Block-Based Carry Speculative Approximate Adder for Energy-Efficient Applications

Farhad Ebrahimi-Azandaryani, Omid Akbari, Mehdi Kamal, Ali Afzali-Kusha, and Massoud Pedram

***Abstract*—In this brief, a low energy consumption block-based carry speculative approximate adder is proposed. Its structure is based on partitioning the adder into some non-overlapped summation blocks whose structures may be selected from both the carry propagate and parallel-prefix adders. Here, the carry output of each block is speculated based on the input operands of the block itself and those of the next block. In this adder, the length of the carry chain is reduced to two blocks (worst case), where in most cases only one block is employed to calculate the carry output leading to a lower average delay. In addition, to increase the accuracy and reduce the output error rate, an error detection and recovery mechanism is proposed. The effectiveness of the proposed approximate adder is compared with state-of-the-art approximate adders using a cost function based on the energy, delay, area, and output quality. The results indicate an average of 50% reduction in terms of the cost function compared to other approximate adders.**

***Index Terms*—Approximate computing, low power, speculative adder, energy-efficient.**

I. Introduction

In current digital systems, one of the key constraint is the thermal design power (TDP) could limit the performance of digital systems. One of the techniques which may help to obtain the most out of this constraint is the use of the approximate computing technique. It may be used for application domains such as multimedia and image processing, digital signal processing, wireless communication, machine learning, and data mining which are inherently error-resilient [1]. The technique may be used to attain more energy reduction and/or performance at the cost of some accuracy loss [1]. In recent years, various approximate computing techniques at different software/hardware levels have been proposed. Examples includes thread fusion and tunable kernels [2], approximate accelerator [3], imprecise logic or arithmetic unit [4], [5] and approximate Instruction Set Architecture (ISA) [6].

In this brief, we deal with approximate adders which are utilized as the basic operator in performing other arithmetic operations such as subtraction, multiplication, and division. Approximate adders have been received many attention by the designers. In the state-of-the-art approximate adders, where most of them are based on the carry propagate structures, the energy and speed gains have been achieved by hardware manipulation, logic simplification, and voltage over scaling [1]. While some of the adders were based on a configurable output accuracy (see [8]–[10]), others had a fixed accuracy level (see [11], [12]). The accuracy configurability imposed some overheads in terms of delay, area, and power which could limit their use in some applications where such re-configurability is not needed [7].

In this brief, we propose a high performance yet low power/energy block-based carry speculative approximate adder structure which is called BCSA adder. In this structure, the adder is partitioned into some non-overlapped parallel blocks, which in the worst-case, the carry output of a block is dependent on the carry output of the previous block. To reduce the critical path more, we suggest an approach to predict the carry output of a block based on its signals as well as of the next block. The structure has a low hardware complexity leading a low delay (on average, about one block) and a rather high quality. To achieve a lower accuracy loss, an error detection and recovery mechanism, which significantly reduces the output error rate, is proposed. The effectiveness of this adder is compared with some of the state-of-the-art approximate adders. Finally, the efficiency of the adder is studied using two image processing applications.

The rest of this brief is organized as follows. Some related works are reviewed in Section II. Section III presents the details of the proposed approximate adder. The efficacy of the proposed adder compared to those of the previous ones and also its applicability in image processing applications are studied in Section IV. Finally, this brief is concluded in Section V.

II. Related Works

In this section, some of the previous works in the field of the approximate adders are briefly reviewed.

Reconfigurable approximate carry look-ahead adder (RAP-CLA), which is an approximate adder designed based on the exact carry look ahead adder, was proposed in [7]. This adder was able to switch between the exact and approximate operating modes during the runtime. In RAP-CLA, fixed-size overlapped sub-blocks (windows) were used to calculate the carry output and sum bits. Some multiplexers were used to provide accuracy re-configurability which resulted in higher delay, power, and area in the exact operating mode compared to those of a pure exact CLA. An approximate carry skip adder (ACSA), which divided an n-bit exact CSA adder into l-bit sub-adders, was suggested in [8]. The ACSA reduced the critical path length by using a carry prediction technique. In

Manuscript received December 23, 2018; revised February 3, 2019; accepted February 19, 2019. Date of publication February 22, 2019; date of current version December 26, 2019. This brief was recommended by Associate Editor C.-T. Cheng. *(Corresponding author: Mehdi Kamal.)*

F. Ebrahimi-Azandaryani, O. Akbari, M. Kamal, and A. Afzali-Kusha are with the School of Electrical and Computer Engineering, College of Engineering, University of Tehran, Tehran 1439957131, Iran (e-mail: farhadebrahimi@ut.ac.ir; akbari.o@ut.ac.ir; mehdikamal@ut.ac.ir; afzali@ut.ac.ir).

M. Pedram is with the Department of Electrical Engineering, University of Southern California, Los Angeles, CA 90089-2562 USA (e-mail: pedram@usc.edu).

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

Digital Object Identifier 10.1109/TCSII.2019.2901060

1549-7747 © 2019 IEEE. Personal use is permitted, but republication/redistribution requires IEEE permission.
See http://www.ieee.org/publications_standards/publications/rights/index.html for more information.

Authorized licensed use limited to: University of Missouri Libraries. Downloaded on January 28,2021 at 20:51:45 UTC from IEEE Xplore. Restrictions apply.

addition, ACSA proposed an error magnitude reduction unit to increase the output accuracy. Due to using the additional circuit, the power consumption became large. Also, ACSA had a long critical path and the error rate was high.

Accuracy-configurable approximate adder (ACAA) discussed in [9] was a runtime quality adjustable segmented adder consisting of $\lceil n/l \rceil - 1$ overlapped sub-adders which operated in parallel. Each block adds $2l$ consecutive bits with l-bit of overlap with the previous one. Due to the large number of sub-adders, it, however, suffered from high power consumption and large area. A High Accuracy Block-based Approximate adder (HABA) was suggested in [10]. This adder proposed an error correction unit, which operated based on using the generate signal to decrease the accuracy loss. Although the power consumption and error rate were low, the critical path delay was still high due to its ripple carry propagation scheme.

In [11], an approximate carry ripple adder (SARA) has been proposed. In this structure, the adder has been segmented into some non-overlapped sub-adders. For the first and last exact FAs of the sub-adders, two modified full adders (FAs) have been employed. While this structure has a high output precision, its delay due to the ripple carry structure is high. In [14], an accuracy gracefully degrading adder (GDA) was proposed which used $\lceil n/l \rceil$ l-bits blocks to calculate the output. GDA utilized multiplexers between sub-blocks to select between the exact and approximate input carries. A generalized model of window-based approximate adders called GeAr was suggested in [13]. This adder utilizes $(n - l)$ l-bits sub-adders which are operating in parallel to produce the final output. The parallel scheme reduces the delay of the GeAr which uses an error correction unit to increase the output accuracy. The proposed error correction unit is a sequential circuit, which depending on the number of sub-adders, takes several cycles for correcting the output [7].

In [14], an error tolerant adder (ETA-I), where the add operation was divided into two independent parts such that the most significant sum bits were calculated by exact FAs while the least significant sum bits were generated by the XOR gates, was suggested. For the least significant part, modified XOR gates were used. Although this adder reduced the power consumption, the error rate/distance was large. In [15], an Approximate Reverse Carry Propagate Adder (RCPA) based on the RCAs was proposed. RCPA included three types of FAs where in the carry input had higher significant than the carry output, and the carry signal was propagated from the most significant bits to the least significant ones. Although the structure is considerably immune to the timing variations, it suffers from large delay due to the RCA structure. In [16], ETA-II was introduced which had a lower relative error distance compared to its predecessor. Both of the adders, however, suffered from large output errors.

III. Internal Structure of the Proposed Adder

The general architecture of an n-bit speculative approximate adder enhanced by a carry predictor unit is illustrated in Fig. 1. The add operation is performed by $\lceil n/l \rceil$ l-bit summation blocks working in parallel where l is the bit-length of each summation block. Each summation block includes an l-bit sub-adder, a Carry Predictor unit, and a Select unit. In this structure, the carry input of the i^{th} sub-adder, is chosen by $(i\text{-}1)^{\text{th}}$ Select unit from the carry signal generated by the $(i\text{-}1)^{\text{st}}$ Carry Predictor unit and the one generated by the $(i\text{-}1)^{\text{th}}$ sub-adder. Selecting the carry output of the Carry

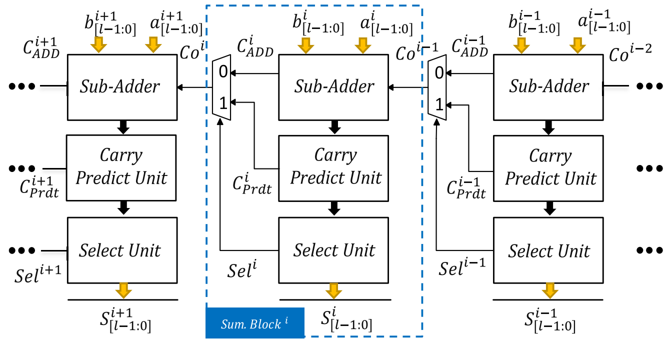

Fig. 1. The general architecture of an approximate adder equipped with carry prediction unit.

Predictor unit leads to a shorter critical path and lower energy consumption [1]. In this case, the dependency between the blocks are cut at the cost of some accuracy loss. Thus, the accuracy of the add operation depends on the accuracy of the Carry Predictor unit, and also, the policy of the carry output signal selection. In our proposed structure, in the worst-case, the length of a carry chain is equal to two blocks (*i.e.*, $2l$).

In most of the state-of-the-art approximate adders, the carry input of each block is chosen only based on the input signals of the previous blocks (see [10], [11]). In this brief, however, we propose a speculative approximate adder that the carry input of its i^{th} block is determined based on some input signals of the current block and those of the next one. As we will show, this approach results in a considerable accuracy improvement compared to the other approximate adders. In BCSA (see Fig. 1), the carry output of the i^{th} summation block (CO^i) is obtained from

$$CO^i = \overline{Sel^i}.C^i_{ADD} + Sel^i.C^i_{Prdt} \qquad (1)$$

where the Sel^i (C^i_{Prdt}) is the output signal of the Select (Carry Predictor) unit. Also, C^i_{ADD} is the carry output of the sub-adder of the i^{th} summation block assuming that the carry input of the block is zero. Thus, in the worst case, the carry is propagated through two blocks (generated in the first bit position of the i^{th} block and propagated in the i^{th} and $(i{+}1)^{th}$ blocks). Now, in this brief, we propose to determine the Sel^i, C^i_{Prdt}, and C^i_{Exact} using

$$Sel^i = K^{i+1}_0 + G^i_{l-1} \qquad (2)$$

$$C^i_{Prdt} = G^i_{l-1} \qquad (3)$$

$$C^i_{ADD} = P^i_{l-1}G^i_{l-2} + P^i_{l-1}P^i_{l-2}G^i_{l-3} + \cdots + \prod_{\text{k}=1}^{l-1} P^i_k.G^i_0 \qquad (4)$$

where the K^{i+1}_0 is the kill signal of the first bit position of the $(i{+}1)^{th}$ block (*i.e.*, $\overline{a^{i+1}_0}.\overline{b^{i+1}_0}$), G^i_{l-1} is the generate signal of the last bit position of the i^{th} block (*i.e.*, $a^i_{l-1}.b^i_{l-1}$), and P^i_{l-1} is the generate signal of the last bit position of the i^{th} block (*i.e.*, $a^i_{l-1} \bigoplus b^i_{l-1}$).

Based on (1)-(4), the structure of the proposed model is depicted in Fig. 2 where the carry propagate and parallel prefix adder structures (*e.g.*, KoggeStone, CLA, and RCA) could be employed for the sub-adders. Based on (1) and (2), the carry output of the i^{th} block is determined under four cases where, for each of them, either C^i_{Prdt} or C^i_{ADD} is selected. In the proposed approach, we focus on reducing the error as

Authorized licensed use limited to: University of Missouri Libraries. Downloaded on January 28,2021 at 20:51:45 UTC from IEEE Xplore. Restrictions apply.

much as possible by selecting a proper carry input on the next block. Therefore, in each of these cases, the carry output is selected with the highest possible accuracy. Thus, in the following paragraphs, we discuss the idea behind the carry output selection for the four cases of Sel^i.

- In the first case ($K_0^{i+1} = 0$ and $G_{l-1}^i = 0$), since $G_{l-1}^i = 0$, the $P(P_{l-1}^i)$ and $P(K_{l-1}^i)$ are 2/3 and 1/3, respectively. $P(x)$ denotes the probability of the signal x. Thus, the C_{Prdt}^i may be incorrect. In addition, when $K_0^{i+1} = 0$, the $P(P_0^{i+1})$ and $P(G_0^{i+1})$ are 2/3 and 1/3, respectively. Note that these probabilities have been obtained by assuming that the distribution of the 1 and 0 in the input operands bits are uniform. Therefore, the carry input of the $(i+1)^{th}$ block may be propagated to its most significant bits. Hence, to reduce the probability of the error propagation in the proposed adder, in this case, the Select unit circuit, chooses the C_{ADD}^i whose error probability is smaller than C_{Prdt}^i.
- In the second case ($K_0^{i+1} = 0$ and $G_{l-1}^i = 1$), because G_{l-1}^i is 1, the speculated carry (C_{Prdt}^i) is correct. Therefore, for this case, the C_{Prdt}^i is selected as the C_{in}^{i+1}.
- In the third case ($K_0^{i+1} = 1$ and $G_{l-1}^i = 0$), because K_0^{i+1} is 1, independent from the accuracy of the carry input of the $(i+1)^{th}$ block, the carry input is not propagated. Therefore, for shortening the critical path, we suggest to select C_{Prdt}^i as the carry output of the i^{th} block.
- In the fourth case (K_0^{i+1}=1 and G_{l-1}^i= 1), similar to the second case, since G_{l-1}^i is 1, the predicated carry is the same as the exact carry output of the block. Therefore, in the proposed approach, C_{Prdt}^i is selected as the C_{in}^{i+1}.

Among these cases, only in the first case, the carry is propagated in two blocks. Therefore, on average, the length of the carry propagation is close to one block.

In the third case, although the carry input of the block is killed and is not propagated, the carry input is employed to determine the first summation bit of the block. Therefore, if the carry input in this case is wrong, it impacts on the output accuracy of the summation. Hence, for improving the accuracy of the proposed adder, we suggest an error recovery unit which generates the first summation bit of the i^{th} block (S_0^i) by

$$S_0^{i+1} = (K_0^{i+1}.C_{ADD}^i) + (P_0^{i+1} \oplus C_{in}^{i+1}) \qquad (5)$$

Note that $P_0^{i+1} \oplus C_{in}^{i+1}$ ($C_{in}^{i+1} = C_{Prdt}^i$) is the approximate summation output in the first bit of the $(i+1)^{th}$ block denoted as AS_0^{i+1} in Fig. 2. Since the ERU is not on the critical path of adder, using the Error Recovery Unit (ERU) leads to improving the accuracy without increasing the delay of the proposed adder structure. Fig. 3 shows the functionality of the proposed speculative approximate adder with and without the ERU. The ERU imposes only about 3% and 2% power and area overheads, respectively. In the rest of this brief, we call the proposed adder enhanced by the ERU as BCSA$_{ERU}$.

IV. Results and Discussion

A. Error Metrics Evaluation

As previously discussed, the Select and Carry Predictor units determine the accuracy of the approximate adders. In this section, the accuracy of the proposed adder is evaluated compared to the four latest state-of-the-art approximate adders. These adders include GeAr [13], RAP-CLA [7], SARA [11], HABA [10], and the HABA equipped with

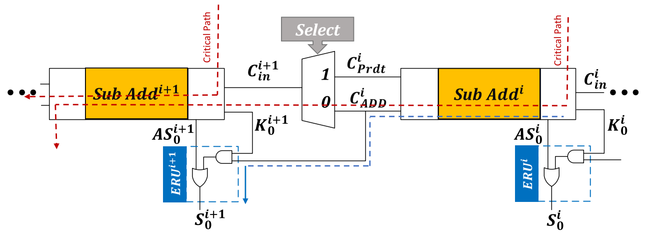

Fig. 2. The structure of the proposed adder with error recovery unit (ERU).

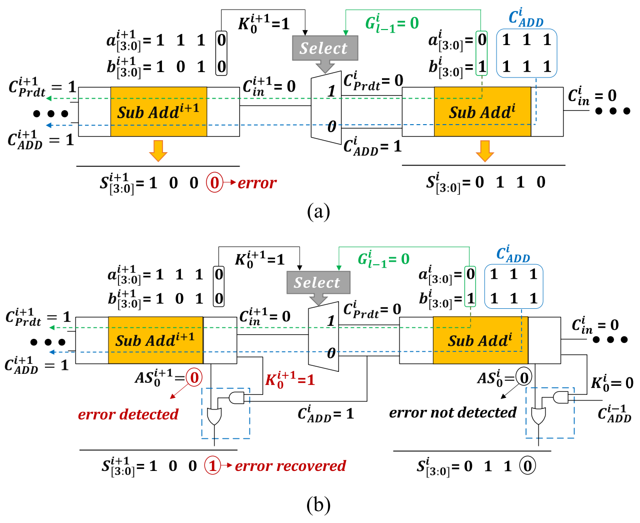

Fig. 3. An exmpale of functionality of the proposed approximate adder, (a) without ERU (BCSA W/O ERU), and (b) with ERU.

the ERU unit proposed in [10] (HABA$_{ERU}$). Note that the ERU circuit in HABA$_{ERU}$ is different from the one we suggested for BCSA$_{ERU}$. For this brief, three error metrics have been considered including Error Rate (ER), Normalized Mean Error Distance (NMED), and Mean Relative Error Distance (MRED) [7]. The NMED and MRED for an n-bit adder are obtained by

$$NMED = \frac{1}{2^n}\sum_{i=1}^{|N|}\frac{|S_i - S_i'|}{2^{|N|}} \qquad (6)$$

$$MRED = \frac{1}{|N|}\sum_{i=1}^{|N|}\frac{|S_i - S_i'|}{S_i} \qquad (7)$$

where $|N|$ shows the number of the input data samples, and S_i (S_i') is the exact (approximate) output of the add operation. The ER, NMED, and MRED of the considered adders under different block sizes and input operand widths are reported in Table I. While only the error metrics for the cases of block sizes of 2, 4 and 8 are reported in this table, we have performed this brief for the block sizes from 2 to 16.

These metrics have been extracted by applying 65,536 (10 million) uniform random numbers in the case of 8-bit (16- and 32-bit) adders. As the results show, in the case of 8-bit adder, the ER, NMED, and MRED of the BCSA$_{ERU}$ are 0 meaning that BCSA$_{ERU}$ is exact.

Among the studied adders, BCSA$_{ERU}$ and HABA [10] have the lowest ER compared to the other ones. On average, the ER of the BCSA is about 80% larger than HABA [10]. On the other hand, the NMED and MRED of the BCSA$_{ERU}$ (BCSA)

Authorized licensed use limited to: University of Missouri Libraries. Downloaded on January 28,2021 at 20:51:45 UTC from IEEE Xplore. Restrictions apply.

TABLE I
COMPARISON OF ACCURACY RESULTS FOR 8-, 16-, AND 32-BIT ADDERS

		8-bit			16-bit			32-bit		
Adder Type	Block Size	ER (%)	NMED ($\times10^{-4}$)	MRED ($\times10^{-4}$)	ER (%)	NMED ($\times10^{-4}$)	MRED ($\times10^{-4}$)	ER (%)	NMED ($\times10^{-4}$)	MRED ($\times10^{-4}$)
HABA	2	12.42	607	621	33.77	608	623	51.23	625	644
	4	1.37	137	136	4.29	155	157	9.18	156	158
	8	0	0	0	0.10	10	10	0.29	10	10
HABA$_{ERU}$	2	12.41	151	155	33.77	154	157	51.21	156	161
	4	1.36	9	9	4.28	10	10	9.17	10	10
	8	0	0	0	0.1	0.04	0.04	0.29	0.04	0.04
GeAr	2	30.06	605	707	61.77	625	730	65.37	625	730
	4	5.47	68	93	16.72	156	190	32.18	156	190
	8	0	0	0	0.68	10	12	2.25	10	12
RAP-CLA	2	15.54	606	646	36.45	626	670	55.71	625	670
	4	2.34	137	147	8.54	129	154	17.17	130	154
	8	0	0	0	0.34	10	11	1.12	10	11
SARA	2	14.43	379	484	35.05	417	531	65.44	563	589
	4	5.46	68	93	16.75	83	112	35.88	83	112
	8	0	0	0	6.09	5	7	17.45	5	7
BCSA$_{ERU}$	2	18.73	167	185	47.86	172	190	74.66	172	189
	4	0	0	0	5.89	20	26	16.66	20	26
	8	0	0	0	0	0	0	0.39	0.1	0.07
BCSA	2	27.47	395	428	60.98	417	554	87.62	417	554
	4	5.46	34	52	21.15	56	81	45.94	55	81
	8	0	0	0	6.20	2	3	18.04	3	4

TABLE II
THE ERRONEOUS OUTPUTS WITH RED SMALLER THAN A SPECIFIC VALUE WHEN $n = 8$ AND $l = 4$

RED(%)	**≤ 5%**	**≤ 10%**	**≤ 20%**	**≤ 50%**	**≤ 100%**
HABA & HABA$_{ERU}$	98.6%	98.6%	98.6%	98.6%	100%
GeAr	95.3%	95.8%	97.2%	99.5%	100%
RAP-CLA	97.7%	97.7%	97.9%	99.0%	100%
SARA	94.5%	94.5%	100%	100%	100%
BCSA W/O ERU	94.5%	94.5%	100%	100%	100%
BCSA W ERU	100%	100%	100%	100%	100%

are, on average, about 87% (52%) and 86% (40%) smaller than those of the other studies adders. In addition, for all the adders, by increasing the block size, the accuracy increases. As an example, in the case of 32-bit BCSA$_{ERU}$ (BCSA), the ER, NMED, and MRED reduced about 100% (93%), 100% (100%), and 100% (100%), respectively, when the block size was increased from 2 to 16. Table II shows the percentages of the summation results with Relative Error Distance (RED) smaller than a specified value for the all the considered 8-bit adders with the block size of 4. The RED is extracted by employing (7) without $1/|N|$ term. As the figures of this table show, all the outputs of the BCSA and BCSA$_{ERU}$ have RED smaller than 10% while in the case of HABA (HABA$_{ERU}$), about 1.5% of the outputs have RED values larger than 50%.

B. Design Parameters Evaluation

The hardware of BCSA, BCSA$_{ERU}$, HABA, HABA$_{ERU}$, GeAr, RAP-CLA and SARA were described by Verilog HDL and synthesized by Synopsys Design Compiler. All the studies in this brief have been performed using the typical process of the 15nm FinFET NanGate technology [17] with the operating voltage level of 0.8V and at the temperature of 25°C. Based on these implementations, the results of the delay and energy of the 32-bit studied adders with the block sizes of 2, 4, 6, 8, 12 and 16 are presented in Fig. 4(a) and (b), respectively. The design parameters of each adder under different block sizes have been reported in these plots where the corresponding block sizes are provided inside the circle. For extracting the power consumption (to obtain the energy consumption), up to 10M random stimuli were injected to the input of the netlist of the synthesized adders and the activity of the internal nodes of them were logged in the VCD format. Then, by employing Synopsys Primetime tool, the power consumptions of the adders were extracted based on their corresponding VCD file. Also, the reported delay for each adder was the critical path delay of the adder determined using the timing analysis of the Synopsys Design Compiler.

The studies show that the delay of SARA, HABA, HABA$_{ERU}$, BCSA$_{ERU}$, BCSA and RAP-CLA are, on average, 88%, 71%,78%, 7%, 5% and 20%, respectively higher than that of the GeAr. In terms of the energy consumption, GeAr consumes lower energy except for the block sizes of 6 and 16 where BCSA has the lowest energy dissipation, while SARA consumes the largest energy consumption among the considered adders. As the results show, SARA, HABA, HABA$_{ERU}$, BCSA$_{ERU}$, C and RAP-CLA consumes, on average, 81%, 60%, 87%, 5%, 4%, and 30%, respectively, higher energy compared to that of GeAr.

To obtain a better grasp of relative performances of the adders, the plots in Fig. 4(c)-(d) show the relations between the accuracies and the EDPs (energy-delay products) of the studied 8- and 16- and 32-bit adders with the block sizes of 2, 4, 8, 12 and 16. the size of each circle indicates the area usage of the adder. For the case of the 8-bit adder, while the EDP of the GeAr in some block sizes are smaller than that of the BCSA and BCSA$_{ERU}$, their accuracies are lower than that of the GeAr. On the other hand, in the case of the 32-bit adder, the EDP and accuracy of the BCSA and BCSA$_{ERU}$ are better than those of the GeAr. Compared to the other adders, the EDP of the BCSA and BCSA$_{ERU}$ are better for different block sizes and bit widths.

The study show that GeAr has the lowest area in the case of the block size of 2, while for the block size of 4, the smallest area is for the BCSA. For other studied block sizes, SARA has the smallest area compared to those of other studies approximate adders. The areas of the HABA, HABA$_{ERU}$, GeAr, BCSA$_{ERU}$, BCSA, and RAP-CLA are, on average, 14%, 19%, 30%, 15%, 12%, and 56%, respectively higher than that of the SARA. Also, as the results shown in Figure 4(c)-(e), the EDP and area usage of the BCSA is smaller than those of the exact adders. In the best (worst) case, the EDP of the BCSA is about 68% (13%) lower than those of the exact CLA at the cost of 0.055 (1e-6) MRED.

Finally, to assess the efficacy of the proposed adder in image processing applications, we employed it (16-bit width with block size of 4) in the sharpening and smoothing applications with an input set of ten image benchmarks. To evaluate the accuracy of the output images, we employed Peak Signal-to-Noise Ratio (PSNR) and Mean Structural Similarity Index Method (MSSIM) metrics used in evaluating the qualities of image processing applications [7]. The results of this brief show that using the BCSA in sharpening and smoothing image processing applications leads to, on average, 33% (12%) and 68% (70%), PSNR (MSSIM) reductions, respectively, compared to BCSA$_{ERU}$. Using the BCSA instead of CLA exact adder led to, on average, 14.4% and 20% energy efficiency and performance gains in the studied image processing applications. It should be pointed out that most portions of the energy and delay in these applications were due to the multiplications which were performed using exact multipliers. By employing the ERU, the PSNR (MSSIM) reductions in the cases of sharpening and smoothing image processing applications improved by 14% (4%) and 50% (67%), respectively compared to the BCSA without the ERU. Four output images of these applications with the lowest PSNR and MSSIM

Authorized licensed use limited to: University of Missouri Libraries. Downloaded on January 28,2021 at 20:51:45 UTC from IEEE Xplore. Restrictions apply.

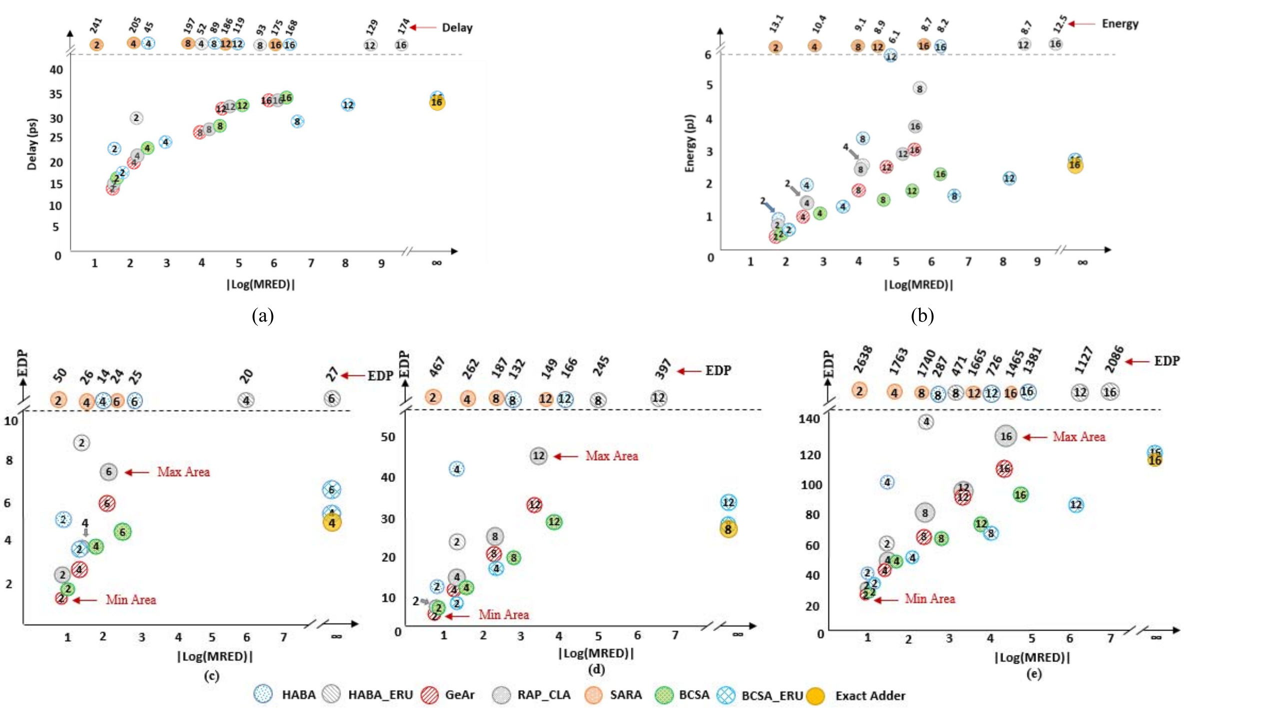

Fig. 4. The a) delay and b) energy consumption of the investigated 32-bit approximate adders and the relations between the accuracies and the EDPs of the investigated c) 8- and d) 16- and e) 32-bit adders under different block sizes.

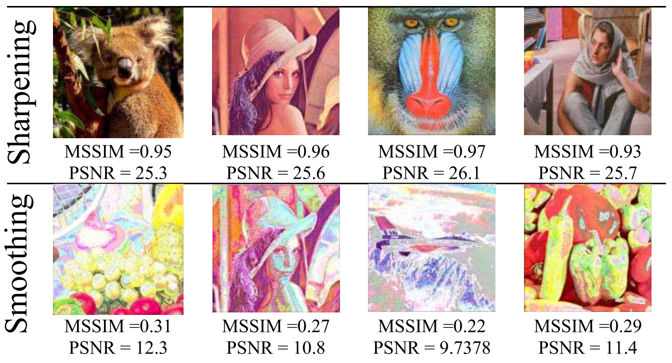

Fig. 5. Four output images of the sharpening and smoothing applications with lowest PSNR and MSSIM using the proposed BCSA_{ERU}.

in the case of using BCSA_{ERU} adder are shown in Fig. 5. The reported PSNR and MSSIM values are with respect to the images produced by the exact adder.

V. Conclusion

In this brief, we proposed a block-based carry speculative approximate adder (BCSA), which was based on dividing an exact adder into some non-overlapped blocks operated in parallel. Each block may be composed of any desired type of adders. In this adder, the length of carry chain was reduced to was utilized to estimate the carry. A select logic was suggested to speculate the carry input of each block based on some input operand bits of the current and next block. In addition, to decrease the accuracy loss, an error detection and recovery mechanism was suggested. Based on the results, for the different approximate operating modes, BCSA_{ERU} showed, on average, about 50% improvements in the cost function, compared to the recently proposed approximate adders.

References

[1] H. Jiang, C. Liu, L. Liu, F. Lombardi, and J. Han, "A review, classification, and comparative evaluation of approximate arithmetic circuits," *ACM J. Emerg. Technol. Comput. Syst.*, vol. 13, no. 4, 2017, Art. no. 60.

[2] M. Samadi, J. Lee, D. A. Jamshidi, A. Hormati, and S. Mahlke, "SAGE: Self-tuning approximation for graphics engines," in *Proc. Micro*, 2013, pp. 13–24.

[3] H. Esmaeilzadeh, A. Sampson, L. Ceze, and D. Burger, "Neural acceleration for general-purpose approximate programs," in *Proc. Micro*, 2012, pp. 449–460.

[4] B. K. Mohanty and S. K. Patel, "Area–delay–power efficient carry select adder," *IEEE Trans. Circuits Syst. II, Exp. Briefs*, vol. 61, no. 6, pp. 418–422, Jun. 2014.

[5] M. Bilal, S. Masud, and S. Athar, "FPGA design for statistics-inspired approximate sum-of-squared-error computation in multimedia applications," *IEEE Trans. Circuits Syst. II, Exp. Briefs*, vol. 59, no. 8, pp. 506–510, Aug. 2012.

[6] M. Kamal, A. Ghasemazar, A. Afzali-Kusha, and M. Pedram, "Improving efficiency of extensible processors by using approximate custom instructions," in *Proc. DATE*, 2014, pp. 1–4.

[7] O. Akbari, M. Kamal, A. Afzali-Kusha, and M. Pedram, "RAP-CLA: A reconfigurable approximate carry look-ahead adder," *IEEE Trans. Circuits Syst. II, Exp. Briefs*, vol. 65, no. 8, pp. 1089–1093, Oct. 2018.

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

[9] A. B. Kahng and S. Kang, "Accuracy-configurable adder for approximate arithmetic designs," in *Proc. DAC*, 2012, pp. 820–825.

[10] J. Hu and W. Qian, "A new approximate adder with low relative error and correct sign calculation," in *Proc. IEEE DATE*, 2015, pp. 1449–1454.

[11] W. Xu, S. S. Sapatnekar, and J. Hu, "A simple yet efficient accuracy configurable adder design," in *Proc. ISLPED*, 2017, pp. 1–14.

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

[13] M. Shafique, W. Ahmad, R. Hafiz, and J. Henkel, "A low latency generic accuracy configurable adder," in *Proc. DAC*, 2015, pp. 1–6.

[14] N. Zhu, W. L. Goh, W. Zhang, K. S. Yeo, and Z. H. Kong, "Design of low-power high-speed truncation-error-tolerant adder and its application in digital signal processing," *IEEE Trans. Very Large Scale Integr. (VLSI) Syst.*, vol. 18, no. 8, pp. 1225–1229, Aug. 2010.

[15] M. Pashaeifar, M. Kamal, A. Afzali-Kusha, and M. Pedram, "Approximate reverse carry propagate adder for energy-efficient DSP applications," *IEEE Trans. Very Large Scale Integr. (VLSI) Syst.*, vol. 26, no. 11, pp. 2530–2541, Nov. 2018.

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

[17] (2016). *NanGate—The Standard Cell Library Optimization Company*. [Online]. Available: http://www.nangate.com/

Authorized licensed use limited to: University of Missouri Libraries. Downloaded on January 28,2021 at 20:51:45 UTC from IEEE Xplore. Restrictions apply.