iJRASET
International Journal For Research in Applied Science and Engineering Technology

# INTERNATIONAL JOURNAL FOR RESEARCH

IN APPLIED SCIENCE & ENGINEERING TECHNOLOGY

**Volume:** 13 **Issue:** IV **Month of publication:** April 2025

**DOI:** https://doi.org/10.22214/ijraset.2025.68534

www.ijraset.com

Call: 08813907089 | E-mail ID: ijraset@gmail.com

International Journal for Research in Applied Science & Engineering Technology (IJRASET)
*ISSN: 2321-9653; IC Value: 45.98; SJ Impact Factor: 7.538*
*Volume 13 Issue IV Apr 2025- Available at www.ijraset.com*

# Design of the Efficient Leading Zero Counter

M. Deepika Krishna[1], G. Sowjanya[2], K. K. V. Sri Vihar[3], J. N. Bhanu Prakash[4], K. Bhanu Sandeep[5]
[1]*Assistant Professor,* [2, 3, 4, 5]*B.Tech Students, Department of ECE, Seshadri Rao Gudlavalleru Engineering College, Gudlavalleru*

***Abstract:*** ***A Leading Zero Counter (LZC) is a digital circuit that determines the number of leading zeros in a binary number, and its design considerations include speed, area, and power consumption. The working of a 64-bit LZC is studied in this paper. The functionality of a 64-bit leading zero counter is investigated through a hierarchical design approach. Starting from a basic 2-bit configuration, the counter's underlying mechanics are analysed and subsequently scaled up to accommodate 64-bit operations. In this paper basic gates such as AND gate, OR gate and inverter are used for implementing leading zero counter. Boolean equations are formed from previously proposed architectures and after simplifying that Boolean expression a new architecture for LZC unit is formed. Further calculation can be performed for a 128-bit leading zero counter. Performance and analysis of the 64-bit LZC can be conducted using this method, and the Xilinx Vivado design suite is utilized to simulate and synthesize the 64-bit leading zero counter.***

***Keywords – Leading Zero Counter (LZC), Digital circuits, Boolean Expression.***

## I. INTRODUCTION

A Leading Zero Counter (LZC) is utilized as a digital circuit that determines the number of consecutive zeros at the beginning of a binary sequence, commonly used in computing for efficient data processing and arithmetic operations. In this, a 64-bit operation is performed in the leading zero counter. A 64-bit Leading Zero Counter is designed to count the leading zeros in a 64-bit binary number. This means very large numbers can be handled and it is suitable for applications that require high-precision arithmetic. A 64-bit binary number is taken as input and an output is produced that represents the number of leading zeros in the input number. The 2-bit LZC as the basic block adopts a different way of dealing with the case in which all the input bits are zero.

The importance of Leading Zero Counters in modern computing systems is emphasized by their ability to optimize performance in areas such as floating-point arithmetic, cryptography, and digital signal processing. A critical operation, the Leading Zero Count (LZC), is performed to determine the number of leading zeros in a binary number. The LZC is widely utilized in floating-point normalization, priority encoding, and data compression. To address this need, a hierarchical, hardware-efficient Leading Zero Counter (LZC) architecture is presented, which is capable of handling 64-bit inputs with minimal latency. The 64-bit input is recursively divided into smaller segments (32-bit, 16-bit, 8-bit, 4-bit, and 2-bit) until the count of leading zeros is efficiently determined. Parallel processing is enabled by the hierarchical design.

The 32-bit Leading Zero Counter (LZC) has been improved to 64-bit to enhance its performance, precision, and scalability. Increased range, higher accuracy, and better compatibility with modern 64-bit computing systems are offered by the 64-bit LZC, making it suitable for demanding applications in fields like scientific simulations, data analytics, and cryptography. A 128-bit Leading Zero Counter can be further calculated by being implemented and evaluated using the Xilinx Vivado Design Suite.

## II. LITERATURE SURVEY

A Leading Zero Counter (LZC) is utilized as a vital component in digital systems, where the number of consecutive leading zeros or ones in a binary number is determined, starting from the most significant bit (MSB). An n-bit binary input is processed, and an output consisting of $\log_2(n) + 1$ bits is generated by the LZC. This output includes a flag bit (V) that indicates whether all input bits are zero, and the remaining bits (Z) represent the count of leading zeros. Various methods have been proposed for efficiently determining the leading zero count, with a focus on FPGA-based implementations. These approaches differ in their design methodologies and underlying logic.

For instance, distinct design techniques are employed by the methods described in references [1], [2], and [3]. The method outlined in reference [3] was initially developed for ASIC designs but was later adapted for FPGA implementations. In contrast, different design methodologies for 8-bit LZCs are presented in references [1] and [2], which are then combined in a hierarchical structure to process wider binary values. Hierarchical designs for 32-bit LZCs, as described in references [1], [2], and [3], are presented. References [1] and [3] share a similar structural approach, although distinct logic circuits are used in their respective 8-bit LZC designs.

Variations in hardware characteristics are resulted from this. A distinct logic mechanism is employed by reference [2] compared to references [1] , [3] [5] and [8], both for the 8-bit LZC and the hierarchical construction of larger LZCs. Instead of computing the inverted versions of V and Z, the values are directly derived by the method in reference [2].

Probabilistic methods are integrated into Leading Zero Counter (LZC) design to enhance performance and energy efficiency. The LZC design is optimized through the reduction of transistors and power consumption, while maintaining accuracy, thereby making it suitable for nanoscale computing applications is described in references [4] and [8].

In reference [6] and [9] up to 31% improvement in accuracy is demonstrated by the proposed method compared to standard fixed-point quantization, and a balance between accuracy and energy consumption is achieved for real-world applications, particularly in edge AI, IoT, mobile, and embedded systems.

This optimized design enhances energy efficiency and speed, making it ideal for low-power, high-performance FPGA accelerators. Improved processing speed and reduced power consumption are achieved, enabling efficient acceleration for various applications are described in the reference [7] and [10].

The method outlined in reference [9], In an innovative approximate addition technique is introduced that efficiently exploits FPGA resources, leading to improved accuracy without compromising power consumption or computational speed. A design is proposed that is particularly beneficial for error-resilient applications requiring high performance and energy efficiency, making it suitable for applications such as machine learning, data analytics, and IoT devices.

## III. PROPOSED METHODOLOGY

Implementing a Leading Zero Counter (LZC) involves creating a circuit that detects the number of leading zeros in a binary number. More effective techniques or a combination of logic gates can be used to accomplish this.

The design process starts with a 2-bit LZC, which consists of two inputs (A1, A0) and two outputs (V and Z). Here, ‘V’ functions as the parity bit or overflow indicator, while ‘Z’ represents the count of leading zeros based on different input values. The two output equations are written here:

$V=\overline{A_0}+\overline{A_1}$

$Z=\overline{A_1}\&A_0$

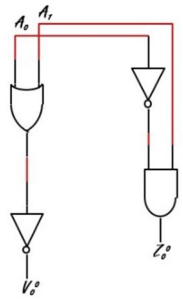


Fig.1: 2-bit leading zero counter

The truth table is written below which is used to check whether our designed LZC able to count the leading zeros for all the cases in it

| Input | | Output | |
|---|---|---|---|
| A1 | A0 | V | Z |
| 0 | 0 | 1 | 10 |
| 0 | 1 | 1 | 01 |
| 1 | 0 | 1 | 00 |
| 1 | 1 | 0 | 00 |

*ISSN: 2321-9653; IC Value: 45.98; SJ Impact Factor: 7.538*
*Volume 13 Issue IV Apr 2025- Available at www.ijraset.com*

And then, a 4-bit LZC is developed with four inputs (A3, A2, A1, A0) and three outputs (V, Z1, Z0). The V output serves as the overflow flag, while Z provides the leading zero count. This module is efficiently structured using two 2-bit LZCs, ensuring an optimized computation process. The output equations are written here:

V= ~((A3 | A2) & (A1 | A0))

Z1=~((A3) | (A2))

Z0=(~((A3) | (A2)) & (~(A3) & (A2))) | ((~((A3) | (A2)) & (~(A3) & (A2)))

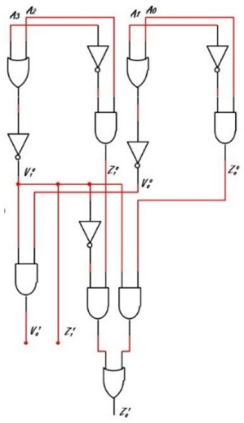


Fig.2: 4-bit leading Zero counter

The logical implementation diagram of a 4-bit leading zero counter is shown in Fig.2. In this, two 2-bit leading counters are used in the 4-bit leading zero counter.

After that, The design is then extended to an 8-bit LZC, which includes eight inputs (A7 to A0) and four outputs (V, Z2, Z1, Z0). It is constructed by integrating two 4-bit LZCs. Here ‘V' is called parity bit or overflow flag and ‘Z' gives the count of leading zeros for different values of inputs.

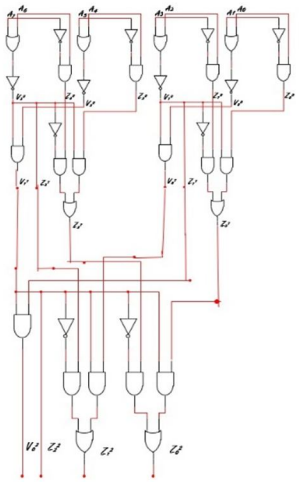


Fig.3: 8-bit leading Zero counter

The logical implementation diagram of a 8-bit leading zero counter is shown in Fig.3.

Next, a 16-bit LZC is designed, incorporating sixteen inputs (A15 to A0) and five outputs (V, Z3, Z2, Z1, Z0). The circuit consists of two 8-bit LZCs, which work together to determine the total number of leading zeros. Here ‘V' is called parity bit or overflow flag and ‘Z' gives the count of leading zeros for different values of inputs.

*ISSN: 2321-9653; IC Value: 45.98; SJ Impact Factor: 7.538*
*Volume 13 Issue IV Apr 2025- Available at www.ijraset.com*

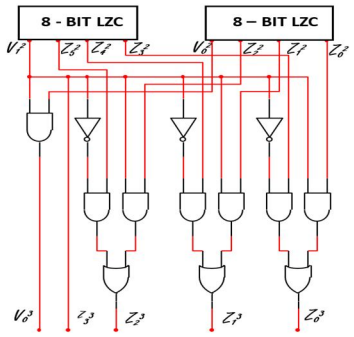



Fig.4: 16-bit leading Zero counter

Then Following this, a 32-bit LZC is implemented with thirty-two inputs (A31 to A0) and six outputs (V, Z4, Z3, Z2, Z1, Z0). This module is constructed by combining two 16-bit LZCs, merging their results to compute the final count with efficiency. Here 'V' is called parity bit or overflow flag and 'Z' gives the count of leading zeros for different values of inputs.

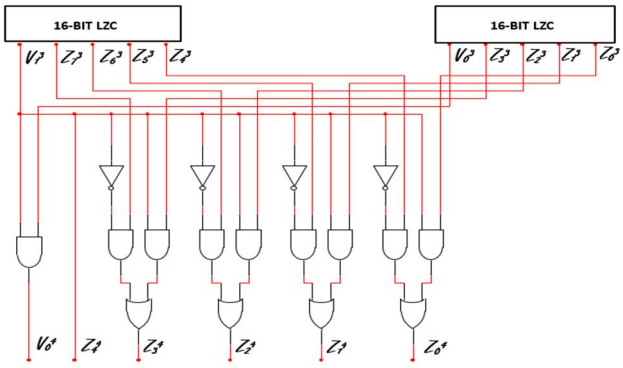



Fig.5: 32-bit leading Zero counter

At the final stage, a 64-bit LZC is developed, featuring sixty-four inputs (A63 to A0) and seven outputs (V, Z5, Z4, Z3, Z2, Z1, Z0). Here 'V' is called parity bit or overflow flag and 'Z' gives the count of leading zeros for different values of inputs.

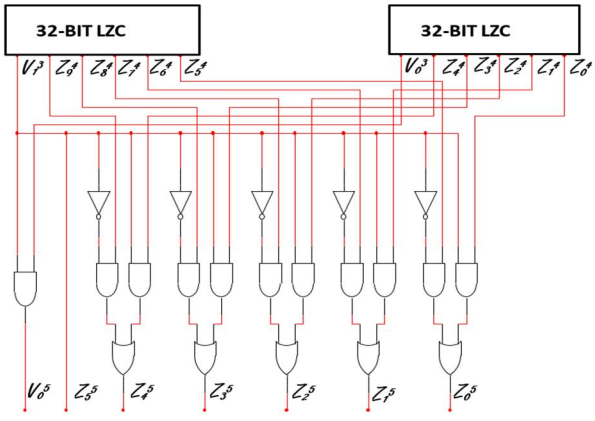



Fig.6: 64-bit leading Zero counter

The logical implementation diagram of a 64-bit leading zero counter is shown in Fig.6. This design consists of two 32-bit LZCs, whose outputs are combined to generate the final leading zero count.

## IV. RESULTS

```
Time = 0, Input = 00(0), Valid = 1, Leading Zero Count = 10
Time = 10000, Input = 01(1), Valid = 1, Leading Zero Count = 01
Time = 20000, Input = 10(2), Valid = 1, Leading Zero Count = 00
Time = 30000, Input = 11(3), Valid = 0, Leading Zero Count = 00
$finish called at time : 40 ns : File "C:/Users/Nithya Bhanu prakash/2bitlzc/2bitlzc.srcs/sim_1/new/lzc_2bit_tb.v" Line 47
INFO: [USF-XSim-96] XSim completed. Design snapshot 'lzc_2bit_tb_behav' loaded.
INFO: [USF-XSim-97] XSim simulation ran for 1000ns
launch_simulation: Time (s): cpu = 00:00:11 ; elapsed = 00:00:15 . Memory (MB): peak = 1016.586 ; gain = 0.000
```

Fig.7: Results of 2-bit leading zero counter

```
Time: 0 | a = 0000( 0) | lzc = 10
Time: 10000 | a = 0001( 1) | lzc = 11
Time: 20000 | a = 0010( 2) | lzc = 10
Time: 30000 | a = 0011( 3) | lzc = 10
Time: 40000 | a = 0100( 4) | lzc = 01
Time: 50000 | a = 0101( 5) | lzc = 01
Time: 60000 | a = 0110( 6) | lzc = 01
Time: 70000 | a = 0111( 7) | lzc = 01
Time: 80000 | a = 1000( 8) | lzc = 00
Time: 90000 | a = 1001( 9) | lzc = 00
Time: 100000 | a = 1010(10) | lzc = 00
Time: 110000 | a = 1011(11) | lzc = 00
Time: 120000 | a = 1100(12) | lzc = 00
Time: 130000 | a = 1101(13) | lzc = 00
Time: 140000 | a = 1110(14) | lzc = 00
Time: 150000 | a = 1111(15) | lzc = 00
INFO: [USF-XSim-96] XSim completed. Design snapshot 'lzc4bit_tb_behav' loaded.
```

Fig.8: Results of 4-bit leading zero counter

Tcl Console × Messages Log

```
# }
# run 1000ns
Time     Input        V   lzc
---------------------------------------
1000     00000000(  0)   1   110
2000     00000001(  1)   0   111
3000     00000010(  2)   0   110
4000     00000011(  3)   0   110
5000     00000100(  4)   0   101
6000     00000101(  5)   0   101
7000     00000110(  6)   0   101
8000     00000111(  7)   0   101
9000     00001000(  8)   0   100
10000    00001001(  9)   0   100
11000    00001010( 10)   0   100
12000    00001011( 11)   0   100
13000    00001100( 12)   0   100
14000    00001101( 13)   0   100
15000    00001110( 14)   0   100
16000    00001111( 15)   0   100
17000    00010000( 16)   0   011
18000    00010001( 17)   0   011
19000    00010010( 18)   0   011
20000    00010011( 19)   0   011
21000    00010100( 20)   0   011
22000    00010101( 21)   0   011
23000    00010110( 22)   0   011
24000    00010111( 23)   0   011
25000    00011000( 24)   0   011
26000    00011001( 25)   0   011
27000    00011010( 26)   0   011
28000    00011011( 27)   0   011
29000    00011100( 28)   0   011
```

Fig.9: Results of 8-bit leading zero counter

```
#    } else {
#       send_msg_id Add_Wave-1 WARNING "No top
#    }
# }
# run 1000ns
Time      a          V    lzc
0     0000000000000000(     0)      1    1110
10    0000000000000001(     1)      0    1111
20    0000000000000010(     2)      0    1110
30    0000000000000011(     3)      0    1110
40    0000000000000100(     4)      0    1101
50    0000000000000101(     5)      0    1101
60    0000000000000110(     6)      0    1101
70    0000000000000111(     7)      0    1101
80    0000000000001000(     8)      0    1100
90    0000000000001001(     9)      0    1100
100   0000000000001010(    10)      0    1100
110   0000000000001011(    11)      0    1100
120   0000000000001100(    12)      0    1100
130   0000000000001101(    13)      0    1100
140   0000000000001110(    14)      0    1100
150   0000000000001111(    15)      0    1100
160   0000000000010000(    16)      0    1011
170   0000000000010001(    17)      0    1011
180   0000000000010010(    18)      0    1011
190   0000000000010011(    19)      0    1011
200   0000000000010100(    20)      0    1011
210   0000000000010101(    21)      0    1011
220   0000000000010110(    22)      0    1011
230   0000000000010111(    23)      0    1011
240   0000000000011000(    24)      0    1011
250   0000000000011001(    25)      0    1011
260   0000000000011010(    26)      0    1011
```

Fig.10: Results of 16-bit leading zero counter

```
# }
# run 1000ns
Time          a                                   v    lzc
0     00000000000000000000000000000000(          0)     1    30
10    11111111111111111111111111111111(4294967295)     0     0
20    10000000000000000000000000000000(2147483648)     0     0
30    01000000000000000000000000000000(1073741824)     0     1
40    00100000000000000000000000000000( 536870912)     0     2
50    00000000000000000000000000000001(          1)     0    31
60    00000000000000000000000000000010(          2)     0    30
70    00000000000000000000000000000100(          4)     0    29
80    10101010101010101010101010101010(2863311530)     0     0
90    01010101010101010101010101010101(1431655765)     0     1
100   00000000000000000000000011111111(        255)     0    24
110   00000000000000001111111111111111(      65535)     0    16
120   00000000111111111111111111111111(   16777215)     0     8
130   11111111000000000000000000000000(4278190080)     0     0
140   11111111111111110000000000000000(4294901760)     0     0
150   11111111111111111111111100000000(4294967040)     0     0
160   00000000000000000000000000001000(          8)     0    28
170   00000000000000000000000100000000(        256)     0    23
180   00000000000000001000000000000000(      32768)     0    16
190   00000000001000000000000000000000(    2097152)     0    10
200   00000000000000001010101010101010(      43690)     0    16
210   11000000000000000000000000000011(3221225475)     0     0
220   11110000000011110000000011110000(4027515120)     0     0
230   00001111000011110000111100001111( 252645135)     0     4
240   00000000000000000000000000001111(         15)     0    28
250   00000000000000000000111111111111(       4095)     0    20
260   11111111111111111111111111110000(4294967280)     0     0
270   00000000000000000000000011110000(        240)     0    24
280   00000000000000001111111111111111(      65535)     0    16
290   11111111111111110000000000000000(4294901760)     0     0
```

Fig.11: Results of 32-bit leading zero counter

```
Test Case 0: a = 0000000000000000 (                   0), V = 1, lzc = 111110
Test Case 1: a = 8000000000000000 ( 9223372036854775808), V = 0, lzc = 000000
Test Case 2: a = ffffffff00000000 (18446744069414584320), V = 0, lzc = 000000
Test Case 3: a = 00000000ffffffff (          4294967295), V = 0, lzc = 100000
Test Case 4: a = 0000ff0000000000 (     280375465082880), V = 0, lzc = 010000
Test Case 5: a = 1234567890abcdef ( 1311768467294899695), V = 0, lzc = 000011
Test Case 6: a = ffffffffffffffff (18446744073709551615), V = 0, lzc = 000000
Test Case 7: a = 00000000000000ff (                 255), V = 0, lzc = 111000
Test Case 8: a = 0000000000ffff00 (            16776960), V = 0, lzc = 101000
Test Case 9: a = 0000000000000001 (                   1), V = 0, lzc = 111111
Test Case 10: a = 000000000000000f (                  15), V = 0, lzc = 111100
Test Case 11: a = abababababababab (12370169555311111083), V = 0, lzc = 000000
Test Case 12: a = 7777777777777777 ( 8608480567731124087), V = 0, lzc = 000001
Test Case 13: a = ff00000000000000 (18374686479671623680), V = 0, lzc = 000000
Test Case 14: a = 0101010101010101 (   72340172838076673), V = 0, lzc = 000111
Test Case 15: a = 9876543210abcdef (10986060915021696495), V = 0, lzc = 000000
Test Case 16: a = 00000000ffffff00 (          4294967040), V = 0, lzc = 100000
Test Case 17: a = 00000000000000ff (                 255), V = 0, lzc = 111000
Test Case 18: a = e4e4e4e4e4e4e4e4 (16493559407081481444), V = 0, lzc = 000000
Test Case 19: a = 0000000000000700 (                1792), V = 0, lzc = 110101
$finish called at time : 200 ns : File "C:/New folder/64bit/64bit.srcs/sources_1
INFO: [USF-XSim-96] XSim completed. Design snapshot 'bit64_tb_behav' loaded.
INFO: [USF-XSim-97] XSim simulation ran for 1000ns
launch_simulation: Time (s): cpu = 00:00:03 ; elapsed = 00:00:11 . Memory (MB):
```

Fig.12: Results of 64-bit leading zero counter

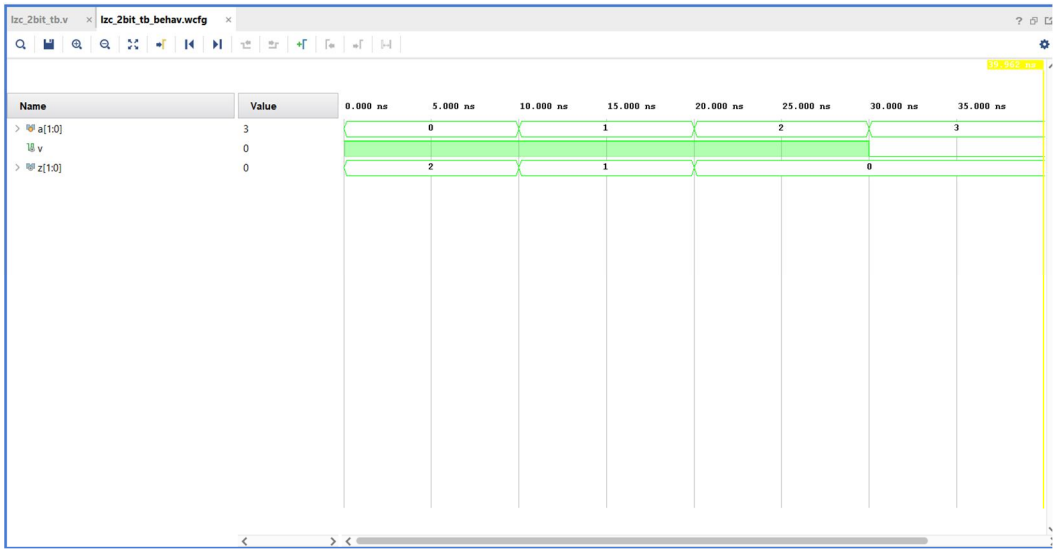



Fig.13: Waveforms for a 2-bit leading zero counter consists of two inputs (A1, A0) and two outputs (V and Z).

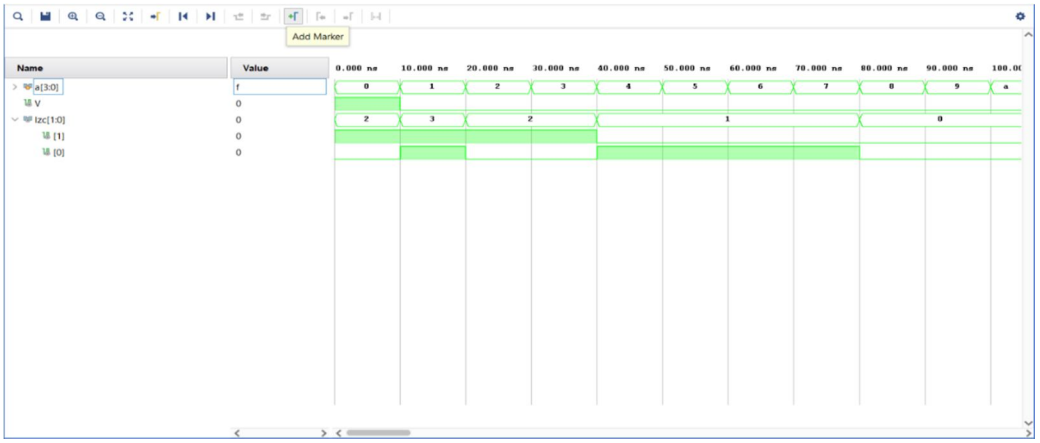



Fig.14: Waveforms for the 4-bit leading zero counter with four inputs (A3, A2, A1, A0) and three outputs (V, Z1, Z0).

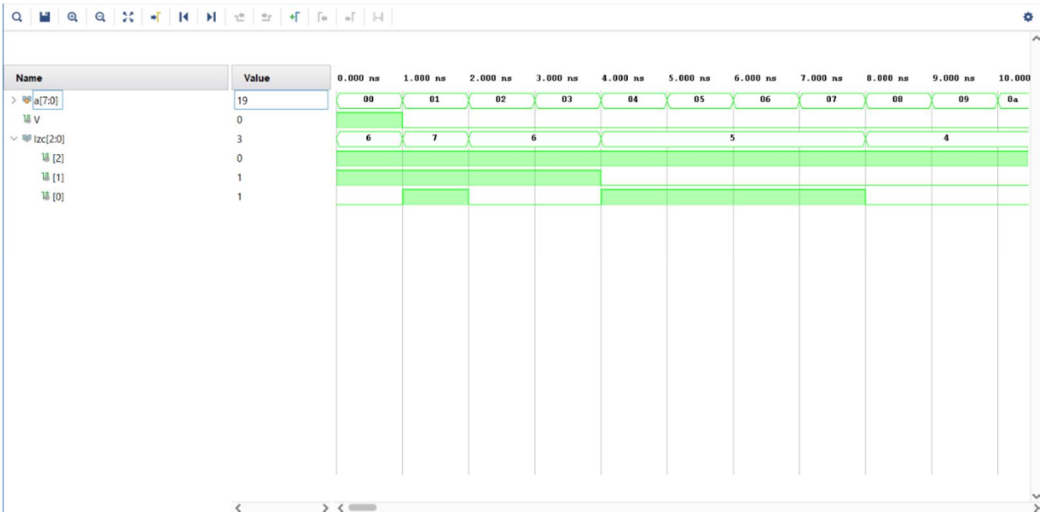



Fig.15: Waveforms for the 8-bit leading zero counter which is having 8 inputs (A7 to A0) and that produced 4 outputs such as V and Z0, Z1 & Z2.

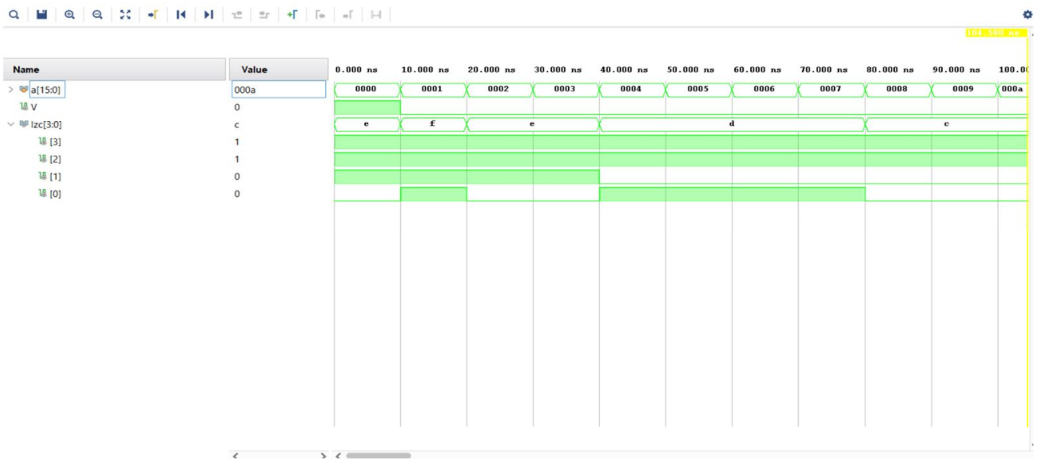



Fig.16: Waveforms for the 16-bit leading zero counter which is having 16 inputs (A15 to A0) and that produced 5 outputs such as V and Z0, Z1, Z2 & Z3.

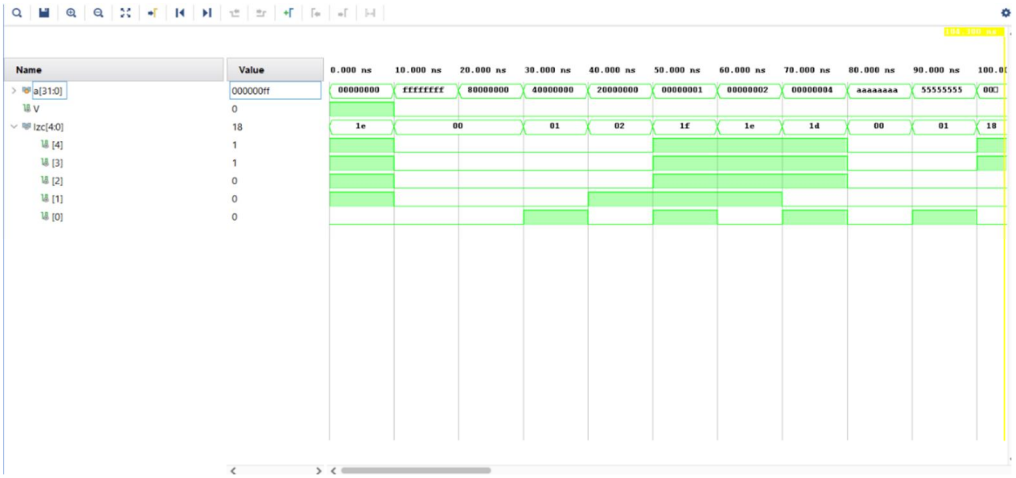



Fig.17: Waveforms for the 32-bit leading zero counter which is having 32 inputs (A31 to A0) and that produced 6 outputs such as V and Z0, Z1, Z2, Z3 & Z4.

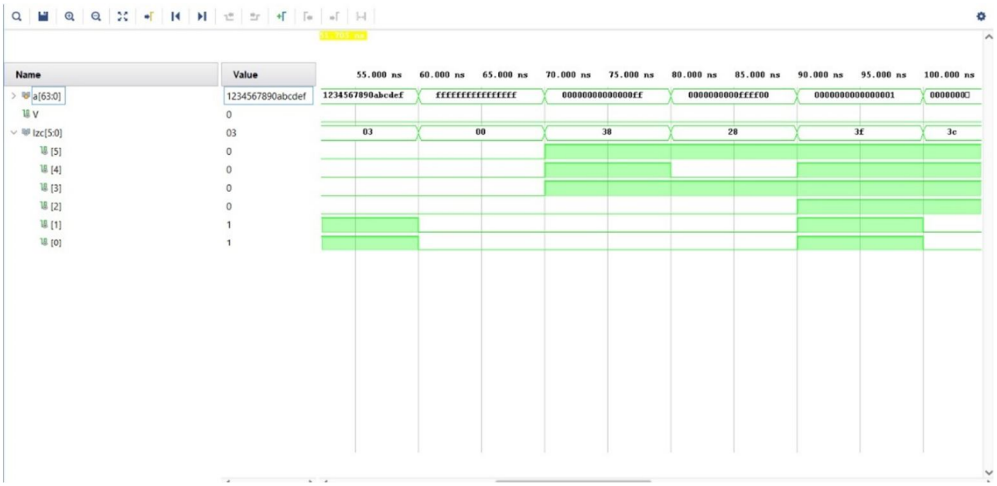



Fig.18: Waveforms for the 64-bit leading zero counter which is having 64 inputs (A63 to A0) and that produced 7 outputs such as V and Z0, Z1, Z2, Z3, Z4 & Z5.

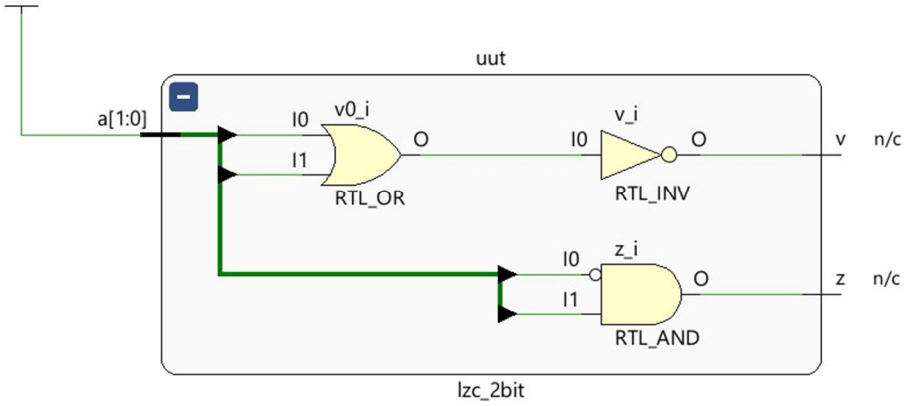



Fig.19: RTL Schematic diagram of 2-bit leading zero counter

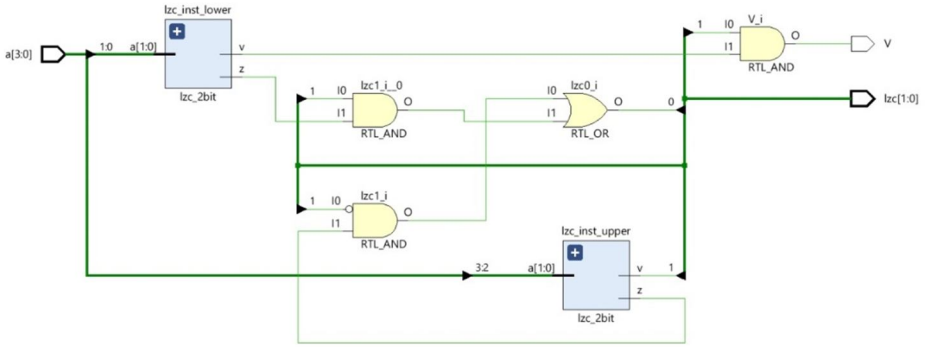



Fig.20: RTL Schematic diagram of 4-bit leading zero counter

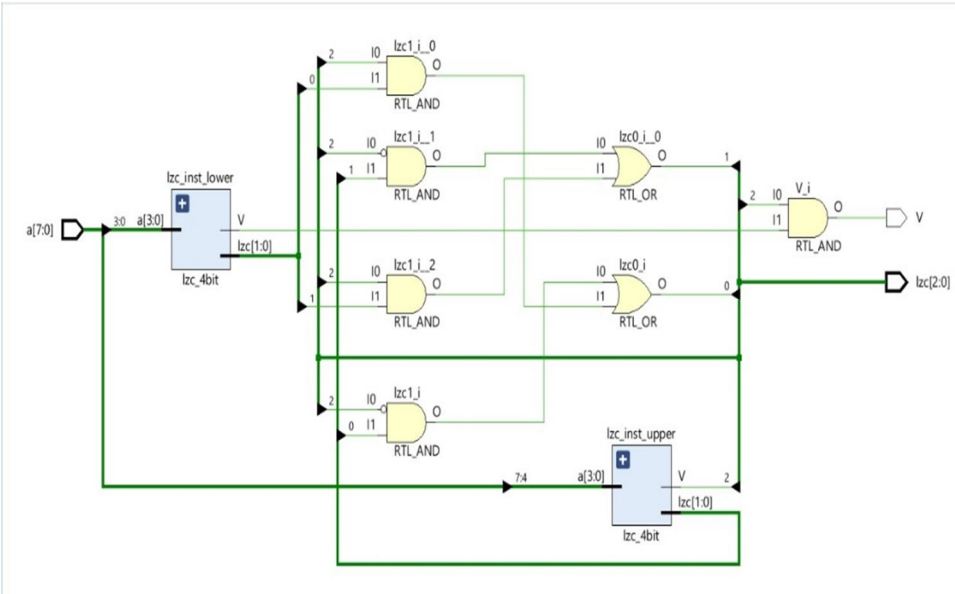



Fig.21: RTL Schematic diagram of 8-bit leading zero counter

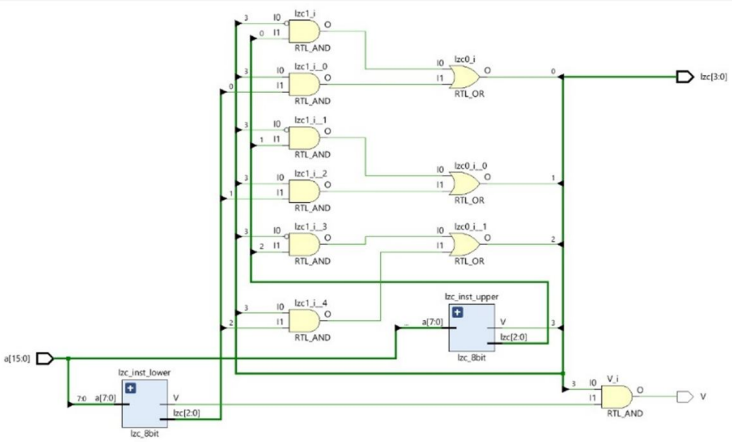



Fig.22: RTL Schematic diagram of 16-bit leading zero counter

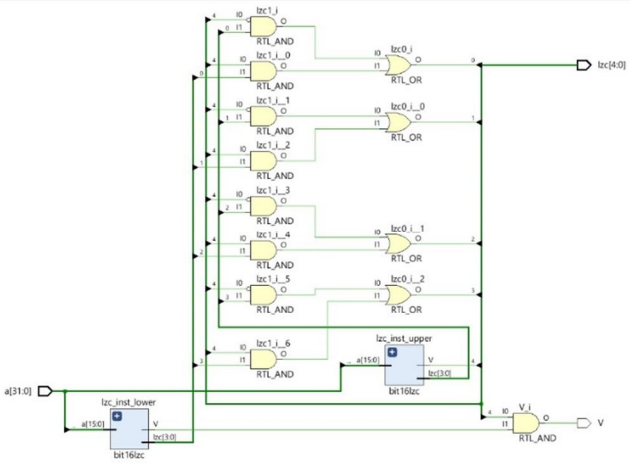



Fig.23: RTL Schematic diagram of 32-bit leading zero counter

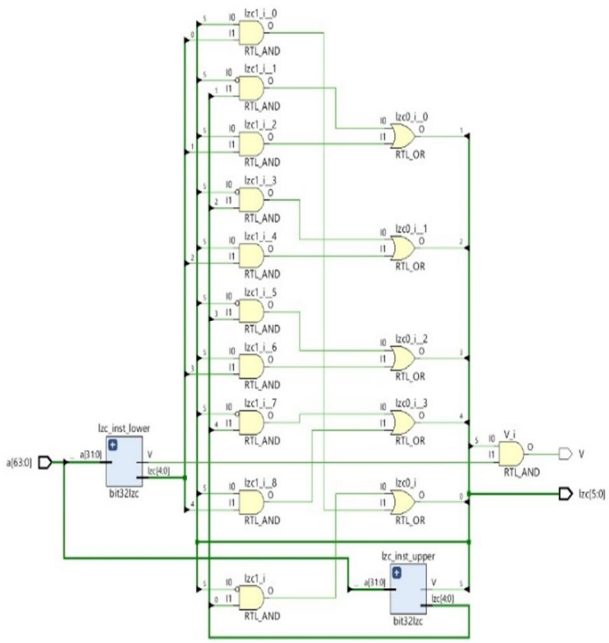



Fig.24: RTL Schematic diagram of 64-bit leading zero counter

Table I
Comparative Analysis of Resource Utilization, LUT's, Slices's, Power in mW, Delay, and PADP for Various Bit-Width Configurations of the Proposed Leading Zero Counter Design

| | Bits | [1] | [3] | [11] | [12] | [2] | proposed |
|---|---|---|---|---|---|---|---|
| LUTs | 8 | 6 | 5 | - | 5 | 4 | 3 |
| | 16 | 16 | 14 | - | 14 | 10 | 10 |
| | 32 | 39 | 33 | 36 | 36 | 26 | 24 |
| | 64 | 75 | 71 | 71 | 73 | 60 | 47 |
| SLICEs | 8 | 2 | 2 | - | 2 | 1 | 1 |
| | 16 | 6 | 5 | - | 5 | 4 | 4 |
| | 32 | 16 | 12 | 12 | 12 | 9 | |
| | 64 | 29 | 28 | 28 | 26 | 24 | 18 |
| POWER | 8 | 4.41 | 4.32 | - | 3.41 | 3.83 | 2.81 |
| (in mw) | 16 | 6.36 | 6.02 | - | 6.49 | 5.03 | 3.25 |
| | 32 | 6.83 | 6.77 | 7.063 | 6.77 | 6.36 | 3.83 |
| | 64 | 8.84 | 8.04 | 7.71 | 8.17 | 7.45 | 5.29 |
| DELAY | 8 | 1.92 | 1.62 | - | 2.2 | 1.87 | 1.52 |
| | 16 | 1.98 | 2.06 | - | 2.38 | 2.71 | 2.96 |
| | 32 | 2.76 | 2.84 | 2.94 | 2.92 | 3.03 | 3.10 |
| | 64 | 3.52 | 3.40 | 3.70 | 3.66 | 3.83 | 3.72 |
| PDAP | 8 | 50.8 | 34.99 | - | 37.51 | 28.87 | 21.35 |
| | 16 | 201.48 | 173.62 | - | 216.25 | 136.31 | 192.40 |
| | 32 | 735.18 | 634.48 | 747.55 | 711.66 | 501.04 | 710.37 |
| | 64 | 2333.76 | 1940.86 | 2025.41 | 2182.86 | 1712.01 | 1967.88 |

## V. CONCLUSION

In this paper, a 64-bit Leading Zero Counter (LZC) was designed using a hierarchical approach, starting from a 2-bit configuration and expanding to handle 64-bit numbers. Basic gates such as AND, OR, and inverters were used to form and simplify Boolean expressions, leading to an optimized LZC architecture. The design was synthesized and analyzed using Xilinx Vivado, where reports on area, power, and timing were obtained, highlighting the trade-offs in resource utilization and performance.

This design serves as a foundation for extending to larger bit-width counters, like 128-bit LZC, and can be applied in fields such as digital signal processing, high-performance computing, and cryptography, where efficient leading zero detection is essential. Future work could focus on further power optimization and integration into larger systems for more complex applications.

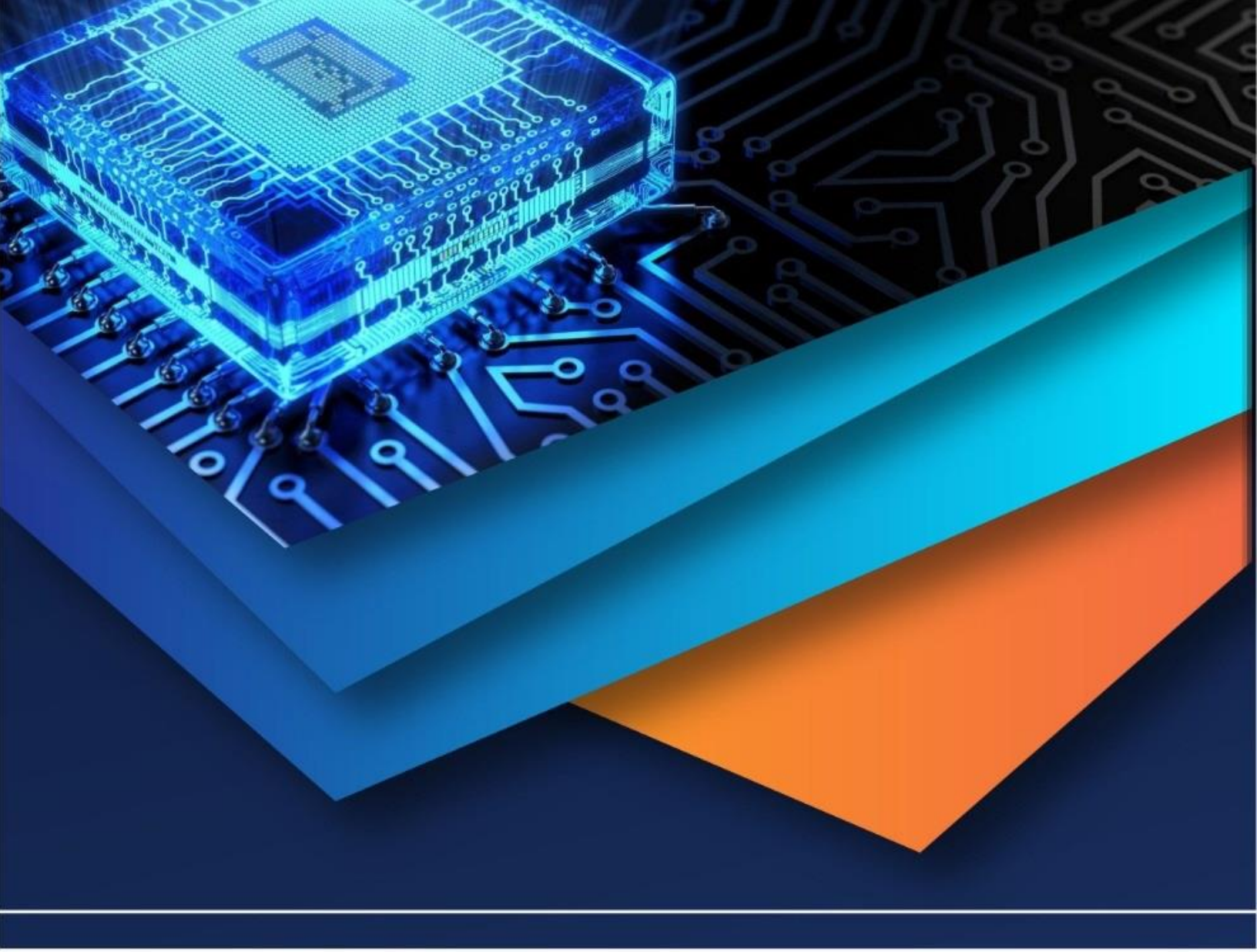
## VI. ACKNOWLEDGEMENT

We would like to extend our gratitude to the faculty-in- charge of Department of Electronics and Communication Engineering of Seshadri Rao Gudlavalleru college, family and friends for providing necessary facilities and support in completing this paper.

## REFERENCES

[1] J. Miao and S. Li, "A design for high speed leading-zero counter," inProc. ISCE, Kuala Lumpur, Malaysia, 2017, pp. 22–23.
[2] A. Zahir, A. Ullah, P. Reviriego, and S. R. U. Hassnain, "Efficient leading zero count (LZC) implementations for Xilinx FPGAs," IEEEEmbedded Syst. Lett., vol. 14, no. 1, pp. 35–38, Mar. 2022.
[3] G. Dimitrakopoulos, K. Galanopoulos, C. Mavrokefalidis, and D. Nikolos, "Low-power leading-zero counting and anticipation logic for high-speed floating point units," IEEE Trans. Very Large Scale Integrate. (VLSI) Syst., vol. 16, no. 7, pp. 837–850, Jul. 2008.
[4] S. Liu, P. Reviriego, P. Junsangsri, and F. Lombardi, "Probabilisticapproximate computing at nanoscales," IEEE Nanotech. Mag., vol. 16, no. 1, pp. 16–24, Feb. 2022.
[5] F. G. Zacchigna, "Methodology for CNN implementation in FPGA based embedded systems," IEEE Embedded Syst. Lett., early access, Jun. 29, 2022, doi: 10.1109/LES.2022.3187382.
[6] H. F. Langorudi, V. Karia, T. Pandit, and D. Kudithipudi, "TENT:Efficient quantization of neural networks on the tiny edge with taperedfixed PoiNT," 2021, arXiv:2104.02233.
[7] S. Ullah, T. D. A. Nguyen, and A. Kumar, "Energy-efficient lowlatency signed multiplier for FPGA-based hardware accelerators," IEEEEmbedded Syst. Lett., vol. 13, no. 2, pp. 41–44, Jun. 2021.

[8] H. Zhang, H. J. Lee, and S.-B. Ko, "Efficient fixed/floating-point mergedmixed-precision multiply-accumulate unit for deep learning processors,"in Proc. ISCAS, Florence, Italy, 2018, pp. 1–5.
[9] S. Perri, F. Frustaci, F. Spagnolo, and P. Corsonello, "Efficient approximate adders for FPGA-based data-paths," Electronics, vol. 9, no. 9,pp. 1–19, Sep. 2020.
[10] S. Perri, F. Frustaci, F. Spagnolo, and P. Corsonello, "Design of realtime FPGA-based embedded system for stereo vision," in Proc. ISCAS,Florence, Italy, 2018, pp. 1–5.
[11] N.Z. Milenkovic, V. V. Stankovi ´ c, M.L. Mili ´ c. "Modular Design Of Fast Leading Zeros Counting Circuit" in Journal of Electrical Engineering, 2015 Nov 1,66(6):329-33.
[12] Xilinx, "Vivado Design Suite User Guide: High Level Synthesis", UG902 (v2017.4) February 2, 2018.



10.22214/IJRASET

45.98

IMPACT FACTOR: 7.129

IMPACT FACTOR: 7.429

# INTERNATIONAL JOURNAL FOR RESEARCH

IN APPLIED SCIENCE & ENGINEERING TECHNOLOGY