Hardware Specialization Proposal: Softmax Accelerator

ELEC 5803

Jesse Levine (101185127)

1 Specialization Target: Softmax Kernel

The target hardware specialization is the Softmax activation function, the standard final layer for multi-class classification networks. Mathematically, for an input vector **x** of length N, the output $\mathbf{y_i}$ is defined as:

$$\mathbf{y_i} = \frac{e^{x_i}}{\sum_{j=1}^{N} e^{x_j}}$$

This requires three distinct computational phases:

1. **Vector Exponentiation:** Calculating e^x for every element in the input vector.
2. **Global Summation:** A reduction operation to computer the sum of all exponential results.
3. **Normalization:** An element-wise division of each exponent by the global sum.

Performance bottlenecks related to the Softmax activation function are further detailed in Section 2.

Furthermore, this project will use a CNN tested on the MNIST (10-element vector) dataset for validation, although the hardware will be designed and meant as a generic vector-processing engine for any Softmax workload.

2 Target Performance Bottleneck

The primary bottleneck addressed by this specialization is computational latency caused by transcendental function emulation.

Transcendental evaluations (e^x) and normalization steps are difficult to parallelize in general-purpose pipelines [1]. On an integer-based RISC-V core, these would incur considerable latency and hardware costs. Further, the bottleneck is not static as the computational load of the Softmax kernel scales with model dimensionality and sequence length, making it a critical point of failure for real-time infrence as network complexity grows [1].

3 Architectural Approach

The hardware specialization will be implemented as a memory-mapped hardware accelerator.

To overcome the high latency associated with standard transcendental math libraries, the architecture utilizes a hybrid approximation strategy combining Piecewise Linear Approximation (PLA) with Lookup Tables (LUTs). In this design, the input range of the exponential function is divided into several segments. The specific coefficients for each segment—representing the slope and intercept of the linear model—are stored in a small, efficient LUT. This hybrid approach replaces iterative Taylor series expansions or massive, high-capacity LUTs with basic fixed-point multiplications and additions. By storing coefficients for the linear equation $y = mx+b$ in a LUT, the hardware can maintain high numerical precision across a wide dynamic range while significantly reducing the computational cycles required for each element.

The internal logic features a pipelined reduction tree designed to calculate the global summation of exponents in parallel. This configuration allows the accelerator to process input vectors with high throughput, shifting the system bottleneck from the CPU's mathematical execution limits to the available memory bus bandwidth. The accelerator is positioned as a peripheral on the system bus, enabling a decoupled execution model where the RISC-V core manages high-level control flow and layer sequencing while the specialized IP block handles the intensive Softmax kernel in the background. This resource-efficient design ensures that the hardware footprint remains small enough for FPGA deployment by utilizing DSP slices for math and minimal BRAM for coefficient storage, providing a deterministic speedup over general-purpose software emulation.

4 Hardware-Software Interface

The RISC-V core functions as the primary orchestrator for the neural network inference, managing the overall control flow and executing non-specialized layers in software. To accelerate the final stage of inference, the core interacts with the Softmax hardware through a memory-mapped I/O (MMIO) interface. This integration allows the CPU to offload the mathematically intensive Softmax kernel to the specialized hardware block while maintaining control over the network's sequencing.

Essentially, the operation flow is as follows: the inference process follows a strict hardware-software handshake. First, the RISC-V core executes the preceding layers of the neural network and writes the resulting logits into a shared BRAM region. Once the data is staged, the core programs the accelerator's configuration registers and asserts the START signal. At this point, the accelerator takes master control of the local bus, streams the data through its hybrid PLA-LUT pipeline, and overwrites the shared memory with the normalized probability distribution. The core, having polled the DONE flag, then performs the final classification step (e.g., argmax). This partitioning ensures that the core is only responsible for high-level logic and simple arithmetic, while the specialized hardware handles the transcendental bottleneck.

Figure 1 below showcases the system-level architecture:

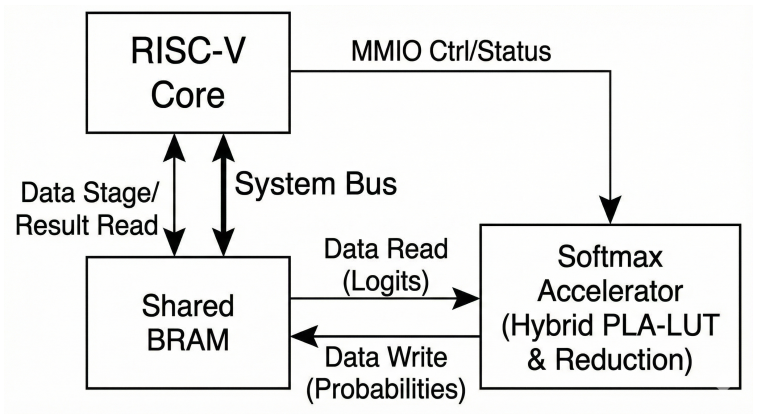

Figure 1: System-Level Block Diagram

5 AI Usage

Gemini was used to write the report and to discuss hardware implementation, e.g. only using PLA, PLA vs LUT approach, hybrid PLA + LUT. ChatGPT was used to explore how the SoC will operate in a high-level sense: NN inference on core, softmax accelerator is separate hardware block that the core invokes while control remains with the core.

References

[1] R. Kim, D. Lee, J. Kim, J. Park, and S. E. Lee, “Hardware accelerator for approximation-based softmax and layer normalization in transformers,” *Electronics*, vol. 14, no. 12, 2025. [Online]. Available: https://www.mdpi.com/2079-9292/14/12/2337