# IMPLEMENTATION OF AN 8-BIT DIVIDER WITH NORMALIZED MANTISSA IN VHDL: AN ANALYSIS OF LOGICAL AND ELECTRICAL INTEGRITY IN AN EMBEDDED SYSTEM

Emiliano Vargas Trejo and Ricardo Francisco Martínez González

Department of Electrical–Electronic Engineering, Tecnológico Nacional de México – Instituto Tecnológico de Veracruz

***ABSTRACT***

*This paper describes the design and physical implementation of an arithmetic division unit for a custom 8-bit floating-point format. The proposed architecture employs a normalized mantissa with an implicit bit, thereby extending the effective precision to five bits. The development, carried out in VHDL and synthesized on an FPGA, incorporates dynamic normalization logic and exponent management in two's complement. Experimental validation, performed using an oscilloscope and quantization tests, yields a maximum relative error of 6.25% and confirms the system's operational stability thanks to the conditioning of the input logic levels. We conclude that the design constitutes a viable alternative for embedded systems with area and power constraints.*

***KEYWORDS***

*VHDL, floating point, FPGA, implicit bit, arithmetic divider.*

## 1. INTRODUCTION

In digital system design, representing real numbers presents a recurring dilemma: the trade-off between dynamic range, precision, and implementation cost. Fixed-point arithmetic, while simple and inexpensive [1], shows significant limitations when signals vary across several orders of magnitude [2]. At the opposite extreme, the IEEE 754 single-precision standard (32 bits) offers ample range and precision, but its implementation on an FPGA consumes considerable logic and memory resources [3]. This situation has driven the development of reduced-precision floating-point formats, particularly in areas such as portable instrumentation, embedded control systems, and quantized neural networks [4].

This work presents an arithmetic division unit for an 8-bit floating-point format with a normalized mantissa and an implicit bit. Our primary goal is to determine whether it is feasible to perform divisions reliably, with bounded error and resource consumption compatible with low-end FPGAs. Unlike other developments limited to simulation, this project includes physical implementation on a development board and verification of the input signals' electrical integrity.

The article is organized into four sections. Section 2 describes the format architecture, the division algorithm, and the details of the VHDL implementation. Section 3 presents the simulation results and the experimental measurements. Finally, Section 4 discusses the implications of the design and outlines possible lines of future work.

DOI: 10.5121/ijist.2026.16201 

# 2. DEVELOPMENT

The development of this digital divider was structured in phases, ranging from the design of the reduced-precision mathematical model to its on-silicon validation. This ensures that each stage complies with the logic synthesis criteria for programmable logic devices.

## 2.1. 8-bit floating-point format

The format defined in this work uses an 8-bit word with the following allocation: bit 7 is assigned to the sign (S), bits 6 to 4 to the exponent (E) in two's complement, and bits 3 to 0 to the mantissa (M). The choice of two's complement for the exponent avoids the need for a bias and simplifies the addition and subtraction logic in hardware [5].

To increase the effective resolution without widening the word, we adopt the convention of a normalized mantissa with an implicit bit. Under this convention, every nonzero number is represented in the form $(1.M)_2$, where the '1' preceding the binary point is not stored but is restored internally during arithmetic operations. In this way, a stored 4-bit mantissa is equivalent to an effective 5-bit mantissa [6].

## 2.2. Division algorithm

The division of two floating-point numbers can be decomposed into three independent operations: sign computation, exponent subtraction, and mantissa division. An XOR gate applied to the signs of the operands obtains the sign of the quotient. The resulting exponent is the subtraction of the dividend exponent and the divisor exponent. The quotient's mantissa is calculated by dividing the expanded five-bit mantissas.

Mantissa division is implemented with a combinational algorithm based on successive subtraction. Figure 1 details the logical control sequence in a flowchart. Since the obtained quotient may not be normalized, a dynamic normalization stage is added: if the quotient's most significant bit is zero, the mantissa is shifted left by one bit and the exponent is decremented. This process repeats until the implicit bit returns to the correct position [7].

## 2.3. VHDL implementation

The VHDL description follows a modular structure. The most representative fragments are reproduced below.

Code 1. Implicit-bit restoration.

```
mantisa_real_A <= '1' & num_A(3 downto 0);
mantisa_real_B <= '1' & num_B(3 downto 0);
```

Concatenating the '1' bit with the four stored bits reconstructs the full five-bit mantissa. This operation, performed at the beginning of the computation, is indispensable for the division to operate on the number's real value [8].

Code 2. Signed exponent subtraction.

```
exp_res <= signed(exp_A) - signed(exp_B);
```

The use of the signed type from the numeric_std library allows the synthesizer to infer a binary subtractor with a two's complement interpretation. This handles negative exponents without additional logic [9].

Code 3. Dynamic normalization.

```
if mantisa_div(4) = '0' then
   sal_mantisa <= std_logic_vector(shift_left(unsigned(mantisa_div), 1));
   sal_exp     <= std_logic_vector(exp_res - 1);
```

If the most significant bit of the quotient mantissa is zero, the vector is shifted left by one position and one unit is subtracted from the exponent. The control logic repeats this operation until reaching the normalized form [10]

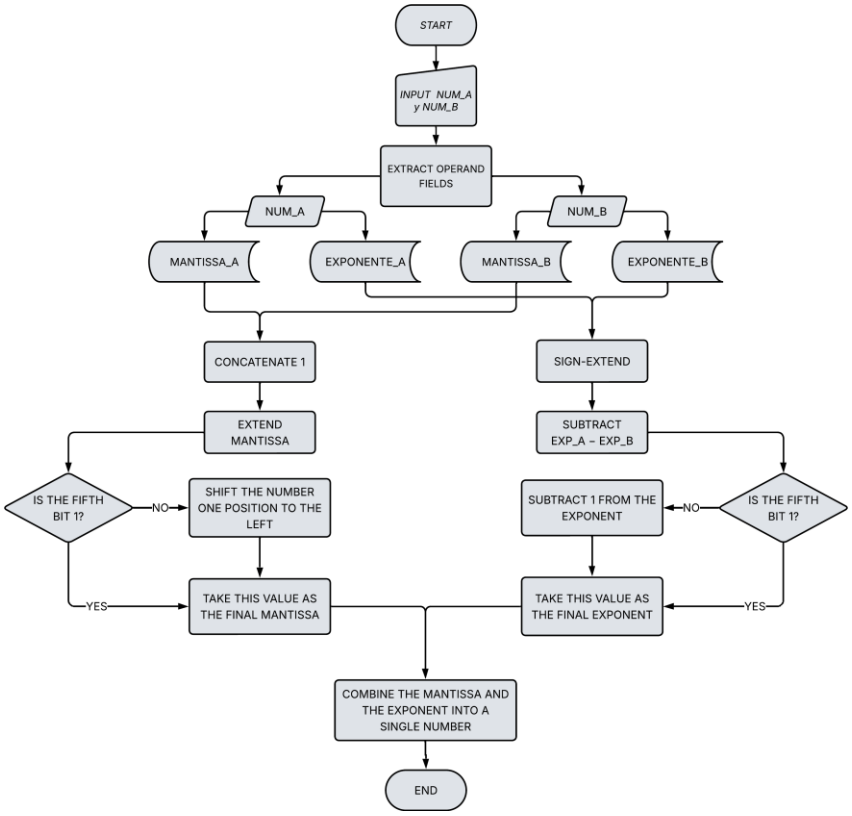


Figure 1. Flowchart of the 8-bit floating-point division algorithm.

## 2.4. Physical implementation and signal conditioning

The design was synthesized for an FPGA from the Cyclone IV family. DIP switches are used to enter the operands, and a row of LEDs displays the result. Table 1 details the pin assignment used.

Table 1. FPGA pin assignment.

| **Component** | **Technical function** | **GPIO Label** | **Logic type** |
|---|---|---|---|
| DIP Switch A | Dividend (8 bits) | GPIO 0 – GPIO 7 | Pull-down input |
| DIP Switch B | Divisor (8 bits) | GPIO 8 – GPIO 15 | Pull-down input |
| LED bar | Quotient (8 bits) | GPIO 16 – GPIO 23 | Output |
| Supply | Reference level | VCC 3.3V | Reference |

To ensure stable logic levels at the inputs, a DNS6000AUD regulator was incorporated to filter fluctuations caused by the mechanical bounce of the switches. This stage is critical, since an input below 2.5 V may be interpreted as a low level even when the switch is in the ON position [11].

The complete breadboard assembly is illustrated in Figure 2. The two DIP switches used to enter the operands (on the left), the LED bar showing the quotient result (on the right), and the FPGA as the central processing element can be identified. This arrangement allows for direct interaction and immediate visual verification of the system's operation.

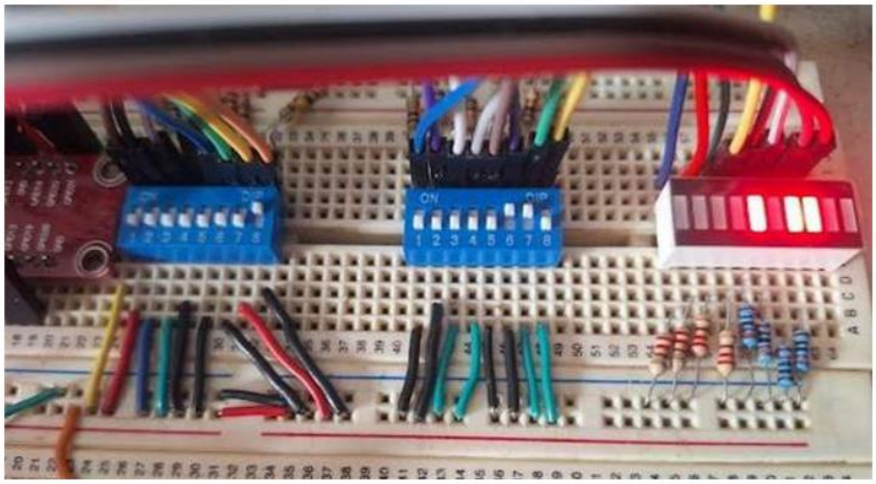

Figure 2. Physical implementation on a breadboard. The two DIP switches for the input operands (left), the LED bar for the quotient output (right), and the FPGA development board are shown.

## 3. RESULTS

The floating-point divider was validated in two stages: first through functional simulation in a VHDL environment and, subsequently, through direct measurement on the synthesized FPGA. This approach made it possible to compare the expected logical behavior with the readings obtained under real operating conditions, as well as to assess the system's sensitivity to imperfections inherent in the physical setup.

### 3.1. Functional tests

Four test cases were selected to cover representative situations: division of positive numbers, division with a negative result, division requiring automatic normalization, and division of a number by itself. Table 2 summarizes the results obtained.

Table 2. Experimental results.

| Case | A (binary) | B (binary) | Decimal operation | Expected (binary) | Obtained |
|---|---|---|---|---|---|
| 1 | 0 001 0100 | 0 000 1000 | 2.5 / 1.5 | 0 001 1010 | Matches |
| 2 | 1 010 0000 | 0 001 0000 | –1.0 / 2.0 | 1 001 0000 | Matches |
| 3 | 0 111 1111 | 0 111 0000 | 7.75 / 4.0 | 0 001 1111 | Matches |
| 4 | 0 000 0001 | 0 000 0001 | 1.0625 / 1.0625 | 0 000 0000 | Matches |

In all cases, the observed output matched the expected value. Case 1 required the intervention of the normalization block, which confirms the correct operation of the shifting logic and exponent

adjustment. Case 4, in turn, verifies the handling of a number divided by itself: the mantissa quotient is $1.0000_2$ and the exponent subtraction is zero, so the final result is exactly 1.0.

### 3.2. Signal integrity measurement

A digital oscilloscope was used to monitor one of the input lines during switch toggling. The waveform showed a clean transition, with a stabilization time of less than 5 ms and no appreciable oscillations above the 2.5 V threshold. This behavior is attributed to the conditioning performed with the DNS6000AUD regulator. No read errors attributable to mechanical bounce were detected.

### 3.3. Quantization error analisis

In a floating-point system, the number of mantissa bits determines the maximum relative error. For an effective mantissa of *k* bits, the machine epsilon (ε) is defined as

$$\varepsilon = 2^{-(k-1)} \quad \text{Eq. 1}$$

In this design, k = 5; therefore, $\varepsilon = 2^{-4} = 0.0625$ (6.25 %). This value is consistent with the experimental results. In the 2.5 / 1.5 operation, for example, the real quotient is 1.666... and the represented value is 1.625, which yields a relative error of 2.5%, below the theoretical maximum [12].

The format's dynamic range extends from $2^{-4} \times 1.0 = 0.0625$ to $2^3 \times 1.9375 = 15.5$. This range is sufficient for numerous instrumentation and control applications.

## 4. CONCLUSIONS

We have presented the design, implementation, and validation of an eight-bit floating-point divider with a normalized mantissa. The results show that it is possible to perform complex arithmetic operations with a short word format, provided that techniques such as the implicit bit and dynamic normalization are employed.

The maximum relative error of 6.25% is acceptable in applications where the priority is not extremely high accuracy, but rather the ability to process signals spanning variable orders of magnitude at a reduced hardware cost. Likewise, we have verified that the electrical stability of the inputs is a determining factor for the system's reliability, which justifies the inclusion of conditioning stages.

This work opens development lines toward more complex arithmetic units—multipliers, adders, or fused multiply-add units—as well as the exploration of this format in stream-processing architectures for FPGA-based machine learning applications.

### REFERENCES

[1] [1] M. M. Mano y C. R. Kime, Logic and Computer Design Fundamentals, 4.ª ed. Upper Saddle River, NJ: Pearson, 2008.
[2] [2] D. A. Patterson y J. L. Hennessy, Computer Organization and Design: The Hardware/Software Interface, 5.ª ed. Waltham, MA: Morgan Kaufmann, 2014.
[3] [3] IEEE Computer Society, IEEE Standard for Floating-Point Arithmetic, IEEE Std 754-2019, 2019.
[4] [4] S. Brown y Z. Vranesic, Fundamentals of Digital Logic with VHDL Design, 3.ª ed. Nueva York: McGraw-Hill, 2009.
[5] [5] J. F. Wakerly, Digital Design: Principles and Practices, 4.ª ed. Upper Saddle River, NJ: Pearson, 2006.

[6] [6] B. Parhami, Computer Arithmetic: Algorithms and Hardware Designs, 2.ª ed. Nueva York: Oxford University Press, 2010.
[7] [7] K. Hwang, Computer Arithmetic: Principles, Architecture, and Design. Nueva York: Wiley, 1979.
[8] [8] P. P. Chu, *FPGA Prototyping by VHDL Examples: Xilinx Spartan-3 Version*. Hoboken, NJ: Wiley, 2008.
[9] [9] V. A. Pedroni, Circuit Design and Simulation with VHDL, 2.ª ed. Cambridge, MA: MIT Press, 2010.
[10] [10] S. Kilts, Advanced FPGA Design: Architecture, Implementation, and Optimization. Hoboken, NJ: Wiley, 2007.
[11] [11] C. H. Roth Jr. y L. L. Kinney, Fundamentals of Logic Design, 7.ª ed. Stamford, CT: Cengage Learning, 2014.
[12] [12] D. M. Harris y S. L. Harris, Digital Design and Computer Architecture, 2.ª ed. Waltham, MA: Morgan Kaufmann, 2013.

## AUTHORS

**Ricardo Francisco Martínez González** was born in Veracruz, Mexico, in 1983. He received the B.S. degree in Electronic Engineering from the Instituto Tecnológico de Veracruz, afterwards the M.Sc. and Ph.D. degrees in Electronics from the Instituto Nacional de Astrofísica, Óptica y Electrónica (INAOE).

He has worked as a developer of online software platforms and mobile applications for the private sector. He is currently with the Department of Electrical-Electronic Engineering at the Tecnológico Nacional de México (campus Veracruz), where he serves as President of the Electronic Engineering Academy. He has directed various research projects in computer vision, control systems, instrumentation, artificial intelligence, steganography, and energy efficiency, among others. He is the author of several scientific articles in the aforementioned fields.

**Emiliano Vargas Trejo** was born in Reynosa, Mexico, in 2003. He is currently pursuing a B.S. degree in Electronic Engineering at the Instituto Tecnológico de Veracruz. He has been actively involved in technological outreach projects and academic mentoring aimed at fostering engineering vocations in younger generations. His research interests include embedded systems, automation, and the Internet of Things (IoT).