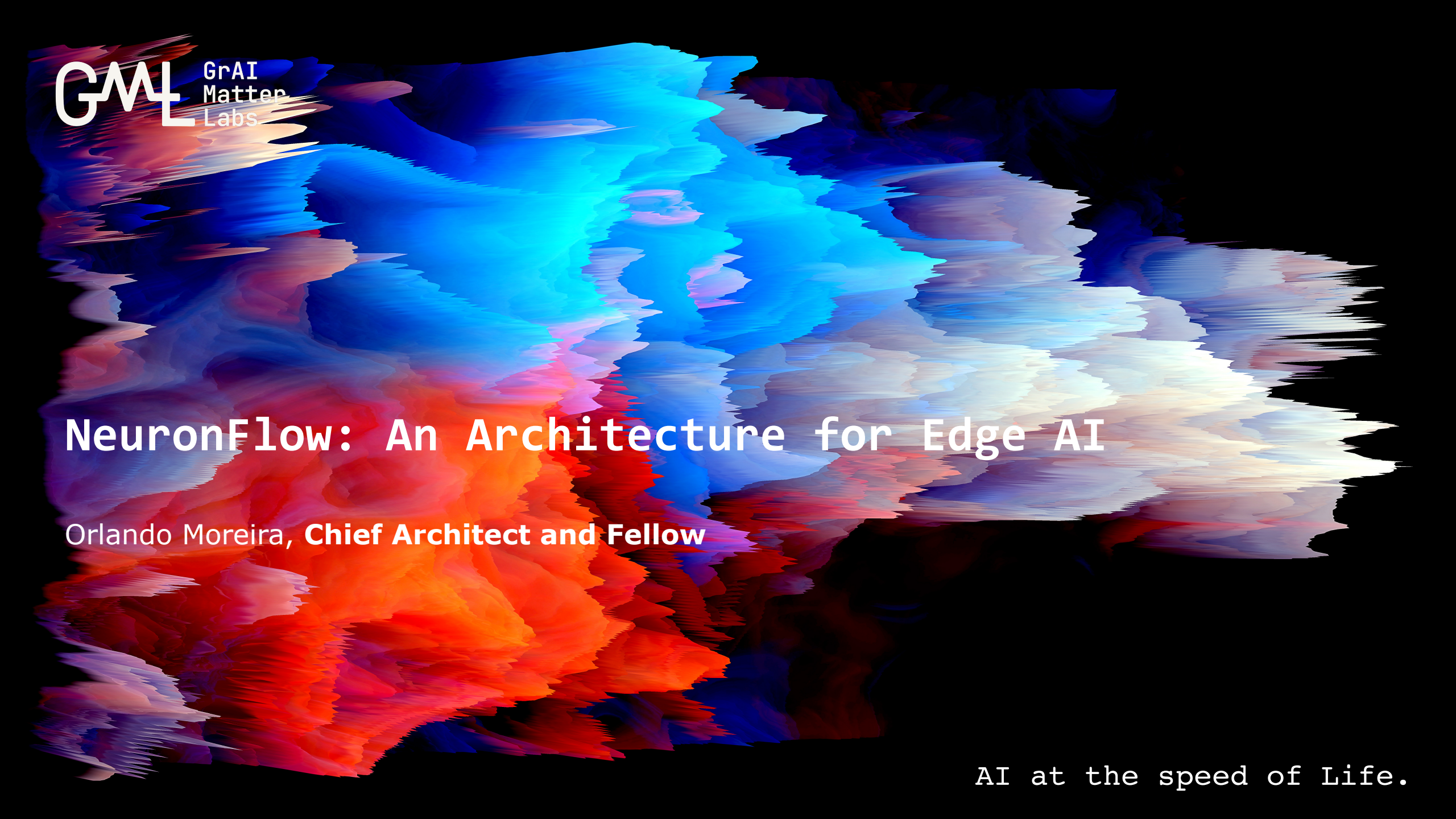


NeuronFlow: An Architecture for Edge AI

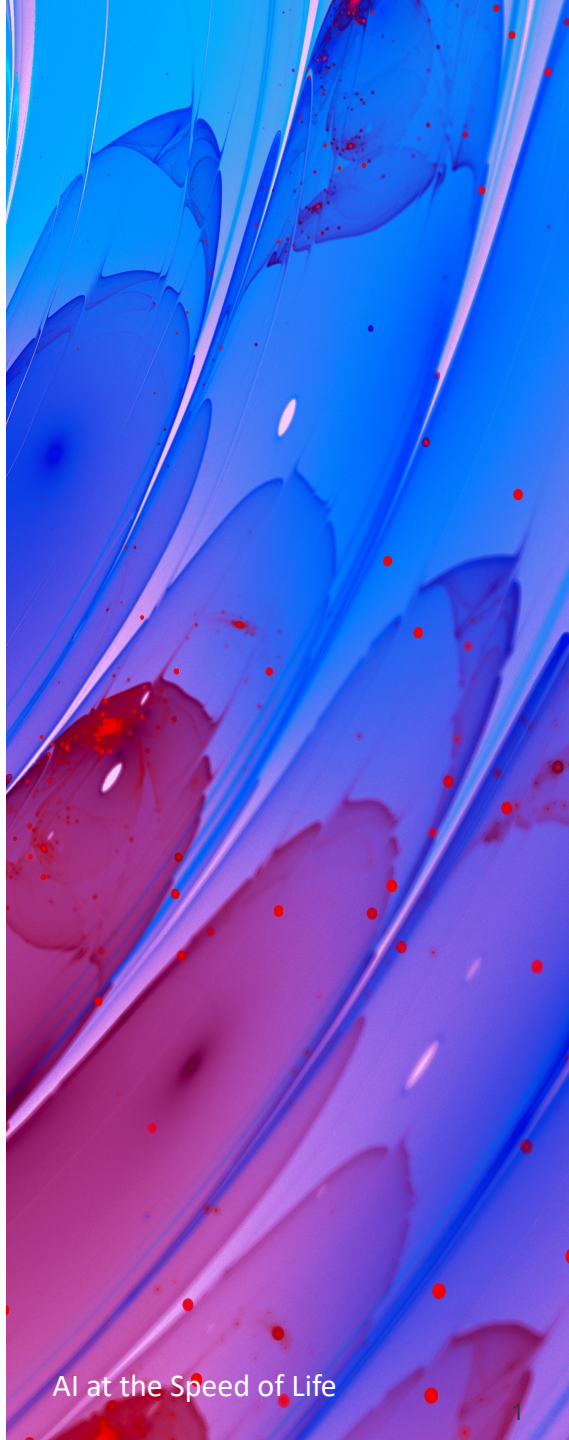
Orlando Moreira, **Chief Architect and Fellow**

AI at the speed of Life.

GrAI Matter Labs

We provide brain-inspired chips for intelligent devices at the Edge:

- Responsive
- Autonomous
- Low Power
- Low Latency



GrAI Matter Labs

Silicon Valley

- Product Marketing & Sales, CEO
- Customer Solutions

Paris

- Science Center
- System & Applications Engineering

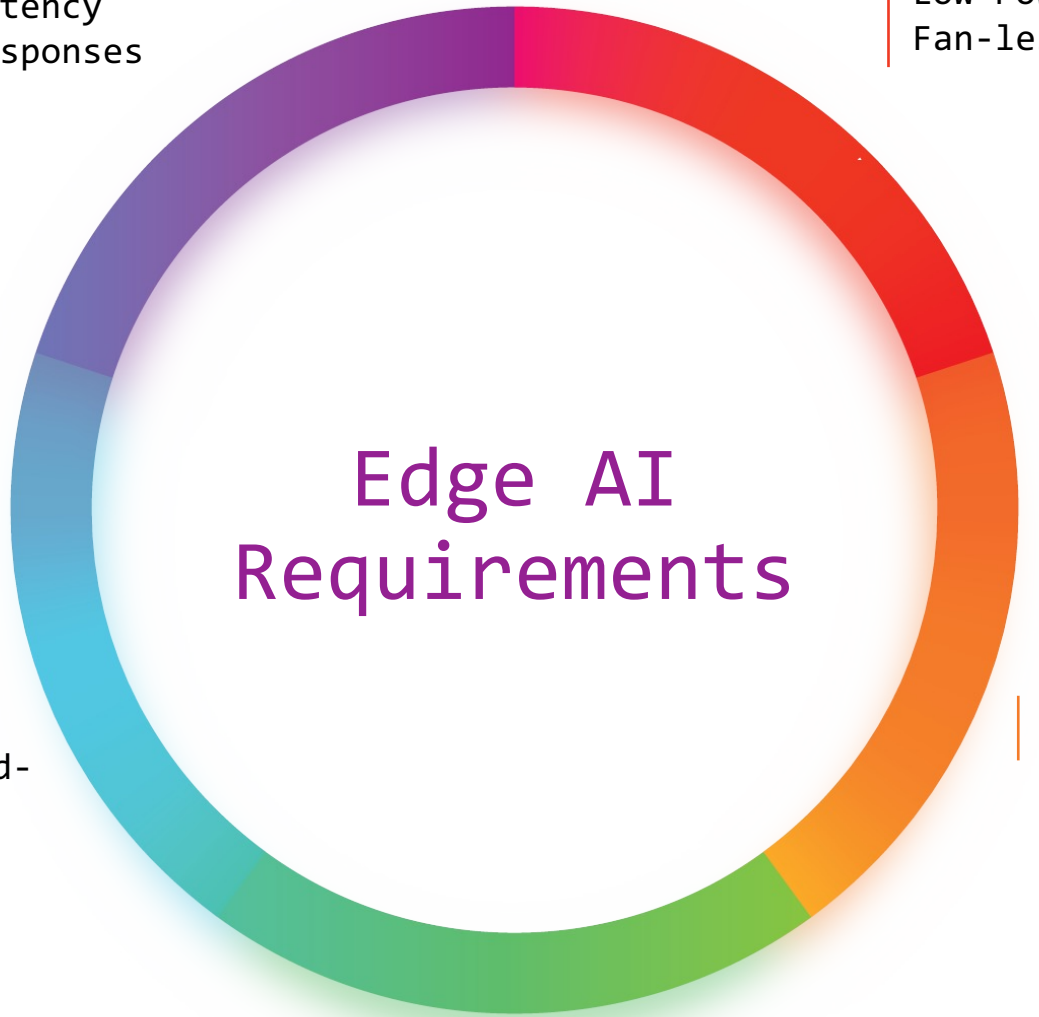
Eindhoven

- Silicon Design Center
- SDK Engineering

Edge AI Requirements

Ultra-Low latency
Real-time Responses

Low Power
Fan-less design

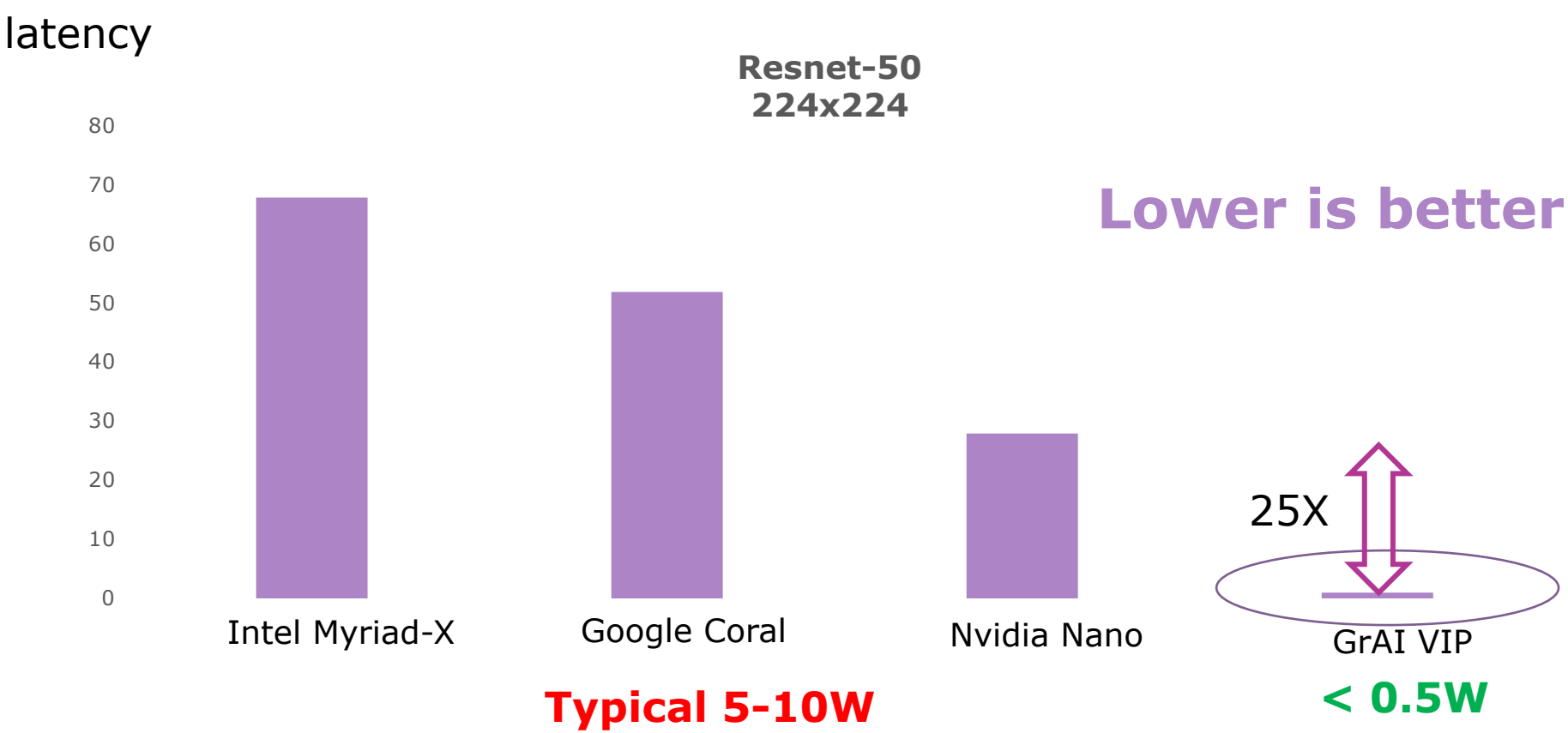


Secure & Small End-point AI

High Accuracy
Reliable and Robust

Ease of Programmability
Fast Time To Market

Life-Ready Ultra-Low Latency at Low Power

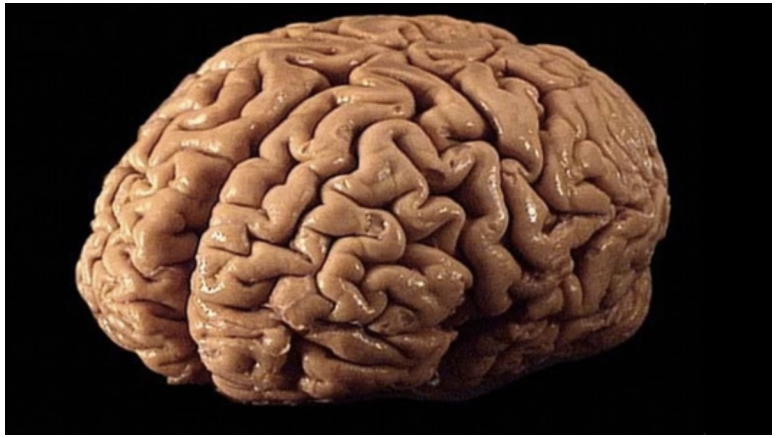


https://coral.ai/docs/edgetpu/benchmarks/
https://docs.openvinotoolkit.org/latest/openvino_docs_performance_benchmarks.html#resnet_50_tf
https://developer.nvidia.com/embedded/jetson-benchmarks
GrAI VIP based on GrAIFlow SDK estimates

Brain inspired Computing

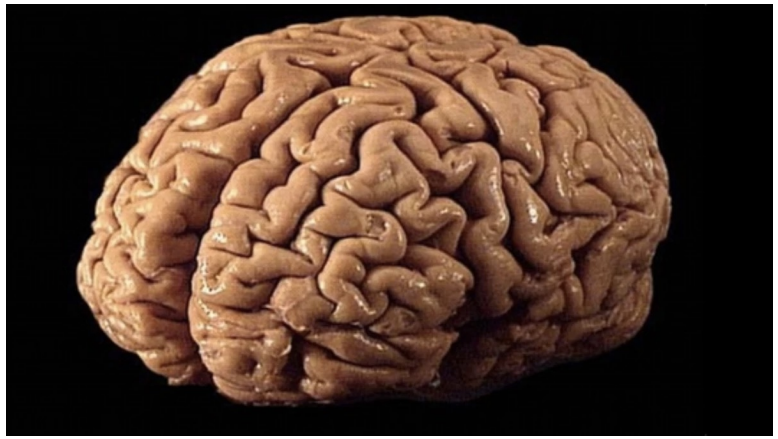
Brain-inspired computing: scalability

- Very simple distributed structure replicated over and over



Brain-inspired computing: low power

- Animal vision much more power efficient than silicon.
- There must be something we can learn from it.
- This leads us to bio-mimicry.
- But how does the brain do what it does?



Vision System	Power consumption
Human Vision	6 W
GPU	~250W

Neuromorphic Model: Spiking Neural networks

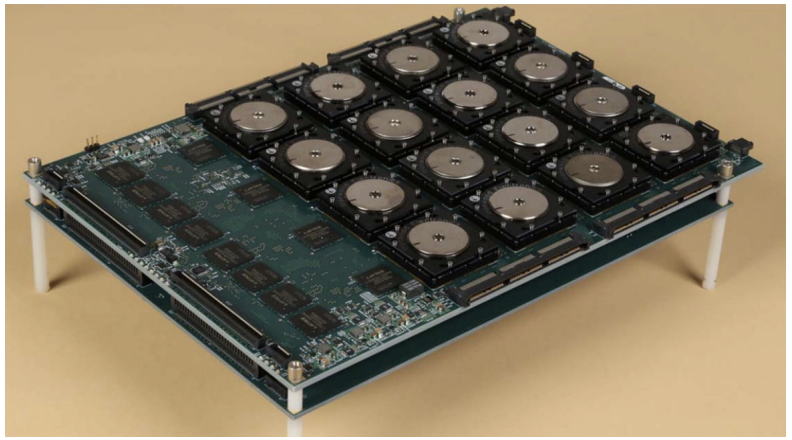
Artificial neurons that closely mimic brain cell behavior:

- neurons communicate through **value-less** spikes.
- weighted synapses with **delayed** spike transport
- neurons with **persistent state**
- temporal execution model
  - synaptic relay of spikes includes a **temporal delay**
  - neuron state **decays** over time
  - requires a global concept of time and time base functions

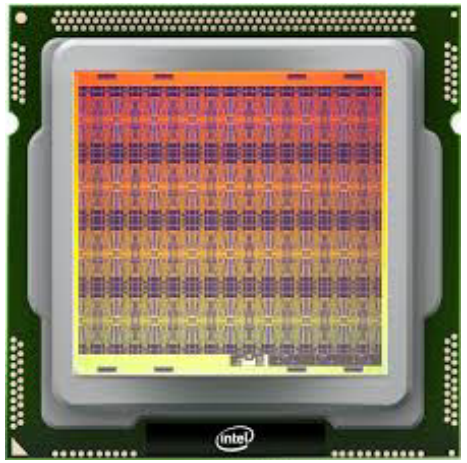
Challenges for Neuromorphic going Digital

- spikes offer little/no gain over values due to addressing and memory access overhead.
- temporal behavior: simulating ”brain” time requires costly global synchronization.
- exponential decay of neuron state is very costly

IBM’s TrueNorth



Intel’s Loihi



Avoiding the pitfalls of biomimicry

We discovered how to fly by
getting inspiration from birds
but not by exactly mimicking birds.

Focus on the secret sauce.

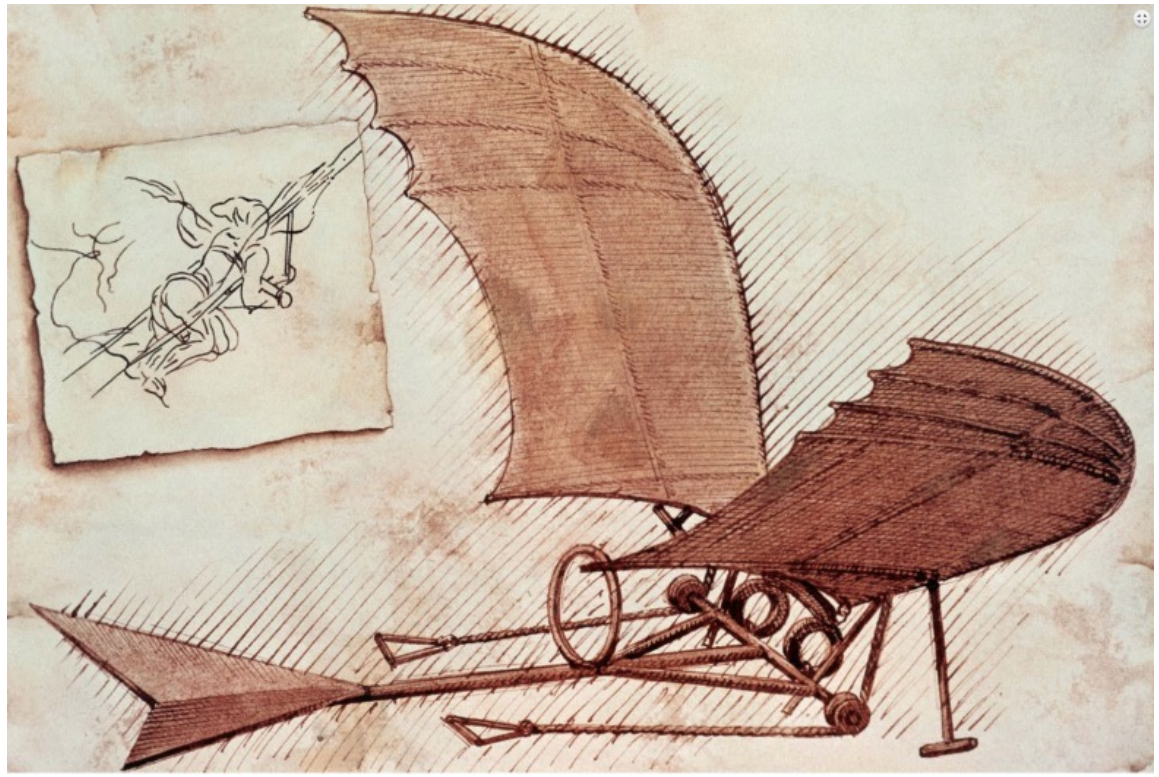


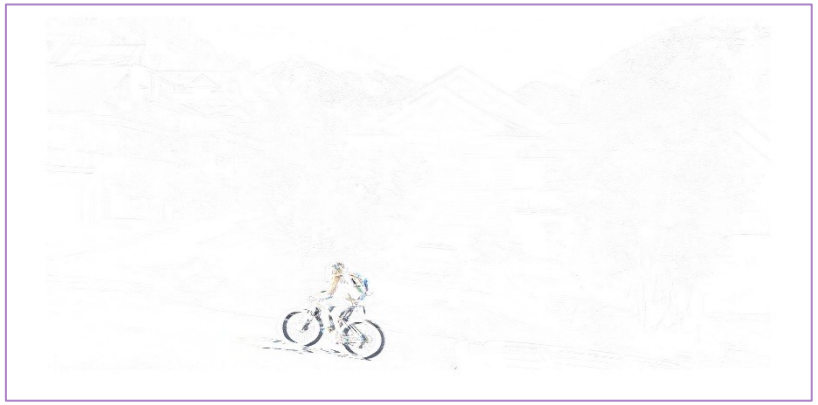
Image from www.leonardodavinci.net

"Just as the Wright brothers did not design an aircraft with wings that flap, but still gained inspiration from observing how birds glide and turn, a practicable approach for replicating animal-like intelligence is to combine mimicry of selected aspects of neurobiological solutions with entirely different implementation mechanisms, such as silicon-based electronics."

McDonnel et al "Engineering intelligent electronic systems based on computational neuroscience"

Sparsity

Sparsity

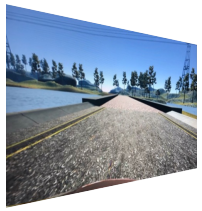


- **Sparsity in structure**
  - Pruning of needless weights
    - *In many image-processing networks **>70%** weights can be pruned without significant loss of accuracy.*

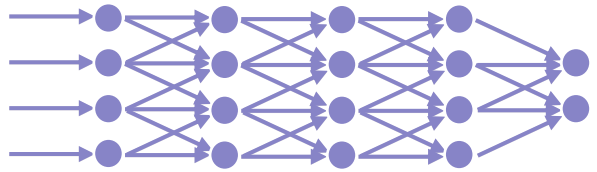
  Sparsity in activation
  - Many pixels have no relevant feature data.
    - resulting in 0-valued activations.
    - *With RELU: ~50% of activations are 0-valued*
      - *Even without training for activation suppression!!!*
- **Sparsity in time**
  - Image changes little from instant to instant
    - *why should we always re-process whole frames?*

Key To Life-Ready AI: Exploiting Sparsity

GPUs, NPUs..



Process everything

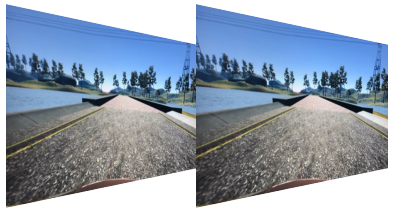


Sparse Compute

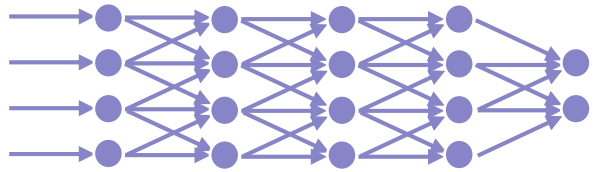
- Sparsity in time
- Sparsity of connections
- Sparsity in activation

Key To Life-Ready AI: Exploiting Sparsity

GPUs, NPUs..



Process everything

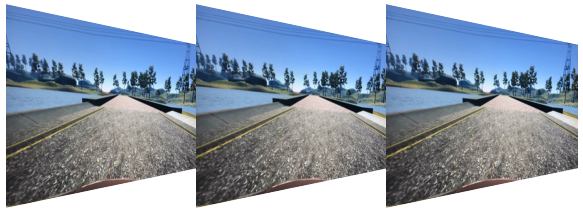


Sparse Compute

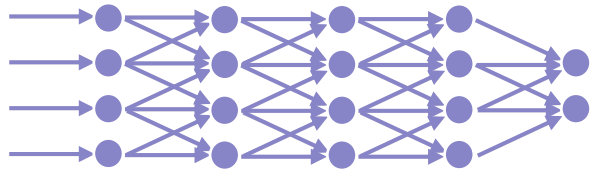
- Sparsity in time
- Sparsity of connections
- Sparsity in activation

Key To Life-Ready AI: Exploiting Sparsity

GPUs, NPUs..



Process everything

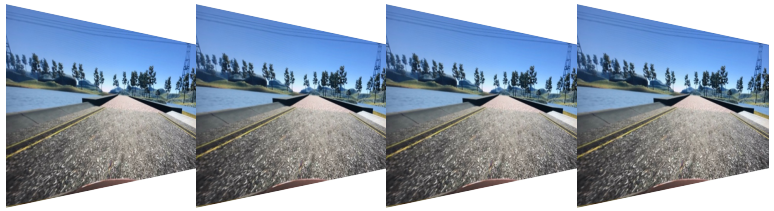


Sparse Compute

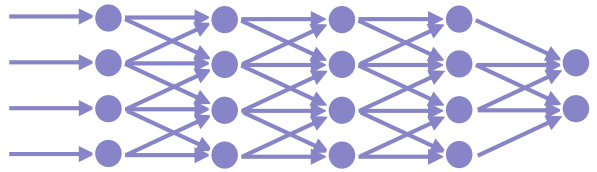
- Sparsity in time
- Sparsity of connections
- Sparsity in activation

Key To Life-Ready AI: Exploiting Sparsity

GPUs, NPUs..



Process everything



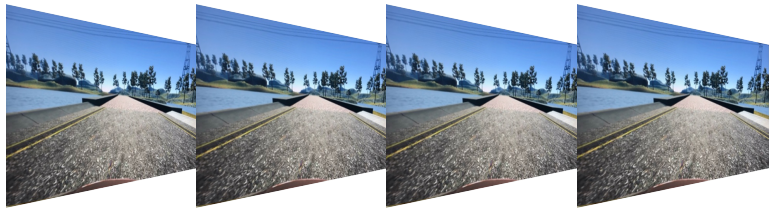
Sparse Compute

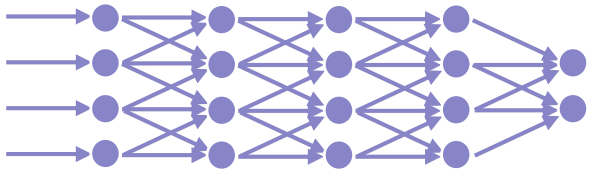
- Sparsity in time
- Sparsity of connections
- Sparsity in activation

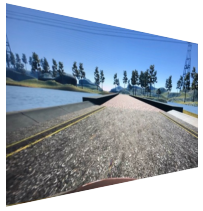
Key To Life-Ready AI: Exploiting Sparsity

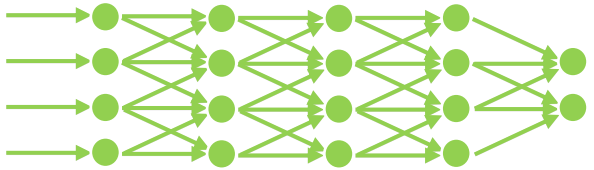
GPUs, NPUs..

Process everything









GrAI Matter Labs

NeuronFlow

Process only the changes

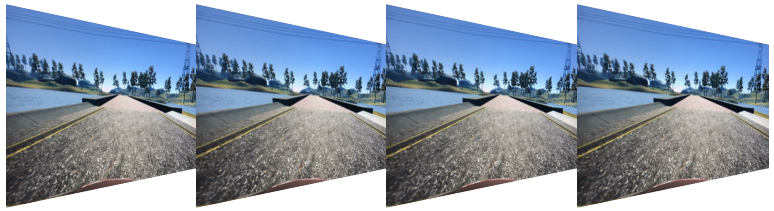
Sparse Compute

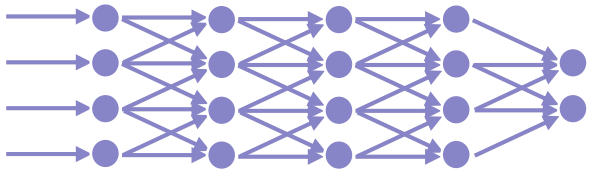
- Sparsity in time
- Sparsity of connections
- Sparsity in activation

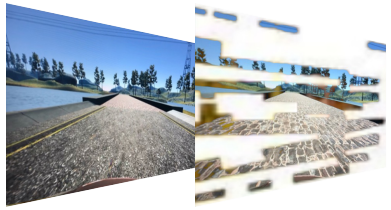
Key To Life-Ready AI: Exploiting Sparsity

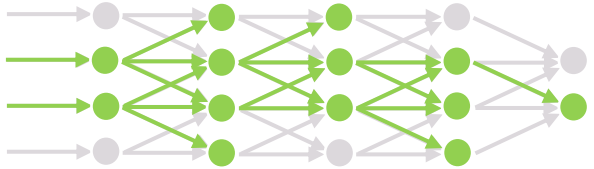
GPUs, NPUs..

Process everything









GrAI Matter Labs

NeuronFlow

Process only the changes

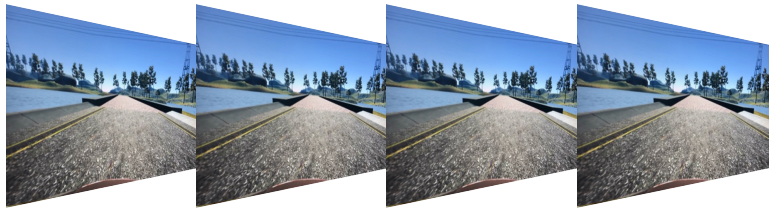
Sparse Compute

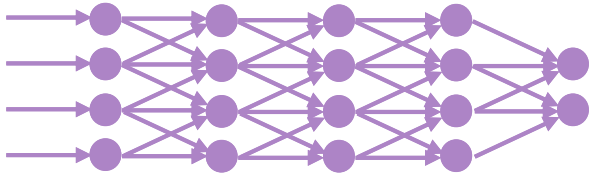
- Sparsity in time
- Sparsity of connections
- Sparsity in activation

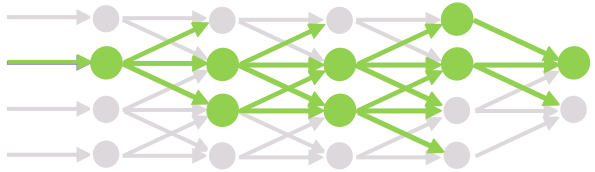
Key To Life-Ready AI: Exploiting Sparsity

GPUs, NPUs..

Process everything







GrAI Matter Labs

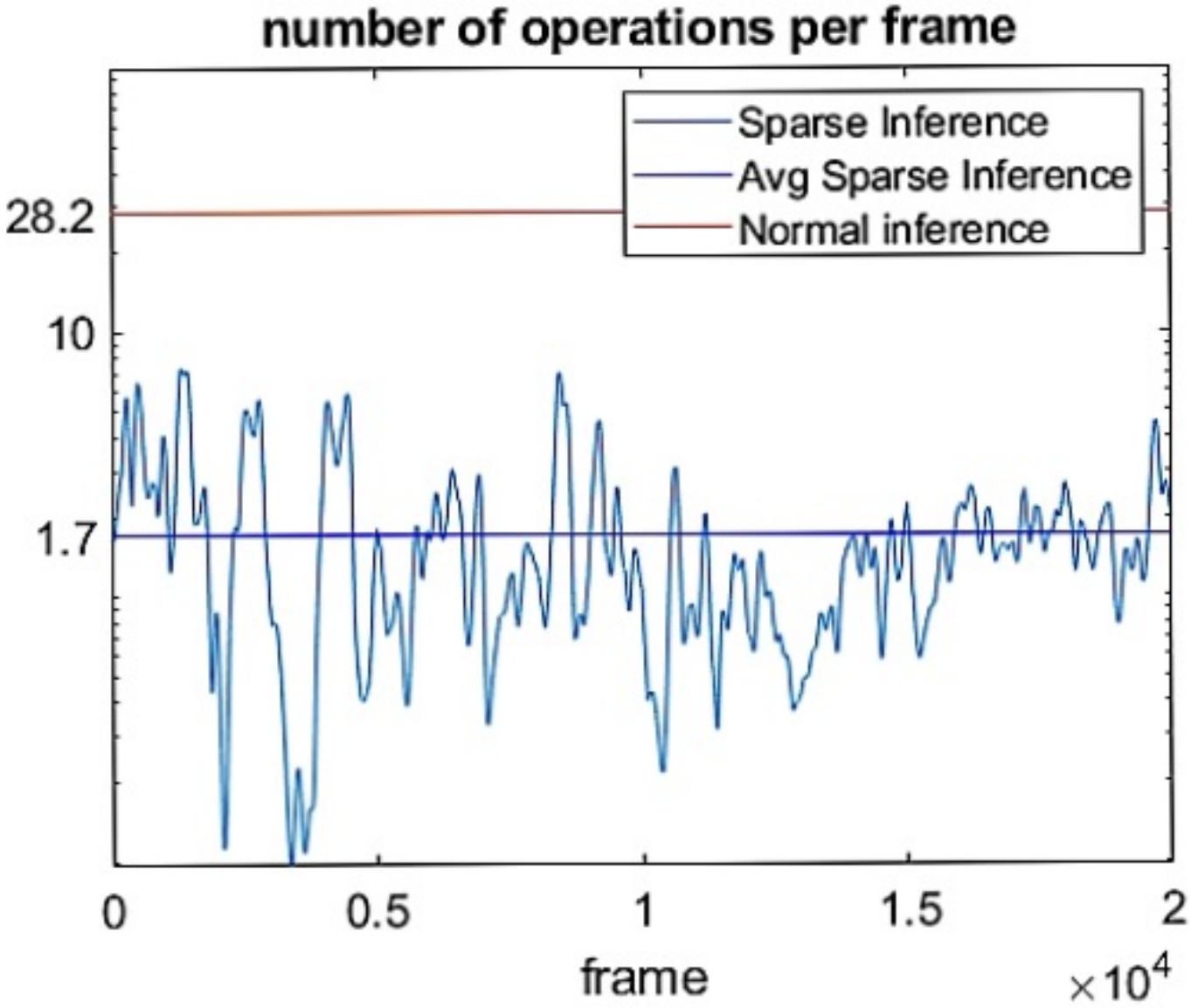
NeuronFlow

Process only the changes

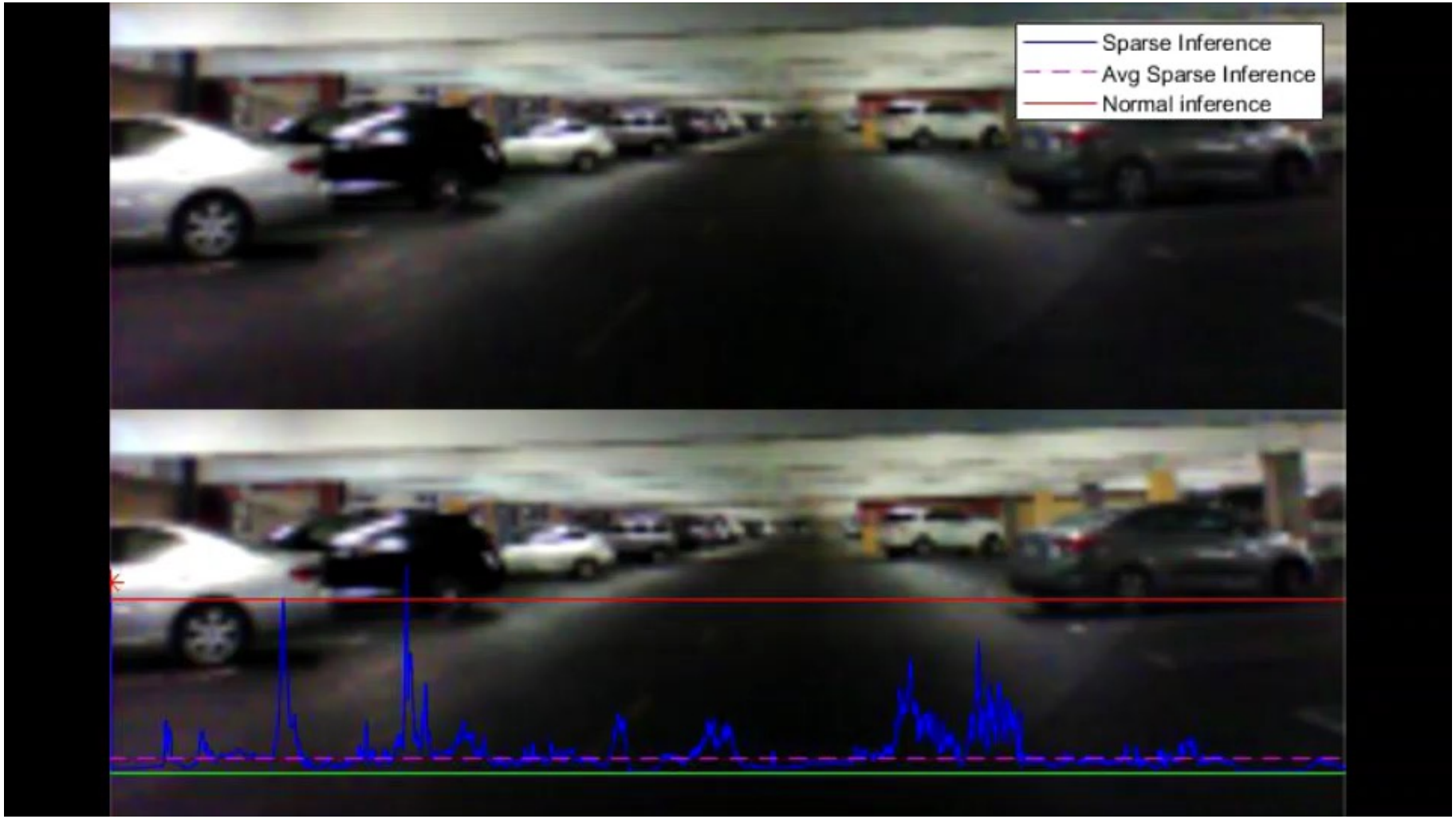
Sparse Compute

- Sparsity in time
- Sparsity of connections
- Sparsity in activation

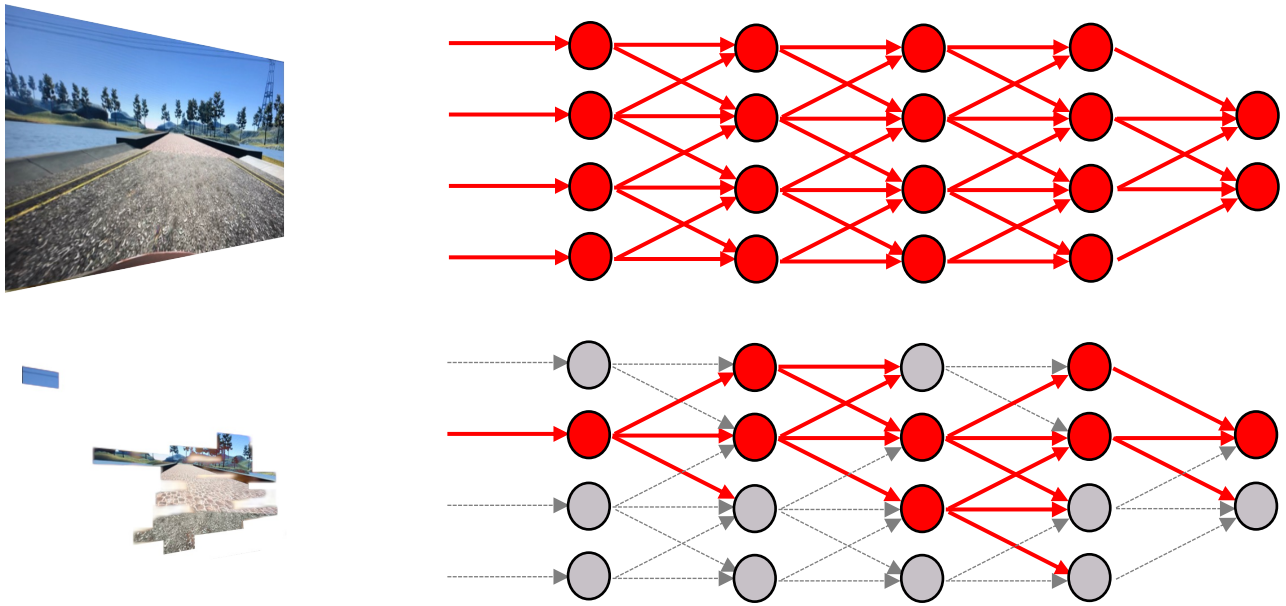
Sparsity in Pilot Net



PilotNet in SparNet



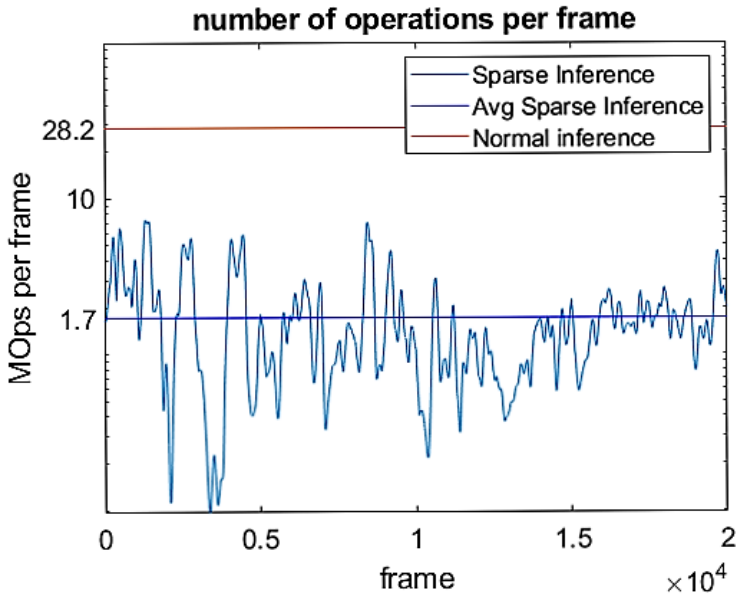
SparNet: Sparse and Event-Based execution model

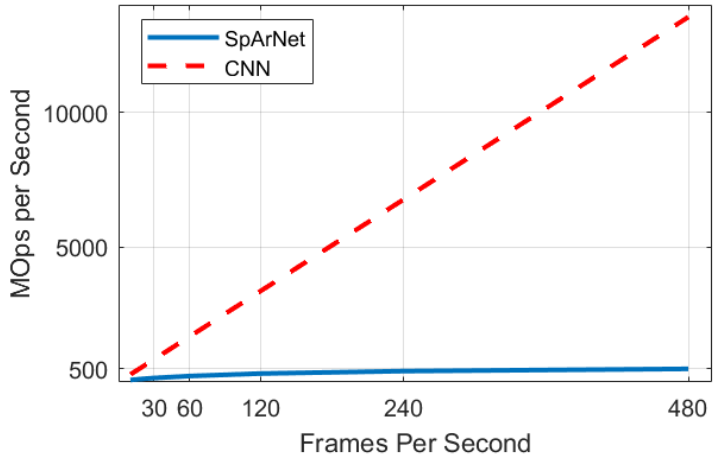


Red = active links and activated neurons

- Exploits time-sparsity in a time series;
- Converts frame-based network to **event-based** inference;
- **Event-based:** change is sent sporadically, so **no frame structure** to input data;
- **Only propagates changes,** thus less work needs to be done;
- Requires **resilient** neuron state;
- **Threshold:** per neuron, defines how much change is needed to warrant propagation.
- To convert a CNN to SparNet, we set a threshold per neuron.

PilotNet using SparNet

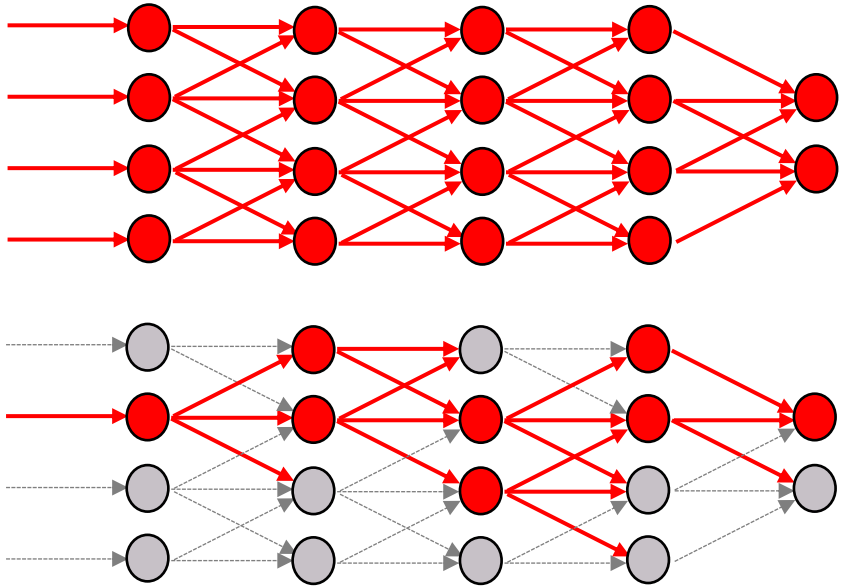


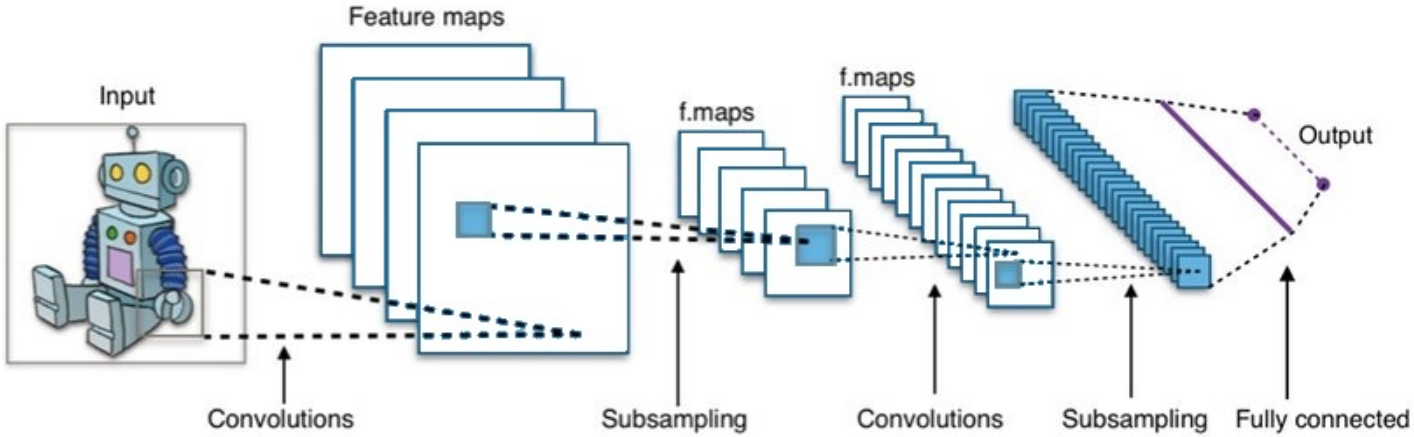


- SparNet dramatically reduces the number of operations required.
- Effect is dramatic at high fps:
  - same amount change per same time interval;
  - but for frame-based processing, load increases linearly with frame rate;
  - higher fps => lower sampling period => lower latency

Consequences of Sparsity Exploitation for Computer Architecture

- **Temporal sparsity requires resilient neurons**
  - suggests **in/near memory computation;**
- **Sparsity reduces regularity in compute demand:**
  - fewer sequential memory accesses as sparsity increases;
    - reduced value of caching, network bursts, dma
    - Eg: *for* loop over input array leads to resource wastage;
  - suggests **event-driven scheduling;**





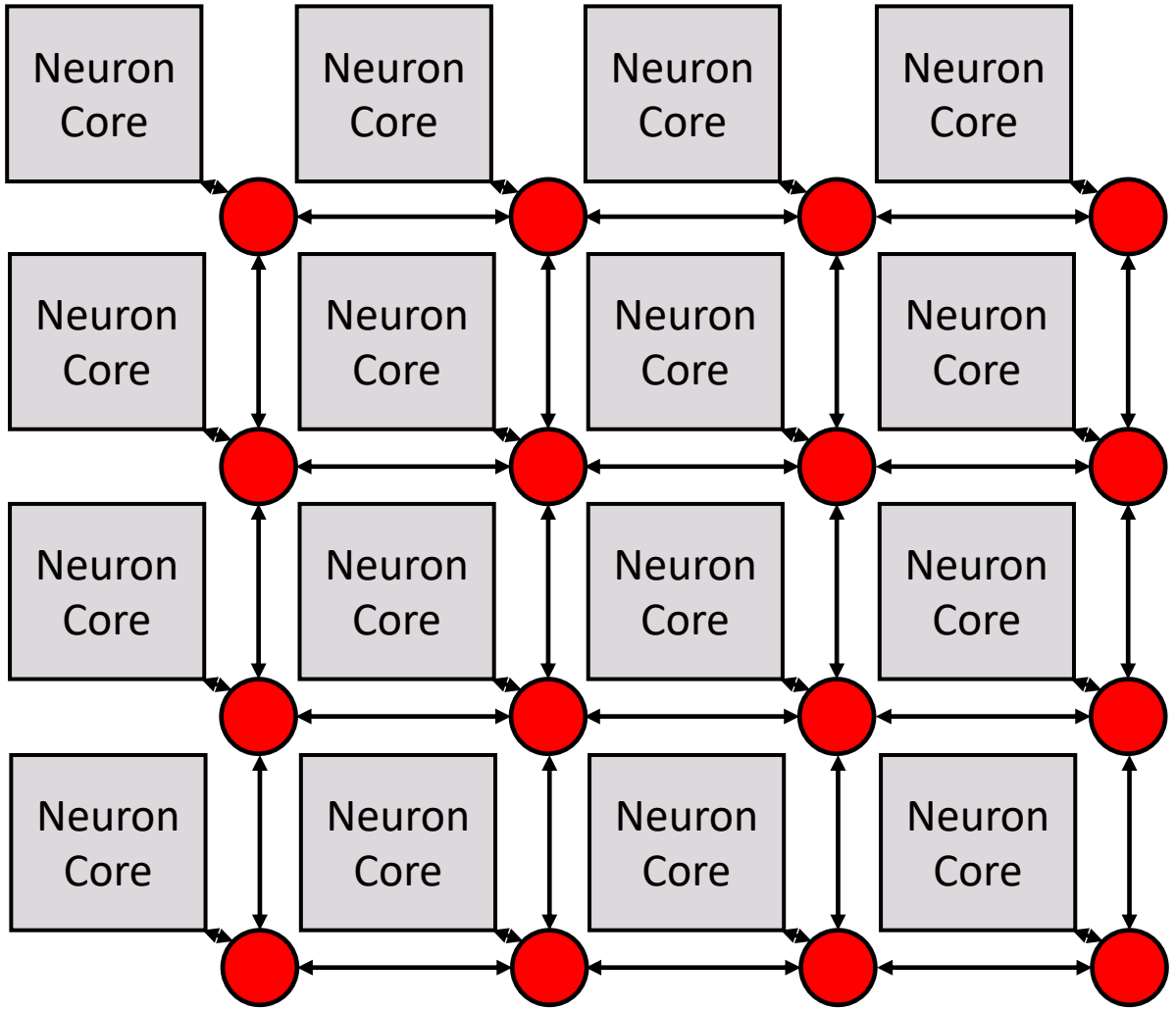
Impact of training in perfomance

- **Activation suppression** yields great benefits
  - event-driven: each event suppressed eliminates all computation triggered by the event…
  - particularly efficient for RELU activation
- **Weight pruning** removes many computations
  - 70% weight pruning in image networks is not uncommon.
- **Weight pruning and quantization** affect mappability by dramatically reducing memory requirements.

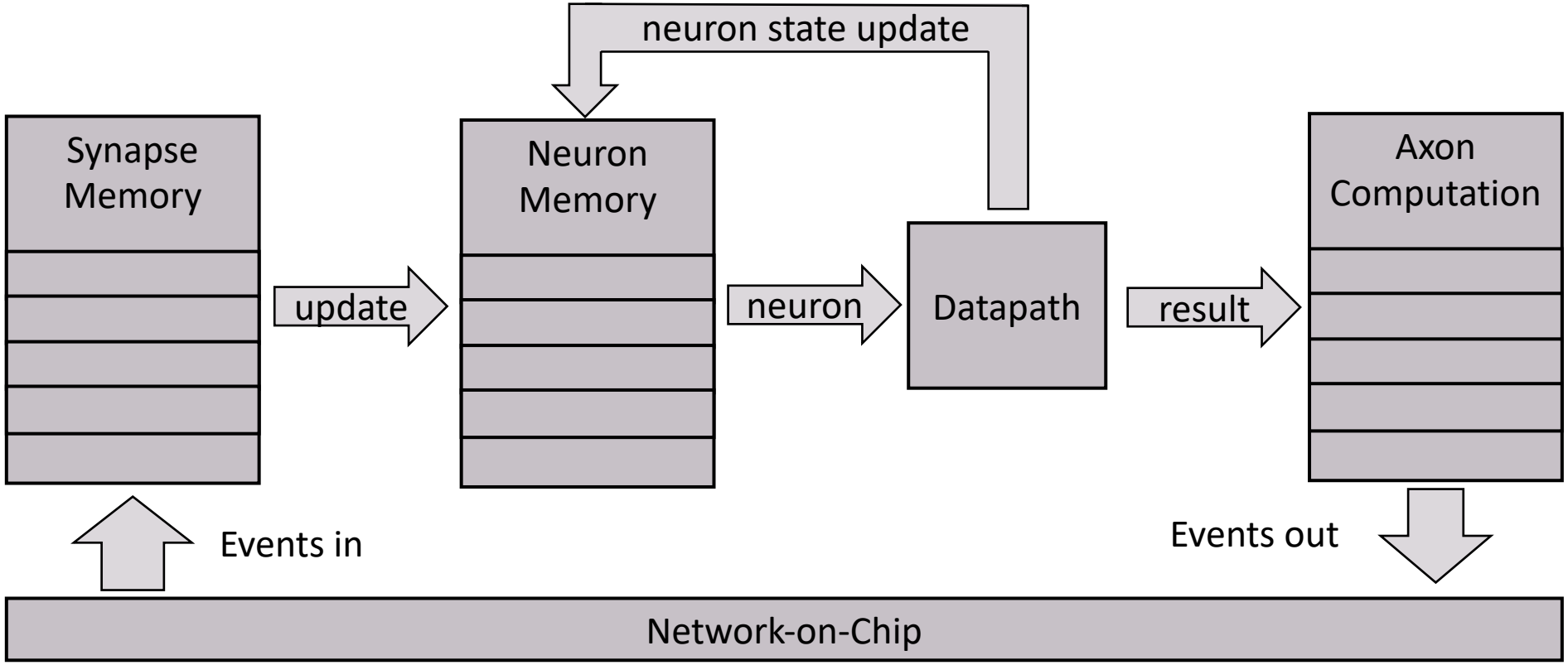
The Neuronflow Architecture

Neuronflow Array

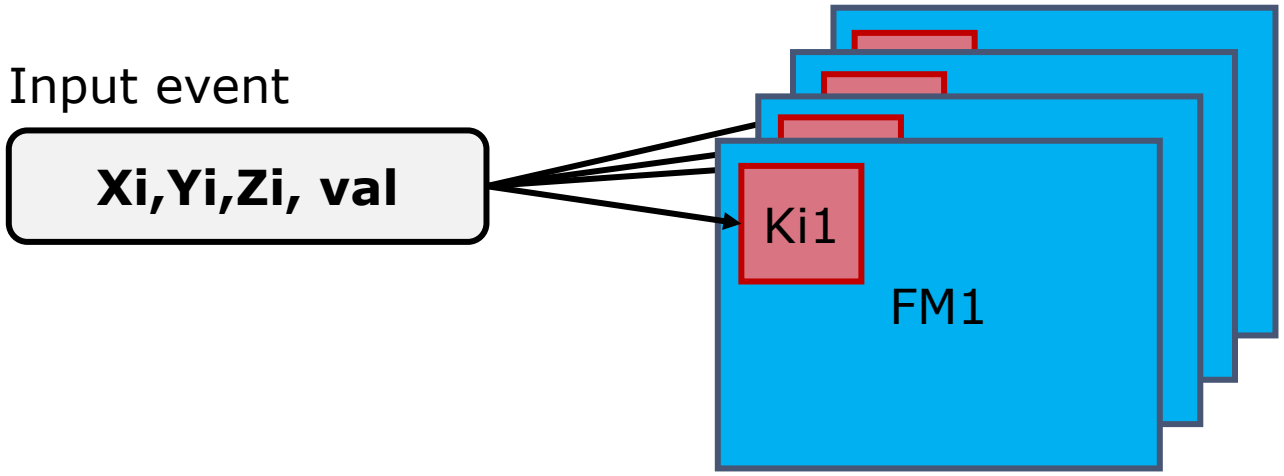
- Sizeable memory per core
  - local storage of neuron states and weights
  - near-memory computation
- **Scalable**, distributed execution.
- Sparse computation, **event-driven** scheduling:
  - avoid bulk data movement
  - in-place updates;
- data-flow synchronization.
- **axon computation unit** dramatically reduces network configuration memory
- **GrAI ViP:** 12x12 cores, 256KB mem per core
- **GrAI ViP:** FP16 SIMDx4 datapath

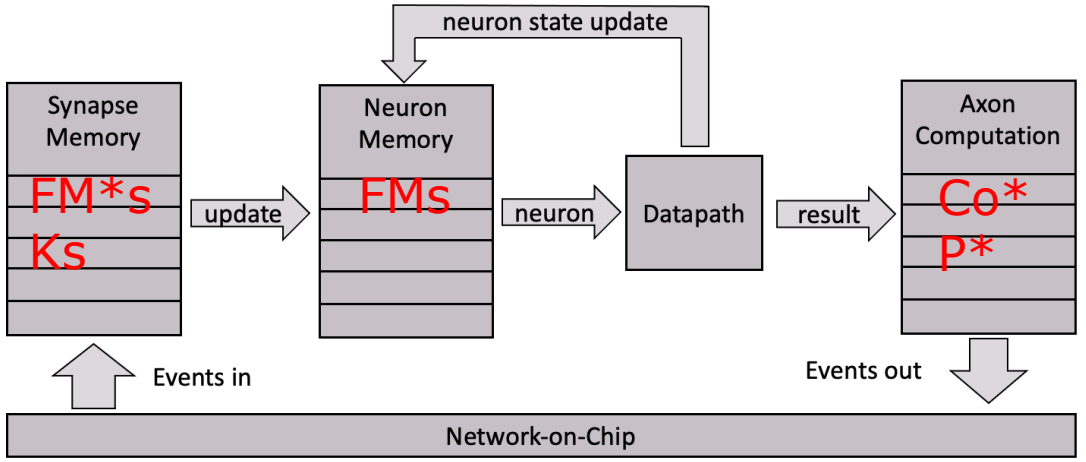


NeuronFlow Core



Core memory layout

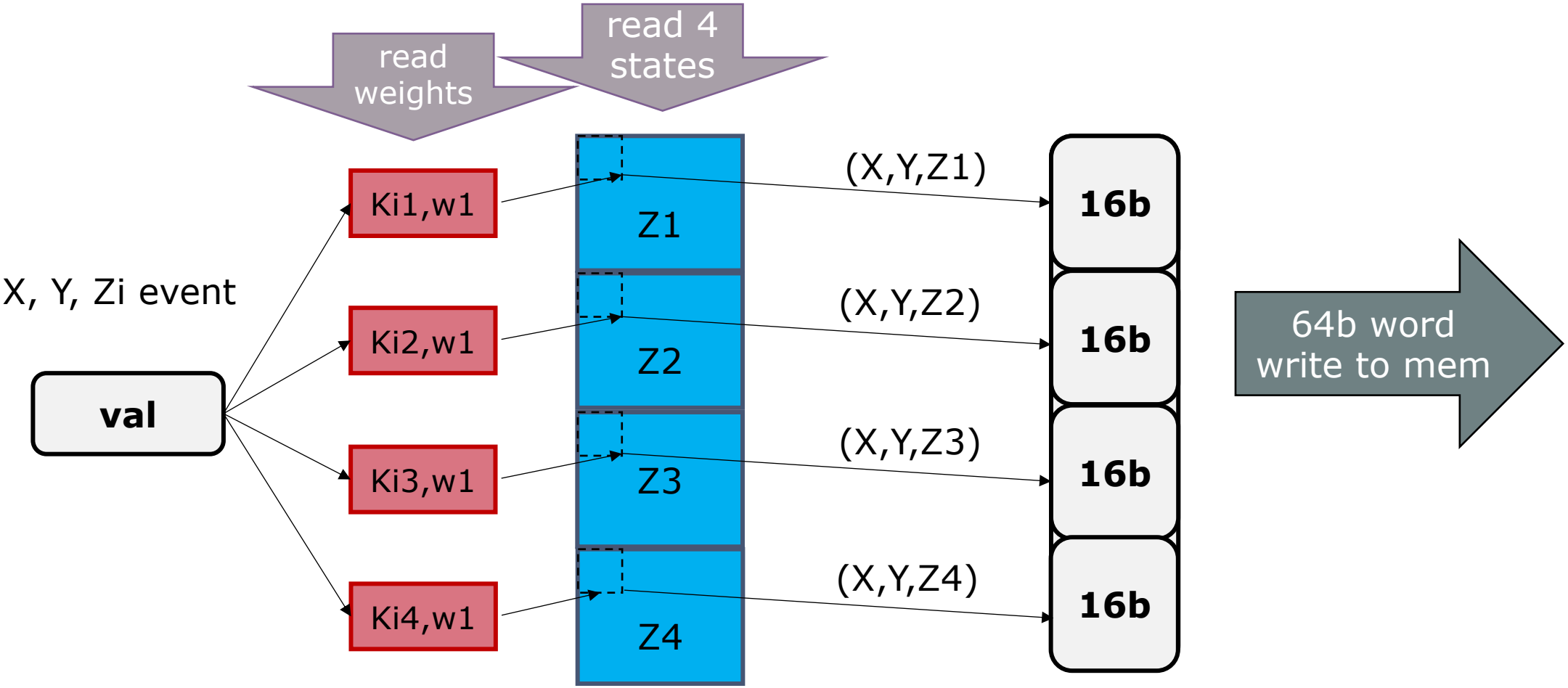




- **Synapse memory:** entries represent **neuron populations**
  - (Xi, Yi, Zi) to a 3D map of weight kernels and output neuron addresses
- **Neuron memory:** stores output neuron states (FMs)
- **Axon memroy:** send output event (Xo,Yo,Zo) to target (Core, Population)

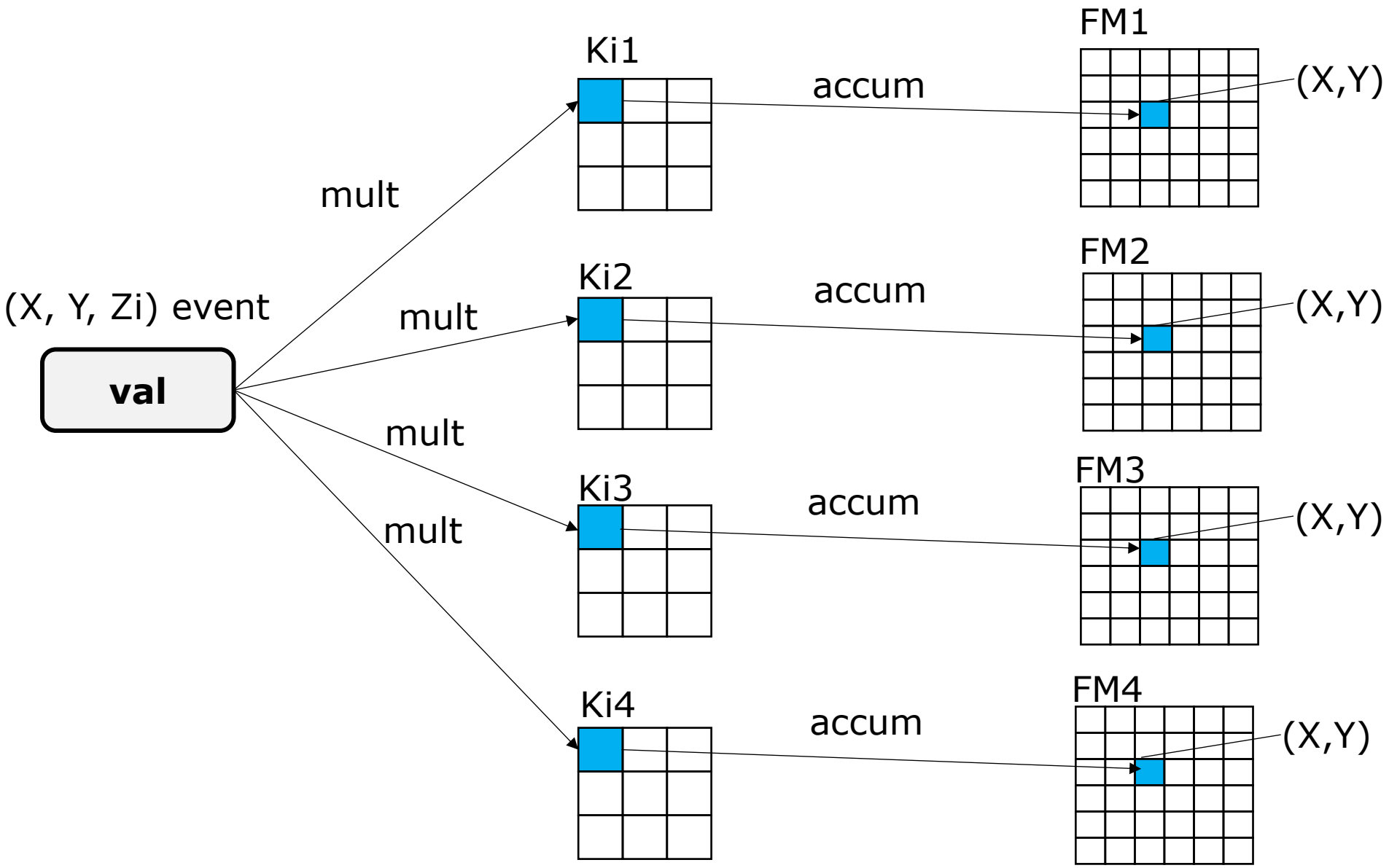
#MACs triggered per input event: #FMs * K^2 -> computational intensity

Input-stationary and SIMD

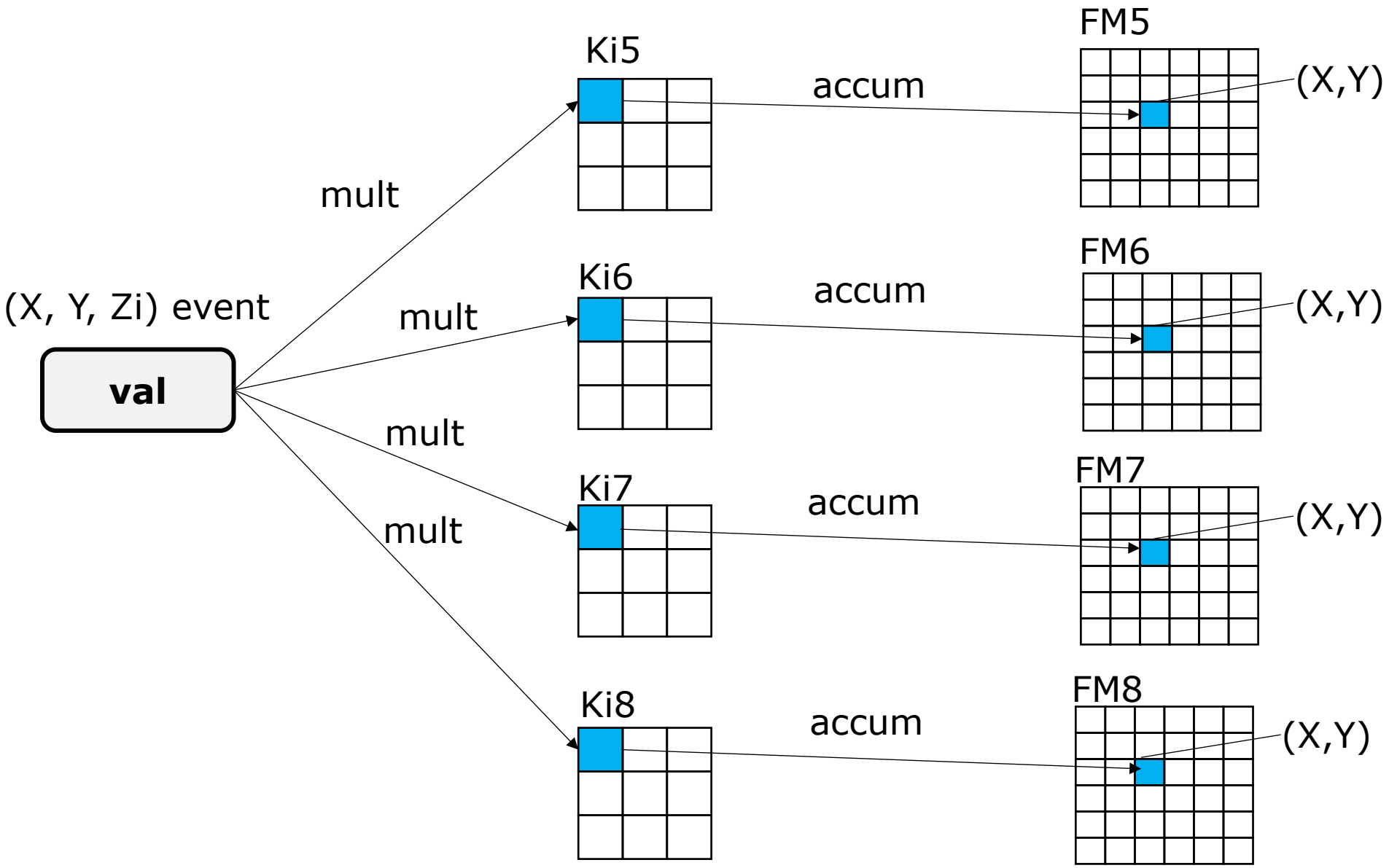


- SIMD: static data paralelism in Z (channel first)
- 4 channels per cycle, 1 weight per kernel per cycle
- Per cycle read weights, neuron states, write neuron states, Z first, then XY

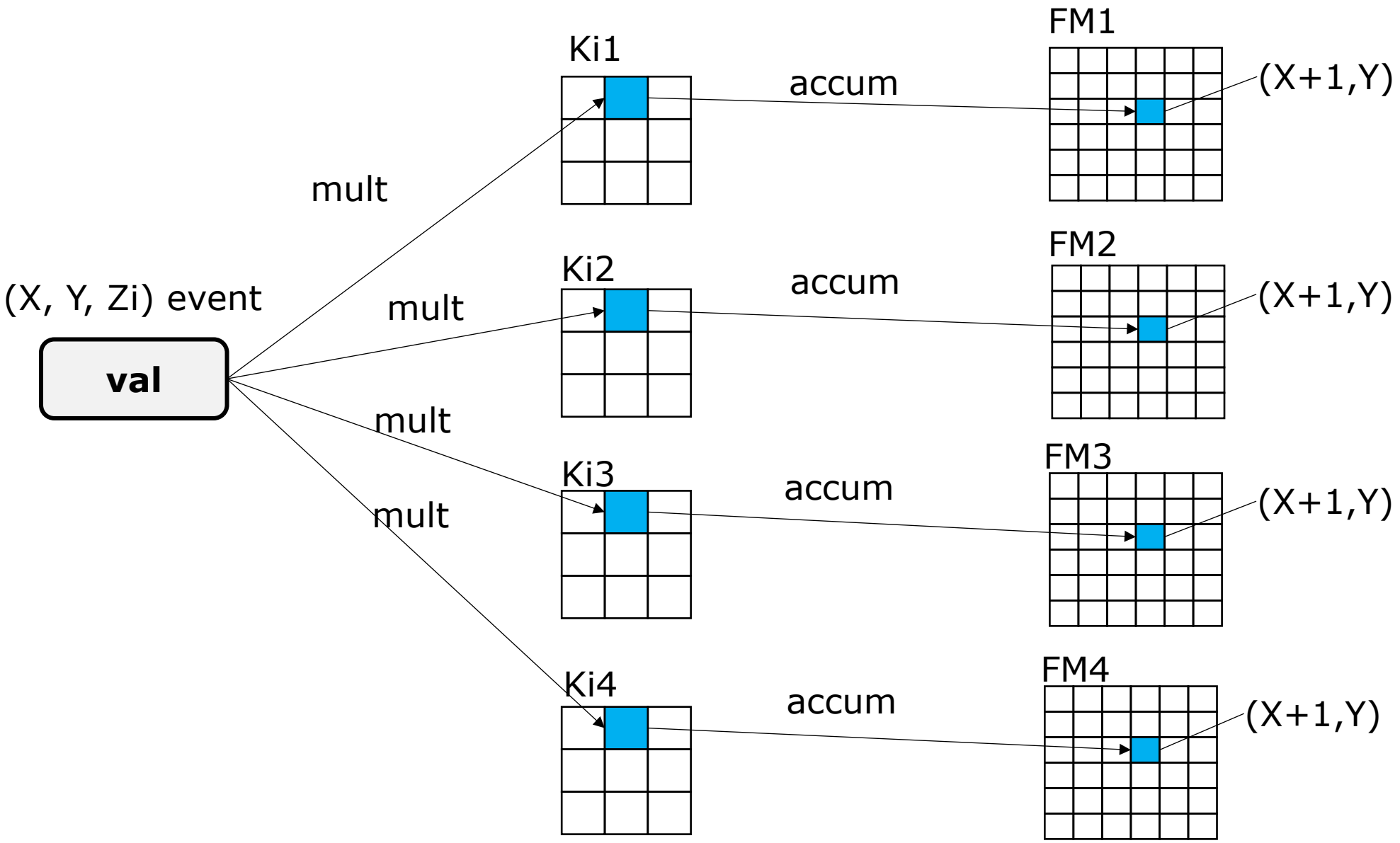
Input-stationary and SIMD



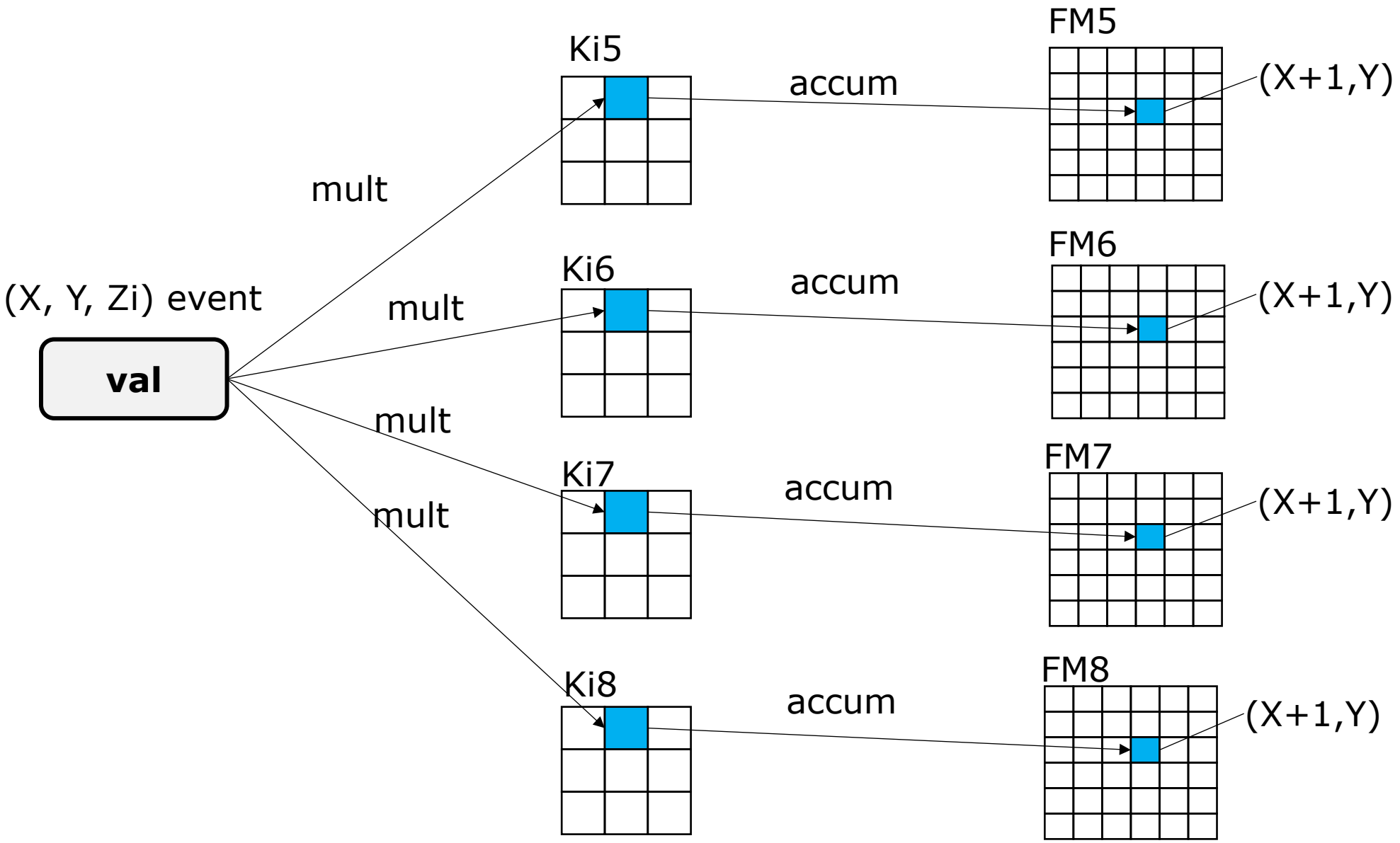
Input-stationary and SIMD



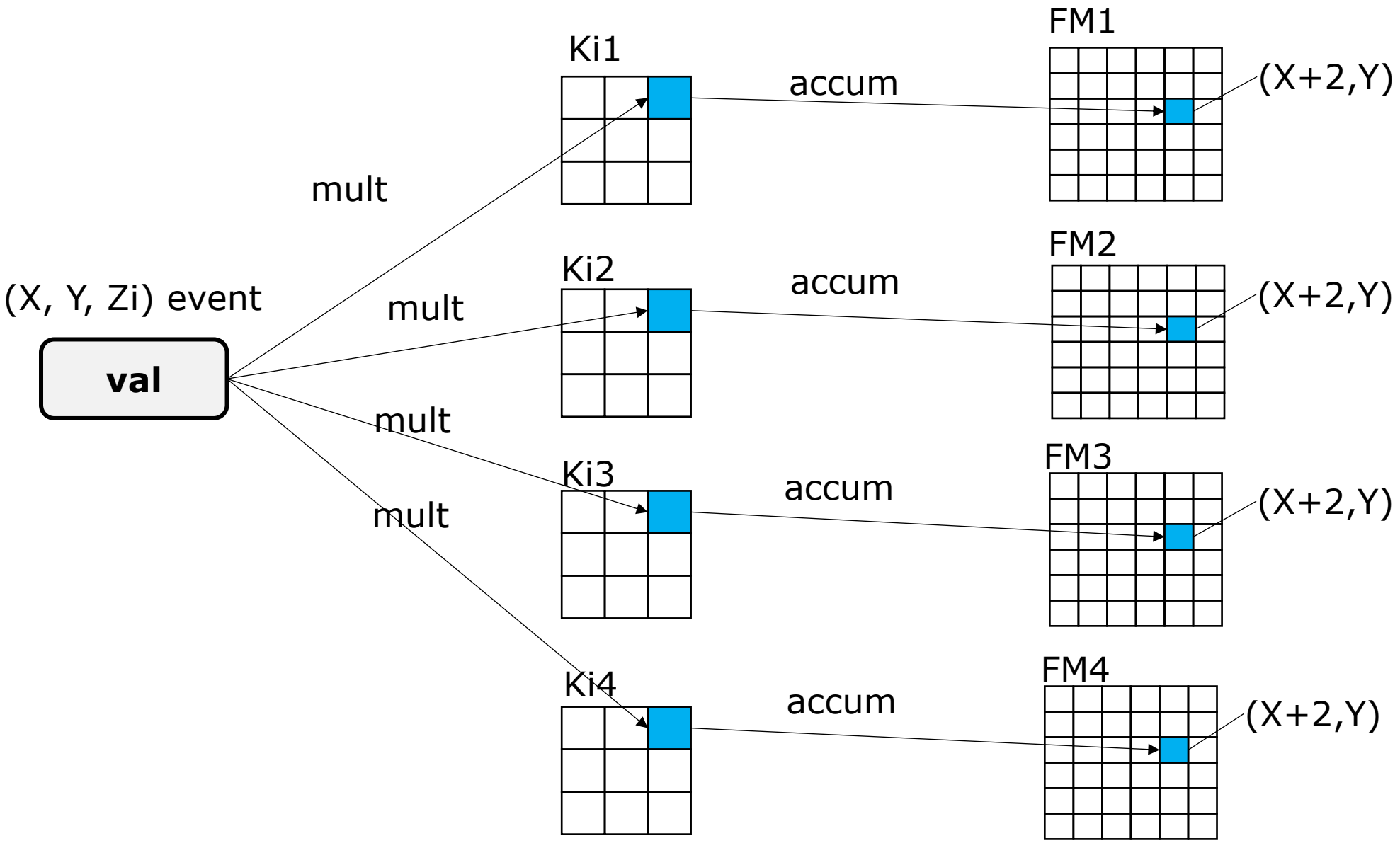
Input-stationary and SIMD



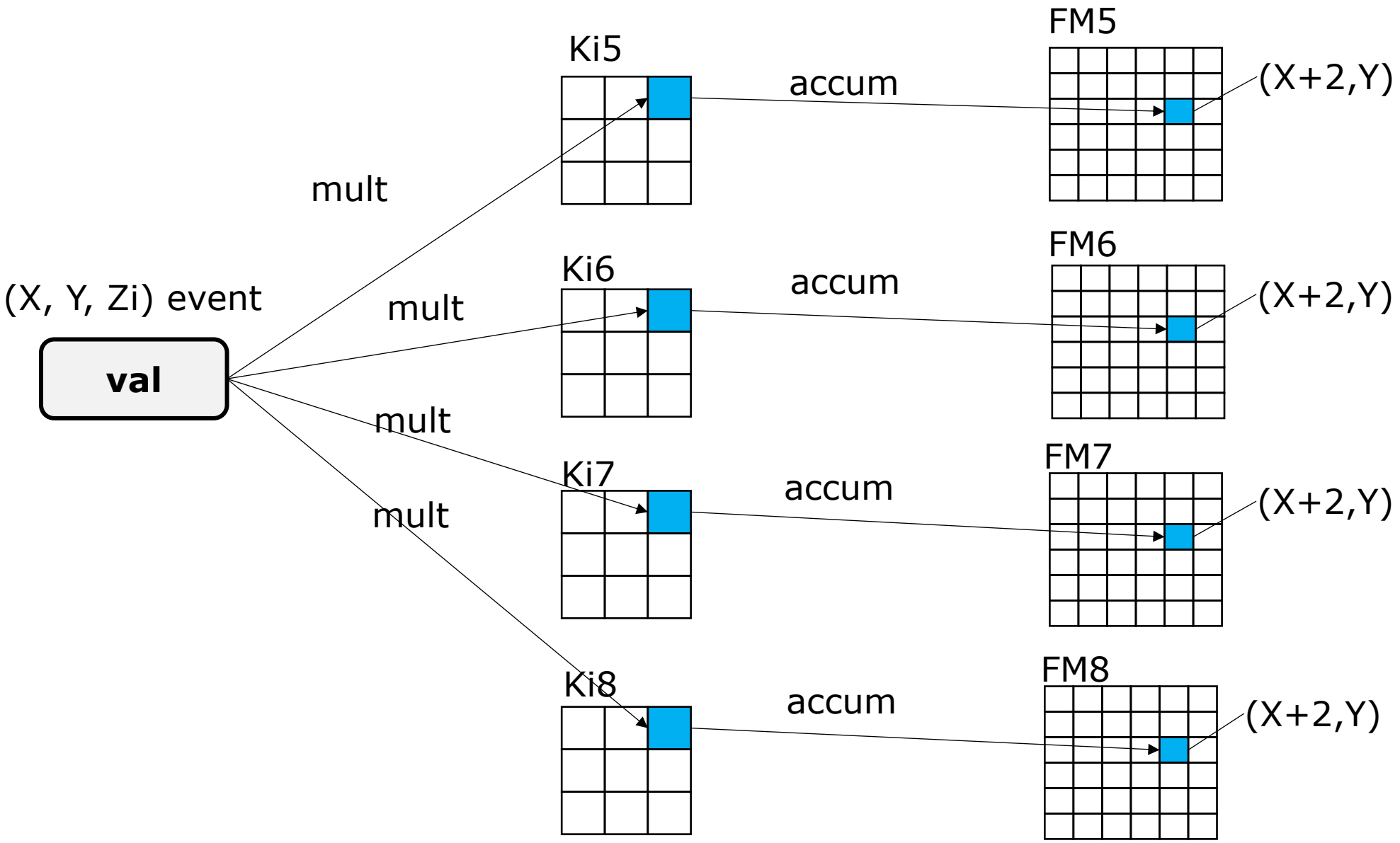
Input-stationary and SIMD



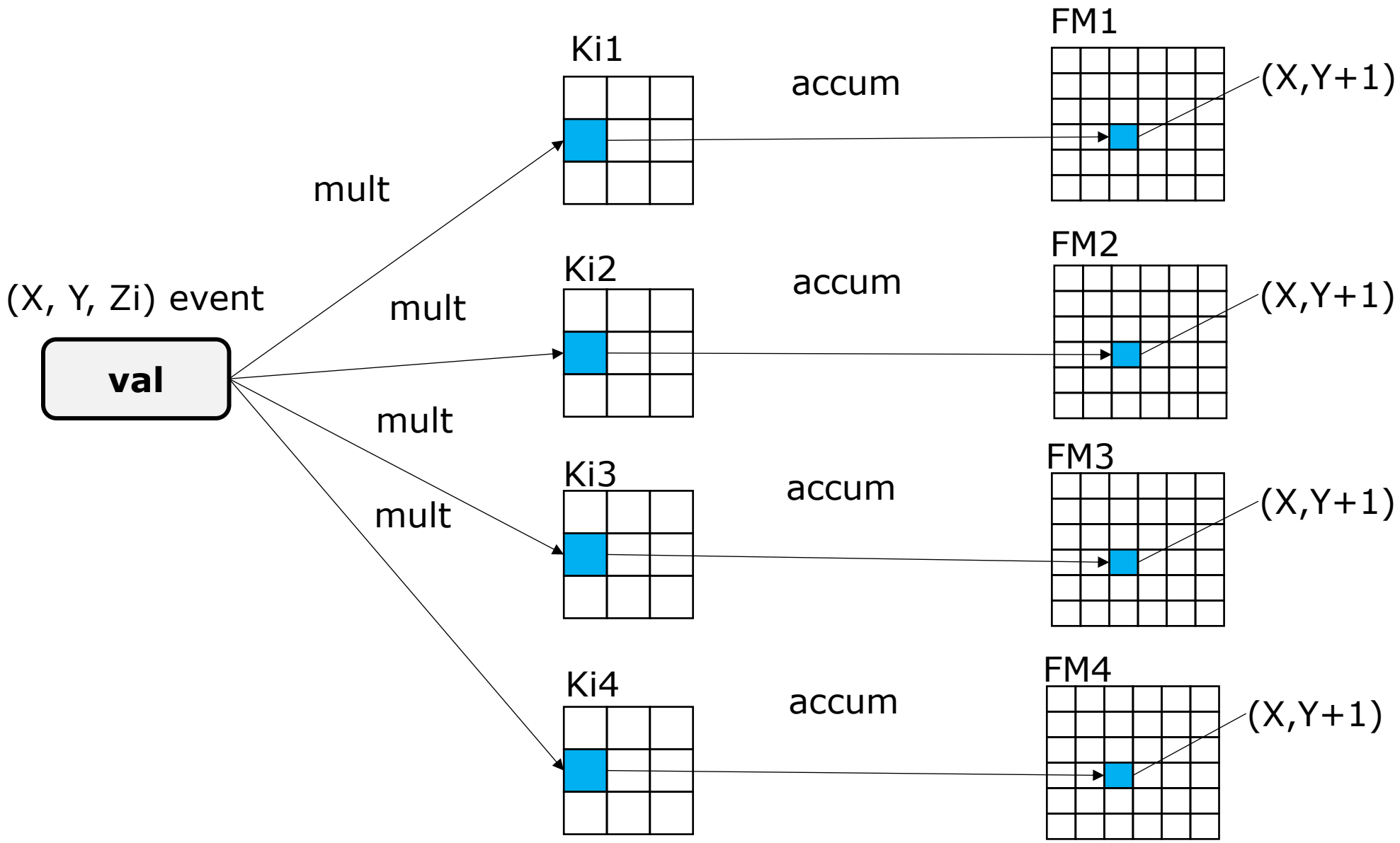
Input-stationary and SIMD



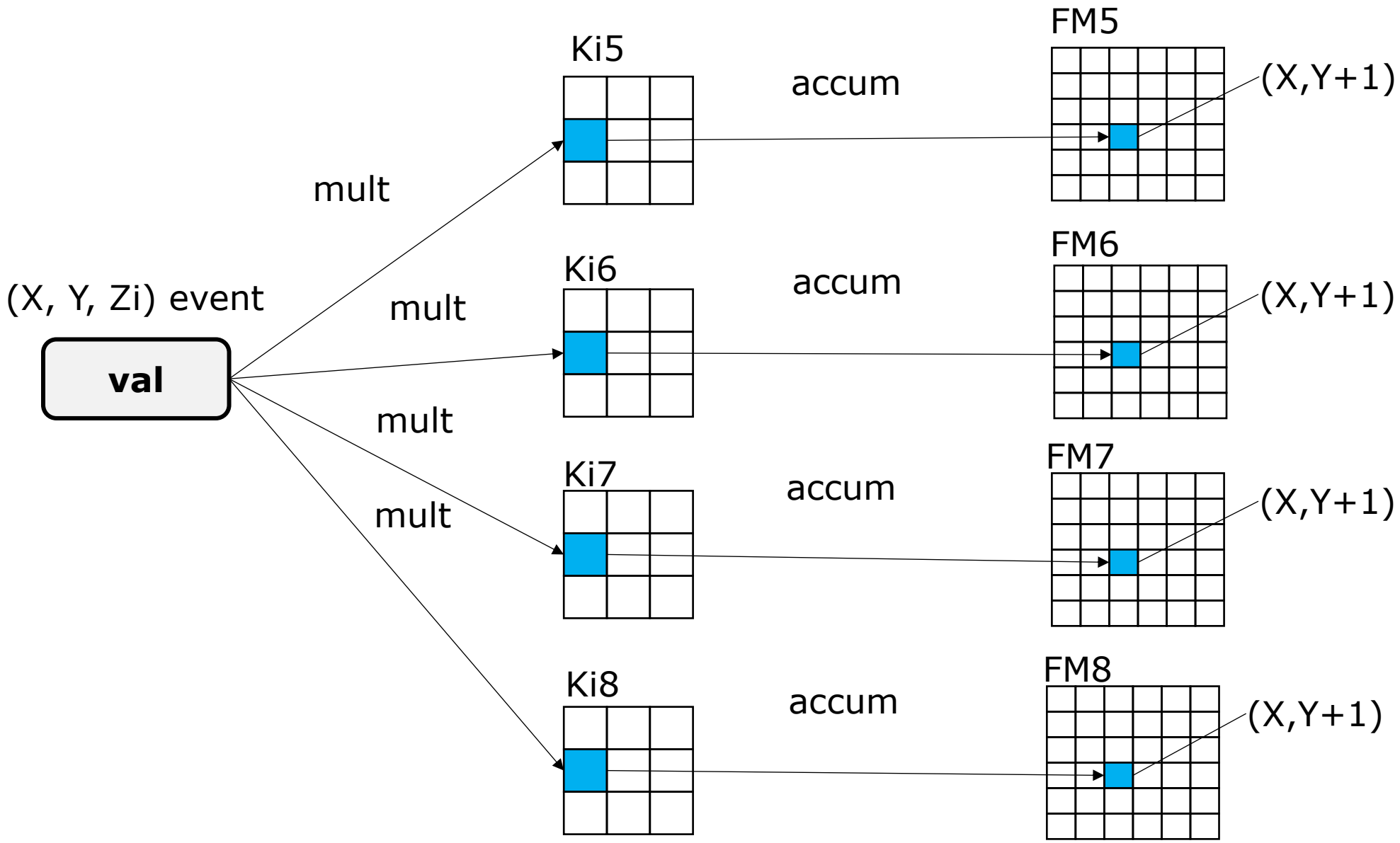
Input-stationary and SIMD



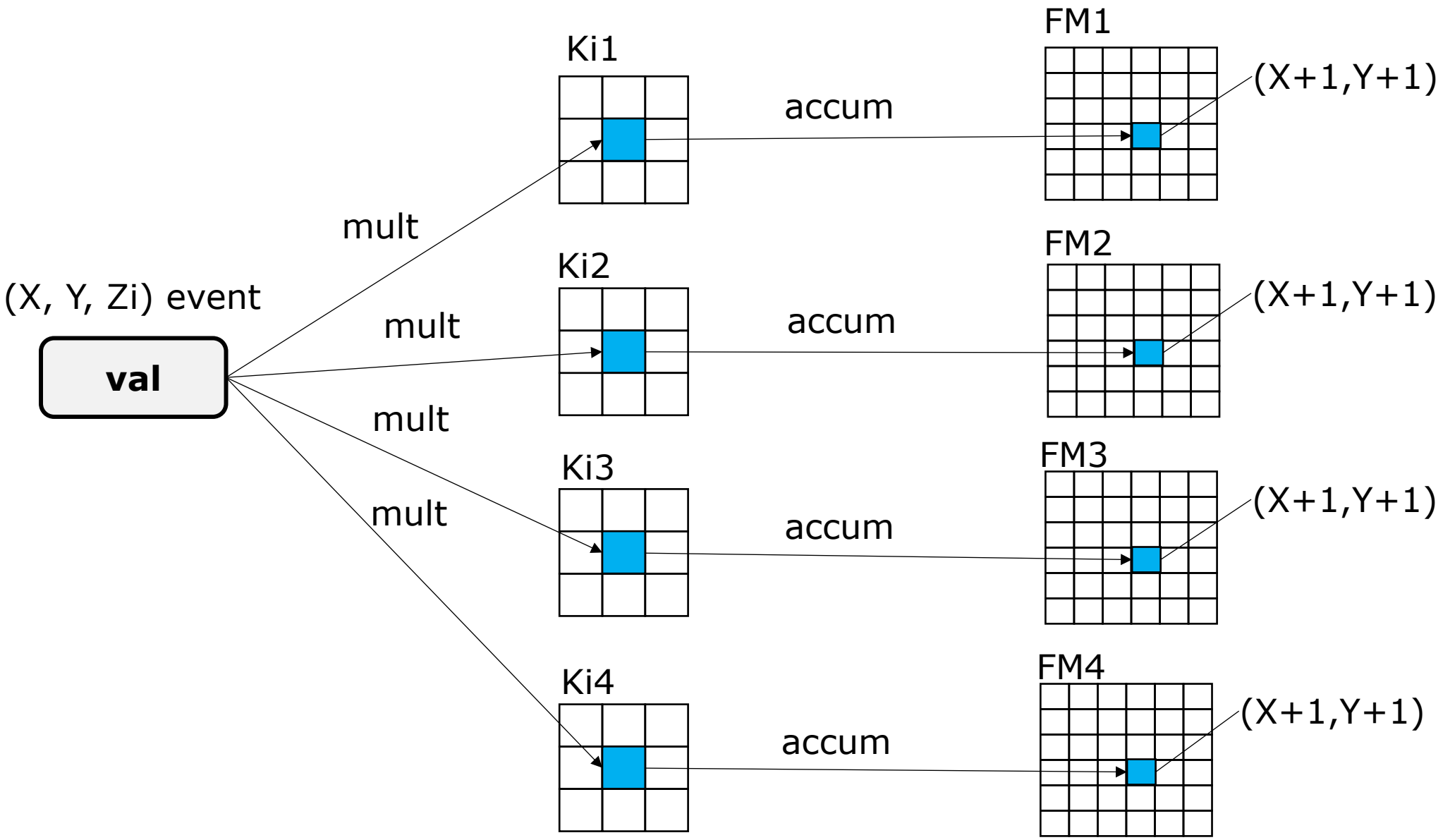
Input-stationary and SIMD



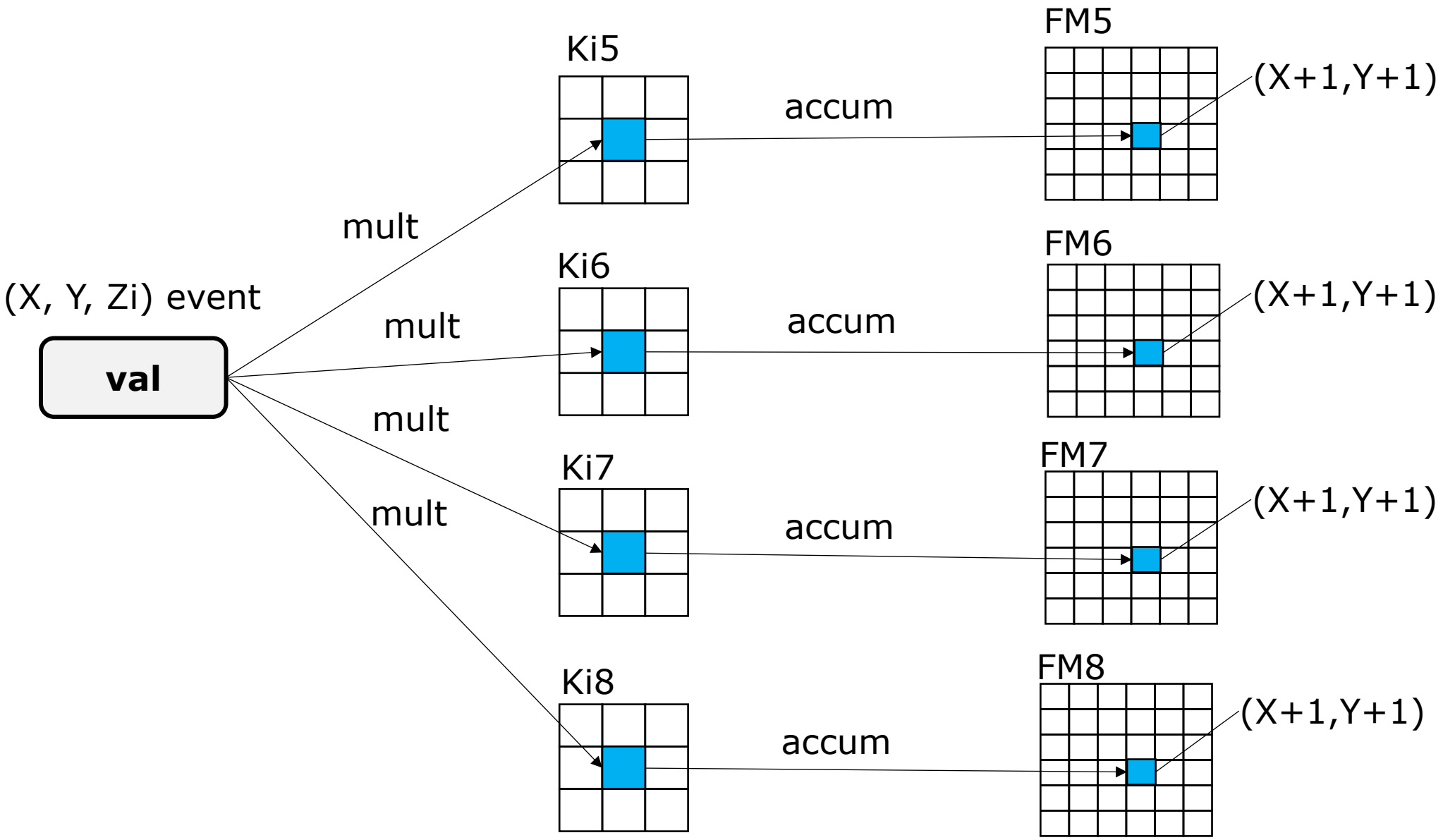
Input-stationary and SIMD



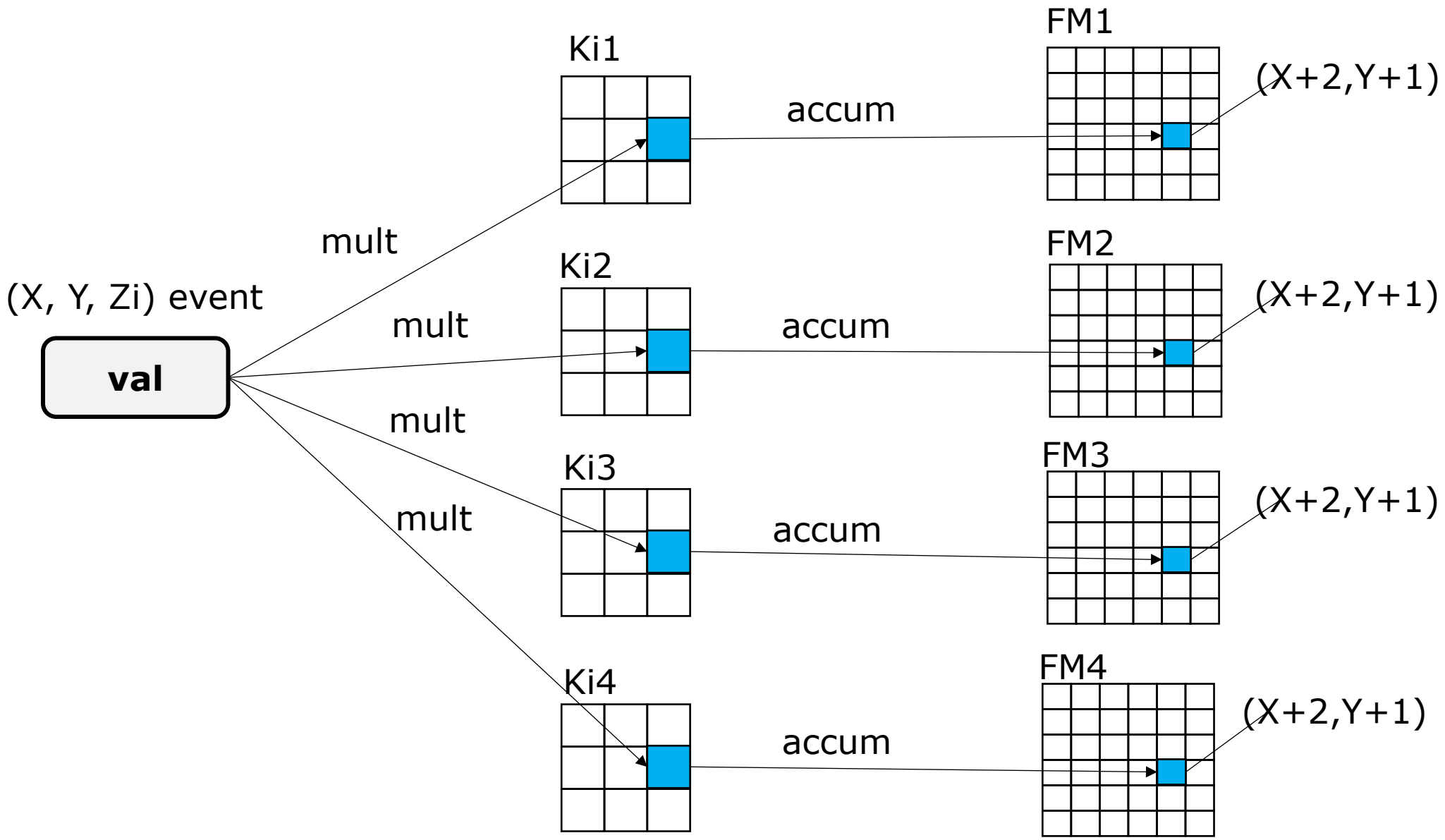
Input-stationary and SIMD



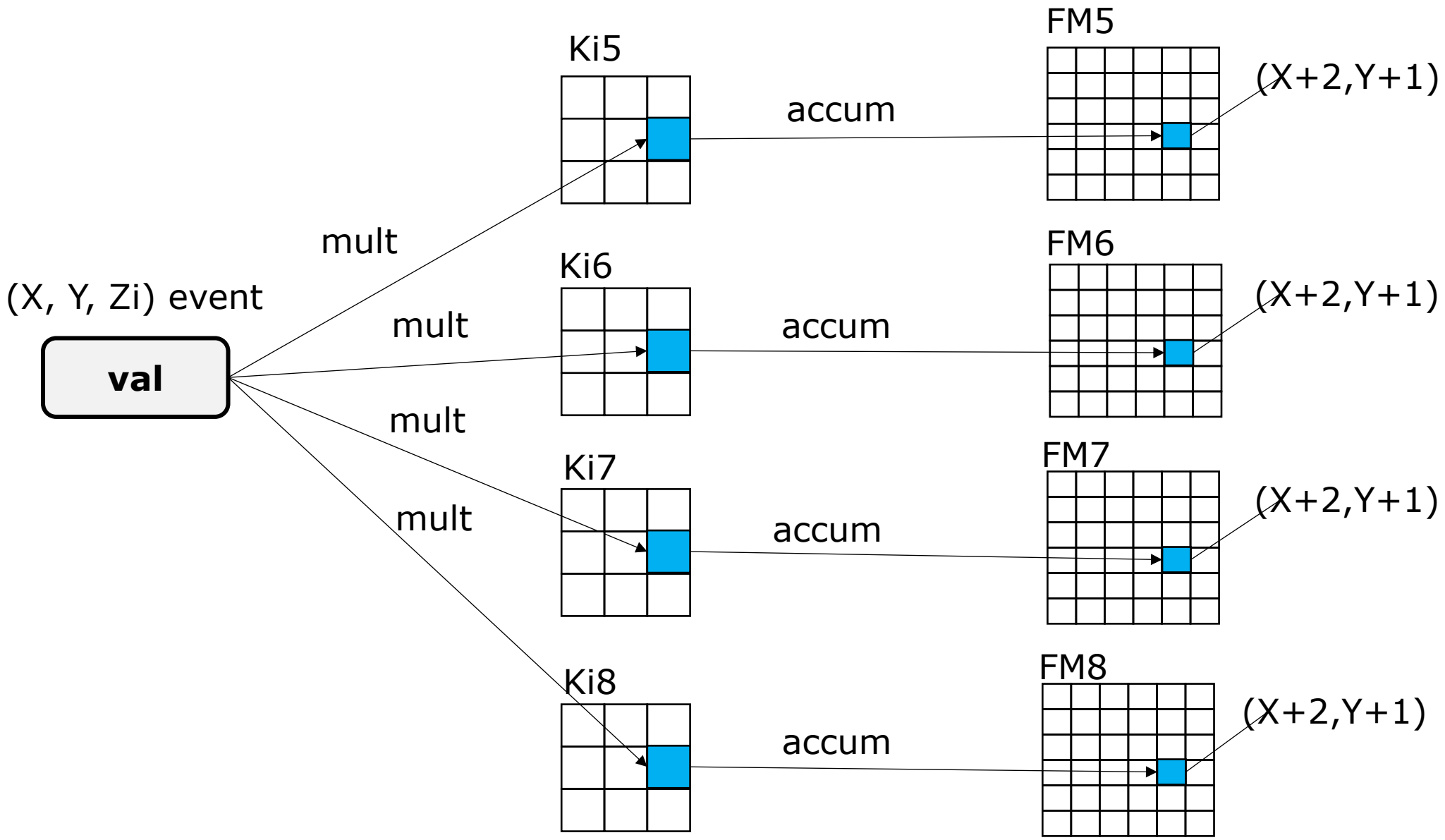
Input-stationary and SIMD



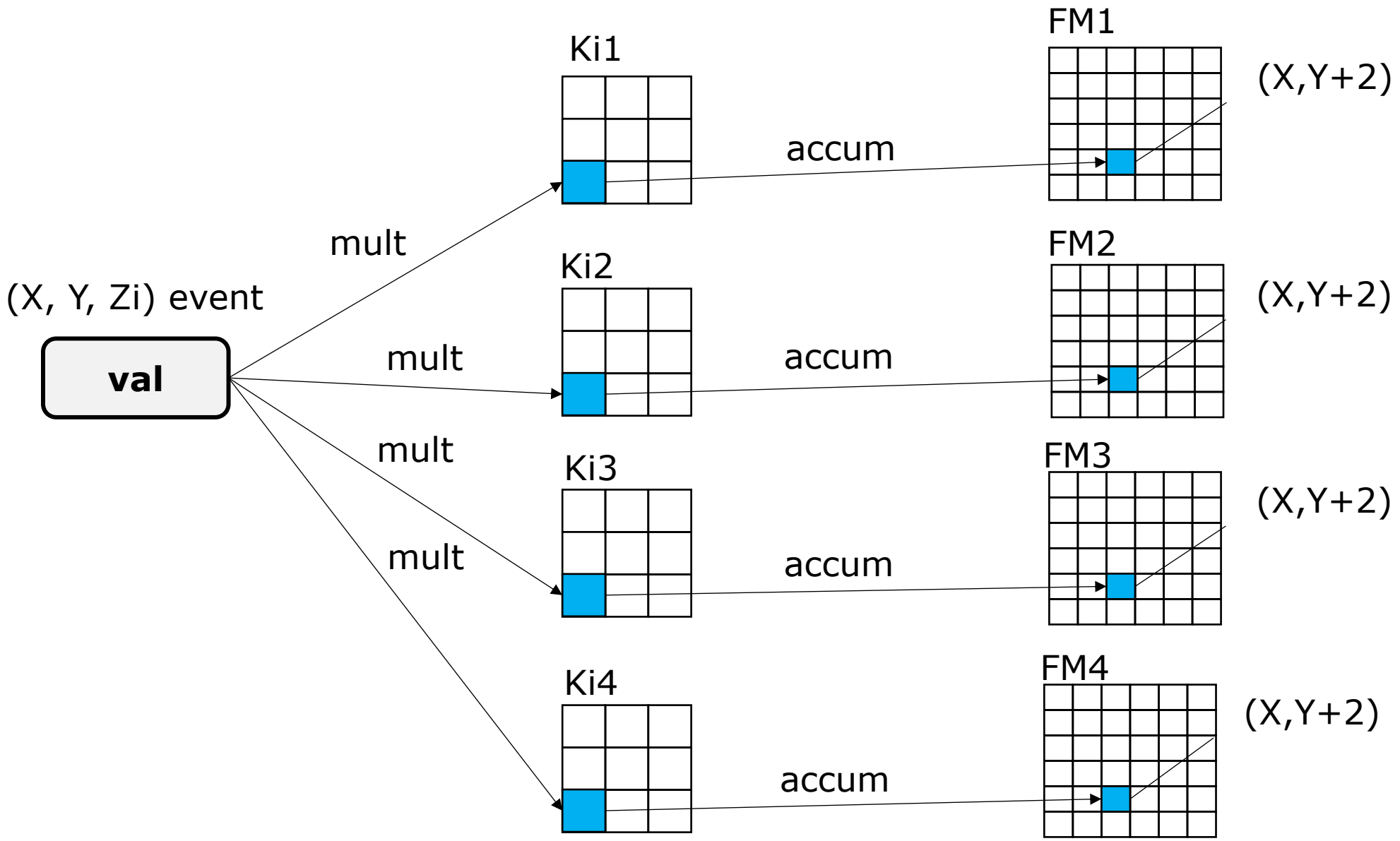
Input-stationary and SIMD



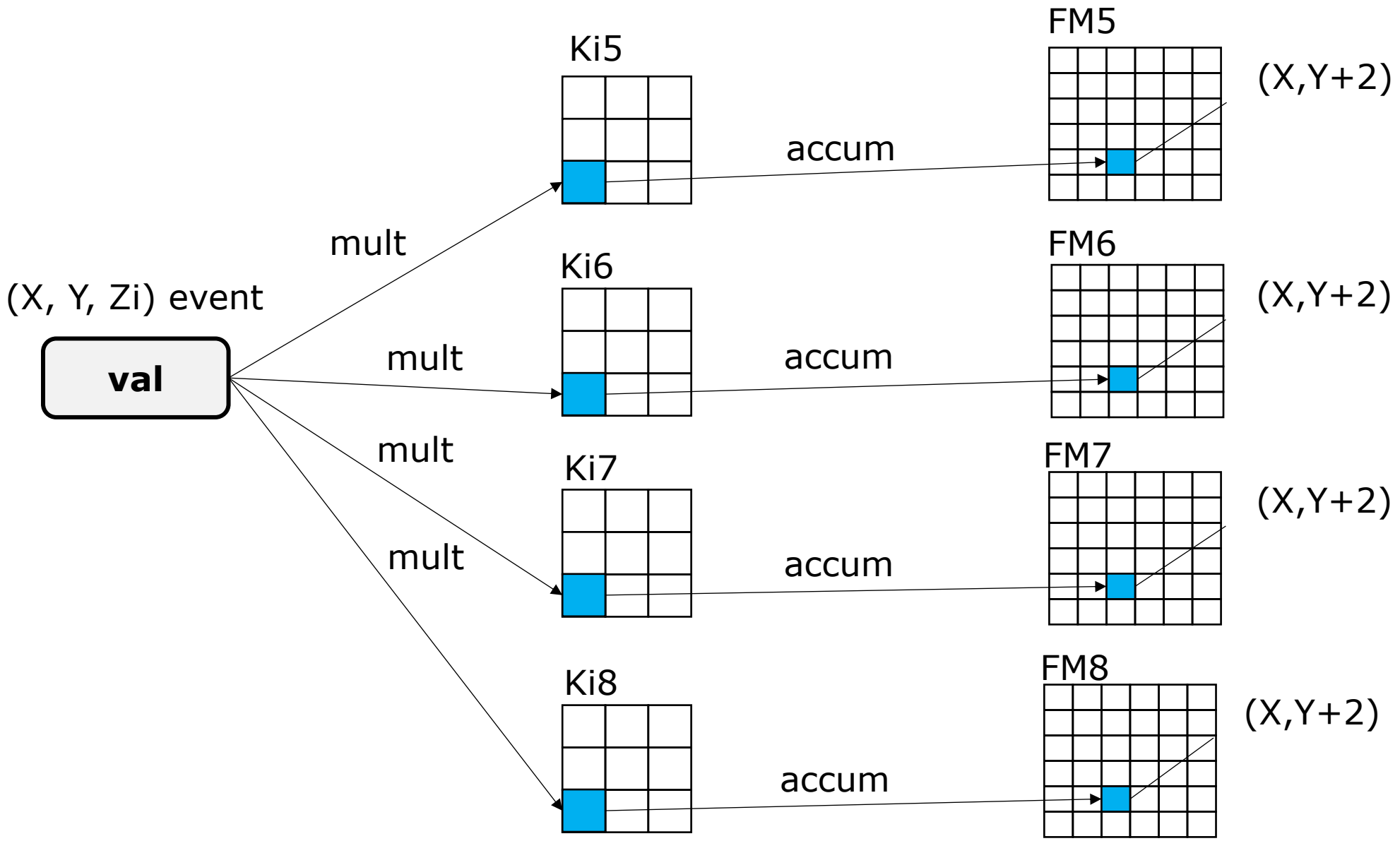
Input-stationary and SIMD



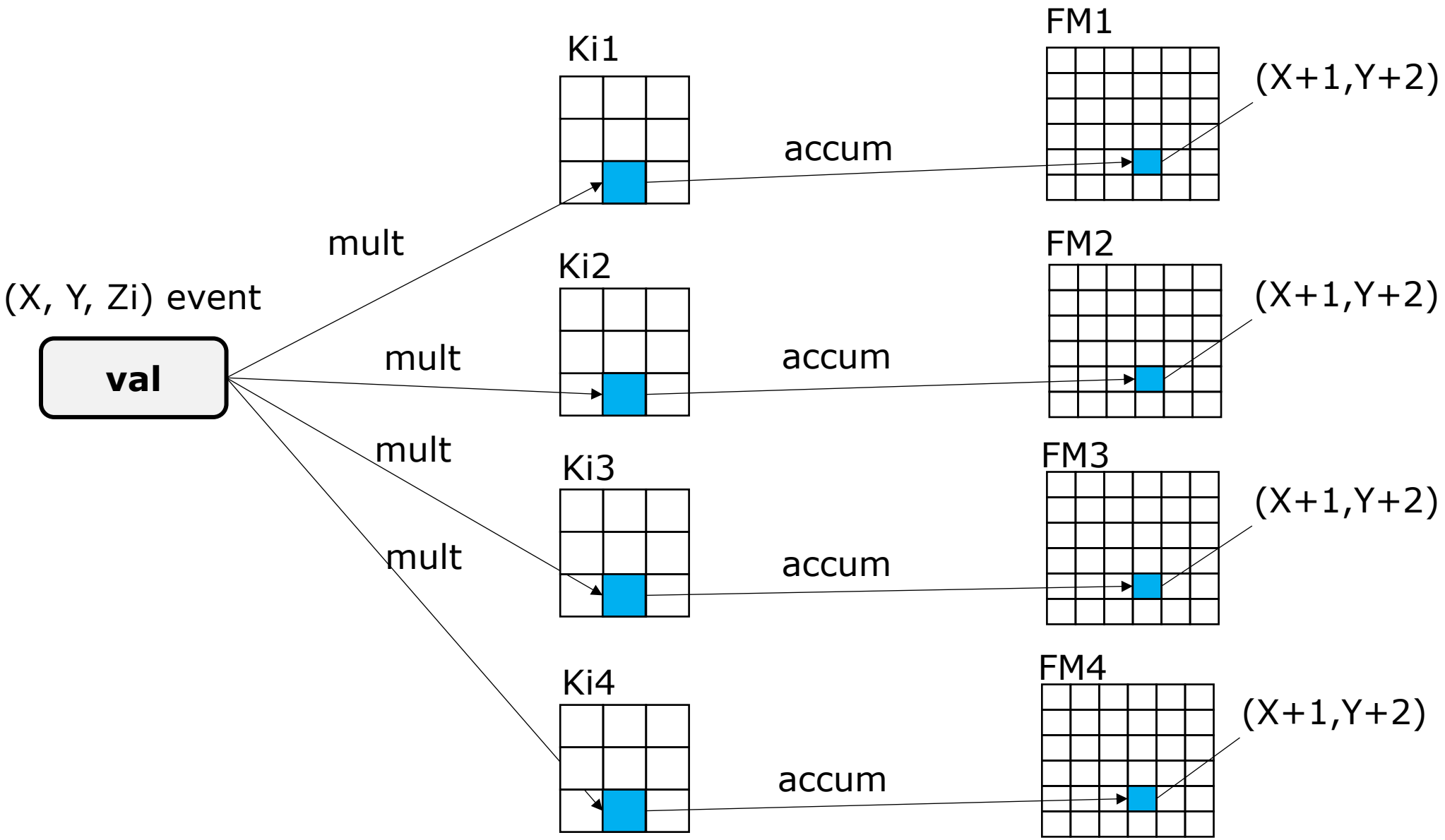
Input-stationary and SIMD



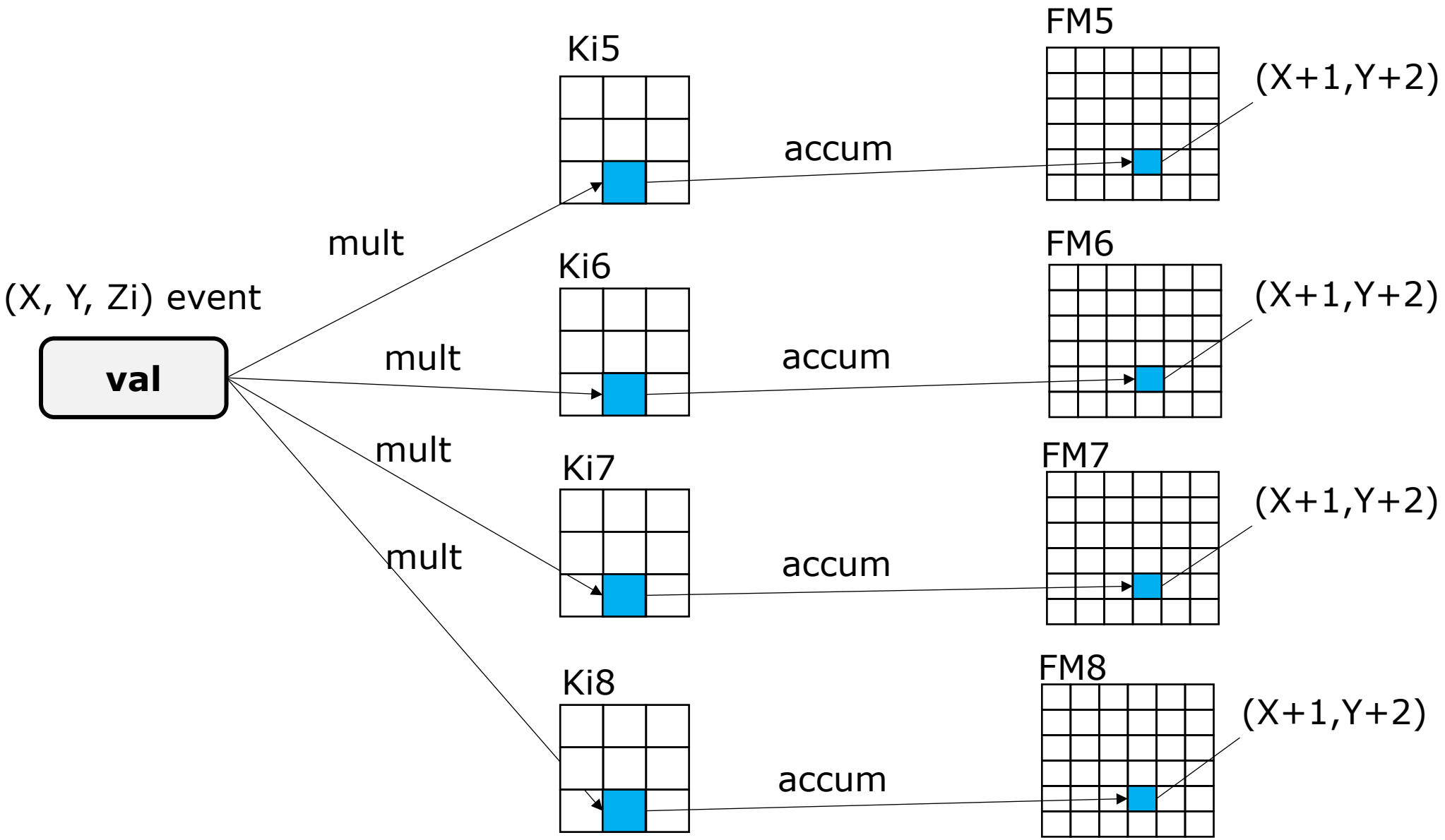
Input-stationary and SIMD



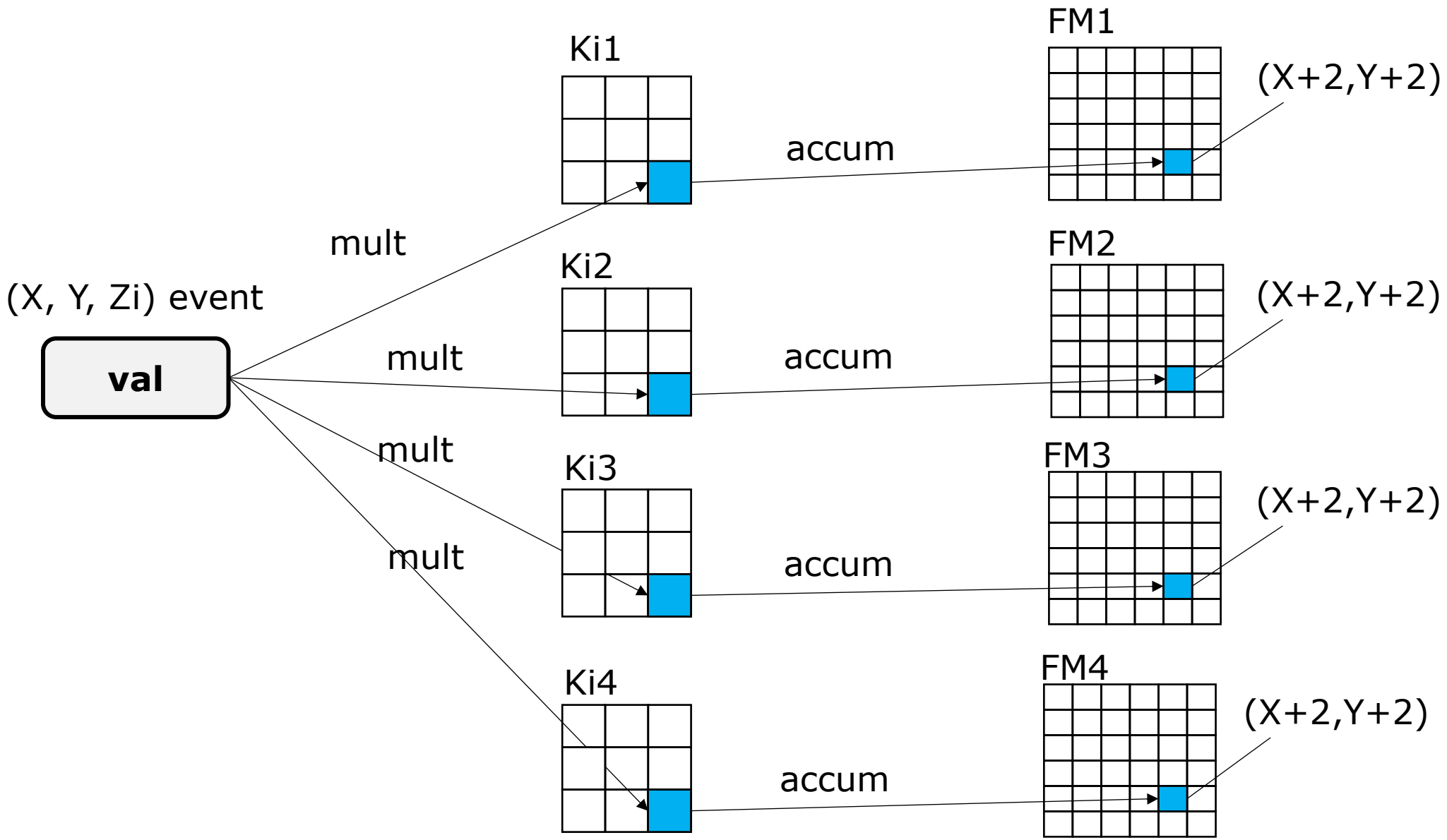
Input-stationary and SIMD



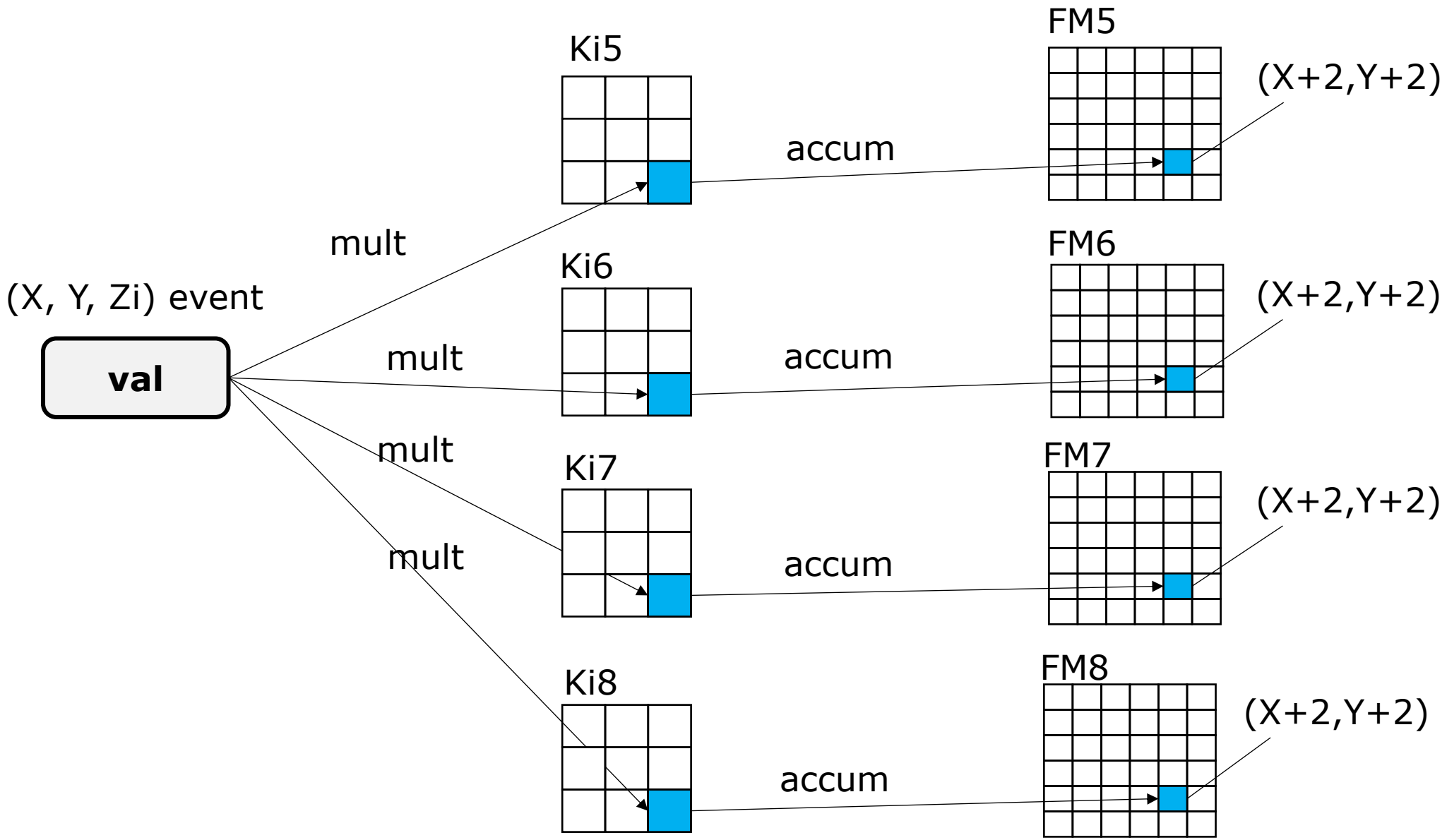
Input-stationary and SIMD



Input-stationary and SIMD



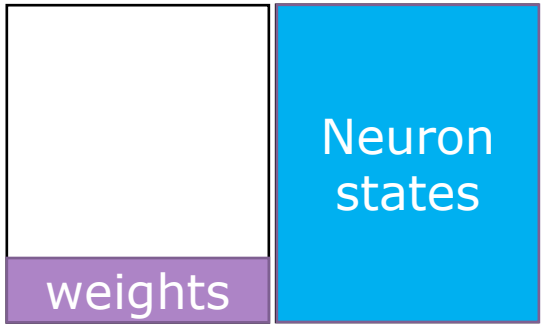
Input-stationary and SIMD



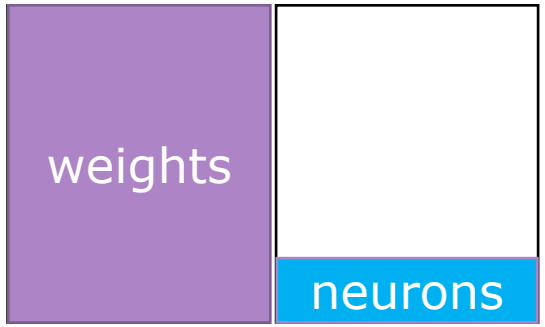
Memory Fragmentation

- With dedicated memories
  - neuron/weight ratio varies **significantly** per layer;
  - memory lost to fragmentation.

Core allocated to 1st layer



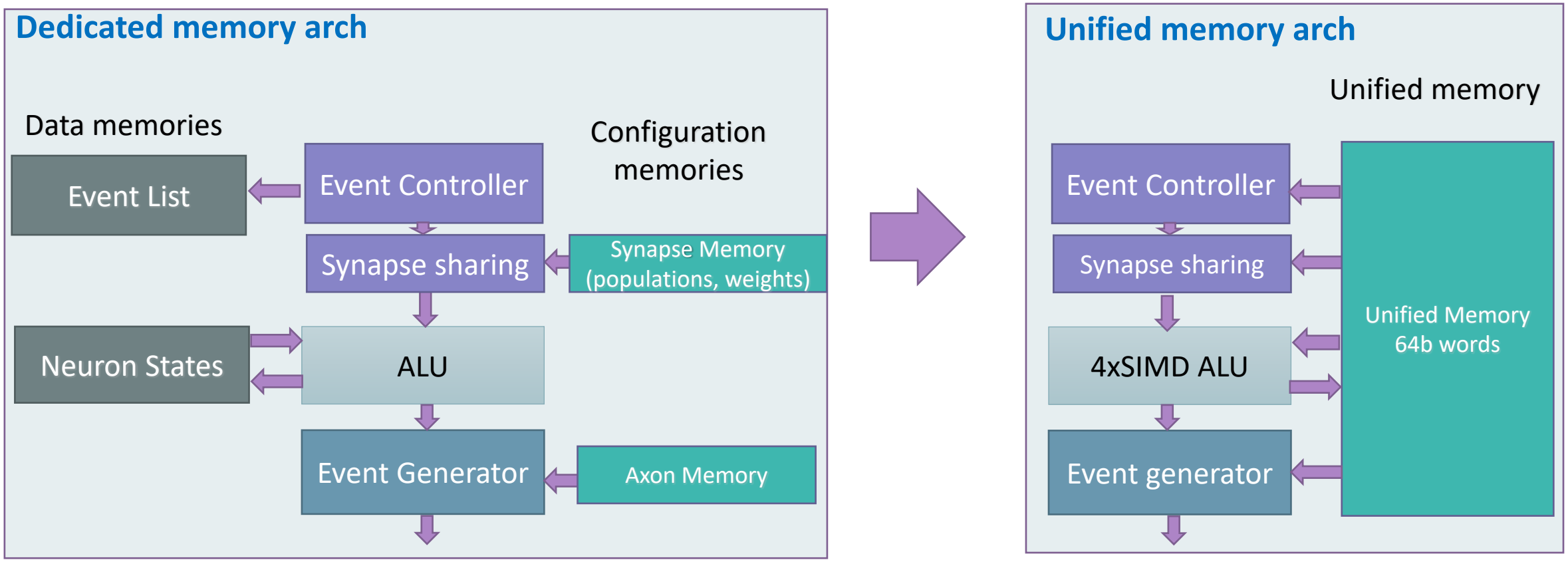
Core allocated to Last layer



RESNET18

	channels	neurons	weights	ratio weight/neuron
input		150528		
conv1	3	802816	9408	0,012
conv2_1	64	200704	36864	0,184
conv2_2	64	200704	36864	0,184
conv2_3	64	200704	36864	0,184
conv2_4	64	200704	36864	0,184
conv3_1	64	100352	73728	0,735
conv3_2	128	100352	147456	1,469
conv3_3	128	100352	147456	1,469
conv3_4	128	100352	147456	1,469
conv4_1	128	50176	294912	5,878
conv4_2	256	50176	589824	11,755
conv4_3	256	50176	589824	11,755
conv4_4	256	50176	589824	11,755
conv5_1	256	25088	1179648	47,020
conv5_2	512	25088	2359296	94,041
conv5_3	512	25088	2359296	94,041
conv5_4	512	25088	2359296	94,041
fc	512	1000	512000	512
Total		2.459.624	11.506.880	4,678308554

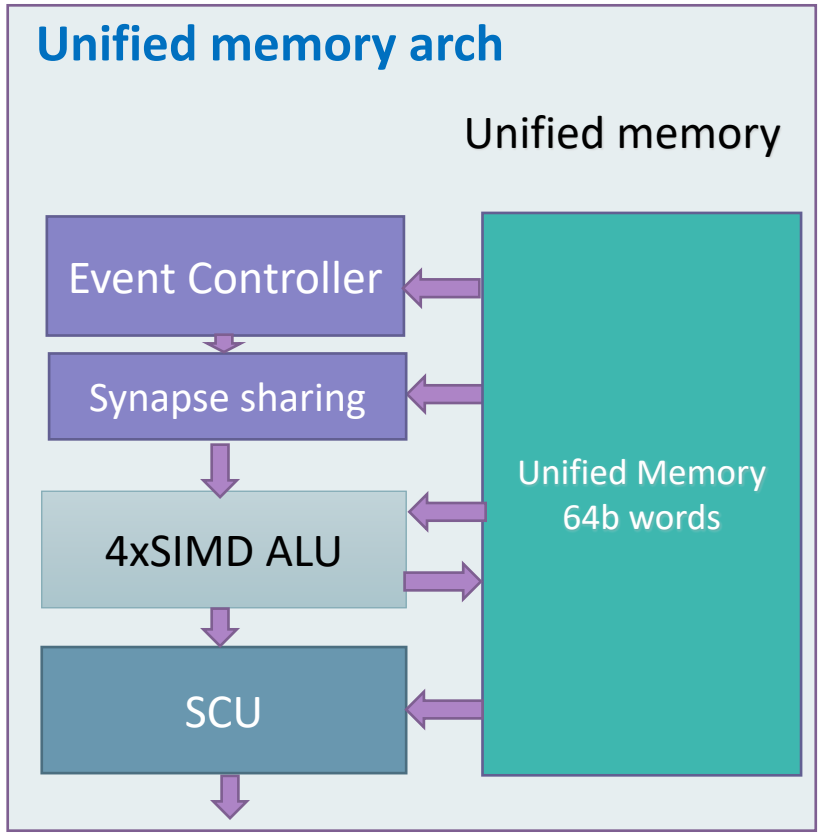
Unified Memory: avoiding fragmentation



events, neuron states, weight tables and axons all must fit in 64b words
…but uniform memory instances **simplify layout** (see layout slide later on).

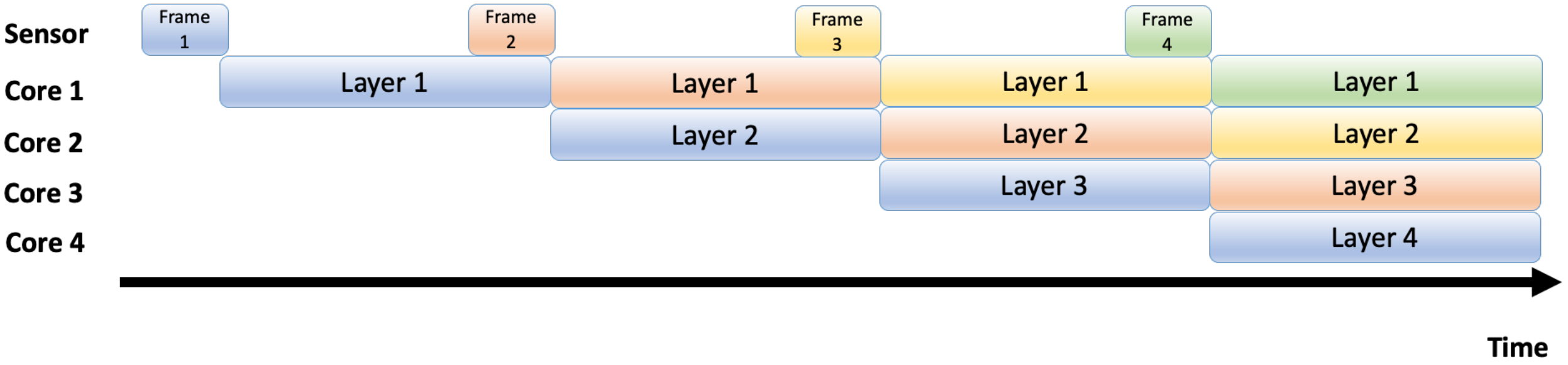
Memory view: unified SRAM

- Memory choice: maximize bw, minimizing size, consumption
  - 64 bit words: 4 FP16 neuron states per entry.
  - 4 memory banks means 4x64b accesses per cycle
  - 4x SIMD
- Requirement: one synapse per MAC per cycle
- Per cycle needs:
  - read 4 neuron states (64b)
  - read 4 kernel weights (64b max)
    - depends on quantization and pruning…
  - write 4 neuron states (write-back) (64b)
- Per input event:read one population
- Per output event: read one or more axons
- Pipeline constraint: read and write of neurons state must not access same bank in same cycle



Pipelined scheduling

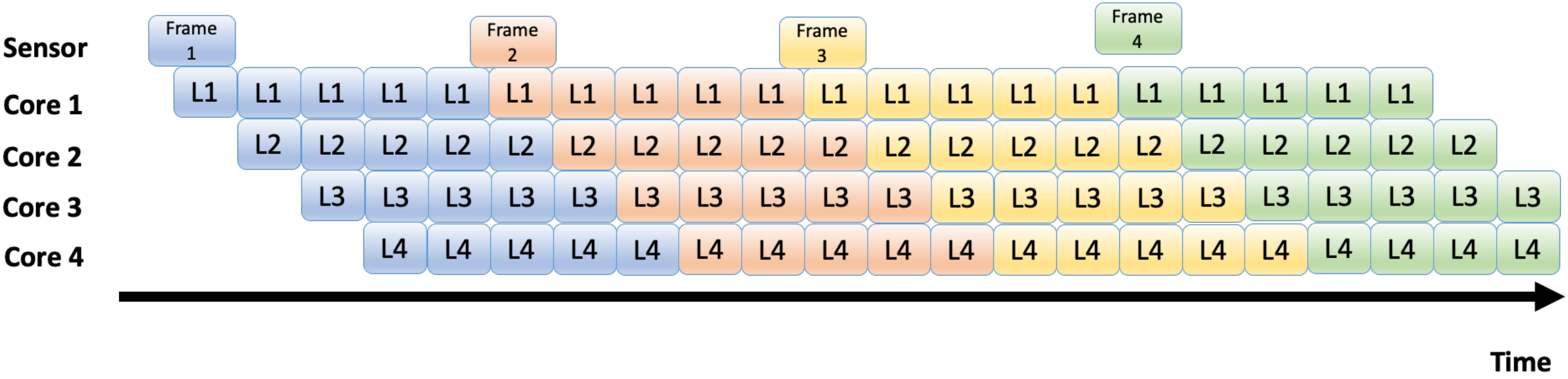
Pipelining: each layer is allocated statically to one core.
Requires buffers to store the output results of one full layer.



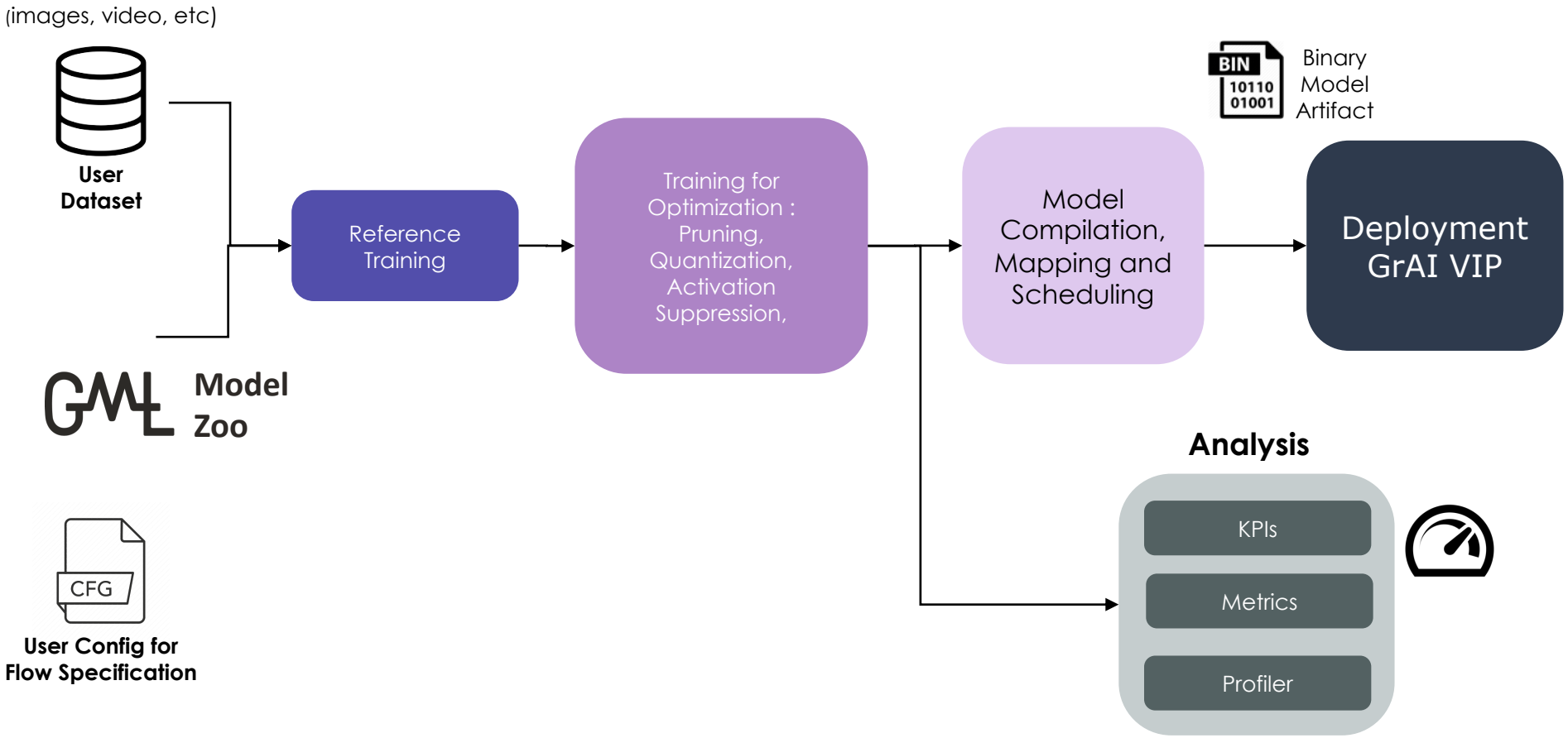
Note: this is very simplified. In practice we can execute several layers per core, and we split layers in several cores (more on that later). Mostly the layer to core allocation is constrained by the size of each layer in weights and the amount of memory per core.

Striping

Striping:
Execution of a layer starts when enough data is ready to start partial evaluation (eg, line by line).
Much smaller output buffers, lower latency.



Programming Tool Flow



Mapping units: Cuts

- **Neuronflow is a self-contained architecture**
  - no external DRAM access
    - No DRAM power consumption, latency…
  - **But the size of the model is very important**
  - all parameters and buffers must reside in device memory.
  - Static layer (cut) to core assignment
  - all weights and input/output buffers must be stored in the memory of the cores

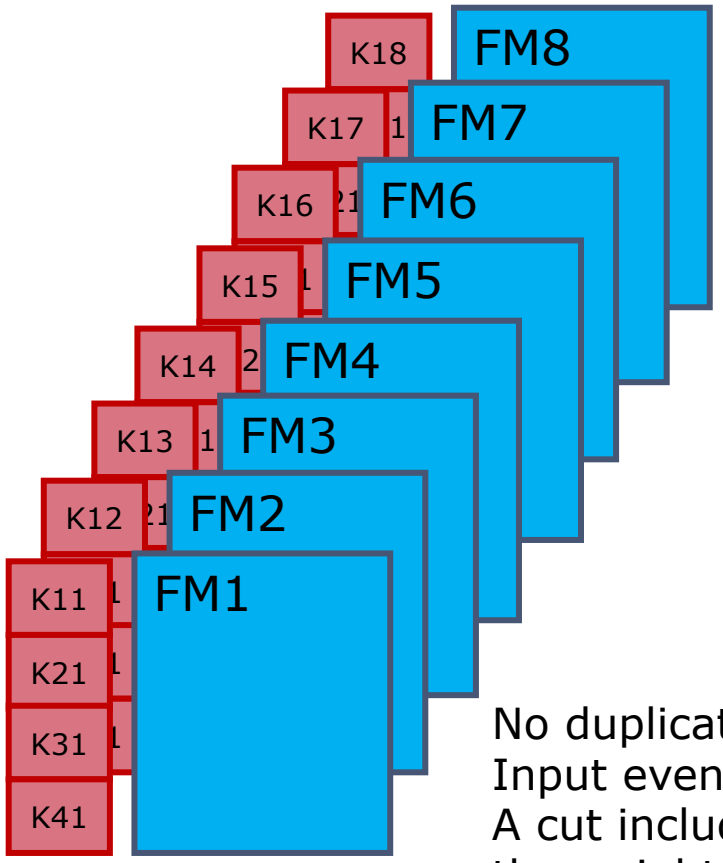
Layers are **cut to fit** in core memory.

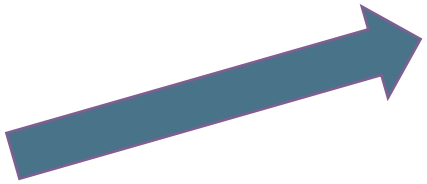
Cuts are the units of mapping.

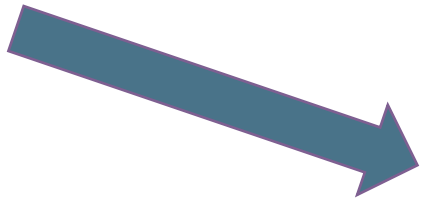
Cuts can be done in X, Y or Z, but Z is better in terms of resource usage.

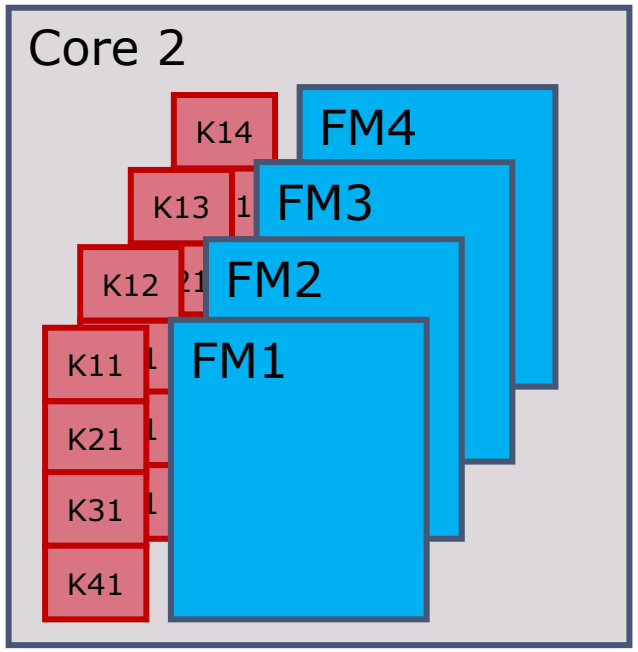
Cuts can be used to increase performance by adding more cores to the processing of one layer.

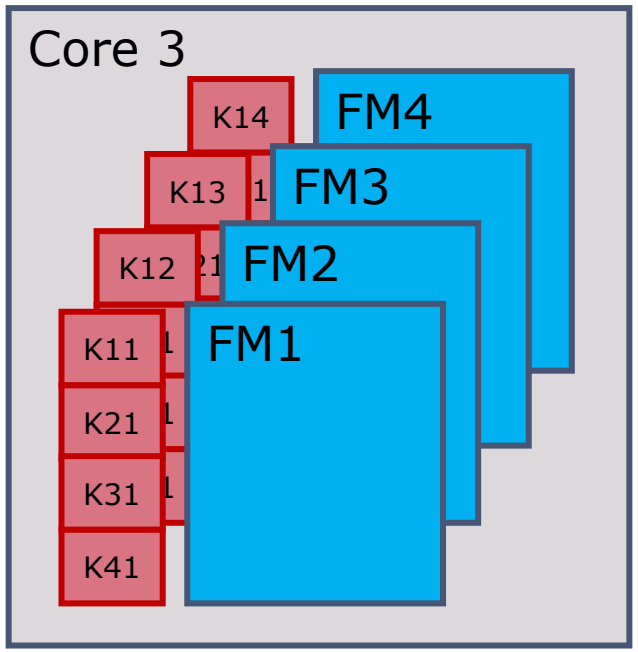
Simple Cut in Z







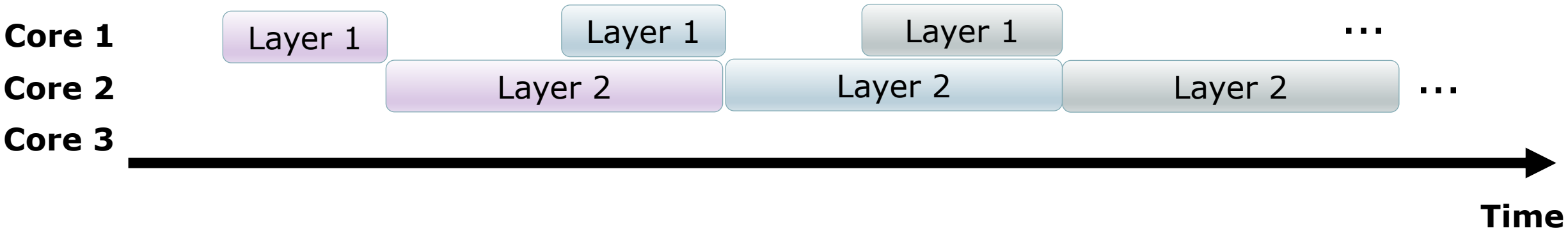




No duplicated weights in memory.
Input event executes in parallel.
A cut includes a number of FMs and all the weights for each FM it contains.
In the example we use an 8 channel layer that receives input from 4 channels in a 4x8 conv.

Pipelined schedule: cutting for performance

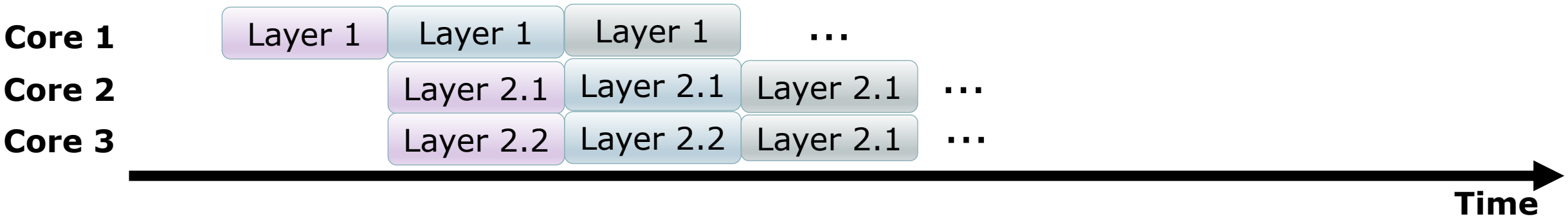
Simple model: Slowest stage of the pipeline limits the throughput – critical stage of the pipe



Execution time of Layer 1 = t
Execution time of Layer 2 = 2t
Latency per stripe = t+2t = 3t
Throughput = min (1/2, 1/2t) = 1/2t

Pipelined schedule: cutting for performance

Throughput can be increased by allocating more cores to a bottleneck layer.
Feature map must be cut.



Execution time of Layer 1 = t
Execution time of Layer 2 = 2t
Cut layer 2 by 2 cores
Latency per stripe = t+t = 2t
Throughput = min (1/t, 1/t, 1/t) = 1/t

How training affects mapping/performance

- Weight quantization and prunning lower memory requirements per layer which frees cores to cut slow layers to improve latency and throughput.
- Prunning also reduces number of macs per inference.
- Activation suppression decreases the number of events to process, lowering load per core and power consumption
- ARTS combines prunning and activation suppression.

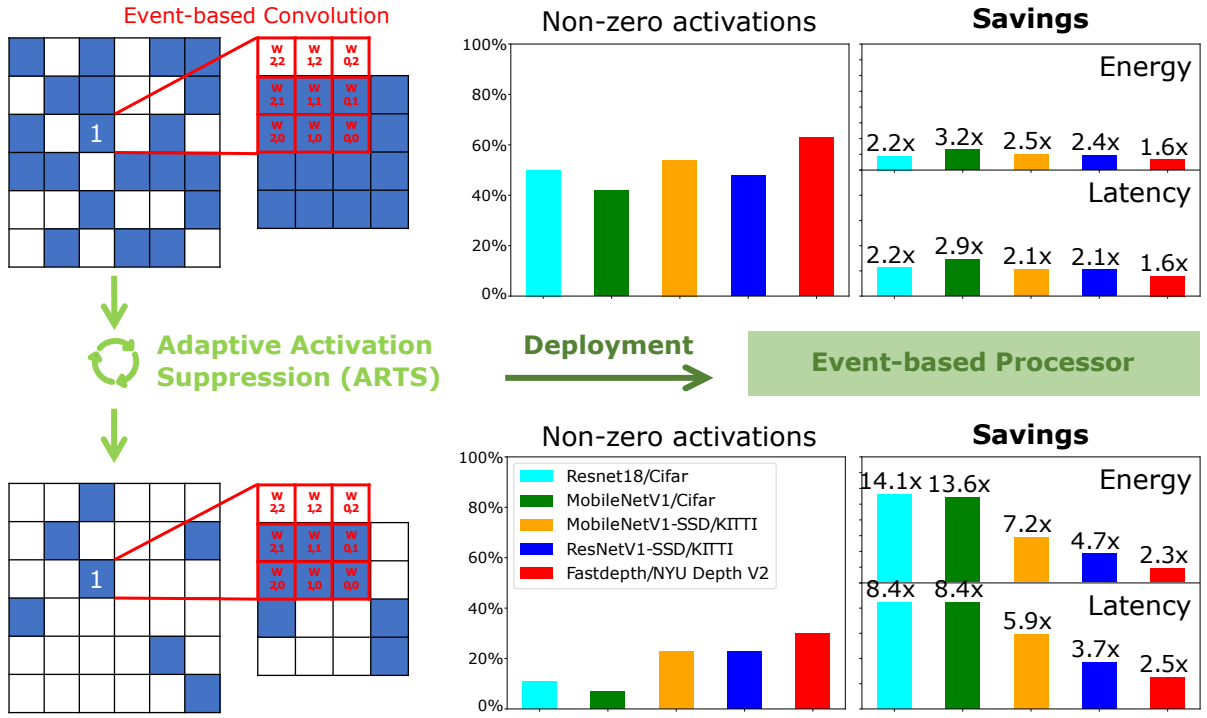


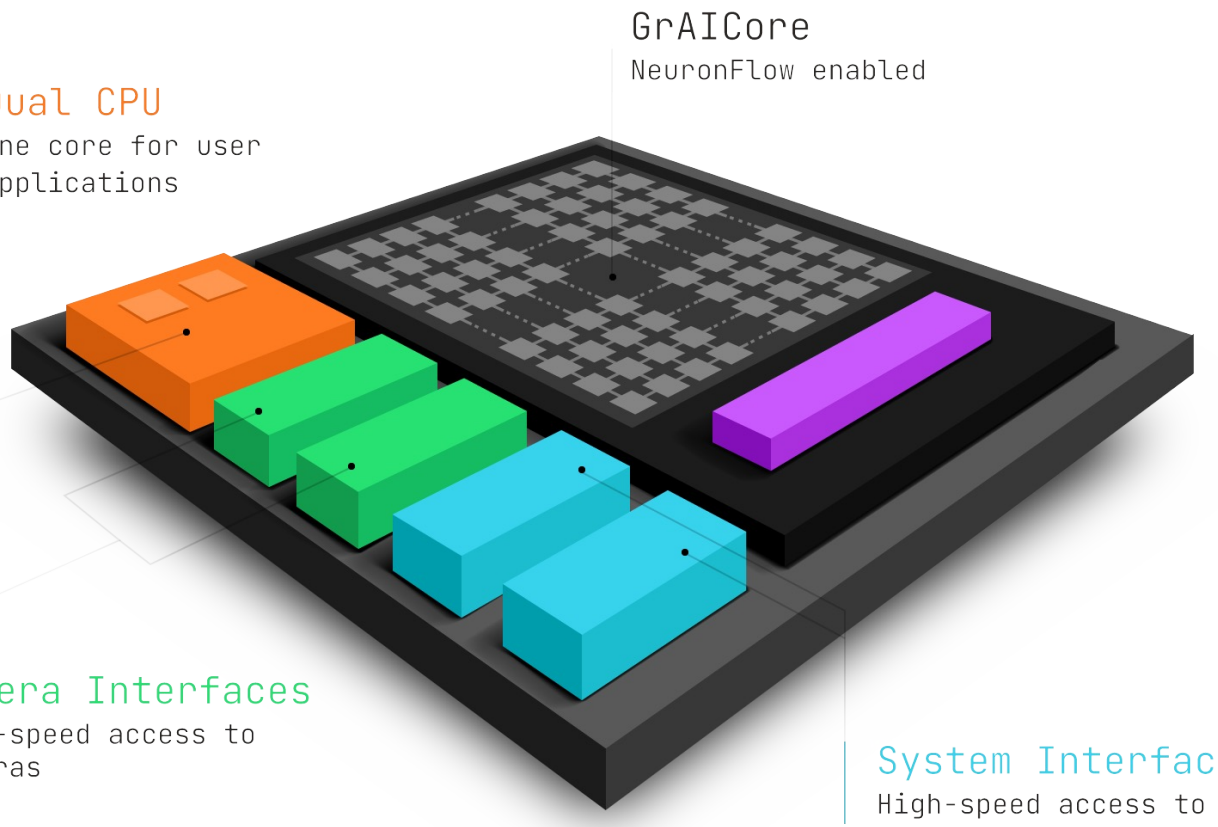
Figure 1. Our proposed method (ARTS) progressively enforces the amount of zeros in the activation maps, leveraged by event-based processors to achieve significant inference energy savings and latency reduction.

GrAI VIP
Vision Inference Processor

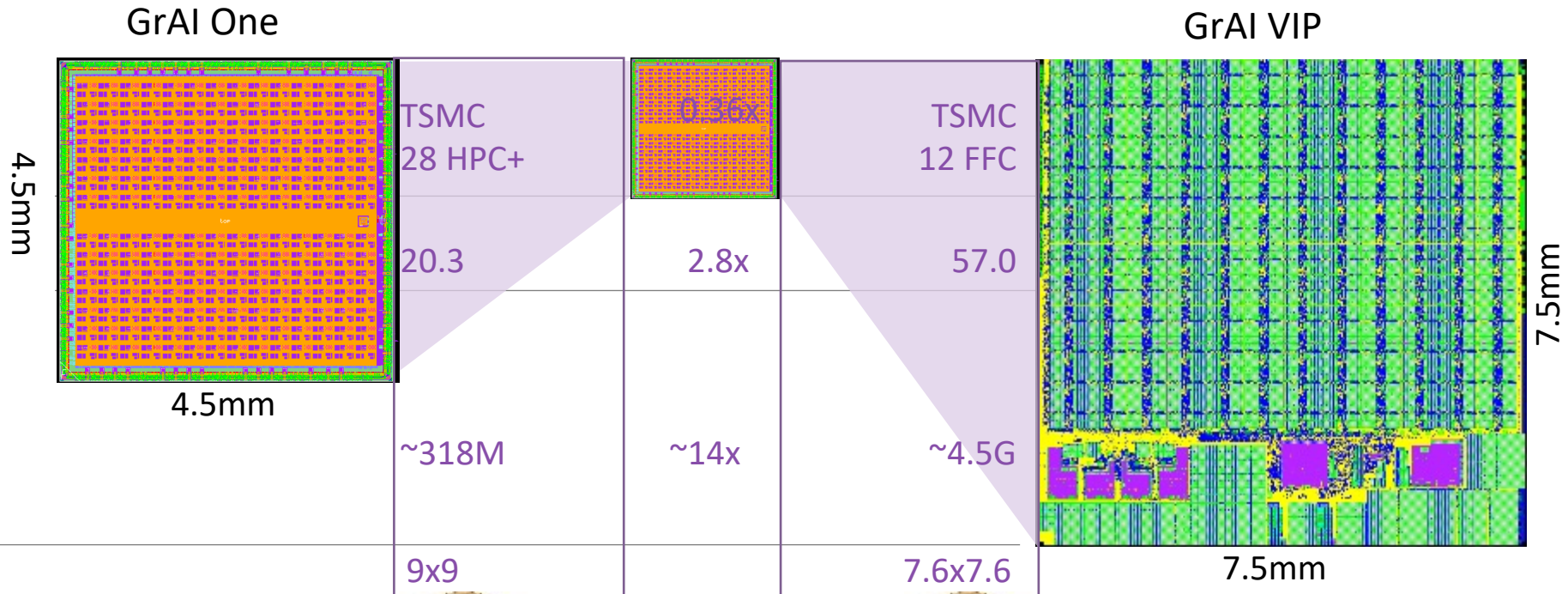
Life-Ready AI in silicon

GrAI VIP

Vision
Inference
Processor

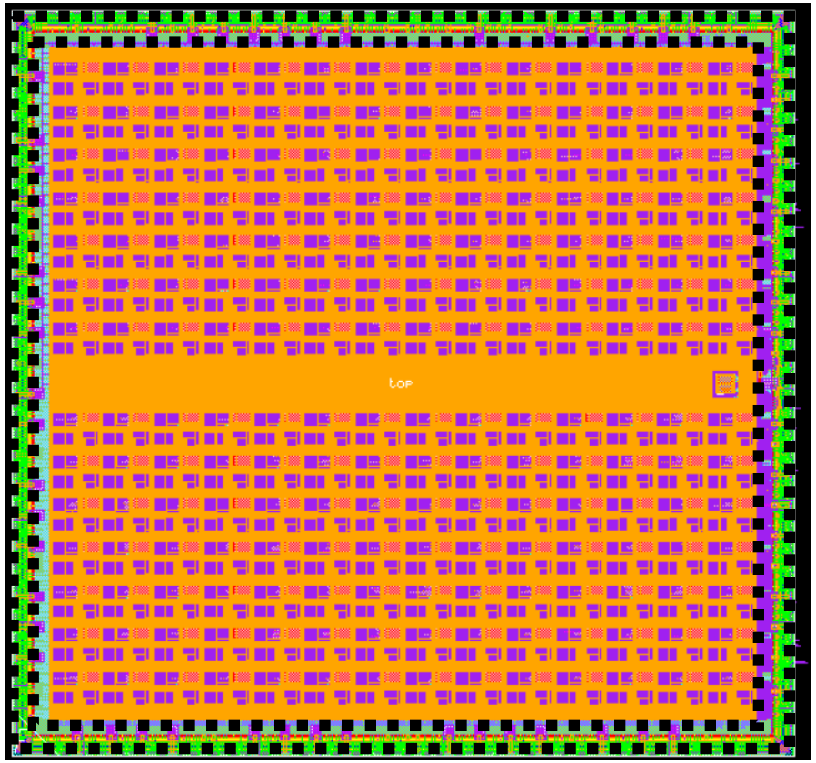


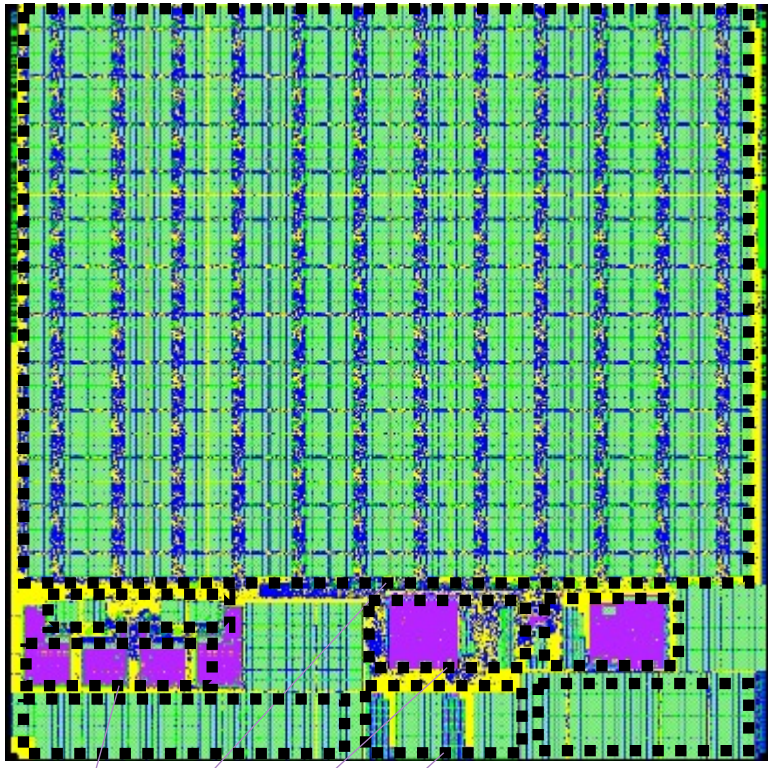
GrAI One to GrAI VIP: Improvements



	GrAI One		GrAI VIP
Technology scaling:	TSMC 28 HPC+	0.36x	TSMC 12 FFC
Silicon Area (mm2)	20.3	2.8x	57.0
Silicon Complexity: (transistors)	~318M	~14x	~4.5G
Package size (mm):	9x9	0.7x	7.6x7.6
Package pins:	120	2.7x	324
Neuron Capacity:	200,704	90x	~18,000,000
Max Nr of Parameters:	250,000	190x	~48,000,000

GrAI One to GrAI VIP: Updates

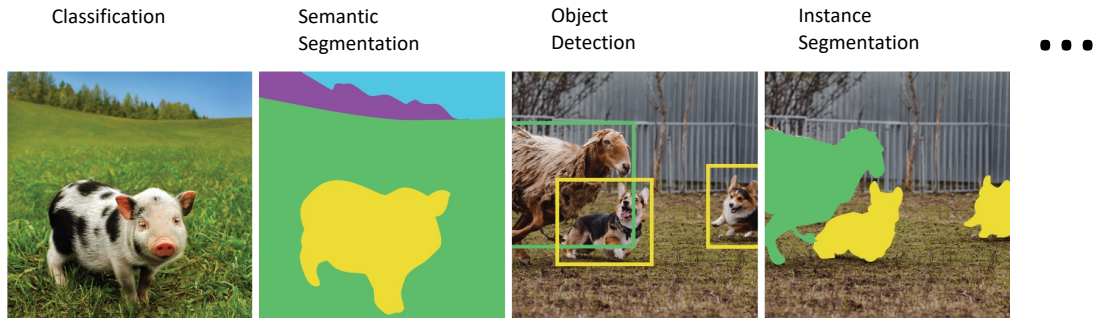




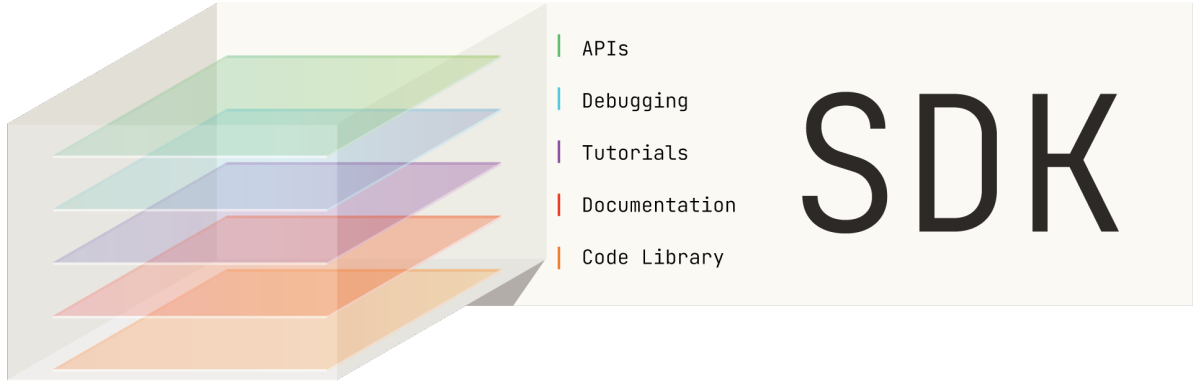
Sensor interfaces:	--	2x MIPI Tx/Rx
Image capture:	--	2x ISP+F2E
GrAI Core technology:	16-bit INT	FP16
Neuron Engines:	196	144
Host interface(s):	AER	PCIe USB3
ARM:	--	2 x CM7
Debug:	--	ARM CS600
External memory:	--	QSPI flash
Other connectivity:	--	I2S, I2C, SPI, GPIO, UART

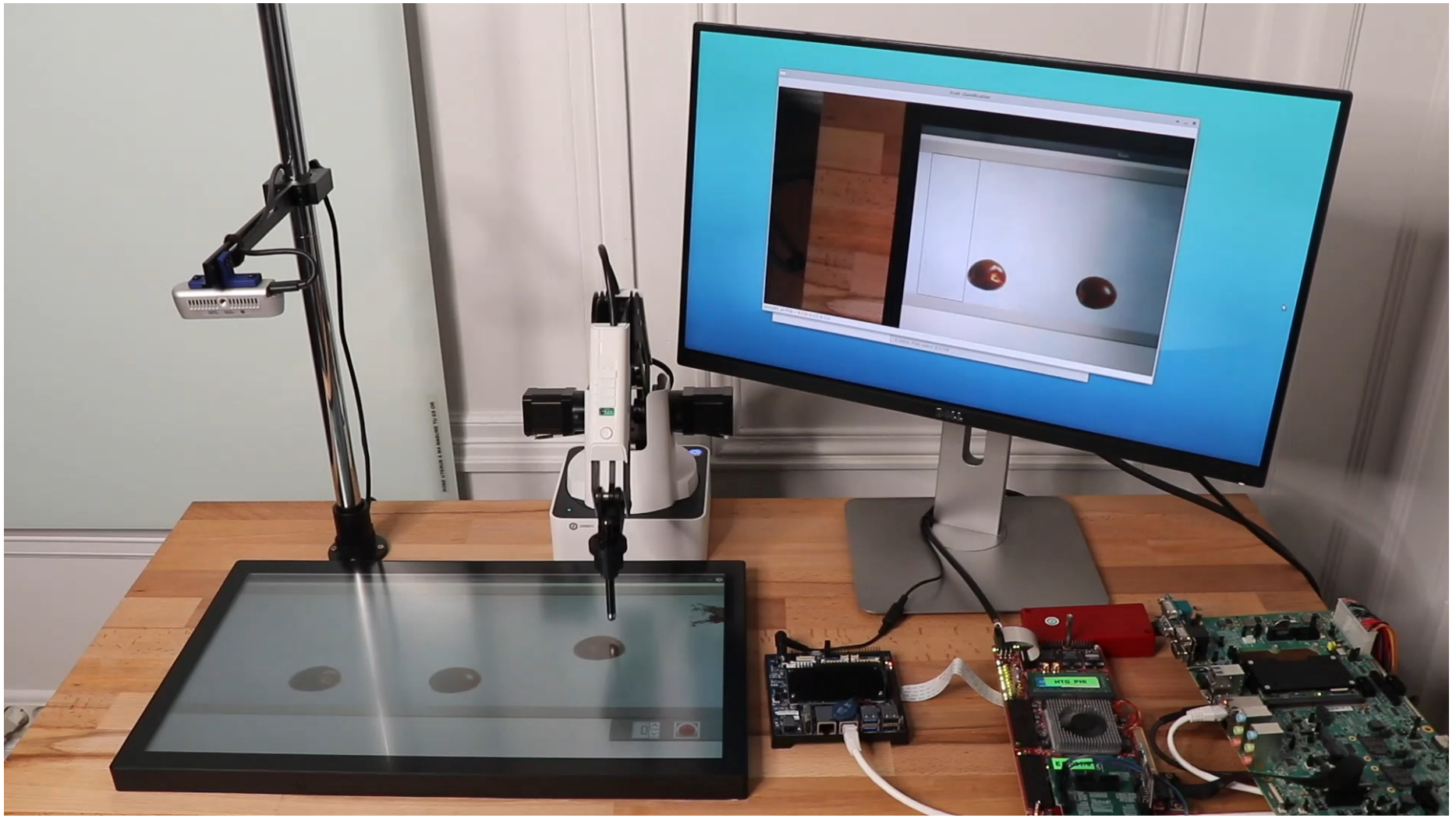
Software Development Support

Model Zoo



GrAIFlow






GrAI
Matter
Labs
Life-Ready AI is here_
www.graimatterlabs.ai
AI at the speed of Life.