IDEC Chip Design Contest

Weight-Only Quantized LLM Acceleration with Analog In-Memory Computing Unit

Jaeyong Jang, Wonkyung Han and Jae-Joon Kim
Seoul National University, Korea

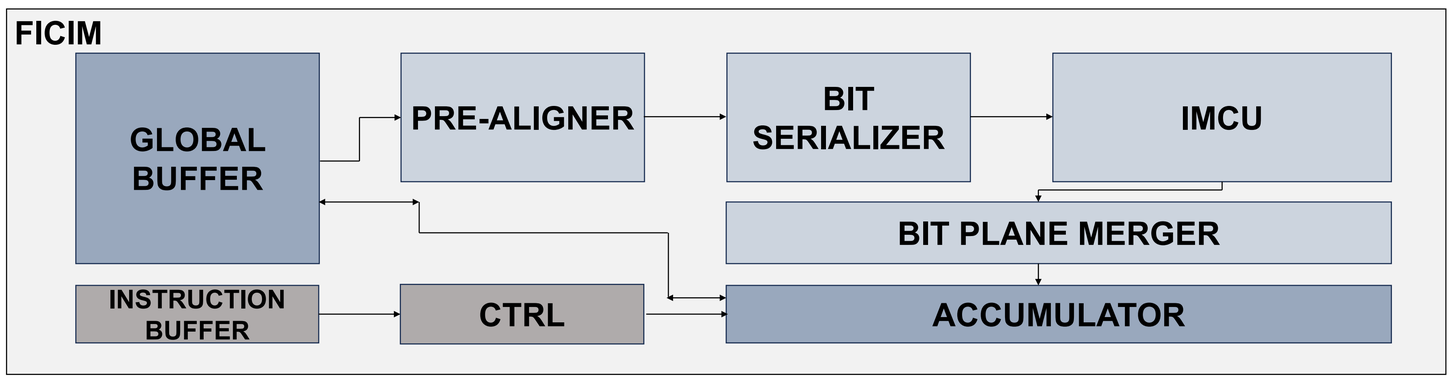
Motivation & Opportunity

Large language models (LLMs) are increasingly deployed in latency- and energy-constrained environments, but their inference cost continues to grow with model size, hidden dimension, and context length. Quantization is one of the most practical approaches for reducing this cost. However, activation quantization remains challenging because activation distributions vary significantly across layers, tokens, and input prompts. Nonlinear operators such as softmax, normalization, and gating functions further amplify this variation, making aggressive activation quantization more likely to degrade model accuracy.

In contrast, weights are static after training and can be characterized offline. For this reason, many practical LLM deployments adopt weight-only quantization, where weights are stored in low-bit integer formats while activations remain in floating point. This reduces memory footprint and bandwidth pressure, but it does not automatically improve compute efficiency. In many conventional runtimes, quantized weights are dequantized back to floating point before matrix multiplication, so the dominant arithmetic still becomes FP×FP computation.

This work targets the natural mixed-precision operation created by weight-only quantization: floating-point activations multiplied by low-bit integer weights. Instead of converting INT weights back to FP values, **FICIM** reshapes the computation so that FP×INT multiplication can be executed as IMCU-friendly INT×INT operations. The goal is to translate the memory-side advantage of weight-only quantization into compute-side energy efficiency.

FICIM Overview

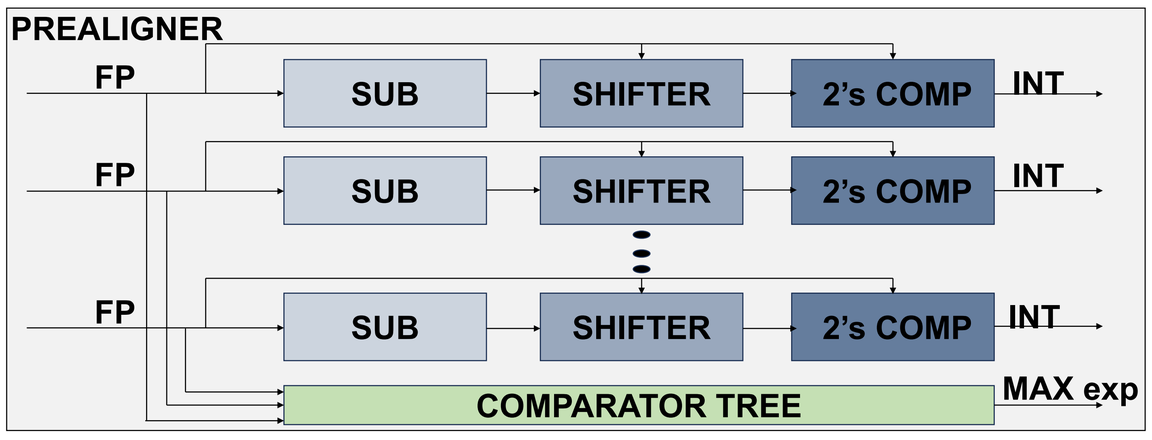


FICIM is organized as a heterogeneous pipeline composed of a global buffer, **prealigner**, bit serializer, analog IMCU, bit-plane merger, accumulator, controller, instruction buffer, and SFU. The global buffer stores activations, quantized weights, and partial sums for the current layer or tile. The **prealigner** converts FP activations into integer-compatible values. The bit serializer transforms these values into activation bit planes. The IMCU computes partial dot products using bit-sliced weights, and the bit-plane merger reconstructs the result by combining temporal activation planes and spatial weight slices. The accumulator aggregates partial sums across tiles, while the SFU handles non-MVM operations such as activation and normalization-related functions.

Prealigner

The **prealigner** bridges floating-point activations and integer-domain IMCU execution. For each FP activation tile, it scans exponent fields, finds the maximum exponent, and shifts each mantissa according to the exponent difference so that all values share a common scale.

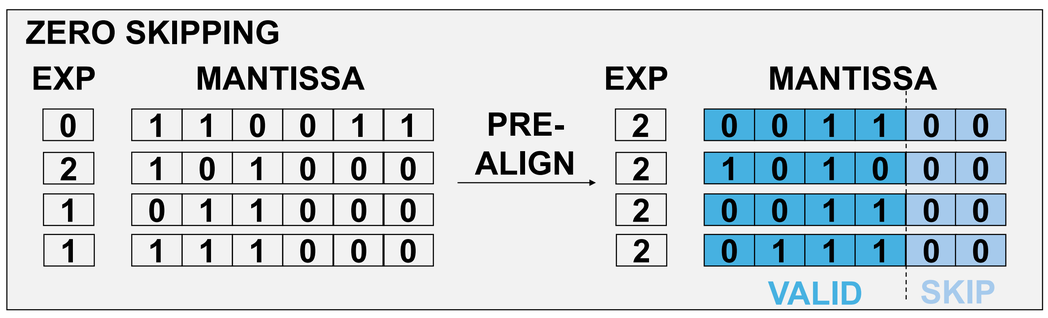
After prealignment, FP activations are represented as uniformly scaled fixed-point integers. Unlike naïve activation quantization, this conversion preserves the effective scaling behavior of FP arithmetic, while guard bits help reduce accuracy loss from truncated least significant bits.



ZERO SKIPPING

Prealignment often creates bit planes that contain no useful information. For example, some least significant bit positions may become zero for every activation in the current tile. If such an all-zero bit plane is sent to the IMCU, the analog MVM result is also zero, so the corresponding IMCU pass wastes cycle time and energy.

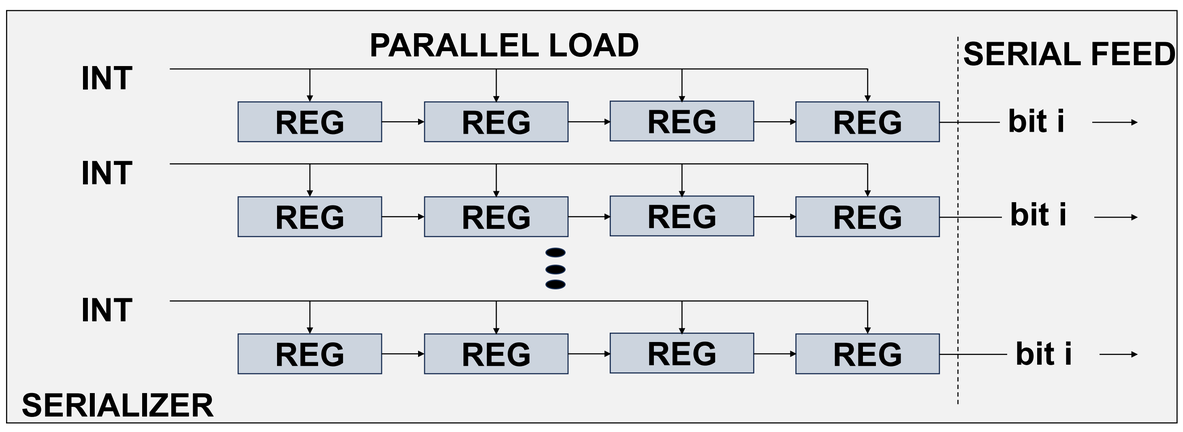
FICIM adds input bit-plane skipping to remove this redundant work. Before transmission, the serializer checks each activation bit plane. If every element in the bit plane is zero, the serializer skips the IMCU pass and records the skipped bit position. The **bit-plane merger** later uses this metadata to apply the correct shift-and-add scaling, so the reconstructed result remains numerically consistent.



Bit Serializer

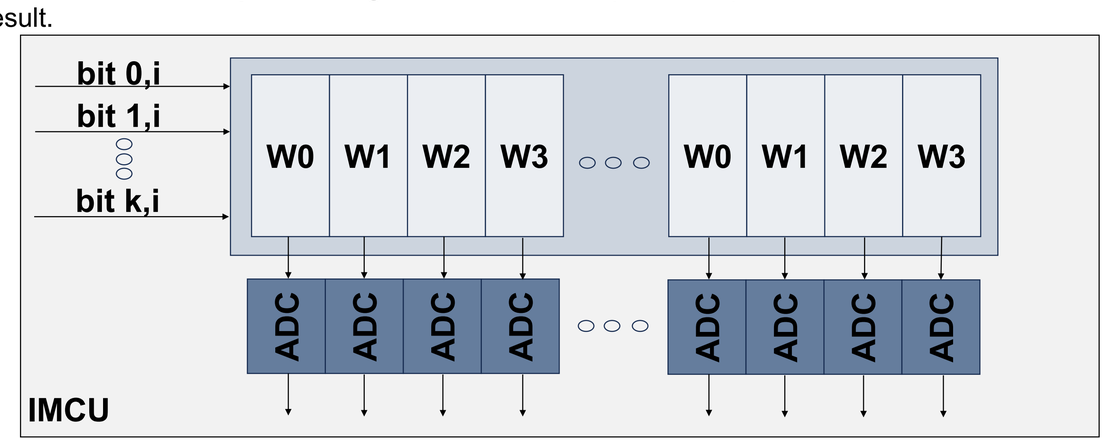
The **bit serializer** converts prealigned integer activations into bit-plane streams for the IMCU. Since the analog IMCU receives one activation bit plane per pass, each multi-bit activation vector is decomposed into single-bit planes.

This bit-serial format reduces input driver complexity and matches the natural operating style of CIM arrays. Each transmitted bit plane produces a partial MAC result, which is later reconstructed by the **bit-plane merger**.



In-Memory Computing Unit (IMCU)

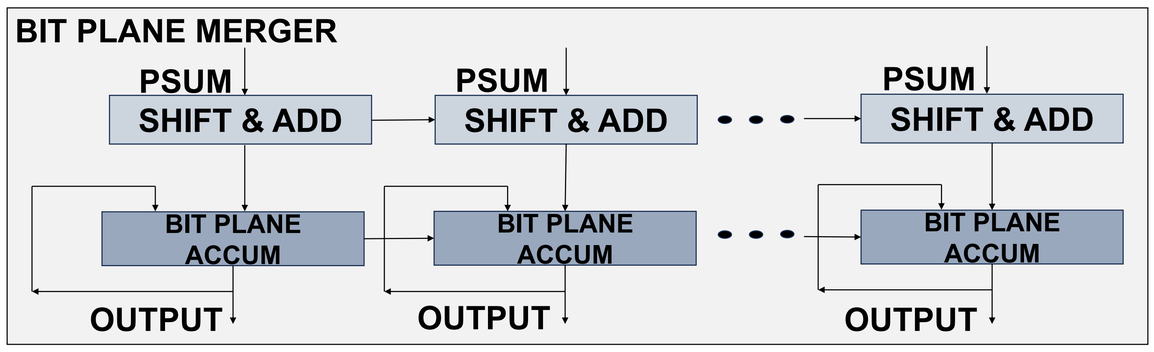
The **IMCU** performs the throughput-critical matrix-vector multiplication. Weights are mapped into the array in a bit-sliced format, and adjacent columns represent different weight bit positions or output slices. During each pass, the IMCU receives one activation bit plane from the serializer. The input bit drives the array as an analog signal, and the stored weight bits modulate the bitline response. Column-wise analog accumulation produces partial MAC outputs, which are digitized by ADCs. Since each pass corresponds to one activation bit plane, the IMCU output is only a partial result. The **bit-plane merger** combines these partial outputs to reconstruct the full MVM result.



Bit-Plane Merger

The **bit-plane merger** reconstructs the dot-product result from IMCU partial outputs. Activations are bit-sliced over time, while weights are bit-sliced across IMCU columns. Therefore, the merger performs both temporal merging for activation bit planes and spatial merging for weight bit slices.

For weight slices, outputs from different columns are combined using shift-and-add logic according to the weight bit significance. For activation planes, outputs from successive IMCU passes are shifted according to their original bit positions and accumulated. When bit-plane skipping is used, the merger refers to the recorded skip mask so that skipped planes are treated as zero-valued contributions rather than changing the numerical scale.



Conclusion

FICIM accelerates FP×INT matrix multiplication in weight-only quantized LLM inference by converting FP activations into prealigned integer values and executing them on an analog IMCU. Bit-serial input delivery and bit-plane merging reconstruct the original computation, while input bit-plane skipping removes redundant IMCU passes. Overall, **FICIM** shows that weight-only quantization can be used not only to reduce memory traffic, but also to improve compute-side energy efficiency.

Acknowledgement: Chip fabrication and EDA tool support were provided by IDEC, Korea.

반도체설계교육센터 IC DESIGN EDUCATION CENTER