*IOSR Journal of VLSI and Signal Processing (IOSR-JVSP)*
*Volume 11, Issue 3, Ser. I (May. - June. 2021), PP 26-31*
*e-ISSN: 2319 – 4200, p-ISSN No. : 2319 – 4197*
*www.iosrjournals.org*

# HUB Floating-point Addition Using Unbiased Rounding

Akbar Shaik[1], Dr.SK.Fairooz[2]
*[1]M.Tech Scholar, ES, ECE, Sreyas Institute of Engineering & Technology, Hyderabad, India,*
*[2] Dr SK.Fairooz, Associate Professor, Dept of ECE, Sreyas Institute of Engineering & Technology, Hyderabad.*

***Abstract:*** *Half-unit-biased (HUB) is a new design that is based on the displacement of the digits represented by half the unit last performed. This arrangement makes the two's complement and round operations closest by avoiding any carry propagation. This saves energy, time and area consumption. Given that the IEEE floating point standard uses unbiased rounding as the default mode, this feature is also desirable for HUB approaches. In this article, we study unbiased rounding for HUB floating point addition both within independent operation and within FMA. We show two different options to eliminate bias by eliminating the sum results either partially or completely. Implementation results of the proposed architecture to help designers decide what their best option.*

***Keywords:*** *HUB format, unbiased rounding, Floating point , IEEE.*

## I. Introduction

Today, special purpose systems are developing due to the emergence of new application areas, such as machine learning [1], [2], Internet of Things [3], green computing [4], [5], [4]. and others. Many of these applications require a temporal point (FP) calculation that is traditionally very expensive.

As a result, in recent years, there has been increased interest in finding new FP approaches that achieve better numbers when implementing these specific systems. Probably the most radical FP approach yet proposed is the Flexible Floating Point Format (FFP) [7] initiated by Synopsis, which is completely different from the IEEE FP standard [8]. FFP changes the representation of the mean and the exponent, uses truncation only as the rounding mode, and does not generalize among other changes.

A more subtle modification is proposed using the HUB approach [9]. HUB is abbreviated to half-unit biased format and is based on changing standard numbers by the last unit (ULP). This representation format has important features [9] such that the two are complemented by a bitwise inversion, the nearest rounding is done by simple truncation and the double rounding error [10] never occurs. Furthermore, the same number of bits are required for storage as it has the same accuracy as its traditional counterpart. In [11], the authors discuss the benefits of using the HUB format for FP additives, multipliers, and converters.

Today, special purpose systems are developing due to the emergence of new application areas, such as machine learning [1], [2], Internet of Things [3], green computing [4], [5], [4]. and others. Many of these applications require a temporal point (FP) calculation that is traditionally very expensive.

As a result, in recent years, there has been increased interest in finding new FP approaches that achieve better numbers when implementing these specific systems. Probably the most radical FP approach proposed so far is the flexible floating point format (FFP) [7] launched by Synopsis, which is completely different from the IEEE FP standard [8]. FFP changes the representation of the mean and the exponent, uses truncation only as the rounding mode, and does not generalize among other changes.

A more subtle modification is proposed using the HUB approach [9]. HUB is abbreviated to half-unit biased format and is based on changing standard numbers by the last unit (ULP). This representation format has important features [9] such that the two are complemented by a bitwise inversion, the nearest rounding is done by simple truncation and the double rounding error [10] never occurs. Furthermore, the same number of bits are required for storage as it has the same accuracy as its traditional counterpart.

In [11], the authors discuss the benefits of using the HUB format for FP additives, multipliers, and converters. In this article, we analyze various sources of bias that exist in the previous HUB additive architecture and propose two new architectures to avoid these sources of bias. We also provide an error analysis and comparison of the implementation of different additives.

The remainder of the document is organized as follows: Section II summarizes the basic principles of HUB representation and reviews the basic HUB additive [11]. Section III is devoted to analyzing various sources of bias, while Section IV proposes the new architecture to perform partial or total unbiased rounding. Section V discusses the implementation of an FMA unit and proper rounding for results. Finally, in the last section, the conclusion.

## II. Hub Format And Adder For Fp Numbers

### II A. HUB format for FP numbers

The mathematical foundations and in-depth analysis of the HUB format can be found in [9]. In this section, we summarize the HUB format defined in [9] and characterize it for standard FP HUB numbers. An FP HUB number is similar to a normal one but it follows the HUB format. Therefore, the only difference compared to the FP binary standard [8] is the format for the mean. Without any loss of generality, in this article, we use FP HUB operations with a generalized meaning in radix - 2. Define x as the number of FP radix - 2 HUB.

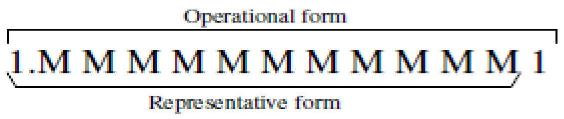


**Fig. 1.** Representative versus operational form

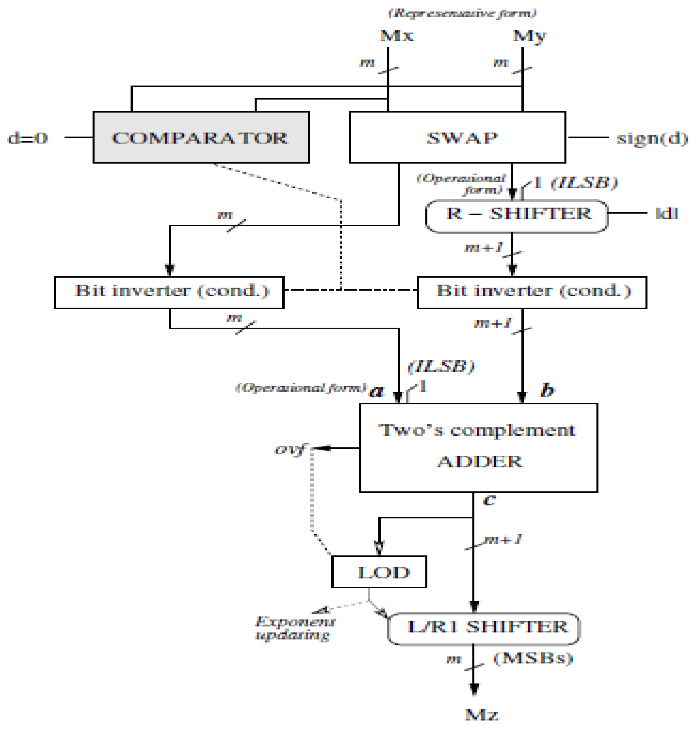


**Fig. 2.** Proposed HUB Floating-point adder (architecture A)

### II B. Basic adder for HUB FP numbers

The simple design of the HUB FP additive is presented in [11] and shown in fig. Compared to the traditional number of counterparts in the FP architecture, the HUB architecture does not use the fixed bit as well as the circuit required for rounding calculation: the closest circuit is only truncated since the HUB rounding is the closest. As a result, a significant reduction in both area and energy consumption is achieved, as demonstrated in [11]. Let A denote the architecture proposed in [11]. Architecture A combines two HUB FP numbers with rounding to the nearest in the case of zero binding. Therefore, the rounding of importance for the linkage case is biased, that is, the rounding is always done in the same direction: up. In the following sections, we analyze in depth when a link condition occurs (or not) and propose two designs to avoid biases to avoid rounding.

## III. Sources Of Bias While Rounding

Some biases can arise under rounding conditions. For HUB numbers, the tie condition occurs when an operation after normalization results in a bit 0 in the ILSB condition and the remaining bits to its right are all 0, that is, the bits removed after sorting are all zeros. Occurs The square brackets () indicate the status of the ILSB.

### III A. Effective addition with d _ 1 (Eop=0)

In this case, the binding condition is never generated because the ILSB is at least 1 bit from the position.

**III B. Effective subtraction with d_ 2 (Eop=1)**

If the result of a subtraction d _ 2 is normalized or requires a left shift of a position [17] (that is, the pattern of the result is 01.xxx ... or 00.1xxx ...). The bind condition is never met in the result because there is at least 1 bit (due to the second operand) beyond the ILSB condition.

**III C. Effective aligned addition (d=0, Eop=0)**

In this case, an overflow always occurs and a correct change of a situation is required. A bind condition occurs if the LSB of the result of the addition is 0.

**III D. Effective aligned subtraction (d=0, Eop=1)**

Let M, V denotes two generalized meanings of two HUB numbers (U; V) with the same exponent. As a result, the result always satisfies the tie condition because the left shift of at least one position is always necessary for normalization, which implies an injection of 0 in the ILSB condition of the normalized result.

**III E. Effective non-aligned subtraction with d=1 (Eop=1)**

If the result of subtraction is already normal, the tie condition is never met because the second operand's ILSB (= 1) is outside the size of the word, which implies at least 1 bit in those cases. If the result subtraction is not normalized and the most significant fractional fraction is 1, the link condition is not met. On the other hand, a tie case is possible if the subtraction result has integer bit 0 and the most significant fraction 0 (that is, the pattern in the result is 00.0xxx ... x (x) 1), in this case, the resulting HUB number. Will always be round (001.01110 (1)), resulting in bias. This is the ultimate source of bias.

## IV. ADDERS FOR UNBIASED ROUNDING

The two new designs proposed in this paper are shown in Fig. 3 and Fig. 4. Architecture A+ (Fig. 3) is designed to perform partial unbiased rounding (only aligned addition case), whereas architecture A++ (Fig. 4) resolves all the previous source of bias, producing unbiased rounding.

**IV A. Architecture for partial unbiased rounding: A+**

The A + architecture shown in Figure 3 avoids bias due to a single source: addition. The way to solve the situation is described in [16] rounding by forcing the LSB of the displaced result.

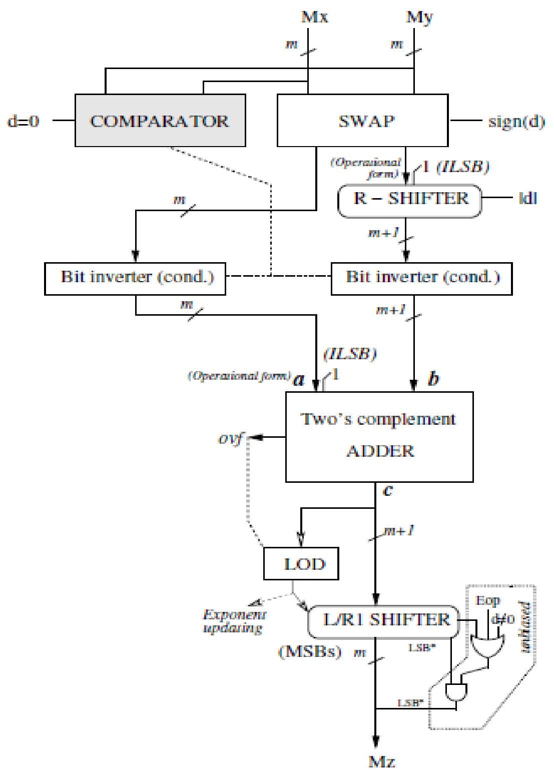


Fig. 3. Partial unbiased HUB FP adder: architecture A+

**IV B. Architecture for unbiased rounding: A++**

The architecture A++ shown in Fig. 4 eliminates the three sources of bias described in previous section. It prevents the significand comparator required in architecture A (Fig. 2). In this way, the result of the operation can be positive or negative (this feature allows preventing the tie cases not supported by A+). A leading zero-one detector (LZOD) is required in the new architecture to detect if the result is negative, as well as the final conditional inverter, in order to obtain the magnitude of the result if this were negative.

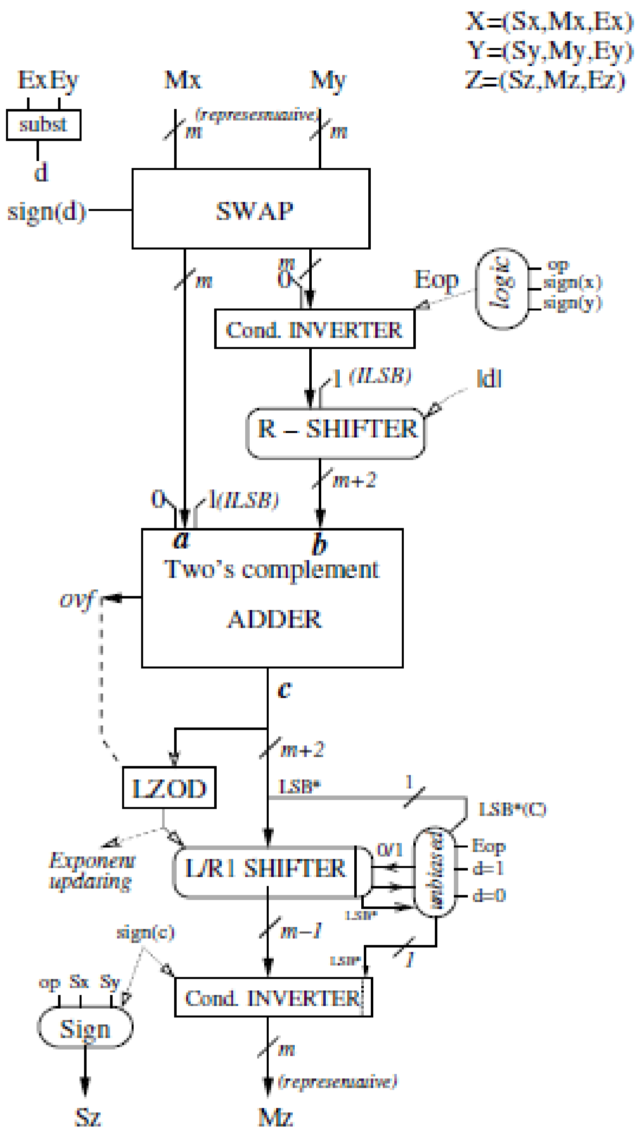


Fig. 4. Unbiased HUB FP adder: architecture A++

## IV. IMPLEMENTATION RESULTS

The A, A + and A ++ architectures were implemented using VHDL and synthesized with the Xilinx simulator. The 32-bit fuel version of these architectures has been synthesized for the same target clock frequencies, and the field and power consumption results have been compiled for comparision.

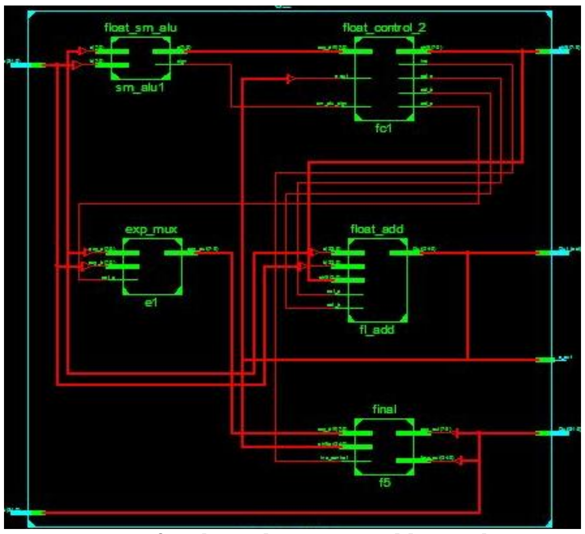


**Fig.5.** RTL view of Unbiased HUB FP adder: architecture A++

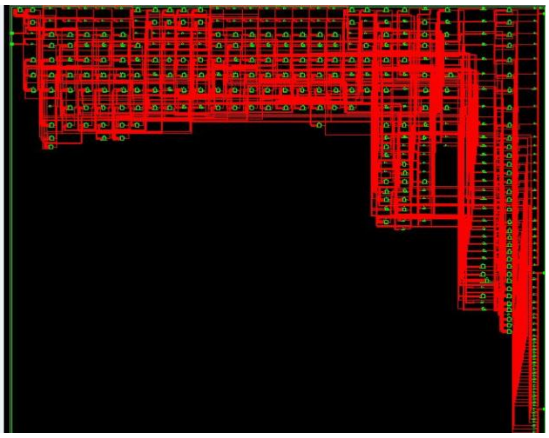
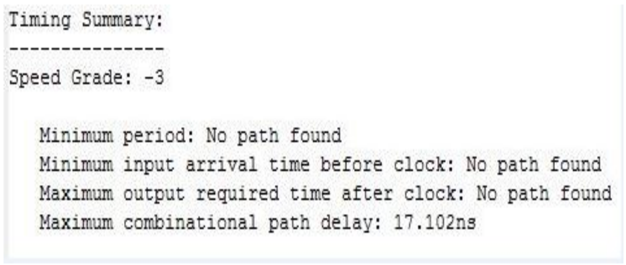


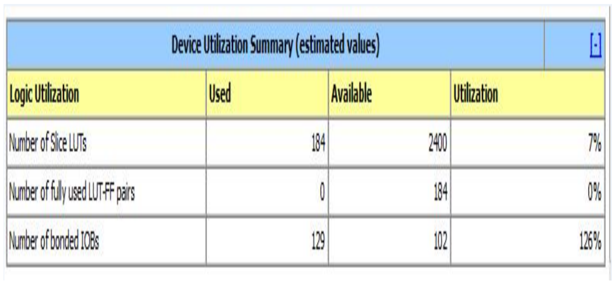
**Fig.6.** Technology schematic of Unbiased HUB FP adder: architecture A++



```
Timing Summary:
---------------
Speed Grade: -3

   Minimum period: No path found
   Minimum input arrival time before clock: No path found
   Maximum output required time after clock: No path found
   Maximum combinational path delay: 17.102ns
```

**Fig.7.** Timing summary of Unbiased HUB FP adder: architecture A++



| Device Utilization Summary (estimated values) | | | |
|---|---|---|---|
| Logic Utilization | Used | Available | Utilization |
| Number of Slice LUTs | 184 | 2400 | 7% |
| Number of fully used LUT-FF pairs | 0 | 184 | 0% |
| Number of bonded IOBs | 129 | 102 | 126% |

**Fig.8.** Device Utilization summary of Unbiased HUB FP adder: architecture A++

## V. Conclusion

In this article, we study the various sources of FP HUB addition and FMA roundness bias. We present and compare two architectures to deal with these sources of bias.

From the results of the implementation, we conclude that Additive A ++ is a solution (with moderate incremental cost compared to Additive A) for applications where prevention of bias is mandatory. On the other hand, a partially unbiased A + architecture should generally be used because it avoids a large bias and involves a negligible increase in hardware costs. Similar solutions can be implemented in the FMA architecture.

## References

[1]. E. Alpaydin, Machine learning : the new AI, 2016.

[2]. M. Kubat, An Introduction to Machine Learning, 2015, no. August.

[3]. J. A. Stankovic, “Research directions for the internet of things,” IEEE Internet of Things Journal, vol. 1, no. 1, pp. 3–9, Feb 2014.

[4]. F. K. Shaikh, S. Zeadally, and E. Exposito, “Enabling technologies for green Internet of Things,” IEEE Systems Journal, vol. PP, no. 99, pp. 983–994, Jun 2015.

[5]. R. Murphy, T. Sterling, and C. Dekate, “Advanced architectures and execution models to support green computing,” Computing in Science and Engineering, vol. 12, no. 6, pp. 38–47, Nov 2010.

[6]. P. P. Pande, A. Ganguly, and K. Chakrabarty, Design technologies for green and sustainable computing systems, P. P. Pande, A. Ganguly, and K. Chakrabarty, Eds. New York, NY: Springer New York, 2013.

[7]. Synopsys, “DWFC Flexible Floating Point Overview,” no. August, pp. 1–6, 2016.

[8]. “IEEE standard for floating-point arithmetic,” IEEE Std 754-2008, pp. 1 –58, 29 2008.

[9]. J. Hormigo and J. Villalba, “New formats for computing with real numbers under round-to-nearest,” IEEE Transactions on Computers, vol. 65, no. 7, pp. 2158–2168, 2016.

[10]. D. R. Lutz and N. Burgess, “Overcoming double-rounding errors under IEEE 754-2008 using software,” in 2010 Conference Record of the Forty Fourth Asilomar Conference on Signals, Systems and Computers. IEEE, Nov 2010, pp. 1399–1401.

[11]. 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.