IJCRT.ORG ISSN : 2320-2882

INTERNATIONAL JOURNAL OF CREATIVE RESEARCH THOUGHTS (IJCRT)

An International Open Access, Peer-reviewed, Refereed Journal

DESIGN METHODOLOGY OF ERROR REDUCED APPROXIMATE ADDERS FOR FPGAS

[1] M.Karthiga and [2] Dr.S.Mohideen Abdul Kadhar

[1]P.G Student ,[2] Professor

[1] Department of ECE,,

[1] SriVidya College of Engineering and Technology,Virudhunagar

Abstract— In this paper, we propose novel low error approximate adder structures for FPGA-based implementations. By using the proposed adder structures, the throughput of an FPGA-based implementation can be significantly increased. We propose two approximate adder structurers for FPGAs in this paper : one is low error and area efficient approximate adder (LEADx), and another one is area and power efficient approximate adder (APEx). Both approximate adders are consists of an accurate and an approximate part. These kind of adders are designed to minimize the mean square error (MSE). When compared with other approximate adders LEADx has lower MSE whereas APEx has smaller area and lower power consumption. When compared with other existing approximate adders LEADx provided better quality for video encoding application.

Keywords:- Approximate computing, approximate adder structure , FPGA, Low power designs, Low area designs LUT

1. INTRODUCTION

Approximate computing may be a new style technique that trades off accuracy for performance, space and/or power consumption for error-tolerant applications like video cryptography. The video compression is error-tolerant in nature since the sole demand is to provide output that has sufficient quality to produce sensible user expertise. Therefore, approximate computing incorporates a large potential to boost the performance, space and/or power consumption of hardware implementations. FPGAs function a superb platform for a good vary of applications from small-scale embedded devices to superior computing systems because of their short time-to-market, increased flexibility and run-time re-configurability. Therefore, besides using ancient energy improvement techniques, there's a necessity for exploring new avenues in energy-efficient computing only for FPGA-based systems. Approximate computing trades the accuracy and preciseness of intermediate or final computations to realize vital gains in essential path delay, area, power and/or energy consumption. This trade-off becomes helpful for applications exhibiting inherent application resilience, i.e., the power to provide viable output despite a number of its computations being inaccurate due to approximations. a good vary of applications like image and video process, data processing, machine learning, etc., within the recognition, mining and synthesis domains exhibit this property. Existing approximate computing techniques and principles is applied to completely different stages of the computing stack, starting from logic and architectures at the hardware layer all the high to compiler and programing language at the software system layer. there's an in depth quantity of analysis associated with approximations at each hardware and software system layers. Voltage over-scaling and purposeful approximation square measure the 2 major approximate computing knobs utilized at the hardware level. Approximations at the software system level is classified into 2 major categories: (i) loop perforation, perform approximation and (ii) programing language support. Approximations at the hardware level square measure targeted on basic computation modules like adders and multipliers. analysis works like concentrate on modeling the error chance of the prevailing state of-the-art ASIC-based adders and algorithmic multiplier factor architectures. The basic building block of associate FPGA square measure the Configurable logic blocks or CLBs. they're accustomed implement any quite logic perform exploitation the switching/routing matrix. every CLB consists of 2 slices FPGA family arranges all the CLBs in columns by deploying Advanced atomic number 14 standard Block (ASMBL) design. every slice during this device consists of 4 6- input LUTs, eight flip-flops associated an economical carry chain logic. The slices act because the main perform generators of any FPGA and within the Virtex-7 family they're categorised as SLICEL or logic slices, and SLICEM or memory slices. The operation tables gift in these slices square measure 5x2 LUTs. This LUT6_2 is fancied exploitation 2 LUT5s and a electronic device. These LUT5s square measure the fundamental SRAM parts that square measure accustomed notice the specified logic perform, by storing the output sequence of the truth-table in 1-bit memory locations, that square measure accessed exploitation the address lines acting as inputs. These LUTs square

measure created accessible employing a big selection of operation table primitives offered by the Xilinx UNISIM library, starting from LUT1 to LUT6. These LUT primitives square measure instantiated with associate INIT attribute, that is that the truth table of the perform needed supported the input logic. The LUT primitives square measure accustomed implement the specified logic perform that square measure then compacted and mapped onto the material resources accessible on the FPGA. every of those LUT primitives soak up the same range of 1-bit inputs, and manufacture a singular 1-bit output. However, at the hardware level, every of those primitives square measure physically mapped to at least one of the 2 LUT5s gift within the four LUT6_2 materials in a very given slice of the CLB.As per studies, the employment of LUT primitives permits for Xilinx to expeditiously optimize the combining and mapping of LUT primitives to cut back the realm and latency of the synthesized styles. we tend to use these LUT primitives to realize vital space and performance gains for the approximate adder styles. Approximate computing trades off accuracy to boost the realm, power, and speed of digital hardware. several computationally intensive applications like video encryption, video process, and computing square measure error resilient naturally because of the constraints of human perception or nonbeing of a golden declare the given downside. Therefore, approximate computing is accustomed improve the realm, power, and speed of digital hardware implementations of those error tolerant applications. A variety of approximate circuits, starting from system level styles to basic arithmetic circuits, are projected within the literature. Adders square measure utilized in most digital hardware, not just for binary addition however additionally for alternative binary arithmetic operations like subtraction, multiplication, and division. Therefore, several approximate adders are projected within the literature. All approximate adders exploit the very fact that essential path in associate adder is rarely used. Approximate adders is loosely classified into the subsequent categories: segmental adders, that divide n-bit adder into many r-bit adders operational in parallel; speculative adders, that predict the carry exploitation solely the few previous bits; and approximate full-adder primarily based adders, that approximate the correct full-adder at electronic transistor OR circuit level. segmental and speculative adders typically have higher speeds and bigger areas than correct adders. Approximate full-adder primarily based approximate n-bit adders use m-bit approximate adder within the least vital half (LSP) and (n − m)-bit correct adder within the most important half (MSP)

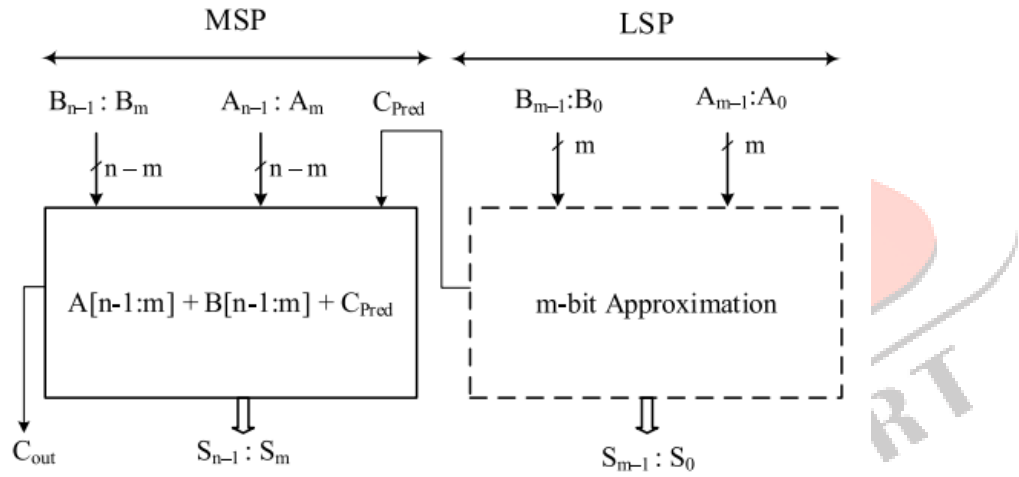

Fig.1 Generalized approximate adder

2. OBJECTIVES OF THE STUDY

In this paper, we tend to propose a technique to cut back the error of approximate adders by expeditiously utilizing FPGA resources, like unused LUT inputs. we tend to propose 2 approximate adders for FPGAs exploitation our methodology supported the design shown in Fig. above. we tend to propose a coffee error and space economical approximate adder (LEADx) for FPGAs. it's lower mean sq. error (MSE) than the approximate adders within the literature. It achieves higher quality than the opposite approximate adders for video coding application. we tend to additionally propose a region and power economical approximate adder (APEx) for FPGAs. though its MSE is on top of that of LEADx, it's under that of the approximate adders within the literature. it's identical space, lower MSE and fewer power consumption than the littlest and lowest power intense approximate adder within the literature. it's smaller space and lower power consumption than the opposite approximate adders within the literature.

3. EXISTING METHOD

Binary addition is key operation in most of the digital circuits. There are such a lot of adders within the digital style. the choice of adder depends on its performance parameters. Adders are necessary components in microprocessors, digital signal processors. ALU and in floating purpose arithmetic units. and memory addressing ,in booth multipliers .they are additionally utilized in real sign process like signal process, image process etc. for people at large arithmetic calculations are simple to calculate once they are decimals i.e. base ten. however they became pragmatic if binary numbers are given. so binary addition is important any improvement in binary addition will improve the performance of system. The quick and accuracy of system depends primarily on adder performance. during this paper planning and implementation of assorted parallel prefix adders on FPGA are delineate. “Lower-Part-OR Adder” Structure Description: Addition may be an elementary operation which will considerably influence the possible performances. The Lower-part-OR Adder (LOA) divides a -bit addition into 2 -bit and -bit smaller components. As shown in Fig.

below, a -bit LOA exploits a daily smaller precise adder (is referred to as sub-adder) that computes the precise values of the ‘ ’ important|most vital|most important} bits of the result (upper-part) at the side of some OR gates that approximate the ‘ ’ least significant result bits (lower-part) by applying bitwise OR to the various input bits. an additional AND circuit is employed to get a carry-in for the higher half sub-adder once the foremost important bits of each lower-part inputs ar one. This helps to require into consideration the trivial carries from lower to higher components of the LOA adder to decrease its inexactitude. The sub-adder may need any desired precise structure like carry look-ahead or ripple-carry adder, whereas exploitation a lot of economical precise adders because the sub-adder directly results in a lot of economical LOA. For a LOA with a continuing Word-Length (WL or ‘ ’), higher ‘ ’ or Lower-Part Length (LPL) values decrease the realm, delay, and power consumption of the LOA whereas at identical time increase its inexactitude

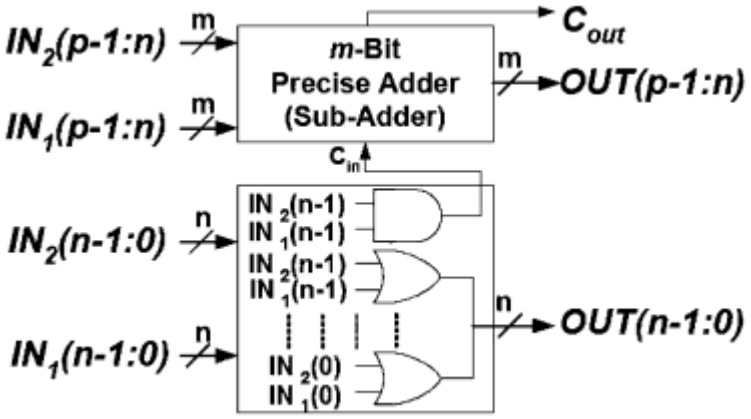

Fig.2 LOA approximate adder

4. PROPOSED METHOD

In the proposed methodology, approximate full adders supported FPGA LUT’s has been projected. 2 styles are projected one establishes a exchange between power and accuracy and also the different on space and accuracy. The projected style methodology uses the approximate full adder based mostly n-bit adder design shown in Fig. Generalized approximate adder, n-bit addition is split into n-bit approximate adder within the LSP and (n−m)-bit correct adder within the MSP. Breaking the carry chain at bit-position m typically introduces a slip-up of 2m within the final total. The error rate and error magnitude is reduced by predicting the carry-in to the MSP (CMSP) a lot of accurately and by modifying the logic perform of LSP to catch up on the error. The carry to the correct half is foretold exploitation any k-bit input pairs from the approximate half specified specified. Most of the prevailing approximate adders use k = one. FPGA implementation of correct adder uses solely two inputs and one output of every 6-input LUT. we tend to propose to utilize the remaining four, offered however unused, inputs of the primary LUT of the MSP to predict CMSP. Therefore, we tend to propose to share the foremost important two bits of each inputs of the LSP with the MSP for carry prediction. Sharing a lot of bits of LSP with MSP can increase the chance of properly predicting CMSP which is able to successively scale back error rate. However, this can additionally increase the realm and delay of the approximate adder. Analyze the exchange between the accuracy and performance of an FPGA-based approximate adder with completely different values of k. For k > 2, the error rate reduces slightly at the price of exaggerated space and delay. On the opposite hand, for k< 2, the delay improves marginally at the price of great increase within the error rate. Therefore, we tend to propose exploitation k = two, because it provides smart balance between accuracy and performance of approximate adders for FPGAs.

4.1 PROPOSED APPROXIMATE ADDERS

In the planned approximate adders, a carry is passed to the MSP if it's generated at bit position m − one, or generated at bit position m − two and propagated at bit position m − one. The CMSP is delineated by (4) wherever Gi and Pi are generate and propagate signals of the ith bit position, severally .Architecture of the planned approximate adders is shown in Fig. below.

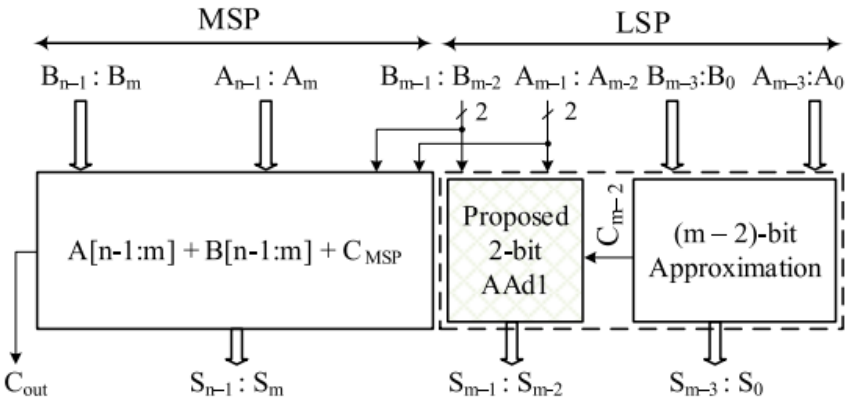

Fig.3 proposed approximate adder

It uses 2 MSBs of LSP to predict the CMSP, whereas their respective sum bits are computed using AAd1. Additional resources or unused LUT inputs are required for accurate prediction of Cout . 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 above.

The two proposed n-bit approximate adders use different approximate functions for the first m − 2 bits of the LSP. 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. To implement a 2-bit approximate adder LUTs are used.

For an area efficient FPGA implementation, we propose to split the first m − 2 bits of LSP into d(m − 2)/2e 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 Cout of ith group to one of the inputs of that group (Ai+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 Si and Si+1 output bits as follows:

- If the Cout is predicted correctly, the sum outputs are also calculated accurately using standard 2-bit addition.
- If the Cout is predicted incorrectly and the predicted value of Cout is 0, both sum outputs are set to 1.
- If the Cout is predicted incorrectly and the predicted value of Cout 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 below. The error cases are shown in red. the error probability of AAd2 is 0.25, Since AAd2 produces an erroneous result in 8 out of 32 cases,.

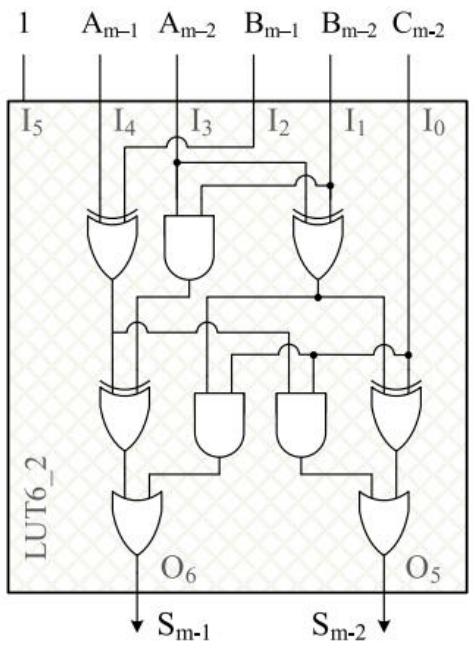

Fig.4 AAd1

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

Table.1 AAd2

4.2 LEADx APPROXIMATE ADDER

The planned LEADx approximate adder is shown in Fig. below. AN n-bit LEADx uses d(m − two)/2e copies of AAd2 adder within the least important m − 2 bits of the approximate adder design shown in Fig. planned approximate adder. In LEADx, Cm−2 = Am−3. AAd2 implements a 5-to-2 logic operate that's mapped to one LUT. Similarly, AAd1 is additionally mapped to one LUT. Therefore, dm/2e LUTs ar used for the LSP. These LUTs add parallel. Therefore, the delay of LSP is adequate the delay of one LUT (tLUT). The important path of LEADx is from the input Am−2 to the output Sn−1.

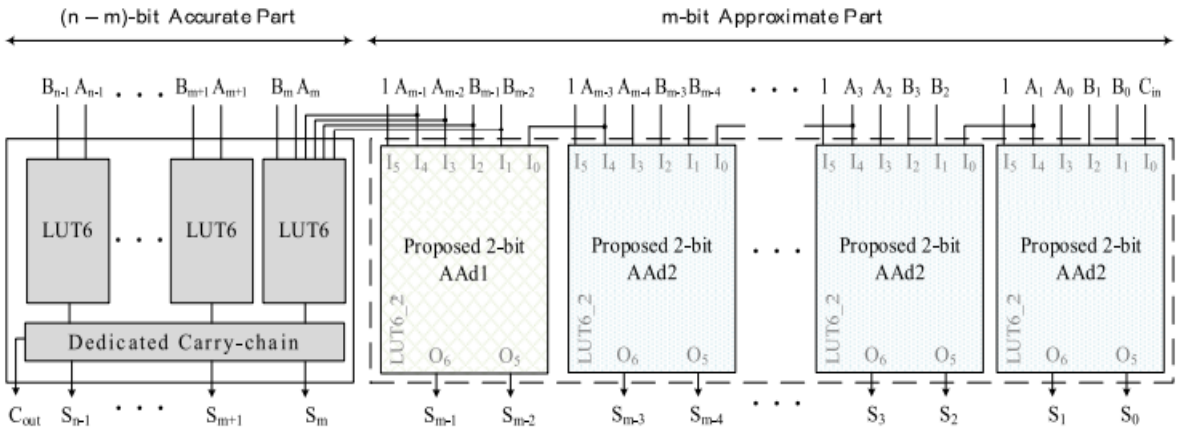

Fig.5 LEADx

4.3 APEx APPROXIMATE ADDER

APEx is also based on the approximate adder architecture shown in Fig. 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 ≤ (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 ith 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. If the least significant m − 2 bits are fixed to 1, the ME occurs when the inputs A0 to Am−3 and B0 to Bm−3 are all 0. With accurate addition, S0 to Sm−3 output bits are all 0 and carry is not propagated to m − 2 bit position. Fixing S0 to Sm−3 to 1 and carry-in for m − 2 bit position to 0 results in ME of 2m−2 − 1. The ME of constant 1 is less than the ME of constant 0.

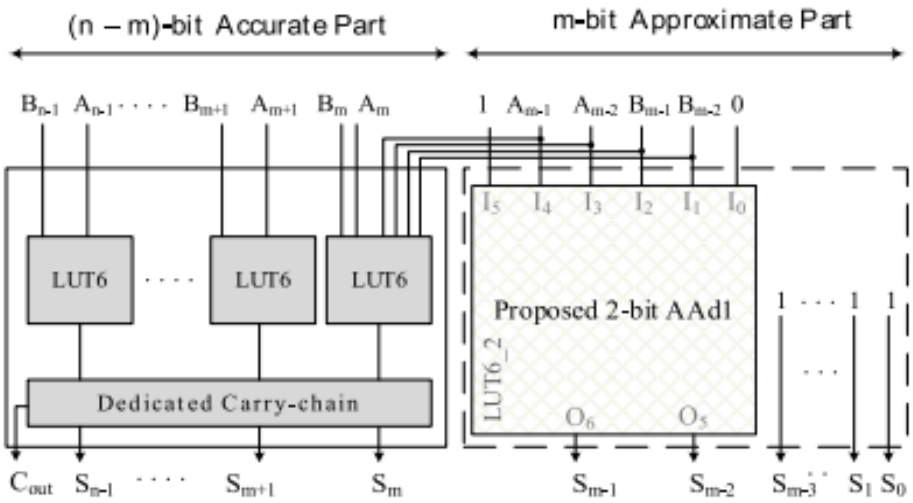

Fig.6 APEx

In the proposed APEx, the S0 to Sm−3 outputs are assigned to 1 and the Cm−2 is assigned to 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 2m+1 − 2 which is much higher than ME of APEx (2m−2 − 1). The proposed APEx approximate adder is shown in Fig. above. Same as LEADx, the critical path of APEx is from the input Am−2 to the output Sn−1.

5. RESULTS AND DISCUSSIONS

In this section, we present experimental results of the proposed approximate adders, LEADx and APEx. We compare LEADx and APEx with LOA. 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: and LOA . 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. 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 107 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.

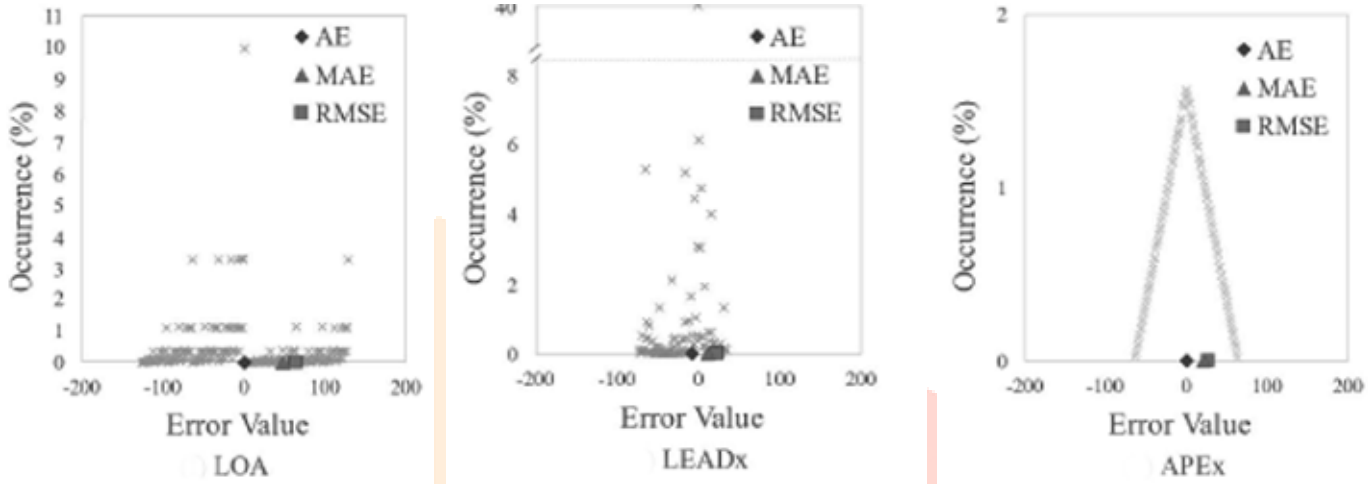

Fig. 7 Error Metrices of LOA, LEADx, APEx

The error distribution is plotted as a function of error value and its respective percentage occurrence. As can be seen in Fig. 7, 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 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 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.

6. IMPLEMENTATION RESULTS

Most of 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. AreaOptimized_high strategy is used for synthesis, and default strategy is used for implementation.

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 utilises 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. When compared to other approximate adders LEADx consumes slightly less power. 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 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. 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.. 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... With 8-bits approximation, LEADx has 7% smaller area and 86% lower MSE than DeMAS. 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. 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

7. 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.

8. 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 highspeed adder for error-tolerant application,’’ in Proc. Int. SoC Design Conf., Incheon, South Korea, Nov. 2010, pp. 323–327.