DOI: 10.48047/IJIEMR/V11/ISSUE 09/03

Title LOW ERROR EFFICIENT APPROXIMATE ADDERS FOR FPGAs

Volume 11, ISSUE 09, Pages: 23-28

Paper Authors

VANKAYALAPATI LAKSHMI PRASANNA, Dr. T. Raghavendra Vishnu

LOW ERROR EFFICIENT APPROXIMATE ADDERS FOR FPGAs

VANKAYALAPATI LAKSHMI PRASANNA
M-tech Student Scholar
Department of Electronics and Communication Engineering, Priyadarshini Institute Of Technology & Science For Women
Chintalapudi-522 306, Near Tenali, Guntur Dt. (A.P).

Dr. T. Raghavendra Vishnu M.Tech, Ph.D
Associate professor& HOD
Department of Electronics and Communication Engineering, Priyadarshini Institute Of Technology & Science For Women
Chintalapudi-522 306, Near Tenali, Guntur Dt. (A.P).

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

1. 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]. Allapproximate 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 implementedon 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.

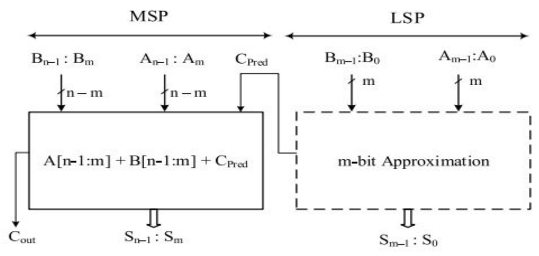


FIGURE 1. Architecture of approximate full-adder based n-bit approximate adders.

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.

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

2: PROPOSED DESIGN METHODOLOGY

The proposed design methodology uses the approximate fulladder based n-bit adder architecture shown in Fig. 1. n-bit addition is divided into n-bit approximate adder in the LSPand (n−m)-bit accurate adder in the MSP. Breaking the carry chain at bit-position m generally introduces an error of 2m in the final sum. The error rate and error magnitude can be reduced by predicting the carry-in to the MSP (CMSP) 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 $k = 1$.

Sharing more bits of LSP with MSP will increase the probability of correctly predicting CMSP 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 CMSP 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.

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 {CMSPSm−1Sm−2}. An error occurs in the n-bit addition if a carry (Cm−2) is generated at bit positioni < (m − 2) and that carry should be propagated to MSP. In this case, the correct result should be {CMSPSm−1Sm−2} = 100. However, without any error reduction mechanism the approximate result will be {CMSPSm−1Sm−2} = 000. To reduce the error magnitude, we propose a 2-bit approximate adder (AAd1) for computing Sm−1 and Sm−2. The functionality of AAd1 is described by (5) and (6). AAd1 is implemented using a single LUT as shown in Fig. 3. When Cm−2 = 1, Pm−2 = 1, and Pm−1 = 1, the approximate result will be {CMSPSm−1Sm−2} = 011, only 1 less than the accurate result. For all other inputs, it will generate the accurate result.

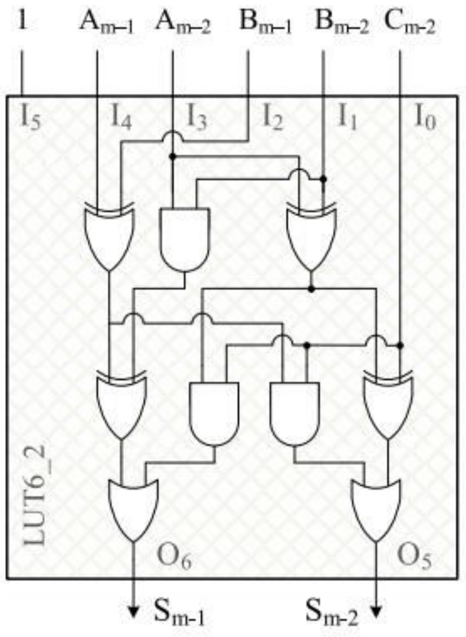


FIGURE 2.Proposed 2-bit approximate adder (AAd1) used in MSBs of LSP.

Architecture of the proposed approximate adders is shown in Fig. 4. It uses 2 MSBs of LSP to predict the CMSP, whereas their respective sum bits are computed using AAd1. AAd1 is only suitable when the Cout of 2-bit inputs is predicted accurately. Accurate prediction of Cout 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.

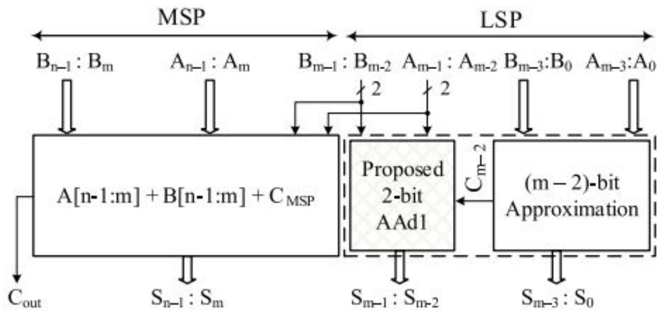


FIGURE 3.Architecture of proposed approximate adders for FPGAs.

3: 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 adderTo 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.

The proposed LEADx approximate adder is shown in Fig. 5. An n-bit LEADx uses d(m − 2)/2e copies of AAd2 adder in the least significant m − 2 bits of the approximate adder architecture shown in Fig. 4. In LEADx, Cm−2 = Am−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, dm/2e 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 (tLUT). The critical path of LEADx is from the input Am−2 to the output Sn−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 (CMSP) is predicted from the two MSBs of LSP as shown in (4).

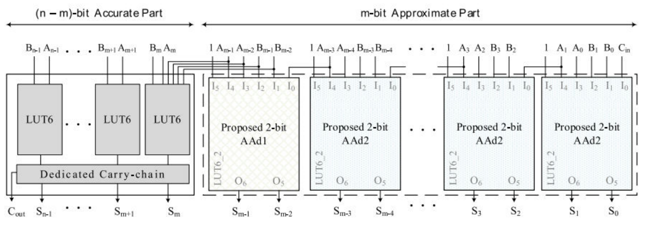


FIGURE 4. Proposed n-bit low error and area efficient approximate adder (LEADx).

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 ≤ (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. 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.

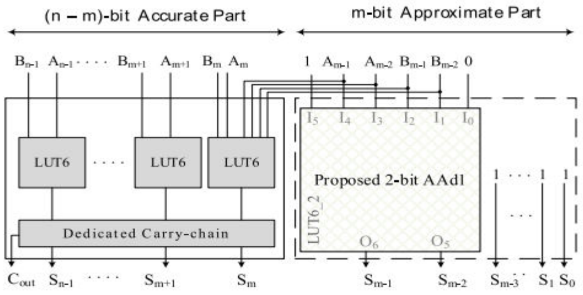


FIGURE 5: Proposed n-bit area and power efficient approximate adder (APEx).

In the proposed APEx, the S0 to Sm−3 outputs are fixed to 1 and the Cm−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 2m+1 − 2 which is much higher than ME of APEx (2m−2 − 1). The proposed APEx approximate adder is shown in Fig. 7. Same as LEADx, the critical path of APEx is from the input Am−2 to the output Sn−1. Similar to (9), the error probability of APEx can be calculated as shown in (13). When Cm−2 is 0, EAAd1 reduces to 0 according to (7). Therefore, the error probability of APEx depends only on the number of outputbits fixed to 1.

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 (CMSP) is predicted from the two MSBs of LSP as shown in (4).

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 interchangeformat (SAIF) files are also generated from these postimplementation 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. 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.

4. SIMULATION RESULTS

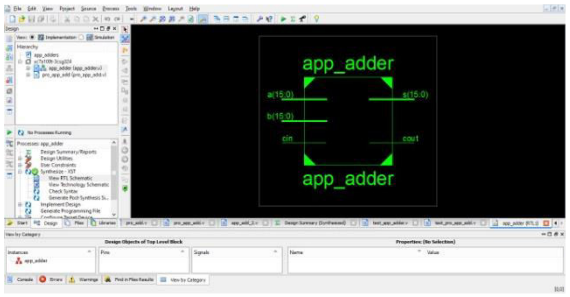


Figure 6: Block diagram of the approximate adder

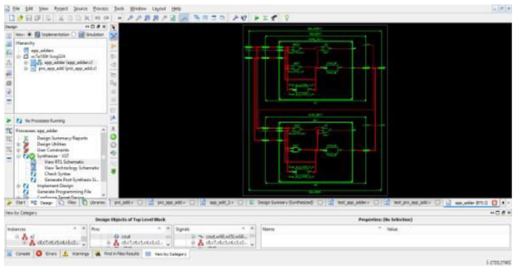


Figure 7: RTL Schematic of the approximate adder

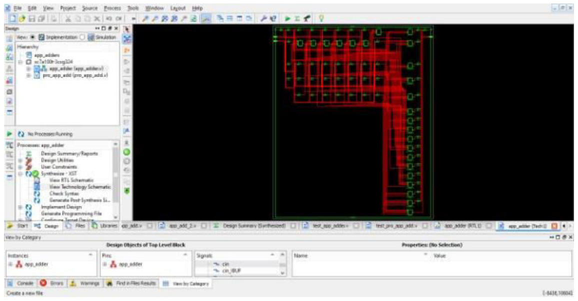
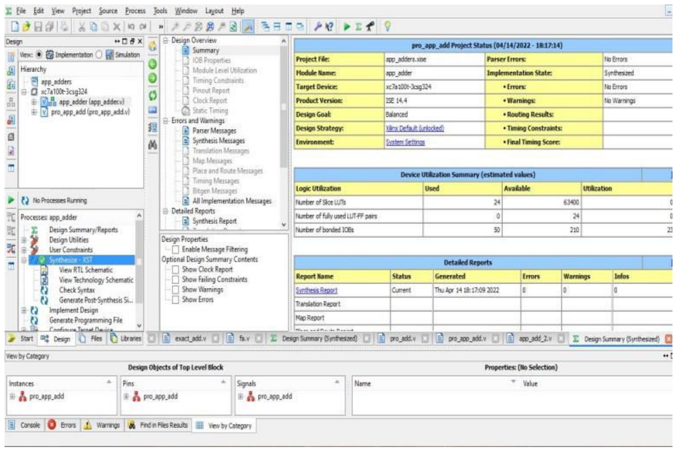


Figure 8: Technology Schematic of the approximate adder

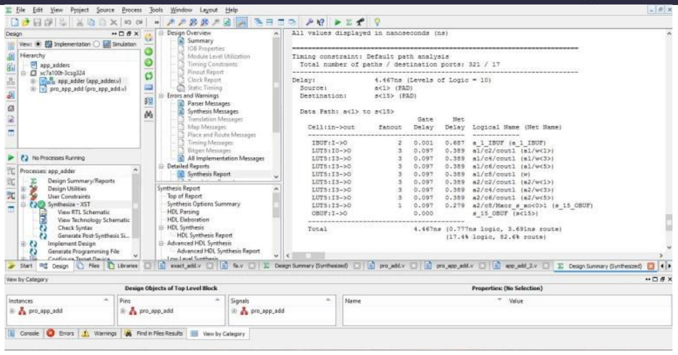


pro_app_add Project Status (04/14/2022 - 18:17:14)			
Project File:	app_adders.xise	Parser Errors:	No Errors
Module Name:	app_adder	Implementation State:	Synthesized
Target Device:	xc7a100t-3csg324	• Errors:	No Errors
Product Version:	ISE 14.4	• Warnings:	No Warnings
Design Goal:	Balanced	• Routing Results:	
Design Strategy:	Xilinx Default (unlocked)	• Timing Constraints:	
Environment:	System Settings	• Final Timing Score:	

Logic Utilization	Used	Available	Utilization
Number of Slice LUTs	24	63400	0
Number of fully used LUT-FF pairs	0	24	0
Number of bonded IOBs	50	210	23

Report Name	Status	Generated	Errors	Warnings	Infos
Synthesis Report	Current	Thu Apr 14 18:17:09 2022	0	0	0
Translation Report					
Map Report					

Figure 9: Summary report of the approximate adder



```
All values displayed in nanoseconds (ns)

=========================================================================
Timing constraint: Default path analysis
  Total number of paths / destination ports: 321 / 17
-------------------------------------------------------------------------
Delay:               4.467ns (Levels of Logic = 10)
  Source:            a<1> (PAD)
  Destination:       s<15> (PAD)

  Data Path: a<1> to s<15>
                                Gate     Net
    Cell:in->out      fanout   Delay   Delay  Logical Name (Net Name)
    ----------------------------------------  ------------
     IBUF:I->O             2   0.001   0.687  a_1_IBUF (a_1_IBUF)
     LUT5:I0->O            3   0.097   0.389  a1/c2/cout1 (a1/w<1>)
     LUT5:I3->O            3   0.097   0.389  a1/c4/cout1 (a1/w<3>)
     LUT5:I3->O            3   0.097   0.389  a1/c6/cout1 (a1/w<5>)
     LUT5:I3->O            3   0.097   0.389  a1/c8/cout1 (w)
     LUT5:I3->O            3   0.097   0.389  a2/c2/cout1 (a2/w<1>)
     LUT5:I3->O            3   0.097   0.389  a2/c4/cout1 (a2/w<3>)
     LUT5:I3->O            3   0.097   0.389  a2/c6/cout1 (a2/w<5>)
     LUT5:I3->O            1   0.097   0.279  a2/c8/Mxor_s_xo<0>1 (s_15_OBUF)
     OBUF:I->O                 0.000          s_15_OBUF (s<15>)
    ----------------------------------------
    Total                      4.467ns (0.777ns logic, 3.691ns route)
                                       (17.4% logic, 82.6% route)
```

Figure 10: Delay report of the approximate adder

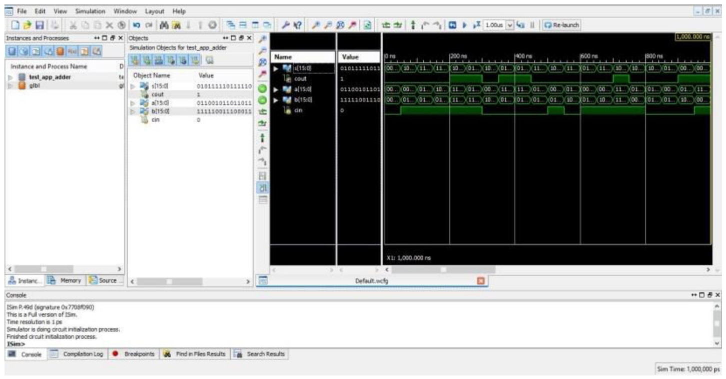


Figure 11: Simulation result of the approximate adder

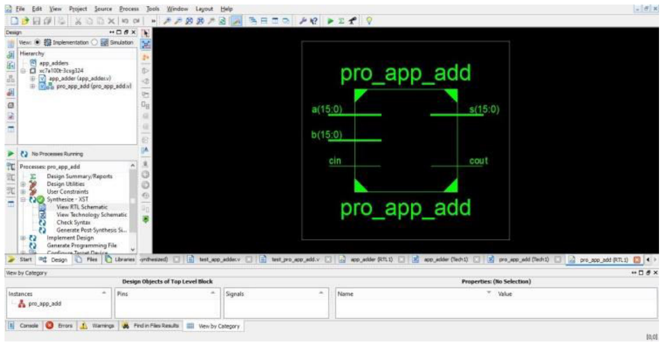


Figure 12: Block diagram of the proposed approximate adder

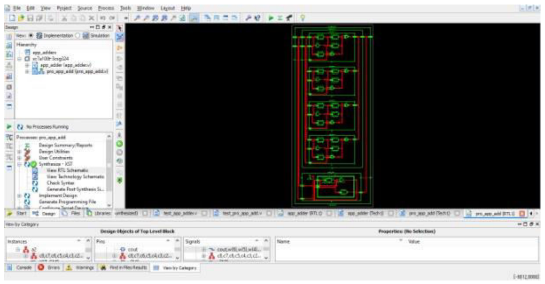


Figure 13: RTL Schematic of the proposed approximate adder

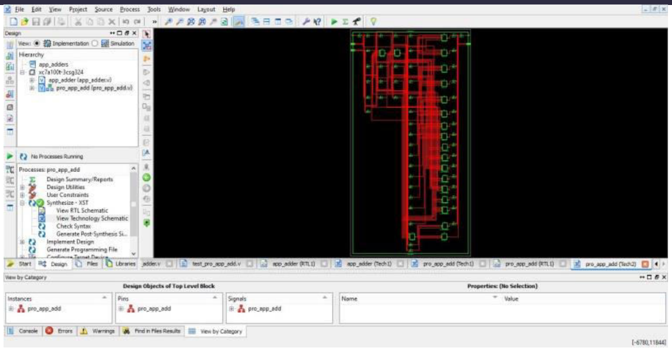
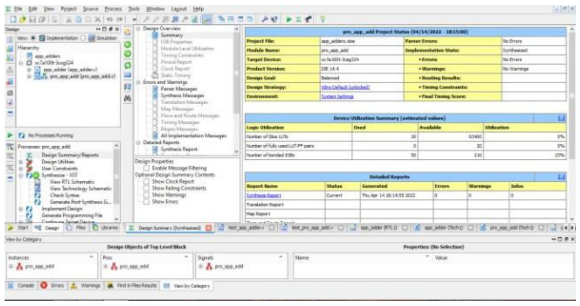


Figure 14: Technology Schematic of the proposed approximate adder

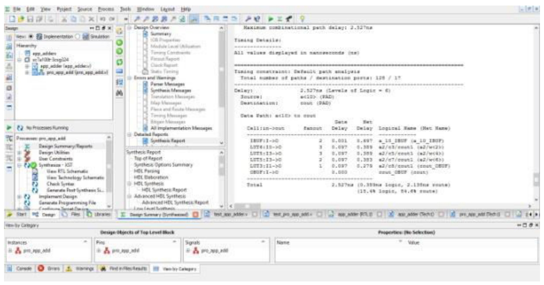


pro_app_add Project Status (04/14/2022 - 18:15:00)			
Project File:	app_adders.xise	Parser Errors:	No Errors
Module Name:	pro_app_add	Implementation State:	Synthesized
Target Device:	xc7a100t-3csg324	• Errors:	No Errors
Product Version:	ISE 14.4	• Warnings:	No Warnings
Design Goal:	Balanced	• Routing Results:	
Design Strategy:	Xilinx Default (unlocked)	• Timing Constraints:	
Environment:	System Settings	• Final Timing Score:	

Logic Utilization	Used	Available	Utilization
Number of Slice LUTs	20	63400	0%
Number of fully used LUT-FF pairs	0	20	0%
Number of bonded IOBs	50	210	23%

Report Name	Status	Generated	Errors	Warnings	Infos
Synthesis Report	Current	Thu Apr 14 18:14:55 2022	0	0	0
Translation Report					
Map Report					

Figure 15: Summary report of the proposed approximate adder



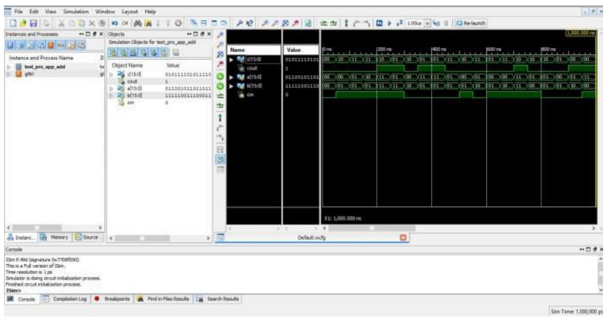
```
   Maximum combinational path delay: 2.527ns

Timing Details:
---------------
All values displayed in nanoseconds (ns)

=========================================================================
Timing constraint: Default path analysis
  Total number of paths / destination ports: 128 / 17
-------------------------------------------------------------------------
Delay:               2.527ns (Levels of Logic = 6)
  Source:            a<10> (PAD)
  Destination:       cout (PAD)

  Data Path: a<10> to cout
                                Gate     Net
    Cell:in->out      fanout   Delay   Delay  Logical Name (Net Name)
    ----------------------------------------  ------------
     IBUF:I->O             2   0.001   0.697  a_10_IBUF (a_10_IBUF)
     LUT6:I0->O            3   0.097   0.389  a2/c3/cout1 (a2/w<2>)
     LUT5:I3->O            3   0.097   0.389  a2/c5/cout1 (a2/w<4>)
     LUT5:I3->O            2   0.097   0.383  a2/c7/cout1 (a2/w<6>)
     LUT3:I1->O            1   0.097   0.279  a2/c8/cout1 (cout_OBUF)
     OBUF:I->O                 0.000          cout_OBUF (cout)
    ----------------------------------------
    Total                      2.527ns (0.389ns logic, 2.138ns route)
                                       (15.4% logic, 84.6% route)
```

Figure 16: Delay report of the proposed approximate adder


17: Simulation result of the approximate adder

5. 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] JasbirKaur, LalitSood “Comparison between various Types of Adder topologies” International journal Computer Science and Technology Vol.6, March 2015.

[2] Jasmine Saini, SomyaAgarwal, Aditi Kansa “Performance, Analysis and Comparison of Digital Adders “International Conference on advances in Computer Engineering and Application (ICACEA) 2015.

[3] R.P.P.Singh, Parveen Kumar “Performance Analysis Fast Adders using VHDL” International conference on Advances in recent technologies in communication and Computing2009.

[4] NidhiTiwari, Ruchi Sharma, Rajesh Pariha “Implementation Of Area and energy efficient full adder Cell” IEEE International Conference on recent advances and innovation in Engineering 2014.

[5] RajkumarSarma and VeeratiRaju “Design and Performance Analysis of hybrid adders for High speed arithmetic circuit” International Journal of VLSI Design and communication Systems (VLSICS) Vol 3, no 3 June2012.

[6] ArkadiyMorgenshtein, Alexander fish and Israel Wagner “Gate Diffusion input (GDI) – a technique for low power Design of Digital circuits Analysis and characterization IEEE International symposium on circuit and system February2012.

[7] Sentamilselvi M, Mahendran P “High performance Adder Circuit in VLSI system” International journal of Technology Enhancements and emerging engineering Research IJTEEE VOL 2 Issue3.