FPHUB-RISCV: HUB Floating-Point Unit in RISC-V Platform – Format definition

Javier Hormigo*, Julio Villalba, Gerardo Bandera, Sonia Gonzalez-Navarro, Alfonso Martínez-Conejo, Alejandro Fuster, Jesús Lastre, Óscar Plata, Emilio L. Zapata†

Dept. Computer Architecture, Institute for Mechatronics Engineering & Cyber-Physical Systems, Universidad de Málaga

Abstract

FPHUB-RISCV is a 2-year "proof of concept" project within the Spanish PERTE chip. It aims to implement a new floating-point arithmetic unit for RISC-V targeting low-resource applications. The unit will use a new floating-point format called FPHUB, which allows us to simplify the implementation logic while keeping the same precision as the IEEE 754 standard.

Introduction

The RISC-V specifications of the Floating-Point (FP) instruction extensions demand that FP operations must comply with the IEEE 754–2008 standard [1], ensuring compatibility and precision across different software and hardware platforms. However, this adherence comes at a significant cost in terms of design complexity and hardware cost. Implementing IEEE-compliant FP units often requires substantial hardware resources to implement functionalities that are not always necessary. Moreover, the availability of open-source FP units for the RISC-V architecture is extremely limited, and most existing implementations are optimized for full IEEE compliance rather than efficiency or simplicity. This situation could be attributed to the challenges and expenses associated with designing an FP unit from scratch, notably when prior experience in this domain is lacking. These problems hinder the adoption of floating-point formats for low-power or resource-constrained environments.

Despite these challenges, there is a growing demand for floating-point capabilities in low-cost, low-power applications, such as IoT devices, edge computing, and embedded systems, where precision is needed, but the full rigor of the IEEE standard is often unnecessary. This gap in the market underscores the need for innovative designs that balance compatibility, performance, cost, and energy efficiency, enabling broader adoption of floating-point arithmetic in resource-constrained environments. Thus, our work aims to fill this technological gap and provide a more accessible and efficient solution for integrating low-resource floating-point units into the RISC-V architecture. To do that, we have been founded by the Spanish government through the project FPHUB-RISCV.

*Corresponding author: fjhormigo@uma.es

†Grant PDC2023-145800-I00, founded by MICIU/AEI /10.13039/501100011033 and European Union Next Generation EU/PRTR; and grant PID2022-136575OB-I00 by MICIU/AEI/10.13039/501100011033

FPHUB-RISCV Project

The FPHUB-RISCV is a "proof of concept" project that aims to valorize our research results in the design of floating-point arithmetic operators for HUB format. To do that, we are developing a low-power FP unit using our HUB technology for floating-point calculations. This FP unit should be versatile and seamlessly fit into various existing or emerging RISC-V architectures. The adoption of a HUB numerical representation format [2] in our project offers a compelling advantage: the ability to significantly reduce the processor's area and power consumption, all while maintaining precision in calculations.

Our HUB approach, introduced in [2], stands out for its ability to simplify arithmetic units at the logic level, providing significant advantages in hardware implementations. The HUB approach concurrently reduces area resources, power consumption, and critical path delay [3]. Our research group has previously undertaken comprehensive investigations into implementing fundamental FP operations, including addition, subtraction, multiplication, division, and square root, using HUB formats at theoretical level [3, 4, 5, 6, 7]. More recently, we developed a functional prototype of a floating-point Adder designed within a RISC-V architecture [8]. We will build upon this experience to develop our FP unit.

Since our main objective is technology transfer, this work will be open source. We will have a public repository [9] with all the development available and the corresponding documentation. We also plan to do rigorous laboratory testing and field validation using FPGAs. To facilitate the test of our new format for third parties, we will develop a software library to simulate the FP calculation using HUB approach. The first version of the FP unit should be available by

next year. We should note that although the source code will be available with a permissive license, the Universidad of Malaga held several Spanish patents concerning HUB circuits. Therefore, commercial use of the developed FP unit within Spain may require a technology transfer agreement with our University.

It is essential to highlight that among our objectives, there is no intention to independently undertake the physical implementation of a RISC-V processor with our FP unit in ASIC. This falls outside our current expertise. However, if there is a suitable partner with the expertise to integrate our design into an ASIC, we are open to collaborating to achieve this goal.

FPHUB: Floating Point HUB format

In this section, we present the specific format that we will implement in this project, which we will call FPHUB. The fundamentals of general HUB FP formats can be found in [2].

An FPHUB number is similar to a binary IEEE standard one but, similarly to the implicit integer bit, its significand also has an Implicit Least Significant Bit (ILSB) set to one. Thus, the normalized FPHUB significand has the form $1.M_x1$, where M_x is the explicit fractional part of the significand with f fractional bits.

Moreover, not-a-number (NaN) and subnormal are not considered, allowing us to expand normalized numbers' exponent. Thus, the bias of the exponent is set to $2^{n_{exp}-1}$ instead of $2^{n_{exp}-1}-1$, where n_{exp} is the number of bits of the exponent. Therefore, a normalized FPHUB number is coded by the triple (S_x, E_x, M_x), which represents the value:

$$X = (-1)^{S_x}(1 + M_x + 2^{-f-1})2^{E_x - 2^{n_{exp}-1}}, \quad (1)$$

where S_x is the sign, and E_x is the biased exponent.

On the other hand, we keep the special signed zero and infinities. However, although zero has the same codification (both exponent and significand equal zero), the infinities are coded with the maximum value of the significand (all 1's) instead of zero. Furthermore, we have also added the signed one as a special case. The one case is coded using "all 0's" for the significand field, and the exponent is coded as $2^{n_{exp}-1}$. In table 1, we summarize the special cases for the example of binary32.

Another critical point is that rounding–to–nearest is the only rounding mode supported by FPHUB, although it could be biased or unbiased for the tie case. The former case is performed simply by truncation [3] of the significand. The latter one also needs that the explicit last significant bit was set to zero when all discarded bits are 0's[10]. Moreover, all numbers lower than the minimum normalized number are flushed to zero with their corresponding sign (underflow).

Case	Code	8 bits	23 bit
0	$(0,0,0)$	0	00000000 000000...0000000
0	$(1,0,0)$	1	00000000 000000...0000000
+1	$(0,128,0)$	0	10000000 000000...0000000
-1	$(1,128,0)$	1	10000000 000000...0000000
$+\infty$	$(0,255,2^{23}-1)$	0	11111111 111111...1111111
$-\infty$	$(1,255,2^{23}-1)$	1	11111111 111111...1111111

Table 1: *Special cases for FPHUB with 8-bit exponent and 24-bit precision*

The FPU will be integrated as a Zfinx extension to reduce the cost. Thus, we will implement a binary32 FPHUB. This first design could be extended to other sizes later.

References

[1] "IEEE Standard for Floating-Point Arithmetic". In: *IEEE Std 754-2008* (Aug. 2008), pp. 1–70. DOI: 10.1109/IEEESTD.2008.4610935.

[2] J. Hormigo and J. Villalba. "New formats for computing with real-numbers under round-to-nearest". In: *IEEE Transactions on Computers* 65.7 (2016), pp. 2158–2168. ISSN: 00189340. DOI: 10.1109/TC.2015.2479623.

[3] Javier Hormigo and Julio Villalba. "Measuring Improvement When Using HUB Formats to Implement Floating-Point Systems under Round-to-Nearest". In: *IEEE Transactions on Very Large Scale Integration (VLSI) Systems* 24.6 (2016), pp. 2369–2377. ISSN: 10638210. DOI: 10.1109/TVLSI.2015.2502318.

[4] J. Villalba-Moreno. "Digit Recurrence Floating-point Division under HUB Format". In: *23rd IEEE Symposium on Computer Arithmetic, Silicom Valley (California, USA)* (July 2016).

[5] Javier Hormigo and Julio Villalba. "HUB Floating-Point for Improving FPGA Implementations of DSP Applications". In: *IEEE Transactions on Circuits and Systems II: Express Briefs* 64.3 (2017), pp. 319–323. DOI: 10.1109/TCSII.2016.2563798.

[6] J. Villalba-Moreno and J. Hormigo. "Floating Point Square Root under HUB Format". In: *2017 IEEE International Conference on Computer Design (ICCD)*. Nov. 2017, pp. 447–454. DOI: 10.1109/ICCD.2017.79.

[7] Javier Hormigo, Julio Villalba-Moreno, and Sonia Gonzalez-Navarro. "Floating–Point Fused Multiply–Add under HUB Format". In: *2020 IEEE 27th Symposium on Computer Arithmetic (ARITH)*. 2020, pp. 1–8. DOI: 10.1109/ARITH48897.2020.00010.

[8] Gerardo Bandera et al. *Floating Point HUB Adder for RISC-V Sargantana Processor*. 2023. URL: https://arxiv.org/abs/2401.09464.

[9] FPHUB-RISCV. *HUBformat Repository*. Accessed: 2025-02-05. 2025. URL: https://github.com/HUBformat.

[10] Julio Villalba-Moreno, Javier Hormigo, and Sonia González-Navarro. "Unbiased Rounding for HUB Floating-Point Addition". In: *IEEE Transactions on Computers* 67.9 (Sept. 2018), pp. 1359–1365. ISSN: 1557-9956. DOI: 10.1109/TC.2018.2807429.