2024 IEEE 17th International Symposium on Embedded Multicore/Many-core Systems-on-Chip (MCSoC) | 979-8-3315-3047-1/24/$31.00 ©2024 IEEE | DOI: 10.1109/MCSoC64144.2024.00050

Improved Decimal Rounding Module based on Compound Adder

Heba Hakim
Computer Engineering Department
Engineering College, University of Basrah
Basrah, Iraq
hiba.abdulzahrah@uobasrah.edu.iq

Hanadi A. Jaber
Computer Engineering Department
Engineering College, University of Basrah
Basrah, Iraq
hanadi.jaber@uobasrah.edu.iq

Zaineb M. Alhakeem
Department of Chemical and Petroleum Refining Engineering
Basrah University for Oil and Gas
Basrah, Iraq
zainebalhakeem@buog.edu.iq

***Abstract*— A new approach and architecture have been presented in this paper to efficiently merge the decimal rounding stage according to the IEEE 754-2008 standard based on the compound adder. This integration serves as a crucial key to improve the computation performance of both decimal floating point multiplication and fused multiply add (FMA) operation. The decimal rounding control unit is based on the IEEE 754-2008 five rounding modes. The decimal combined add/round module has been coded in the VHDL and verified using the Xilinx ISE 13.2. The overall critical path delay is compared with another design based on the BCD adder with the decimal rounding. The results have shown at least 3.4 % improvement in terms of delay reduction.**

Keywords—DFP, BFP, FMA, Rounding module, VHDL

I. INTRODUCTION

The accurate representation of decimal numbers with limited digits is a perpetual challenge in computer systems. The rounding problem happens when only a portion of the digits in the calculation result can be preserved. Rounding a number means estimate or approximate it, on the other hand, rounding might reduce the digits in a number without effect the expected value. So, the result will be less accurate but easier to use. The five rounding modes defined by the IEEE 754-2008 must provide exactly rounded results (Markstein, 2008).Rounding operation is used to approximate a number to a specific value, selecting one of the two nearest possible decimal floating point (DFP) numbers based on to the specified rounding direction (rounding mode). Floating-point number (FP) is a fractional number that result from the division of two integers. A computer can process and recognize real number in a form of complex coded [1, 2]. The design of Decimal floating point has garnered considerable interest due to the development of real-time applications in recent years. This leads to a significant demand for high-performance of adder and multiplier units [3].

The most important issue in the Binary floating point (BFP) numbers even in the DFP is that, the fractional numbers representation. Sometimes, the fractional numbers (i.e. 0.1, 0.2 or 0.3) cannot be represented accurately using the BFP. This issue is considered as a main challenge in the most of engineering, financial and commercial application. Especially, when the errors have been occurred due to the error propagation into the rounding module after execution of an arithmetic operation (i.e. multiplication, addition, subtraction and division) with the involvement of fractional numbers [4, 5]. For example, if two fractional numbers are added (i.e. 0.6 + 0.3) then the result should be produced 0.9, but actually it is 0.8999999999999991. The DFP format can be used to fix the accuracy representation problem and it should be enhanced using accurate round module to decrease the error effect [6].

This issue is considered as one of the most important issues in the design of the arithmetic processor. In commercial and financial applications, calculations adhere to the human rules and standards of decimal arithmetic, which can differ from the traditional arithmetic used in scientific calculations. So, the decimal numbers that utilized in financial applications are typically represented as the integer coefficients scaled by a power of 10. For example, the value 834.50 is denoted as an integer coefficient 83450 with an exponent -2, which is expressed as 83450 × 10^(-2). Since more than one coefficient can denote the same value, this integer scaled encoding is redundant. Both coefficients 050 (with an exponent 1) and 005 (with an exponent 2) denote the value 500. Although it is possible to utilize a normalized fixed-point (non-redundant) coefficient as well, this is more suitable for scientific computations [7, 8].

The rounding error is considered as a common issue between the both DFP and BFP. As a consequence, this problematic issue cannot be avoided any more especially when a finite number of bits could be used to represent a fractional number. The rounding errors and its impact can be reduced when the precision digits in the DFP is increased.

Therefore, when required, implementations of decimal arithmetic should have the capability to retain the complete precision of the numbers, including the trailing fractional zeroes, in addition to calculating numerical values. To support the full accuracy and range that necessary for the financial computations, early computers utilized exact decimal arithmetic. For example, in order to perform precise decimal multiplication, the precision digits of the largest input value must be doubled. Thus, the series of multiplications would rapidly exceed any hardware precision capability. Therefore, rounding is required in commercial and financial applications in two various ways:

- rounding placed by precision: to ensure an exact approximation in numerous complex computations.
- rounding placed by legal requirements: in order to decrease the exact result to a lower precision required by the application.

This paper is organized as follow: section 2 mentions the overview of the related work on the rounding operation; section 3 illustrates in details the proposed decimal add/round module; section 4 describes the comparison result, and the conclusion is presented in section 5.

2771-3075/24/$31.00 ©2024 IEEE
DOI 10.1109/MCSoC64144.2024.00050

II. BACKGROUNDIN

According to the guidelines of IEEE 754 standard for binary floating-point arithmetic, FP64 denoted the format of data that utilized 8 bytes for both encoding and storage. FP64 storage space includes three components as demonstrated in Figure 1: (a) the sign (S) which is represented by most significant bit; (b) the exponent (E) which is expressed by the middle 11bits; and (c) the fraction (M) which is represented by the lowest 52bits. The normalized number of FP64 can be mathematically expressed by the following equation:

$$N = (-1)^S \times 2^{E-1023} \times (1 \times M) \quad (1)$$

The value of actual exponent represents the difference between E and exponential bias. To compare the sizes of exponent for 2 floating-point numbers, all values of exponent can be expressed by unsigned integers to make it easiest.

In engineering application, the operation $(a \times b + c)$ is often needed and executed in 2 steps involving 2 rounding operation.

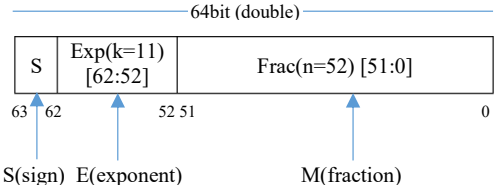

Fig. 1. FP64 storage format

The FMA operator is used in the execution of single instruction with operands of single/double precision floating point. Since there is a single rounding operation is executed on the merged full precision sum and product by employing the FMA operator. So, the latency is reduced and enhanced arithmetic precision for floating point [1,4,9].

To obtain exact final result during the DFP/BFP multiplication or FMA operation, the two vectors (sum and carry) which outputs from the partial reduction tree (i.e. 128-bit per each) should be added. The same concept in the addition/subtraction operation must be achieved to get the final result, too. Comprehensive works have been implemented in both decimal/binary multiplication and addition field. A. Vazquaz el. [10] accomplishes the decimal rounding operation after performed the BCD addition operation. The conventional BCD addition stage usually consists of three parts: pre-correction, compound addition and post-correction. The rounded model which is presented in [10] will be the reference to the proposed decimal rounding module in this work in term of delay comparison. When the multiple stages can be migrated in the same module then the computation performance will be enhanced. The proposed decimal add/round architecture presents the following advantage:

- For the trailing 9's detection, decimal rounding will be performed without additional carry propagation. The signals (C5 and C6) which are required for the rounding decision could be computed using the enhanced compound adder.
- The result of binary compound adder (sum and sum+1) can be corrected using a fast and simple decimal post-correction.
- Enables the rounding control unit to compute the round, guard and sticky digits concurrently with the operation of addition for two operands using the compound adder.
- Simplest and high-performance achievement.
- Two stages are implemented concurrently.

For instance, accurate decimal rounding is necessary to provide precisely rounded results for approximative computations

III. PROPOSED DECIMAL ADD/ROUND ARCHITECTURE

In this section, the proposed decimal add/round module will be presented in details. Figure 2 demonstrates the regular stages for the addition and rounding of 2 vectors (sum and carry) after the normalization staged is performed in the decimal multiplication or FMA operation. It consists mainly of a series of pre-correction (i.e. conventional (3:2) carry save adder (CSA)), a compound adder (i.e. 64-bit prefix adder), rounding control unit and finally, a series of post-correction circuits.

A. Pre-correction

The word length of the two operands is assumed (3p+1) digits, where p is equal to 16 digits. Before employing a prefix adder tree network-based rapid binary adder, the pre-correction of the (3p+1) digits of the two operands, sum (S) and carry (H), is performed. A conventional (3:2) CSA adds 0110_2 (+ 6) to the two operands (S and H) (i.e. S= $s_3\ s_2\ s_1\ s_0$ and H=$h_3\ h_2\ h_1\ h_0$) at the same time to obtain intermediate result (4-bit sum and 4-bit carry), this correction operation will be performed to all digits in parallel. The next step is that the intermediate result of adding two vectors (sum and carry) which are added together using a 64-bit prefix adder. Figure 2 (a) shows the block diagram of the pre-correction circuit, while Figure 3 illustrates the top level of the proposed decimal add/round architecture.

B. Compound Adder

A more efficient alternative of the low latency of the proposed add/round architecture based on the implementation of the compound adder which computes sum = (S + H) and sum + 1 = (S + H + 1), simultaneously. The benefit of this implementation is being able to incorporate a late complement or late increment into a 9's complement carry-propagate adder within a small constant time increment. A prefix adder tree implements the binary carry recurrence $C_{i+1} = g_i \vee (a_i \wedge c_i)$, where a_i and g_i are the carry alive functions and binary carry generate for bit number *i,* respectively. The decimal carries (c_1, c_2, c_5, and c_6) are the binary carries at different decimal positions. In this approach, the addition of prefix is performed in three stages:-

Stage 1) Pre-processing: This stage is implemented by a simple half adder. According to computation of prefix, generate (g_i), carry alive (a_i) and and propagate (p_i) signals are expressed in equations 2, 3, and 4 respectively.

$$g_i = sum_i \wedge carry_i \quad (2)$$

$$a_i = sum_i \vee carry_i \quad (3)$$

$$p_i = sum_i \oplus carry_i \quad (4)$$

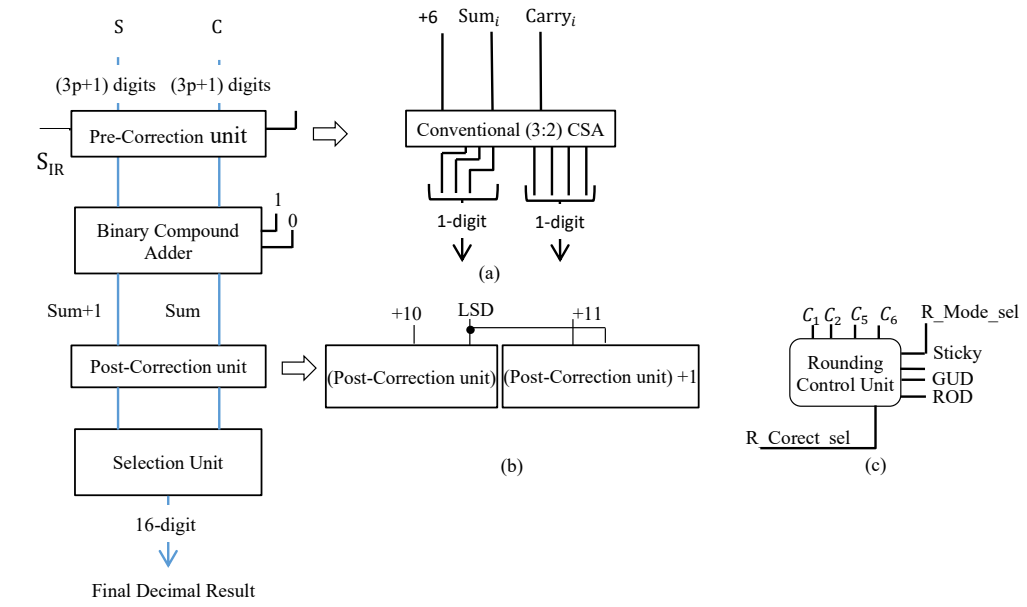

Fig. 2. Decimal Combined Add/Round Block Diagram

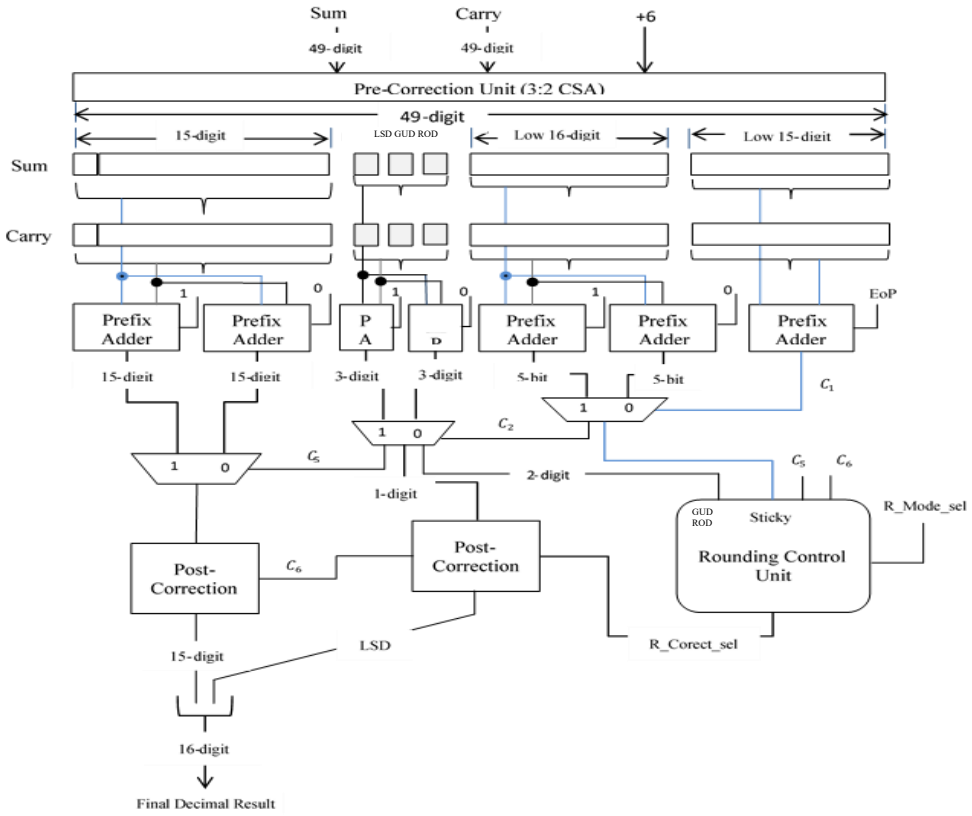

Fig. 3. Top level of decimal comined Add/Round module

Stage 2) Prefix Computation: The construction of prefix combinational is based on the concept of a collection carry propagate generate signals. It is defined by the equations 5 and 6 respectively.

$$G_{[i:m]} = \begin{Bmatrix} g_i & if\ i = m \\ P_{[i:j]} \vee G_{[i:j]} \wedge G_{[i-1:m]} & otherwise \end{Bmatrix} \quad (5)$$

$$P_{[i:m]} = \begin{Bmatrix} P_i & if\ i = m \\ P_{[i-1:m]} \vee P_{[i:j]} & otherwise \end{Bmatrix} \quad (6)$$

The representation of P and G can be simplified, an operator is called dot operator and it can be represented by '*' which could be introduced to create group of propagate and group of generate, they are expressed by the equation 7 as follow:

$$(G,P)_{[i:m]} = (G,P)_{[i-1:m]} * (G,P)_{[i:j]} \quad (7)$$

Stage 3) Post-processing: The information of sum and carry bits for each operand bit is determined in this stage. The equations (8) and (9) are expressed c_i and s_i respectively.

$$c_i = G_{[i:0]} \quad (8)$$

$$s_i = p_i \oplus c_{i-1} \quad (9)$$

In general, a prefix combinational network of m-bit inputs $b_{m-1}, b_{m-2}, \cdots, b_0$ uses the associative (arbitrary) operator ($\circ$) to produce the vector of the outputs described by:

$$z_i = x_i \circ x_{i-1} \circ x_{i-2} \circ \cdots \circ x_1 \circ x_0 \quad (10)$$

The carry computation can be defined as below:

$$z_i = \left(g_{(i,0)}, a_{(i,0)}\right), b_i = (g_i, a_i) \quad (11)$$

It has to be noticed that it can be implemented by a cell contains inputs of two pairs of bits (g_L, g_R) and (a_L, a_R) , where R and L represents (g_{out}, a_{out}), such that:

$$g_{out} = g_L \vee (a_L \wedge g_R) \quad (12)$$

$$a_{out} = (a_L \wedge a_R) \quad (13)$$

According to the above two equations, a variety of cells in multiple levels of the prefix adder network can be used to compute the carry bits in different positions. The point is that the carry bit named c_i engages to generate signal spanning the bit position (-1) to (i_{-1}). The relation between the generate signal (i_{-1},-1) and c_i defines as below:

$$c_i = g_{(i-1-1)}, where\ (g_{-1}a_{-1}) = (c_0, c_0) \quad (14)$$

The next step, an interconnection of these cells together is used to produce $g_{(i_{-1,-1})}$ for all i_i. After that, these carry bits are used to determine the final summation result, such that:

$$S_i = P_i \oplus C_i \quad (15)$$

C. Rounding Set-up Unit

The main target of the rounding set-up unit is to compute the guard (GUD), round digits (ROD), sticky bit (STK), and potential carry-in bit to p-digits (the most significant digits) concurrently with the execution of the addition operation.

D. General Algorithm

The rounding setup unit is put together as a conditional adder to ensure that at critical path only p-digit carry rippling delay. As shown in Figure 3, the (3p+1) digits width is divided into four groups (p-1), (p), (p-1) and 3-digit (i.e. LSD, GUD, and ROD), then all of these digits dispatched to the compound adders in parallel to avoid two successive addition steps. The least significant (p-1) digits are calculated once using 64-prefix adder network while the other groups are calculated twice to produce the sum and sum + 1. One of them assumes the carry-in bit to prefix adder is equal to zero (C_{in}= 0) while the other could be assumed that the carry-in bit is equal to one (i.e. C_{in}= 1).

The resulting carry-out (C_{in}) signal from the least significant (p-1) digits is using to select the appropriate carry-out signal to control the path of the other groups. The next p-digit groups are added to determine the sticky bit and carry-out (C_2) signal. The appropriate ROD, GUD and LSD should be selected using the carry-out (C_2) signal. Then, the three digits (i.e. LSD, GUD, ROD) of the two vectors are added using the compound adder. As a consequence, these three digits can be used to produce the carry-out (C_5) signal and intermediate vector of 3-digit. The latest two significant digits (i.e. GUD and ROD) of this vector should be dispatched to the rounding control unit. The remaining digit is considered as the LSD of the final result which should be corrected using the post-correction circuit. Finally, the most significant (p-1) digits should be input to the compound adder and the final (p-1) digit of the result could be selected using the carry (C_5) signal. To find the correct (p-1) digits and the LSD, a small circuitry of post-correction is utilized. However, the less significant digit could be contributed only to the sticky and their exact values are not required. Instead, the rounding control unit is utilized to determine their correct sticky bit. In the meantime, the carry-out (C_5 and C_6) two signals are fed into the rounding control unit.

As mentioned previously, the correct round and guard digits can be selected using the carry-out signals which are computed from multiple groups of least significant digits via the prefix adder network. This method is equivalent to determine the most significant (p) digits except the LSD. The EoP signal represents the carry-in. The intermediate result of the addition step should be included two vectors (sum and sum+1). Moreover, In the case of negative result, it should be complemented. At last, the rounding decision depends on the value of both guard and round digits which are computed in the rounding set-up module and the carry-out signals (C_5 and C_6) with signal (R_mode_sel) together could be used to select the final correct rounded result (R).

As a total, the final result (after complementation if need) could be selected according the value of the following signals: C_5,C_6, S_IR, R_mode_sel and R_corect_sel. The S_IR signal should be obtained previous to the stage of combined add/round. It can be produced with a simple comparator (the block of intermediate sign detection) that operates concurrently with the addition stage to indicate which the operand is greater one. Figure 4 shows the unrounded result can produce from the addition operation of the two vectors (sum and carry) and there are four different cases could be fed to the combined add/round module. The rounding position has two possibilities based on the value of the LSD and the temporary GUD which are produced from the rounding set-up unit.

As shown in Figure 4, the temporary round (ROD) and sticky digit (STK) are generated from the rounding set-up unit. This approach requires determining whether the value of GUD, ROD, and STK digits of the un-rounded result is zero or non-zero. Furthermore, these signals indicate whether the minimum exponent has been generated or the preferred exponent is determined.

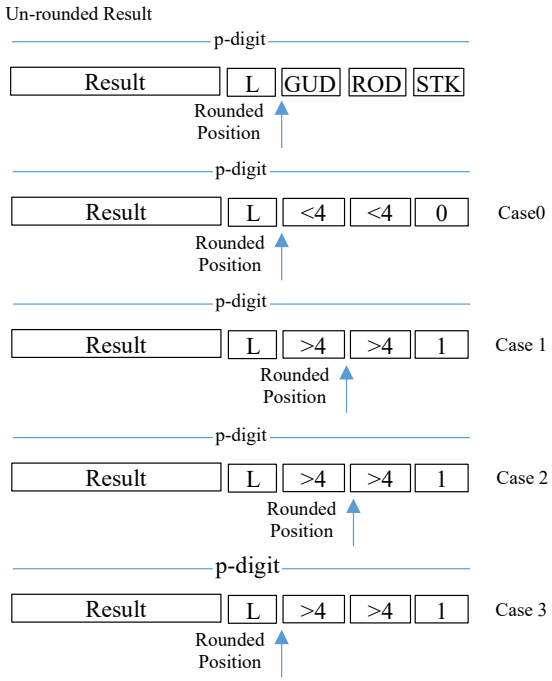

Fig. 4. FP64 storage format

E. General Algorithm

In parallel with binary prefix addition and post-correction, the decimal rounding control unit determines the selection signals and a decision of the rounding mode in cooperate with ROD and GUD. An important section of the rounding control unit is that the rounding decision. It is mainly depending on the value of round, guard, sticky digits and intermediate sign which are determined in the rounding set-up unit. In the meantime, the (p-1) digits of un-rounding result, LSD, and two signals (C_5 andC_6) could be used to select the appropriate rounding result. The correct most (p-1) significant digits of the un-rounded result can be predicted using the MSD of sum or sum+1 based on the value of the signal (C_5). It consists of two parts:

- Rounding Logic: A combinational logic implementation can be used directly to obtain the rounding condition of every decimal rounding mode. Furthermore, various conditions must be implemented on 4 probable rounding positions. Alternatively stated, the rounding operation should be implemented speculatively on sum or sum+1 which represent two possible rounding positions. The signals $C_1$and C_2) can be used to perform the rounding decision either at LSD as shown in Fig.4 case (1) or at guard digit as shown in Figure 4 case (2, 3), after that the carry is propagated via the rounding digit. Finally, the (2:1) multiplexer can be used to select the final summation result based on the rounding position.
- Rounding Conditions: The proposed decimal rounding architecture is supported the five modes of IEEE 754-2008 decimal rounding. Table I provides a summary of the conditions governing each mode of decimal rounding. The logical operator '∨' and '∧' denote logical OR and logical AND, respectively. Figure 4 demonstrates the meaning of symbols that are used in Table I.

TABLE I. Top Level of Decimal Combined Add/Round Module

Rounding Mode	Rounding Position	Action
Round Ties to Even	GUD	If (ROD>4 ∨ ROD=5 ∧ STK=1) Round to the nearest up Elseif (ROD=4 ∧ STK=0 ∧ GUD→odd) Round to the nearest up Elseif (ROD=4 ∧ STK=0 ∧ GUD→odd) truncate
	LSD	If (ROD>4 ∨ ROD=5 ∧ (ROD>0 ∨ STK=1)) Round to the nearest up Elseif (ROD=5 ∧ ROD=0 ∧ STK=0 ∧ LSD→odd) Round up Elseif (ROD=5 ∧ ROD=0 ∧ STK=0 ∧ LSD→odd)truncate
Round Toward Zero	GUD	Truncate
	LSD	Truncate
Round Ties To Away	GUD	If (GUD≥5) round up Else truncate
	LSD	If (LSD≥5) round up Else truncate
Round Toward Positive	GUD	If (S_{IR}=0 ∧ (ROD>0 ∨ STK=1)) round up Else truncate
	LSD	If (S_{IR}=0 ∧ (GUD>0 ROD>0 ∨ STK=1)) Round up Else truncate
Round Toward Negative	GUD	If (S_{IR}=1 ∧ (ROD>0 ∨ STK=1)) Round up Else truncate
	LSD	If (S_{IR}=1 ∧ (GUD>0 ROD>0 ∨ STK=1)) round up Else truncate

F. Post-correction and Final Selection Stage

The layout of the post-correction circuit has been shown in Figure 5 and the implementation of this circuit is very simple. The importance of this circuit, when some digits exceeded the permitted range of decimal numbers, then it should be used to correct them.

In case of the intermediate result is greater than 9 then the 1010 should be added digitally to correct the selected summation result in order to produce the post-corrected result. It is equivalent to subtract 0110 from each digit. The correct output of Sum or Sum + 1 is selected. The selected final results for all possible cases are shown in Table II.

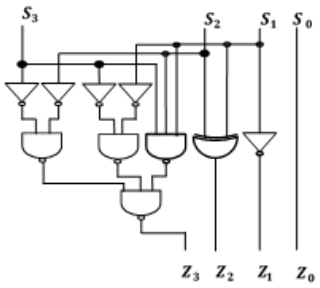

Fig. 5. Post-correction Circuit

TABLE II. Selected Final Result

Case #	cmp	C_1	C_2	C_5	Final Result
1	0	0	0	0	{sum,LSD}
2	0	1	0	0	{sum,LSD}
3	0	0	1	0	{sum,LSD+1}
4	0	1	1	0	{sum,LSD+1}
5	0	0	0	1	{sum+1,LSD}
6	0	1	0	1	{sum+1,LSD}
7	0	0	1	1	{sum+1,LSD+1}
8	0	1	1	1	{sum+1,LSD+1}
9	1	0	0	0	{$\overline{sum}$,LSD}
10	1	1	0	0	{$\overline{sum}$,LSD}
11	1	0	1	0	{$\overline{sum}$,LSD+1}
12	1	1	1	0	{$\overline{sum}$,LSD+1}
13	1	0	0	1	{$\overline{sum+1}$,LSD}
14	1	1	0	1	{$\overline{sum+1}$,LSD}
15	1	0	1	1	{$\overline{sum+1}$,LSD+1}
16	1	1	1	1	{$\overline{sum+1}$,LSD+1}

IV. Implementation Result

The Xilinx Vertex-5 FPGA family has been used to synthesis and verify the performance of the proposed decimal add/round method. The results of this architecture have been compared with the decimal rounding model presented in [10]. Table III provides the performance comparison result in term of # FO4 delay, a metric that is used to measure the taken time in logic gate to drive four times of its load capacitance.

TABLE III. Performance Comparison

Stage (#FO4)	Proposed Add/Round architecture	Decimal Rounding Model [10]
Pre-correction	6.5	6.5
Binary Adder	10.3	10.5
Rounding	2.4	2.5
Selection	6	6.6
Total (#FO4)	25.2	26.1

V. Conclusion

In this research, a new decimal add/round approach has been presented according to the IEEE 754-2008. The floating-point multiplication and FMA are considered as an important key component in many engineering applications; the speed in the rounding architecture could be improved the overall performance. The latency of the proposed decimal add/round architecture is improved 3.4% than fast decimal rounding model. Future work may include implementing this architecture on different hardware platforms and expanding the design to work on extend-precision 128-bit and support broader application where decimal precision is critical.

REFERENCES

[1] V. Dasu and K. Ragini, "Implementation of Unbiased Rounding for 64-Bit Floating Point Adder," IEEE International Conference on Recent Trends in Microelectronics, Automation, Computing and Communications Systems (ICMACC), pp. 309–394, 2022. [Online]. Available: *https://doi.org/10.1109/ICMACC54824.2022.10093518*

[2] B. Harish, M. Rukmini, and K. Sivani, "Design of MAC unit for digital filters in signal processing and communication", International J Speech Technol, pp.561–565, 2022. [Online]. Available: https://doi.org/10.1007/s10772-021-09824-0

[3] P. Kuo, Y. Huang, and J. Huang, "Configurable Multi-Precision Floating-Point Multiplier Architecture Design for Computation in Deep Learning", IEEE 5th International Conference on Artificial Intelligence Circuits and Systems (AICAS), pp. 1-5, 2023. [Online]. Available: https://doi.org/10.1109/AICAS57966.2023.10168572

[4] D. Li, K. Mo, L. Liu, B. Pan, W. Li, W. Kang, and L. Li, "All-Digital Computing-in-Memory Macro Supporting FP64-Based Fused Multiply-Add Operation", Applied Sciences vol. 13, pp. 1-13, 2023. [Online]. Available: https://doi.org/10.3390/app13074085.

[5] P. Markstein, "The New IEEE-754 Standard for Floating Point Arithmetic", IEEE Std 754-2019-Redline, pp.1-148, 2019.

[6] M. Nabil, F. Al-Assfor, and M. Al-Ebadi, " Fast Combined Decimal/Binary Multiplier Based on Redundant BCD 4221-8421Digit Recoding", Basrah journal for engineering science, pp.40-47, 2017. [Online]. Available: https://doi.org/10.33971/bjes.17.1.6

[7] P. Sri, V. R S, and C. Poongodi, "A Low Power 10NM FinFET design of the GRFU-Multiply Accumulate Unit for DNN Accelerators", Research Square Journal, pp. 1-23, 2023. [Online]. Available: https://doi.org/10.21203/rs.3.rs-3249825/v1.

[8] R. Turaka, K. Bonagiri, T. Rao, G. Kumar, S. Jayabalan, V. Sreenivasulu, A. Panigrahy, and M. Prakash, "Design of approximate reverse carry select adder using RCPA", International Journal of Electronics Letters, pp. 146-156, 2023. [Online]. Available: https://doi.org/10.1080/21681724.2022.2062791.

[9] R. Vincent and S. Anju, "Decimal floating point format based on commonly used precision for embedded system applications", IEEE Annual International Conference on Microelectronics, Communication and Renewable Enenry, pp. 1-4,2013. [Online]. Available:https://doi.org/10.1109/AICERA-CMiCR.2013.6575957

[10] A. Vazquez and E. Antelo, "A High-Performance Significand BCD Adder with IEEE 754-2008 Decimal Rounding", 19th IEEE Symposium on Computer Arithmetic, pp.135-144, 2009. [Online]. Available: https://doi.org/10.1109/ARITH.2009.30.