A Comparative Evaluation of Approximate Multipliers

Honglan Jiang*, Cong Liu*, Naman Maheshwari†, Fabrizio Lombardi‡ and Jie Han*
*Department of Electrical and Computer Engineering
University of Alberta, Edmonton, AB T6G 1H9, Canada
Email: {honglan, cong4, jhan8}@ualberta.ca
†Department of Electrical and Electronics Engineering
Birla Institute of Technology and Science, Pilani, Rajasthan, India
Email: naman.mah1993@gmail.com
‡Department of Electrical and Computer Engineering
Northeastern University, Boston, MA, USA
Email: lombardi@ece.neu.edu

***Abstract*—A multiplier has a significant impact on the speed and power dissipation of an arithmetic processor. Precise results are not always required in many algorithms, such as those for classification and recognition in data processing. Moreover, many errors do not make an obvious difference in applications such as image processing due to the perceptual limitations of human beings. Error-tolerant algorithms and applications have promoted the development of approximate multipliers to tradeoff accuracy for speed, implementation area and/or power efficiency. This paper briefly reviews the current designs of approximate multipliers and provides a comparative evaluation of their error and circuit characteristics. Image sharpening is performed using the considered approximate multipliers to assess their performance in such applications.**

***Index Terms*—Approximate computing, Multiplier, Accuracy**

I. Introduction

Arithmetic units such as adders and multipliers are key components in a logic circuit. The speed and power consumption of arithmetic circuits significantly influence the performance of a processor. High-performance arithmetic circuits such as carry look ahead adders (CLAs) and Wallace tree multipliers have been widely utilized. However, traditional arithmetic circuits that perform exact operations are encountering difficulties in performance improvement. Approximate arithmetic that allows a loss of accuracy can reduce the critical path delay of a circuit. Since most approximate designs leverage simplified logic, they tend to have a reduced power consumption and area overhead. Thus, approximate arithmetic is advocated as an approach to improve the speed, area and power efficiency of a processor due to the error-resilience of some algorithms and applications [1].

As an important arithmetic module, the multiplier has been redesigned to many approximate versions. The often conflicting advantages and disadvantages of these designs make it difficult to select the most suitable approximate multiplier for a specific application. Thus, approximately redesigned multipliers are reviewed in this paper and a comparative evaluation is performed by considering both the error and circuit characteristics.

II. Review and Classification

Generally, a multiplier consists of stages of partial product generation, accumulation and final addition. The commonly used partial product accumulation structures include the Wallace, Dadda trees and a carry-save adder array. In a Wallace tree, $log_2(n)$ layers are required for an n-bit multiplier. The adders in each layer operate in parallel without carry propagation, and the same operation repeats until two rows of partial products remain. Therefore, the delay of the partial product accumulation stage is $O(log_2(n))$. Moreover, the adders in a Wallace tree can be considered as a 3:2 compressor and can be replaced by other counters or compressors (e.g. a 4:2 compressor) to further reduce the delay. The Dadda tree has a similar structure as the Wallace tree, but it uses as few adders as possible.

For a carry-save adder array, the carry and sum signals generated by the adders in a row are connected to the adders in the next row. Adders in a column operate in series. Hence the partial product accumulation delay of an n-bit multiplier is approximately $O(n)$, longer than that of the Wallace tree. However, an array requires a smaller area and thus a lower power dissipation due to the simple and symmetric structure.

Three methodologies are applicable to approximate a multiplier: i) approximation in generating the partial products [2], ii) approximation (including truncation) in the partial product tree [3]–[5], and iii) using approximate designs of adders [6], counters [7] and/or compressors [8], [9] to accumulate the partial products. Following this classification, existing designs of approximate multipliers are briefly reviewed next.

A. Approximation in generating partial products

The underdesigned multiplier (UDM) utilizes an approximate 2×2 bit multiplier block obtained by altering a single entry in the Karnaugh Map (K-Map) of its function [2]. In this approximation, the accurate result "1001" for the multiplication of "11" and "11" is simplified to "111" to save one output bit. Assuming the value of each input bit is equally likely, the error rate of the 2×2 bit multiplier block is $(\frac{1}{2})^4 = \frac{1}{16}$. Larger multipliers can be designed based on the 2×2 bit multiplier. This multiplier introduces an error when generating partial products, however the adder tree remains accurate.

B. Approximation in the partial product tree

A bio-inspired imprecise multiplier referred to as a broken-array multiplier (BAM) is proposed in [3]. The BAM operates by omitting some carry-save adders in an array multiplier in both horizontal and vertical directions.

The error tolerant multiplier (ETM) [4] is divided into a multiplication section for the MSBs and a non-multiplication section for the LSBs. A NOR gate based control block is used to deal with two cases: i) if the product of the MSBs is zero, then the multiplication section is activated to multiply the LSBs without any approximation, and ii) if the product of the MSBs is nonzero, the non-multiplication section is used as an approximate multiplier to process the LSBs, while the multiplication section is activated to multiply the MSBs.

The static segment multiplier (SSM) [10] was further proposed using a similar partition scheme. Different from ETM, no approximation is applied to the LSBs in the SSM. Either the MSBs or the LSBs of each of the operands is accurately multiplied depending on whether its MSBs are all zeros. [11] has shown that a small improvement in accuracy and hardware cost is achieved compared to the ETM, thus this design is not considered further in the comparison study.

A power and area-efficient approximate Wallace tree multiplier (AWTM) is based on a bit-width aware approximate multiplication and a carry-in prediction method [5]. An n-bit AWTM is implemented by four $n/2$-bit sub-multipliers, and the most significant $n/2$-bit sub-multiplier is further implemented by four $n/4$-bit sub-multipliers. The AWTM is configured into four different modes by the number of approximate $n/4$-bit sub-multipliers in the most significant $n/2$-bit sub-multiplier. The approximate partial products are then accumulated by a Wallace tree.

C. Using approximate counters or compressors in the partial product tree

In the inaccurate counter based multiplier (ICM), an approximate (4:2) counter is proposed for an inaccurate 4-bit Wallace multiplier [7]. The carry and sum of the counter are approximated as “10” (for “100”) when all input signals are ‘1’. As the probability of obtaining a partial product of ‘1’ is $\frac{1}{4}$, the error rate of the approximate (4:2) counter is $(\frac{1}{4})^4 = \frac{1}{256}$. The inaccurate 4-bit multiplier is then used to construct larger multipliers with error detection and correction circuits.

In the compressor based multiplier, accurate (3:2) and (4:2) compressors are improved to speed up the partial product accumulation stage [12]. By using the improved compressors, better energy and delay characteristics are obtained for a multiplier. To further reduce delay and power, two approximate (4:2) compressor designs (AC1 and AC2) are presented in [9]; these compressors are used in a Dadda multiplier with four different schemes. Approximate counters in which the more significant output bits are ignored, are presented and evaluated in [13]; several signed multipliers are also implemented using these approximate counters. As only unsigned multipliers are discussed in this paper, the more accurate schemes 3 and 4 of the approximate compressor based multiplier (referred to as ACM-3 and ACM-4) in [9] are considered in the comparison.

In the approximate multiplier with configurable error recovery, the partial products are accumulated by a novel approximate adder [6]. The approximate adder utilizes two adjacent inputs to generate a sum and an error bit. The adder processes data in parallel, thus no carry propagation is required. Two approximate error accumulation schemes are then proposed to alleviate the error of the approximate multiplier (due to the approximate adder). OR gates are used in the first error accumulation stage in scheme 1 (AM1), while in scheme 2 (AM2), both OR gates and approximate adders are used. The truncation of 16 LSBs in the partial products in AM1 and AM2 results in TAM1 and TAM2 respectively [14].

III. Comparison of Error and Circuit Characteristics

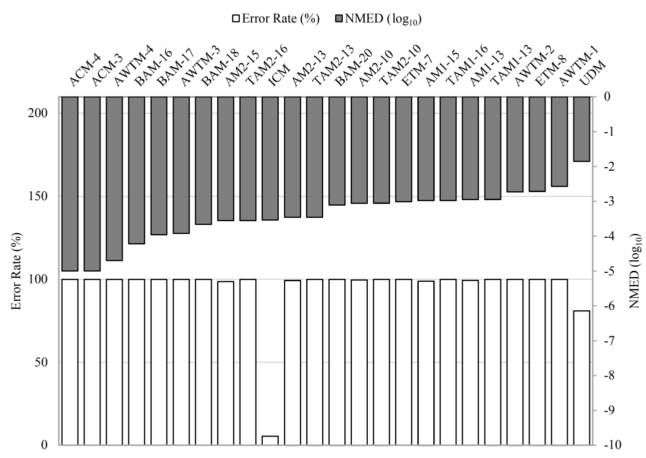
A. Error Characteristics

The error rate (ER), mean relative error distance ($MRED$), and normalized mean error distance ($NMED$) [15] are used as metrics to assess the error characteristics of approximate multipliers. ER is defined as the probability of producing an incorrect result. $MRED$ is the average value of all possible relative error distances (RED). RED is calculated by $RED = \frac{ED}{M}$, where $ED = |M' - M|$ is the error distance, M' and M are the approximate and accurate results, respectively [15]. $NMED$ is the normalization of the mean error distance (MED) by the maximum output of the accurate design.

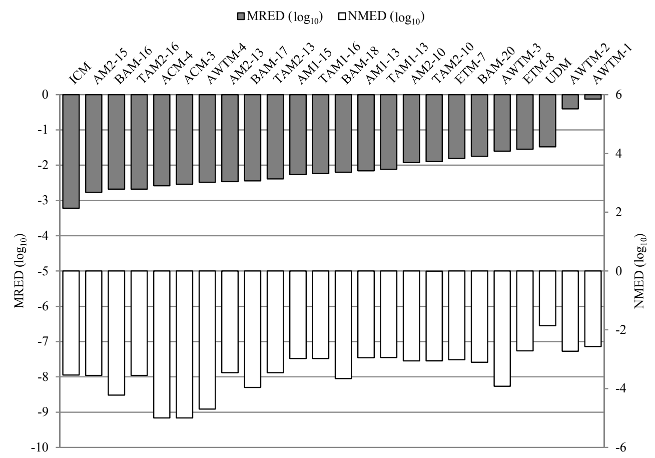
Monte Carlo simulation is performed to evaluate the accuracy of approximate multipliers. The considered approximate multipliers (16×16 bit) are simulated by MATLAB with 10^8 random input combinations and the $NMED$, $MRED$ and ER are obtained, as shown in Fig. 1, with ascending $NMED$ values in Fig. 1(a) and $MRED$ values in Fig. 1(b).

According to Fig. 1(a), most of the designs, especially those with truncation, have large ERs. For example, ETM and BAM result in nearly 100% error rates. However, ICM has a relatively low ER of 5.45%, because it uses just one approximate counter in a 4×4 bit sub-multiplier with an error rate of only $\frac{1}{256}$. UDM also shows a lower ER than the other approximate multipliers.

Among the approximate multipliers, UDM has the largest $NMED$ while ACM-4 has the smallest. Since only the LSBs are approximated in ACM-4, ACM-3 and AWTM-4, they achieve a high accuracy in terms of $NMED$. However, ACM-3, ACM-4 and AWTM-4 have a low accuracy for small operands, and they even generate non-zero product when the accurate product is zero; this is the reason for the $MRED$s to be not as small. For calculating the $MRED$, both input operands are selected to be non-zero. ICM, AM2-15 and TAM2-16 have similar values of $NMED$, however ICM has the smallest $MRED$, while the $MRED$ of TAM2-16 is the



(a)



(b)

Fig. 1. A comparison of ER, $NMED$ and $MRED$ of the considered approximate multipliers with data sorted on (a) $NMED$ and (b) $MRED$. The parameter k follows the acronym of each approximate multiplier. For AM1, AM2, TAM1 and TAM2, this parameter refers to the number of MSBs for error reduction and for ETM, the number of LSBs in the inaccurate part. It is the mode number in AWTM and ACM, and the vertical broken length (VBL) for BAM.

largest. Therefore ICM has the highest accuracy in terms of $MRED$, while TAM2-16 is the least accurate among these three approximate multipliers. This indicates that multipliers with simple truncation tend to have larger $MRED$s when their $NMED$s are similar. It can also be seen that ETM and BAM have relatively large $MRED$s due to truncation. AWTM-1 and AWTM-2 also have very large $MRED$s especially when one of the input is zero, in which case it has an infinite RED (because of its non-zero output).

In summary, ICM is the most accurate design with the lowest ER, $MRED$ and moderate $NMED$. ACM-4, ACM-3, AWTM-4, BAM-16, AM2-15 and TAM2-16 also show good accuracy among all considered approximate multipliers with both low $NMED$s and $MRED$s. ETM, AWTM-1, AWTM-2 and UDM are not very accurate in terms of these metrics. Some designs may have a low $NMED$ (e.g., BAM-18 and AWTM-3), however their $MRED$s are relatively high.

B. Circuit Characteristics

The considered 16×16 bit approximate multipliers are implemented in VHDL and synthesized using the Synopsys Design Compiler (DC) based on an STM CMOS 28 nm process. For a fair comparison, all designs use the same process, voltage and temperature configurations and optimization option. Both the P-channel and the N-channel transistors used in the designs have a typical design corner with a regular threshold voltage. The supply voltage of all designs are set to 1.0 V; the simulation temperature is 25 °C. Critical path delays and areas are reported by the Synopsys DC. The power dissipation is measured by the PrimeTime-PX tool at a clock period of 4 ns with 10 million random input combinations. The accurate Wallace multiplier (WallaceM) and array multiplier (ArrayM) are also simulated for comparison. To reduce the effect of the final addition, the same multi-bit adder in the tool library (hence, the one with the best performance and most power efficient) is utilized in all approximate multiplier designs as the final adder. Fig. 2 shows the critical path delay, area and power of the considered multipliers. The leakage power dissipations for the considered designs are very low (less than 1 uW), thus they are not considered in the comparison.

Fig. 2(a) shows that the accurate array multiplier (ArrayM) is the slowest and the Wallace multiplier (WallaceM) is the most power consuming; this is consistent with the theoretical analysis. Due to the expressively fast carry-ignored operation, AM1/TAM1, AM2/TAM2 have smaller delays even with a 16-bit error reduction compared to the other types of designs. BAM is significantly slower due to its array structure. AWTM, UDM, ICM and ACM have larger delays than the other multipliers, while their power consumptions show a quite different trend. BAM consumes very low power, the power consumptions of AWTM and ACM are in the medium range, while UDM and ICM incur relatively high power consumptions. ETM has small values for both delay and power dissipation.

Fig. 2(b) indicates that a multiplier with a higher power dissipation usually has a larger area. In terms of power and area, ETM, TAM1/TAM2 and BAM are among the best designs. A common feature of these designs is that they all use truncation, which can significantly affect $MRED$ while $NMED$ may not be changed as much. If most of the inputs have large values, the error introduced by truncation can be tolerated; thus truncation is a useful scheme to save area and power. Otherwise, truncation-based designs may yield unacceptably inaccurate results. Without truncation, a multiplier whose design is approximated when generating partial product (e.g. UDM) tends to have a large delay, power

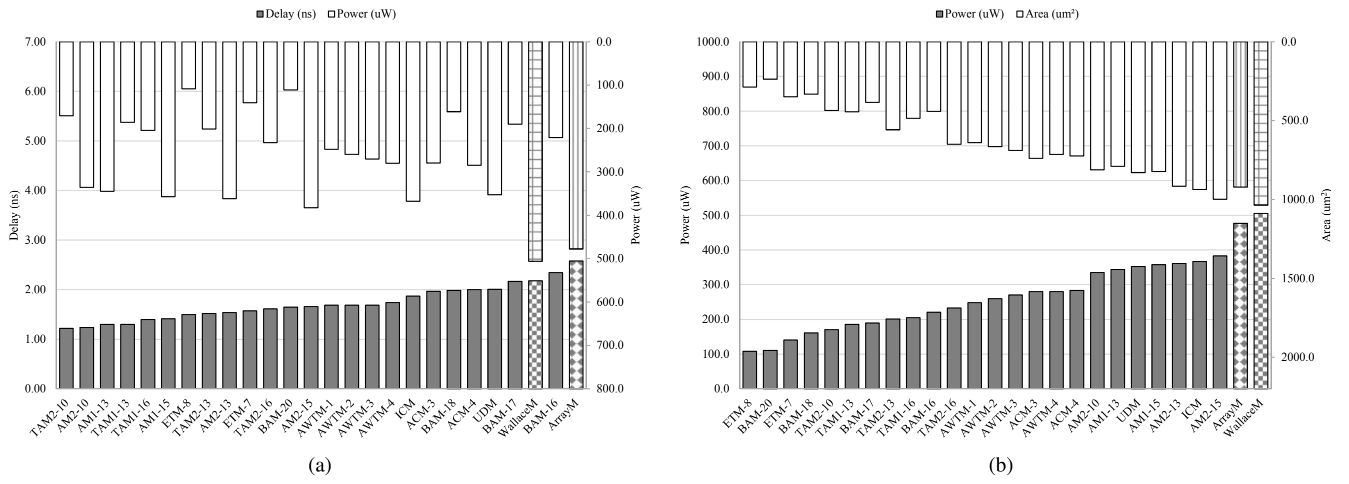


Fig. 2. A comparison of delay, power and area of the considered multipliers with data sorted on (a) delay and (b) power.

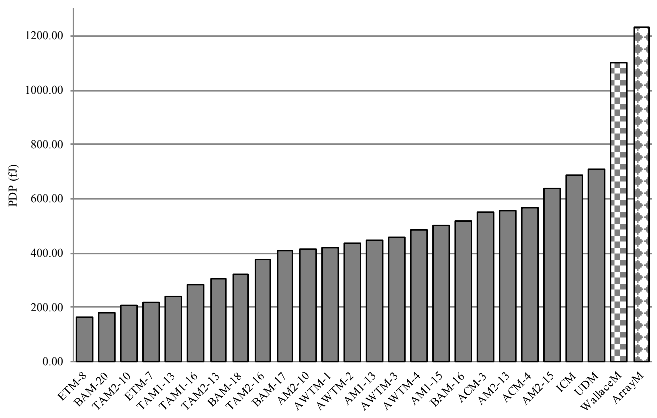


Fig. 3. Power-delay products of the considered multipliers.

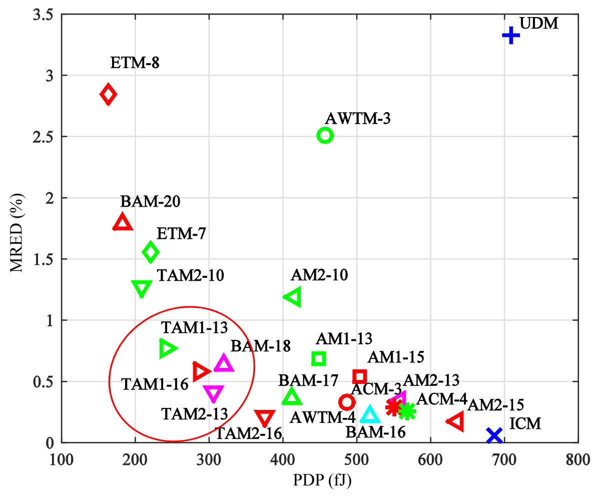


Fig. 4. Mean relative error distances and power-delay products of the approximate multipliers.

and area. These measures for the multiplier approximated in the partial product tree (e.g. AWTM) are moderate, while the multipliers using approximate counters or compressors (ICM, ACM, AM1, AM2) require higher power and area.

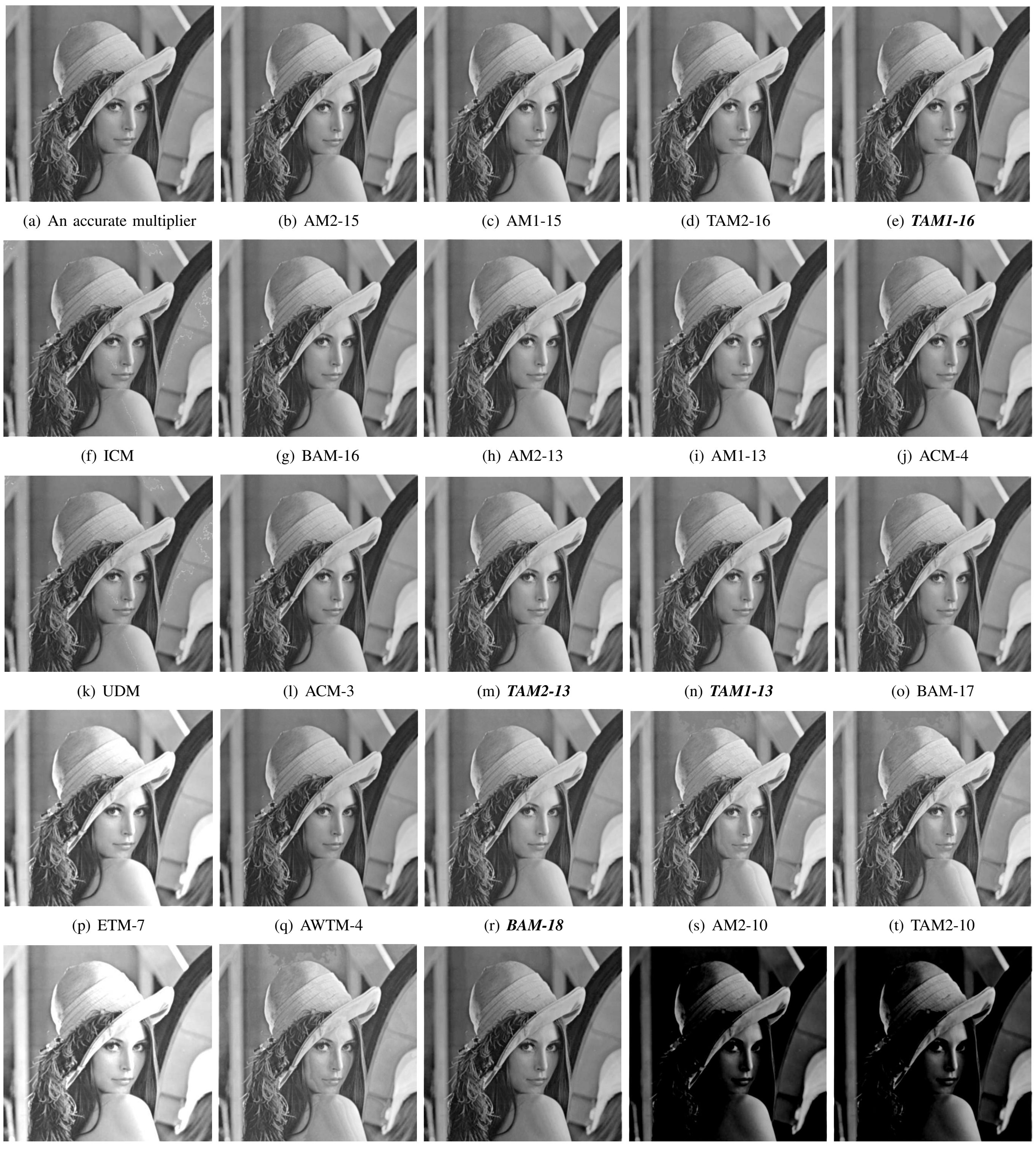
The values of power-delay product (PDP) for the considered multipliers are shown in Fig. 3 for a better overview of the circuit characteristics. ETM, TAM1 and TAM2 have very small PDPs due to their small values of power and delay, while ICM and UDM are the opposite. The values of PDP for AM1 and AM2 are in the medium range. The PDPs of BAM vary according to the number of truncated bits.

C. Discussion

$MRED$ and PDP are jointly considered for an overall evaluation of the different approximate multipliers, as shown in Fig. 4. TAM1-13, TAM1-16 TAM2-13 and BAM-18 have both small PDPs and MREDs. Most of the other designs have at least one major shortcoming. ICM and ACM incur a very low error, but their PDPs are very high. ETM-8 has the smallest PDP but a rather large $MRED$. UDM shows a poor performance in both PDP and $MRED$. Even though most BAM configurations have small PDPs, their delays are generally large (Fig. 2); moreover, some BAM configurations have low accuracies. AWTMs have large PDPs and only AWTM-4 has a high accuracy.

IV. Image Processing Application

For further comparison, the considered approximate multipliers are applied to image sharpening as an application [16]. The sharpened images are shown in Fig. 5. The multiplications in the algorithm are implemented by the considered approximate multipliers, while the division and subtraction are accurate. The quality of images sharpened by AM2-10, TAM2-10, ETM-8, BAM-20, AWTM-3, AWTM-2 and AWTM-1 is not very high due to their low accuracy (i.e., by large $NMED$s and $MRED$s). Likewise, The multipliers with higher accuracy (i.e., with smaller $NMED$s and $MRED$s), AM2-15, AM1-15, TAM2-16 and TAM1-16, show better performance for image sharpening. As for the other images,


(a) An accurate multiplier (b) AM2-15 (c) AM1-15 (d) TAM2-16 (e) ***TAM1-16***

(f) ICM (g) BAM-16 (h) AM2-13 (i) AM1-13 (j) ACM-4

(k) UDM (l) ACM-3 (m) ***TAM2-13*** (n) ***TAM1-13*** (o) BAM-17

(p) ETM-7 (q) AWTM-4 (r) ***BAM-18*** (s) AM2-10 (t) TAM2-10

(u) ETM-8 (v) BAM-20 (w) AWTM-3 (x) AWTM-2 (y) AWTM-1

Fig. 5. Image sharpened using different multipliers. The approximate multipliers with a good tradeoff in MRED and PDP are highlighted in bold italics.

it is difficult to distinguish the quality. Therefore, the peak signal noise ratio ($PSNR$) is computed, as shown in Table I. Moreover, the ratios of the PDP for each multiplier normalized by the accurate Wallace multiplier are also presented. TAM1-16 achieves an image sharpening result imperceptibly different from the exact multiplier, but with only a 25.99% PDP of the accurate Wallace multiplier. The image sharpening results of TAM2-13, TAM1-13 and BAM-18 are also acceptable with less than 30% PDP compared to WallaceM.

The $PSNR$s for the sharpened images show that multipliers with smaller values of $NMED$ and $MRED$ perform better in image sharpening (except for ICM and UDM). Although ICM has the smallest values of ER and $MRED$ among all considered approximate multipliers, its sharpened image does not show the best quality. This occurs because the rarely occurred errors in ICM are very large (as per its small value of ER); therefore very large errors lead to the obvious large white (erroneous) areas in Fig. 5(f). UDM has very large $NMED$ and $MRED$, however the sharpened image has a $PSNR$ value similar to ICM.

TABLE I. Peak signal noise ratios (PSNRs) of the sharpened images and the power-delay-product (PDP) values of the approximate multipliers normalized by the accurate Wallace multiplier. The approximate multipliers with good tradeoffs of MRED and PDP are highlighted in bold.

Multiplier	PSNR (dB)	PDP (%)
AM2-15	57.89	57.69
AM1-15	53.56	45.74
TAM2-16	48.28	34.03
TAM1 − 16	**46.97**	**25.99**
BAM-16	46.18	46.99
AM2-13	45.88	50.56
AM1-13	45.12	40.64
ACM-4	43.72	51.56
ACM-3	43.39	49.97
TAM2 − 13	**41.87**	**27.77**
TAM1 − 13	**41.42**	**21.92**
ICM	40.45	62.35
UDM	40.14	64.33
BAM-17	40.09	37.32
AWTM-4	38.63	44.21
BAM − 18	**33.99**	**29.09**
AM2-10	28.07	37.71
TAM2-10	27.49	18.88
BAM-20	22.06	16.62
AWTM-3	21.52	41.45
ETM-7	17.42	20.02
ETM-8	14.09	14.77
AWTM-2	8.84	39.78
AWTM-1	8.84	38.00

V. Conclusion

Approximate unsigned multipliers are comparatively evaluated for both error and circuit characteristics. Among the considered approximate multipliers, truncation on part of the partial products is an effective way to reduce circuit complexity. However, it incurs a large ER and moderate $NMED$ and $MRED$. UDM shows a low accuracy, especially in terms of the error distance, and a relatively high circuit overhead because the 2×2 bit approximate multiplier is used to compute the most significant bits and accurate adders are utilized to accumulate the generated partial products. When truncation is not used, multipliers approximated in the partial product tree tend to have a poor accuracy and moderate hardware consumption, while multipliers using approximate counters or compressors are usually very accurate with relatively high power dissipation and hardware consumption.

Image sharpening is used as an application to evaluate the performance of the considered approximate multipliers. In general, multipliers with smaller values of $NMED$ and $MRED$ perform better. However, a small ER does not guarantee a high performance of a multiplier for image sharpening because the error magnitude can be large.

References

[1] J. Han and M. Orshansky, "Approximate Computing: An Emerging Paradigm For Energy-Efficient Design," in *the 18th IEEE European Test Symposium (ETS)*, Avignon, France, May 2013, pp. 1–6.

[2] P. Kulkarni, P. Gupta, and M. Ercegovac, "Trading accuracy for power with an underdesigned multiplier architecture," in *International Conference on VLSI Design*, 2011, pp. 346–351.

[3] 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 Trans. Circuits and Systems*, vol. 57, no. 4, pp. 850–862, Apr. 2010.

[4] K. Y. Kyaw, W. L. Goh, and K. S. Yeo, "Low-power high-speed multiplier for error-tolerant application," in *EDSSC*, 2010, pp. 1–4.

[5] K. Bhardwaj, P. S. Mane, and J. Henkel, "Power- and area-efficient Approximate Wallace Tree Multiplier for error-resilient systems," in *Proceedings of the 15th International Symposium on Quality Electronic Design*. IEEE, Mar. 2014, pp. 263–269.

[6] 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*. European Design and Automation Association, 2014.

[7] C.-H. Lin and I.-C. Lin, "High accuracy approximate multiplier with error correction," in *ICCD*. IEEE, Oct. 2013, pp. 33–38.

[8] J. Ma, K. L. Man, N. Zhang, S.-U. Guan, and T. T. Jeong, "High-speed area-efficient and power-aware multiplier design using approximate compressors along with bottom-up tree topology," in *Fifth International Conference on Machine Vision (ICMV): Algorithms, Pattern Recognition, and Basic Technologies*, 2013.

[9] A. Momeni, J. Han, P. Montuschi, and F. Lombardi, "Design and Analysis of Approximate Compressors for Multiplication," *IEEE Transactions on Computers*, vol. 64, no. 4, pp. 984–994, 2015.

[10] 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 Transactions on VLSI Systems*, vol. 23, no. 6, pp. 1180–1184, 2015.

[11] C. Liu, H. Jiang, F. Lombardi, and J. Han, "High-performance approximate multiplier designs with configurable error recovery," *submitted for publication.*

[12] D. Baran, M. Aktan, and V. G. Oklobdzija, "Energy efficient implementation of parallel CMOS multipliers with improved compressors," in *Proceedings of the 16th ACM/IEEE international symposium on Low power electronics and design*. ACM, 2010, pp. 147–152.

[13] 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.

[14] C. Liu, "Design and analysis of approximate adders and multipliers," Master's thesis, University of Alberta, Canada, 2014.

[15] J. Liang, J. Han, and F. Lombardi, "New metrics for the reliability of approximate and probabilistic adders," *IEEE Transactions on Computers*, vol. 62, no. 9, pp. 1760–1771, 2013.

[16] M. S. Lau, K.-V. Ling, and Y.-C. Chu, "Energy-aware probabilistic multiplier: design and analysis," in *International Conference on Compilers, Architecture, and Synthesis for Embedded Systems*, 2009, pp. 281–290.