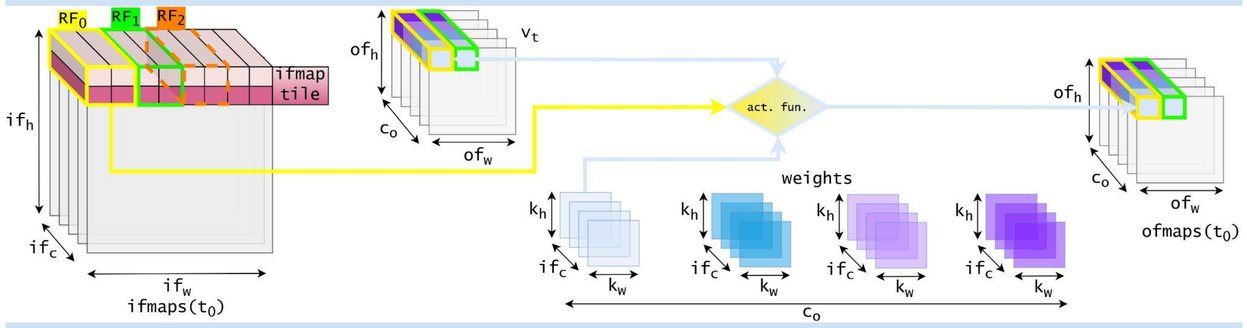
SpikeStream: Accelerating Spiking Neural Network Inference on RISC-V Clusters with Sparse Computation Extensions

S. Manoni[1], P. Scheffler[2], L. Zanatta[3], A. Acquaviva[1], L. Benini[1,2], A. Bartolini[1]

[1]*Department of Electrical, Electronic, and Information Engineering (DEI) – University of Bologna, Italy*
[2]*Integrated Systems Lab (IIS) - ETH Zurich, Switzerland*
[3]*Autonomous Robots Lab - Norwegian University of Science and Technology (NTNU) , Norway*

1. Introduction and Motivation

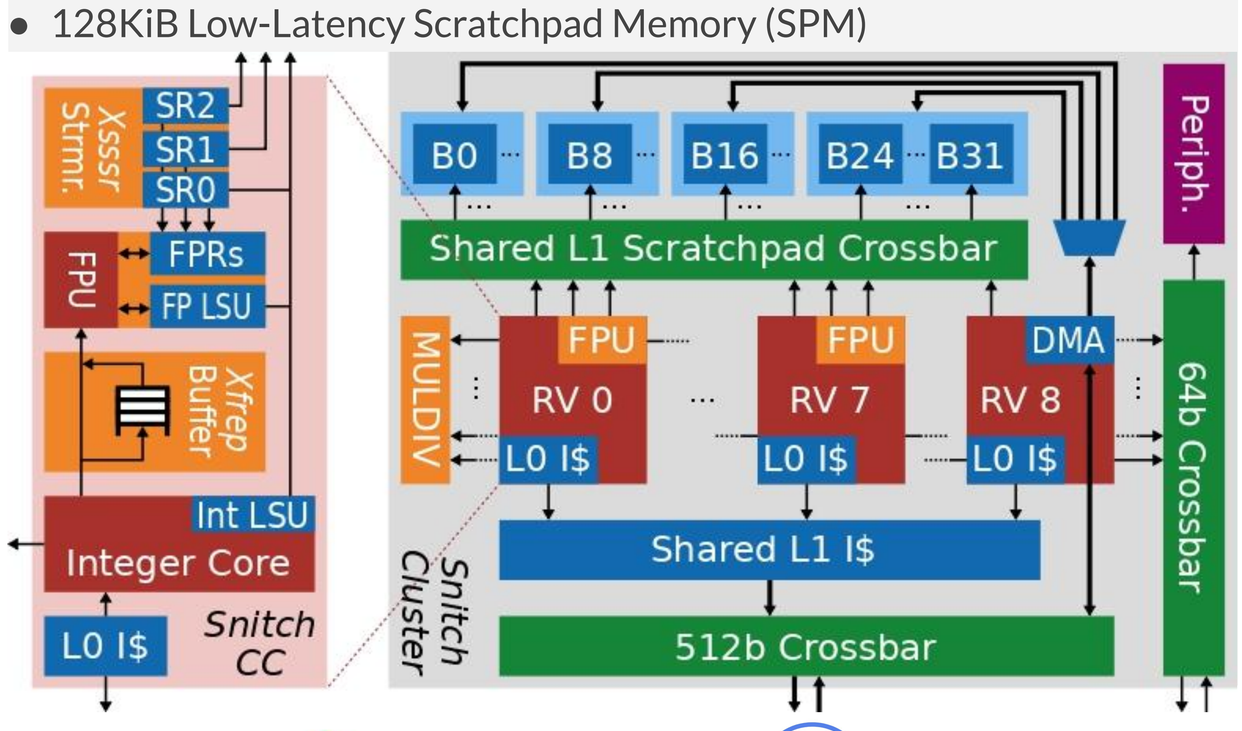
- The pursuit of efficient, low-latency machine intelligence has led to Neuromorphic systems inspired by the brain. These models rely on computations and algorithms based on **Spiking Neural Networks**
- **SNNs** incorporate **spike**-based communication between neurons, **activation sparsity**, and **complex neuronal dynamics** while maintaining traditional neural network topologies



- Traditional CPUs and GPUs struggle to deliver high-efficiency in the presence of spikes and sparsity
- Dedicated **accelerators** are often entirely designed only for SNN models, making them an **expensive** and **inflexible** solution
- **Stream Registers** (SRs) emerged as a **CPU extension** to overcome memory bottlenecks, enabling **hardware-managed data streaming** with **support** for **indirect** (gather/scatter) **access** for sparse workloads

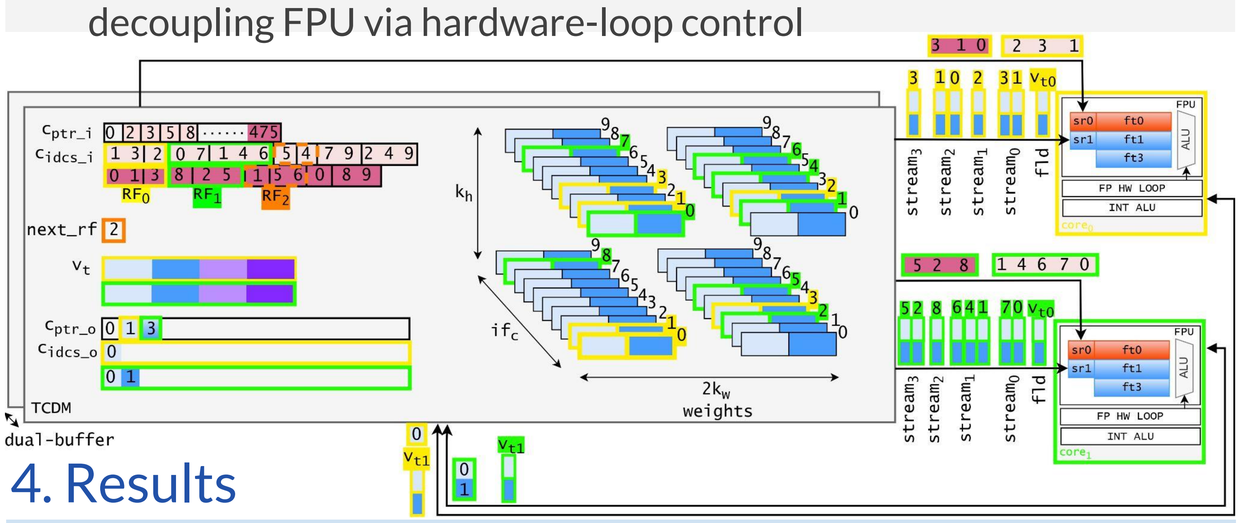
2. Target Platform: Snitch Cluster

- 8 RV32G Cores enhanced with
  - Double-precision FPU
  - Indirect SRs
  - Floating-point HW loops
  - FP SIMD extension
- 1 DMA Core
- 128KiB Low-Latency Scratchpad Memory (SPM)



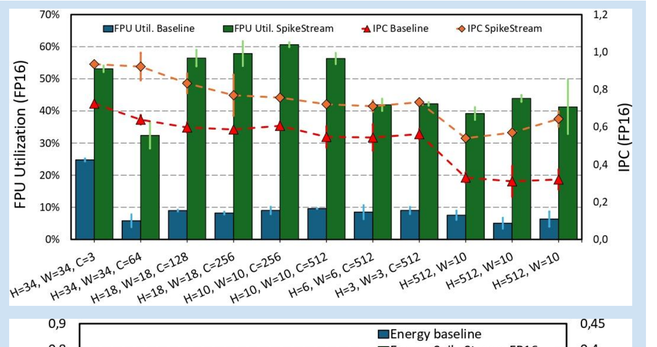
3. SpikeStream Software Architecture

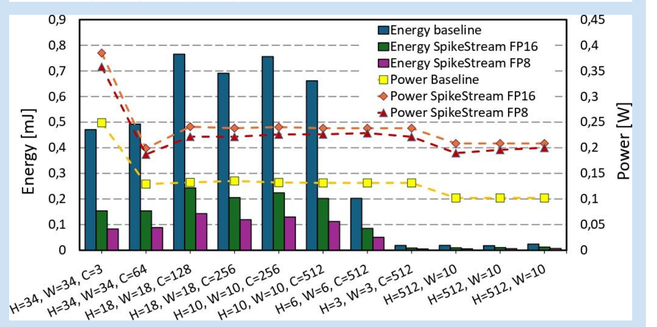
- We introduce target architecture-aware optimizations
  - **Tensor compression**: CSR-derived fibre tree format storing binary activations as spike positions using indices and spatial pointers
  - **Task Parallelization**: Computation parallelized across Snitch Cluster cores (receptive field per core). Workload-stealing with atomic tagging balances irregular parallelization.
  - **Data parallelization**: Batched HWC weight layout enables output channel parallelism across FPU lanes
  - **Double Buffering:** DMA core used for sparse activation tiling
  - **Streaming Acceleration (SA)**: Indirect weight loads mapped to indexed streams (SR-managed address gen/memory ops), decoupling FPU via hardware-loop control

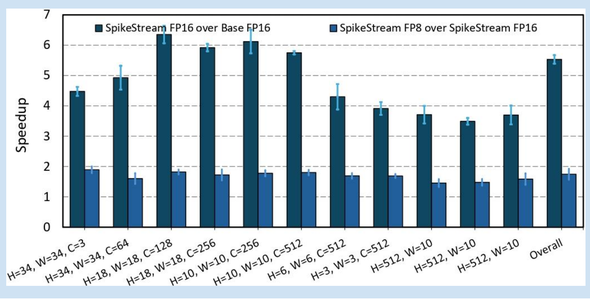


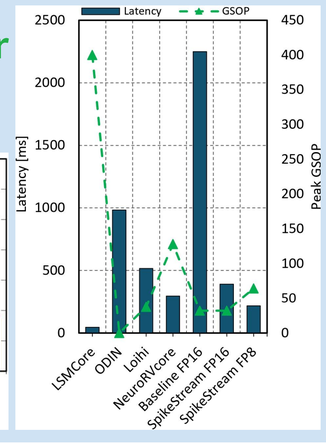
4. Results

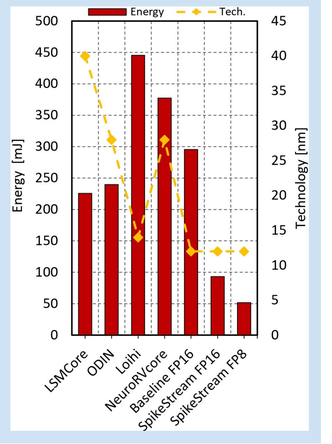
- Synth in **GF12LP+**. Energy from post-layout sim @1GHz-0.8V:
  - **Performance:**
    - Comparison over w.r.t **non-SA FP16: Avg. 5.4x (FP16), 9.8x (FP8)**
    - SpikeStream FP8 is **4.71× slower** than **LSMCore** but **2.38× faster** than **Loihi**
  - **Energy:**
    - SpikeStream FP16 (FP8) achieves **2.37 (3.46)× lower energy** vs. **LSMCore**











s.manoni@unibo.it