DATE 25
DESIGN, AUTOMATION AND TEST IN EUROPE
THE EUROPEAN EVENT FOR ELECTRONIC SYSTEM DESIGN & TEST
31 MARCH – 2 APRIL 2025
LYON, FRANCE
CENTRE DE CONGRES DE LYON

SpikeStream: Accelerating Spiking Neural Network Inference on RISC-V Clusters with Sparse Computation Extensions

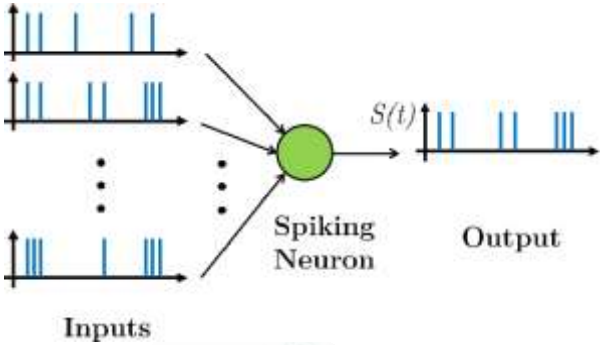
Simone Manoni*, Paul Scheffler[†], Luca Zanatta[‡], Andrea Acquaviva*, Luca Benini*[,†], Andrea Bartolini*

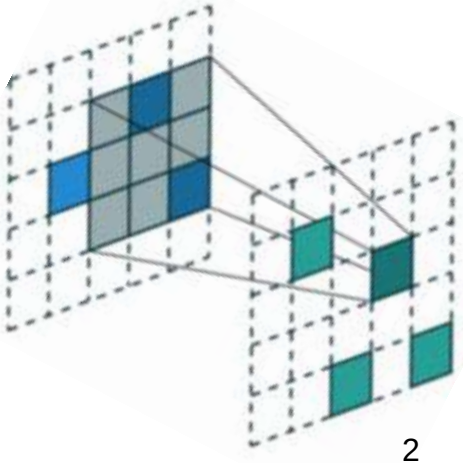
**University of Bologna*
[†]*Integrated System Laboratory, ETH Zürich*
[‡]*Autonomous Robots Lab, Norwegian University of Science and Technology (NTNU)*




Spiking Neural Networks

- **Features**:
  - Traditional neural network layers
  - **Spikes** → Binary activation tensors
  - **Spatio-temporal** processing
    - e.g. event-camera streams
  - Unstructured **Sparse activations**
    - Can reduce computational load





Challenges

- **CPUs/GPUs struggle** with **sparsity** and spikes:
  - Optimized for regular kernels
  - Irregular memory access patterns → low-utilization

Challenges

- **CPUs/GPUs struggle** with **sparsity** and spikes:
  - Optimized for regular kernels
  - Irregular memory access patterns → low-utilization

- Custom **accelerators drawbacks** (e.g. LSMCore[1], Loihi[3]):
  - Limited arithmetic precision [1-4]
  - Hardwired activation functions [1-4]
  - No General-Purpose (GP) computing [1-3]

[1] Lei Wang *et al*., ''A 69ksynapse/mm2 single-core digital neuromorphic processor for liquid state machine.'', IEEE TCAS-I '22
[2] C. Frenkel *et al*., ''A 0.086-mm2 12.7-pj/sop 64k-synapse 256-neuron online-learning digital spiking neuromorphic processor in 28-nm cmos'', IEEE TBioCAS '19
[3] M Davies *et al*., ''Loihi: A neuromorphic manycore processor with on-chip learning'', IEEE Micro '18
[4] Zhijie Yang *et al*., ''Back to homogeneous computing: A tightly-coupled neuromorphic processor with neuromorphic isa'', IEEE TPDPS '23

Challenges

- **CPUs/GPUs struggle** with **sparsity** and spikes:
  - Optimized for regular kernels
  - Irregular memory access patterns → low-utilization

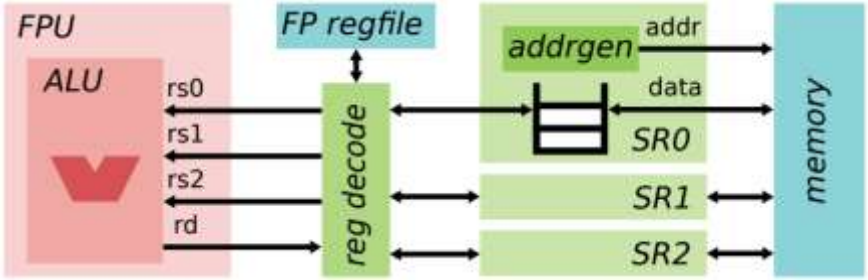
- Custom **accelerators drawbacks** (e.g. LSMCore[1],
  - Limited arithmetic precision [1-4]
  - Hardwired activation functions [1-4]
  - No General-Purpose (GP) computing [1-3]

Research question:
- Are custom SNN accelerators always necessary?
- Can GP sparse ISA extensions and smart software unlock efficient SNN execution?

[1] Lei Wang *et al*., "A 69ksynapse/mm2 single-core digital neuromorphic processor for liquid state machine.", IEEE TCAS-I '22
[2] C. Frenkel *et al*., "A 0.086-mm2 12.7-pj/sop 64k-synapse 256-neuron online-learning digital spiking neuromorphic processor in 28-nm cmos", IEEE TBioCAS '19
[3] M Davies *et al*., "Loihi: A neuromorphic manycore processor with on-chip learning", IEEE Micro '18
[4] Zhijie Yang *et al*., "Back to homogeneous computing: A tightly-coupled neuromorphic processor with neuromorphic isa", IEEE TPDPS '23

Lightweight Sparsity Support for CPUs

- **Stream Registers** (SRs) [1]
  - Proposed as ISA extension for CPUs
  - Map CPU memory streams to architectural register W/Rs
  - Address calculations + Loads & Stores → dedicated hardware units
  - **Indirect-SRs**: **base addresses** + **index arrays** to scatter-**gather** data:
    - Lightweight support for sparse workload



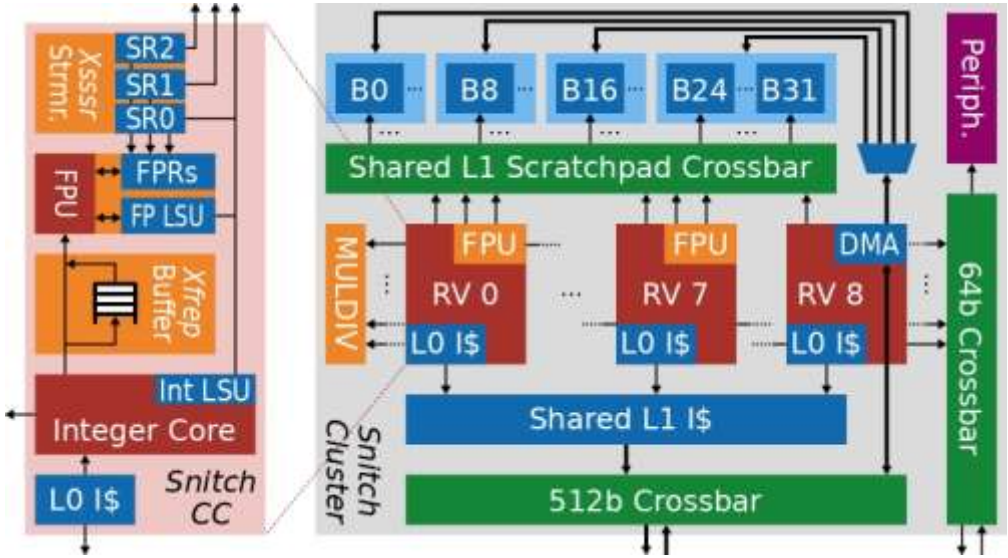
[1] Paul Scheffler *et al.*, ''SARIS: Accelerating Stencil Computations on Energy-Efficient RISC-V Compute Clusters with Indirect Stream Registers'', IEEE/ACM DAC '24.

Contributions

- ***SpikeStream***
  - SNN inference optimization based on Indirect-SRs
- **SW implementation**:
  - Parallel (8-cores) RV32G SIMD baseline
  - SpikeStream-accelerated code variant
- Average **Speedup** over **non-streaming** implementation
  - 5.62× (FP16)
  - 7.29× (FP8)
- Comparison with Neuromorphic processors
  - 3.46× energy gain over LSMCore and a performance gain of 2.38× over Loihi

SpikeStream: Target Platform

- **Snitch Cluster:**
  - 8 RV32G In-order *worker* cores
    - 64b-wide FPU
      - FP SIMD ISA extension [1]
      - FP hardware-loop [2]
    - **Indirect-SRs** ISA extension [3]
  - 1 RV32G *DMA* core
  - 128KiB L1 SPM



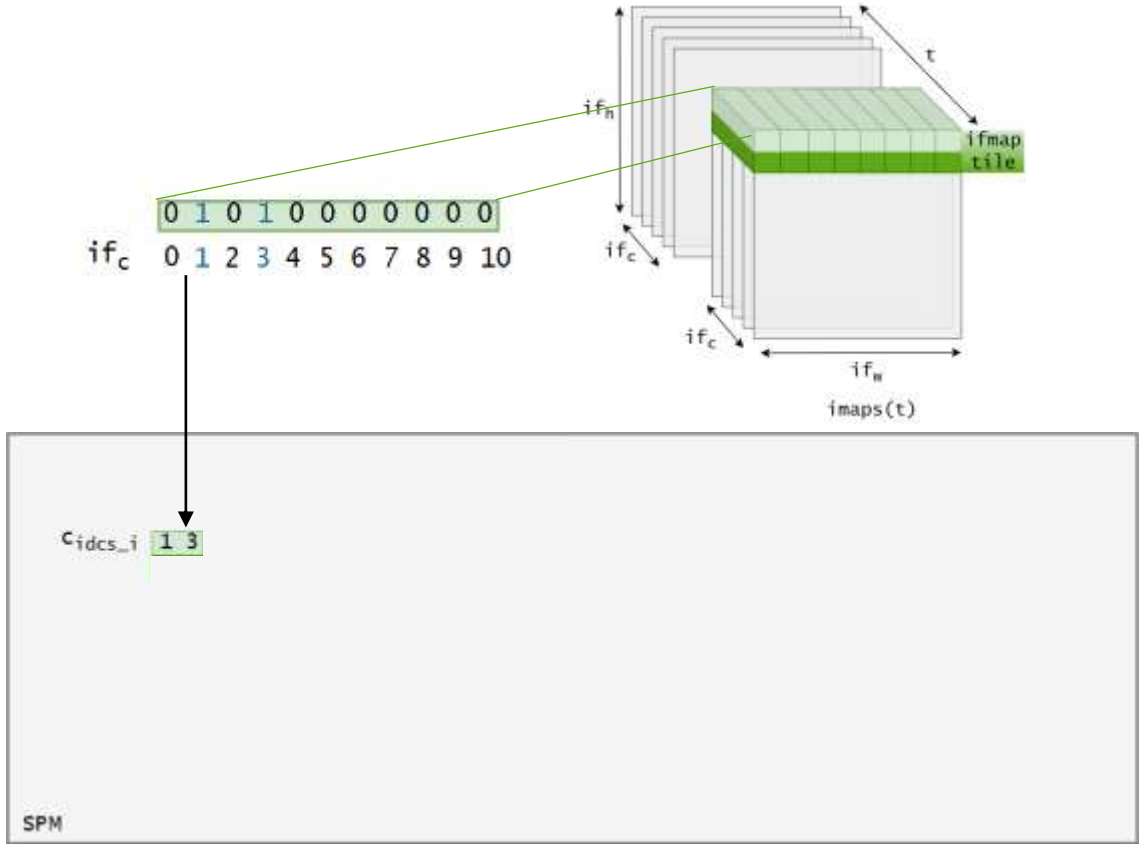
[1] G. Tagliavini *et al.*, "A transprecision floating-point platform for ultra-low power computing" IEEE *DATE '18.*
[2] F. Zaruba *et al.*, "Snitch: A Tiny Pseudo Dual-Issue Processor for Area and Energy Efficient Execution of Floating-Point Intensive Workloads," in IEEE TCOMP '21.
[3] P. Scheffler *et al.*, "Sparse Stream Semantic Registers: A Lightweight ISA Extension Accelerating General Sparse Linear Algebra'', IEEE TPDPS '23.

SpikeStream: SW Architecture

- I. Tensor Compression
- II. Data Parallelization
- III. Task Parallelization
- IV. Streaming Acceleration
- V. Double Buffering

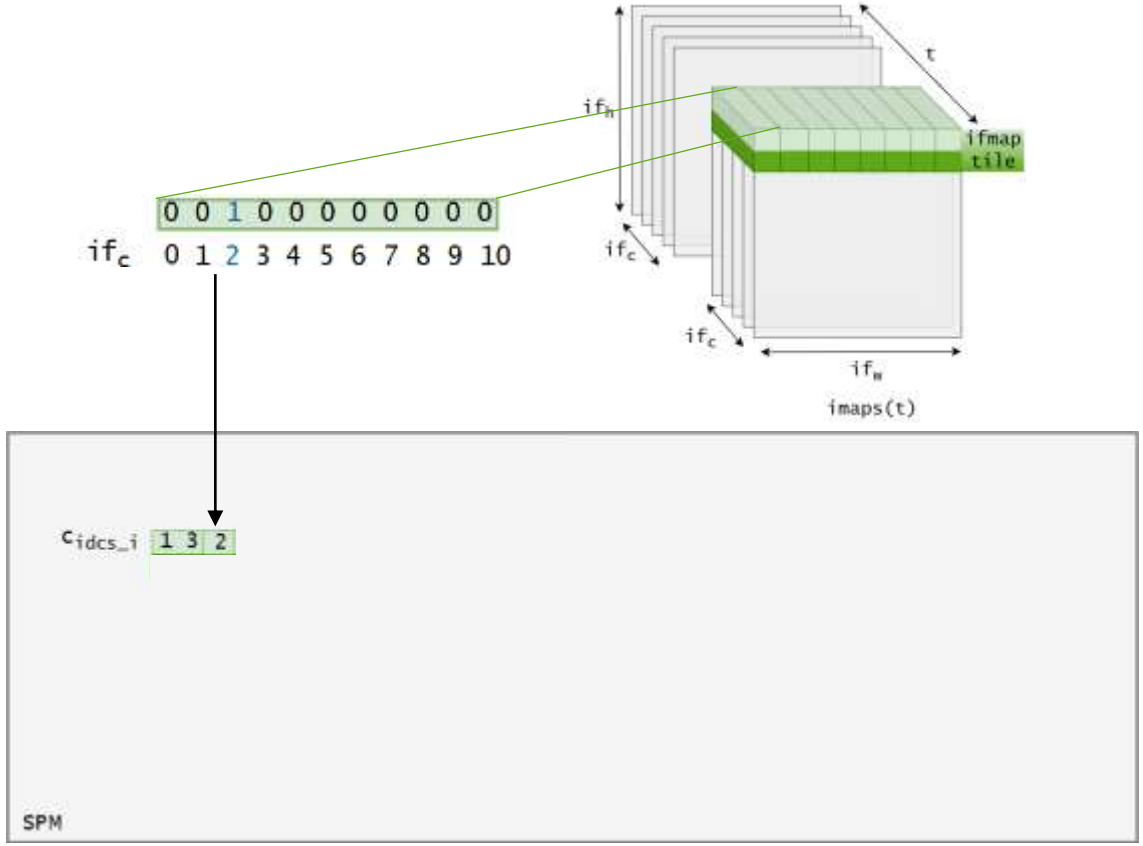
SpikeStream: SW Architecture

- **I. Tensor Compression**
- II. Data Parallelization
- III. Task Parallelization
- IV. Streaming Acceleration
- V. Double Buffering



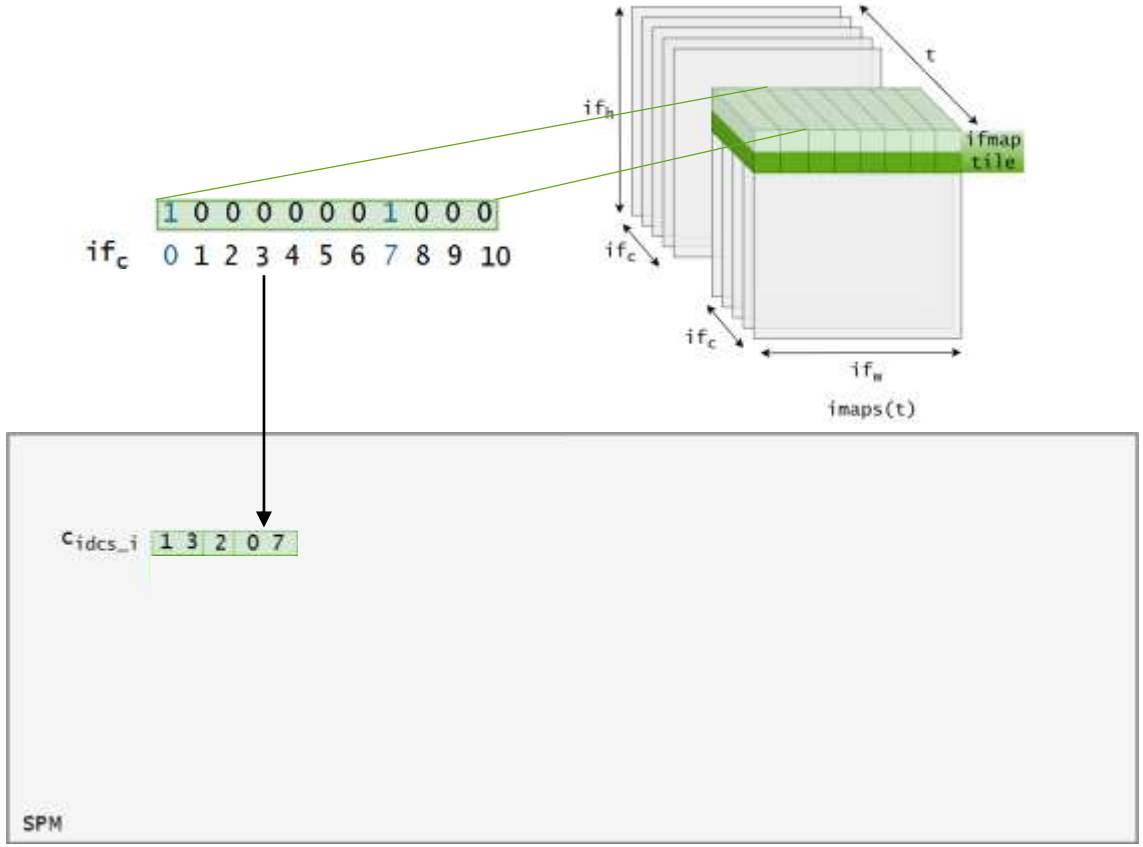
SpikeStream: SW Architecture

- **I. Tensor Compression**
- II. Data Parallelization
- III. Task Parallelization
- IV. Streaming Acceleration
- V. Double Buffering



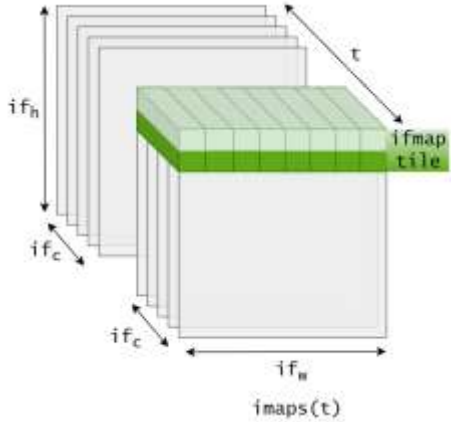
SpikeStream: SW Architecture

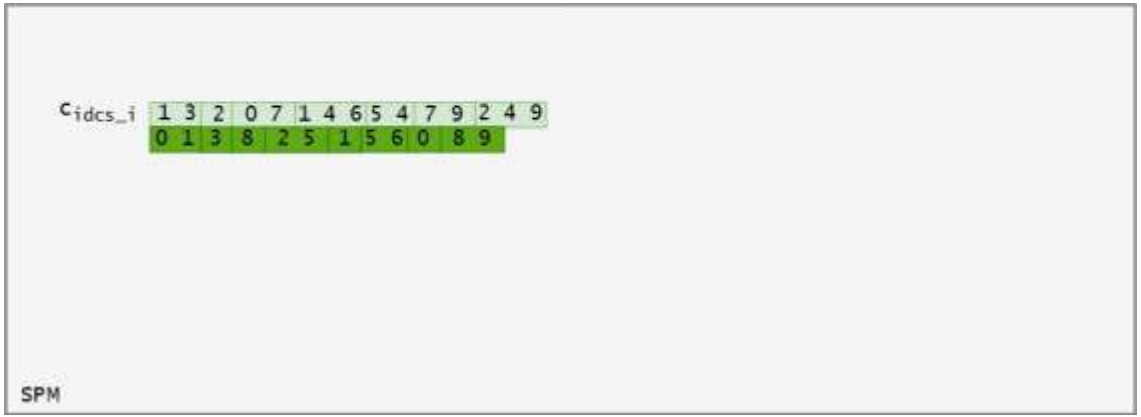
- **I. Tensor Compression**
- II. Data Parallelization
- III. Task Parallelization
- IV. Streaming Acceleration
- V. Double Buffering



SpikeStream: SW Architecture

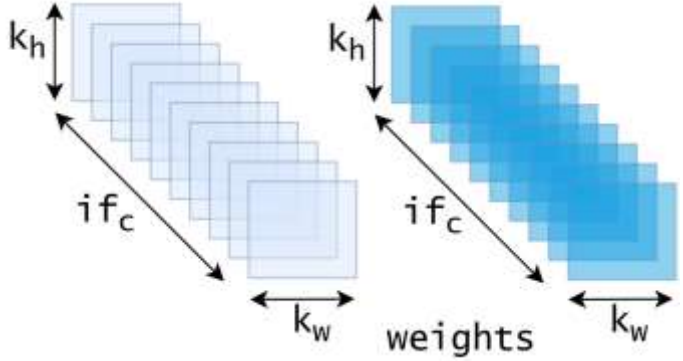
- **I. Tensor Compression**
- II. Data Parallelization
- III. Task Parallelization
- IV. Streaming Acceleration
- V. Double Buffering

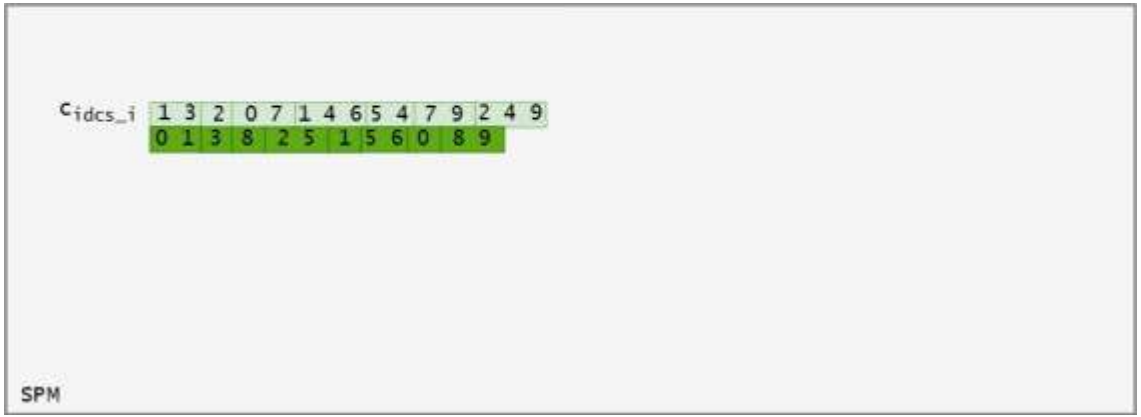




SpikeStream: SW Architecture

- **I. Tensor Compression**
- II. Data Parallelization
- III. Task Parallelization
- IV. Streaming Acceleration
- V. Double Buffering



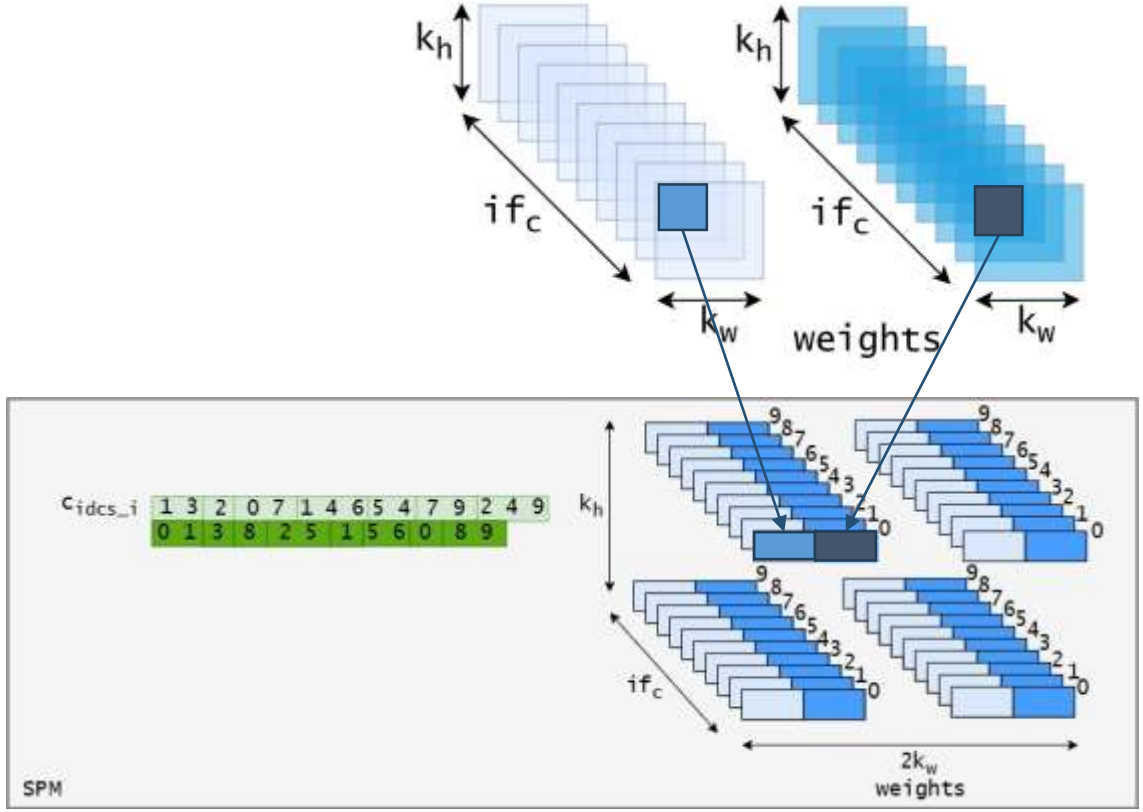


SpikeStream: SW Architecture

- I. Tensor Compression
- **II. Data Parallelization**
- III. Task Parallelization
- IV. Streaming Acceleration
- V. Double Buffering

Batched HWC memory layout:

- FP32 → 2 FPU Lanes
- FP16 → 4 FPU Lanes
- FP8 → 8 FPU Lanes

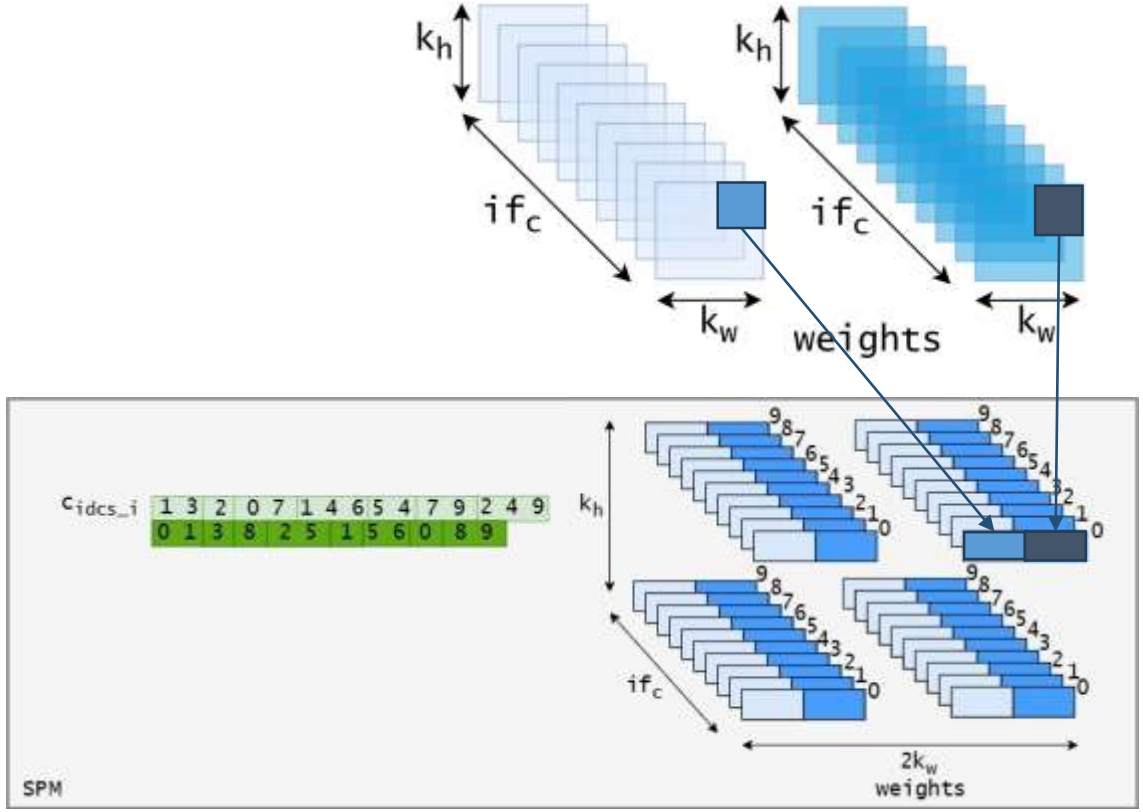


SpikeStream: SW Architecture

- I. Tensor Compression
- **II. Data Parallelization**
- III. Task Parallelization
- IV. Streaming Acceleration
- V. Double Buffering

Batched HWC memory layout:

- FP32 → 2 FPU Lanes
- FP16 → 4 FPU Lanes
- FP8 → 8 FPU Lanes

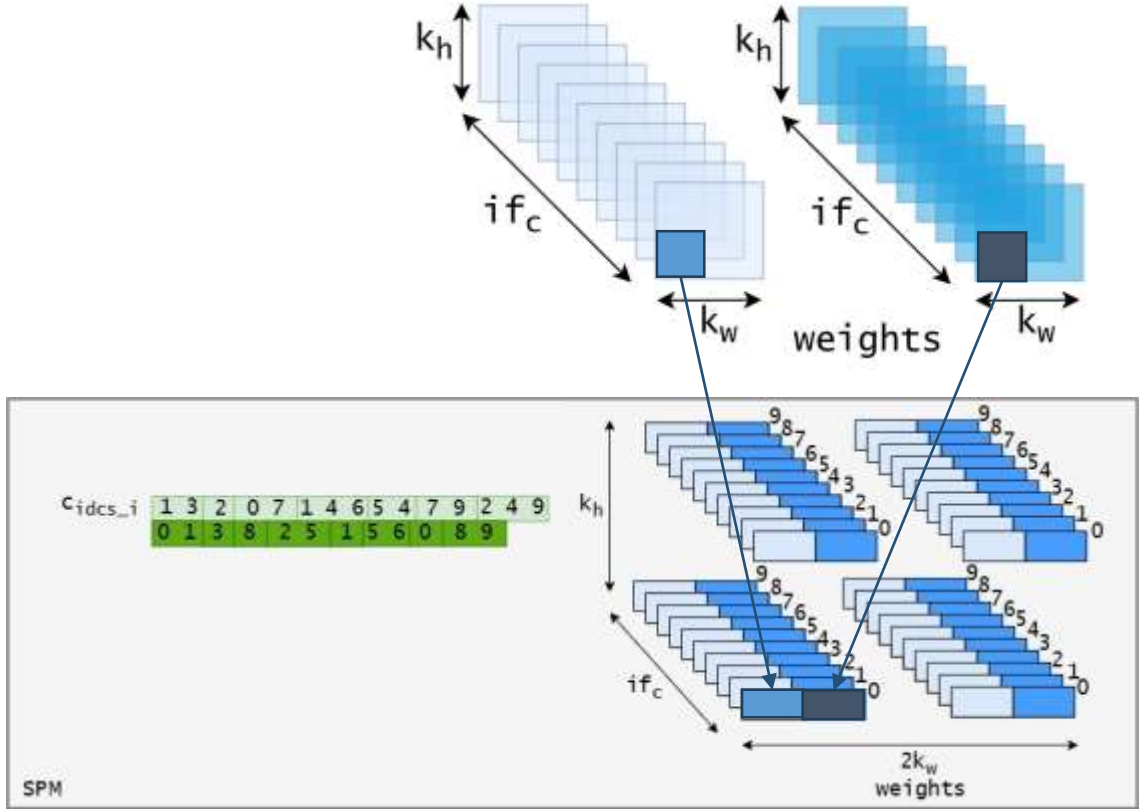


SpikeStream: SW Architecture

- I. Tensor Compression
- **II. Data Parallelization**
- III. Task Parallelization
- IV. Streaming Acceleration
- V. Double Buffering

Batched HWC memory layout:

- FP32 → 2 FPU Lanes
- FP16 → 4 FPU Lanes
- FP8 → 8 FPU Lanes

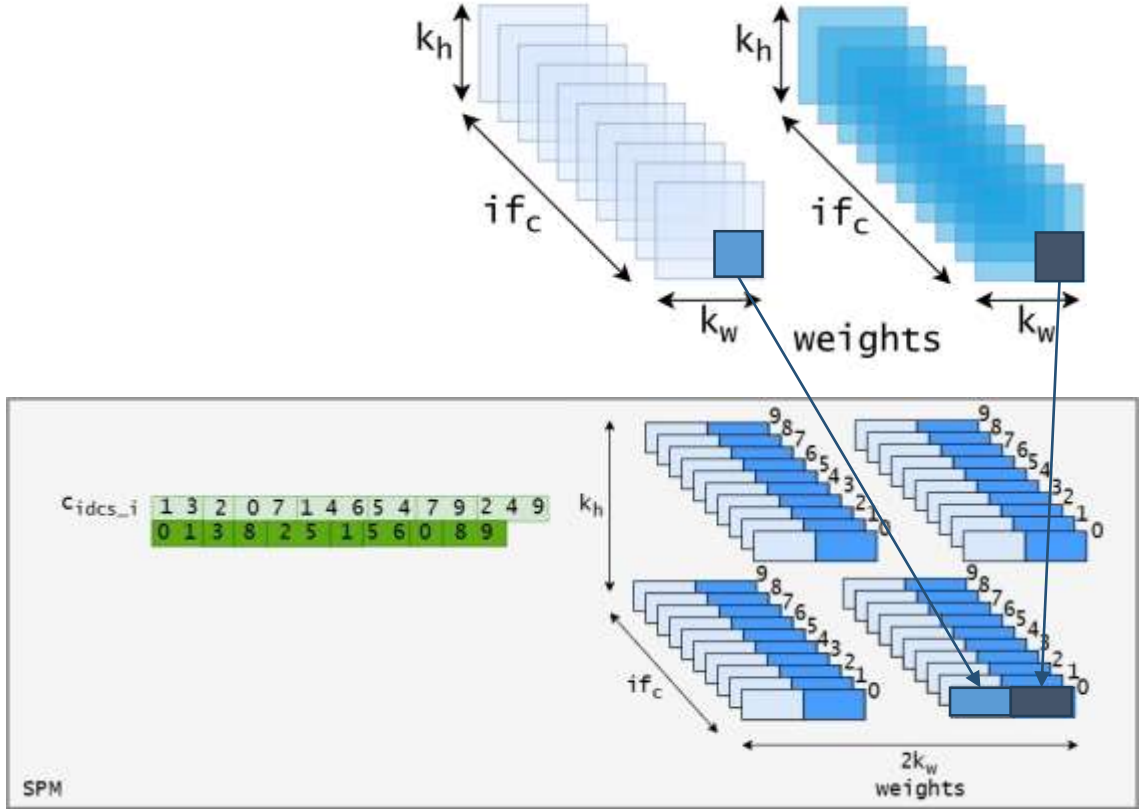


SpikeStream: SW Architecture

- I. Tensor Compression
- **II. Data Parallelization**
- III. Task Parallelization
- IV. Streaming Acceleration
- V. Double Buffering

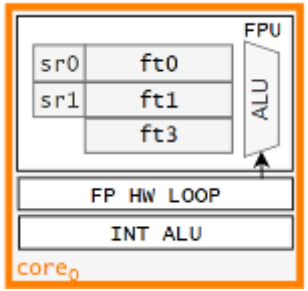
Batched HWC memory layout:

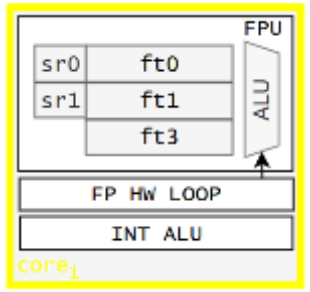
- FP32 → 2 FPU Lanes
- FP16 → 4 FPU Lanes
- FP8 → 8 FPU Lanes

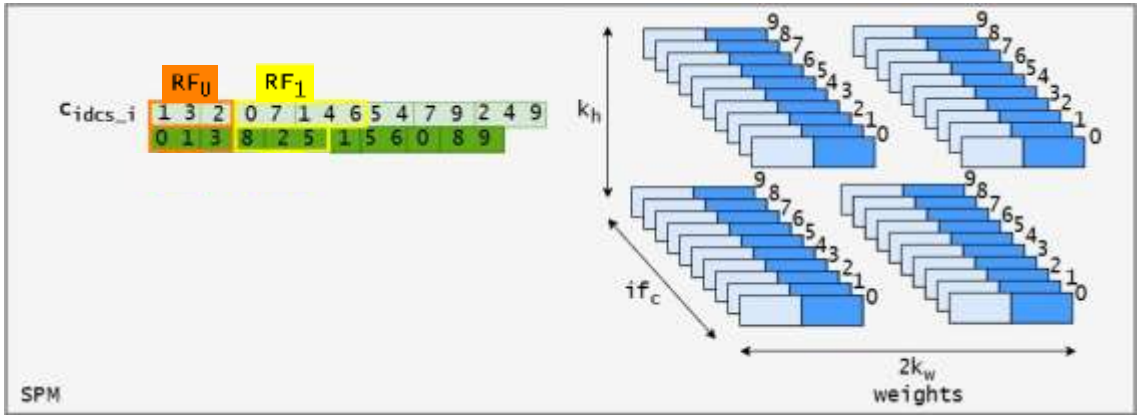


SpikeStream: SW Architecture

- I. Tensor Compression
- II. Data Parallelization
- **III. Task Parallelization**
- IV. Streaming Acceleration
- V. Double Buffering







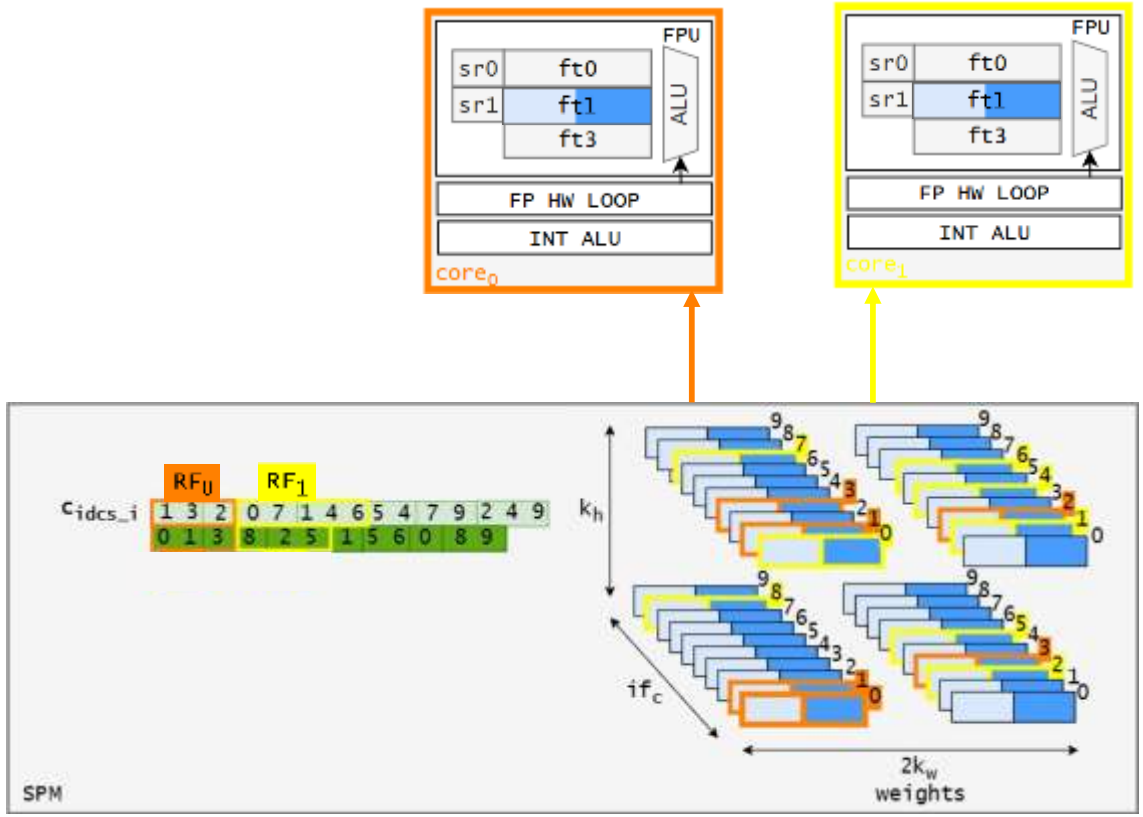
SpikeStream: SW Architecture

- I. Tensor Compression
- II. Data Parallelization
- III. Task Parallelization
- **IV. Streaming Acceleration**
- V. Double Buffering

```
SpVA: lw    t0, 0(%c_idcs_i)
      slli  t0, t0, 3
      add   t0, t0, %w
      fld   ft1, 0(t0)
      addi  %c_idcs_i, %c_idcs_i, 2
      addi  %iter, %iter, 1
      fadd  %ic, ft1, %ic
      bne   %iter, %s_len, SpVA
```

hot-loop

Baseline: RV32G

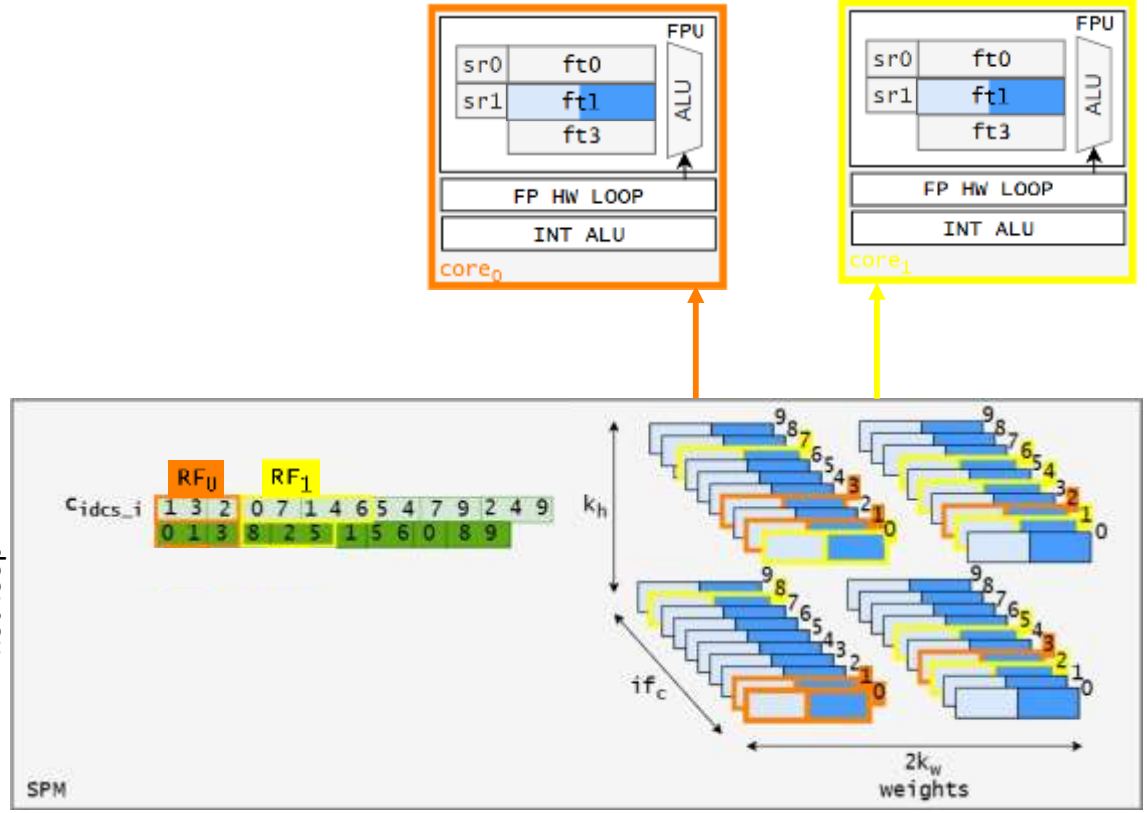


SpikeStream: SW Architecture

- I. Tensor Compression
- II. Data Parallelization
- III. Task Parallelization
- **IV. Streaming Acceleration**
- V. Double Buffering

```
cfg_sr_ft1 %w, %c_idcs_i, %s_len
frep 1, %s_len
fadd   %ic, ft1, %ic
```

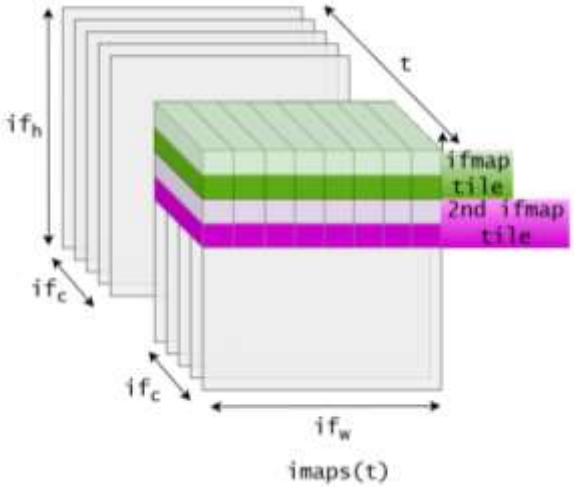
hot-loop

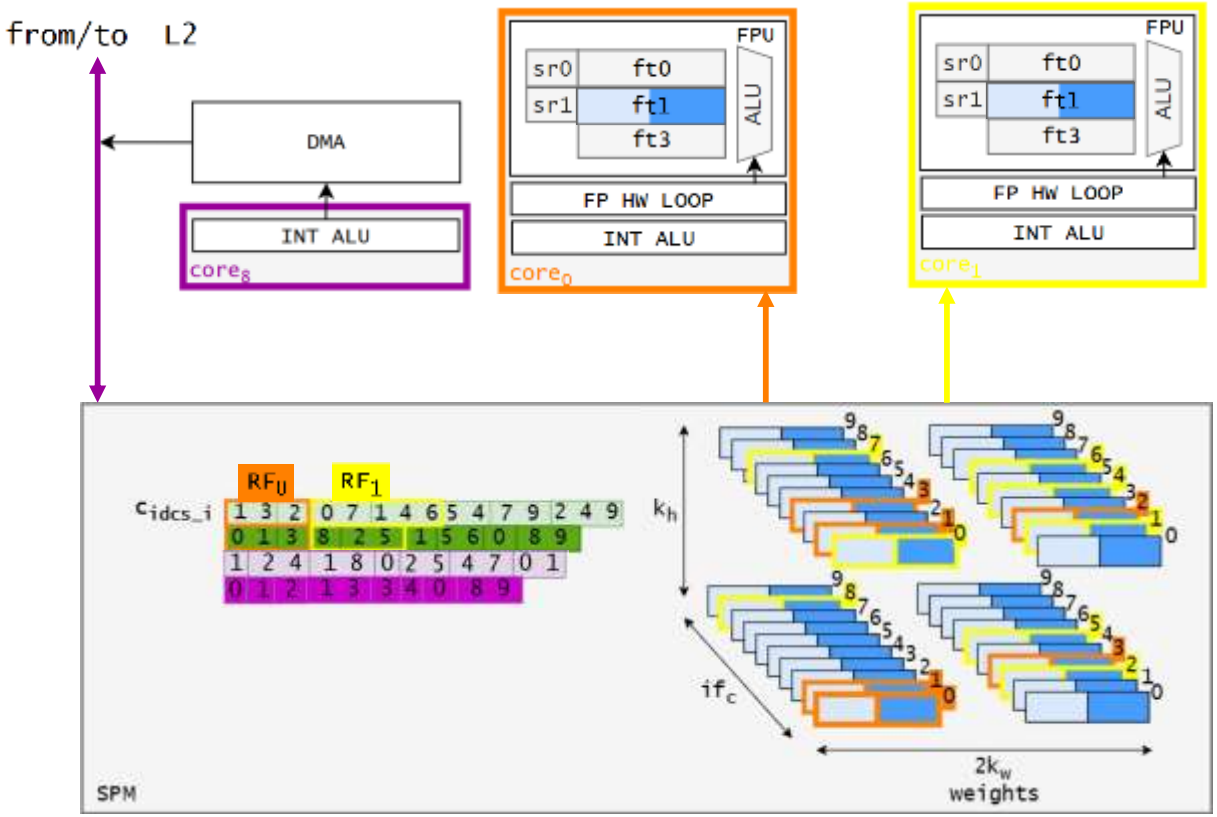


SpikeStream: RV32G
+ Indirect-SR
+ frep

SpikeStream: SW Architecture

- I. Tensor Compression
- II. Data Parallelization
- III. Task Parallelization
- IV. Streaming Acceleration
- **V. Double Buffering**





Conclusions & Results

- Proposed *SpikeStream*, an **SNN** inference **optimization** based on **Indirect-SRs**
- Average **Speedup over non-streaming implementation** (FP16):
  - 5.62× (FP16)
  - 7.29× (FP8)
- Performance gain:
  - 2.38× over Loihi
- Energy-efficiency gain:
  - 2.37x over Loihi
  - 3.46× over LSMCore
- Main SNN bottleneck on In-Order CPUs: memory inefficiencies from irregular accesses