Received: 28 March 2024 | *Revised:* 28 June 2024 | *Accepted:* 5 July 2024
DOI: 10.37917/ijeee.22.1.28 *Vol. 22 | Issue 1 | June 2026*

Iraqi Journal for Electrical and Electronic Engineering
Original Article

Open Access
IJEEE University of Basrah College of Engineering

Elastic Fixed-Point Arithmetic-Logic Unit Based on Hybrid Adder And Vedic Mathematic

Fatima GH. Ali*[1], Raqia Samir Almusali[2], Fatemah K. AL-Assfor[1]
[1]Computer Engineering Department, College of Engineering, University of Basrah, Basrah, Iraq
[2]Electrical and Computer Engineering Department, Altinbaş University, Istanbul, Türkeye

Correspondance
*Fatima GH. Ali
Department of Computer Engineering
University of Basrah, Basrah, Iraq
Corresponding address
Email: pgs.fatimah.ali@uobasrah.edu.iq

Abstract
High speed and area reduction of the Arithmetic-Logic Unit (ALU) have a fundamental role in modern processors, especially in digital signal processor (DSP). In this paper, a new elastic fixed-point (Fx-P) ALU module is proposed to perform multiple operations on real and complex numbers. The arithmetic part of the ALU executes operations such as addition, subtraction, increment, decrement, and multiplication on real numbers. For complex operands, the proposed ALU executes three operations comprising addition, subtraction, complex conjugate of complex numbers. The logical part performs the basic operations including AND, OR, NAND, NOR, XOR, XNOR, NOT and BUFFER operations. The proposed design is based on utilizing an enhanced design of a hybrid adder consists of a Han Carlson adder with a carry-select adder (EHC-CSLA) and an improved design of the Vedic multiplier to achieve multiplication operation of real numbers. A 16-bit and a 32-bit EHC-CSLA are designed first to perform real/complex addition-and- subtraction on both data types. Then, an improved-Vedic multiplier (IVM) is designed to perform multiplication on two real operands. The proposed EHC-CSLAs, numerous bit-sizes of the IVMs, and the elastic design of real/ complex ALU modules in this work are coded in VHDL, simulated, and synthesized by Xilinx ISE14.7 tool on different FPGA families. The performance results demonstrate appreciable reductions in delay and area usage in comparison to the most counterpart multipliers and ALU designs.

Keywords
Real /Complex ALU, Enhanced Hybrid Adder (EHA), Improved -Vedic Multiplier (IVM), FPGA Families.

I. Introduction

ALU represents the fundamental building block of a digital-signal processing (DSP), mobile, wristwatches, and other [1, 2].Further, in modern-day processors, a concoction of ALUs are utilized to build the graphical-processing unit (GPU) [3–6].The ALU is a multifunctional module that conditionally carries out one operation from a set of arithmetic or logical operations on one or two input operands based on selected control signals [7–9]. It can carry out their operations either on fixed-point (F_X-P) or floating-point (F_L-P) real numbers. Real F_X-P ALU is widely used in DSP since it allots the performance more significance than precision.
The elementary building block of the arithmetic part of the ALU consists of multiplexers, and parallel adders (namely; carry-ripple adder (CR-A)). The logic part of ALU comprises logic gates, and multiplexers only. The speed of ALU is mainly relied on the speed of the utilized adders and multipliers [10, 11].Modern ALUs comprises a multiplier to perform DSP algorithms and applications, like video

processing, Discrete-Fourier transform (DFT), impulse response (FIR) filters, intelligent systems and others. Thus,

an efficient design of the multiplier is a prime requirement in terms of high-performance and cost-efficient for these applications [12–14] .

Numerous multiplication algorithms have been presented and implemented in the last decades. One of these algorithms is the utilization of Vedic-mathematics to minimize the multiplier execution time, area-usage, and the power consumption [15]. A Urdhva-Tiryagbhyam Sutra (UT-SU) Vedic multiplier (VM) is one of the fast multipliers that can be employed in ALU to perform multiplication on real F_X-P or F_L-P numbers, since it allows the partial-product to be generated in parallel, eliminates unnecessary multiplication stages that involve zeros, and scales to a higher bit level [16].

The speed of any multiplier is mostly determined by the multiplication algorithm employed, as well as the accumulation of the sum and carry vectors to produce the eventual multiplication result. Consequently, minimizing the delay and area usage of the multiplier is one of the design goals in this work.

In addition to the importance of the real numbers in DSP algorithms and applications, complex numbers have a crucial role in DSP systems, mainly in applications, such as filtering, modulation, and Fourier-transform [17]. Consequently, ALU should have capabilities for handling both real and complex numbers.

A F_X-P complex number (A) is fundamentally a combination of a real part and an imaginary part, as: $A = (A\_re + iA\_im)$, where A_reis the real part of A, $A_imis$an imaginary part, and $i = \sqrt{-1}$.

The addition and subtraction of two complex numbers (A and B) can be accomplished as follows [18] :

$$\begin{aligned} A+B &= (A_{re}+iA_{im})+(B_{re}+iB_{im}) \\ A+B &= (A_{re}+B_{re})+i(A_{im}+B_{im}) \end{aligned} \tag{1}$$

and

$$\begin{aligned} A-B &= (A_{re}+iA_{im})-(B_{re}+iB_{im}) \\ A-B &= (A_{re}-B_{re})+i(A_{im}-B_{im}) \end{aligned} \tag{2}$$

A conjugate value of a complex number (A) is another complex number A^* that has the same real part as the original complex number, and the imaginary part has the same magnitude but opposite sign:

$$A^* = (A_{re}+iA_{im})^* = A_{re}-iA_{im} \tag{3}$$

Numerous ALU architectures have been offered and realized utilizing FPGA architecture:

R. Jaikumar et al. [19] had offered an 8-functions (4-bit) ALU encompassed three arithmetic operations (Add, Sub, and Mul), and five logical operations (two Not, And, OR, and XOR). They had used 4-bit RC-A to perform addition and subtraction, and (4X4)-bit VM to perform multiplication. Although they had used a VM, but their design had incurred high delay and area due to the utilization of eight full adders (FA)s and three half adders ((HA)s to design their VM. Accordingly, the delay of their ALU is also high.

P. Nautiyal, et al. [20] had introduced ALU multi bit-sizes: 8, 16, 32, and 64 bits. Their ALU designs had include a revised form of square-root CS-A which comprised of the CS-A and carry-lookahead adder (CL-A). They had used Verilog language to perform their designs. However, their designs suffered from the delay propagation generated by the CL-AS.

Chetan B V et al. [21] had proposed an 8-bit ALU with four basic arithmetic operations; namely: Add, Sub, Mul, and Div. Their ALU comprised of 8-bit RC-A to perform addition and subtraction, A (8X8)-bit VM based on the UT-SU algorithm along with three (8-bit) RC-As, and 8-bit divider based on the approach of Nikhilam-Sutra (N-SU) Vedic algorithm. Their design had introduced high area occupation and high delay due to the use of several RC-As.

G. M. Tang et al. [22] have designed a 16-bit bit-slice ALU for 32/64-bit Fast single flux-quantum (FSFQ) microprocessors. Their design involves all the operations needed for MIPS32 instructions set. They had employed Ladner-Fischer (LF-A) adders to perform the basic arithmetic operations.

T. B. Yadav et al. [23] have designed a 16-bit ALU architecture using Vedic computations approach to perform the four basic arithmetic operations (Add, Sub, Mul and Div) The design had generated relatively high delay due to the utilization of RC-As in designing the ALU adder and multiplier.

S.B. Shirol et al. [24] had introduced an 8 and a 16-bit ALUs utilizing a reversible- logic gates (RL-G) s. Their designs implicated a hybrid-adder composed of carry-select adder (CSL-A) and a Kogg-Stone adder (KS-A) to carry out the addition and subtraction operations, and a VM to perform multiplication. The authors had used a hybrid adder formed of CS-A and RC-A to add the partial products of VM. Their ALUs designs had better performance than the previous designs due to the use of the hybrid adders. But the overall delay of the multipliers and accordingly the ALUs had remained relatively high due to the use of RC-As to reduce the partial products of the multiplier.

J. L. V. Ramana Kumari et al. [25] have designed an 8-function (4-bit) ALU by utilizing 8-bit hybrid-ring counter, rather than using multiplexers. The hybrid (or programmable)-ring counter is a mix of regular and twisted ring counters. Their design performance has incurred high delay due to the utilization of the ring counter.

M. A. Raza et al. [26] have offered and realized two ALUs. The first one performed eight function on 8-bit data (two arith-

metic and six logic operations) while the 2nd one performed 16-function for a 16-bit data (six arithmetic operations and ten logic operations) with relatively good performance metrics. G. Surekha et al. [27] have offered 9-function ALU performed on different data sizes. The nine functions include (add, sub, MUl, compare, AND, OR, NOT, XOR, and shift). They have used CR-A and CL-A to perform addition and subtraction and VM to perform multiplication. However, their designs have introduced high delay and hardware occupation (area) due to the use of CR-As and CL-As.

The main new contribution of this work is the design of an elastic ALU to efficiently performs arithmetic operations on real and complex numbers, through performing the following:
• Design an efficient adder to perform FX-P real/complex addition and subtraction utilizing an enhanced design of nonlinear hybrid adder consists of CSL-A and a Han-Carlson adder (HC-A). The adder is called here EHC-CSLA.
• Design an improved VM (called (IVM) through using the EHC-CSLA to add the intermediate sum and carry vectors yielded by the CS-A to obtain the final multiplication result. This step is important since the ALU's speed is mainly relied on the multiplier speed.
• Employ the adjusted XOR-gate designed in [28] to design the entire modules of the proposed ALU.

This work is organized as follows: Section II. offers an enhanced design of a hybrid adder comprised of a CSL-A and a HC-A called EHC-CSLA to perform real/complex additions and subtraction efficiently and makes down an improved design of the VM (IVM) with different bit sizes to perform multiplication of two real operands based on the use of the proposed EHC-CSLA. Section III. proposes a real/complex ALU with two different bit-size (16-bit and 32-bit). Section IV. highlights the simulation results and performance appraisal of the IVM and FX-P real/complex ALU. In section V. , the conclusion of this work is presented.

II. Proposed Efficient Adder and Multiplier

The key components of any ALU are the adder and multiplier. Thus, an efficient adder and multiplier have designed first in this article to be used in designing efficient and elastic ALU.

A. Enhanced Han- Carlson Carry-Select Adder (EHC-CSLA)

The article offers an efficient hybrid adder composed of an enhanced design of the HC-A (which is a kind parallel prefix adders) and CSL-A (EHC-CSLA). The proposed architecture of the EHC-CSLA encompasses a set of diverse bit-sizes of HC-A, binary-to-excess-1 convertors (BE1C) s, and MUXs.

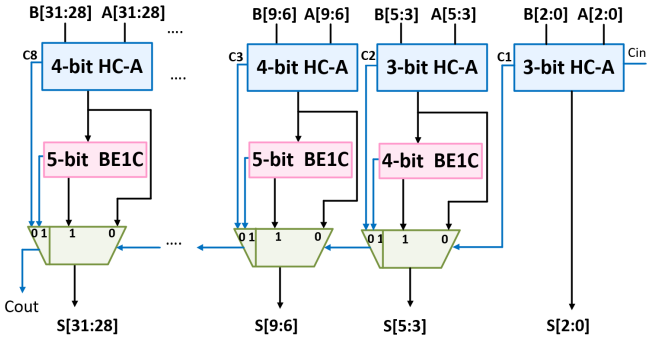


Fig. 1. Proposed 32-bit EHC-CSLA.

The purpose of BE1C is to increment the result of each HC-A by 1.
The proposed adder can be used as a parallel adder for the ALU to perform the addition and subtraction and to generate the product result of the multiplication operation. Fig. 1 illustrates the proposed architecture of the (32-bit) EHC-CSLA.

B. Improved-Vedic Multiplier (IVM)

Recently, Vedic Mathematics is attained high importance, thus numerous studies have highlighted the use of these mathematics. Multiplication using VM relies on the UT-SU (or vertical-cross) approach [29] .
The basic algorithm to multiply two (n-bit) operands (such as A and B) using UT-SU comprises the following steps:
Step 1: Generate a set of four partial-products (PP) vectors.
Step 2: Reduce the PP vectors to two vectors by employing a set of adders such as RC-As, Carry-save adders (CS-A), CL-As, etc. These two vectors are called "redundant product vectors" (or simply: sum and carry).
Step 3: Generate the final product (result of the multiplication) from the two redundant product vectors of step 2 using another adder.
Based on UT-SU, the conventional (2X2)-bit VM accepts two (2-bit) input data ($A \equiv A[1:0]$ and $B \equiv B[1:0]$) to provide 4-bit product result (namely Pr[3:0]) [24] . The architecture of the (2X2)-bit VM embraces four AND gates and two half-adders (HA)s [30] . Mathematically, the algorithm can be illustrated as

$$\begin{aligned} Pr[0] &= A[0]XB[0] \\ C_{ro1}Pr[1] &= A[1]XB[0] + A[0]XB[1] \\ Pr[3:2] &= A[1]XB[1] + C_{ro1} \end{aligned} \tag{4}$$

where $Pr[i]$: is the i^th -bit of the product result. A (4X4)-bit IVM module is designed first in this work through employing four (2X2)-bit VM modules, a (4-bit) CSA, a (3-bit) EHC-CSL-A to accomplish the addition in the second (middle)

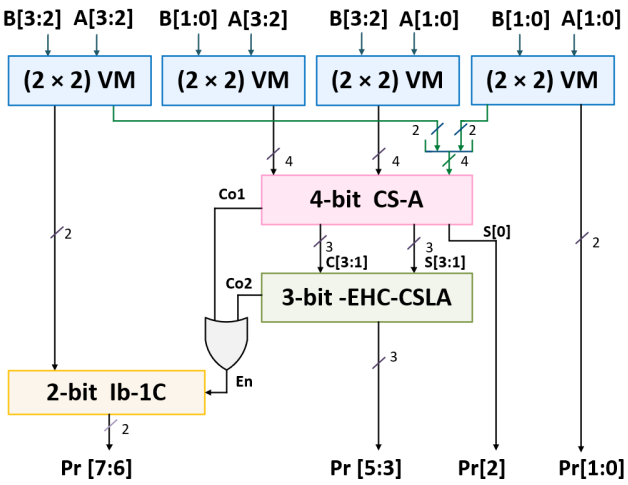
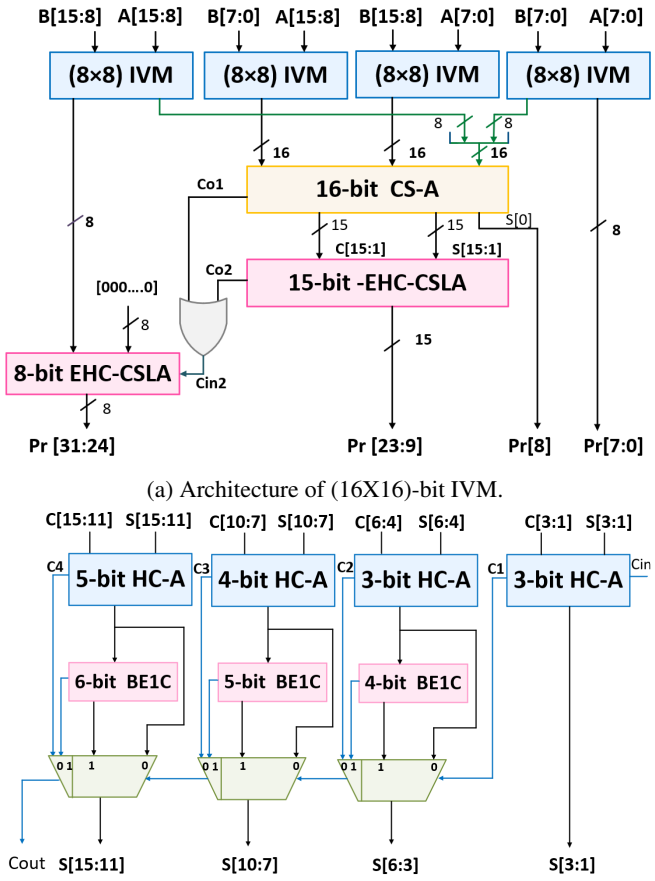


Fig. 2. Proposed (4X4)-bit IVM.



(a) Architecture of (16X16)-bit IVM.

(b) Internal organization of 15-bit EHC-CSLA.

Fig. 3. The Proposed (16X16)-bit IVM utilizing an EHC-CSLA.

stage of the multiplier, and a 2-bit increment by-1converter (Ib-1C) to generate the bits in the third stage (most significant bits (MSB)s) of multiplication as illustrated in Fig. 2.

The (4X4)-bit IVM is then extended to design an (8X8)-bit, which in turns utilized to design the (16X16)-bit, and the (32X32)-bit IVMs as illustrated in Fig. 3 and Fig. 4. Fig. 4a expounds a (32X32)-bit VM module. It encompasses four (16X16)-bit IVM modules to generate four (32-bit) partial-products (Pr) vectors. Next, a (32-bit) CS-A is used to minify the s to two (32-bit) vectors; i.e. S [31:0] and C [31:0].

The CS-A is then followed by a 31-bit EHC-CSLA to add these two vectors to produce the final product of the middle stage (namely: Pr[47:17]. It consists of eight blocks of variable bit sizes of HC-adders along with seven BE1C and a set of MUXs to select the final product result according to the input carry as demonstrated in Fig. 4b.

The two output carries (Co1 and Co2) of the CS-A and the EHC-CSLA are OR-ed together to find the carry-in for a 16-bit EHC-CSLA which is utilized for generating most significant 16-bits of the product (Pr).

III. Real/Complex ALU Design

A novel design of elastic FX-P Real/Complex ALU block is proposed in this article to supports two modes of arithmetic operations; namely arithmetic operations for real-input operands and complex-input operands in addition to the logical operations on real numbers.

Fig. 5 and Table I expose the block scheme of the proposed ALU and the diverse arithmetic and logic operations that can be executed, respectively. As listed in Table I, the proposed ALU adepts of performing eleven arithmetic operations (eight of them for real-operands and the rest for complex operands), in addition to eight logic operations.

The arithmetic unit speed is of enormous importance and highly relied on the speed of the utilized multiplier. The proposed architecture of the elastic Real/Complex ALU embraces of an n-bit EHC-CSLA (as a parallel- adder) to carry out the real/complex arithmetic operations, an n-bit IVM to perform real-number multiplications, a set of logic-gates to perform the logic operations in logic unit, a demultiplexer (DEMUX) to distinguish between the real and imaginary parts of complex operands, and a set of multiplexers (MUX) to select the required operation to be executed.

The ALU utilizes six control signals as follows: The signals S4-S0 are utilized to select and implement the numerous arithmetic and logic functions, and the Real/Complex signal or (simply R/C) which is employed to select the real or the complex arithmetic operations within the arithmetic unit.

To make the EHC-CSLA elastic in performing addition and subtraction on real and complex operands, the adder archi-

TABLE I.
ARITHMETIC – LOGIC UNIT FUNCTIONS

Control inputs						Output	Function	
R/C	S4	S3	S2	S1	S0	Z		
0	0	0	0	0	0	A+B	Addition	Real-Numbers
0	0	0	0	0	1	A-B	Subtraction	
0	0	0	0	1	0	A-1	Decrement A	
0	0	0	0	1	1	A+1	Increment A	
0	0	0	1	0	0	A-B-1	Subraction with borrow	
0	0	0	1	0	1	A+B+1	Addition with carry	
0	0	0	1	1	0	2A	Doubling A	
0	0	1	X	X	X	A*B	Multiplication	
1	0	0	X	0	0	A+B	Addition	Complex Numbers
1	0	0	X	0	1	A-B	Subtraction	
1	0	0	X	1	0		Complex conjugate (A)	
0	1	X	0	0	0	A.B	AND	Logical operations on Real Numbers
0	1	X	0	0	1	A+B	OR	
0	1	X	0	1	0	$\overline{A.B}$	NAND	
0	1	X	0	1	1	$\overline{A+B}$	NOR	
0	1	X	1	0	0	$A \oplus B$	XOR	
0	1	X	1	0	1	$A \odot B$	XNOR	
0	1	X	1	1	0	$\overline{A}$	NOT	
0	1	X	1	1	1	A	BUFFER	

tecture in Fig. 1 has amended by inserting three multiplexers controlled by the signal R/C at the center of the adder as revealed in Fig. 6.
The R/C signal is used for adapting this adder to perform addition or subtraction according to the type of operands received. If R/C = 0 the adder will perform addition or subtraction on real input-operands, else the adder performs addition or subtraction on complex input-operands.

The two n-bit FX-P input-operands of the EHC- CSLA module are either real numbers or complex numbers, as manifested in Fig. 7.

The proposed ALU has been designed for two data-sizes; namely: 16-bit and 32-bit. Fig. 8 demonstrates the architecture of 32-bit elastic ALU. To extremely improving the area utilization and the speed of the elastic ALU, the adjusted XOR-gate in [28] has utilized in designing the ALU.

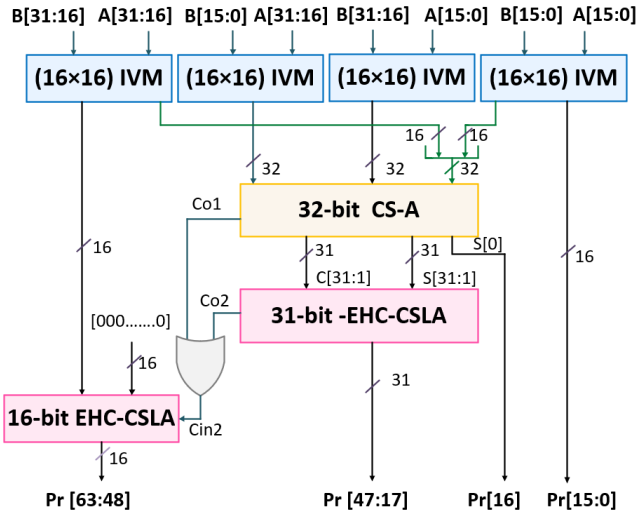
IV. RESULTS AND COMPARISON

A. Simulation and synthesis

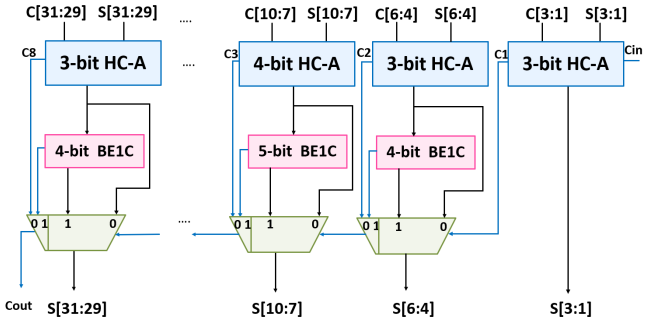
The proposed EHC-CSLA, IVMs, and elastic Real/Complex ALU modules are coded in VHDL and their performances in terms of the FPGA area occupation and the delay (speed=1/delay) are estimated in Xilinx tool using the following seven FPGA families: Spartan 3E, Spartan-6, Artix-7, Virtex-(5-to-7), and Zynq, as demonstrated in Table II, Table III and Table IV.

To best of our knowledge, there has no such ALU design for elastic data types coming to light in the literature. Fig. 9 displays the synthesized register-transfer-level (RTL) scheme of the 32-bit ALU. It can be discerned that the 32-bit ALU architecture has exploited three main modules; namely: 32-bit arithmetic-unit, 32-bit Logic unit, and 64-bit (2:1) MUX. Moreover, Fig. 10 demonstrates a deeper view for the internal design of the proposed ALU.
It incorporates the following modules: A (32×32)-bit IVM, a 32-bit EHC-CSLA, a 32-bit of ((3:1) (6:1) and (8:1)) MUXs, a 32-bit (1:2) DEMUX, numerous (2:1) MUXs, in addition to a set of logical gates (AND, OR, adjusted-XOR, NOT) for the logical unit.

The proposed ALU is simulated to functionally proving its capability in execution multi operations on real and complex



(a) (Architecture of (32X32)-bit IVM.



(b) Internal organization of 31-bit EHC-CSLA.

Fig. 4. The Proposed (32X32)-bit IVM utilizing an EHC-CSLA

operands. Fig. 11a and Fig. 11b states the 32-bit elastic ALU simulation result for execute nineteen arithmetic and logic functions (set R/C = 0) for real operands and (set R/C = 1) for complex operands. It can be seen that hexadecimal numbers

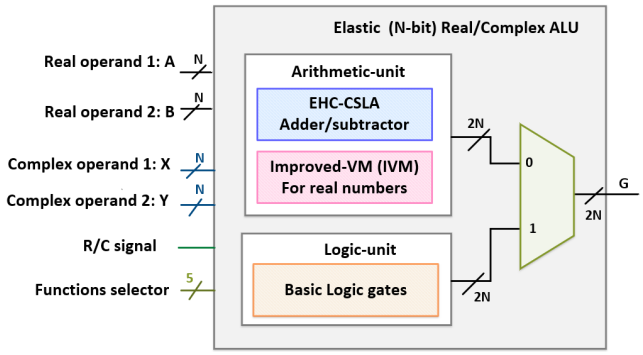


Fig. 5. The Block scheme of an n-bit Real/Complex ALU.

TABLE II.
EHC-CSLA AREA AND DELAY RESULTS

Parameter	FPGA Family	Proposed EHC-CSLA	
		16-bit	32-bit
No. of FPGA LUT	Spartan-3E	56	98
	Spartan-6	41	65
	Artix-7	34	56
	Virtex-5	34	56
	Virtex-6	34	56
	Virtex-7	34	56
	Zynq	34	56
Delay (ns)	Spartan-3E	5.340	9.383
	Spartan-6	4.871	7.561
	Artix-7	2.561	3.276
	Virtex-5	3.85	4.677
	Virtex-6	2.589	3.458
	Virtex-7	2.127	3.274
	Zynq	2.127	3.274

TABLE III.
IVM RESULTS: AREA AND DELAY

Metric	FPGA Family	IVM (bit)			
		4X4	8X8	16X16	32X32
No. of FPGA LUT	Spartan-3E	23	125	576	2326
	Spartan-6	21	113	502	2079
	Artix-7	21	113	502	1581
	Virtex-5	21	115	510	1610
	Virtex-6	21	113	502	1579
	Virtex-7	21	113	502	1579
	Zynq	21	113	502	1579
Delay (ns)	Spartan-3E	10.409	18.220	30.613	44.006
	Spartan-6	7.181	11.755	18.676	27.212
	Artix-7	3.200	5.92	9.011	13.391
	Virtex-5	6.056	8.762	12.577	18.373
	Virtex-6	2.41	4.21	8.86	12.839
	Virtex-7	2.339	4.075	7.756	12.470
	Zynq	2.339	4.075	7.756	12.470

were utilized in the simulation to verify the ALU operation.

B. Comparison with Recent Counterpart Works

Table V exhibits a comparison of the proposed IVM module with the counterparts in the literature based on delay and area for diverse bit-width. It can be perceived that the proposed IVM attains the foremost speed up and the lowest area usage compared with other designs in literature. As an example, the proposed IVM reaches highest speedup of 30.37% and lowest area usage of 34.21 % than the (8×8)-bit VM introduced

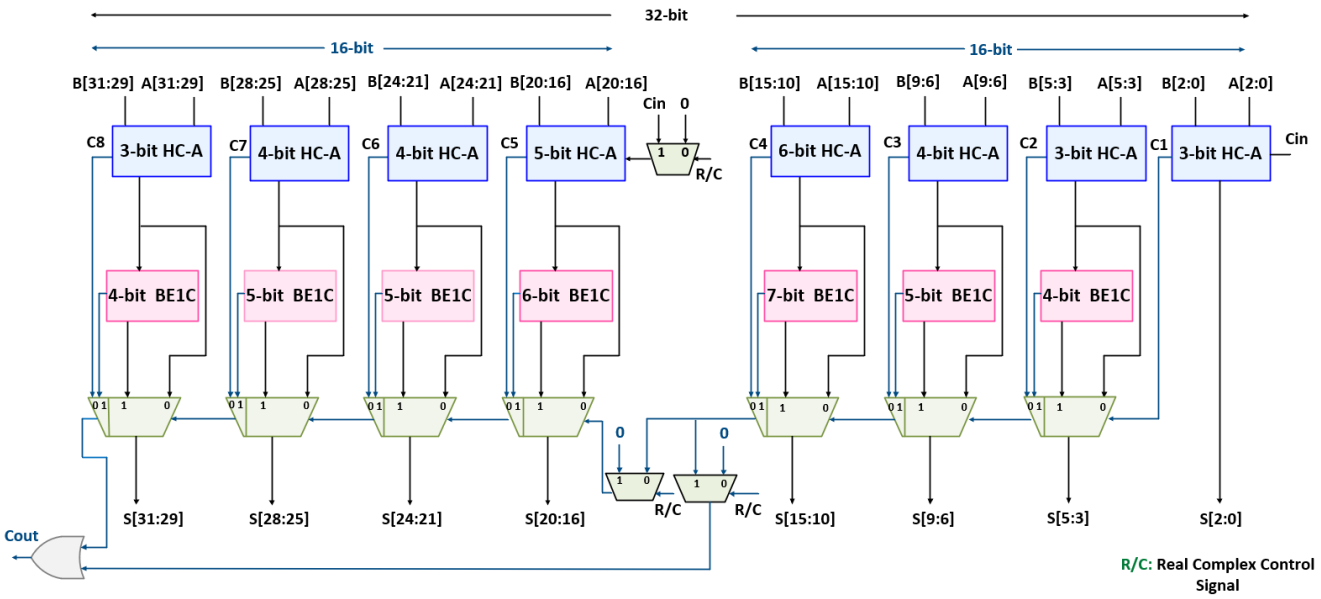


Fig. 6. Elastic Real/Complex EHC-CSLA Design

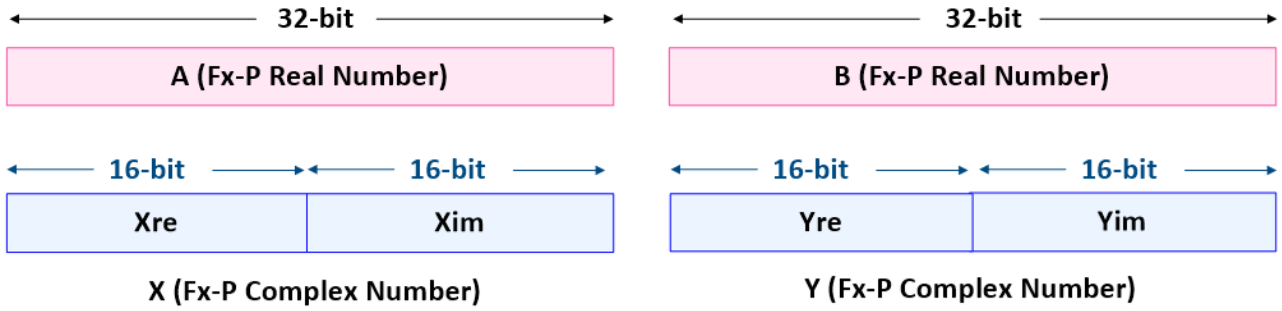


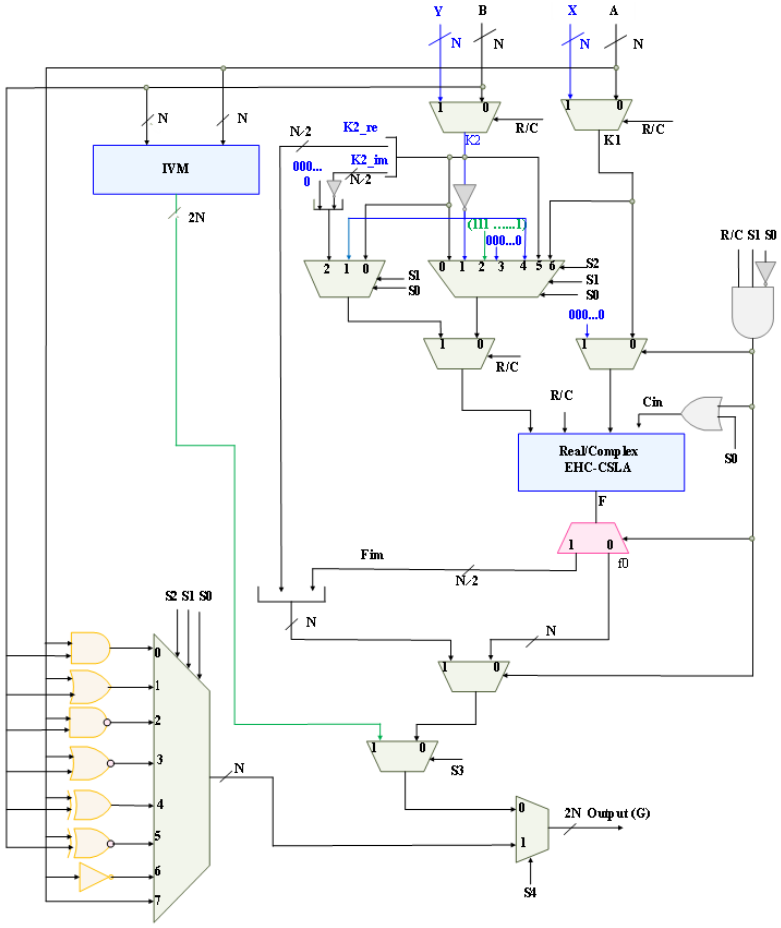
Fig. 7. 32-bit FX-P Real and complex formats.

in [14] for the same FPGA family (Spartan 3E). Also, the (16X16)-bit IVM has gained a speed up reaches the double (50.06%) compared with (16X16)-bit VM in [13] but the area of IVM is larger than their design for the same FPGA family (Vertix-7).

Table VI evinces that the proposed elastic FX-P Real/ Complex 16-bit and the 32-bit ALU blocks are speedier (i.e. achieved less delay) and consumes lesser area resources than all the ALU blocks in the literature.

V. Conclusion

This work introduces a novel design of elastic FX-P Real/ Complex ALU. The proposed design is based on utilizing an enhanced design of a hybrid adder consists of a Han-Carlson adder with a carry-select adder (EHC-CSLA) to perform multi-arithmetic operations on real or complex operands, in addition to logical operations. Further, the arithmetic unit of the proposed ALU involves an improved design of the Vedic multiplier to achieve multiplication operation of real numbers. The proposed ALU has been designed for two data-sizes; namely: 16-bit and 32-bit. The performance-metric results demonstrate noticeable reduction in delay and area resources in comparison to the most present ALU and Vedic-multiplier designs. The proposed IVM achieves reduction in delay and FPGA area occupation by 30.37% and 34.21 % respectively for (8X8)-bit IVM design, and reduction in delay of 50.06 % for (16X16)-bit IVM and the 32-bit proposed ALU design have attained lowest delay and area in comparison to all ALU designs in the literature.



A, B: Real Numbers.
X, Y: Complex Numbers
R/C: Real/Complex Control Signal

Fig. 8. Elastic FX-P Real/Complex 32-bit ALU.

Conflict of Interest

The authors have no conflict of relevant interest to this article.

References

[1] S. Sasikala, S. Balambigai, P. Sivaranjani, and V. Udhayasuriyan, "Design of arithmetic logic unit using hybrid power reduction methodologies for super computer applications," in *2023 14th International Conference on Computing Communication and Networking Technologies (ICCCNT)*, pp. 1–8, IEEE, 2023.

[2] U. Lakadiwala, S. Hirapara, R. Ramani, and N. Chaudhary, "Implementation of alu on fpga," *International Research Journal of Engineering and Technology (IRJET) Vol*, vol. 3, 2016.

[3] J. Swathi, K. R. L. Tangudu Santhoshi, S. M. P. Yugandhar, and U. T. Sai, "Implementation of arithmetic logic unit using high speed carry-skip adder and booth's multiplier," *International Research Journal of Modernization in Engineering Technology and Science*, vol. 103, pp. 103–108, 2022.

[4] M. Rangari, R. Saraswat, and R. Jain, "Design of reversible logic alu using reversible logic gates with low delay profile," *International Journal of Advanced Research in Computer and Communication Engineering*, vol. 4, pp. 344–348, 2015.

[5] S. Rasappan, G. Thangavel, S. R. Ahmed, S. A. Nishath, M. A. Ali, and N. M. Tahir, "Design and implementation of 3–bit calculator for an alu using vertical and crosswise multiplication," in *2023 IEEE 13th International Con-*

TABLE IV.
PERFORMANCE RESULTS OF THE OFFERED F_X-P ALUS

Parameter	FPGA Family	Proposed ALU	
		16-bit	32-bit
No. of FPGA LUT	Spartan3E	689	2647
	Spartan-6	621	2309
	Artix-7	611	2309
	Virtex-5	621	2316
	Virtex-6	611	2309
	Virtex-7	611	2309
	Zynq	611	2309
Delay (ns)	Spartan3E	31.59	45.015
	Spartan-6	19.016	27.941
	Artix-7	9.197	13.858
	Virtex-5	12.965	18.739
	Virtex-6	9.093	13.031
	Virtex-7	8.065	12.639
	Zynq	8.065	12.639

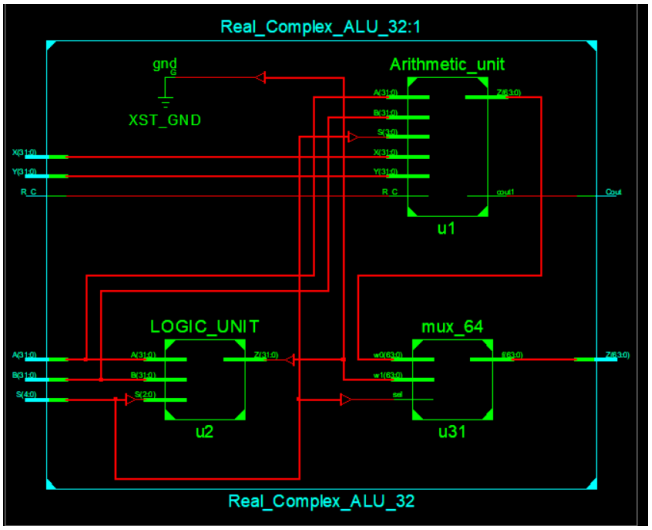


Fig. 9. RTL-diagram of the proposed 32-bit F_X-P ALU

ference on Control System, Computing and Engineering (ICCSCE), pp. 309–313, IEEE, 2023.

[6] N. K. Kachhwaha and S. Shah, "Implementation of efficient fixed point alu with 32 bit processing capability," 2017.

[7] A. A. Purohit, M. R. Ahmed, and R. V. S. Reddy, "Design of area optimized arithmetic and logical unit for microcontroller," in *2020 IEEE VLSI DEVICE CIRCUIT AND SYSTEM (VLSI DCS)*, pp. 335–339, IEEE, 2020.

TABLE V.
COMPARISON OF MULTIPLIERS BASED ON DELAY AND AREA FOR DIVERSE BIT-WIDTH

Ref.	FPGA Family	VM size (bit)	Delay (n sec)	Area Utilization (LUT No.)
[23]	Virtex-7	(16X16)	11.586	-
[29]	Spartan-6	(4X4)	10.43	24
		(8X8)	18.46	125
	Virtex-4	(4X4)	8.54	30
		(8X8)	13.87	153
[14]	Spartan3E	(8X8)	26.169	190
[13]	Spartan-6	(16X16)	32.175	164
	Artix-7		24.903	221
	Virtex-6		17.776	157
	Virtex-7		15.531	185
Proposed	Spartan3E	(4X4)	10.409	23
		(8X8)	18.220	125
		(16X16)	30.613	576
		(32X32)	44.006	2326
	Spartan-6	(4X4)	7.181	21
		(8X8)	11.755	113
		(16X16)	18.676	502
		(32X32)	27.212	2079
	Aritex-7	(4X4)	3.200	21
		(8X8)	5.920	113
		(16X16)	9.011	502
		(32X32)	13.391	1581
	Virtex-5	(4X4)	6.056	21
		(8X8)	8.762	115
		(16X16)	12.577	510
		(32X32)	18.373	1610
	Virtex-6	(4X4)	2.41	21
		(8X8)	4.201	113
		(16X16)	8.86	502
		(32X32)	12.839	1579
	Virtex-7	(4X4)	2.339	21
		(8X8)	4.075	113
		(16X16)	7.756	502
		(32X32)	12.470	1579
	Zynq	(4X4)	2.339	21
		(8X8)	4.075	113
		(16X16)	7.756	502
		(32X32)	12.470	1579

[8] S. Jamuna, P. Dinesha, K. Shashikala, and K. K. Kumar, "Area optimized run-time reconfigurable alu for digital systems," in *2019 Second International Conference on Advanced Computational and Communication Paradigms (ICACCP)*, pp. 1–5, IEEE, 2019.

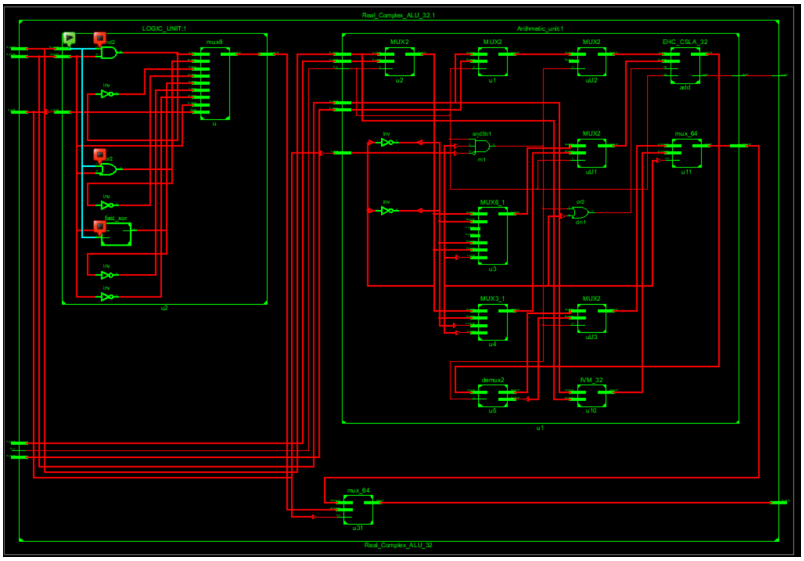
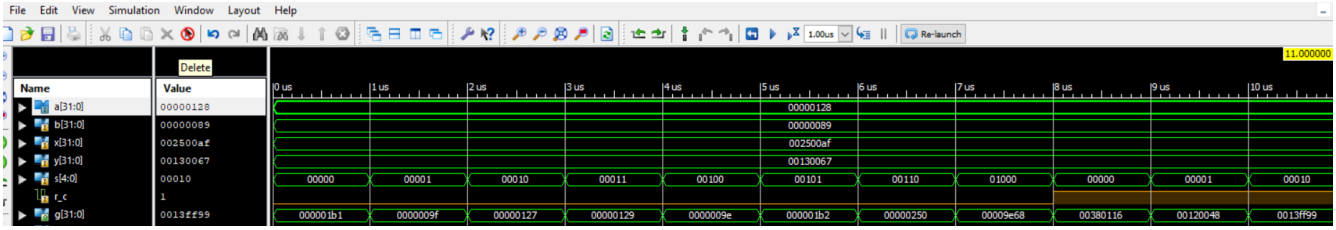


Fig. 10. RTL internal-schematic of 32-bit F_X-P ALU design



(a) (Arithmetic unit waveform

(b) logic unit waveform

Fig. 11. (a, b) Input/output waveform of 32-bit F_X-P ALU.

[9] I. Said and M. Çavuş, "Alu design by vhdl using fpga technology and micro learning in engineering education," *British Journal of Computer, Networking and Information Technology, January*, pp. 1–18, 2018.

[10] A. Kumar, S. K. Gupta, and P. Kota, "4-trit cnfet-based arithmetic logic unit," in *2023 International Conference on Device Intelligence, Computing and Communication Technologies,(DICCT)*, pp. 34–38, IEEE, 2023.

[11] A. Jose and K. Jyothisree, "8-bit arithmetic logic unit (alu) using full swing restored m-gdi technique," in *International Conference on Communication, Embedded-VLSI Systems for Electric Vehicle (ICCEVE 2023)*, vol. 2023, pp. 49–53, IET, 2023.

[12] V. P. Brahmaiah, V. K. Gurrala, S. T. Tuduru, and K. H. Kishore, "Design of area and power-optimized vlsi architecture of aludesign using signed multiplier," in *2022 International Conference on Recent Trends in Microelectronics, Automation, Computing and Communications*

TABLE VI.
COMPARISON OF MULTIPLIERS BASED ON DELAY AND AREA FOR DIVERSE BIT-WIDTH

Ref.	FPGA Family	ALU size (bit)	Delay (n sec)	Area Utilization (LUT No.)
[17]	-	32	70.646	4242
[12]	-	32	20.347	4656
[23]		16	34.168	-
[24]	-	8	19.133	251
		16	29.64	950
Proposed	Spartan3E	16	31.59	689
		32	45.015	2647
	Spartan-6		19.016	621
			27.941	2309
	Aritex-7		9.197	611
			13.858	2309
	Virtex-5	16	12.965	621
		32	18.739	2316
	Virtex-6	16	9.093	611
		32	13.031	2309
	Virtex-7	16	8.065	611
		32	12.639	2309
	Zynq	16	8.065	611
		32	12.639	2309

Systems (ICMACC), pp. 276–280, IEEE, 2022.

[13] P. Sairam, K. Manikumar, Y. S. Reddy, B. U. Narayana, and K. Gowreesrinivas, "Fpga implementation of area efficient 16-bit vedic multiplier using higher order compressors," in *2023 IEEE Devices for Integrated Circuit (DevIC)*, pp. 404–407, IEEE, 2023.

[14] J. K. KJ, P. Kaythry, V. Gokhulesh, and K. Sudharsan, "Efficient fpga implementation of rsa algorithm using vedic multiplier," in *2023 International Conference on Wireless Communications Signal Processing and Networking (WiSPNET)*, pp. 01–05, IEEE, 2023.

[15] J. Nayak and S. B. Kaje, "Fast image convolution and pattern recognition using vedic mathematics on field programmable gate arrays (fpgas)," in *2022 OITS International Conference on Information Technology (OCIT)*, pp. 569–573, IEEE, 2022.

[16] S. S. Saokar, R. Banakar, and S. Siddamal, "High speed signed multiplier for digital signal processing applications," in *2012 IEEE International Conference on Signal Processing, Computing and Control*, pp. 1–6, IEEE, 2012.

[17] V. Swathi, K. Panduga, and G. S. Kumari, "Design of high performance alu using vedic mathematics," in *Journal of Physics: Conference Series*, p. 62031, IOP Publishing, 2021.

[18] Y. Zhao, "Discovery of complex numbers," *Highlights in Science, Engineering and Technology*, vol. 38, pp. 138–143, 2023.

[19] R. Jaikumar, P. Poongodi, and R. Lavanya, "Implementation of high speedarithmetic logic using vedic mathematics techniques,"

[20] P. Nautiyal, P. Madduri, and S. Negi, "Implementation of an alu using modified carry select adder for low power and area-efficient applications," in *2015 International Conference on Computer and Computational Sciences (ICCCS)*, pp. 22–25, IEEE, 2015.

[21] M. V. Chetan B V, Arpitha H V, "Fpga implementation of alu using vedic mathematics," *IOSR Journal of VLSI and Signal Processing*, vol. 6, pp. 8–12, 2016.

[22] G.-M. Tang, P.-Y. Qu, X.-C. Ye, and D.-R. Fan, "Logic design of a 16-bit bit-slice arithmetic logic unit for 32-/64-bit rsfq microprocessors," *IEEE Transactions on Applied Superconductivity*, vol. 28, no. 4, pp. 1–5, 2018.

[23] D. S. C. T. B. Yadav, "Implementation of arithmetic logic unit using vedic mathematics," *JETIR*, 2019.

[24] S. B. Shirol, S. Ramakrishna, and R. B. Shettar, "A novel design and implementation of 8-bit and 16-bit hybrid alu," in *Intelligent Systems Design and Applications: 18th International Conference on Intelligent Systems Design and Applications (ISDA 2018) held in Vellore, India, December 6-8, 2018, Volume 1*, pp. 32–42, Springer, 2020.

[25] J. R. Kumari, Y. Varshitha, C. Gopi, and M. Rakesh, "Design and implementation of alu using ring counters," in *2023 International Conference on Signal Processing, Computation, Electronics, Power and Telecommunication (IConSCEPT)*, pp. 1–5, IEEE, 2023.

[26] M. A. Raza, I. Shahzad, H. Anwar, M. A. Qureshi, F. H. Malik, and M. U. A. Khan, "An optimum design and implementation of a 16-bit alu on cadence using risc-v architecture," in *2023 IEEE International Conference on Emerging Trends in Engineering, Sciences and Technology (ICES&T)*, pp. 1–5, IEEE, 2023.

[27] G. Surekha, G. Madesh, M. P. Kumar, and H. Sriramoju, "Design and implementation of arithmetic and logic unit

(alu)," in *2023 2nd International Conference on Applied Artificial Intelligence and Computing (ICAAIC)*, pp. 1530–1536, IEEE, 2023.

[28] F. K. Al Assfor, I. S. Al-Furati, and A. T. Rashed, "Vedic-based squarers with high performance," *Indonesian Journal of Electrical Engineering and Informatics (IJEEI)*, vol. 9, no. 1, pp. 163–172, 2021.

[29] S. Akhter and S. Chaturvedi, "Modified binary multiplier circuit based on vedic mathematics," in *2019 6th international conference on signal processing and integrated networks (SPIN)*, pp. 234–237, IEEE, 2019.

[30] P. V. K. Kanth, "Simulation and implementation of vedic multiplier using vhdl," *Department of Electronics & Communication Engineering R.V.R. & J.C. College of Engineering, (Affiliated to Acharya Nagarjuna University) Chandramoulipuram,Chowdavaram GUNTUR – 522019, Andhra Pradesh, INDIA*, 2015.