Improving Accuracy, Area and Speed of Approximate Floating-Point Multiplication Using Carry Prediction

Marzie Fathi*
Faculty of Computer Engineering, Najafabad Branch, Islamic Azad University, Najafabad, Iran
fathi.marziye@gmail.com
Hooman Nikmehr
Department of Computer Architecture, University of Isfahan, Isfahan, Iran
nikmehr@eng.ui.ac.ir

Received: 27/Jul/2016 Revised: 31/Aug/2016 Accepted: 26/Sep/2016

Abstract

Arithmetic units are essential in digital circuit construction, and the enhancement of their operation would optimize the whole digital system. Among them, multipliers are the most important operational units and are used in a wide range of digital systems such as telecommunication signal processing, embedded systems, and mobile technology. The main drawback to a multiplication unit is its high computational load, which leads to considerable power consumption and an increased area of silicon. This also reduces the speed, which negatively affects the digital host functionality. Estimating arithmetic is a new branch of computer arithmetic implemented by discarding or manipulating a portion of arithmetic circuits and/or intermediate computations. Applying estimated arithmetic in arithmetic units would improve the speed, power consumption, and the implementation area by sacrificing a slight amount of result accuracy. This article develops and analyzes an estimated truncated floating-point multiplier for single precision operands that is capable of compensating for errors to a desired level by applying the least significant columns of the partial product matrix. These errors are caused by removing a number of carry digits in the partial product matrix which make a direct contribution to rounding the floating-point numbers. The evaluation results indicate that the proposed method improves speed, accuracy, and silicon area, in comparison with common truncated multiplication methods.

Keywords: estimated arithmetic; partial product matrix; rounding; truncated multiplier; error correction.

1. Introduction

Multipliers are one of the major arithmetic units commonly applied to digital systems like digital signal processing, telecommunication signal processing, embedded systems, and mobile systems. The major deficiency in these systems is their low speed, high power consumption, and high silicon-covered area, which cause a shortfall in digital system operation [1]. The Fast Fourier Transform (FFT) constitutes the basis of most telecommunication systems, like DVBT, UWB, WIMAX, WLAN, ADSL, and wireless telecommunication systems [2]. This function is implemented by multiplication and represents a complex function of products applied for converting the signal from frequency to time phase and vice versa.

$$x_k = \sum_{n=0}^{N-1} d_n . e^{\frac{j2\pi nk}{N}} \qquad k = 0, 1, 2, \cdots, N-1 \tag{1}$$

This function is applicable in telecommunication signal processing as well as image processing and digital signal processing. Optimized implementation of this block influences telecommunication systems with respect to speed and accuracy [3]-[4]. In addition to its multiplication or Multi-Accumulate (MAC) on some sources of Fused Multiply-Add (FMA), this function is a component other major arithmetical factors like Finite Impulse Response (FIR) [5], Discrete Cosine Transform (DCT) [5]-[6], and Multiple Input Multiple Output (MIMO) [7]. Consequently, any improvement in the multiplying algorithm would also contribute to both of these functions and to the telecommunication system in general.

In today's applications, where high-value calculation is of the essence, applying floating-point unit in FFT calculations and other telecommunication signal processing functions could be beneficial and efficient. In the recent past, the mentality and assumption was that floating-point has big area and high power; therefore, designs preferred to convert codes into fixed points or simulate floating-point functionality. This, of course, was a time-consuming and erroneous, with 30% of software design time attributed to it [8]. Nevertheless, this point is not always accurate , while applying optimized floating-point unit in telecommunication digital signal can even reduce energy consumption by 30% [9]-[10]. The focus of most research is now on approximated or estimated computations as a new approach in applications where the circuit would be able to generate inaccurate and faulty results. This approach is implemented by eliminating a portion of intermediate calculation. Although this function causes an increase in speed and a decrease in the silicon-covered area and power consumption, it slightly reduces arithmetic circuit accuracy [11]. Implementation of arithmetic units like multiplication have different methods. This counter, which replaces the common method and 3:2 counters, when using a 4:2 counter (four entry and two exit bits) in an 8×8 tree multiplier, is illustrated in Fig. 1. When all four bits are 1, an error

* Corresponding Author

occurs. In part (b) of Fig. 1, a reduction in multiplication, circuit capacity, and implementation time are observed [12].

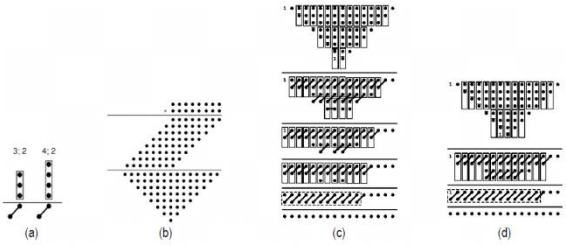

Figure.1. a) 4:2 and 3:2 counters, b) partial product generation in $(\mathbf{8}\times\mathbf{8})$-bit multiplier, c) the $(\mathbf{8}\times\mathbf{8})$ marked accurate multiplier using 3:2 counter, d) the$(\mathbf{8}\times\mathbf{8})$ estimated marked accurate multiplier using 4:2 counter[12]

The parallel truncated multiplier, developed in 1992 [13], was the first sample applied in estimated computing techniques and accepted in this field. In this method, eliminating the partial product LSP (Least Significant Part) matrix and keeping the MSP (Most Significant Part) section would lead to circuit area reduction. Of course, this phenomenon would generate error; so far, there exist no studies regarding removal of these errors [14]-[15].

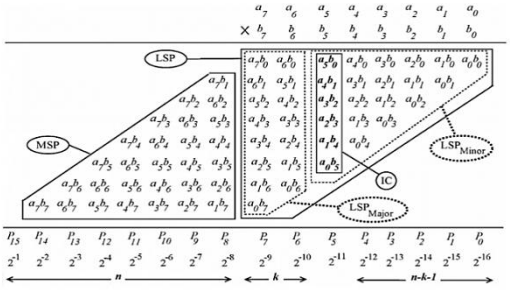

Figure. 2. The $(\mathbf{8}\times\mathbf{8})$ truncated multiplier and different elements of the partial product matrix for errror correction

Applying the LSP column of partial products to estimate the carry digit to MSP is another method that has been used in [16]-[17]. This method depends on multiplying the circuit entrance by a complicated estimating circuit but retaining accuracy. Here, we see implementation of a sequential estimated truncated floating-point digit at single precision, where the partial product of the matrix yield from the multiplication of mantissas of two digit floating-points are involved. By applying the four columns in LSP of the matrix and applying one small circuit for the carry digit estimation from the 5th column, the error is reduced in a significant manner and the outcome converges to the real value of multiplication.

In this article, the rounding of floating-point technique is introduced in Sec. 2, the combinational estimated multiplier is described in Sec. 3, the probability of maximum error is computed in Sec. 4, the proposed simulation method and its comparison with other states is addressed in Sec. 5, and the article is concluded in Sec. 6.

2. Rounding Floating –Point Numbers

Since 1990, floating-point numbers have been exhibited in accordance with IEEE 754-2008 standard. The two single and double precision widths are introduced for these numbers in this standard [1]. The partial product matrix generated from multiplying the mantissas by two floating-point digits at single precision is shown in Fig. 3. This product has 48 bits, but due to the standard limitations, only 24 MSP bits (23 bits of mantissas and one bit, the product of normalization) are held. To compensate for the error caused by eliminating the LSP bits, the rounding is performed by applying the three sticky (S), Gard (G) and Round (R) bits located in the LSP, calculated in accordance with Standard S from Eq. (2) [1].

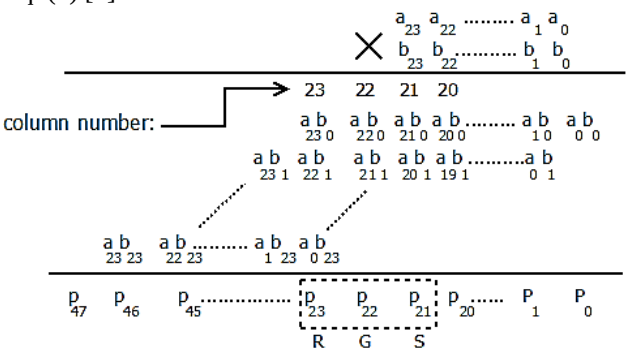

Figure. 3. Partial product matrix and the two mantissas floating-point digits at single precision

$$S = P_{21}\,\mathrm{OR}\,P_{20}\,\mathrm{OR}\ldots P_0 \tag{2}$$

3. Sequential Estimated Multiplier Capable of Error Compensation

3.1 Algorithm

Hardware implementation of this multiplier, including the 20 bits register A (for storing the 1st to 20th multiplicand), the 24 bits register B (for storing the multiplier), and 28 bits register P (for storing the product) with initial value of zero, is illustrated in Fig. 4. Since this multiplier is a sequential truncated to 28 bits and there is no need for the final response of the 20 LSP bits, the section of the circuit that computes these 20 LSP bits can be eliminated. To accomplish this, the 27 bits register, the Extended Register (Ex) with its initial value, is defined according to Eq. (3).

$$\begin{cases} Ex(3:0) = A(23:20) \\ A = A(19:0) \end{cases} \tag{3}$$

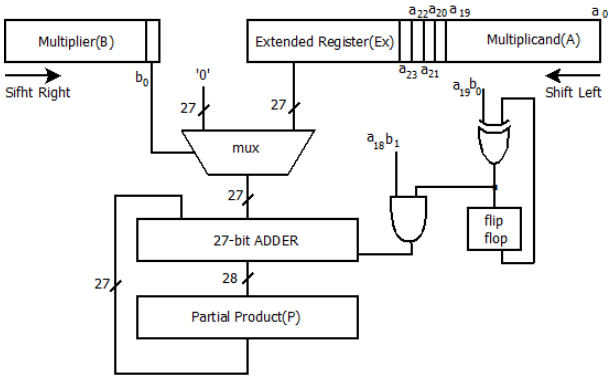

Figure. 4. Hardware implementation of multiplier, W4CE state

The operation of this multiplier is subject to the following steps:

1. In the first cycle, provided that the least significant bit of register *B* (b_0) is 1, the initial value of *EX* is added to *P* and the product is placed in *P*; otherwise, *P* holds its previous value.

$$P(i)=\begin{cases} P_{i-1}+Ex(i) & \text{if} \quad b_i=1 \\ P_{i-1} & \text{if} \quad b_i=0 \end{cases} \tag{4}$$

2. In every cycle *i*, first the *P* and *A* registers shift to the left by one bit and the exiting bit enters the *EX* register. Here, *B* shifts to the right by one bit.
3. In cycle *i*, step one is repeated 24 times until the last cycle yield is placed in *P*.

3.2 Error Compensating Circuit

As explained earlier, in the proposed multiplier, 20 columns have been eliminated from the LSP section of the partial product matrix, which in turn would lead to the elimination of the carry digit in this section, an error occurs. To compensate for this error, it should be estimated somehow; hence, what bits are involved in the creation of this carry digit should be known.

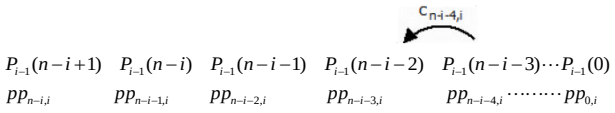

Figure. 5. Adding function in cycle i^{th} of partial product matrix

In this proposed method, this digit is estimated from the 19th column in cycle$i$ $(c_{n-i-4,i})$, three bits are required:

- The partial product created in the i^{th} cycle and at the $(n-i-4)^{th}$ space of $(pp_{n-i-4,i})$
- The intermediate total product in the $(i-1)^{th}$ cycle of $(P_{i-1}(n-i-3))$
- The carry generated in the i^{th} cycle and $(n-i-5)^{th}$ space of $(c_{n-i-5,i})$

These three elements are the entries of full adder; therefore, circuits must be designed that would receive these three entries and produce the $c_{n-i-4,i}$ and $P_{n-i-4,i}$ outputs. Since this proposed multiplier is of a sequential type and multiplication occurs in any part of the cycle, not all the above-mentioned entries are accessible. And among them, only $pp_{n-i-4,i}$ are accessible in every cycle and $c_{n-i-5,i}$ is the carry digit of the column adjacent, which in turn requires these three similar elements for its production and makes the circuit issue more complicated; for that reason, it is not considered. To compute $P_{i-1}(n-i-3)$, it can be said that this value in approximation is equal to the total of intermediate multiplication products in previous cycles, where the computation of function $c_{n-i-5,i}$ is not considered. Hence, to make a distinction between this estimated value and $P_{i-1}(n-i-3)$, it is presented as an accumulated value ($AV_{n-i-3,i-1}$) with a name change; that is, the total partial product must be protected in any cycle to allow access to $AV_{n-i-3,i-1}$. Here, a flip-flop with an initial value of 0 can be applied, the value of which in the i^{th} cycle would equal FF_i.

$$AV_{n-i-3,i-1}=FF_i=\begin{cases} 0 & \text{if} \quad i=0 \\ FF_{i-1}+pp_{n-i-3,i-1} & \text{if} \quad 0<i\le 20 \end{cases} \tag{5}$$

Equation 5 describes the total of two bits, for implementation of which a half adder could be applied. According to the descriptions above and the half-adder equation, the $c_{n-i-4,i}$ value can be obtained from Eq. (6-7).

$$AV_{n-i-3,i-3}=FF_i=FF_{i-1}\ \text{XOR}\ pp_{n-i-3,i-3} \tag{6}$$

$$c_{n-i-4,i}=AV_{n-i-3,i-1}\ \text{AND}\ pp_{n-i-4,i} \tag{7}$$

The implementation of this circuit is illustrated in Fig. 6, where ECV is the carry digit that is added to the 20th column in every cycle.

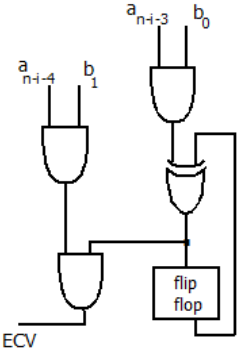

Figure. 6. Error compensation circuit

4. Computing Probability of Maximum Error

Maximum error occurs when the *R*, *G*, *S* bits and other effective bits are not available or, if available, their values are not appropriate for computing *S*; thus, no rounding takes place.

The bits *R* and *G*, in fact, represent the bits P_{23} and P_{22} respectively, (refer to Fig. 3), and $S=P_{21}\,\text{OR}\,P_{20}\,\text{OR}$

P_{19}. To compute the maximum error probability, it is necessary to calculate the probability of each one of P_{19} to P_{23} bits being 1 (here P_{19} is computed in both the truncated and estimated states).

To accomplish this task, the probabilities of the elements from partial product matrix involved in computing these bits being 1 must be computed. For example, the probability of P_{23} or *R* being 1 depends on the probability of every element being 1 in the 23rd column (Fig. 3) in every cycle, and each of these intermediate elements depends on another element; that is, every intermediate product like $P_i(n-i+1)$ in cycle *i* depends on $pp_{n-i+1,i}$ (intermediate product in cycle *i*), $P_{i-1}(n-i+2)$ (intermediate product in $(i-1)^{th}$ cycle), and $c_{n-i,i}$ (carry bit from adjacent column in cycle *i*).

If the probability of $P_i(n-i+1)$ being 1 is defined as $P_1(P_i(n-i+1))$. According to Table 1, its being 1 is achieved in different states.

Table 1. Possible states in computing $\boldsymbol{P_i(n-i+1)}$

$P_{i-1}(n-i+2)$	$pp_{n-i+1,i}$	$c_{n-i,i}$	$P_i(n-i+1)$	$c_{n-i+1,i}$
0	0	0	0	0
0	0	1	1	0
0	1	0	1	0
0	1	1	0	1
1	0	0	1	0
1	0	1	0	1
1	1	0	0	1
1	1	1	1	1

Consequently, the following equation is obtained:

$$\begin{aligned}P_1(P_i(n-i+1)) &= [\{P_1(P_{i-1}(n-i+2)) \times P_1(pp_{n-i+1,i})\} \\ &+ \{P_0(P_{i-1}(n-i+2) \times P_0(pp_{n-i+1,i})\}] \\ &\times P_1(c_{n-i,i}) \\ &+ [\{P_1(P_{i-1}(n-i+2)) \times P_0(pp_{n-i+1,i})\} \\ &+ \{P_0(P_{i-1}(n-i+2)) \times P_1(pp_{n-i+1,i})\}] \\ &\times P_0(c_{n-i,i})\end{aligned} \quad (8)$$

$P_1(pp_{n-i+1,i})$, $P_1(P_{i-1}(n-i+2))$, and $P_1(c_{n-i,i})$ are the probability of the intermediate product being 1 at i^{th} cycle, their being 1 at $(i-1)^{th}$ cycle, and their being 1 for carry digit bits from the adjacent column and i^{th} cycle, respectively (probability of being 0 is defined in the same order).

To compute the probability of $pp_{n-i+1,i}$ being 1, represented by $P_1(pp)$, it can be assumed that every intermediate bit is a random variable; thus:

$$P_1(pp) = \frac{1}{2} \times \frac{1}{2} = \frac{1}{4} \quad (9)$$

$$P_0(pp) = 1 - \frac{1}{4} = \frac{3}{4} \quad (10)$$

Likewise, it can be said that the probability $P_{i-1}(n-i+2)$ is in fact the probability $P_i(n-i+1)$, in the $(i-1)^{th}$ cycle; therefore, it depends on the same parameters but at i^{th} cycle, in a sense that it has a recessive state, the end condition of which is computation of $P_1(P_1(n-1))$ (an addition that occurs in the first cycle). This value in the first cycle is subject to values $pp_{n-1,1}$, $pp_{n-1,2}$ and $c_{n-3,2}$. By applying a table similar to Table 1, this can be presented as:

$$\begin{aligned}P_1(P_1(n-1)) &= [\{P_1(pp_{n-2,2}) \times P_1(pp_{n-1,1})\} \\ &+ \{P_0(pp_{n-2,2}) \times P_0(pp_{n-1,1})\}] \\ &\times P_1(c_{n-3,2}) \\ &+ [\{P_1(pp_{n-2,2}) \times P_0(pp_{n-1,1})\} \\ &+ \{P_0(pp_{n-2,2}) \times P_1(pp_{n-1,1})\}] \\ &\times P_0(c_{n-3,2}) \\ &= [\{\frac{1}{4} \times \frac{1}{4}\} + \{\frac{3}{4} \times \frac{3}{4}\} \times P_1(c_{n-3,2})] \\ &+ [\{\frac{1}{4} \times \frac{3}{4}\} + \{\frac{3}{4} \times \frac{1}{4}\}] \times P_0(c_{n-3,2}) \\ &= \frac{10}{16} \times P_1(c_{n-3,2}) + \frac{10}{16} \times P_0(c_{n-3,2})\end{aligned} \quad (11)$$

All values in Eq. (11) are random variables, the probability of which is computable through Eq. (9-10). To compute $P_1(c_{n-i,i})$ in Eq. (8), Eq. (12) must be applied, which is obtained from Table 1.

$$\begin{aligned}P_1(c_{n-i,i}) &= [P_1(P_{i-1}(n-i+1)) \times P_1(pp_{n-i,i})] \\ &+ [\{P_1(P_{i-1}(n-i+1) \times P_0(pp_{n-i,i})\} \\ &+ \{P_0(P_{i-1}(n-i+1) \times P_1(pp_{n-i,i}) \\ &\times P_1(c_{n-i-1,i})]\end{aligned} \quad (12)$$

Eq. (12) is a returnable one, the ending condition of it is determining the second cycle carry digit ($c_{n-2,2}$).

According to Fig. 7, computing $c_{n-2,2}$ is subject to two a_{n-2} and b_{n-2} values (intermediate product) and its value is1 when the volume of either a_{n-2} or b_{n-2} is 0 and the other is 1 and the carry digit of the adjacent column (c_{n-3}) is 1. This equation is a returnable one as well. Eventually, the need to compute the probability of c_1 would rise, which would be subject to the probability of a_0 and b_0 being 1; hence:

$$P_1(c_1) = P_1(a_0) \times P_1(b_0) = \frac{1}{4} \times \frac{1}{4} = \frac{1}{16} \quad (13)$$

Consequently, the probability of c_{n-2} being 1 can be computed through Eq. (14).

$$P_1(c_{n-2}) = (\frac{1}{4}\times\frac{1}{4}) + \frac{3}{8}(\frac{1}{4}\times\frac{1}{4}) + \ldots \frac{3}{8}(\frac{1}{4}\times\frac{1}{4}) = \frac{1}{16}\sum_{k=0}^{n-2}\left(\frac{3}{8}\right)^k \quad (14)$$

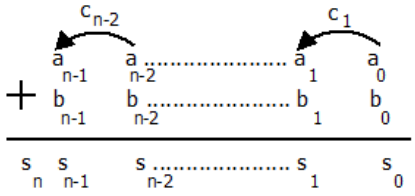

Figure. 7. Adding function of the two-bit digits

The $P_1(c_{n-3,2})$ and $P_0(c_{n-3,2})$ in Eq. (11) are computed with respect to the above-mentioned description. In the same sequence, all parameters in Eq. (8) can be computed.

4.1 Maximum Error Probability for No Error Compensation State

To determine the effectiveness of this proposed method of compensating for errors caused by truncation in a partial product matrix, different state of truncation is of concern.

In the first state, the product contains only 24 bits and no extra bit is held for rounding (WO3C). In the second state, in addition to the 24 bits, bits P_{23}, P_{22} and P_{21} are considered as *R*, *G*, and *S* bits held (W2CE). In the next state, bits P_{23}, P_{22}, and P_{21} are considered as *R*, *G*, and *S*, and P_{20} is considered as a bit following *S* (W3CE). In the last state, bits P_{23}, P_{22} and P_{21} are considered as bits *R*, *G*, and *S*, and bits P_{20}, P_{19} are considered as the two bits following *S*, which are held (W4CE). The maximum error and rounding are defined based on occurrences of different states of bits *R*, *G*, and S, Table 2.

Table 2. Rounding states occurrence

R	*G*	*S*	Rounding
0	0	0	0
0	0	1	0
0	1	0	0
0	1	1	0
1	0	0	0
1	0	1	1
1	1	0	1
1	1	1	1

Now, if the probabilities of *R*, *G*, and *S* bits being 1 equal $P_1(R)$, $P_1(G)$, and $P_1(S)$ respectively, and the probability of maximum error is $P_{\max}(e)$, Eq. (15) yields:

$$P_{\max}(e) = [P_1(R)\times P_0(G)\times P_1(S)] + [P_1(R)\times P_1(G)\times P_0(S)] + [P_1(R)\times P_1(G)\times P_1(S)] \quad (15)$$

In fact, Eq. (15) is the maximum error occurrence probability at WO3C state in Table 2. At W2CE, the maximum error occurs when *R*=1, *G*=0, *S*=0, and P_{20} = 1; but since this bit is not available, the rounding would be calculated wrongly. In W3CE, maximum error occurs when *R*=1, *G*=0, *S*=0, P_{20} = 0, and P_{19} =1. A similar result is obtained for W4CE state when *R*=1, *G*=0, *S*=0, P_{20} = 0, P_{19} = 0, and P_{18} = 1. The results obtained by these computations will be explained later. In all cases, the probability of every bit being 1 or 0 is calculated according to the method described in Sec. 4.

4.2 Maximum Error Probability for Error Compensation States

Computing probability at this state is related to a time when the error compensation circuit is applied for carry propagation estimation. Applying this circuit predicts the carry propagation wrongly in both states, Table 1, Therefore, by knowing the existence of error in computing carry propagation, and its appearance in the product, with reference to Eq. (16) the following is yielded:

$$AV_{n-i+1,i} = \begin{cases} AV_{n-i+2,i-1} \oplus pp_{n-i+1,i} & \text{if} \quad 2 < i \le 24 \\ AV_{n,i-1} & \text{if} \quad i = 2 \end{cases} \quad (16)$$

According to the definition of exclusive OR function function, this product would be equal 1, only if the number of bits that equal 1 is an odd number. Hence, $P_1(AV_i)$, the probability function of exclusive OR function can be represented as

$$P_1(AV_i) = \sum_{k=1}^{\left\lceil \frac{i}{2} \right\rceil} \binom{i}{2k-1} \left(\frac{1}{4}\right)^{2k-1} \left(\frac{3}{4}\right)^{i-(2k-1)} \quad (17)$$

The results obtained by replacing the digits in all the above-mentioned equations in both cases of error state, with and without compensation, are presented in Table 3. As observed these results are very close to one another, indicating the error compensation circuit accuracy.

Table 3. Digit values of muximum error probability

Proposed method	Maximum error probability for no compensation state	maximum error probability for error compensation state
WO3C	0.4977495060	—
W2CE	0.06249765004	0.062621245550
W3CE	0.0314991165	0.03143649401
W4CE	0.01581255649	0.01587555472

5. Simulation

This simulation is made through *MATLAB* software and the implementation area and delay are assessed through *Synopsys Design Compiler* with *TSMC18* typical library.

To determine the error compensation circuit effect, in addition to simulation of this proposed multiplier for

complete multiplication, the following steps are applied: elimination of LSP section (WO3C), LSP three columns (W3CO), two columns and carry propagation estimation from the third column (W2CE), three columns and carry propagation estimation from the fourth column LSP (W3CE).

5.1 Accuracy Assessment

Here, the absolute and relative error, a product of two random digit multiplications at 24 cycle, is calculated and its diagram is drawn (Fig. 8 and Fig. 9). The cycles presented are 1, 4, 8, 12, 16, 20, and 24 only.

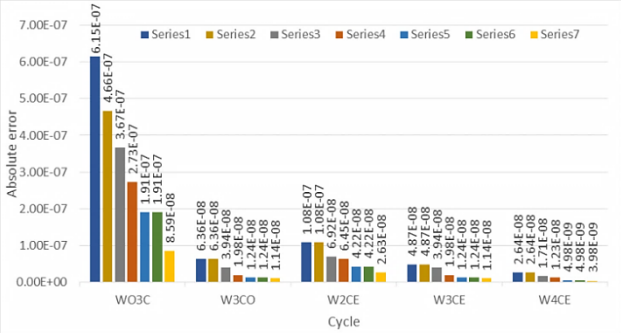

Figure. 8. Selected absolute error diagram in seven cycles

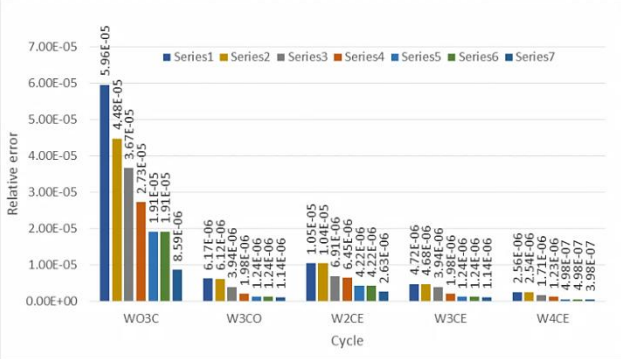

Figure. 9. Selected relative error diagram in seven cycles

The proposed design was simulated for 5,000 random digits as well, and the absolute and relative errors in the seven cycles defined were calculated and plotted (Fig. 10 and Fig. 11). The results indicate a reduction in error and the positive effect of the error compensation circuit.

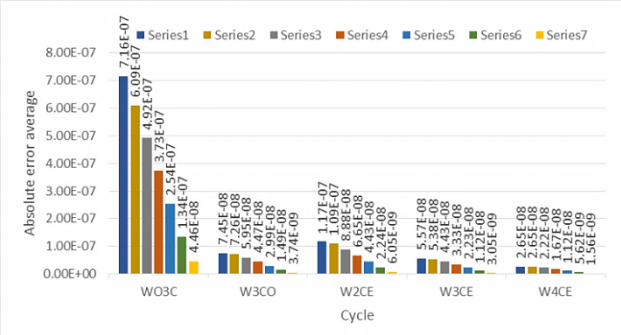

Figure. 10. Absolute error average of 5,000 random digits in seven cycles

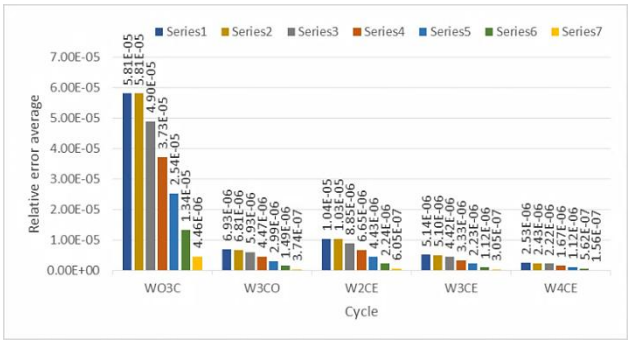

Figure. 11. Relative error average of 5,000 random digits in seven cycles

To determine error convergence, the fitting curve of the absolute and relative error average of 5,000 random digits for all states is drawn in Fig. 12. This curve indicates the error converges towards 0.

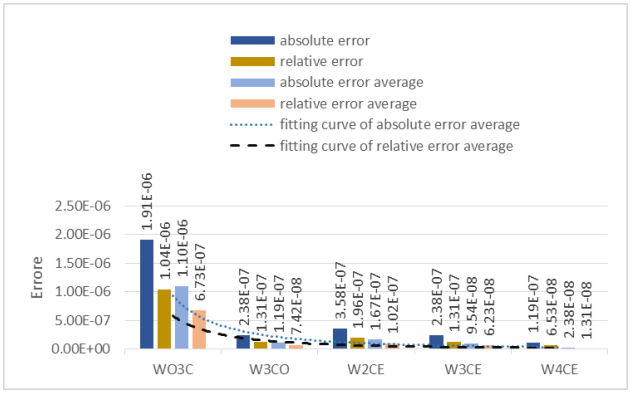

Figure. 12. The error coverges towards zero diagram

5.2 Area and Delay Assessment

In area assessment, the results of this proposed multiplier and truncated conditions are presented in the two combinational and non-combinational (TCA), and combinational and non-combinational area of circuits and intermediate connections (TA) are reported (Table 4). Comparison of different states, specifically, W2CE and W3CE, indicates the effective contribution of the estimating circuit.

Table 4. Truncated multipliers area report

	WO3C	W3CO	W2CE	W3CE	W4CE
TCA (um^2)	18431.894617	27219.185976	26255.382937	27071.902759	28077.475626
TA (um^2)	314113.38741	428701.223207	425790.735110	437927.316211	437626.215616

The timing report of all truncated multiplier states and this proposed multiplier are presented in Table 5. Comparison of different states, W2CE and W3CO in specific, indicates the effective contribution of the error compensation circuit. There exists a direct relation

between an increase in LSP column for error compensation and circuit delay.

Table 5. Truncated multipliers timing report

Truncated multipliers	WO3C	W3CO	W2CE	W3CE	W4CE
Delay (ns)	473.33	521.33	473.33	593.33	593.33

In Table 6, previous works has been summarized. Comparisons have been done for two parameters area and delay, and for both integer and floating-point numbers. In all of the papers presented, only two parameters area or delay have been discussed and have not been reported simultaneously.

As are observed, the proposed multiplier is in the proper position, exception in case [22] where this paper has not obtained any report about accuracy and area. Likewise, only there is one reported area that relates to a 16-bits integer multiplier.

Table 6. Comparison of proposed multiplier and previous work where fp represents floating-point numbers

Mult. method	Size (bits)	Area (um^2)	Delay (ns)
[18]	16 int	1.22×10^5	Not reported
[19]	8 int	Not reported	940.8
[20]	32 fp	Not reported	3055.2
[21]	16 int	Not reported	836.32
[22]	32 fp	Not reported	163.68
[23]	32 fp	Not reported	2099.424
[23]	32 fp	Not reported	2264.136
Proposed	32 fp	4.38×10^5	593.33

To determine the contribution of the LSP column number in error compensation, the diagram of relative error average and delay is presented in Fig. 13, where an increase in LSP column number for the purpose of compensation results in an increase in delay. This is why no attempt is made to increase the column number.

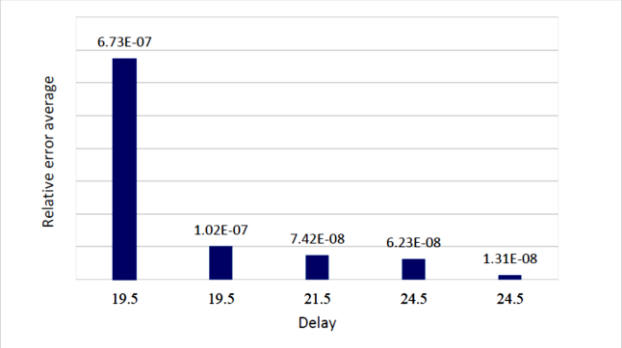

Figure. 13. Relative error average and delay diagram

6. Conclusion

The algorithm for and implementation of a sequential truncated multiplier for floating-point digits at single-precision that is capable of error compensation are proposed. This method is implemented by applying four MSPS from a semi-LSP matrix of a partial product and estimated carry from the fifth column using a small circuit. Results indicate that the error compensation circuit can significantly reduce the error caused by lake of carry digits from the eliminated columns. Among all simulation states, W4CE is the best candidate with respect to finding a trade-off between speed, accuracy, and area, and it can be used in digital signal processing, telecommunication signal processing, embedded systems, mobile systems that require a small area, and low power consumption.

References

] B. Parhami, Computer Arithmetic, New York: Oxford University Press, 2000.

[2] X. Guan, Y. Fei and H. Lin, "Hierarchical design of an application-specific instruction set processor for high-throughput and scalable FFT processing." IEEE Transactions on Very Large Scale Integration (VLSI) Systems, vol. 20, no.3, (2012), pp. 551-563.

[3] A. V. Oppenheim, R. W. Schafer and J. R. Buck, Discrete-Time Signal Processing, USA: Prentice-Hall, 1998.

[4] J. R. Choi, H. G. Kim, S. S. Han and S. C. Hwang, "Variable 2K/4K/8K-point FFT/IFFT with compact memory for OFDM-based DVB-T system," International Conference on Systems and Informatics (ICSAI), May 2012, pp. 977-980.

[5] H. M. Hassan, K. Mohammad and A. F. Shalash, "Implementation of a reconfigurable ASIP for high throughput low power DFT/DCT/FIR engine," EURASIP Journal on Embedded Systems, No.1, 2012, pp. 1-18.

[6] J. Sohn and E. E. Swartzlander Jr, "Improved architectures for a fused floating-point add-subtract unit," Circuits and Systems I: Regular Papers, IEEE Transactions on, vol. 59, no. 10, 2012, pp. 2285-2291.

[7] X. Chen, A. Minwegen, Y. Hassan, D. Kammler, S. Li, T. Kempf and G. Ascheid, "Efficient multi-mode MIMO detection using reconfigurable ASIP," 20th Annual International Symposium on Field-Programmable Custom Computing Machines (FCCM), April 2012, pp. 69-76.

[8] D. Menard, D. Chillet, F. Charot and O. Sentieys, "Automatic floating-point to fixed-point conversion for DSP code generation," ACM. In Proceedings of the 2002 international conference on Compilers, architecture, and synthesis for embedded systems, October 2002, pp. 270-276.

[9] S. Z. Gilani, N. S. Kim and M. Schulte, "Virtual floating-point units for low-power embedded processors," 23rd International Conference on Application-Specific Systems, Architectures and Processors (ASAP), July 2012, pp. 61-68.

[10] S. Z. Gilani, N. S. Kim and M. Schulte, “Energy-efficient floating-point arithmetic for software-defined radio architectures,” 2011 IEEE International Conference on Application-Specific Systems, Architectures and Processors (ASAP), September 2011, pp. 122-129.
[11] P. Korkmaz, B. E. Akgul and K. V. Palem, “Energy, performance, and probability tradeoffs for energy-efficient probabilistic CMOS circuits,” Circuits and Systems I: Regular Papers, IEEE Transactions on, vol. 55, no. 8, 2008, pp. 2249-2262.
[12] D. Kelly, B. Phillips and S. Al-Sarawi, “Approximate signed binary integer multipliers for arithmetic data value speculation,” In Conference on Design & Architectures for Signal and Image Processing, 2009.
[13] Y. C. Lim, “Single-precision multiplier with reduced circuit complexity for signal processing applications,” Computers, IEEE Transactions on, vol. 41, no. 10, 1992, pp. 1333-1336.
[14] N. Petra, D. D. Caro, V. Garofalo, E. Napoli and A. G. Strollo, “Truncated binary multipliers with variable correction and minimum mean square error,” Circuits and Systems I: Regular Papers, IEEE Transactions on, vol. 57, no. 6, 2010, pp. 1312-1325.
[15] V. Garofalo, N. Petra and E. Napoli, “Analytical calculation of the maximum error for a family of truncated multipliers providing minimum mean square error,” Computers, IEEE Transactions on, vol. 60, no. 9, 2011, pp. 1366-1371.
[16] E. J. King and E. E. Swartzlander, “Data-dependent truncation scheme for parallel multipliers,” Conference Record of the Thirty-First Asilomar Conference on Signals, Systems & Computers, vol. 2, November 1997, pp. 1178-1182.
[17] E. E. Swartzlander, “Truncated multiplication with approximate rounding,” Conference Record of the Thirty-Third Asilomar Conference on Signals, Systems, and Computers, vol. 2, October 1999, pp. 1480-1483.
[18] F. Farshchi, M. Abrishami, and S.M. Fakhraie, "New approximate multiplier for low power digital signal processing." The 17th CSI International Symposium on Computer Architecture & Digital Systems, October 2013, pp. 25-30.
[19] Z. Vasicek and L. Sekanina, "Evolutionary design of approximate multipliers under different error metrics." Design and Diagnostics of Electronic Circuits & Systems, 17th International Symposium on. 2014 Apr 23, pp.135-140.
[20] C. M. Guardia and E. Boemo, "FPGA implementation of a binary32 floating point cube root." Programmable Logic (SPL), Nov 2014, pp. 1-6.
[21] A. Sunny, B. K. Mathew and P. B. Dhanusha, "Area Efficient High Speed Approximate Multiplier with Carry Predictor." Procedia Technology 24, 2016, pp. 1170-1177.
[22] S. Sivanantham, "Design of low power floating point multiplier with reduced switching activity in deep submicron technology." International Journal of Applied Engineering Research, vol. 8, no. 7, 2013, pp. 851-59.
[23] P. Koneru, T. Sreenivasu and A. P. Ramesh, "Asynchronous Single Precision Floating Point Multiplier using Verilog HDL." IJ of Advanced Research in Electronics and Communication Engineering, 2013 Nov.

Marziye Fathi received her B.SC degree in Computer Hardware Engineering and M.SC degree in Computer Architecture Engineering both from Islamic Azad University of Najafabad (IAUN), Najafabad, Iran, in 2010 and 2015, respectively. Her area research interests include digital arithmetic, VLSI and image processing.

Hooman Nikmehr received his BSc in Electronic Engineering and MSc in Computer Architecture Engineering both from University of Tehran, Tehran, Iran, in 1992 and 1997, respectively, and PhD degree in Computer Engineering from the University of Adelaide, Adelaide, Australia, in 2005. He is an Assistant Professor with the Department of Computer Architecture, University of Isfahan, Isfahan, Iran. His current research interests include VLSI, digital arithmetic, computer architecture, reconfigurable hardware design and low-power design.