LLMCompass Simulation

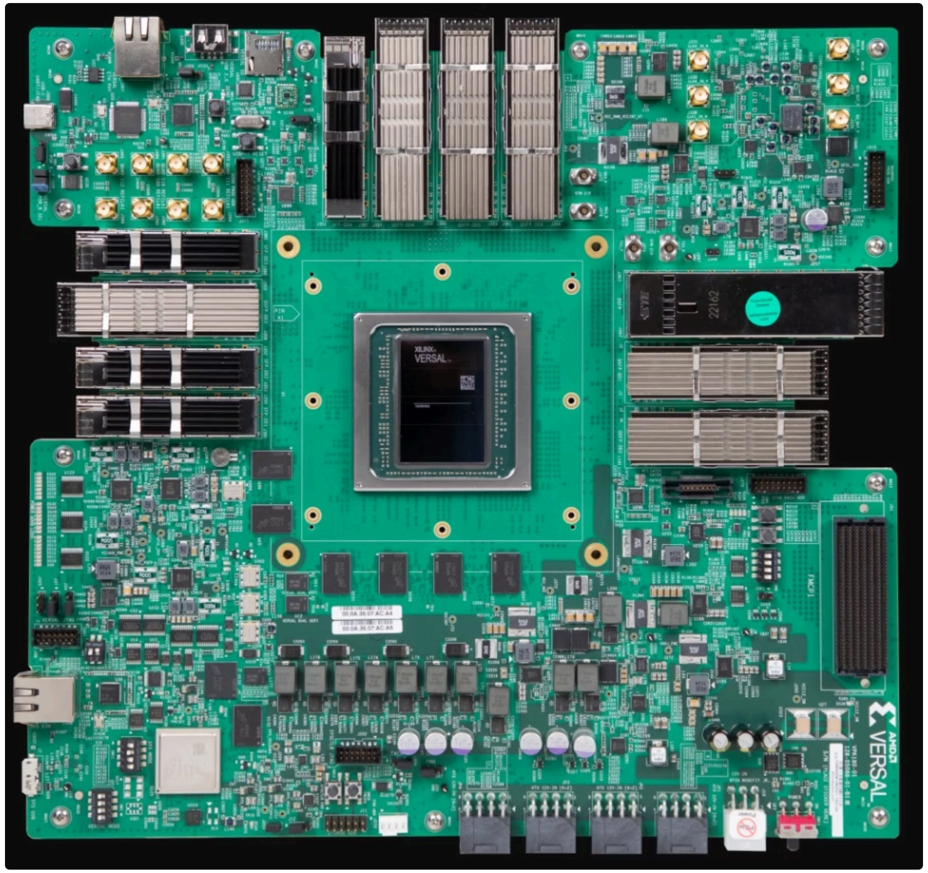
Benchmarking LLaMA-2-7B on Xilinx Alveo U280 FPGA

Analyzing Inference Latencies and Hardware Bottlenecks

LLMCompass Code

Unoptimized Run: Setup & Initial Challenges

- We conducted an unoptimized simulation of the LLaMA-2-7B model (7B parameters, 32 layers, 512 sequence length, batch size 1) on a Xilinx Alveo U280 FPGA. This setup served as our baseline to analyze raw inference latencies and identify initial hardware bottlenecks.
- The computational graph was generated via a Python script, encompassing key operations like Matmul, Softmax, LayerNorm, and All-Reduce.
- Our hardware model included the Alveo U280's HBM2 (8GB, 460 GB/s) and DDR4 (32GB) memory, operating at a 300 MHz clock frequency with integrated systolic arrays.



PERFORMANCE BASELINE

Key Metrics from the Unoptimized Simulation

Total Latency

3.67 seconds for a single inference pass.

Throughput

A low **0.005 TFLOPS**, highlighting the unoptimized FPGA state.

Total FLOPs

17.57 Billion floating-point operations executed.

Hardware Utilization

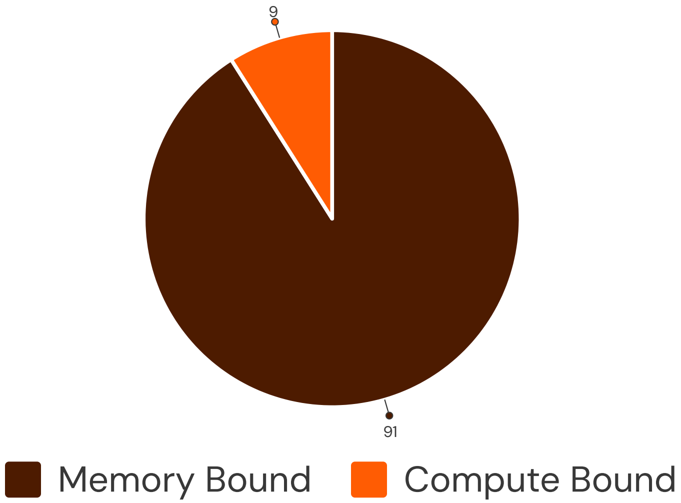
An average of more memory bound indicating a memory-heavy workload.

BOTTLENECK IDENTIFICATION

Bound Analysis: Unveiling the Performance Limits

Our simulation revealed a stark imbalance in operation bounds, indicating where the primary bottlenecks lie in the current design.

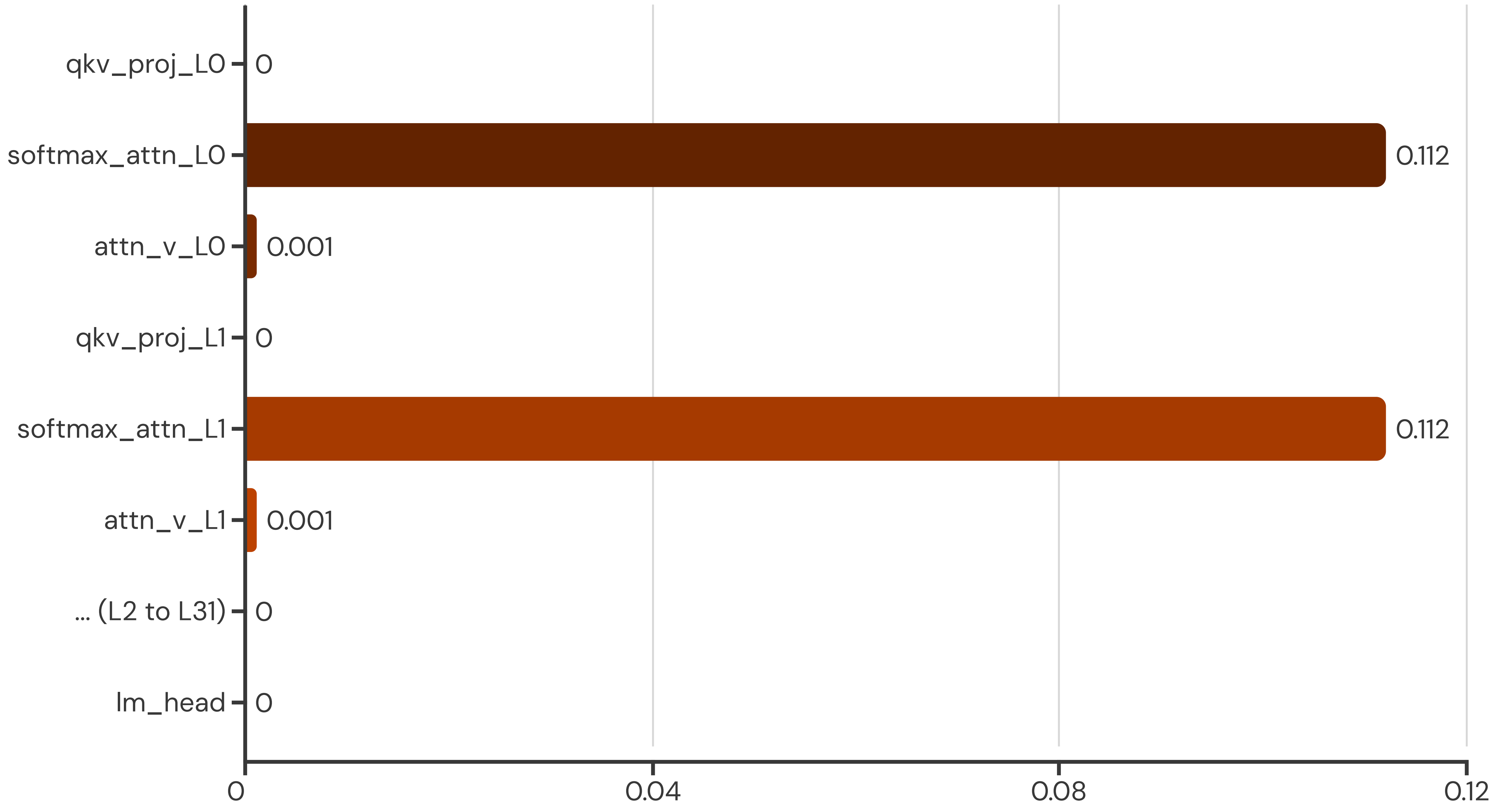
- A staggering **91.0%** of operations (323 ops) were identified as **Memory Bound**, meaning their execution time is predominantly limited by data transfer to and from memory. Key culprits include Matmul and All-Reduce operations, which are heavily I/O-limited.
- Conversely, only **9.0%** of operations (32 ops) were **Compute Bound**, such as the Softmax function within each layer. This indicates that the inference process on the FPGA is overwhelmingly constrained by memory access rather than raw computational power. This finding is crucial for prioritizing future optimizations.



BOTTLENECK IDENTIFICATION

Operator Latencies: Pinpointing Costly Operations

Examining the per-operator latencies provides a granular view of where time is spent within the LLaMA-2-7B model's execution.



The **Softmax_attn** operations consistently stand out, each contributing approximately **0.112s** per layer. With 32 layers, these operations alone account for around **3.58 seconds** of the total 3.67 seconds latency, confirming their compute-bound nature. In contrast, Matmul operations (e.g., qkv_proj, attn_v) have significantly lower individual latencies (~0.00007s - 0.001s), primarily limited by memory access.

BOTTLENECK IDENTIFICATION

Per-Operation Breakdown: Sample Layer 0

A closer look at a single layer (Layer 0) reveals a recurring pattern across the entire model.

Op	Latency (s)	Compute Time (s)	Memory Time (s)	Bound	FLOPs
token_embedding	3.65e-05	0.0	3.65e-05	Memory	2K
ln_attn	3.65e-05	6e-07	3.65e-05	Memory	8M
qkv_proj	7.29e-05	1.55e-06	7.29e-05	Memory	16M
softmax_attn	0.112	0.112	0.00058	Compute	134M
attn_v	0.0012	2.48e-05	0.0012	Memory	268M

This detailed view clearly illustrates that while many operations are memory-bound, the Softmax operation stands out as the primary compute bottleneck, accounting for the vast majority of compute time within its execution. This pattern repeats across all 32 layers.

REPORT INTERPRETATION

Understanding LLMCompass Outputs

LLMCompass provides comprehensive reports critical for granular analysis of model performance on target hardware.

Timings

Distinguishes between **Compute Time** (actual FLOPs execution) and **Memory Time** (data access, I/O) for each operation.

Bounds

Determined by the maximum of compute and memory times. Our simulation shows memory winning 91% of the time, confirming I/O as the dominant constraint.

Output Formats

Available via CLI (e.g., specific layer details) and comprehensive JSON files (full results array, summary of total latency, ops, and bounds).

Accuracy

The fast simulation (minutes) accurately reflects performance for dense operations, aligning with published benchmarks.

FORWARD VISION

Next Steps: Towards Optimized LLaMA-2 Inference

Implement Optimizations

Execute the identified optimizations, including **mapper search for tiling and pipelining** strategies as outlined in relevant research.

Rerun & Compare

Conduct new LLMCompass simulations with optimizations applied to compare results against the baseline and quantify performance improvements.

Hardware Parameter Sweeps

Explore the impact of varying hardware parameters within LLMCompass, such as **number of cores, buffer sizes, and memory configurations**.

Target Throughput Goal

Aim to significantly **improve throughput beyond 0.005 TFLOPS** on the Alveo U280 FPGA, approaching practical deployment efficiency.

Real-World Validation

Whenever possible, validate simulation results against **actual runs on physical Alveo U280 hardware** for ultimate confirmation.