International Journal of Integrated Engineering, Vol. 10 No. 1 (2018) p. 114-133

DOI: https://doi.org/10.30880/ijie.2018.10.01.018

High performance 8-bit approximate multiplier using novel 4:2 approximate compressors for fast image processing

Fatemeh Ranjbar[1], Yahya Forghani[1*], Davoud Bahrepour[1]

[1]Department of Computer
Islamic Azad University, Mashhad Branch, Mashhad, IRAN.

Received 24 February 2018; accepted 9 April 2018, available online 30 April 2018

Abstract: In this paper, a novel 8-bit approximate multiplier is proposed based on three novel 4:2 approximate compressors which its delay and error is less than those of the multipliers constructed by traditional 4:2 approximate compressors, and its delay is also less than that of an 8-bit multiplier constructed by using 3:2 precise compressors. To do so, each novel compressor is designed such that its output carry is independent of the output carry of its previous compressor in the multiplier. Therefore, the problem of carry propagation delay is eliminated and a fast multiplier is constructed. To obtain the most accurate multiplier, the best compressor of the three proposed compressors for each multiplier's column is determined using the genetic algorithm. Moreover, one can use the approximate compressors only at the k least significant multiplier's columns for more error reduction. The proposed multiplier is used for image blending and image compression. Our simulations show that for example the error and the delay of the proposed method for k=9 is at-least 32.52% and 33.10% less than those of traditional 4:2 approximate compressor based multipliers, respectively.

Keywords: Approximate Compressors, Dadda Multiplier, Genetic Algorithm, Image Blending, Compression

1. Introduction

In many signal processing problems such as lossy compression of images, sounds and films, the results of using approximation computations and precise computations have no significant difference from the user's point of view. JPEG, MP3 and MPEG are some well-known algorithms for the mentioned lossy compression problems. Each of these algorithms makes some distortion on the original file in order to achieve a better compression ratio. This amount of distortion of an image, sound or film is usually ignorable from the user's viewpoint. In such problems, approximate computations can be used instead of precise computations to reduce the number of transistors, power consumption or delay.

Approximate addition and approximate multiplication are some aspects of approximate computations. In [1-4], some approximate adders and approximate multipliers were studied and analyzed, elaboratively. In [2], in order to reduce the number of transistors and power consumption, an approximate mirror full-adder (AMA) was proposed. Then, this approximate full-adder was utilized in the JPEG algorithm. In [3], another approximate full-adder was proposed based on probabilistic CMOS. This technology consumes very low power. These approximate full-adders can be used to construct an approximate multiplier. In [5], an n-bit approximate adder named low-part-or-adder (LOA) was proposed. This adder computes the summation of each of k(<=n) least significant bits approximately by using only an Or-gate instead of a half or full-adder. This adder ignores carry propagation in its k least significant bits. This fast adder was then used in a neural network and a fuzzy system utilized in fast face recognition. In [6], an approximate booth multiplier was proposed which then was used in low-pass finite impulse response and then applied to digital signal processing. In [7], in order to reduce delay and the number of transistors, a truncated multiplier was proposed. In this approximate multiplier, the k least significant bits of partial products are truncated or ignored, and the remaining most significant bits of partial products are added with each other and the result then is added with a constant to compensate the truncation, and the final result is rounded to p bits. In [8], in order to increase the accuracy of truncated multiplier, the compensated value is determined based on the value of truncated bits. In other words, the compensated value is not constant any longer. Another truncated multiplier was proposed in [9] where the maximum absolute error is guaranteed to be no more than 1 unit of least position. This multiplier was implemented in Field Programmable Gate Array (FPGA), and then it was applied for image blending [10]. In [11], an approximate 2-bit multiplier was proposed to reduce power consumption, and then bigger multipliers were constructed based on the mentioned 2-bit multiplier. This multiplier then was applied to design an approximate Gaussian smoothing image improvement filter for noise reduction. In [12], an approximate signed multiplier was proposed which is 20% faster than a precise signed multiplier. In [13, 14], in order to reduce the carry propagation delay, an approximate adder was proposed that compute i-th bit of the summation of the two number

*Corresponding author: yahyafor2000@yahoo.com

A and B, i.e. S_i, based on only the i-th and (i-1)-th bit of the two numbers. Then, to reduce the error, an error signal is also produced based on the same two bits. If the error signal is added to S, the accurate value of sum of A and B is obtained. But, this involves using a time-consuming Carry Propagation Adder (CPA). Therefore, to increase the speed, the sum of S and the error signal were computed approximately by using only some or-gates instead of some half and full-adders. In each of [15] and [16], a 4:2 approximate compressor with the mean square error (MSE) of 0.25 was proposed and then each one was used to design a fast 8-bit Dadda multiplier. The delay of each of these two 4:2 approximate compressors is less than that of 4:2 precise compressors.

The delay of an 8-bit approximate Dadda multiplier constructed by using the traditional 4:2 approximate compressors is also less than that of an 8-bit Dadda multiplier constructed by using 4:2 precise compressors [15]. But, our experiments show that the delay of the former multiplier and an 8-bit Dadda multiplier constructed by using only some half-adders and full-adders (3:2 precise compressors) does not differ. However, the number of transistors and the power consumption of the former multiplier is less than the successor multiplier.

In this paper, a novel 8-bit approximate multiplier is proposed based on three novel 4:2 approximate compressors which its delay and error is less than those of the Dadda multipliers constructed by the traditional 4:2 approximate compressors, and its delay is less than that of Dadda multiplier constructed by using only some half-adders and full-adders (3:2 precise compressors). To do so, the novel compressor is designed such that its output carry is independent of the output carry of its previous compressor in the multiplier. Therefore, the problem of carry propagation delay is eliminated and a fast multiplier is constructed. Using each of the proposed compressors at each column of partial products has different effect on the multiplier error and also affects the next column of partial products, because the output carry of each compressor is connected to the next compressor input. To obtain the most accurate multiplier, the best compressor of the three proposed compressors for each column of partial products is determined using the genetic algorithm. Moreover, one can use the approximate compressors only at k least significant columns of partial products for more error reduction. Therefore, for each k, a different multiplier is constructed.

The proposed 8-bit multiplier is applied for image blending and image compression. Simulations show that the delays and also the errors of the multipliers constructed by the traditional 4:2 approximate compressors are more than those of the proposed multipliers for some k. For example, the error and the delay of the proposed method for k=12 is at-least 32.52% and 33.10% less than those of the traditional 4:2 approximate compressor based multipliers, respectively.

The innovations of this paper are as follows:

- Introducing a novel 4:2 approximate compressor where its output carry is independent of some of its inputs.
- Introducing a novel approximate 8-bit multiplier based on the proposed compressor.
- Using genetic algorithm to decrease the proposed multiplier error.

In continue, in section 2, traditional compressor-based multipliers are introduced. Then, in section 3, our novel compressors and multipliers are proposed. In section 4, by using some simulations, the proposed multipliers are compared with some other multipliers. Then, our proposed approximate multipliers are applied for image blending and image compression in section 5. Finally, the paper is concluded in section 6.

2. Traditional 4:2 compressor-based multipliers

A compressor computes the sum of some 1-bit numbers. Fig. 1.a shows the general form of a 4:2 compressor, and Fig. 2 depicts an especial implementation of a 4:2 compressor based-on full-adder. Fig. 3 shows a low delay implementation [17]. Similar implementations can be found in [18-23].

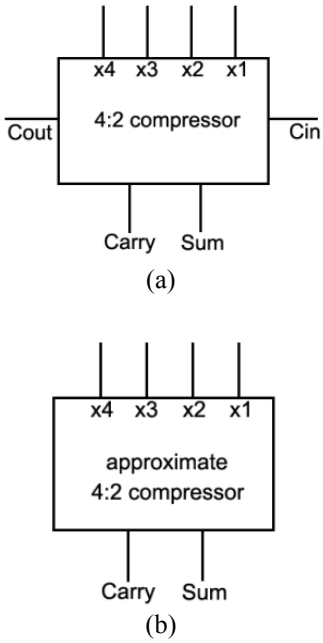


Fig. 1. The general form of a 4:2 compressor (a) with input carry, (b) without input carry [15].

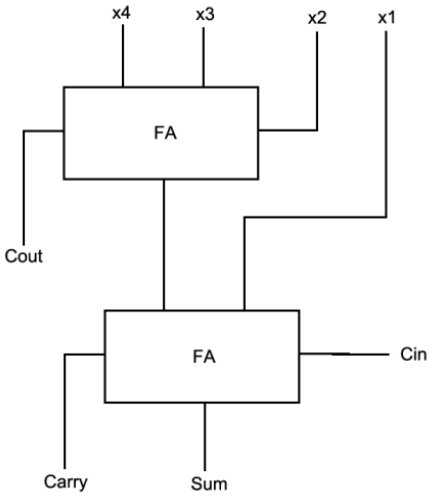


Fig. 1. The Full-adder-based implementation of 4:2 precise compressor [15].

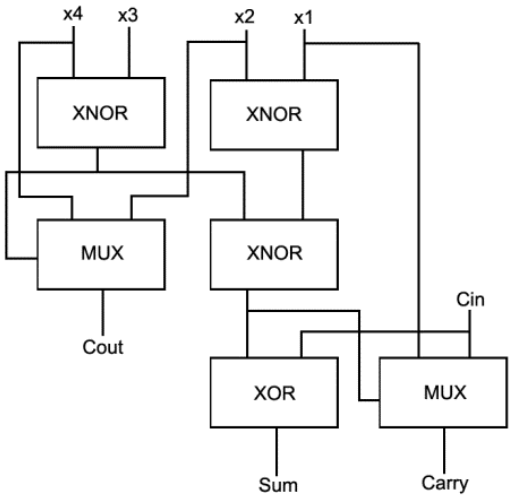


Fig. 2. A low delay implementation of 4:2 precise compressor [17].

In each of the 4:2 approximate compressors proposed in [15] and [16], the input carry was supposed to be zero. Therefore, the input carries were removed from their input list. Fig. 1.b shows the general form of a 4:2 compressor without input carry. Fig. 4 depicts two special implementations of the mentioned approximate compressors [15,16]. Table 1 shows the truth tables of these two circuits. According to these tables, the MSE of each of these two compressors is 0.25.

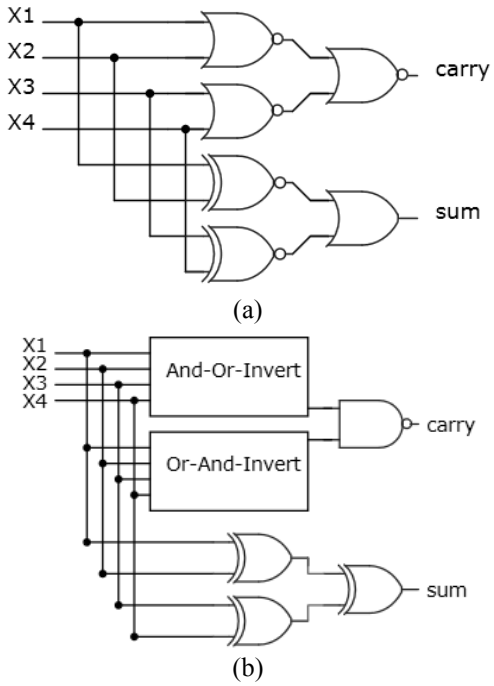


Fig. 3. The 4:2 approximate compressors without input carry proposed in (a) [15], and (b) [16].

Table 1. The Truth tables of 4:2 approximate compressors proposed in (a) [15], and (b) [16]. The difference column indicates the difference between the output of approximate and precise compressors.

(a)

x4	x3	x2	x1	carry	sum	difference
0	0	0	0	0	1	1
0	0	0	1	0	1	0
0	0	1	0	0	1	0
0	0	1	1	0	1	-1
0	1	0	0	0	1	0
0	1	0	1	1	0	0
0	1	1	0	1	0	0
0	1	1	1	1	1	0
1	0	0	0	0	1	0
1	0	0	1	1	0	0
1	0	1	0	1	0	0
1	0	1	1	1	1	0
1	1	0	0	0	1	-1
1	1	0	1	1	1	0
1	1	1	0	1	1	0
1	1	1	1	1	1	-1

(b)

x4	x3	x2	x1	carry	sum	difference
0	0	0	0	0	0	0
0	0	0	1	0	1	0
0	0	1	0	0	1	0
0	0	1	1	1	0	0
0	1	0	0	0	1	0
0	1	0	1	1	0	0
0	1	1	0	1	0	0
0	1	1	1	1	1	0
1	0	0	0	0	1	0
1	0	0	1	1	0	0
1	0	1	0	1	0	0
1	0	1	1	1	1	0
1	1	0	0	1	0	0
1	1	0	1	1	1	0
1	1	1	0	1	1	0
1	1	1	1	1	0	-2

The second stage of Fig. 5 shows partial product matrix of an 8-bit multiplier using the dot notation. Each dot is an unspecified bit. Each partial product is computed using an AND gate. Partial products may be rearranged in a tree-like format as the first stage of Fig. 6. Each multiplier must compute the summation of the partial products. In other words, each multiplier must reduce these eight rows of partial products to two rows, then, the final results is produced by the summation of these two binary numbers using a CPA.

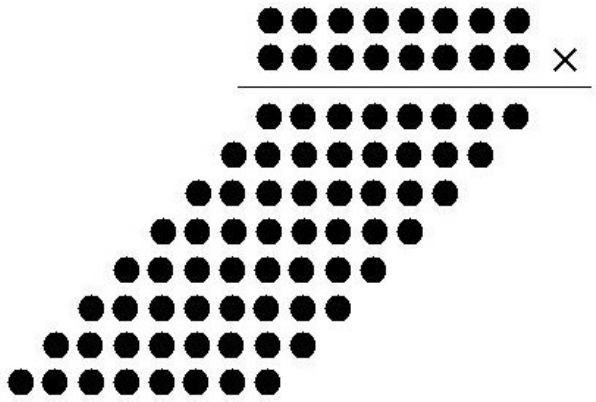


Fig. 4. Partial product matrix of an 8-bit multiplier

Fig. 6 shows an 8-bit Dadda multiplier constructed by only some half and full-adders, and Fig. 7 depicts an 8-bit Dadda multiplier constructed by some half and full-adders and some 4:2 compressors. Each rectangle represents a half adder, full adder or 4:2 compressor. Each of half adder, full adder and 4:2 compressor output a summation and a carry shown with two dots in the same column and the next column of the next stage, respectively.

As it can be seen, the former multiplier is performed in five stages while the successor multiplier is performed in three stages. In the two first stages of the successor multiplier shown in Fig. 7, Carry Save Adders (CSA) are used to decrease the eight rows of partial products to two rows, and then a CPA is used to compute the final result. In [15] and [16], in order to decrease the power consumption and delay of multiplier shown in Fig. 7, 4:2 approximate compressor was used instead of precise compressor. The delay of an 8-bit approximate Dadda multiplier constructed by using the traditional 4:2 approximate compressors [15, 16] is less than that of an 8-bit Dadda multiplier constructed by using 4:2 precise compressors. But, our experiments show that the delay of the former and 8-bit Dadda multiplier constructed by using only some half-adders and full-adders (3:2 precise compressors) does not differ. However, the number of transistors and the power consumption of the former multiplier is less than the successor multiplier.

In this paper, a novel 8-bit approximate multiplier is proposed based on three novel 4:2 approximate compressors which its delay and error is less than those of the Dadda multipliers constructed by the traditional 4:2 approximate compressors, and its delay is also less than that of the Dadda multiplier constructed by using only some half-adders and full-adders (3:2 precise compressors). To do so, each novel compressor is designed such that its output carry is independent of the output carry of its previous compressor in the multiplier. Therefore, the problem of carry propagation delay is eliminated and a fast multiplier is constructed.

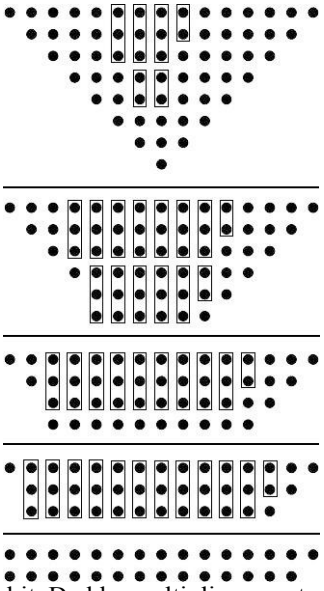


Fig. 5. An 8-bit Dadda multiplier constructed by only some half and full-adders (each rectangle represents a half adder or full adder) [18].

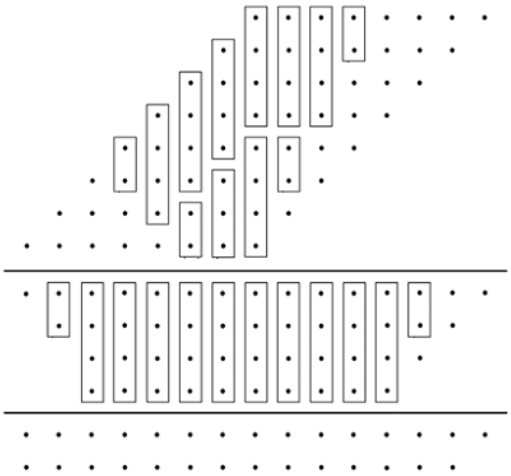


Fig. 6. Using 4:2 compressors to construct an 8-bit multiplier (each rectangle represents a half adder, full adder or 4:2 compressor) [15].

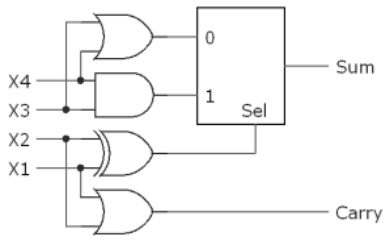
3. Our proposed multipliers

Before proposing our novel multipliers, its novel components, i.e. the novel compressors, must be introduced.

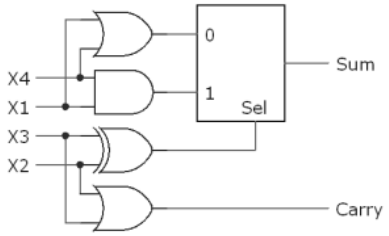
3.1 Our Proposed Compressors

Fig. 8 shows the circuits of the proposed approximate compressors and Table 2 shows their truth tables. As it can be seen, the output of each of these approximate compressors differs from the output of precise compressor for four truth table states. Therefore, the MSE of each of these approximate compressors is equal to 0.25 which is the same as that of the compressors proposed in [15] and [16]. The advantage of each proposed approximate compressor with respect to the traditional approximate 4:2 compressors is that its output carry is independent of its two inputs. For example, the output

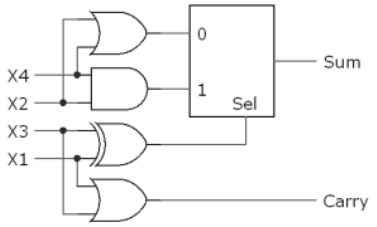
carry of the first proposed approximate compressor, shown in Fig. 8.a, is independent of x3 and x4 (third and fourth input). Therefore, if the output carry of an instance of this compressor is connected to the input x3 or x4 of another instance of this compressor, then the output carry of the successor can be produced even if the output carry of the former is not ready. Therefore, if the components of a CPA are such compressors, then this CPA does not have the carry propagation delay problem and consequently is very rapid.



(a)



(b)



(c)

Fig. 7. The three proposed 4:2 compressors.

Table 2. Truth tables of the three proposed compressors.

Input				1st compresor			2st compresor			3st compresor		
x4	x3	x2	x1	carry	sum	differ.	carry	sum	differ.	carry	sum	differ.
0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	1	1	0	1	0	1	0	1	0	1
0	0	1	0	1	0	1	1	0	1	0	1	0
0	0	1	1	1	0	0	1	0	0	1	0	0
0	1	0	0	0	1	0	1	0	1	1	0	1
0	1	0	1	1	0	0	1	0	0	1	0	0
0	1	1	0	1	0	0	1	0	0	1	0	0
0	1	1	1	1	1	0	1	1	0	1	1	0
1	0	0	0	0	1	0	0	1	0	0	1	0
1	0	0	1	1	0	0	0	1	1	1	0	0
1	0	1	0	1	0	0	1	0	0	0	1	1
1	0	1	1	1	1	0	1	1	0	1	1	0
1	1	0	0	0	1	1	1	0	0	1	0	0
1	1	0	1	1	1	0	1	1	0	1	1	0
1	1	1	0	1	1	0	1	1	0	1	1	0
1	1	1	1	1	1	-1	1	1	-1	1	1	-1

3.2 Our Proposed CPA

Each of Fig. 9, Fig. 10 and Fig. 11 depicts a CPA which computes the sum of three 3-bit numbers, i.e. the numbers A, B and C, by using different 4:2 approximate compressors, i.e. the compressors proposed in [15] and [16], and our proposed approximate compressors, respectively. In the CPA constructed by using our proposed compressors, the output carry of each of proposed compressor was connected to the fourth input (x4) of its successor compressor. Since the output carry of the proposed compressors is independent of the fourth input, the proposed CPA does not have the carry propagation delay problem. In the mentioned figures, the longest path of each CPA was shown with a thick line. As can be seen, the longest path of each CPA constructed by the compressors proposed in [15] or [16] is the path from input carry of CPA to the output carry of the last compressor, while the longest path of the CPA constructed by our proposed compressor is the path from an input of a compressor to the sum pin of its successor compressor. Meanwhile, the longest path to the output carry of the proposed CPA is the path from an input of its last compressor to that compressor output carry which is too short. This path was also depicted in Fig. 11 by a thick line.

Fig. 12 depicts the longest path in an 8-bit CPA constructed by the compressors proposed in [15] or [16], and Fig. 13 depicts the longest path in an 8-bit CPA constructed by the proposed compressors. As can be seen, the longest path of a CPA constructed by the compressors proposed in [15] and [16] becomes too longer for the bigger CPA, while the longest path of a CPA constructed by the proposed compressors is constant. Strictly speaking, regardless of the CPA length, the longest path of the CPA constructed by our proposed compressor is the path from an input of a compressor to the sum pin of its successor compressor.

Now, we determine the upper bound error of the proposed CPA. Consider the proposed CPA of the length t. If only t-th compressor of this CPA is approximate compressor, the MSE of CPA becomes $2^{t-1}\times\frac{4}{16}$, because the MSE of the proposed compressor is $\frac{4}{16}=0.25$. Therefore, an upper bound MSE for the proposed CPA is $\left(2^{t-1}\times\frac{4}{16}\right)+2^{t-2}-1$ when all compressors are approximate compressors, because the value of t-1 least significant bits of CPA is no more than $2^{t-2}-1$. To obtain this upper bound, we suppose that the value of t-1 least significant bits of CPA is always $2^{t-2}-1$ more than or less than its real value, while according to the truth table of our proposed compressor, the error probability of the proposed compressor is 0.25. Therefore, a tighter upper bound MSE for the proposed CPA is $\left(2^{t-1}\times\frac{4}{16}\right)+0.25\times\left(2^{t-2}-1\right)$.

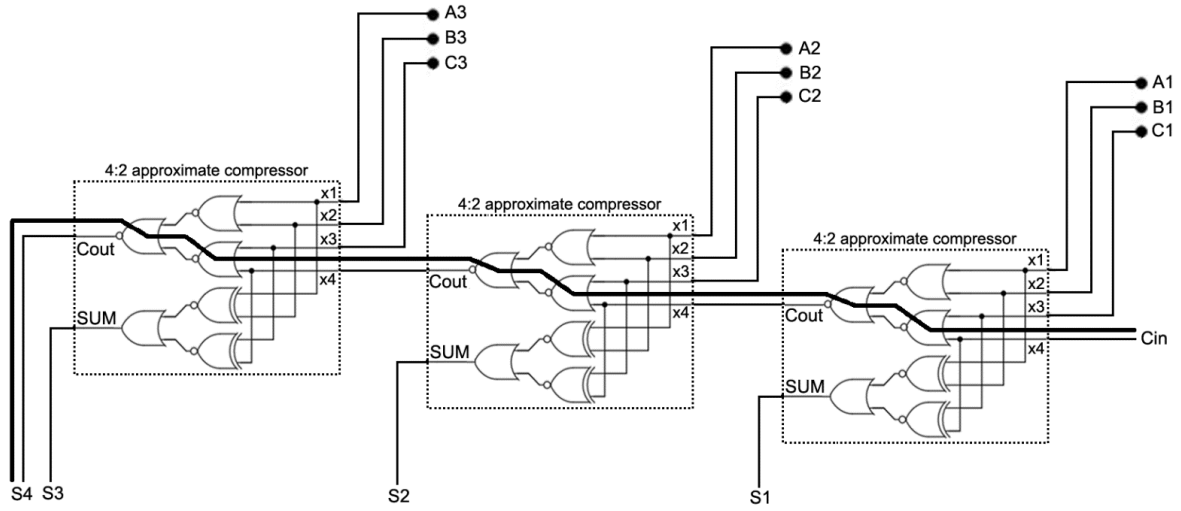


Fig. 8. A CPA constructed by using the 4:2 approximate compressor proposed in [15] to compute the sum of three 3-bit numbers. The thick line shows its longest path.

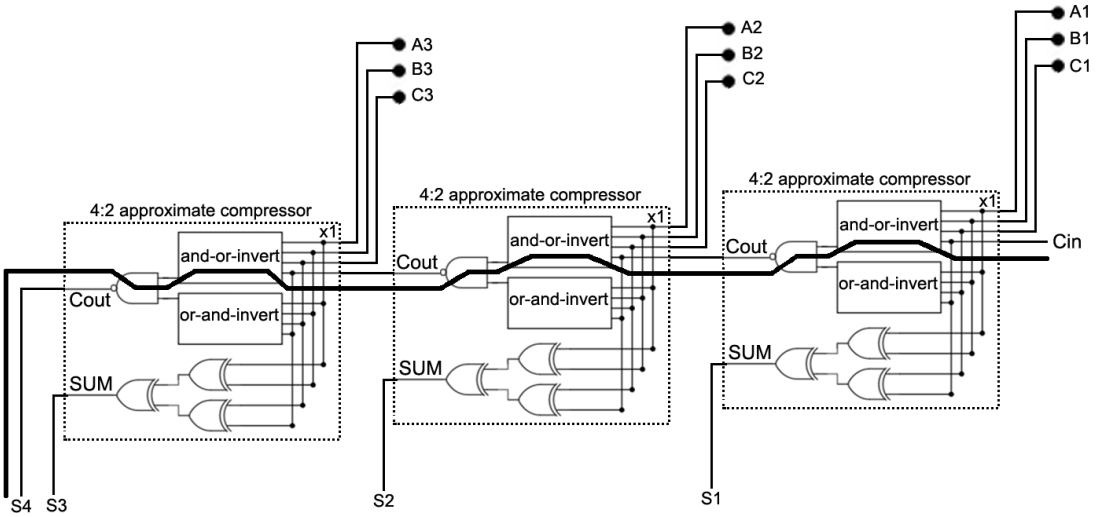


Fig. 9. A CPA constructed by using the 4:2 approximate compressor proposed in [16] to compute the sum of three 3-bit numbers. The thick line shows its longest path.

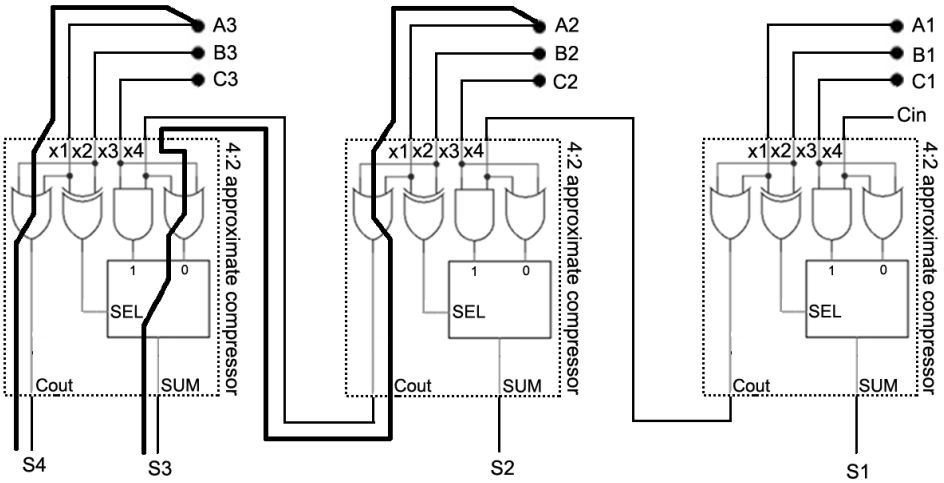


Fig. 10. A CPA constructed by using our proposed 4:2 approximate compressor. The thick lines show its overall longest path and the longest path to S4.

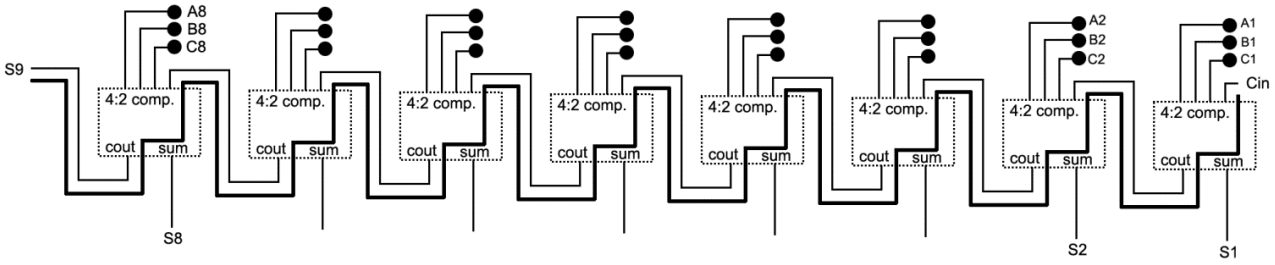


Fig. 11. A CPA constructed by using the 4:2 approximate compressor proposed in [15] or [16] to compute the sum of three 8-bit numbers. The thick line shows its longest path.

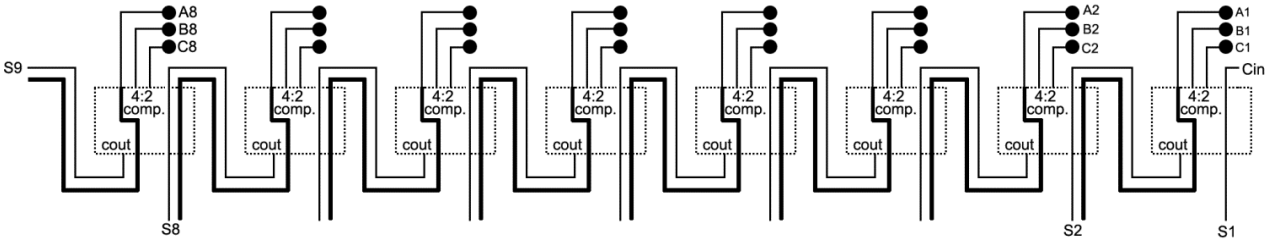


Fig. 12. A CPA constructed by using our proposed 4:2 approximate compressor to compute the sum of three 8-bit numbers. Each thick line can be the longest path.

3.3 Our First Proposed Approximate Multiplier

The approximate compressor shown in Fig. 8.a is used to construct our first approximate multiplier. Fig. 14 depicts this multiplier which is similar to the Dadda multiplier shown in Fig. 6. The number of stages of the proposed multiplier is one stage less than that of the former. At the last stage of the proposed multiplier, in order to obtain the summation of the three remaining rows of partial product, an especial CPA is used which was constructed by some half adders and some proposed approximate compressors. In each of the columns 2 and 15 of this CPA, a half adder is used, and in each of the columns 3 to 14, the first proposed compressor is used. Indeed, columns 3-14 of this CPA is our proposed CPA introduced in the previous sub-section which does not have the carry propagate delay problem. Meanwhile, one can use the approximate compressors only in the k least significant bits of the CPA to decrease its error. Fig. 15 shows this multiplier for k=8.

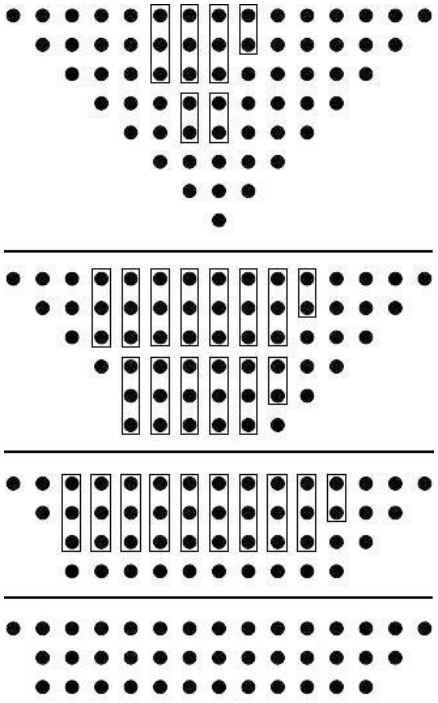


Fig. 13. The proposed multiplier. Each rectangle represents a half adder or a full adder. Sum of the last stage's columns are computed using a CPA which is constructed by some half-adders, full-adders, and proposed 4:2 compressors.

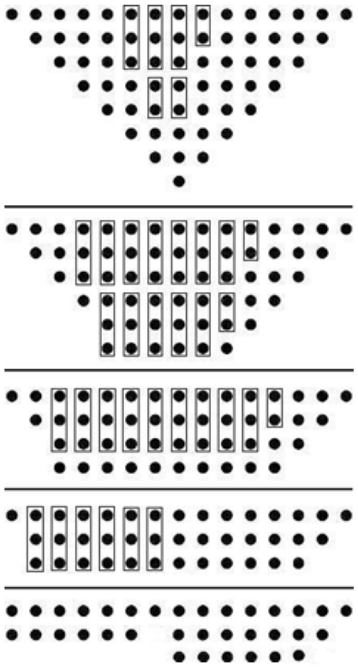


Fig. 14. The proposed multiplier for k=8. Each rectangle represents a half adder or a full adder. Sum of the last stage's columns are computed using a CPA which is constructed by some half-adders, full-adders, and proposed 4:2 compressors.

3.4 Our Second Proposed Approximate Multiplier

Consider the three proposed approximate compressors shown in Fig. 8. The only difference between these three compressors is the sequence of their inputs. For example, if the second input of compressor shown in Fig. 8.b is swapped with the third, the resulting circuit is the third compressor. Notice that changing the sequence of input of a precise compressor does not alter its output because a precise compressor computes the sum of its inputs, and its output is independent of the inputs sequence. But, this is not true for the approximate compressors.

As it can be seen in the truth Table 2, the output carry of each of the three proposed approximate compressors is independent of its fourth input. Therefore, if each of these three approximate compressors is used in the last stage of the proposed approximate multiplier to construct the CPA, the carry propagation delay problem is eliminated as long as the output carry of each of these compressors is connected to the fourth input of its next compressor. But, using each of the proposed compressors at each columns of CPA has different effect on the multiplier error and also affects the next column of CPA, because the output carry of each proposed compressor is connected to the next proposed compressor input. The proposed multiplier for an especial k was shown in the Fig. 15. To determine the type of approximate compressors for each columns 3 to k of CPA of multiplier to construct a multiplier with the least possible error, the following problem must be solved:

$$\min \quad MSE = \sum_{i=0}^{2^k-1}\sum_{j=0}^{2^k-1}(preciseMultiplication(i,j) - approximateMultiplication(i,j))^2$$

Subject to: types of compressors of i-th column of CPA of approximate multiplier $\in \{1,2,3\}$, $i = 3,4,...,k.$ (1)

To solve the problem (1), each time a different combination of the three proposed compressors was used at the columns of CPA of multiplier, and the MSE of the constructed multiplier was calculated. Then, the best multiplier with the least MSE and the types of compressors used in it was registered in Table 3.

Table 3. The best compressor type of each columns of CPA of multiplier for different value of k obtained by solving the problem (1).

k \ Column number	3	4	5	6	7	8	9	10	11	12	13	14
3	2											
4	3	1										
5	3	1	1									
6	1	1	1	1								
7	1	1	1	3	2							
8	2	2	1	1	2	2						
9	3	3	3	1	3	2	2					
10	3	3	2	3	3	3	2	2				
11	3	2	3	1	3	3	3	2	2			
12	3	1	2	1	3	1	3	3	2	2		
13	2	3	2	2	2	3	1	1	1	2	2	
14	1	1	3	2	3	1	1	3	1	3	2	1

If the k in the problem (1) is a big number, the number of combinations of compressors types for the columns 3 to k of CPA of an 8-bit multiplier becomes a big number. For example, the number of combinations of compressors types for k=14 is equal to $3^{(14-2)}$=531,441. The number of combinations of compressors types for a bigger k in a 16-bit multiplier increases exponentially and it isn't possible to compare all of combinations of compressors types in a reasonable time. To obtain the best types of compressors, an integer mathematical programming (Problem 1) must be solved. One can use the genetic algorithm to solve it instead of comparing all possible combinations of compressors types. In this paper, the basic genetic algorithm was used to solve the problem (1) only for k=13 and k=14 and the result was registered in Table 3. The length of chromosomes in the genetic algorithm is equal to the number of compressors in the CPA of multiplier. The i-th gene of chromosome is a number between 1 to 3 which denotes compressor type. The fitness function of the genetic algorithm for solving the problem (1) is the negative of multiplier MSE for a combination of compressors types in the CPA of multiplier.

4. Simulations

In this section the proposed multipliers are compared with ten other multipliers: truncated1 [7], truncated2 [8], truncated3 [9], LOA [5], Momeni [15], Ma [16], Liu [13,14], Kulkani [11], Accurate3:2 (the Dadda multiplier constructed by 3:2 precise compressors), and Accurate4:2 (the Dadda multiplier constructed by 4:2 precise compressors).

Table 4 shows the MSE, the delay, the number of transistors, and the product of delay and the number of transistors (PDT) of each multiplier. This table also shows the percentages of delay improvement, the percentages of the number of transistors improvement, the percentages of PDT improvement for each multiplier with respect to the multiplier Accurate3:2. According to this table, for each k, the MSE of the second approximate proposed multiplier is less than that of the first approximate proposed multiplier. To compute the MSE, each multiplier was simulated in MATLAB, and for computing the delay and the number of transistors needed to construct each multiplier, each multiplier was simulated in HSPICE at 16 nm CMOS technology based on the best gates of [21] (See Fig. 16). Working voltage and the temperature were supposed to be 3.3v and 27 degrees centigrade.

According to Table 4, when the multiplication operands are supposed to be selected from a uniform distribution,

- For each k, the MSE of the second approximate proposed multiplier is less than that of the first approximate proposed multiplier.

For k=3,4,…,12, the error and the delay of the second proposed method are less than those of the traditional 4:2 approximate compressor-based multipliers. Table 5(a) (b)

- Table 1 shows the least error and delay improvement of the second proposed method with respect to the traditional 4:2 approximate compressor-based multipliers.
- The second proposed multiplier has less delay than that of the other multipliers with the same MSE level, except the LOA for some k, or the second proposed multiplier has less MSE than that of the other multipliers with the same delay level, except the LOA for some k. The second proposed multiplier for k=8,9,…,14 has also less delay than that of the LOA with the same MSE level.

Table 4. Comparison of different multipliers.

Multiplier	Log(MSE)	#Transistors	#Transistors impr. with respect to Acc3:2	Delay (ns)	Delay improvement with respect to Accurate3:2	PDT	PDT impr. with respect to Acc3:2
truncated1 [7] ,P=11,K=4	2.49	1542	20.26	1.67	18.84	2578.68	35.29
truncated2 [8] ,P=11,K=4	2.03	1700	12.09	1.75	14.99	2978.06	25.27
truncated3 [9] ,P=8,k=4	4.11	1382	28.54	1.53	25.51	2121.37	46.77
LOA [5] , k=3	0.17	1900	1.75	1.91	7.19	3633.75	8.82
LOA [5] , k=4	0.54	1874	3.10	1.84	10.61	3451.72	13.39
LOA [5] , k=5	1.40	1844	4.65	1.76	14.43	3251.52	18.41
LOA [5] , k=6	2.01	1822	5.79	1.69	17.85	3084.09	22.61
LOA [5] , k=7	2.70	1796	7.13	1.61	21.66	2899.10	27.25
LOA [5] , k=8	3.46	1770	8.47	1.54	25.09	2732.17	31.44
LOA [5] , k=9	4.05	1744	9.82	1.44	30.08	2512.75	36.95
LOA [5] , k=10	4.66	1718	11.16	1.32	35.65	2278.06	42.83
LOA [5] , k=11	5.25	1692	12.51	124	39.56	2107.04	47.13
LOA [5] , k=12	5.80	1666	13.85	1.14	44.55	1903.57	52.23
LOA [5] , k=13	6.32	1640	15.20	0.99	51.79	1629.17	59.12
LOA [5] , k=14	6.85	1614	16.54	0.84	59.02	1362.70	65.80
LOA [5] , k=15	7.38	1588	17.89	0.69	66.26	1103.81	72.30
Momeni [15]	7.21	1550	19.85	2.06	0	3194.08	19.85
Ma [16]	6.07	1686	12.82	2.06	0	3474.34	12.82
Liu [13,14]	6.12	2024	4.65-	1.12	45.46	2274.77	42.92
Kulkani [11]	6.81	1512	21.82	2.32	12.77-	3513.88	11.83
Proposed1,k=3	0.69	1908	1.34	1.91	7.15	3650.57	8.40
Proposed1, k=4	1.19	1882	2.68	1.79	13.03	3372.73	15.37
Proposed1, k=5	1.72	1856	4.03	1.73	15.62	3227.02	19.02
Proposed1, k=6	2.31	1830	5.37	1.65	19.59	3032.12	23.91
Proposed1, k=7	3.03	1804	6.72	1.60	22.20	2892.17	27.43
Proposed1, k=8	3.62	1778	8.06	1.51	26.69	2685.66	32.61
Proposed1, k=9	4.19	1752	9.41	1.37	33.10	2415.30	39.39
Proposed1, k=10	4.83	1726	10.75	1.31	36.18	2269.69	43.04
Proposed1, k=11	5.42	1700	12.09	1.21	41.17	2060.91	48.28
Proposed1, k=12	6.03	1674	13.44	1.06	48.41	1779.62	55.34
Proposed1, k=13	6.56	1648	14.78	0.91	55.64	1506.27	62.20
Proposed1, k=14	7.18	1622	16.13	0.65	68.14	1064.68	73.28
Proposed2, k=3	0.65	1908	1.34	1.91	7.15	3650.57	8.40
Proposed2, k=4	1.16	1882	2.68	1.79	13.03	3372.73	15.3727
Proposed2, k=5	1.71	1856	4.03	1.73	15.62	3227.02	19.02
Proposed2, k=6	2.31	1830	5.37	1.65	19.59	3032.12	23.91
Proposed2, k=7	2.95	1804	6.72	1.60	22.20	2892.17	27.43
Proposed2, k=8	3.52	1778	8.06	1.51	26.69	2685.66	32.61
Proposed2, k=9	4.09	1752	9.41	1.37	33.10	2415.30	39.39
Proposed2,k=10	4.69	1726	10.75	1.31	36.18	2269.69	43.04
Proposed2, k=11	5.28	1700	12.09	1.21	41.17	2060.91	48.28
Proposed2, k=12	5.89	1674	13.44	1.06	48.41	1779.62	55.34
Proposed2, k=13	6.48	1648	14.78	0.91	55.64	1506.27	62.20
Proposed2, k=14	7.04	1622	16.13	0.65	68.14	1064.68	73.28
Accurate3:2	-	1934	0	2.06	0	3985.39	0
Accurate4:2	-	1950	0.82-	2.36	14.70-	4609.21	15.65-

Table 5. The least error and delay improvement of the second proposed method with respect to the traditional 4:2 approximate compressor-based multipliers (%).

	The least error improvement (%)	The least delay improvement (%)
K=3	89.25	7.15
K=4	80.89	13.03
K=5	71.83	15.63
K=6	61.95	19.60
K=7	51.35	22.20
K=8	42.00	26.70
K=9	32.52	33.10
K=10	22.65	36.19
K=11	13.06	41.17
K=12	2.96	48.41

Fig. 17 shows the delay of each multiplier versus its MSE. There is a trade off between the delay and the MSE of a multiplier. The less the delay of a multiplier, the more its MSE. But, according to this Fig., the second proposed multiplier has less delay than that of the other multipliers with the same MSE level, except the LOA for some k, or the second proposed multiplier has less MSE than that of the other multipliers with the same delay level, except the LOA for some k. The second proposed multiplier for k=8,9,…,14 has also less delay than that of the LOA with the same MSE level.

Fig. 18 shows the PDT of each multiplier versus its MSE. According to this Fig., the second proposed multiplier has less PDT than that of the other multipliers with the same MSE level, except LOA for some k and the truncated multipliers, or the second proposed multiplier has less MSE than that of the other multipliers with the same PDT level, except LOA for some k and the truncated multipliers. The second proposed multiplier for k=8,9,11,…,14 has also less PDT than that of LOA with the same MSE level.

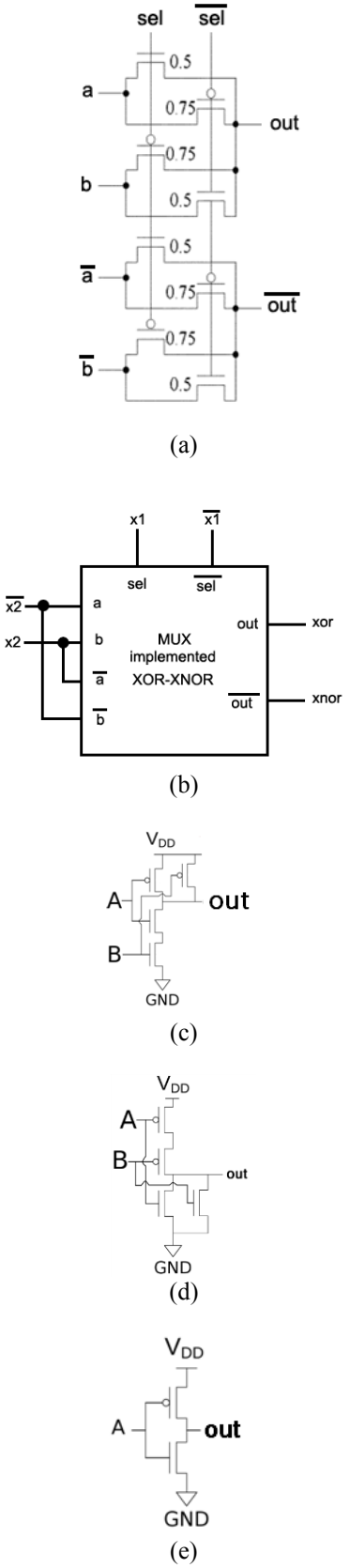


Fig. 15. The gates used for simulation: (a) XOR-XNOR [17], (b) MUX [17], (c) NOR, (d) NAND, (e) NOT.

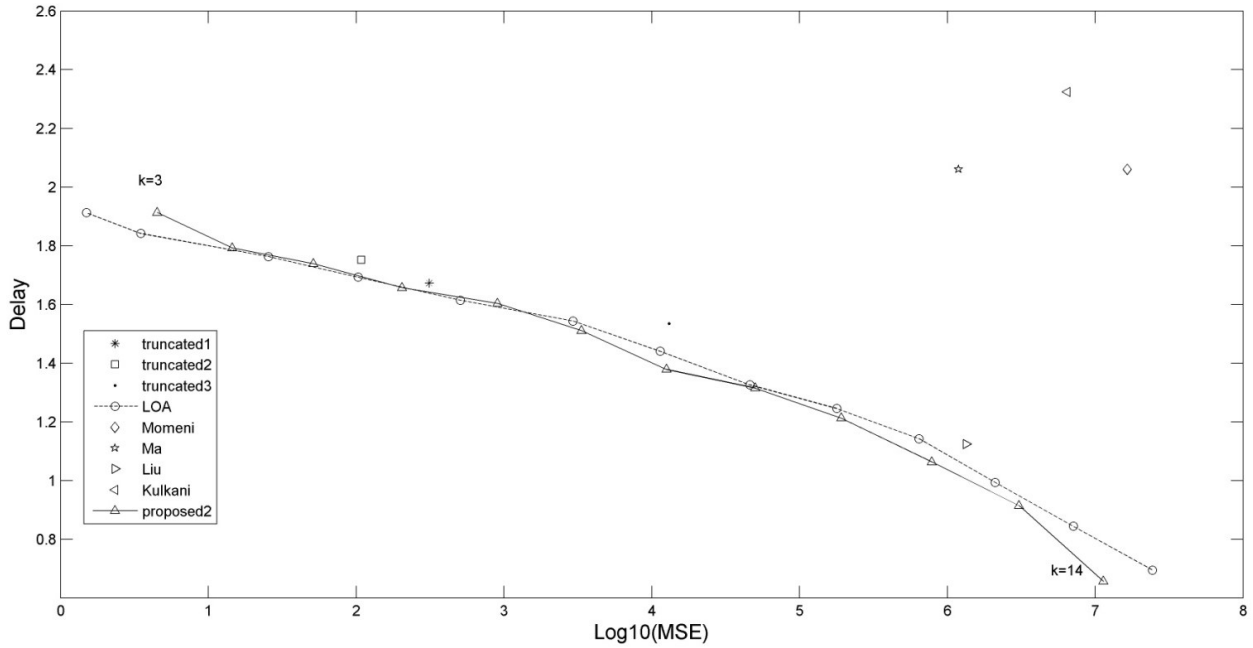


Fig. 16. The delay of each multiplier versus its MSE.

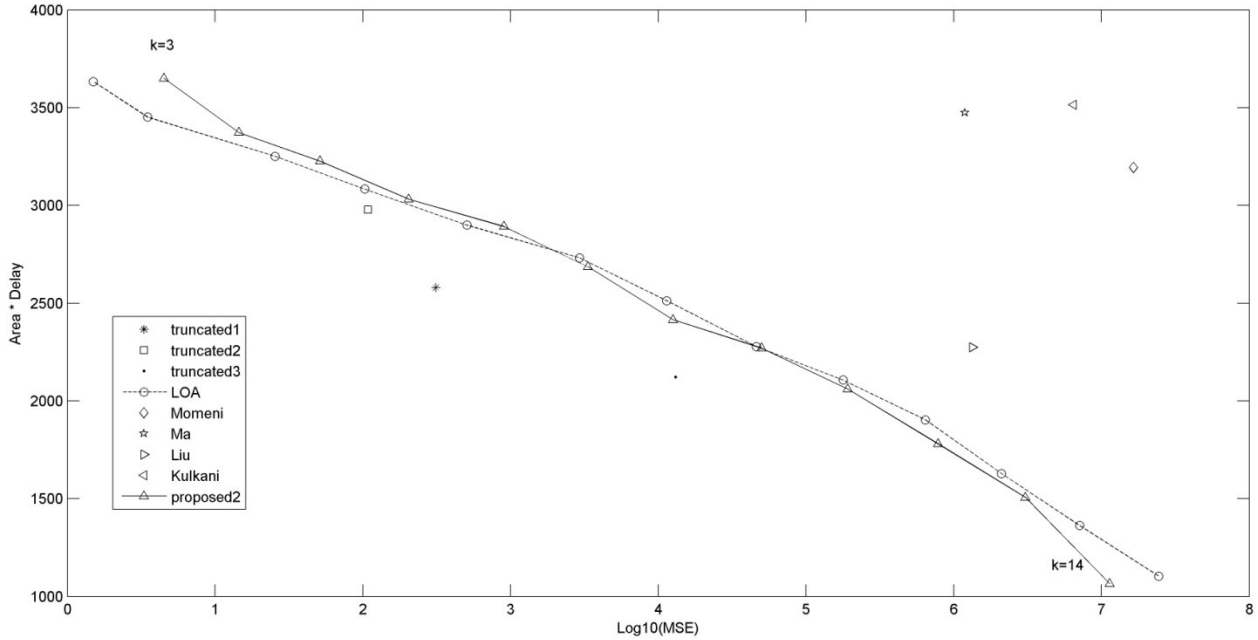


Fig. 17. The PDT of each multiplier versus its MSE.

5. Applications

5.1 Image Blending

Fig. 19 shows an application of multiplication in image processing. The right blended 8-bit image in this Fig. was obtained by peer-to-peer pixel multiplication of the two left 8-bit images using a precise multiplier and truncation of the 16-bit result to 8-bit. In this sub-section, the approximate multipliers are used to blend the two right images. Table 6 shows the PSNR of the blended images for each multiplier. The PSNR formulation is as follows:

$$PSNR = 10\log_{10}\left(\frac{\left(Peak\ \ Signal\ \ Value\right)^2}{MSE}\right) \quad (2)$$

The blended images obtained by different approximate multipliers were shown in Fig. 21. As it can be seen, the most of the blended images obtained by approximate multipliers do not differ significantly from that of obtained by precise multiplier.

(a) (b) (c)

Fig. 18. Blended images using precise multiplier (a) the first image, (b) the second image, (c) Blended image.

Fig. 20 shows the delay of each multiplier versus the negative of its PSNR for the image blending. There is a trade off between the delay and the -PSNR of a multiplier. The less the delay of a multiplier, the more its -PSNR. But, according to this Fig., the second proposed multiplier has less delay than that of the other multipliers with the same PSNR level, except LOA for some k, or the second proposed multiplier has less -PSNR than that of the other multipliers with the same delay level, except LOA for some k. The second proposed multiplier for k=8,9,…,14 has also less delay than that of LOA with the same PSNR level.

Table 6. PSNR of blended images for different approximate multipliers.

Multiplier	PSNR	Multiplier	PSNR
truncated1 [7], p=11,k=4	71.36	Proposed1, k=5	79.02
truncated2 [8], p=11,k=4	75.81	Proposed1, k=6	72.93
truncated3 [9], p=8,k=4	55.35	Proposed1, k=7	65.78
LOA [5], k=3	94.44	Proposed1, k=8	59.93
LOA [5], k=4	90.68	Proposed1, k=9	54.32
LOA [5], k=5	82.18	Proposed1, k=10	47.80
LOA [5], k=6	76.16	Proposed1, k=11	41.97
LOA [5], k=7	69.59	Proposed1, k=12	35.23
LOA [5], k=8	61.52	Proposed1, k=13	29.21
LOA [5], k=9	55.44	Proposed1, k=14	24.26
LOA [5], k=10	49.34	Proposed2, k=3	89.70
LOA [5], k=11	43.10	Proposed2,k=4	83.73
LOA [5], k=12	37.12	Proposed2,k=5	77.70
LOA [5], k=13	33.06	Proposed2,k=6	72.04
LOA [5], k=14	24.29	Proposed2,k=7	66.34
LOA [5], k=15	19.64	Proposed2,k=8	61.20
Momeni [15]	27.86	Proposed2,k=9	55.44
Ma [16]	42.08	Proposed2,k=10	49.44
Liu [13,14]	33.64	Proposed2,k=11	43.78
Kulkani [11]	29.36	Proposed2,k=12	37.20
Proposed1, k=3	89.23	Proposed2,k=13	31.68
Proposed1, k=4	84.30	Proposed2,k=14	28.63

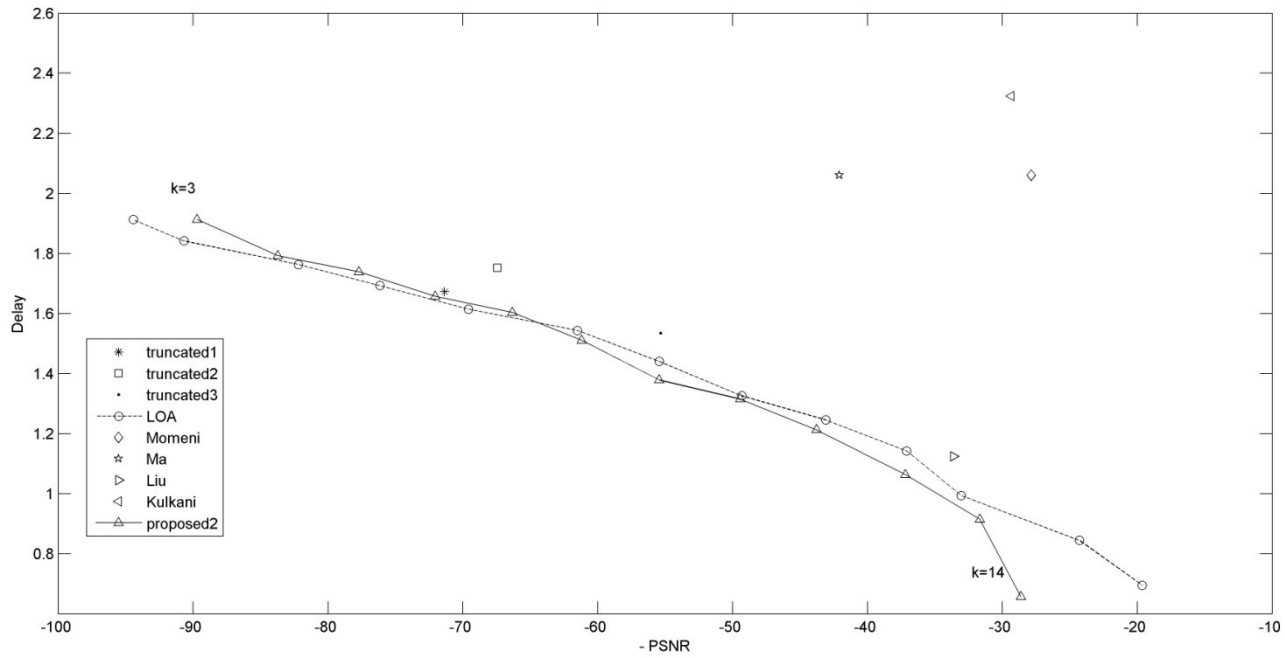
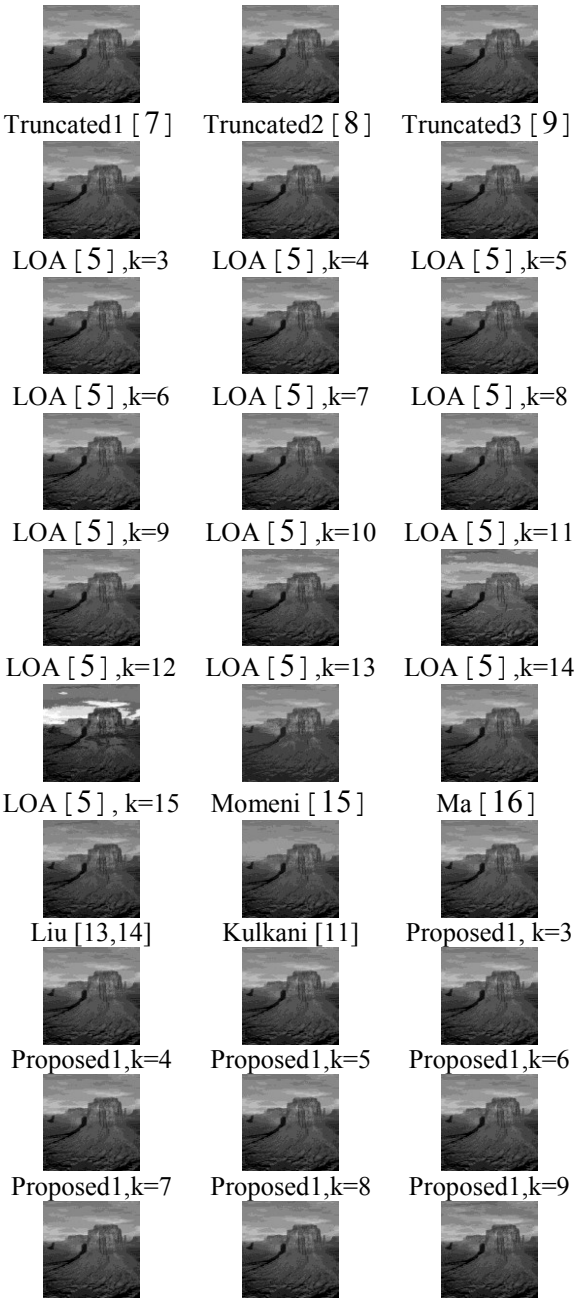


Fig. 19. The delay of each multiplier versus the negative of its PSNR for the image blending.



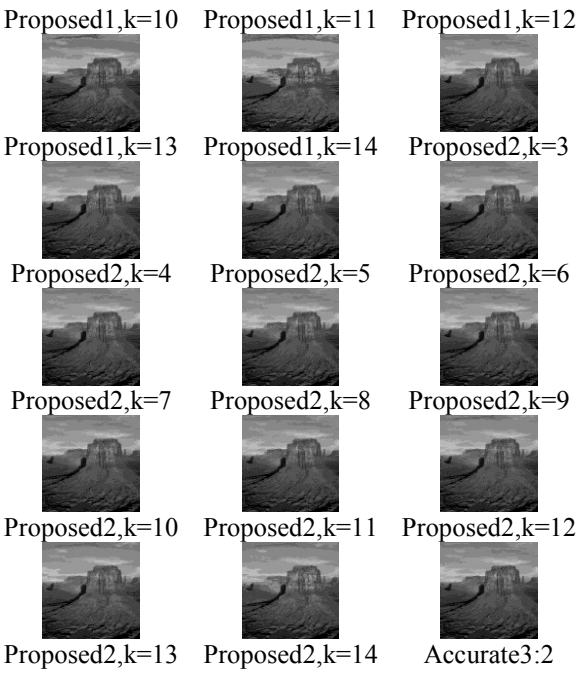


Fig. 20. The blended images obtained by different approximate multipliers.

Fig. 22 shows the PDT of each multiplier versus the negative of its PSNR for the image blending. According to this Fig., the second proposed multiplier has less PDT than that of the other multipliers with the same PSNR level, except LOA for some k and truncated1 and truncated2, or the second proposed multiplier has less -PSNR than that of the other multipliers with the same PDT level, except LOA for some k and truncated1 and truncated2. The second proposed multiplier for k=8,9,…,14 has also less PDT than that of LOA with the same PSNR level.

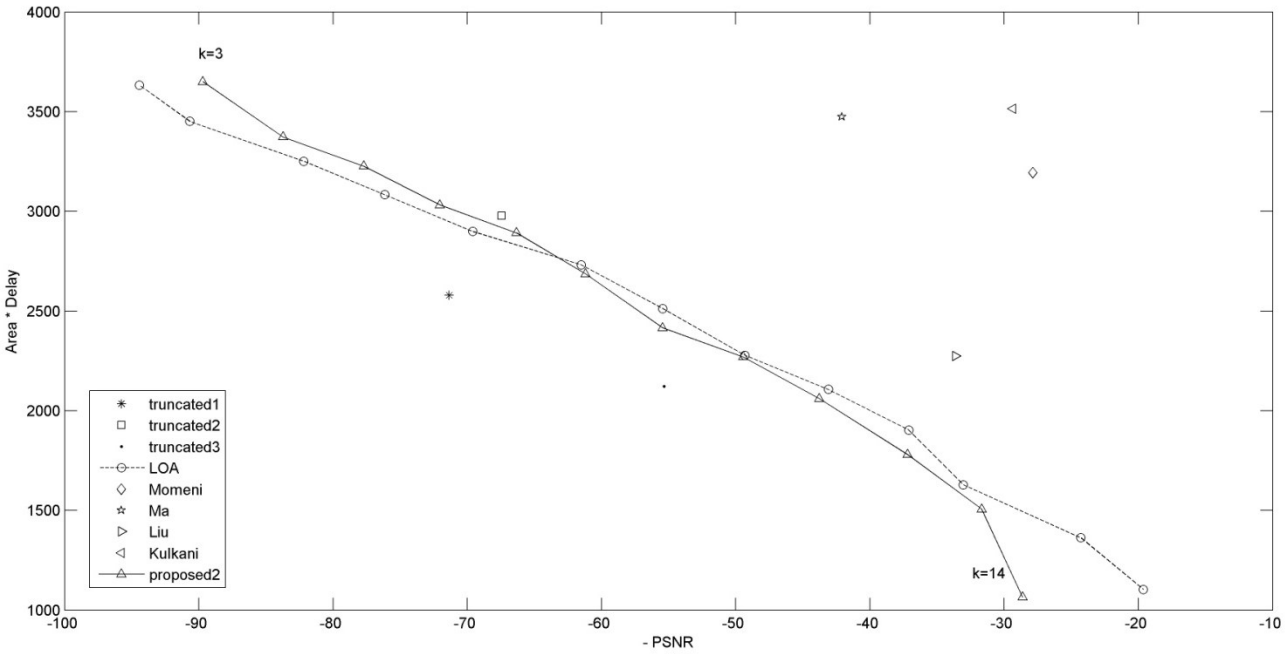


Fig. 21. The PDT of each multiplier versus the negative of its PSNR for the image blending.

5.2 Image Compression

In this section, the proposed multiplier is used in the JPEG algorithm for lossy image compression. The block diagram of JPEG algorithm was shown in Fig. 23. In this algorithm, first, the image is encoded to the color coding YUV. Then, their 8×8 blocks of each channel Y, U and V shown by f is transformed from the time domain into the frequency domain using Discrete Cosine Transform (DCT) [24] to obtain a new 8×8 blocks named F. the DCT formulation is as follows:

$$F(u,v)=\frac{C(u)C(v)}{4}\times \sum_{i=0}^{7}\sum_{j=0}^{7}\cos\left(\frac{(2i+1)u\pi}{16}\right)\cos\left(\frac{(2j+1)u\pi}{16}\right)f(i,j) \quad (3)$$

where

$$c(u)=\begin{cases}\frac{\sqrt{2}}{2} & if\ u=0,\\ 1 & otherwise.\end{cases}$$

In this paper, integer value version of this transformation [25] is used which is as follows:

$$F(u,v)=\frac{1}{1024}\sum_{i=0}^{7}\sum_{j=0}^{7}z(i,j)f(i,j), \quad (4)$$

where $z(i,j)=round(1024\times\frac{C(u)C(v)}{4}\times \cos\left(\frac{(2i+1)u\pi}{16}\right)\cos\left(\frac{(2j+1)u\pi}{16}\right))$ is an 8-bit integer value.

z(i,j) for each i and j is calculated one time, and is stored in a table to be used in the compression phase of each images [26]. Therefore, since f(i,j) is considered to be an 8-bit value, F(u,v) can be computed using some 8-bit multiplications and 8-bit additions, and then by dividing the result to 1024 or right shifting it 10 times. In this paper, the additions are calculated using a precise adder and only the multiplications were calculated using an approximate multiplier. Each times, each of the six standard images shown in Fig. 24 was compressed using an approximate multiplier and then decompressed. Then, the similarity amount of the original image and the decompressed image was calculated using the PSNR, and was registered then in Table 7. In this table, the mean of the PSNR of the six decompressed images for each approximate multiplier, and also the percentages of improvement of these mean with respect to the mean PSNR of decompressed images obtained by Accurate3:2 were registered. One series of the decompressed images each one obtained by using a different multiplier was shown in Fig. 25. As it can be seen, the original image and the decompressed images often do not differ significantly.

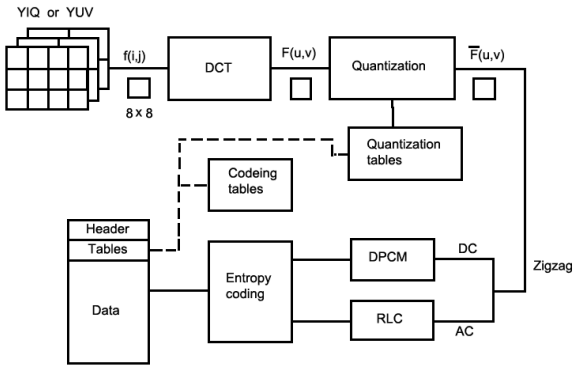


Fig. 22. Block diagram of JPEG algorithm [24].

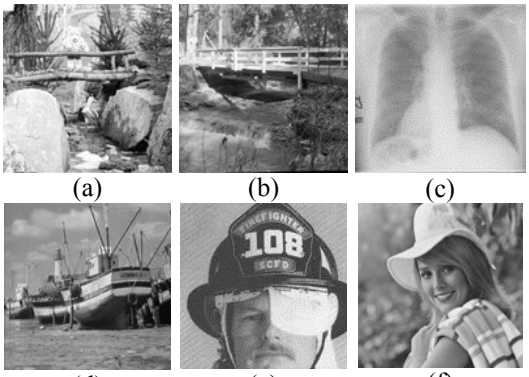
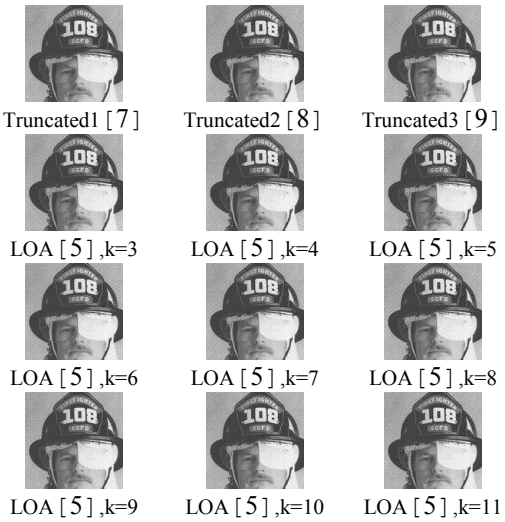


Fig. 23. Six benchmark images.



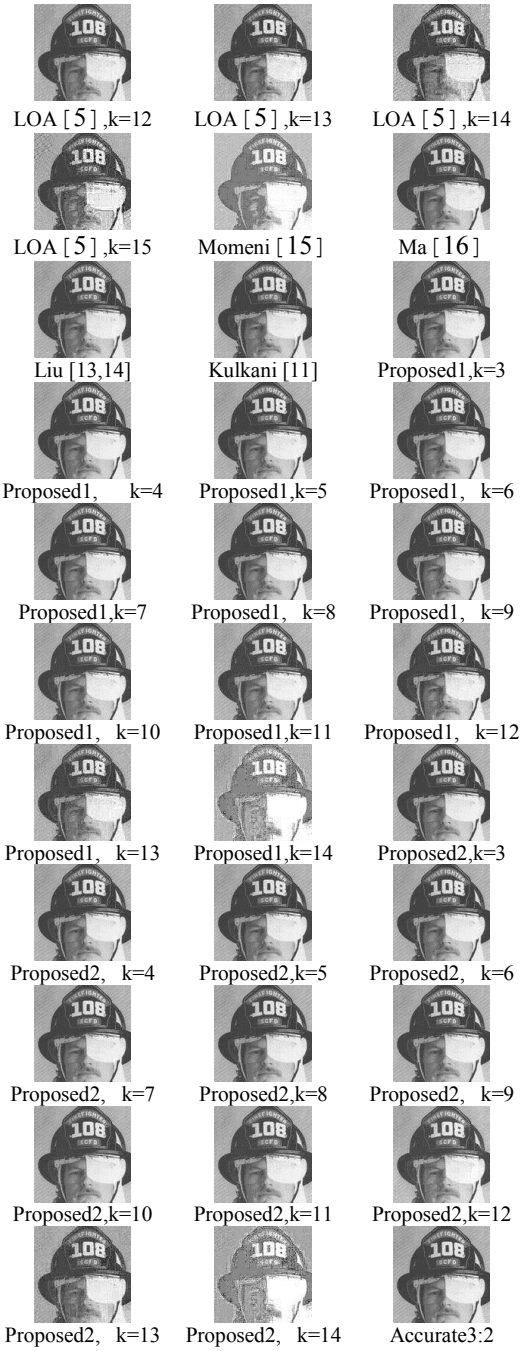


Fig. 24. Decompressed images for different multipliers.

Table 7. PSNR or similarity amount of the original images and the corresponding decompressed images for different multipliers.

Multiplier	PSNR of each image type						Mean of PSNR	PSNR mean improvement with respect to Accurate3:2
	a	b	c	d	e	f		
truncated1 [7], P=11,K=4	65.09	65.26	65.64	65.36	65.25	65.33	65.30	14.4777
truncated2 [8], P=11,K=4	72.23	73.22	76.16	73.74	73.01	73.58	73.55	3.6726
truncated3 [9], P=8,k=4	54.46	55.76	49.91	49.88	49.96	55.36	49.87	34.68
LOA [5], k=3	74.02	75.48	83.25	76.41	75.18	76.08	76.26	0.12
LOA [5], k=4	73.94	75.41	82.90	76.32	75.10	76.02	76.17	0.24
LOA [5], k=5	73.41	74.74	79.91	75.42	74.49	75.16	75.32	1.35
LOA [5], k=6	71.87	72.56	75.42	73.06	72.89	72.99	73.15	4.20
LOA [5], k=7	67.63	68.15	69.55	68.74	68.15	68.22	68.85	9.83
LOA [5], k=8	61.44	61.70	62.16	62.19	61.48	61.81	62.64	17.96
LOA [5], k=9	55.73	55.68	56.18	55.71	56.04	55.76	56.51	25.98
LOA [5], k=10	49.59	49.71	49.88	49.76	50.00	49.52	50.69	33.61
LOA [5], k=11	43.73	44.48	44.10	44.24	44.51	44.20	45.20	40.79
LOA [5], k=12	38.98	39.36	39.73	39.75	39.55	39.18	39.83	47.83
LOA [5], k=13	34.64	36.15	35.11	36.18	35.78	35.56	36.38	52.34
LOA [5], k=14	29.87	30.55	31.24	31.81	31.26	30.43	31.51	58.73
LOA [5], k=15	26.05	28.46	28.52	30.27	28.43	27.97	29.42	61.46
Momeni [15]	23.44	23.00	22.37	22.37	22.74	23.06	22.95	69.94
Ma [16]	42.99	48.06	46.92	51.98	46.71	48.67	49.22	35.54
Liu [13,14]	41.86	42.64	43.15	43.05	42.92	42.63	43.00	43.67
Kulkani [11]	39.28	42.15	44.44	43.56	42.07	40.78	42.90	43.81
Proposed1, k=3	73.96	75.40	82.89	76.32	75.13	75.99	76.17	0.24
Proposed1, k=4	73.72	75.09	81.35	75.91	74.80	75.63	75.70	0.86
Proposed1, k=5	72.77	73.85	77.67	74.42	73.60	74.33	74.26	2.74
Proposed1, k=6	70.32	70.91	73.04	71.21	71.24	71.33	71.43	6.44
Proposed1, k=7	66.04	66.35	67.07	66.32	66.86	66.33	66.55	12.84
Proposed1, k=8	57.17	57.21	57.43	57.36	57.23	57.27	57.34	24.90
Proposed1, k=9	51.42	51.50	51.64	51.49	51.57	51.51	51.70	32.28
Proposed1, k=10	45.33	45.49	45.63	45.59	45.44	45.53	45.70	40.15
Proposed1, k=11	39.50	39.50	39.82	39.66	39.89	39.56	39.63	48.10
Proposed1, k=12	37.35	37.49	38.21	37.88	38.12	37.32	37.73	50.58
Proposed1, k=13	32.59	33.03	34.15	33.66	34.18	33.15	33.79	55.73
Proposed1, k=14	20.58	20.75	21.40	25.10	20.20	21.43	21.91	71.30
Proposed2, k=3	73.97	75.44	83.09	76.35	75.16	76.06	76.21	0.18
Proposed2, k=4	73.76	75.18	81.52	76.03	74.92	75.71	75.82	0.70
Proposed2, k=5	73.11	74.28	78.56	74.91	74.02	74.67	74.69	2.18
Proposed2, k=6	71.32	72.08	73.86	72.44	71.98	72.23	72.31	5.29
Proposed2, k=7	67.90	68.02	68.86	68.50	68.23	68.08	68.25	10.61
Proposed2, k=8	58.06	58.15	58.32	58.10	58.21	58.06	58.12	23.88
Proposed2, k=9	52.25	52.32	52.49	52.43	52.42	52.32	52.49	31.25
Proposed2, k=10	46.17	46.31	46.54	46.41	46.24	46.38	46.48	39.12
Proposed2, k=11	40.21	40.14	40.59	40.35	40.53	40.27	40.24	47.29
Proposed2, k=12	34.22	34.15	34.54	33.74	34.73	34.21	34.23	55.17
Proposed2, k=13	27.69	28.24	29.23	28.57	27.44	28.34	28.25	63.00
Proposed2, k=14	21.96	21.36	21.43	25.10	20.93	22.52	22.21	70.90
Accurate3:2	74.02	75.54	83.65	76.49	75.24	76.16	76.36	0

Fig. 26 shows the delay of each multiplier versus the negative of its PSNR for the image compression. There is a trade off between the delay and the -PSNR of a multiplier. The less the delay of a multiplier, the more its -PSNR. But, according to this Fig., the second proposed multiplier has less delay than that of the other multipliers with the same PSNR level, except LOA for some k, or the second proposed multiplier has less -PSNR than that of the other multipliers with the same delay level, except LOA for some k. The second proposed multiplier for k=4,5,6,7 has also less delay than that of LOA with the same PSNR level. The obtained result is different from the results obtained in section IV. The reason is that in

section IV the MSE was calculated based on the all of 8-bit multiplier truth table states and thus multiplication operands were supposed to be from a uniform distribution, while in this section the PSNR (which related directly to MSE) was computed based on some of multiplier truth table states. These states depend on multiplication operands, pixels values, histogram, or data distribution of each image. Therefore, one can obtain different result for some other images probably. If it is supposed that the multiplication operands in the JPEG compression have a uniform distribution, then the result will be the same as what were shown in Table 4 not what where shown in Table 7.

Fig. 27 shows the PDT of each multiplier versus the negative of its PSNR. According to this Fig., the second proposed multiplier has less PDT than that of the other multipliers with the same PSNR level, except LOA for some k and the truncated multipliers, or the second proposed multiplier has less -PSNR than that of the other multipliers with the same PDT level, except LOA for some k and the truncated multipliers. The second proposed multiplier for k=4,6,7 has also less PDT than that of LOA with the same PSNR level.

6. Conclusion

In this paper, a novel 8-bit approximate multiplier based on three novel 4:2 approximate compressors was proposed which its delay and error was less than those of the multipliers constructed by traditional 4:2 approximate compressors, and its delay is also less than that of an 8-bit multiplier constructed by using 3:2 precise compressors. To do so, each novel compressor was designed such that its output carry was independent of some of its inputs. One of these inputs was connected to the output carry of its previous compressor in the CPA of multiplier. Therefore, the problem of carry propagation delay of multiplier's CPA was eliminated and a fast multiplier was constructed. This was the first proposed multiplier. To obtain the most accurate multiplier, the best compressor of the three proposed compressors for each multiplier's columns is determined using the genetic algorithm. The constructed multiplier called the second proposed multiplier. Meanwhile, for more error reduction, the approximate compressors were used only at the k least significant columns of multiplier. The proposed multipliers were used for image blending and image compression.

According to the simulations results (Table 4), when the multiplication operands are supposed to be selected from a uniform distribution, for each k, the MSE of the second approximate proposed multiplier is less than that of the first approximate proposed multiplier. For k=3,4,…,12, the error and the delay of the second proposed method are less than those of the traditional 4:2 approximate compressor-based multipliers (See Table 5 for detail). Meanwhile, the second proposed multiplier has less delay than that of the other multipliers with the same MSE level, except the LOA for some k, or the second proposed multiplier has less MSE than that of the other multipliers with the same delay level, except the LOA for some k. The second proposed multiplier for k=8,9,…,14 has also less delay than that of the LOA with the same MSE level.

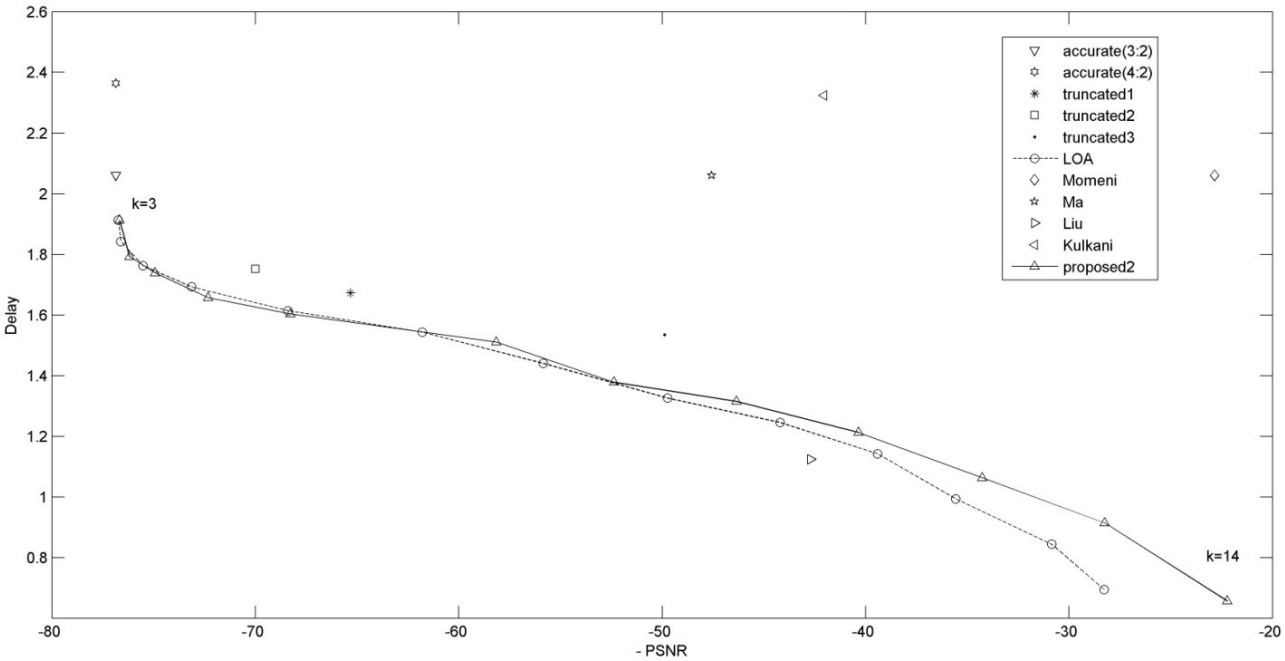


Fig. 25. The delay of each multiplier versus the negative of its PSNR for the image compression.

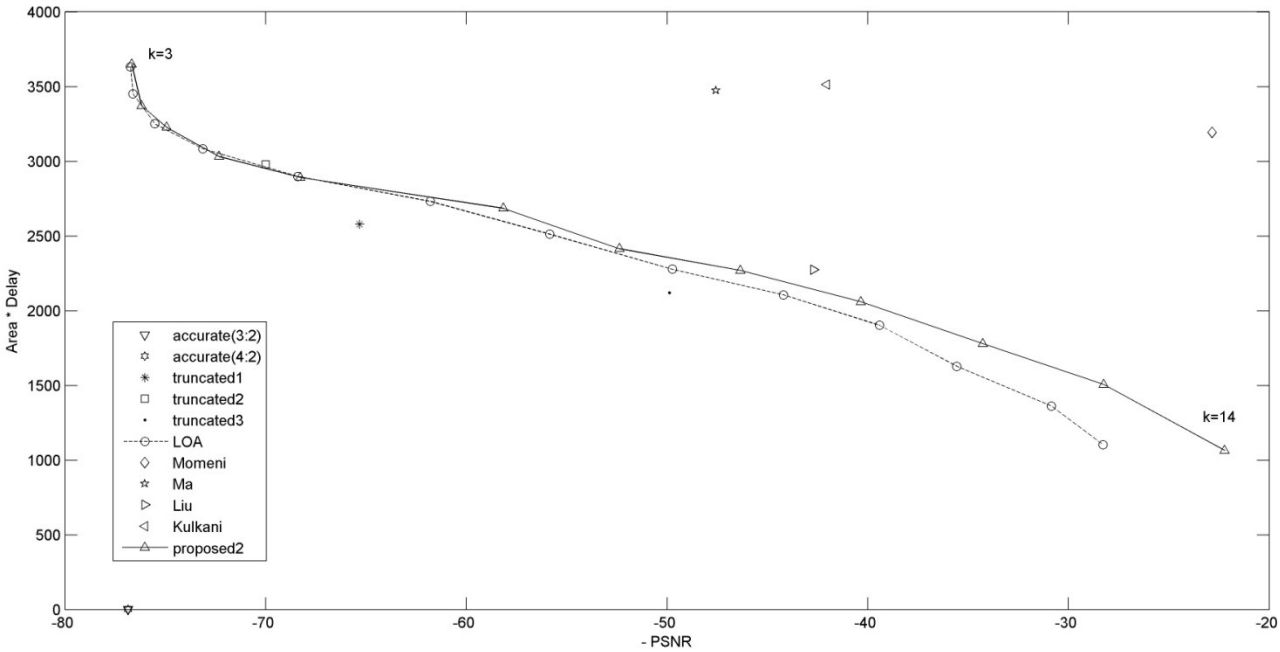


Fig. 26. The PDT of each multiplier versus the negative of its PSNR for the images compression.

References

[1] J. Liang, J. Han, and F. Lombardi, "New metrics for the reliability of approximate and probabilistic adders," *IEEE Transactions on Computers,* vol. 62, pp. 1760-1771, 2013.

[2] V. Gupta, D. Mohapatra, S. P. Park, A. Raghunathan, and K. Roy, "IMPACT: imprecise adders for low-power approximate computing," in *Proceedings of the 17th IEEE/ACM international symposium on Low-power electronics and design*, 2011, pp. 409-414.

[3] S. Cheemalavagu, P. Korkmaz, K. V. Palem, B. E. Akgul, and L. N. Chakrapani, "A probabilistic CMOS switch and its realization by exploiting noise," in *IFIP International Conference on VLSI*, 2005, pp. 535-541.

[4] A. B. Kahng and S. Kang, "Accuracy-configurable adder for approximate arithmetic designs," in *Proceedings of the 49th Annual Design Automation Conference*, 2012, pp. 820-825.

[5] H. R. Mahdiani, A. Ahmadi, S. M. Fakhraie, and C. Lucas, "Bio-Inspired Imprecise Computational Blocks for Efficient VLSI Implementation of Soft-Computing Applications," *IEEE TRANSACTIONS ON CIRCUITS AND SYSTEMS,* vol. 57, pp. 850-862, 2011.

[6] K.-J. Cho, K.-C. Lee, J.-G. Chung, and K. K. Parhi, "Design of low-error fixed-width modified booth multiplier," *IEEE Transactions on Very Large Scale Integration (VLSI) Systems,* vol. 12, pp. 522-531, 2004.

[7] M. J. Schulte and E. E. Swartzlander, "Truncated multiplication with correction constant [for DSP]," in *VLSI Signal Processing, VI, 1993., [Workshop on]*, 1993, pp. 388-396.

[8] E. J. King and E. E. Swartzlander, "Data-dependent truncation scheme for parallel multipliers," in *Signals, Systems & Computers, 1997. Conference Record of the Thirty-First Asilomar Conference on*, 1997, pp. 1178-1182 vol.2.

[9] H.-J. Ko and S.-F. Hsiao, "Design and Application of Faithfully Rounded and Truncated Multipliers With Combined Deletion, Reduction, Truncation, and Rounding," *IEEE TRANSACTIONS ON CIRCUITS AND SYSTEMS,* vol. 58, pp. 304-308, 2011.

[10] Harish Rao. B and R. K. V, "IMPLEMENTATION OF 8X8 DADDA MULTIPLIER USING APPROXIMATE COMPRESSION FOR IMAGE ENHANCEMENT," *International Journal of Advances in Engineering Research,* vol. 10, pp. 70-81, 015.

[11] P. Kulkarni, P. Gupta, and M. Ercegovac, "Trading accuracy for power with an underdesigned multiplier architecture," in *VLSI Design (VLSI Design), 2011 24th International Conference on*, 2011, pp. 346-351.

[12] D. Kelly, B. Phillips, and S. Al-Sarawi, "Approximate signed binary integer multipliers for arithmetic data value speculation," 2009.

[13] C. Liu, "Design and analysis of approximate adders and multipliers," 2014.

[14] C. Liu, J. Han, and F. Lombardi, "A low-power, high-performance approximate multiplier with configurable partial error recovery," in *Proceedings of the conference on Design, Automation & Test in Europe*, 2014, p. 95.

[15] A. Momeni, H. Jie, P. Montuschi, and F. Lombardi, "Design and Analysis of Approximate Compressors for Multiplication," *Computers, IEEE Transactions on,* vol. 64, pp. 984-994, 2015.
[16] J. Ma, K. Man, T. Krilavicius, S. Guan, and T. Jeong, "Implementation of High Performance Multipliers Based on Approximate Compressor Design," presented at the International Conference on Electrical and Control Technologies Yichang, China, 2011.
[17] C.-H. Chang, J. Gu, and M. Zhang, "Ultra low-voltage low-power CMOS 4-2 and 5-2 compressors for fast arithmetic circuits," *IEEE Transactions on Circuits and Systems I: Regular Papers,* vol. 51, pp. 1985-1997, 2004.
[18] P. Behrooz, "Computer arithmetic: Algorithms and hardware designs," *Oxford University Press,* vol. 19, pp. 512583-512585, 2000.
[19] J. Gu and C.-H. Chang, "Ultra low voltage, low power 4-2 compressor for high speed multiplications," in *Circuits and Systems, 2003. ISCAS'03. Proceedings of the 2003 International Symposium on*, 2003, pp. V-V.
[20] M. Margala and N. G. Durdle, "Low-power low-voltage 4-2 compressors for VLSI applications," in *Low-Power Design, 1999. Proceedings. IEEE Alessandro Volta Memorial Workshop on*, 1999, pp. 84-90.
[21] K. Prasad and K. K. Parhi, "Low-power 4-2 and 5-2 compressors," in Signals, Systems and Computers, 2001. Conference Record of the Thirty-Fifth Asilomar Conference on, 2001, pp. 129-133.
[22] M. D. Ercegovac and T. Lang, *Digital arithmetic*: Elsevier, 2004.
[23] D. Baran, M. Aktan, and V. G. Oklobdzija, "Energy efficient implementation of parallel CMOS multipliers with improved compressors," in *Low-Power Electronics and Design (ISLPED), 2010 ACM/IEEE International Symposium on*, 2010, pp. 147-152.
[24] G. K. Wallace, "The JPEG still picture compression standard," *IEEE transactions on consumer electronics,* vol. 38, pp. xviii-xxxiv, 1992.
[25] K. K. Parhi, VLSI digital signal processing systems: design and implementation: John Wiley & Sons, 2007.
[26] N. Banerjee, G. Karakonstantis, and K. Roy, "Process variation tolerant low power DCT architecture," in *Proceedings of the conference on Design, automation and test in Europe*, 2007, pp. 630-635.