International Journal of Engineering Science and Advanced Technology (IJESAT)
Vol 26 Issue 11, November 2026
Received: 05-10-2026 Accepted: 08-11-2026 Published: 16-11-2026

Design of Generic Vedic ALU Using Reversible Logic

[1]A. Jayasilpa, [2]Smt. Dr. G. Jyoshna
[1]PG Student
[2]Assistant Professor
[1,2]Department of Electronics and Communication Engineering,
[1,2]Sri Krishnadevaraya University College of Engineering and Technology (SKUCET), Anantapuramu, Andhra Pradesh, India
*Corresponding author: jayashilpaa47@gmail.com

***Abstract*—Power dissipation is a first-order constraint in digital system design, and logically irreversible computation carries a fundamental lower bound on energy per erased bit. Reversible logic, in which every output vector identifies a unique input vector, is one route to circuits that avoid this bound in principle. This paper presents the design, Verilog implementation and verification of a parameterized generic arithmetic logic unit (ALU) built from reversible gates around an Urdhva Tiryakbhyam (Vedic) multiplier. The ALU combines a ripple-carry adder/subtractor made of HNG gates, a recursive N×N Vedic multiplier whose 2×2 base block uses BVF, BME and Peres gates, a logic unit, and a three-way output multiplexer made of BHA gates. The architecture and gate library follow a published design; the implementation reconciles an inconsistent control encoding and uses a second Peres gate in place of a CNOT stage in the 2×2 block. A self-checking testbench on a 64-bit instance completed 928 checks with no mismatches, and the multiplier was verified exhaustively up to 8×8 operands. Mapping to Xilinx 7-series LUTs gives 41 to 11,430 LUTs for 4- to 64-bit operands, and the reversible gate count follows $G(N)=2.5N^2-2.5N$ for the multiplier. A gate-level audit also shows that the BME gate, as coded, is not bijective. Timing, power and a like-for-like comparison with the baseline remain future work.**

***Index Terms*—Reversible logic, Vedic mathematics, Urdhva Tiryakbhyam, arithmetic logic unit, HNG gate, Peres gate, Verilog HDL, FPGA**

I. Introduction

Every irreversible logic operation discards information, and Landauer showed that erasing one bit dissipates at least kT ln 2 of energy [1]. Bennett later showed that computation can be organized so that no information is erased, which removes this lower bound in principle [2]. A reversible gate has as many outputs as inputs and realizes a one-to-one mapping between them; fan-out and feedback are not allowed inside a purely reversible network [3], [4]. Circuits built from such gates therefore need extra constant inputs and produce garbage outputs whose count must be kept small.

The arithmetic logic unit (ALU) is the central datapath block of a processor, and its multiplier dominates both area and delay. The Urdhva Tiryakbhyam ("vertically and crosswise") sutra of Vedic mathematics [5] forms all partial products of a multiplication at the same time and combines them by column, which suits hardware realization. Several works combine Vedic multiplication with reversible logic, either as stand-alone multipliers [7]–[9] or inside an ALU [6], [10].

This paper describes a parameterized reversible Vedic ALU written in Verilog, with operand width W as a parameter. The architecture and gate library follow the design of Tiwari [10]; the present work is an independent implementation and verification of that architecture, together with a quantitative study of how its resources scale. The contributions are:

(i) a parameterized Verilog model of the ALU that is built only from the reversible gates of Section III (apart from the select logic of the logic unit) and that was exercised at 4 to 64 bits; (ii) a reversibility and functional audit, comprising a truth-table check of every gate, exhaustive verification of the multiplier up to 8×8 operands and a self-checking 64-bit ALU test; (iii) a closed-form gate-count expression for the recursive multiplier and LUT estimates for 4- to 64-bit instances; and (iv) a record of the points where the implementation departs from, or could not be reconciled with, the source design.

The rest of the paper is organized as follows. Section II reviews related work, Section III defines the gate library and cost metrics, Section IV describes the architecture, Section V explains the verification and synthesis setup, Section VI presents and discusses the results, and Section VII concludes.

II. Related Work

The thermodynamic motivation for reversible computing was established in [1] and [2]. Conservative logic and the Fredkin and Toffoli gates [3], the Peres gate [4], and Feynman's analysis of quantum computers [11] supply the gate vocabulary used by most later arithmetic designs. Thapliyal and Ranganathan argued that the number of reversible gates is an incomplete figure of merit because gates differ in quantum cost, and that delay and garbage outputs must also be optimized; they demonstrated this on reversible latches, flip-flops and a universal shift register verified through Verilog simulation [12].

For addition, Haghparast and Navi proposed the HNG gate, which realizes a complete reversible full adder in a single gate, and the HNFG gate for copying two bits [13]. For multiplication, Bhagyalakshmi and Venkatesha introduced the

International Journal of Engineering Science and Advanced Technology (IJESAT)
Vol 26 Issue 11, November 2026
Received: 05-10-2026 Accepted: 08-11-2026 Published: 16-11-2026

BVF gate to copy operand bits and reported that their multiplier needs 12 fan-out gates where earlier designs used 24 Feynman gates [14]. Thapliyal and Srinivas generated partial products in parallel with Fredkin gates and reduced the addition to $\log_2 N$ steps with a parallel adder built from TSG gates [15], and Shams, Haghparast and Navi built a 4×4 multiplier from MKG and Peres gates that can be generalized to $N \times N$ operands [16].

Reversible Vedic multipliers have recently emphasized parity preservation, which allows faults to be detected by comparing input and output parity. Rashno et al. proposed a parity-preserving reversible Vedic multiplier [7]. Noorallahzadeh et al. introduced three parity-preserving blocks and used them to build a 2-bit reversible Vedic multiplier and 4-bit ripple-carry adders, which were then combined into a 4-bit multiplier [8]. A later design of the same group offers a 2×2 parity-preserving quantum Vedic multiplier and 4-, 8- and 16-bit extensions [9]. At the ALU level, Jadhav et al. combined the speed of Vedic algorithms with application-specific reversibility [6], and Tiwari reported a generic low-power Vedic ALU on an Artix-7 FPGA that uses reversible gates, Vedic mathematics to shorten critical paths and reduce garbage outputs, and vinculum encoding, and that is scalable to wider operands [10].

The literature therefore offers many optimized multiplier blocks, mostly at 2×2 to 16×16 widths, and a smaller number of complete ALUs. What is less visible in the sources consulted for this study is an openly documented, parameterized RTL model of a complete reversible Vedic ALU that is checked gate by gate for reversibility, verified exhaustively at the block level, and characterized for how its resources grow with width. This paper does not claim a new ALU architecture; it addresses that documentation and verification gap for the architecture of [10].

III. Reversible Gate Library and Cost Metrics

A gate with n inputs and n outputs is reversible if its truth table is a bijection on $\{0,1\}^n$. The gates used in the design are listed in Table I together with the output equations as implemented in the RTL. Two of them need comment. First, the BVF gate is implemented as two Feynman gates, $(W,X)\rightarrow(W, W\oplus X)$ and $(Y,Z)\rightarrow(Y, Y\oplus Z)$, because with $X = Z = 0$ this form delivers two copies of each operand bit, which is the role the BVF gate plays in [14]; the equations as printed in [10] were found not to duplicate a signal. Second, the HNG gate computes a full adder when $Z = 0$:

$$S = A \oplus B \oplus C_{in}, \qquad C_{out} = (A \oplus B)\, C_{in} \oplus AB\,. \qquad (1)$$

TABLE I
Reversible gates used in the ALU (equations as implemented in the RTL)

Gate (size)	Output equations	Used for	Bijective
Feynman (2×2)	$A=X,\ B=X\oplus Y$	XOR, OR, b⊕sub	Yes (4/4)
Toffoli (3×3)	$A=X,\ B=Y,\ C=XY\oplus Z$	AND	Yes (8/8)
Peres (3×3)	$P=A,\ Q=A\oplus B,\ R=AB\oplus C$	Half adder	Yes (8/8)
BME (4×4)	$A=W,\ B=WX\oplus Y,\ C=WZ\oplus Y,\ D=W'X\oplus Y\oplus Z$	Partial products	No (8 of 16 outputs distinct)
BVF (4×4)	$A=W,\ B=W\oplus X,\ C=Y,\ D=Y\oplus Z$	Operand fan-out	Yes (16/16)
HNG (4×4)	$A=W,\ B=X,\ C=W\oplus X\oplus Y,\ D=(W\oplus X)Y\oplus(WX\oplus Z)$	Full adder (Z=0)	Yes (16/16)
BHA (3×3)	$P=A,\ Q=AB\oplus A'C,\ R=A'B\oplus AC$	2:1 select, $Q=A\ ?\ B:C$	Yes (8/8)

The usual cost metrics are the number of gates, the number of constant inputs, the number of garbage outputs and the quantum cost, together with the logic depth or delay [12], [17]. This paper reports the first three, counted directly from the RTL instantiations (Section VI-D). Quantum cost is not evaluated: the quantum costs of the BME, BVF and BHA gates are not available from the sources used here, and no value has been assumed. Delay is discussed analytically in Section VI-D, because no timing analysis was possible.

IV. Proposed Architecture

A. Top-Level ALU

The ALU of Fig. 1 takes two W-bit operands a and b, a 2-bit unit selector sel and a 4-bit function code $fcon$, and produces a $2W$-bit result y and a carry flag. Four units operate on the operands in parallel: an adder/subtractor, a Vedic multiplier and a logic unit, whose outputs are chosen by a reversible multiplexer, and a small decoder that derives the select signals. Add, subtract and logic results occupy the lower W bits of y and the upper half is zero. The control encoding is given in Table II. A registered wrapper (`rev_vedic_alu64_top`) adds input and output registers around the 64-bit instance so that FPGA tools have a clocked path to time.

2250-3676 ijesatj@gmail.com https://www.ijesat.com/

International Journal of Engineering Science and Advanced Technology (IJESAT)
Vol 26 Issue 11, November 2026
Received: 05-10-2026 Accepted: 08-11-2026 Published: 16-11-2026

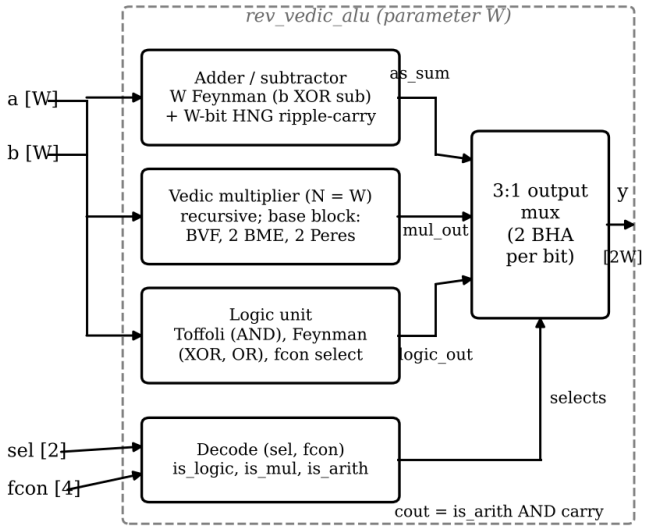


Fig. 1. Block diagram of the generic reversible Vedic ALU, drawn from the RTL (`rev_vedic_alu`).

TABLE II
CONTROL ENCODING IMPLEMENTED IN THE RTL

sel	fcon	Operation
00	1010	Multiply, $y=a\times b$
00	other, fcon[0] = 0	Add, $y=a+b$
00	other, fcon[0] = 1	Subtract, $y=a-b$
01	see Table III	Logic operation
10	any	Multiply, $y=a\times b$
11	fcon[0] = 0 / 1	Add / subtract

TABLE III
LOGIC-UNIT FUNCTION CODES (SEL = 01)

fcon	Function	fcon	Function	fcon	Function
0000	0	0110	$a \oplus b$	1010	$\bar{a}$
0001	$a \wedge b$	0111	$a \vee b$	1100	$\bar{b}$
0010	b	1000	NOR	1110	NAND
0101	a	1001	XNOR	1111	all ones

Codes not listed return 0.

The control encoding was reconciled in this implementation because the figure and the tables of the source design [10] do not agree with one another; the multiplier can therefore be selected either by *sel* = 10 or by *sel* = 00 with *fcon* = 1010.

B. Adder and Subtractor

Subtraction uses the identity $a - b = a + \bar{b} + 1$. A row of W Feynman gates forms $b \oplus sub$, where sub = *fcon*[0], and the result feeds a W-bit ripple-carry adder made of W HNG gates with carry-in sub. A carry-out of 1 during subtraction therefore means that no borrow occurred.

C. 2×2 Urdhva Tiryakbhyam Base Block

For two 2-bit operands the sutra gives the product bits

$$p_0 = a_0\, b_0, \quad p_1 = a_1\, b_0 \oplus a_0\, b_1, \quad c = a_1\, b_0 \cdot a_0\, b_1\,, \qquad (2)$$

$$p_2 = a_1\, b_1 \oplus c, \quad p_3 = a_1\, b_1 \cdot c\,. \qquad (3)$$

Fig. 2 shows the reversible realization. One BVF gate copies b_0 and b_1; two BME gates with a constant-0 input generate the four partial products; the first Peres gate acts as a half adder and delivers p_1 and the carry c; and a second Peres gate delivers p_2 and p_3. The source design uses a CNOT for the last stage; here a second Peres gate replaces it because p_3 is the AND of $a_1 b_1$ and the carry, which a CNOT alone cannot produce.

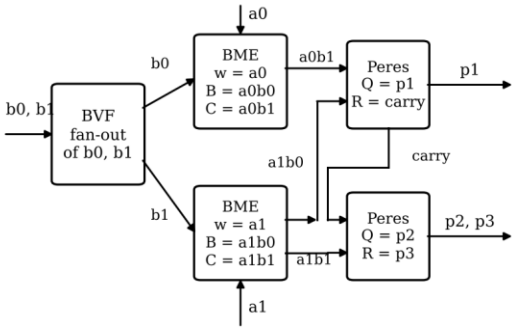


Fig. 2. 2×2 Urdhva Tiryakbhyam block as implemented (`ut2x2`): BVF, two BME and two Peres gates.

D. Recursive N×N Multiplier

An N-bit multiplication is split into four $(N/2)$-bit multiplications, with $A = \{A_H, A_L\}$ and $B = \{B_H, B_L\}$:

$$P = A_H\, B_H\ 2^N + (A_H\, B_L + A_L\, B_H)\ 2^{N/2} + A_L\, B_L\,. \qquad (4)$$

Fig. 3 gives the structure. The cross terms p_1 and p_2 are added by an N-bit HNG ripple-carry adder, and the upper half is formed by adding the sum to $\{p_3, p_0[N-1{:}N/2]\}$ in an $(N+N/2)$-bit HNG ripple-carry adder. The recursion ends at the block of Fig. 2, so the design is defined for N equal to a power of two. Because each level uses four sub-multipliers and $2.5N$ HNG gates, the multiplier gate count $G(N)$ satisfies

$$G(N) = 4\, G(N/2) + 5/2N, \quad G(2)=5 \quad \Rightarrow \quad G(N)=5/2N^2-5/2N\,. \qquad (5)$$

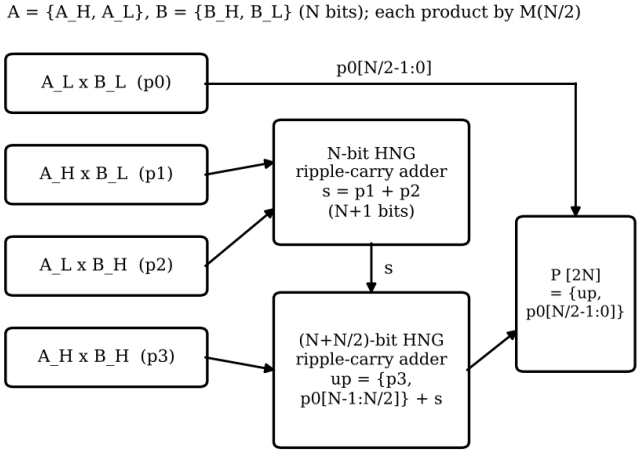


Fig. 3. Recursive structure of the $N\times N$ multiplier (`vedic_mult`).

2250-3676 ijesatj@gmail.com https://www.ijesat.com/

International Journal of Engineering Science and Advanced Technology (IJESAT)
Vol 26 Issue 11, November 2026
Received: 05-10-2026 Accepted: 08-11-2026 Published: 16-11-2026

E. Logic Unit and Output Multiplexer

The logic unit forms AND with a Toffoli gate whose target input is 0, XOR with a Feynman gate, and OR as $ab \oplus (a \oplus b)$ with a second Feynman gate. The remaining functions of Table III are chosen from these results and from a and b by a `case` statement on *fcon*, and the inversions use the Verilog complement operator; this select logic is behavioral and is not built from reversible gates. The output multiplexer uses two BHA gates per result bit. The first selects between the logic and arithmetic results and the second between the multiplier and the first stage, using the property $Q = A \ ? \ B : C$ of the BHA gate. The carry flag is the adder carry gated by the arithmetic-select signal.

V. Implementation and Verification Setup

The design is written in Verilog and consists of the gate library and the modules `rev_rca`, `ut2x2`, `vedic_mult`, `rev_vedic_alu` and the registered wrapper `rev_vedic_alu64_top`, with a self-checking testbench. Operand width is a parameter; the multiplier requires a power-of-two width. The default and the verified maximum is 64 bits.

All simulation results in this paper were produced with Icarus Verilog 12.0. A Vivado simulation project accompanies the source files, but the results of that run were not available, so no Vivado result is reported. Verification used three levels. First, the truth table of every gate was enumerated and tested for bijectivity. Second, the multiplier was compared with the Verilog product operator for all input pairs at 2×2, 4×4 and 8×8 widths and for 20,000 random pairs at 16×16. Third, a self-checking testbench applied 58 operand pairs to the 64-bit ALU (eight directed and boundary cases, including the examples of Section VI-B, and 50 random pairs) with 16 operation settings each, and compared the result y and the carry *cout* with behavioral expressions after every setting. A separate testbench exercised the registered wrapper to produce the waveforms of Fig. 4.

Area was estimated by technology mapping with Yosys 0.33 (`synth_xilinx`, 7-series, hierarchy flattened) of the unregistered ALU. This yields LUT counts but no placement, routing, timing or power, and it is not a substitute for a Vivado implementation.

VI. Results and Discussion

A. Gate Reversibility Audit

Exhaustive enumeration of the gates of Table I confirms that the Feynman, Toffoli, Peres, BVF, HNG and BHA gates are bijective. The BME gate, with the equations in Table I, is not: it produces only eight distinct output vectors for its sixteen inputs. For example, $(W,X,Y,Z) = (0,1,0,0)$ and $(0,0,0,1)$ both give $(A,B,C,D) = (0,0,0,1)$. The collision persists in the way the gate is used in the 2×2 block ($Y = 0$), where $(X,Z) = (0,1)$ and $(1,0)$ with $W = 0$ become indistinguishable. The two AND outputs used by the multiplier, B and C, are nevertheless correct, which is why the functional results below are unaffected. The origin of the discrepancy (the printed equations of [10] or their transcription) has not been established, and the 2×2 block cannot be called fully reversible until the BME definition is confirmed.

B. Functional Verification

TABLE IV
Functional verification results (Icarus Verilog 12.0)

Test	Vectors	Errors
2×2 multiplier, exhaustive	16	0
4×4 multiplier, exhaustive	256	0
8×8 multiplier, exhaustive	65,536	0
16×16 multiplier, random	20,000	0
64-bit ALU, 16 settings × 58 operand pairs	928	0

Table IV lists the outcome. Every test passed. The directed examples $9 \times 9 = 81$, $15 \times 15 = 225$ and $5 + 4 = 9$ also matched, and the maximum-times-maximum product, $(2^{64}\text{-}1)^2$, was returned correctly. Because the testbench uses behavioral operators as the reference, these results show that the gate-level network computes the intended functions; they do not by themselves prove that the network is reversible (Section VI-A).

C. Simulation Waveforms

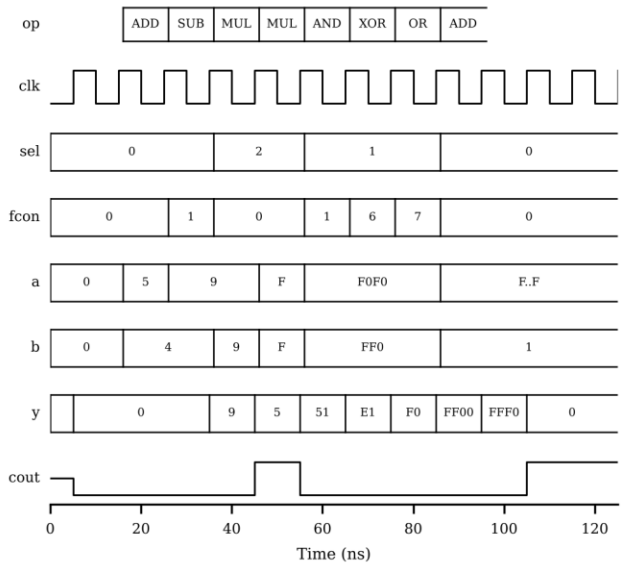


Fig. 4. Simulated waveforms of the registered 64-bit ALU (Icarus Verilog, zero-delay RTL; values in hexadecimal, 64-bit all-ones abbreviated F..F). Row *op* marks the stimulus applied to the inputs.

Fig. 4 shows the registered wrapper executing add, subtract, two multiplications and the AND, XOR and OR functions, followed by an add that overflows. The results 9, 5, 51 (= 81), E1 (= 225), F0, FF00 and FFF0 appear at y two clock cycles after the corresponding inputs, which reflects the input and output registers, and *cout* is 1 for the subtraction (no borrow)

2250-3676 ijesatj@gmail.com https://www.ijesat.com/

International Journal of Engineering Science and Advanced Technology (IJESAT)
Vol 26 Issue 11, November 2026
Received: 05-10-2026 Accepted: 08-11-2026 Published: 16-11-2026

and for the overflowing addition. Since the RTL is simulated with zero gate delay, the waveform shows function and latency but not propagation delay.

D. Resource Counts and Scaling

TABLE V
REVERSIBLE-GATE RESOURCES COUNTED FROM THE RTL AND YOSYS 7-SERIES LUT ESTIMATE

W	Multiplier gates	ALU gates	Constant inputs	Unused outputs	LUTs
4	30	66	52	104	41
8	140	212	198	336	159
16	600	744	770	1,184	672
32	2,480	2,768	3,034	4,416	2,657
64	10,080	10,656	12,042	17,024	11,430

Gates, constant inputs and unused outputs are for the ALU unless stated; the logic-unit select logic is behavioral and not counted. LUTs: Yosys 0.33, flattened ALU.

Table V gives the resources. The multiplier gate counts agree exactly with the closed form of Section IV-D: for $W = 64$, $2.5 \times 64^2 - 2.5 \times 64 = 10{,}080$ gates, which accounts for about 95% of the 10,656 gates of the ALU. The LUT estimate grows by a factor between 3.9 and 4.3 for each doubling of W (Fig. 5), so area scales roughly with the square of the operand width. This differs from the sub-quadratic growth one might expect from a divide-and-conquer multiplier: the recursion of Section IV-D keeps four sub-products per level, whereas Karatsuba-type schemes need only three [18]. The statement that a hierarchical Vedic multiplier of this kind has area of order $N^{1.585}$ therefore does not apply to the present implementation.

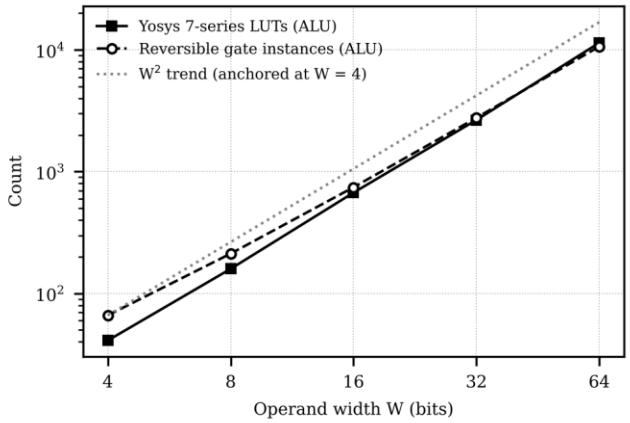


Fig. 5. Growth of the reversible gate count (counted) and of the Yosys LUT estimate (mapped) with operand width.

The same structure bears on delay. Both adders of each recursion level are ripple-carry chains whose length grows with N, and the chains are in series with the sub-multipliers, so the critical path is expected to grow roughly linearly with N rather than logarithmically. This is an analytical expectation only; no timing analysis was performed.

E. Comparison with the Baseline Design

TABLE VI
QUALITATIVE COMPARISON WITH THE SOURCE DESIGN [10]

Aspect	Source design [10]	This work
Platform	Artix-7 FPGA (as reported)	RTL simulation (Icarus); Yosys mapping to 7-series LUTs, no place and route
Arithmetic basis	Vedic multiplication, reversible gates, vinculum encoding (as reported)	Urdhva Tiryakbhyam, reversible gates; no vinculum encoding
Scalability	Higher bit widths (as reported)	Parameter W; verified 4 to 64 bits
Control encoding	Figure and tables found inconsistent	Reconciled (Table II)
2×2 final stage	CNOT	Second Peres gate
Delay, power, area figures	Reported in [10]	Not obtained; Vivado needed

No numerical comparison is made.

Table VI compares the two designs qualitatively. No numerical comparison is attempted. The figures of [10] come from a different tool flow and device, and a fair comparison requires implementing both designs under the same constraints in the same tool; reporting the two sets of numbers side by side would risk presenting an indicative comparison as a direct one.

F. Limitations

The study has several limits. The BME gate is not bijective as coded. The logic unit selects its function with behavioral Verilog. Delay and power are not measured. Quantum cost is not computed. Area is a Yosys estimate and not a Vivado result. The thesis-level description of this project contains a shifter and a control state machine; neither is part of the RTL verified here. Finally, the reversibility audit is structural, so a passing functional test cannot replace a proof that the whole network is reversible.

VII. CONCLUSION AND FUTURE WORK

A parameterized reversible Vedic ALU built from HNG, BVF, BME, Peres, Toffoli, Feynman and BHA gates was implemented and verified in simulation. The 64-bit instance passed all 928 self-checking tests, and the multiplier passed exhaustive tests through 8×8 operands. The multiplier gate count follows $2.5N^2 - 2.5N$ and the mapped area grows roughly fourfold per doubling of width. The audit found that the BME gate, as coded, is not reversible and that the control encoding of the source design needed reconciliation.

Future work should (i) confirm the BME definition against the original gate and correct the 2×2 block if needed, (ii) run Vivado implementation on Artix-7 to obtain timing, power and utilization, (iii) compute quantum cost and garbage outputs under an agreed convention, (iv) replace the ripple-carry adders by faster reversible adders to test the expected delay reduction, and (v) implement the baseline of [10] under the same conditions for a direct comparison.

2250-3676 ijesatj@gmail.com https://www.ijesat.com/

Received: 05-10-2026 Accepted: 08-11-2026 Published: 16-11-2026

REFERENCES

[1] R. Landauer, "Irreversibility and heat generation in the computing process," *IBM J. Res. Dev.*, vol. 5, no. 3, pp. 183–191, Jul. 1961.

[2] C. H. Bennett, "Logical reversibility of computation," *IBM J. Res. Dev.*, vol. 17, no. 6, pp. 525–532, Nov. 1973, doi: 10.1147/rd.176.0525.

[3] E. Fredkin and T. Toffoli, "Conservative logic," *Int. J. Theor. Phys.*, vol. 21, nos. 3–4, pp. 219–253, 1982.

[4] A. Peres, "Reversible logic and quantum computers," *Phys. Rev. A*, vol. 32, no. 6, pp. 3266–3276, Dec. 1985, doi: 10.1103/PhysRevA.32.3266.

[5] Bharati Krishna Tirthaji Maharaja, *Vedic Mathematics: Sixteen Simple Mathematical Formulae from the Vedas*. Delhi, India: Motilal Banarsidass, 1965.

[6] K. Jadhav, A. Vibhute, S. Iyer, and R. Dhanabal, "Novel Vedic mathematics based ALU using application specific reversibility," in *Proc. IEEE 9th Int. Conf. Intell. Syst. Control (ISCO)*, 2015, doi: 10.1109/ISCO.2015.7282231.

[7] M. Rashno, M. Haghparast, and M. Mosleh, "Designing of parity preserving reversible Vedic multiplier," *Int. J. Theor. Phys.*, vol. 60, no. 8, pp. 3024–3040, 2021, doi: 10.1007/s10773-021-04903-z.

[8] M. Noorallahzadeh, M. Mosleh, S.-S. Ahmadpour, J. Pal, and B. Sen, "A new design of parity preserving reversible Vedic multiplier targeting emerging quantum circuits," *Int. J. Numer. Model.: Electron. Netw. Devices Fields*, vol. 36, no. 5, 2023, Art. no. e3089.

[9] M. Noorallahzadeh and M. Mosleh, "A novel and ultra-efficient design of reversible quantum Vedic multiplier," *Quantum Inf. Comput.*, vol. 25, pp. 552–567, 2025, doi: 10.2478/qic-2025-0030.

[10] K. S. Tiwari, "Design of generic vedic ALU using reversible logic," *Memories – Mater. Devices Circuits Syst.*, vol. 9, Art. no. 100121, Apr. 2025, doi: 10.1016/j.memori.2025.100121.

[11] R. P. Feynman, "Quantum mechanical computers," *Optics News*, vol. 11, pp. 11–20, 1985.

[12] H. Thapliyal and N. Ranganathan, "Design of reversible sequential circuits optimizing quantum cost, delay, and garbage outputs," *ACM J. Emerg. Technol. Comput. Syst.*, vol. 6, no. 4, Art. 14, Dec. 2010, doi: 10.1145/1877745.1877748.

[13] M. Haghparast and K. Navi, "A novel reversible BCD adder for nanotechnology based systems," *Am. J. Appl. Sci.*, vol. 5, no. 3, pp. 282–288, 2008, doi: 10.3844/ajassp.2008.282.288.

[14] H. R. Bhagyalakshmi and M. K. Venkatesha, "An improved design of a multiplier using reversible logic gates," *Int. J. Eng. Sci. Technol.*, vol. 2, no. 8, pp. 3838–3845, 2010.

[15] H. Thapliyal and M. B. Srinivas, "Novel reversible multiplier architecture using reversible TSG gate," arXiv:cs/0605004.

[16] M. Shams, M. Haghparast, and K. Navi, "Novel reversible multiplier circuit in nanotechnology," *World Appl. Sci. J.*, vol. 3, no. 5, pp. 806–810, 2008.

[17] M. Perkowski and P. Kerntopf, "Reversible logic," invited tutorial, in *Proc. EURO-MICRO*, Warsaw, Poland, Sep. 2001.

[18] A. Karatsuba and Y. Ofman, "Multiplication of multidigit numbers on automata," *Sov. Phys. Dokl.*, vol. 7, pp. 595–596, 1963.