PERCIVAL: Integrating Posit and Quire Arithmetic into the RISC-V Ecosystem

github.com/artecs-group/PERCIVAL

David Mallasén, Raul Murillo, Alberto A. Del Barrio, Guillermo Botella, Luis Piñuel, and Manuel Prieto-Matias

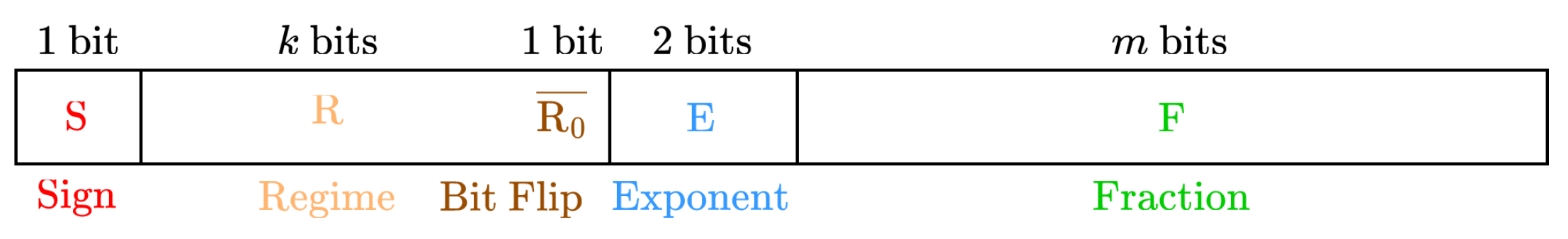
Corresponding: dmallasen@ucm.es
Facultad de Informática,
Universidad Complutense de Madrid, Spain

Abstract

Posit arithmetic [1] is an alternative to IEEE 754 standard floating-point [2] that presents promising properties in areas such as high-performance computing or artificial intelligence. The open-source PERCIVAL [4] posit RISC-V core integrates posit arithmetic and quire capabilities into hardware. In addition, Xposit, a RISC-V custom extension for posit operations allows for the compilation of C programs with inline assembly posit and quire instructions. PERCIVAL is based on the application-level CVA6 core developed by the PULP Platform and maintained by the OpenHW Group. As a study platform, it has support for both posit and IEEE 754 formats, further permitting the comparison of these arithmetic representations.

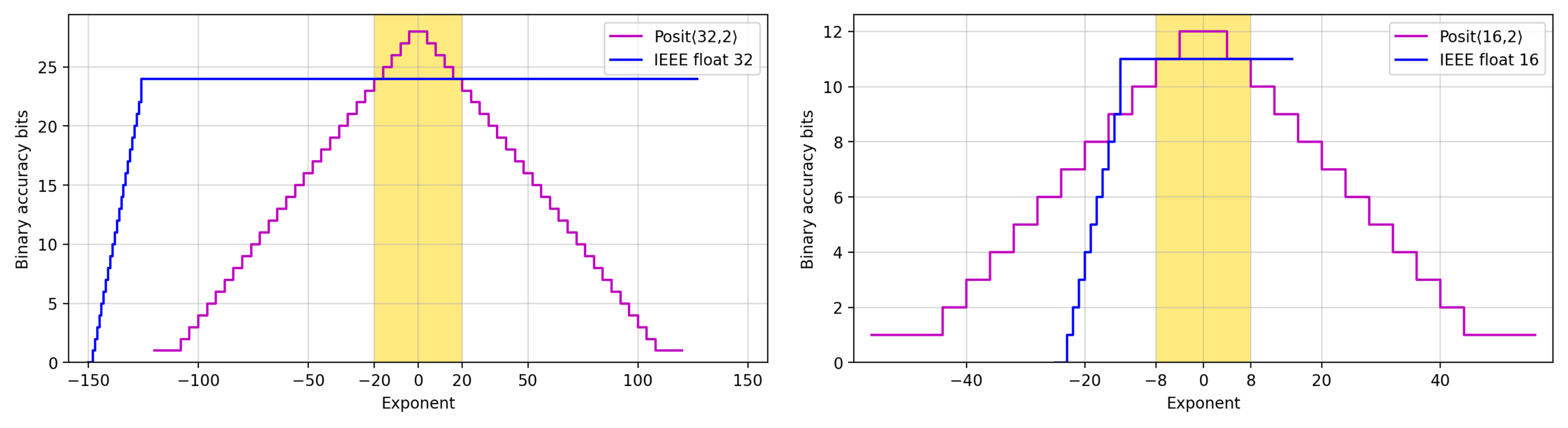
Posit Arithmetic

The posit number format defines a posit configuration from its total bit-width n. One of the main benefits of posit arithmetic is that they have only two special cases. The value zero and the Not-a-Real (NaR). The rest of the values are composed of four fields: Sign bit, variable-length regime, 2 exponent bits, and variable-length fraction field.



From these fields we can calculate the real value p of a generic posit as:

$$p = ((1 - 3s) + f) \times 2^{(1-2s)\times(4r+e+s)}.$$

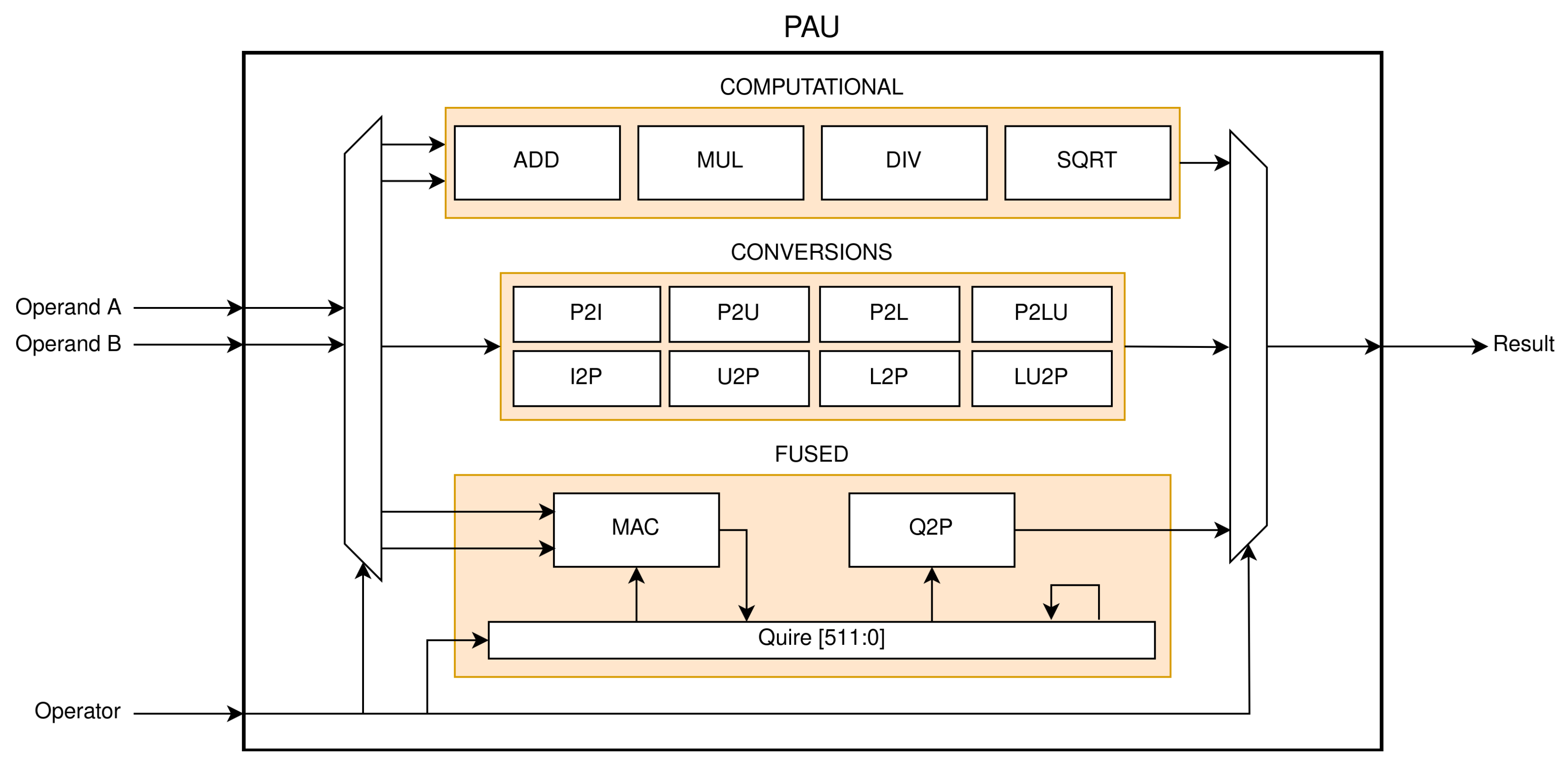


Posit arithmetic also includes fused operations using the quire, a $16n$-bit fixed-point 2's complement register. This special accumulation register allows for the execution of up to $2^{31} - 1$ Multiply-Accumulate (MAC) operations without intermediate rounding or accuracy loss. These operations are very common when computing dot products, matrix multiplications, or other more complex algorithms.

PERCIVAL

PERCIVAL [4] is a RISC-V core based on the application-level CVA6 core (ex. PULP's Ariane [5]). The main objective has been to maintain the compatibility between the IEEE 754 floating-point operations and the new functionality.

The Posit Arithmetic Unit (PAU) consists of the posit arithmetic and conversion modules, the quire operations, and a top-level module that orchestrates the execution of the instructions. This top-level module uses a synchronous handshake interface to transfer the data and read the control signals.



Xposit is a custom RISC-V extension that adds posit and quire instructions as well as a posit register file. It has been integrated into LLVM to allow the compilation of C programs together with inline assembly for the new instructions. The posit instruction set mimics the structure of the F RISC-V standard extension, adapting it to this novel arithmetic.

inst[4:2] / inst[6:5]	000	001	010	011	100	101	110	111 ($> 32b$)
00	LOAD	LOAD-FP	XPOSIT	MISC-MEM	OP-IMM	AUIPC	OP-IMM-32	$48b$
01	STORE	STORE-FP	*custom-1*	AMO	OP	LUI	OP-32	$64b$
10	MADD	MSUB	NMSUB	NMADD	OP-FP	*reserved*	*custom-2/rv128*	$48b$
11	BRANCH	JALR	*reserved*	JAL	SYSTEM	reserved	*custom-3/rv128*	$\geq 80b$

Synthesis Results

Synthesis results of PERCIVAL provide some insight into the total hardware cost of a posit- and quire-enabled CPU. Furthermore, we can compare the resource usage of posits in contrast to IEEE floats. Synthesis was executed on Vivado v.2020.2 for a Xilinx Kintex-7 FPGA with a target frequency of 50MHz (set by the CVA6 core).

32-bit	LUTs	FFs	DSP blocks
FPU	4046	973	2
PAU	11879	2985	4

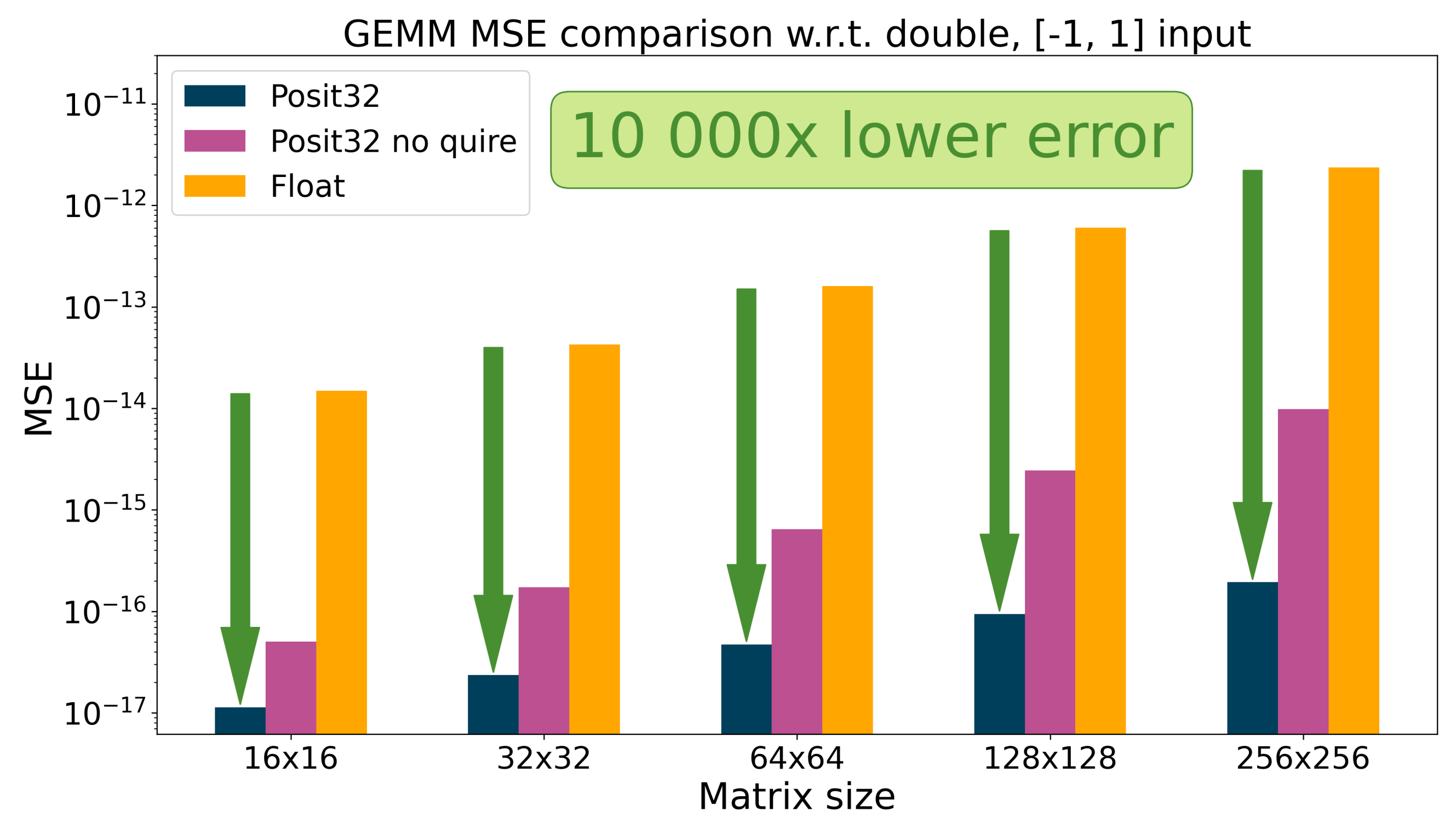
When comparing the PAU with quire with the Floating-Point Unit (FPU), the PAU requires around 3 times more resources. However, the quire MAC occupies 50% of the area. If we do not include the quire, we follow the results from previous works which reported an increase of around 30-35% more resources when using posit arithmetic when compared to IEEE floats.

- Quire MAC uses ∼50% of the area
- Posit32 without quire: ∼33% more area than FP

Name	LUTs	FFs
PAU top	593	1063
Posit Add	784	106
Posit Mult	736	73
Posit ADiv	413	43
Posit ASqrt	426	33
Quire MAC	**5644**	**1541**
Quire to Posit	**889**	**126**
Int to Posit	176	0
Long to Posit	331	0
ULong to Posit	425	0
Posit to Int	499	0
Posit to Long	379	0
Posit to UInt	228	0
Posit to ULong	358	0
PAU total	11879	2985
PAU w/o quire	**5346**	**1318**

Benchmarks

The General Matrix Multiplication (GEMM) showcases the accuracy benefits of using posit arithmetic. The results obtained using 64-bit doubles are used to compute the Mean Squared Error (MSE) of posit32 and 32-bit floats. The difference between MSEs is between 3 and 4 orders of magnitude in favor of posits when using fused operations. This is reduced to two orders of magnitude if the quire is not used. Overall, the impact of the quire is significant among all test cases, and its extra hardware cost is justified by the accuracy gains. Moreover, there is no performance penalty when using fused MAC operations and the quire, as posits execute as fast as floats.



Conclusions

Increased accuracy | Same performance | More area

In the past years, new emerging floating-point representations have provided alternatives to the widespread IEEE 754 format. In particular, posit arithmetic has been shown to have compelling benefits in areas such as machine learning or high-performance computing. The PERCIVAL posit core advances the native integration of posit arithmetic and quire in hardware. It provides a complete platform in which to further study the use of posit arithmetic. Since it also includes an IEEE 754 FPU, accurate comparisons can be made between these two arithmetic formats.

We are currently testing a posit64 version of PERCIVAL, which targets scientific computing, on more complex linear algebra kernels [3]. Furthermore, we are also developing a standalone posit co-processor that could be attached to other RISC-V cores both in the embedded and HPC domains.

References

[1] J. L. Gustafson and I. T. Yonemoto. Beating floating point at its own game: Posit arithmetic. *Supercomputing Frontiers and Innovations*, 4(2):71–86, Apr. 2017.

[2] IEEE Computer Society. IEEE Standard for Floating-Point Arithmetic. *IEEE Std 754-2019 (Revision of IEEE 754-2008)*, pages 1–84, July 2019.

[3] D. Mallasén, A. A. Del Barrio, and M. Prieto-Matias. Big-PERCIVAL: Exploring the Native Use of 64-Bit Posit Arithmetic in Scientific Computing, May 2023.

[4] D. Mallasén, R. Murillo, A. A. D. Barrio, G. Botella, L. Piñuel, and M. Prieto-Matias. PERCIVAL: Open-Source Posit RISC-V Core With Quire Capability. *IEEE Transactions on Emerging Topics in Computing*, 10(3):1241–1252, 2022.

[5] F. Zaruba and L. Benini. The Cost of Application-Class Processing: Energy and Performance Analysis of a Linux-Ready 1.7-GHz 64-Bit RISC-V Core in 22-nm FDSOI Technology. *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, 27(11):2629–2640, Nov. 2019.