

Article

An Optimized Floating-Point Unit Set for FPGA-Based DSP: Improving Area, Energy, and Throughput Trade-Offs

Fernando Flores ^{1,2,*} , Juan Portela Queimaño ³ , Jesús Manuel Costa Pazo ³ , María Dolores Valdés-Peña ¹ , Camilo Quintáns Graña ¹  and José Manuel Villapún Sánchez ² 

¹ Department of Electronics Technology, Universidade de Vigo, 36310 Vigo, Spain; mvaldes@uvigo.es (M.D.V.-P.); quintans@uvigo.es (C.Q.G.)

² Department of Digital Hardware, Indra, 36202 Vigo, Spain; jmvillapun@indra.es

³ Department of Signal Processing, Indra, 36202 Vigo, Spain; jportelaq@indra.es (J.P.Q.); jmc_pazo@indra.es (J.M.C.P.)

* Correspondence: fernando.flores@uvigo.es; Tel.: +34-658-599-331

Abstract

Floating-point arithmetic provides the dynamic range that fixed-point lacks for digital signal processing (DSP) algorithms with widely varying operand magnitudes. This work presents a parameterizable floating-point unit set for field programmable gate array (FPGA)-based DSP. The set consists of five units: adder/subtractor, multiplier, multiply-accumulate (MAC), fixed-to-float and float-to-fixed converters. Two architectural choices distinguish the proposed format from IEEE-754: configurable exponent and mantissa widths during synthesis and a $0.f$ significand encoding that reduces corner-case logic at the cost of one additional mantissa bit. The format is therefore IEEE-754-inspired rather than fully compliant: special values (NaN, $\pm\infty$) are not implemented, and overflow and underflow are handled through saturation to predefined constants. The design is implemented in standard VHDL-2008 without relying on high-level synthesis (HLS) tools or vendor-specific primitives, ensuring portability across different FPGA families and application-specific integrated circuits (ASICs). The multiplier and MAC are evaluated in two configurations: inferring DSP blocks or look-up table (LUT)-only, both close timing at 300 MHz on Artix-7 and Kintex Ultrascale devices. The proposed blocks outperform vendor IP Cores and recent academic designs in terms of area-throughput-power (ATP), achieving improvements from 10% to 108%, except for the adder/subtractor, which does not outperform two optimized Xilinx IP cores (HS-R and HS-P) and is therefore included for design coherence rather than as a strict resource improvement over all vendor IPs. All these blocks meet the theoretical error bound, and a representative 200-tap finite impulse response (FIR) filter built from them closes timing at 300 MHz with 76% LUT utilization.

Keywords: digital signal processing; floating-point arithmetic; field programmable gate arrays; IEEE-754



Academic Editors: Yafei Ning, Jie Su and Yanpeng Shi

Received: 27 May 2026

Revised: 25 June 2026

Accepted: 26 June 2026

Published: 30 June 2026

Copyright: © 2026 by the authors. Licensee MDPI, Basel, Switzerland. This article is an open access article distributed under the terms and conditions of the [Creative Commons Attribution \(CC BY\)](https://creativecommons.org/licenses/by/4.0/) license.

1. Introduction

Digital signal processing (DSP) algorithms in radar and communication systems present significant challenges under fixed-point arithmetic [1]. High-order fixed-point Fast Fourier Transforms (FFTs) can suffer from overflow, reduced precision, or a low signal-to-quantization-noise ratio if not designed carefully [2,3]. Matrix operations such as inversion, Cholesky factorization, and singular value decomposition—common in radar and multiple-input multiple-output systems [4–6]—require handling elements with widely

varying magnitudes. Floating-point arithmetic guarantees approximately the same relative precision for a number and its inverse, a property not inherent to fixed-point representations. Operations relying on divisions or inverse calculations, such as recursive least squares or QR-based decompositions, become particularly fragile under fixed-point: datapath widths grow, and convergence can fail if precision is insufficient [7,8]. Floating-point arithmetic provides the dynamic range and precision needed for these calculations.

However, implementing floating-point arithmetic on field programmable gate arrays (FPGAs) or application-specific integrated circuits (ASICs) tends to be costly in terms of hardware resource and power consumption. The main drawbacks of existing FPGA floating-point implementations can be summarized along five recurring axes:

- High look-up table (LUT) and register utilization, dominated by mantissa multiplication, operand alignment, normalization, and rounding.
- Long critical paths through the exponent-comparison, alignment, normalization, and rounding stages, which limit the achievable frequency or force deep pipelines.
- Elevated dynamic power as a direct consequence of the two previous points.
- Limited portability, since many high-performance designs rely on high-level synthesis (HLS) flows or on vendor-specific primitive instantiations and are therefore tied to a particular device family.
- Additional logic required by full IEEE-754 special-value handling and by fixed single/double precision, which wastes resources when the application does not need them.

These drawbacks, rather than the limitations of fixed-point arithmetic discussed above, are the ones that the present work targets. Existing research has mainly addressed these limitations through two approaches: the adoption of advanced high-performance hardware platforms [9,10] and the development of custom floating-point units optimized for FPGA implementations.

In DSP systems, the arithmetic operators that form the computational core of fundamental algorithms, such as FFTs, finite impulse response (FIR) filters, and adaptive filtering techniques, are the adder/subtractor, multiplier, and multiply-accumulate (multiply-accumulate (MAC)) units. As discussed above, achieving efficient FPGA implementations of these operators remains challenging due to the inherent trade-off between hardware resource utilization and computational performance.

This paper presents a custom floating-point arithmetic unit set that explicitly addresses this trade-off through five jointly designed FPGA-optimized pipelined blocks: an adder/subtractor, a multiplier, a non-fused MAC constructed from the previous two units, and fixed-to-float and float-to-fixed conversion blocks. All units share a consistent floating-point representation and pipeline interface, simplifying their integration into larger DSP datapaths. In particular, the inclusion of the conversion units enables seamless interoperability with mixed-signal front-ends, allowing direct interfacing with analog-to-digital converters (ADCs) and digital-to-analog converters (DACs).

The proposed architecture is based on the IEEE-754 floating-point representation, which remains one of the most widely adopted floating-point standards [11], although alternative formats such as POSIT [12,13] have also been investigated. At the representation level, two design choices distinguish the proposed representation format from a strict IEEE-754 implementation. First, the exponent and mantissa widths are configurable at synthesis time, allowing the datapath precision and dynamic range to be adapted to the target application. Second, a $0.f$ significand encoding is adopted, where one additional mantissa bit is exchanged for simpler corner-case handling in the multiplier and adder normalization stages. Beyond these two choices, the proposed format is IEEE-754-inspired but not fully compliant: it does not implement the IEEE-754 special values (NaN, $\pm\infty$) nor

subnormal numbers, and overflow and underflow are resolved by saturation to predefined constants, as detailed in Section 3.2. This design decision is consistent with the guard-logic approach commonly adopted at the system level in DSP datapaths, and it must be taken into account when comparing the proposed units against fully compliant IEEE-754 operators.

Configurable-precision floating-point operators for FPGAs are not new. Tools such as FloPoCo and several vendor intellectual-property (intellectual property core (IP Core)) cores have provided parameterizable exponent and mantissa widths for more than a decade. Consequently, the contribution of this work does not lie in proposing a novel arithmetic algorithm but rather in consolidating a coherent and portable floating-point unit set, thereby characterizing it across two FPGA generations under a uniform area-equivalent metric and quantitatively positioning it with respect to vendor IP cores and recent academic implementations. To avoid overstating novelty, we note that the contributions are of two kinds. The representation-level refinements are incremental with respect to existing parameterizable floating-point units: the explicit $0.f$ significand encoding and the synthesis-time configurable exponent and mantissa widths. The engineering contributions carry the practical value: an architecture definition and single VHDL-2008 description of the complete unit set with no HLS flow and no vendor-specific primitive instantiation (portable across FPGA families and ASIC technologies), the dual DSP-inferred and LUT-only deployment of that same source, and the uniform area-equivalent ATP characterization across two FPGA generations.

The complete floating-point unit set is described exclusively in VHDL-2008, without relying on HLS flows or vendor-specific primitive instantiations. As a result, the design is portable across FPGA families and ASIC technologies that support VHDL-2008 synthesis. Implementations on AMD Xilinx Artix-7 (28 nm) and Kintex UltraScale (20 nm) FPGAs demonstrate that the proposed multiplier, MAC, and conversion units improve the area-throughput-power (ATP) metric with respect to vendor IP cores and recent academic designs by at least 108%, 10%, and 30%, respectively. The proposed adder/subtractor also outperforms previous academic implementations and several Xilinx IP cores. In addition, a 200-tap direct-form FIR filter case study achieves timing closure at 300 MHz under realistic routing pressure (76% LUT utilization), demonstrating that the proposed unit set does not become a routing or timing bottleneck in high-performance FPGA-based DSP systems.

To make the aim and positioning of this work explicit, we can summarize it as follows. The problem addressed is the inherent trade-off between hardware-resource utilization and computational performance that makes efficient floating-point arithmetic on FPGAs costly for DSP datapaths. The main goal is to provide a coherent, parameterizable, and fully portable floating-point unit set (adder/subtractor, multiplier, MAC, and fixed/float converters) that improves the area-throughput-power trade-off for FPGA-based DSP. The main idea used to solve the problem combines an explicit $0.f$ significand encoding that simplifies corner-case logic, synthesis-time configurable exponent and mantissa widths, and a single VHDL-2008 description that is mapped onto either DSP slices or onto pure LUT logic. The novelty is not a new arithmetic algorithm: the units integrate known techniques into a coherent, portable framework and are characterized under a uniform area-equivalent metric across two FPGA generations, which, to the best of our knowledge, has not been reported together for a complete and vendor-independent unit set. The contribution to the field is therefore a portable, openly characterized floating-point unit set that measurably outperforms vendor IP cores and recent academic designs in ATP for the multiplier, MAC, and conversion blocks, together with the dual deployment strategy and the methodology that places all designs on a common comparison footing.

The remainder of this paper is organized as follows: Section 2 reviews the most relevant related work. Section 3 details the proposed architectures and their FPGA implementation,

including the modifications introduced to the IEEE-754 format. Section 4 presents the experimental evaluation. Section 5 describes the representative FIR filter case study, and Section 6 concludes the paper.

2. Related Works

A thorough review of the state of the art has been carried out, analyzing the different solutions proposed in the literature for each of the building blocks considered in this work, namely, the floating-point multiplier, the floating-point adder/subtractor, and the floating-point multiply-accumulate (MAC) unit. In the following subsections, the most relevant characteristics of some of the analyzed works are described, grouped by block type, and a final discussion summarizes the main trends and trade-offs identified across the surveyed designs.

2.1. Floating-Point Multiplier Unit

The design of floating-point multipliers is driven by the large mantissa product, which usually requires either wide integer multipliers or structured decompositions. Existing state-of-the-art works explore Booth encoding, recursive partitioning, and Karatsuba-based methods to reduce partial products and improve scalability across single, double, and higher precisions.

In [14], the authors present an IEEE-754 compliant multiplier for single and double precision, with a product engine based on radix-4 Booth encoding. The sign is computed by an XOR gate, while exponents are summed and bias-adjusted (127 for single precision and 1023 for double precision). Mantissas are expanded to the 1.M mantissa format and multiplied; the use of Booth encoding reduces partial products, yielding a 46-bit product for single precision and 104-bit for double. Exponent adjustment handles overflow and underflow, while normalization and rounding (nearest-even, toward zero, $\pm\infty$) complete the result assembly.

The multiplier in [15] has a straightforward four-step datapath: preprocessing and denormalization, mantissa multiplication, normalization, and rounding/packing. The sign is computed as an XOR of the operand signs. The exponents are added, and the bias is subtracted to form the unnormalized exponent, with handling of underflow and overflow (the exponent is set to zero or to $2^b + 1$, respectively). Mantissas are formed in 1.M format before full-width multiplication. The product is then normalized: if the most significant bit (MSB) is 1, the mantissa is shifted right and the exponent incremented, otherwise it is already aligned. Rounding to nearest-even is applied using guard, round, and sticky (GRS) bits, and overflow from rounding is propagated into the exponent if needed.

The work presented in [16] implements three different LUT-based multipliers for single, double, and quadruple precision. The divide-and-conquer approach first partitions the mantissas into fixed-width segments, which are multiplied in smaller blocks. The results are accumulated using adders and barrel shifts and are recursively extended to higher precisions. On the other side, the Karatsuba/Ofman method restructures cross-terms to reduce submultiplications, at the cost of additional add/sub stages. Finally, a hybrid approach applies the divide-and-conquer partitioning method with Karatsuba at the lower levels to minimize block multipliers. In all cases, exponent addition, normalization, and rounding follow standard IEEE-754 stages.

The study in [17] designs three multipliers, where the pipeline of the datapath is divided into sign, exponent, and significand flows. The sign is computed again as the XOR of the operand signs, while the exponent path adds E1 and E2 and subtracts the bias. Each significand is expanded with its hidden bit and multiplied using one of three alternative schemes. The Optimized Schoolbook Multiplier (OSBM) arranges the partial

products in a regular array, reducing the number of additions by grouping terms efficiently. The Hybrid Karatsuba Multiplier (HKM) applies Karatsuba decomposition only at the top level, splitting operands into halves and reducing the number of large multipliers required. The Hybrid Recursive Karatsuba Multiplier (HRKM) extends this by applying Karatsuba recursively to sub-blocks, further lowering the multiplication complexity for wide mantissas. In all three cases, the raw product is a 2m-bit word passed through normalization (conditional right shift and exponent increment if MSB = 1), and rounding-to-nearest-even using GRS bits.

2.2. Floating-Point Adder/Subtractor Unit

Floating-point adders are often limited by the cost of exponent comparison, mantissa alignment, and post-operation normalization. Reported architectures mainly differ in their treatment of denormal numbers, pipeline depth, and rounding strategy, reflecting the trade-offs between low latency and hardware efficiency in FPGA devices.

For instance, the work in [18] implements a single-precision floating-point adder with full support for denormal inputs, organized as an eight-stage pipeline. The datapath begins with exponent comparison and operand swapping, followed by mantissa alignment using a barrel shifter with GRS bits. A fixed 27-bit two's-complement adder performs the significand operation, and the result passes through a leading-one detector and barrel shifter for normalization. The datapath is completed by rounding to the nearest even (via GRS) and exception handling (overflow, underflow, Not-a-Number (NaN), $\pm\infty$), with each stage being registered to achieve high operating frequency.

In contrast, in [14], the authors present a configurable floating-point unit supporting both single and double precision. It also integrates the adder into a DSP pipeline alongside a Booth multiplier and a MAC unit. A modular, parameterized organization enables the same datapath to operate in either single or double precision while preserving a consistent pipeline structure. The adder follows canonical IEEE-754 stages: exponent alignment by right-shifting the smaller operand, significand addition/subtraction using two's complement, and normalization to restore the 1.M mantissa format. A rounding unit supports multiple modes (nearest-even, toward zero, $\pm\infty$), while a control block manages operation sequencing and exception flags.

In line with the approach of [14], the work in [15] also proposes an adder for single and double precision, partitioned into four stages. Operand comparison identifies the larger input and computes the exponent difference. Denormalization extends the mantissa with implicit and GRS bits, while shifting the smaller operand accordingly. The significand addition produces sum and carry terms, which are normalized by conditional shifting: if a carry occurs, the mantissa is right-shifted and the exponent incremented. If the MSB is zero, leading-one detection determines the required left shifts. Rounding to nearest-even finalizes the result before assembling the IEEE-754 word. The proposed four-stage structure achieves full IEEE-754 addition functionality in a compact form.

2.3. Floating-Point Multiply–Accumulate Unit

MAC units integrate multiplication with iterative addition, and are central to DSP kernels. The most recent designs range from conventional pipelined MACs optimized for real-time DSP to specialized schemes such as systolic arrays or fixed-point accumulation tailored for sparse linear algebra.

The work in [14] integrates a Booth-based multiplier with an accumulation stage, enabling fused multiply–add operations for DSP. Products are accumulated using an adder and FIFO buffering, with results being stored in an output register for iterative accumulation. The architecture basically combines the adder and multiplier described in

the previous sections. The design also supports both single and double precision, with double precision expanding mantissas to 104 bits.

In line with the work of [14], the MAC presented in [19] is a fully pipelined MAC that combines a radix-4 Booth multiplier with a single-cycle accumulator optimized for FPGAs. The multiplier uses Wallace tree compression to reduce partial products, while the accumulator employs bidirectional shift alignment to handle exponent mismatch, a 3:1 compressor to shorten the feedback path, and a three-operand leading-zero predictor to accelerate normalization. These optimizations remove the need for a critical carry-propagation adder.

In contrast, in [20], authors restructure the MAC datapath through a multi-fused multiply-accumulate (MFMA) scheme embedded in a three-dimensional systolic array. Normalization and rounding are postponed until the final accumulation, thereby reducing hardware cost and latency. Mantissas are decomposed into 8-bit-width sub-blocks and spliced into 16-bit and 32-bit floating-point formats via a two-step splicing method. Exponent alignment is performed hierarchically with bias adjustments.

Lastly, the work in [21] targets sparse linear algebra using a deeply pipelined FPGA-oriented MAC. The multiplier employs radix-4 Booth encoding with a Wallace tree combining 5–3 and 3–2 carry-save adders to reduce LUT usage and delay. Accumulation is performed in fixed-point format, with a fixed decimal point position removing the variable shifter from the feedback path. This reduces the loop delay to a single LUT. A LUT-based leading-zero anticipator predicts normalization shifts in parallel with accumulation. The final stage restores IEEE-754 compliance with overflow protection.

2.4. Discussion

Considering all of the above, the state of the art can be organized along three axes relevant for FPGA-based DSP: (i) IEEE-754 compliance and supported rounding modes; (ii) pipeline granularity vs. latency/frequency trade-off; (iii) use of hardened DSP blocks vs. pure LUT-based logic.

For multipliers, full IEEE-754 compliance with implicit $1.f$, multi-mode rounding, and $\pm\infty$ /NaN handling is reported in [14,15,17]; differences lie in the partial-product reduction strategy (Booth, school-book, Karatsuba) and DSP-slice usage. LUT-only designs [16,17] scale to higher precisions but with substantial LUT cost. Surveyed multipliers report an ATP figure on different device families.

Existing floating-point adder architectures range from deeply pipelined implementations with full denormal support [18] to compact reduced-pipeline solutions [15] that trade functionality or critical-path length for lower resource usage, with [14] occupying an intermediate position. The cost and delay of the leading-one detector (or, equivalently, the leading-zero counter/anticipator) that drives post-subtraction normalization is a well-studied problem. A long line of work distinguishes exact leading-zero detection from inexact leading-zero anticipation (LZA), the latter operating in parallel with the adder at the price of a possible one-bit error that must then be detected and corrected. More recent FPGA-oriented studies optimize the leading-zero counter specifically for the LUT/carry-chain fabric [22], and the MAC designs surveyed here already exploit such ideas: [19] uses a three-operand leading-zero predictor and [21] a LUT-based leading-zero anticipator, both to shorten the normalization path. Against this background, none of the reviewed adder designs reports a strategy to bound the cost of the leading-one detector after significant subtraction, despite this block being one of the dominant contributors to resource utilization within the unit. The present work addresses this gap pragmatically by restricting the priority encoder to the most significant half of the result vector, as justified in Section 3,

which halves the encoder width without loss of precision rather than introducing an inexact LZA and its associated correction logic.

Two primary philosophies coexist for MAC design: chained multiplier-adders [14] and domain-specific variants, such as MFMA [20] or fixed-point accumulation [21]. The former prioritizes modularity, often employing fused designs [19–21] that postpone rounding and normalization to reduce both error and the critical path. In contrast, the latter trades generality for efficiency in specific kernels. The MAC presented in this work follows the first philosophy.

3. Design of the Floating-Point Units

Before detailing the proposed floating-point units, the floating-point representation and the corresponding rounding and exception-handling policies are defined.

3.1. Additions to the IEEE-754 Standard for Floating-Point Arithmetic

The IEEE-754 standard, on which the floating-point format used in this work is based, represents a value x in normalized form as

$$x = (-1)^s \cdot 2^{e-b} \cdot (1.f), \quad (1)$$

where s denotes the sign, e the exponent, b the exponent bias, and f the mantissa. The normalized mantissa $1.f$ can be expressed explicitly as

$$1 + \frac{f}{2^{n_f}}, \quad f \in \{0, 1, \dots, 2^{n_f} - 1\},$$

with n_f being the number of mantissa bits. The exponent bias b depends on the width of the exponent field (n_e) and is defined as

$$b = 2^{n_e-1} - 1.$$

The components are grouped in a word as illustrated in Figure 1. The main parameters for single and double precision IEEE-754 formats are summarized in Table 1:

Table 1. IEEE-754 floating-point format parameters.

Parameter	Single Precision	Double Precision
Sign bits (s)	1	1
Exponent bits (n_e)	8	11
Mantissa bits (n_f)	23	52
Exponent bias (b)	127 (01111111b)	1023 (01111111111b)
Exponent range ($e - b$)	−126 to 127	−1022 to 1023
Mantissa f	0 to $2^{23} - 1$	0 to $2^{52} - 1$



Figure 1. Structure of the IEEE-754 standard floating-point word format.

The key additional features of the IEEE-754 standard format are:

- Normalized representations with an implicit MSB for extra precision.
- Special values like infinity ($\pm\infty$), NaN, and zero (± 0) are used to represent results of undefined operations, overflows and underflows.
- Fixed field lengths for e and f for single, double and so on precision.

However, direct implementation of IEEE-754 in FPGA-based DSP is costly: exponent alignment, mantissa computation, normalization, and rounding produce long critical

paths. In this work we propose two modifications to the IEEE-754 representation aimed at hardware efficiency while preserving the dynamic range needed by typical DSP workloads.

The first modification replaces the implicit $1.f$ significand by an explicit $0.f$ representation. The motivation is hardware-oriented, not algorithmic; with an explicit significand, the multiplier and adder pipelines do not require a separate code path to insert or strip the implicit leading bit, which simplifies the normalization stage and removes a small amount of multiplexing on the critical path. In structural terms, an implicit $1.f$ datapath must, at every stage that consumes or produces a significand, prepend the hidden leading 1 before any arithmetic (multiplication, alignment, or addition) and conditionally strip and re-insert it after normalization. With the explicit $0.f$ encoding the significand is already complete at the register boundary, so these insertion and stripping multiplexers are absent from the multiplier and adder normalization paths, and the leading-one detector operates directly on the stored significand without a special case for the hidden bit. The saving is a few multiplexers and the associated control per significand path rather than a large block, which is why the benefit is presented here as a simplification of the corner-case logic rather than as a dominant area reduction. The cost is one additional mantissa bit to preserve dynamic range: a $0.f$ value with n_f explicit bits represents the same set of normalized magnitudes as a $1.f$ value with $n_f - 1$ explicit bits. It is worth stressing that this encoding is precision-lossless: for the same number of stored mantissa bits the $0.f$ format would lose exactly one bit of significand precision (its representable magnitudes coincide with those of a $1.f$ format having one fewer explicit bit), but adding the single compensating bit restores bit-for-bit the same relative precision and the same per-operation $\frac{1}{2}$ ULP rounding bound as the equivalent $1.f$ format. The quantifiable cost is therefore one bit of storage density, not accuracy. This is confirmed by the numerical validation of Section 4.4, where the units configured with $n_f = 24$ match the accuracy of an IEEE-754 single-precision reference. The trade-off favors implementation simplicity over density. The configurable mantissa width introduced next allows the designer to compensate for this overhead at synthesis time. A direct, quantitative comparison against a functionally equivalent $1.f$ implementation of the same unit set, isolating the area, timing, and power contribution of this encoding choice, is not reported here and is identified as future work in Section 6.

The second modification enables configurable widths for the exponent (n_e) and mantissa (n_f) fields at synthesis time, sizing the datapath to application precision and dynamic range without the overhead of fixed single or double precision. The resulting custom format is

$$x = (-1)^s \cdot 2^{e-b} \cdot 0.f, \quad (2)$$

where s , e , and f denote the sign, exponent, and mantissa, respectively, and b is the same bias as explained before. Here, $0.f$ denotes the denormalized mantissa, which can be expressed explicitly as

$$\frac{f}{2^{n_f}}, \quad f \in \{2^{n_f-1}, \dots, 2^{n_f} - 1\},$$

with n_f being the number of mantissa bits.

3.2. Rounding Mode and Exception Handling

The proposed units use round-to-nearest-ties-to-even (RNE) as their rounding policy, which is the default mode in IEEE-754 and bounds the per-operation rounding error at 0.5 unit in the last place (ULP) in the last place (ULPULPs). Internally, the post-normalization mantissa carries guard, round, and sticky bits that drive the standard RNE decision logic. This solution adds a carry-propagation chain to the rounding stage and a small amount of soft logic, but the resulting design is unbiased and matches the accuracy of correctly rounded operators.

The proposed units (adder/subtractor, multiplier, MAC and fixed-to-float and float-to-fixed converters) do not implement the full IEEE-754 special-value taxonomy ($\pm\infty$, NaN). Each unit reports overflow and underflow through saturation flags and replaces the output with predefined constants (maximum-magnitude representable value on overflow, zero on underflow). DSP datapaths typically replace special-value propagation by guard logic at the system level. When full IEEE-754 special-value handling is required, the units can be wrapped in a thin compatibility layer at the cost of additional LUTs and one extra pipeline register per output.

Beyond special values, the proposed units also omit subnormal (denormalized) numbers and implement a single rounding mode (RNE) instead of the four IEEE-754 directed modes. The impact of these limitations is application-dependent. They are immaterial for the streaming DSP datapaths targeted here, such as FIR/IIR filtering, FFT, and matrix kernels operating on bounded-range, scaled signals, where overflow and underflow are managed by system-level guard logic and gradual underflow is not relied upon, and where RNE is the natural unbiased choice. Conversely, the omitted features do matter for general-purpose or library-grade floating-point that must propagate NaN/ $\pm\infty$ and preserve bit-exact IEEE-754 results. Algorithms that depend on gradual underflow near zero (some iterative solvers and ill-conditioned linear algebra) would see an abrupt flush-to-zero and also applications requiring directed rounding, such as interval arithmetic or rigorous error bounding, which need the toward-zero and toward- $\pm\infty$ modes. For these domains the compatibility wrapper and additional rounding logic mentioned above would be required, with the corresponding area and latency overhead.

3.3. Core Arithmetic Units

The core arithmetic units provide the basic floating-point operations required by typical DSP kernels. Three units are described in this section: a multiplier, an adder/subtractor, and a MAC unit that combines the previous two. All of them share the same custom floating-point format and a fully pipelined, synchronous design, so they can be freely interconnected within a larger datapath while keeping a predictable latency and throughput.

3.3.1. Multiplier Unit

The multiplier unit receives two floating-point values (hereafter denoted as A and B) and computes their product. Its corresponding hardware block diagram is presented in Figure 2.

In the first stage, the output sign is computed as the XOR of the input signs, while the input exponents are added concurrently. In parallel, a NOR operation on the input mantissas is used to detect zero operands, while the mantissa multiplication is performed by the DSP block shown in Figure 2. The mantissa product is implemented behaviorally using the standard VHDL multiplication operator applied to registered inputs.

Two pipeline registers are inserted in the multiplication so that the synthesis tool is free to map the operator to whichever resource it considers optimal. Because the multiplication is described behaviorally, no DSP primitive instantiation appears in the source code, and the design is portable across FPGA families and ASIC technologies.

In the second stage, the result of adding the exponents is examined to determine whether the floating-point product value leads to saturation (“corner cases flag” block in Figure 2). At the same time, the provisional sign is registered to synchronize with the DSP slice output corresponding to the mantissa’s multiplication result. Concurrently, the bias is subtracted from the sum in order to obtain the output exponent, according to

$$e_{out} = e_a + e_b - (bias + \alpha), \quad (3)$$

where α is an adjustment factor whose value can be either zero or one, as explained below.

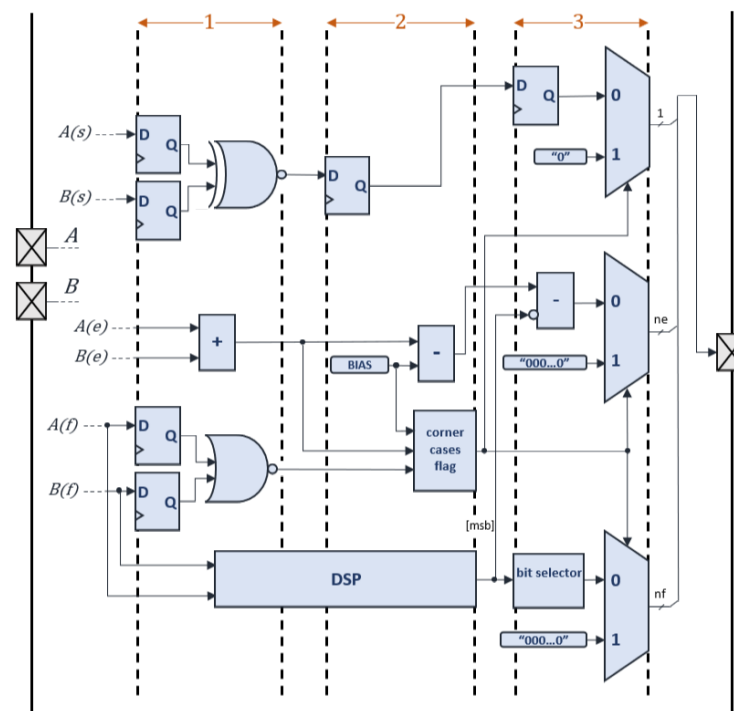


Figure 2. Hardware block diagram of the multiplier unit. Dotted lines denote the pipeline stage boundaries. Top labels indicate the corresponding pipeline stages.

Finally, in the third stage, the corner-case flags are combined with the provisional output fields to determine the output value. The final exponent may be equal to the provisional exponent minus one when the MSB of the multiplication result is '0'. Since the input mantissas are normalized within the range $[0.5, 1)$, their product lies within the interval $[0.25, 1)$. Consequently, an adjustment (α) is required to preserve the mantissa width, unless the MSB of the product is '1', in which case no adjustment is necessary.

As an illustration, consider the multiplication $A \times B$ where $A = 1.5$ and $B = 0.75$, encoded with $n_e = 8$ and $n_f = 24$ (single-precision-like widths in the $0.f$ form, $b = 127$). In the proposed format, A is encoded with sign $s_A = 0$, exponent $e_A = 128$ (i.e., 2^1), and mantissa f_A representing 0.75 ($1.5 = 2^1 \cdot 0.75$). B is encoded with $s_B = 0$, $e_B = 127$ (2^0), and f_B representing 0.75 ($0.75 = 2^0 \cdot 0.75$). In stage 1 the sign is computed as $s_A \oplus s_B = 0$, the mantissa product yields $0.75 \times 0.75 = 0.5625$, and the exponent sum yields $e_A + e_B = 255$. In stage 2 the bias is subtracted: $e_{out} = 255 - (127 + \alpha)$. Since the MSB of the product is '1' ($0.5625 \geq 0.5$), $\alpha = 0$ and $e_{out} = 128$, which corresponds to 2^1 . The output mantissa is the upper n_f bits of the product, representing 0.5625 . The reconstructed value is $2^1 \cdot 0.5625 = 1.125$, which matches the expected result.

3.3.2. Adder/Subtractor Unit

This unit receives two floating-point values (referred to as A and B hereafter) and computes either their sum or difference. Figure 3 shows the FPGA-oriented hardware implementation block diagram of the unit. It is shown how the used logic is divided into five different synchronous stages.

In the first stage, the input signs are compared, and a flag is generated to indicate whether they coincide. Concurrently, a comparator receives both input exponents and outputs the greater value along with a flag indicating the corresponding input. The difference between the exponents is then calculated to determine the alignment required for

the input mantissas. The mantissas are also compared in order to generate a flag indicating the greater one.

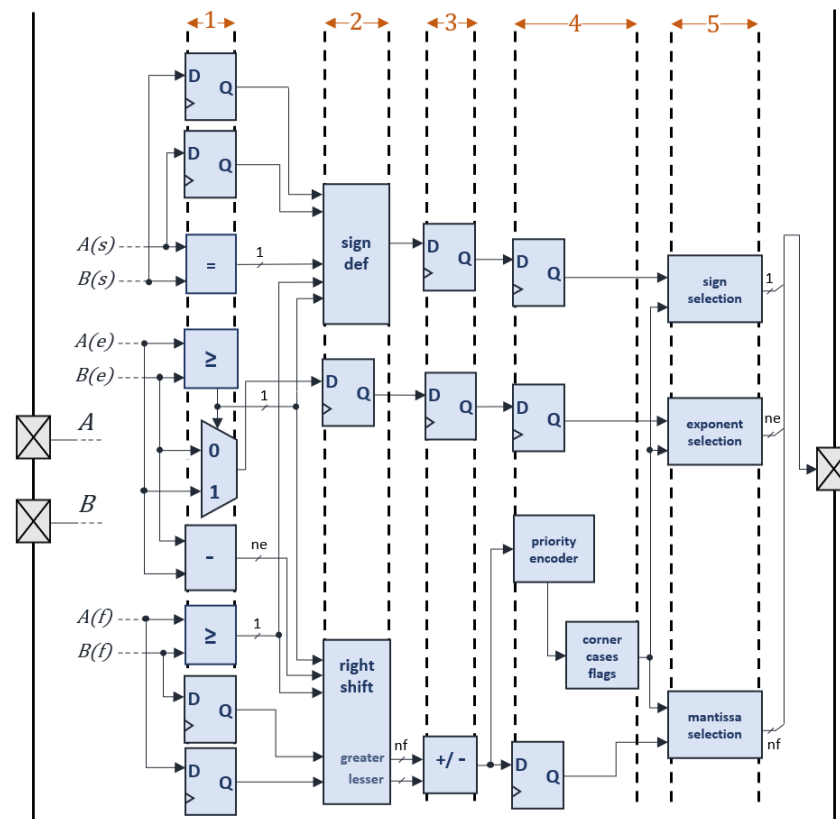


Figure 3. Hardware block diagram of the Adder/Subtractor unit, with dotted lines indicating pipeline stage boundaries. Top labels indicate the corresponding pipeline stages.

The second stage addresses both sign definition and mantissa alignment. The sign is derived from the previously generated flags, selecting the sign of the input with the greatest absolute value. In subtraction operations where the magnitude of input B exceeds that of A, the sign is set to the opposite of B. Mantissa alignment is subsequently achieved by applying a right-shift operation to the mantissa associated with the smaller value. In order to operate with maximum precision both mantissas are added a zero-padding vector as least significant bits before the alignment. The padding vector has the same width as the mantissa, so every possible case is considered.

The third stage mainly performs the output mantissa computation. In the case of subtraction, the smaller mantissa is consistently subtracted from the larger one, independently of their origin. This approach guarantees that the result is expressed in absolute value.

The fourth stage incorporates a priority encoder that receives the result from the preceding computation and identifies the position of the msb. This information is used to adjust the output exponent, select the relevant mantissa bits, and detect corner cases such as underflow and overflow.

It should be noted that the priority encoder is a resource-intensive operation. Consequently, only the most significant half of the result vector is analyzed. This optimization is justified by the fact that, for the msb to appear in the least significant half, three conditions must be simultaneously fulfilled: the input exponents must be equal, the operation must be a subtraction, and the two mantissas must be identical. However, if the exponents are equal, no alignment shift is applied to either mantissa, which guarantees that the msb remains within the most significant half in all cases. This constitutes a contradiction, confirming

that the msb can never be located in the least significant half, and therefore restricting the encoder's scope to the upper half introduces no loss of generality nor loss of precision.

Finally, the fifth stage produces the output value. At this point, the corner-case flags determine whether the output sign, exponent, and mantissa correspond to the values obtained from the computation, or are replaced by predefined constants in underflow or overflow situations.

3.3.3. MAC Unit

The MAC unit incorporates the functionality of the previously described multiplier and adder/subtractor units. It receives three floating-point values (denoted as A, B, and C) and performs the operation

$$R = (A \times B) + C. \quad (4)$$

Its hardware block diagram is shown in Figure 4. The design connects the multiplier and adder units in a straightforward manner, without applying any additional fusion or optimization techniques.

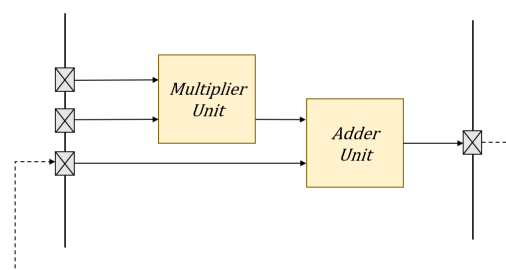


Figure 4. Hardware block diagram of the MAC unit.

This non-fused construction is a deliberate trade-off. A fused multiply-add (FMA) computes $R = A \cdot B + C$ with a single rounding step at the end, so its error with respect to the real-arithmetic result is bounded by $0.5 \text{ ULP}(R)$. The chained design rounds twice: the multiplier rounds the intermediate product $P = A \cdot B$ and the adder rounds the final sum $P + C$. With RNE on both stages, the chained error is bounded by

$$|\hat{R} - R| \leq \frac{1}{2} \text{ULP}(\hat{P}) + \frac{1}{2} \text{ULP}(\hat{R}), \quad (5)$$

that is, at most one extra 0.5 ULP term with respect to FMA, weighted by the ULP of the intermediate product. In practice this extra term is bounded above by $0.5 \text{ ULP}(\hat{R})$ whenever $|P| \leq |R|$, which holds when the additive operand C does not cancel the magnitude of the product (the common case in FIR-style accumulations), the worst-case chained error is therefore $1 \text{ ULP}(\hat{R})$, only twice the FMA bound. The chained design preserves the modular structure of the unit set (the multiplier and adder can be used as standalone blocks with identical pipeline interfaces) and is acceptable for the bounded-depth accumulations typical of the DSP kernels targeted by this work. An FMA variant that recovers the 0.5 ULP bound and shortens the critical path by one normalization stage is identified as future work in Section 6.

3.4. Format Conversion Units

3.4.1. Fixed-to-Float Unit

This conversion unit receives a fixed-point value and produces its representation in the custom floating-point format. Both the fixed-point and floating-point field widths can be customized. The hardware block diagram of the unit is shown in Figure 5.

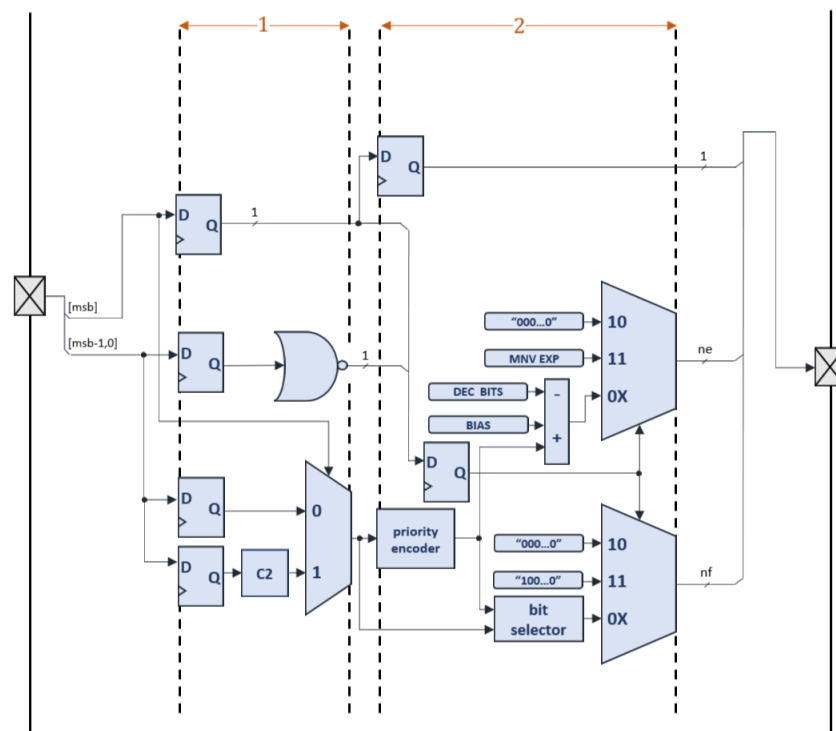


Figure 5. Hardware block diagram of the fixed-to-float conversion unit, where dotted lines denote the pipeline stage boundaries. Top labels indicate the corresponding pipeline stages.

The output sign is identical to the input sign, so it is forwarded directly through pipeline registers. Regarding the mantissa, as it has to be represented in absolute value, the first step is to choose the input mantissa or its two's complement. This operation is implemented using a multiplexer whose selection signal corresponds to the input sign. In parallel, a NOR gate is used to verify whether the input value is equal to zero.

The output of the multiplexer is connected to a priority encoder in order to determine the position of the MSB. The result of the priority encoder, together with the input sign and the *zero_flag*, is subsequently used to define the output exponent and mantissa.

When the *zero_flag* is active, the output exponent and mantissa are both set to predefined constants. Otherwise, the exponent is computed by adding the output of the priority encoder to the exponent bias and subtracting the number of fractional bits in the fixed-point format. On the other hand, the mantissa is obtained by shifting its absolute value according to the position indicated by the priority encoder output, thereby achieving normalization. This process is illustrated in detail in the second stage of Figure 5.

3.4.2. Float-to-Fixed Unit

This unit receives a value in floating-point format and produces its equivalent fixed-point representation. Both the floating-point and fixed-point field widths are configurable. The block also provides two single-bit outputs to indicate overflow and underflow conditions. The hardware block diagram of the unit is shown in Figure 6.

In the first stage, the input signal is decomposed into its sign, exponent, and mantissa. The bias is subtracted from the exponent to determine the position of the MSB of the conversion result. This defines the number of shifts that need to be applied to the mantissa. Simultaneously, either the mantissa or its two's complement is selected via a multiplexer using the input sign as the selection signal.

Finally, the MSB of the input exponent, together with the input sign and the *shift* signal, are used to detect potential saturation. If saturation occurs, the output value is

replaced by a predefined vector; otherwise, it is equal to the multiplexer output with its associated sign bit.

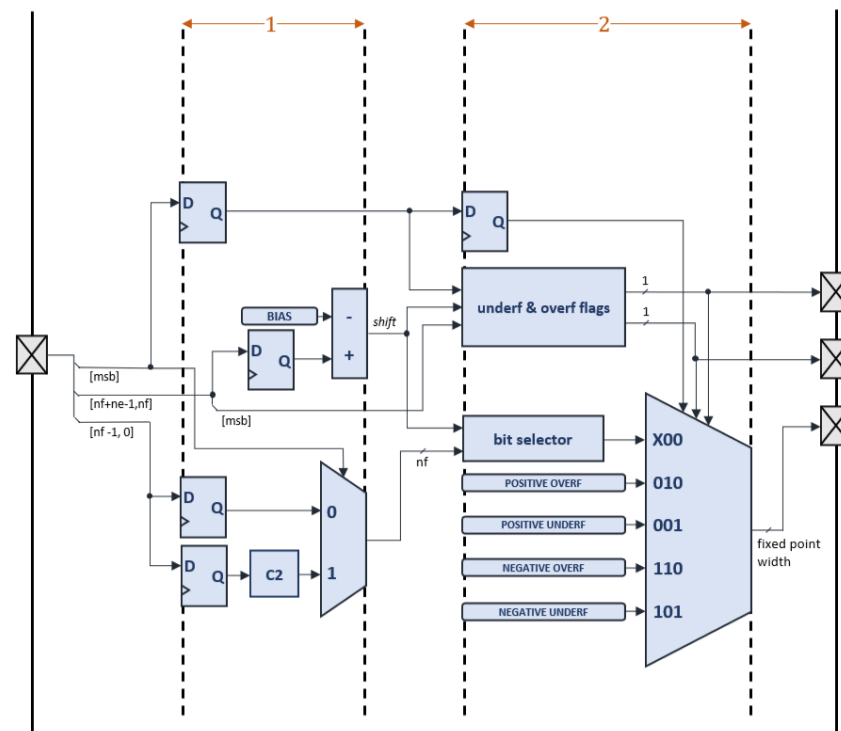


Figure 6. Hardware block diagram of the float-to-fixed conversion unit. Dotted lines denote the pipeline stage boundaries. Top labels indicate the corresponding pipeline stages.

4. Experimental Results and Discussion

To evaluate the performance of the floating-point unit set, benchmarks were conducted on mid-end and high-end FPGAs. For this purpose, the Artix-7 AC701 and Kintex Ultrascale KC105 evaluation boards from AMD Xilinx are selected. These boards were deliberately selected to contrast a modest, earlier-generation device with a high-end, more advanced one, while ensuring comparability with the most recent state-of-the-art works. The conducted benchmarks focus on hardware resources usage, power consumption, maximum operating frequency, latency, and throughput. The implementations were carried out using Vivado 2023.2 and VHDL-2008-compliant code.

The AC701 board integrates the Artix-7 XC7A200T-2FBG676C device, which uses 28 nm process technology and provides 134,600 LUTs, 269,200 flip-flops (FFs), and 740 DSP slices [23]. In comparison, the KC105 board features the Kintex Ultrascale XCKU040-2FFVA1156E device, based on a 20 nm process, offering 242,400 LUTs, 484,800 FFs, and 1920 DSP slices [24]. The clear distinction in process technology and available resources highlights the gap between mid-range 7-series devices and high-end Ultrascale devices.

The benchmarks consist of implementations of the proposed floating-point units parameterized according to the data widths of the IEEE-754 single precision format. This configuration facilitates a more effective comparative analysis with recent state-of-the-art works.

The multiplier and MAC units are implemented with two synthesis configurations of the same VHDL source: Using DSP slices or LUT-only. Both units have been evaluated under the same pair of strategies. The DSP-inferred variant is the default Vivado mapping on AMD/Xilinx targets, in which the synthesis tool absorbs the mantissa product (and, in the case of the MAC, the subsequent accumulation path) and its surrounding registers into cascaded DSP48 slices. The LUT-only variant is obtained by forcing the multiplication

and the associated arithmetic in the MAC into soft logic. Both strategies are portable to any FPGA family or ASIC. The exponent path, sign and corner-case logic, and pipeline structure are identical in both cases for each unit. The LUT-only configuration is intended for designs that need to keep DSP slices free for other functions, or for devices without hardened DSP macros. Hardware cost and energy figures for both configurations of the multiplier and the MAC are reported separately in this section.

4.1. Methodology and Reporting Conventions

To delimit the scope of the comparison, the following conventions apply throughout this section:

- **Operating frequency:** The proposed units close timing at 300 MHz on both boards. This value is close to the maximum frequency achievable by the designs and is sustained under the routing pressure of the FIR case study (Section 5). It also serves as a uniform working point for like-for-like throughput, energy, and ATP comparison. Surveyed works are listed at their own reported frequency.
- **Power:** Reported values are dynamic power estimated by Vivado 2023.2 with default switching activity at 300 MHz, static power is excluded. Figures imported from surveyed works are reported as published and therefore also use dynamic-power estimations from different EDA tools.
- **Validation:** Each unit was validated against a MATLAB 2023b double-precision reference, using (i) random input vectors covering the supported exponent range, (ii) directed corner cases (near zero, near saturation, catastrophic-cancellation pairs in the adder), and (iii) sequences from a FIR workload. For the random campaign, two independent test sets of 5000 input vectors each were generated per unit: one drawn from a discrete uniform distribution and one from a standard normal distribution, so that both the uniform coverage of the dynamic range and the operand statistics typical of real DSP signals are exercised. Validation was carried out in two stages: first by behavioral simulation in Vivado 2023.2 against the MATLAB reference, and then by on-board execution on the target devices, where the input vectors were stored in on-chip memories, streamed into the unit under test, and the results read back from memory for comparison with the reference. The absolute error is bounded by the per-operation ULP bound derived in Section 4.4, and all units pass it on the full test set.
- **Area-equivalent metric:** Mixing LUTs, FFs, DSP slices, and LUTRAMs in a single-resource comparison is misleading; the area-equivalent figure used throughout this section is defined in Equation (6) from the LUT/DSP/FF equivalences in the AMD/Xilinx datasheets of the target devices [23,24]. The weights are an engineering convention tied to one device family; results are therefore presented as relative trends within the family. Sensitivity to alternative weights is largest for DSP-based vs. LUT-only comparisons, which are discussed explicitly when relevant.
- **Cross-technology and cross-vendor scope:** Several surveyed works target Spartan-6 (45 nm), Virtex-5 (65 nm), or Virtex-7 (28 nm) devices, comparisons against them focus on architectural trends, since process scaling alone affects frequency and power. The proposed units are described in pure VHDL-2008 with no HLS flow and no vendor primitive instantiation, and are therefore portable to any FPGA family or ASIC technology supporting a VHDL-2008 synthesis flow. Furthermore, the units are implemented in two device families to obtain results from high-end and more economic device families. From the logic architecture point of view, the multiplier and MAC are reported in two synthesis configurations: a default mapping in which DSP slices are inferred for the mantissa product, and a LUT-only mapping. Cross-

vendor implementation is not characterized in this paper. The consequence for the conclusions is that comparisons spanning different process nodes (45 nm, 65 nm, or 28 nm parts versus the 28 nm and 20 nm devices used here) must be read as indicative of architectural and resource-efficiency trends, not as absolute frequency or power superiority, since process scaling alone shifts both. Accordingly, the strongest and most controlled evidence for the headline ATP claims comes from the same-device, same-node comparisons, namely, the proposed units versus the AMD Xilinx IP cores on the Kintex UltraScale (20 nm) part, where the area-equivalent metric, the operating frequency, and the power-estimation flow are identical for all designs. The cross-technology figures against academic works on older nodes (only 2 cases of a total of 15) are reported for completeness and context, and the qualitative ranking they support is consistent with the same-node results, but the quantitative percentages involving those works should be interpreted with this caveat in mind.

4.2. Implementation Details and Resource Utilization

Table 2 summarizes the implementation details of the units on Ultrascale and Artix-7. All units close timing at 300 MHz with correct functional behavior on the boards.

Table 2. Implementation details of the proposed floating-point units on Kintex Ultrascale (KU) and Artix-7 (A7), parameterized according to the IEEE-754 single-precision widths.

Unit	Latency (cycles)	Power (mW)		LUTs		FFs		Other HW	
		KU	A7	KU	A7	KU	A7	KU	A7
Multiplier (DSP)	3	15	16	75	76	68	68	2 DSPs	2 DSPs
Multiplier (LUT)	3	17	19	367	360	133	133	-	-
Add/Sub	5	27	30	543	550	281	281	1 LU-TRAM	1 LU-TRAM
MAC (DSP)	8	41	44	628	633	414	414	2 DSPs 48 LU-TRAMs	2 DSPs 57 LU-TRAMs
MAC (LUT)	8	46	50	920	914	479	479	48 LU-TRAMs	57 LU-TRAMs
Fix-Float	2	7	7	65	75	44	44	-	-
Float-Fix	3	9	10	133	133	64	64	-	-

To contextualize these results, Table 3 compares the proposed designs with the state-of-the-art works of Section 2 and the Xilinx IP cores. Each Xilinx unit offers four configurations from two pairs of optimization goals: High-Speed (HS) vs. Low-Latency (LL), and Resource (R) vs. Performance (P), giving HS-R, HS-P, LL-R, and LL-P. The area-equivalent figure of merit reported in the last column is defined as

$$\text{Area}_{eq} = \text{LUTs} + \text{LUTRAMs} + \frac{\text{FFs}}{2} + 150 \cdot \text{DSPs} \quad (6)$$

and is derived from the device datasheets: for the XC7A200T, 1 DSP equals 182 LUTs and 1 FF equals 0.5 LUTs [23], for the XCKU040, 1 DSP equals 126 LUTs and 2 FFs equal 1 LUT [24]. The factor 150 averages the two devices for cross-board comparability. The exact arithmetic mean of the two device-specific weights is $(182 + 126)/2 = 154$. Therefore, the value 150 is adopted as a rounded convention of the same order, and the conclusions are not sensitive to this rounding for two reasons. First, all comparisons against the Xilinx IP cores are made between designs that use the same number of DSP slices as the proposed

units (two), so the DSP term $150 \cdot \text{DSPs}$ cancels in the numerator of any area difference, and the relative ranking is therefore invariant to the chosen weight; only the percentage scale shifts slightly through the denominator. Second, the academic designs surveyed are LUT-only (zero DSPs), so their area-equivalent figures are completely independent of the DSP weight. The only comparison genuinely sensitive to the weight is DSP-inferred versus LUT-only, which is reported explicitly and discussed as such. As a check, recomputing the proposed multiplier with DSPs using the device-specific weights instead of 150 changes its area-equivalent figure by about -12% on UltraScale (126 vs. 150) and about 16% on Artix-7 (182 vs. 150) and does not alter any of the qualitative conclusions. The detailed unit-by-unit discussion of Table 3 is deferred to the joint analysis of hardware cost and energy/ATP given after Table 4.

Table 3. Hardware cost and target technology of the units parameterized according to the IEEE-754 single-precision widths. Unit sources in bold font indicate the units proposed in this work.

Unit	Source	Tech.	LUTs	FFs	DSPs	LUTRAMs	Area Eq.
Multiplier	Proposed DSP (Ultrascale)	Kintex-US	75	68	2	0	409
	Proposed LUT (Ultrascale)	Kintex-US	367	133	0	0	434
	Proposed DSP (Artix-7)	Artix-7	76	68	2	0	410
	Proposed LUT (Artix-7)	Artix-7	360	133	0	0	427
	Xilinx IP: HS-R	Kintex-US	117	228	2	0	531
	Xilinx IP: HS-P	Kintex-US	135	266	2	16	584
	[14]	Spartan-6	1734	427	0	0	1948
	[15]	Kintex-US	106	96	2	0	454
	[16]-Standard	Artix-7	657	96	0	0	705
	[16]-DivConq	Artix-7	668	96	0	0	716
	[16]-Karatsuba/Ofman	Artix-7	648	96	0	0	696
	[17]-OSBM	Kintex-US	696	0	0	0	696
	[17]-HKM	Kintex-US	639	0	0	0	639
	[17]-HRKM	Kintex-US	671	0	0	0	671
Add/Sub	Proposed (Ultrascale)	Kintex-US	543	281	0	1	685
	Proposed (Artix-7)	Artix-7	550	281	0	1	692
	Xilinx IP: HS-R	Kintex-US	383	290	0	0	528
	Xilinx IP: HS-P	Kintex-US	394	328	0	16	574
	Xilinx IP: LL-R	Kintex-US	462	375	0	0	650
	Xilinx IP: LL-P	Kintex-US	477	413	0	16	700
	[14]	Spartan-6	376	224	0	0	488
	[15]	Kintex-US	263	96	0	0	311
	[18]	Virtex-5	432	433	0	0	649
MAC	Proposed DSP (Ultrascale)	Kintex-US	628	414	2	48	1183
	Proposed LUT (Ultrascale)	Kintex-US	920	479	0	48	1208
	Proposed DSP (Artix-7)	Artix-7	633	414	2	57	1197
	Proposed LUT (Artix-7)	Artix-7	914	479	0	57	1211
	Xilinx IP: HS-R	Kintex-US	703	408	2	7	1214
	Xilinx IP: HS-P	Kintex-US	701	447	2	25	1250
	Xilinx IP: LL-R	Kintex-US	703	408	2	7	1214
	Xilinx IP: LL-P	Kintex-US	701	447	2	25	1250
	[14]	Spartan-6	3912	1999	0	0	4912
	[19]	Virtex-7	2869	979	0	0	3359
	[20]	Zynq-US+	3071	1500	0	0	3821
	[21]	Virtex-7	2659	1711	0	0	3515

Table 3. Cont.

Unit	Source	Tech.	LUTs	FFs	DSPs	LUTRAMs	Area Eq.
Fix-to-Float	Proposed (Ultrascale)	Kintex-US	65	44	0	0	87
	Proposed (Artix-7)	Artix-7	75	44	0	0	97
	Xilinx IP: HS-R	Kintex-US	83	64	0	0	115
	Xilinx IP: HS-P	Kintex-US	103	92	0	16	166
Float-to-Fix	Proposed (Ultrascale)	Kintex-US	133	64	0	0	165
	Proposed (Artix-7)	Artix-7	133	64	0	0	165
	Xilinx IP: HS-R	Kintex-US	105	71	0	0	141
	Xilinx IP: HS-P	Kintex-US	121	93	0	8	176

Table 4. Performance metrics of the units parameterized according to the IEEE-754 single-precision widths. Unit sources in bold font indicate the units proposed in this work.

Unit	Source	Frequency (MHz)	Latency (cycles)	Latency (ns)	Throughput (MSps)	Power (mW)	Energy (nJ/op)	ATP (area × nJ/op)
Multiplier	Proposed DSP (Ultrascale)	300	3	10	300	15	0.050	20.450
	Proposed DSP (Artix-7)	300	3	10	300	16	0.053	21.867
	Proposed LUT (Ultrascale)	300	3	10	300	17	0.057	24.537
	Proposed LUT (Artix-7)	300	3	10	300	19	0.063	27.034
	Xilinx IP: HS-R	300	4	13.3	300	24	0.080	42.480
	Xilinx IP: HS-P	300	6	20	300	26	0.087	50.613
	[14]	108	–	–	108	50	0.463	901.620
	[15]	141	1	7.1	141	184	1.310	592.454
	[16]–Standard	105	1	9.5	105	10	0.095	67.143
	[16]–DivConq	121	1	8.3	121	12	0.099	71.008
	[16]–Karatsuba/Ofman	93	1	10.75	95	9	0.097	65.937
	[17]–OSBM	239	1	4.2	239	17	0.071	49.506
	[17]–HKM	283	1	3.5	283	19	0.067	42.901
	[17]–HRKM	346	1	2.9	346	24	0.069	46.543
Add/Sub	Proposed (Ultrascale)	300	5	16.7	300	27	0.090	61.650
	Proposed (Artix-7)	300	5	16.7	300	30	0.100	69.200
	Xilinx IP: HS-R	300	4	13.3	300	24	0.080	42.240
	Xilinx IP: HS-P	300	6	20	300	27	0.090	51.660
	Xilinx IP: LL-R	300	4	13.3	300	29	0.097	62.785
	Xilinx IP: LL-P	300	6	20	300	31	0.103	72.282
	[14]	103	–	–	103	14	0.136	66.330
	[15]	121	1	8.3	121	187	1.545	480.636
	[18]	263	8	30.4	263	62	0.236	152.878
MAC	Proposed DSP (Ultrascale)	300	8	26.7	300	41	0.137	161.504
	Proposed LUT (Ultrascale)	300	8	26.7	300	46	0.153	184.840
	Proposed DSP (Artix-7)	300	8	26.7	300	44	0.147	175.591
	Proposed LUT (Artix-7)	300	8	26.7	300	50	0.167	201.837
	Xilinx IP: HS-R	300	7	23.3	300	44	0.147	178.053
	Xilinx IP: HS-P	300	9	30	300	45	0.150	187.425
	Xilinx IP: LL-R	300	7	23.3	300	44	0.147	178.053
	Xilinx IP: LL-P	300	9	30	300	45	0.150	187.425
	[14]	110	–	–	110	131	1.191	5849.150
	[19]	383	9	23.5	383	176	0.460	1543.332
	[20]	173	–	–	173	161	0.931	3555.960
	[21]	461	12	26	461	186	0.403	1417.998
Fix-to-Float	Proposed (Ultrascale)	300	2	6.7	300	7	0.023	2.030
	Proposed (Artix-7)	300	2	6.7	300	7	0.023	2.263
	Xilinx IP: HS-R	300	2	6.7	300	8	0.027	3.067
	Xilinx IP: HS-P	300	4	13.3	300	11	0.037	6.087
Float-to-Fix	Proposed (Ultrascale)	300	3	10	300	9	0.030	4.950
	Proposed (Artix-7)	300	3	10	300	10	0.033	5.500
	Xilinx IP: HS-R	300	1	3.3	300	8	0.027	3.760
	Xilinx IP: HS-P	300	3	10	300	11	0.037	6.453

In addition to the hardware usage analysis shown in Table 3, Table 4 reports on performance-related metrics, namely, maximum operating frequency, latency (in both cycles and absolute time), throughput and dynamic power. The latter is complemented by

two derived metrics: energy per operation and area-throughput-power (ATP). The energy consumed per operation is defined as

$$E = \frac{P}{\text{Throughput}}, \quad (7)$$

where P is the average power consumption and Throughput is given in operations per second.

To combine hardware cost and energy efficiency, the ATP metric is computed as

$$\text{ATP} = \text{Area}_{eq} \cdot E, \quad (8)$$

where Area_{eq} is the area-equivalent figure of merit defined in Equation (6).

4.3. Results and Discussion

The following analysis combines the hardware cost from Table 3 with the performance and energy metrics from Table 4, and is summarized in Table 5. Table 5 reports the relative differences with respect to the Ultrascale implementation of the proposed units, computed as

$$\Delta X = \frac{X_{\text{proposed (Ultrascale)}} - X_{\text{design}}}{X_{\text{proposed (Ultrascale)}}} \cdot 100, \quad (9)$$

where X is area, energy per operation, or ATP.

Table 5. Variations of area, energy, and ATP relative to the proposed Ultrascale implementation (using DSPs in the multiplier and MAC units). Unit sources in gray bold font indicate the other units proposed in this work. Green cells indicate improvement, red indicate degradation and gray text marks alternative configurations of the proposed design.

Unit	Source	vs. Proposed (Ultrascale) *		
		ΔArea_{eq} (%)	ΔEnergy (%)	ΔATP (%)
Multiplier	Proposed DSP (Artix-7)	0	−6	−7
	Proposed LUT (Ultrascale)	−6	−14	−20
	Proposed LUT (Artix-7)	−4	−26	−32
	Xilinx IP: HS-R	−30	−60	−108
	Xilinx IP: HS-P	−43	−74	−147
	[14]	−376	−826	−4309
	[15]	−11	−2520	−2796
	[16]-Standard	−72	−90	−228
	[16]-DivConq	−75	−98	−247
	[16]-Karatsuba/Ofman	−70	−94	−222
	[17]-OSBM	−70	−42	−142
	[17]-HKM	−56	−34	−110
	[17]-HRKM	−64	−38	−128
Add/Sub	Proposed (Artix-7)	−1	−11	−12
	Xilinx IP: HS-R	23	11	31
	Xilinx IP: HS-P	16	0	16
	Xilinx IP: LL-R	5	−8	−2
	Xilinx IP: LL-P	−2	−14	−17
	[14]	29	−51	−8
	[15]	55	−1617	−680
	[18]	5	−162	−148

Table 5. Cont.

Unit	Source	vs. Proposed (Ultrascale) *		
		$\Delta Area_{eq}$ (%)	$\Delta Energy$ (%)	ΔATP (%)
MAC	Proposed LUT (Ultrascale)	−2	−12	−14
	Proposed DSP (Artix-7)	−1	−7	−9
	Proposed LUT (Artix-7)	−2	−22	−25
	Xilinx IP: HS-R	−3	−7	−10
	Xilinx IP: HS-P	−6	−10	−16
	Xilinx IP: LL-R	−3	−7	−10
	Xilinx IP: LL-P	−6	−10	−16
	[14]	−315	−769	−3522
	[19]	−184	−236	−855
	[20]	−223	−580	−2102
	[21]	−197	−194	−778
Fix-to-Float	Proposed (Artix-7)	−11	0	−11
	Xilinx IP: HS-R	−32	−14	−51
	Xilinx IP: HS-P	−90	−57	−198
Float-to-Fix	Proposed (Artix-7)	0	−10	−11
	Xilinx IP: HS-R	15	11	24
	Xilinx IP: HS-P	−6	−22	−30

* For the multiplier and the MAC, the baseline is the DSP-inferred synthesis configuration on Ultrascale; the LUT-only configuration of the same VHDL source is reported separately.

4.3.1. Multiplier

The proposed multiplier is reported in two synthesis configurations of the same VHDL source:

- In the DSP-inferred configuration on UltraScale, the synthesis tool (Vivado) maps the 24×24 -bit mantissa product to two cascaded DSP48 slices, absorbing the surrounding pipeline registers as the internal pipeline registers of the slices and reaching the working frequency of 300 MHz. This yields 75 LUTs, 68 FFs and 15 mW of dynamic power, well below the 117–135 LUTs and 24–26 mW of the Xilinx HS-R/HS-P IPs. The 0.f encoding removes the implicit-MSB insertion and stripping multiplexers across the pipeline. The cost is a 3-cycle latency against the single-cycle designs in [15,17], which, for streaming DSP workloads at 300 MHz, is acceptable. Overall, the DSP-inferred configuration improves ATP by 108% against Xilinx HS-R, 147% against HS-P, and by 110–4309% against academic designs.
- The LUT-only configuration forces the same multiplication into distributed logic via a synthesis attribute. The implementation obtained would be the one on platforms without hardened DSP macros, for example. It reaches $Area_{eq}$ values of 434 (Ultrascale) and 427 (Artix-7), only 4–6% larger than the DSP-inferred baseline and clearly below the 639–716 $Area_{eq}$ reported by the LUT-only academic designs of [16,17]. The energy and ATP follow the same trend: the LUT-only variant pays 14–32% more in ATP than the DSP-inferred baseline, but still beats every Xilinx IP and every LUT-only academic design surveyed. This makes the LUT-only configuration suitable for DSP-free pipelines and for cross-vendor portability without sacrificing competitiveness against pure soft-logic prior art.

4.3.2. Adder/Subtractor

The adder/subtractor is the least competitive unit of the set. The dominant cost is the leading-one detector after significant subtraction, feeding the post-normalization barrel

shifter. Even with the priority encoder restricted to the upper half of the result vector (Section 3), the LUT count (543 on Ultrascale) exceeds the Xilinx HS-R IP (383 LUTs) and the compact design of [15] (263 LUTs). The adopted 5-stage pipeline is driven by the latency requirements of exponent comparison, mantissa alignment, subtraction, normalization, and exception masking. Combining any of these stages was found to reduce the achievable F_{max} .

In ATP terms, the proposed unit is 31% worse than Xilinx HS-R and 16% worse than HS-P, but improves over the low-latency variants by 2–17% (LL-R, LL-P) and over every academic design surveyed (8–680%). The adder/subtractor is therefore included in the set for design coherence (uniform pipeline interface, identical $0.f$ format), not as a strict resource improvement over all vendor IPs, but as a consistently resource-efficient alternative to recent academic implementations reported in the literature.

4.3.3. MAC

The 8-cycle latency is the sum of the multiplier (3) and adder (5) stages, consistent with a non-fused chained design. Because the new multiplier and adder are tighter than the previous iteration of the work, the MAC inherits those gains: on Ultrascale the DSP variant occupies 628 LUTs, 414 FFs, 2 DSPs and 48 LUTRAMs, with 41 mW dynamic power and an $Area_{eq}$ of 1183, below all Xilinx IP variants (1214–1250). In ATP terms the DSP-inferred MAC is 10% better than Xilinx HS-R/LL-R and 16% better than HS-P/LL-P. It is between 2 and 35 times better in ATP than the academic proposals [14,19–21], mainly because those designs are LUT-only and do not exploit DSP slices.

The LUT-only MAC variant pays 14% more ATP than the DSP variant on Ultrascale, but still matches Xilinx HS-R/LL-R within 4% and beats HS-P/LL-P by 2%, while removing the dependency on hardened DSP macros. Both variants stay within the chained worst-case rounding error bound of 1 ULP derived in Section 3. An FMA implementation that recovers the 0.5 ULP bound and shortens the critical path by one normalization stage is left as future work.

It must be stressed that this MAC comparison is not made on an equal-accuracy footing. The proposed unit is a non-fused chained design with two rounding steps and a worst-case error of 1 ULP($\hat{R}$), whereas several of the surveyed works [19–21] adopt fused multiply-add architectures that perform a single final rounding and therefore reach the tighter 0.5 ULP($\hat{R}$) bound. Fused designs trade this extra accuracy for additional internal datapath width and, frequently, a longer or more complex normalization stage, while the chained design trades one extra 0.5 ULP term for modularity, since the multiplier and adder remain reusable standalone blocks with identical pipeline interfaces. Consequently, the ATP advantages reported above should be read together with this accuracy difference: the proposed MAC is more area-, energy-, and ATP-efficient, but at the cost of one additional ULP of worst-case error relative to a fused operator. The bounded-depth accumulations targeted in this work tolerate this trade-off, and the FMA variant identified as future work would close the accuracy gap.

The latency dimension follows the same trade-off. The non-fused MAC has a latency equal to the sum of the multiplier and adder pipelines (8 cycles), because the intermediate product is fully normalized and rounded before being fed to the adder. A fused multiply-add merges the two normalization/rounding stages into one, which both removes the intermediate rounding (recovering the 0.5 ULP bound) and shortens the datapath by one normalization stage, so it is expected to reduce the latency of the present design by roughly one cycle in addition to improving accuracy. The non-fused choice therefore costs one extra rounding and one extra normalization stage relative to an FMA, which is the price paid for keeping the multiplier and adder reusable as independent blocks. Quantifying this latency/accuracy/area trade-off against an FMA implementation is part of the future work.

4.3.4. Format Conversion Units

The fix-to-float unit on Ultrascale uses 65 LUTs, 44 FFs and 7 mW dynamic power. Compared with the Xilinx HS-R IP it is 32% smaller in area, 14% lower in energy per operation, and 51% better in ATP. Against HS-P the gap widens to 90%, 57% and 198% respectively. The dominant cost is the leading-one detector that determines the post-conversion exponent, which reuses the encoder optimization of the adder.

The float-to-fix unit performs comparably to Xilinx HS-R: 15% larger in area but with similar energy and a 24% higher ATP, mostly explained by the extra pipeline stage (3 cycles vs. 1). Against HS-P the proposed unit is 6% smaller, 22% better in energy and 30% better in ATP. Both blocks integrate the floating-point datapath with ADC/DAC interfaces without becoming a bottleneck, and on the Artix-7 device, the metrics track within a few per cent of the Ultrascale figures.

4.3.5. Summary

The proposed unit set is particularly competitive in the multiplier and format-conversion blocks, where the use of inferred hardware DSP blocks together with the simplified $0.f$ corner-case logic yields measurable gains in area, energy consumption, and ATP compared with vendor IPs and recent academic proposals. The MAC unit also achieves competitive results, although its ATP improvement over Xilinx IPs is limited to 10–16%. Its main advantage appears against LUT-only academic implementations, where the gap extends to one or two orders of magnitude. Conversely, the adder/subtractor remains the weakest component of the set, thereby delimiting the scope of the conclusions presented in Section 4.1.

The proposed unit set is most competitive in the multiplier and the format conversion blocks, where the combination of inferred DSP slices (when available) and the simplified $0.f$ corner-case logic translates into measurable area, energy, and ATP gains over both vendor IPs and recent academic proposals. The MAC is competitive, but the margin against Xilinx IPs is modest (10–16% ATP); its main advantage is against pure LUT-only academic designs, where the gap is one to two orders of magnitude. The adder/subtractor remains the main weakness of the set and the methodology in Section 4.1, which delimits the scope of the conclusions.

From the data above, a practical rule for choosing between the two synthesis modes can be stated in terms of the FPGA resource budget. The DSP-inferred mode is preferable whenever DSP slices are available, because it delivers the best ATP at a cost of two DSP slices per multiplier or MAC while keeping the LUT footprint minimal. The LUT-only mode of the same source costs only 4–6% more area and 14–32% more ATP in the multiplier ($\approx 2\%$ area and 14% ATP in the MAC), in exchange for using no DSP slices. Quantitatively, a design instantiating N multiply-class units needs $2N$ DSP slices in DSP-inferred mode. The LUT-only mode should be selected for the $N - \lfloor DSP_{\text{avail}}/2 \rfloor$ units that exceed the available DSP budget DSP_{avail} , or for all of them when DSP slices must be reserved for other functions, when targeting devices without hardened DSP macros, or when cross-vendor portability is required. The FIR case study of Section 5 illustrates the trade-off: its DSP-inferred mapping uses 400 DSP slices (20.9% of the XCKU040), so a device with a tighter DSP budget would migrate part of the taps to the LUT-only mode at the moderate area/ATP premium quantified above, without any change to the VHDL source.

4.4. Numerical Validation

The arithmetic correctness of the proposed units was assessed against a MATLAB double-precision reference. As summarized in Section 4.1, each unit was exercised with 5000 input vectors per distribution, drawn from a discrete uniform distribution and from

a standard normal distribution, complemented by directed corner cases and FIR-derived sequences, and the same vectors were run both in Vivado behavioral simulation and on the physical devices through memory-backed input/output. For an operation $c = f(a, b)$, let $\hat{a}$ and $\hat{b}$ denote the encoded inputs and $\hat{c}$ the decoded unit output. Two error sources contribute to $|\hat{c} - c|$: the input quantization in the custom format and the rounding of the result mantissa. Under a first-order error-propagation analysis with RNE rounding, the total error is bounded by

$$|\hat{c} - c| \leq |\partial_a f| \cdot \text{ULP}(\hat{a}) + |\partial_b f| \cdot \text{ULP}(\hat{b}) + \frac{1}{2} \text{ULP}(\hat{c}), \quad (10)$$

where $\text{ULP}(\hat{x})$ is the weight of the least significant mantissa bit of $\hat{x}$, and $\partial_a f, \partial_b f$ are the partial derivatives of the operation, which scale the propagated input quantization. The third term is the worst-case rounding error of the operator itself.

For addition/subtraction, $|\partial_a f| = |\partial_b f| = 1$ and the bound reduces to

$$|\hat{c} - c| \leq \text{ULP}(\hat{a}) + \text{ULP}(\hat{b}) + \frac{1}{2} \text{ULP}(\hat{c}). \quad (11)$$

For multiplication, $\partial_a f = b$ and $\partial_b f = a$, the operator rounding remains bounded at $\frac{1}{2} \text{ULP}(\hat{c})$. The non-fused MAC is a special case: with two rounding stages (intermediate product and final sum), its operator-induced bound doubles to $\text{ULP}(\hat{c})$ in the common case $|P| \leq |c|$, as derived in Section 3 (Equation (5)).

On the entire test set done, all units stay within their respective bounds: the operator-induced error of the adder, multiplier, and conversion units is at or below $\frac{1}{2} \text{ULP}$ of the result, matching a correctly rounded RNE implementation, while the chained MAC stays at or below $\text{ULP}(\hat{c})$, matching its theoretical limit.

To provide statistical rather than worst-case-only evidence, Table 6 reports the mean, root-mean-square (RMS), and maximum operator-induced error, expressed in ULP of the result, over the two 5000-vector campaigns (discrete-uniform and standard-normal inputs) for the adder, multiplier, and MAC. The multiplier and adder errors are confined to $[0, 0.5] \text{ULP}$ with a mean near 0.25 ULP and an RMS near 0.29 ULP, i.e., the uniform-rounding statistics expected from a correctly rounded RNE operator. For the chained MAC, the figures are given in the regime $|P| \leq |R|$ (the common case for which the 1 ULP bound was derived), where the error stays at or below 1 ULP with a mean near 0.28 ULP. The complementary cases, in which the addend nearly cancels the product ($|P| > |R|$, about 22–27% of the random vectors), exhibit a larger relative error in ULP of the much smaller result, exactly as predicted by the chained-rounding analysis of Section 3. This catastrophic-cancellation regime is precisely what a fused multiply–add would mitigate, and motivates the FMA variant left as future work. Figure 7 shows the corresponding error distributions for the multiplier and the MAC.

Table 6. Operator-induced error in ULP of the result over the 5,000-vector validation campaigns.

Unit	Uniform			Normal		
	Mean	RMS	Max	Mean	RMS	Max
Multiplier	0.249	0.288	0.500	0.250	0.289	0.500
Add/Sub	0.181	0.274	0.500	0.187	0.277	0.500
MAC	0.277	0.333	0.995	0.286	0.344	0.994

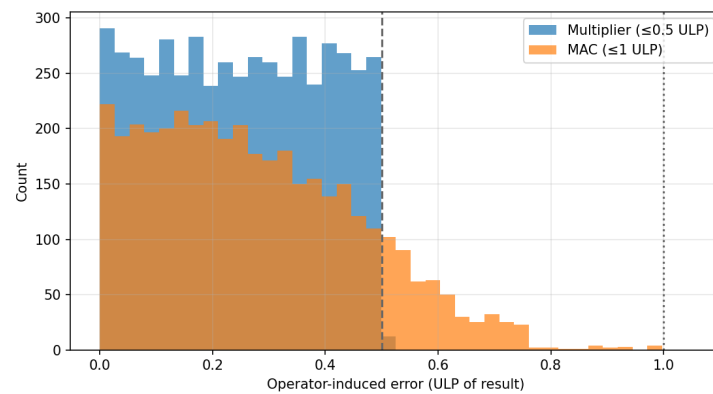


Figure 7. Distribution of the operator-induced error (in ULP of the result) for the multiplier and the chained MAC over the uniform campaign. The multiplier stays within 0.5 ULP and the MAC within 1 ULP in the $|P| \leq |R|$ regime.

5. DSP Application: Impulse Response Filter

A direct-form serial FIR filter was used as a representative DSP benchmark to evaluate the proposed unit set under demanding FPGA routing and resource-utilization conditions. The design incorporates 200 coefficients and an input-sequence generator, with the resulting implementation metrics reported in Table 7.

Table 7. Ultrascale resource utilization of the FIR filter implementation.

Resource	Utilization	Utilization (%)	Available
LUTs	183,375	75.6	242,400
FFs	125,339	25.8	484,800
DSPs	400	20.9	1920
LUTRAMs	7002	6.3	112,800
BRAMs	6	1.0	600

The implementation closes timing at 300 MHz with LUT utilization above 75%, confirming that the proposed units do not become a routing or timing bottleneck when integrated in a non-trivial DSP datapath. This is primarily a timing-closure and resource-scaling check. To additionally assess the signal-processing behavior, the numerical accuracy of the filter response is characterized below in terms of output signal-to-quantization-noise ratio (SQNR) and magnitude frequency response, comparing the proposed floating-point arithmetic against a fixed-point and an IEEE-754 single-precision implementation of the same filter. A full head-to-head comparison against a Xilinx-IP-based realization of the complete filter system is left as future work in Section 6.

Numerical Behavior of the FIR Filter Using the Proposed Format

The 200-tap filter was evaluated with a bit-accurate model of the proposed units that reproduces the $0.f$ format, the RNE rounding, the saturation policy, and the non-fused chained MAC (two roundings per tap). This model reproduces the same per-operation behavior that was validated on hardware at the unit level in Section 4.4. The FIR-level figures are therefore obtained by propagating that bit-accurate behavior through the 200-tap datapath. The filter output was compared against a double-precision reference for three arithmetics, parameterized to a common 32-bit word: the proposed floating-point format ($n_e = 8$, $n_f = 24$), a 32-bit fixed-point format (Q1.30), and IEEE-754 single precision.

Table 8 reports the output SQNR as a function of the input level, and Figure 8 plots it over a wider dynamic range. At full scale the 32-bit fixed-point format attains the highest SQNR because its 30 fractional bits exceed the 24-bit significand precision of the floating-

point formats. However, the fixed-point SQNR degrades by approximately 1 dB per dB of input attenuation, whereas the floating-point SQNR stays essentially constant across the whole range, owing to the exponent that preserves relative precision. The two formats cross over near -50 dBFS: below that level the proposed floating-point unit set outperforms the equal-width fixed-point implementation, which is precisely the wide-dynamic-range regime that motivates floating-point arithmetic in radar and communication DSP. As expected from their identical 24-bit significand precision, the proposed format and IEEE-754 single precision yield the same SQNR. Figure 9 shows that the magnitude frequency response of the three arithmetics is visually indistinguishable from the double-precision reference across the passband and stopband, confirming that the floating-point datapath does not distort the filter response.

Table 8. FIR output SQNR (dB) versus input level, for the proposed floating-point format, a 32-bit fixed-point (Q1.30) implementation, and IEEE-754 single precision, all referred to a double-precision reference.

Input Level	Proposed FP	Fixed-Point 32-bit	IEEE-754 Single
0 dBFS	131.8	162.7	131.8
-40 dBFS	131.8	139.6	131.8
-80 dBFS	131.8	99.6	131.8

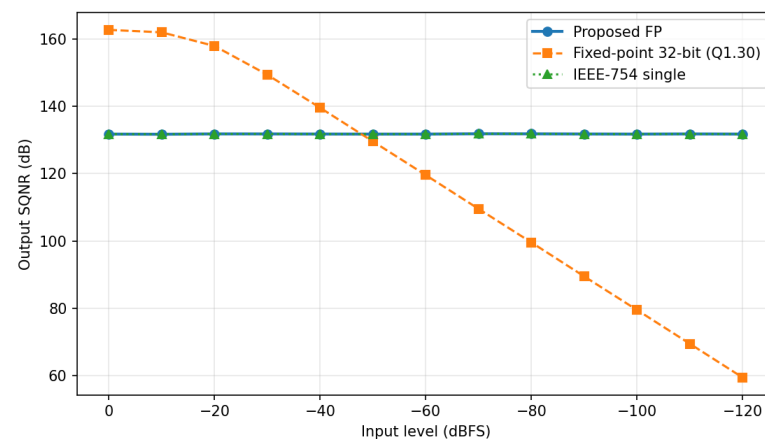


Figure 8. FIR output SQNR versus input level.

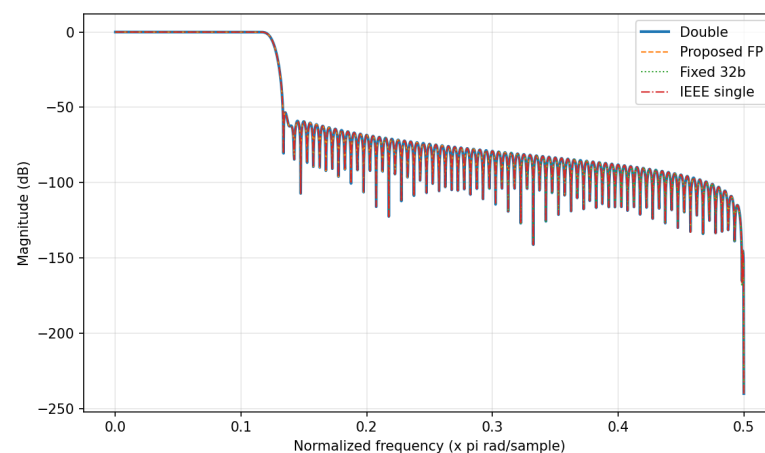


Figure 9. FIR magnitude frequency response for the proposed floating-point, fixed-point, and IEEE-754 single-precision implementations, overlaid on the double-precision reference.

6. Conclusions

This work presented a parameterizable floating-point arithmetic unit set tailored to FPGA-based DSP datapaths, including pipelined adder/subtractor, multiplier, MAC, and fixed-to-float/float-to-fixed converters. Two design choices distinguish the format from strict IEEE-754: configurable mantissa and exponent widths set at synthesis time and a $0.f$ significand encoding that modifies corner-case logic at the cost of one additional mantissa bit. The format is IEEE-754-inspired but not fully compliant as special values (NaN, $\pm\infty$) and subnormals are not implemented and out-of-range results are saturated to predefined constants. The whole unit set is described in portable VHDL and supports two synthesis strategies, DSP-inferred and LUT-only, making it portable to any FPGA family or ASIC target.

Tested on Artix-7 (28 nm) and Kintex Ultrascale (20 nm), parameterized to single-precision widths, the proposed multiplier, MAC, and conversion blocks improve ATP by 51–198% over Xilinx IP cores and by 27% to over an order of magnitude against recent academic designs. The proposed adder/subtractor improves over academic works (8–680%) and most Xilinx IPs (2–17%) but is outperformed by HS-R and HS-P (16–31%). It is included for design coherence and as a resource-efficient alternative to recent academic designs rather than a strict improvement over all vendor IPs. Numerical validation against MATLAB confirms that the absolute error stays below 0.5 ULP (1 ULP for the MAC unit) of the operator result on every tested vector. A 200-tap FIR filter closes timing at 300 MHz with 76% LUT utilization. For this filter, the output SQNR and magnitude frequency response were additionally characterized against fixed-point and IEEE-754 single-precision implementations: the proposed format keeps an almost constant SQNR across the input dynamic range and outperforms an equal-width 32-bit fixed-point implementation below approximately -50 dBFS, while preserving the filter frequency response.

Direct extensions of this work include a thin compatibility wrapper for full IEEE-754 special-value handling, a fused multiply-add MAC variant, a direct quantitative comparison against a functionally equivalent $1.f$ implementation to isolate the area, timing, and power impact of the $0.f$ encoding, characterization at half/bfloat16 and double precision with accuracy-versus-area benchmarks on representative DSP kernels (FFT, FIR, matrix factorization) and extending the SQNR comparison against fixed-point and IEEE-754 baselines—reported here for the FIR filter—to those additional kernels and to a full Xilinx-IP-based filter realization, and the addition of non-linear operators (logarithm, inverse square root, arctangent) and complex-number special arithmetic for higher-level DSP algorithms.

Author Contributions: Conceptualization, F.F., J.P.Q. and J.M.C.P.; methodology, F.F., J.P.Q. and J.M.V.S.; software, F.F. and J.P.Q.; validation, F.F. and J.M.V.S.; formal analysis, M.D.V.-P. and C.Q.G.; investigation, F.F., J.P.Q. and J.M.C.P.; resources, J.M.C.P. and J.M.V.S.; writing, original draft preparation, F.F. and J.P.Q.; writing, review and editing, M.D.V.-P., C.Q.G.; supervision, J.M.C.P., M.D.V.-P. and C.Q.G. All authors have read and agreed to the published version of the manuscript.

Funding: Funding for Article Processing Charges: Universidade de Vigo/CISUG.

Data Availability Statement: The raw data supporting the conclusions of this article will be made available by the authors on request.

Conflicts of Interest: The authors declare no conflicts of interest.

Abbreviations

The following abbreviations are used in this manuscript:

ADC	analog-to-digital converter
ASIC	application-specific integrated circuit

ATP	area-throughput-power
BRAM	block RAM
DAC	digital-to-analog converter
DSP	digital signal processing
EDA	electronic design automation
FF	flip-flop
FFT	Fast Fourier transform
FIFO	first-in, first-out
FIR	finite impulse response
FMA	fused multiply-add
FPGA	field programmable gate array
GRS	guard, round, and sticky
HLS	high-level synthesis
IIR	infinite impulse response
IP	intellectual property
LUT	look-up table
LUTRAM	look-up table RAM
LZA	leading zero anticipation
MAC	multiply-accumulate
MFMA	multi-fused multiply-accumulate
MSB	most significant bit
NaN	not a number
RMS	root-mean-square
RNE	round-to-nearest-ties-to-even
SQNR	signal-to-quantization-noise ratio
ULP	unit in the last place
VHDL	VHSIC hardware description language

References

1. Menard, D.; Caffarena, G.; Lopez, J.A.; Novo, D.; Sentieys, O. Fixed-Point Refinement of Digital Signal Processing Systems. In *Digitally Enhanced Mixed Signal Systems*; The Institution of Engineering and Technology: Stevenage, UK, 2019; Chapter 1, pp. 1–37. [\[CrossRef\]](#)
2. Yan, C.; Zhao, X.; Zhang, T.; Ge, J.; Wang, C.; Liu, W. Design of High Hardware Efficiency Approximate Floating-Point FFT Processor. *IEEE Trans. Circuits Syst. I Regul. Pap.* **2023**, *70*, 4283–4294. [\[CrossRef\]](#)
3. Paz, P.; Garrido, M. A 12.8-GS/s 32-Parallel 1 Million-Point FFT. In Proceedings of the IEEE Conference on Design of Circuits and Integrated Systems (DCIS), Catania, Italy, 13–15 November 2024; pp. 336–341. [\[CrossRef\]](#)
4. Hussain, A.A.; Tayem, N.; Soliman, A.H.; Radaydeh, R.M. FPGA-Based Hardware Implementation of Computationally Efficient Multi-Source DOA Estimation Algorithms. *IEEE Access* **2019**, *7*, 88845–88858. [\[CrossRef\]](#)
5. Zhang, X.W.; Zuo, L.; Li, M.; Guo, J.X. High-Throughput FPGA Implementation of Matrix Inversion for Control Systems. *IEEE Trans. Ind. Electron.* **2021**, *68*, 6205–6216. [\[CrossRef\]](#)
6. Hu, T.; Li, X.; Yu, X.; Ren, S.; Yan, L.; Bai, X.; Xu, Z.; Zhu, S. A Novel Fully Hardware-Implemented SVD Solver Based on Ultra-Parallel BCV Jacobi Algorithm. *IEEE Trans. Circuits Syst. II Express Briefs* **2022**, *69*, 5114–5118. [\[CrossRef\]](#)
7. Lin, A.Y.; Gugel, K.S.; Principe, J.C. Feasibility of Fixed-Point Transversal Adaptive Filters in FPGA Devices with Embedded DSP Blocks. In Proceedings of the IEEE International Workshop on System-on-Chip for Real-Time Applications, Calgary, AB, Canada, 2 July 2003; pp. 157–160. [\[CrossRef\]](#)
8. Soh, J.; Wu, X. An FPGA-Based Unscented Kalman Filter for System-On-Chip Applications. *IEEE Trans. Circuits Syst. II Express Briefs* **2017**, *64*, 447–451. [\[CrossRef\]](#)
9. Flores, F.; Pena, M.D.V.; Sanchez, J.M.V.; Pazo, J.M.C.; Grana, C.Q. Evaluation of the Versal Intelligent Engines for Digital Signal Processing Basic Core Units. In Proceedings of the IEEE Conference on Design of Circuits and Integrated Systems (DCIS), Catania, Italy, 13–15 November 2024. [\[CrossRef\]](#)
10. Flores, F.; Sanchez, O.L.; de Ojeda, L.J.R.; Pena, M.D.V.; Sanchez, J.M.V. Acceleration of a Compute-Intensive Algorithm for Power Electronic Converter Control Using Versal AI Engines. In Proceedings of the IEEE Conference on Design of Circuits and Integrated Systems (DCIS), Catania, Italy, 13–15 November 2024. [\[CrossRef\]](#)
11. *IEEE Std 754-2019 (Revision of IEEE Std 754-2008)*; IEEE Standard for Floating-Point Arithmetic. IEEE: New York, NY, USA, 2019; pp. 1–84. [\[CrossRef\]](#)

12. Ternovoy, E.; Popov, M.G.; Kaleev, D.V.; Savchenko, Y.V.; Pereverzev, A.L. Comparative Analysis of Floating-Point Precision of IEEE 754 and POSIT Standards. In Proceedings of the IEEE Conference of Russian Young Researchers in Electrical and Electronic Engineering (EIConRus), Moscow, Russia, 27–30 January 2020; pp. 1883–1886. [\[CrossRef\]](#)
13. Forget, L.; Uguen, Y.; de Dinechin, F. Comparing POSIT and IEEE-754 Hardware Cost. 2021. Available online: <https://hal.science/hal-03195756v3> (accessed on 20 April 2026).
14. Malkapur, S.B.; Rajput, R.P. Design of Generic Floating Point Pipeline Based Arithmetic Operation for DSP Processor. In Proceedings of the International Conference on Inventive Research in Computing Applications (ICIRCA), Coimbatore, India, 15–17 July 2020; pp. 1059–1064. [\[CrossRef\]](#)
15. Yacoub, M.H.; Ismail, S.M.; Said, L.A. Hardware Realization of High-Speed Area-Efficient Floating Point Arithmetic Unit on FPGA. In Proceedings of the 2024 International Conference on Machine Intelligence and Smart Innovation (ICMISI), Alexandria, Egypt, 12–14 May 2024; pp. 190–193. [\[CrossRef\]](#)
16. Chabini, N.; Wolf, M.C.; Beguenane, R. LUT-Based Multipliers for IEEE-754 Floating Point Arithmetic on FPGAs. In Proceedings of the 2024 IEEE 15th Annual Ubiquitous Computing, Electronics and Mobile Communication Conference (UEMCON), New York, NY, USA, 17–19 October 2024; pp. 698–701. [\[CrossRef\]](#)
17. Singh, U.; Mundliya, R.; Maharana, R.; Jajodia, B. Floating Point Multipliers on FPGAs. In Proceedings of the 2025 Devices for Integrated Circuit (DevIC), West Bengal, India, 5–6 April 2025; pp. 702–707. [\[CrossRef\]](#)
18. Shirke, M.; Chandrababu, S.; Abhyankar, Y. Implementation of IEEE 754 Compliant Single Precision Floating-Point Adder Unit Supporting Denormal Inputs on Xilinx FPGA. In Proceedings of the 2017 IEEE International Conference on Power, Control, Signals and Instrumentation Engineering (ICPCSI), Chennai, India, 21–22 September 2017; pp. 408–412. [\[CrossRef\]](#)
19. Zhou, B.; Wang, G.; Jie, G.; Liu, Q.; Wang, Z. A High-Speed Floating-Point Multiply-Accumulator Based on FPGAs. *IEEE Trans. Very Large Scale Integr. (VLSI) Syst.* **2021**, *29*, 1782–1789. [\[CrossRef\]](#)
20. Liu, H.; Lu, X.; Yu, X.; Li, K.; Yang, K.; Xia, H.; Li, S.; Deng, T. A 3-D Multi-Precision Scalable Systolic FMA Architecture. *IEEE Trans. Circuits Syst. I Regul. Pap.* **2025**, *72*, 265–276. [\[CrossRef\]](#)
21. Li, K.; Hao, X.; Ma, Z.; Yu, F.; Zhang, B.; Xing, Q. A Fast Floating-Point Multiply-Accumulator Optimized for Sparse Linear Algebra on FPGAs. *IEEE Trans. Very Large Scale Integr. (VLSI) Syst.* **2025**, *33*, 2592–2596. [\[CrossRef\]](#)
22. Perri, S.; Spagnolo, F.; Frustaci, F.; Corsonello, P. Design of Leading Zero Counters on FPGAs. *IEEE Embed. Syst. Lett.* **2023**, *15*, 149–152. [\[CrossRef\]](#)
23. AMD. 7 Series FPGAs Data Sheet: Overview (DS180). 2020. Available online: https://docs.amd.com/v/u/en-US/ds180_7Series_Overview (accessed on 16 September 2025).
24. AMD. UltraScale Architecture and Product Data Sheet: Overview (DS890). 2025. Available online: <https://docs.amd.com/v/u/en-US/ds890-ultrascale-overview> (accessed on 16 September 2025).

Disclaimer/Publisher’s Note: The statements, opinions and data contained in all publications are solely those of the individual author(s) and contributor(s) and not of MDPI and/or the editor(s). MDPI and/or the editor(s) disclaim responsibility for any injury to people or property resulting from any ideas, methods, instructions or products referred to in the content.