Prosperity: An RTL Accelerator Exploiting Product Sparsity in Spiking Neural Networks

Shri Vishakh Devanand
Veermata Jijabai Technological Institute
shrivishakhdevanand@gmail.com

Gopalkrishnan Srinivasan
RISE Lab, IIT Madras
sgopal@cse.iitm.ac.in

***Abstract*—This report summarises my internship designing *Prosperity*, a from-scratch RTL accelerator for Spiking Neural Networks (SNNs). The architecture exploits *product sparsity*—the intersection of sparse spike activations and pruned synaptic weights—to reduce compute demand. A novel three-stage pipeline (*Detector–Pruner–Dispatcher*) implemented in synthesizable Verilog attains the theoretical performance target of 4 cycles per row. A comprehensive verification framework confirms functional correctness, timing closure, and scalability to 256-neuron layers.**

***Index Terms*—Spiking Neural Networks, Neuromorphic Hardware, RTL Design, Product Sparsity, Verification, FPGA Prototyping**

I. Introduction

Neuromorphic computing promises ultra-low-power inference by mimicking the brain's event-driven information flow. However, practical deployment demands hardware that capitalises on the inherent sparsity of SNN workloads. **Prosperity** addresses this gap by detecting and eliminating zero-work multiply–accumulates (MACs) at run-time, thereby accelerating inference without accuracy loss.

II. Literature Review and Background

A. Spiking Neural Networks

SNN neurons integrate discrete spikes according to

$$V(t+1) = \alpha V(t) + \sum_i w_i s_i(t),$$

firing only when the membrane potential exceeds a threshold. Such event-driven behaviour yields natural *activation sparsity* ($\sim 5{-}15\%$ firing rate).

B. Product Sparsity

Combining activation sparsity with *weight sparsity* ($70{-}90\%$ pruned weights) produces *product sparsity*. Hardware needs only evaluate

$$\texttt{Active\_MAC} = \texttt{Spike\_Vector} \wedge \texttt{Weight\_Mask}.$$

C. Prosperity Paper Target

The original Prosperity paper specifies a performance goal of four pipeline cycles per SNN row but omits RTL details. This internship delivers a complete hardware realisation meeting that target.

III. Architecture and Implementation

A. Three-Stage Pipeline

Detector: bitwise product detection on 256-bit spike and mask vectors.
Pruner: filters low-activity results via configurable thresholds.
Dispatcher: schedules surviving MACs into a parallel array while preserving temporal order.

B. Sparse Data Handling Innovations

- 256-bit popcount unit for spike reduction
- Parallel address generation for active weights
- Stall-free handshaking tolerant of bursty activity

C. Memory Hierarchy

A three-level FIFO/cache stack balances spike bandwidth against BRAM footprint, dynamically allocating resources per activity window.

IV. Verification Methodology

A. Test-Bench Strategy

SystemVerilog/Cocotb test-benches supply:

- Unit tests for all RTL modules
- Integration tests for Detector–Pruner–Dispatcher flow
- Randomised edge-case generators (empty input, overflow)

B. Coverage Metrics

100% functional coverage for sparsity patterns, thresholds, and dispatcher queues; 96% toggle coverage across pipeline registers.

V. Results and Analysis

A. Performance

Measured using Verilator:

- **4 cycles per row** sustained on 256-neuron layers

VI. Innovation and Contributions

1) First RTL accelerator to exploit *product sparsity* at run-time
2) Configurable three-stage pipeline scalable to arbitrary layer sizes
3) Verification harness combining unit, integration, and random testing

VII. Future Work

While the core RTL pipeline achieves theoretical throughput, further steps are needed to validate real-world applicability:

- **Model-Level Verification:** Integrate the accelerator into full SNN inference flows (e.g., MNIST, DVS Gesture datasets) and compare outputs against software baselines.
- **Benchmarking:** Evaluate performance and energy efficiency across diverse SNN architectures (e.g., feedforward, recurrent, convolutional).
- **FPGA Prototyping:** Deploy on FPGA to measure real timing, area, and power tradeoffs under realistic workloads.
- **ISA/Compiler Co-Design:** Explore instruction extensions or compilation techniques to better feed sparse data to the accelerator.

VIII. Conclusion

This internship translated the abstract concept of product sparsity into a fully synthesizable RTL design. The Prosperity accelerator meets the theoretical performance target of four cycles per SNN row and is validated through rigorous simulation and verification. Although not yet deployed on physical hardware, the architecture demonstrates scalability and correctness through comprehensive testbenches. These design and verification artefacts provide a robust foundation for future work in neuromorphic accelerator development and real-world deployment.

References

[1] Y. Liu *et al.*, "Prosperity: Product-Sparsity Accelerator for Spiking Neural Networks," *arXiv preprint* arXiv:2503.03379, 2025. [Online]. Available: https://arxiv.org/pdf/2503.03379

[2] S. V. Devanand, "Spiking Neural Networks: A Complete Guide," 2025. [Online]. Available: https://blog.sravjti.in/2025/07/14/spikingnns.html

[3] A. Author, "The Complete Guide to Spiking Neural Networks," Toward AI, 2024. [Online]. Available: https://pub.towardsai.net/the-complete-guide-to-spiking-neural-networks-d0a85fa6a64

[4] Waterloo AI Lab, "Spiking Neural Networks Explained," YouTube video, 2024. [Online]. Available: https://www.youtube.com/watch?v=PeW-TN3P1hk

[5] D. Shorak, "Spiking Neural Networks: The Next Big Thing in AI," Medium, 2023. [Online]. Available: https://medium.com/@deanshorak/spiking-neural-networks-the-next-big-thing-in-ai-efe3310709b0