Multipliers With Approximate 4-2 Compressors and Error Recovery Modules

Minho Ha and Sunggu Lee, *Member, IEEE*

***Abstract*—Approximate multiplication is a common operation used in approximate computing methods for high performance and low power computing. Power-efficient circuits for approximate multiplication can be realized with an approximate 4-2 compressor. This letter presents a novel design that uses a modification of a previous approximate 4-2 compressor design and adds an error recovery module. The proposed design, even with the additional error recovery module, is more accurate, requires less hardware, and consumes less power than previously proposed 4-2 compressor-based approximate multiplier designs.**

***Index Terms*—Approximate computing, arithmetic and logic units, error recovery, multiplier.**

I. Introduction

APPROXIMATE computing is a promising solution for fast computation with restricted resource budgets [1]. As a commonly used arithmetic unit, multipliers are a desirable target for circuit-level approximate computing [2]–[9]. Multipliers are commonly implemented using three phases. Given two n-bit operands to be multiplied.

1) The bits of the multiplicand are AND'ed with the bits of the multiplier (and shifted left by the multiplier bit position) to produce a set of n partial product terms.
2) The n partial product terms are compressed (or reduced) down to two terms by adding them together in some manner.
3) The final two terms are added together to form the final product.

Of the above three phases used in a multiplier, phase 2) typically requires the dominant proportion of area, delay, and power [10]. Also, methods developed for phase 2) can be included with other methods developed for phases 1) and 3). Thus, focusing specifically on phase 2), an n-to-2 compressor can be designed by using a sequence of k-2 compressors [10], full adders, and half adders placed at strategic locations in an n-to-2 compressor tree. Although the 3-2 compressor, also known as a carry-save adder, is the most well known, Chang *et al.* [10] have shown that a 4-2 compressor can be used to produce better results. By using *approximate* 4-2 compressors (shown in Fig. 1) [11]–[13], it is also possible to design approximate multipliers with significantly decreased power, area, and delay characteristics.

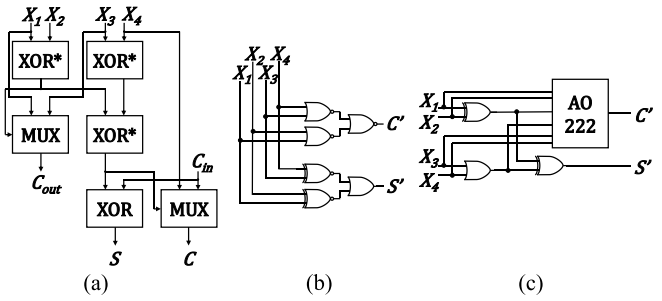


Fig. 1. Previous 4-2 compressor designs: (a) accurate 4-2 compressor design in [10] (XOR* is an XOR–XNOR module), (b) approximate 4-2 compressor design in [11] and [12], and (c) approximate 4-2 compressor design in [13] (AO222 refers to a custom AND-OR complex compound gate).

This letter improves upon the results of [12] and [13] by presenting an approximate multiplier design-based on a modified 4-2 compressor and a novel error recovery unit design. Example image processing applications are used to show that the proposed approximate multiplier is practical and produces acceptable results with real applications. Error analysis and application-specified integrated circuit (ASIC) postsynthesis results are used to show that, even with the addition of the error recovery module, the proposed design improves upon the previous work. 8-32 bit multipliers with the proposed design require 23.2%–24.4% smaller hardware area, 22.4%–24.5% lower power usage, and 11.2%–17.0% shorter delay than a corresponding exact multiplier. Also, when compared to [13], which outperforms [12], the accuracy is improved by 11.7% and the error distribution shows a lower and more even distribution.

II. Multiplication Using Approximate 4-2 Compressors

A. Previous Approximate 4-2 Compressor Designs

Recent research in [11] and [12] used the approximate 4-2 compressor design of Fig. 1(b). The main idea behind their design is the observation that, if a small number of erroneous entries are permitted, the truth table for an exact 4-2 compressor can be modified to yield a simpler logic implementation. Yang *et al.* [13] then proposed an improvement on the approximate 4-2 compressor design of [12]. A low error rate approximate 4-2 compressor was designed by considering the characteristics of the AND gate that is used in partial product generation. The probability that a partial product bit equals 1 and 0 are 1/4 and 3/4, respectively. Thus, an input to

Manuscript received January 11, 2017; revised May 23, 2017; accepted August 23, 2017. Date of publication August 29, 2017; date of current version February 24, 2018. This work was supported by the Samsung Research Funding and Incubation Center of Samsung Electronics under Project SRFC-TB1703-07. This manuscript was recommended for publication by S. Hellebrand, J. Henkel, A. Raghunathan, and H.-J. Wunderlich. *(Corresponding author: Sunggu Lee.)*

The authors are with the Department of Electrical Engineering, Pohang University of Science and Technology, Pohang 790-784, South Korea (e-mail: mh0205@postech.ac.kr; slee@postech.ac.kr).

Color versions of one or more of the figures in this paper are available online at http://ieeexplore.ieee.org.

Digital Object Identifier 10.1109/LES.2017.2746084

TABLE I
TRUTH TABLE OF APPROXIMATE 4-2 COMPRESSOR DESIGN IN [13] AND PROPOSED DESIGN

$X_4X_3X_2X_1$	C'_{yang}	S'_{yang}	D'_{yang}	$C'_{proposed}$	$S'_{proposed}$	$D'_{proposed}$
0 0 0 0	0	0	0	0	0	0
0 0 0 1	0	1	0	0	1	0
0 0 1 0	0	1	0	0	1	0
0 0 1 1	1	0	0	1	0	0
0 1 0 0	0	1	0	0	1	0
0 1 0 1	1	0	0	1	0	0
0 1 1 0	1	0	0	1	0	0
0 1 1 1	1	1	0	1	1	0
1 0 0 0	0	1	0	0	1	0
1 0 0 1	1	0	0	1	0	0
1 0 1 0	1	0	0	1	0	0
1 0 1 1	1	1	0	1	1	0
1 1 0 0	**1**	1	**1**	**0**	1	**-1**
1 1 0 1	1	0	-1	1	0	-1
1 1 1 0	1	0	-1	1	0	-1
1 1 1 1	1	1	-1	1	1	-1

a 4-2 compressor is three times more likely to be a 0 than a 1. Also, if there is no carry-in, a carry-out can only occur when all four input bits are 1. Denoting the two outputs of a 4-2 compressor as C' and S' and simply letting $C'S'$ equal 11 (instead of 00 with a carry-out of 1) when all inputs (X_1, X_2, X_3, and X_4) are 1, (1) and (2) can be used for the carry C' and sum S' bits. This not only obviates the need for carry-in and carry-out bits, it also makes the resulting logic implementation simpler. The tradeoff is a small loss in accuracy, as reflected in the truth table of Table I. In Table I, D refers to the difference between the outputs of approximate and exact 4-2 compressor circuits

$$C'_{\text{yang}} = X_1 \cdot X_2 + X_1 \cdot X_3 + X_1 \cdot X_4 + X_2 \cdot X_3 + X_2 \cdot X_4 + X_3 \cdot X_4 \quad (1)$$

$$S'_{\text{yang}} = (X_1 \oplus X_2) \oplus (X_3 + X_4). \quad (2)$$

B. Proposed 4-2 Compressor and Error Recovery Module

In this letter, an improved multiplier design has been devised by manipulating the truth table entries that result in erroneous outputs in a 4-2 compressor. For inputs 1100, the exact outputs should be 10, while the compressor in [13] produces 11 ($S' = 1$ instead of 0). Thus, $D_{\text{yang}} = 11 - 10 = 1$ using binary arithmetic. However, of the four inexact entries in Table I, this is the only row that has a difference of 1 (the others have difference -1).

As shown in the last column of Table I, in the proposed design, all truth table entries are designed to either produce $D = 0$ or $D = -1$. This is done by producing outputs 01 when the inputs are 1100, while leaving all of the other entries of Table I the same. Since there are now the same number of inexact entries (in the last four rows of Table I), but the error is always in the same direction (toward making the outputs smaller), a simple error recovery circuit can be used to correct the error when $X_4X_3 = 11$. Equations (3) and (4) show the logic equations for the C' and S' terms in the proposed 4-2 compressor. Notably, C' has one fewer product term than in the design of [13]. New approximate multiplier circuits can be designed using the proposed 4-2 compressor design in combination with a simple error recovery module

$$C'_{\text{proposed}} = X_1 \cdot X_2 + X_1 \cdot X_3 + X_1 \cdot X_4 + X_2 \cdot X_3 + X_2 \cdot X_4 \quad (3)$$

$$S'_{\text{proposed}} = (X_1 \oplus X_2) \oplus (X_3 + X_4). \quad (4)$$

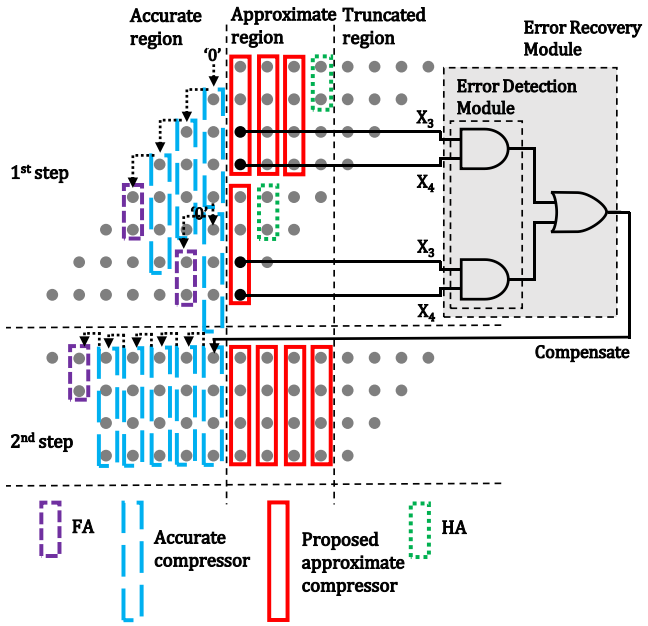


Fig. 2. Partial product accumulation process using truncation and proposed approximate compressors for an 8-bit approximate multiplier with error detection module and error recovery module.

To illustrate the proposed multiplier design, the phase 2) multiplier part (n-2 compression) for an 8-bit approximate multiplier design is shown in Fig. 2. As in [13], the partial product accumulation part of an approximate multiplier is divided into three regions: 1) a 7-bit accurate region; 2) a 4-bit approximate region; and 3) a 4-bit truncated region. Carry propagation, shown with dashed and solid arrows in Fig. 2, only occurs in a few places and in a noncascading manner (a carry propagates only once and can be computed in parallel with other sum and carry terms). The dashed arrows show the carry propagation used in an exact 4-2 compressor (the interested reader is referred to [13] for details). The solid arrow shows the only carry propagation used in the proposed multiplier design. The two pairs of four entries in the most significant portion of the approximate region produce two error recovery terms. These are then OR'ed together to produce the carry into the least significant bit of the accurate region. Error recovery is only used at this position because it has the largest influence on accuracy. A 16-bit (32-bit) approximate multiplier would have 2 (4) such carry terms produced from 4 (8) error recovery terms in the most significant portion of the approximate region. Since the error recovery module is only used in the most significant column of the approximate region, the overall hardware overhead is small.

In the proposed n-bit approximate multiplier, partial product accumulation requires $\log n - 1$ steps since the n partial product terms can be reduced to 2 terms with $\log n - 1$ steps of 4-2 compression. Using truncation and approximate compressors in the less significant bits of the partial and accumulated products results in decreased power consumption and circuit area, while using accurate compressors in the most significant

TABLE II
MED AND NUMBER OF CORRECT OUTPUTS COMPARISON (8-BIT CASE)

	MED	*Maximum ED*	*Std. Dev.*	*# exact (out of 65536)*
Design in [12]	48.68	176	38.01	4928
Design in [13]	31.76	641	52.11	11200
Proposed Design	**28.05**	**385**	**39.02**	**11200**

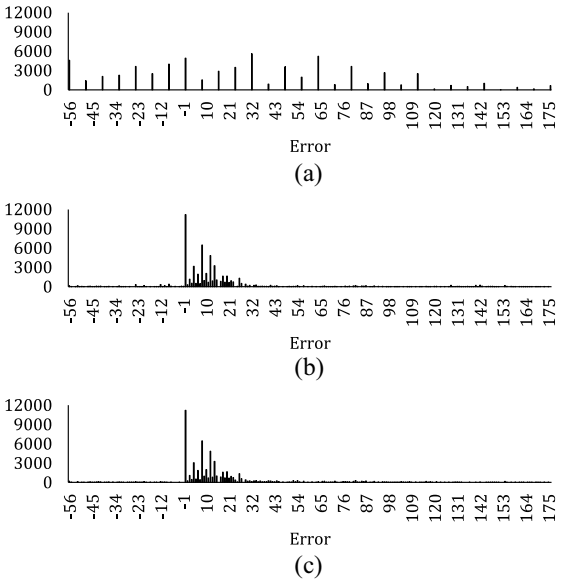


Fig. 3. Error distribution in 8-bit approximate multiplier designs: (a) approximate multiplier design in [12]; (b) approximate multiplier design in [13]; and (c) proposed approximate multiplier.

bits can mitigate the errors caused by truncation and the use of approximate compressors. As shown in Fig. 2, the proposed error recovery module is a simple 2-level AND–OR circuit.

III. ANALYSIS AND EXPERIMENTAL RESULTS

Approximate multipliers using the proposed 4-2 compressor and error recovery module have been compared with possible alternatives [12] and [13] ([11] uses the same compressor as [12]) using analysis and ASIC postsynthesis implementation.

A. Error Analysis

Two common metrics for analyzing the accuracy of approximate multipliers are the mean error distance (MED) and number of exact outputs. The MED of an n-bit approximate multiplier can be defined as

$$\text{MED} = \frac{\sum_{i=0}^{2^n-1}\sum_{j=0}^{2^n-1}\left|\text{product}_{\text{acc}}(i,j) - \text{product}_{\text{appr}}(i,j)\right|}{2^{2n}}. \tag{5}$$

Using $n = 8$ as an example, Table II shows the MED, maximum error distance (ED), standard deviation, and number of correct outputs with the proposed approximate multiplier and the approximate multipliers in [12] and [13]. Approximate multipliers based on the design of [13] and the proposed design have the same number of correct outputs. However, because of the error recovery module, the proposed approximate multiplier has the smallest MED value.

TABLE III
POSTSYNTHESIS RESULTS FOR MULTIPLIERS WITH 4-2 COMPRESSORS

		Area (μm^2)	*Power (mW)*	*Delay (ns)*
8-bit	Accurate design	674.56	0.2153	3.04
	Design in [12]	620.80	0.1760	2.79
	Design in [13]	521.28	0.1724	2.70
	Proposed Design	**510.08**	**0.1646**	**2.70**
16-bit	Accurate design	2667.52	0.9772	5.90
	Design in [12]	2401.60	0.8249	5.55
	Design in [13]	2101.12	0.7985	5.07
	Proposed Design	**2048.32**	**0.7584**	**5.02**
32-bit	Accurate design	10750.40	4.5206	11.66
	Design in [12]	9453.44	3.6205	11.35
	Design in [13]	8391.68	3.5874	9.79
	Proposed Design	**8163.52**	**3.4149**	**9.73**

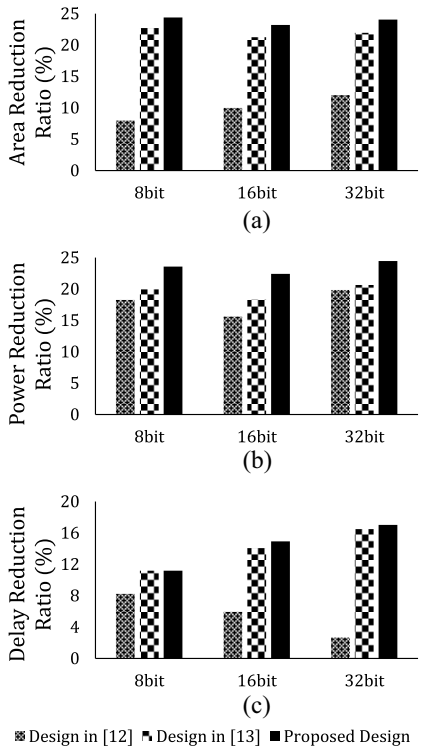
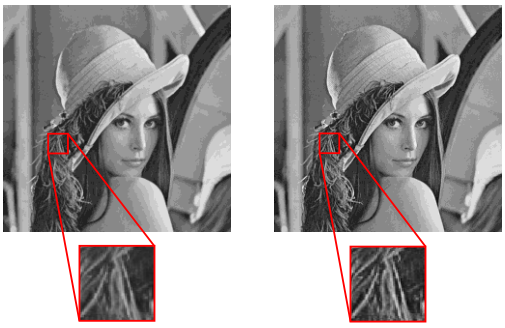


Fig. 4. (a) Area, (b) power, and (c) delay reduction ratio of the proposed multipliers and multipliers in [12] and [13].

In order to analyze the behavior of approximate multiplier designs, a plot of the distribution of errors were generated, as shown in Fig. 3. The proposed method has an error distribution with most values concentrated at lower ED values.

B. ASIC Postsynthesis Results

For a hardware overhead and power/delay comparison, the proposed approximate multiplier and the approximate multipliers in [12] and [13] have been synthesized using 65 nm CMOS cell library. To synthesize these approximate multiplier designs, Synopsys Design Compiler has been used. The postsynthesis results are summarized in Table III. Even though the proposed design has an added error recovery module, the proposed design still has better area, power, and delay characteristics than the previous approximate multiplier designs in [12] and [13]. Fig. 4 shows the reductions in area, power, and delay for 8, 16, and 32-bit multipliers.



(a) (b)

Fig. 5. Image sharpening example using the proposed approximate multiplier design: (a) original image and (b) sharpened image.

TABLE IV
PSNR RESULTS OF IMAGE SHARPENING METHOD

	$PSNR_{image1}$	$PSNR_{image2}$	$PSNR_{image3}$
Design in [12]	33.7787	40.9170	45.6257
Design in [13]	48.1978	45.5306	50.1626
Proposed Design	**48.2951**	**48.1506**	**53.5515**

IV. PRACTICAL USAGE OF PROPOSED METHOD

To demonstrate the practicality of the proposed approximate multiplier for error-resilient applications, an image processing application, *image sharpening*, has been used. The main calculation required for image sharpening, as shown in [13], is

$$S(x,y) = 2I(x,y) - \frac{1}{273}\sum_{i=-2}^{2}\sum_{j=-2}^{2} G(i+3,j+3)I(x-i,y-j) \tag{6}$$

where $I(S)$ is the original (sharpened) image and G is

$$G = \begin{bmatrix} 1 & 4 & 7 & 4 & 1 \\ 4 & 16 & 26 & 16 & 4 \\ 7 & 26 & 41 & 26 & 7 \\ 4 & 16 & 26 & 16 & 4 \\ 1 & 4 & 7 & 4 & 1 \end{bmatrix}. \tag{7}$$

In this method, only the multiplications are approximate, while all others are exact operations. This method has been tested using MATLAB R2016a. Fig. 5 shows an example of the use of this method with the proposed approximate multiplier design.

The peak signal-to-noise ratio (PSNR) is a common metric used to evaluate the performance of image processing algorithms. Table IV compares the PSNR values achieved with the proposed design and previous designs in [12] and [13] when tested with three test images. The PSNR of the proposed design is better than [12] and [13], by up to 30.0%, for all three test images.

V. CONCLUSION

Arguably the most important operations in a digital signal processor are addition and multiplication. Focusing on the latter, this letter has proposed a novel 4-2 compressor design that can be used in the partial product compression phase of a binary multiplier. The proposed design uses a small modification of an existing 4-2 compressor design that changes the error profile such that *all* errors result in slightly decreased values (over the exact values). The resulting error profile is such that *all* such error points can be easily located with a simple 2-input AND. A simple error recovery module can then be used to recover from such errors when desired. An approximate multiplier can be designed with accurate and approximate regions. Then, the proposed 4-2 compressor can be used in the approximate region and a few error recovery modules can be used in the most significant bit position of the approximate region.

An approximate 8-32 bit multiplier with this proposed design requires 23.2%–24.4% smaller hardware area, 22.4%–24.5% lower power usage, and 11.2%–17.0% shorter delay than an exact multiplier. It is notable that all three metrics have been improved with no tradeoffs necessary. Finally, error analysis shows that the proposed multiplier design improves the MED of multiplication results by 11.7% and more over previous approximate multiplier designs.

REFERENCES

[1] S. Mittal, "A survey of techniques for approximate computing," *ACM Comput. Surveys*, vol. 48, no. 4, p. 62, May 2016.

[2] S. Narayanamoorthy, H. A. Moghaddam, Z. Liu, T. Park, and N. S. Kim, "Energy-efficient approximate multiplication for digital signal processing and classification applications," *IEEE Trans. Very Large Scale Integr. (VLSI) Syst.*, vol. 23, no. 6, pp. 1180–118, Jun. 2015.

[3] B. Shao and P. Li, "Array-based approximate arithmetic computing: A general model and applications to multiplier and squarer design," *IEEE Trans. Circuits Syst. I, Reg. Papers*, vol. 62, no. 4, pp. 1081–1090, Apr. 2015.

[4] H. Jiang, J. Han, F. Qiao, and F. Lombardi, "Approximate radix-8 Booth multipliers for low-power and high-performance operation," *IEEE Trans. Comput.*, vol. 65, no. 8, pp. 2638–2644, Aug. 2016.

[5] G. Zervakis, S. Xydis, K. Tsoumanis, D. Soudris, and K. Pekmestzi, "Hybrid approximate multiplier architectures for improved power-accuracy trade-offs," in *Proc. IEEE/ACM Int. Symp. Low Power Electron. Design*, Rome, Italy, 2015, pp. 79–84.

[6] S. Hashemi, R. I. Bahar, and S. Reda, "DRUM: A dynamic range unbiased multiplier for approximate applications," in *Proc. IEEE/ACM Int. Conf. Comput.-Aided Design*, Austin, TX, USA, 2015, pp. 418–425.

[7] K. Bhardwaj, P. S. Mane, and J. Henkel, "Power- and area-efficient approximate wallace tree multiplier for error-resilient systems," in *Proc. Int. Symp. Qual. Electron. Design*, Santa Clara, CA, USA, 2014, pp. 263–269.

[8] R. Zendegani, M. Kamal, M. Bahadori, A. Afzali-Kusha, and M. Pedram, "RoBA multiplier: A rounding-based approximate multiplier for high-speed yet energy-efficient digital signal processing," *IEEE Trans. Very Large Scale Integr. (VLSI) Syst.*, vol. 25, no. 2, pp. 393–401, Feb. 2017.

[9] M. Shafique, R. Hafiz, S. Rehman, W. El-Harouni, and J. Henkel, "Cross-layer approximate computing: From logic to architectures," in *Proc. ACM/EDAC/IEEE Design Autom. Conf.*, Austin, TX, USA, 2016, pp. 1–6.

[10] 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 Trans. Circuits Syst. I, Reg. Papers*, vol. 51, no. 10, pp. 1985–1997, Oct. 2004.

[11] N. Maheshwari, Z. Yang, J. Han, and F. Lombardi, "A design approach for compressor based approximate multipliers," in *Proc. Int. Conf. VLSI Design*, Bengaluru, India, 2015, pp. 209–214.

[12] A. Momeni, J. Han, P. Montuschi, and F. Lombardi, "Design and analysis of approximate compressors for multiplication," *IEEE Trans. Comput.*, vol. 64, no. 4, pp. 984–994, Apr. 2015.

[13] Z. Yang, J. Han, and F. Lombardi, "Approximate compressors for error-resilient multiplier design," in *Proc. IEEE Int. Symp. Defect Fault Tolerance VLSI Nanotechnol. Syst.*, Amherst, MA, USA, 2015, pp. 183–186.

[14] J. Liang, J. Han, and F. Lombardi, "New metrics for the reliability of approximate and probabilistic adders," *IEEE Trans. Comput.*, vol. 62, no. 9, pp. 1760–1771, Sep. 2013.