HUB-Floating-Point for improving FPGA implementations of DSP Applications

Javier Hormigo, and Julio Villalba, *Member, IEEE*

***Abstract*—The increasing complexity of new digital signal-processing applications is forcing the use of floating-point numbers in their hardware implementations. In this brief, we investigate the advantages of using HUB formats to implement these floating-point applications on FPGAs. These new floating-point formats allow for the effective elimination of the rounding logic on floating-point arithmetic units. Firstly, we experimentally show that HUB and standard formats provide equivalent SNR on DSP application implementations. We then present a detailed study of the improvement achieved when implementing floating-point adders and multipliers on FPGAs by using HUB numbers. In most of the cases studied, the HUB approach reduces resource use and increases the speed of these FP units, while always providing statistically equivalent accuracy as that of conventional formats. However, for some specific sizes, HUB multipliers require far more resources than the corresponding conventional approach.**

***Index Terms*—FPGA, floating-point, DSP applications, HUB-format**

I. Introduction

NOWADAYS, many Digital Signal Processing (DSP) applications, such as graphics, wireless communications, industrial control, and medical imaging require the use of linear algebra or other complex algorithms. The use of Floating-Point (FP) arithmetic is quickly becoming a requirement in these applications due to its extended dynamic range and precision. For this reason, FP arithmetic is being introduced on FPGA implementations, as a soft-core [1] [2], or even as a hardware block in the newest Altera devices [3]. Although these embedded hardware blocks are more efficient and cost effective than their equivalent soft-core designs, the latter are still very useful. Firstly, low-cost devices do not offer these FP embedded blocks and it is not clear that other FPGA brands are going to include something similar in their devices in the near future. Secondly, up to now, only single precision has been directly supported in DSP blocks [4]. Therefore, improvements to the soft-core implementations are of great value.

Some of these solutions are being designed to follow the IEEE754 standard [5]. However, in many applications, compliance with this standard is sacrificed to obtain more efficient implementations regarding area and performance. In relation to FPGAs, much more efficient designs are obtained by using more flexible implementations of FP numbers and ensuring the fulfillment of certain quality parameters at the output. These flexible implementations could utilize word-length optimization [6] [7], high-radix representation [8], and fused datapath synthesis [9], or avoid the implementation of unnecessary rounding modes [2], exceptions, or subnormals support [1].

Generally, both hard and soft cores only support the round-to-nearest-even (RNE) mode, since this is the most useful of the rounding modes. In these FP cores, a significant amount of resource use and delay is due to the rounding logic. However, two new families of formats, HUB (Half-Unit biased) [10] and Round-to-Nearest [11] representations, allow RNE to be performed simply by truncation, which could make rounding logic negligible. Here, we focus on HUB formats. HUB Fixed-point formats were used in [12] and [13] to improve DSP implementations, since they allow better word-length optimization. The ASIC implementation of HUB-FP units has been studied for binary16 (half), binary32 (single), and binary64 (double) [5], and important improvements have been achieved [14] [15]. In this brief communication, we extend this analysis to FPGAs over a wide range of sizes. Compared to previous articles, we provide:

- An experimental error analysis of the implementation of FIR filters, which shows that the HUB approach provides similar statistical parameters to those of standard FP implementations, including the SNR.
- The results of FPGA implementation of a basic FP adder and multiplier for a wide range of exponent and mantissa bit-widths under HUB and conventional approaches and their comparison.

In most of the cases studied, the HUB format reduces resource use and increases the speed of these FP units. Furthermore, due to its simplicity, any existing soft or hard core could be easily enhanced by using the proposed approach. Therefore, based on basic architectures, our aim is to encourage researchers to improve their optimized FP cores or DSP applications by using HUB-FP formats.

II. HUB-FP numbers and associated circuits

Firstly, we summarize the main characteristics of the HUB-FP formats and circuits presented in [10] [15]. For demonstrations or further explanations, please refer to these papers.

A HUB-FP number is an FP number such that its mantissa (or significand) has an Implicit Least Significant Bit (ILSB) which equals one. Compared with a standard format, it has the same number of explicit bits and precision, but the same bit-vector represents a value biased half Unit-in-the-Last-Place (ulp) [10]. For example, using an m-bit HUB number

This work was supported in part by the Ministry of Education and Science of Spain under contracts TIN2013-42253-P.

The authors are with the Department of Computer Architecture, Universidad de Málaga, Málaga E-29071 Spain (e-mail: fjhormigo@uma.es; jvillaba@uma.es).

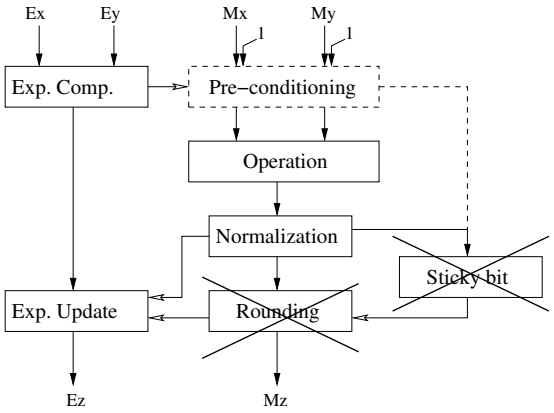


Fig. 1. Basic HUB-FP arithmetic architecture

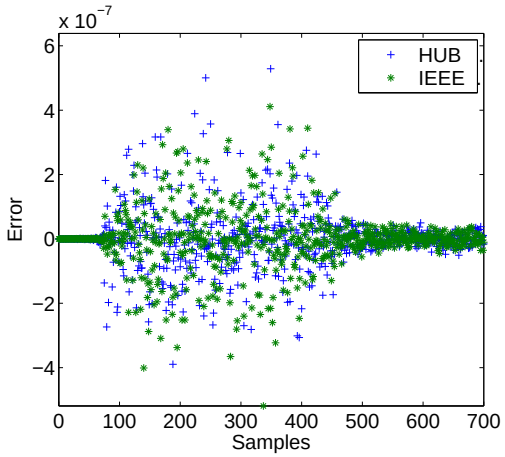


Fig. 2. Absolute error of the output of 150-tap FIR filter when using single instead of double precision

normalized between $(1,\ 2)$ for the mantissa (M_x), a sign bit (S_x), and an exponent E_x, the HUB-FP X'= $(S_x, E_x, M_x)'$ represents

$$(S_x, E_x, M_x)' = (-1)^{S_x} \cdot \left(\left[\sum_{i=-m+1}^{0} X_i \cdot 2^i \right] + 2^{-m} \right) \cdot 2^{E_x} \tag{1}$$

where X_i are the m bits of the mantissa M_x. Now, let us consider $m = 4$: the mantissa 1.001 represents 1.125 under conventional format, but 1.1875 under the HUB format, both with an error bound of ± 0.0625. Therefore, an exact real value is represented for different numbers under each approach, and a different rounding error is also produced, but both errors are within the ± 0.5ulp bound.

The main advantages of using the HUB format are that two's complement is computed by only bit-wise inversion, the truncation of a value to obtain a HUB number produces an equivalent RNE, and a sticky-bit computation is not required for most operations. Moreover, the conversion to a conventional format only requires explicitly appending the ILSB to the original number. Therefore, HUB numbers could be easily operated by conventional arithmetic circuits, while practically eliminating the rounding logic. Furthermore, the impact of the inclusion of the ILSB is limited since it is constant.

A basic general architecture to operate HUB-FP numbers is shown in Fig. 1, where a conventional FP arithmetic unit has been conveniently modified. Firstly, the ILSBs are appended to the mantissas of the input operands before using them. As consequence they are converted to conventional format. Next, a conventional datapath is used, but at first the mantissa data-path has to be one-bit wider. Since the final result is in HUB format, a simple truncation is performed for rounding. Thus, after the arithmetic operation itself, a conventional normalization logic is utilized, but no guard bit is provided at the output. The rounding logic of the conventional architecture is simply eliminated (crossed out in Fig. 1).

Taking into account that the ILSB is a constant, this general architecture could be further optimized depending on the specific architecture. A detailed architecture for addition and multiplication is provided in [15].

TABLE I
STATISTICAL PARAMETERS OF THE ROUNDING ERROR DISTRIBUTION.

	IEEE	HUB	IEEE	HUB	IEEE	HUB	IEEE	HUB
Taps	min$_{(10^{-7})}$		mean$_{(10^{-9})}$		max$_{(10^{-7})}$		$\sigma_{(10^{-8})}$	
FIR100	-5.25	-5.17	0.036	-0.32	4.83	4.39	0.86	0.87
FIR125	-5.32	-4.98	-1.84	-1.61	6.02	5.30	1.02	1.00
FIR150	-5.51	-6.11	-0.54	-1.90	5.88	5.88	1.05	1.04
FIR175	-7.47	-7.35	1.33	0.38	5.31	6.10	1.20	1.18
FIR200	-7.95	-8.08	-2.01	-1.55	7.17	7.03	1.22	1.20

III. ROUNDING ERROR OF HUB-FP COMPUTATION

The equivalence between truncation under HUB formats and RNE under conventional formats has been theoretically demonstrated in [10] and experimentally demonstrated for isolated operations in [15]. In this section, we show that although the specific error for each output value is always different, the statistical error performance of the HUB approach is very similar to the conventional one for DSP applications.

Specifically, several FIR filters have been implemented using IEEE double-precision as reference designs. Similarly, the output of the same filters was computed using single-precision for both the IEEE-754 standard and the corresponding HUB format. Conventional computation was performed using MATLAB in a PC, whereas the results of the HUB approach were obtained through VHDL simulation. Next, we analyze the error observed between double- and single-precision implementations.

As an example, Fig. 2 shows the absolute error of the output samples for both approaches, corresponding to a 150-tap low-pass FIR filter when a chirp signal is introduced. As expected, the error corresponding to conventional and HUB approaches is always different. Since the exactly-represented numbers of both approaches are different, the rounding error cannot coincide. However, they are distributed in the practically same way. To measure this outcome, similar experiments were performed for several low-pass filters, using a chirp input signal with 10000 samples. Table I shows some statistical parameters of the error for these experiments, including the

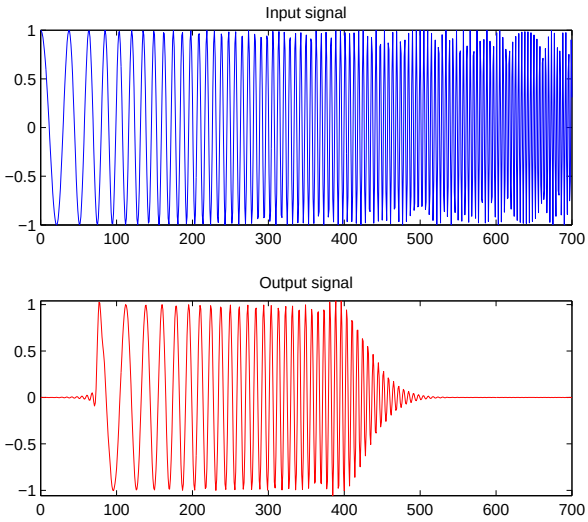


Fig. 3. Input/output signals of a 150-tap low-pass FIR filter example

bounds, the bias (mean), and the standard deviation. It can be seen that, in general, the values corresponding to both approaches are very similar for all parameters. Nevertheless, depending on the specific filter used, the results are slightly better for the IEEE standard or for the HUB format. We found that this behavior depends on how well the coefficients of the filter are represented under each approach, i.e., the amount of rounding error produced when representing the coefficients on each format. This depends on the values of the coefficients themselves and the mantissa bit-width. Thus, it happens in arbitrary manner and should not be considered to be a difference between both approaches. The same behavior is observed for the relative error measured by their SNR_{dB}, as shown in the first column of Table II.

On the other hand, Fig.2 clearly shows two different areas: the magnitude of the error is greater for the first half of the samples than for the second half. To explain this, we refer to the input signal and the output signal of the mentioned FIR filter in Fig.3. The magnitude of the absolute error decreases because the output signal goes down to nearly zero when the frequency of the input signal goes above the cutoff frequency. However, the relative error increases, since many catastrophic cancellations (i.e., subtraction of numbers with similar magnitudes) take place in the computation to produce this attenuated output.

To estimate the accuracy of the HUB-FP computation when many cancellations occur, the second and third columns of Table II show the SNR when only taking into account the first and the last 30% of output signal samples, respectively. Clearly, the relative error increases when the output signal approaches zero, but the behavior is the same for both approaches. The small differences in the SNR values again arise from the error of the representation of the coefficients for each specific filter. Therefore, taking into account these results and the previous ones, we conclude that the accuracy of both approaches is statistically equivalent.

TABLE II
SIGNAL TO ERROR NOISE RATIO (dBs) (SNR_{dB})

	IEEE	HUB	IEEE	HUB	IEEE	HUB
Taps	Full signal		Low Freq.		High Freq.	
FIR100	136.24	136.21	136.56	136.37	107.70	107.31
FIR125	134.84	134.94	135.08	135.10	97.63	98.90
FIR150	134.56	134.66	134.82	134.96	93.09	92.88
FIR175	133.47	133.58	133.58	133.70	90.60	90.50
FIR200	133.32	133.44	133.62	133.57	88.61	88.11

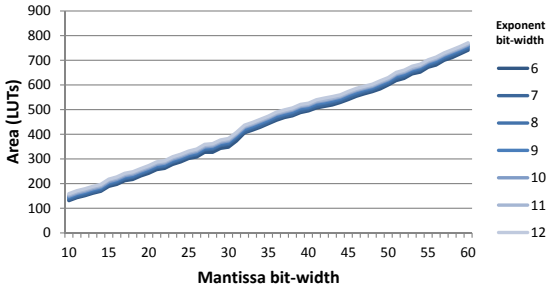
IV. IMPLEMENTATION RESULTS ANALYSIS

We now analyse and compare the main results of the FPGA implementation of a HUB-FP adder and multiplier to the results of the corresponding conventional ones. Since these are the main operations involved in DSP applications, this approach allows us to estimate the benefits of using HUB formats to implement FP computation in DSP applications. In order to measure the impact of the HUB approach alone, and to keep the implementation as flexible and general as possible, our implementations allow any bit-width, but do not support special cases, subnormal cases, or any optimization of the datapath. Therefore, the results obtained in this study should be considered to be an estimation of the improvement that can be achieved by using HUB-FP formats and to encourage further investigation using optimized cores and specific applications.

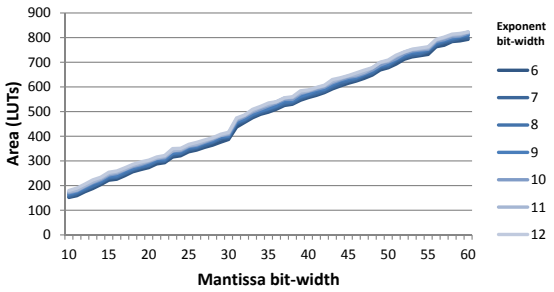
To perform this study, the basic architectures of the adders and multipliers presented in [15] for conventional and HUB-FP numbers were described in VHDL, such that the bit-width of the mantissa and exponent were configurable. Moreover, to facilitate comparisons, all the designs were fully combinational, although in future research, we will try to confirm that similar behavior occurs for pipeline implementations. The adders and multipliers for both approaches were synthesized using Xilinx ISE 14.3 and targeting Xilinx Virtex-6 FPGA xc6vlx240t-1 for a wide range of formats. Specifically, we used all FP formats with mantissa sizes ranging from 10 to 60 bits and exponent sizes ranging from 6 to 12 bits.

Fig. 4 shows the area (LUTs) occupied by the conventional adders and the proposed FP adders. The mantissa bit-width is represented in the x-axis, and the exponent bit-width is represented by different coloured lines. It can be seen that the exponent bit-width has very little impact on the area, which rises slightly when it increases. The proposed adder requires significantly less area than the conventional one. To quantitatively indicate the improvement obtained by using HUB formats, Fig. 5 shows the area of the new HUB adders divided by the area of their corresponding conventional adders. The area savings range from 6% to 18% with a mean of about 11%.

Similarly, the delay of the critical path corresponding to the FP adders is shown in Fig 6. It can be seen that the lines are more irregular than in the case of the area and are particularly irregular in the conventional approach. However, in general, the proposed HUB approach is faster than the conventional one. This is more clearly seen in Fig 7, which shows the speedup achieved in each case. Except for one case

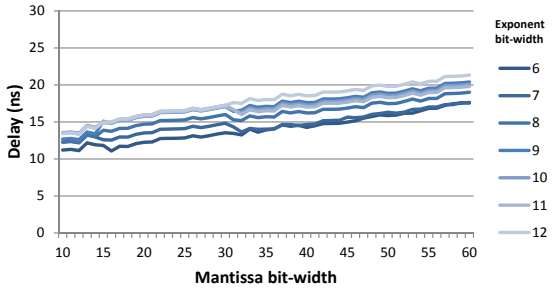


(a) HUB approach

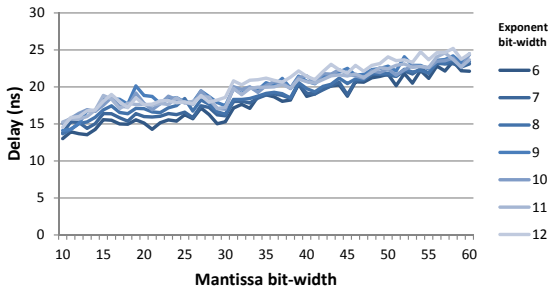


(b) Conventional approach

Fig. 4. Area used by FP-adders under both approaches



(a) HUB approach



(b) Conventional approach

Fig. 6. Delay of FP-adders under both approaches

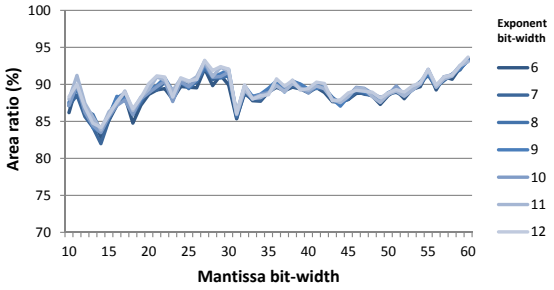


Fig. 5. Ratio of adder areas under both approaches (HUB-FP/conventional)

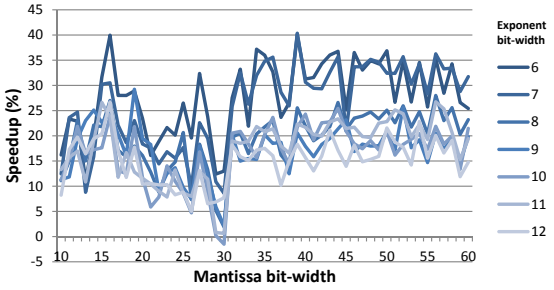


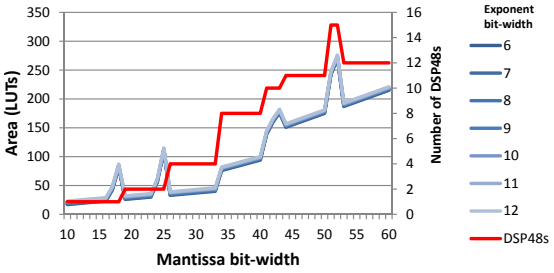
Fig. 7. Speedup of HUB-FP adders versus conventional ones

(specifically, for a 30-bit mantissa and 10-bit exponent), the HUB FP adder is faster than the equivalent conventional one, achieving an acceleration of up to 40% and a mean speedup of 20%.

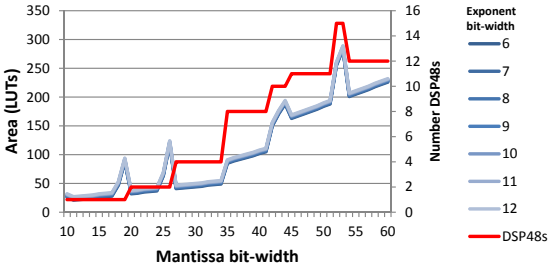
Fig. 8 presents the area used to implement the FP multipliers, and includes the number of LUTs and the number of built-in multipliers (DSP48 blocks). Note that the number of DSP48 blocks used is automatically selected by the synthesis software. It can be observed that the influence of the exponent bit-width is practically negligible and the number of LUTs dramatically increases just before the new DSP48 blocks are occupied. The number of DSP48 blocks is the same under both approaches, but this number increases one bit earlier under the HUB approach (i.e., both red lines are identical, but shifted by one position). The number of LUTs undergoes a similar shift, but in this case the number is lower for the HUB multipliers. This fact is better observed in Fig. 9 in which the number of LUTs is represented as a ratio. In a few cases, the HUB multipliers require up to 75% more LUTs than under the conventional approach, whereas the reduction in the remaining cases ranges from 70% to 5%.

On the other hand, Fig. 10 shows the delay of the FP multipliers. Since the delay is independent of the exponent bit-width for the conventional multipliers, both approaches are presented in the same graph. In most cases, the HUB multipliers are considerably faster than their equivalent conventional multipliers. This speedup is presented in Fig. 11. It can be observed that the speedup reaches up to 40% for short mantissas, whereas it is greater than 10% for long mantissas; however, the HUB approach is around 5% slower for 26 and 34 mantissa bit-widths.

In general, significant improvements are achieved on the FPGA implementation of both FP adders and multipliers when using HUB formats. It is important to highlight that these improvements are simultaneously obtained in both area and delay, although these two characteristics are inversely proportional. These enhancements are more noticeable for the FP adders and for the FP multipliers when the mantissa bit-width is less than 30 bits. Since these improvements are achieved due to the simplification of the operations, a reduction in the power consumption of the HUB unit is also expected. We will try to confirm this prediction in a future study.



(a) HUB approach



(b) Conventional approach

Fig. 8. Area used by FP-multipliers under both approaches

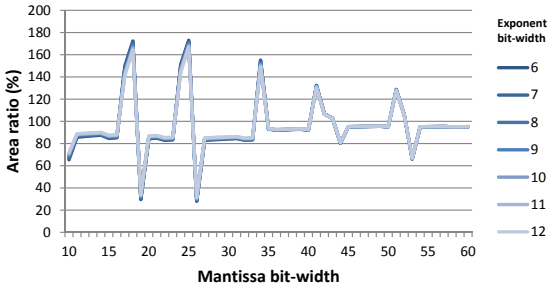


Fig. 9. Ratio of multiplier areas under both approaches (HUB-FP/conventional)

V. Conclusions

In this brief, we investigated the use of HUB-FP formats to enhance the implementation of DSP applications on FPGA. Firstly, the statistical equivalence of the accuracy of HUB and standard FP computations was empirically verified by the implementation of FIR filters. It was shown that although the values of the results are different, the SNR of both approaches was practically the same. The advantages of implementing HUB-FP arithmetic units on FPGA instead of standard ones were measured for addition and multiplication, which are the key operations on most DSP applications. The elimination of the rounding logic can significantly reduce both area and delay. We studied this improvement for a wide range of mantissa and exponent bit-widths and showed that that HUB units were clearly superior in most of the cases analyzed. Furthermore, due to the nature of the improvement, most current soft or hard cores could be easily enhanced by using the proposed approach. We should also note that several patent applications have been filed regarding several HUB circuits.

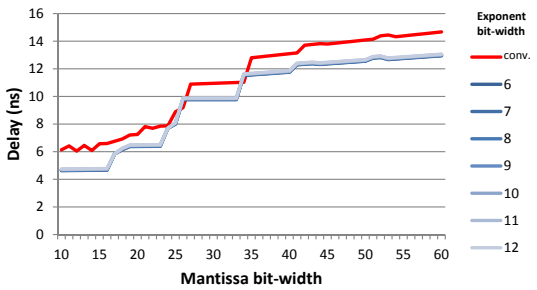


Fig. 10. Delay of FP-multiplier under both approaches

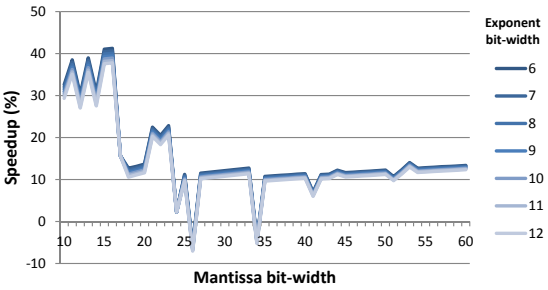


Fig. 11. Speedup of HUB-FP multipliers versus conventional ones

References

[1] F. de Dinechin and B. Pasca, "Designing custom arithmetic data paths with FloPoCo," *Design Test of Computers, IEEE*, vol. 28, no. 4, pp. 18–27, July 2011.

[2] Xilinx, "LogiCORE IP floating-point operator v7.0, product guide, PG060," *www.xilinx.com/support/documentation*, 2014.

[3] Altera, "Arria10 device overview," *https://www.altera.com*, 2015.

[4] M. Langhammer and B. Pasca, "Design and implementation of an embedded FPGA floating point DSP block," in *Computer Arithmetic (ARITH), 2015 IEEE 22nd Symposium on*, June 2015, pp. 26–33.

[5] IEEE Task P754, *IEEE 754-2008, Standard for Floating-Point Arithmetic*, Aug. 2008.

[6] A. Gaffar, O. Mencer, and W. Luk, "Unifying Bit-Width Optimisation for Fixed-Point and Floating-Point Designs," in *IEEE Symp. on Field-Programmable Custom Computing Machines*, 2004, pp. 79–88.

[7] D. Boland and G. Constantinides, "A scalable precision analysis framework," *Multimedia, IEEE Trans.*, vol. 15, no. 2, pp. 242–256, Feb 2013.

[8] A. Ehliar, "Area efficient floating-point adder and multiplier with IEEE-754 compatible semantics," in *Field-Programmable Technology (FPT), 2014 International Conference on*, Dec 2014, pp. 131–138.

[9] M. Langhammer, "Floating point datapath synthesis for FPGAs," in *Field Programmable Logic and Applications, 2008. FPL 2008. International Conference on*, Sept 2008, pp. 355–360.

[10] J. Hormigo and J. Villalba, "New formats for computing with real-numbers under round-to-nearest," *Computers, IEEE Transactions on*, vol. PP, no. 99, pp. –, 2015, early access.

[11] P. Kornerup, J.-M. Muller, and A. Panhaleux, "Performing arithmetic operations on round-to-nearest representations," *Computers, IEEE Trans. on*, vol. 60, no. 2, pp. 282–291, Feb 2011.

[12] J. Hormigo and J. Villalba, "Optimizing DSP circuits by a new family of arithmetic operators," in *Signals, Systems and Computers, Asilomar Conference on*, Nov 2014, pp. 871–875.

[13] S. D. Muñoz and J. Hormigo, "Improving fixed-point implementation of QR decomposition by rounding-to-nearest," in *Consumer Electronics (ISCE 2015), 19th IEEE Int. Symp. on*, June 2015, pp. 1–2.

[14] J. Hormigo and J. Villalba, "Simplified floating-point units for high dynamic range image and video systems," in *Consumer Electronics (ISCE 2015), 19th IEEE Int. Symp. on*, June 2015, pp. 1–2.

[15] ——, "Measuring improvement when using HUB formats to implement floating-point systems under round-to-nearest," *Very Large Scale Integration (VLSI) Systems, IEEE Transactions on*, vol. PP, no. 99, pp. 1–9, 2015, early access.