International Journal of Scientific Research in Science and Technology
Print ISSN: 2395-6011 | Online ISSN: 2395-602X (www.ijsrst.com)
doi : https://doi.org/10.32628/IJSRST

Design And Implementation of Imprecise Multiplier for FPGA Applications Based on LUT Based Adders

Priyanka Babalad[1], Pawankumar C Patil[2]

[1]PG Scholar, Department of ECE (VLSI & EMBEDDED SYSTEM), Sharanbasva University, Kalaburgi, India

[2]ASSISTANT PROFFESOR, Department of ECE, Sharanbasva University, Kalaburgi, India

Article Info
Volume 9, Issue 4
Page Number : 296-302

Publication Issue
July-August 2022

Article History
Accepted : 05 July 2022
Published : 20 July 2022

ABSTRACT

A methodology for constructing low-error, high-efficiency approximate adders has been provided in this project. To reduce the inaccuracy of approximation adders, the suggested solution effectively utilizes FPGA resources. We propose two approximate adders for FPGAs using our methodology: low error as well as area efficient approximate adder (LEADx), as well as area as well as power efficient approximate adder (APEx). Both approximate adders are composed of an accurate as well as 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. APEx has smaller area as well as lower power consumption than the other approximate adders than the existing adders. As a case study, In video encoding applications, approximation adders are employed. In the video encoding application, LEADx outperformed the other approximation adders. As a result, our proposed approximate adders can be used to create error-tolerant applications in FPGAs with efficiency. The effectiveness of the proposed method is synthesized as well as simulated using Xilinx ISE 14.7.

Keywords :- Approximate computing, approximate adder, FPGA, low error, low power, LUT.

I. INTRODUCTION

For error-tolerant multimedia applications coding, approximate computing is a new design strategy that trades accuracy for performance, size, as well as/or power consumption. Because the main criterion is to produce output of sufficient quality to give a decent user experience, the video compression is error-tolerant in nature. As a result, approximation computing provides a lot of promise for improving optimal design performance, area, as well as/or energy usage. Due to its rapid time-to-market, increased flexibility, as well as run-time re-configurability, FPGAs are a great platform for a variety of applications ranging from small-scale embedded devices to high-performance computing systems.

FPGA-based systems, on the other has well as, generally consume more power as well as/or energy than ASIC-based systems, despite providing specialized hardware accelerators as well as co-processors. As just a consequence, alternative routes in energy-efficient computation for FPGA-based systems should be explored in addition to existing energy optimization techniques.

An Approximation Computation paradigm is however one appealing tendency, which is re-emerging as a result of Moore's law as well as Dennard scaling collapsing, as well as the growing desire for high-performance as well as energy efficiency. To obtain significant improvements in critical chain latency, area, energy, as well as/or energy consumption, approximate computing compromises the accuracy as well as precision of intermediate or final computations.

This trade-off is advantageous for applications that have inherent application resilience, or the capacity to provide usable output even if part of the computations are wrong due to approximations. This property could be found in a wide range of identification, mining, as well as synthesis applications such as image as well as video processing, information retrieval, machine learning, as well as whatever else.

Current approximation computation methodologies as well as concepts could be implemented to various levels of the computer stack, from hardware circuitry as well as architectures to software compilers as well as computer languages. Both at the hardware as well as software layers, there is a lot of study on computations. The 2 significant approximation computation knobs employed at the hardware level are power over-scaling as well as functional approximation. Loop incision, linear regression, as well as computer language options are indeed the two major kinds of software estimates. Basic compute modules, such as adders but also multipliers, are indeed the subject of approximations at the hardware level. Modelling the prediction error of present state-of-the-art ASIC-based adders as well as recursive multiplier architectures is the topic of research works like this.

An Advanced Silicon Modular Block (ASMBL) architecture in the FPGA family arranges all of the CLBs in columns. This device has four 6-input LUTs, eight flip-flops, as well as fast carry chain logic in each slice. SLICEL (logic slices) as well as SLICEM (memory slices) are the fundamental function generator in any FPGA. Some slice feature 5x2 LUTs for search tables. A multiplexer as well as two LUT5s were used to create this LUT6 2. These LUT5s are the basic SRAM components that are employed to execute the required logic by storing the truth-table output in 1-bit memory locations that are accessed through the address lines that act as inputs. The Xilinx UNISIM library provides a broad array of lookup table troglodytes, spanning from LUT1 to LUT6 that can be used to access these LUTs. The truth table of the function required based on input logic is instantiated with the INIT property on these LUT primitives.

According to studies, Xilinx can efficiently optimize the merging as well as mapping of LUT primitives to reduce the space as well as latency of synthesis designs utilizing LUT primitives. The LUT primitives are used for the approximation adder designs to generate significant area as well as performance advantages. In order to reduce the size, power, as well as speed of digital technology, approximate computing sacrifices off accuracy. Because of the limitations of human visual perception or the lack of a golden answer for a particular problem, many computationally intensive technologies such as video encoding, video processing, as well as artificial intelligence are error resistant by nature.

II. EARLIER WORK

Approximate complete adders based on FPGA LUTs were proposed as part of the proposed technique. One design establishes a compromise between power and precision, while the other establishes a tradeoff between area and accuracy. The proposed design methodology employs the n-bit adder architecture depicted in Fig. 1 that is based on an approximation complete adder. In the LSP, n-bit addition is split into n-bit approximation adder and (nm)-bit accurate adder, while in the MSP, n-bit addition is split into n-bit approximate adder and (nm)-bit accurate adder.

In general, interrupting the carry chains at bit location m introduces a 2m mistake in the final sum. By the more accurately forecasting the carry-in to the MSP (CMSP) and altering the logic of the LSP to adjust for the error, the error rate and magnitude can be minimised. Any k-bit input pair from the approximation component can be used to anticipate the carry to a correct part if k m. k = 1 is used in the majority of current approximate adders.

Each 6-input LUT employs only 2 inputs & 1 output inside the FPGA implementation of the accurate adder. To forecast CMSP, authors propose using the first LUT of the MSP's remaining four available but unused inputs. As a result, for carry prediction, we propose sharing the most significant two bits of both LSP inputs with the MSP. More bits of LSP shared with MSP increases the likelihood of accurately predicting CMSP, lowering mistake rates. Nevertheless, this will increase the approximate adder's area as well as the delay.

It investigate how different values of k influence the reliability and efficiency of an FPGA-based approximation adder. Whenever k is more than two, then error rate was decreased marginally at the expense of increased area and delay. However, for k 2, the latency increases only slightly just at expense of a large rise in the error rate. As a result, we recommend adopting k = 2, which offers a good mix for precision for FPGA approximation adders.

If a carry is formed at bit position m 1 and propagate at bit position m 1, it is passed to the MSP in the proposed approximatios adder. (4) describes the CMSP, where Gi and Pi respectively create and propagate signals of the ith bit position. The proposed approximate adders are designed in the following way is shown in Fig.1 below.

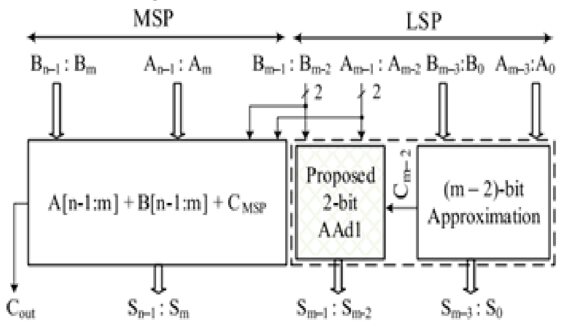


Fig.1: proposed approximate adder

. AAd1 is only appropriate when the CMSP is predicted precisely using two MSBs of LSP, with their corresponding total bits computed using Cout of two-bit inputs. Cout requires additional resources or unused LUT inputs for accurate prediction. AAd1 is thus not used in the least-significant m 2 bits of the LSP when designing area efficient approximate adders for FPGAs. In this paper, we use the design to propose two n-bit approximation adders in Fig 1above.

The first m 2 bits of the LSP are approximated differently in the two proposed n-bit approximate adders. 6-input LUTs are used in high-end FPGAs. Two 5-input functions can be realised using these LUTs. Performance of LUT-based implementation is unaffected by the complexity of the implemented logic function. The inputs and outputs of a 2-bit adder are both 5. As a result, a 2-bit approximation adder can be implemented with a LUT.

We propose dividing a first m 2 bits of LSB in d(m 2)/2e groups of 2-bit inputs, each of which is routed to a single LUT, for an area efficient FPGA implementation. Using a 2-bit approximation adder, thus every group adds two 2-bit inputs using carry-in (AAd2). We suggest equating Cout of ith group to one of that group's inputs (Ai+1) to eliminate the carry chain in LSP. This leads to a mistake in 8 of the 32 possible situations, with a magnitude of absolute error of 4 in each erroneous case. We propose computing the Si and Si+1 output bits as following that lessen the number of a error:

- If indeed the Cout is properly anticipated, the total outputs are calculated correctly as well, using standard 2-bit addition.
- Both sum outputs are set to 1 if the Cout is mistakenly anticipated and the projected value of Cout is 0.
- Cout is 1. Both sum outputs are set to 0 if the Cout is mistakenly anticipated and the predicted value of Cout is 1.

In two circumstances, the absolute error magnitude is reduced to 2, and in the remaining six cases, it is reduced to 1. Table 1 shows the AAd2 truth table as a result. Red denotes the error cases. The error probability of AAd2 is 0.25, because it provides an incorrect result in 8 out of 32 occurrences.

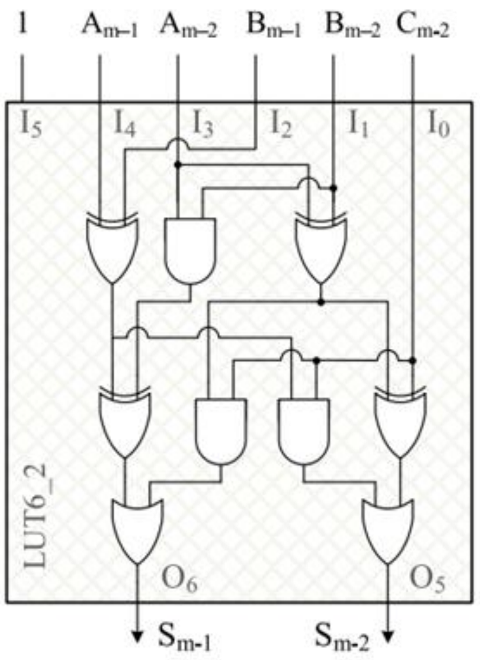


Fig.2: Structure of AAd1

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

III. PROPOSED WORK

The planned LEADx approximation adder is illustrated in figure 3.. In the least important m 2 bits of the approximate adder architecture presented in Fig. suggested approximate adder, an n-bit LEADx employs d(m 2)/2e copies of the AAd2 adder. Cm2 = Am3 in the LEADx system. AAd2 is a single-LUT implementation of a 5-to-2 logic function. AAd1 has a single LUT as well. As a result, the LSP is implemented using dm/2e LUTs. These LUTs are used in conjunction with one another. As a result, LSP's delay is the same as a single LUT's delay (tLUT). From the input Am2 to the output Sn1, LEADx's critical route is formed.

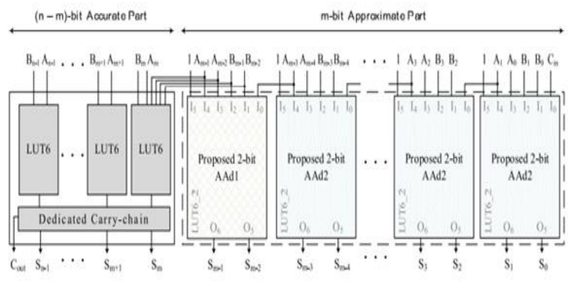


Fig.3: Architecture of LEADx

The approximation adder design depicted in Fig. 2 is also used in APEx. The goal is to find an expected completion that has no data reliance for the LSP's least significant m 2 bits. Carry should not be produced or used to calculate sums. Only a 1-bit sum output should be produced from a 1-bit input pair at any bit point I (m 2). In general, any logic function having a 1-bit output could be used to approximate the sum of 1-bit inputs at the ith bit location.

An approximation function with such a constant 0 or 1 as the output is also valid. Because no hardware is required for sum computation, setting the output to 0 or 1 lowers the space and power consumption of the approximate adder. When the least significant m-2 bits are set to 1, the ME occurs when all of the inputs A0 to Am-3 and B0 to Bm-3 are zero. S0 to Sm-3 output bits are all 0, and carry is not propagated to the m 2 bit position, indicating that accurate addition is used. Fixing S0 to Sm-3 to 1 and m 2 bit position carry-in to 0 yields ME of 2m 2 1. Constant 1's ME is smaller than Constant 0's ME.

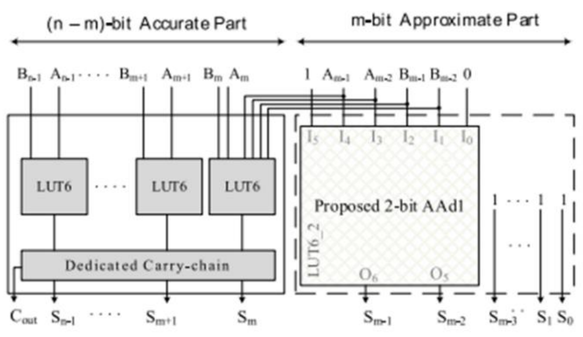

(n – m)-bit Accurate Part
m-bit Approximate Part
B_{n-1} A_{n-1} · · · · B_{m+1} A_{m+1} B_m A_m
1 A_{m-1} A_{m-2} B_{m-1} B_{m-2} 0
I_5 I_4 I_3 I_2 I_1 I_0
LUT6
LUT6
LUT6
Proposed 2-bit AAd1
LUT6_2
1 · · · 1 1
Dedicated Carry-chain
O_6 O_5
C_{out} S_{n-1} · · · · S_{m+1} S_m S_{m-1} S_{m-2} S_{m-3} S_1 S_0

Fig.4 Structure of APEX

The S0 to Sm-3 outputs, and the Cm-2, are all set to 1 in the proposed APEx. At the cost of a minor quality loss, this results in huge space and power savings. This differs from the bit truncation technique, which sets the both sum and carry outputs to 0. The ME of the truncation adder is 2m+1- 2, which is significantly greater than the ME of the APEx (2m-2- 1). In Figure 4, you can see the proposed APEx approximation adder. APEx's critical path, like LEADx's, runs from Am-2 to Sn-1.

The rate of the multiplication operation affects the speed of the intended circuit, hence fast multipliers are required in digital signal processing. Digital signal processing systems use fast multipliers as a component. Today, the speed of multiply operations is essential in both digital signal processing and general-purpose processors, especially since media processing became popular. Multiplication used to be done by adding, subtracting, and shifting numbers. Multiplication is a sequence of additions that are repeated. The multiplicand is the integer to be multiplied, the multiplier is the number of times it must be multiplied, and the result is the result. Each adding step yields a half-finished product. Most computers have the same number of bits as the operands. Whenever the operands are converted to integers, the result is usually half as long as the operas well as in order to keep the information content.

The following diagram depicts the general approach for unsigned multiplication in base b.

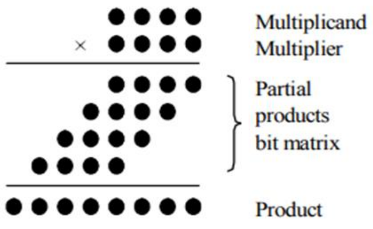


Fig.5: Multiplication process

When multiplying each digit of the multiplier times the multiplicand, each row or partial product is obtained. The low order digit of a partial product 4 is defined by just one multiplicand digit, but the effects of the carry from of the digits to the right are included in the other digits. Partial products in binary are simple – either one copy of the multiplicand or zero. The calculated result is the sum of the partial products.

In computer based systems, a multipliers is among the most significant arithmetic units. To develop an effective multiplier, a number of arithmetic strategies have been discussed. Wallace tree technique is a good example of a good method. This technique can generate functional hardware in three steps, consisting of full-adder and half-adder circuits that perform multiplication in parallel. Three stages make up the multiplication procedure:

Stage1. Multiply each bit of multiplicand by each bit of multiplier to get n2 partial products.

Stage2. By combining the layers of an FA and a HA block, you can reduce the amount of incomplete products.

Stage3. The preceding procedure yielded an n-bit adder by combining two n-sets. The second stage is carried out in the following manner. FA receives three bits of the same value, resulting in two bits of distinct values. (one bit of same value one and bit of a greater value).

- Put two bits with the identical value into the HA if they remain.
- Transfer the bit to the next layer if there is just one bit.

Our proposed adders can be used to minimize partial products in this case for an 8 bit multiplier. It's possible to see it like this:

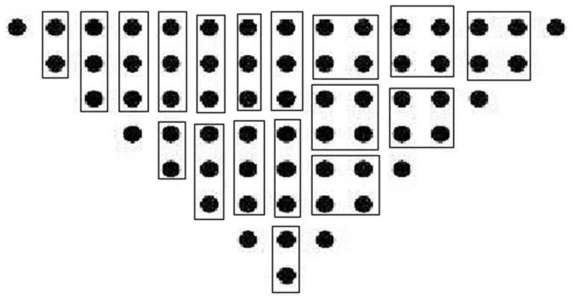


Fig.6: 8 bit multiplier partial products reduction with proposed 2 bit adder

IV. EXPERIMENTAL RESULTS

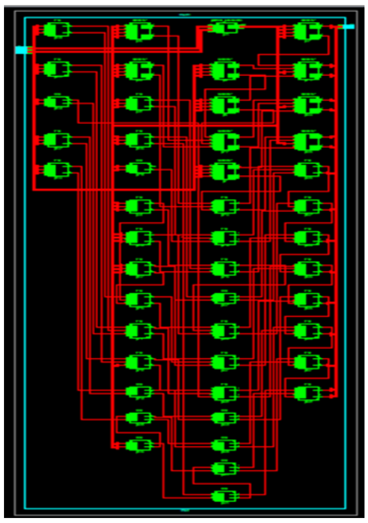


Fig.7: RTL schematic of proposed circuit.

Figure 7 shows RTL schematic of proposed approximate adder

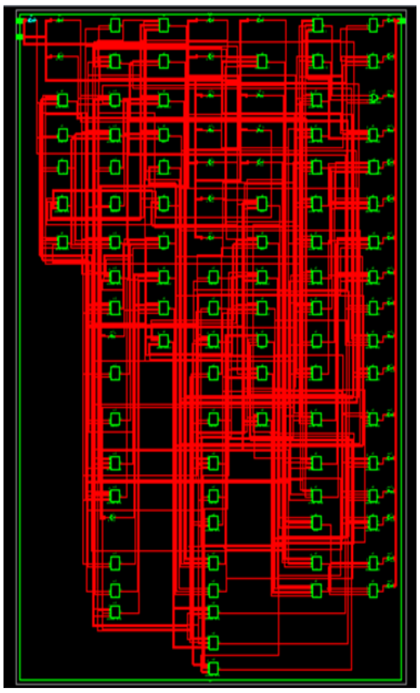
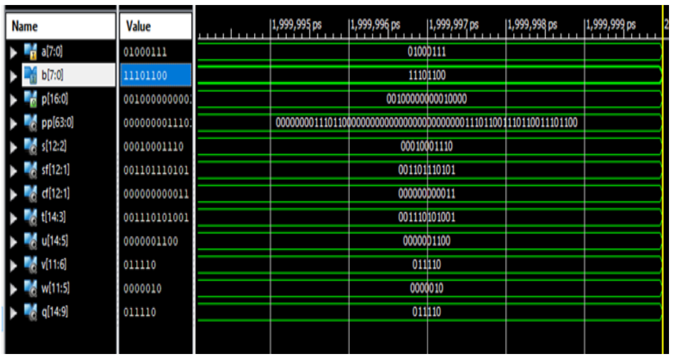


Fig.8 Technology schematic.

Figure 8 displays the Technology schematic of our suggested circuit.



Name	Value	
a[7:0]	01000111	01000111
b[7:0]	11101100	11101100
p[16:0]	001000000000	00100000000010000
pp[63:0]	0000000011101	0000000011101100000000000000000000000000111011001110110011101100
s[12:2]	00010001110	00010001110
sf[12:1]	001101110101	001101110101
cf[12:1]	000000000011	000000000011
t[14:3]	001110101001	001110101001
u[14:5]	0000001100	0000001100
v[11:6]	011110	011110
w[11:5]	0000010	0000010
q[14:9]	011110	011110

Fig9: Simulation results of proposed circuit.

V. CONCLUSION

Two low-error fast approximation adders for FPGAs are proposed in this study, with approximation taking place only in the LSP and the MSP remaining correct. In comparison to exact adder implementations, LEADx, the first approximate adder, achieves better area and delay. APEx, the second approximate adder, has a smaller footprint and consumes less energy than that of the existing adders. It takes up less space and uses less power than other approximation adders described in the literature. As a result, FPGA implementations of error-tolerant applications can benefit from the proposed approximate adders. By creating efficient ways in carry chains or accurate adders, the proposed adders can be developed efficiently by providing a trade-off between error and factors like area, delay, and power. It could also be used in multiplier architectures during the partial product reduction phase to improve design efficiency.

VI. REFERENCES

[1]. G. A. Gillani, M. A. Hanif, B. Verstoep, S. H. Gerez, M. Shafique, as well as A. B. J. Kokkeler, "MACISH: Designing approximate MAC accelerators with internal-self-healing," IEEE Access, vol. 7, pp. 77142–77160, 2019.

[2]. E. Kalali as well as I. Hamzaoglu, "An approximate HEVC intra angular prediction hardware," IEEE Access, vol. 8, pp. 2599–2607, 2020.

[3]. T. Ayhan as well as M. Altun, "Circuit aware approximate system design with case studies in image processing as well as neural networks," IEEE Access, vol. 7, pp. 4726–4734, 2019.

[4]. W. Ahmad as well as 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, as well as J. Han, "A review, classification, as well as 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, as well as 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 as well as 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, as well as F. Lombardi, "Design, evaluation as well as 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, as well as 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.

Cite this article as :

Priyanka Babalad, Pawankumar C Patil, "Design And Implementation of Imprecise Multiplier for FPGA Applications Based on LUT Based Adders", International Journal of Scientific Research in Science and Technology (IJSRST), Online ISSN : 2395-602X, Print ISSN : 2395-6011, Volume 9 Issue 4, pp. 296-302, July-August 2022.
Journal URL : https://ijsrst.com/IJSRST229447