FACTA UNIVERSITATIS
Series: **Electronics and Energetics** Vol. 39, N° 1, March 2026, pp. 137 - 148
https://doi.org/10.2298/FUEE2601137K

Original scientific paper

SIGNED MULTIPLICATION ARCHITECTURE USING VEDIC URDHVA TIRYAGBHYAM ALGORITHM

Dharamvir Kumar[1], Manoranjan Pradhan[1], Neeraj Kumar Misra[2], Bandan Kumar Bhoi[1]

[1]Department of Electronics and Telecommunication,
Veer Surendra Sai University of Technology, Burla, Sambalpur 768018, Odisha, India
[2]School of Electronics Engineering, VIT-AP University, Amaravati,
Andhra Pradesh 522237, India

ORCID iD:	Dharamvir Kumar	https://orcid.org/0000-0001-8941-3917
	Manoranjan Pradhan	https://orcid.org/0000-0003-4520-5280
	Neeraj Kumar Misra	https://orcid.org/0000-0002-7907-0276
	Bandan Kumar Bhoi	https://orcid.org/0000-0003-2916-2903

Abstract. *This paper introduces a novel architecture for signed multiplication, representing a significant advancement in VLSI design for high-end computation. We have implemented signed number multiplier numbers using the Urdhva Tiryagbhyam (UT) algorithm. The purpose of the proposed architecture is to overcome the limitations of previous approaches, through the integration of Virtex-7 and Spartan-7 hardware, in addition to improving power and delay performance. As a result of using vertical and crosswise techniques, the proposed parallel multiplication scheme for signed numbers is implemented in this paper. We have synthesized and simulated this structure using Vivado 2020.1 software and Virtex-7 and Spartan-7 FPGA devices for low-power VLSI computational applications, such as checking the speed and delay of the design post-layout. A comparison is made between the proposed design and existing architectures in terms of area, delay, and power consumption. According to the results, the proposed architecture improves speed by 40% over prior architectures. For faster computing applications, a high-speed proposed multiplier will be useful for complex design architecture. The work contributes significantly to the improvement of high-performance computing and digital signal processing.*

Key words: *Computational, Vedic multiplier, High performance computing, FPGA, System design; urdhva tiryagbhyam*

Received April 19, 2025; revised June 22, 2025; accepted July 29, 2025
Corresponding author: Neeraj Kumar Misra
School of Electronics Engineering, VIT-AP University, Amaravati, Andhra Pradesh 522237, India
E-mail: neeraj.misra@vitap.ac.in

1. INTRODUCTION

In VLSI system design, multipliers are the basic building blocks for many operations such as filtering and microprocessor design [1]. The multiplier block is used in the convolution process when different signals are filtered through it. An integer multiplier block is used for large integer multiplicities in cases of encryption and decryption of information in applications that use public key cryptography to protect sensitive information. Further, the performance of the processor is determined by the speed of the multiplication operation. Most of the processors have included multiplier blocks as part of their processor architectures. Most of the research targets low power multiplier design and improves the performance parameters such as the area, computational delay, and power consumption.

In Vedic mathematics, there are sixteen sutras, including the UT algorithm [2]. The authors [3] and [4] have compared both the Nikhilam and Urdhva Tiryagbhyam algorithms and claim that the Nikhilam multiplier is faster. In addition, [5], [6], [7], and [8] provide efficient Vedic multiplier implementations. In spite of this, their works are restricted to unsigned numbers. In addition, there is no discussion of signed implementation. Many real-time systems and image-processing applications can be implemented using the Vedic method presented in [9]. In comparison to array multipliers and booth multipliers, the design is faster [9]. However, there has been no discussion of how signed numbers will be implemented. The authors of [10] demonstrate how their squaring algorithm improved in terms of area and speed by implementing the squaring technique. Their structure, however, is limited to unsigned numbers. In [11], authors used compressors to reduce the combinational delay time of their proposed multiplier and claimed better speed over conventional designs. A wide variety of research has been done on adders and multipliers, such as [12], [13], [14], [15] and [16]. Multiplier and square implementations as [12] and [13] are useful for unsigned operands. In previous work [14], the Vedic algorithm was used to implement efficient ASIC and FPGA cube architectures. In addition to signed numbers, this implementation can also be used with unsigned numbers. There is, however, a redundant binary number system used in the implementation of signed multi-pliers in [15]. The Nikhilam algorithm was used in [16] for the computation of signed multipliers. In [17], the UT method is used to design a Vedic multiplier. Based on the Zynq 7000 FPGA board, the authors of [18] implemented a 32-bit Wallace multiplier and a 32-bit systolic multiplier. However, there is no discussion of the realization of signed operands. The authors of [19] have also used both unsigned-signed and signed-unsigned converters to support signed implementation. Their paper discusses the use of quantum-dot cellular automata (QCA) technology to implement a two-bit Vedic multiplier. In this paper, the UT algorithm is used to design a signed multiplication architecture. In addition to overcoming previous limitations, the proposed architecture also improves performance.

This research paper contributions include:

- We investigate the design of a signed multiplier utilizing the UT algorithm that offers speed improvement to the prior reported design.
- Parallel generation of vertical and sequential partial products, we eliminate the need for sequential calculation of partial products, which reduces the overall combinational path delay of the multiplier.
- Parallel addition of partial products: The partial products are added in parallel which also provides significant critical path delay reduction.
- Novel Architecture for Signed Multiplier: The contributions in the context of Vedic multiplication algorithms, represent significant advances in the field of signed multiplication. This paper introduces a novel architecture for signed multiplication,

representing a key advancement in the field of signed multiplication. Using the UT algorithm and extending it to handle signed numbers, the proposed architecture not only overcomes previous limitations but also improves performance.

- Parallel Generation of Vertical and Sequential Partial Products: One noteworthy improvement is the removal of the requirement for the sequential calculation of partial products. The suggested architecture lowers the multiplier's overall combinational path latency by permitting the concurrent creation of vertical and sequential partial products. This method of parallel processing improves throughput and computational efficiency.
- Parallel Addition of Partial Products: The parallel addition of partial products makes an additional important contribution. The suggested architecture delivers notable reductions in critical path delay by adding partial products in parallel. The multiplier's overall performance is significantly improved by this parallel processing strategy.

In the context of FPGA-based design, this proposed methodology can provide advantages such as:

1. Resource Efficiency: A UT algorithm optimizes FPGA resources for high-performance multiplication while making better use of FPGA resources.
2. High-Speed Operation: A FPGA-based system allows fast multiplication operations, which are essential for signal processing, cryptography, and digital signal processing.
3. Low Power Consumption: In many applications, particularly those with portable or battery-powered components, efficient multiplication algorithms reduce power consumption.

The rest of the paper is divided into sections. Section 2 shows UT algorithm basic and examples for signed multiplication architecture. Section 3 shows the proposed architecture of signed multiplier. A comparison and implementation results are presented in section 4. In section 5, a conclusion is drawn.

2. URDHVA TIRYAGBHYAM ALGORITHM

There are two classes of numerical values in computer science and mathematics: unsigned and signed numbers. The following is a brief summary of each classification:

There are only non-negative values that can be represented using unsigned numbers when they are unsigned. In this case, the range begins at zero and continues exclusively with positive values; the sign bit is not included in their range. An example of a binary encoding would be the use of all available bits to indicate the magnitude of the number; no bit is set aside to indicate the sign of the number in binary encoding. For an 8-bit unsigned integer include 0, 1, 2, 3..., 255.

Signed Numbers have the ability to represent both positive and negative values. They have a sign bit in them that establishes the number's sign. Usually, the sign is indicated in the leftmost bit, which is also the most crucial bit. The representation: Binary representation uses the two's complement representation to describe the magnitude of the number, with the leftmost bit serving as the sign bit (0 for positive, 1 for negative). For an 8-bit signed integer, the values are -128, -127, -2, -1, 0, 1, 2, 127. In comparison, when compared to signed numbers with the same number of bits, unsigned numbers have a simpler representation and can represent a larger positive range. A signed number can represent either positive or negative values, making it more versatile but also requiring specific representations and arithmetic considerations. The use of unsigned numbers is common when only non-negative values are relevant, such as when representing quantities, indices, or measurements. When positive and negative values need to

be represented, such as in arithmetic calculations or financial transactions, signed numbers are used.

The meaning of the UT algorithm is vertical and crosswise technique. The proposed architecture computes vertical and crosswise partial products in parallel. Further, partial products are also added in parallel to provide significant critical path delay reduction. For signed multiplication utilizing an unsigned multiplier, multiplicand, and multiplier are converted to 2's complement form. The product can be either unsigned or signed form. For signed products, it is further converted using 2's complement method to unsigned form. So, two additional converters, one for unsigned-signed and the other for signed-unsigned are required which increases the critical path delay and area of signed multiplier. Further, most of the earlier reported Vedic multipliers are based on unsigned numbers. In this study, we have tried to extend the proposed architecture for signed numbers.

By extending a Vedic multiplier architecture originally intended for unsigned numbers to support signed integers, additional complexity is introduced in terms of conversion logic, critical route time, and area utilization. Although these difficulties can be overcome, it requires careful consideration and trade-offs to ensure effective operation that supports both signed and unsigned arithmetic.

2.1. Equations for UT algorithm

The product of two numbers $ax + b$ and $cx + d$ i.e., $x^2ac + x(ad + bc) + bd$ is derived from the UT algorithm as:

1. The coefficient of x^2 is the vertical multiplication of a and c.
2. The coefficient of x is the by the crosswise multiplication of a with d and b with c, along with an addition operation of both products.
3. The constant term is the vertical multiplication of b with d.

The mathematical formula for The Urdhva Tiryagbhyam algorithm for 2-digit and 3-digit multiplication are shown in equations 1 and 2 respectively. Where x represents as base.

For two digits numbers, Let X=ax+b and Y=cx+d, So, multiplication of X and Y

$$P = X \times Y = (ax + b) \times (cx + d)$$

$$= ax(cx + d) + b(cx + d)$$

$$= acx^2 + adx + bcx + bd$$

$$= acx^2 + (ad + bc)x + bd \quad (1)$$

Similarly, for three digits numbers, Let $X = (a_1x^2 + b_1x + c_1)$ and $Y = (a_2x^2 + b_2x + c_2)$, multiplication of X and Y

$$P = X \times Y = (a_1x^2 + b_1x + c_1) * (a_2x^2 + b_2x + c_2)$$

$$= a_1x^2(a_2x^2 + b_2x + c_2) + b_1x(a_2x^2 + b_2x + c_2) + c_1(a_2x^2 + b_2x + c_2)$$

$$= (a_1a_2)x^4 + (a_1b_2x^3 + a_2b_1x^3) + (a_1c_2x^2 + b_1b_2x^2 + a_2c_1x^2) + (b_1c_2x + b_2c_1x) + c_1c_2$$

$$= x^4(a_1a_2) + x^3(a_1b_2 + a_2b_1) + x^2(a_1c_2 + b_1b_2 + a_2c_1) + x(b_1c_2 + b_2c_1) + c_1c_2 \quad (2)$$

The decimal UT algorithm for two-digit numbers X=ax+b and Y=cx+d and Their product XY=P3P2P1 can be represented in the following steps:
Step 1: The base (x) is taken as 10.
Step 2: P1 (Vertical multiplication) =b x d.
Step 3: P2 (Crosswise multiplication and addition) = a x d + b x c.
Step 4: P3 (Vertical multiplication) =a x c.
Step 5: Final Product P=X×Y = Concatenation of P3, P2 and P1=P3P2P1

Initially, base (radix) is taken as (10)10. The final product consists of the concatenation of P3, P2, and P1. P1 is computed by vertical multiplication of b and d. The addition of cross multiplication of a with d and b with c is P2. Similarly, P3 is the vertical multiplication of a with c. Final product P is the concatenation of P3, P2 and P1.

Let the two-digit numbers are X=12=1x10 +2 and Y=13= 1x10 +3

1. For the decimal UT algorithm, the base (x) is taken as 10.
2. P1 (Vertical multiplication) =b x d= 2 x 3= 6
3. P2 (Crosswise multiplication and addition) = a x d + b x c = 1x3 +2x1=5
4. P3 (Vertical multiplication) =a x c= 1x1 = 1
5. Final Product P=X×Y = Concatenation of P3, P2 and P1=P3P2P1
   = (1,5,6)
   =156

The base is taken as (10) 10. Unit digit 2 of number X is vertically multiplied with the unit digit 3 of number Y to get partial product 6 (P1). The cross multiplication of the tenth digit 1 of number X with unit digit 3 of number Y is added with the cross multiplication of unit digit 2 of number X with tenth digit 1 of number Y to get partial product 5 (P2). Similarly, the tenth digit 1 of number X is vertically multiplied by the tenth digit 1 of number Y to get partial product 1(P3). Final product 156 (P) is the concatenation of 1(P3), 5 (P2) and 6 (P1).

Let the two-digit numbers are X=12=1x10 +2 and Y=-13= (-1) x10 +(-3)

1. For the signed decimal UT algorithm, the base (x) is taken as 10.
2. P1 (Vertical signed multiplication) =b x d= 2 x (-3) = -6
3. P2 (Crosswise signed multiplication and addition) = a x d + b x c = 1x (-3) +2x (-1) = -5
4. P3 (Vertical signed multiplication) =a x c= 1x (-1) = -1
5. Final Product P= X×Y = Concatenation of P3, P2 and P1=P3P2P1
   = (-1, -5, -6)
   =-156

The base is taken as (10) 10. Unit digit 2 of number X is vertically multiplied with unit digit -3 of number Y to get partial product -6 (P1). The cross multiplication of the tenth digit 1 of number X with unit digit -3 of number Y is added with the cross multiplication of unit digit 2 of number X with tenth digit -1 of number Y to get partial product -5 (P2). Similarly, the tenth digit 1 of number X is vertically multiplied with the tenth digit -1 of number Y to get partial product -1(P3). The final product -156 (P) is the concatenation of -1 (P3), -5 (P2) and -6 (P1).

2.2. Conventional 2's complement signed binary number

Table 1 shows the signed multiplication of two binary numbers $(12)_{10}$ and $(-13)_{10}$ using the traditional 2's complement multiplication method. Here 12 is an unsigned number and

(-13) is a signed number, so 2's complement of (-13) is $(10011)_2$. So $(12)_{10} = (0\ 1\ 1\ 0\ 0)_2$ and $(-13)_{10} = (1\ 0\ 0\ 1\ 1)_2$

Table 1 Signed multiplication using conventional 2's complement method

Multiplicand = 01100= $(12)_{10}$							0	1	1	0	0
Multiplier = 10011= $(-13)_{10}$							1	0	0	1	1
Partial product 1	0	0	0	0	0	0	0	1	1	0	0
Partial product 2	0	0	0	0	0	0	1	1	0	0	
Partial product 3	0	0	0	0	0	0	0	0	0		
Partial product 4	0	0	0	0	0	0	0	0			
Partial product 5	1	1	1	1	0	1	0	0			
Multiplication result $(-156)_{10}$	1	1	1	0	1	1	0	0	1	0	0

Multiplication result of $(12)_{10}$ x $(-13)_{10} = (-156)_{10}$. We get the signed multiplication result by converting the signed number using 2's complement method. The binary value of $(12)_{10}$ is $(1\ 1\ 0\ 0)_2$, as it is a signed multiplication, we can write it as $(0\ 1100)_2$. The binary value of $(13)_{10}$ is $(1\ 1\ 0\ 1)_2$, for the signed number we have to take the 2's complement of $(1\ 1\ 0\ 1)_2$ which we can write as $(-13)_{10} = (1\ 0011)_2$. Here 0 and 1 are used as sign extensions for positive and negative signs respectively. After doing the multiplication we get the product as $(-156)_{10} = (1\ 01100100)_2$. There are five partial products with sign extension for the negative partial product (pp5). The final product is the addition of five partial products.

2.3. The Proposed signed multiplication using the UT algorithm

In Figure 1 also base is taken as $(10000)_2$. The 8-bit multiplicand X is represented by the least significant 4-bit binary number b = $(0010)_2$ and the most significant 4-bit binary number a = $(0001)_2$. Similarly, the 8-bit multiplier Y is represented by the least significant 4-bit binary number d = $(1101)_2$ and the most significant 4-bit binary number c = $(1111)_2$.

The least significant 4-bit binary number $(0010)_2$ of number X is vertically multiplied with the least significant 4-bit binary number $(1101)_2$ of number Y to get the 8-bit partial product $(11111010)_2$ (P1). However, the number of binary digits of P1 should be four as there are four zeroes in the base of the proposed algorithm. So P1 becomes $(1010)_2$ considering 4-bit including sign bit.

The cross multiplication of $(0001)_2$ of number X with $(1101)_2$ of number Y is added with cross multiplication $(0010)_2$ of number X with $(1111)_2$ of number Y to get the 8-bit partial product $(11111011)_2$ (P2). As per the proposed algorithm, P2 also becomes $(1011)_2$ considering 4-bit including sign bit.

Similarly, the most significant 4-bit binary number $(0010)_2$ of number X is vertically multiplied with the most significant 4-bit binary number $(1101)_2$ of number Y to get the 8-bit partial product $(11111111)_2$ (P3). The final 16-bit product P is the concatenation of 8-bit P3, 4-bit P2, and 4-bit P1.

Example1: of the proposed signed multiplication using UT algorithm
Multiplicand X (8-bit binary) = ab: a (4-bit binary) = $(0001)_2 = (1)_{10}$ and b (4-bit binary) = $(0010)_2 = (2)_{10}$
Multiplier Y (8-bit binary) =cd: c (4-bit binary) = $(1111)_2 = (-1)_{10}$ and d (4-bit binary) = $(1101)_2 = (-3)_{10}$
Product: P (16-bit binary) = $(11111111\ 1011\ 1010)_2$
Intermediate parts = P1 (4-bit binary) = $(1010)_2 = (-6)_{10}$ //Right Part of Result
P2 (4-bit binary) = $(1011)_2 = (-5)_{10}$ //Middle Part of Result
P3 (8-bit binary) = $(11111111)_2 = (-1)_{10}$ //Left part of Result

1. For the 4-bit signed binary Urdhva Tiryagbhyam algorithm, the base (x) is taken as $(1000)_2$.
2. P1 (Vertical signed multiplication) =b x d
   $= (0010)_2 \times (1101)_2$
   $= (11111010)_2$
   $= (1010)_2$ //considering 4-bit including sign bit
3. P2 (Crosswise signed multiplication and addition) = a x d + b x c
   $= \{(0001)_2 \times (1101)_2 + \{(0010)_2 \times (1111)_2\}$
   $= (11111101)_2 + (11111110)_2$
   $= (11111011)_2$
   $= (1011)_2$ //considering 4-bit including sign bit
4. P3 (Vertical signed multiplication) =a x c
   $= (0010)_2 \times (1101)_2$
   $= (11111111)_2$
5. Final Product P= X×Y = Concatenation of P3, P2 and P1= (P3, P2, P1)
   $= (11111111,1011,1010)_2$

3. The Proposed Architecture

The proposed 8x8 signed binary multiplier architecture using the Vedic UT algorithm is shown in Figure 1. The Proposed multiplier architecture is implemented using four numbers of 4x4 signed multiplier blocks (4x4 SM1, 4x4 SM2, 4x4 SM3 and 4x4 SM4) and one parallel adder block. The 4x4 SM block is implemented using the signed product of a 4-bit multiplicand with a 4-bit multiplier. The two 8-bit inputs of the proposed architecture are X (A [4:1] B [4:1]) and Y (C [4:1] D [4:1]). The 16-bit output is P3[8:1] P2[4:1] P1[4:1].

The 8-bit multiplicand X is represented by the least significant 4-bit binary number B [4:1] and the most significant 4-bit binary number A [4:1]. Similarly, the 8-bit multiplier Y is represented by the least significant 4-bit binary number D [4:1] and the most significant 4-bit binary number C [4:1].

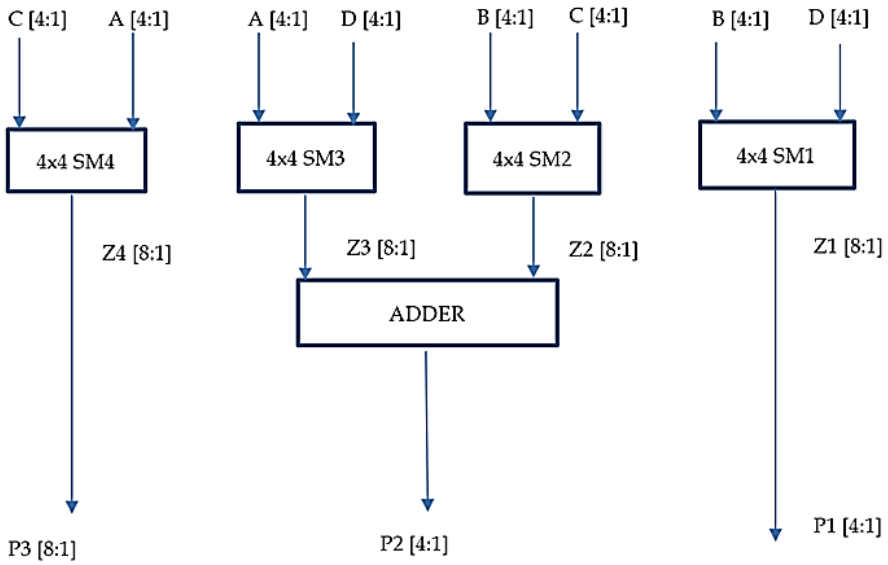

Fig. 1 Proposed signed multiplication architecture using UT algorithm

The least significant 4-bit binary number B [4:1] of number X is vertically multiplied with the least significant 4-bit binary number D [4:1] of number Y using 4x4 SM1 to get 8-bit partial product Z1 [8:1]. However, the number of binary digits of P1 should be four as there are four zeroes in the base of the proposed algorithm. So P1 [4:1] becomes Z1 [4;1] considering the 4-bit result including the sign bit. The most significant 4-bits of Z1 become carry inputs to the parallel adder. The cross-multiplication product Z2 [8:1] between B [4:1] with C [4:1] is implemented using 4x4 SM2.Similarly, cross multiplication product Z3 [8:1] between A [4:1] with D [4:1] is realized using 4x4 SM3. Further Z2 [8:1], and Z3 [8:1] are added using a parallel adder to get sum P2 [4:1]. As per the proposed algorithm, the least significant 4-bit sum becomes P2 [4:1] considering the 4-bit result including the sign bit. Similarly, the most significant 4-bit binary number A [4:1] is vertically multiplied by the most significant 4-bit binary number C [4:1] using 4x4 SM4 to get an 8-bit partial product Z4 [8:1]. Z4 [8:1] is equal to P3 [8:1]. The final 16-bit product P is the concatenation of 8-bit P3, 4-bit P2, and 4-bit P1.

Similarly,16x16,32x32, and 64x64 signed binary multiplier architectures using the proposed design are implemented accordingly.

4. Result and Discussion

The proposed architecture is coded using Verilog HDL. The functionality of the design is verified using the Vivado 2020.1 simulator tool. Synthesis and implementation have been done using Vivado 20.1 software tools for different Virtex-7 and Spartan-7 FPGA devices for comparison. After the implementation area, power and delay are obtained from the utilization report, power report, and timing report respectively. FPGA is an integrated circuit consisting of a configuration logic block, input-output block, and vertical and horizontal routing channels.

Configuration logic blocks include lookup tables (LUTs), associated carry chains, multiplexers, and flip-flops.DSP slices in an FPGA are specialized blocks designed to perform mathematical operations efficiently in filtering and digital signal processing applications. The proposed signed multiplier has been compared with the implementations reported in [9],[10] and [15].

Table 2 shows the area, power, and delay comparison of the proposed signed multiplier in Virtex-7 device vlx15sf363-12 with different state-of-the-art multipliers and square modules reported in [9], [10], and [15]. As shown in the table, our proposed signed multiplier provides higher speed performance than the state-of-the-art multiplier and square unit presented in [9], [10], and [15]. For example, our proposed 8x8 signed multiplier reduces critical path delay by 35%,13%,93%, and 81% as compared to the design presented in [9], [10] and [15] respectively.

It can be observed from the table, except for 8x8 size, our proposed design requires a smaller number of LUTs than other state-of-the-art different bit size multipliers. For example, our 16x16 proposed implementation requires only 182 4-input LUTs whereas 294 4-input LUTs are required in Vedic square unit presented in [9].

Table 3 depicts the delay comparison of the proposed signed multiplier in Spartan-7 device with different state-of-the-art multipliers reported in [10],[11], and [15]. As shown in the table, our proposed signed multiplier provides higher speed performance than state of the art multiplier presented in [10],[11], and [15]. For example, our proposed 16x16 signed multiplier reduces critical path delay by 59%,52%and 49% as compared to the design presented in [10],[11] and [15] respectively.

Figure 2 also presents the graphical representation of the critical path delay of our proposed design with other implementations in [10],[11] and [15]. It is the graphical representation of Table 3. The differences in the bars of our design with other schemes show a clear indication of performance improvement. Our signed multiplier outperforms other state-of-the-art designs in delay metric.

Table 2 Area, power, and delay comparison for the proposed multiplier

Bit size	Performance parameter	Vedic squaring unit [9]	Booth multiplier [9]	Squaring circuit [10]	Signed vedic multiplier [15]	vedic multiplier [21]	Proposed signed multiplier
8x8	4-input LUTs	35	186	49	--	51	0.25%
	Power, W						
	Delay, ns	14.256	15.718	8.948	15.22	15.45	10.835
		--	--	--	--	---	13.437
% Improvement w.r.to Power		23.99	31.06	NI	28.81	42.59	
16x16	4-input LUTs	294	880	233	--	240	1.25%
	Power, W						
	Delay, ns	33.391	36.657	18.564	19.58	18.45	28.786
		--	--	---	---	--	15.291
% Improvement w.r.to Power		13.79	21.47	NI	NI	NI	
32x32	4-input LUTs	1034	2760	985	--	842	5%
	Power, W						
	Delay, ns	68.125	74.432	37.345	22.32	21.56	62.024
		---	---	---	---	---	17.909
% Improvement w.r.to Power		8.955	16.67	NI	NI	NI	

Table 3 Delay comparison for the proposed multiplier in Spartan-7 device

16x16 bit multiplier	Signed/ Unsigned multiplier	Combinational delay (ns)	Percentage of improvement (%)
Array multiplier in [11]	Unsigned	43.946	65
Booth multiplier in [11]	Signed	37.041	58
Vedic multiplier in [10]	Unsigned	37.50	59
Vedic multiplier using compressor adder in [11]	Unsigned	32	52
Signed vedic multiplier in [15]	Signed	30.16	49
Proposed signed multiplier	Signed	15.291	---

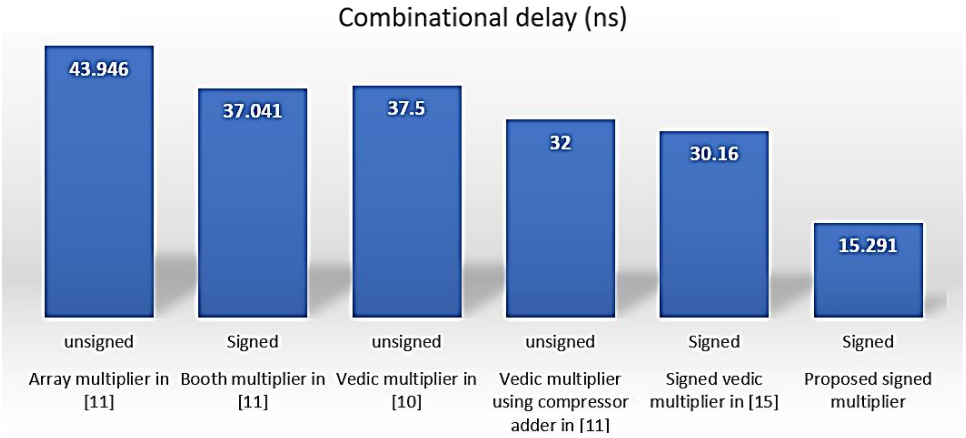

Fig. 2 Combinational delay chart

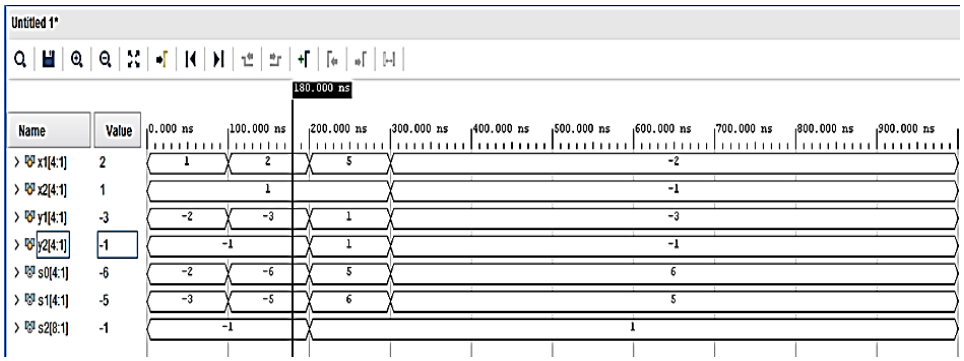

Fig. 3 Simulation result for proposed 8x8 signed multiplier

Figure 3 shows the simulation results of the proposed 8x8 signed multiplier. The signed multiplication of 12 and -13 is -156 which verifies the correct functionality of the architecture.

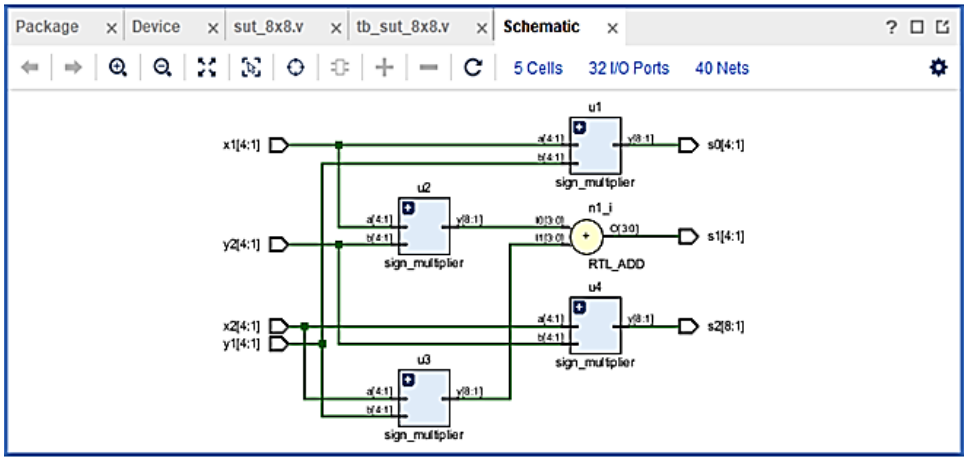

Fig. 4 Hardware realization for proposed 8x8 signed multiplier

The hardware realization for the proposed 8x8 signed multiplier is shown in Figure 4. This RTL schematic is generated after the synthesis and implementation of the design. The architecture utilizes four 4x4 signed multipliers and one adder block. Conventional 8x8 signed multiplier can process operands between -128 to +127, while the proposed 8x8 signed multi-plier can only process -88 to +77 because the representation range of 4-bit signed binary number is -8 to +7. Further, the comparison of the proposed 8x8 signed multiplier is fair with the squaring operation both of [9] and [10] because squaring is nothing but the multiplication of the same operand.

5. CONCLUSION

The Vedic UT algorithm has been used as a basis for designing signed multipliers for FPGA-based systems with real hardware applications. We introduced a novel architecture for signed multiplication, representing a significant advance in VLSI design for high-end computational applications. We have targeted real hardware devices like Virtex-7 and Spartan-7 FPGA devices to test post-layout parameters such as speed and delay for multiplier architectures. Comparing the proposed architecture with existing ones offers valuable insight into its efficiency and effectiveness. According to the proposed design, it exhibits 59% speed improvements over previously reported architectures, which may enhance the performance of high-performance signed number computations in the future. High-performance computing is likely to benefit from further research in this area, especially in the area of integer number computation.

REFERENCES

[1] B. Parhami, *Computer Arithmetic: Algorithms and Hardware Designs*, Oxford University Press, 2000. doi: 10.1093/acprof:oso/9780195125832.001.0001
[2] S. B. K. Tirtha and V. S. Agrawala, *Vedic Mathematics*, Motilal Banarsidass Publishers, India, 1992.
[3] H. Thapliyal and H. R. Arbania, "A Time-Area-Power Efficient Multiplier and Square Architecture Based on Ancient Indian Vedic Mathematics," in *Proceedings of the Int. Conf. on VLSI*, Las Vegas, Nevada,

2004, pp. 434–439.
[4] P. D. Chidgupkar and M. J. Karad, "The implementation of Vedic algorithms in digital signal processing," *Global Journal of Engineering Education*, vol. 8, no. 2, pp. 153–158, 2004.
[5] H. S. Dhillon and A. Mitra, "A Reduced-Bit Multiplication Algorithm for Digital Arithmetic," *International Journal of Computational and Mathematical Sciences*, vol. 2, no. 2, pp. 759–764, 2008.
[6] M. Ramalatha, K. D. Dayalan, P. Dharani, and S. Deeban, "High-speed energy efficient ALU design using Vedic multiplication techniques," in *Proceedings of the Int. Conf. on Advances in Computational Tools for Engineering Applications*, Zouk Mosbeh, Lebanon, 2009, pp. 600–603.
[7] P. Mehta and D. Gawali, "Conventional versus Vedic mathematical method for hardware implementation of a multiplier," in *Proceedings of the Int. Conf. on Advances in Computing, Control and Telecommunication Technologies*, 2009, pp. 640–642.
[8] R. Pushpangadan, V. Sukumaran, R. Innocent, D. Sasikumar, and V. Sundar, "High-speed Vedic multiplier for digital signal processors," *IETE Journal of Research*, vol. 55, no. 6, pp. 282–286, 2009.
[9] P. S. Kasliwal, B. P. Patil, and D. K. Gautam, "Performance evaluation of squaring operation by Vedic mathematics," *IETE Journal of Research*, vol. 57, no. sup1, pp. 39–41, 2011.
[10] K. Sethi and R. Panda, "Multiplier less high-speed squaring circuit for binary numbers," *International Journal of Electronics*, vol. 102, no. 3, pp. 433–443, 2015.
[11] Y. Bansal and C. Madhu, "A novel high-speed approach for 16 × 16 Vedic multiplication with compressor adders," *Computers and Electrical Engineering*, vol. 52, pp. 39–49, 2016.
[12] M. Pradhan and R. Panda, "High-speed multiplier using Nikhilam Sutra Algorithm of Vedic Mathematics," *International Journal of Electronics*, vol. 101, no. 3, pp. 300–307, 2014.
[13] R. K. Barik and M. Pradhan, "Area-time efficient square architecture," *AMSE Journal of Advances in Modeling and Analysis D*, vol. 20, no. 2, pp. 21–34, 2015.
[14] R. K. Barik and M. Pradhan, "Efficient ASIC and FPGA Implementation of Cube Architecture," *IET Computers & Digital Techniques*, vol. 11, no. 1, pp. 43–49, 2017.
[15] R. K. Barik, M. Pradhan, and R. Panda, "Time efficient signed Vedic multiplier using redundant binary representation," *IET Journal of Engineering*, vol. 2017, no. 10, pp. 60–68, 2017.
[16] S. Sahu, B. Bhoi, and M. Pradhan, "Fast signed multiplier using Vedic Nikhilam algorithm," *IET Circuits, Devices & Systems*, vol. 14, no. 5, pp. 623–629, 2020.
[17] S. K. Panda, R. Das, and T. R. Sahoo, "VLSI implementation of Vedic multiplier using Urdhva-Tiryakbhyam sutra in VHDL environment," *IOSR Journal of VLSI and Signal Processing*, vol. 5, no. 5, pp. 17–24, 2015.
[18] P. Vamsi Krishna, K. Nirosha, G. Amala, N. Manikanta, and J. Venkata Suman, "Design and Implementation of FPGA Based 32-bit WALLACE and SYSTOLIC Multipliers," *International Journal of Creative Research Thoughts*, vol. 6, no. 1, pp. 162–166, 2018.
[19] G. K. Ganjikunta, S. I. Khan, and M. Mahaboob Basha, "A High-Performance Signed-Unsigned Multiplier Using Vedic Mathematics," *Journal of Low Power Electronics*, vol. 15, no. 3, pp. 302–308, 2019.
[20] J. J. Huang and S. V. Lale, "A novel nano-scale architecture of Vedic multiplier using majority logic in quantum-dot cellular automata technology," *Electronics Letters*, vol. 58, no. 17, pp. 660–662, 2022.
[21] C. Shylaja, A. Rai, and P. K. Mishra, "Modelling and simulation of 16-bit vedic multiplication using FPGA," *Journal of Physics: Conference Series*, vol. 2007, no. 1, p. 012003, 2021.