Accuracy Configurable Adders with Negligible Delay Overhead in Exact Operating Mode

FARHAD EBRAHIMI-AZANDARYANI, University of Tehran
OMID AKBARI, Tarbiat Modares University
MEHDI KAMAL, University of Tehran and University of Southern California
ALI AFZALI-KUSHA, University of Tehran
MASSOUD PEDRAM, University of Southern California

In this paper, two accuracy configurable adders capable of operating in approximate and exact modes are proposed. In the adders, which include a block-based carry propagate and a parallel prefix structure, the carry chains are cut off in the approximate mode limiting the carry chain depth to two blocks. In the case of parallel prefix adder, we propose a special carry generate tree equipped with a power gating means. In both of the proposed structures, the critical paths of the adders are not increased in the exact operating mode. Thus, the main objective of proposing these approximate adder structures is to present an accuracy configurable adder structure whose delay in the exact mode is almost the same as an exact adder. The efficacies of the proposed accuracy configurable adders are compared with some state-of-the-art adder structures using a 15nm CMOS technology. In addition, their efficacies are evaluated in two error-resilient applications. These studies show that the proposed carry-propagate adder has 22% (51%) lower energy consumption (error rate) compared to the best prior works. Also, the proposed parallel prefix adder provides, on average, 20% lower energy consumption compared to the exact parallel prefix adders.

CCS Concepts: • **Hardware → Arithmetic and datapath circuits**;

Additional Key Words and Phrases: Approximate adder, accuracy configurable, carry propagate adder, parallel prefix adder

ACM Reference format:
Farhad Ebrahimi-Azandaryani, Omid Akbari, Mehdi Kamal, Ali Afzali-Kusha, and Massoud Pedram. 2022. Accuracy Configurable Adders with Negligible Delay Overhead in Exact Operating Mode. *ACM Trans. Des. Autom. Electron. Syst.* 28, 1, Article 13 (December 2022), 14 pages.
https://doi.org/10.1145/3549936

Authors' addresses: F. Ebrahimi-Azandaryani and A. Afzali-Kusha, School of Electrical and Computer Engineering, University of Tehran, North Kargar st., Tehran, 14395-515, Iran; emails: {farhadebrahimi, afzali}@ut.ac.ir; O. Akbari, Department of Electrical and Computer Engineering, Tarbiat Modares University, Jalal AleAhmad, Tehran, 14115-111, Iran; email: o.akbari@modares.ac.ir; M. Kamal, School of Electrical and Computer Engineering, University of Tehran, North Kargar st., Tehran, 14395-515, Iran and Ming Hsieh Department of Electrical and Computer Engineering, University of Southern California, 3740 McClintock Ave, Los Angeles CA 90089; email: mehdi.kamal@usc.edu; M. Pedram, Ming Hsieh Department of Electrical and Computer Engineering, University of Southern California, 3740 McClintock Ave, Los Angeles CA 90089; email: pedram@usc.edu.

1084-4309/2022/12-ART13 $15.00
https://doi.org/10.1145/3549936

1 INTRODUCTION

Transistor counts in digital integrated circuits are growing rapidly, which gives rise to an increase in the power consumption. On the other hand, as the user applications are becoming more complex, higher performance computing is required to address the high performance and low power needs of these applications. Approximate computing is considered as one of the promising solutions to achieve a good balance between computational speed and power efficiency. Fortunately, there are a large number of applications in different domains which can tolerate a certain amount of output error. Examples are multimedia and image processing, wireless communication, machine learning, data mining, and some digital signal processing applications [1]. Approximate computing techniques may be employed at different design abstraction levels such as **Instruction Set Architecture (ISA)** [2], architecture [3], and circuit level [4–6].

Arithmetic components are the heart of computational units, and hence, their latencies and energy consumptions have strong impacts on characteristics of the whole system. Among the arithmetic units, adders are the basic operators for performing the other arithmetic operations like subtraction, multiplication, and division [1]. Therefore, improving the efficacy of adders can lead to significant improvement in the overall circuit's efficacy. In recent years, designing approximate adders has received a lot of attention from researchers (*e.g.*, see, [7–11]). Examples of the techniques employed in approximating the adders include logic simplification [5], voltage over scaling [12], and truncating the carry chain [13].

While most of the proposed approximate adders have a fixed output accuracy level [13, 15, 16], some state-of-the-art approximate adders provide configurable output accuracy levels (see, *e.g.*, [17, 18]). The runtime **accuracy configurability (AC)**, however, is achieved normally at the cost of some timing, area, and power overheads [19]. For example, in some circuits, generating the exact output requires several extra clocks (*e.g.*, [20]). In addition, most AC adders are designed based on a specific type of adder, limiting their usage in the low power/high performance applications (*e.g.*, [9]) when the underlying adder type is not the optimal one for a given application.

In this paper, we present two circuit-level adder structures including block-based carry propagates and tree-based AC adders whose operation delay overhead in the exact mode is negligibly small. Having the ability to dynamically configure the accuracy provides us with online **quality of service (QoS)** and energy management [19]. In the approximate operating mode of the proposed block-based carry propagate adder, the carry chain becomes limited to two blocks. In addition, to remove the delay overhead of the switches controlling the operating mode, we present a structure where the required logic is embedded into the carry generator logic. It should be mentioned that the summation block of this adder could be implemented based on either carry look-ahead adder or any carry propagate adder types. In the case of the proposed parallel prefix adder, we present a carry generation tree structure where a power gating technique, which shortens the carry chain and imposes some area overhead, is invoked. The structure of the latter adder in the approximate mode is similar to that of the approximate block-based adder. To the best of our knowledge, this is the first accuracy configurable parallel prefix adder which provides negligibly small delay overhead in the exact operating mode. The accuracy and design parameters of the proposed AC adders are compared with some of the state-of-the-art structures. Also, the efficacies of the structures in two error resilient applications are studied.

The rest of this paper is organized as follows. Prior works are studied in Section 2. The structural details of the proposed accuracy configurable adders are provided in Section 3. In Section 4, the efficacies of the proposed adders are compared to those of the prior works. Finally, the paper is concluded in Section 5.

2 RELATED WORKS

In this section, some of prior configurable accuracy adders are reviewed. An accuracy **gracefully degrading adder (GDA)** was proposed in [22]. In this adder, $[n/l]$ l-bits sub-adders were employed to calculate the output. The input carry of the sub-adders could be exact or predicted. The GDA utilized some multiplexers to select between the exact and approximate input carries. A generalized model of window-based approximate adders called GeAr was suggested in [20]. The adder utilized $(n-l)$ l-bits sub-adders operating in parallel to produce the final output. In addition, the GeAr utilized an error correction unit to increase the output accuracy. The error correction unit was a sequential circuit which took several cycles to correct the output depending on the number of sub-adders. In [21] an accuracy-configurable adder based on the carry look-ahead adder structure was proposed. In this adder, the first part of the conventional **carry-look ahead adder (CLA)** which generated the propagate and generate signals, was replaced by a **carry-maskable half-adder (CMHA)** structure providing the carry chain masking as well as the accuracy reconfigurability.

A **reconfigurable approximate carry look-ahead adder (RAP-CLA)** designed based on the exact CLA was proposed in [19]. The adder was able to switch between exact and approximate operating modes during the runtime. In the RAP-CLA, fixed-size overlapped blocks (windows) were employed to calculate the carry output and sum bits. To design dynamically energy-quality scalable adders with graceful degradation, a bit truncation approach was proposed in [16]. The key idea was to set the truncated input bits to some constant values (instead of zero) for maximizing the output quality. The authors of [17] proposed a **carry-predicting adder (**called **CPredA)** which did not either require additional circuits or recovery logic blocks for the carry-in prediction. It was constructed using one **carry predicting full adder (CPFA)** and one **carry-maskable full adder (CMFA)**. These two modified full adders provided reconfigurability by predicting the carry and cutting the critical path in the desired length. In [18], an accuracy-configurable block-based **Carry Look-ahead Adder (AC-CLA)** whose structure employed the **voltage over scaling (VOS)** as the online approximation knob for adjusting the output quality was proposed. Under a given accuracy level, some blocks of the AC-CLA operated at the low operating voltage level (approximate mode), whereas other blocks operated at the nominal operating voltage level (accurate mode). In this structure for passing signals from low voltage domain to the high voltage domain, voltage level shifter was required imposing delay and power overheads.

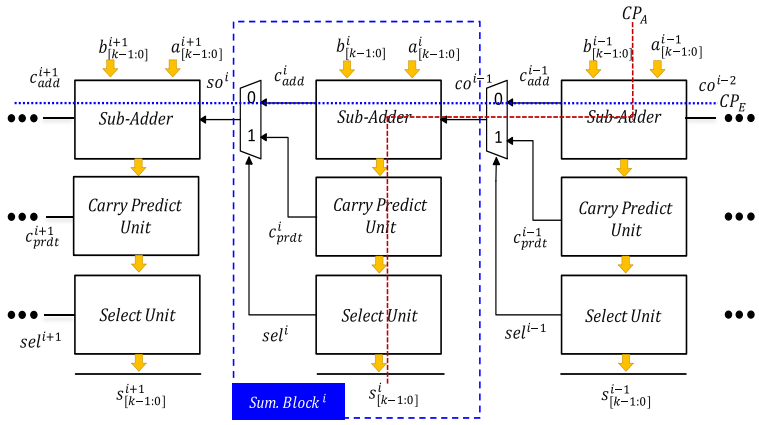
An accuracy configurable adder whose architecture consisted of an accurate part and an inaccurate part was proposed in [15]. In the inaccurate part, except for the two most significant sum bits, the sum bits were assigned a constant value of 1. Also, the most significant bit of the inaccurate part has been employed to predict (by using AND gate) the carry input of the accurate part. To provide the ability to switch from the approximate to exact mode, all of these works added additional circuits to the critical path of the adder. This made the delay of the adders in the exact mode higher than that of the conventional exact adder. While the main objective of proposing the approximate adder structures in this work is to present an accuracy configurable adder structure whose delay in the exact mode is almost the same as an exact adder.

3 THE PROPOSED ADDER STRUCTURES (ND-ACA)

In this section, the architectures of the proposed approximate adder for the two adder classes of **carry propagate adder (CPA)** and **parallel prefix adder (PPA)** are presented.

3.1 Carry-Propagate Adder (ND-ACA$_{CP}$)

The typical architecture of an approximate adder equipped with a carry prediction unit is illustrated in Figure 1. In this structure, the adder is segmented into several summation blocks. The a



CP_E : longest carry propagation path (exact mode) CP_A : longest carry propagation path (approximate mode)

Fig. 1. The general architecture of an accuracy configurable block-based CPA equipped with the carry prediction unit.

and b are the n-bit input operands of the adder while s and c_{out} are the sum and carry outputs, respectively. The add operation is performed by $\lceil n/k \rceil$ k-bit summation blocks. For this adder, the longest carry propagation paths in the exact and approximate modes are shown as CP_E and CP_A, respectively. Each summation block includes a k-bit sub-adder, a *Carry Predict* unit, and a *Selection* unit. The sub-adders may be composed of any desired adder type, *e.g.*, **Ripple Carry Adder (RCA), Carry Look-ahead Adder (CLA)**, and **Carry Skip Adder (CSA)**. Additionally, selecting the output carry by the carry prediction unit results in a shorter critical path and lower energy consumption [5]. In this case, the dependency between the blocks is removed at the cost of some accuracy loss. Therefore, the accuracy of the add operation, in the approximate mode, depends on the accuracy of the carry prediction unit and the output signal of the carry selection unit. This structure provides multi-precision quality at run-time while imposing some power and area overheads compared to the exact operating mode. In Figure 1, the output carry of the i^{th} block (*i.e.*, c_O^i) is determined by [19]

$$c_O^i = g_{k-1}^i + p_{k-1}^i g_{k-2}^i + \cdots + \prod_{n=1}^{k-1} p_n^i g_0^i + \prod_{n=0}^{k-1} p_n^i c_{in}^i \tag{1}$$

where g_{k-1}^i, p_{k-1}^i, and c_{in}^i are the generate ($a_{k-1} \bullet b_{k-1}$), propagate ($a_{k-1} \oplus b_{k-1}$) of the $k-1^{th}$ bit in i^{th} block, and the carry input signal of this block, respectively.

Let us rephrase (1) for the i^{th} block as follows

$$c_o^i = g_{k-1}^i + p_{k-1}^i \bullet \left(g_{k-2}^i + \cdots + \prod_{n=1}^{k-2} p_n^i g_0^i + \prod_{n=0}^{k-2} p_n^i c_{in}^i \right) \tag{2}$$

where $\bullet$ denotes the Boolean AND operation. In addition, C_{in}^j is determined by the previous summation block. Next, we rewrite (2) as where, in this case, the carry output of the i^{th} sub-adder block (in its exact operating mode) is obtained through a multiplexer whose selector signal is P_{k-1}^i.

$$c_{o-exact}^i = \begin{cases} g_{k-2}^i + \cdots + \prod_{n=1}^{k-2} p_n^i g_0^i + \prod_{n=0}^{k-2} p_n^i c_{in}^i & p_{k-1}^i = 1 \\ g_{k-1}^i & p_{k-1}^i = 0 \end{cases} \tag{3}$$

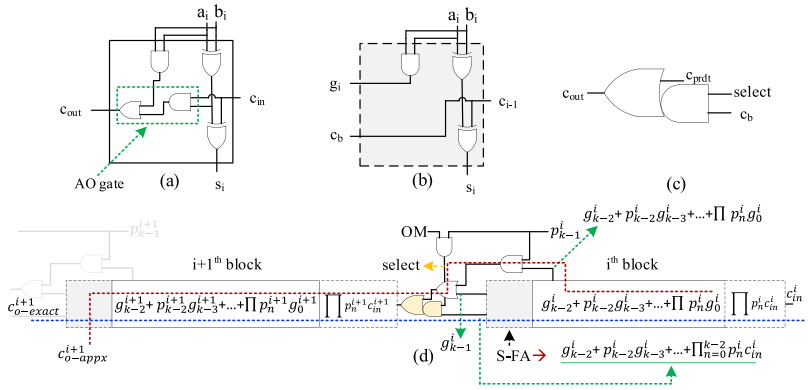


Fig. 2. (a) Conventional FA, (b) the proposed S-FA, (c) the employed AO gate, and (d) the general architecture of the proposed accuracy configurable CPA.

The multiplexing operation inside the FA is implemented by a AND-OR gate (see Figure 2(a)). Now, based on this, we suggest omitting the AO gate inside the conventional block-based adder structure and using the AO gate of the summation block (based on (3)) in place of that in the internal structure of the block-based AC adder. Using this modification, the critical path of the exact mode of the add operation is shortened by $\lceil n/k \rceil \times D_{AO}$ (D_{AO} is the delay of the AO gate) compared to the delay of the conventional block-based accuracy configurable adders. In this case, the delay of the adder in the exact mode becomes the same as that of the conventional CPA making the approach free of any delay overheads. On the other hand, in the approximate mode, the output carry of each block in the proposed adder is determined by assuming that the carry input for that block is logic 0. Thus, the output carry in the approximate mode (c^i_{o-appx}) is obtained from

$$c^i_{o-appx} = g^i_{k-1} + p^i_{k-1} \bullet \left(g^i_{k-2} + \cdots + \prod_{n=1}^{k-2} p^i_n g^i_0 \right) \tag{4}$$

To implement the proposed approach, we suggest to employ our suggested **simplified full adder (S-FA)** shown in Figure 2(b) which is placed in the most significant bit position of the sub-adder blocks. The S-FA provides run-time accuracy re-configurability when used along with the suggested AO gate structure shown in Figure 2(c). S-FA has three outputs comprising a sum signal denoted by s, a carry bypass (c_b) signal which is directly connected to its c_{in}, and a generate signal (g). Since the S-FA is employed as the last FA in the sub-adder block, the two inputs of the AO gate are c_b and g^i_{k-1}. The general structure of the proposed approximate block-based adder is depicted in Figure 2(d). In this structure, the *select* signal of the AO gate is $OM \bullet p^i_{k-1}$, where the signal ***OM* (operating mode)** determines the operating mode of the summation block. Based on these explanations, the output carry of the i^{th} block of the proposed adder may be formulated as

$$c^i_o = \left(g^i_{k-1} + p^i_{k-1} \bullet \left(g^i_{k-2} + \cdots + \prod_{n=1}^{k-2} p^i_n g^i_0 \right) \right) + OM \bullet p^i_{k-1} \bullet \left(g^i_{k-2} + \cdots + \prod_{n=1}^{k-2} p^i_n g^i_0 + \prod_{n=0}^{k-2} p^i_n c^i_{in} \right) \tag{5}$$

Here, when OM = 1, the above equation reduces to (1) and when OM = 0, it reduces to (4). In this work, we only consider the proposed CPA structure in the cases that all the OM signals of the blocks are the same. Also, it is possible to control the OM signals differently providing us

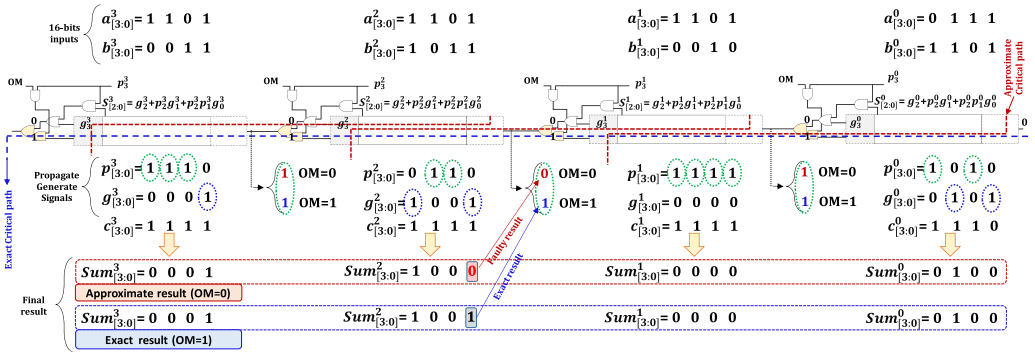


Fig. 3. The function and the carry chain of the proposed carry propagate adder under two input samples.

with multi-accuracy structures. The function and the carry chain of the proposed adder under two input samples and corresponding propagate and generate signals with each pair of bits, are illustrated in Figure 3. In the figure, $c^i[j]$ is the exact input carry of the j^{th} bit position in i^{th} block. As depicted in the figure, the least significant bit of the 2^{nd} block is wrong. It originates from the fact that when $OM = 0$ (approximate mode), the input carry of 2^{nd} block is the approximate output carry of its previous block (*i.e.*, c^1_{o-appx}). More specifically, in the exact operation, based on the input operands, the 1^{st} block propagates c^1_{in} (c^0_O) to the next block where $c^1_{in} = 1$. However, in the approximate mode, the predicted carry for the next block is generated without considering the input carry and thus $c^2_{in} = 0$. However, the correct input carry of 2^{st} block should be 1. Hence, in this particular bit position, an incorrect result is generated.

3.2 Parallel Prefix Adder (ND-ACA$_{PP}$)

The general structure of a **parallel prefix adder (PPA)** has three stages, including pre-processing, carry propagation, and final summation [23, 24]. In the first stage, the generate (g_i) and propagate (p_i) signals of the i^{th} bit position ($n > i \geq 0$) are obtained. In this case, the input carry of the $(i+1)^{st}$ bit position (c_{i+1}) is equal to g_i. Also, the sum signal in the i^{th} bit position (s_i) is $p_i \oplus c_i$. The Boolean operation $\circ$ (called prefix operation) is widely used in PPAs and is applied on the pair of generate and propagate signals. Generally, the prefix operation is defined as

$$(g, p) \circ (g', p') = (g + p \bullet g', p \bullet p') = (g'' \cdot p'') \tag{6}$$

The pair of $(g_{i:j}, p_{i:j})$ (where $j \leq i$) denotes the carry generation and propagation signals for the bits in the range of $(i, i-1, i-2, \ldots, j+1 \cdot j)$ inclusive of i and j, which could be determined by associating the two overlapping terms of $(g_{i:k}, p_{i:k})$ and $(g_{r:j}, p_{r:j})$ (where $j < k \leq r < i$) using (6). Thus, in this case, $g_{i:j}$ and $p_{i:j}$ are obtained from

$$g_{i:j} = g_{i:k} + p_{i:k} \bullet g_{r:j} \tag{7}$$

$$p_{i:j} = p_{i:k} \bullet p_{r:j} \tag{8}$$

Based on these explanations, Figure 4(a) shows the general structure of the proposed accuracy-configurable parallel prefix adder, its structure during the approximate mode, and its basic blocks. The gray circle is used to show passing the generate and propagate signals to the next level without applying any function on them. The square block generates the p and g signals. The gray diamond block implements the prefix operation based on (7) and (8). The dot filled diamond is an extension of the gray diamond and performs the prefix operation on the three input pairs of (p_i, g_i), (p_{i-1}, g_{i-1}), and (p_{i-2}, g_{i-2}). This block is only used in the first level of the carry generation step to form

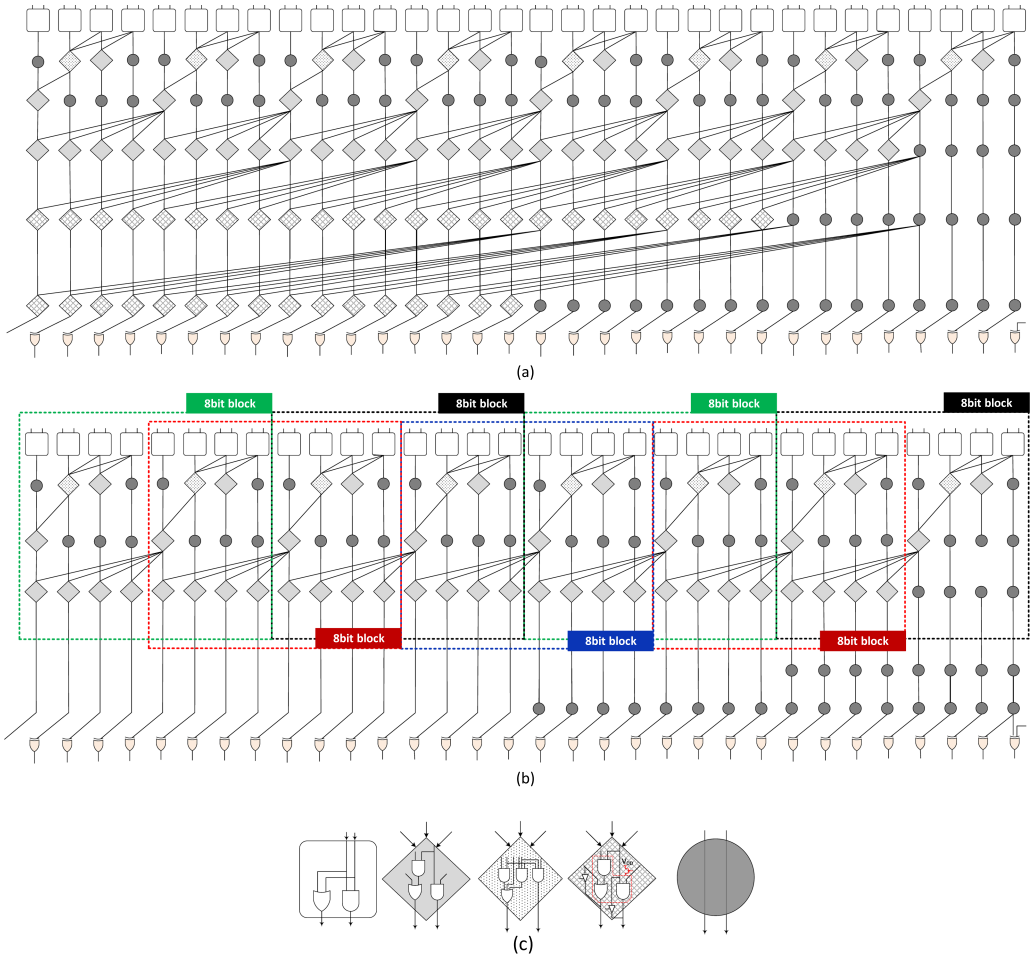


Fig. 4. (a) The general structure of the proposed 32-bit accuracy configurable PPA (b) its equivalent circuit in the approxmiate mode (c) internal structure of logic blocks.

the carry propagation in a block-based approach without increasing the carry generation stages. To support accuracy configurability, we suggest the hatched filled diamond, which implements the 2-input prefix operation (similar to the gray diamond block). In other words, in the approximate mode (as determined by the *OM* signal in CPA structure), its AND and OR gates are turned off (using power gating technique) and the outputs of the block become $g_{i:j} = g_{i:k}$ and $p_{i:j} = p_{i:k}$. Hatched block is used instead of the gray diamonds to cut the carry chains and reduce the latency as well as the power consumption (by power gating of the unused logics). This block in the exact mode generates the proper generate (g) and propagate (p) signals based on its g and p input signals and by using two-level logics (AND-OR logic). Thus, in the exact mode, it imposes two-level logic delay. However, in the case of the approximate mode, its g and p input signals would be passed to the next level through the tristate buffers, and only impose one-level logic delay (buffer logic).

This means that the proposed PPA structure with a bit-width equal or lower than 8 bits ($i \leq 3$) would operate just like the exact adder limiting the application of the structure to data-paths with bit widths larger than 8. Thus, in the proposed structure, each two adjacent n-bit blocks have an overlap of 4 bits. As an example, in Figure 4(b), by exploiting the hatch filled diamond blocks in the

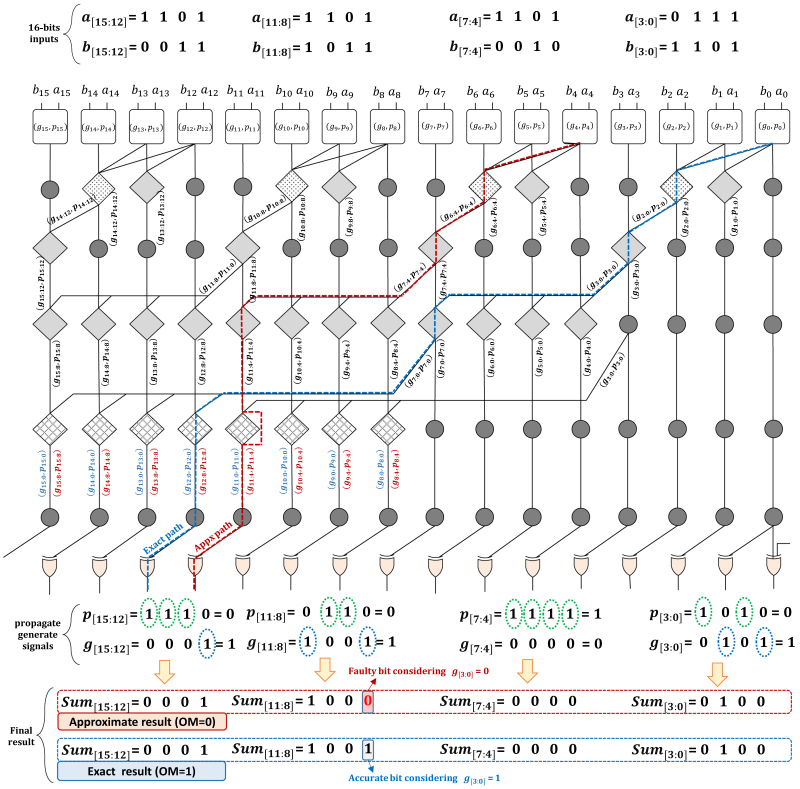


Fig. 5. The function and the carry chain of the proposed parallel prefix adder under two input samples.

last two levels of the carry reduction step, the number of stages in the critical path (in the carry generation step) during the approximate mode operation is 3 (plus two tristate gates) whereas in the exact operating mode, the critical path has five stages (log 32).

Thus, in the approximate mode, the 32-bit adder is constructed by seven overlapping blocks (as shown in Figure 4(b)). Moreover, in this figure, the dashed rectangle shows the independent adder blocks in the approximate operating mode. As mentioned before, the hatched filled diamond can be employed in the last levels ($3 < Level\ \# \leq \log n$) of an n-bit proposed PPA. In this case, the number of 8-bits blocks is $2^{(\log_2 n)-2} - 1$ where $n \in 2^i\ and\ i \geq 3$. Based on this, the smallest block size for the proposed PPA is 8 bits. For example, in our proposed structure, to create a 32-bit proposed adder module, seven 8-bit blocks are needed ($2^{(\log_2 32)-2} - 1 = 7$). Finally, while the considered tristate buffers (for bypassing the powered off gates) in the hatched-filled diamonds were not placed in the critical path, the total capacitance of the critical path increases negligibly. This is due to the fact that this capacitance increase is considerably smaller than those of the output of the blocks and the wires of the critical path. Our studies showed that the delay overhead was below 1%.

Finally, the function and the carry generation tree of the proposed adder under two input samples are illustrated in Figure 5. Just like the CPA structure, when $OM = 1$, the carry generation tree is in the exact mode. It means that the carry input of the i^{th} bit position is determined by (6) and (7). Otherwise, it is in the approximate mode and the input carry of the i^{th} bit ($4q \leq i < 4(q+1)$) is determined by $(g_{i-1:j}, p_{i-1:j})$, where $j = 4(q-1)$ and $q \geq 1$. Note that in the case of the exact operating mode, j is 0. As depicted in the figure, the 8$^{\text{th}}$ bit position of the final result, whose carry input is determined by pair $\{(g_{7:4} \cdot p_{7:4}) |\ q = 2\ \ and\ j = 4\}$, is wrong. It originates from the fact that

Table 1. Comparison of Accuracy Results for 8-, 16-, and 32-bit Adders

		8-bit			16-bit			32-bit		
Adder Type	BS	ER (%)	NMED ($\times 10^{-4}$)	MRED ($\times 10^{-4}$)	ER (%)	NMED ($\times 10^{-4}$)	MRED ($\times 10^{-4}$)	ER (%)	NMED ($\times 10^{-4}$)	MRED ($\times 10^{-4}$)
ND-ACA$_{CP}$	2	18.73	167	185	47.86	172	189	74.66	172	190
	4	0	0	0	4.4	20	26	12.4	20	26
	8	0	0	0	0	0	0	0.40	0.07	0.1
GeAr	2	30.06	605	707	61.77	625	730	65.37	625	730
	4	5.5	68	93	16.7	156	190	32.2	156	190
	8	0	0	0	0.68	10	12	2.25	10	12
RAP-CLA	2	15.54	606	646	36.45	625	670	55.71	626	670
	4	2.3	137	147	8.54	129	154	17.2	130	154
	8	0	0	0	0.34	10	11	1.12	10	11
GDA	2	27.47	395	428	60.98	417	554	87.62	417	554
	4	5.5	34	52	21.2	56	81	45.9	55	81
	8	0	0	0	6.20	2	3	18.0	3	4
AC-CLA	2	25.44	621	742	59.67	689	804	91.2	689	804
	4	3.75	59	70	13.91	98	110	28.4	100	110
	8	0	0	0	0.57	16	18	14.5	10	11
CMHA	2	68.32	146	191	96.83	156	200	99.87	157	203
	4	51.49	1.24	1.58	87.95	10	13	98.64	10	13
	5	0	0	0	68.32	0.6	0.8	89.91	0.6	0.9
ND-ACA$_{PP}$	2	-	-	-	-	-	-	-	-	-
	4	-	-	-	-	-	-	-	-	-
	8	0	0	0	4.4	20	26	12.4	20	26
	16	0	0	0	0	0	0	0.40	0.07	0.1

when $OM = 0$, the input carry of this particular bit position is $g_{7:4} + p_{7:4}g_{3:0} = 0$ where $g_{7:4}$ is 0 and $g_{3:0}$ is also 0 because of the bypass of the exact value of $g_{3:0}$ (see (7)). While the correct value of $g_{3:0}$ is 1 and since $p_{7:4} = 1$, the exact input carry to this bit position should be $g_{7:4} + p_{7:4}g_{3:0} = 1$. This is the reason for having the wrong result.

4 RESULTS AND DISCUSSION

In this section, the accuracy and design parameters of the proposed adders are compared to those of the prior work. Also, the efficacies of the proposed adders in error-resilient applications are studied.

4.1 Accuracy Analysis Comparison of Accuracy

The accuracies of the proposed carry propagate and parallel prefix ND-ACA (*i.e.*, ND-ACA$_{CP}$ and ND-ACA$_{PP}$) in approximate operating mode are compared with those of five **state-of-the-art (SOTA)** AC adders in Table 1. The set of considered SOTA run-time reconfigurable adders consists of AC-CLA [18], RAP-CLA [19], GeAr [20], CMHA [21], and GDA [22]. The considered error metrics in this study were **Error Rate (ER), Normalized Mean Error Distance (NMED)**, and **Mean Relative Error Distance (MRED)** [19]. In Table 1, the accuracies were reported for the block sizes of 2, 4, and 8 and the operating widths of 8, 16, and 32 bits for all studies structures. However, in the case of the ND-ACA$_{PP}$, since, as mentioned before, the smallest block size is 8, we reported its accuracy in the case of the 16-bit block size as well. The accuracy metrics were

extracted by applying 65,536 (10M) uniform random numbers in the case of 8-bit (16- and 32-bit) adders to perform unsigned add operations.

As the results show, due to similar carry signal paths, the accuracy of the proposed carry propagates ND-ACA with the block size of 4 (8) is the same as that of the proposed parallel prefix ND-ACA with block size of 8 (16). This originates from the fact that, as an example, the length of carry chain in the cases of ND-ACA$_{CP}$ with the block size of 4 and ND-ACA$_{PP}$ with the block size of 8 are the same and it is equal to 4. In the case of the 8-bit (16-bit) ND-ACA$_{CP}$ with the block size of 4 (8), the proposed adder in the approximate mode operates as an exact adder having precise outputs. The reported figures in this table reveal that, in the case of 8-bit and 16-bit adders, the ER of the ND-ACA is lower than the other adders. More specifically, the ER of the ND-ACA is up to 51% lower than those of the AC-CLA, RAP-CLA, GeAr, CMHA, and GDA adders. Also, The NMED and MRED of the ND-ACA for 32-bit and 16-bit adders are up to 75% and 76% smaller than those of the other adders for same block size.

4.2 Design Parameters

In this subsection, the design parameters (*i.e.*, delay, energy, and area) of the proposed carry propagate adder are compared to those of the AC-CLA, RAP-CLA, GeAr, CMHA, and GDA in the approximate and exact modes as well as the exact **carry look- ahead adder (CLA)** in the case of the carry propagate adders. In addition, the design parameters of the proposed parallel prefix adder are compared to those of the exact Kogge-Stone, Brent Kung and Sklansky adders [23]. All of these adders were described by Verilog HDL and synthesized by Synopsys Design Compiler using maximum executable frequency constraint. Also, all the studies in this work were performed using the typical process of the 15nm FinFET NanGate technology [25] at the operating voltage level of 0.8V and at the operating temperature of 25°C (with FO4 delay of 2.6ps). In this study, the power consumptions of the adders were obtained by extracting the internal signal activity of the adders when injecting 10M random inputs. The energies, delays and area of the considered carry propagate adders in the approximate and exact modes are shown in Figure 6(a) and (b), respectively. Note that since the area usage in approximate and exact modes are the same, the area usage is only reported in Figure 6(b). Also, the energy, delay and area of the considered parallel prefix adders are illustrated in Figure 6(c). For the purpose of comparison, in the carry propagate adders, the block size of ND-ACA$_{CP}$ was 4. For this block size, we chose the window size/block size of the considered prior works in a way that their MREDs were about the same as that of ND-ACA$_{CP}$ leading to window/block sizes of 7, 7, 4, and 4 for the RAP-CLA, GeAr, CMHA, and GDA, respectively. Also, the width (operating voltage) of the approximate part of the AC-CLA was 4 (0.6V). Moreover, the block size of ND-ACA$_{PP}$ was considered as 8 in these studies to have the same accuracy with the ND-ACA$_{CP}$ with the block size of 4.

In the case of the considered accuracy configurable CPAs in the approximate mode, the proposed ND-ACA$_{CP}$ provided lower delay, area, and energy consumption. The advantage of the proposed structure for these parameters improved by increasing the bit width. As the results show (Figure 6(a) and (b)), on average, the proposed 8-bit (32-bit) carry propagate structure in the approximate mode led to 13% (22%), 8% (15%), and 8% (14%) lower energy, delay, and area usage, respectively, compared to those of the other considered designs. Also, in the exact mode, the energy, delay, and area overheads of the proposed structure are lower than those of the other accuracy configurable designs.

The energy and area overheads of ND-ACA$_{CP}$, on average, are about 3% and 4% compared to the exact CLA, respectively while the delays are almost the same. These values for the other designs are, on average, about 27%, 12%, and 13%, respectively. Finally, the results in Figure 6(c) show that the proposed parallel prefix adder in the exact mode has similar design parameters to those of the Sklansky adder. Although, the ND-ACA$_{PP}$ structure leads to up to 9% more delay to generate the

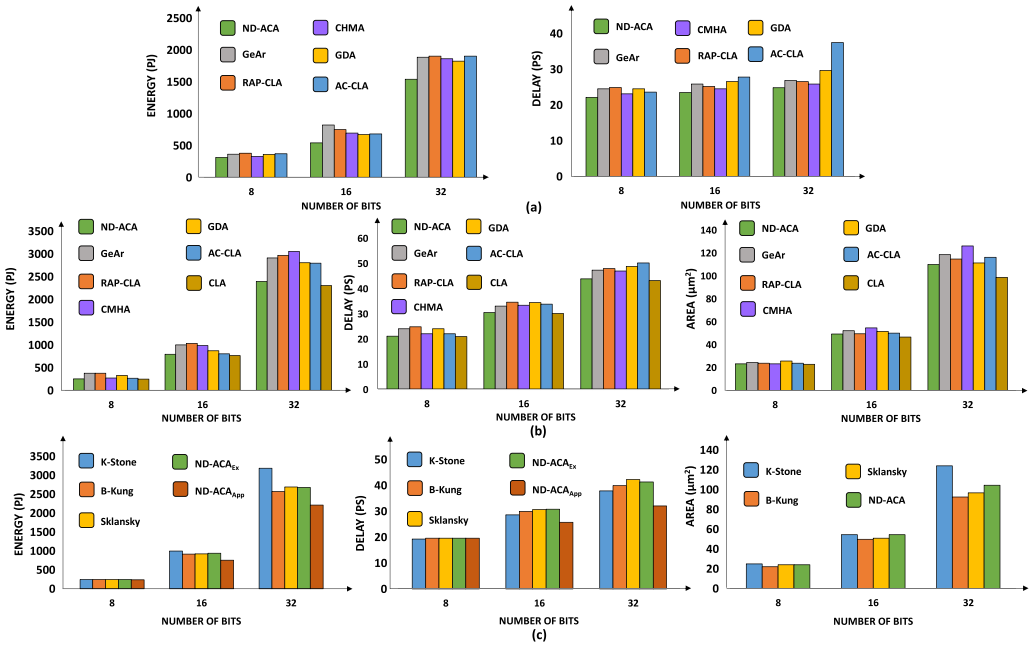


Fig. 6. The energy, delay and area consumption of 8, 16, and 32-bit (a) approximate CPAs (b) exact CPAs (c) PPAs.

final result in comparison with the Kogge-Stone adder, it consumes up to 14% lower energy in the exact model. In the approximate mode, however, its energy and delay are, on average, about 20% and 13% lower than the considered exact parallel prefix adders. Finally, as mentioned before and verified by the results, the design parameters of the 8-bit parallel ND-ACAPP in both approximate and exact mode are as the same.

4.3 Efficiency Evaluation Using Error Resilient Applications

To evaluate the efficacies of the proposed adders in error resilient applications, digit recognition and image sharpening applications were employed. In this evaluation, each sub-adder used four previous bits to predict the input carry of its next sub-adder. We have used the proposed CPA structure with the block size of 4 for the studied real-world applications. However, as illustrated in Table 1, the ND-ACA$_{CP}$ with the block size of i and ND-ACA$_{PP}$ with the block size of $2i$ have a similar output accuracy in the approximate mode. Thus, by considering the proposed tree adder structure in the real-world applications study, similar accuracies could be obtained. For the digit recognition application, we exploited a **multilayer perceptron (MLP)** neural network with one hidden layer trained for the MNIST dataset. A subset of 50K of the dataset images (with the size of 28×28 pixels) were used for the training where the remaining images (10K) were considered for the test phase. The considered MLP was implemented in the fixed-point format, where our proposed ND-ACA adder as well as other adders were utilized as the accumulator inside the MAC operator of the neurons. An exact multiplier structure was utilized for the required multiplication operation. In this study, the MLP was implemented by the adders with the width of $2i$ bits ($0 < i \leq 8$) where 8 bits considered for the integer part. Thus, for the cases with $2i \leq 8$, the fraction part size was zero.

The accuracies of these implementations are reported Table 2 where the accuracy of the network in the case of the floating-point operations (golden model) is 98.8%. In the cases of the 12

Table 2. The Accuracy of SOTA Adders in a Character Recognition Application

	Accuracy (%)							
Adder Type	2-bit	4-bit	6-bit	8-bit	10-bit	12-bit	14-bit	16-bit
GeAr	32.37	64.53	78.87	93.20	94.46	95.27	95.69	95.70
RAP-CLA	33.56	65.47	79.34	93.21	94.44	95.31	95.70	95.73
GDA	30.41	62.03	76.96	91.89	93.81	95.13	95.51	95.56
AC-CLA	32.76	63.89	78.01	93.37	93.79	94.82	94.91	95.17
CMHA	38.73	69.04	80.35	94.17	94.89	95.62	95.91	96.03
ND-ACA	41.08	72.81	83.44	95.05	95.52	96.08	96.37	96.41

Table 3. PSNR and MSSIM of the Sharpening Application on Ten Input Images

Image Name	Barbara		Mandrill		Lena		Parrot		Kid		House		Lake		Airplane		Peppers		Female	
	PSNR	SSIM	PSNR	SSIM	PSNR	SSIM	PSNR	SSIM	PSNR	SSIM	PSNR	SSIM	PSNR	SSIM	PSNR	SSIM	PSNR	SSIM	PSNR	SSIM
ND_ACA	26.96	0.85	26.82	0.94	29.6	0.89	26.92	0.73	28.43	0.85	26.93	0.82	27.16	0.87	25.14	0.67	27.26	0.83	28.36	0.79
GDA	19.11	0.76	19.23	0.86	21.66	0.7	19.65	0.52	20.47	0.78	20.71	0.65	19.82	0.64	17.84	0.48	19.63	0.59	21.27	0.66
GeAr	16.9	0.6	16.46	0.73	19.94	0.71	16.95	0.39	21.59	0.71	15.95	0.53	17.89	0.68	16.19	0.55	17.66	0.57	18.98	0.55
Rap_CLA	18.49	0.6	18.24	0.73	19.81	0.8	13.93	0.41	18.27	0.69	13.63	0.52	15.18	0.68	14.04	0.71	14.59	0.62	15.91	0.54
CMHA	23.11	0.83	21.1	0.83	27.91	0.94	27.97	0.89	28.9	0.97	28.35	0.89	26.97	0.93	27.61	0.79	27.98	0.97	28.83	0.92
AC_CLA	16.92	0.68	16.86	0.77	19.58	0.76	15.49	0.58	19.75	0.71	15.68	0.66	15.52	0.76	14.2	0.77	14.87	0.71	16.55	0.54

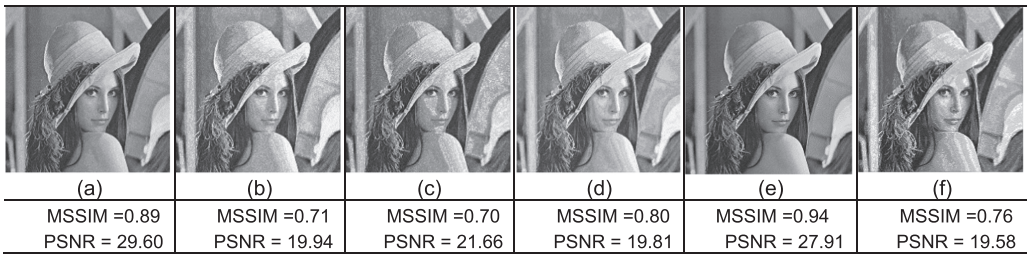


Fig. 7. The output image and its quality of the sharpening application by employing (a) ND-ACA, (b) GDA, (c) GeAr, (d) RAP-CLA, (e) CMHA, and (f) AC-CLA.

to 16-bit structures, the accuracies of the all the considered adders were almost the same (~96%). For the smaller bit widths (from 2 to 8 bits), however, our structure resulted, on average, about 7.5% accuracy improvement compared to those of RAP-CLA, GeAr, CMHA, and GDA. To evaluate the efficacies of the considered adders in the sharpening application, we used ten input images and extracted the **Peak Signal-to-Noise Ratio (PSNR)** and **Mean Structural Similarity Index Method (MSSIM)** metrics of the output images [26]. PSNR and MSSIM of the sharpening application on ten considered input images are reported in Table 3. Also, as an example, Figure 7 shows the output image when the Lena image was the input. The results show that using the ND-ACA, in image sharpening application leads to, on average, 15.7% energy saving while having only 11% (4%) reduction in PSNR (MSSIM), compared to the exact one. When GDA, GeAr, CMHA, and RAP-CLA were employed, however, on average, about 26% (22%) PSNR (MSSIM) reduction was observed.

5 CONCLUSION

In this paper, two accuracy configurable adders of block-based CPA and PPA were proposed. The adders, which in general denoted as (ND-ACA), operated in approximate and exact modes with

imposing negligible delay overhead in the latter mode. Thus, the main objective of proposing these approximate adder structures has been to present an accuracy configurable adder structure whose delay in the exact mode is almost the same as an exact adder. The characteristic parameters of ND-ACA were compared with some prior works. The comparison included two error resilient applications. The evaluation showed that the proposed carry propagate adder led to 22% (51%) lower energy consumption (error rate) compared to that of the prior works. Also, the proposed parallel prefix adder provided, on average, 20% lower energy consumption compared to that of the exact parallel prefix adders.

REFERENCES

[1] Jiang et al. 2020. Approximate arithmetic circuits: A survey, characterization, and recent applications. *Proceedings of the IEEE*.

[2] M. Kamal, A. Ghasemazar, A. Afzali-Kusha, and M. Pedram. 2014. Improving efficiency of extensible processors by using approximate custom instructions. In *Proc. DATE*.

[3] G. Zervakis, S. Xydis et al. 2018. Multi-level approximate accelerator synthesis under voltage island constraints. *IEEE Trans. Circuits Syst. II, Exp. Briefs* (2018).

[4] S. Amanollahi et al. 2020. Circuit-level techniques for logic and memory blocks in approximate computing systems. *Proceedings of the IEEE*, 2020.

[5] 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 Trans. Circuits Syst. I Reg. Papers* 57, 4 (Apr. 2010), 850–862.

[6] G. Dimitrakopoulos, K. Papachatzopoulos, and V. Paliouras. 2021. Sum propagate adders. In *IEEE Transactions on Emerging Topics in Computing* 9, 3 (1 July-Sept. 2021), 1479–1488.

[7] A. Kanani, J. Mehta, and N. Goel. 2020. ACA-CSU: A carry selection based accuracy configurable approximate adder design. In *2020 IEEE Computer Society Annual Symposium on VLSI (ISVLSI)*, 434–439.

[8] S. Angizi, H. Jiang, R. F. DeMara, J. Han, and D. Fan. 2018. Majority-based Spin-CMOS primitives for approximate computing. In *IEEE Transactions on Nanotechnology* 17, 4 (July 2018), 795–806.

[9] C. Efstathiou, Z. Owda, and Y. Tsiatouhas. 2013. New high-speed multioutput carry look-ahead adders. *IEEE Trans. Circuits Syst. II Exp. Briefs* 60, 10 (Oct. 2013), 667–671.

[10] B. K. Mohanty and S. K. Patel. 2014. Area–delay–power efficient carry selects adder. *IEEE Trans. Circuits Syst. II Exp. Briefs* 61, 6 (Jun. 2014), 418–422.

[11] V. Camus, J. Schlachter, and C. Enz. 2015. Energy-efficient inexact speculative adder with high performance and accuracy control. In *Circuits and Systems (ISCAS), 2015 IEEE International Symposium on* (May 2015), 45–48.

[12] D. Mohapatra, V. K. Chippa, A. Raghunathan, and K. Roy. 2011. Design of voltage-scalable meta-functions for approximate computing. In *Proc. IEEE DATE*, 950–955.

[13] J. Lee, H. Seo, H. Seok, and Y. Kim. 2021. A novel approximate adder design using error reduced carry prediction and constant truncation. *In IEEE Access* 9 (2021), 119939–119953.

[14] F. Ebrahimi-Azandaryani, O. Akbari, M. Kamal, A. Afzali-Kusha, and M. Pedram. 2019. Block-based carry speculative approximate adder for energy-efficient applications. *IEEE Transactions on Circuits and Systems II: Express Briefs*, 1–1.

[15] P. Balasubramanian, R. Nayar, D. L. Maskell, and N. E. Mastorakis. 2021. An approximate adder with a near-normal error distribution: Design, error analysis and practical application. In *IEEE Access* 9 (2021), 4518–4530.

[16] F. Frustaci, S. Perri, P. Corsonello, and M. Alioto. 2019. Energy-Quality scalable adders based on nonzeroing bit truncation. In *IEEE Transactions on Very Large Scale Integration (VLSI) Systems* 27, 4 (April 2019), 964–968.

[17] T. Sato et al. 2019. Trading accuracy for power with a configurable approximate adder. *IEICE Trans. Electronics* E102-C, 4 (2019).

[18] H. Afzali-Kusha, M. Kamal, and M. Pedram. 2020. Low-power accuracy-configurable carry look-ahead adder based on voltage over scaling technique. *2020 21st International Symposium on Quality Electronic Design (ISQED)*, Santa Clara, CA, USA, 67–72.

[19] O. Akbari, M. Kamal, A. Afzali-Kusha, and M. Pedram. 2018. RAP-CLA: A reconfigurable approximate carry look-ahead adder. *IEEE Transactions on Circuits and Systems II: Express Briefs* 65, 8 (2018), 1089–1093.

[20] M. Shafique, W. Ahmad, R. Hafiz, and J. Henkel. 2015. A low latency generic accuracy configurable adder. In *Proc. IEEE DAC*, 1–6.

[21] T. Yang, T. Ukezono, and T. Sato. 2018. A low-power yet high-speed configurable adder for approximate computing. *2018 IEEE International Symposium on Circuits and Systems (ISCAS)*, 1–5.

[22] R. Ye, T. Wang, F. Yuan, R. Kumar, Q. Xu. 2013. On reconfiguration oriented approximate adder design and its application. *International Conference on Computer-Aided Design (ICCAD)*, 48–54.

[23] D. Harris. 2003. A taxonomy of parallel prefix networks. In *Proc. 37th Asilomar Conf. Computing, Circuits and Systems*, (Nov. 2003), 2213–2217.
[24] G. Chen, X. Song, G. Yang, T. Wang, X. Mu, and Y. Fan. 2021. A formal proof of PG recurrence equations of parallel adders. In *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems* 40, 7 (July 2021), 1489–1494.
[25] wwww.nangate.com, NanGate - The Standard Cell Library Optimization Company, 2016 [On-line]. Available: http://www.nangate.com/.
[26] H. R. Myler and A. R. Weeks. 2009. *The Pocket Handbook of Image Processing Algorithms in C*. Englewood Cliffs, NJ, USA: Prentice-Hall, (2009).

Received 18 November 2021; revised 1 June 2022; accepted 8 July 2022