Mixed-Precision Matrix Multiplication on FPGA: Architecture, Measurement, and Trade-offs

Ashwin Sabani
Email: ashwin@example.com

***Abstract*—We present a compact, parameterizable mixed-precision GEMM pipeline that targets INT8, FP16, and FP32, implemented in SystemVerilog and instrumented with a hardware-aware results suite. The design exposes precision at the processing element and composes under a controller for matrix multiply with on-chip BRAM. We evaluate accuracy and throughput using the experiment logs included in this repository. On 8×8 matrices, FP16 tracks FP32 accuracy for well-conditioned inputs while INT8 requires explicit scaling to avoid large bias. We report simulated throughput, error statistics, and range utilization, and discuss how to extend the design toward a RISC-V accelerator or larger systolic arrays.**

I. Introduction

Matrix multiplication dominates modern ML and scientific codes. The choice of numeric format drives the balance among accuracy, bandwidth, area, and power. The community often drops from FP32 to FP16 or INT8 for inference, but the reasons span more than the arithmetic units; storage and memory movement matter as much.

Our goal is a clear mixed-precision GEMM pipeline whose precision can be selected without touching control logic. The design is written in SystemVerilog, uses BRAM-backed matrices, and reports both numerical and hardware-aware metrics. This paper documents the architecture and presents results drawn directly from the CSVs and plots included with this project.

A. Background

RISC-V INT8. The RISC-V vector and packed-SIMD extensions provide integer dot products that execute multiple INT8 multiplies and accumulations per instruction. INT8 reduces traffic fourfold versus FP32 and increases arithmetic density, which is critical for bandwidth-bound inference. Correct range management and accumulation width are essential to avoid saturation and sign errors [1].

FP16 on RISC-V. The Zfh extension introduces IEEE 754 half precision [2], [3]. FP16 halves storage versus FP32 and often doubles effective register bandwidth. Many hardware implementations flush subnormals and simplify rounding to keep latency low; this can affect ill-conditioned problems. With values scaled into range, FP16 typically tracks FP32 closely on common workloads.

Why mixed precision. No single format is best across layers, matrices, or devices. A pipeline that can be built for INT8, FP16, or FP32 lets a system choose the format that fits accuracy and energy targets for each stage.

II. Architecture

The pipeline composes a processing element (PE) into a controller-driven GEMM with row-major BRAM interfaces. Precision is a parameter that flows through the PE and can be selected at build time.

Precision definitions. The package `mp_types.sv` defines the precision enum with `PREC_INT8`, `PREC_FP16`, and `PREC_FP32`, and allows an override via a preprocessor define. INT8 data width is 8 bits with a 32 bit accumulator; FP16 and FP32 follow IEEE 754.

Processing element. The PE computes acc $\leftarrow$ acc $+ a \cdot b$: INT8 uses a signed 8 bit multiply widened to 16 bit, accumulated into 32 bit; FP16 and FP32 paths wrap multiply and add units and form a two-stage pipeline.

Controller. A state machine streams A and B from BRAM, performs K MACs per output element, and writes back C. Indices are row-major, and a debug port can read back C.

Array shell. A minimal systolic shell is provided for scaling to $N \times N$ tiles, though the measurements here use the single-PE controller for clarity.

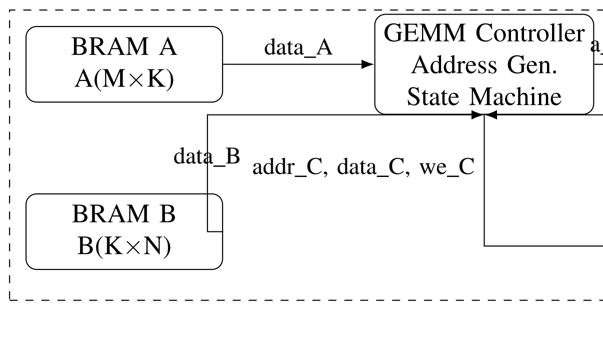

Fig. 1: Block diagram of the mixed-precision GEMM pipeline showing BRAM-backed operands, controller, and the precision-parameterized processing element.

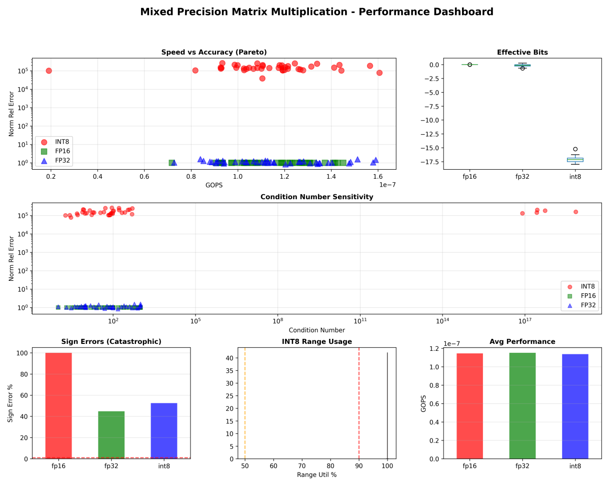

Fig. 2: Dashboard plot from this repo summarizing error and performance metrics for a representative run.

III. Methods

Data. Inputs are synthetic 8×8 matrices with recorded requested and measured condition numbers for A and B. Values are bounded to small ranges to keep FP16 within dynamic range without special handling.

Simulation. Cycle-accurate SystemVerilog simulation. The same GEMM program is run under INT8, FP16, and FP32, and results are written to CSV along with derived hardware-aware metrics.

Metrics. We use the columns documented in the repository's metrics file [4], including throughput (ops/s, GOPS), accuracy (MAE, RMSE, percentiles), floating-point ULP error, error-shape indicators (RMSE/MAE, p99/p50), range utilization for INT8, and sign-error counts.

IV. Results

We summarize observations from the CSVs in this project for 8×8 GEMM.

A. Throughput

All three precisions report simulated throughput in a similar band for these small matrices. Median operations per second from the CSV with hardware metrics are shown in Table I. These should be interpreted as relative simulation values, not device-level claims.

TABLE I: Medians across runs from `results/comprehensive_results_with_hw_metrics.csv`.

Precision	Median ops/s	Median RMSE	Median MAE
INT8	140.590	2 563 785.896	2 172 450.727
FP16	137.534	2.840	2.091
FP32	143.974	3.106	2.372

B. Accuracy

FP16 closely tracks FP32 on these inputs, with single-digit MAE/RMSE in both modes. Representative FP16 rows show `ulp_error_mean` near 1 and `ulp_error_max` below 2, consistent with correct rounding for well-conditioned cases. INT8, run without per-tile scaling, shows large bias and error due to full-range utilization and saturation, as seen in the CSV columns `int8_range_utilization_pct` (often 100) and `bias_fraction` (often near 1).

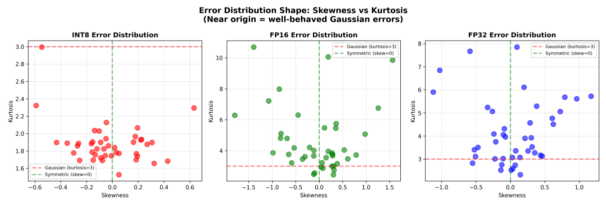

Fig. 3: Error distribution shape: RMSE/MAE and interquartile spread highlight outliers.

C. Range Utilization and Sign Errors

For INT8, range utilization is high without scaling, increasing the chance of saturation and sign errors. The INT8 trail shows high `sign_error_count` and `sign_error_pct` in some runs, signaling catastrophic direction flips that require mitigation.

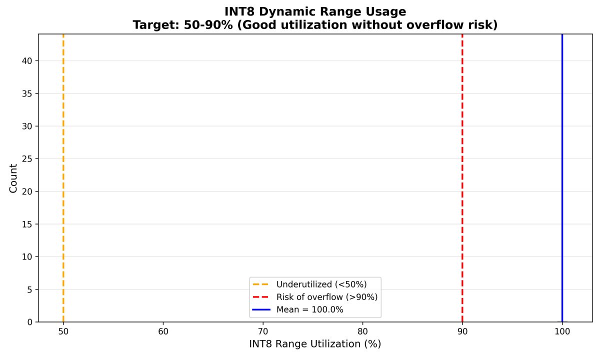

Fig. 4: INT8 range utilization in representative runs. Values near 100% indicate little headroom.

D. Pareto View

The Pareto plot mixes normalized throughput with accuracy. FP16 dominates many well-conditioned cases; INT8 improves once accuracy constraints are relaxed or proper scaling is introduced.

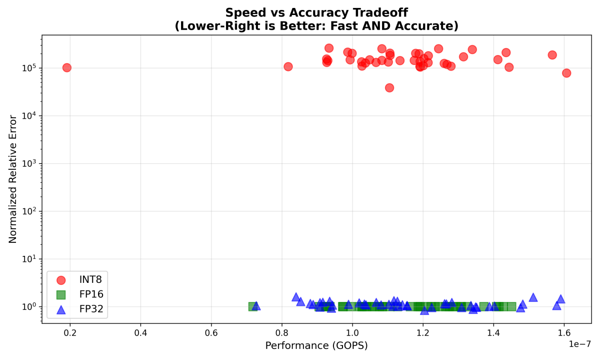

Fig. 5: Pareto trade-off between simulated throughput and accuracy.

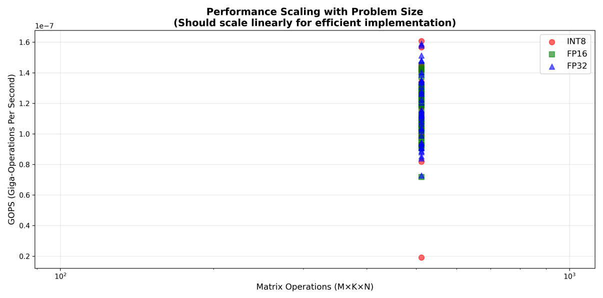

Fig. 6: Simulated performance scaling across test conditions in this repo.

V. Discussion

Precision and accumulation. INT8 is compelling for bandwidth and area, but requires a scaling policy and, for larger K, either block floating-point or a wider accumulator. FP16 can accumulate in FP16 for speed or in FP32 for stability.

Controller and memory. With a single PE, BRAM bandwidth is sufficient. Scaling to an array demands careful tiling and reuse to keep the array fed. The systolic shell can amortize A and B reads and increase arithmetic intensity.

Mapping to RISC-V. The INT8 PE aligns with vector dot-product instructions in RISC-V, and FP16 maps to Zfh units. The metrics used here—throughput, range utilization, sign errors—carry over to bring-up on a RISC-V SoC.

VI. Limitations

Simulation time dominates the throughput metrics, so we use them for relative comparison only. Matrices are small (8×8), which understates pipeline fill/drain effects present in larger arrays. INT8 results are intentionally unscaled to expose raw behavior.

VII. Future Work

Add per-tile scaling and block floating-point for INT8; accumulate FP16 into FP32 for ill-conditioned inputs; scale the systolic array and add a memory model to estimate device-level throughput; and integrate a RISC-V host to stream tiles on hardware.

VIII. Conclusion

A small, modular mixed-precision GEMM pipeline makes precision trade-offs explicit. FP16 delivers accuracy close to FP32 on these inputs. INT8 achieves compact data movement but requires scaling to avoid bias. The architecture here is a practical base for a RISC-V accelerator or FPGA prototype.

References

[1] B. Jacob, S. Kligys, B. Chen, M. Zhu, M. Tang, A. Howard, H. Adam, and D. Kalenichenko, "Quantization and training of neural networks for efficient integer-arithmetic-only inference," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR) Workshops*, 2018.

[2] RISC-V International, "Risc-v zfh extension for half-precision floating-point," 2022, rISC-V Unprivileged Spec, Extension Zfh.

[3] IEEE Std 754-2019, "Ieee standard for floating-point arithmetic," IEEE, 2019.

[4] A. Sabani, "Hardware-aware metrics implemented in the mixed-precision gemm repository," 2025, see docs/NEW_HARDWARE_METRICS_IMPLEMENTED.md in this repository.