Characterizing Approximate Adders and Multipliers Optimized under Different Design Constraints

Honglan Jiang
Institute of Microelectronics
Tsinghua University
Beijing, China
honglan@ualberta.ca

Francisco J. H. Santiago
Mohammad Saeed Ansari
University of Alberta
Edmonton, AB, Canada
{fh,ansari2}@ualberta.ca

Leibo Liu
Institute of Microelectronics
Tsinghua University
Beijing, China
liulb@tsinghua.edu.cn

Bruce F. Cockburn
University of Alberta
Edmonton, AB, Canada
cockburn@ualberta.ca

Fabrizio Lombardi
Northeastern University
Boston, MA, USA
lombardi@ece.neu.edu

Jie Han
University of Alberta
Edmonton, AB, Canada
jhan8@ualberta.ca

ABSTRACT

Taking advantage of the error resilience in many applications as well as the perceptual limitations of humans, numerous approximate arithmetic circuits have been proposed that trade off accuracy for higher speed or lower power in emerging applications that exploit approximate computing. However, characterizing the various approximate designs for a specific application under certain performance constraints becomes a new challenge. In this paper, approximate adders and multipliers are evaluated and compared for a better understanding of their characteristics when the implementations are optimized for performance or power. Although simple truncation can effectively reduce the hardware of an arithmetic circuit, it is shown that some other designs perform better in speed, power and power-delay product. For instance, many approximate adders have a higher performance than a truncated adder. A truncated multiplier is faster but consumes a higher power than most approximate designs for achieving a similar mean error magnitude. The logarithmic multipliers are very fast and power-efficient at a lower accuracy. Approximate multipliers can also be generated by an automated process to be very efficient while ensuring a sufficiently high accuracy.

CCS CONCEPTS

• **General and reference** → **Surveys and overviews**; **Evaluation**; *Measurement*; • **Hardware** → **Arithmetic and datapath circuits**; *Combinational circuits*.

KEYWORDS

approximate computing, adder, multiplier, speed, power

ACM Reference Format:
Honglan Jiang, Francisco J. H. Santiago, Mohammad Saeed Ansari, Leibo Liu, Bruce F. Cockburn, Fabrizio Lombardi, and Jie Han. 2019. Characterizing Approximate Adders and Multipliers Optimized under Different Design Constraints. In *GLSVLSI '19: ACM Great Lakes Symposium on VLSI, May 09–11, 2019, Washington, DC.* ACM, New York, NY, USA, 6 pages. https://doi.org/10.1145/1122445.1122456

GLSVLSI '19, May 09–11, 2019, Washington, DC

ACM ISBN 978-1-4503-9999-9/18/06…$15.00
https://doi.org/10.1145/1122445.1122456

1 INTRODUCTION

As a potential technique for implementing complex computations with reasonable speed and power consumption, approximate computing has been explored actively from circuits to programming languages. Among circuit designs, adders and multipliers have been a focus because they play a pivotal role in determining the performance and power dissipation of many compute-intensive applications [8]. Therefore a large variety of approximate adders and multipliers have been proposed. However, these designs usually seek tradeoffs among accuracy, performance and power consumption, i.e., a design may be very fast but with a low accuracy or high power, or a very power-efficient design may have a significantly low speed or accuracy. Also, different designs in the literature have been evaluated using different synthesis tools and technologies. These differences have made it difficult to choose a suitable approximate design for a specific application with designated purposes.

An evaluation and comparison of the accuracy and circuit characteristics was provided in [11] for approximate adders, multipliers and dividers. However, the circuit measurements were obtained without considering any design constraints and, hence, the generic power-delay product (PDP) metric was mainly used for hardware evaluation. At a similar PDP, it was concluded that a truncated arithmetic circuit results in a smaller error magnitude (in the mean relative error distance (MRED)) than most of the other approximate designs.

In this paper, rather than considering the overall measurement, high performance and power efficiency are respectively pursued as independent design metrics. For example, an approximate design with a high speed is useful for coping with aging-induced timing errors in adders and multipliers [1]. Also, high-performance arithmetic circuits are preferred in real-time machine learning systems [10]. For mobile and embedded devices, power-efficient arithmetic circuits would be key to the extended use given limited battery life. In this work, therefore, the approximate circuits are optimized for maximizing performance (through delay) or minimizing power (through area). Compared to previous studies, several recent approximate arithmetic designs are included for comparison and to provide new insights to approximate arithmetic circuit design.

In the remainder of this paper, Section 2 introduces the simulation methodology. Sections 3 and 4 review and comparatively evaluate approximate adders and multipliers, respectively. Section 5 concludes the paper.

2 SIMULATION METHODOLOGY

The accuracy of the approximate designs is evaluated through Monte Carlo simulations. The error distance (ED = $|M' - M|$) and the relative error distance (RED = $|\frac{\text{ED}}{M}|$) are then calculated, where M' and M are the approximate and accurate results, respectively [18]. The mean error distance (MED) is calculated by averaging all of the obtained EDs. The error characteristics of the approximate designs are assessed by the error rate (ER, the probability of producing an incorrect result), the normalized MED (NMED, the normalization of MED by the maximum output of the accurate design) and the MRED (the average value of all of the obtained REDs).

To assess the circuit characteristics, the approximate designs are implemented in VHDL and synthesized using the Synopsys Design Compiler (DC) in STMicro's 28-nm CMOS technology, with a supply voltage of 1.0 V at a temperature of 25°C. For a fair comparison, all designs use the same process, voltage and temperature with the same optimization option. To compare speed and power, the approximate circuits are synthesized under different constraints. The critical path delay of a design is constrained to the smallest value without a timing violation for the delay-optimized synthesis, whereas the area is minimized for the area-optimized synthesis. The DesignWare library and "ultra compile" are used in the synthesis for optimization. The critical path delay and area are reported by the Synopsys DC. Power dissipation is measured by the PrimeTime-PX tool with 10 million random input combinations.

3 APPROXIMATE ADDERS

Two basic adders for binary addition are the ripple-carry adder (RCA) and the carry lookahead adder (CLA). For an n-bit RCA, the delay and circuit complexity increase proportionally with n (denoted by $O(n)$). The delay of an n-bit CLA is logarithmic in n (or $O(log(n))$), thus significantly shorter than that of an n-bit RCA. However, a CLA requires a larger circuit area (in $O(nlog(n))$), which results in a higher power dissipation.

To reduce the critical path delay of an accurate adder, many approximation methodologies have been proposed, including involving speculative operation, segmentation, carry selection and an approximate full adder.

3.1 Review

In this section approximate adders are briefly introduced. Please refer to [11] for a detailed introduction to these designs.

3.1.1 Speculative Adders. In an n-bit speculative adder, k ($k < n$) least significant bits (LSBs) are used to predict the carry input for each sum bit [22]. Thus, the critical path delay is reduced to $O(log(k))$ (for a parallel implementation such as the CLA, the same throughout this section unless otherwise noted). The almost correct adder (ACA) is designed to reduce the hardware overhead by sharing part of the carry-generation circuits [32].

3.1.2 Segmented Adders. A segmented adder is implemented by using several smaller adders operating in parallel. They include the equal segmentation adder (ESA) [26], the error-tolerant adder type II (ETAII) [36], the accuracy-configurable approximate adder (ACAA) [13] and the dithering adder [24]. The delays of the segmented adders grow with $O(log(k))$ and the circuit complexities grow with $O(nlog(k))$ for ESA and ETAII, and with $O((n-k)log(k))$ for ACAA [11].

3.1.3 Carry Select Adders. An n-bit carry select adder consists of $m = \lceil \frac{n}{k} \rceil$ blocks, and the carry input for each block is selected using different schemes in different designs. In the speculative carry select adder (SCSA), each block is made of two k-bit adders: adder0 with carry-in "0" and adder1 with carry-in "1"; the sum is selected by a multiplexer according to the carry-out of adder0 in the previous block [6]. SCSA and ETAII achieve the same accuracy for the same value of k due to the same carry prediction function. For the carry skip adder (CSA), each block consists of a sub-carry generator and a sub-adder [14]. The carry-in of the $(i+1)^{th}$ sub-adder is determined by the propagate signals of the i^{th} block, which enhances the carry prediction accuracy. In the carry speculative adder (CSPA), each block contains one sum generator, two internal carry generators and one carry predictor; fewer than k input bits are used in a carry predictor [20]. In the consistent carry approximate adder (CCA), the carry prediction depends not only on its LSBs, but also on the higher bits [17]. The generate signals-exploited carry speculation adder (GCSA) has a similar structure as CSA and uses the generate signals for carry speculation [9]. In the gracefully-degrading accuracy-configurable adder (GDA), control signals are used to configure the accuracy by selecting an accurate or approximate carry-in signal using a multiplexer for each sub-adder [34]. In the carry cut-back adder (CCBA), the full carry propagation is prevented by a controlled multiplexer or an OR gate for a high-speed operation [5]. Although the critical path delays of the carry select adders vary, they generally grow with $O(log(k))$, where k is the size of the sub-adder.

3.1.4 Approximate Full Adders. In this class of designs, the LSBs are implemented by approximate full adders. These adders include the simple use of OR gates (and one AND gate for carry propagation) in the lower-part-OR adder (LOA) [23], the approximate designs of the mirror adder [7] and the approximate XOR/XNOR-based full adders [33]. For the LOA, the critical path is approximately $O(log(n-l))$, where l is the number of approximate LSBs. Finally, an adder whose k LSBs are truncated is referred to as a truncated adder (TruA) that works with a lower precision. It is considered as a baseline design.

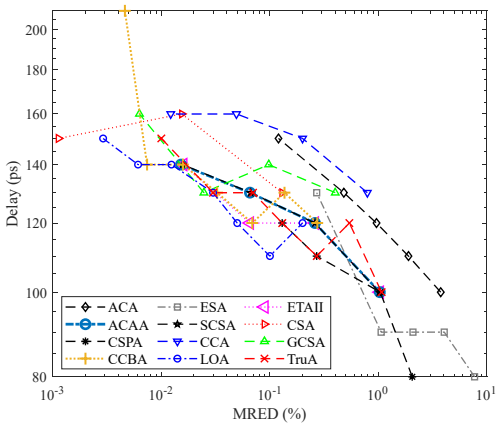
3.2 Evaluation

In this evaluation, 16-bit approximate adders are considered. For CSPA, the size of the carry predictor is $\lceil k/2 \rceil$. The global speculative carry for CCA is "0". All sub-adders are implemented as CLAs because most approximate adders are designed based on the CLA.

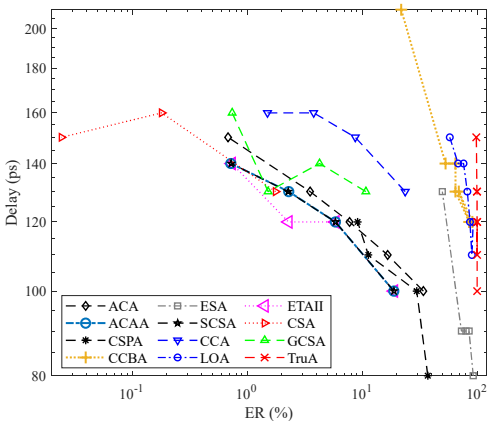
The simulation results show similar performance trends for the MRED and NMED of the approximate adders, so only the MRED is considered in the comparison. Figs. 1, 2 and 3 show the comparison of MRED, ER, delay (for delay-optimized synthesis), power (for area-optimized synthesis) and PDP.

Fig. 1 shows that, among the adders with small MREDs, LOA and ETAII are faster than the other designs, whereas CCA is the slowest followed by CSA. CSA-6 is not shown in the figures because it is accurate due to the precise carry generated for every block, so the ER and MRED of CSA-6 are 0. For a high MRED, ESA and CSPA are faster. When the same ER is considered, ETAII, SCSA and ACAA are among the fastest designs.

Fig. 2 shows that in terms of power consumption, CCBA, LOA and TruA are the most efficient designs, while ACA is very power consuming. However, CCBA, LOA and TruA have very high ERs. CSA has a rather low ER. For a similar ER, ETAII and

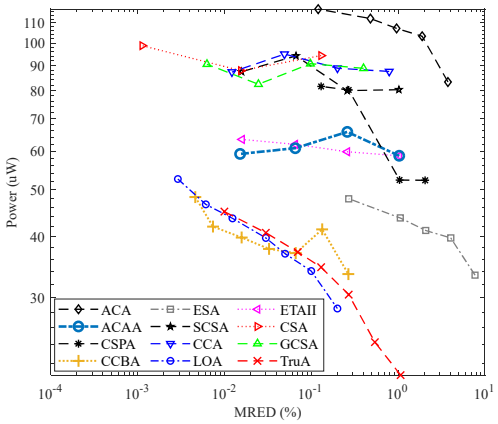


(a) MRED vs. Delay (delay-optimized)

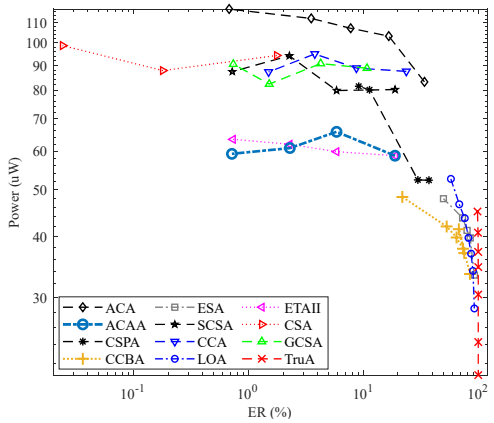


(b) ER vs. Delay (delay-optimized)

Figure 1: **A comparison of optimized delay for the approximate 16-bit adders using different error metrics. The parameter k for LOA and TruA ranges from 3 to 9 from left to right, from 8 down to 3 for ESA and ACA, and from 6 down to 3 for the other adders from left to right. For CCBA, the configurations with the smallest PDPs are chosen for a similar MRED.**



(a) MRED vs. Power (area-optimized)



(b) ER vs. Power (area-optimized)

Figure 2: **A comparison of power for the area-optimized approximate 16-bit adders using different error metrics.**

ACAA are very power-efficient, while ACA consumes relatively high power.

As shown in Fig. 3, CCBA, LOA and TruA have smaller PDPs than the other designs for a similar MRED; ACA and CCA have larger PDPs. ETAII, SCSA, CSPA and GCSA show moderate PDPs. ESA has a rather small PDP, but a considerably larger ER and MRED. With the highest ERs, CCBA, LOA and TruA show the smallest PDPs for a moderate MRED. In fact, these approximate adders show a decent tradeoff in ED and hardware efficiency. In particular, they are useful in applications in which ER is not important. Consequently, truncation can be used to design a hardware-efficient approximate adder (albeit with a high ER). On the other hand, the carry select scheme is more effective for an approximate adder design to achieve a high accuracy.

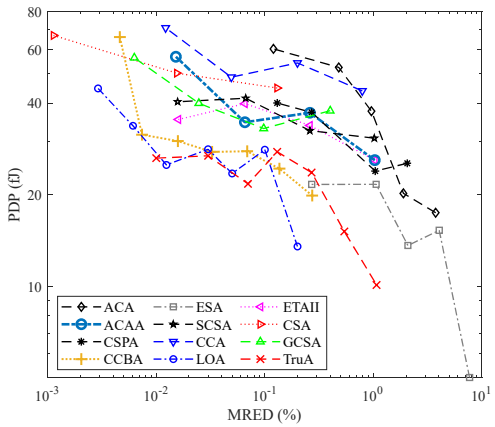
Table 1 summarizes the different approximate adders, with their advantages and disadvantages highlighted (i.e., the metrics with moderate values are not shown). The MRED and NMED of approximate adders are represented by ED in the table.

Table 1: Summary of approximate adders.

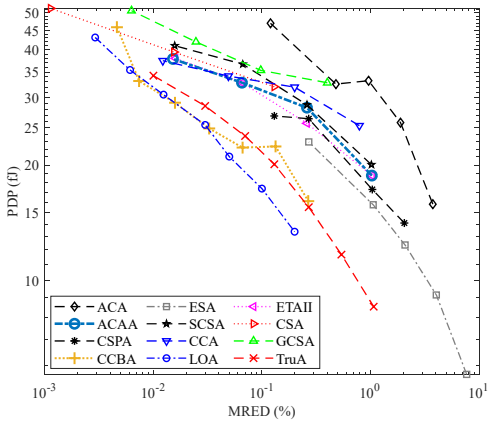
Adder	ER	ED	speed (delay-optimized)	power (area-optimized)	PDP
ACA		high		high	high
ESA	high	high	high	low	low
ETAII					
ACAA					
SCSA					
CSA	low	low			
CSPA		high	high		
CCA			low		
GCSA					
CCBA	high	low		low	low
LOA	high	low		low	low
TruA	high			low	low

4 APPROXIMATE MULTIPLIERS

Multiplication is implemented by partial product generation, accumulation and a final addition. A partial product is usually generated by an AND gate. The most common partial product accumulation structures are the Wallace and Dadda trees and the carry-save adder array. In a Wallace tree, the FAs (or half adders (HAs)) in each layer operate in parallel without carry propagation. Thus, for an $n \times n$ multiplier, the delay of the Wallace tree grows with $O(log(n))$. Also, the FAs in a Wallace tree can be considered to be (3:2) compressors 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. In a carry-save adder array, the carry and sum signals generated by the FAs (or HAs) in a row are passed on to the FAs in the next row. FAs in a column operate in series. Hence the delay of an $n \times n$ carry-save adder array is approximately $O(n)$, longer than that of a Wallace tree.



(a) MRED vs. PDP (delay-optimized)



(b) MRED vs. PDP (area-optimized)

Figure 3: **A comparison of power-delay product (PDP) and MRED for the approximate 16-bit adders.**

Five main methodologies are used for approximating an unsigned multiplier: approximation in generating the partial products, approximation (including truncation) in the partial product tree, using approximate adders, counters or compressors in the partial product tree, using logarithmic approximation, and using an automated process such as a genetic programming method.

4.1 Review

4.1.1 Approximation in Generating Partial Products. In [15], the accurate 2×2 multiplication result "1001" is simplified to "111" to save one output bit when both the inputs are "11", which results in an approximate 2×2 multiplier. Larger approximate multipliers are then designed by using the approximate 2×2 multiplier, which is referred to as the underdesigned multiplier (UDM). UDM introduces an error when generating the partial products, however the accumulation remains accurate.

4.1.2 Approximation in the Partial Product Tree. The broken-array multiplier (BAM) omits some carry-save adders in an array multiplier in both the horizontal and vertical directions [23]. A more straightforward approach is to truncate some LSBs on the input operands and, thus, a smaller multiplier is used to process the remaining most significant bits (MSBs). This truncated multiplier (TruM) will be considered as a baseline design. In [35], partial product perforation is applied to different multiplier structures (PPAM), which ignores several consecutive rows of partial products (not necessarily starting from the LSB). The error tolerant multiplier (ETM) consists of a multiplication section and a non-multiplication section [16]. A control block is used to decide if the multiplication section is to be activated to multiply the LSBs or the MSBs. In the static segment multiplier (SSM), no approximation is applied to the LSBs; either the MSBs or the LSBs of the operands are accurately multiplied depending on whether its MSBs are all zeros [30]. In [4], a bit-width aware approximate multiplication and a carry-in prediction scheme are used to construct an approximate Wallace tree multiplier (AWTM).

4.1.3 Using Approximate Counters or Compressors in the Partial Product Tree. An approximate (4:2) counter is proposed for an inaccurate 4×4 Wallace multiplier [19]. The carry and sum are approximated as "10" for "100" in the approximate counter when all input signals are "1." The inaccurate 4×4 multiplier is then used to construct a larger multiplier that is referred to as ICM. In [27], two approximate (4:2) compressor designs are used in a Dadda multiplier with four different schemes. In this paper, the more accurate schemes 3 and 4 of the approximate compressor based multiplier (referred to as ACM-3 and ACM-4) are considered for comparison. An approximate 4:2 compressor with encoded inputs and improved accuracy is proposed for 4×4 multipliers that are used to build larger multipliers [3] (referred to as M16). A novel approximate adder is proposed to accumulate the partial products for the approximate multiplier with configurable error recovery [12]. Two approximate error accumulation schemes are proposed to compensate the error generated by the approximate adder. In scheme 1 (AM1), errors are accumulated by using OR gates, while both OR gates and the approximate adders are used in scheme 2 (AM2). The truncation of half LSBs in the partial products in AM1 and AM2 results in TAM1 and TAM2, respectively. In [31], approximate half adders, full adders and 4:2 compressors are utilized to accumulate the altered partial products; it is denoted as the USask design.

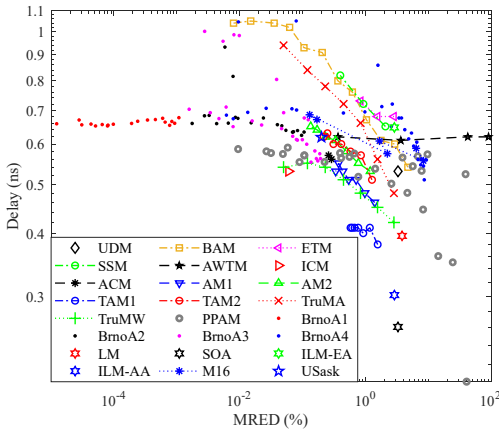
4.1.4 Using Logarithmic Approximation. By using the logarithm and anti-logarithm approximations of a binary number, [25] proposes a logarithmic multiplier (LM) implemented by shifting and addition; it is the baseline design for LMs. The accuracy of a LM is improved in [21] by using a set-one-adder (SOA) and in [2] by using an improved algorithm with exact and approximate adders (ILM-EA and ILM-AA).

4.1.5 Using an Automated Process. In [28], 471 8×8 approximate multipliers are evolved by a multi-objective Cartesian genetic programming method, which are then used to construct 16×16 approximate multipliers [29] (i.e., BrnoA1, BrnoA2, BrnoA3 and BrnoA4 for different construction methods).

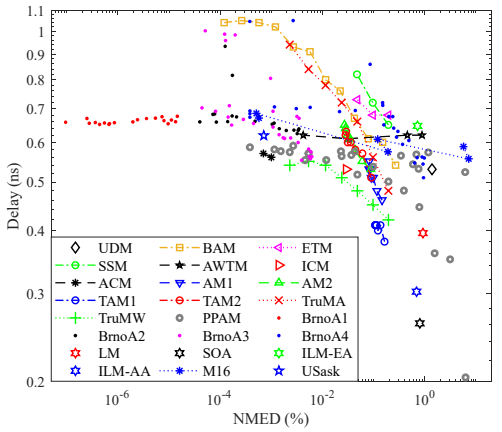
4.2 Evaluation

Monte Carlo simulation results for 16×16 multipliers show that most of the approximate designs result in large ERs close to 100%. However, ICM has a low ER of 5.45%; some configurations of BrnoA1, BrnoA2, BrnoA3 and BrnoA4 also show lower ERs than the other designs. Thus, MRED, NMED, delay (for delay-optimized synthesis), power (for area-optimized synthesis) and PDP (for both delay- and area-optimized syntheses) are jointly considered to evaluate the approximate multipliers, as shown in Figs. 4, 5 and 6.

Fig. 4 shows that BrnoA1 is the most accurate design with very small values of MRED and NMED because only one approximate 8×8 approximate multiplier is used to construct a 16×16 BrnoA1. ICM and the truncated Wallace multiplier (TruMW) are faster than the other designs for a low MRED and NMED, whereas TAM1, ILM-AA, SOA, PPAM are the fastest while the MRED

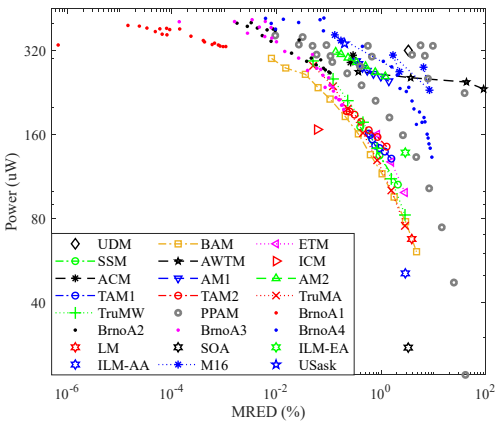


(a) MRED vs. Delay (delay-optimized)

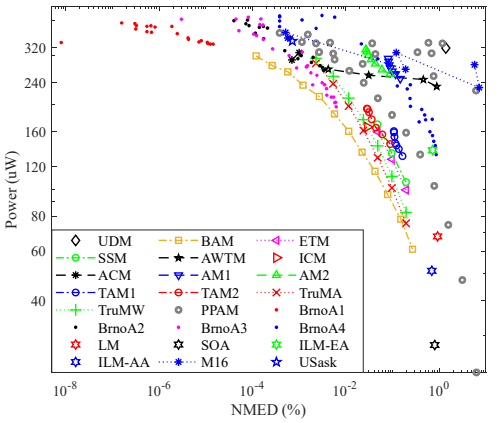


(b) NMED vs. Delay (delay-optimized)

Figure 4: **A comparison of optimized delay for the approximate 16×16 multipliers using different error metrics. The number of truncated LSBs for TruMA and TruMW is from 2 to 8 from left to right, and from 11 to 22 for BAM. The number of MSBs used for error compensation is from 16 to 10 for AM1, AM2, TAM1 and TAM2. The size of the accurate sub-multiplier is from 10 to 8 for ETM and SSM. The mode number is from 4 to 1 for AWTM, from 4 to 3 for ACM, and from 1 to 6 for M16. For BrnoA1, BrnoA2, BrnoA3 and BrnoA4, the configurations with the smallest PDPs are shown for a specific MRED, selected from 500 configurations for each design.**



(a) MRED vs. Power (area-optimized)



(b) NMED vs. Power (area-optimized)

Figure 5: **A comparison of power for the area-optimized approximate 16×16 multipliers considering MRED and NMED.**

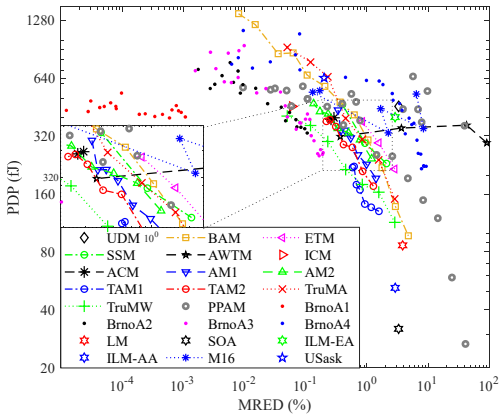
and NMED are higher. AM1 is also very fast; however, BAM, SSM, ETM and the truncated array multiplier (TruMA) are relatively slow. As shown in Fig. 5, BAM is the most power-efficient followed by TruMA and BrnoA3, while UDM, AM1, AM2, M16 and BrnoA4 are relatively power-hungry. The power dissipation of TAM1 and TAM2 is in the medium range.

As shown in the delay-optimized results in Fig. 6(a), TAM1, BrnoA2 and BrnoA3 have smaller PDPs than other approximate designs (including TruMW) for a similar MRED. For the area-optimized results in Fig. 6(b), ICM, TAM1, TAM2 and BrnoA3 show smaller PDPs than other designs (including TruMA).

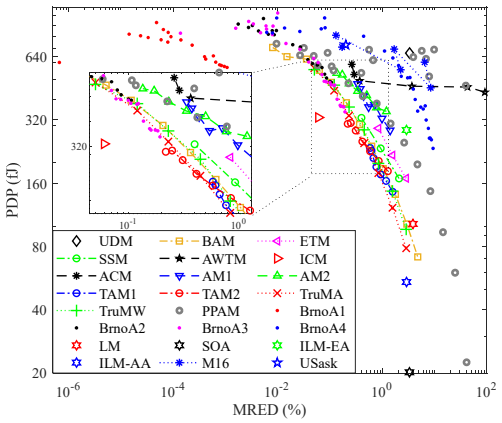
A summary of the accuracy and circuit characteristics for the approximate multipliers is shown in Table 2. Truncation is an effective scheme for a high-speed operation with a relatively low PDP. By using the logarithmic approximation, a multiplier tends to be hardware-efficient but with relatively low accuracy. The approximate multipliers designed using the automated process perform well when a high accuracy is required.

Table 2: Summary of approximate multipliers.

Multiplier	MRED	NMED	speed (delay-optimized)	power (area-optimized)	PDP
UDM	high	high		high	high
BAM			low	low	
PPAM					
ETM			low		
SSM			low		
AWTM					
ICM	low				
ACM		low		high	
M16				high	high
AM1				high	
AM2				high	
TAM1			high		low
TAM2					
USask		low		high	high
BrnoA1	low	low		high	
BrnoA2					
BrnoA3					
BrnoA4					
LM	high	high	high	low	
SOA	high	high	high	low	low
ILM-EA	high	high			
ILM-AA	high	high	high	low	low
TruMA					low
TruMW			high		



(a) MRED vs. PDP (delay-optimized)



(b) MRED vs. PDP (area-optimized)

Figure 6: **A comparison of power-delay product and MRED for the approximate 16×16 unsigned multipliers.**

5 CONCLUSION

In this paper, designs of approximate adders and multipliers are evaluated in terms of accuracy and circuit characteristics. Specifically, the performance and power consumption are compared by obtaining the synthesis results under delay and area constraints.

The simulation results show that truncation is an effective scheme to improve the hardware efficiency of an arithmetic circuit. Most of the approximate adders in the literature are designed for a high-speed and low ER (e.g., 0.02% for CSA-5) by cutting off the carry propagation chain. A truncated adder has a high ER close to 100%; however, it has a lower power dissipation and lower PDP than most approximate designs at a similar MRED (except for LOA and CCBA). The performance (or speed) of a truncated adder is not as high as some approximate adders.

Unlike the adder, a truncated Wallace multiplier is faster than most approximate multipliers (except for ICM and TAM1), but with a higher power at a similar MRED. For a similar accuracy, BAM and BrnoA3 consume less power than a truncated multiplier. In terms of PDP, TAM1, TAM2, ICM, BrnoA2 and BrnoA3 have smaller values than a truncated multiplier for a given MRED.

REFERENCES

[1] H. Amrouch, B. Khaleghi, and A. Gerstlauer. 2017. Towards aging-induced approximations. In *DAC*. 1–6.

[2] M.S. Ansari, B.F. Cockburn, and J. Han. 2019. A hardware-efficient logarithmic multiplier with improved accuracy. In *DATE*. 1–4.

[3] M.S. Ansari, H. Jiang, B.F. Cockburn, and J. Han. 2018. Low-power approximate multipliers using encoded partial products and approximate compressors. *JETCAS* 8, 3, 404–416.

[4] K. Bhardwaj, P.S. Mane, and J. Henkel. 2014. Power- and area-efficient approximate Wallace tree multiplier for error-resilient systems. In *ISQED*. 263–269.

[5] V. Camus, M. Cacciotti, J. Schlachter, and C. Enz. 2018. Design of approximate circuits by fabrication of false timing paths: the carry cut-back adder. *JETCAS* 8, 4, 746–757.

[6] Kai D., P. Varman, and K. Mohanram. 2012. High performance reliable variable latency carry select addition. In *DATE*. 1257–1262.

[7] V. Gupta, D. Mohapatra, A. Raghunathan, and K. Roy. 2013. Low-power digital signal processing using approximate adders. *TCAD* 32, 1, 124–137.

[8] J. Han and M. Orshansky. 2013. Approximate computing: an emerging paradigm for energy-efficient design. In *ETS*. 1–6.

[9] J. Hu and W. Qian. 2015. A new approximate adder with low relative error and correct sign calculation. In *DATE*. 1449–1454.

[10] G.B. Huang, Q.Y. Zhu, and C.K. Siew. 2006. Real-time learning capability of neural networks. *IEEE Trans. Neural Networks* 17, 4, 863–878.

[11] H. Jiang, C. Liu, L. Liu, F. Lombardi, and J. Han. 2017. A review, classification, and comparative evaluation of approximate arithmetic circuits. *JETC* 13, 4, 60.

[12] H. Jiang, C. Liu, F. Lombardi, and J. Han. 2019. Low-power approximate unsigned multipliers with configurable error recovery. *TCASI* 66, 1, 189–202.

[13] A.B. Kahng and S. Kang. 2012. Accuracy-configurable adder for approximate arithmetic designs. In *DAC*. 820–825.

[14] Y. Kim, Y. Zhang, and P. Li. 2013. An energy efficient approximate adder with carry skip for error resilient neuromorphic VLSI systems. In *ICCAD*. 130–137.

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

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

[17] L. Li and H. Zhou. 2014. On error modeling and analysis of approximate adders. In *ICCAD*. 511–518.

[18] J. Liang, J. Han, and F. Lombardi. 2013. New metrics for the reliability of approximate and probabilistic adders. *TC* 62, 9, 1760–1771.

[19] C. Lin and I. Lin. 2013. High accuracy approximate multiplier with error correction. In *ICCD*. 33–38.

[20] I. Lin, Y. Yang, and C. Lin. 2015. High-performance low-power carry speculative addition with variable latency. *TVLSI* 23, 9, 1591–1603.

[21] W. Liu, J. Xu, D. Wang, C. Wang, P. Montuschi, and F. Lombardi. 2018. Design and evaluation of approximate logarithmic multipliers for low power error-tolerant applications. *TCASI* 65, 9, 2856–2868.

[22] S.L. Lu. 2004. Speeding up processing with approximation circuits. *Computer* 37, 3, 67–73.

[23] H.R. Mahdiani, A. Ahmadi, S.M. Fakhraie, and C. Lucas. 2010. Bio-inspired imprecise computational blocks for efficient VLSI implementation of soft-computing applications. *TCASI* 57, 4, 850–862.

[24] J. Miao, K. He, A. Gerstlauer, and M. Orshansky. 2012. Modeling and synthesis of quality-energy optimal approximate adders. In *ICCAD*. 728–735.

[25] J.N. Mitchell. 1962. Computer multiplication and division using binary logarithms. *IRE Trans. Electronic Computers* 4, 512–517.

[26] D. Mohapatra, V.K. Chippa, A Raghunathan, and K. Roy. 2011. Design of voltage-scalable meta-functions for approximate computing. In *DATE*. 1–6.

[27] A. Momeni, J. Han, P. Montuschi, and F. Lombardi. 2015. Design and analysis of approximate compressors for multiplication. *TC* 64, 4, 984–994.

[28] V. Mrazek, R. Hrbacek, Z. Vasicek, and L. Sekanina. 2017. EvoApprox8b: Library of approximate adders and multipliers for circuit design and benchmarking of approximation methods. In *DATE*. 258–261.

[29] V. Mrazek, Z. Vasicek, L. Sekanina, H. Jiang, and J. Han. 2018. Scalable construction of approximate multipliers with formally guaranteed worst case error. *TVLSI* 99, 2572–2576.

[30] S. Narayanamoorthy, H.A. Moghaddam, Z. Liu, T. Park, and N.S. Kim. 2015. Energy-efficient approximate multiplication for digital signal processing and classification applications. *TVLSI* 23, 6, 1180–1184.

[31] S. Venkatachalam and S.B. Ko. 2017. Design of power and area efficient approximate multipliers. *TVLSI* 25, 5, 1782–1786.

[32] A.K. Verma, P. Brisk, and P. Ienne. 2008. Variable latency speculative addition: a new paradigm for arithmetic circuit design. In *DATE*. 1250–1255.

[33] Z. Yang, A. Jain, J. Liang, J. Han, and F. Lombardi. 2013. Approximate XOR/XNOR-based adders for inexact computing. In *IEEE-NANO*. 690–693.

[34] R. Ye, T. Wang, F. Yuan, R. Kumar, and Q. Xu. 2013. On reconfiguration-oriented approximate adder design and its application. In *ICCAD*. 48–54.

[35] G. Zervakis, K. Tsoumanis, S. Xydis, D. Soudris, and K. Pekmestzi. 2016. Design-efficient approximate multiplication circuits through partial product perforation. *TVLSI* 24, 10, 3105–3117.

[36] N. Zhu, W.L. Goh, and K.S. Yeo. 2009. An enhanced low-power high-speed adder for error-tolerant application. In *ISIC 2009*. 69–72.