A Review, Classification, and Comparative Evaluation of Approximate Arithmetic Circuits

HONGLAN JIANG and CONG LIU, University of Alberta
LEIBO LIU, Tsinghua University
FABRIZIO LOMBARDI, Northeastern University
JIE HAN, University of Alberta

Often as the most important arithmetic modules in a processor, adders, multipliers, and dividers determine the performance and energy efficiency of many computing tasks. The demand of higher speed and power efficiency, as well as the feature of error resilience in many applications (e.g., multimedia, recognition, and data analytics), have driven the development of approximate arithmetic design. In this article, a review and classification are presented for the current designs of approximate arithmetic circuits including adders, multipliers, and dividers. A comprehensive and comparative evaluation of their error and circuit characteristics is performed for understanding the features of various designs. By using approximate multipliers and adders, the circuit for an image processing application consumes as little as 47% of the power and 36% of the power-delay product of an accurate design while achieving similar image processing quality. Improvements in delay, power, and area are obtained for the detection of differences in images by using approximate dividers.

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

Additional Key Words and Phrases: Approximate computing, approximate circuit, adder, multiplier, divider, hardware, accuracy, image processing

ACM Reference Format:
Honglan Jiang, Cong Liu, Leibo Liu, Fabrizio Lombardi, and Jie Han. 2017. A review, classification, and comparative evaluation of approximate arithmetic circuits. J. Emerg. Technol. Comput. Syst. 13, 4, Article 60 (August 2017), 34 pages.
DOI: http://dx.doi.org/10.1145/3094124

1. INTRODUCTION

Although computational errors generally are not desirable, applications such as multimedia, wireless communication, recognition, and data mining are tolerant of the occurrence of some errors [Han and Orshansky 2013]. Due to the perceptual limitations of humans, these errors do not make an obvious difference in applications such as image, audio, and video processing. Moreover, in many digital signal processing

This work was partly supported by the University of Alberta and the Natural Sciences and Engineering Research Council (NSERC) of Canada.
Authors' addresses: H. Jiang, C. Liu, and J. Han, Department of Electrical and Computer Engineering, University of Alberta, Edmonton, AB T6G 1H9, Canada; emails: {honglan, cong4, jhan8}@ualberta.ca; L. Liu, Institute of Microelectronics, Tsinghua University, Beijing 100084, China; email: liulb@tsinghua.edu.cn; F. Lombardi, Department of Electrical and Computer Engineering, Northeastern University, Boston, MA; email: lombardi@ece.neu.edu.

DOI: http://dx.doi.org/10.1145/3094124

(DSP) systems, inputs from the outside world are noisy, so there is a limit in precision or accuracy in the computed results. Many applications are based on statistical or probabilistic computation, such as classification and recognition algorithms. Due to the nature of these applications, trivial errors in computation do not result in significant performance degradation. Therefore, approximate computing is applicable in many applications that can tolerate the loss of certain accuracy [Venkatesan et al. 2010].

Past research on approximate computing has spanned from circuits to programming languages [Han 2016]. In Datla et al. [2009], an approximate squaring circuit is proposed. A new logic synthesis approach is introduced to reduce the area of a synthesized circuit for a given threshold of error rate in Shin [2010]. In Chippa et al. [2010], a scalable effort design approach is proposed to implement highly efficient hardware for error-resilient applications. Automated design processes have been proposed for approximate digital circuit design using Cartesian genetic programming [Vasicek and Sekanina 2015; Mrazek et al. 2016]. Sampson et al. [2011] developed the EnerJ language, an extension to Java. This language supports approximate data types for low-power computation. Various computing and memory architectures have been proposed for supporting approximate computing applications [Esmaeilzadeh et al. 2012; Miguel et al. 2015]. In this article, we focus on approximate circuit designs and particularly approximate arithmetic circuits of adders, multipliers, and dividers.

Design metrics and analytical approaches have been proposed for the evaluation of approximate adders [Liang et al. 2013; Huang et al. 2012; Miao et al. 2012; Venkatesan et al. 2010; Liu et al. 2015; Mazahir et al. 2017]. Monte Carlo simulation has been employed to acquire data for analysis. In this article, the error rate (ER), the error distance (ED), and the average error are used to evaluate the error characteristics of the approximate designs. Hardware-related figures of merit including critical path delay, circuit area, and power dissipation, as well as compound metrics including the power-delay product (PDP) and area-delay product (ADP), are utilized to assess the circuit characteristics of these designs.

Image processing has been essential in diverse applications including multimedia, biomedical imaging, and pattern recognition [Acharya and Ray 2005]. Taking advantage of its inherent error resilience, image processing can be efficiently implemented by using approximate arithmetic circuits. Therefore, image sharpening and change detection are considered for further evaluation of the approximate circuits in addition to the evaluation using design metrics. The simulation results show that the image sharpening circuit using approximate adders and multipliers saves as much as 53% of power and 58% of area compared to an accurate design with a similar accuracy. The change detection circuit using approximate dividers achieves as much as 40% improvement in speed and 25% improvement in power compared to an accurate design at a similar accuracy. Although some preliminary results have been presented in Jiang et al. [2015, 2016b], this article makes the following new contributions:

(1) A larger set of approximate adders that include more recent designs are considered in this work. Moreover, a new error metric (average error) is used to measure the output bias of an approximate design, which is very important in an accumulative operation.
(2) Approximate Booth multipliers for signed multiplication are included in the evaluation, whereas only unsigned approximate multipliers are considered in Jiang et al. [2016b].
(3) Current approximate dividers are reviewed, and their features are discussed with respect to performance, accuracy, and hardware consumption.
(4) The STM CMOS 28nm process is used in the circuit synthesis throughout this work, whereas an older 65nm technology was used in Jiang et al. [2015].

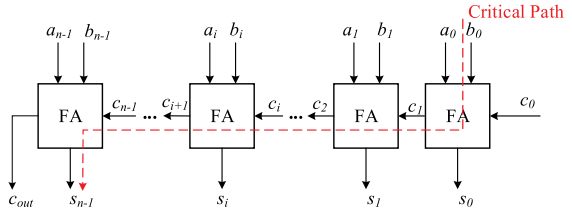


Fig. 1. The n-bit RCA. FA, a 1-bit full adder.

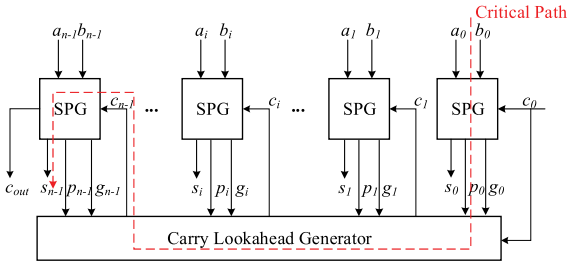


Fig. 2. The n-bit CLA. SPG, the cell used to produce the sum, generate ($g_i = a_i b_i$), and propagate ($p_i = a_i + b_i$) signals.

(5) The considered approximate arithmetic circuits of adders, multipliers, and dividers are applied to two image processing applications for further evaluation. The accuracy and circuit characteristics are obtained by simulation and synthesis.

2. APPROXIMATE ADDERS

An adder performs the addition of two binary numbers. Two basic adders are the ripple-carry adder (RCA) (Figure 1) and the carry lookahead adder (CLA) (Figure 2). In an n-bit RCA, the carry of each full adder is propagated to the next full adder, and thus the delay and circuit complexity increase proportionally with n (or $O(n)$). An n-bit CLA consists of n units that operate in parallel to produce the sum and the generate ($g_i = a_i b_i$) and propagate ($p_i = a_i + b_i$) signals for generating the lookahead carries. The delay of CLA is logarithmic in n (or $O(log(n))$), significantly shorter than for RCA. However, a CLA requires a larger circuit area (in $O(nlog(n))$), thus incurring a higher power dissipation.

Many approximation schemes have been proposed by reducing the critical path and hardware complexity of an accurate adder. An early methodology is based on a speculative operation [Lu 2004; Verma et al. 2008]. In an n-bit speculative adder, each sum bit is predicted by its previous k less significant bits (LSBs) ($k < n$). As the carry chain is shorter than n, a speculative adder is faster than a conventional design. A segmented adder is implemented by several smaller adders operating in parallel [Mohapatra et al. 2011; Zhu et al. 2009; Kahng and Kang 2012; Yang et al. 2016]. Hence, the carry propagation chain is truncated into shorter segments. Segmentation is also utilized in numerous works [Du et al. 2012; Kim et al. 2013; Ye et al. 2013; Lin et al. 2015; LI and Zhou 2014; Hu and Quian 2015; Miao et al. 2012; Camus et al. 2016], but the carry input for each subadder is selected differently. This type of adder is referred to as a carry select adder. Another method for reducing the critical path delay and power dissipation is by approximating a full adder [Mahdiani et al. 2010; Gupta et al. 2013; Yang et al. 2013; Cai et al. 2016; Angizi et al. 2017]. The approximate full adder is then used to implement the LSBs in an accurate adder. Thus, approximate adders are divided into four categories, as briefly summarized next.

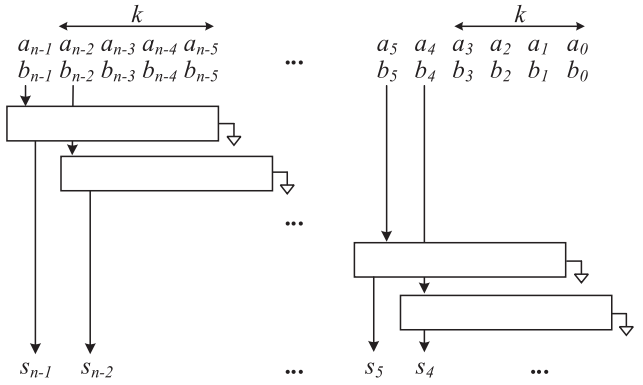


Fig. 3. The ACA. ▭, the carry propagation path of the sum.

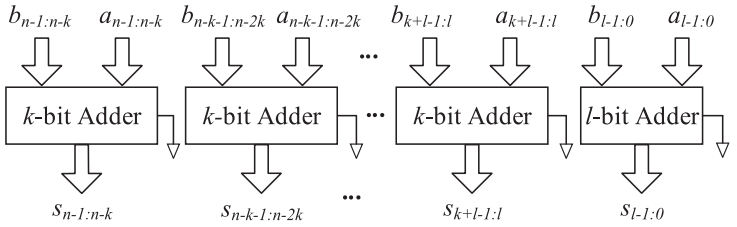


Fig. 4. The ESA. k, the maximum carry chain length; l, the size of the first subadder ($l \leq k$).

2.1. Classification of Approximate Adders

2.1.1. Speculative Adders. The almost correct adder (ACA) [Verma et al. 2008] is based on the speculative adder design of Lu [2004]. In an n-bit ACA, k LSBs are used to predict the carry for each sum bit ($n > k$), as shown in Figure 3. Therefore, the critical path delay is reduced to $O(log(k))$ (for a parallel implementation such as CLA, the same in the following). The design in Lu [2004] requires $(n - k)$ k-bit subcarry generators in an n-bit adder, and thus the hardware consumption is rather high (in $O((n-k)klog(k))$). This overhead is reduced in Verma et al. [2008] by sharing some components among the subcarry generators.

2.1.2. Segmented Adders. The equal segmentation adder (ESA) divides an n-bit adder into a number of smaller k-bit subadders operating in parallel with fixed carry inputs, so no carry is propagated among the subadders (Figure 4) [Mohapatra et al. 2011]. The delay of ESA is $O(log(k))$ and the circuit complexity is $O(nlog(k))$. Its hardware overhead is significantly lower than ACA.

The error-tolerant adder type II (ETAII) consists of parallel carry generators and sum generators [Zhu et al. 2009], as shown in Figure 5. The carry signal from the previous carry generator propagates to the next sum generator. Therefore, ETAII utilizes more information to predict the carry and thus is more accurate than ESA for the same k. The circuit of ETAII is more complex than that of ESA, whereas its delay is larger due to the longer critical path ($2k$).

In an n-bit accuracy-configurable approximate adder (ACAA), $\lceil \frac{n}{k} - 1 \rceil$ $2k$-bit subadders are required [Kahng and Kang 2012]. Each subadder adds $2k$ consecutive bits with an overlap of k bits, and all $2k$-bit subadders operate in parallel to reduce the delay to $O(log(k))$. In each subadder, half of the most significant sum bits is selected as the partial sum. The accuracy of ACAA can be configured at runtime. Moreover, ACAA has

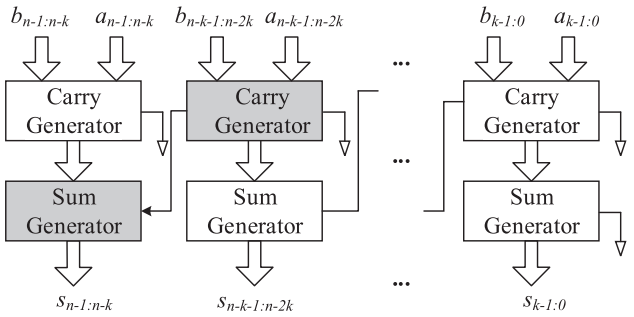


Fig. 5. The ETAII [Zhu et al. 2009]: the carry propagates through the two shaded blocks.

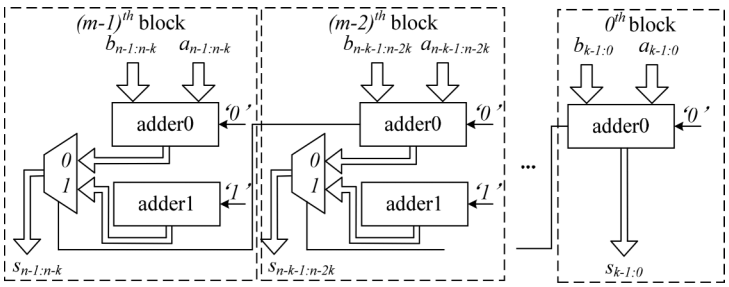


Fig. 6. The SCSA.

the same carry propagation path as ETAII for each sum, so they are equally accurate for the same k.

The dithering adder divides an adder into an accurate, more significant subadder and a less significant subadder with upper and lower bounding modules [Miao et al. 2012]. The output of the less significant subadder is conditionally selected. An effective "Dither Control" enables a smaller variance in the overall error.

To reduce the error distance, an error control and compensation method is proposed for a segmented adder in Yang et al. [2016]. This method employs a multistage latency to compensate the carry prediction error in a more significant segmentation, thus trading off computing efficiency for an improved accuracy.

The delays of the segmented adders are $O(log(k))$ and the circuit complexities are $O(nlog(k))$ for ESA and ETAII, and $O((n-k)log(k))$ for ACAA.

2.1.3. Carry Select Adders. In a carry select adder, several signals are commonly used. For the i^{th} block, generate $g_{i,j} = a_{i,j}b_{i,j}$, propagate $p_{i,j} = a_{i,j} \oplus b_{i,j}$, and $P_i = \prod_{j=0}^{k-1} p_{i,j}$, where $a_{i,j}$ and $b_{i,j}$ are the j^{th} LSBs of the input operands. $P_i = 1$ indicates that all k propagate signals in the i^{th} block are true.

An n-bit speculative carry select adder (SCSA) consists of $m = \lceil \frac{n}{k} \rceil$ subadders (or window adders) [Du et al. 2012]. Each subadder is made of two k-bit adders: adder0 with carry-in "0" and adder1 with carry-in "1." The carry-out of adder0 is connected to a multiplexer to select the addition result as part of the final result, as shown in Figure 6. SCSA and ETAII achieve the same accuracy for the same value of k due to the same carry predict function, whereas SCSA uses an additional adder and multiplexer in each block.

Similar to SCSA, an n-bit adder is divided into $\lceil \frac{n}{k} \rceil$ blocks in the carry skip adder (CSA) [Kim et al. 2013]. Each block in CSA consists of a subcarry generator and a

subadder. The carry-in of the $(i+1)^{th}$ subadder is determined by the propagate signals of the i^{th} block: it is the carry-out of the $(i-1)^{th}$ subcarry generator when all propagate signals are true ($P_i = 1$); otherwise, it is the carry-out of the i^{th} subcarry generator. Therefore, the critical path delay of CSA is $O(log(k))$. This carry select scheme improves the carry prediction accuracy.

Different from SCSA, the carry speculative adder (CSPA) in Lin et al. [2015] contains one sum generator, two internal carry generators (one with carry-0 and one with carry-1), and one carry predictor in each block. The output of the i^{th} carry predictor is used to select carry signals for the $(i+1)^{th}$ sum generator. l input bits (rather than $k, l < k$) in a block are used in a carry predictor. Therefore, the hardware overhead is reduced compared to SCSA.

The consistent carry approximate adder (CCA) is similar to SCSA in that each block of CCA consists of adders with carry-0 and carry-1 [Li and Zhou 2014]. The select signal of a multiplexer is determined by the propagate signal (i.e., $S_i = (P_i + P_{i-1})SC + \overline{(P_i + P_{i-1})}C_{i-1}$), where C_{i-1} is the carry-out of the $(i-1)^{th}$ adder0 and SC is a global speculative carry. In CCA, the carry prediction depends not only on its LSBs but also on the higher bits; its critical path delay is similar to that of SCSA.

The generate signals–exploited carry speculation adder (GCSA) has a similar structure as CSA and uses the generate signals for carry speculation [Hu and Qian 2015]. The difference between them lies in the carry selection; the carry-in for the $(i+1)^{th}$ subadder is selected by its own propagate signals rather than its previous block. The carry-in is the most significant generate signal $g_{i,k-1}$ of the i^{th} block if $P_i = 1$, or else it is the carry-out of the i^{th} subcarry generator. This carry selection scheme effectively controls the maximum relative error.

In the gracefully degrading accuracy-configurable adder (GDA) [Ye et al. 2013], the control signals are used to configure the accuracy by selecting an accurate or approximate carry-in signal using a multiplexer for each subadder. The delay of GDA is determined by the carry propagation and thus by the control signals to the multiplexers.

In the carry cut-back adder (CCBA) [Camus et al. 2016], the full carry propagation is prevented by a controlled multiplexer or an OR gate for a high-speed operation. The multiplexer is controlled by a carry propagate block at a higher-significance position to cut the carry propagation at a lower-significance position. The delay and accuracy of the CCBA largely depend on the distance between the propagate block and the cutting multiplexer, thus allowing a high accuracy with a marginal overhead.

The critical path delays of the carry select adders are given by $O(log(k))$, where k is the size of the subadder.

2.1.4. Approximate Full Adders. In this type of design, approximate full adders are implemented in the LSBs of a multibit adder. It includes the simple use of OR gates (and one AND gate for carry propagation) in the so-called lower-part-OR adder (LOA) (Figure 7) [Mahdiani et al. 2010], the approximate designs of the mirror adder (AMAs) [Gupta et al. 2013], and the approximate XOR/XNOR-based full adders (AXAs) [Yang et al. 2013]. Additionally, emerging technologies such as magnetic tunnel junctions have been considered for the design of approximate full adders for a shorter delay, a smaller area, and a lower power consumption [Cai et al. 2016; Angizi et al. 2017].

The critical path of this type of adders depends on its approximation scheme. For LOA, it is approximately $O(log(n-l))$, where l is the number of bits in the lower part of an adder. In the evaluation, LOA is selected as the reference design because the other designs require customized layouts at the transistor level; hence, they are not comparable with the other types of approximate adders that are approximated at the logic gate level. Finally, an adder with the LSBs truncated is referred to as a truncated adder that works with a lower precision. It is considered a baseline design.

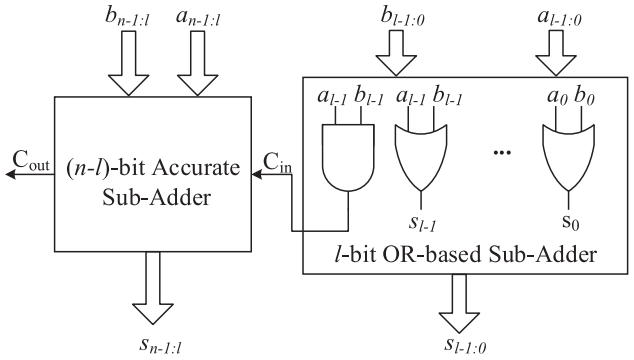


Fig. 7. The LOA.

2.2. Evaluation of Approximate Adders

2.2.1. Error Characteristics. Monte Carlo simulation is performed to evaluate the accuracy of the approximate adders. The error distance (ED) and the relative error distance (RED) are calculated as $ED = |M' - M|$ and $RED = \frac{ED}{M}$, where M' is the approximate result and M is the accurate result [Liang et al. 2013]. The mean error distance (MED) is the mean of all possible EDs. 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 mean relative error distance ($MRED$, the average value of all possible REDs) are used to assess the error characteristics of the approximate designs. Moreover, the average error (the mean of all possible errors ($M' - M$)) is used to evaluate the bias of an approximate arithmetic design.

The functions of 16−bit approximate adders are simulated by MATLAB using 10 million uniformly distributed random input combinations. Table I shows the simulation results. The size of the carry predictor for CSPA is $\lceil k/2 \rceil$ in this evaluation. The global speculative carry SC for CCA is "0," which is proved to be more accurate than using "1." Additionally, the adder with k LSBs truncated (TruA-k) is simulated for comparison.

As shown in Table I, ETAII, ACAA, and SCSA have the same error characteristics (in ER, $NMED$, and $MRED$) due to the same carry propagation chain for each sum bit. The $NMED$ and $MRED$ show the same trend, so only $MRED$ and ER are considered in the comparison, as shown in Figure 8. An equivalent carry propagation chain is selected for the considered approximate adders (i.e., the parameter k for ACA, ESA, LOA, and TruA is 8), whereas it is 4 for CSA, GCSA, ETAII, ACAA, SCSA, CCA, and CSPA. These approximate adders are considered equivalent approximate adders.

Among these approximate adders, CSA is the most accurate, and GCSA is the second most accurate in terms of $MRED$. LOA has a structure different from the other approximate adders. Its more significant part is fully accurate, whereas the approximate part is less significant. Therefore, the $MRED$ of LOA is rather small, but its ER is very large. For a similar reason, TruA has the highest ER and very large $MRED$. The information used to predict each carry in ESA and CSPA is rather limited, so the ER and the $MRED$ of ESA and CSPA are larger than most of the other approximate designs. Compared to the other approximate adders, CCA, ETAII, SCSA, and ACAA show moderate ER and $MRED$. In terms of average error, LOA has the lowest value because it produces both positive and negative errors that can compensate each other, whereas errors are accumulated for the other approximate adders since only negative errors are generated. Therefore, LOA is suitable for an accumulative operation.

Table I. Simulation Results of the Error Characteristics for the Approximate Adders

Adder Type	ER (%)	$NMED$ (10^{-3})	$MRED$ (10^{-3})	Average Error
Speculative Adders				
ACA-4	16.66	7.80	18.90	–1,024.6
ACA-5	7.76	3.90	9.60	–511.7
Segmented Adders				
ESA-4	85.07	15.70	40.40	–2,047.5
ESA-5	80.03	7.80	20.80	–1,023.4
ETAII-4	5.85	0.97	2.60	–127.5
ETAII-5	2.28	0.24	0.65	–31.6
ACAA-4	5.85	0.97	2.60	–127.5
ACAA-5	2.29	0.24	0.65	–31.6
Carry Select Adders				
SCSA-4	5.85	0.97	2.60	–127.5
SCSA-5	2.28	0.24	0.65	–31.6
CSA-4	0.18	0.06	0.15	–7.4
CSA-5	0.02	0.004	0.01	–0.5
CSPA-4	29.82	3.90	10.40	–511.4
CSPA-5	11.31	0.98	2.70	–128.3
CCA-4	8.71	0.98	2.00	–128.3
CCA-5	3.78	0.25	0.49	–32.2
GCSA-4	4.26	0.48	0.98	–63.2
GCSA-5	1.52	0.12	0.25	–16.1
Approximate Full Adders				
LOA-6	82.19	0.09	0.25	0.2
LOA-8	89.99	0.37	1.00	0.2
Truncated Adders				
TruA-6	99.98	0.48	1.30	–63.0
TruA-8	100.0	1.95	5.40	–255.0

Note: The number following the name of each approximate adder is the number of LSBs used for the carry speculation in the speculative adders, the length of the segmentation in the segmented adders, and the number of approximated and truncated LSBs in the approximate full adder–based and truncated adders.

In summary, the carry select adders and the speculative adder (ACA) are very accurate with small values of ER and $MRED$ (except for CSPA using a small number of bits for carry prediction). Represented by LOA, an approximate full adder–based adder has a moderate $MRED$, the lowest average error but a very large ER. The segmented adders are not very accurate in terms of $NMED$ and $MRED$. With very large values of ER and $MRED$, the truncated adder is the least accurate among the equivalent designs. Three different types of approximate adders, ETAII, ACAA, and SCSA, have the same error characteristics.

2.2.2. Circuit Characteristics. To assess the circuit characteristics, 16-bit approximate adders and the accurate CLA are implemented in VHDL and synthesized using the Synopsys Design Compiler (DC) based on an STM CMOS 28-nm process 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. Both the P-channel and the N-channel transistors used in the designs have a typical design corner with a regular threshold voltage. The critical path delay and area are reported by the Synopsys DC. Power dissipation is measured by the PrimeTime-PX tool at a 1ns clock period with 10 million random input combinations. All adders and subadders are implemented as CLA in this article unless otherwise noted.

Table II reports the results for the delay, power dissipation, PDP, circuit area, and ADP of the considered adders. Two structures of the accurate CLA are implemented: CLAC is realized by four cascaded 4-bit CLAs, whereas CLAG is realized by four parallel 4-bit CLAs and a carry look-ahead generator. Among ETAII, SCSA, and ACAA

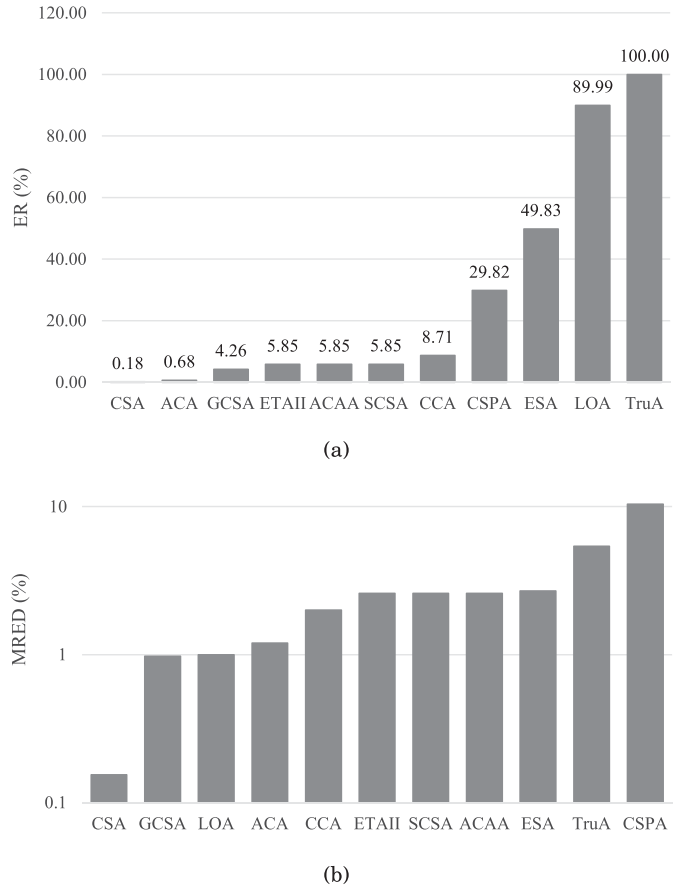


Fig. 8. A comparison of error characteristics of the approximate adders: *ER* (a) and *MRED* (b). The parameter k is 4 for CSA, GCSA, ETAII, ACAA, SCSA, CCA, and CSPA, and it is 8 for ACA, ESA, LOA, and TruA for an equivalent carry propagation chain.

(with the same error characteristics when the same value of k is selected), SCSA, albeit being the fastest, incurs the largest power dissipation and area because two subadders and one multiplexer are utilized in each block. ACAA is very slow due to its long critical path. The block of ETAII (a carry generator and a sum generator) is significantly simpler than those of SCSA and ACAA. Therefore, ETAII consumes less power and requires a smaller area than SCSA and ACAA.

In general, a circuit with a larger area is likely to consume more power except for CSA with a relatively low power dissipation but a large area. This is due to its short critical path and enhanced carry select that results in a complex wiring. Figure 9 shows the delay, power, and PDP of the equivalent adders. As expected, the accurate CLAC has the longest delay among all adders but not the highest power dissipation. Compared to CLAC, CLAG is significantly faster and consumes more power and area. TruA is not very fast, but it is the most power- and area-efficient design. LOA is also very power and area efficient compared to the other approximate adders. ESA is the slowest, but it is very power and area efficient due to its simple segmentation structure. CCA is very fast but is the most power- and area-consuming design due to its complex speculative circuit. Both CSPA and GCSA have moderate power dissipations,

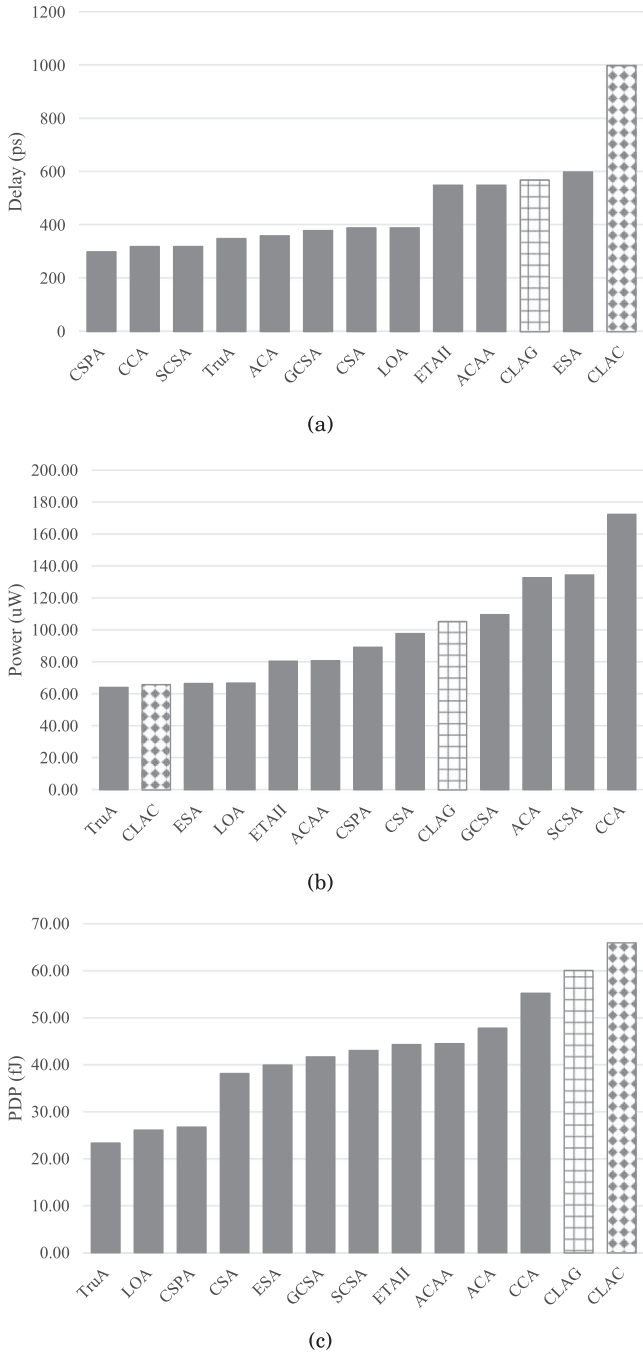


Fig. 9. A comparison of delay and power of the approximate adders delay (a), power (b), and PDP (c).

Table II. Circuit Characteristics of the Approximate Adders

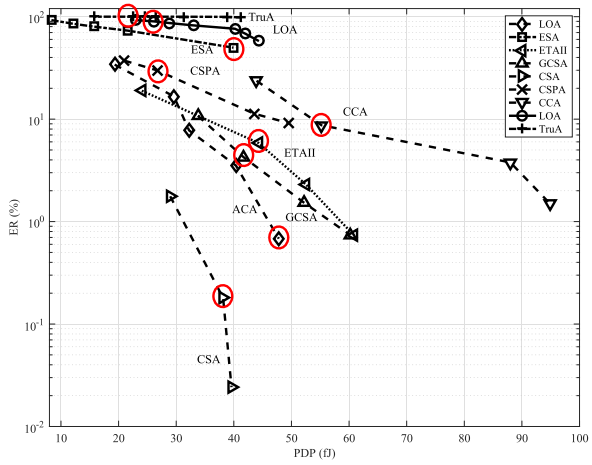
Adder Type	Delay (ps)	Power (uW)	PDP (fJ)	Area (um^2)	ADP ($um^2.ns$)
CLAC	1000	65.9	65.9	60.7	60.7
CLAG	570	105.4	60.1	84.2	48.0
Speculative Adders					
ACA-4	250	118.4	29.6	73.8	18.5
ACA-5	270	119.4	32.2	71.8	19.4
Segmented Adders					
ESA-4	260	47.0	12.2	49.9	13.0
ESA-5	310	50.6	15.7	51.7	16.0
ETAII-4	550	80.6	44.3	71.6	39.4
ETAII-5	670	78.5	52.6	70.2	47.0
ACAA-4	550	80.9	44.5	70.8	38.9
ACAA-5	650	87.3	56.8	74.6	48.5
Carry Select Adders					
SCSA-4	320	134.5	43.0	109.2	34.9
SCSA-5	400	163.0	65.2	126.2	50.5
CSA-4	390	97.8	38.1	142.5	55.6
CSA-5	420	94.3	39.6	131.2	55.1
CSPA-4	300	89.2	26.8	83.7	25.1
CSPA-5	370	117.6	43.5	100.7	37.3
CCA-4	320	172.6	55.2	131.4	42.0
CCA-5	420	209.5	88.0	155.0	65.1
GCSA-4	380	109.7	41.7	74.3	28.2
GCSA-5	460	113.6	52.3	73.3	33.7
Approximate Full Adders					
LOA-6	440	75.1	33.0	58.8	25.9
LOA-8	390	66.9	26.1	53.2	20.8
Truncated Adders					
TruA-6	390	67.9	26.5	52.4	20.4
TruA-8[1]	350	64.2	22.5	46.2	16.2

[1]TruA-8 is synthesized at a medium mapping effort, different from the high mapping effort used for the other designs. In this case, TruA-8 attains a shorter critical path delay but a similar PDP and ADP as the results synthesized using the high mapping effort.

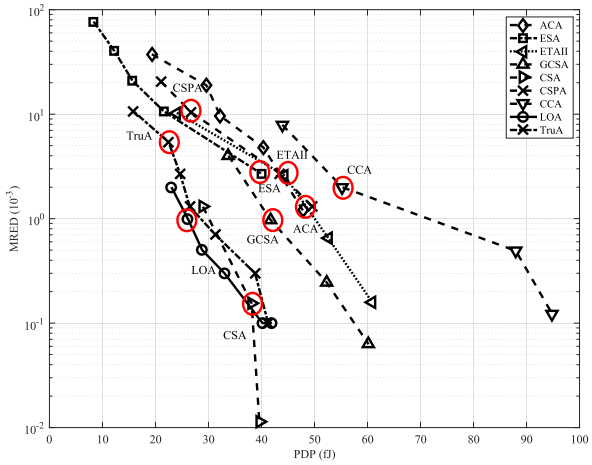
but CSPA is faster and GCSA uses a smaller area. Both the speed and power dissipation of CSA are in the medium range. In terms of PDP and ADP (shown in Table II), they show a very similar trend. TruA, LOA, and CSPA have very small values of PDP and ADP, whereas these values are relatively large for ACA and CCA (shown in Figure 9(c)).

In conclusion, the carry select adders are likely to have large values of power dissipation and area at a moderate performance. The segmented adders are power and area efficient. A speculative adder is very fast, but it is also very power consuming with a moderate area. Conversely, the approximate full adder–based adder is slow, but it consumes a low power and area. The approximate full adders are very efficient in PDP and ADP, whereas the speculative adders are not. The truncated adder is very power and area efficient, but with a relatively long delay.

2.2.3. Discussion. Since the ADP shows a similar trend as the PDP, the PDP is considered for a comprehensive comparison of the approximate adders, as shown in the two-dimensional (2-D) plots of Figure 10. CSA-6 is accurate due to the precise carry generated for every block, so the ER and $MRED$ of CSA-6 are 0. Therefore, they are not shown in Figure 10. The equivalent adders are marked by circles. Among adders with the same accuracy (ETAII, SCSA, and ACAA), ETAII is the most efficient in terms of delay, power, and area. Thus, it is shown as a representative in Figure 10. Compared to



(a)



(b)

Fig. 10. Comprehensive comparison of the approximate adders: *ER* and PDP (a) and *MRED* and PDP (b). The parameter k for LOA and TruA ranges from 9 down to 3 from left to right, is 3 to 8 for ESA and ACA, and is from 3 to 6 for the other adders from left to right. The adders marked by circles are equivalent in terms of carry propagation and are thus representatives of different designs.

the other approximate adders, CCA has the largest PDP and moderate *ER* and *MRED*. Among the schemes with moderate PDPs (CSPA, GCSA, and ETAII), ETAII and GCSA have moderate *MRED*s and *ER*s, whereas CSPA shows slightly higher values of these measures. ESA has a rather small PDP but a considerably large *ER* and *MRED*. ACA has a larger PDP than ESA, but it has both lower *ER* and *MRED*. Among all approximate adders, CSA shows the best performance with very small PDP, *ER*, and *MRED* values.

With the highest *ER*s, LOA and TruA show the smallest PDPs for a similar *MRED* due to their low power dissipation. In fact, these approximate adders show a decent trade-off in error distance and hardware efficiency. In particular, they are useful in applications in which hardware efficiency is of the utmost importance.

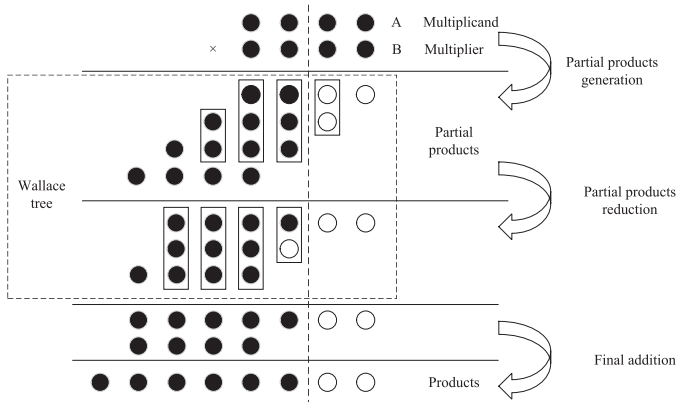


Fig. 11. Basic arithmetic process of a 4 × 4 unsigned multiplication with possible truncations to a limited width. ●, an input, a partial product, or an output product; ○, a truncated bit; □, a full adder or a half adder.

3. APPROXIMATE MULTIPLIERS

3.1. Classification of Approximate Multipliers

Generally, a multiplier consists of stages of partial product generation, accumulation, and a final addition, as shown in Figure 11 for a 4 × 4 unsigned multiplication. Let A_i and B_j be the i^{th} and j^{th} least significant bits of inputs A and B, respectively; a partial product $P_{j,i}$ is usually generated by an AND gate (i.e., $P_{j,i} = A_i B_j$). The commonly used partial product accumulation structures include the Wallace, Dadda trees, and a carry-save adder array. The Wallace tree for a 4 × 4 unsigned multiplier is shown in the dotted box of Figure 11. The adders in each layer operate in parallel without carry propagation, and the same operation repeats until two rows of partial products are left. For an n-bit multiplier, $log(n)$ layers are required in a Wallace tree. Therefore, the delay of the partial product accumulation stage is $O(log(n))$. Moreover, the adders in Figure 11 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.

A carry-save adder array is shown in Figure 12; the carry and sum signals generated by the adders in a row are passed on 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 due to the simple and symmetric structure.

Three main methodologies are used for the approximate design of a multiplier: (i) approximation in generating the partial products [Kulkarni et al. 2011]; (ii) approximation (including truncation) in the partial product tree [Mahdiani et al. 2010; Kyaw et al. 2010; Bhardwaj et al. 2014]; and (iii) using approximate designs of adders [Liu et al. 2014], counters [Lin and Lin 2013], or compressors [Ma et al. 2013; Momeni et al. 2015] to accumulate the partial products. For a signed integer operation, Booth multipliers have been widely used due to the fast operation on a reduced number of partial products. Some recent designs use shifting and addition to obtain the final product by rounding the inputs to a form of 2^m (m is a positive integer) [Hashemi et al. 2015; Zendeganie et al. 2017; Mitchell 1962].

Based on the different schemes in approximation, approximate multipliers are classified into three unsigned types and signed Booth multipliers. Following this classification, existing designs of approximate multipliers are briefly reviewed next.

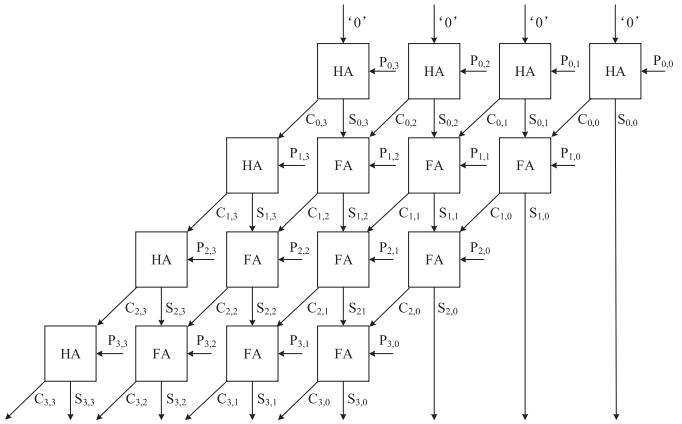


Fig. 12. Partial product accumulation of a 4 × 4 unsigned multiplier using a carry-save adder array. HA, a half adder; FA, a full adder.

Table III. K-Map for the 2 × 2 Underdesigned Multiplier Block

$M_2M_1M_0$		B_1B_0			
		00	01	11	10
A_1A_0	00	000	000	000	000
	01	000	001	011	010
	11	000	011	111	110
	10	000	010	110	100

3.1.1. Approximation in Generating Partial Products. The underdesigned multiplier (UDM) utilizes an approximate 2 × 2 multiplier obtained by altering a single entry in the Karnaugh Map (K-Map) of its function (as highlighted in Table III) [Kulkarni et al. 2011]. Table III shows the K-Map of the approximate 2 × 2 multiplier, where A_1A_0 and B_1B_0 are the two 2-bit inputs, and $M_2M_1M_0$ is the 3-bit output. In this approximation, the accurate multiplication result "1001" is simplified to "111" to save one output bit when both the inputs are "11." Assuming that the value of each input bit is equally likely, the error rate of the 2 × 2 multiplier is then $(\frac{1}{2})^4 = \frac{1}{16}$. Larger multipliers can be designed based on the 2 × 2 multiplier. This multiplier introduces an error when generating the partial products; however, the adder tree remains accurate.

3.1.2. Approximation in the Partial Product Tree. A bioinspired imprecise multiplier, referred to as a broken-array multiplier (BAM), is proposed in Mahdiani et al. [2010]. The BAM operates by omitting some carry-save adders in an array multiplier in both horizontal and vertical directions (Figure 13). A more straightforward approach to truncation is to truncate some LSBs on the input operands, and thus a smaller multiplier is sufficient for the remaining MSBs. This truncated multiplier (TruM) is considered as a baseline design.

The error-tolerant multiplier (ETM) is divided into a multiplication section for the MSBs and a nonmultiplication section for the LSBs [Kyaw et al. 2010]. Figure 14 shows the architecture of a 16-bit ETM. A NOR gate–based control block is used to deal with the following two cases: (i) if the product of the MSBs is zero, then the upper accurate 8-bit multiplier is activated to multiply the LSBs without any approximation, and (ii) if the product of the MSBs is nonzero, the nonmultiplication section is used as an approximate multiplier to process the LSBs, whereas the multiplication section is activated to accurately multiply the MSBs.

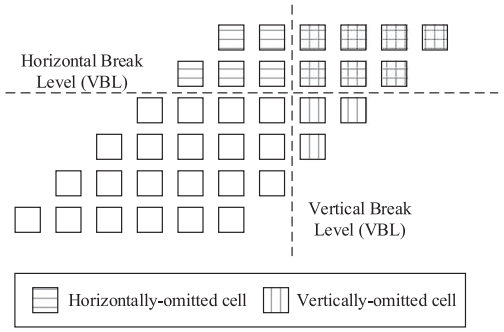


Fig. 13. The BAM with four vertical lines and two horizontal lines omitted [Mahdiani et al. 2010]. □, a carry-save adder cell.

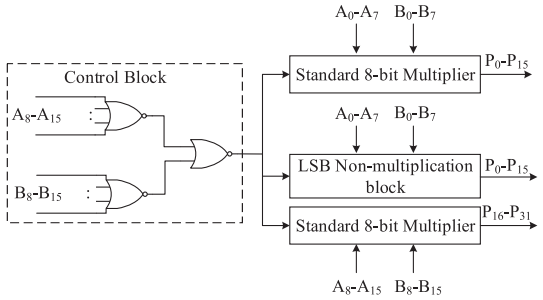


Fig. 14. The 16-bit ETM of Kyaw et al. [2010].

The static segment multiplier (SSM) was proposed using a similar partition scheme [Narayanamoorthy et al. 2015]. Different from ETM, no approximation is applied to the LSBs in the SSM. Either the MSBs or the LSBs of the operands are accurately multiplied depending on whether its MSBs are all zeros. Liu et al. [2017a] has shown that only small improvements in accuracy and hardware are achieved compared to ETM, and thus this design is not further considered in this 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 [Bhardwaj et al. 2014]. An n-bit AWTM is implemented by four $n/2$-bit submultipliers, as shown in Figure 15, where the most significant submultiplier $A_H B_H$ is further implemented by four $n/4$-bit submultipliers. The AWTM is configured into four different modes by the number of approximate $n/4$-bit submultipliers in the most significant $n/2$-bit submultiplier, whereas the other three multipliers ($A_H B_L$, $A_L B_H$, and $A_L B_L$) are approximate. The approximate partial products are then accumulated by a Wallace tree.

3.1.3. Using Approximate Counters or Compressors in the Partial Product Tree. An approximate (4:2) counter is proposed in Lin and Lin [2013] for an inaccurate 4-bit Wallace multiplier. Table IV shows the K-Map of the approximate (4:2) counter, where $X_1 \cdots X_4$ are the four input signals of a (4:2) counter (i.e., the partial products in the partial product tree of a multiplier), and C and S are the carry and sum, respectively. The values of CS in the box are approximated as "10" for "100" in the approximate counter 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.

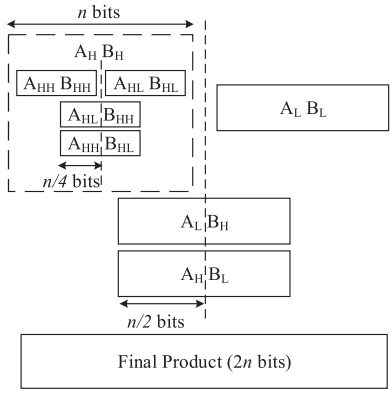


Fig. 15. The AWTM [Bhardwaj et al. 2014]. n is the width of the multiplier; A_HB_H, A_LB_H, A_HB_L, and A_LB_L are partial products generated by the $n/2$-bit submultipliers; and $A_{HH}B_{HH}$, $A_{HL}B_{HH}$, $A_{HH}B_{HL}$, and $A_{HL}B_{HL}$ are partial products generated by the $n/4$-bit submultipliers.

Table IV. K-Map for the 4:2 Approximate Counter

CS		X_1X_0			
		00	01	11	10
X_3X_4	00	00	01	10	01
	01	01	10	11	10
	11	10	11	10	11
	10	01	10	11	10

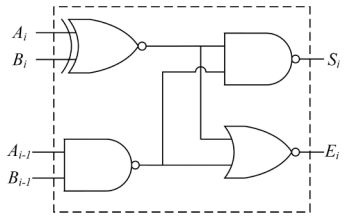


Fig. 16. The approximate adder cell. S_i, the sum bit; E_i, the error bit [Liu et al. 2014].

In the compressor-based multiplier, accurate (3:2) and (4:2) compressors are improved to speed up the partial product accumulation [Baran et al. 2010]. 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 Momeni et al. [2015]; 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 Kelly et al. [2009]. Several signed multipliers are then implemented using these approximate counters. The more accurate schemes 3 and 4 of the approximate compressor–based multiplier (referred to as ACM-3 and ACM-4) in Momeni et al. [2015] are considered for comparison.

In the approximate multiplier with configurable error recovery, the partial products are accumulated by a novel approximate adder (Figure 16) [Liu et al. 2014]. The approximate adder utilizes two adjacent inputs to generate a sum and an error bit. The adder processes data in parallel, and 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

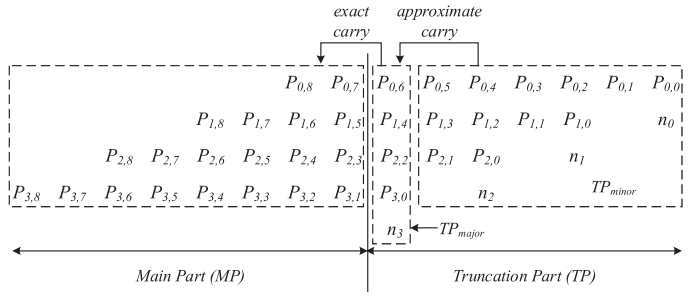


Fig. 17. Partial products for an 8-bit fixed-width modified Booth multiplier [Cho et al. 2004]. $P_{i,j}$ is the j^{th} partial product in the i^{th} partial product vector, and n_i is the sign of the i^{th} partial product vector.

first error accumulation stage in scheme 1 (AM1), whereas in scheme 2 (AM2) both OR gates and the approximate adders are used. The truncation of 16 LSBs in the partial products in AM1 and AM2 results in TAM1 and TAM2, respectively [Liu 2014].

3.1.4. Approximate Booth Multipliers. The Booth recoding algorithm handles binary numbers in 2's complement. The modified (or radix-4) Booth algorithm is commonly used due to its ease in generating partial products. Little work has been reported for approximate Booth multipliers, whereas the fixed-width Booth multiplier utilizes a truncation-based approach that has been studied for more than a decade. The conventional fixed-width multiplier generates an output with the same width as the input operand by truncating the lower half of the product. Truncation of half of the partial products is widely used because the posttruncated scheme does not achieve a significant circuit advantage over the accurate multiplier. A direct truncation of partial products incurs a large error, so many error compensation schemes have been proposed [Cho et al. 2004; Song et al. 2007; Wang et al. 2011; Chen and Change 2012]. Another approach is to use an approximate Booth encoder with a simple circuit [Liu et al. 2017b]. Most of the approximate Booth multipliers are based on the modified Booth algorithm; the partial products are accumulated by an array structure in Cho et al. [2004], Song et al. [2007], Wang et al. [2011], and Farshchi et al. [2013], whereas a parallel carry-save-adder tree is used in Chen and Chang [2012] and Liu et al. [2017b]. The approximate Booth multiplier in Jiang et al. [2016a] is based on the radix-8 Booth algorithm.

Figure 17 shows the partial products of an 8-bit fixed-width modified Booth multiplier with error compensation [Cho et al. 2004]. The final product is the addition of the main part (MP) and the carry signals generated in the truncation part (TP). The carry signals are approximated by the output of Booth encoders. The approximate carry σ is $\sigma = \lfloor 2^{-1}(\sum_{i=0}^{n/2-2} \overline{zero_i} + 1) \rfloor$, where n is the multiplier width and $\overline{zero_i}$ is "1" if the i^{th} partial product vector is not zero or $\overline{zero_i} = 0$ otherwise. This multiplier is referred to as BM04.

The multiplier in Song et al. [2007] can adaptively compensate the quantization error by keeping different numbers of the most significant columns of the partial products (ω ($\omega \geq 0$)). Two types of binary thresholding are proposed for error compensation. Different from BM04, n rather than $(n-1)$ columns of partial products are truncated for an n-bit multiplier. The error compensation for each type of binary thresholding varies with the value of ω and the partial products of the ω^{th} column in the TP (from left to right). This multiplier is denoted as BM07.

The multiplier presented in Wang et al. [2011] uses n columns of partial products in the TP for an n-bit multiplier; the most significant one column in the TP is reserved for error compensation. The error compensation using a simplified sorting network significantly reduces the mean and mean-squared errors by making the error symmetric,

as well as centralizing the error distribution around zero. This design is referred to as BM11.

A fixed-width Booth multiplier was designed based on a probabilistic estimation bias in Chen and Chang [2012]. Therefore, this multiplier is referred to as PEBM. The number of columns of the accumulated partial products varies in accordance with the desired trade-off between hardware and accuracy. The error compensation formula is derived from a probability analysis rather than a time-consuming exhaustive simulation. The carry generated by the TP is approximated by $\sigma = \lfloor 2^{-1}(\sum_{i=0}^{n/2-1-\lfloor \omega/2 \rfloor} z_i - 1) \rfloor$, where $z_i = P_{0,n/2-1} + n_{n/2-1}$ when i is $n/2-1$ and $z_i = \overline{zero_i}$ otherwise.

Based on BAM, the broken Booth multiplier (BBM) uses a modified Booth algorithm to generate partial products and omits carry-save adders to the right of a vertical line [Farshchi et al. 2013]. BBM has a smaller PDP for the same mean-squared error compared to BAM.

An approximate recoding adder is proposed in Jiang et al. [2016a] for calculating the triple multiplicands to reduce the additional delay encountered in a radix-8 Booth multiplier. A Wallace tree and a truncation technique are then utilized for partial product accumulation to reduce power and delay. The most efficient fixed-width multiplier ABM2_R15 is considered in this comparison and referred to as ABM2 in this article for simplicity. In Liu et al. [2017b], two approximate radix-4 Booth encoders are designed for the partial product generation by simplifying the exact K-Map. The generated partial products are then accumulated by using exact 4:2 compressors.

3.2. Evaluation of Approximate Multipliers

3.2.1. Error Characteristics. The considered (16×16) approximate multipliers are simulated by MATLAB with 10 million uniformly distributed random input combinations. The *ER*, *NMED*, *MRED*, and average error are obtained and shown in Table V. TruM-k represents the truncated multiplier with k LSBs truncated in the input operands.

According to Table V, most of the designs, especially those with truncation, have large *ER*s close to 100%. However, ICM has a relatively low *ER* of 5.45%, because it uses just one approximate counter in a 4×4 submultiplier with an error rate of only $\frac{1}{256}$. UDM also shows a lower *ER* than the other approximate multipliers. In terms of the average error, ACMs have the smallest value, whereas the average errors for all other approximate unsigned multipliers show the same trend with the *NMED*. This is because ACMs produce both positive and negative errors, but the other approximate unsigned multipliers produce either negative or positive errors.

Figure 18 shows the *NMED*s and *MRED*s of the equivalent approximate multipliers that are configured to have 16-bit accurate MSBs (except for ICM and UDM that have only one configuration). Thus, the truncated LSBs in the partial product is 16 for BAM; the number of MSBs used for error compensation is 16 for AM1, AM2, TAM1, and TAM2; the size of the accurate submultiplier is 8 for ETM; 8 LSBs are truncated for TruM; and the mode number of ACM and AWTM is 4. Among the unsigned approximate multipliers, UDM has the largest *NMED*, whereas ACM has the smallest. ICM, AM2, and TAM2 have similar values of *NMED*; however, ICM has the smallest *MRED*, whereas the *MRED* of TAM2 is the largest. Therefore, ICM has the highest accuracy in terms of *MRED*, whereas TAM2 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. BAM has moderate values of *NMED* and *MRED*, whereas ETM and TruM have both large *MRED* and *NMED*.

Hence, ICM is the most accurate design with the lowest *ER*, *MRED*, and a moderate *NMED*. ACM, AWTM, BAM, AM2, and TAM2 also show good accuracy among all

Table V. Error Characteristics of the Approximate Multipliers

Multiplier Type	ER (%)	$NMED$ (10^{-3})	$MRED$ (%)	Average Error (10^4)
Multipliers With Approximation in Generating Partial Products				
UDM	80.99	13.92	3.33	–5,974.9
Multipliers With Approximation in the Partial Product Tree				
BAM-16	99.99	0.06	0.21	–24.6
BAM-17	99.99	0.11	0.36	–49.2
BAM-18	99.99	0.22	0.63	–95.0
BAM-20	100.00	0.79	1.79	–337.5
ETM-7	99.99	0.97	1.56	–413.4
ETM-8	100.00	1.94	2.85	–825.1
AWTM-1	100.00	2.70	75.00	1,159.6
AWTM-2	100.00	1.88	39.37	807.4
AWTM-3	99.99	0.12	2.51	51.5
AWTM-4	99.94	0.02	0.33	8.6
Multipliers Using Approximate Counters or Compressors				
ICM	5.45	0.29	0.06	–124.2
ACM-3	99.99	0.01	0.29	3.95
ACM-4	99.97	0.01	0.26	1.44
AM1-13	99.38	0.90	0.54	–385.5
AM1-16	98.22	0.81	0.34	–347.9
AM2-10	99.64	0.88	1.20	–379.8
AM2-13	99.36	0.35	0.34	–148.9
AM2-16	97.96	0.27	0.13	–115.1
TAM1-13	99.99	1.14	0.77	–488.5
TAM1-16	99.99	1.06	0.58	–457.1
TAM2-10	99.99	0.90	1.27	–384.2
TAM2-13	99.99	0.36	0.41	–153.3
TAM2-16	97.99	0.28	0.22	–121.0
Truncated Unsigned Multipliers				
TruM-4	99.61	0.11	0.23	–49.2
TruM-8	100.0	1.94	2.85	–834.1
Approximate Booth Multipliers				
PEBM	99.99	0.023	0.27	–1.02
BBM	100.00	0.092	0.57	–9.83
BM11	99.99	0.022	0.18	–0.003
BM07	99.99	0.024	0.16	–1.56
BM04	99.99	0.027	0.48	–2.66
ABM2	99.99	0.034	0.44	–0.614

Note: The parameter k follows the acronym of each approximate multiplier. For AM1, AM2, TAM1, and TAM2, this parameter refers to the number of MSBs used 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.

considered approximate multipliers with both low $NMED$s and $MRED$s. ETM, TruM, and UDM are not very accurate in terms of these metrics.

For the approximate Booth multipliers in Table V, a column of the most significant partial products in the TP (adjacent to the MP) is kept for PEBM, BM07, and BM04. The 15-bit columns of partial products are truncated in BBM and ABM2 to keep the same width of the output as the other designs. The ERs of all approximate multipliers are close to 100% due to truncation. Most designs have similar $NMED$s except for BBM. BBM has the largest $NMED$ and $MRED$, because there is no error compensation. BM07 and BM11 have very small $MRED$ values, whereas PEBM has a slightly larger value. BM11 has the lowest average error, and the average error of ABM2 is also very small.

In summary, as a multiplier approximated in generating the partial products, UDM has very large values of $NMED$ and $MRED$, and a relatively small ER. The multipliers approximated in the partial product tree mostly have moderate $NMED$s and very large $MRED$s (except for BAMs with fewer than or equal to 18 truncated bits and AWTM-4).

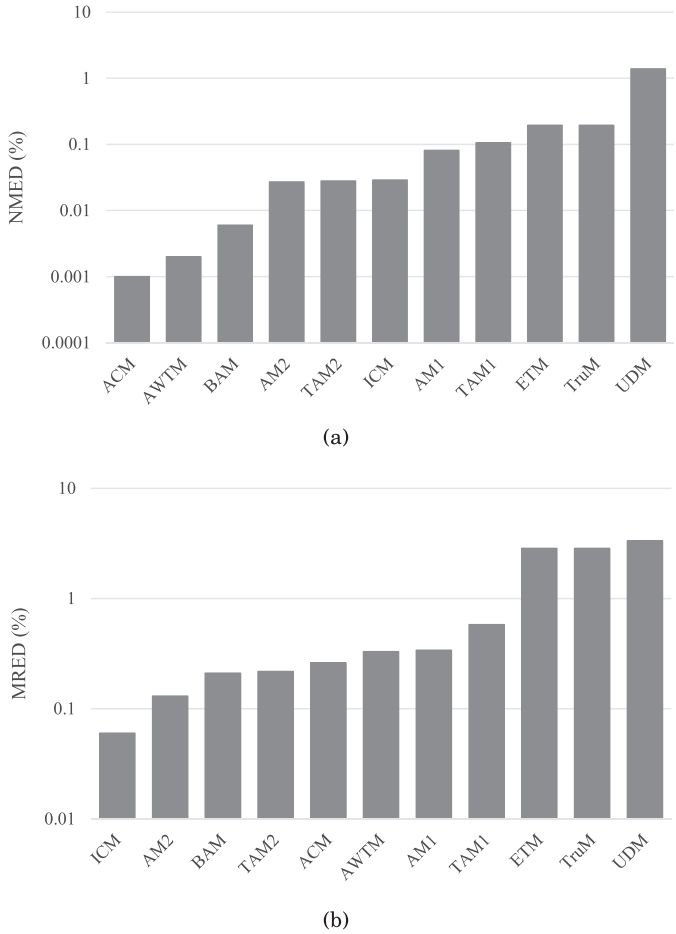


Fig. 18. Comparison of *NMED* and *MRED* of the approximate multipliers with increasing *NMED* (a) and *MRED* (b). ACM and AWTM represent ACM-4 and AWTM-4, respectively. The truncated number of LSBs in the partial product is 16 for BAM, the number of MSBs used for error compensation is 16 for AM1; AM2, TAM1, and TAM2; and 8 LSBs are truncated for TruM. ETM is ETM-8 in Table V.

The multipliers approximated using approximate counters or compressors have small values of both *NMED* and *MRED*, whereas the multipliers truncated on the input operands have large values of both metrics (when the truncated number of LSBs is larger than 4). Among the considered approximate Booth multipliers, BBM shows the lowest accuracy in terms of both *NMED* and *MRED*. BM11 has the smallest average error. The other approximate Booth multipliers show similar *NMED*s and various *MRED*s.

3.2.2. Circuit Characteristics. The 16 × 16 approximate multipliers are implemented in VHDL and synthesized using the same tool and process as in the simulation of approximate adders. The only difference is that the clock period is 4ns for the power estimation of the multipliers because of a longer critical path delay. The accurate Wallace multiplier (WallaceM) optimized for speed [Oklobdzija et al. 1996] and array multiplier (ArrayM) are also simulated for comparison. To reduce the effect of the final addition, the same multibit adder in the tool library is utilized in all approximate multiplier

Table VI. Circuit Characteristics of the Approximate Multipliers

Multiplier Type	Delay (ns)	Power (uW)	PDP (fJ)	Area (um^2)	ADP ($um^2.ns$)
ArrayM	2.58	477.4	1,231.7	921	2,375.7
WallaceM	2.03	461.3	936.4	934	1,896.0
Multipliers With Approximation in Generating Partial Products					
UDM	2.01	352.7	708.9	829	1,666.7
Multipliers With Approximation in the Partial Product Tree					
BAM-16	2.34	221.3	517.8	441	1,031.9
BAM-17	2.17	189.5	411.2	384	833.3
BAM-18	1.99	161.1	320.6	331	658.7
BAM-20	1.65	111.0	183.2	237	390.4
ETM-7	1.57	140.5	220.6	349	547.6
ETM-8	1.50	108.5	162.8	288	431.4
AWTM-1	1.69	247.8	418.8	640	1,081.8
AWTM-2	1.69	259.4	438.4	665	1,123.7
AWTM-3	1.69	270.3	456.8	690	1,165.6
AWTM-4	1.74	280.0	478.2	715	1,243.2
Multipliers Using Approximate Counters or Compressors					
ICM	1.87	367.4	687.0	937	1,751.4
ACM-3	1.97	279.5	550.6	738	1,454.5
ACM-4	2.00	284.1	568.2	724	1,447.0
AM1-13	1.38	355.4	490.5	819	1,128.8
AM1-16	1.57	380.6	597.5	878	1,378.5
AM2-10	1.29	336.8	434.5	816	1,052.6
AM2-13	1.53	364.2	557.2	919	1,406.1
AM2-16	1.71	400.4	684.7	1,051	1,797.2
TAM1-13	1.31	192.0	251.5	460	602.6
TAM1-16	1.45	214.6	311.2	516	748.2
TAM2-10	1.23	180.2	221.6	477	586.7
TAM2-13	1.48	212.5	314.5	584	864.3
TAM2-16	1.62	244.9	396.7	693	1,122.7
Truncated Unsigned Multipliers					
TruMA-4	1.89	243.5	460.2	503	950.6
TruMA-8	1.19	92.1	109.6	211	250.5
TruMW-4	1.62	262.4	425.1	561	908.2
TruMW-8	1.10	98.4	108.2	239	263.0
Approximate Booth Multipliers					
PEBM	1.83	264.3	483.7	528	966.2
BBM	1.91	250.3	478.1	487	930.2
BM11	1.96	258.1	505.9	475	931.0
BM07	2.03	270.4	548.9	528	1,071.8
BM04	2.05	249.8	512.1	447	916.4
ABM2	2.25	208.0	468.0	424	954.0

designs as the final adder. Table VI shows the critical path delay, area, power, PDP, and ADP of the considered multipliers. TruMA and TruMW are the truncated array and Wallace multipliers, respectively.

Figure 19 shows the comparison of delay, power, and PDP of the equivalent approximate multipliers. The accurate array multiplier (ArrayM) is the slowest, and the Wallace multiplier (WallaceM) consumes more area (as per Table VI); this is consistent with the theoretical analysis. Due to the expressively fast carry-ignored operation, AM1/TAM1, AM2/TAM2 have smaller delays compared to most of the other designs. BAM is significantly slower due to its array structure. AWTM, UDM, ICM, and ACM have larger delays than the other approximate multipliers. BAM consumes very low power and the power consumptions of AWTM and ACM are in the medium range, whereas UDM and ICM incur a relatively high power consumption. TruMA, TruMW, and ETM have both a short delay and a low power dissipation.

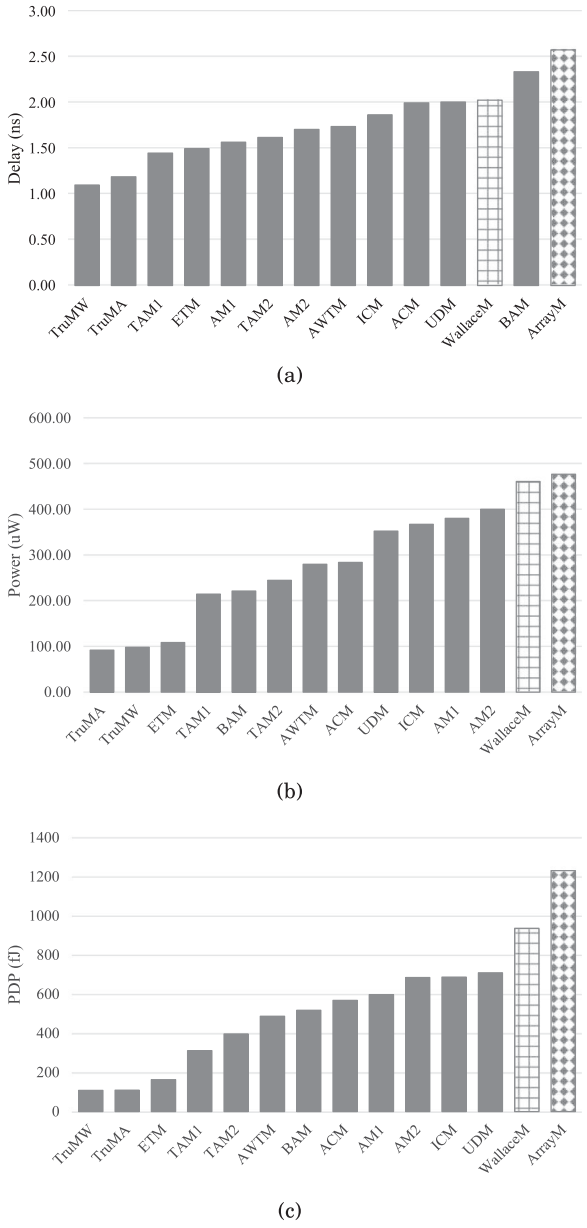


Fig. 19. Comparison of delay, power, and PDP of the approximate multipliers: delay (a), power (b), and PDP (c).

A multiplier with a higher power dissipation usually has a larger area and thus larger PDP and ADP. In terms of power and area, TruMA, TruMW, 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 the $MRED$, whereas the $NMED$ may not be significantly changed. 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.

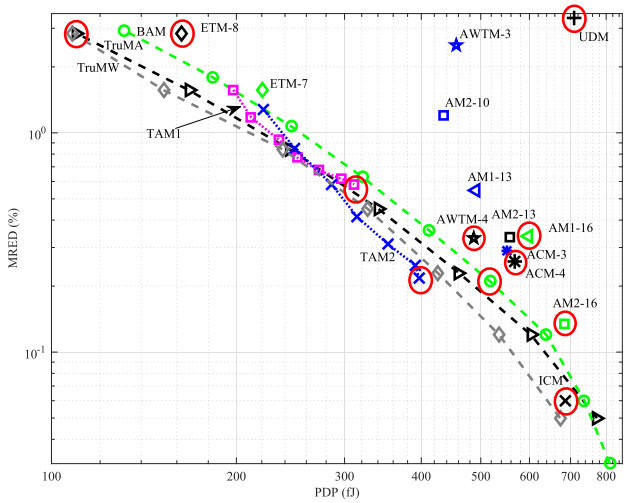


Fig. 20. *MRED* and PDP of the approximate unsigned multipliers. The parameter k for TruMA and TruMW is from 8 down to 2 from left to right, is 21 down to 13 for BAM, and is 10 to 16 for TAM1 and TAM2. The multipliers marked by circles are equivalent in terms of the number of accurate MSBs and are thus representatives of different designs.

Without truncation, a multiplier whose design is approximated in generating partial product (e.g., UDM) tends to have a large delay, power, and area. These measures for the multipliers approximated in the partial product tree (e.g., AWTM) are moderate, whereas the multipliers using approximate counters or compressors (ICM, ACM, AM1, AM2) require higher power and area.

In terms of PDP (see Figure 19(c)) and ADP, TruMA, TruMW, ETM, TAM1, and TAM2 have very small values, whereas ICM, UDM, and AM2 are the opposite. The values of PDP and ADP for AM1, ACM, BAM, and AWTM are in the medium range.

For the approximate Booth multipliers, PEBM is the fastest, but it is very power and area consuming due to the use of a carry save adder tree for the parallel accumulation. ABM2 is the slowest but the most power- and area-efficient design due to the smaller number of partial products generated by the time-consuming recoding adder in the radix-8 algorithm. Thus, it has the smallest PDP and a moderate value of ADP. With no error compensation, BBM shows a small delay, low power dissipation, and small circuit area, and thus smaller PDP and ADP compared to most other designs (except ABM2 and BM04). BM11 and BM04 have similar values for all circuit metrics. BM07 has a similar delay, but with a higher power and area, and thus a larger PDP and ADP.

3.2.3. Discussion. *MRED* and PDP are jointly considered next for an overall evaluation of the approximate multipliers, as shown in Figures 20 and 21.

Figure 20 shows that TruMW has a smaller PDP than TruMA when the same number of LSBs is truncated. Among the truncation-based designs, the truncated unsigned multipliers (TruMA and TruMW) are slightly more accurate (in *MRED*) than BAM and ETM for a similar PDP. TruMW has a smaller *MRED* than most other approximate designs (except TAM1, TAM2, and ICM).

In Figure 20, TAM1-13, TAM1-16, TAM2-13, TruMA-6, TruMW-6, and BAM-18 have both small PDPs and *MRED*s. 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. Other than the truncated designs, ETM-8 has the smallest PDP but a rather large *MRED*. UDM shows poor performance in both PDP and *MRED*. Even though some BAM

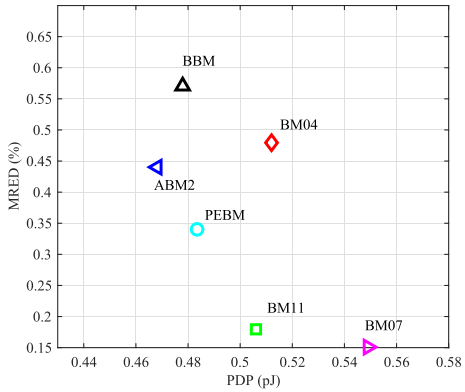


Fig. 21. *MRED* and PDP of the approximate Booth multipliers.

configurations have small PDPs, their delays are generally large (see Figure 19(a)); moreover, some BAM configurations have low accuracies. AWTMs have large PDPs, and only AWTM-4 has a high accuracy.

As for the approximate Booth multipliers (see Figure 21), ABM2 shows the lowest PDP and moderate accuracy. BM11 and BM07 are very accurate in terms of *MRED* but have relatively poor PDPs. PEBM shows both a moderate PDP and *MRED*.

4. APPROXIMATE DIVIDERS

4.1. Classification

The divider is a less frequently used arithmetic module compared to the adder and multiplier; therefore, less research has been pursued on an approximate design.

Two methodologies have been advocated for sequential division: the digit recurrent algorithm [Liu and Nannarelli 2012] and the functional iterative algorithm (e.g., using the Newton-Raphson algorithm [Flynn 1970]). A sequential divider has a low hardware complexity; however, its delay is considerably longer than an adder and a multiplier, so it significantly affects the overall performance of a processor. Thus, dividers made of combinational logic circuits are discussed in this article. Like multiplication, division can also be implemented by an array structure, in which adder cells are replaced by subtractor cells. Two approximations are made on the array divider while retaining a high-speed and low-power operation [Chen et al. 2015; Hashemi et al. 2016]. In addition, different approximate divider designs based on rounding [Zendegani et al. 2016] and curve fitting [Low and Jong 2013; Wu and Jong 2015] are also proposed.

4.1.1. Approximate Array Dividers. Four types of approximate unsigned integer nonrestoring divider (AXDnr) are presented in Chen et al. [2015]. Three approximate subtractors (AXCSs) are designed for the array of an unsigned divider by simplifying the circuit of an exact subtractor cell. The AXCSs are then used to replace the exact subtractor cells at the least significant vertical, horizontal, square, or triangle cells of the array divider. Moreover, a truncation scheme is utilized by discarding the approximate subtractors for comparison. Based on the same theory and design, four types of approximate restoring divider (AXDr) are further proposed by using the proposed AXCSs [Chen et al. 2016]. It has been shown that AXDrs are slightly more accurate and consume lower power than AXDnrs.

To make the remaining subtractor cells more efficient, a dynamic approximate divider (DAXD) is proposed by dynamically selecting the inputs of the subtractor cells [Hashemi et al. 2016]. DAXD selects a fixed number of bits in the input operands from

the most significant nonzero bit and then truncates the least significant bits. Therefore, it can be implemented by two leading-one detectors, two multiplexers, a smaller array divider, and a barrel shifter.

4.1.2. Curve Fitting–Based Approximate Dividers. A widely used methodology to reduce the hardware overhead of a divider is based on binary logarithms—that is, to obtain the antilogarithmic value of the difference between the logarithmic values of the dividend and the divisor. Mitchell [1962] first developed the logarithm approximation of a binary number by shifting and counting. Inspired by it, a new antilogarithmic algorithm was proposed using a piecewise linear approximation [Low and Jong 2013]. A high-speed divider (HSD) based on this algorithm was then presented. The antilogarithmic algorithm is directly approximated from the input operands, and thus only look-up tables and multiplications are required (i.e., no logarithmic or subtraction operation is needed). HSD achieves better accuracy and much higher speed (but at a larger area) than the divider implemented directly by Mitchell's algorithm.

A similar curve fitting approach was used for the design of a floating-point divider (FPD) [Wu and Jong 2015]. FPD partitions the curved surfaces of the quotient into several square or triangular regions and linearly approximates each region by curve fitting. Finally, the division of the mantissas is implemented by a comparison module, a look-up table, shifters, and adders. This approximate divider achieves a better accuracy than that in Low and Jong [2013] with similar circuit characteristics.

4.1.3. Rounding-Based Approximate Dividers. A high-speed and energy-efficient rounding-based approximate divider (SEERAD) is presented in Zendegani et al. [2016]. It transforms the division to a smaller multiplication by rounding the divisor B to a form of $2^{K+L}/D$, where K shows the bit position of the most significant "1" of B ($K = \lfloor log_2 B \rfloor$), and L and D are constant integers found from an exhaustive simulation by the condition of obtaining the lowest mean relative error. Different accuracy levels are considered to improve the accuracy of SEERAD by varying D and L with combinations of the more relevant bits of B after the most significant "1." The multiplier in SEERAD is implemented by several shift units and an adder block. Thus, SEERAD is very fast.

4.2. Discussion

The approximate array dividers, AXDnr, AXDr, and DAXD, are designed for the unsigned $n/(\frac{n}{2})$ division, where n is the width of the dividend. As the borrow signal passes through all subtractor cells, the critical path of an array divider is $O(n^2)$ [Parhami 2000]. Thus, the speed of the approximate array dividers is not very fast, but the area and power dissipation are relatively low among the approximate designs. The accuracy of the array dividers depends on the number of replaced cells (AXDnr and AXDr) and the size of the subdivider (DAXD).

The approximate dividers based on curve fitting are very accurate (e.g, the maximum relative error distance of a 16/16 HSD and FPD are 0.20% and 0.14%, respectively). Although curve fitting using software is complex, the hardware implementation consisting of look-up tables, smaller multipliers, and adders is very simple. Thus, curve fitting–based dividers are usually fast. However, the look-up tables used for storing the constant coefficients make the approximate dividers area and power consuming.

The rounding-based approximate divider has a maximum relative error distance of 6.25% (for the design with the highest accuracy level) in a 16/16 divider, and hence it is not very accurate. Its hardware implementation is simpler than those of the curve fitting–based dividers. Therefore, the rounding-based approximate divider has a very high speed and relatively large area and power dissipation due to the use of look-up tables.

Fig. 22. The image sharpened using an accurate multiplier and an accurate adder.

5. IMAGE PROCESSING APPLICATION

Low power dissipation and small circuit area are basic requirements for consumer electronic products, especially for mobile devices with stringent battery restrictions. Therefore, approximate designs have been pervasively considered for implementations in image processing. Approximate multipliers have been utilized for image sharpening [Jiang et al. 2016b]. In this work, both the adder and multiplier in the image sharpening algorithm are replaced by approximate designs. Moreover, approximate dividers are used to detect the difference between two images to show the changes. This application is known as change detection.

5.1. Image Sharpening

The image sharpening algorithm computes $R(x, y) = 2I(x, y) - S(x, y)$ [Lau et al. 2009], where I is the input image, R is the sharpened image, and S is given by

$$S(x, y) = \frac{1}{4368} \sum_{m=-2}^{2} \sum_{n=-2}^{2} G(m+3, n+3) I(x-m, y-n), \tag{1}$$

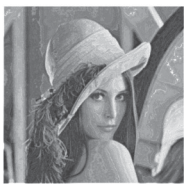
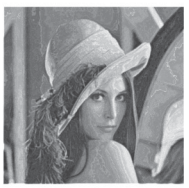
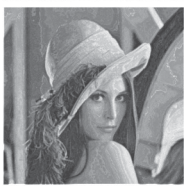
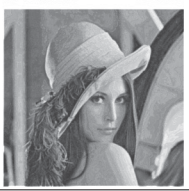
where G is a 5×5 matrix given by

$$G = \begin{bmatrix} 16 & 64 & 112 & 64 & 16 \\ 64 & 256 & 416 & 256 & 64 \\ 112 & 416 & 656 & 416 & 112 \\ 64 & 256 & 416 & 256 & 64 \\ 16 & 64 & 112 & 64 & 16 \end{bmatrix}. \tag{2}$$

Simulation results in Jiang et al. [2016b] show that AM2-15, AM1-15, TAM2-16, TAM1-16, BAM-16, AM2-13, AM1-13, ACM-4, ACM-3, TAM2-13, TAM1-13, BAM-17, AWTM-4, and BAM-18 achieve visually acceptable image sharpening results. Among these multipliers, the ones with a moderate hardware overhead (AM1-13, TAM2-13, TAM1-16, TAM1-13, BAM-17, and BAM-18) are selected in this work for image sharpening. Likewise, the approximate adders LOA, CSA, ETAII, and CSPA are selected. As the multiplication result of a 16×16 multiplier is 32 bits wide, 32-bit approximate adders are used for image sharpening. The value of parameter k is 8 for CSA, ETAII, and CSPA, and 16 for LOA.

The results for image sharpening using the selected approximate multipliers and adders are given in Table VII, and the accurate result is shown in Figure 22. The images sharpened using CSPA have unacceptable defects and some defects (white dots) can be seen in the image sharpened by AM1-13 and ETAII-8 when zooming into the images in Table VII. Other images processed by the approximate designs show similar quality with the accurate result. This is also confirmed by the peak signal-to-noise ratio (PSNR), as shown in Table VIII. The PSNRs of the images sharpened by a truncation based multiplier (i.e., TAM1-16, TAM2-13, TAM1-13, BAM-17, or BAM-18) are fixed as the adder is changed among LOA-16, CSA-8, and ETAII-8. This occurs because the

Table VII. Images Sharpened Using Different Approximate Adder and Multiplier Pairs

Approximate design	LOA-16	CSA-8	ETAII-8	CSPA-8
TAM1-16				
AM1-13				
TAM2-13				
TAM1-13				
BAM-17				
BAM-18				

lower 16 bits of the multiplication results generated by these multipliers are zeros, and hence only the higher half of an approximate adder (as an accurate 16-bit adder for LOA-16, CSA-8, or ETAII-8) is used.

The image sharpening algorithm is implemented in VHDL by using the selected approximate adders and multipliers. In the implementation, no pipelining or memory unit is used for exclusively showing the hardware characteristics of the approximate arithmetic circuits. It is synthesized by Synopsys DC using the same process, voltage, and temperature as in the simulation of the approximate adders. The 512×512 pixel

Table VIII. PSNRs of the Sharpened Images (*dB*)

Multiplier \ Adder	LOA-16	CSA-8	ETAII-8	CSPA-8
TAM1-16	46.97	46.97	46.97	25.01
AM1-13	45.21	45.06	36.86	24.20
TAM2-13	41.87	41.87	41.87	24.32
TAM1-13	41.42	41.42	41.42	24.35
BAM-17	40.09	40.09	40.09	25.19
BAM-18	33.99	33.99	33.99	24.21

Table IX. Delay, Power, and Area of Image Sharpening Using Approximate Multipliers and Adders

Multiplier	Adder	Delay (ns)	Power (mW)	PDP (pJ)	Area (um^2)	ADP ($um^2.ns$)
ArrayM	**CLAG**	**6.74**	**1.995**	**13.45**	**31,183.9**	**210,179.5**
TAM1-16	LOA-16	5.36	0.9723	5.21	18,139.0	97,225.0
TAM1-16	CSA-8	7.45	1.032	7.69	23,652.1	176,208
TAM1-16	ETAII-8	5.34	0.9643	5.15	18,056.8	96,423.3
AM1-13	LOA-16	5.41	1.193	6.45	26,644.0	144,144
AM1-13	CSA-8	7.41	1.377	10.20	30,586.5	226,646
AM1-13	ETAII-8	6.40	1.369	8.76	28,214.7	180,574
TAM2-13	LOA-16	5.25	1.055	5.54	17,057.8	89,553.5
TAM2-13	CSA-8	6.43	1.053	6.77	20,526.6	131,986
TAM2-13	ETAII-8	5.22	1.041	5.43	16,975.6	88,612.6
TAM1-13	LOA-16	5.25	0.9467	4.97	17,221.0	90,410.3
TAM1-13	CSA-8	7.45	0.9942	7.41	22,734.1	169,369
TAM1-13	ETAII-8	5.34	0.9350	4.88	17,138.8	89,464.5
BAM-17	LOA-16	6.14	1.226	7.53	14,993.8	92,061.9
BAM-17	CSA-8	7.36	1.247	9.17	16,533.0	121,683
BAM-17	ETAII-8	6.13	1.211	7.42	14,868.5	91,143.9
BAM-18	LOA-16	5.97	1.097	6.55	13,285.3	79,313.2
BAM-18	CSA-8	6.89	1.117	7.70	16,901.8	116,453
BAM-18	ETAII-8	5.96	1.076	6.41	13,156.0	78,409.8

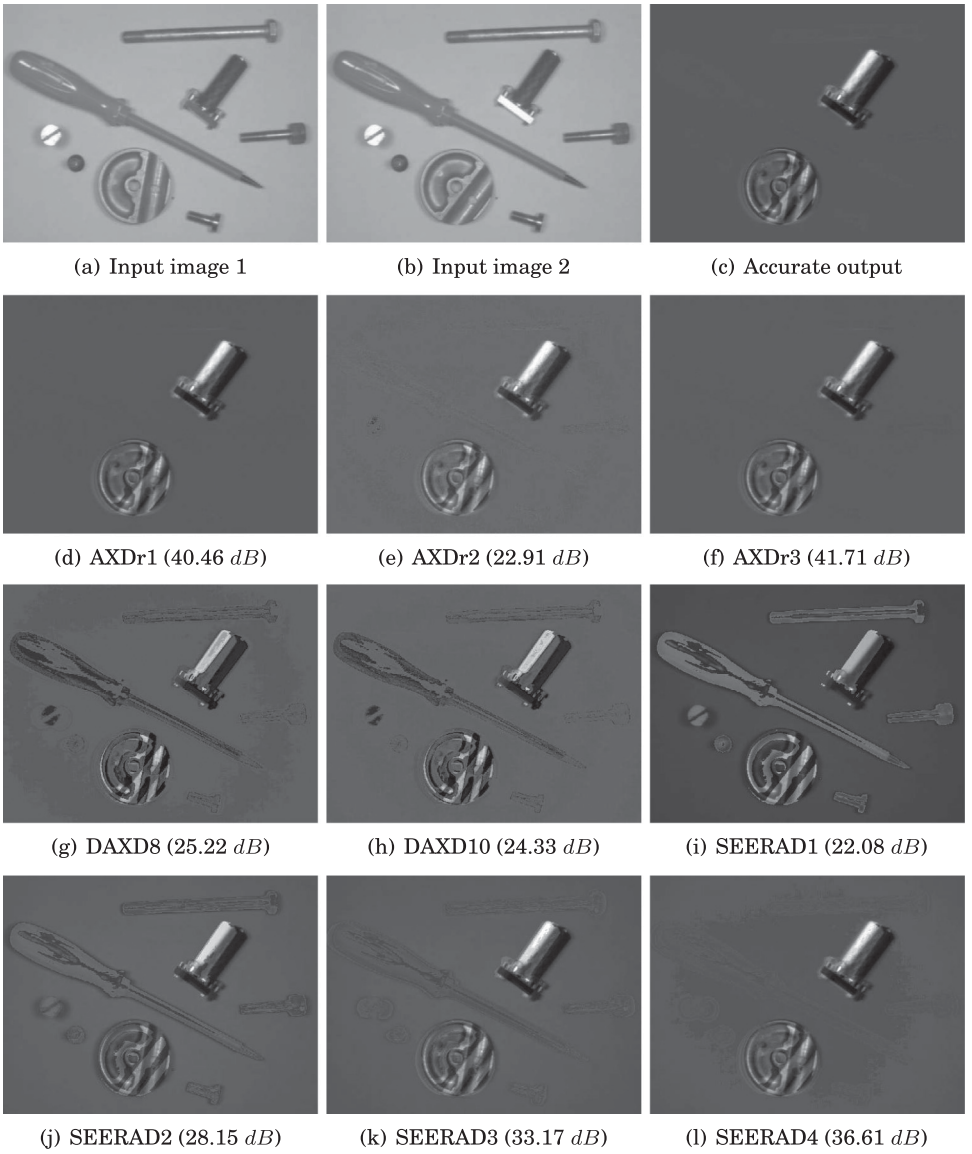
values of the image shown in Table VII are used as inputs for assessing the power dissipation using the PrimeTime-PX tool at a clock period of 10ns.

Table IX shows the circuit characteristics of image sharpening using approximate multipliers and adders. Using the same multiplier, the image sharpening implementations with LOA-16 and ETAII-8 show similar values for the delay, power, and area (except for AM1-13), whereas the implementations using CSA-8 have relatively larger values for these metrics. Likewise, the image sharpening circuits have similar characteristics using the same adder, but the schemes based on AM1-13, BAM-17, and BAM-18 show slightly larger values.

Compared to the image sharpening circuit using accurate multipliers and adders, the approximate designs using CSA-8 or AM1-13 achieve small improvement in terms of delay and area because CSA and AM1 are less efficient in delay and area compared to the other approximate designs. By using LOA-16, ETAII-8, TAM2-13, BAM-17, or BAM-18, the circuit can be 23% faster and saves as much as 53% in power, 58% in area, 64% in PDP, and 62% in ADP compared to the accurate design.

5.2. Change Detection

In the application of change detection, the ratio between two corresponding pixel values is calculated by a divider [Chen et al. 2016]. The changes in two images are then highlighted by normalizing the pixel ratios. In this section, 16/8 divider designs are used to calculate the division of two 8-bit gray-level images, as shown in Figure 23(a) and (b). To ensure a higher accuracy, the pixel values of the first image are multiplied by 64. As



(a) Input image 1 (b) Input image 2 (c) Accurate output

(d) AXDr1 (40.46 dB) (e) AXDr2 (22.91 dB) (f) AXDr3 (41.71 dB)

(g) DAXD8 (25.22 dB) (h) DAXD10 (24.33 dB) (i) SEERAD1 (22.08 dB)

(j) SEERAD2 (28.15 dB) (k) SEERAD3 (33.17 dB) (l) SEERAD4 (36.61 dB)

Fig. 23. Change detection using different approximate dividers.

HSD and FPD are designed for floating-point division, AXDr, DAXD, and SEERAD are selected for the change detection. Among the four types of AXDr, the triangle replacement has been shown to achieve the best results for image processing [Chen et al. 2016]. Therefore, three designs of AXDr with the triangle replacement of depth 8, AXDr1 (using approximate subtractor 1), AXDr2 (using approximate subtractor 2), and AXDr3 (using approximate subtractor 3), are used for the change detection. For DAXD, 8/4 and 10/5 accurate dividers are utilized in DAXD8 and DAXD10, respectively. Four accuracy levels are considered in SEERAD; they are referred to as SEERAD1, SEERAD2, SEERAD3, and SEERAD4. Moreover, the accurate array divider (denoted as ArrayD) is simulated for comparison.

Table X. Accuracy and Circuit Characteristics of the Change Detection Using Approximate Dividers

Divider	PSNR (dB)	Delay (ns)	Power (uW)	PDP (fJ)	Area (um^2)	ADP ($um^2.ns$)
ArrayD	**—**	**4.08**	**54.29**	**221.50**	**425.8**	**1,737.2**
AXDr1	40.46	3.85	50.71	195.23	415.5	1,599.7
AXDr2	22.91	4.36	54.08	235.79	408.2	1,779.6
AXDr3	41.71	4.58	40.55	185.72	376.2	1,722.9
DAXD10	24.33	2.43	40.25	97.84	375.7	912.9
SEERAD3	33.17	1.83	60.27	110.29	615.8	1,126.8
SEERAD4	36.61	2.43	70.62	181.33	765.4	1,859.9

Figure 23 shows the change detection results by the dividers, and the obtained PSNR value is shown in parentheses. It is clear that AXDr1, AXDr3, and SEERAD4 perform very well in the application of change detection, whereas the results by the other designs are of lower quality.

The characteristics of the change detection circuits are obtained by using the same tool and process as in the simulation of approximate adders. The clock period is 6ns, and the input combinations for the power evaluation are the two images shown in Figure 23(a) and (b) with 384×507 pixels. The synthesis results are shown in Table X. To be consistent with the other designs, AXDr1, AXDr2, and AXDr3 are implemented at the gate level rather than at the transistor level as in Chen et al. [2016].

Table X shows that the array-based dividers (ArrayD, AXDrs, and DAXD) are more power and area efficient than the rounding-based approximate dividers (SEERADs). However, they are very slow, except for DAXD, which uses a smaller accurate array divider. SEERADs are very fast, but they consume more power and area due to the use of look-up tables and the large output size (including the fractional part). In terms of PDP and ADP, DAXD10 has the smallest values followed by SEERAD3. Among the approximate dividers that produce excellent change detection results, SEERAD4 has the smallest PDP but the largest ADP. AXDr1 results in the largest PDP and the smallest ADP, whereas AXDr3 has both moderate PDP and ADP. To be more specific, the change detection using AXDr3 saves 25% of power and 12% of area compared to the accurate design. It is 40% faster with 18% reduction in PDP by using SEERAD4.

6. CONCLUSION

In this article, designs of approximate arithmetic circuits are reviewed. Their error and circuit characteristics are evaluated using functional simulation and hardware synthesis with an industrial 28 nm technology library.

Approximate adders. In general, approximate speculative adders show high accuracy and relatively small PDPs. The approximate adders using approximate full adders in the LSBs are slow, but they are power efficient with high ERs (due to the approximate LSBs), low average error, and moderate $NMED$ and $MRED$ values (due to the accurate MSBs). The error and circuit characteristics of the segmented and carry select adders vary with the prediction of the carry signals.

A truncated adder has a smaller $MRED$ (an indicator of a smaller error magnitude) than most approximate designs at a similar PDP, except for LOA and CSA. However, it has lower performance and significantly higher ER compared to the other approximate designs. As a result, a simple truncation of the LSBs in an adder causes a high ER and does not significantly improve the performance of the adder, but it has a relatively small error distance.

Approximate multipliers. For approximate multipliers, truncation of part of the partial products is an effective scheme to reduce hardware complexity while preserving a

moderate *NMED* and *MRED*. Similarly, truncating some LSBs of the input operands can efficiently reduce the hardware overhead of a multiplier and result in a moderate *MRED* (an indicator of the error magnitude) that is smaller than most other approximate designs, except for TAM1, TAM2, and ICM, for a similar PDP.

Albeit with a relatively low ER, UDM shows low accuracy in terms of the error distance and relatively high circuit overhead, because the 2×2 approximate multiplier is used to compute the most significant bits and accurate adders are utilized to accumulate the generated partial products. ICM has the lowest ER among all designs. When truncation is not used, multipliers approximated in the partial product tree tend to have a poor accuracy (except AWTM-3 and AWTM-4) and moderate hardware consumption, whereas multipliers using approximate counters or compressors are usually very accurate with relatively high power dissipation and hardware consumption. The approximate Booth multipliers show different characteristics in hardware efficiency and accuracy.

Approximate dividers. For the dividers, the approximate array dividers are slow, but they are hardware efficient with variable accuracy depending on the approximation parameters. The dividers based on curve fitting are very accurate and fast, but they require a large area and high power dissipation due to the utilization of look-up tables. The rounding based approximate dividers have very high speed, large area and power dissipation, with relatively low accuracy.

Application. Image sharpening is implemented using the selected approximate multipliers and adders. The accuracy and circuit characteristics of the image sharpening obtained by simulation indicate significant savings in hardware while producing similar sharpening quality as the accurate design. On average, the designs using approximate adders and multipliers with an acceptable accuracy consume only 55% of power, 62% of area, 51% of PDP, and 57% of ADP compared to the accurate design.

Approximate dividers are utilized in the change detection of images. The simulation and synthesis results show that the change detection circuits using the approximate array dividers, AXDr1 and AXDr3, are power and area efficient but very slow, whereas the circuit using the rounding-based approximate divider, SEERAD4, consumes more power and area with a high performance for excellent detection accuracy.

ACKNOWLEDGMENTS

The authors would like to thank Vincent Camus from EPFL for his contributions in proof reading the article.

REFERENCES

Tinku Acharya and Ajoy K. Ray. 2005. *Image Processing: Principles and Applications*. John Wiley & Sons.

Shaahin Angizi, Zhezhi He, Ronald F. DeMara, and Deliang Fan. 2017. Composite spintronic accuracy-configurable adder for low power digital signal processing. In *Proceedings of the 2017 18th International Symposium on Quality Electronic Design (ISQED'17)*. IEEE, Los Alamitos, CA.

Dursun Baran, Mustafa Aktan, and Vojin G. Oklobdzija. 2010. Energy efficient implementation of parallel CMOS multipliers with improved compressors. In *Proceedings of the ACM/IEEE International Symposium on Low Power Electronics and Design*. ACM, New York, NY, 147–152.

Kartikeya Bhardwaj, Pravin S. Mane, and Jorg Henkel. 2014. Power- and area-efficient approximate Wallace tree multiplier for error-resilient systems. In *Proceedings of the 2014 15th International Symposium on Quality Electronic Design (ISQED'14)*. 263–269.

Hao Cai, You Wang, Lirida A. B. Naviner, Zhaohao Wang, and Weisheng Zhao. 2016. Approximate computing in MOS/spintronic non-volatile full-adder. In *Proceedings of the 2016 International Symposium on Nanoscale Architectures (NANOARCH'16)*. IEEE, Los Alamitos, CA, 203–208.

Vincent Camus, Jeremy Schlachter, and Christian Enz. 2016. A low-power carry cut-back approximate adder with fixed-point implementation and floating-point precision. In *Proceedings of the 53rd Annual Design Automation Conference*. ACM, New York, NY, 127.

Linbin Chen, Jie Han, Weiqiang Liu, and Fabrizio Lombardi. 2015. Design of approximate unsigned integer non-restoring divider for inexact computing. In *Proceedings of the 25th edition of the Great Lakes Symposium on VLSI (GLSVLSI'15)*. ACM, New York, NY, 51–56.

Linbin Chen, Jie Han, Weiqiang Liu, and Fabrizio Lombardi. 2016. On the design of approximate restoring dividers for error-tolerant applications. *IEEE Transactions on Computers* 65, 8, 2522–2533.

Yuan-Ho Chen and Tsin-Yuan Chang. 2012. A high-accuracy adaptive conditional-probability estimator for fixed-width Booth multipliers. *IEEE Transactions on Circuits and Systems I: Regular Papers* 59, 3, 594–603.

Vinay K. Chippa, Debabrata Mohapatra, Anand Raghunathan, Kaushik Roy, and Srimat T. Chakradhar. 2010. Scalable effort hardware design: Exploiting algorithmic resilience for energy efficiency. In *Proceedings of the 2010 47th ACM/IEEE Design Automation Conference (DAC'10)*. ACM, New York, NY, 555–560.

Kyung-Ju Cho, Kwang-Chul Lee, Jin-Gyun Chung, and Keshab K. Parhi. 2004. Design of low-error fixed-width modified booth multiplier. *IEEE Transactions on VLSI Systems* 12, 5, 522–531.

S. R. Datla, M. A Thornton, and D. W. Matula. 2009. A low power high performance radix-4 approximate squaring circuit. In *Proceedings of the 2009 20th IEEE International Conference on Application-Specific Systems, Architectures, and Processors (ASAP'09)*. 91–97.

K. Du, P. Varman, and K. Mohanram. 2012. High performance reliable variable latency carry select addition. In *Proceedings of the 2012 Design, Automation, and Test in Europe Conference and Exhibition (DATE'12)*. 1257–1262.

Hadi Esmaeilzadeh, Adrian Sampson, Luis Ceze, and Doug Burger. 2012. Architecture support for disciplined approximate programming. *ACM SIGPLAN Notices* 47, 301–312.

Farzad Farshchi, Muhammad Saeed Abrishami, and Sied Mehdi Fakhraie. 2013. New approximate multiplier for low power digital signal processing. In *Proceedings of the 2013 17th CSI International Symposium on Computer Architecture and Digital Systems (CADS'13)*. IEEE, Los Alamitos, CA, 25–30.

Michael J. Flynn. 1970. On division by functional iteration. *IEEE Transactions on Computers* 100, 8, 702–706.

V. Gupta, D. Mohapatra, A. Raghunathan, and K. Roy. 2013. Low-power digital signal processing using approximate adders. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems* 32, 1, 124–137.

Jie Han. 2016. Introduction to approximate computing. In *Proceedings of the 2016 34th IEEE VLSI Test Symposium (VTS'16)*. IEEE, Los Alamitos, CA, 1.

Jie Han and Michael Orshansky. 2013. Approximate computing: An emerging paradigm for energy-efficient design. In *Proceedings of the 2013 18th IEEE European Test Symposium (ETS'13)*. IEEE, Los Alamitos, CA, 1–6.

S. Hashemi, R. Bahar, and S. Reda. 2015. Drum: A dynamic range unbiased multiplier for approximate applications. In *Proceedings of the IEEE/ACM International Conference on Computer-Aided Design*. IEEE, Los Alamitos, CA, 418–425.

S. Hashemi, R. Bahar, and S. Reda. 2016. A low-power dynamic divider for approximate applications. In *Proceedings of the 53rd Annual Design Automation Conference*. ACM, New York, NY, 105.

Junjun Hu and Weikang Qian. 2015. A new approximate adder with low relative error and correct sign calculation. In *Proceedings of the 2015 Design, Automation, and Test in Europe Conference and Exhibition (DATE'15)*. 1449–1454.

Jiawei Huang, John Lach, and Gabriel Robins. 2012. A methodology for energy-quality tradeoff using imprecise hardware. In *Proceedings of the 49th ACM Annual Design Automation Conference*. 504–509.

Honglan Jiang, Jie Han, and Fabrizio Lombardi. 2015. A comparative review and evaluation of approximate adders. In *Proceedings of 2015 ACM Great Lakes Symposium on VLSI*. 343–348.

Honglan Jiang, Jie Han, and Fabrizio Lombardi. 2016a. Approximate radix-8 booth multiplier for low-power and high-performance operation. *IEEE Transactions on Computers* 65, 8, 2638–2644.

Honglan Jiang, Jie Han, and Fabrizio Lombardi. 2016b. A comparative evaluation of approximate multipliers. In *Proceedings of the IEEE/ACM International Symposium on Nanoscale Architectures*.

Andrew B. Kahng and Seokhyeong Kang. 2012. Accuracy-configurable adder for approximate arithmetic designs. In *Proceedings of the 49th ACM Annual Design Automation Conference*. 820–825.

D. Kelly, B. Phillips, and S. Al-Sarawi. 2009. Approximate signed binary integer multipliers for arithmetic data value speculation. In *Proceedings of the 2009 Conference on Design and Architectures for Signal and Image Processing*.

Yongtae Kim, Yong Zhang, and Peng Li. 2013. An energy efficient approximate adder with carry skip for error resilient neuromorphic VLSI systems. In *Proceedings of the 2013 IEEE/ACM International Conference on Computer-Aided Design (ICCAD'13)*. 130–137.

Parag Kulkarni, Puneet Gupta, and Milos Ercegovac. 2011. Trading accuracy for power with an underdesigned multiplier architecture. In *Proceedings of the 2011 International Conference on VLSI Design*. 346–351.

Khaing Yin Kyaw, Wang Ling Goh, and Kiat Seng Yeo. 2010. Low-power high-speed multiplier for error-tolerant application. In *Proceedings of the 2010 IEEE International Conference of Electron Devices and Solid-State Circuits (EDSSC'10)*. 1–4.

Mark S. K. Lau, Keck-Voon Ling, and Yun-Chung Chu. 2009. Energy-aware probabilistic multiplier: Design and analysis. In *Proceedings of the 2009 International Conference on Compilers, Architecture, and Synthesis for Embedded Systems (CASES'09)*. 281–290.

Li Li and Hai Zhou. 2014. On error modeling and analysis of approximate adders. In *Proceedings of the 2014 IEEE/ACM International Conference on Computer-Aided Design (ICCAD'14)*. 511–518.

J. Liang, J. Han, and F. Lombardi. 2013. New metrics for the reliability of approximate and probabilistic adders. *IEEE Transactions on Computers* 62, 9, 1760–1771.

Chia-Hao Lin and Ing-Chao Lin. 2013. High accuracy approximate multiplier with error correction. In *Proceedings of the 2013 IEEE 31st International Conference on Computer Design (ICCD'13)*. IEEE, Los Alamitos, CA, 33–38.

Ing-Chao Lin, Yi-Ming Yang, and Cheng-Chian Lin. 2015. High-performance low-power carry speculative addition with variable latency. *IEEE Transactions on Very Large Scale Integration (VLSI) Systems* 23, 9, 1591–1603.

Cong Liu. 2014. *Design and Analysis of Approximate Adders and Multipliers*. Master's Thesis. University of Alberta, Canada.

Cong Liu, Jie Han, and Fabrizio Lombardi. 2014. A low-power, high-performance approximate multiplier with configurable partial error recovery. In *Proceedings of the 2014 Design, Automation, and Test in Europe Conference and Exhibition (DATE'14)*.

Cong Liu, Jie Han, and Fabrizio Lombardi. 2015. An analytical framework for evaluating the error characteristics of approximate adders. *IEEE Transactions on Computers* 64, 5, 1268–1281.

Cong Liu, Honglan Jiang, Fabrizio Lombardi, and Jie Han. 2017a. High-performance approximate unsigned multipliers with configurable error recovery. *IEEE Transactions on Circuits and Systems I*. Under revision.

Wei Liu and Alberto Nannarelli. 2012. Power efficient division and square root unit. *IEEE Transactions on Computers* 61, 8, 1059–1070.

Weiqiang Liu, Liangyu Qian, Chenghua Wang, Honglan Jiang, Jie Han, and Fabrizio Lombardi. 2017b. Design of approximate radix-4 booth multipliers for error-tolerant computing. *IEEE Transactions on Computers* PP, 99, 1.

Joshua Yung Lih Low and Ching Chuen Jong. 2013. Non-iterative high speed division computation based on Mitchell logarithmic method. In *Proceedings of the 2013 IEEE International Symposium on Circuits and Systems (ISCAS'13)*. 2219–2222.

Shih-Lien Lu. 2004. Speeding up processing with approximation circuits. *Computer* 37, 3, 67–73.

Jieming Ma, Ka Lok Man, Nan Zhang, Sheng-Uei Guan, and Taikyeong Ted Jeong. 2013. High-speed area-efficient and power-aware multiplier design using approximate compressors along with bottom-up tree topology. In *Proceedings of SPIE 8784: The 5th International Conference on Machine Vision (ICMV'12): Algorithms, Pattern Recognition, and Basic Technologies*. 87841Z.

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. *IEEE Transactions on Circuits and Systems* 57, 4, 850–862.

Sana Mazahir, Osman Hasan, Rehan Hafiz, Muhammad Shafique, and Jörg Henkel. 2017. Probabilistic error modeling for approximate adders. *IEEE Transactions on Computers* 66, 3, 515–530.

Jin Miao, Ku He, Andreas Gerstlauer, and Michael Orshansky. 2012. Modeling and synthesis of quality-energy optimal approximate adders. In *Proceedings of the 2012 ACM International Conference on Computer-Aided Design*. 728–735.

Joshua San Miguel, Jorge Albericio, Andreas Moshovos, and Natalie Enright Jerger. 2015. Doppelgänger: A cache for approximate computing. In *Proceedings of the 2015 48th International Symposium on Microarchitecture*. ACM, New York, NY, 50–61.

John N. Mitchell Jr. 1962. Computer multiplication and division using binary logarithms. *IRE Transactions on Electronic Computers* 4, 512–517.

D. Mohapatra, V. K. Chippa, A. Raghunathan, and K. Roy. 2011. Design of voltage-scalable meta-functions for approximate computing. In *Proceedings of the 2011 Design, Automation, and Test in Europe Conference and Exhibition (DATE'11)*. 1–6.

Amir Momeni, Jie Han, Paolo Montuschi, and Fabrizio Lombardi. 2015. Design and analysis of approximate compressors for multiplication. *IEEE Transactions on Computers* 64, 4, 984–994.

Vojtech Mrazek, Syed Shakib Sarwar, Lukas Sekanina, Zdenek Vasicek, and Kaushik Roy. 2016. Design of power-efficient approximate multipliers for approximate artificial neural networks. In *Proceedings of the 2016 International Conference on Computer Aided Design (ICCAD'16)*. 7.

Srinivasan Narayanamoorthy, Hadi Asghari Moghaddam, Zhenhong Liu, Taejoon Park, and Nam Sung Kim. 2015. Energy-efficient approximate multiplication for digital signal processing and classification applications. *IEEE Transactions on Very Large Scale Integration (VLSI) Systems* 23, 6, 1180–1184.

Vojin G. Oklobdzija, David Villeger, and Simon S. Liu. 1996. A method for speed optimized partial product reduction and generation of fast parallel multipliers using an algorithmic approach. *IEEE Transactions on Computers* 45, 3, 294–306.

Behrooz Parhami. 2000. *Computer Arithmetic*. Oxford University Press, Oxford, England.

Adrian Sampson, Werner Dietl, Emily Fortuna, Danushen Gnanapragasam, Luis Ceze, and Dan Grossman. 2011. EnerJ: Approximate data types for safe and general low-power computation. *ACM SIGPLAN Notices* 46, 6, 164–174.

Doochul Shin. 2010. Approximate logic synthesis for error tolerant applications. In *Proceedings of the 2010 Design, Automation, and Test in Europe Conference and Exhibition (DATE'10)*. IEEE, Los Alamitos, CA, 957–960.

Min-An Song, Lan-Da Van, and Sy-Yen Kuo. 2007. Adaptive low-error fixed-width Booth multipliers. *IEICE Transactions on Fundamentals of Electronics, Communications and Computer Sciences* 90, 6, 1180–1187.

Zdenek Vasicek and Lukas Sekanina. 2015. Evolutionary approach to approximate digital circuits design. *IEEE Transactions on Evolutionary Computation* 19, 3, 432–444.

Rangharajan Venkatesan, Amit Agarwal, Kaushik Roy, and Anand Raghunathan. 2010. MACACO: Modeling and analysis of circuits for approximate computing. In *Proceedings of the 2010 International Conference on Computer-Aided Design (ICCAD'11)*. 667–673.

Ajay K. Verma, Philip Brisk, and Paolo Ienne. 2008. Variable latency speculative addition: A new paradigm for arithmetic circuit design. In *Proceedings of the 2008 Conference on Design, Automation, and Test in Europe (DATE'08)*. 1250–1255.

Jiun-Ping Wang, Shiann-Rong Kuang, and Shish-Chang Liang. 2011. High-accuracy fixed-width modified Booth multipliers for lossy applications. *IEEE Transactions on Very Large Scale Integration (VLSI) Systems* 19, 1, 52–60.

Lei Wu and Ching Chuen Jong. 2015. A curve fitting approach for non-iterative divider design with accuracy and performance trade-off. In *Proceedings of the 2015 13th IEEE International New Circuits and Systems Conference (NEWCAS'15)*. 1–4.

Xinghua Yang, Yue Xing, Fei Qiao, Qi Wei, and Huazhong Yang. 2016. Approximate adder with hybrid prediction and error compensation technique. In *Proceedings of the IEEE Computer Society Annual Symposium on VLSI (ISVLSI'16))*. IEEE, Los Alamitos, CA, 373–378.

Zhixi Yang, Ajaypat Jain, Jinghang Liang, Jie Han, and Fabrizio Lombardi. 2013. Approximate XOR/XNOR-based adders for inexact computing. In *Proceedings of the 2013 13th IEEE Conference on Nanotechnology (IEEE-NANO'13)*. 690–693.

Rong Ye, Ting Wang, Feng Yuan, Rakesh Kumar, and Qiang Xu. 2013. On reconfiguration-oriented approximate adder design and its application. In *Proceedings of the 2013 IEEE/ACM International Conference on Computer-Aided Design (ICCAD'13)*. 48–54.

Reza Zendegani, Mehdi Kamal, Milad Bahadori, Ali Afzali-Kusha, and Massoud Pedram. 2017. RoBA multiplier: A rounding-based approximate multiplier for high-speed yet energy-efficient digital signal processing. *IEEE Transactions on Very Large Scale Integration (VLSI) Systems* 25, 2, 393–401.

Reza Zendegani, Mehdi Kamal, Arash Fayyazi, Ali Afzali-Kusha, Saeed Safari, and Massoud Pedram. 2016. SEERAD: A high speed yet energy-efficient rounding-based approximate divider. In *Proceedings of the 2016 Design, Automation, and Test in Europe Conference and Exhibition (DATE'16)*. IEEE, Los Alamitos, CA, 1481–1484.

Ning Zhu, Wang Ling Goh, and Kiat Seng Yeo. 2009. An enhanced low-power high-speed adder for error-tolerant application. In *Proceedings of the 2009 12th International Symposium on Integrated Circuits (ISIC'09)*. 69–72.

Received September 2016; revised January 2017; accepted April 2017