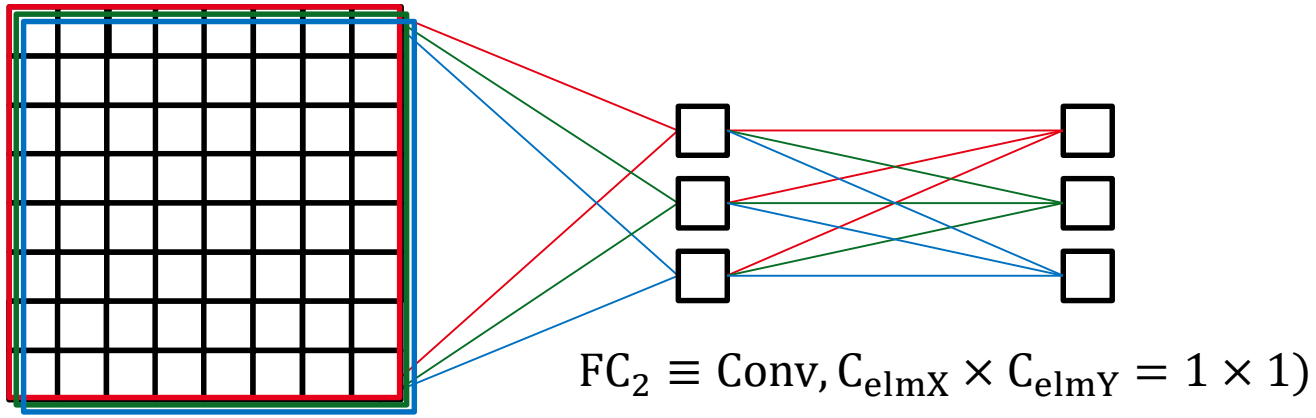
Allow multiple layer in one NeuroSpike



Hardware blocks added to the computation core to enable all CNN topologies and main layer types

GENERALIZE THE ARCHITECTURE FOR CNN LAYERS : FC LAYER

- **Fully connected layer**
  - Equivalent to a convolutional layer with kernel size = input size

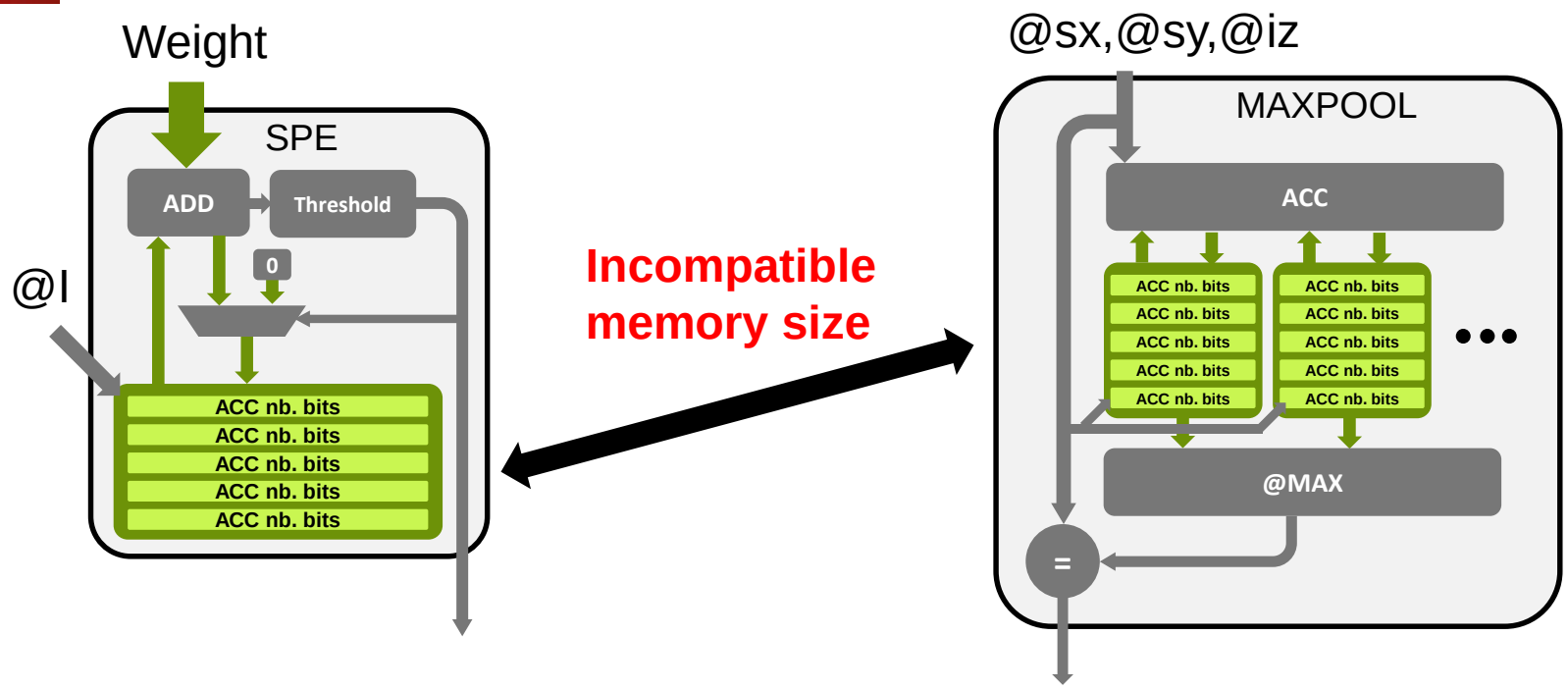


$$FC_2 \equiv Conv, C_{elmX} \times C_{elmY} = 1 \times 1)$$

$$FC_1 \equiv Conv, C_{elmX} \times C_{elmY} = I_{elmX} \times I_{elmY} = I$$

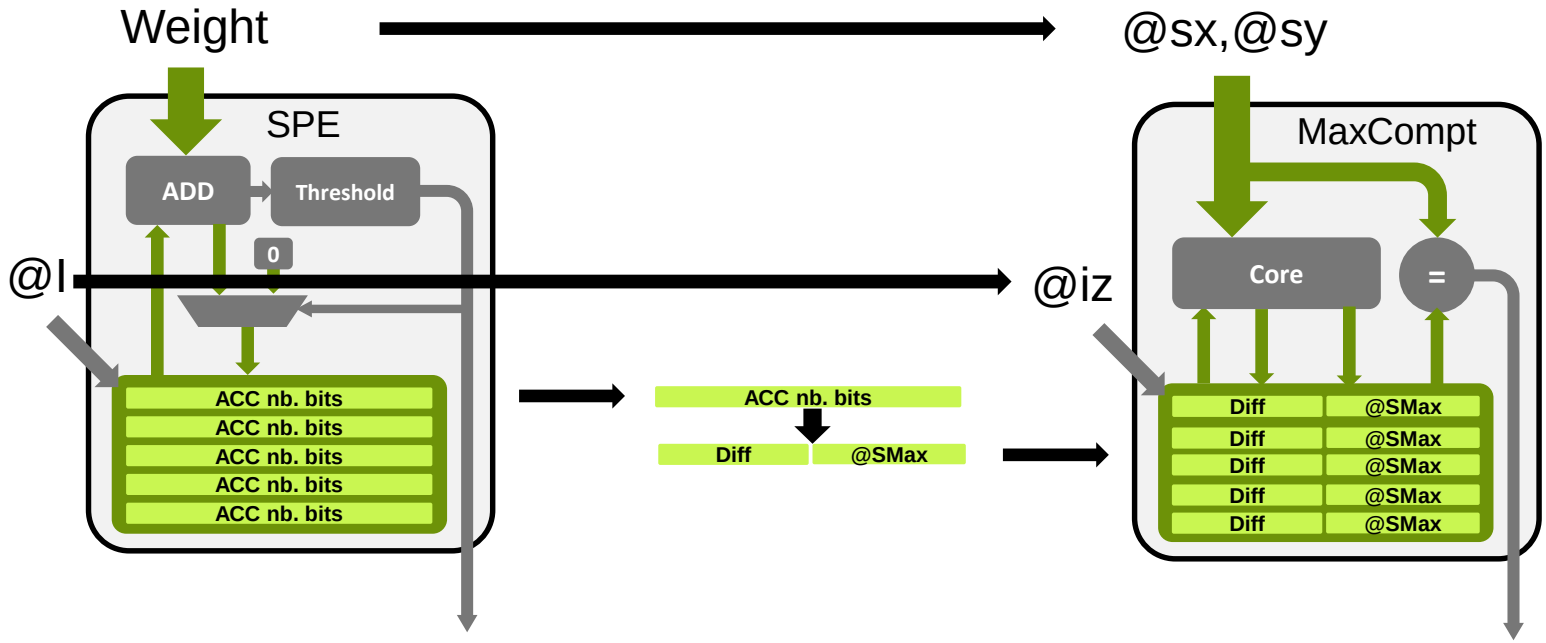
  - Full convolution hardware reuse
  - Hardware limit for FC ➔ $Nb_{SPE} = C_{elm_{MAX}} \geq I$

GENERALIZE THE ARCHITECTURE FOR CNN LAYERS : MAXPOOLING



- **MaxPooling ➔ MAX activity determination**
  - Same connectivity as Conv layers
  - MaxPooling is expensive
    - Memory ➔ store all synaptic activities
    - Computation ➔ max of all synaptic activities
  - Incompatible with the IF neuron implementation

GENERALIZE THE ARCHITECTURE FOR CNN LAYERS : MAXPOOLING



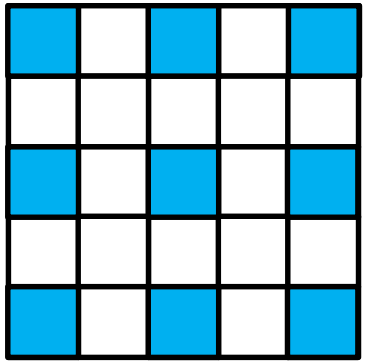
- **MaxCompt ➔ MaxPooling approximation**
  - Same connectivity as Conv layer use of weights for synapse @
  - Compatible with the IF neuron implementation
  - MaxCompt is less expensive :
    - Memory ➔ store 1 differential activity and 1 address
    - Computation ➔ single address comparison
- **Loss <0.1% on the recognition rate**

GENERALIZE THE ARCHITECTURE FOR CNN LAYERS : STRIDE

- **Deduction from the output map with** $\mathrm{St_x} = \mathrm{St_y} = 1$

$$\begin{cases} \mathrm{new}(o_x) = \left\lfloor \frac{o_x}{St_x} \right\rfloor \text{ if } o_x \bmod St_x = 0 \\ \mathrm{new}(o_y) = \left\lfloor \frac{o_y}{St_y} \right\rfloor \text{ if } o_y \bmod St_y = 0 \end{cases}$$

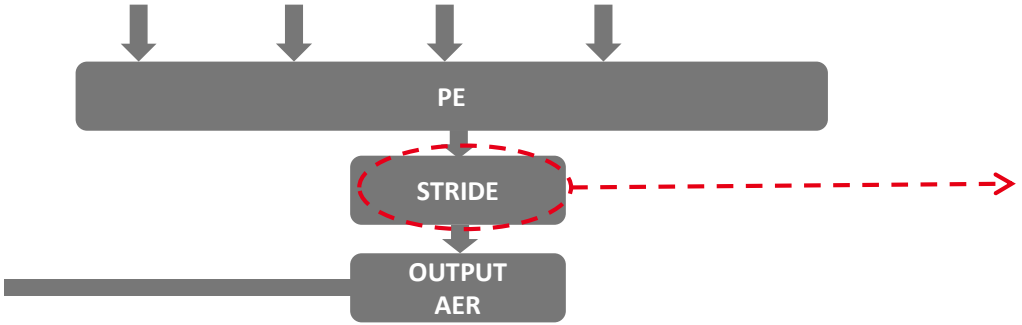
Output map



Stride = 1

Stride = 2

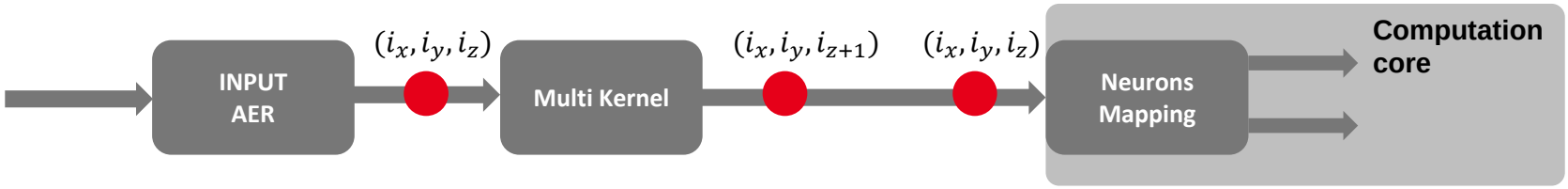
- **Discard of unused outputs and address remap of remaining outputs in *post-calcul***



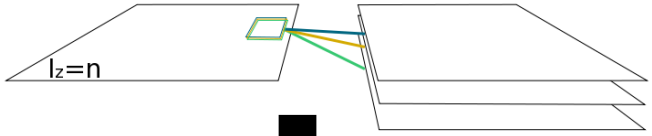
1 ALU instead of $\mathrm{C_{elm_{MAX}}}$

GENERALIZE THE ARCHITECTURE FOR CNN LAYERS : MULTI OUTPUT MAP

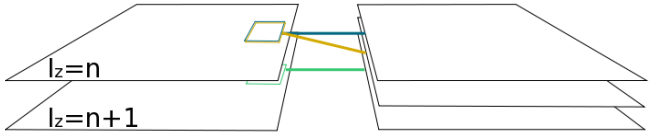
- **Multi Kernel ➔ time unfolding**
  - **All filters cannot be stored in one weight memory line**
  - **Duplicate ≡ multiple inputs with the same pulses**
  - **Repetition of the same entry with i_z ascending**



Desired connection

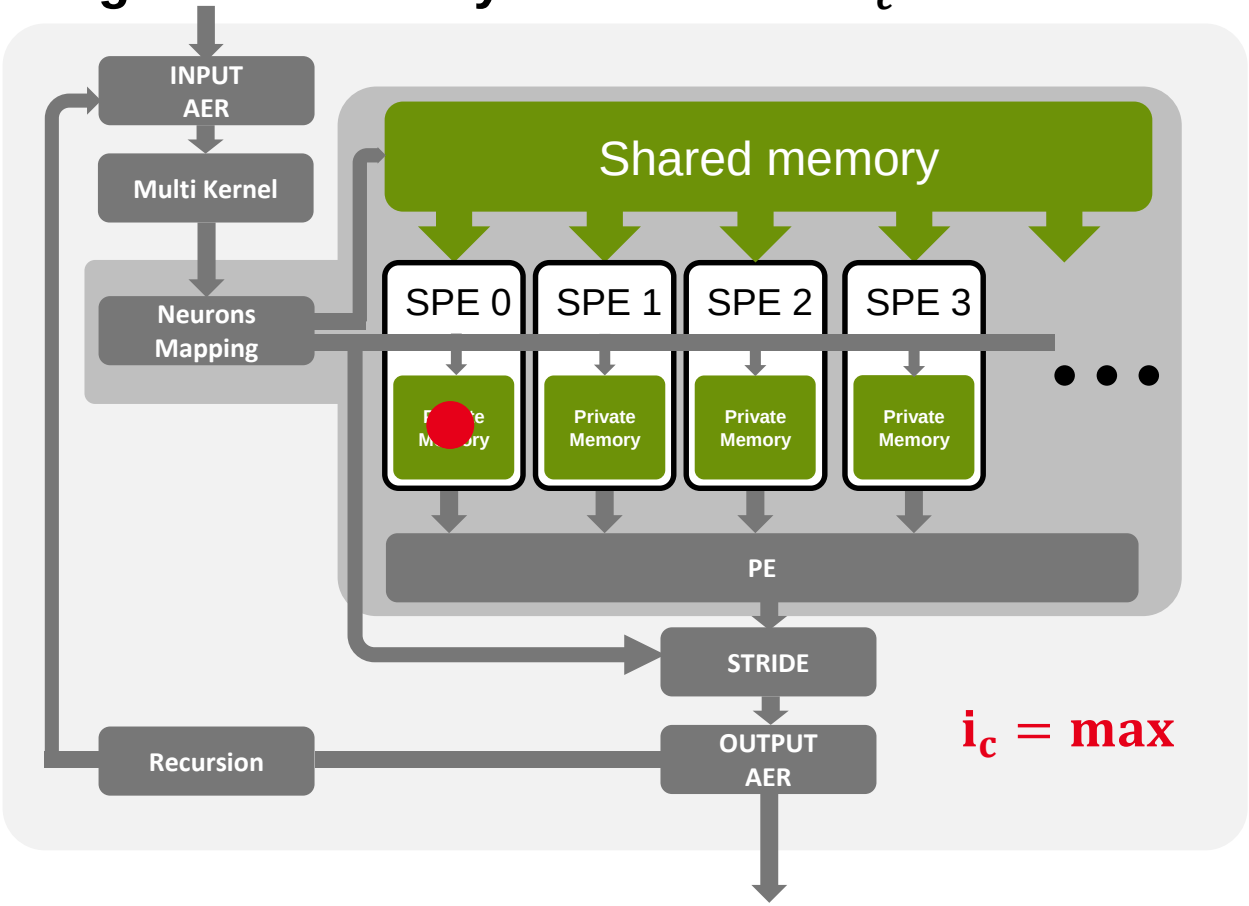


Implemented connection



GENERALIZE THE ARCHITECTURE FOR CNN LAYERS : MULTI LAYERS

- **Multi layers**
  - Recursion of spike processing
  - Looping of the output spike to the input
  - Adding an internal layer coordinate i_c



THE RESULTS

- OP a common metric for Spiking architecture
- STOA: spiking architectures
- NeuroSpike Measurement and conditions
- NeuroSpike Performance

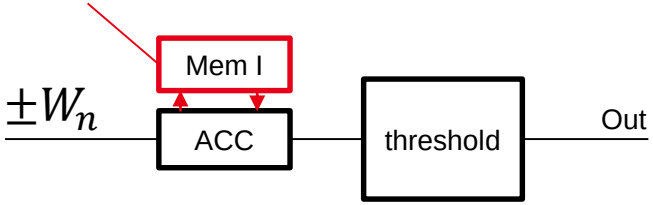
OP A COMMON METRIC FOR SPIKING ARCHITECTURE

- Case study: digital implementation
- Goal: find a common metric for spike comparison

Intrinsically required memory

Spike

Out = Threshold($\int X_n W_n dt$)



- **ACC(N_{bit}) ➔ ADD(N_{bit})**
- **ADD(N_{bit}) = $\left((N_{bit} - 1) + \frac{1}{2}\right)$**

ADD(8bits) = 7,5 OP

OP_i = ACC(N_{bit}) × nbACC

OP/j = $\frac{Kernel_{max} * ADD(N_{bit})}{W * t_{process}}$

OP/s = $\frac{Kernel_{max} * ADD(N_{bit})}{t_{process}}$

Common metric: “atomic” operation with data/weights precision of N_bit

STOA: SPIKING ARCHITECTURES

Generic

	TrueNorth	BrainScaleS
$Kernel_{\max}$	256	112
ACC //	256	8
$t_{process}$	1 ms	100 ms
N_{bit}	1 bit	4-8 bits
W	10^{-2}mW	1,7W

- **// read of weigths**
- **// process**
- **No weights sharing**

Partial hardware sharing

Conv

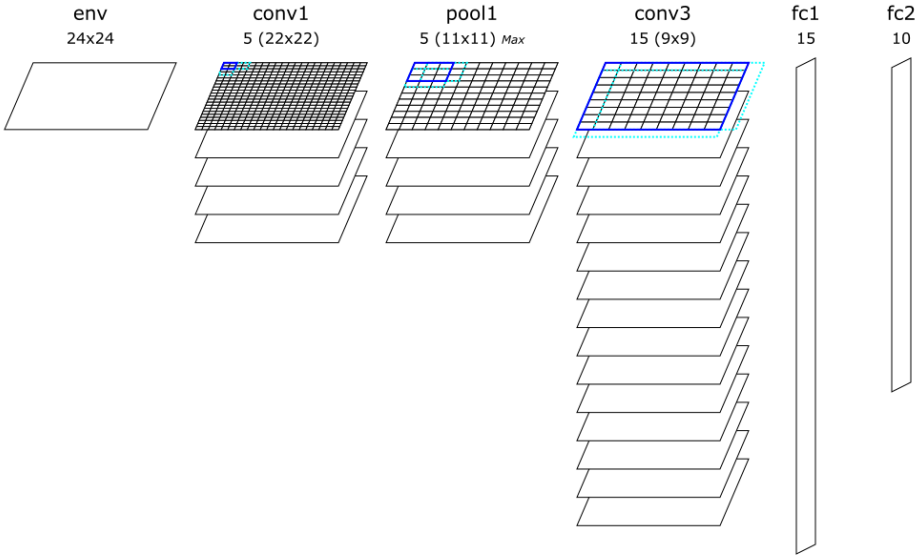
	Serrano	Camuas 11
$Kernel_{\max}$	1024	1024
ACC //	1	1
$t_{process}$	30-340 ns	50-565 ns
N_{bit}	3 bits	18 bits
W	150 mW	200 mW

	Camuas 12	Multiplex
$Kernel_{\max}$	4096	16k
ACC //	1	128
$t_{process}$	60-680 ns	24-774 ns
N_{bit}	6 bits	6 bits
W	200 mW	NC

- **sequential (AER) input spikes**
- **Weights sharing**
- **Only convolution**

MEASUREMENT AND CONDITIONS

Network for MNIST database



	Frame DNN	Spiking DNN
Quantization	Yes	Yes
Approx. computing	No	Yes
Base operation	Multiply-Accumulate (MAC)	Accumulate only
Activation function	Non-linear function	Threshold (+ refractory period)*
Parallelism	Spatial	Spatial and temporal
Memory reutilisation	Yes	No

**Not required for ReLU activation function*

	Frame		Spiking		
	Score	**MACs**	**Δ**	**Score**	**spikes/MAC**
MNIST ReLU	98.1%	94k	5	**98.1%**	1.27

Transcoded CNN used in PrimeTime

MEASUREMENT AND CONDITIONS

NeuroSpike configuration for the simulations

Nb. layers	$C_{MaxX} \times C_{MaxY}$	Nb. filters	$I_{elmX} \times I_{elmY}$	Neurons	Process
8	11 x 11	2048	32 x 32	30,976	FDSOI 28

NeuroSpike Simulation results without memory (typical 25°C, 0.9V)

Simulation	Power	Area	Freq
DC	70.9 mW	0.309 mm²	400 MHz
Prime time	67.8 mW	0.309 mm²	400 MHz

Bodies bias memory projection size and power (typical 25°C, 0.9V)

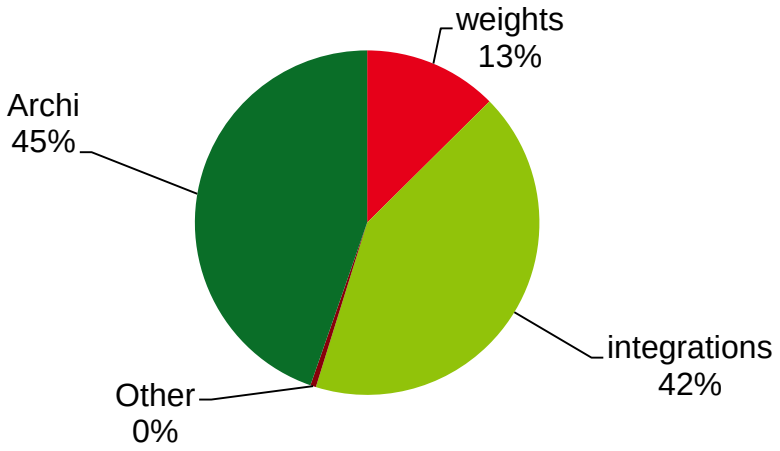
Memory	Power	Area	Rd%	Wr%	Freq
Weights	19.0 mW	1.5 mm²	20	0	400 MHz
Integrations	64.1 mW	0.3 mm²	20	20	400 MHz
Others	0.8 mW	0.2 mm²	3	0	400 MHz

Read and write rate determined using the RTL simulation

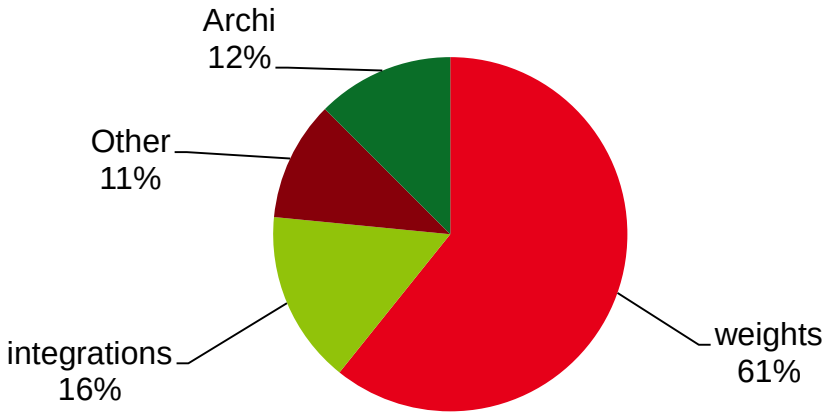
NEUROSPIKE PERFORMANCE

PrimeTime simulation gives:

Consumption/Power: 153.4 mW



Area 2.466 mm²

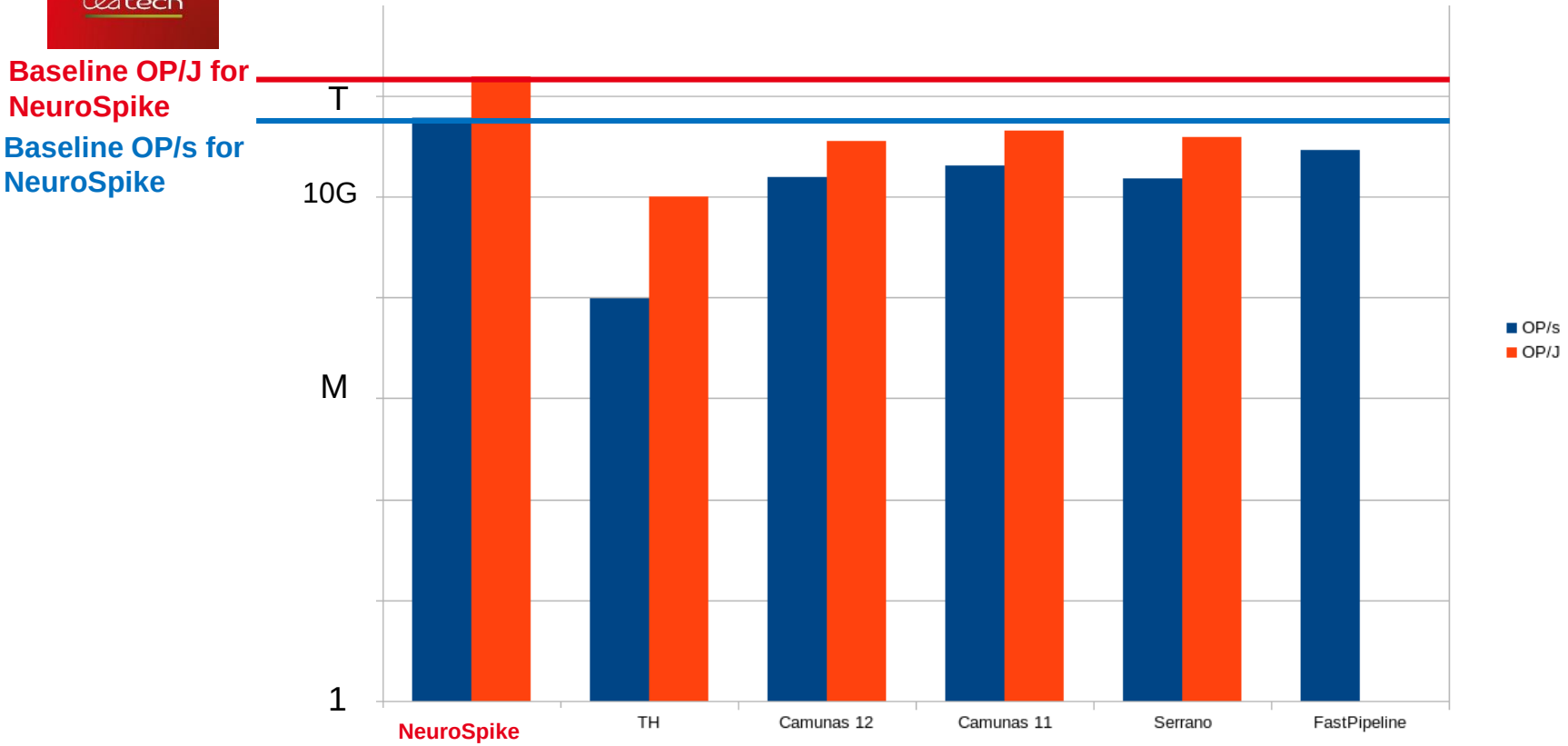


Power consumption ~ evenly distributed between memory and logic

Weights (R only) power << Integrations (R/W) power

Weights area >> Integrations area

NEUROSPIKE PERFORMANCE

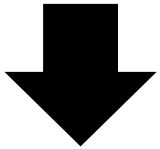


- **Performance in OP vs Camunas 11**
  - Gain of **x4** in processing time
  - Gain of **x11** in energy consumption
  - Flexibility in term of topology and models (MaxPooling, FC)
  - Full CNN implementation

- **Perspectives**
  - **The future of NeuroSpike**
  - **Works in process**
- **Dneuro**

NEXT : EVENT BASED DATA, UNSUPERVISED LEARNING

- **Event based data:**
  - **Reducing the number of spike needed for an application**
  - **Need an efficient event-based Learning for CNN**
- **Unsupervised learning**
  - **may lead to more efficient data representation, hence energy efficiency**

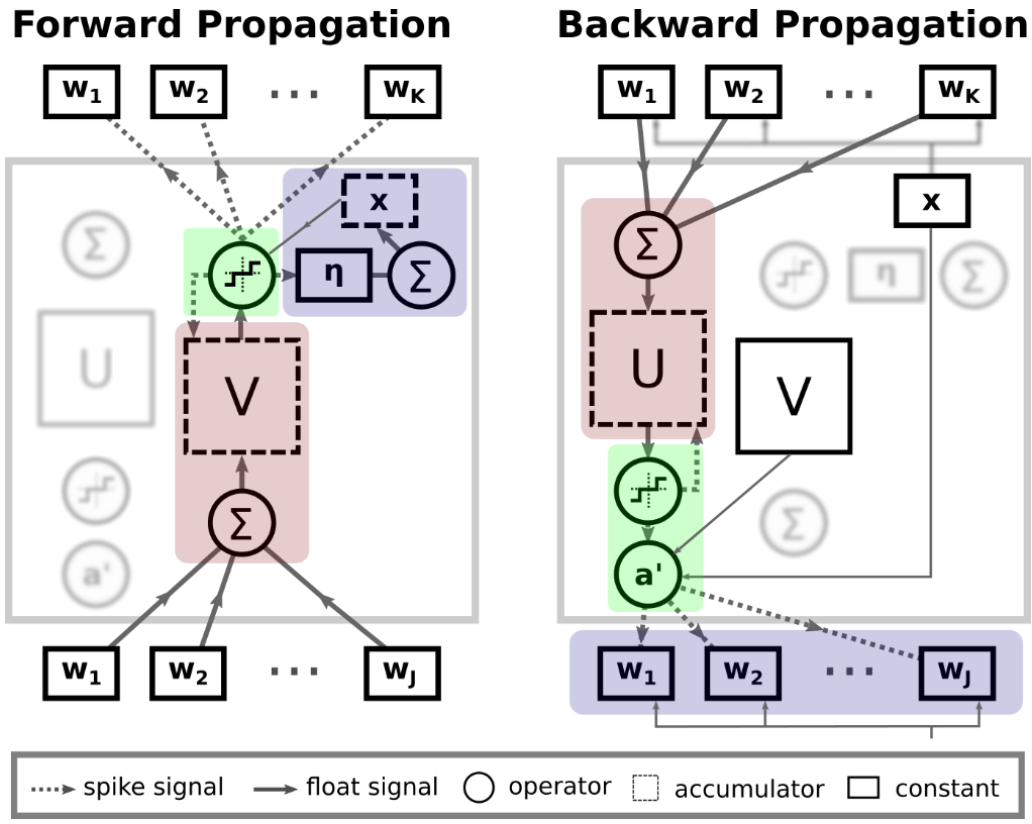


Johannes Thiele, Thesis

- **First fully event-based implementation of multi-layer backpropagation**
- **Allows multi-layer optimization**
- **Allows exact definition of objective function (supervised and unsupervised)**

EVENT-BASED IMPLEMENTATION OF THE BACKPROPAGATION ALGORITHM FOR SPIKING NEURONS

- Dynamic error ternarization by second integrator U
- Translates SNN dynamic precision to BP
- Accumulations and comparisons only
- Promising preliminary results on MNIST (99.05% with CNN)



"Ternarized gradients for efficient on-chip training of spiking neural networks", J. C. Thiele, O. Bichler & A. Dupret, Cognitive Computing 2018
"*Retro-propagation d'Erreurs Sous Forme Impulsionnelle Dans Un Réseau De Neurones Impulsionnels*", J. C. Thiele & O. Bichler , European Patent under Review

DNEURO

RTL HW library

- Set of DNN layer kernels optimized in RTL
  - Generic RTL
  - Support convolutional layers… (Fully-CNN)
    - Today: fully connected, convolution and max pooling layers
    - Plan: unit map connectivity and stride support

Fully automatic DNN RTL generation

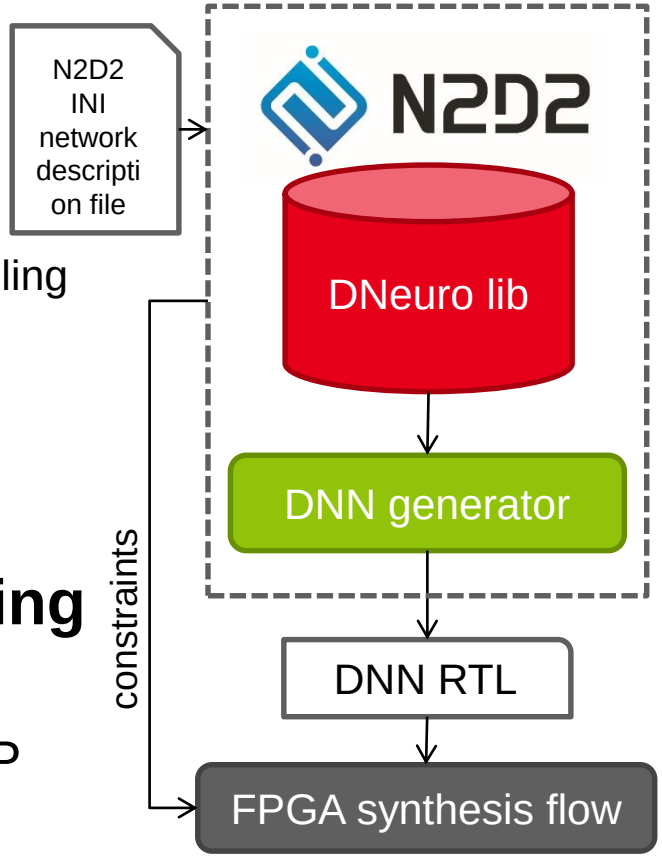
- Dataflow computation

Early projection performances (ongoing work)

- e.g. 250 GOPS on Virtex 7 UltraScale+ VU7P @100MHz (~20W)

Future works

- Performance optimizations (e.g. use of DSP, increase of frequency…)
- New DNN layers support (e.g. FasterRCNN…)

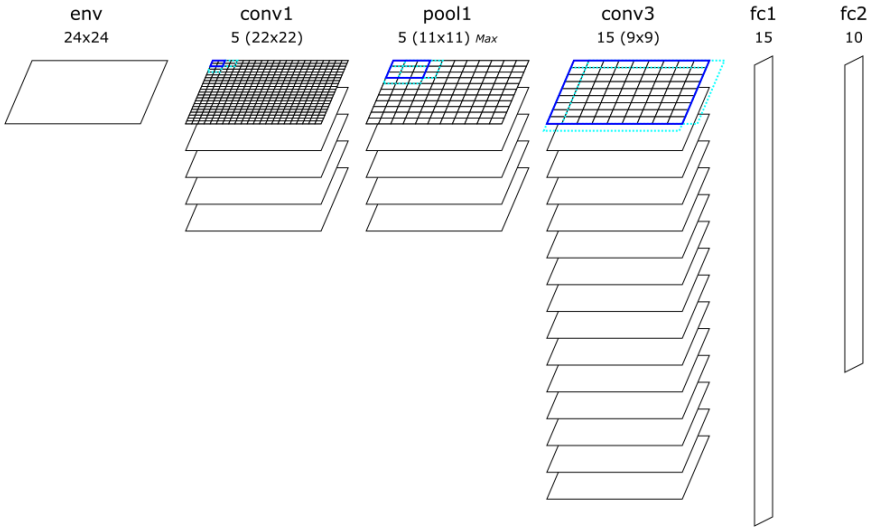


Centre de Saclay
Nano-Innov PC 172 - 91191 Gif sur Yvette Cedex

INSTITUT CARNOT CEA LIST

EXAMPLE OF SPIKE-CODING RESULTS

Network for MNIST database



	Frame DNN	Spiking DNN
Quantization	Yes	Yes
Approx. computing	No	Yes
Base operation	Multiply-Accumulate (MAC)	Accumulate only
Activation function	Non-linear function	Threshold (+ refractory period)*
Parallelism	Spatial	Spatial and temporal
Memory reutilisation	Yes	No

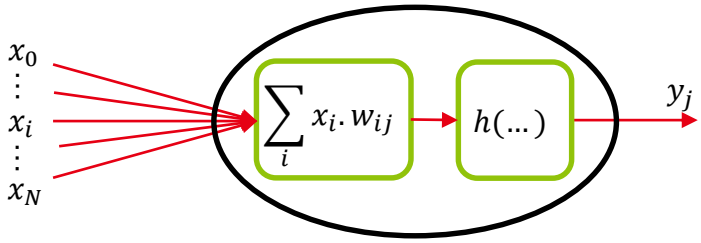
**Not required for ReLU activation function*

	Frame		Spiking		
	Score	**MACs**	**Δ**	**Score**	**spikes/MAC**
MNIST ReLU	98.1%	94k	5	**98.1%**	1.27

TRANSPOSITION PRINCIPLE

"Formal" neural network model

Multiply-Accumulate (MAC) + non-linear operation



$$y_j = h\left(\sum_i x_i.w_{ij}\right)$$

"Spiking", rate-based equation equivalence

- Neuron model: Integrate & Fire (IF)
- Neuron thresholds: $x_{th+} > 0$ and $x_{th-} < 0$
  - Integration is reset to its value minus $x_{th\ sign(n_j)}$
- Input / output spikes over duration T_{acc}: n_i / n_j

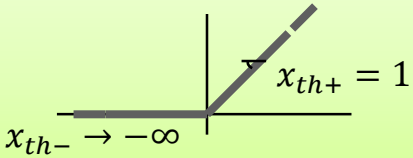
Approximation for $n_j \gg 1$:

$$\frac{n_j}{T_{acc}} \approx \frac{\sum_i n_i.w_{ij}}{\left|x_{th\ sign(n_j)}\right|.T_{acc}}$$

***h*(…) = ReLU(…) activation equivalence:**

$$\begin{cases} x_i \equiv \dfrac{n_i}{T_{acc}} \\ y_j \equiv \dfrac{n_j}{T_{acc}} \end{cases}$$

with



Mathematical convergence

➔ use the result of frame-based learning in spike

NEURON DISTRIBUTION INTO NEUROSPIKE

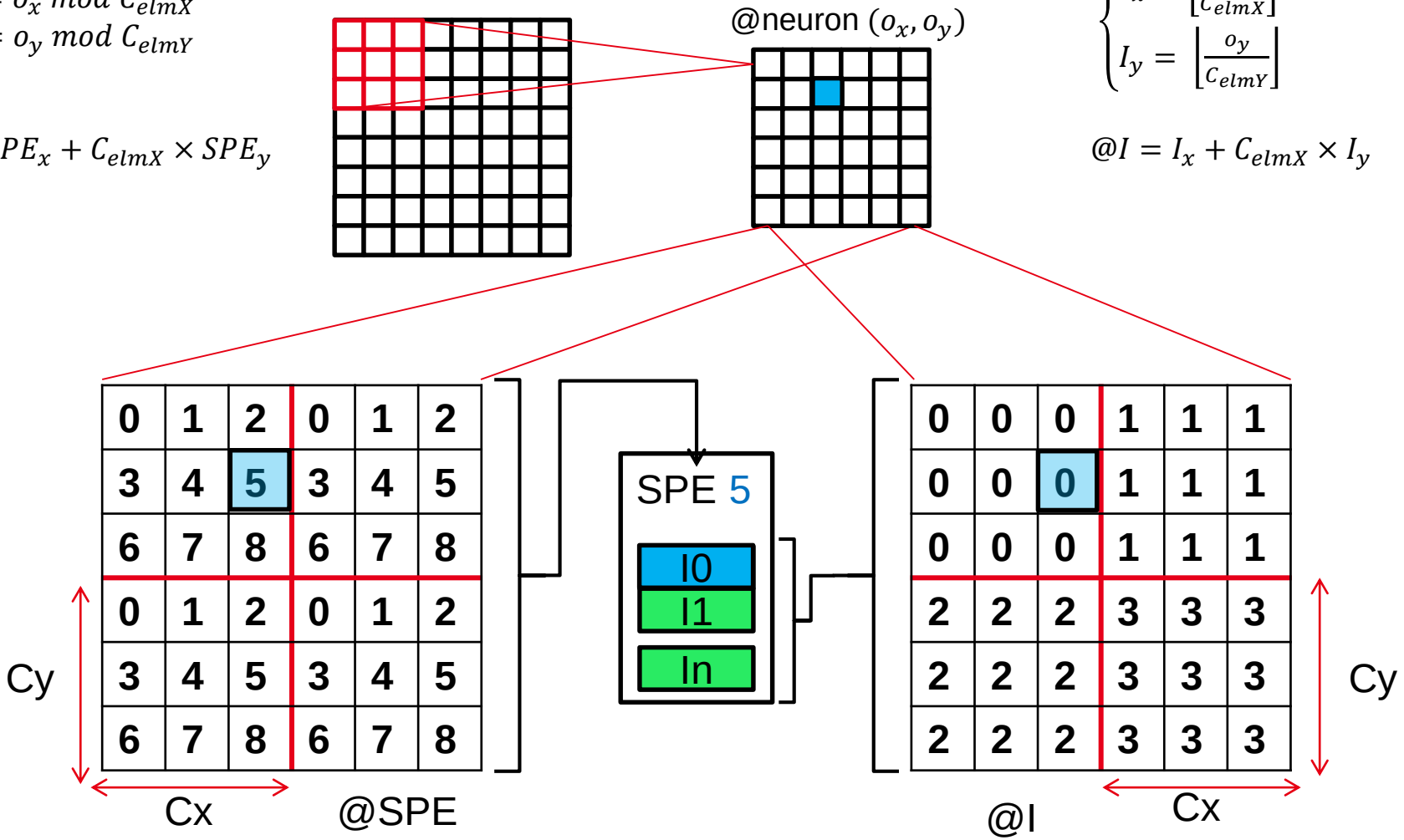
I = 8 x8, C = 3 x 3 , O = 6 x 6

$$\begin{cases} SPE_x = o_x \ mod \ C_{elmX} \\ SPE_y = o_y \ mod \ C_{elmY} \end{cases}$$

$$SPE = SPE_x + C_{elmX} \times SPE_y$$

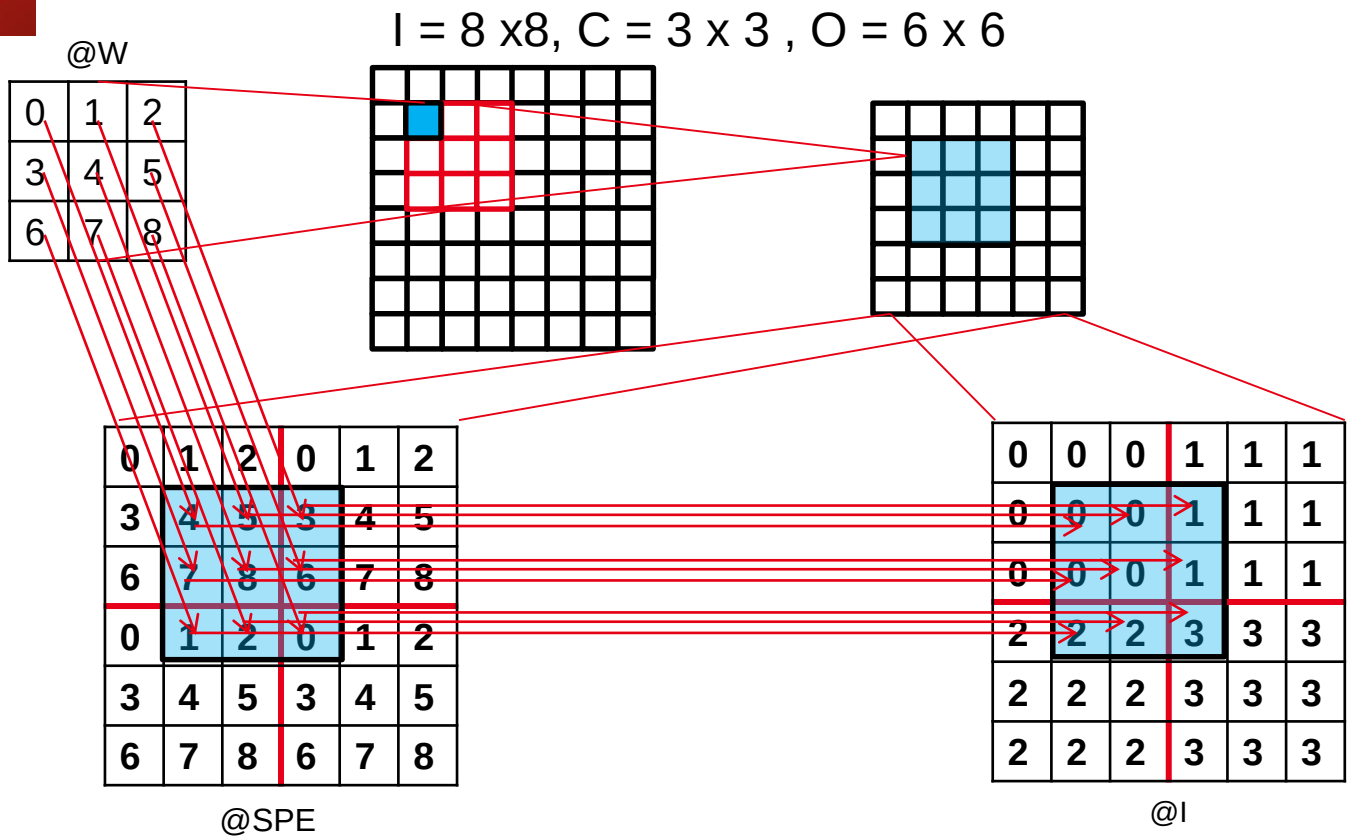
$$\begin{cases} I_x = \left\lfloor \frac{o_x}{C_{elmX}} \right\rfloor \\ I_y = \left\lfloor \frac{o_y}{C_{elmY}} \right\rfloor \end{cases}$$

$$@I = I_x + C_{elmX} \times I_y$$



In NeuronSpike the @neuron is defined by (@SPE, @I)

ADDRESSING NEURONS



Nb SPE = size of filter max ($C_{elm_{MAX}}$)

Max 4 possibilities of @I per input spike

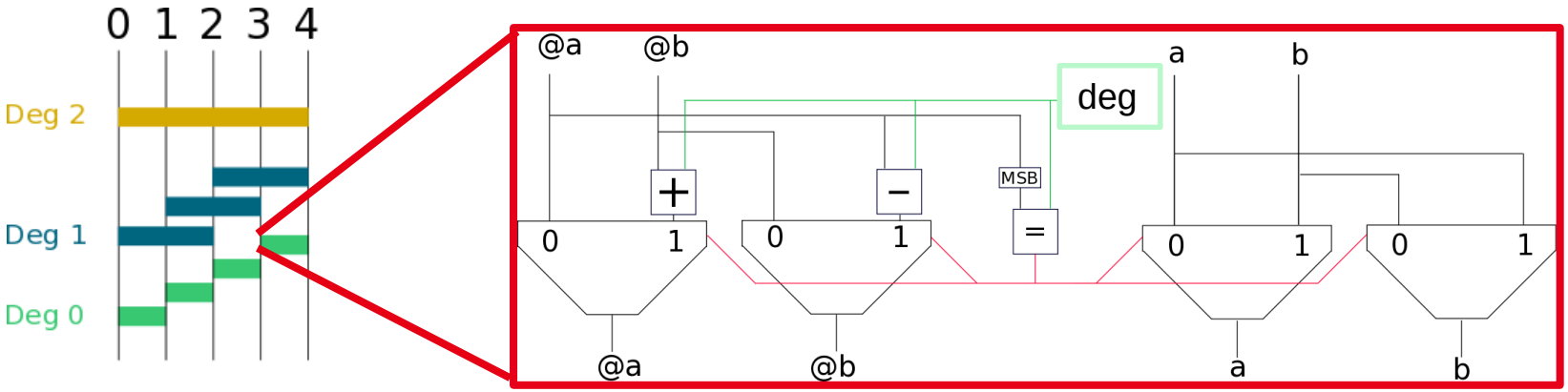
The number of SPE is independent of the number of neurons

The number of SPE = $C_{elm_{MAX}} = MAX(NbC_{neuro})$

➔Full hardware sharing between neurons

SHARING THE WEIGHTS

- The distributed @



$$\begin{pmatrix} 0 & 1 & 2 & 3 & 4 \\ 2 & 3 & 4 & 0 & 1 \end{pmatrix} = 2 \quad 2 \quad 2 \quad -3 \quad -3$$

Addressing vector based on the departure to arrival distance