# Mega: A 22 nm Convolutional Spiking Neural Network Accelerator Achieving 0.375 pJ/SOP for Efficient Edge Vision

Rick Luiken*, Manil Dev Gomony and Sander Stuijk
Dept. of Electrical Engineering, Eindhoven University of Technology, The Netherlands
*Correspondence to h.j.luiken@tue.nl

***Abstract*—Convolutional Spiking Neural Networks (SNN) offer the potential for highly energy-efficient vision processing by exploiting sparse, event-driven computation. However, existing SNN accelerators underutilize the inherent parallelism of convolutional layers and lack the flexibility to accommodate varying memory demands and input sparsity across layers. This paper presents Mega, a digital architecture for convolutional SNNs that addresses these limitations through three key contributions: (1) highly parallel acceleration of $3 \times 3$ convolutions, (2) a unified data memory for spikes, neuron states, and weights, and (3) efficient spike map processing with low-overhead spike detection. Fabricated in GlobalFoundries 22 nm FDSOI technology, Mega achieves an energy efficiency of 0.375 pJ/SOP, improving the state of the art by $4\times$.**

***Index Terms*—SNNs, neuromorphic computing, edge AI**

## I. Introduction

Enabling vision-based applications at the edge, where energy and compute resources are scarce, requires highly efficient processing. Dynamic Vision Sensors (DVS) [1] address this need by converting changes in visual scenes into sparse, event-driven spikes with high temporal resolution. Convolutional Spiking Neural Networks (SNNs) naturally exploit this sparsity, enabling energy-efficient inference for vision tasks.

Despite the potential, accelerating convolutional SNNs in hardware remains challenging. Existing SNN accelerators often fail to fully exploit the parallelism of convolutional layers, particularly for small kernels such as $3 \times 3$, which dominate network designs. Memory demands also vary across layers: initial layers often require high-resolution feature maps, making neuron state memory dominant, while deeper layers have many channels, making weight memory dominant. Additionally, common spike communication schemes, such as Address-Event Representation (AER), are efficient only at very high sparsity. However, typical sparsity levels vary per layer, reducing their effectiveness.

To address these challenges, this paper presents Mega, a digital architecture for convolutional SNNs. Mega introduces three key innovations, illustrated in Fig. 1:

1) Highly parallel $3 \times 3$ spiking convolution acceleration, exploiting kernel-level and output channel parallelism.
2) A unified data memory for spikes, neuron states, and weights that supports different layer types.
3) Low-overhead spike detection and address conversion for efficient event-driven processing.

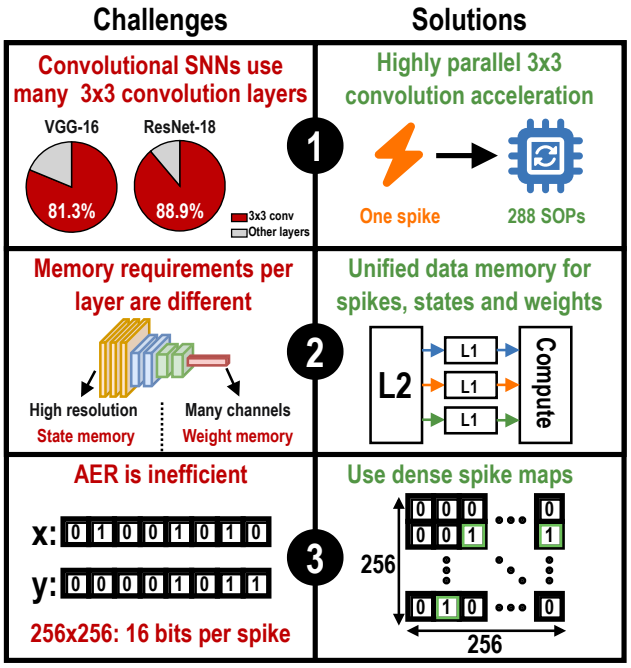


Fig. 1. **Challenges and Solutions**. The left side illustrates common challenges encountered when accelerating convolutional SNNs, while the right side presents our corresponding solutions.

## II. Hardware Architecture

Fig. 2 shows the hardware architecture of Mega. The design is centered around nine compute clusters that collectively perform parallel $3\times3$ spiking convolutions.

At a high level, input spikes are loaded by the spike streamer, which converts spike vectors into spike addresses. These addresses are sent to the clusters, where they are broadcast to the convolution units (CUs). The CUs use these addresses to fetch the corresponding weights and neuron states to perform accumulation in parallel. After all spikes within a timestep have been processed, the threshold unit in each cluster generates the output spikes.

### A. Spiking Convolution on Compute Clusters

Each compute cluster integrates 32 CUs, along with local neuron state memory and a threshold unit.

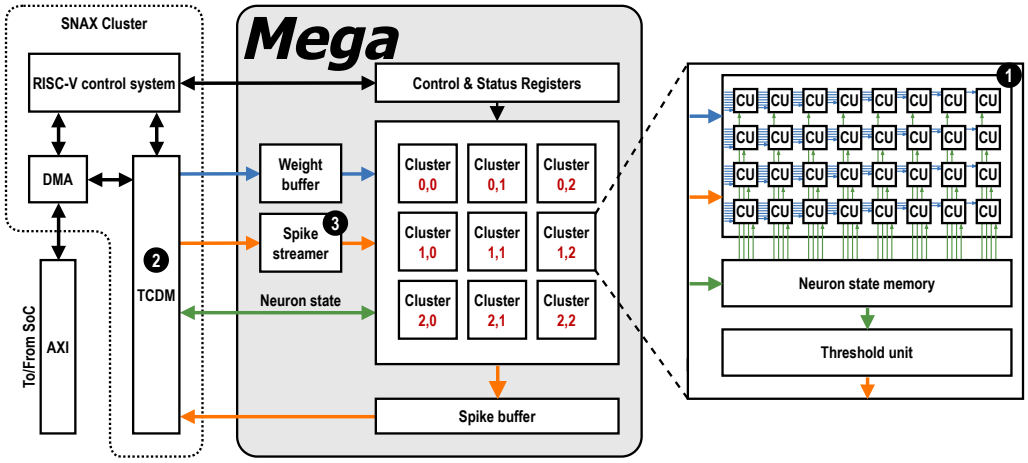


Fig. 2. **Mega architecture diagram**. Mega is built around nine compute clusters that compute $3 \times 3$ convolutions in parallel. Each cluster updates 32 neuron states in parallel using 32 convolution units (CUs). Spikes, weights, and neuron states are fetched from the integrated SNAX cluster [2]. The spike streamer converts dense spike maps into sparse spike addresses, which are forwarded to the compute clusters.

*1) Spiking convolution:* Convolutional Neural Networks (CNNs) commonly employ a sliding window, frame-based approach. For a $3 \times 3$ kernel $W$ and input $I$, the output at location $(x, y)$ is given by

$$O[x, y] = \sum_{a=-1}^{1} \sum_{b=-1}^{1} I[x + a, y + b]W[a, b]. \quad (1)$$

This formulation is output-centric, requiring all input elements contributing to a given output location to be fetched. In SNNs, where spikes are highly sparse, this approach leads to fetching many zeros, which makes processing inefficient.

An alternative approach, first proposed in [3], defines the convolution in an event-driven manner. For an incoming spike at location $(u, v)$, the spike $S[u, v]$ is included in the sum for neuron state $V[x, y]$ when

$$(u, v) = (x + a, y + b), \quad (a, b) \in \{-1, 0, 1\}^2, \quad (2)$$

which can be rewritten as

$$x = u - a, \quad y = v - b. \quad (3)$$

This leads to the following update rule:

$$\Delta V[u - a, v - b] \mathrel{+}= W[a, b] \text{ for } (a, b) \in \{-1, 0, 1\}^2. \quad (4)$$

This event-driven update rule produces the same results as the frame-based approach. However, since the update rule is event-driven, computation scales with the number of input spikes.

*2) Neuron state memory:* The update rule in (4) allows for nine neuron updates in parallel. To fetch the nine neuron states in parallel, we extend the memory interlacing scheme used in [4]. As shown in Fig. 3, neuron states are distributed across nine dual-ported SRAM banks, such that each $3 \times 3$ window of neuron states can be accessed concurrently. To further increase throughput, we exploit parallelism across output channels. Unlike in [4], each SRAM word stores 32 neuron states

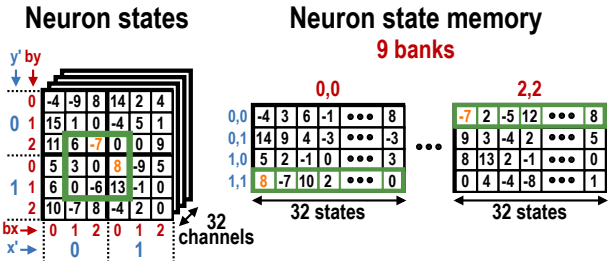


Fig. 3. **Neuron state memory organization**. Neuron states are distributed across nine SRAM banks. The $(x, y)$ coordinate of a state is transformed into $(x', y', bx, by)$, as shown on the left. The bank indices $(bx, by)$ select one of the nine banks, while $(x', y')$ defines the word address within the bank. Each memory word stores 32 neuron states corresponding to 32 output channels.

corresponding to 32 output channels. With nine SRAM banks, the architecture supports 288 concurrent neuron state reads and writes per cycle. Each SRAM bank has a capacity of 16 kB, resulting in a total neuron state memory of 144 kB.

*3) Convolution unit:* Mega employs nine compute clusters, each responsible for one of the nine kernel offsets in (4), enabling fully parallel kernel updates. Within each cluster, 32 CUs operate in parallel to process 32 output channels, resulting in a total of 288 neuron updates per cycle.

Each CU implements a four-stage pipeline:

1) **Address generation**: Compute the target neuron address from the input spike coordinates.
2) **Neuron state fetch**: Read the 8-bit neuron state from the neuron state memory.
3) **Neuron update**: Add the weight to the neuron state. Overflow is handled by clipping the neuron state.
4) **Write-back**: Store the updated neuron state in the neuron state memory.

Read-After-Write (RAW) hazards can occur when a neuron state is read before it has been updated or written back. These

hazards are detected during the address calculation stage and are resolved by data forwarding, eliminating pipeline stalls.

*4) Threshold unit:* The threshold unit applies leakage and thresholding. Mega employs leaky integrate-and-fire (LIF) neurons with linear decay. Thresholding is performed in a row-wise manner, activating three of the nine thresholding units concurrently. Output spikes are buffered before being written back to the shared memory.

The threshold unit also implements a four-stage pipeline:

1) **Address calculation**: Neuron states are accessed sequentially. The address is generated using a counter.
2) **Neuron state fetch**: Read 32 neuron states in parallel.
3) **Leak and threshold**: Apply linear leakage to the signed 8-bit neuron states, followed by the threshold comparison. Neurons exceeding the threshold emit a spike.
4) **Write-back**: Store the updated neuron states in the neuron state memory. Spiking neurons are reset to zero; otherwise, the leaked neuron state is written back.

### B. SoC Integration

To increase flexibility, Mega is integrated with the SNAX framework [2], which provides a RISC-V control system and a Tightly-Coupled Data Memory (TCDM).

The RISC-V control system communicates with Mega through the Control and Status Registers (CSRs) interface of Mega, enabling the configuration of parameters such as neuron leakage and the threshold. The RISC-V controller also manages data transfers between Mega and the TCDM. The TCDM consists of 32 single-port SRAM banks of 8 kB each, totaling 256 kB. It stores spikes, neuron states, and weights before they are loaded into Mega's local memories. Mega can access the entire TCDM, providing flexibility to efficiently handle layers with either large neuron states or large weight kernels.

### C. Spike streamer

The spike streamer converts dense binary spike maps into sparse spike addresses, enabling the spiking convolution explained in Section II-A. It fetches 96-bit spike vectors from the TCDM and finds spikes using a Leading Zero Counter (LZC), which is implemented efficiently as a tree structure. After a spike is extracted, the corresponding bit is cleared, allowing the LZC to locate the next spike. Once all spikes in a vector have been processed, the spike streamer proceeds to the next vector.

To avoid stalls when fetching the next spike vector, we prefetch it while the current vector is being processed. A second LZC operates on the prefetched vector, allowing immediate continuation without stalls.

The spike location $(x, y)$ is derived from the combination of the spike vector address and the LZC output. Due to the memory interlacing scheme used, the location needs to be converted to the $(x', y', bx, by)$ format. To avoid expensive divide-by-3 and modulo-3 operations, $y'$ and $by$ are generated using lightweight counters. However, since the $x$ coordinate depends directly on the LZC output, we cannot use counters to compute $x'$ and $bx$. Therefore, we employ two small lookup tables (LUTs) with 96 entries, which store the $x'$ and $bx$ corresponding to each LZC output.

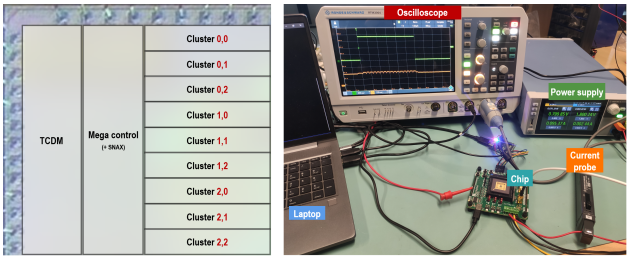


Fig. 4. Chip micrograph and measurement setup

TABLE I
CHIP SUMMARY

| | | |
|---|---|---|
| **Technology** | 22 nm FDSOI CMOS | |
| **Area** | 1.12 mm$^2$ | |
| **Supply voltage** | 0.55 V - 1.1 V | |
| **Frequency** | 155 MHz - 600 MHz | |
| **Network type** | Convolutional SNN | |
| **Weight/State precision** | INT4/INT8 | |
| | 0.55 V @ 155 MHz | 1.1 V @ 600 MHz |
| **Power consumption** | 14.4 mW | 144.7 mW |
| **Energy efficiency** | 0.375 pJ/SOP | 0.978 pJ/SOP |
| **Throughput** | 38.4 GSOP/s | 148.7 GSOP/s |

## III. MEASUREMENT RESULTS

Mega is fabricated as part of a larger SoC tapeout in GlobalFoundries 22 nm FDSOI. It occupies 1.12 mm$^2$, which is 23.0 % of the SoC area. The chip micrograph and measurement setup are shown in Fig. 4. The chip summary is given in Table I.

As Mega is integrated within a larger SoC sharing a single power domain, its individual contributions to leakage and dynamic power must be determined. Leakage power is found by measuring the total leakage and scaling it proportionally to the area occupied by Mega. Dynamic power is obtained by measuring the total chip power with and without activity on Mega, and taking the difference between the measurements.

Fig. 5 shows the energy and latency of Mega across different input sparsity levels. Both the energy and latency scale well with the input sparsity as a result of the event-driven spiking convolution. Fig. 6 presents the energy efficiency and performance tradeoff at different voltages. All tested samples of the chip operate from 0.55 V to 1.1 V. At 0.55 V, Mega runs at up to 155 MHz, consuming just 0.375 pJ/SOP.

We evaluate Mega on the IBM DVS gestures dataset [10], a widely used benchmark for event-based vision. The implemented network, shown in Fig. 7, achieves an accuracy of 96.1 %. Each convolutional layer is deployed and evaluated individually on Mega using inputs with representative sparsity levels. Aggregating the per-layer measurements, Mega consumes 174.9 µJ/inference, demonstrating its suitability for energy-efficient edge vision applications.

TABLE II
COMPARISON OF NEUROMORPHIC ACCELERATORS

| | ReckOn [5] ISSCC'22 | ANP-I [6] ISSCC'23 | Sparse-IMC [7] ISSCC'24 | SpiDR [8] ESSERC'25 | SpikeRAM [9] ISSCC'26 | **Mega This work** |
|---|---|---|---|---|---|---|
| **Technology** | 28 nm | 28 nm | 22 nm | 65 nm | 65/40 nm | **22 nm** |
| **Voltage** | 0.5-0.8V | 0.56-0.9V | 0.55-0.9V | 0.9-1.2V | 0.8-1.1V | **0.55-1.1V** |
| **Clock frequency** | 13-115 MHz | Async | 51-280 MHz | 50-150 MHz | 50-210 MHz | **155-600 MHz** |
| **Network type** | SRNN | FC SNN | Conv SNN | FC/Conv SNN | FC/Conv SNN | **Conv SNN** |
| **State resolution** | INT16 | (N/A) | (N/A) | INT7, INT11, INT15 | INT16 | **INT8** |
| **Weight resolution** | INT8 | INT8 | INT4, INT8 | INT4, INT6, INT8 | INT8 | **INT4** |
| **Neuron model** | LIF | LIF | LIF | IF, LIF, RMP | LIF | **LIF** |
| **DVS gestures Accuracy** | 87.3 % | 92.0 % | 94.0 % | 93.5 % | 92.9 % | **96.1 %** |
| **Energy Efficiency** | 5.3-12.8 pJ/SOP | 1.5 pJ/SOP | 3.87 pJ/SOP | 5 TOPS/W* | 11.67 pJ/SOP | **0.375 pJ/SOP** |

*Measured at 95 % sparsity, includes operations skipped due to zeros.

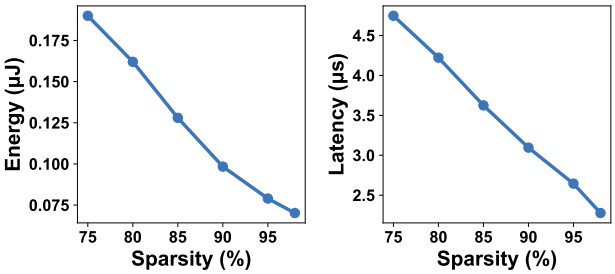


Fig. 5. Energy and latency versus input sparsities for a $96 \times 50$ spike map.

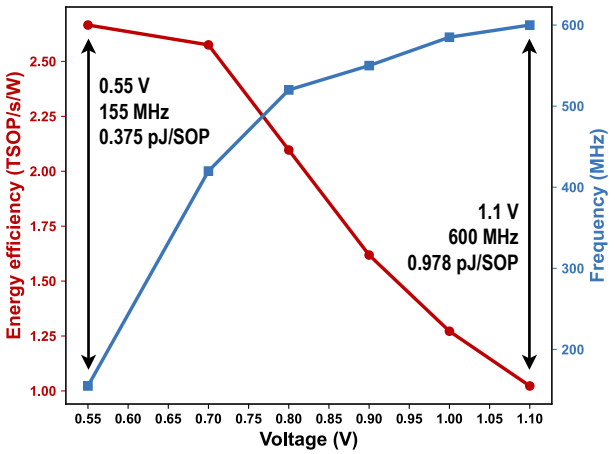


Fig. 6. Energy efficiency and performance tradeoff.

Table II compares Mega with recent state-of-the-art neuromorphic accelerators. ReckOn [5] and ANP-I [6] target (recurrent) fully connected networks, limiting their scalability to larger vision workloads. In contrast, Mega supports convolutional SNNs, enabling efficient execution of more complex and larger-scale networks. [7]–[9] adopt Compute-In-Memory (CIM) to support convolutional SNNs. Despite not using CIM, Mega achieves the highest energy efficiency, improving on prior work by $4\times$.

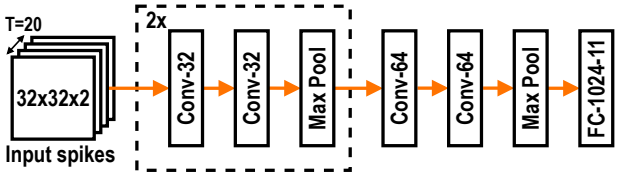


Fig. 7. SNN architecture used for the DVS gestures task.

## IV. CONCLUSION

This paper presented Mega, a digital convolutional SNN accelerator. It combines event-driven spiking convolution with low-overhead spike detection for efficient operation. To improve flexibility, Mega employs a unified memory for spikes, states, and weights. Implemented in 22 nm, Mega consumes 0.375 pJ/SOP, achieving state-of-the-art energy efficiency.


## ACKNOWLEDGMENT

This work is funded by EU's Horizon Europe research and innovation programme under grant agreement No. 101070374. The authors thank Henk Corporaal for his valuable feedback.


## REFERENCES


[1] P. Lichtsteiner *et al.*, "A 128× 128 120 db 15 $\mu$s latency asynchronous temporal contrast vision sensor," JSSC, 2008.

[2] R. A. Antonio *et al.*, "An open-source HW-SW Co-development framework enabling efficient multi-accelerator systems," ISLPED, 2025.

[3] R. Tapiador-Morales *et al.*, "Neuromorphic LIF row-by-row multiconvolution processor for FPGA," TBioCAS, 2019.

[4] J. Sommer *et al.*, "Efficient Hardware Acceleration of Sparsely Active Convolutional Spiking Neural Networks," TCADICS, 2022.

[5] C. Frenkel *et al.*, "ReckOn: A 28nm Sub-mm$^2$ Task-Agnostic Spiking Recurrent Neural Network Processor Enabling On-Chip Learning over Second-Long Timescales," ISSCC, 2022.

[6] J. Zhang *et al.*, "ANP-I: A 28nm 1.5pJ/SOP Asynchronous Spiking Neural Network Processor Enabling Sub-0.1 $\mu$J/Sample On-Chip Learning for Edge-AI Applications," ISSCC, 2023.

[7] Y. Liu *et al.*, "A 22nm 0.26nW/Synapse Spike-Driven Spiking Neural Network Processing Unit Using Time-Step-First Dataflow and Sparsity-Adaptive In-Memory Computing," ISSCC, 2024.

[8] D. Sharma *et al.*, "A 65nm 5 TOPS/W digital CIM accelerator with reconfigurable precision and temporal pipelining for spiking neural networks," ESSERC, 2025.

[9] H. Fu *et al.*, "SpikeRAM: A 48.1pW/Synapse/Bit Event-Driven Spiking Compute-Near/In-Memory Processor with Neuromorphic Sensor Enabling Life-Long On-Chip Learning," ISSCC, 2026.

[10] A. Amir *et al.*, "A low power, fully event-based gesture recognition system," CVPR, 2017.