Design of High Precision Radix-8 MAF Unit with Reduced Latency

Bayadir A. Issa
Iinformation Technology Department
Southren Technical University
Basrah Iraq
bayader.phd@gmail.com

Israa Sabri A. AL-Forati
Computer Engineering Department
University of Basrah
Basrah, Iraq
israa.subri.1@gmail.com

Mohammed A. Al-Ibadi
Computer Engineering Department
University of Basrah
Basrah, Iraq
m.a.al-ebadi@ieee.org

Hayder M. Amer
Computer Engineering Department
Southren Technical University
Basrah, Iraq
hayder.amer@stu.edu.iq

Abdulmuttalib Turky Rashid
Electrical Engineering Department
University of Basrah
Basrah, Iraq.
abdturky@gmail.com

Osama T. Rashid
Computer Engineering Department
Iraq University College
Basrah, Iraq
turkey782002@yahoo.com

***Abstract*— Signal processing and image processing applications require floating-point computation in digital circuit designs. The floating-point calculation is recommended in almost all digital design applications to improve the accuracy of results. Different mathematical operations use floating-point mathematical operations and can be used to perform calculations and different logical units. The proposed work sugestes a double-MAF unit that uses radix-8 for multiplication and uses a common addition block for addition and multiplication operations. Firstly, floating-point numbers are transformed into IEEE 754 format, next to both addition and multiplication calculations are achieved. Extracts the fractional part of a number and performs a calculation based on the exponential difference between the numbers. The proposed curriculum is designed with a parallel structure that first extracts the decimal and exponential values of the first unit and then performs multiplication and addition operations in the second block. At last, the output is made in the third block where normalization and zero detection are performed. The proposed approach is then compared to the baseline approach, which shows improvements in energy consumption and maximum common path delay. The results showed that the delay is reduced by about 57%.**

Keywords: Radix-8, VHDL, MAF unit, IEEE- 754, double-precision.

I. Introduction

The fused multiply-add unit (MAF) has become the IEEE-754-2008 standard operator. It is a combination of improved performance (one and two operations) and improved precision (one rounding). The latter allows for many algorithm improvements, such as efficient division and implementation of the square root. FMA can also be used as an adder or a multiplier. The latest instruction set (including IBM Power / PowerPC and Intel / HP IA64, but also the latest graphics processing unit) builds a floating-point unit (FPU) around FMA [1]. FMA combines multiplication and addition into one instruction as described by Eq.1[2]. Two operands can be added by making X=1 and two operands can be multiplied by making W=0. This flexibility means there is no need for an additional multiplier (that does only multiplication) or adder (that will perform addition only) as long as there is an FMA [3]. In general, the unit increases throughput due to fusing of the two operations and also increases accuracy since rounding is carried out once instead of twice.

$$M_i = ((X_i \Uparrow Y_i) + W_i) \quad (1)$$

The FMA can be broken down into six basic operations that work together to carry out the fused multiply-addition. The following are the FMA's basic operations;
1) Multiplication
2) Alignment
3) Addition
4) Leading Zero Anticipation (LZA)
5) Normalization, and
6) Rounding.

The fused multiplier- adder (FMA) advantages are:

1- Improve the application performance by performs the multiplication and subsequent addition recursively,
2- Perform floating-point multiplication or addition by put in a static constant in the data path. Though, compete with this floating-point multiplier or floating-point adder is not costly. This is because the additional hardware in the block enforces further latency on separate commands related to the unique unit. Furthermore, the bit width and interconnectivity of FMA internal blocks is typically more than twice the bit width of floating-point adders and floating-point multipliers, achieving design and routing process and timing goals. It gets complicated.
3- Finally, although the MAF unit is described by high consumption of power, the continual response for 3D graphs, software submissions, and different traditional handling algorithms results in infused multiply-add unit performance. The advantages outweigh its disadvantages [4, 5].

However, apart from the above advantages, combining two procedures in one instruction has some disadvantages. Using a constant to compute a single multiply and add operation in the MAF unit will result in longer latency than if it were performed in a single floating-point multiplier or adder, respectively. Also, the MAF unit adds hardware difficulty

978-1-7281-9352-6/20/$31.00 ©2020 IEEE

and avoids parallel floating-point addition and multiplication [6].

This paper describes a double precision MAF architecture that executes double-precision arithmetic in parallel. This design shows better performance by decreasing latency for MAF instructions and latency for floating-point addition. The idea is to use a radix-8 Booth encoder multiplier that speeds up partial products in half. Also, you can use Wallace Trees to reduce multiplier delays. Wallace tree multipliers consume less power than booths and arrays. You can combine the features of both multipliers to create faster, lower power multipliers. When floating point multiplication is performed. Also, normalization is accomplished previously than rounding to combine rounding with the final addition to further reduce latency. The presented architecture makes use of previously introduced techniques, but this is the first time these techniques have been combined into a double-precision MAF unit. Therefore, the main contribution of this paper is to present an efficient double-precision MAF unit, evaluate its performance, and identify the hardware cost. The rest of this paper is organized as follows:

Section 2 analyses related work and describe the basic MAF unit structure. Section 3 details the general architecture of the proposed double-mode MAF unit and the basic MAF module. Section 4 presents the implementation results, and finally, Section 5 contained the conclusion.

II. Related Works

The fusion MAF unit was first proposed in 1990 [2]. Since then, there has been some work on improving the FMA architecture shown in Fig. 1, which is often targeted to Independent Application Integrated Circuits (ASICs) or ISU target pipelines in general. In [7], the author introduces the FMA structure that was developed in 2007. In which the fusion operators have also been applied to other computation such as the dot product and FFT. Recommended for use with FPGA with nonstandard application [8] to improve numerical accuracy. Whence to use a nonstandard performance improvement model described in [9] for the use of multiply-add units (MAF). Partial Carry Save adders are used in this design to reach a low latency in an added stage, but with some knowledge of the application field of input and output values. Also, the author provides only the result of the implementation of the adder. The application of radix- 4 and 16 showed that multiplication is slow [10]. Contrary to popular perception, high-value representations are said to be beneficial for FPGA applications requiring IEEE 754 compliance because they can provide excellent digital performance while using low FPGA resources. The work in [4] illustrates the basics of FPGA floating-point computation. The author assigned a 9-stage addition pipeline and multiplied 14 stages to Xilinx Virtex 4 FPGA (-11 speed grade). Dual-precision floating-point performance compatible with IEEE 270 MHz with hardware pipeline.. Their area requirements are about 500 slices for adders and less than 750 slices for multipliers. While The [11] explores current solutions and uses two new fusible overlap structures in floating-point using heterogeneous input formats and considers the impact of various organizational features of modern FPGA structures. The units are evaluated at the application level as follows: Modify your standing advanced synthesis system to repeatedly include different units for calculation in the critical path of three different convex solvents. This component recovers performance by up to 2.5x compared to the closest competitor of the latest model and improves digital accuracy. However, this result appears to come at the cost of a significant rise in the required resources (4x to 5x the Xilinx score).

III. The Suggested Multiply-add Design

The goal of the proposed MAF unit is to reduce the latency of the FP addition with respect to FP MAF and also the FP multiplication operations. Recently, the FP adder unit has been implemented using the pipeline which designed in three phases. The first stage modifies the input parameter bits to produce the appropriate binary, exponent, and signal, depending on the mode of

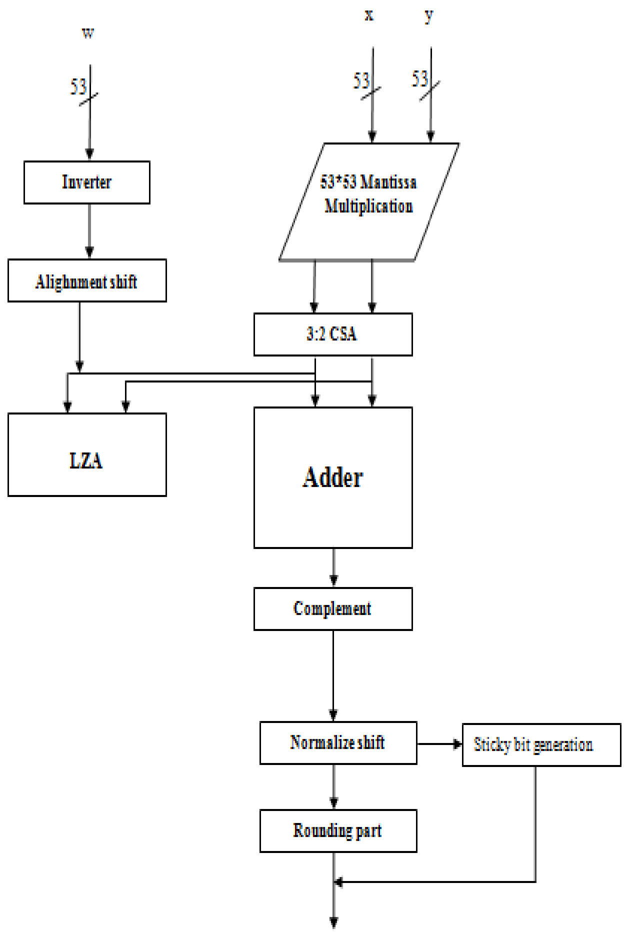


Fig. 1. The block diagram of standered design of MAF unit.

operation. Then do the exponential and binary multiplication. In the second stage, the architectural data path is divided into two tracks: Far path and close Far. Closed paths handle instructions that cause a large amount of cancellation. This is an effective proposition with bases that differ from the max. 1. The far path is active in the remaining cases where the indicators are far away. The decision to choose the path is based on the input index and is made primarily. closed paths perform alignment and normalization. The dual path of the proposed solution reduces latency and benefits from the fact that the calculations are mutually exclusive, which improves the efficiency of the devices. Finally, the third stage performs the final addition and rounding in parallel by performing the normalization before addition. The proposed design relies on devices for double precision MAF unit but has been expanded to implement two instructions with one precision in parallel. To this end, several components have been combined and redesigned to introduce additional devices. However, these additional hardware requirements provide better results for the region rather than repeating the components of each precision mode. Many multiple transmitters throughout the design manage the flow of different precision vectors. The control signal is multiplied by the multiplexer with other components of the design. For double-precision, this control signal is set. The remainder of this section provides details on the basic units of the MAF unit.

A. *First Stage: Exponent Handling and Multiplication*

1. *Exponent Handling:* Exponentiation determines the transitions of alignment and normalization. Once double-precision is accomplished, the exponent variance equals to

$$df = \mathrm{EW} - (\mathrm{EX} + \mathrm{EY} - 1023) \tag{2}$$

and the shift amount is

$$sh = 56 - d = \mathrm{EX} + \mathrm{EY} - \mathrm{EW} - 967 \tag{3}$$

2. *Multiplication:* The MAF unit uses radix8 multiplier which can perform double-precision multiplication. In order to recode the radix-8 multiplier, quad-bit is applied as an alternative of triple. Every quadrant is noted as a signed number. The radix-8 algorithm decreases the number of partial products to n / 3, as shown in Fig. 2. where n is the number of doubled bits [14]. The partial product generator is designed to multiply the multiplier A by 0, 1, -1, 2, -2, -3, -4, 3, 4 to produce the product. For product constructor, multiplication by zero multiplies the multiplication by "0". Multiplying by "1" incomes that the product will remain the same as the value of the multiplicand. Multiplying by "-1" means the product is the two's complement to the number. Multiplication by "-2" displaces the complement of the remaining multiple in one portion. Multiplying by "2" moves the number in one place to the left. Multiplication by "-4" changes the complement of 2 of the multiple that you left the titan. Multiplying by "2" changes the number that leaves two places. Here we have a single multiple of the 3Y multiplier, and it is not immediately available. The previous addition must be made to generate it. 2Y + Y = 3Y Add the same number to the left with one number. Therefore, you can increase the time by summing the partial products. One-digit product of the multiplicand and the multiplier, if the multiplier has more than one digit. Partial products are used as an intermediate step to compute larger products. Uses the (4: 2) CSA tree method to generate a partial product of two rows that can be added at the final stage. It also reduces the number of critical paths and adders compared to traditional parallel adders. The (4: 2) CSA tree here helps to increase the speed of partial products addition. The benefits are even more pronounced when using 32-bit or 64-bit multipliers. Multiplier speed, area, and power consumption are directly proportional to multiplier efficiency. Compressor. The block diagram in Fig. 2 and Table I show the radix8 booth encoder.

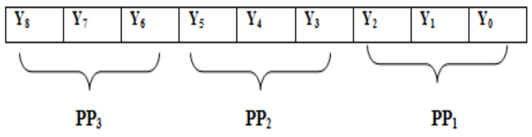


Fig. 2. Grouping of bits in Radix-8 method

TABLE I. RADIX-8 BOOTH RECODING

Multiplier Bits $Y_{i+2} Y_{i+1} Y_1 Y_{i-1}$	*Recoded Operation on multiplicand, X*
0000	*0X*
0001	*+X*
0010	*+X*
0011	*+2X*
0100	*+2X*
0101	*+3X*
0110	*+3X*
0111	*+4X*
1000	*-4X*
1001	*-3X*
1010	*-3X*
1011	*-2X*
1100	*-2X*
1101	*-X*
1110	*-X*
1111	*0X*

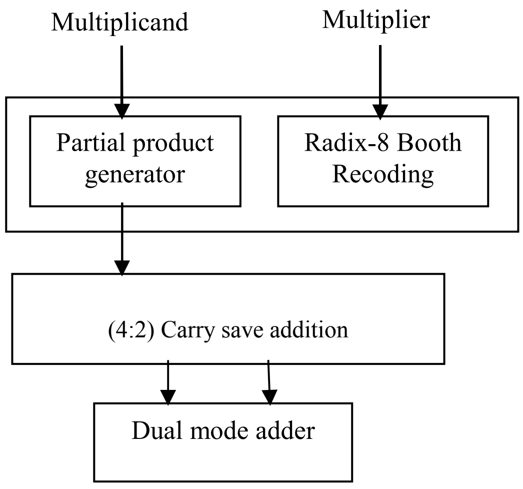


Fig. 3. Standard chart of Radix-8 Booth Encoder multiplier

B. *The Second Stage: Double-Path Association*

In the second stage of the floating point multiplication addition architecture, the double-path configuration has been presented as an effective design method for floating-point adders as follow:

```
- IF df > 1 then
    $P = (x*y) + w$    (4)
  Elseif $(-1 \leqslant df \leqslant +1)$ then
    P = (x*y)-w    (5)
  End IF
```

The advantage of splitting the data path in two is that Bypassing the first stage of the design when performing floating-point addition reduces the total latency of the design. To achieve this, two adjustments need to be made.

1- First, duplicate the exponentiation logic. Next stage. The hardware overhead essential for this step is least. This is because the additional exponentiation part takes up less space.

2- The second adjustment is that the placement shifters should be placed after the multipliers rather than in parallel. This is in the standard MAF construction. However, altering the alignment shifter location affects the MAF critical path. This is since the critical path contains double shifters (an alignment shifter and a normalization shifter) instead of one. Dual-pass tissues overcome this problem by taking improvement of the information that alignment shifts and normalization shifts are interrelated, where each pass gets only one of the shifters above: Far data The path uses the alignment shifter only, whereas the Close data path only needs the normalization shifter. After the alignment is complete, either pass will result in 4:2 CSA. And finally, it is normalized. HA for each pass is required for precise rounding. The vectors in both data paths are signified in two's complement. Once the MAF unit executes a double-precision instruction, only one of the paths is active in the design. Depending on the different exponents, it may be necessary to have both paths active at the same time.

C. *Third Stage: Collective Addition with Rounding*

Floating-point multiplication or addition and rounding combinations are presented in [12]. The indication of this scheme is based on the resulting facts. Execution addition and rounding in parallel reduces latency by eliminating the additional addition that rounding can cause.

TABLE II. SUMMARY OF THE LATENCIES FOR THE IMPLEMENTATIONS OF MAF UNITS

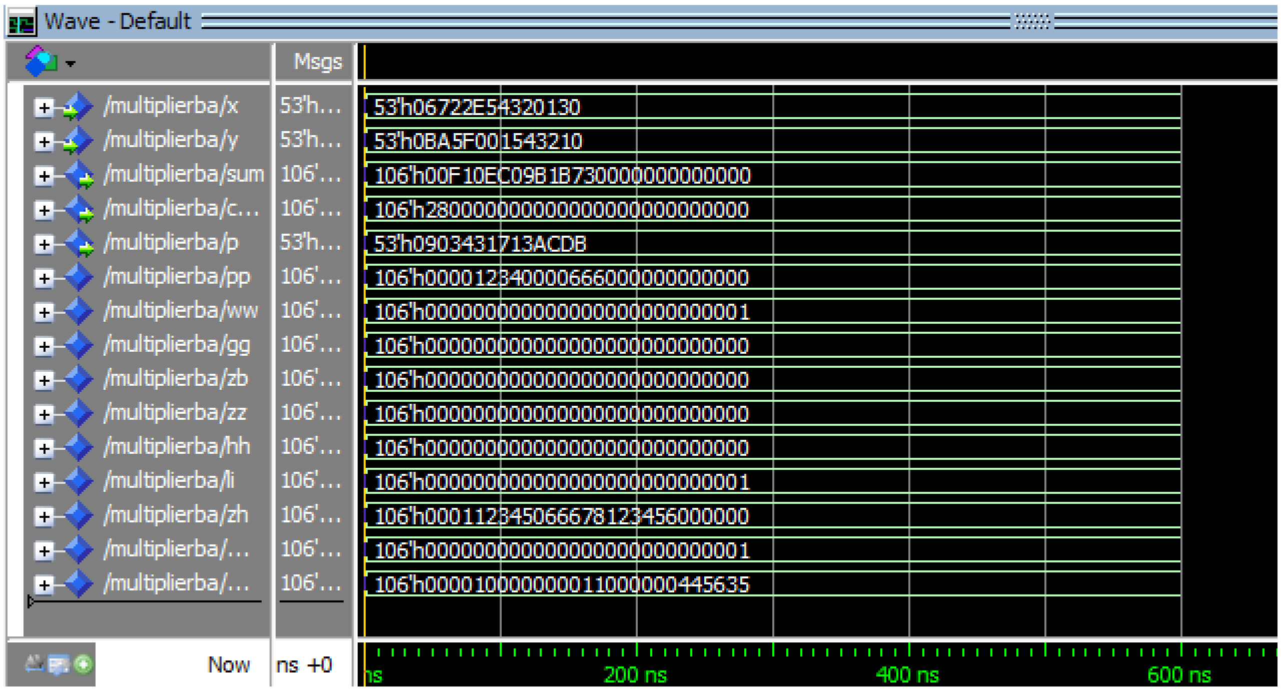
Latency / in MAF scheme	FP MAF operation		FP addition operation	
	t_{nd2}	ns	t_{nd2}	ns
FP MAF in [6]	107	10.77	107	10.77
FP MAF in [6]	91	9.1	91	9.1
Proposed MAF	74	7.4	74	7.4

IV. Estimation and Comparison of the MAF Units

Estimated critical path latency for the proposed MAF unit are accomplished at a specific level because a further correct estimation depending on the Floating Point Unit (FPU) requirements and technical factors. This is the latency of the two input NAND gates (which is equivalent to two inverters with four fans, or FO4). Then compare it to the previous single Datapath architecture proposed in [6] and [7]. The units were tested with valid randomly generated data that follow a standard allocation. Legal numbers means applying some limitations to the check numbers to better understand the exactness results. The situation of overflow/underflow that raises an exception is avoided. Also, except when the exponent of the multiplication result is greater than the exponent of the other operand, we have finished adding zeros to the other operand, so that there is no significant difference between the two exponents and the multiplication result and limits the exponent of the second add operand. Sticky bits can contribute to the decision to round the result. During the simulation, 100 samples were generated for each of the three operands contained in the operation of multiply-add supplied to the hardware unit. A high-level implementation was formed to calculate the outcome of each multiplication-addition task that emulates unfused single and double precision units. The results of the double-precision unit are the basis for comparing our unit with the non-fusion type single-precision unit. However, before comparing, the double-precision result is rounded to the nearest even rule according to IEEE rounding. The process of calculating 100 samples was repeated 20 times to ensure consistent results. The results for the FMA unit were compared to the [6,7] MAF unit. We have found that the hardware always achieves the highest possible accuracy, but this is not always the case. In Table II, Fig. 4, and Fig. 5, the results of three different versions of the FMA unit and a non-fusion version of the Xilinx base. The low latency version uses only LUTs to implement the floating-point adder. This reduces routing delay overhead and can lead to higher frequencies. Comparing the three low latency units shows that the version uses fewer resources than the equivalent unit at [6,7]. However, this benefit is composed by the detail that the maximum frequency of the unit is low. Even if the frequency means to consume more resources, it was improved by making more efforts to parallelize the hardware at some stages to achieve better frequency with the same latency. Resource usage of the non-fusion unit, excluding the number of registers used.

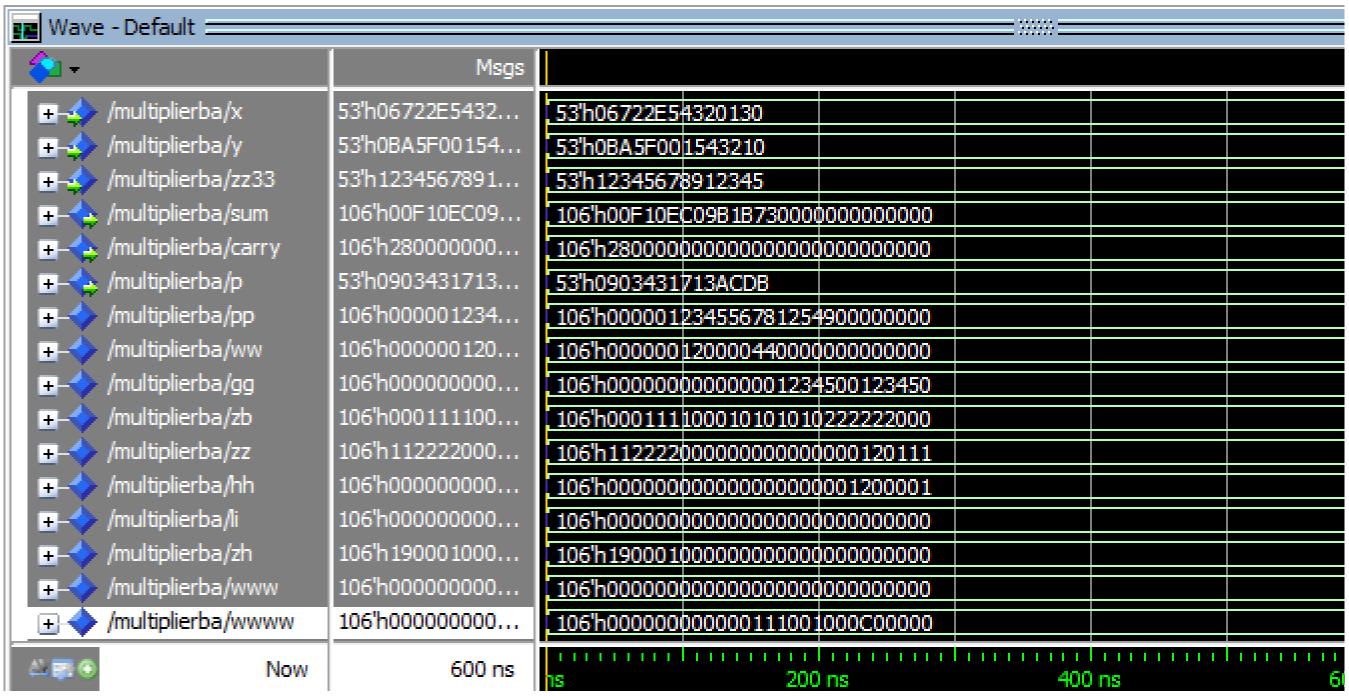
V. Conclusions

Fast and efficient floating-point (***FP***) computational units have been proposed in this paper. The proposed units provide the performance needed to meet the needs of today's computationally complex applications. Two computational units were configured in this paper to compute the merged function ***X+Y.W*** with dual-mode two 32bits or one 64bits double data path multiply-added fused (***MAF***) units. The proposed architecture for the floating-point multiply-add (MAF) unit waits for floating-point multiplication using a radix-8 Booth encoder that cuts the latency of partial products by half compared to previous MAF implementations. It is advisable to save time. This architecture is based on the earlier MAF architecture, which uses dual adders to combine final addition and rounding to enable pre-addition normalization. The proposed MAF architecture incorporates the previously used dual datapath configuration. Floating-point adders and MAF units show that full-length alignment shifts and full-length normalized shifts are mutually exclusive, and only one such shift appears in the proposed tissue critical path. I will. I. Take advantage of the facts. We estimated the delay of the proposed architecture, compared it to the single datapath MAF architecture and performed the addend alignment after the multiplication, rather than the parallel processing done in the previous architecture.



```
Wave - Default
                       Msgs
/multiplierba/x        53'h...   53'h06722E54320130
/multiplierba/y        53'h...   53'h0BA5F001543210
/multiplierba/sum      106'...   106'h00F10EC09B1B730000000000000
/multiplierba/c...     106'...   106'h28000000000000000000000000
/multiplierba/p        53'h...   53'h0903431713ACDB
/multiplierba/pp       106'...   106'h000012340000666000000000000
/multiplierba/ww       106'...   106'h000000000000000000000000001
/multiplierba/gg       106'...   106'h000000000000000000000000000
/multiplierba/zb       106'...   106'h000000000000000000000000000
/multiplierba/zz       106'...   106'h000000000000000000000000000
/multiplierba/hh       106'...   106'h000000000000000000000000000
/multiplierba/li       106'...   106'h000000000000000000000000001
/multiplierba/zh       106'...   106'h000112345066678123456000000
/multiplierba/...      106'...   106'h000000000000000000000000001
/multiplierba/...      106'...   106'h000010000000011000000445635
Now                    ns +0     ns   200 ns   400 ns   600 ns
```

Fig. 4: Simulation Waveform of the Proposed MAF unit with multiplication and addition operation



```
Wave - Default
                            Msgs
/multiplierba/x        53'h06722E5432...   53'h06722E54320130
/multiplierba/y        53'h0BA5F00154...   53'h0BA5F001543210
/multiplierba/zz33     53'h1234567891...   53'h12345678912345
/multiplierba/sum      106'h00F10EC09...   106'h00F10EC09B1B730000000000000
/multiplierba/carry    106'h280000000...   106'h28000000000000000000000000
/multiplierba/p        53'h0903431713...   53'h0903431713ACDB
/multiplierba/pp       106'h000001234...   106'h000001234556781254900000000
/multiplierba/ww       106'h000000120...   106'h000000120000440000000000000
/multiplierba/gg       106'h000000000...   106'h000000000000000123450012345 0
/multiplierba/zb       106'h000111100...   106'h000111100010101010222222000
/multiplierba/zz       106'h112222000...   106'h112222000000000000000120111
/multiplierba/hh       106'h000000000...   106'h000000000000000000001200001
/multiplierba/li       106'h000000000...   106'h000000000000000000000000000
/multiplierba/zh       106'h190001000...   106'h190001000000000000000000000
/multiplierba/www      106'h000000000...   106'h000000000000000000000000000
/multiplierba/wwww     106'h000000000...   106'h000000000000011100100 0C00000
Now                    600 ns              ns   200 ns   400 ns   6
```

Fig. 5: Simulation Waveform of the Proposed MAF unit with addition operation

REFERENCES

[1] N. Brunie, F. de Dinechin, B. de Dinechin" Mixed-precision Fused Multiply and Add" 45th A silomar Conference on Signals, SystemsComputers, Nov 2011, United States. pp.165-169.

[2] [2] L. Huang, L. Shen, K. Dai, Z. Wang " A New Architecture For Multiple-Precision Floating-Point Multiply-Add Fused Unit Design" National University of Defense Technology Changsha, 410073, P.R.China.

[3] F. Aliyu " Design and Analysis of a Floating Point Fused Multiply Add Unit using VHDL" International Journal of Engineering Trends and Technology (IJETT) Vol.24, No.4, PP.1-9, 29 June 2015.

[4] W. Jos´e, A. Silva, H. Neto, M´ario, et al." Floating-Point Single-Precision Fused Multiplier-adder Unit on FPGA" Atas X Conference on Reconfigurable Systems, Vilamoura – Algarve, PP.15-21, April 13, 2014

[5] I. Ishteyaq, K. Guarav, H. Gupta "A Low Power Design Of Floating Point Multiply Add Unit" IJEDR1703037 International Journal of Engineering Development and Research 243, Vol. 5, No.3, PP.243-247,2017.

[6] K. Manolopoulos, D. Reisis V.A. Chouliaras "An Efficient Dual-Mode Floating-Point Multiply-Add Fused Unit" IEEE, PP.1-8, 2010.

[7] J. Bruguera, T. Lang "Floating—Point Fused Multiply—Add: Reduced Latency for Floating-Point Addition" IEEE, PP.1-10,2005.

[8] E. Quinnell, E.Swartzlander, C. Lemonds "Floating-Point Fused multiply-add Architectures" IEEE PP.331-337,2007.

[9] A. Azad1, A. Mollahoseini2 " Design and Implementation of a New Multi-Functional Fused Dot Product in FPGA" International Journal of Computer Science and Mobile Computing, Vol. 3, No. 2, PP.917 – 923, February 2014.

[10] Ms. Aruna Jyothi C B" Design & Analysis of High-Performance Floating-point Fused multiply-add with Reduced Latency" Journal of Advanced Research and Science,Vol.1,PP.1-7,2015.

[11] M. Thomas "Design and Simulation of Radix-8 Booth Encoder Multiplier for Signed and Unsigned Numbers International Journal for Innovative Research in Science & Technology,Vol. 1, No. 1, PP.1-10 , June 2014.

[12] B. Kumar Mohanty1 · A. Choubey "Efficient Design for Radix-8 Booth Multiplier and Its Application in Lifting 2-D DWT" Springer Science, Business Media New York,PP.1-21, 2016.

[13] P. Patil, L.Kumre "High Speed-Low Power Radix-8 Booth Decoded Multiplier" International Journal of Computer Applications, Vol. 73, No.14, PP.0975 – 8887 July 2013.

[14] R. Muralidharan, "Radix-4 and Radix-8 Booth Encoded

Multi-Modulus Multipliers" IEEE TRANSACTIONS ON CIRCUITS AND SYSTEMS.PP.1-13,2013.

[15] H. Jiang "Approximate Radix-8 Booth Multipliers for Low-Power and High-Performance Operation" IEEE Transactions on Computers, PP.1-8, 2015.