Journal of VLSI Circuits and Systems, ISSN: 2582-1458 Vol. 7, No. 2, 2025 (pp. 1-8)
WWW.VLSIJOURNAL.COM

Unified Multimodal 64-Bit Arithmetic Logic Unit for High-Performance Computing Architectures

Shilpi Birla[1], Neha Singh[1]*, Jeevani G.S.N.[1], Renu Kumawat[2], Avireni Srinivasulu[3]

[1]*Department of Electronics and Communication Engineering, Manipal University Jaipur, Jaipur, Rajasthan, India.*
[2]*Department of IoT and Intelligent Systems, Manipal University Jaipur, Jaipur, Rajasthan, India.*
[3]*Department of Electronics and Communication Engineering, Mohan Babu University, Tirupati, Andhra Pradesh, India.*

KEYWORDS:
Unified ALU;
Parallel processing;
Floating-point arithmetic;
Fixed-point arithmetic;
Machine learning hardware;
Cryptographic computations

ARTICLE HISTORY:
Received: 06.10.2025
Revised: 28.10.2025
Accepted: 05.11.2025

DOI:
https://doi.org/10.31838/jvcs/07.02.01

Abstract

An arithmetic logic unit (ALU) is the core component for any processing unit to perform arithmetical and logical operations for modern computing. The design for ALUs for specific tasks, including integer and floating-point arithmetic, logical operations, data movement, and control functions, influences the CPU architecture and digital system design. This work presents a unified ALU implemented in Verilog hardware description language (HDL), capable of performing arithmetic and logic operations across diverse numerical representations. The ALU is engineered to integrate logic unit, signed arithmetic processor, unsigned arithmetic processor, and floating-point arithmetic processor. The selection of operation is governed by select line signals to facilitate versatile user-driven functionality at the hardware level. To enhance the computational efficiency for handling multiplication of 64-bit signed and unsigned operands which generates a 128-bit result, the proposed ALU architecture employs parallelism by processing the most significant and least significant bits simultaneously. A flexible mechanism for selective output enables the user to extract the desired segment of the result. By consolidating floating-point and fixed-point computations within a single ALU instance, the proposed architecture reduces the silicon area with reduced power dissipation and computational latency while streamlining routing complexities. This multimodal ALU design with high throughput is particularly suited for deployment in heterogeneous computing environments such as general-purpose processors, cryptographic accelerators, and machine intelligence hardware, where the rapid processing of heterogeneous data types is essential for workload optimization and energy-efficient system operation.

Authors' e-mail: shilpi.birla@jaipur.manipal.edu; neha.singh@jaipur.manipal.edu; g.229202080@muj.manipal.edu; renu.kumawat@jaipur.manipal.edu; avireni@ieee.org

Authors' ORCID IDs: 0000-0002-4239-4912; 0000-0001-6492-4278; 0009-0001-2755-1115; 0000-0002-3938-9485; 0000-0001-6254-7990

How to cite this article: Shilpi Birla et al. Energy-Aware Physical Synthesis of Deep Neural Networks for Edge-AI Applications in Robotics and VLSI Systems, Journal of VLSI Circuits and System, Vol. 7, No. 2, 2025 (pp. 1-8).

Introduction

An arithmetic logic unit (ALU) is an indispensable component of modern computing systems, as it serves as the fundamental processing unit for arithmetic and logic operations. It enables the central processing unit (CPU) to execute arithmetic operations like addition, subtraction, multiplication, and division along with logical operations like AND, OR, XOR, and XNOR. ALU for fixed-point operations offer power-efficient and faster computation with efficient hardware utilization for resource-constrained environments like IoT devices, consumer electronics, and automotive control systems. However, to support numerical accuracy for scientific computations and complex mathematical operations for IoT systems and high-performance computing,[1] ALUs are designed to precisely handle real numbers with fractional components. Moreover, modern computational needs for machine learning implementation systems require to consolidate fixed-point and floating-point operations for handling broad spectrum of computational tasks with multiple data types. For example, hardware accelerators for artificial intelligence-based systems work with floating-point numbers at the training stage for high precision and use integers during the inference stage[2] General-purpose ALUs are extended to support floating-point operations and signed integer operations by adding a specific hardware block or by using software routines to break them into integers. This consumes longer processing time with increased power and area requirement as well as routing complexity. To address the demand for diverse computational workloads for versatile, high-performance computing systems, a unified ALU is needed to handle signed and unsigned integers with floating-point arithmetic operations and logical operations. Having a unified block is also advantageous in the perspective of compactness, efficient utilization of resources, and flexibility, and simplifies the control logic as it need not send instructions to multiple units. This work proposes a unified multimodal 64-bit ALU for high throughput, which employs parallelism to achieve faster speed while providing flexibility to the user for selecting the operational needs and output. The multiplier generates a 128-bit output, which is processed in two segments catering to most significant and least significant bits, to allow the user to choose the desired part of the resultant multiplication to be forwarded to the next phase.

Literature Review

Conventional ALU designs focused on unsigned integer arithmetic, while signed integer and floating-point operations were often implemented with separate hardware blocks or software routines. This results in increased power consumption and latency with additional hardware expense. Floating-point unit (FPU) with IEEE 754 standards[3] for 64-bit ALUs using hardware descriptive languages, incorporates modules for addition, subtraction, multiplication, and division using pipelining techniques to improve throughput while managing hardware complexity, as reported in references.[4,5] The authors in reference[5] suggested to use a converter for converting decimal numbers to IEEE 754 format and vice versa at the input and output, respectively, of the floating-point ALU to enable the user to provide decimal inputs to the ALU.

A floating-point arithmetic unit with IEEE 754 standard is presented in references[6,7] to reduce power consumption by employing reversible logic techniques. The floating-point operations such as addition, normalization, and rounding is performed using reversible technology. Another energy-efficient FPU is discussed in reference,[8] focusing on optimizing throughput within area and power constraints. This work achieves high floating-point operation density with the help of architectural techniques such as lower supply voltage, shallower pipelines, and relaxed gate sizing.

A high-performance floating-point arithmetic implementation on field programmable gate array (FPGA) is presented in references,[9,10] which focuses on optimizing speed and resource utilization. Single and double precision FPUs employed pipelining for improved computation throughput and efficient utilization of FPGA resources through architectural enhancements. The work highlights trade-offs between latency, area, and performance in FPGA-based floating-point arithmetic designs. FPGA implementation of a single precision floating-point ALU is presented in references,[11,12] which is suggested to be extended for approximate computing applications.

A 32-bit floating-point ALU is presented in reference[13] with the ability to manage rounding, normalization, underflow, overflow, not-a-number condition, and handling infinity values for precise numerical calculations. Other 32-bit and 64-bit floating-point ALUs are presented in references,[14-16] respectively, each of which employs Vedic mathematical principles for implementing arithmetic operations for reduced latency.

Based on the study, this work aims to design a flexible multimodal unified ALU that offers signed and unsigned fixed-point as well as floating-point arithmetic and logical operations into a single module, thereby reducing chip area, routing complexity, power consumption, and latency. Overall, the progression toward unified 64-bit ALUs capable of handling multiple arithmetic data types in one module is supported by recent research. These advancements promise enhanced performance and efficiency necessary for next-generation computing systems.

Design Methodology

The proposed design follows a bottom-up methodology by designing independent segments for signed ALU, unsigned ALU, and floating-point ALU using behavioral modelling in Verilog hardware design language (HDL). Each module is individually simulated and functionally verified using Verilog testbench. Later, all the design blocks are integrated into unified ALU through the structural coding style. There are four modules that are discussed in this section.

64-Bit Unsigned ALU

Unsigned ALU is the traditional ALU design block, which deals with unsigned integers that are represented as unsigned binary bits at the hardware level. It consists of two units, Arithmetic unit, which performs unsigned addition, subtraction, multiplication, division, and a logical unit, which performs logical AND, OR, XOR, and XNOR operations, as given in Figure 1. A select line (sel) is used to select the required unit to be operated. While all the Arithmetic operations of the 64-bit unsigned ALU provides the 64-bit output, multiplying two 64 bits result in 128-bit output, and most of the traditional ALU designs either store MSB or LSB of result causing loss of data. In the proposed design, the unsigned ALU uses parallel approach to multiply LSBs of two operands and MSBs of two operands simultaneously and store the results in two individual registers. This parallel approach provides the flexibility to either display LSB (least 64 bits) or MSB (most 64 bits) based on designer's requirement using a select line (M). Using parallel processing will store the complete resultant (128 bits) of multiplication resolving the issue of loss of data without increasing the output width. The simulation output for the 64-bit unsigned multiplication using the unsigned ALU segment is shown in Figure 2.

64-Bit Signed ALU

The 64-bit signed ALU proposed in this paper is responsible for performing arithmetic operations like addition, subtraction, multiplication, and division for unsigned numbers. The signed ALU takes the negative binary numbers and converts the operands into 2's complement form before performing any Arithmetic operations. Like unsigned ALU, the signed ALU also follows parallel multiplication of MSB and LSB both individually and simultaneously, and provides flexibility to the designer for selecting the most significant or least significant bits of the output for display. The signed ALU uses the traditional 2's compliment method to perform addition and subtraction, as shown in Figure 3. The sign bit is taken care of during all arithmetic operations. The output waveform for performing division using the proposed 64-bit signed ALU is shown in Figure 4.

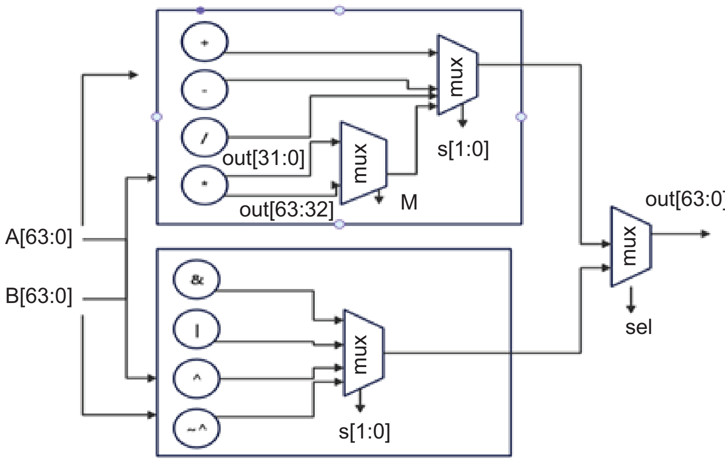
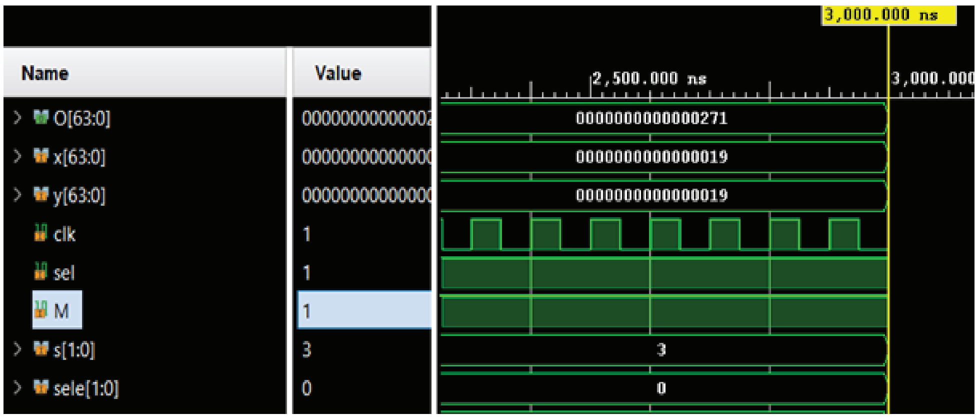


Fig. 1: 64-bit unsigned ALU block diagram.



Name	Value	
O[63:0]	000000000000002	0000000000000271
x[63:0]	0000000000000	0000000000000019
y[63:0]	0000000000000	0000000000000019
clk	1	
sel	1	
M	1	
s[1:0]	3	3
sele[1:0]	0	0

Fig. 2: Output waveform for 64-bit unsigned multiplication using unsigned ALU

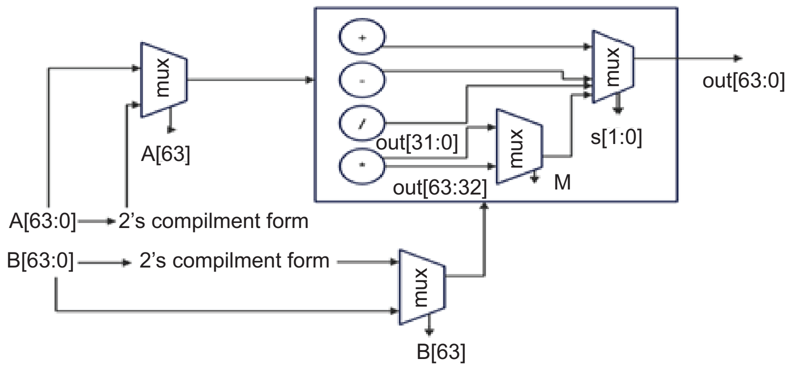
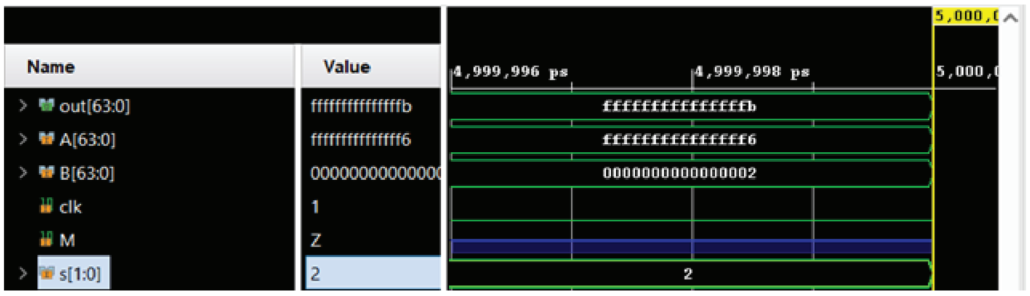


Fig. 3: Signed ALU block diagram.



Name	Value
out[63:0]	fffffffffffffffb
A[63:0]	fffffffffffffff6
B[63:0]	0000000000000(
clk	1
M	Z
s[1:0]	2

Fig. 4: Simulation output for division using 64-bit signed ALU.

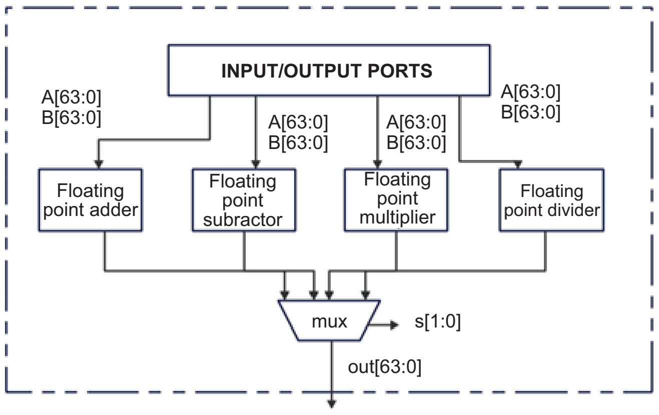


Fig. 5. Block diagram for floating-point ALU.

64-Bit Floating-point ALU

The 64-bit floating-point ALU is a specialized execution unit, which can perform computations on real numbers that are represented in the IEEE754 format. The floating-point ALU, as proposed in this paper, following the IEEE-754 double precision format is shown in Figure 5. Floating-point ALU is designed using the structural approach in Verilog HDL, where the floating-point adder, multiplier, and divisor are designed individually through behavioral modelling and explicitly called inside the floating-point ALU module. The blocks that are used in this module are as discussed here.

Double precision floating-point adder/subtractor

Double precision floating-point adder is a block, which can perform addition and subtraction on real numbers that are represented in the IEEE-754 double precision format, as discussed in reference[17]; therefore, the bias would be 1024 bits. Based on the sign bit of the number, the floating-point adder can selectively act as adder or subtractor. In the floating-point adder/subtractor, the exponents are first aligned, and based on that, the mantissa of the smallest operand is adjusted, finally rounding off the result. The output for subtraction operation performed using the proposed floating-point ALU is shown in Figure 6.

Double precision floating-point multiplier

Double precision floating-point multiplier is designed to perform multiplication on floating-point numbers that are represented using the 64-bit IEEE format. To perform multiplication of two real numbers, this floating-point multiplier directly multiplies the mantissa and adds the exponent. The sign will be determined through XOR operation performed on the sign bit of both operands. The double precision floating-point multipliers are known for their fast and accurate multiplications popularly used in digital signal processing applications.

Double precision floating-point divider

As the name suggests, the double precision floating-point divider handles the division of two 64-bit floating-point numbers. It divides the mantissa but performs subtraction on the exponent bits. The floating-point division is used in applications that involve scaling like fast Fourier transforms (FFT). Figure 5 shows the block diagram of the floating-point ALU.

Top module—Unified ALU

The top module is responsible for integrating all the design blocks, namely, signed ALU, unsigned ALU, and floating-point ALU. The unified ALU implemented on the Spartan 7 FPGA board is shown in Figure 7. The top module uses explicit calling for the subblocks. The 64-bit floating-point ALU consists of three submodules, namely, f_add (used for both addition and subtraction), f_mul, and f_div. The hierarchy is structured such that all special cases are tested in the main module. For example, if either input has all exponent bits as one and a non-zero mantissa, the result is immediately set to a NaN and an invalid_op flag is raised. If both operands are 0 or infinity, then the invalid_op is raised and the result will be NAN. Table 1 compiles the type of operation performed by unified ALU according to the select line signals.

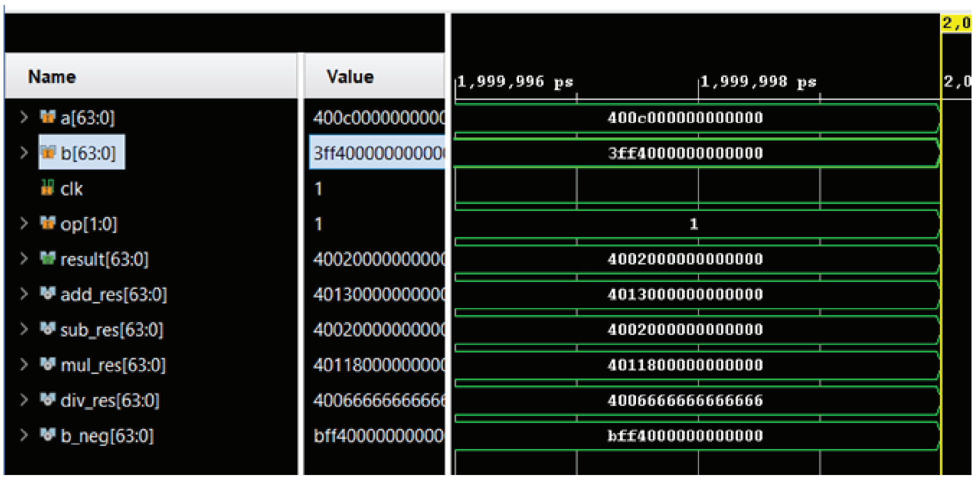

```
Name            Value               1,999,996 ps        1,999,998 ps        2,0
a[63:0]         400c0000000000      400c000000000000
b[63:0]         3ff40000000000      3ff4000000000000
clk             1
op[1:0]         1                   1
result[63:0]    40020000000000      4002000000000000
add_res[63:0]   40130000000000      4013000000000000
sub_res[63:0]   40020000000000      4002000000000000
mul_res[63:0]   40118000000000      4011800000000000
div_res[63:0]   4006666666666       4006666666666666
b_neg[63:0]     bff40000000000      bff4000000000000
```

Fig. 6. Output waveform for subtraction using floating-point ALU.

Table 1. Operations performed by unified ALU.

Select lines						Operations
select		S		sel	M	
0	0	0	0	0	-	Logical AND
0	0	0	1	0	-	Logical OR
0	0	1	0	0	-	Logical XOR
0	0	1	1	0	-	Logical XNOR
0	0	0	0	1	-	Unsigned addition
0	0	0	1	1	-	Unsigned subtraction
0	0	1	0	1	-	Unsigned division
0	0	1	1	1	0	Unsigned multiplication with MSB output
0	0	1	1	1	1	Unsigned multiplication with LSB output
0	1	0	0	-	-	Floating-point addition
0	1	0	1	-	-	Floating-point subtraction
0	1	1	0	-	-	Floating-point multiplication
0	1	1	1	-	-	Floating-point division
1	0	0	0	-	-	Signed addition
1	0	0	1	-	-	Signed subtraction
1	0	1	0	-	-	Signed division
1	0	1	1	-	0	Signed multiplication with MSB output
1	0	1	1	-	1	Signed multiplication with LSB output

Results

Each subunit of an unified ALU is functionally verified using individual Verilog testbenches. After the integration of all design blocks in a structural manner under the top module of the unified ALU, the design is simulated using Xilinx Vivado and verified functionally, as shown in Figures 8–10, with the help of waveforms for logical XOR operation, floating-point division, and addition of two signed numbers, respectively. The area utilization in terms of core configurable hardware resources on the Spartan 7 FPGA board, for the unified ALU presented in this paper, is compiled in Table 2. Table 3 shows the resource utilization by unsigned ALU, signed ALU, and floating-point ALU in comparison to the proposed unified

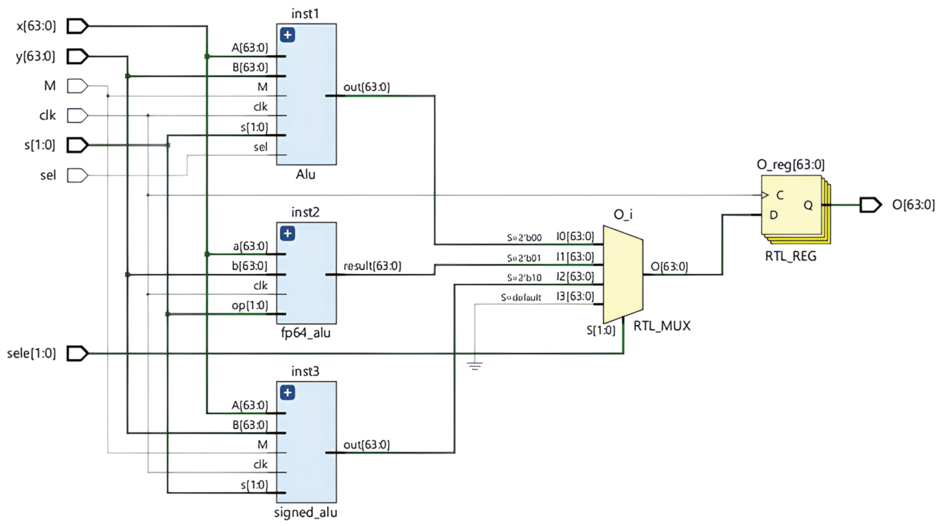
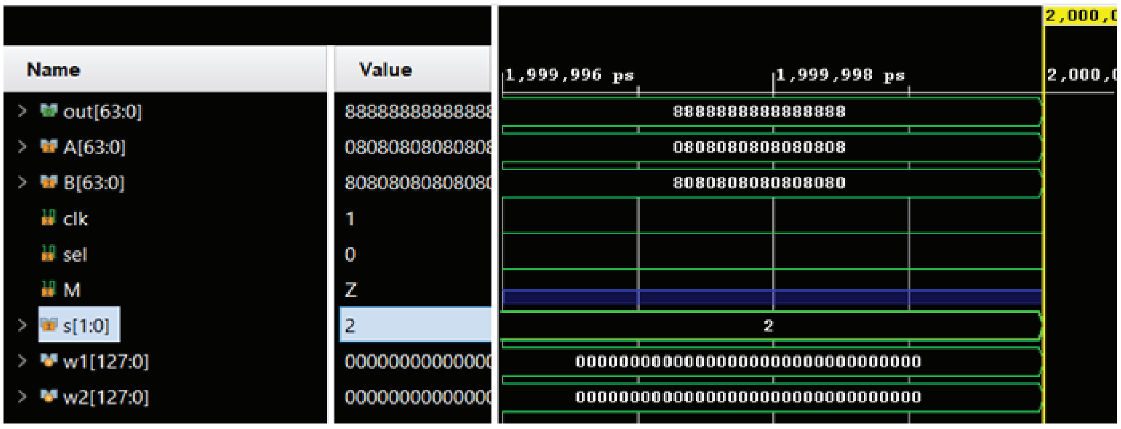
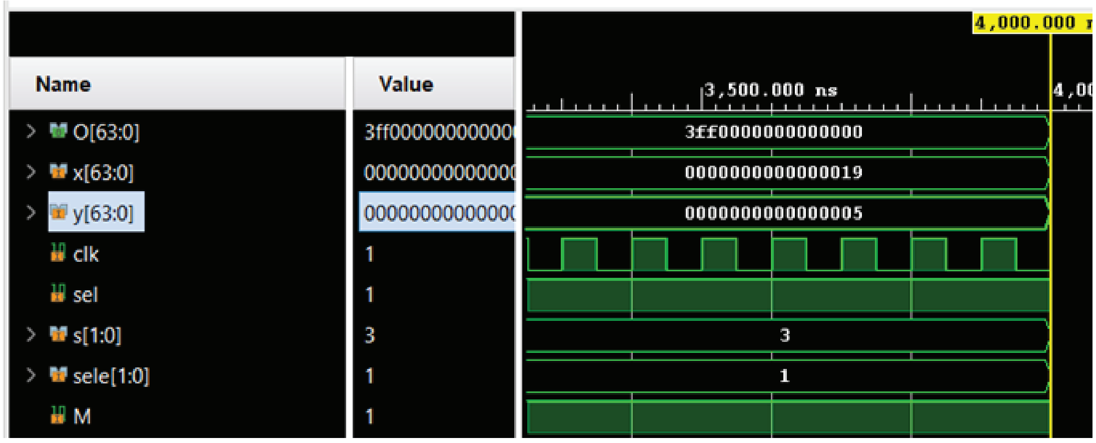


Fig. 7: Implementation of an unified ALU.



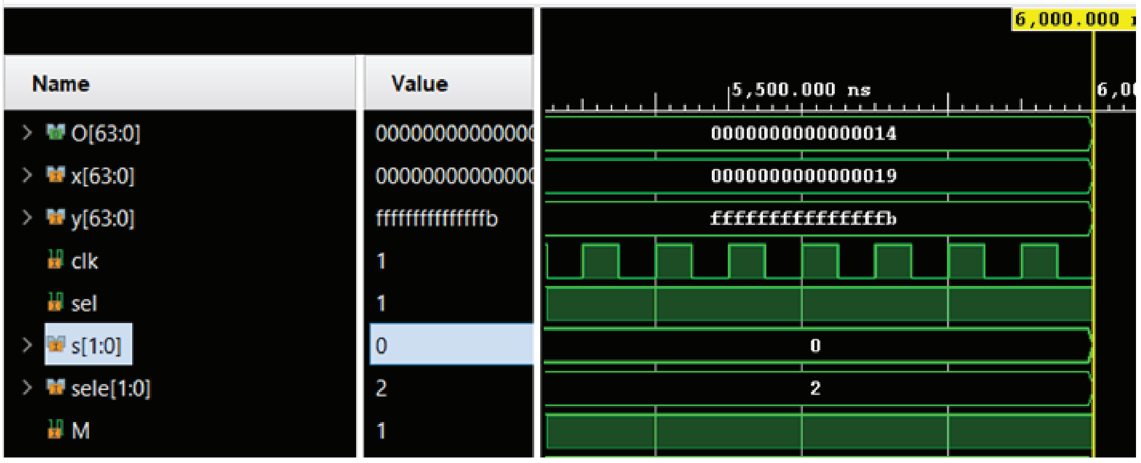
Name	Value	
out[63:0]	88888888888888	8888888888888888
A[63:0]	08080808080808	0808080808080808
B[63:0]	80808080808080	8080808080808080
clk	1	
sel	0	
M	Z	
s[1:0]	2	2
w1[127:0]	0000000000000	00000000000000000000000000000000
w2[127:0]	0000000000000	00000000000000000000000000000000

Fig. 8: Output simulation for XOR operation using an unified ALU.



Name	Value	
O[63:0]	3ff00000000000	3ff0000000000000
x[63:0]	00000000000000	0000000000000019
y[63:0]	00000000000000	0000000000000005
clk	1	
sel	1	
s[1:0]	3	3
sele[1:0]	1	1
M	1	

Fig. 9: Output simulation for floating-point division using an unified ALU.



Name	Value
O[63:0]	00000000000000
x[63:0]	00000000000000
y[63:0]	fffffffffffffffb
clk	1
sel	1
s[1:0]	0
sele[1:0]	2
M	1

Fig. 10: Simulation output for signed addition operation using an unified ALU.

Table 2: Area utilization by an unified ALU.

Resource	Utilization	Available	Utilization (%)
LUT	22129	53200	41.6
FF	1266	106400	1.19
DSP	21	220	9.55
IO	1	125	0.80

Table 3: Comparison of area utilization (%) by different ALUs.

Resource	Unsigned ALU	Signed ALU	Floating-point ALU	Unified ALU
LUT	10.15	10.81	23.61	41.6
FF	1.59	1.59	1.71	1.19
DSP	7.27	3.64	4.09	9.55
IO	0.8	0.8	0.8	0.80

Table 4: On-chip static and dynamic power consumption.

	Unsigned ALU	Signed ALU	Floating-point ALU	Unified ALU
Static power (W)	1.029	1.039	1.039	1.038
Dynamic power (W)	149.740	163.246	317.427	467.844

Table 5: On-chip power consumption.

	Unsigned ALU	Signed ALU	Floating-point ALU	Unified ALU
Signals	35.969	41.391	108.565	178.129
Logic	63.236	72.008	158.720	274.806
DSP	2.800	6.32	7.108	14.905
Input/output	47.734	43.527	43.033	0.004

ALU. With an occupancy of 41.6% of the look-up tables (LUTs), 9.55% of digital signal processors (DSP), 1.19% of flip-flops (FF), and 1 input-output (IO) port, the on-chip power consumption in terms of static and dynamic power consumption by the individual ALU segment and the unified ALU is presented in Table 4. Power consumption by the signals, logical implementation, DSP, and IO ports by the individual ALU segment and the unified ALU is presented in Table 5.

Conclusion and Future Scope

The multimodal ALU design presented in this paper aims to unify the processing of signed, unsigned, and floating-point arithmetic operations within a singular hardware module, thereby obviating the need for disparate arithmetic units and mitigating routing complexities. This integrated design paradigm accommodates both fixed-point and floating-point computations, rendering it highly applicable to domains such as scientific computing, engineering simulations, and real-time embedded systems where precision and computational throughput are imperative. The framework anticipates extensibility toward supporting heterogeneous data formats, including INT16, INT32, BF16, and BF32, thereby broadening its utility across diverse application scenarios. The proposed unified ALU design is intrinsically suitable for realization of application-specific integrated circuits (ASIC) as pipelined modules for multimode execution path for efficient arithmetic and logical tasks. This module will enable seamless switching with minimal pipeline stalls or mode reconfiguration overhead to boost performance and energy efficiency while processing cryptographic data or other applications requiring mixed-precision computation.

Moreover, future developments like vectorization through single instruction multiple data (SIMD) can further boost the ALU's capability by allowing parallel processing of multiple data elements. This evolution will make the multimodal ALU a valuable component for next-generation high-performance computing requirements for contemporary and future workloads.

References

1. Fei, X. (2025). Optimized design and applications of arithmetic logic units: Addressing power efficiency and performance in diverse computing applications. Applied and Computational Engineering, 128. https://doi.org/10.54254/2755-2721/2025.20216
2. Aarthy, M. (2014). ASIC implementation of 32 and 64-bit floating point ALU using pipelining. International Journal of Computer Applications, 94(17). https://doi.org/10.5120/16452-6184
3. Adela, N. A. S., Yousuf, A. N. B., & Eljhani, M. M. (2023). Design and implementation of single precision floating-point arithmetic logic unit for RISC processor on FPGA. In 2023 IEEE 3rd International Maghreb Meeting of the Conference on Sciences and Techniques of Automatic Control and Computer Engineering (MI-STA). IEEE. https://doi.org/10.1109/MI-STA57575.2023.10169623
4. Sahu, L., & Dev, R. (2012). An efficient IEEE754 compliant floating point unit using Verilog (Doctoral dissertation). National Institute of Technology Rourkela. Retrieved from http://ethesis.nitrkl.ac.in/3638/1/thesis_final.pdf
5. Savaliya, Y., & Rudani, J. (2020). Design and simulation of 32-bit floating point arithmetic logic unit using VerilogHDL. International Research Journal of Engineering and Technology, 7(12): 1467–1474. Retrieved from https://www.irjet.net/archives/V7/i12/IRJET-V7I12262.pdf
6. Jain, J., & Agrawal, R. (2015). Design and development of efficient reversible floating point arithmetic unit. In 2015 Fifth International Conference on Communication Systems and Network Technologies (pp. 811–815). IEEE. https://doi.org/10.1109/CSNT.2015.215
7. Khan, A. (2025). Challenges and solutions in low-power ASIC design for edge computing applications. Journal of Integrated VLSI, Embedded and Computing Technologies, 2(3), 12–22. https://doi.org/10.31838/JIVCT/02.03.02
8. Galal, S., & Horowitz, M. (2011). Energy-efficient floating-point unit design. IEEE Transactions on Computers, 60(7), 913–922. https://doi.org/10.1109/TC.2010.121
9. Govindu, G., Zhuo, L., Choi, S., & Prasanna, V. (2004). Analysis of high-performance floating-point arithmetic on FPGAs. In 18th International Parallel and Distributed Processing Symposium, 2004. Proceedings (p. 149). IEEE. https://doi.org/10.1109/IPDPS.2004.1303135
10. Veerappan, S. (2023). Designing voltage-controlled oscillators for optimal frequency synthesis. National Journal of RF Engineering and Wireless Communication, 1(1), 49–56. https://doi.org/10.31838/RFMW/01.01.06
11. Azhar, Y. N. J., & Adersh, V. R. (2023). FPGA implementation of a high speed efficient single precision floating point ALU. In 2023 International Conference on Control, Communication and Computing (ICCC). IEEE. https://doi.org/10.1109/ICCC57789.2023.10165441
12. Abdullah, D. (2024). Leveraging FPGA-based design for high-performance embedded computing. SCCTS Journal of Embedded Systems Design and Applications, 1(1), 37–42. https://doi.org/10.31838/ESA/01.01.07. https://doi.org/10.1109/DICCT64131.2025.10986594
13. C., S., S. M., R., M. M., & R, L. T. (2025). Comprehensive development and testing of a 32-bit floating-point ALU in Verilog. In International Conference on Electronics and Renewable Systems (ICEARS). IEEE. https://doi.org/10.1109/ICEARS64219.2025.10940212
14. Kourav, S., Thakur, D. S., Shah, S. K., & Verma, K. (2024). Area and speed-efficient floating-point arithmetic logical unit implementation on FPGA. In 2024 IEEE 13th International Conference on Communication Systems and Network Technologies (CSNT). IEEE. https://doi.org/10.1109/CSNT60213.2024.10545952
15. Arunkumar, K., Pavithra, V., & Prithyusha, V. (2025). High-performance 64-bit ALU for AI applications using Vedic Sutras and adaptive accuracy. In 2025 3rd International Conference on Device Intelligence, Computing and Communication Technologies (DICCT). IEEE. https://doi.org/10.1109/DICCT64131.2025.10986594
16. Al-Yateem, N., Ismail, L., & Ahmad, M. (2024). A comprehensive analysis on semiconductor devices and circuits. Progress in Electronics and Communication Engineering, 2(1), 1–15. https://doi.org/10.31838/PECE/02.01.01
17. Xilinx Inc. (2014). Vivado High-Level Synthesis user guide (UG902). Retrieved from https://www.xilinx.com/support/documentation/sw_manuals/xilinx2014_1/ug902-vivado-high-level-synthesis.pdf