IEEE Access
Multidisciplinary | Rapid Review | Open Access Journal

Received July 31, 2021, accepted August 20, 2021, date of publication August 24, 2021, date of current version August 30, 2021.

Digital Object Identifier 10.1109/ACCESS.2021.3107370

Low Error Efficient Approximate Adders for FPGAs

WAQAR AHMAD, BERKE AYRANCIOGLU, AND ILKER HAMZAOGLU, (Senior Member, IEEE)
Faculty of Engineering and Natural Sciences, Sabanci University, 34956 Tuzla, Istanbul, Turkey
Corresponding author: Ilker Hamzaoglu (hamzaoglu@sabanciuniv.edu)

This work was supported in part by the Scientific and Technological Research Council of Turkey (TUBITAK) under Contract 118E134.

ABSTRACT In this paper, we propose a methodology for designing low error efficient approximate adders for FPGAs. The proposed methodology utilizes FPGA resources efficiently to reduce the error of approximate adders. We propose two approximate adders for FPGAs using our methodology: low error and area efficient approximate adder (LEADx), and area and power efficient approximate adder (APEx). Both approximate adders are composed of an accurate and an approximate part. The approximate parts of these adders are designed in a systematic way to minimize the mean square error (MSE). LEADx has lower MSE than the approximate adders in the literature. The 32-bit LEADx with 16-bit approximation has 20% lower MSE than the approximate adder with the lowest MSE in the literature. The 16-bit APEx with 8-bit approximation has the same area, 60% lower MSE, and 4.5% less power consumption in Xilinx Virtex 7 FPGA than the smallest and lowest power consuming approximate adder in the literature. APEx has smaller area and lower power consumption than the other approximate adders in the literature. As a case study, the approximate adders are used in video encoding application. LEADx provided better quality than the other approximate adders for video encoding application. Therefore, our proposed approximate adders can be used for efficient FPGA implementations of error tolerant applications.

INDEX TERMS Approximate computing, approximate adder, FPGA, low error, low power, LUT.

I. INTRODUCTION

Approximate computing trades off accuracy to improve the area, power, and speed of digital hardware. Many computationally intensive applications such as video encoding, video processing, and artificial intelligence are error resilient by nature due to the limitations of human visual perception or nonexistence of a golden answer for the given problem. Therefore, approximate computing can be used to improve the area, power, and speed of digital hardware implementations of these error tolerant applications.

A variety of approximate circuits, ranging from system level designs [1]–[4] to basic arithmetic circuits [5], have been proposed in the literature. Adders are used in most digital hardware, not only for binary addition but also for other binary arithmetic operations such as subtraction, multiplication, and division [6]–[8]. Therefore, many approximate adders have been proposed in the literature [9]–[24]. All approximate adders exploit the fact that critical path in an adder is seldom used.

Approximate adders can be broadly classified into the following categories: segmented adders [10], which divide n-bit adder into several r-bit adders operating in parallel; speculative adders [9], which predict the carry using only the few previous bits; and approximate full-adder based adders [12]–[17], which approximate the accurate full-adder at transistor or gate level. Segmented and speculative adders usually have higher speeds and larger areas than accurate adders [5], [13]. Approximate full-adder based approximate n-bit adders use m-bit approximate adder in the least significant part (LSP) and $(n - m)$-bit accurate adder in the most significant part (MSP), as shown in Fig. 1.

Most of the approximate adders in the literature have been designed for ASIC implementations. These approximate adders use gate or transistor level optimizations. Recent studies have shown that the approximate adders designed for ASIC implementations either do not yield the same area, power, and speed improvements when implemented

The associate editor coordinating the review of this manuscript and approving it for publication was Stavros Souravlas.

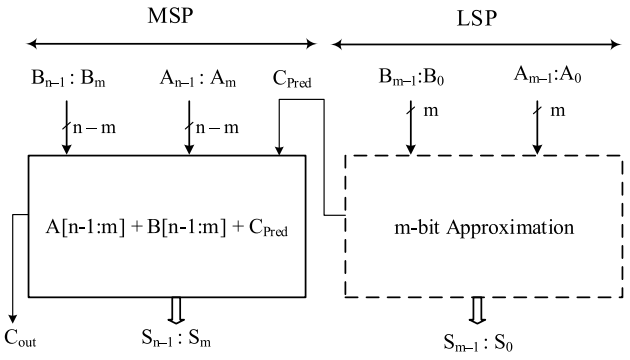

FIGURE 1. Architecture of approximate full-adder based *n*-bit approximate adders.

on FPGAs or fail to utilize FPGA resources efficiently to improve the output quality [20], [26].

This is mainly due to the difference in the way logic functions are implemented in ASICs and FPGAs. The basic element of an ASIC implementation is a logic gate, whereas FPGAs use lookup tables (LUTs) to implement logic functions. Therefore, ASIC based optimization techniques cannot be directly mapped to FPGAs.

FPGAs are widely used to implement error-tolerant applications using addition and multiplication operations. The efficiency of FPGA-based implementations of these applications can be improved through approximate computing. Only a few FPGA specific approximate adders have been proposed in the literature [19]–[23]. These approximate adders focus on improving either the efficiency or accuracy. Therefore, the design of low error efficient approximate adders for FPGAs is an important research topic.

In this paper, we propose a methodology to reduce the error of approximate adders by efficiently utilizing FPGA resources, such as unused LUT inputs. We propose two approximate adders for FPGAs using our methodology based on the architecture shown in Fig. 1.

We propose a low error and area efficient approximate adder (LEADx) for FPGAs. It has lower mean square error (MSE) than the approximate adders in the literature. It achieves better quality than the other approximate adders for video encoding application.

We also propose an area and power efficient approximate adder (APEx) for FPGAs. Although its MSE is higher than that of LEADx, it is lower than that of the approximate adders in the literature. It has the same area, lower MSE and less power consumption than the smallest and lowest power consuming approximate adder in the literature. It has smaller area and lower power consumption than the other approximate adders in the literature.

We provide mathematical models to estimate the error rate (ER), MSE, and mean absolute error (MAE) of the proposed approximate adders. We compare the proposed approximate adders with the approximate adders in the literature.

The rest of the paper is organized as follows. Section II provides an overview of related works and the necessary background to understand the proposed approximate adders. Section III presents the proposed approximate adders and the mathematical models to compute their error metrics. Their error analyses and implementation results are given in Section IV. Section IV also presents the results of using approximate adders in video encoding as a case study. Finally, Section V concludes the paper.

II. BACKGROUND

A. RELATED WORKS

Bit truncation in least significant bit positions is a well-known approximation technique. In truncate adder, the output of LSP is fixed to zero. Although, the truncate adder provides significant improvements in speed, area, and power consumption, it has high error rate and MSE [12], [13].

Lower-part-OR adder (LOA) is proposed in [14]. Its LSP consists of 2-input OR gates, whereas the MSP is accurate. A carry is sent to the MSP if it is generated at most significant bit position of the LSP. An approximate adder, OLOCA, is proposed in [15] by optimizing the LOA architecture. OLOCA uses only two OR gates in the LSP to compute the two most significant sum bits. Rest of the LSP is approximated to a fixed value. An approximate adder with near-normal error distribution (HOAANED) is proposed in [16]. HOAANED has similar architecture to OLOCA, however, it uses more resources to compute the two most significant sum bits of LSP. Therefore, HOAANED has better quality than OLOCA at the expense of slight increase in area.

Dutt *et al.* [17] proposed an approximate full adder based multibit adder (AFA). The sum of each bit of LSP is computed accurately whereas its respective carry out is equated to one of the inputs.

In recent years, a few approximate adders are proposed specifically for FPGAs. A LUT-based approximate adder (LBA) is proposed in [19]. The LSP and MSP, both perform accurate addition. A carry is passed to MSP only if it is generated at the most significant bit (MSB) of the LSP. If any other carry, that needs to be propagated to the MSP, is detected, then all bits of LSP are set to 1. LBA has high accuracy, but it does not provide performance improvement compared to the accurate adder synthesized by FPGA synthesis tool [20].

A methodology to design approximate adders (DeMAS) for FPGAs is presented in [20]. The methodology is based on an optimized truth table of approximate full-adder. Eight different variants of multibit approximate adder are presented using the optimized truth table. All these variants use same number of LUTs but differ in their error metrics.

Quaternary addition based approximate adder using the fast carry chains of FPGAs is presented in [21]. The accurate quaternary adder uses two carry inputs and generates two carry outputs. However, the authors in [21] proposed to use only one carry in the quaternary addition, hence generating an approximate result.

A single exact dual adder (SEDA) is proposed for FPGAs in [22]. The adder can either perform accurate addition of

TABLE 1. Probability of the length of a carry being equal to L bits.

Adder Bit Width (n)	Probability (%)					
	L = 1	L = 2	L = 3	L = 4	L = 5	L = 6
16	53.09	25.01	11.72	5.49	2.54	1.17
32	51.59	24.97	12.10	5.86	2.84	1.37
64	50.78	25.00	12.31	6.05	2.98	1.46
128	50.38	24.99	12.41	6.16	3.05	1.51

single n-bit input or approximate addition of two n-bit inputs. Carry of 2-bit addition is computed accurately, while the sum bits are equated to inverse of carry out.

High speed segmented approximate adders (xUAV) for FPGAs are proposed in [23]. Segmentation is done in 2, 3, or 5-bit groups for efficient mapping to LUTs. However, the proposed adders use more area and consume more power than accurate adder. These adders also have very large MAE and MSE as the size of adder is increased.

B. LENGTH OF CARRY

The key principle of approximate addition is to shorten the critical path of an adder by breaking the carry chain at one or multiple positions. This technique improves the speed of an adder at the expense of accuracy loss. In this section, we briefly explain the rationale for this technique.

The length of a carry signal in n-bit binary addition is defined as the number of bits it propagates before being killed or regenerated. For example, if a carry signal is generated at ith bit position and killed or regenerated at jth bit position ($j > i$), the length of that carry signal is defined as $j - i$ bits.

In n-bit binary addition, the outgoing carry signal at any bit position i is determined by the current and previous input bits. Bit position i is said to generate a carry if both the input bits at ith position are 1, propagate the incoming carry if both the input bits at ith position are different, and kill the incoming carry if both the input bits at ith position are 0.

In the worst case, a carry signal is generated in the least significant bit (LSB) and propagated to the most significant bit (MSB). In this case, the length of carry signal is equal to the adder bit width. However, the worst case rarely happens, and the average length of a carry signal is usually much shorter than the adder bit width [9].

We implemented and simulated n-bit accurate adder using 10^7 independent random number pairs extracted from uniformly distributed sample space between 0 and $2^n - 1$. Based on these simulation results, probability of the length of a carry signal being equal to L bits is given in Table 1. As can be seen from this table, the length of a carry signal is rarely longer than 5 bits. The length of a carry signal is shorter than 5 bits with more than 90% probability.

Since the worst case of carry propagation (length of carry = n-bits) rarely happens, in most cases, the carry can be correctly predicted by considering only a few previous input bits.

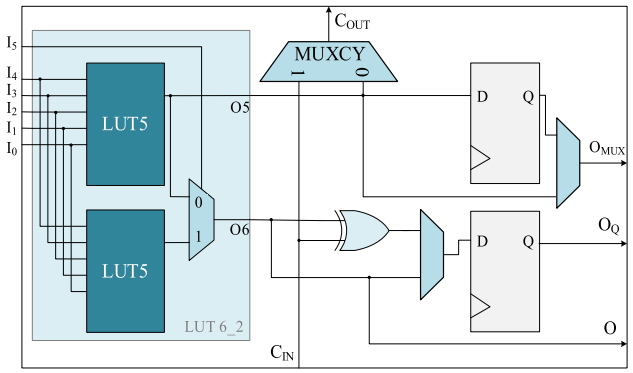

FIGURE 2. Simplified architecture of a slice in Xilinx Virtex 7 FPGA.

C. XILINX VIRTEX FPGA

The main logic resource in a Xilinx Virtex FPGA is configurable logic blocks (CLBs) [27]. Each CLB contains two slices. Simplified architecture of a slice in Xilinx Virtex 7 FPGA is shown in Fig. 2. Each slice is composed of 4 such elements with carry-chain cascaded in series. Therefore, each slice has four 6-input LUTs. Each LUT can be used to implement two 5-input combinational logic functions or one 6-input combinational logic function. Furthermore, each slice also contains a 4-bit carry-chain and eight flip-flops.

An efficient FPGA-based implementation should be able to effectively utilize these resources. This is particularly important for implementing arithmetic functions that can utilize the fast carry-chains. Therefore, it is important to understand how the arithmetic operations are implemented on FPGAs. Particularly, we consider the mapping of a full adder to a Xilinx FPGA.

Typically, a full adder is implemented as shown in (1) and (2), where A and B represent the inputs, S is sum, and C_{IN} and C_{OUT} are carry-in and carry-out, respectively.

$$S = (A \oplus B) \oplus C_{IN} \tag{1}$$

$$C_{OUT} = (A \oplus B)\, C_{IN} + AB \tag{2}$$

However, when implementing a full adder on a Xilinx FPGA, the synthesis tool rewrites (2) as (3).

$$C_{OUT} = (A \oplus B)\, C_{IN} + \overline{(A \oplus B)}B \tag{3}$$

This simplification allows the reuse of A ⊕ B logic function for computing both S and C_{OUT} [28]. This term is used as input to XOR gate for sum computation and as select input of mux for selecting the appropriate signal for C_{OUT}. However, since only 4-bit carry chain is available in a slice, an n-bit adder uses n LUTs such that 4 inputs and one output of each LUT are not used. These unused resources can be utilized to implement additional logic with an adder without increasing area.

III. PROPOSED DESIGN METHODOLOGY

The proposed design methodology uses the approximate full-adder based n-bit adder architecture shown in Fig. 1. n-bit addition is divided into n-bit approximate adder in the LSP

TABLE 2. Effects of increasing the number of bits (k) for carry prediction in a 64-bit approximate adder with 12-bits LSP.

k	LUT	Delay (ns)	ER (%)
0	64	1.57	50.04
1	64	1.59	24.98
2	64	1.62	12.51
3	65	1.85	6.26
4	65	1.90	3.10
5	65	2.03	1.55

and $(n-m)$-bit accurate adder in the MSP. Breaking the carry chain at bit-position m generally introduces an error of 2^m in the final sum. The error rate and error magnitude can be reduced by predicting the carry-in to the MSP (C_{MSP}) more accurately and by modifying the logic function of LSP to compensate for the error.

The carry to the accurate part can be predicted using any k-bit input pairs from the approximate part such that $k \leq m$. Most of the existing approximate adders use $k = 1$.

As discussed in Section II, FPGA implementation of accurate adder uses only 2 inputs and 1 output of each 6-input LUT. We propose to utilize the remaining 4, available but unused, inputs of the first LUT of the MSP to predict C_{MSP}. Therefore, we propose to share the most significant 2 bits of both inputs of the LSP with the MSP for carry prediction.

Sharing more bits of LSP with MSP will increase the probability of correctly predicting C_{MSP} which will in turn reduce error rate. However, this will also increase the area and delay of the approximate adder.

To analyze the tradeoff between the accuracy and performance of an FPGA-based approximate adder with different values of k, we performed synthesis and simulation experiments on a Xilinx Virtex 7 FPGA.

The results for a 64-bit adder with 12-bits LSP using k bits to predict C_{MSP} are shown in Table 2. For $k > 2$, the error rate reduces slightly at the cost of increased area and delay. On the other hand, for $k < 2$, the delay improves marginally at the cost of significant increase in the error rate.

Therefore, we propose using $k = 2$, as it provides good balance between accuracy and performance of approximate adders for FPGAs. In the proposed approximate adders, a carry is passed to the MSP if it is generated at bit position $m-1$, or generated at bit position $m-2$ and propagated at bit position $m-1$. The C_{MSP} can be described by (4) where G_i and P_i are the generate and propagate signals of the ith bit position, respectively.

$$C_{MSP} = G_{m-1} + P_{m-1}G_{m-2} \tag{4}$$

The error in higher bit positions has more impact on the error magnitude of an approximate adder. As described in (4), the carry-in to MSP is predicted using two most significant bits of LSP. These 2 bits effectively implement a 3-output function $\{C_{MSP}S_{m-1}S_{m-2}\}$. An error occurs in the n-bit addition if a carry (C_{m-2}) is generated at bit position

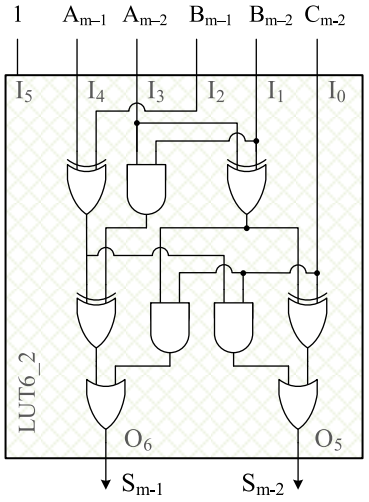

FIGURE 3. Proposed 2-bit approximate adder (AAd1) used in MSBs of LSP.

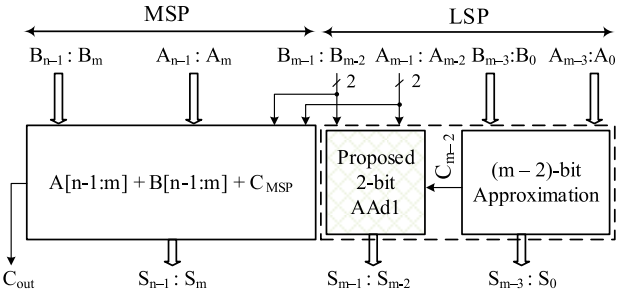

FIGURE 4. Architecture of proposed approximate adders for FPGAs.

$i < (m-2)$ and that carry should be propagated to MSP. In this case, the correct result should be $\{C_{MSP}S_{m-1}S_{m-2}\} = 100$. However, without any error reduction mechanism the approximate result will be $\{C_{MSP}S_{m-1}S_{m-2}\} = 000$.

To reduce the error magnitude, we propose a 2-bit approximate adder (AAd1) for computing S_{m-1} and S_{m-2}. The functionality of AAd1 is described by (5) and (6). AAd1 is implemented using a single LUT as shown in Fig. 3. When $C_{m-2} = 1$, $P_{m-2} = 1$, and $P_{m-1} = 1$, the approximate result will be $\{C_{MSP}S_{m-1}S_{m-2}\} = 011$, only 1 less than the accurate result. For all other inputs, it will generate the accurate result.

$$S_{m-2} = (P_{m-2} \oplus C_{in}) + (P_{m-1}C_{m-2}) \tag{5}$$

$$S_{m-1} = (P_{m-1} \oplus G_{m-2}) + (P_{m-2}C_{m-2}) \tag{6}$$

For uniformly distributed inputs, the carry-in has equal probability of being 1 or 0. The probability of inputs at bit position i propagating a carry is $P_i = 1/2$. Therefore, in the proposed n-bit approximate adders, the probability of S_{m-2} and S_{m-1} generating an error is 0.125 as shown in (7). Throughout this paper, E_x represents the cases when hardware x generates an error.

$$\begin{aligned} Pr\,[E_{AAd1}] &= Pr\,[C_{m-2} \wedge P_{m-2} \wedge P_{m-1}] \\ &= \frac{1}{2} \times \frac{1}{2} \times \frac{1}{2} = 0.125 \end{aligned} \tag{7}$$

TABLE 3. Truth table of proposed 2-bit approximate adder (AAd2) used for approximation in least-significant m−2 bits of LEADx.

A_{i+1}	A_i	B_{i+1}	B_i	C_{in}	C_{i+2}	S_{i+1}	S_i
0	0	0	0	0	0	0	0
0	0	0	0	1	0	0	1
0	0	0	1	0	0	0	1
0	0	0	1	1	0	1	0
0	0	1	0	0	0	1	0
0	0	1	0	1	0	1	1
0	0	1	1	0	0	1	1
0	0	1	1	1	0	1	1
0	1	0	0	0	0	0	1
0	1	0	0	1	0	1	0
0	1	0	1	0	0	1	0
0	1	0	1	1	0	1	1
0	1	1	0	0	0	1	1
0	1	1	0	1	0	1	1
0	1	1	1	0	0	1	1
0	1	1	1	1	0	1	1
1	0	0	0	0	1	0	0
1	0	0	0	1	1	0	0
1	0	0	1	0	1	0	0
1	0	0	1	1	1	0	0
1	0	1	0	0	1	0	0
1	0	1	0	1	1	0	1
1	0	1	1	0	1	0	1
1	0	1	1	1	1	1	0
1	1	0	0	0	1	0	0
1	1	0	0	1	1	0	0
1	1	0	1	0	1	0	0
1	1	0	1	1	1	0	1
1	1	1	0	0	1	0	1
1	1	1	0	1	1	1	0
1	1	1	1	0	1	1	0
1	1	1	1	1	1	1	1

Architecture of the proposed approximate adders is shown in Fig. 4. It uses 2 MSBs of LSP to predict the C_{MSP}, whereas their respective sum bits are computed using AAd1. AAd1 is only suitable when the C_{out} of 2-bit inputs is predicted accurately. Accurate prediction of C_{out} requires additional resources or unused LUT inputs. Therefore, to design area efficient approximate adders for FPGAs, AAd1 is not used in the least-significant $m-2$ bits of the LSP. In this paper, we propose two n-bit approximate adders using the architecture in Fig. 4. The two proposed n-bit approximate adders use different approximate functions for the first $m-2$ bits of the LSP.

A. PROPOSED LOW ERROR AND AREA EFFICIENT APPROXIMATE ADDER FOR FPGAs

In this section, we propose a low error and area efficient approximate adder (LEADx) for FPGAs.

State-of-the-art FPGAs use 6-input LUTs. These LUTs can be used to implement two 5-input functions. The complexity of the implemented logic function does not affect performance of LUT based implementation. A 2-bit adder has 5 inputs and two outputs. Therefore, a LUT can be used to implement a 2-bit approximate adder.

For an area efficient FPGA implementation, we propose to split the first $m-2$ bits of LSP into $\lceil (m-2)/2 \rceil$ groups of 2-bit inputs such that each group is mapped to a single LUT. Each group adds two 2-bit inputs with carry-in using an approximate 2-bit adder (AAd2).

To eliminate the carry chain in LSP, we propose to equate C_{out} of ith group to one of the inputs of that group (A_{i+1}). This results in error in 8 out of 32 possible cases with an absolute error magnitude of 4 in each erroneous case. To reduce the error magnitude, we propose to compute the S_i and S_{i+1} output bits as follows:

- If the C_{out} is predicted correctly, the sum outputs are also calculated accurately using standard 2-bit addition.
- If the C_{out} is predicted incorrectly and the predicted value of C_{out} is 0, both sum outputs are set to 1.
- If the C_{out} is predicted incorrectly and the predicted value of C_{out} is 1, both sum outputs are set to 0.

This modification reduces the absolute error magnitude to 2 in two cases, and to 1 in the other six cases. The resulting truth table of AAd2 is given in Table 3. The error cases are shown in red. Since AAd2 produces an erroneous result in 8 out of 32 cases, the error probability of AAd2 is 0.25 as shown in (8).

$$Pr\left[E_{AAd2}\right] = 0.25 \tag{8}$$

The proposed LEADx approximate adder is shown in Fig. 5. An n-bit LEADx uses $\lceil (m-2)/2 \rceil$ copies of AAd2 adder in the least significant $m-2$ bits of the approximate adder architecture shown in Fig. 4. In LEADx, $C_{m-2} = A_{m-3}$. AAd2 implements a 5-to-2 logic function that is mapped to a single LUT. Similarly, AAd1 is also mapped to a single LUT. Therefore, $\lceil m/2 \rceil$ LUTs are used for the LSP. These LUTs work in parallel. Therefore, the delay of LSP is equal to the delay of a single LUT (t_{LUT}). The critical path of LEADx is from the input A_{m-2} to the output S_{n-1}.

Fig. 6 shows an example of the functionality of 16-bit LEADx with 8-bit approximation. The outputs of bits enclosed in dotted lines are computed using AAd1. The outputs of the other bits of the approximate part (LSP) are computed using three copies of AAd2. The carry-in to the accurate part (C_{MSP}) is predicted from the two MSBs of LSP as shown in (4).

The error probability of n-bit LEADx depends on the number of approximate 2-bit adders used in the approximate part. Error in any of these 2-bit adders can contribute to the error in the sum output. Therefore, the error probability of LEADx is given as the union of error probabilities of the individual 2-bit approximate adders. Let there be N copies of AAd2 in LEADx and E_{AAd2-i} represents the error in ith copy of AAd2, then the error probability of LEADx can be calculated as shown in (9).

$$\begin{aligned} &Pr\left[E_{LEADx}\right] \\ &\quad = Pr\left[E_{AAd1} \vee E_{AAd2-1} \vee E_{AAd2-2} \ldots \vee E_{AAd2-N}\right] \end{aligned} \tag{9}$$

Since error in two or more of these 2-bit adders can occur concurrently, occurrence of error in these adders are not mutually exclusive. Therefore, (9) can be evaluated using

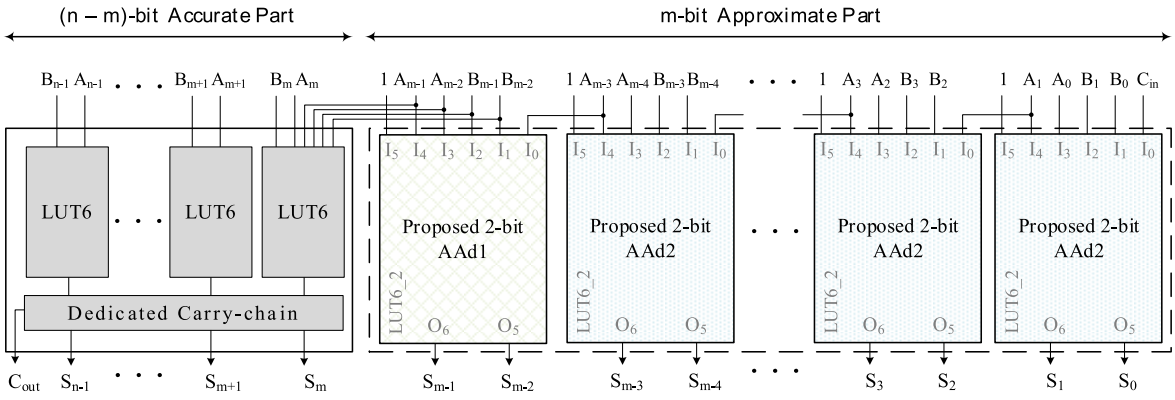

FIGURE 5. Proposed *n*-bit low error and area efficient approximate adder (LEADx).

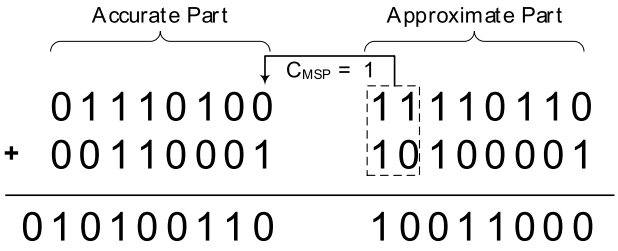

FIGURE 6. Example of 16-bit LEADx with 8-bit approximation.

inclusion-exclusion principle [29]. For example, the error probability of LEADx with 4-bits LSP, for uniformly distributed inputs, can be calculated as shown in (10).

$$\begin{aligned} &Pr\left[E_{LEADx|m=4}\right] \\ &= Pr\left[E_{AAd1} \vee E_{AAd2}\right] \\ &= Pr\left[E_{AAd1}\right] + Pr\left[E_{AAd2}\right] - Pr\left[E_{AAd1} \wedge E_{AAd2}\right] \\ &= 0.125 + 0.25 - (0.125 \times 0.25) \\ &= 0.34375 \end{aligned} \tag{10}$$

Similarly, the error probability of LEADx with 6-bits LSP, for uniformly distributed inputs, can be calculated as shown in (11).

$$\begin{aligned} &Pr\left[E_{LEADx|m=6}\right] \\ &= Pr\left[E_{AAd1} \vee E_{AAd2} \vee E_{AAd2}\right] \\ &= Pr\left[E_{AAd1}\right] + Pr\left[E_{AAd2}\right] + Pr\left[E_{AAd2}\right] \\ &\quad - Pr\left[E_{AAd1} \wedge E_{AAd2}\right] - Pr\left[E_{AAd1} \wedge E_{AAd2}\right] \\ &\quad - Pr\left[E_{AAd2} \wedge E_{AAd2}\right] + Pr\left[E_{AAd1} \wedge E_{AAd2} \wedge E_{AAd2}\right] \\ &= 0.125 + 0.25 + 0.25 - (0.125 \times 0.25) - (0.125 \times 0.25) \\ &\quad - (0.25 \times 0.25) + (0.125 \times 0.25 \times 0.25) \\ &= 0.50781 \end{aligned} \tag{11}$$

B. PROPOSED AREA AND POWER EFFICIENT APPROXIMATE ADDER FOR FPGAs

In this section, we propose an area and power efficient approximate adder (APEx) for FPGAs.

TABLE 4. Error characterization of constant approximate functions for 1-bit addition.

A	B	Accurate Addition	Constant 1		Constant 0	
			Sum	Error	Sum	Error
0	0	00	1	1	0	0
0	1	01	1	0	0	-1
1	0	01	1	0	0	-1
1	1	10	1	-1	0	-2
Error Cases			2		3	
Average Error			0		-1	
Mean Square Error			½		⅔	

APEx is also based on the approximate adder architecture shown in Fig. 4. For the least significant $m-2$ bits of the LSP, the aim is to find an approximate function with no data dependency. Carry should neither be generated nor used for sum computation. A 1-bit input pair at any bit position $i \leq (m-2)$ should produce a 1-bit sum output only.

In general, any logic function with 1-bit output can be used as an approximate function to compute the approximate sum of 1-bit inputs at *i*th bit position. A constant 0 or constant 1 at the output are also valid approximate functions. Fixing the output to 0 or 1 will reduce the area and power consumption of the approximate adder because no hardware will be required for sum computation.

We evaluated error metrics of both constant functions for 1-bit addition, as shown in Table 4. Fixing the output to 0 introduces error in 3 out of 4 cases with an average error (AE) of -1 and MSE of $3/2$ for uniformly distributed inputs. Fixing the output to 1 introduces error in 2 out of 4 cases with 0 AE and MSE of 1/2 for uniformly distributed inputs. Therefore, constant 1 provides a better approximation.

We further analyze the error metrics of *n*-bit approximate adder architecture shown in Fig. 4 when approximate constant functions are used in the least significant $m-2$ bits of its LSP. If the least significant $m-2$ bits are fixed to 0, the maximum error (ME) occurs when the inputs A_0 to A_{m-3} and B_0 to B_{m-3} are all 1. With accurate addition, S_1 to S_{m-3} output bits are all 1 and a carry is propagated to $m-2$ bit position. Fixing S_0 to S_{m-3} to 0 and carry-in for $m-2$ bit position to 0 results in ME of $2^{m-1}-2$.

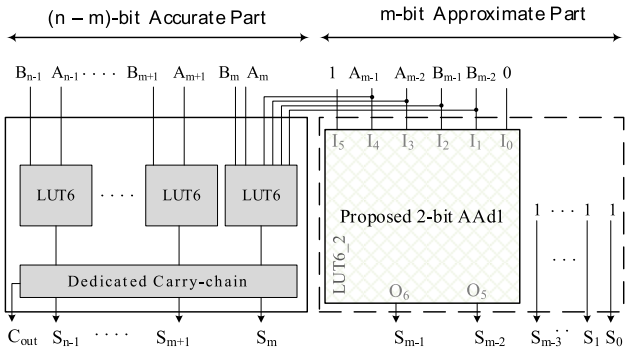

FIGURE 7. Proposed *n*-bit area and power efficient approximate adder (APEx).

If the least significant $m-2$ bits are fixed to 1, the ME occurs when the inputs A_0 to A_{m-3} and B_0 to B_{m-3} are all 0. With accurate addition, S_0 to S_{m-3} output bits are all 0 and carry is not propagated to $m-2$ bit position. Fixing S_0 to S_{m-3} to 1 and carry-in for $m-2$ bit position to 0 results in ME of $2^{m-2}-1$. The ME of constant 1 is less than the ME of constant 0.

Furthermore, assume that a constant value V is used to approximate the function $F = A + B$. The resulting absolute error is defined as $|F - V|$. The aim is to find a constant value V such that MSE is minimized. This is a well-known problem with a well-defined solution: using mean of distribution of F as V minimizes the MSE [30], [31].

Let us consider that A and B have uniform input distribution with values between 0 and $2^n - 1$, then F has a symmetric triangular distribution in the range [0, $(2^{n+1} - 2)$] [18]. In the case of symmetric distribution, the mean and median are the same and located at the center of the sample space [32]. Therefore, mean and median of F are located at $2^n - 1$, which is the halfway point of [0, $2^{n+1} - 2$]. The binary representation of $2^n - 1$, in $n+1$ bit sample space, is 0111 ... 1. Therefore, using constant 1 as the sum output and 0 as carry-out minimizes the MSE of the approximate output. If i bits are fixed to 1, the probability of error in the sum output is calculated as shown in (12).

$$Pr\left[E_{const1}\right] = \frac{2^i - 1}{2^i} \tag{12}$$

In the proposed APEx, the S_0 to S_{m-3} outputs are fixed to 1 and the C_{m-2} is 0. This provides significant area and power consumption reduction at the expense of slight quality loss. It is important to note that this is different from bit truncation technique which fixes both the sum and carry outputs to 0. The ME of truncate adder is $2^{m+1} - 2$ which is much higher than ME of APEx ($2^{m-2} - 1$).

The proposed APEx approximate adder is shown in Fig. 7. Same as LEADx, the critical path of APEx is from the input A_{m-2} to the output S_{n-1}. Similar to (9), the error probability of APEx can be calculated as shown in (13). When C_{m-2} is 0, E_{AAd1} reduces to 0 according to (7). Therefore, the error probability of APEx depends only on the number of output bits fixed to 1.

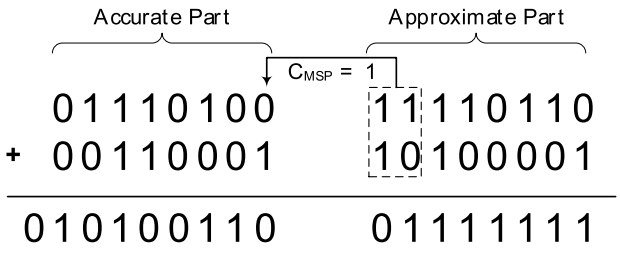

FIGURE 8. Example of 16-bit APEx with 8-bit approximation.

$$\begin{aligned} Pr\left[E_{APEx}\right] &= Pr\left[E_{AAd1|C_{m-2}=0} \vee E_{Const1|i=m-2}\right] \\ &= \frac{2^{m-2} - 1}{2^{m-2}} \end{aligned} \tag{13}$$

Fig. 8 shows an example of the functionality of 16-bit APEx with 8-bit approximation. The outputs of the bits enclosed by dotted lines are computed using AAd1. The outputs of the other bits of the approximate part (LSP) are fixed to 1. The carry-in to the accurate part (C_{MSP}) is predicted from the two MSBs of LSP as shown in (4).

IV. EXPERIMENTAL RESULTS AND DISCUSSION

In this section, we present experimental results of the proposed approximate adders, LEADx and APEx. We compare LEADx and APEx with other FPGA-specific approximate adders in the literature: LBA [18], DeMAS [19], and SEDA [21]. DeMAS can be built using different configurations. For a given number of approximate bits, each of these configurations has the same area. Therefore, we chose the configuration with the lowest average error for comparison.

We also compare LEADx and APEx with power and area efficient ASIC-based approximate adders in the literature: AFA [17], HOAANED [16], and LOA [14]. Each of these approximate adders is based on the approximate adder architecture shown in Fig. 1, where approximation is done only in the LSP and the MSP is kept accurate. We also compare the proposed approximate adders with the segmented and speculative approximate adders in the literature.

A. ERROR METRICS

The functional models of these approximate adders are implemented in C++. Error metrics of these approximate adders are determined using their functional models for 16, 32, and 64-bit addition, with varying number of approximate bits, using 10^7 uniform random numbers as inputs.

The error value for each input is calculated by subtracting the accurate result from the approximate result. Error value may be positive, negative, or zero. The average error (AE) is defined as the average of all the error values. MAE, also known as mean error distance [33], is the average of the absolute values of all the error values. MAE is always positive. MSE is the average of the squares of all the error values. RMSE is the square root of MSE.

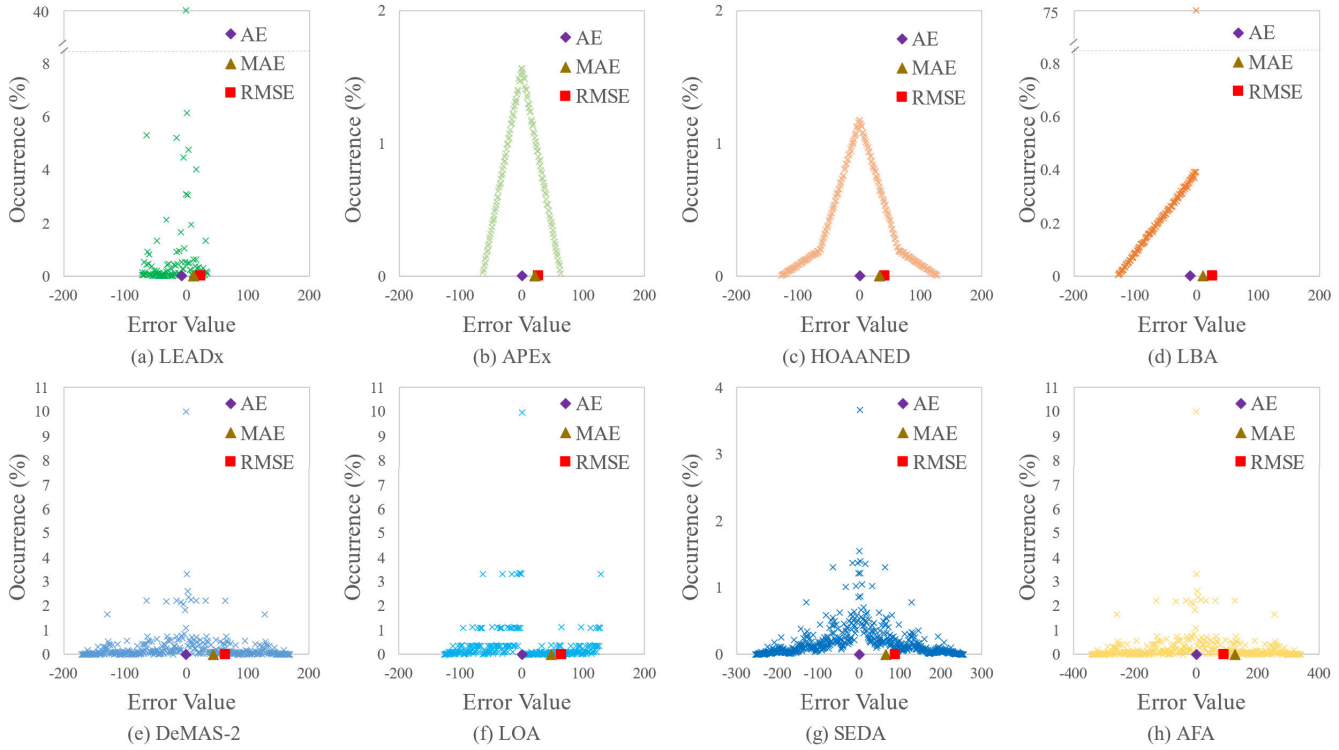

FIGURE 9. Error distribution and error metrics of 16-bit approximate adders with 8-bit approximation.

The MAE and MSE of LEADx can be calculated using (14) and (15), respectively. Similarly, the MAE and MSE of APEx can be calculated using (16) and (17), respectively. An empirical approach is used to determine these mathematical models, i.e., these formulas are determined using experimental results.

$$MAE_{LEADx} = \left(\frac{3}{16} \times 2^{m-2}\right) + 2^{m-9} \tag{14}$$

$$MSE_{LEADx} \approx 2^{2m-7} + 2^{2m-11} - 2^{1.5m-9.2} \tag{15}$$

$$MAE_{APEx} = \frac{2^{m-2}}{3} \tag{16}$$

$$MSE_{APEx} \approx \frac{5}{24} \times 4^{m-2} \tag{17}$$

As can be observed in these equations, error metrics of the proposed approximate adders depend only on the number of approximate bits (m), and they are independent of the bit width (n) of the adder.

Error metrics and the error distribution of 16-bit approximate adders with 8-bit approximation are shown in Fig. 9. The error distribution is plotted as a function of error value and its respective percentage occurrence. As can be seen in Fig. 9, the maximum errors of the proposed approximate adders are less than those of other approximate adders.

The error distribution of LEADx is skewed to the negative side. This indicates that, in most of the cases, the result of LEADx is less than the accurate result, leading to a negative AE. Whereas, plotting the error distribution of APEx results in a symmetrical triangular shape centered at zero, indicating that APEx has equal probability of negative and positive errors. Therefore, APEx has almost zero AE.

The error distribution of LBA indicates that its erroneous output is always less than the accurate result. All other approximate adders in the literature have almost symmetrical error distribution. However, their error values are spread over a wide range, resulting in much larger MAE and MSE as compared to the proposed approximate adders.

The error metrics of 64-bit adders with 4 to 12-bits of approximation are reported in Table 5. Our proposed approximate adders have the lowest MSE. The MSE of the LEADx is at least 20% less than that of the approximate adders in the literature.

LBA has the lowest MAE. However, it has the worst area and power consumption results, as reported in the next section. The MAE of the proposed approximate adders is second only to that of LBA.

The ER of LEADx and APEx validate the analytical error probability results given in Section II. All the approximate adders, except LBA and LEADx, have high ER.

These adders follow the fail-small approach [25]. In the fail-small approach, even if ER is high, error magnitudes are small. The rationale behind this approach is that small errors are naturally masked by algorithms, and they have less impact on MSE. Therefore, they slightly degrade the quality of applications.

The error magnitude of our proposed approximate adders is significantly reduced by accurately predicting the carry to the MSP using unused LUT inputs. AAd1 and AAd2, both fully utilize the LUT inputs to achieve low error. The

TABLE 5. Error metrics of 64-bit approximate adders.

	Adder	Approximate Bits				
		4	6	8	10	12
MSE ($x10^2$)	LEADx	0.019	0.333	5.43	87.10	1392
	APEx	0.025	0.425	6.83	109.09	1746
	AFA [17]	0.639	10.240	163.62	2620.20	41894
	DeMAS–2 [20]	0.160	2.560	40.91	655.05	10473
	HOAANED [16]	0.065	1.066	17.10	273.14	4363
	LBA [19]	0.026	0.428	6.84	110.07	1751
	LOA [14]	0.159	2.560	41.00	656.66	10494
	SEDA [22]	0.303	4.920	78.56	1257.80	20130
MAE	LEADx	0.69	3.08	12.56	50.41	201
	APEx	1.25	5.31	21.33	85.27	341
	AFA [17]	5.51	22.35	89.54	358.27	1432
	DeMAS–2 [20]	2.75	11.17	44.77	179.14	716
	HOAANED [16]	1.94	7.98	32.02	128.02	511
	LBA [19]	0.66	2.67	10.68	42.91	171
	LOA [14]	2.87	11.88	47.92	192.16	768
	SEDA [22]	4.05	16.47	65.99	264.10	1056
ER (%)	LEADx	34.44	50.83	63.10	72.31	79.26
	APEx	74.99	93.75	98.44	99.61	99.91
	AFA [17]	68.30	82.14	89.97	94.38	96.83
	DeMAS–2 [20]	68.31	82.14	89.97	94.39	96.84
	HOAANED [16]	81.26	95.34	98.82	99.71	99.93
	LBA [19]	21.88	24.31	24.85	25.04	25.04
	LOA [14]	68.35	82.21	90.01	94.37	96.82
	SEDA [22]	80.85	91.61	96.33	98.40	99.29

LEADx is designed in a way that not only the error values are reduced but also the number of error cases are reduced. The experimental results show that LEADx has indeed higher accuracy and lower MSE than the other approximate adders. Similarly, the logic function of the approximate part of APEx is determined to reduce the MSE. The experimental results show that the MSE of APEx is indeed less than that of the approximate adders in the literature.

B. IMPLEMENTATION RESULTS

All the approximate adders are implemented using Verilog HDL. The accurate part of all the adders is identical and implemented using addition operator. Verilog RTL codes are synthesized and implemented on a Xilinx Virtex 7 FPGA with speed grade 3 using Vivado 2020.1. AreaOptimized_high strategy is used for synthesis, and default strategy is used for implementation.

The quality metrics are extracted from post-implementation timing simulations using 1 million uniform random numbers. The quality metrics are cross verified with C++ simulations. For power estimation, switching activity interchange format (SAIF) files are also generated from these post-implementation timing simulations at 100 MHz for all adders. The power consumption of each approximate adder FPGA implementation is estimated with Vivado 2020.1 using the corresponding SAIF file.

TABLE 6. FPGA implementation results of 16-bit adders with 8-bit approximation.

Adder	LUTs		Delay (ns)	Power (mW)
	MSP	LSP		
Accurate	8	8	1.35	6.16
LEADx	8	4	1.35	6.09
APEx	8	1	1.35	4.32
AFA [17]	8	5	1.35	6.17
DeMAS–2 [20]	8	5	1.35	6.12
HOAANED [16]	8	1	1.35	4.52
LBA [19]	8	11	1.56	6.10
LOA [14]	8	4	1.35	5.66
SEDA [22]	8	6	1.35	6.16

The implementation results of 16-bit adders with 8-bit approximation are given in Table 6. All the adders are implemented with input and output registers. SEDA and LBA are slower than the accurate adder because of carry propagation in their LSPs. All other 16-bit approximate adders have the same delay as the accurate adder. It is important to note that their delay is limited by the maximum frequency of Virtex 7 FPGA. It does not necessarily mean that the critical path of these adders is the same.

All the approximate 16-bit adders, except LBA, use fewer LUTs than the accurate adder. Since an accurate adder is used in the MSP of all these adders, the reduction in LUTs occurs only in the LSP. Since LEADx performs 2-bit addition in a single LUT, its LSP uses 50% fewer LUTs than the accurate adder.

APEx and HOAANED use the lowest number of LUTs. For these two adders, a significant reduction in number of LUTs occurs because of the use of constant functions in their LSPs. For other approximate adders, the reduction in number of LUTs occurs because of the approximation techniques used, which allow the synthesis tool to merge two sum outputs to a single LUT.

LEADx consumes slightly less power than the accurate adder. APEx consumes the lowest power among all the approximate adders. For the 16-bit adder with 8-bit approximation, the power consumption of APEx is 29% less than that of the accurate adder and 4.5% less than that of the second lowest power consuming adder, HOAANED.

The LUTs vs MSE and power vs MSE graphs of 32-bit approximate adders are given in Fig. 10. These results are plotted for 4-bit to 20-bit approximation in a 32-bit adder. The 32-bit accurate adder uses 32 LUTs and consumes 10.75 mW power.

While the number of LUTs used by most of the approximate adders decreases linearly with the increase in

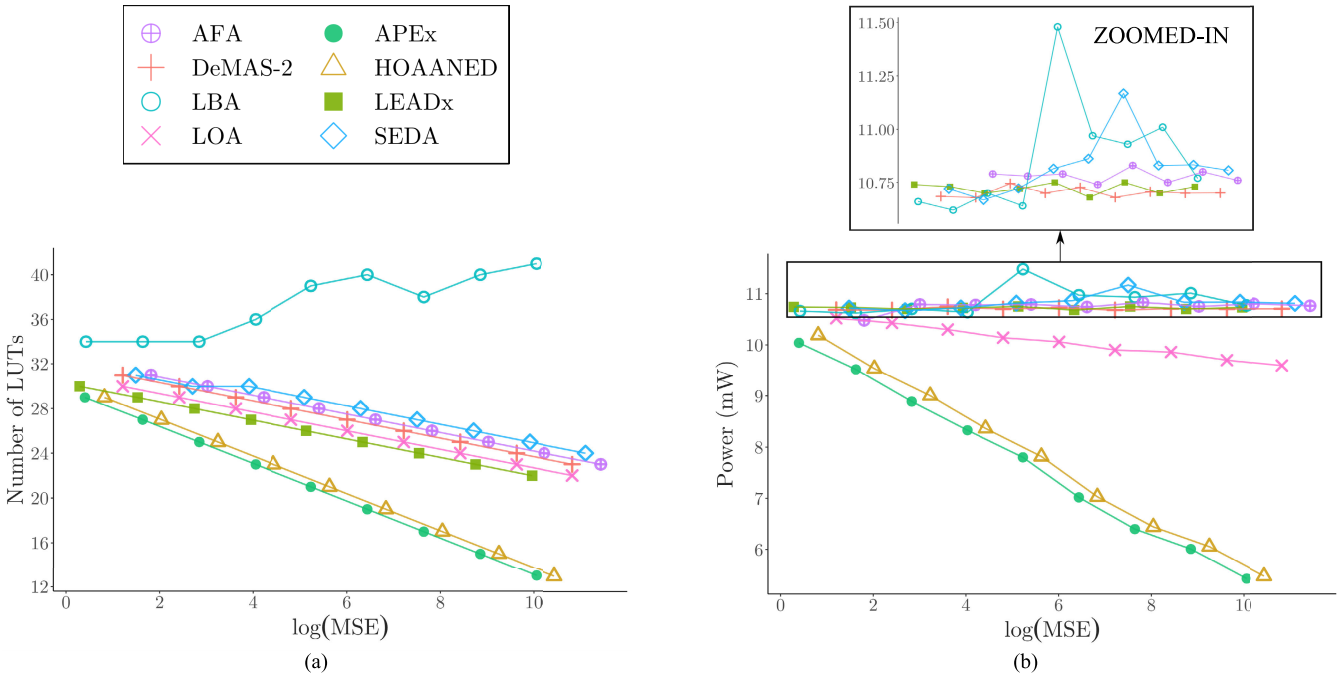

FIGURE 10. Comparison of 32-bit approximate adders with 4-bit to 20-bit approximation (left to right). (a) LUTs vs MSE. (b) Power vs MSE.

approximation, their respective power reductions do not follow the same trend. However, APEx provides significant power reduction compared to the accurate adder at the cost of a slight loss in accuracy.

LUTs, power consumption and delay reductions achieved by 64-bit approximate adders with 16-bit approximation compared to 64-bit accurate adder are shown in Fig. 11. LEADx reduced the LUTs by 12.5% compared to the accurate adder. APEx reduced the LUTs by 23.4% and power consumption by 21% compared to the accurate adder.

LBA performs worse than the accurate adder in all these metrics. Among other FPGA specific adders, DeMAS provided no power reduction but reduced the LUTs by 11% compared to the accurate adder. The performance of HOAANED is compatible with APEx. However, as discussed earlier, it has lower quality than both LEADx and APEx.

These results show that our proposed LEADx has smaller area, lower power, and better quality than the FPGA specific adders in the literature. The results show that DeMAS is the most efficient FPGA specific approximate adder in the literature. With 8-bits approximation, LEADx has 7% smaller area and 86% lower MSE than DeMAS. LOA is one of the most efficient ASIC-based approximate adders in the literature [5]. LEADx has better quality than LOA at the same cost when implemented on an FPGA. With 8-bits approximation, LEADx has 87% lower MSE than LOA at the same cost. HOAANED is suitable for FPGA implementation. However, APEx has less power and better quality than HOAANED at the same cost, when implemented on an FPGA. APEx has more than 60% lower MSE than HOAANED at the same cost.

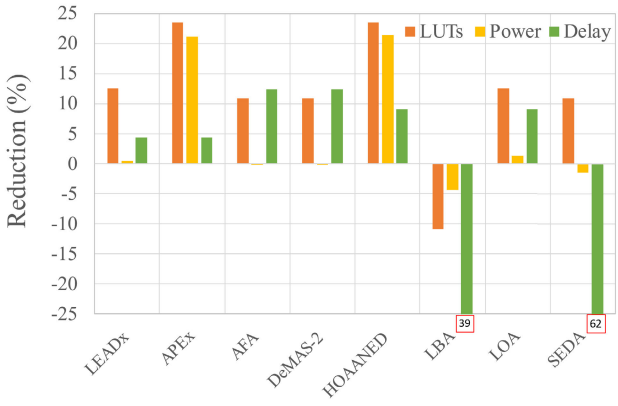

FIGURE 11. Area, power, and delay reduction achieved with 16-bit approximation in 64-bit approximate adders compared to 64-bit accurate adder.

C. COMPARISON WITH SEGMENTED AND SPECULATIVE APPROXIMATE ADDERS

In this section, we compare the proposed approximate adders with segmented and speculative adders in the literature; Almost Correct Adder (ACA-I) [9], Accuracy Configurable Adder (ACA-II) [11], Block-based Carry Speculative Adder (BCSA) [24], Error-tolerant adder II (ETA-II) [10], and xUAV [23].

The quality and implementation results of 16-bit adders with different approximation amounts are given in Table 7. These adders have same delay (1.35 ns). These adders are implemented with input and output registers. Therefore, although their critical paths are different, their speed is limited by the maximum frequency supported by Virtex 7 FPGA.

xUAV is an FPGA-specific segmented adder. Several configurations of xUAV are proposed in [23]. We used two most

TABLE 7. Comparison of 16-bit proposed approximate adders with 16-bit segmented and speculative approximate adders.

Adder	m*	r*	LUTs	Power (mW)	ME	RMSE	ER (%)
LEADx	4	–	14	6.13	4	1.39	34.88
	8	–	12	6.09	72	23.29	63.15
APEx	4	–	13	5.45	3	1.58	74.90
	8	–	9	4.32	63	26.13	98.04
ACA-I [9]	4	1	29	6.25	34944	6702	34.13
	8	1	72	6.78	32768	1689	1.59
ACA-II [11]	4	2	22	6.29	17472	5232	47.88
	8	4	24	6.23	4096	703	5.90
BCSA [24]	4	4	24	6.42	4368	1029	6.20
	8	8	16	6.12	256	63	17.04
ETA-II [10]	4	2	22	6.30	17472	5232	47.88
	8	4	29	6.20	4096	703	5.90
xUAV [23]	3	1	16	6.24	37448	9615	61.84
	5	1	26	6.42	33824	4754	16.72

*m is the size of approximate part (LSP) of LEADx and APEx. For other adders, m is the segment size. For segmented and speculative adders, r is the number of resultant bits contributing to the final sum from each segment.

TABLE 8. Impact of approximate adders on HEVC encoder bitrate and PSNR.

Adder	Video Sequence							
	Traffic (2560x1600)		BQ Terrace (1920x1080)		Four People (1280x720)		Party Scene (832x480)	
	ΔBR (%)	PSNR (dB)	ΔBR (%)	PSNR (dB)	ΔBR (%)	PSNR (dB)	ΔBR (%)	PSNR (dB)
Accurate	–	37.35	–	34.69	–	39.58	–	33.44
LEADx	3.93	37.05	1.49	34.51	2.07	39.38	1.86	33.29
APEx	4.33	37.03	1.98	34.50	2.08	39.38	1.98	33.28
AFA [17]	4.46	36.89	2.28	34.50	2.55	39.34	2.90	33.24
DeMAS–2 [20]	3.98	37.06	2.64	34.50	2.20	39.29	1.87	33.29
HOAANED [16]	10.70	36.76	3.54	34.44	4.68	39.28	4.37	33.18
LBA [19]	11.35	36.77	3.57	34.44	3.89	39.27	4.49	33.18
LOA [14]	11.78	36.76	3.77	34.45	3.63	39.30	3.89	33.19
SEDA [22]	11.61	36.75	3.66	34.44	2.87	39.27	4.26	33.15

efficient configurations; one with the lowest error ($m = 5$, $r = 1$) and the other with low error and low area ($m = 3$, $r = 1$).

The segmented and speculative adders follow fail-rare approach [25]. They have low ER. But their error magnitudes are usually large. Therefore, these adders have high MAE and MSE. For example, ACA-I with 8-bit segmentation has only 1.5% ER. However, its ME is 2^{15}. Most of the errors that occur in ACA-I have large magnitude, resulting in significantly high MSE.

Among the segmented and speculative adders, BCSA with 8-bit segmentation has the best quality. ETA-II and ACA-II have similar architecture. Therefore, their error metrics are similar. However, for 8-bit segmentation, ACA-II is more area efficient than ETA-II.

LEADx and APEx have better quality, smaller area, and lower power consumption than the segmented and speculative adders. These results show that, for uniformly distributed inputs, fail-small approach gives better quality than fail-rare approach.

D. CASE STUDY: MOTION ESTIMATION IN VIDEO ENCODING

We also assessed the impact of the proposed approximate adders and the other approximate adders on video encoding quality. C++ implementations of 8-bit adders with 4-bit approximation are integrated into High Efficiency Video Coding (HEVC) reference software HM 16.14 video encoder.

The approximate adders are used for sum of absolute difference (SAD) computations for motion estimation (ME). ME accounts for approximately 70% of the computational complexity of video encoding [13]. The search strategy is set to fast test zone search (TZ). The quality results are obtained for four video sequences with different spatial resolutions.

For each approximate adder, PSNR result in dB and the percentage increase in bitrate (ΔBR) with respect to using accurate adder are shown in Table 8. LEADx has the least quality loss, i.e., lowest PSNR decrease and lowest bitrate increase, compared to the other approximate adders.

V. CONCLUSION

In this paper, two low error efficient approximate adders for FPGAs are proposed. The first approximate adder, LEADx, has lower MSE than the approximate adders in the literature. It also achieves better quality than the other approximate adders for video encoding application. The second approximate adder, APEx, has same area, lower MSE and less power consumption than the smallest and lowest power consuming approximate adder in the literature. It has smaller area and lower power consumption than the other approximate adders in the literature. Its MSE is second only to LEADx. Therefore, the proposed approximate adders can be used for FPGA implementations of error tolerant applications.

REFERENCES

[1] G. A. Gillani, M. A. Hanif, B. Verstoep, S. H. Gerez, M. Shafique, and A. B. J. Kokkeler, "MACISH: Designing approximate MAC accelerators with internal-self-healing," *IEEE Access*, vol. 7, pp. 77142–77160, 2019.
[2] E. Kalali and I. Hamzaoglu, "An approximate HEVC intra angular prediction hardware," *IEEE Access*, vol. 8, pp. 2599–2607, 2020.
[3] T. Ayhan and M. Altun, "Circuit aware approximate system design with case studies in image processing and neural networks," *IEEE Access*, vol. 7, pp. 4726–4734, 2019.
[4] W. Ahmad and I. Hamzaoglu, "An efficient approximate sum of absolute differences hardware for FPGAs," in *Proc. IEEE Int. Conf. Consum. Electron. (ICCE)*, Las Vegas, NV, USA, Jan. 2021, pp. 1–5.

[5] 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, pp. 1–34, Aug. 2017.

[6] A. C. Mert, H. Azgin, E. Kalali, and I. Hamzaoglu, "Novel approximate absolute difference hardware," in *Proc. 22nd Euromicro Conf. Digit. Syst. Design (DSD)*, Kallithea, Greece, Aug. 2019, pp. 190–193.

[7] N. Van Toan and J.-G. Lee, "FPGA-based multi-level approximate multipliers for high-performance error-resilient applications," *IEEE Access*, vol. 8, pp. 25481–25497, 2020.

[8] L. Chen, J. Han, W. Liu, P. Montuschi, and F. Lombardi, "Design, evaluation and application of approximate high-radix dividers," *IEEE Trans. Multi-Scale Comput. Syst.*, vol. 4, no. 3, pp. 299–312, Jul. 2018.

[9] A. K. Verma, P. Brisk, and P. Ienne, "Variable latency speculative addition: A new paradigm for arithmetic circuit design," in *Proc. Design, Automat. Test Eur. (DATE)*, Munich, Germany, Mar. 2008, pp. 1250–1255.

[10] N. Zhu, W. L. Goh, G. Wang, and K. S. Yeo, "Enhanced low-power high-speed adder for error-tolerant application," in *Proc. Int. SoC Design Conf.*, Incheon, South Korea, Nov. 2010, pp. 323–327.

[11] A. B. Kahng and S. Kang, "Accuracy-configurable adder for approximate arithmetic designs," in *Proc. 49th Annu. Design Automat. Conf. (DAC)*, New York, NY, USA, 2012, pp. 820–825.

[12] V. Gupta, D. Mohapatra, A. Raghunathan, and K. Roy, "Low-power digital signal processing using approximate adders," *IEEE Trans. Comput.-Aided Design Integr. Circuits Syst.*, vol. 32, no. 1, pp. 124–137, Jan. 2013.

[13] W. Ahmad, B. Ayrancioglu, and I. Hamzaoglu, "Comparison of approximate circuits for H.264 and HEVC motion estimation," in *Proc. 23rd Euromicro Conf. Digit. Syst. Design (DSD)*, Kranj, Slovenia, Aug. 2020, pp. 167–173.

[14] H. R. Mahdiani, A. Ahmadi, S. M. Fakhraie, and C. Lucas, "Bio-inspired imprecise computational blocks for efficient VLSI implementation of soft-computing applications," *IEEE Trans. Circuits Syst. I, Reg. Papers*, vol. 57, no. 4, pp. 850–862, Apr. 2010.

[15] A. Dalloo, A. Najafi, and A. Garcia-Ortiz, "Systematic design of an approximate adder: The optimized lower part constant-OR adder," *IEEE Trans. Very Large Scale Integr. (VLSI) Syst.*, vol. 26, no. 8, pp. 1595–1599, Aug. 2018.

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

[17] S. Dutt, S. Nandi, and G. Trivedi, "Analysis and design of adders for approximate computing," *ACM Trans. Embedded Comput. Syst.*, vol. 17, p. 40, Dec. 2017.

[18] D. Celia, V. Vasudevan, and N. Chandrachoodan, "Optimizing power-accuracy trade-off in approximate adders," in *Proc. Design, Automat. Test Eur. Conf. Exhib. (DATE)*, Dresden, Germany, Mar. 2018, pp. 1488–1491.

[19] A. Becher, J. Echavarria, D. Ziener, S. Wildermann, and J. Teich, "A LUT-based approximate adder," in *Proc. IEEE 24th Annu. Int. Symp. Field-Program. Custom Comput. Mach. (FCCM)*, Washington, DC, USA, May 2016, p. 27.

[20] B. S. Prabakaran, S. Rehman, M. A. Hanif, S. Ullah, G. Mazaheri, A. Kumar, and M. Shafique, "DeMAS: An efficient design methodology for building approximate adders for FPGA-based systems," in *Proc. Design, Automat. Test Eur. Conf. Exhib. (DATE)*, Dresden, Germany, Mar. 2018, pp. 917–920.

[21] S. Boroumand, H. P. Afshar, and P. Brisk, "Approximate quaternary addition with the fast carry chains of FPGAs," in *Proc. Design, Automat. Test Eur. Conf. Exhib. (DATE)*, Dresden, Germany, Mar. 2018, pp. 577–580.

[22] C. K. Jha, K. Prasad, A. S. Tomar, and J. Mekie, "SEDAAF: FPGA based single exact dual approximate adders for approximate processors," in *Proc. IEEE Int. Symp. Circuits Syst. (ISCAS)*, Seville, Spain, Oct. 2020, pp. 1–5.

[23] T. Nomani, M. Mohsin, Z. Pervaiz, and M. Shafique, "XUAVs: Towards efficient approximate computing for UAVs-low power approximate adders with single LUT delay for FPGA-based aerial imaging optimization," *IEEE Access*, vol. 8, pp. 102982–102996, 2020.

[24] F. Ebrahimi-Azandaryani, O. Akbari, M. Kamal, A. Afzali-Kusha, and M. Pedram, "Block-based carry speculative approximate adder for energy-efficient applications," *IEEE Trans. Circuits Syst. II, Exp. Briefs*, vol. 67, no. 1, pp. 137–141, Jan. 2020.

[25] V. K. Chippa, S. T. Chakradhar, K. Roy, and A. Raghunathan, "Analysis and characterization of inherent application resilience for approximate computing," in *Proc. 50th Annu. Design Automat. Conf. (DAC)*, Austin, TX, USA, 2013, pp. 1–9.

[26] Y. Wu, C. Shen, Y. Jia, and W. Qian, "Approximate logic synthesis for FPGA by wire removal and local function change," in *Proc. 22nd Asia South Pacific Design Automat. Conf. (ASP-DAC)*, Chiba, Japan, Jan. 2017, pp. 163–169.

[27] Xilinx. (Sep. 2016). *7 Series Configurable Logic Block*. Accessed: 2021. [Online]. Available: https://www.xilinx.com/support/documentation/user_guides/ug474_7Series_CLB.pdf

[28] A. Ehliar, "Optimizing Xilinx designs through primitive instantiation," in *Proc. FPGA World Conf.*, Copenhagen, Denmark, 2010, pp. 20–27.

[29] A. Leon-Garcia, *Probability, Statistics, and Random Processes for Electrical Engineering*. Upper Saddle River, NJ, USA: Prentice-Hall, 2008.

[30] R. J. Hyndman and G. Athanasopoulos, *Forecasting: Principles and Practice*, 2nd ed. Melbourne, VIC, Australia: OTexts, 2018.

[31] E. T. Jaynes, *Probability Theory: The Logic of Science*. Cambridge, U.K.: Cambridge Univ. Press, 2003.

[32] D. P. Doane and L. E. Seward, "Measuring skewness: A forgotten statistic?" *J. Statist. Educ.*, vol. 19, no. 2, pp. 1–18, Jul. 2011.

[33] C. Niemann, M. Rethfeldt, and D. Timmermann, "Approximate multipliers for optimal utilization of FPGA resources," in *Proc. 24th Int. Symp. Design Diag. Electron. Circuits Syst. (DDECS)*, Vienna, Austria, Apr. 2021, pp. 23–28.

WAQAR AHMAD received the B.S. degree in electronics engineering from Air University, Islamabad, Pakistan, in 2007, and the M.S. degree in computer sciences and technology from Northwestern Polytechnic University, Xi'an, China, in 2011. He is currently pursuing the Ph.D. degree in electronics engineering with Sabanci University, Istanbul, Turkey. His research interests include approximate computing, FPGA design optimization, and energy-efficient digital hardware design for video coding.

BERKE AYRANCIOGLU received the B.S. degree in electronics engineering from Sabanci University, Istanbul, Turkey, in 2019, where he is currently pursuing the M.S. degree in electronics engineering. His research interests include low-power and high-performance digital hardware design, FPGA design, video compression, and approximate hardware design.

ILKER HAMZAOGLU (Senior Member, IEEE) received the B.S. and M.S. degrees in computer engineering from Bogazici University, Istanbul, Turkey, in 1991 and 1993, respectively, and the Ph.D. degree in computer science from the University of Illinois at Urbana–Champaign, IL, USA, in 1999. He worked as a Senior and Principle Staff Engineer with Multimedia Architecture Laboratory, Motorola Inc., Schaumburg, IL, USA, from August 1999 to August 2003. He is currently an Associate Professor with Sabanci University, Istanbul, where he has been working as a Faculty Member, since September 2003. His research interests include low-power digital hardware design for video processing and compression, system-on-chip (SoC) ASIC and FPGA design, approximate computing, and high level synthesis.