Simulation of a recurrent, sparsely connected spiking neural network (LSM reservoir)

The paper I am using (references) implemented an LIF-equivalent spiking neuron and built a sparse recurrent Liquid State Machine. The implied computational workload is an event-driven, memory-bound SNN simulation with irregular parallelism, which maps non-trivially to GPUs.

My core computer loop, regardless of my hardware :

1)Membrane potential

2)Threshold crossing

3)Spiking / firing

4)Leaky integrate-and-fire (LIF)

The proposed quantum-tunneling neuron is shown to be functionally equivalent to an ideal LIF neuron model ie. The proposed hardware neuron is functionally equivalent to a leaky integrate-and-fire (LIF) neuron, enabling abstraction at the algorithmic level.

We will try to justify an algorithmic LIF simulation in a cleanly defined gpu workload.

This is an event driven computation in which not all neurons will be activating at every timestep, which makes the GPU mapping non-trivial and interesting.

Computationally in Liquid state machines, three things are important to us:

1) Fixed recurrent connectivity
2) Sparse Synapses
3) Continuous time or discretized updates

The hardware neuron motivates efficient- event driven LIF computation. Neuronal computation is event-driven, with sparse spiking activity leading to irregular parallelism.

For time being, I am ignoring these, they are not that imp for GPU execution model analysis:

1)Exact energy-per-spike numbers

2)capacitor values(fF)

3)Precise Fabrication details

4)Circuital blocks(comparators,I-V converters)

Simulation of a recurrent, sparsely connected spiking neural network (LSM reservoir)

The target workload is an event-driven simulation of a recurrent spiking neural network (Liquid State Machine) composed of input, reservoir and output layers.

This is where I am going to focus majorly:

1) Operation breakdown: Neuron update and spike propagation
2) Compute vs Memory : Sparse events, Memory-bound
3) Expected GPU Mapping : threads(neurons or spikes)

My diagram annotations:

1)Threads: neuron updates and spike events

2)Warp divergence :sparse firing

3)Memory bottleneck:synaptic adjacency lists

4)synchronization:spike accumulations/reductions

The workload is event-driven and spike-triggered, with sparse connectivity, lightweight per-neuron state and highly non-uniform computation dominated by irregular memory accesses during spike propagation.

The reservoir layer uses fixed, sparse recurrent synapses, resulting in irregular memory access and load imbalance during simulation.

Spiking activity is sparse and temporally irregular, leading to bursty execution patterns rather than uniform per-timestep computation.

The demonstrated equivalence to an ideal LIF neuron allows the workload to be analyzed purely at the algorithmic level, independent of device physics.

While the reservoir dynamics are fixed, output layer computation and learning are typically performed in software, motivating efficient GPU-based simulation and analysis.

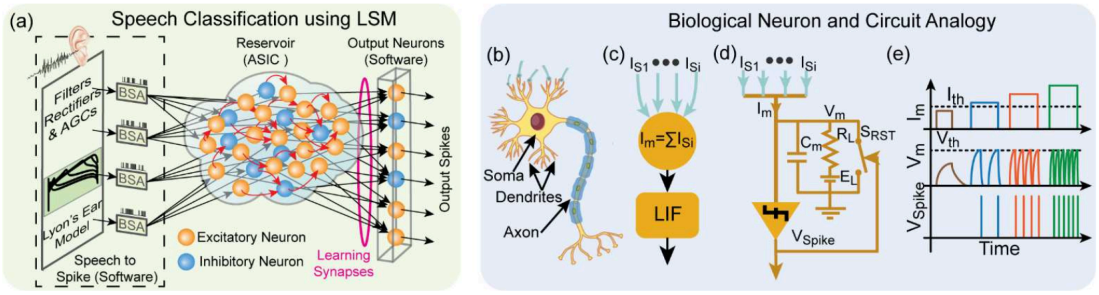


Fig. 1. **Speech classification using LSM, biological neuron structure, and circuit analogy.** (a) Implemented Spiking Neural Network used for speech classification. (b) Biological neuron mainly consists of soma, dendrites, and axon. (c) Basic neuron algorithm, i.e., summation, leaky integration, and fire. (d) Circuit equivalent of a neuron. (e) The transient waveform of a typical LIF neuron. Neuron starts spiking if $I_m > I_{th}$ and spiking frequency increases proportionally to the input current. The neuron is reset to resting potential E_L by switch S_{RST} if the membrane potential exceeds the threshold ($V_m > V_{th}$).

The workload consists of spike-encoded inputs driving a sparse, recurrent LSM reservoir, followed by a software-based readout layer.

Here, neurons are abstracted as computational units that integrate synaptic inputs and communicate via discrete spike events.

The algorithm:

For each neuron:

Im = sum(Is_i)

Vm = leak(Vm, Im)

if Vm > Vth:

emit spike

reset Vm

GPU mapping

Thread - one neuron update

Reduction - summation of synaptic inputs

Branch - spike/no-spike (divergence)

Threshold crossing triggers a reset operation.

As already said earlier and also clear from e):

Spiking activity is temporally sparse and input-dependent, resulting in irregular, event-driven computation rather than uniform per-timestep workloads.

Task 1:

Operation Breakdown

Core operations per simulation step

Each neuron independently performs:

1. Input accumulation from incoming synaptic spikes
2. Leaky integration of membrane potential

3. Threshold check
4. Spike generation and reset (if threshold crossed)
5. Spike propagation to connected post-synaptic neurons

Simplified Pseudocode:

```
for each timestep:
    for each neuron (parallel):
        Vm += sum(incoming_spikes) - leak(Vm)
        if Vm > Vth:
            emit spike
            Vm = reset
            for each connected synapse:
                enqueue spike event
```

Compute vs Memory Behavior

The workload is **memory-bound rather than compute-bound**. Per-neuron computation involves a small number of arithmetic operations for leaky integration and threshold comparison, while spike propagation requires accessing sparse synaptic data structures with irregular memory access patterns.

Due to sparse firing activity, only a small subset of neurons generate spikes at any timestep, leading to limited data reuse and poor memory coalescing during synaptic updates. As a result, overall performance is dominated by global memory bandwidth and access latency rather than floating-point throughput.

Parallelism

Parallelism arises at two levels:

- **Neuron-level parallelism**, where each neuron's LIF state update can be computed independently.
- **Synapse-level parallelism**, where spike events are propagated in parallel to multiple post-synaptic neurons.
- **Event-level parallelism**: multiple spikes can be processed concurrently

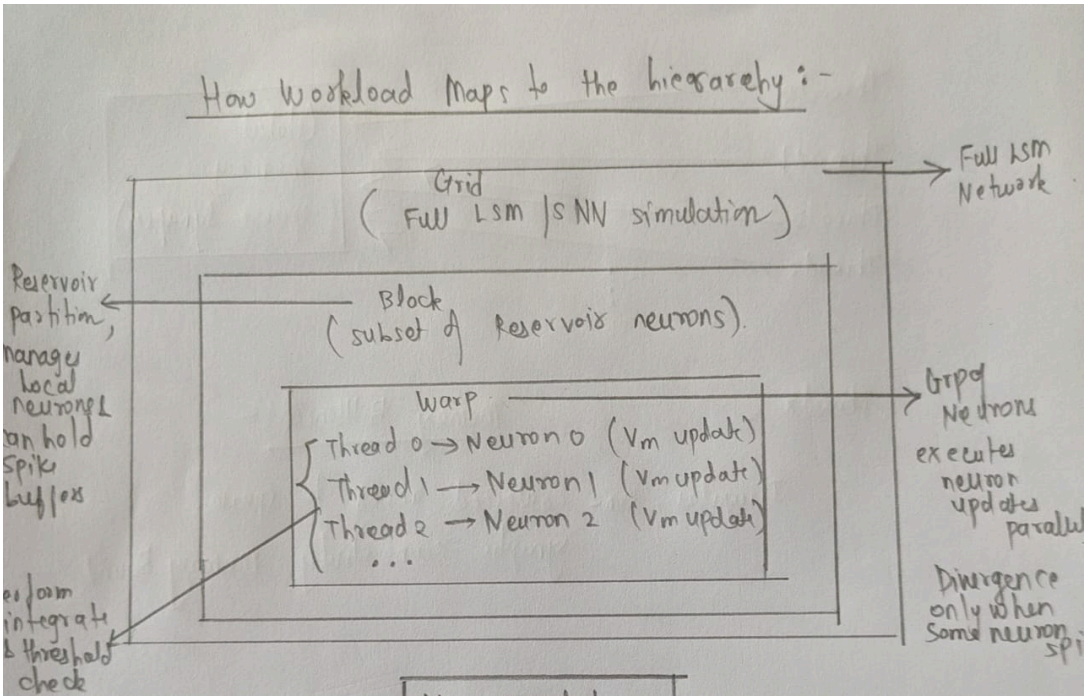
However, this parallelism is highly irregular, as the number of active neurons and synapses varies dynamically with time and input activity.

GPU Mapping and Potential Issues

On a GPU, threads can be mapped to neurons during the neuron state update phase and to individual spike events during the spike propagation phase. Thread blocks correspond to subsets of neurons within the reservoir, while warps process groups of neurons concurrently.

Sparse and input-dependent spiking leads to warp divergence, as only a fraction of threads execute spike-related code paths. Additionally, irregular synaptic fan-out results in load imbalance and non-coalesced memory accesses, making the workload predominantly memory-bound.

Task 2: CUDA Execution Model Diagram



Mapping

Thread - neuron state update or spike event

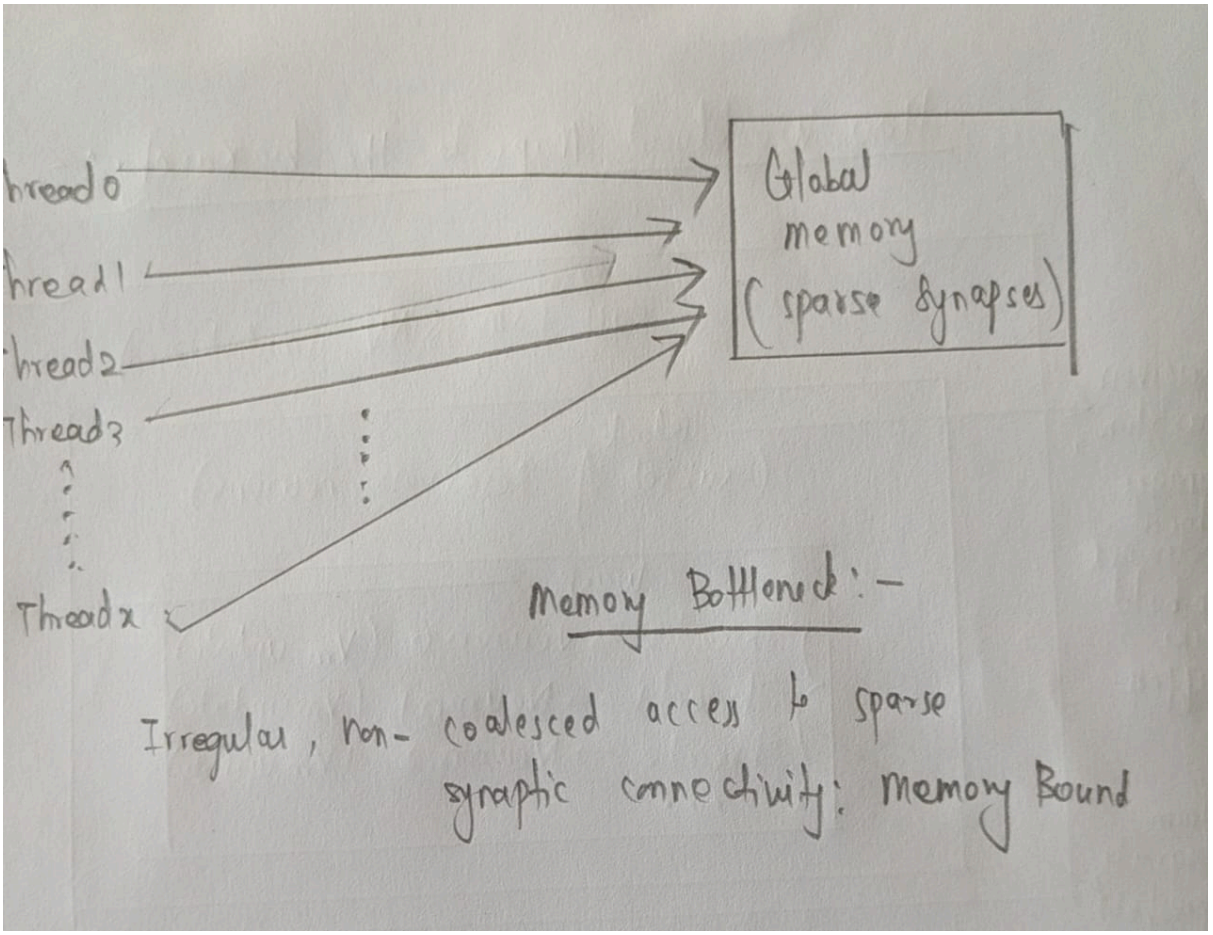
Warp - group of neurons

Block - subset of reservoir neurons

Bottlenecks

Sparse and irregular synaptic memory accesses

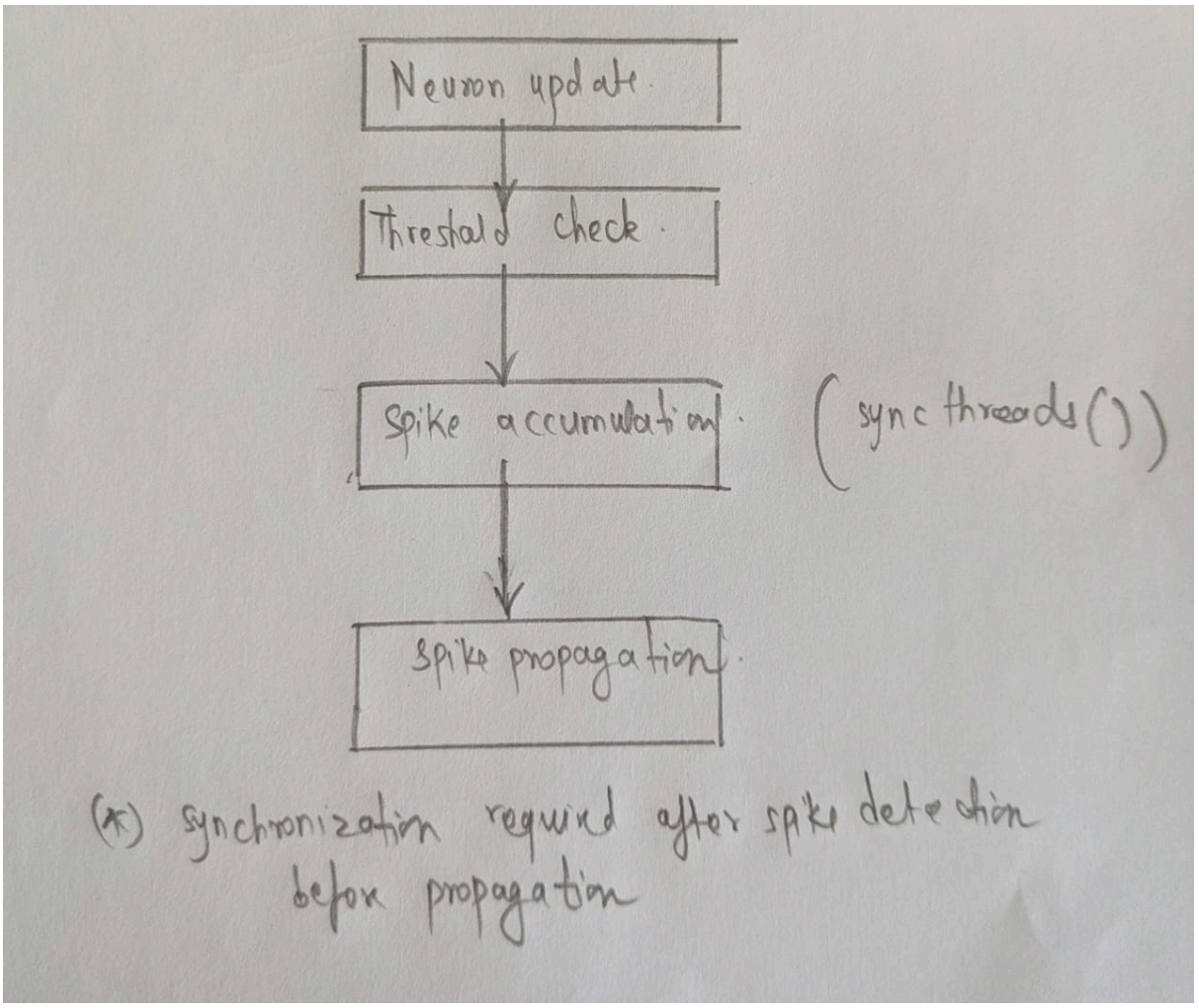
Branch divergence at the threshold comparison



Synchronization

Local synchronization may be required within a block or warp during spike accumulation or reduction.

Global synchronization across all neurons is typically avoided due to event-driven execution.



Note: Shared memory could be used for temporary spike accumulation within a block before writing membrane updates to global memory, though sparsity limits reuse

Neuron update rule:

Each neuron computes the sum of incoming synaptic currents, performs leaky integration to update its membrane potential and generates a spike followed by a reset when a fixed threshold is exceeded.

Spike propagation semantics:

The generated spike travels through the axon and other connected neurons are communicated about the occurred event.Spike propagation is event-driven and only occurs for firing neurons, resulting in sparse and irregular synaptic updates.

The hardware neuron is designed to efficiently implement leaky integration and thresholding with ultra-low power, motivating large-scale neuromorphic systems.

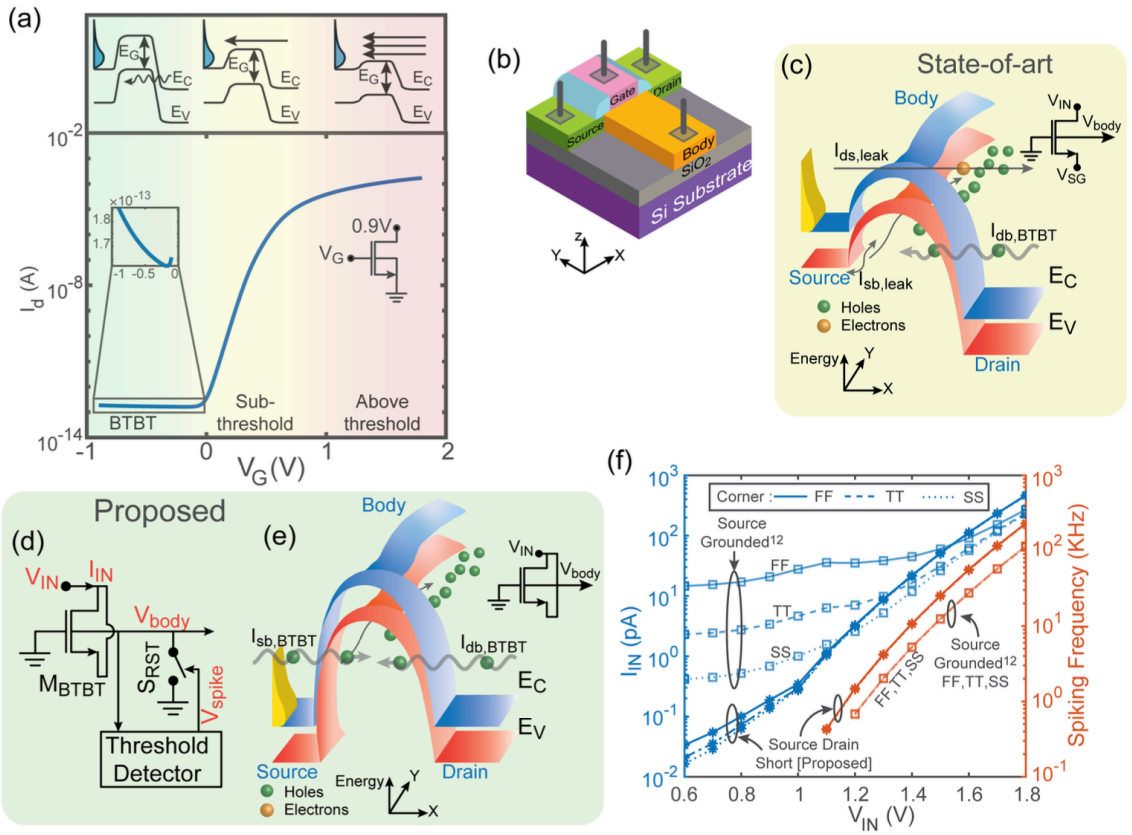


Fig. 2. **BTBT operation and motivation.** (a) BTBT operates at an extremely low current ensuring power saving. (b) PDSOI MOSFET architecture with body contact. (c) Band diagram of PDSOI with large drain bias applied, resulting in BTBT operation. (d) Proposed neuron circuit using BTBT in PDSOI MOSFET as the leaky integrator. Drain and source shorted to enable BTBT at both terminals. (e) Band diagram of PDSOI with a large bias applied to drain and source shorted, resulting in BTBT current from source as well as from drain to the body while eliminating the source to drain leakage. (f) Power and dynamic range improvement by proposed integrator using M_{BTBT}.

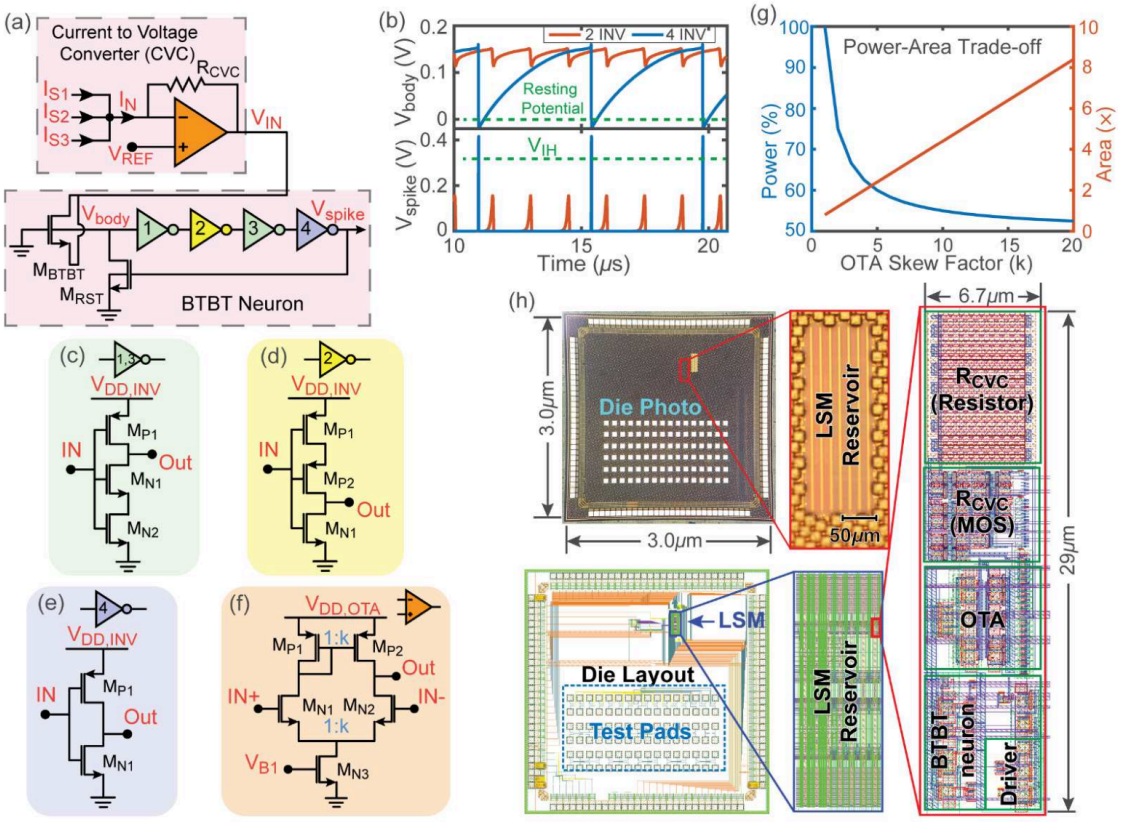
The proposed hardware enables ultra-low-current leaky integration, motivating energy-efficient large-scale spiking neural networks.

The neuron dynamics consist of current integration followed by threshold detection and reset, matching the standard LIF update rule used in spiking neural network simulations.

Spiking frequency varies with input strength, resulting in temporally bursty and input-dependent execution.

Spiking activity is input-dependent, with higher input currents producing higher spike rates, leading to temporally bursty execution.

Spiking activity is input-dependent and temporally irregular, with bursty spike generation once neuronal thresholds are crossed.Since most neurons remain inactive for long durations, effective GPU utilization is limited by sparse activation and event-driven execution.



Circuit-level optimizations primarily affect power and robustness, while the algorithmic LIF dynamics and event-driven execution model remain unchanged.

Hardware linearization techniques enable BTBT neurons to closely approximate ideal LIF behavior, allowing GPU simulations to safely assume ideal LIF dynamics without changing the event-driven execution structure.

Threads map to neurons, blocks to reservoir partitions and spike-driven execution causes warp divergence and memory-bound behavior due to sparse synaptic access.

References:

1)NVIDIA CUDA Programming Guide — Chapter 1
https://docs.nvidia.com/cuda/cuda-c-programming-guide/

2)Quantum Tunnelling based Ultra-compact and Energy Efficient Spiking Neuron enables Hardware Liquid State Machine-IITB Published paper