Hardware Support for Variable Precision Floating Point Formats Exploration

Riccardo Alidori[1], Andrea Bocco[1] *

[1]CEA-LIST, Grenoble, France

Abstract

Variable Precision (VP) Floating Point (FP) is a solution to compensate accumulation and rounding errors during computing. Hardware approaches to VP FP are often preferred, since software solutions result in algorithmic performance degradation and computational instability. We propose three new VP formats and evaluate them using a RISC-V platform running on a FPGA. Preliminary results show that our VP formats are beneficial for certain applications with low-precision requirements.

Introduction

Variable Precision (VP) Floating Point (FP) is a solution to compensate accumulation and rounding errors during computing. Compared to the IEEE-754 FP standard [1], VP FP is based on the assumption that each variable is fine-tuned in its length and precision by the programmer, optimizing the algorithmic error, latency or memory footprint. State of the art VP FP formats are UNUM [2] and Posit [3].

Hardware approaches to VP FP are often preferred, since software solutions result in algorithmic performance degradation and computational instability. This work extends the VaRiable Precision (VRP) accelerator [4] functionalities. This RISC-V CVA6 based accelerator handles 512 bits extended FP variables. It uses two FP formats, one internal used in the VRP FPU, *P*, and one for memory operation, *X*.

The objective of this work is twofold: 1/ It proposes three new VP FP formats, Custom Posit (PCUST), Not Contiguous Posit (NCP) and Modified Posit (MP); 2/ It presents the micro-architecture of the extended VRP Load and Store Unit (LSU), enabling the VRP to multiple VP FP formats support. The goal is to have a common hardware platform to explore VP FP formats, allowing the programmer to dynamically choose among several supported VP FP formats, as well as fine-tuning their precision and configuration at runtime. As a proof-of-concept, we report results of two test applications run on the modified VRP: 1/ Gauss Elimination as high-, and 2/ FFT as low-precision applications.

Variable Precision Floating Point

The FP IEEE-754 2008 standard [1] extends its previous format definition, *double* and *float*, which have fixed size fields, to FP encoding with arbitrary field size. Contrary to the IEEE-754, a VP FP format shrinks the exponent bit-width for exponent values close to zero. The saved bits are used to enlarge the mantissa precision. The major difference among VP FP formats is their exponent encoding. The exponent fields bit-width varies according to the encoded value.

UNUM [2] is the first proposal of VP FP format. Posit (UNUM Type III) [3] (PSTD) updates the previous one, even though it shows some flaws [5]. Hence, we propose three new VP FP formats.

A. *Custom Posit* (PCUST): a Posit optimization. 1/ It adds Infinity and NaN symbols encoding, 2/ It limits the maximum Regime Bits field size to gain in mantissa precision, 3/ It avoids a two's complement to encode negative FP values.

B. *Not-Contiguous Posit* (NCP): a mix of IEEE-754 and Posit. An additional *flag* bit indicates if the encoding is a IEEE-Like or a Posit-Like. This choice is dynamic. It favours the exponent encoding with minimal footprint between the two, based on the represented value.

C. *Modified Posit* (MP): It provides an exponent encoding which is a trade-off between the one used in Posit and the UNUM type I.

This work extends the functionalities of the VRP, a RISC-V CVA6 based accelerator for extendable FP computation [4]. VRP aims to facilitate the use of variable extended precision in order to improve the numerical stability of big linear algebra algorithms. The VRP adopts two FP formats: 1/ *P*, used into the computing unit of the processor, VRP FPU, which is in charge of performing basic FP operations such as additions and multiplications; 2/ *X*, IEEE-754 extendable format used to store (and load) in (and from) memory. Both *P* and *X* have a bit-length from 8 to 512 bits, tunable in length with a bit granularity. Using *X* only in memory allows a simpler VRP FPU. The 512 bits FP computing is performed by iterating on 64 bits data chunks. The VRP Load and Store Unit (LSU) performs the conversion on the fly during

*Corresponding author: riccardo.alidori@cea.fr

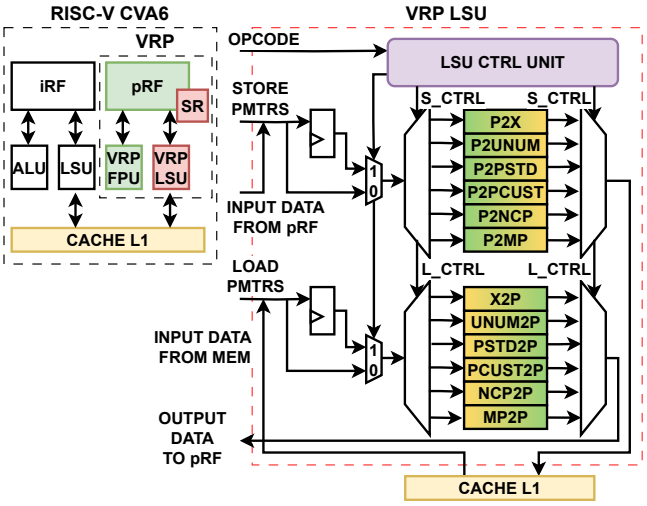

Figure 1: *The VRP and its modified LSU*

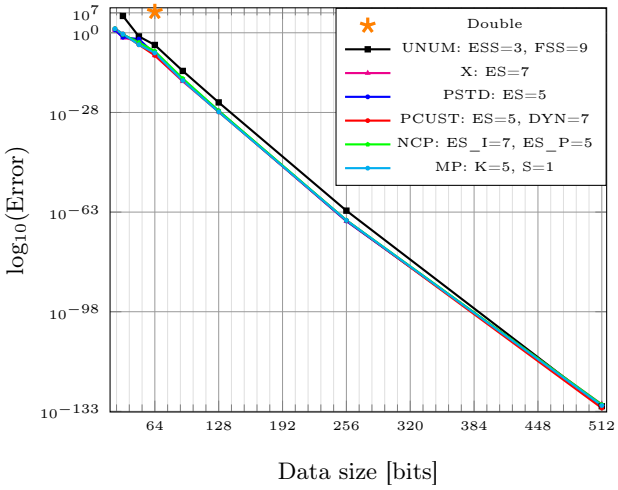

Figure 2: *GE error results. Best format configurations*

loads, *X2P*, and stores operations, *P2X*, by means of dedicated hardware converters. Format configuration is handled by apposite Status Registers (SR).

RISC-V Integration & Validation

To support all the aforementioned formats, we designed the hardware memory converters from P to the proposed VP FP formats for load and store operations, e.g. *MP2P* & *P2MP*, etc. We extended the VRP LSU [4], as well as the VRP software interface. Figure 1 depicts the modified VRP LSU in details: all the load and store converters share the same input and output. Depending to the current operation, format and parameters from the SR, the VRP LSU CTRL Unit can stall load/store instructions when required, as well as manage mutual exclusion between converters.

The circuit is compiled onto a Xilinx Virtex UltraScale+ board. The modified VRP occupies 37.17% of the available LUTs, and operates at a 50MHz clock

Bits	UNUM	PSTD	PCUST	NCP	MP	X	IEEE-754
8		75.65%	75.65%	79.67%	75.65%	77.65%	
16	89.85%	99.43%	99.43%	99.43%	99.42%	98.82%	BFloat16, 94.54%
24	99.42%	99.99%	99.99%	99.98%	99.98%	99.94%	
32	99.95%	100%	100%	100%	100%	100%	Float, 100%

Table 1: *FFT accuracy results. Best format configurations*

frequency. We benchmarked our architecture with the Gaussian Elimination (GE) and FFT for high- and low-precision applications. Both algorithms are coded in C and run in the VRP. The two applications are executed for each supported format and configuration. For the GE algorithm we use a 100x100 Hilbert matrix. Figure 2 shows the algorithm residual error. These results show that except for UNUM which performs worst, the choice between VP FP formats is not relevant for this high-precision application. Also, we analyze a 8192 samples audio with an in-place FFT butterfly algorithm. Accuracy is calculated from the ratio between the original and the inverse-FFT reconstructed signals, shown in Table 1. Results show that VP can achieve better accuracy with the respect to the standard FP formats. Not Contiguous Posit performs best with 8 bits variables.

Conclusion

In this work we give VP FP support to the VRP, a RISC-V based extendable-precision FP accelerator, with the aim of performing VP FP format exploration. We integrate several VP FP hardware converters into the VRP LSU. Experimental results show that VP format choice may be irrelevant for high-precision applications, while it has some benefit when handling small-size variables.

Acknowledgment

This work has received funding from the European Union's Horizon 2020 research and innovation program under Grant Agreement EPI-SGA1: 826647 (EPI Project).

References

[1] *IEEE Standard for Floating-Point Arithmetic.* DOI: 10.1109/IEEESTD.2008.4610935.

[2] John Gustafson. *The End of Error: Unum Computing.* DOI: 10.1201/9781315161532.

[3] Gustafson and Yonemoto. *Beating Floating Point at Its Own Game: Posit Arithmetic.* DOI: 10.14529/jsfi170206.

[4] Yves Durand et al. *Accelerating Variants of the Conjugate Gradient with the Variable Precision Processor.* DOI: 10.1109/ARITH54963.2022.00017.

[5] Florent Dinechin et al. *Posits: the good, the bad and the ugly.* DOI: 10.1145/3316279.3316285.