International Journal of Recent Technology and Engineering (IJRTE)
ISSN: 2277-3878 (Online), Volume-7, Issue-6, March 2019

Am-Multiplication: A Novel Multiplication Algorithm Based Binary Multiplexer

Amit Verma, Manish Prateek

Abstract: ***Multiplication always remain one of the important operation in arithmetic. Researchers have proposed various methods of multiplication using Vedic literature. However, mainly two approaches of arithmetic multiplication, namely, urdhva tiryakbhya and nikhilam sutra from Vedic literature used by many re-searchers for designing binary multiplexer circuit. Such circuits are complex be-cause of large number of electronic components and interconnection overhead. In this work, we proposed a novel multiplication algorithm (am-MULTIPLICATION) for the arithmetic multiplication of two unsigned whole numbers. The algorithm is extended for performing multiplication operation based on binary numerals that is 0 and 1. The proposed algorithm makes use of three sub-algorithm namely MIN, MAX and SUMMATION for calculating the multiplication of unsigned whole numbers based on the equations for developing n number of sets. A circuit is also designed for performing multiplication operation of binary numerals based on serial shift register and carry look-ahead full adder. The simulation of the circuit is presented using software proteus-8, calculated combinational delay is according to VHDL synthesis report.***

Index Terms**: *Vedic literature, multiplexer, binary.***

I. INTRODUCTION

Multiplication is one of the important basic operation in mathematics. The roots of multiplication methods can be found in Vedic literature. Vedanga jyotisha [1] can be consider as one of the earliest evidence for ganita (Mathematics). Different types of ganita like finger mathematics, mental arithmetic and higher mathematics have also been mentioned in Buddhist literature [2]. The higher mathematics considered as pati-ganita (procedural mathematics, algorithms) and bija-ganita (mathematics of algebra) both are regarded separately by Sridharacharya [3] in Trisatika (Patiganitasara), Patiganita, Bijaganita, Navasati, and Brhatpati. Similarly, the Yajurveda Samhita [4] is considered as the early source of hindu numerals providing the list of hindu numerals of large numbers. The same list of hindu numerals were seen in taittiriya, maitrayani and kathaka Samhita. Pingala chandah-sutra is discussed with an example to demonstrate the sutra introduced by Pingala [5] to find the total number of arrangements of two things in n places.

Revised Manuscript Received on 30 March 2019.
***** Correspondence Author
Amit Verma*, School of Computer Science, department of informatics, University of petroleum and energy studies Dehradun. India.
Manish Prateek, School of Computer Science, University of petroleum and energy studies Dehradun. India.

From past many years Vedic literature has been translated in many regional languages for understanding the basic mathematics. For example, Datta et al. [2] discussed the history of Hindu mathematics that provided the translation and explanation of many Vedic mathematical methods (Sutras) originally in Sanskrit script. Likewise, Brahmagipta et al. [6] discussed the various methods of twenty arithmetic operations like addition, subtraction, multiplication, division, square, square-root, cube, cube-root, five standard forms of fractions, the rule of three, etc. The authors have discussed particularly four methods of multiplication gomutrika, khanda, bheda and ista. More importantly, the multiplication method is considered as an important area of research mainly to reduce the number of steps required in performing the operation. In [7], the authors have discussed about the large percentage of CPU cycle utilization in performing multiplication and division operation. The authors have designed the multiplexer based on combinational logic for generating the product of two numbers. Likewise, the authors in [8] used the Vedic method named Urdhva Tiryakbhyam and proposed a design of multiplier to enhance the speed of computation for performing the multiplication of two numbers. The Vedic method is used for generating partial product in parallel and doing the summation of the partial products. From the literature analysis, we have noticed that the Vedic multiplexer are mainly based on the principle of two Vedic sutras nikhilam sutra and urdhva tiryakbhyam. However, there are many arithmetic methods that are defined in the Vedic manuscripts on which various commentaries have been done by several mathematician such as Arayabhata I, Bhaskara I, Brahmagupta, Srid-hara, Mahavira, Arayabhata II, Sripati, Narayana, Bhaskara II, Ganesha. Inspired from Vedic literature, in this paper, we propose a multiplication algorithm that is based on the process of appending zeros and addition operations. Therefore, the circuit for implementing the algorithm is simple require less number of components, interconnection overhead and delays as compare to the existing algorithm and proposed circuitry for binary multiplication. Our algorithm is works on the development of various sets and equations for searching the presence of multiplier or multiplicand in the developed set(s). The algorithm can be extended for binary multiplication by adding 0s in the end of binary equivalent based on the presence of multiplier or multiplicand in number of set(s), followed by binary addition. The main contributions of the paper are as follows: Firstly, we propose a novel multiplication algorithm for performing arithmetic multiplication. Secondly, a circuit is designed for performing the multiplication of binary data.

Finally, results are shown on the basis of the proposed algorithm and the simulation of the circuit is presented using software proteus-8, delay according to VHDL synthesis report. The rest of the paper is organized as follows: In Section 2, we discuss the works of researchers who proposed various binary multiplication methods and circuitry based on existing arithmetic multiplication method from Vedic literature. The pro-posed arithmetic multiplication algorithm presented in Section 3.The circuit for the binary multiplication based the proposed algorithm in shown in 3.4. Results are discussed in Section 4 with the simulation of the circuit using proteus-8 and VHDL. Finally, we conclude in Section 5 by highlighting some of the future ex-tensions of the presented work.

II. RELATED WORK

Here, we discuss the work of authors who proposed different multiplication methodologies using Vedic literature. colebrooke et al. [9] have discussed eight operations in arithmetic methods mentioned in LILAVATI, treatise of Bhaskaracharya. The authors have mentioned multiple examples based on the various method for per-forming these eight arithmetic operations. In [10], the work provided the rules for solving problems involving linear equations, indeterminate equations of the second degree, arithmetical progressions, quadratic equations, approximate evaluations of square roots, complex series, problems of type $x(1-a_1)(1-a_2).(1-a_3) = P$, the computation of the fineness of gold, income and expenditure, profit and loss. Various example of multiplication are mentioned including dvigunam, asta gunam, gunita jatam, gunita jata, anena gunitam jatam, phalam. The authors in [11] have discussed the Bija-Ganita of Bhaskara and also presented topics from Brahmaguptas Ganita and Bhaskaras Lilavati. In Arithmetic, various rules for performing fundamental operation, multiplication, division, squaring, square root, cubing, cube root, summation and vyutkalita are explained. Nowadays, many researcher are working on the utilization of various Vedic methods of arithmetic to propose architectures for adders, multiplexers to enhance the performance of the current method. In [12], the authors have designed a multipliers over the conventional multiplier using Vedic methods. Their multiplier was based on the concept of urdhva tiryakbhyam sutra for increasing the speed of multiplication. The authors mentioned list of 16 sutras with their meaning. Reduced-bit multiplication algorithm is proposed in [13] using the urdhva tiryakbhyam sutra and nikhilam sutra. The authors have proposed the multiplier design based on both the Vedic methods of multiplication and suggested the multiplier based on urdhva tiryakbhyam sutra is applicable for all sort of multiplication operation and multiplier based on nikhilam sutra is applicable increasing the efficiency of the multiplier for performing multiplication operation of two large numbers instead of small numbers. The authors in [14] considered the booths algorithm for multiplication and modify the booths algorithm by generating regular partial product array instead of irregular partial product array generated by the booths algorithm. They considered the change applicable for reducing the delays, area, negligible overhead and reduce power consumption by the modified multiplier. In [15], the authors have proposed a multiplier based on the Vedic multiplication methods urdhva tiryakbhyam sutra and nikhilam sutra, and showed the architecture of 16 x 16 Vedic multiplier. Likewise, in [16], the importance of Vedic methods has been mentioned for increasing the efficiency of the current system. Vedic division method dhvajanka upasutra for transistor level implementation of divisor circuitry. A new high speed architecture has been proposed in [17] for the multiplexer based on the Vedic method of multiplication nikhilam sutra. The multiplication and addition of the complement of two large operands was performed instead of multiplication of two large operands. The result of the architecture was also compared with the earlier multiplier architecture to show the better utilization and high speed of the binary machine. An 8 and 16 bit Vedic multiplier [18] has been proposed for performing faster multiplication using fast adders for high-speed and low-power applications. Leibniz et al. [19] in their work proposed the binary number system and all the basic mathematical operations such as addition, multiplication, subtraction and division based on binary number system. Vedic hindu mathematics also shows the evidences of binary number system in Pingala chandah-sutra [5]. In [20], the importance of an algorithm and the feasible hardware for doing faster multiplication and addition in the field of Digital Signal Processing (DSP) has been discussed. Also, the authors have mentioned the use of Vedic multiplication methods and their implementation on 8085 and 8086 microcontroller. Moreover, the proposed work can be used in various concept of machine learning [22][23]. Summary of related work is shown in Table 1, where we noticed that most of the researcher have proposed the design for multiplexer for reducing the processing time of multiplication operation using mainly two Vedic methods, namely, urdhva tiryakbhyam sutra and nikhilam sutra.

III. PROPOSED METHODOLOGY

In the section, we present the details of the proposed arithmetic multiplication algorithm. With the concern that the multiplication operation is one of the complex operation which takes more time as compare to other arithmetic operation like addition and subtraction. The present work is motivated from the Hindu Vedic literature. Our proposed algorithm am-MULTIPLICATION consist of three sub-algorithm namely, MIN, MAX and SUMMATION, where MIN and MAX re-turn the minimum and maximum number among the multiplier and multiplicand respectively. The am-MULTIPLICATION algorithm mainly find the number of set(s) to which the minimum among multiplier and multiplicand belongs. Then, the number of set(s) are return to SUMMATION algorithm for the calculation of the final result that is multiplication of multiplier and multiplicand. The work ow for the proposed algorithm in shown in Fig. 1 which involves the development of sets followed by finding minimum and maximum number among the multiplexer and multiplicand that is *a* and *b* respectively. Then, we find the number of set(s) to which *a* belong. Finally, appending the numbers of 0s at the end of binary representation of *b* according to the set(s) to which *a* belong followed by binary addition.

A. Development of Sets

Firstly, we have calculated the set(s) require to perform the multiplication operation. Few examples of set(s) are shown below from S0 to S3. Next, separate general equations are proposed for finding out the values in particular set. After that, finding out the relation among the values of multiple set(s) a fuzzy equation (1) is proposed to and the any number of set(s) to check the presence and absence of the number return by the MIN algorithm among multiplier and the multiplicand.

S_0 = 1, 3, 5, 7, 9, 11, 13, 15
S_1 = 2, 3, 6, 7, 10, 11, 14, 15
S_2 = 4, 5, 6, 7, 12, 13, 14, 15, 20, 21, 22, 23
S_3 = 8, 9, 10, 11, 12, 13, 14, 15, 20, 21, 22, 23

Here is the general equation(s) are shown below for finding the values in particular set, where O is the set of odd numbers.

$O = \{1, 3, 5, 7, 9, 11, 13...\}$
$S_0 = \{2n + 1, n \in N \cup \{0\}\}$
$S_1 = \{2^1 * i + j\}$ where $i \in O$ & $\forall i \in O$, we have $0 \leq j \leq 2^1-1$.
$S_2 = \{2^2 * i + j\}$ where $i \in O$ & $\forall i \in O$, we have $0 \leq j \leq 2^2-1$.
$S_3 = \{2^3 * i + j\}$ where $i \in O$ & $\forall i \in O$, we have $0 \leq j \leq 2^3-1$.
$S_4 = \{2^4 * i + j\}$ where $i \in O$ & $\forall i \in O$, we have $0 \leq j \leq 2^4-1$.

This single equation can be used for finding *n* number of sets like from S_0 to S_n.

$$Sn = \{2^n * i + j\} \text{ where } n \in N \cup \{0\}\ i \in O \ \&\ \forall i \in O \text{ we have } 0 \leq j \leq 2^n - 1. \quad (1)$$

B. Developed Equations

After finding the set(s) and the fuzzy equation for calculating n number of set(s), three equations (2-4) are developed to find the numbers of set(s) to which the number return by MIN algorithm, which takes multiplier and multiplicand as parameter and return the minimum among them belong to. The number can belong to one or multiple set(s) but sets are designed in such a way so that the number will belong to at-least one set.

$$x = 2^i * p \text{ where } i \in N, \quad (2)$$

$$y = x - (x\%2^i), \quad (3)$$

$$p = y/2^i, \quad (4)$$

where *x* will be the minimum among the multiplier and the multiplicand, with the first equation among three the value of p will be calculated for i = 0 and if the value of p is non-fractional than we can decide whether the value belong to the concern set that is S0 or not. If the value of p is fractional than rest two equations are used to calculate the non-fractional value of p to decide whether the p belong to the particular set of not. This process continue from i = 0 to N until the value of 2^i is lesser than or equal to the value of x that is minimum value among multiplier and multiplicand. Here, one theorem is proposed which is used for performing the multiplication operation on the binary equivalent of numbers.

Theorem. If we add 0 at the end of any number n of base m where $m \leq 10$ then the decimal equivalent of the resultant number will be m times the decimal equivalent of number n.

Proof. Let the number be n of base m that is n_m and the decimal equivalent of the number n is x_{10}. So, if we add 0 at the end of number n then the decimal equivalent of the number n will be $(m.x)_{10}$

For example,

$(1101)_2 = (?)_{10}$
$1 * 2^3 + 1 * 2^2 + 0 * 2^1 + 1 * 2^0 = (13)_{10}$

Now, after adding 0 at the end of the binary number we will get the following result

$(11010)_2 = (26)_{10}$
$1 * 2^4 + 1 * 2^3 + 0 * 2^2 + 1 * 2^1 + 0 * 2^0 = (26)_{10}$

And 26 is two times of the decimal equivalent of $(1101)_2$ that is 13.
Lets take another example

$(13)_4 = (?)_{10}$
$1 * 4^0 + 3 * 4^1 = (7)_{10}$

Now, if we add 0 at the end of the number that is 13 and find out the decimal equivalent of the same than we will get following result.

$(130)_4 = (?)_{10}$
$1 * 4^2 + 3 * 4^1 + 0 * 4^0 = (28)_{10}$

And 28 is four times of the decimal equivalent of $(13)_4$ that is 7.

C. Proposed am-MULTIPLICATION algorithm

Proposed algorithm for the multiplication of unsigned whole numbers is divided into four procedures namely am-MULTIPLICATION, MIN, MAX and SUMMATION. For multiplication, we call the procedure am-MULTIPLICATION 1 which takes input as the multiplier m and the multiplicand n. In the algorithm, we call the procedure MIN and MAX which takes the input as m and n, return the minimum and the maximum number respectively which get stored in the variable a and b. If any of the value among a and b is zero then algorithm return 0 and exit. Otherwise the value of p is calculated initially for i = 0 and if the value of p belong to O, then the set according to the value of i, is considered as selected. If the value of p belong to E, where E is the set of even numbers E = 2, 4, 6, 8, 10, 12, 14... then the particular set is not considered as selected set and finally if the value of p is any fractional number then value of p is recalculated and again get compared to find the selected set. The value of i get incremented and the same procedure remain continue to find the other sets to which the number a belongs uptil the value of a is greater than or equal to 2^i

---

Algorithm 1 am-MULTIPLICATION(m, n)

Require Multiplier m and multiplicand n as parameter
Ensure: Find the number of set(s) to which the minimum among m and n belong.
1: a = MIN(m, n)
2: b = MAX(m, n)
3: if a = = 0 || b = = 0
4: return 0
5: i = 0
6: $p = a/2^i$
7: if p ∈ O // O is set of all odd numbers

```
 8:      S = ϕ ∈ S_i      // update set S
 9: else if p ∈ E        // E is set of all even numbers
10:      S ≠ ϕ ∈ S_i      // dont update Set S
11: else if p ∈ f        // f is any fractional number
12:      y = a - (a % 2^i)
13:      p = y/2^i
14:      goto step 7
15: i = i + 1
16: if 2^i ≤ a
17:      goto step 6
18: else r = SUMMATION(S, b)
```

In am-MULTIPLICATION algorithm, sub-algorithm MIN 2 takes input as m and n, return the minimum among the multiplier and the multiplicand, if both the number are equal then the algorithm will return m. MAX 3 also takes input as m and n, return the maximum among them.

Algorithm 2 MIN(m, n)

Require: Multiplier m and multiplicand n as parameter
Ensure: Find the minimum among m and n.

```
1: If m  n
2:      return m
3: else
4:      return n
```

Algorithm 3 MAX(m, n)

Require: Multiplier m and multiplicand n as parameter
Ensure: Find the maximum among m and n.

```
1: If m > n
2:      return m
3: else
4:      return n
```

For am-MULTIPLICATION algorithm, SUMMATION 4 takes input as S, a, b represents the set of selected sets, minimum and maximum number among m, n respectively. Finally, result r get return to the am-MULTIPLICATION algorithm from where the procedure has been called.

Algorithm 4 SUMMATION(S, a, b)

Require Set S containing the set(s) to which a belong to, a and b as parameter.
Ensure: Calculate result

```
1: i = 0
2: z = 0
3: c = 0
4: while 2^i ≤ a
5:     if S_i ∈ S
6:         z = z + ∑ b        //from 0 to 2^i
7:         return m
8:     i = i + 1
9: return z
```

D. Circuit diagram for Binary Multiplexer using am-MULTIPLICATION algorithm

Here the circuit is proposed for the multiplication of two binary numbers according to the am-MULTIPLICATION algorithm 1. After finding the minimum number among multiplier and multiplicand using MIN algorithm 2, the number of set(s) that is S is identified to which the number, let the number be a return by MIN algorithm belong to using am-MULTIPLICATION 1. On the basis of the set(s) to which a belong, equal number of serial shift registers as shown in the Fig. 2 are used. If set S contain two sets like S = {S_0, S_1} then two serial shift registers will be used. Now according to the set S, 0s are added after the last bit of binary equivalent of the number return by MAX algorithm 3, let the number be b. Now the output of serial shift register(s) is given to the carry look-ahead full adder as shown in Fig. 3 for the summation.

As shown in the Fig. 3 the carry look-ahead full adder accept three inputs in which two bit for the summation and one carry generated in the previous state as represented by C(N-1) in the figure, and CN is representing the carry generated in the current state.

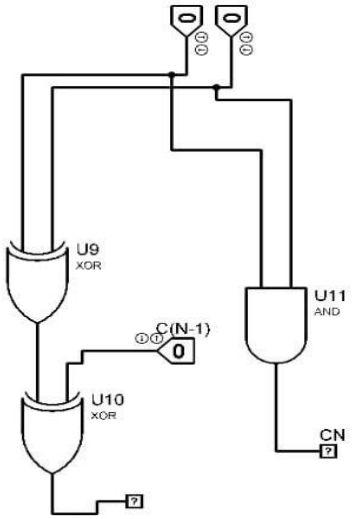

Fig. 3 Carry Look-ahead Full Adder for the summation of generated carry and the input received form shift registers.

The complete circuit diagram of am-MULTIPLICATION algorithm for multiplication of two binary numbers, where the number is return by MIN algorithm 2 belongs to two sets. As discussed above, two serial shift registers are required as depicted in the Fig. 4. And the output of the shift registers will be consider as input to the carry look-ahead full adder where CN is representing the carry generated in current state and C(N-1) is representing the carry from the previous state. Therefore, initially C(N-1) is considered as 0 and for the summation of the next two bits the value of CN will be consider as the value of C(N-1).

IV. RESULTS AND DISCUSSION

In this section, we are considering two unsigned whole numbers for multiplication according to the proposed am-ALGORITHM 1. Let the numbers are m = 3 and n = 9. Now, according to the algorithm MIN 2 and MAX 3 return minimum and maximum among m; n that is a = 3 and b = 9. Next, we step by step follow the am-ALGORITHM 1 to calculate the multiplication of m and n in which we calculate the value of p to find the number of set(s) to which a = 3 belong as shown below.

- a = 3, b = 9, i = 0
- $p = 3/2^0$, p = 3
- p ∈ {odd numbers}, update S = {S_0}
- i = i + 1, i = 1

- $p = 3/2^1$, $p \in f$
- $y = 3 - (3 \% 2^1)$, $y = 2$
- $p = 2/2^1$, $p = 1$
- $p \in$ {odd numbers}, update $S = \{S_0, S_1\}$
- Now $2^2 > 3$
- Final set $S = \{S_0, S_1\}$

Append the number of 0s according to S in binary equivalent of b = 9 that is 1001 we get, x = 1001 and y = 10010

The above multiplication operation is simulate over the circuit using simulation software proteus-8 for the multiplication of a = 3 and b = 9 after finding out the set $S = \{S_0, S_1\}$ to which a = 3 belong to. Next, the binary numbers based on the set S = {S0, S1}, i.e., x = 1001 and y = 10010 are simulate on the circuit for getting the final result r = 11011. As shown in the Fig. 5 where (a) The output of the first and second shift register that is the last bit of y and x which are 0,1 act as input to carry look-ahead full adder so the output will be sum = 1 and carry for the current state CN = 0 where the C(N-1) initially remain 0. (b) Now the next two bits that is 1 and 0 from the sift register is provided to the carry look-ahead adder with C(N-1) = 0, so we get sum = 1 and CN = 0. Here C(N-1) is 0 which is equivalent to the CN of the previous state. (c) Similarly, next bits that is 0 and 0 will be the output of the sift register which will act as input to the carry look-ahead adder with C(N-1) = 0 to get the sum = 0 and CN = 0. (d) And now for next two bits, 0, 1 and C(N-1) = 0 we get sum = 1 and CN = 0. (e) For last pair of bits 1, 0 and C(N-1) = 0 we get sum = 1 and CN = 0. In this way after keeping the track of all the sum values, the result will be r = 11011 which is equivalent to the 27_{10}.

In Fig.5 red and blue LED representing 1 and 0 respectively.

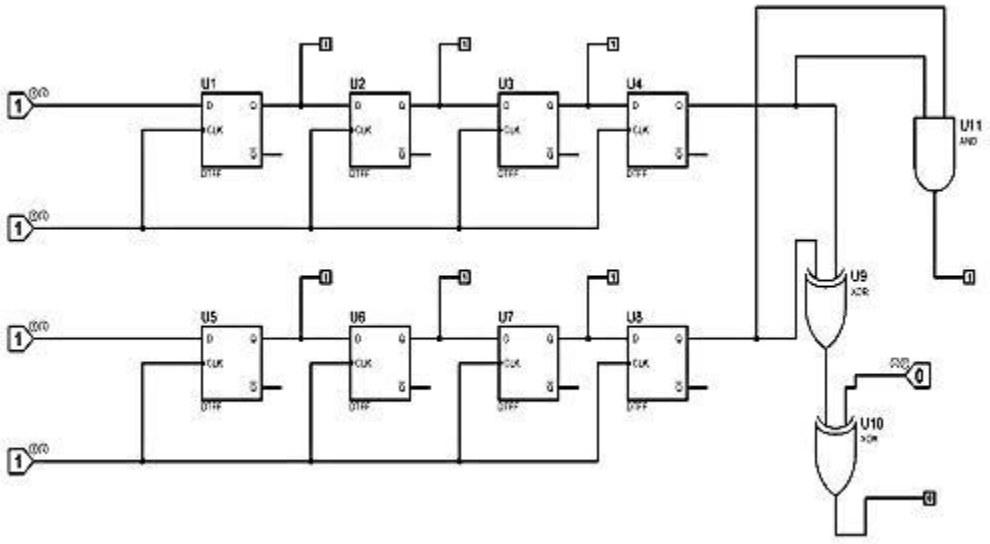

Fig. 4 Circuit based on am-MULTIPLICATION algorithm in which the output of the shift registers and the generated carry of previous state act as input to carry look-ahead adder for binary addition

The speed of the multiplexer can be expressed in terms of basic gate delays, number of gate count in the circuit and the power consumption by the circuit. Therefore, the comparison of the proposed circuit has been done with existing techniques. For example, in [21], the authors shown the circuit for 8x8 combinatorial multiplier and Wallace-tree implementation of 8x8 multiplier, to ease the calculation of the speed of multiplication in terms of gate delays and gate count. The total number of (AND/OR) gates count was 624 including 56 binary adders for 8x8 combinatorial multiplier and a gate delay of 57 was recorded. For Wallace-tree implementation of 8x8 multiplier, the total gate count was 564 gates which include 22 binary adders and 23 gate delays. The proposed multiplexer circuit based on am-MULTIPLICATION algorithm for 8x8 multiplier require 8 carry look-ahead adders and 8 shift register each carry look-ahead adder comprises of 30 (AND/OR) gates where each shift register is the combination of 4 D flip- flop depicted in Fig. 2. The four flip-flops are used to maintain the stability of the bit propagation. However, only one D flip- flop can be used to propagate single bit with each clock pulse. And each D flip- flop require 4 gates, so the total number of gates will be 272 gates and 9 gate delays where 1 gate delay is required for the generation of partial product.

The gate delay and gate count for the proposed circuitry is much more lesser as compare to the combinatorial and Wallace-tree based multiplier as shown in the Table 2.

Table 2 Gate count and gate delay for three types of 8x8 multiplexers.

In [20], the authors provided the result of delays in modified booth Wallace multiplier and proposed Vedic multiplexer based urdhva tiryakbhyam sutra simulated on VHDL. The result comparison for delay is calculated with the proposed circuit based on am-MULTIPLICATION algorithm ccording to VHDL

Multiplexer	Gate count	Gate delay
Combinatorial	624	57
Wallace-tree	564	23
am-Multiplication(Proposed)	272	9

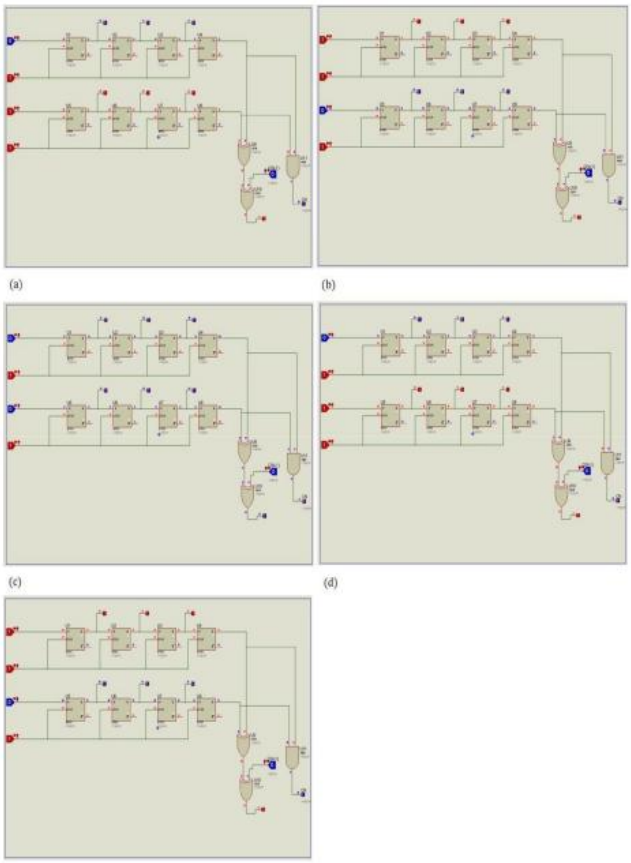

Fig. 5 Results on Proteus 8 (a) The output of carry look-ahead adder Sum = 1 for y = 0, x = 1 generated from shift registers and carry C(N-1) = 0 (b) Now, Sum = 1 for y = 1, x = 0 and C(N-1) = 0 (c) Sum = 0 for y = 0, x = 0 and C(N-1) = 0 (d) Sum = 1 for y = 0, x = 1 and C(N-1) = 0 (e) Sum = 1 for y = 1, x = 0 and C(N-1) = 0.

synthesis report as shown in Table 3. Similarly, the authors in [15] have proposed a 16x16 Vedic multiplexer based on urdhva tiryakbhyam and Nikhilam Sutra. The results were reported using VHDL synthesis. We have calculated the delay of our proposed circuit using VHDL synthesis which is much lesser as compare to the results of Vedic multiplexer based on urdhva tiryakbhyam and Nikhilam Sutra as presented in Table 4.

Table 3 Calculated delays in ns based on VHDL Synthesis report for 8x8 multiplexers.

Multiplexer	Gate delay
Modi ed booth Wallace	15.815
Vedic	15.685
am-Multiplication (Proposed)	8.653

Table 4 Calculated delays in ns based on VHDL Synthesis report for 16x16 multiplexers.

Multiplexer	Gate delay
urdhva tiryakbhyam sutra	41.751
Nikhilam Sutra	33.729
am-Multiplication (Proposed)	16.797

I. CONCLUSION AND FUTURE SCOPE

In this paper, we proposed a novel multiplication algorithm for unsigned whole numbers. The algorithm works on mathematical equations for generating complete set of sets S = S_0, S_1, S_2…S_n and the equations to check to which set(s) the minimum number among multiplier and multiplicand belongs. The methodology is based on searching and appending 0's to the binary equivalent of maximum number among multiplier and multiplicand accordingly and finally adding the binary equivalent of the numbers generated according to set S. The algorithm can be used for performing multiplication of binary numbers for the current digital machines. The circuitry required for the multiplication of binary numerals based on propose algorithm is simple, require less number of electronic components, interconnection overhead and delays. In future, if the computer machine develop for higher base then also the algorithm can be used for performing multiplication of unsigned whole numbers. The Algorithm can be extent for the multiplication of signed and fractional numbers.

REFERENCES

1. PV Holay. Vedic astronomy. Shri Babasaheb Apte Smarak Samitee, Nagpur, 1994.
2. Bibhutibhusan Datta and Avadhesh Narayan Singh. History of Hindu mathematics. Asia Publishing House; Bombay, 1935.
3. K Shankar Shukla. On sridharas rational solution of nx^ 2+ 1= y^ 2. Ganita, 1:1-12, 1950.
4. Albrecht Weber. The history of Indian literature. Routledge, 2013.
5. R Sridharan. Sanskrit prosody, pigala sutras and binary arithmetic. In Contributions to the History of Indian Mathematics, pages 33-62. Springer, 2005.
6. ACHARYAVARA RS SHARMA. Shri brahmagupta viracita brahma-sphuta siddhanta. Vol. I, Indian Institute of Astronomical and Sanskrit Research, New Delhi, Kuttakadhyya, pages 64-65, 1966.

7. D Jacobsohn. A suggestion for a fast multiplier. IEEE Transactions on Electronic Com-puters, 6(EC-13):754, 1964.
8. Himanshu Thapliyal and Hamid R Arabnia. A time-area-power e cient multiplier and square architecture based on ancient indian vedic mathematics. In ESA/VLSI, pages 434-439, 2004.
9. Henry Thomas Colebrooke. Algebra with arithmetic and mensuration from the sanskrit of brahmegupta and bhaskara (london: 1817).
10. Svami Satya Prakash Sarasvati and Usha Jyotishmati. The bakhshali manuscript. an ancient treatise of indian arithmetic, 1979.
11. M Rangacarya. The ganita-sara-sangraha of mahav racarya. Bull. Amer. Math. Soc, 1912.
12. Honey Durga Tiwari, Ganzorig Gankhuyag, Chan Mo Kim, and Yong Beom Cho. Multi-plier design based on ancient indian vedic mathematics. In SoC Design Conference, 2008. ISOCC'08. International, volume 2, pages II-65. IEEE, 2008.
13. Harpreet Singh Dhillon and Abhijit Mitra. A reduced-bit multiplication algorithm for digital arithmetic. International Journal of Computational and Mathematical Sciences, 2(2), 2008.
14. Shiann-Rong Kuang, Jiun-Ping Wang, and Cang-Yuan Guo. Modi ed booth multipliers with a regular partial product array. IEEE Transactions on Circuits and Systems II: Express Briefs, 56(5):404-408, 2009.
15. Manoranjan Pradhan, Rutuparna Panda, and Sushanta Kumar Sahu. Speed comparison of 16x16 vedic multipliers. International Journal of Computer Applications (0975-8887), 21(6), 2011.
16. Prabir Saha, Deepak Kumar, Partha Bhattacharyya, and Anup Dandapat. Vedic division methodology for high-speed very large scale integration applications. The Journal of Engineering, 2014(2):51-59, 2014.
17. Manoranjan Pradhan and Rutuparna Panda. High speed multiplier using nikhilam sutra algorithm of vedic mathematics. International Journal of Electronics, 101(3):300-307, 2014.
18. G Ganesh Kumar and Subhendu K Sahoo. Implementation of a high speed multiplier for high-performance and low power applications. In VLSI Design and Test (VDAT), 2015 19th International Symposium on, pages 1-4. IEEE, 2015.
19. G Leibniz. 1703. explication de larithmetique binaire [explanation of binary arithmetic]; gerhardt, mathematical writings.
20. Ramesh Pushpangadan, Vineeth Sukumaran, Rino Innocent, Dinesh Sasikumar, and Vaisak Sundar. High speed vedic multiplier for digital signal processors. IETE journal of research, 55(6):282-286, 2009.
21. Shlomo Waser. High-speed monolithic multipliers for real-time digital signal processing. Computer, (10):19-29, 1978.
22. Pradeep Kumar, Rajkumar Saini, Santosh Kumar Behera, Debi Prosad Dogra, and Partha Pratim Roy. Real-time recognition of sign language gestures and air-writing using leap motion. In 2017 Fifteenth IAPR International Conference on Machine Vision Applications (MVA), pages 157{160. IEEE, 2017.
23. Pradeep Kumar, Rajkumar Saini, Pawan Kumar Sahu, Partha Pratim Roy, Debi Prosad Dogra, and Raman Balasubramanian. Neuro-phone: An assistive framework to operate smartphone using eeg signals. In 2017 IEEE Region 10 Symposium (TENSYMP), pages1-5. IEEE, 2017.

AUTHORS PROFILE

Amit Verma presently working as assistant professor in School of Computer Science, department of informatics, University of petroleum and energy studies Dehradun. He is Ph.D. scholar of University of petroleum and energy studies Dehradun and his area of interest includes multi-valued logic, basic electronics components. Recently he has filed a patent using the concept of solenoid, which is published in Indian patent journal.

Manish Prateek presently working as a professor and dean School of Computer Science, University of petroleum and energy studies Dehradun. His area of interest includes multi-valued logic, robotics, image processing, and computer organization. He is having many national and international publications in the mentioned areas and attended multiple conferences in India and abroad. Recently he has filed a patent using the concept of solenoid, which is published in Indian patent journal.