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

David Mallasén[1]*, Raul Murillo[2], Alberto A. Del Barrio[1], Guillermo Botella[1], Luis Piñuel[2] and Manuel Prieto-Matias[1]

[1]Facultad de Informática, Universidad Complutense de Madrid
[2]Facultad de Ciencias Físicas, Universidad Complutense de Madrid

Abstract

Posit arithmetic is an alternative to IEEE 754 standard floating-point that presents promising properties in areas such as high-performance computing or artificial intelligence. The open-source PERCIVAL 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.

Introduction

Representing and operating on real numbers in a microprocessor presents unique challenges not encountered with the set of integers. Working with real numbers introduces additional concepts such as precision, that is, the error made between the number with which we want to operate and the approximation that we can represent in a finite number of bits. Currently, the universally extended way of representing the set of real numbers is using floating-point numbers defined by the IEEE 754 standard. Posits [1] are an alternative number representation format introduced in 2017 to mitigate the problems inherent to the floating-point standard.

The recent PERCIVAL[1] [2] proposal is the first core that supports all posit and quire operations natively in hardware. In addition, Xposit[2], a RISC-V custom extension for posit operations allows for the compilation of C programs with inline assembly posit and quire instructions. This leverages the extensibility and modular design of the RISC-V ISA.

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 bit patterns are composed of four fields: sign bit, variable-length regime (long-range dynamic exponent), 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 [3].

PERCIVAL

PERCIVAL [2] is a RISC-V core based on the application-level CVA6 core (ex. PULP's Ariane) [4], that includes support for posit arithmetic and quire in hardware. To test this new arithmetic format on HPC and AI problems, we have also included support for the Xposit custom extension to the RISC-V backend of the LLVM compiler.

The main objective behind PERCIVAL was to maintain the compatibility between the existing operations and the new functionality. This includes the IEEE 754 floating-point operations of the FPNew [5] FPU available in the CVA6. Thus, some modules in CVA6 were extended and a Posit Arithmetic Unit (PAU) was added to include the new instructions.

The PAU consists of the posit arithmetic and conversion modules, and a top-level module that orchestrates the execution of the instructions (Fig. 1). This top-level module uses a synchronous handshake interface to transfer the data and read the control signals. Fused MAC operations make use of the quire accumulator register. In our design, this 512-bit register is allocated internally in the PAU.

*Corresponding author: dmallasen@ucm.es
[1] https://github.com/artecs-group/PERCIVAL
[2] https://github.com/artecs-group/llvm-xposit

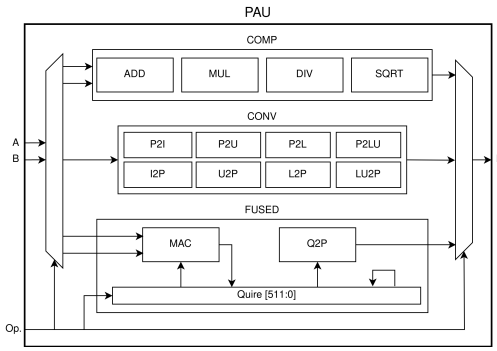

Figure 1: *Overview of the Posit Arithmetic Unit (PAU).*

Xposit

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.

The Xposit extension occupies the *custom-0* opcode. Currently, this is enough as there is only support for 32-bit posits. However, to support more posit bit-widths in the future, loads and stores could be separated into additional custom opcodes. This would follow the same OP-FP, LOAD-FP and STORE-FP structure as the F standard extension.

Results

For this project, we benchmarked the General Matrix Multiplication (GEMM) operation. The results obtained using the 64-bit IEEE 754 format were used to compute the Mean Squared Error(MSE) of the 32-bit posit and the 32-bit IEEE 754 floating point. As shown in Fig. 2, 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 the accuracy gains justify its extra hardware cost.

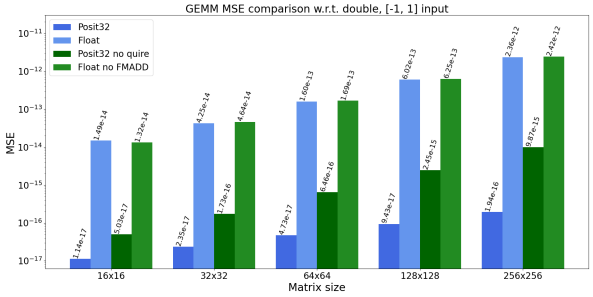

Figure 2: *MSE results of posits and floats when executing the GEMM operation with input values in* $[-1, 1]$.

Regarding the execution time of the previous GEMM test, when using fused MAC operations and the quire, the execution time of 32-bit posits is practically the same as that of single-precision floats. Therefore, especially for larger matrix sizes, which are common in scientific applications and Deep Neural Networks (DNNs), 32-bit posits perform equally as 32-bit floats and outperform 64-bit floats. However, the main drawback comes in the form of an additional cost in area.

We synthesized PERCIVAL targeting the Genesys II (Xilinx Kintex 7) FPGA and the PAU using TSMC's 45 nm standard-cell library to further study their hardware cost in ASIC. In both cases, results show that the 32-bit PAU with quire occupies between 2.5 and 3 times more resources than its floating-point counterpart. However, the quire MAC unit occupies half of the total area of the PAU. If we subtract the cost of the quire, the resulting overhead is around 30%. This is still a significant extra cost which goes in line with previous proposals at the individual operator level. However, the accuracy benefits without performance degradation provide a potential alternative when operating with real numbers.

Next Steps

We are currently testing a posit64 version of PERCIVAL, which targets scientific computing, on more complex linear algebra kernels. 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] Posit Working Group. *Standard for Posit Arithmetic (2022)*. Feb. 2022. URL: `https://posithub.org/docs/posit_standard-2.pdf` (visited on 05/16/2022).

[2] David Mallasén et al. "PERCIVAL: Open-Source Posit RISC-V Core With Quire Capability". In: *IEEE Transactions on Emerging Topics in Computing* 10.3 (2022), pp. 1241–1252. DOI: `10.1109/TETC.2022.3187199`.

[3] Raul Murillo et al. "Energy-Efficient MAC Units for Fused Posit Arithmetic". In: *2021 IEEE 39th International Conference on Computer Design (ICCD)*. Oct. 2021, pp. 138–145. DOI: `10.1109/ICCD53106.2021.00032`.

[4] Florian Zaruba and Luca 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". In: *IEEE Transactions on Very Large Scale Integration (VLSI) Systems* 27.11 (Nov. 2019), pp. 2629–2640. DOI: `10.1109/TVLSI.2019.2926114`.

[5] Stefan Mach et al. "FPnew: An Open-Source Multiformat Floating-Point Unit Architecture for Energy-Proportional Transprecision Computing". In: *IEEE Transactions on Very Large Scale Integration (VLSI) Systems* 29.4 (Apr. 2021), pp. 774–787. DOI: `10.1109/TVLSI.2020.3044752`.