Fast HUB Floating-point Adder for FPGA

Julio Villalba, Javier Hormigo and Sonia González-Navarro

***Abstract*—Several previous publications have shown the area and delay reduction when implementing real number computation using HUB formats for both floating-point and fixed-point. In this paper, we present a HUB floating-point adder for FPGA which greatly improves the speed of previous proposed HUB designs for these devices. Our architecture is based on the double path technique which reduces the execution time since each path works in parallel. We also deal with the implementation of unbiased rounding in the proposed adder. Experimental results are presented showing the goodness of the new HUB adder for FPGA.**

***Index Terms*—Floating-point (FP), field-programmable gate array (FPGA), half-unit biased (HUB) format, addition, unbiased rounding.**

I. Introduction

When specific Floating-Point (FP) is needed for some applications, Field-Programmable Gate Array (FPGA) design allows to meet the required features. Thus, nowadays many systems are not implemented in ASIC but using FPGAs [1]. Traditionally FPGA implementations use fixed-point arithmetic mainly because many of the Digital Signal Processing (DSP) applications tolerate error precision providing low-cost implementation at the same time. However, in the last years a fast growth of floating-point implementations and studies has been seen in the literature [2], [3], [4], [5]. There are more DSP applications implementing complex algorithms which require extended dynamic range and higher precision. The drawback is that the implied implementations on FPGA are costlier than their fixed-point counterparts. However, there are some promising researches proposing designs of adders and multipliers (key units on most DSP applications) which use other format than the IEEE-754 standard for binary floating point with lower cost [6], [7], [8], [5]. Specifically the implementations on FPGA of an adder and multiplier are analyzed in [5] having simultaneously less area and delay (compared to conventional implementations). In this brief, we use the same format as that used in [5] named HUB format.

HUB is the acronym of Half-Unit-Biased format and it is based on shifting the standard numbers by half unit in the last place (ULP). Some of its important features are that the two's complement is carried out by bit-wise inversion, the round-to-nearest is performed by simple truncation, and requires the same number of bits for storage as its conventional counterpart for the same precision [9]. Thanks to those characteristics, it is possible to eliminate the rounding logic which significantly reduces both area and delay [9], [5].

This work has been supported in part by the Spanish projects: TIN2016-80920-R, JA2012 P12-TIC-1692, JA2012 P12-TIC-1470.

The authors are with the Department of Computer Architecture, Universidad de Málaga, Spain, E-29071.E-mail: {jvillalba,fjhormigo, sgn}@uma.es

A floating-point HUB number, x, has a similar representation as the binary floating-point standard [10]. The difference between both representations is in the mantissa. A normalized HUB mantissa, M_x, has both a value $1 < M_x < 2$ and an implicit least significant bit (ILSB) which equals one [9].

The efficiency of using HUB formats for floating-point approach has been demonstrated in several works, such as [9] and [5]. In [9], the authors analyze the benefits of using HUB format for floating-point adders, multipliers, and converters from a quantitative point of view for ASIC implementation. The HUB adder proposed in [9] is optimized for FPGA devices in [5] achieving excellent results. In this paper we present new architectures based on the double-path technique that speed up the previous results of HUB addition in FPGA devices presented in [5]. Moreover, the problem of bias when rounding in the previous architectures is overcome by adapting the proposal in [11].

The rest of the paper is organized as follows: in Section II we deal with the design of the proposed adder. Next in Section III, we show how to improve the architecture to include unbiased rounding. The implementation results are presented in Section IV. Finally, in the last section, we give the summary and conclusion.

II. Floating-point HUB adder based on double-path

The simple design of floating-point HUB adder (a single-path architecture) presented in [9] and studied for FPGA in [5] is shown in Fig. 1. This architecture carries out the addition of two floating-point HUB numbers with rounding-to-nearest and tie-away-from-zero (i.e. when the result is exactly in the middle of two exactly representable numbers, the higher magnitude is selected). In comparison with the counterpart floating-point architecture for conventional numbers, the HUB architecture does not use the round-to-nearest circuit as well as the circuits required for calculating sticky bit, since the HUB round-to-nearest is carried out simply by truncation. As a consequence, an important reduction both in area and delay is achieved as proved in [5] and [9].

In this section, we propose a new HUB adder architecture using the double path approach to improve the speed of the HUB adder of Fig. 1 when implemented on FPGA. In the single-path HUB architecture the critical path goes through two variable shifters (shadowed in Fig. 1). It is a well known fact that *variable* shifters perform very poorly in FPGA [6], [7], [8], being responsible for from 25% to more than 50% of the total delay. Therefore, if at least one of the variable shifter is removed from the critical path, the gain on speed may be very significant. This is accomplished using a double-path approach [12]. The double-path approach separates these variable shifters so that they belong to two parallel

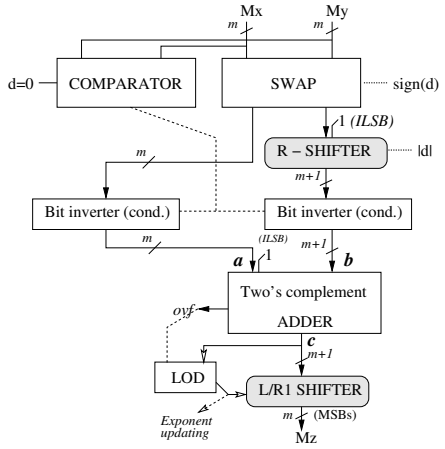


Fig. 1. Floating-point HUB adder proposed in [9]

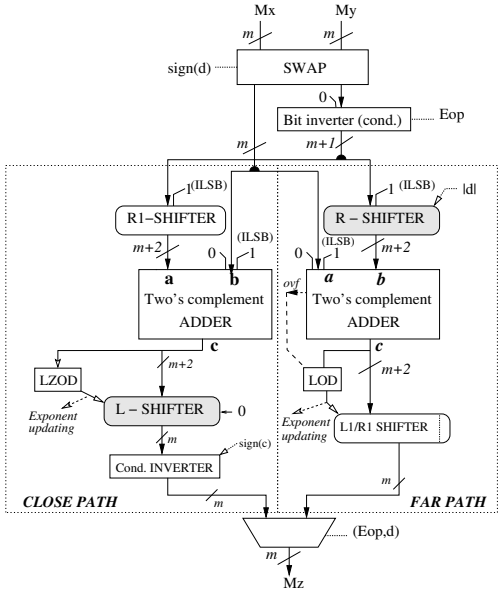


Fig. 2. Double-path HUB adder

disjoint paths [13], and therefore, a significant reduction of the latency is expected. On the other hand, using two disjoint paths implies duplicating some logic (such as the mantissa adder) and increasing control logic. As a consequence, an area increase is also expected.

The proposed double-path HUB adder is presented in Fig. 2. It has a global structure similar to that of the classic double-path implementation (see [13]), except that the circuits required for rounding have been eliminated to handle floating-point HUB numbers. The left adder forms the Close path, and the right adder forms the Far path, both including only one variable shifter (R-SHIFTER in the Far path and L-SHIFTER in the Close path, shadowed in Fig. 2). The variable right shifter, R-SHIFTER of Fig. 1, is now placed in the Far path and the variable left- and one-bit right shifter L/R1-SHIFTER of Fig. 1 is now placed in the Close path.

Apart from the aforementioned differences, when comparing the double-path approach (Fig. 2) with the previous single-path approach (Fig. 1) we can see that the comparator of Fig. 1 has been prevented and the inversion of one of the operands (if required) is performed before shifting in the double-path architecture (Fig. 2). We can also see that there is a fixed R1-SHIFTER in the Close path and a fixed L1/R1-SHIFTER in the Far path. These modules are explained later and the logic needed to implement them is very simple (similar to one multiplexer).

Before entering in Close and Far path, the swap module of Fig. 2 places the mantissa of the highest exponent in the left output depending on the sign of the difference of exponents (d). This ensures that the mantissa of the greatest number is located at the left output of the swap module except when the difference of exponent is 0 ($d = 0$), in which case the position of the greatest operand is unknown. The top conditional inverter of Fig. 2 inverts the operand if the Effective Operation (Eop) is a subtraction only. Next, the operands arrive to both the Close and Far paths.

Note that in both paths the Implicit Least Significant Bit (ILSB) of the operands is incorporated as the Least Significant Bit (LSB) of the operation in the suitable modules in the architecture (R1-SHIFTER, R-SHIFTER and both adders). Note, as well, that sign extension (bit 0) is incorporated in the top conditional inverter and at the second input of the two adders.

Let us deal with the Close and the Far paths separately:

- **The Close path**. The Close path is intended for effective subtraction of two floating-point HUB numbers with a difference between the exponents less than two ($d = 0, 1$). In this case, aligning the input operands is almost not required, but normalizing the results may required large left shifting. The R1-SHIFTER module in the Close path of Fig. 2 is used when $d = 1$ to perform a fixed shift of one position to the right (if $d = 0$ it allows the data to go through without shifting).
  The output c of the adder is suitably shifted by the variable left-shifter, L-SHIFTER. The number of bits to be shifted is calculated by the module LZOD which detects the leading 0 or 1 depending on the sign of the output c. Note that the prevention of the comparator of Fig. 1 makes a negative output c possible. This happens when the input operands have the same exponent (d=0) and the swap module place the mantissa of the lowest operand in the left output. Thus, a conditional inverter is required at the end of the Close path (Cond. Inverter module in the bottom of Fig. 2). The left shift is filled with 0's (for implementation of the unbiased rounding for the tie case this pattern changes slightly as we will see in Section III).
- **The Far path**. This path covers the rest of the cases (i.e. subtraction with $d > 1$ and addition). In this case, aligning the input operands may be required but, at most, one-bit shifting may be required for normalization. Therefore, a variable shifter is required at the input of the adder for alignment (R-SHIFTER in the Far path of Fig. 2). The final L1/R1-SHIFTER module corrects a possible overflow in the addition operation (one-bit right shift) or

the case of having a pattern 0.1xxx in the result of a subtraction operation (one-bit left shift). Note that for the subtraction of the mantissas of two HUB numbers, it is not necessary to calculate the sticky bit since it is always one due to the ILSB of the second operand (the right-shifted operand), and thus, it always involves an incoming carry to the adder (sticky=1).

After the Close and Far paths, the final multiplexer of Fig. 2 selects the result of either the Close or the Far path depending of the effective operation (Eop) and the difference of exponents (d).

III. Unbiased round-to-nearest

When the result of an operation is just in the middle of two exactly representable numbers (tie case), rounding may be performed in any direction. However, the careless election of this direction may produce a statistical bias in the results. To avoid annoying statistical anomalies of some applications due to this bias, we should round either up or down with similar probability for the tie case. In [11] a deep analysis of the three source of bias for HUB-FP addition is presented. A bias can be produced under any of the next operations [11]:

1) aligned addition ($d = 0$),
2) aligned subtraction ($d = 0$),
3) subtraction with difference of exponents of one ($d = 1$).

In [11] the algorithms to reduce, or even, prevent the bias for the tie case are also proposed. Next, we introduce this solution to our double-path architecture.

The first source of bias (aligned addition) occurs only in the Far path (which is devoted to perform subtractions with $d > 1$ and additions), whereas the other two always happen in the Close path (which is devoted to perform subtractions with $d = 0, 1$). Hence, the adaptation of the solution proposed in [11] to our architecture is almost straightforward. Fig. 3 shows the modifications required in the architecture of Fig. 2 to support the unbiased solution. The differences with Fig. 2 are the two modules entitled "unbiased" (shadowed in Fig. 3). These modules involve a very simple logic such that the hardware cost and the penalty time are very small as can be seen in the experimental results presented in Section IV (details of the logic of these modules can be found in [11]).

On the one hand, the logic to prevent the bias for the aligned addition is included at the output of the Far path. On the other hand, the bias for the aligned subtraction is, in fact, prevented in the original double-path architecture and only the logic to eliminate the bias for the third source is needed. In this case, depending on the LSB of the result, the pattern 0111... is inserted in the left shifting (more details can be found in [11]).

We should note that in the simple HUB adder of [5] (Fig. 1) it is possible to prevent the bias only for the case of aligned addition (if the corresponding logic is appended). To eliminate the other two sources of bias a negative results at the output of the adder is required [11]. This is not possible in the simple path approach due to the fact that the comparator in the architecture of Fig. 1 ensures a positive result at the output.

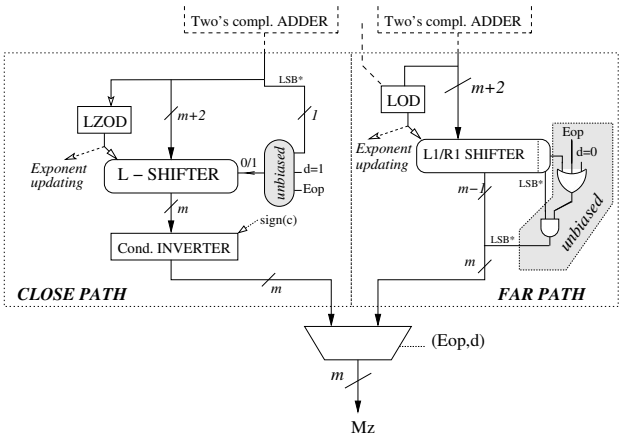


Fig. 3. Unbiased Double-path HUB adder

IV. FPGA Implementation Results

In this section, we analyze and compare the main results of the FPGA implementation of the new floating-point HUB double-path adder (denoted as DPath in the sequel) with the ones corresponding to the HUB-FP single-precision adder (denoted as SPath in the sequel) presented in [5]. All our implementations were described in VHDL such that the bit widths of the mantissa and exponent were configurable. To facilitate the comparison, all designs are fully combinational but they do not support special cases or subnormal. All the architectures have been synthesized using Xilinx ISE 14.3 targeting Xilinx Virtex-6 FPGA xc6vlx240t-1 for a wide range of formats. Specifically, we have synthesized all FP formats with the size of the mantissa ranging from 10 to 60 bits and the size of the exponent from 6 to 12 bits (as in [5]).

Fig. 4, Fig. 5, Fig. 6 and Fig. 7 show the analysis performed over the two studied adder architectures (without bias prevention) controlling when the mantissa bit width varies. Each architecture uses lines printed in the same color and line style, whereas each exponent bit width is represented in lines with different color intensity. The label "DpathX" (in the legends of the figures) represents the architecture with double-path and the number X indicates the bit width of the exponent. Similarly "SpathX" corresponds to the basic single-path architecture [5]. To facilitate the comparison, the mean of all exponent bit width for each architecture is also shown using the corresponding line stile but with red color. Fig. 4 shows the delay of the critical path for the analyzed architectures. It is easily observed that the influence of the exponent bit width is much more important in the single-path architectures than in the double-path ones. Consequently, the delay reduction is very significant in all cases, specially when the exponent bit width rises.

Fig. 5 shows the area in LUTs for all the analyzed architectures. In this case the exponent bit width has less influence in both architectures. As expected, the double-path architecture requires significantly more area than the single-path one.

Fig. 6 and Fig. 7 provide, respectively, the power and energy to perform one calculation for all the analyzed architectures. Both power and energy have been estimated using Xilinx

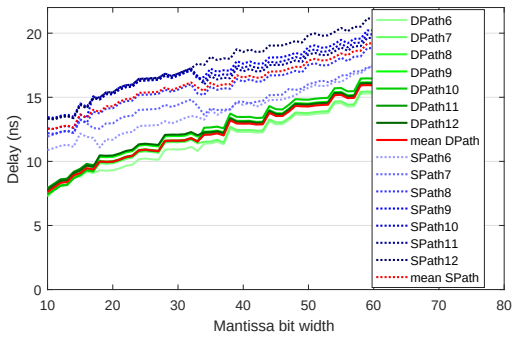


Fig. 4. Delay of adders: double-path (DPath) and simple-path [5] (SPath)

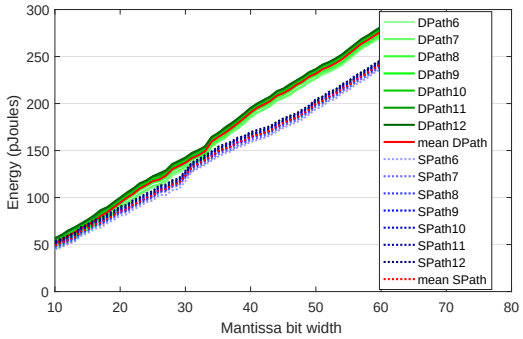


Fig. 7. Energy of adders: double-path (DPath) and simple-path [5] (SPath)

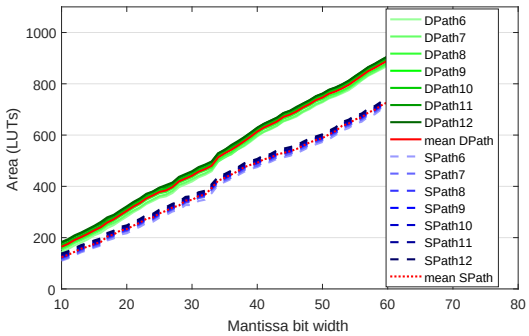


Fig. 5. LUTs used for implementing adders: double-path (DPath) and simple-path [5] (SPath)

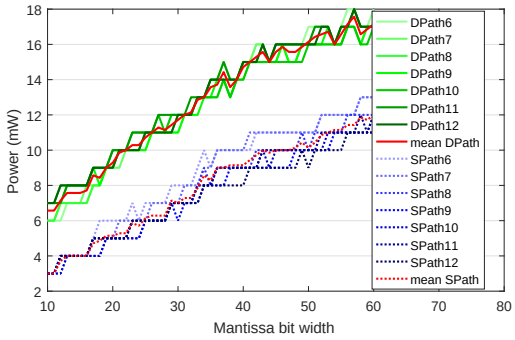


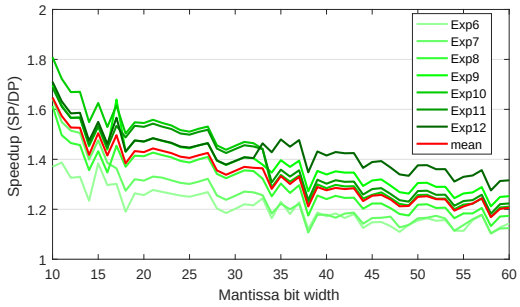
Fig. 6. Power of adders: double-path (DPath) and simple-path [5] (SPath)

Power Estimator (XPE) considering the maximum achievable frequency. As expected, the power of "DpathX" adders is significantly greater than that of the "SpathX" adders. The increase in energy of "DpathX" is very low for short mantissas and increases slowly for larger mantissas (up to 20%). We can also observe that the shape of the curve of Fig. 7 is similar to that of the area (Fig. 5), but the relative increase in energy (13% on average) is less than the relative increase in area (26% on average). Thus, the energy consumption shows a very good behaviour.

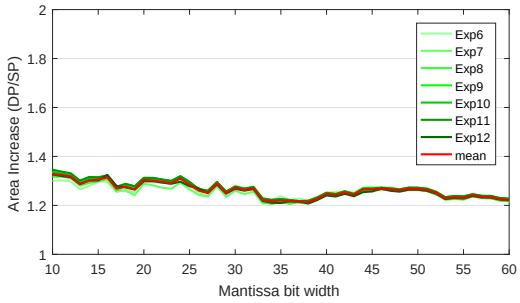
To facilitate the comparison between both approaches, Fig. 8 represents some typical rates. Fig. 8a shows the speedup when using the proposed double-path architecture. It has a significant variability, but in general the speedup increases when the exponent bit width increases and asymptotically decrease when the bit width of the mantissa increases. Considering all implemented combination of bit width, the speedup ranges from 80% to 10% with a mean of 34%. For a typical 24-bit mantissa, the speedup ranges from 25% to 50%. On the other hand, considering the mean of all exponent, the speedup ranges from 65% for 10 bits and 20% for 60 bits. Similarly, Fig. 8b represents the increment of area required by the double-path architecture respect to the single-path one. In this case the variation when changing the exponent bit width is irrelevant. The area increment ranges between 35% and 21% with a mean of 26%. Finally, Fig. 8c shows the Energy-Delay-Product (EDP) in relative terms (Dpath/Spath). The EDP is a fused metric used to compared low power designs. The lower the EDP is, the better. In Fig. 8c, it is observed that the ratio is generally lower than one, which means our proposal performs better. Thus, in general, the area and energy cost of the proposed double-path architecture is very reasonable (26% and 13% on average, respectively) related to the gain in speed (34% on average) which means good EDP decrease (15 % on average).

Nonetheless, we must point out that the changes in the global results of a system when the proposed double-path adder replaces the single-path one may vary drastically since it depends on many factors (such as the relative weight of the adder in the maximum delay and global area, percentage of resources utilized in the device, etc.). Hence, the convenience of using the proposed adder should be analyzed on each case. The ideal case would be when the adder is on the critical path and slightly influences on the area of the system. Then, an important delay reduction with a negligible cost is expected. However, if adders have a great influence on the global area and the initial system has a high occupancy of the device, the resulting delay may be even worse than the original one due to routing problems.

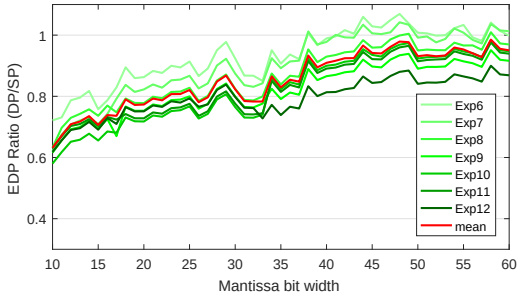
We have also studied the cost of implementing the unbiased rounding described in Section III. The architecture Dpath+ refers to the elimination of the bias only in the far path whereas, Dpath++ refers to the elimination of all sources of bias [11]. Fig. 9 shows the relative cost of implementing this logic in the proposed double-path architecture. To facilitate



(a) Speedup

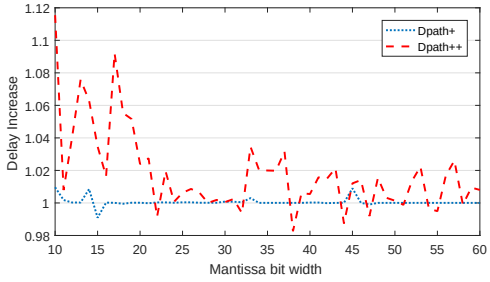


(b) Area increase

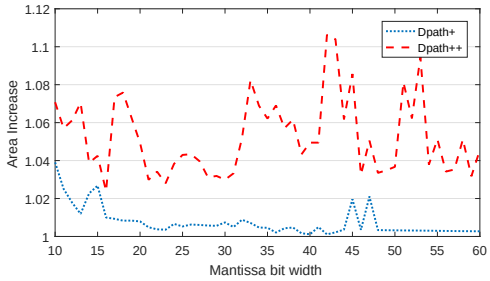


(c) EDP Ratio

Fig. 8. Speedup, area and EDP ratios between the double-path adder and the simple-path adder [5]



(a) Delay increase



(b) Area increase

Fig. 9. Delay and area increase when adding the logic for unbiased rounding to the double-path adder

the comprehension of the plot, we only show the mean value among the different exponent bit width. Fig. 9a represents the delay increment, in relative terms, for both Dpath+, and Dpath++ respect to the Dpath architecture. It is clear that the delay increase of Dpath+ is negligible. For Dpath++ is below 3% for most of the cases, with a maximum of 12% and a mean of 1.8%. The area is a little more affected as we can observe in Fig. 9b. For Dpath++ the area increase ranges between 10.6% and 2.4% with a mean of 5%. This increase is much lower for Dpath+ which is below 1% for most of the cases.

V. Conclusion

In this paper, we design a double-path based HUB-FP adder to speed up the computation on FPGA devices. Compared to the single-path adder implementations shown in [5], the cost in area and energy of the double-path approach is reasonable, giving a substantial speedup, especially for short bit-width mantissas. In addition, we analyze the impact of adding the hardware required to produce unbiased addition in the proposed architecture. In this case, with a slight increment in the delay and area, the new architecture prevents the three sources of bias.

References

[1] J. J. Rodriguez-Andina, M. D. Valdes-Pena, and M. J. Moure, "Advanced features and industrial applications of FPGAs: A review," *IEEE Trans. on Industrial Informatics*, vol. 11, no. 4, pp. 853–864, aug 2015.

[2] L. Zhuo and V. K. Prasanna, "High-performance designs for linear algebra operations on reconfigurable hardware," *IEEE Transactions on Computers*, vol. 57, no. 8, pp. 1057–1071, aug 2008.

[3] C. H. Ho, C. W. Yu, P. Leong, W. Luk, and S. J. E. Wilton, "Floating-point FPGA: Architecture and modeling," *IEEE Trans. on Very Large Scale Integration (VLSI) Systems*, vol. 17, no. 12, pp. 1709–1718, 2009.

[4] G. Caffarena and D. Menard, "Quantization noise power estimation for floating-point dsp circuits," *IEEE Transactions on Circuits and Systems II: Express Briefs*, vol. 63, no. 6, pp. 593–597, jun 2016.

[5] J. Hormigo and J. Villalba, "HUB floating point for improving FPGA implementations of DSP applications," *IEEE Transactions on Circuits and Systems II: Express Briefs*, vol. 64, no. 3, pp. 319–323, mar 2017.

[6] B. Catanzaro and B. Nelson, "Higher radix floating-point representations for FPGA-based arithmetic," in *13th IEEE Symposium on Field-Programmable Custom Computing Machines (FCCM'05)*, 2005, pp. 161 – 170.

[7] A. Ehliar, "Area efficient floating-point adder and multiplier with IEEE-754 compatible semantics," in *Proc. International Conference on Field-Programmable Technology (FPT'14)*, dec 2014, pp. 131–138.

[8] M. Langhammer, "Floating point datapath synthesis for FPGAs," in *Proc. International Conference on Field Programmable Logic and Applications (FPL'08)*, sept 2008, pp. 355–360.

[9] J. Hormigo and J. Villalba, "Measuring improvement when using HUB formats to implement floating-point systems under round-to-nearest," *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, vol. 24, no. 6, pp. 2369–2377, 2016.

[10] *IEEE Standard for Floating-Point Arithmetic*, IEEE Std. 754, 2008.

[11] J. Villalba, J. Hormigo, and S. Gonzalez-Navarro, "Unbiased rounding for HUB floating-point addition," *IEEE Transactions on Computers*, vol. Early access, no. 99, pp. 1–1, 2018.

[12] Farmwald and P. Michael, "On the design of high performance digital arithmetic units," Ph.D. dissertation, Stanford University, 1981.

[13] M. D. Ercegovac and T. Lang, *Digital Arithmetic*. Morgan Kaufmann, San Francisco, 2004.