# An efficient dual-mode floating-point Multiply-Add Fused Unit

**Conference Paper** · January 2011

DOI: 10.1109/ICECS.2010.5724440 · Source: IEEE Xplore

**3 authors:**

Konstantinos Manolopoulos
National and Kapodistrian University of Athens
**27** PUBLICATIONS **186** CITATIONS
SEE PROFILE

D. Reisis
National and Kapodistrian University of Athens
**87** PUBLICATIONS **966** CITATIONS
SEE PROFILE

V.A. Chouliaras
Loughborough University
**87** PUBLICATIONS **531** CITATIONS
SEE PROFILE

# An Efficient Dual-Mode Floating-Point Multiply-Add Fused Unit

K. Manolopoulos, D. Reisis
Electronics Laboratory, Department of Physics
National and Kapodistrian University of Athens
Athens, Greece
Email: kmanolo@phys.uoa.gr, dreisis@phys.uoa.gr

V.A. Chouliaras
Department of Electronic and Electrical Engineering,
Loughborough University, Loughborough,
LEICS, LE11 3TU, UK
Email: V.A.Chouliaras@lboro.ac.uk

***Abstract*—Multiply-Add Fused (MAF) units play a key role in the processor's performance for a variety of applications. Aiming at improving the MAF functionality this paper presents a dual-mode MAF architecture, which is able to perform either one double-precision or two single-precision operations in parallel. The design attains low latency by following a dual-path approach and by combining final addition with rounding. The organization performs a MAF instruction in three cycles, while single floating-point addition in two cycles. The design has been validated and implemented with TSMC 0.13um.**

## I. INTRODUCTION

Floating-Point calculations are imposed by a large number of modern applications such as high performance computers, graphic acceleration, FFTs, DSPs etc. [1]-[3].Moreover, applications in the area of 3D graphics require the parallel execution of floating point operations, which usually involve operands with both single and/or double precision format.

To meet these demands, novel processor designs tend to provide Floating-Point Units (FPUs), which support not only single but also double precision operations. The fact that two of the most commonly used calculations in Floating-Point arithmetic are floating point addition and multiplication led researchers to replace the traditional FPUs with Multiply-Add-Fused (MAF) units, which have been used in many applications.

A Multiply-Add Fused Unit realizes the $(A \times B)+C$ single-instruction equation. Instead of using distinct floating-point multiplier and floating-point adder, a MAF unit combines both in order to reduce the latency of dot-product calculations. Using the MAF unit instead of separating floating point multiplication and addition provides a number of advantages. First, the MAF unit demonstrates an improved precision. Since it requires a single rounding operation, at the final stage of the calculations, both multiplication and addition are performed with full precision, providing therefore improved accuracy. Second, the MAF unit performs normalization and rounding only once every two consecutive floating-point operations. Therefore, it requires reduced hardware complexity and reveals an improved latency, when compared to the traditional implementation of performing first a multiplication and then an addition. Finally, the MAF can still perform standard floating-point operations by using constants. Making $C = 0.0$ or $B = 1.0$ the MAF unit performs floating-point multiplication or floating-point addition respectively.

Apart the above advantages though, the combination of two operations in a single instruction has certain drawbacks. The use of constants to calculate operations of single multiplication and addition in the MAF unit imposes greater latencies, compared to their execution in a single floating-point multiplier or adder respectively. Moreover, the MAF unit increases the complexity of the hardware and prevents the parallel execution of floating-point additions and multiplications.

This paper presents a dual-mode MAF architecture that performs either one double-precision or two single-precision operations in parallel. The design demonstrates an improved performance by reducing the latency of the MAF instruction and of the floating-point addition. The idea is to bypass the multiplier when floating-point addition is performed. To achieve this, the alignment shifter is removed from the first stage and a dual-path organization is utilized. Moreover, normalization is performed before rounding so as to combine rounding with the final addition, thus further reducing the latency. Even though the presented architecture utilizes techniques previously introduced, this is the first time that these techniques have been incorporated in a multiple precision MAF unit. Hence, the main contribution of this paper is that it presents an efficient multiple precision MAF unit, evaluates its performance and identifies the hardware cost.

The remainder of this paper is organized as follows: Section 2 reviews the related work and describes the structure of a basic MAF unit. Sections 3 presents the general architecture of the proposed dual-mode MAF unit and a detailed description of the basic MAF modules. Section 4 presents the implementation results and finally, Section 5 concludes this paper.

## II. CONVENTIONAL MAF UNIT

Since the Multiply-Add Fused unit of the IBM POWER1 processor MAF architectures have been studied and implemented [4]-[8]. [4] compares complexities between single and dual-pass through a multiply array for a fused multiply add floating point unit. In [5] a MAF floating-point unit with signed digit addition is presented. [6] describes a MAF floating-point unit able to process denormalized number. In [7] a fully pipelined MAF unit combines the final addition with rounding. [8] presents a multiple precision floating-point MAF unit capable of performing either one double precision or two single precision operations.

As all the hitherto published MAF designs follow the same basic scheme, we outline here its typical features to use them as a point of reference for the proposed architecture. Figure 1 presents the block diagram of a typical floating-point MAF unit.

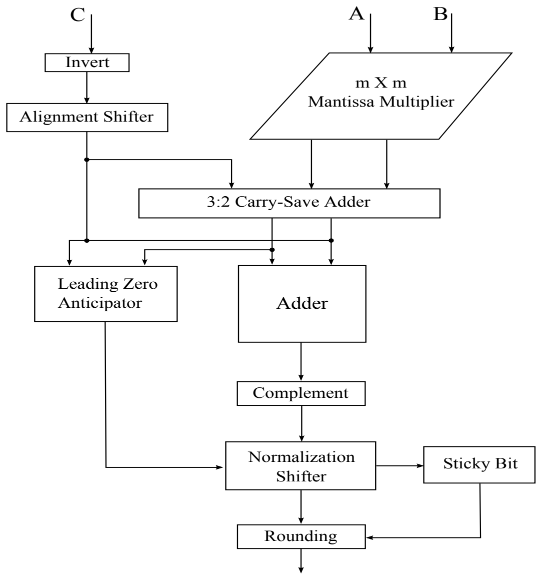

Fig. 1. Block diagram of a basic MAF architecture

The necessary steps to complete the Multiply-Add Fused operation, in double-precision format are the following:

1. Multiplication Stage: it uses a $53 \times 53$- bit multiplier for the significants of the operands A and B and produces the 106-bit result in a carry-save format. At the same time, the alignment of the third operand C takes place.

2. Addition Stage: it adds the product $A \times B$ and aligned operand C by using a 3:2 CSA and a 161-bit adder. The Leading Zero Anticipator unit computes, in parallel with the addition, the necessary shift amount for normalization.

3. Normalization and Rounding: The 161-bit result is normalized, based on the estimation of the LZA unit. Finally, rounding is performed and a possible post-normalization if it is required.

## III. The Proposed MAF Architecture

The proposed dual mode MAF unit supports two different IEEE precisions. It can perform either one double-precision or two single-precision instructions in parallel. Moreover, the proposed design follows a dual-path approach, which reduces the latency of the floating-point addition. Finally, it combines the final addition and rounding by anticipating the normalization before the addition and thus, it improves the latency of the MAF instruction.

Figures 2a and 2b show a 64-bit register stores either one double-precision or two single-precision operands. Figure

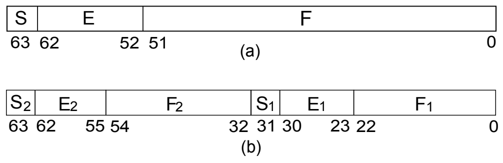

Fig. 2. (a) Double precision format, (b) Single precision format

3 depicts the architecture of the proposed dual-mode MAF unit. The design is pipelined with three stages. The first stage modifies the input operand bits and depending on the operation mode, it produces the proper mantissas, exponents and signs. Then, it performs the exponent processing and the mantissa multiplication. In the second stage, the datapath of the architecture is divided into two paths: the $Close$ path and the $Far$ path. The *Close* path handles instructions where a massive cancellation occurs, that is effective subtractions with exponents that differ at most by 1. The *Far* path is activated in the rest of the cases, which have distant exponents. The decision of the path to be chosen is based upon the input exponents and it is taken during the first stage. The activated path performs the alignment and normalization. The dual path in the proposed solution reduces the latency and it is still hardware efficient taking advantage of the fact that the computations are mutually exclusive. Finally, the third stage realizes the final addition and rounding in parallel by performing the normalization before the addition.

Even though the proposed design is based on the hardware of a double-precision dual-path MAF unit, it is enhanced to perform the execution of two single-precision instructions in parallel. For this purpose, several components have been combined and redesigned, and extra hardware has been introduced. Nevertheless, these extra hardware requirements still provide better area results, instead of duplicating the components for each precision mode. A number of multiplexers throughout the design manages the flow of the different precision vectors. The control signal *double* handles the multiplexers, along with other components of the design. For double-precision this control signal is set. The following of this section gives the details of the basic modules of the MAF unit.

### A. First Stage: Exponent Processing and Multiplication

*1) Exponent Processing:* The exponent processing determines the shift operations for the alignment and normalization. When double-precision is performed, the exponent difference equals $diff = expC - (expA + expB - 1023)$ and the shift amount is $sh = 56 - d = expA + expB - expC - 967$. In single precision mode the exponent difference is $diff = expC - (expA + expB - 127)$ and the shift amount corresponds to $sh = 27 - d = expA + expB - expC - 100$. The required hardware for processing the double-precision exponents has been extended to also handle the single-precision exponents.

*2) Multiplication:* The array multiplier of the MAF unit can either perform one double-precision multiplication or two single-precision multiplications in parallel. To accomplish

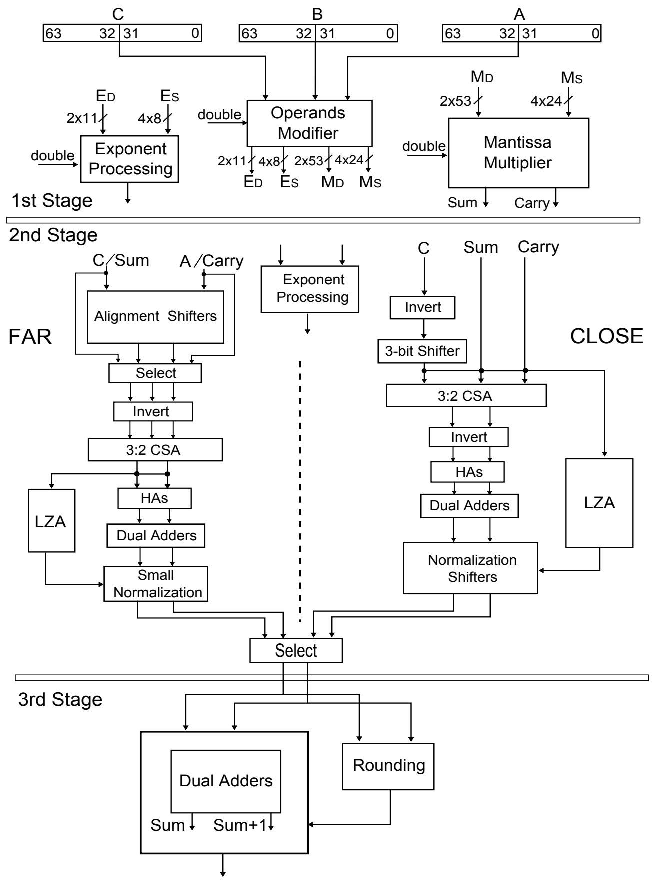

Fig. 3. Block diagram of the dual-mode MAF architecture

this task we use the sub-word partitioning technique [11]. By using this technique the MAF performs both traditional multiplication and subword parallel multiplication, while it uses for both the same counter tree and carry-propagate adder.

When performing a double-precision multiplication, the array multiplier of the MAF unit operates as a traditional $53 \times 53$ bit multiplier. On the other hand, when single-precision multiplication is required, the partial product bits within the regions labeled $Z_{24}$ (Figure 4) are not part of the subword-parallel multiplication. Therefore, these bits are set to zero to avoid altering the final result. This way, two $24 \times 24$ bit multiplications are performed in parallel.

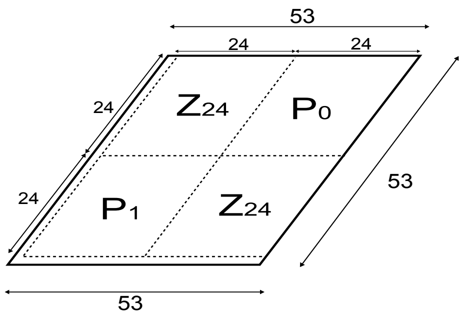

Fig. 4. Dual-mode multiplier

## B. Second Stage: Dual-Path Organization

The second stage of the MAF architecture performs the alignment and normalization. The dual-path organization has been introduced as an efficient designing technique of floating-point Adders and has also been used in designing MAF units [9], [10]. As mentioned above the datapath splits in two different paths, the *Far* path and the *Close* path. The *Far* path processes multiply-add and multiply-subtract instructions, of operands whose exponents difference is larger than 1 $(diff > 1)$. The *Close* path calculates the cases of effective multiply-subtractions, where the exponent difference lies between $\pm 1 (-1 \leq diff \leq +1)$

The advantage of splitting the datapath into two, is that it reduces the total latency of the design when it performs floating-point addition, by bypassing the first stage of the design. In order to achieve that, two adjustments have to take place. First, we duplicate the exponent processing logic at the beginning of the second stage. The hardware overhead required for this step is minimal, since the extra exponent processing module occupies a small amount of area. The second adjustment is that the alignment shifter needs to be placed after the multiplier, instead of placing it in parallel as it is in the classic MAF architecture. Changing the place of the alignment shifter though, affects the critical path of the MAF. This is because the critical path will now include two shifters instead of one (the alignment shifter and the normalization shifter), which will increase the delay instead of improving it. The dual-path organization overcomes this problem by taking advantage of the fact that the alignment shift and the normalization shift are mutually exclusive operations. Therefore, each path utilizes only one of the above shifters: the *Far* datapath uses only the alignment shifter, while the *Close* datapath only requires the normalization shifter.

After the alignment has completed, the results in either path enter a 3:2 CSA. and finally they are normalized. The HAs in each path are necessary for correct rounding [12]. The vectors in both datapaths are represented by two's complement.

When the MAF unit performs a double-precision instruction only one of the paths is activated. When however the design executes two single-instructions in parallel there can be cases, depending on the exponent differences, where both paths need to be active at the same time.

## C. Third Stage: Combined Addition with Rounding

Combining a floating-point multiplication or addition with rounding has been preciously introduced in [12]. The idea of this method lies on the following fact: performing addition and rounding in parallel eliminates the possible extra addition caused by rounding and thus, it reduces the latency. This is accomplished by exchanging the order of executing rounding and normalization (perform normalization before rounding) and combine the addition with rounding. The problem that arises however, when executing a MAF instruction, is that the rounding position is known only after the normalization has finished. For this reason, we perform the normalization before the addition in the previous stage. The third stage computes the vectors *sum* and *sum+1* and depending on the result of rounding, the correct result is forwarded to the output.

## IV. Synthesis Results

The presented MAF unit has been designed and implemented in VHDL. It is pipelined in order to achieve a latency of 3 cycles and has a throughput of one result per cycle. The design was targeted to the TSMC 0.13 1P8M standard cell process. It was synthesized for a varying clock cycle, from 10ns to 2ns. The VLSI implementation achieved a clock period of 3.43ns (291 MHz) and a final area (routed) of 286766.316$um^2$ while consuming 35.22 mW.

Two instances of the MAF unit have also been integrated in the LE1 Multi-Cluster VLIW processor [13]. The LE1 is a configurable, extensible, multi-cluster VLIW architected in a way to allow for the customization of the architecture (Register files, size, number, SIMD capability) and microarchitecture (custom ISA support, streaming memories etc). The LE1 has been customized to include a floating-point core and associated CPU state (FP register file, flags), primarily targeted to MAF-intensive applications. The LE1 implementation results demonstrate a maximum operating frequency of 132MHz (not speed-optimized), 1203 cell rows, $96 \times 1024 \times 32$RAMs (IRAM, STRMEM) and a total area of 19705429.027 $um^2$. Figures 5 and 6 depict the resulting VLSI cells.

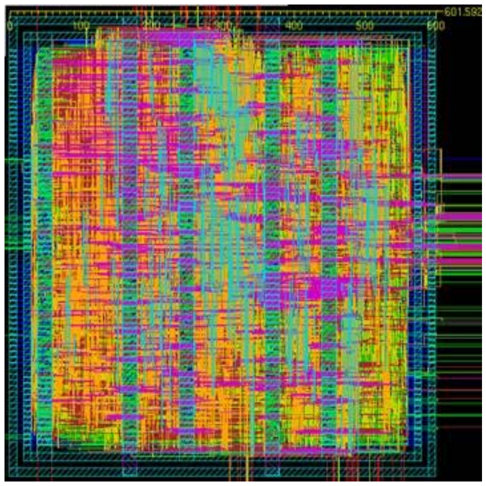

Fig. 5. The VLSI implementation cell of the dual-mode MAF unit

## V. Conclusions

This paper presented a dual-mode floating-point MAF unit, that performs either one double-precision or two single-precision floating-point instructions in parallel. The design demonstrates a reduced latency in both cases of executing either a MAF instruction or a single floating-point addition. This is achieved by incorporating well-known techniques such as a dual-path scheme and the combination of the final addition with rounding. Several adjustments have been introduced compared to traditional single or double-precision MAF units. The design has been validated and implemented with TSMC 0.13.

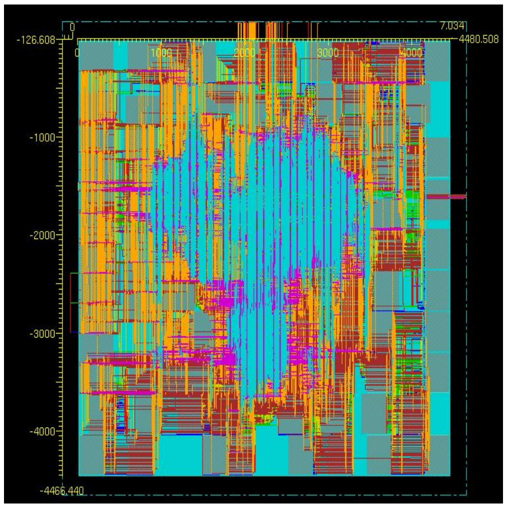

Fig. 6. The VLSI implementation cell of the LE1 VLIW processor using two dual-mode MAF units

## References

[1] C. Hinds, "An Enhanced Floating Point Coprocessor for Embedded Signal Processing and Graphics Applications," IEEE Conference Record of the Thirty-Third Asilomar Conference on Signals, Systems, and Computers, pp. 147-151, 1999.
[2] Y. Voronenko and M. Puschel, "Automatic Generation of Implementations for DSP Transforms on Fused Multiply-Add Architectures," International Conference on Acoustics, Speech and Signal Processing, pp. V-101-4, 2004.
[3] E.N. Linzer, "Implementation of Efficient FFT Algorithms on Fused Multiply-Add Architectures," IEEE Transactions on Signal Processing, Vol. 41, No. 1, pp. 93-107, Jan, 1993.
[4] R. M. Jessani, M. Putrino, "Comparison of Single- and Dual-Pass Multiply-Add Fused Floating-Point Units," IEEE Transactions on Computers, vol. 47 no. 9, pp. 927-937, 1998.
[5] C. Chen, L-A. Chen, J-R. Chen, "Architectural Design of a Fast Floating-Point Multiplication-Add Fused Unit Using Signed-Digit Addition," Proceedings of the Euromicro Symposium on Digital Systems Design (DSD'01), pp. 346, 2001.
[6] H. He, Z. Li, "Multiply-add fused float point unit with on-fly denormalized number processing," 48th IEEE Midwest Symposium on Circuits and Systems, Vol. 2, pp. 1466 - 1468, August 2005.
[7] G. Li, Z. Li, "Design of A Fully Pipelined Single-Precision Multiply-Add-Fused Unit," 20th International Conference on VLSI Design (VLSID'07), pp.318-323, 2007.
[8] L. Huang, L. Shen, K. Dai, Z. Wang, "A New Architecture For Multiple-Precision Floating-Point Multiply-Add Fused Unit Design," Proceedings of the 18th IEEE Symposium on Computer Arithmetic, pp. 69-76, 2007.
[9] J. D. Bruguera, T. Lang, "Floating-Point Fused Multiply-Add with Reduced Latency," 2002 IEEE International Conference on Computer Design (ICCD'02), pp.145, 2002.
[10] H. Sun, M. Gao, "A Novel Architecture for Floating-Point Multiply-Add-Fused Operation," Proceedings of the 2003 4th Pacific Rim Conference on Multimedia, Vol. 3, pp. 1675-1679, Dec. 2003.
[11] S. Krithivasan, M. J. Schulte, "Multiplier Architecture for Media Processing," Proc. 37th Asilomar Conf. Signals, Systems, and Computers, pp.2193-2197, 2003.
[12] G. Even, P. M. Seidel, "A Comparison of Three Rounding Algorithms for IEEE Floating-Point Multiplication," IEEE Trans. Comput., Vol. 49, No. 7, pp.638-650, 2000.
[13] V.-A. Chouliaras, K Koutsomyti, T. Jacobs, S. Parr, D. Mulvaney and R. Thomson, "SystemC-defined SIMD instructions for high performance SoC architectures" Proceedings of the 13th IEEE International Conference on Electronics, Circuits and Systems, Nice, France, Dec 10 - 13, 2006.