Communications on Applied Electronics (CAE) – ISSN : 2394-4714
Foundation of Computer Science FCS, New York, USA
Volume 7– No. 33, February 2020 – www.caeaccess.org

Design of a Novel Fused Add-Sub Module for IEEE 754-2008 Floating Point Unit in High Speed Applications

Abhyarthana Bisoyi
College of Engineering & Technology
Bhubaneswar, India

Aruna Tripathy, PhD
College of Engineering & Technology
Bhubaneswar, India

ABSTRACT

A multiplier block can be implemented either by shift add technique, Booth algorithm or Vedic algorithm, in DSP applications. However, these techniques do not work for a certain class of numbers known as exceptions. They are +infinity, -infinity and Not a Number (NaN). The solution to address these exceptions is a "Fused add-subtract" module. The addition and subtraction modules are fused together to give two outputs giving both addition and subtraction results. The time delay and the number of Look-Up Tables (LUTs) of the existing Fused add-subtract unit have been found to be quite high to meet the present-day requirements of speed. Therefore, a novel algorithm for fused add-subtract has been proposed in this paper. In the floating-point unit (FPU), building blocks of the addition and subtraction are fused together, resulting in reduction of the number of computations as well as the area usage. The existing fused add-sub module is compared with the proposed module in terms of delay and the number of LUTs. The new algorithm is observed to reduce the time delay and area by 12.5% and 18.878% respectively as compared to the conventional one.

Keywords

Floating Point Unit, fused add-sub, Multipliers, Time Delay, Functional Area, Digital Circuits, Digital Signal Processing, FPGA

1. INTRODUCTION

Multipliers are the basic blocks of various applications. For instance, Digital Signal Processing (DSP) applications revolve around techniques that require an efficient multiplier, so that the time delay can be minimized. Not only DSP but also areas of circuit design, communications, embedded system design, etc. need an optimized multiplier based on reduced power and area usage. The various multipliers have evolved from Booth multipliers to fused add multipliers and so on. This paper is about the fused add-sub module which is obtained from the fusion of addition and subtraction unit. The conventional one has some cons, which is overcome by the proposed fused add-sub unit. This module is observed to have optimized area, less time delay and power too.

The literature survey of the IEEE 754-2008 floating point unit (FPU), multiplier block associated with it and fused add-sub multiplier has been discussed in section 2. This section also includes the problem statement formulation after realizing the conventional add-sub multiplier in Vivado 2015.4. The detailed operations of the existing and the proposed multiplier are described in section 3. The simulations of the multipliers and a comparison between them based on certain parameters are discussed in section 4. The paper is concluded in section 5. This section also focusses on the future aspect of the proposed multiplier that is the aim of our upcoming work. At last, section 6 includes the references.

2. PROBLEM STATEMENT

Different multipliers are detailed in this section. The basic IEEE 754-2008 FPU is discussed first. Then the multiplier block followed by the fused add-sub unit is discussed.

2.1 IEEE 754-2008 floating point unit

The basic idea about an IEEE 754-2008 FPU is obtained from [1]. It discusses the bitwise representation of single and double precision format of the aforementioned standard. [2] includes the matrix multiplication using FPGA, giving a brief idea about the implementation of the process on boards. The various arithmetic operations carried out in an FPU are discussed in [3]. Among the various multipliers, the Vedic multiplier is discussed in [4], showing the advantage of using a Vedic multiplier over a conventional one. A better multiplier is proposed and analyzed in [5] that are basically a shift-add multiplier technique, showing some more advantages than the Vedic multiplier.

2.2 Fused Add-Subtract Module of IEEE 754-2008 Single Precision

There exists another module that is a fused add-sub module, as discussed in [6, 7] for the FPU. This technique is used in the formulation of another technique called the fused multiply-add unit [8, 9] for DSP applications. Further, the technique used to reduce the latency in the FPU, is discussed in [10-12]. The Standard gives a platform for two radix- 2 and 10, which can further be extended for rounded mixed-radix operations [13]. The correct average of floating-point numbers (FPU) was proposed for radix 2 and 10, this is useful for implementation of IEEE 754-2008 [14].

The fused Add-Subtract algorithm has addition and subtraction modules fused together to give two outputs, both addition and subtraction results. However, this algorithm also has high values of time delay and a significant number of Look-Up Tables. Therefore, a novel algorithm of fused add-subtract has been proposed in our work. Here, there will be a single output as its result depending upon the type of operation it is supposed to perform.

3. PROPOSED ALGORITHM

In this section, the detailed architecture along with the flow diagrams and algorithms of the previous discussions (as mentioned in section II) are presented. The initial portion in this section includes the disadvantage of IEEE 754-2008 FPU Single Precision of multiplier that is the "exceptions". The next portions include the arithmetic operations done in the above-mentioned standard. The conventional and the proposed fused add-sub unit in an FPU multiplier are discussed next.

Communications on Applied Electronics (CAE) – ISSN : 2394-4714
Foundation of Computer Science FCS, New York, USA
Volume 7– No. 33, February 2020 – www.caeaccess.org

3.1 IEEE 754-2008 FPU Single Precision Multiplication

Single-precision format is a format that takes 32 bits in a computer memory. It represents the values using a floating point.

The number is represented by the following: [1]

((-1)^S) x (1.M) x (2^ (E-127)) (1)

Here S stands for Sign bit, M stands for 23-bit mantissa (significand); E stands for 8 bits exponent.

Table 1. Bitwise Representation of Single Precision IEEE 754 Standard

Precision	Sign	Exponent	Mantissa
Single	[31]	[30:23] = 8 bits	[22:0] = 23 bits

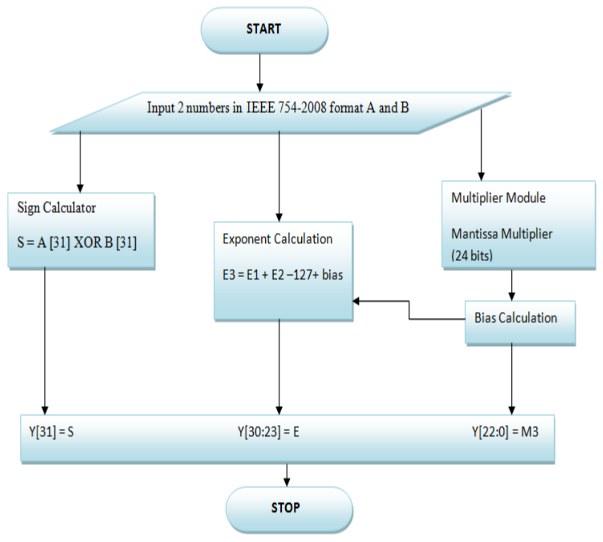


Fig. 1: Block schematic of Multiplication scheme in IEEE 754-2008 single precision FPU.

From the flow chart shown in figure 1,

Result of Y=A*B

= (-1) s1 (M1 x 2 E1) * (-1) s2 (M2 x 2 E2) (2)

S1, S2 are the: Sign bits (32nd bits of number A & B).

E1, E2 are the exponent bits of number A & B.

M1, M2 are the: Mantissa bits of Numbers A& B. [1]

Step 1: S3 = S1 XOR S2

Step 2: M3 = M1 * M2 (Multiplier Block)

Step 3: E3 = E1 + E2 – 127 + bias (Exponent Block)

Step 4: The bias value is calculated from the number of shifts done by the multiplier

S3 occupies the 31st bit of the result. M3 occupies [22:0] bits of the result. E3 occupies [30:23] bits of the result.

3.2 Exceptions

The multiplier block shown in figure 1, is used in various multipliers. But multiplier schemes such as Booth, Vedic, shift-add [1] do not work for a certain class of numbers. These numbers are exceptions. They are +infinity, -infinity and Not a Number (NaN).

3.3 IEEE 754-2008 Single Precision Addition Module

The basic condition for the module as illustrated in figure 2 to perform, is that ‘a’ and ‘b’ can only be added if the exponents are the same that is e1=e2 [6, 13].

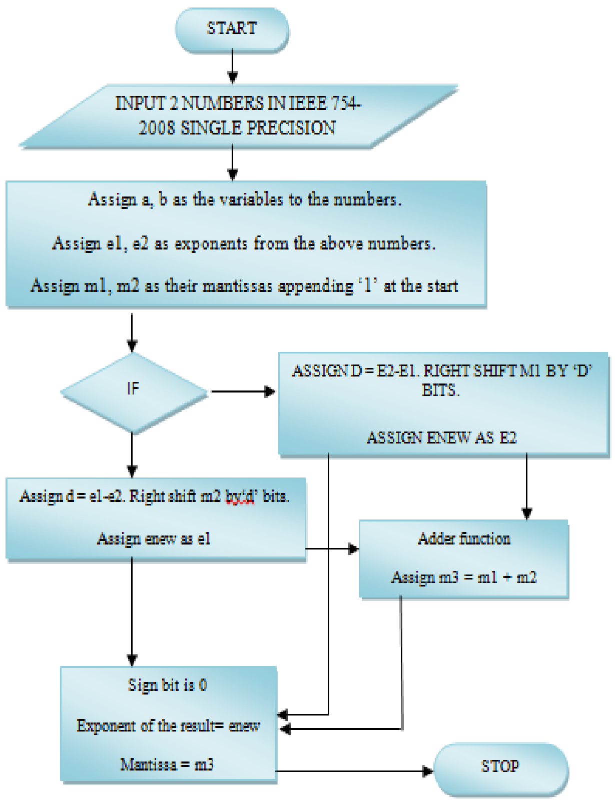

START
INPUT 2 NUMBERS IN IEEE 754-2008 SINGLE PRECISION
Assign a, b as the variables to the numbers.
Assign e1, e2 as exponents from the above numbers.
Assign m1, m2 as their mantissas appending ‘1’ at the start
IF
ASSIGN D = E2-E1. RIGHT SHIFT M1 BY ‘D’ BITS.
ASSIGN ENEW AS E2
Assign d = e1-e2. Right shift m2 by’d’ bits.
Assign enew as e1
Adder function
Assign m3 = m1 + m2
Sign bit is 0
Exponent of the result= enew
Mantissa = m3
STOP

Fig. 2: Flowchart of IEEE 754-2008 Single Precision Addition Module.

If e1>=e2, calculate the difference of the exponent. Shift E2 by d number of bits. Assign enew as E1; else calculate the difference of exponent. Shift E1 by d number of bits. Assign enew as E2. Compute the sum of the mantissa m1 and m2 and store it in m3. And sign bit being zero as it is the addition the value is stored as sign bit is zero, the exponent is assigned enew and mantissa is assigned m3.

3.4 IEEE 754-2008 Single Precision Subtraction Module

The basic condition for the module as illustrated in figure 3, to perform that is ‘a’ and ‘b’ can only be added if the exponents are the same that is e1=e2. [6, 13].

If e1>e2, calculate difference of exponent (d). Assign enew as e1. Assign greater value ‘s’ as m1 and smaller value ‘t’ as m2. Else, if e1<e2, calculate difference of exponent (d). Assign enew as e2. Assign greater value ‘s’ as m2 and smaller value ‘t’ as m1. Else If e1 = e2, then find out the greater of the M1 and M2. The smaller mantissa gets shifted to right by d

number of bits. Compute the difference of the mantissa. Assign it to ‘r’. In other words, r = s – t. Then iteration is performed to find out number of bits (i) after which bit ‘1’ is found. The normalized exponent becomes enew-i.

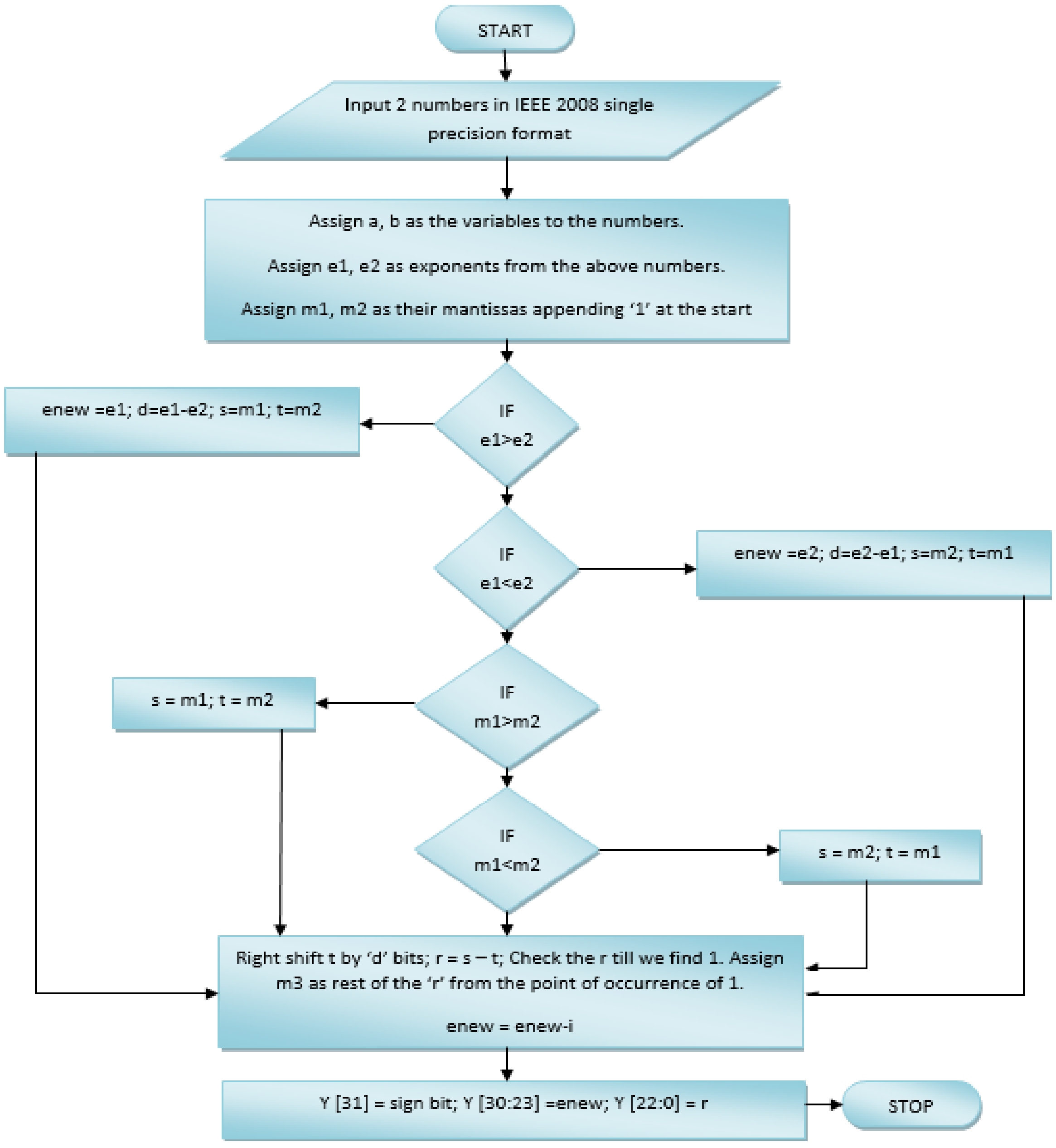

START
Input 2 numbers in IEEE 2008 single precision format
Assign a, b as the variables to the numbers.
Assign e1, e2 as exponents from the above numbers.
Assign m1, m2 as their mantissas appending ‘1’ at the start
IF e1>e2
enew =e1; d=e1-e2; s=m1; t=m2
IF e1<e2
enew =e2; d=e2-e1; s=m2; t=m1
IF m1>m2
s = m1; t = m2
IF m1<m2
s = m2; t = m1
Right shift t by ‘d’ bits; r = s – t; Check the r till we find 1. Assign m3 as rest of the ‘r’ from the point of occurrence of 1.
enew = enew-i
Y [31] = sign bit; Y [30:23] =enew; Y [22:0] = r
STOP

Fig. 3: Flowchart of IEEE 754-2008 Single Precision Subtraction Module.

3.5 Fused Add-Subtract Module of IEEE 754-2008 Single Precision

The method as shown in figure 4 was proposed in [6]. The first two steps are the same as that of addition or subtraction. Assigning of the values to the variables also follows the previous methods of addition and subtraction as outlined in figure 2 and 3. The difference lies in the fact that both addition and subtraction are carried out simultaneously and both the result of addition and subtraction is presented. In some DSP applications like the FFT, there is usage of addition and subtraction in the same equation. So here the advantage is there is no need to call the addition and subtraction separately for the parameters to perform the arithmetic operations over and over again.

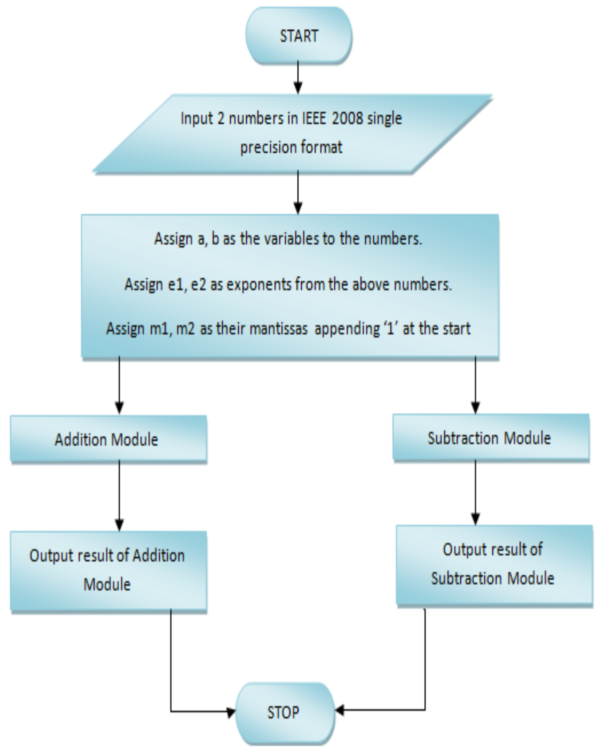

START
Input 2 numbers in IEEE 2008 single precision format
Assign a, b as the variables to the numbers.
Assign e1, e2 as exponents from the above numbers.
Assign m1, m2 as their mantissas appending '1' at the start
Addition Module
Subtraction Module
Output result of Addition Module
Output result of Subtraction Module
STOP

Fig. 4: **Flow chart of Fused Add-Subtract Module of IEEE 754-2008 Single Precision.**

3.6 Proposed IEEE 754 2008 Single Precision Fused Add-Subtract

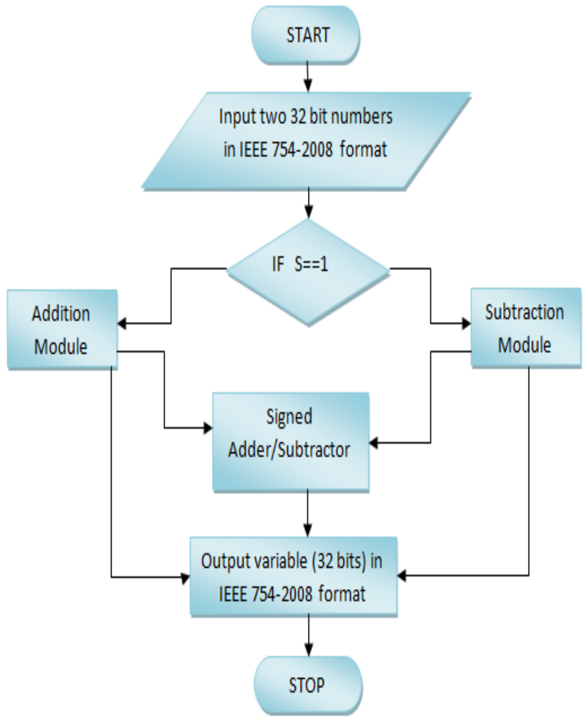

START
Input two 32 bit numbers in IEEE 754-2008 format
IF S==1
Addition Module
Subtraction Module
Signed Adder/Subtractor
Output variable (32 bits) in IEEE 754-2008 format
STOP

Fig. 5: **Flowchart of Proposed IEEE 754 2008 Single Precision Fused Add-Subtract.**

The proposed algorithm here has both addition and subtraction into a single unit. The motivation behind this is the same process followed in adding or subtracting two binary numbers. The process can be seen in figure 5. First, two inputs are given in the IEEE 754-2008 single precision format and then a conditional checker identifies which operation to follow depending on the sign bits of the given inputs. After deciding which module, it will execute it goes towards that particular module. The addition block is very direct as is explained in section 3.5. The subtraction block however follows a different approach. Here the mantissa part is converted into 2's complement notation and then the signed adder-subtractor is called. As the subtraction of two numbers is same as the addition of their 2's complement form. Hence both the modules are called with the same signed adder/subtractor. Thereby it reduces the time delay as well as its area utilization. This leads to less area utilized as well as reduction in time delay.

4. RESULT ANALYSIS

In this section, the simulation results of the techniques mentioned in section 3 are discussed. Simulations are carried out in Vivado 2015.4 in Verilog HDL. The initial parts of this section consist of the simulation results and discussion of an IEEE 754-2008 single precision FPU with three exception cases. The exceptions indicate the limitation of the multipliers. The cases are +infinity, -infinity and Not-a-number (NaN). These results are followed by the addition unit, subtraction unit, and existing fused add-sub unit and the proposed algorithm.

4.1 Exceptions in IEEE 754-2008 FPU

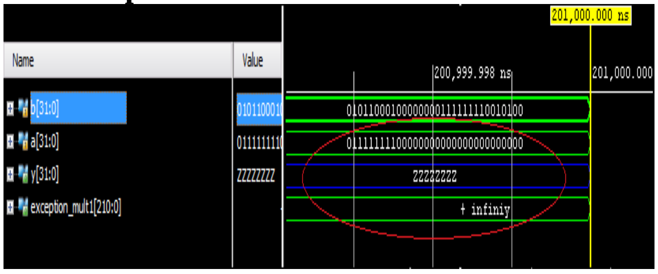

```
Name                     Value
b[31:0]                  0101100010   01011000100000000111111110010100
a[31:0]                  0111111110   01111111100000000000000000000000
y[31:0]                  ZZZZZZZZ     ZZZZZZZZ
exception_mult1[210:0]                + infiniy
```

Fig. 6: **Simulation of +Infinity in IEEE 754-2008 single precision FPU.**

In figure 6, the exception flag can be seen as +infinity if any of the input has an all '1' exponent and mantissa as zero. The sign bit is zero.

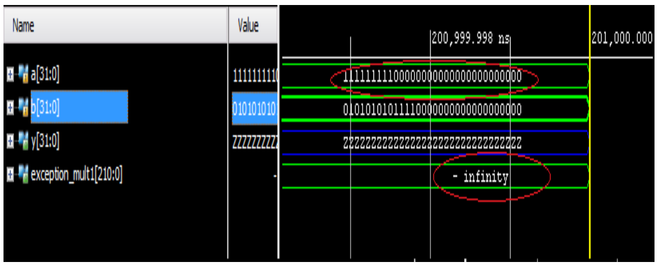
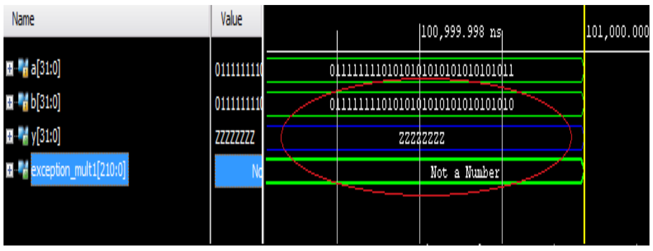

```
Name                     Value
a[31:0]                  1111111110   11111111100000000000000000000000
b[31:0]                  0101010101   01010101011111000000000000000000
y[31:0]                  ZZZZZZZZZZ   ZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZ
exception_mult1[210:0]   -            - infinity
```

Fig. 7: **Simulation of - Infinity in IEEE 754-2008 single precision FPU.**

In figure 7, the exception flag can be seen as +infinity if any of the input has an all '1' exponent and mantissa as zero. The sign bit is 1.

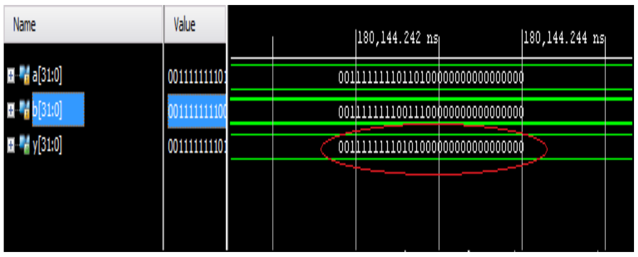


```
Name                      Value        |100,999.998 ns          101,000.000
a[31:0]                   0111111110   01111111101010101010101010101011
b[31:0]                   0111111110   01111111101010101010101010101010
y[31:0]                   ZZZZZZZZ     ZZZZZZZZ
exception_mult1[210:0]    No           Not a Number
```

Fig. 8: Simulation of Not a Number in IEEE 754-2008 single precision FPU.

In figure 8, the exception flag can be seen as Not a Number if any of the input has an all '1' exponent and mantissa as non-zero. The sign bit is may be '1' or '0'.

4.2 IEEE 754-2008 Single Precision Addition Module



```
Name       Value          |180,144.242 ns         |180,144.244 ns
a[31:0]    0011111111101  00111111110110100000000000000000
b[31:0]    0011111111100  00111111110011100000000000000000
y[31:0]    0011111111101  00111111110101000000000000000000
```

Fig. 9: Simulation of IEEE 754 single Precision of Addition.

The values of A and B are as follows,

A = 17.86 = 00111111110110100000000000000000

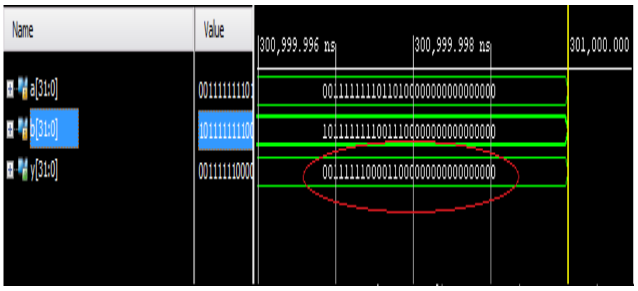
B = 12.111=00111111110011100000000000000000

Upon Adding A and B,

Result is 29.971=00111111110101000000000000000000

The above result is obtained from the flow chart given in figure 2. The results were verified from the simulation result shown in figure 9.

4.3 IEEE 754-2008 Single Precision Subtraction Module



```
Name       Value          |300,999.996 ns     |300,999.998 ns      301,000.000
a[31:0]    0011111111101  00111111110110100000000000000000
b[31:0]    1011111111100  10111111110011100000000000000000
y[31:0]    0011111110000  00111111100001100000000000000000
```

Fig. 10: Simulation of IEEE 754 2008 Single Precision Subtraction Module.

Here the values that are simulated are:

A = 17.86 = 00111111110110100000000000000000

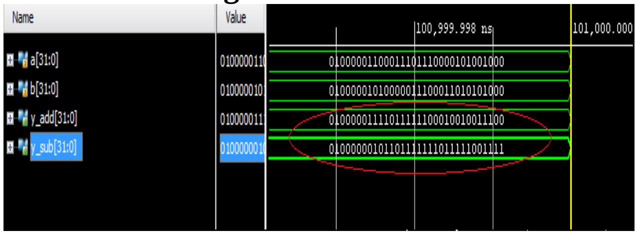
B = 12.111=00111111110011100000000000000000

Upon Subtracting A and B,

Result is 5.749 =00111111100001100000000000000000

The above result is obtained from the flow chart given in figure 3. The results were verified from the simulation result shown in figure 10.

4.4 Fused Add-Subtract Module of IEEE 754-2008 Single Precision



```
Name          Value        |100,999.998 ns          101,000.000
a[31:0]       0100000110   01000001100011101110000101001000
b[31:0]       0100000101   01000001010000011100011010101000
y_add[31:0]   0100000111   01000001111011111100010010011100
y_sub[31:0]   0100000010   01000000101101111111011111001111
```

Fig. 11: Simulation of Fused Add Subtract.

Here the values that are simulated are

A = 17.86 = 00111111110110100000000000000000

B = 12.111 = 00111111110011100000000000000000

Upon adding A and B,

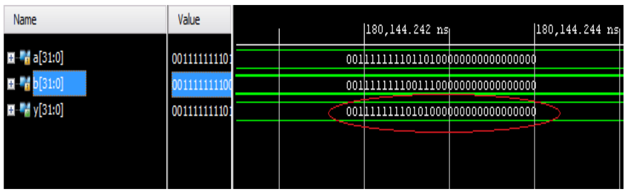
Result is 29.971 = 00111111110101000000000000000000

Upon Subtracting A and B,

Result is 5.749 =00111111100001100000000000000000

The above result is obtained from the flow chart given in figure 4. The results were verified from the simulation result shown in figure 11. Here both addition and subtraction values are obtained in a single function call.

4.5 Proposed IEEE 754 2008 Single Precision Fused Add-Subtract



```
Name       Value          |180,144.242 ns         |180,144.244 ns
a[31:0]    0011111111101  00111111110110100000000000000000
b[31:0]    0011111111100  00111111110011100000000000000000
y[31:0]    0011111111101  00111111110101000000000000000000
```

Fig. 12: Simulation of Fused Add-Subtract with addition inputs.

Here the values that are simulated are

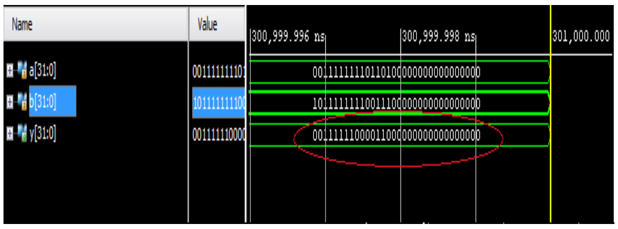
A = 17.86=00111111110110100000000000000000

B = 12.111=00111111110011100000000000000000

Upon adding A and B,

Result is 29.971 = 00111111110101000000000000000000

The value is verified as shown in the simulation done using Vivado 2015.4 in Verilog HDL in figure 12.



```
Name       Value          |300,999.996 ns     |300,999.998 ns      301,000.000
a[31:0]    0011111111101  00111111110110100000000000000000
b[31:0]    1011111111100  10111111110011100000000000000000
y[31:0]    0011111110000  00111111100001100000000000000000
```

Fig. 13: Simulation of Fused Add-Subtract with subtraction inputs.

Upon subtracting same numbers, the result obtained is,

5.749 =00111111000011000000000000000000

The value is verified as shown in the simulation obtained in figure 13.

Figure 14 represents the RTL schematic of the proposed Fused Add-Subtract module. The small boxes are called as Look-Up Tables (LUTs). The number of LUTs used in this technique is 434 out of a total of 53200. Hence the usage percentage of LUTs in this technique is 0.0081 %.

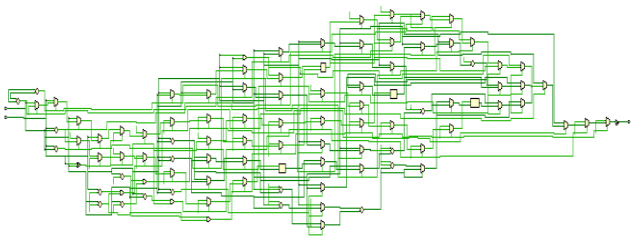
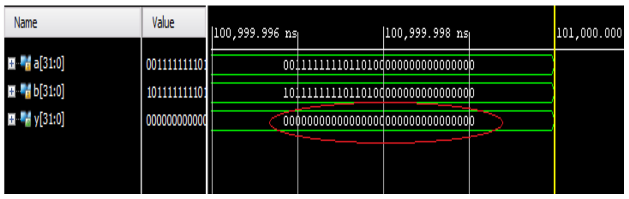


Fig. 14: RTL Schematic of proposed Fused Add-Subtract.

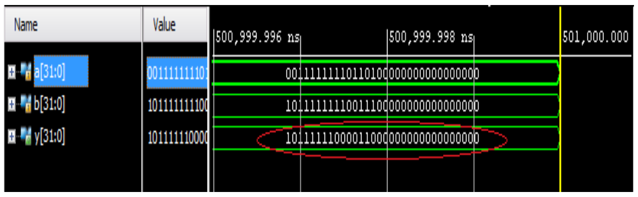


```
Name      Value         |100,999.996 ns      |100,999.998 ns      |101,000.000
a[31:0]   00111111110        00111111110110100000000000000000
b[31:0]   10111111110        10111111110110100000000000000000
y[31:0]   00000000000        00000000000000000000000000000000
```

Fig. 15: Simulation of Fused Add/Subtract when A = B and Sign bit is different

The mantissa and exponents of both A and B are equal but the sign bits are unequal.

The output is equal to 0 as shown in figure 15.



```
Name      Value         |500,999.996 ns      |500,999.998 ns      |501,000.000
a[31:0]   00111111110        00111111110110100000000000000000
b[31:0]   10111111110        10111111110011100000000000000000
y[31:0]   10111111000        10111111000011000000000000000000
```

Fig. 16: Simulation of Fused Add-Subtract when A < B and Subtraction is carried out.

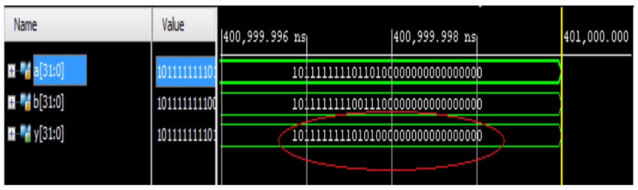
Let us take two values A and B.

A = 1.609375 = 00111111110110100000000000000000

B = -1.703125 = 10111111110011100000000000000000

Output(Y)=-0.09375=
10111111000011000000000000000000

The output is verified as shown in figure 16.



```
Name      Value         |400,999.996 ns      |400,999.998 ns      |401,000.000
a[31:0]   10111111110        10111111110110100000000000000000
b[31:0]   10111111110        10111111110011100000000000000000
y[31:0]   10111111110        10111111110101000000000000000000
```

Fig. 17: Simulation of Fused Add-Subtract When both A and B are negative.

Let us take two values A and B.

A = -1.609375 = 10111111110110100000000000000000

B = -1.703125 = 10111111110011100000000000000000

Output(Y)=-3.3125

=10111111110101000000000000000000

The output is verified as in figure 17.

Table 2. Comparison Between the Proposed and the Existing Fused Add-Subtract in Terms Of Time Delay

Algorithm	**Fused Add-Subtract**	**Proposed Fused Add-Subtract**
Max. Combinational Path Delay (in ns)	21.36	18.69

It is observed from Table II that the proposed Fused Add-Subtract is has less time delay as compared to the existing Fused Add-Subtract.

Table 3. Comparison Between Proposed and Existing Fused Add/Subtract in Terms Of Area Usage

Algorithm	**Fused Add-Subtract**	**Proposed Fused Add-Subtract**
Number of LUTs	535	434
% Area Usage (out of 53200 LUTs)	0.0100	0.0081

It is observed from Table III that the proposed Fused Add-Subtract is has less area usage as compared to the existing one.

5. CONCLUSION

The conventional fused Add-Subtract are useful for limited operations such as FFT where in one equation, the addition and subtraction of the same operands was obtained. The proposed Fused Add-Subtract is found to have a very less time delay 18.69 nanoseconds as compared to the 21.36 nanoseconds of the existing Fused Add-Subtract resulting in a reduction of time delay by 12.50 %. In addition to a reduced time delay, it is also found to have an area usage of 434 LUTs as compared to 535 of the existing Fused Add-Subtract, thereby reducing the area usage by 18.878 %. Three different cases of Fused Add-Subtract have been discussed and simulated. The new unit can be implemented in DSP applications using system generator, TMS320C6748 and TMS320C6713 DSP kit. It can also be implemented through code composer studio toolbox.

6. REFERENCES

[1] Anjanasasidharan and P. Nagarajan, "VHDL implementation of IEEE 754 floating point unit", In *Proceedings of the IEEE Conference on Information Communication and Embedded Systems (ICICES '14), pp.* 1-5, 2014.

[2] F. Bensaali, A. Amira, and R. Sotudeh, "Floating-Point Matrix Product on FPGA", In *Proceedings of*

theIEEE/ACS International Conference on Computer Systems and Applications (AICCSA'07), pp. 466-473, 2007.

[3] Muller, J.M., Brisbare, N., de Dinechin, F., et al. 2009. Handbook of floating point arithmetic. (Birkhäuser, 2009)

[4] A. Bisoyi, M. Baral, and M. K. Senapati, "Comparison of a 32-bit Vedic multiplier with a conventional binary multiplier", In *Proceedings of the IEEE International Conference on Advanced Communications, Control and Computing Technologies* (ICACCCT'14), pp. 1757-1760, 2014.

[5] T. B. Juang, and Y. R. Lee, "Seamlessly Pipelined Shift-and-Add Circuits Based on Precise Delay Analysis and Its Applications", In *Proceedings of theIEEE Computer Society Annual Symposium on VLSI* (ISVLSI'16), pp. 625-630, 2016.

[6] H. Saleh, and E. E. Swartzlander, "A floating-point fused add-subtract unit", In *Proceedings of the 51st Midwest Symposium on Circuits and Systems* (MWSCAS'08), pp. 519-522, 2008.

[7] A. Sharma, S. Singh and A. Sharma, "Implementation of single precision conventional and fused floating point add-sub unit using Verilog", In *Proceedings of theIEEE International Conference on Wireless Communications, Signal Processing and Networking* (WiSPNET'17), pp. 169-171, 2017.

DOI=10.1109/WiSPNET.2017.8299741.

[8] A. Amaricai, O. Boncalo, and C. E. Gavriliu, "Low-precision DSP-based floating-point multiply-add fused for field programmable gate arrays", *IET Computers & Digital Techniques*, vol. 8 issue 4, pp. 187-197, 2014.

DOI=10.1049/iet-cdt.2013.0128.

[9] P. K. Meher, "Seamless pipelining of DSP circuits", *Journal on Circuits, Systems, and Signal Processing*, vol. 35 issue 4, pp.1147-1162, 2016.

DOI= 10.1007/s00034-015-0089-2.

[10] B. Xue, P. Chatterjee, and S. K. Shukla, "Simplification of C-RTL equivalent checking for fused multiply add unit using intermediate models", In *Proceedings of the18th Asia and South Pacific Design Automation Conference* (ASP-DAC'13), pp. 723-728, 2013.

[11] J. D. Bruguera, and T. Lang, "Floating-point fused multiply-add: reduced latency for floating-point addition", In *Proceedings of the 17th IEEE Symposium on Computer Arithmetic* (ARITH'05), pp. 42-51, 2005.

[12] E. Quinnell, E. E. Swartzlander, and C. Lemonds, "Bridge Floating-Point Fused Multiply-Add Design", *IEEE Trans. on Very Large-Scale Integration (VLSI) Systems*. Vol. 16 issue 12, pp. 1727-1731, 2008.

[13] C. Jeangoudoux and C. Lauter, "A Correctly Rounded Mixed Radix Fused-Multiply-Add", *IEEE 25th Symposium on Computer Arithmetic (ARITH)*, pp. 21-28, 2018.

[14] S. Boldo, F. Faissole and V. Tourneur, "A Formally-Proved Algorithm to Compute the Correct Average of Decimal Floating-Point Numbers", *IEEE 25th Symposium on Computer Arithmetic (ARITH)*, pp. 69-75, 2018.

[15] A. Mohapatra, A. Bisoyi and A. Tripathy, "Design of Novel Multipliers-Vedic and Shift-Add for IEEE 754-2008 Single Precision Floating-point Unit in High Speed Applications", *Proceedings of 5th IEEE International Symposium on Smart Electronic Systems (IEEE-iSES, formerly IEEE-iNIS)*, pp. 159-162, 2019.

[16] V. Leon, S. Xydis, D. Soudris and K. Pekmestzi, "Energy-efficient VLSI implementation of multipliers with double LSB operands", in *IET Circuits, Devices & Systems*, vol. 13, no. 6, pp. 816-821, 2019.

[17] K. Chen, Y. Hwang and Y. Liao, "VLSI Design of a High Throughput Hybrid Precoding Processor for Wireless MIMO Systems", in *IEEE Access*, vol. 7, pp. 85925-85936, 2019.

[18] H. Vu and K. Chen, "A Low-Power Broad-Bandwidth Noise Cancellation VLSI Circuit Design for In-Ear Headphones", in *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, vol. 24, no. 6, pp. 2013-2025, June 2016.

[19] H. Kultala *et al.*, "LordCore: Energy-Efficient OpenCL-Programmable Software-Defined Radio Coprocessor," in *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, vol. 27, no. 5, pp. 1029-1042, May 2019.

[20] S. Friedrichs, M. Függer and C. Lenzen, "Metastability-Containing Circuits," in *IEEE Transactions on Computers*, vol. 67, no. 8, pp. 1167-1183, 1 Aug. 2018.