Design, Implementation and Performance Analysis of an Integrated Vedic Multiplier Architecture

Ramachandran.S*, Kirti.S.Pande
Department of Electronics and Communication
Amrita School of Engineering
Kasavanahalli, Bangalore- 560035, India

Abstract

Fundamental and the core of all the Digital Signal Processors (DSPs) are its multipliers and speed of the DSPs is mainly determined by the speed of its multipliers. Multiplication is the most fundamental operation with intensive arithmetic computations. Two important parameters associated with multiplication algorithms performed in DSP applications are latency and throughput. Latency is the "real delay of computing a function". Throughput is a measure of "how many computations can be performed in a given period of time". The execution time of most DSP algorithms is dependent on its multipliers, and hence need for high speed multiplier arises.

Urdhva tiryakbhyam sutra performs faster for small inputs and Nikhilam sutra for larger inputs. Here a novel Integrated Vedic multiplier architecture, which by itself selects the appropriate multiplication sutra based on the inputs, is proposed. So depending on inputs, whichever sutra is faster, that sutra is selected by the proposed integrated Vedic multiplier architecture. In the simulation results, it can be seen that Urdhva performs faster for small inputs, but Nikhilam performs better for large inputs (more than twice as much for 64 bit multiplicands).

Keywords— Integrated Vedic multiplier architecture, Nikhilam sutra architecture, Urdhva tiryakbhyam sutra.

I. Introduction

Vedic mathematics is an extract from four Vedas (books of wisdom). It is actually a sub module of "Sthapatya-veda" (book on civil engineering and architecture), which is an upya-veda (supplement) of Atharva Veda. Owing to its simplicity and regularity, it finds its utility and applications in the fields of geometry, trigonometry, quadratic equations, factorization and calculus.

His holiness Jagadguru Shankaracharya Bharati Krishna Teerthaji Maharaja (1884-1960) comprised all this work. He did an extensive research in the Vedas and came up with the simplified form of calculations which are yet so powerful. He came up with complete explanations and presented them in the form sutras. He constructed 16 sutras (formulae) and 16 Upya sutras (sub formulae) after extensive research in Atharva Veda. As all these sutras were extracts from swamiji's own findings and research, these are not found explicitly in the Veda. 'Vedic' is a term derived from the word 'veda' which means the store-house of all knowledge.

The power of Vedic mathematics is not only confined to its simplicity, regularity, but also it is logical. Its high degree of eminence is attributed to the aforementioned facts. It is these phenomenal characteristics, which made Vedic mathematics, become so popular and thus it has become one of the leading topics of research not only in India but abroad as well.

Vedic mathematics' logics and steps can be directly applied to problems involving trigonometric functions, plane and sphere geometry, conics, differential calculus, integral calculus and applied mathematics of various kind.

The awe striking and the mind boggling feature of Vedic mathematics lies in the fact that it simplifies the complicated looking calculations in conventional mathematics to a simple one in a much faster and efficient manner. This is attributed to the fact that the Vedic formulae are claimed to be based on the "natural principles on which the human mind works". Hence this presents some effective algorithms which can be applied to various branches of engineering.

The architecture of Multipliers can be generally classified into three categories. First is the "serial multiplier" which emphasizes on hardware optimization of chip area. Second is "parallel multiplier" which performs high speed mathematical operations, the drawback being relatively larger chip area consumption. The final one is "serial-parallel multiplier" which is a trade-off between time consuming serial multipliers and the area consuming parallel multipliers.

II. Related Works

A high speed multiplier design (ASIC) using Vedic mathematics was presented in [1]. The idea for designing the multiplier and adder unit was adopted from ancient Indian mathematics "Vedas". Based on those formulae, the partial products and sums are generated in single step which reduces the carry propagation from LSB to MSB. The implementation of the Vedic mathematics and their application to the complex multiplier ensured substantial reduction of propagation delay in comparison with Distributed Array (DA) based architecture and parallel adder based implementation which are most commonly used architectures.

The implementation of the Vedic algorithms in DSP is highlighted in [2]. In this, multiplication process based on Vedic mathematics and its implementation on 8085 and 8086 microprocessors was shown. A comparative study of processing time of conventional multipliers for 8085 and 8086 was done. It was shown that there is an appreciable saving in the processing time of the Vedic multiplier as when compared to that of a conventional multiplier.

A time, area, power efficient multiplier architecture using Vedic mathematics was shown in [3]. In this a comparative study of the array multiplier, Carry save multiplier, Wallace tree multiplier, Booth multiplier and Vedic multiplier was done in detail. The study clearly showed that though array and booth multipliers are faster among the conventional multipliers, they are so because of some trade-off with complexity and high power consumption respectively.

A fast, low power multiplier architecture based on Vedic mathematics was shown in [4]. This paper presented a new architecture for multiplication which uses the modified binary tree network (MBT). This architecture focuses on generating all partial products in one step. The generated partial products are added by the MBT network. This also showed evidence of increase in speed.

Reduced bit multiplication algorithm for digital arithmetic was shown in [5]. It mainly consisted of the in depth explanation of Urdhva tiryakbhyam sutra and the Nikhilam sutra. These sutras are the extracts from the Vedas which are the store house of knowledge. The former was suggested for smaller numbers and the latter was suggested for larger numbers. This paper showed that multiplication of two 8 bit numbers can be effected by reducing it further into two 4 bit numbers and likewise.

VHDL implementation of a NXN multiplier based on the Vedic mathematics was shown in [6]. This proposed a way to implement the design of the Urdhva sutra based multiplier as a bottom up design methodology.

A novel design for square and cube architecture was shown in [7]. It was very clearly evident from the explanations that, the Vedic square and cube architecture were faster than the conventional square and cube calculations.

All the Vedic based calculations, sutra explanations and complete discussions were made in [8]. This does all explanations right from scratch. This contains the complete essence of the Vedic techniques, calculations and all the associated sutras with the explanations in detail. This forms the base and is the fundamental of any Vedic mathematics based calculations.

III. Design Approach

A. Urdhva tiryakbhyam sutra illustration

Multiplication is based on an algorithm called Urdhva Tiryakbhyam (Vertical & Crosswise) of ancient Indian Vedic Mathematics. Urdhva Tiryakbhyam Sutra is a general multiplication formula applicable to all cases of multiplication. The Sanskrit term means “Vertically and crosswise”. The idea here is based on a concept which results in the generation of all partial products along with the concurrent addition of these partial products in parallel [5]. The parallelism in generation of partial products and their summation is obtained using Urdhva Tiryakbhyam explained in Fig. 1.1. Since there is a parallel generation of the partial products and their sums, the processor becomes independent of the clock frequency. Thus the multiplier will require the same amount of time to calculate the product and hence is independent of the clock frequency. The advantage here is that parallelism reduces the need of processors to operate at increasingly high clock frequencies. A higher clock frequency will result in increased processing power, and its demerit is that it will lead to increased power dissipation resulting in higher device operating temperatures. By employing the Vedic multiplier, all the demerits associated with the increase in power dissipation can be negotiated. Since it is quite faster and efficient its layout has a quite regular structure. Owing to its regular structure, its layout can be done easily on a silicon chip [4]. The Vedic multiplier has the advantage that as the number of bits increases, gate delay and area increases very slowly as compared to other multipliers, thereby making it time, space and power efficient. It is demonstrated that this architecture is quite efficient in terms of silicon area/speed [4].

To illustrate this multiplication scheme, let us consider the multiplication of two decimal numbers (325 * 738). Line diagram for the multiplication is shown in Fig.1.2. Initially the LSB digits on the both sides of the line are multiplied and added with the carry from the previous step. This generates one of the bits of the result and a carry. This carry is added in the next step and the process goes on likewise. If more than one line are there in one step, all the results are added to the previous carry. In each step, least significant bit act as the result digit and all other digits act as carry for the next step. Initially the carry is taken to be zero.

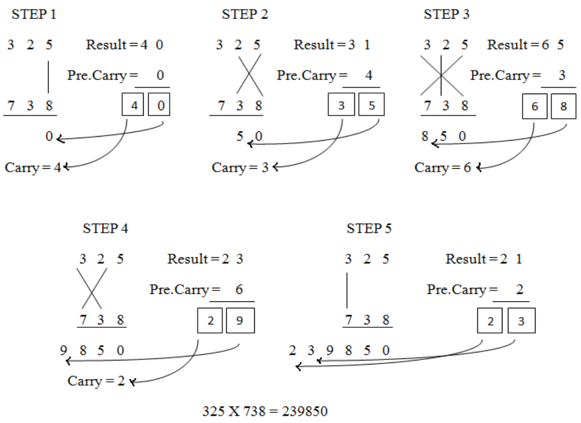


Fig. 1.1 Multiplication of two numbers using Urdhva sutra

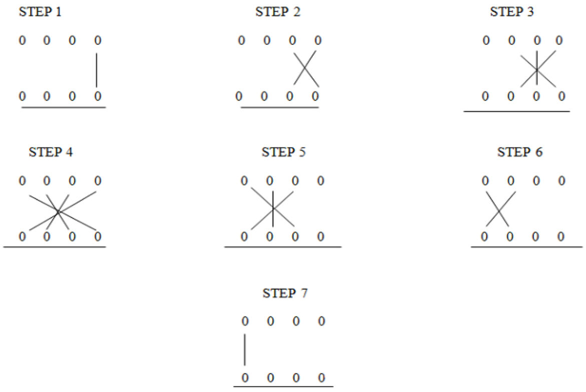


Fig. 1.2 Line diagram of the multiplication

To make the methodology more clear, an alternate illustration is given with the help of line diagrams in Fig. 1.2, where the dots represent bit “0” or “1” [4]. Here in order to illustrate the multiplication algorithm, we consider the multiplication of two binary numbers a3a2a1a0 and b3b2b1b0. As the result of this multiplication would be more than 4 bits, it is expressed as ...r3r2r1r0. Line diagram for multiplication of two 4-bit numbers is shown in Fig. 1.2. This just maps the illustration in Fig.1.1 in binary system. Least significant bit r0 is obtained by multiplying the least significant bits of the multiplicand and the multiplier. The process is followed as per the steps shown in Fig. 1.1. The same method is used and the design for multiplication of two 64 bit numbers is done using the bottom up methodology. The architecture is shown in Fig. 1.3 which is followed as per the concept of [9][10].

A31-A0 and B31-B0 generates Q31-Q0 of the output and the remaining bits are forwarded as carry to next. A31-A0, B63-B32 and A63-A32, B31-B0 and the carry from previous step together forms Q63-Q32 of the output and the remaining bits are forwarded to the next. Finally A63-A32 and B63-B32 and the carry from the previous step together form the Q127-Q64 of the output. Thus a 64 bit multiplication results by multiplying four 32 bit numbers in parallel.

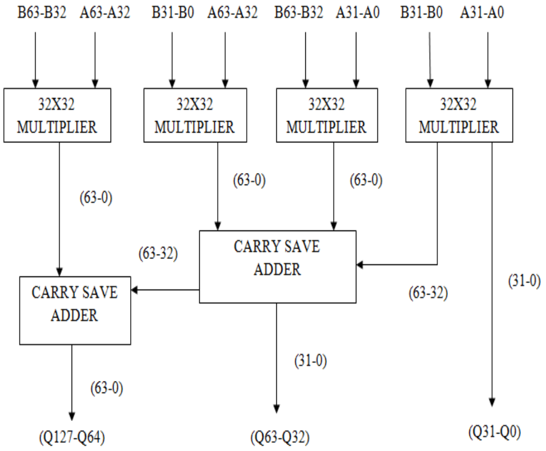


Fig. 1.3. Architecture of 64 bit Urdhva sutra [9][10]

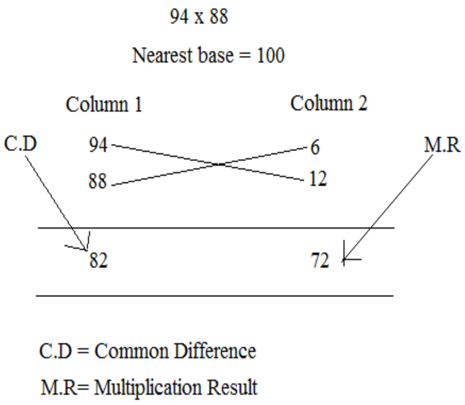


Fig. 1.4. Nikhilam sutra illustration [4]

B. Nikhilam sutra illustration

The Sanskrit term Nikhilam means “all from 9 and last from 10”. It is also applicable to all cases of multiplication, but it tends to be more efficient when the numbers involved are large. This is because, it just finds out the compliment of the large number from its nearest base to perform the multiplication operation on it [4]. Larger the original number, lesser the complexity of the multiplication. This sutra is illustrated by considering the multiplication of two decimal numbers (94 * 88) where the chosen base is 100 which is nearest to and greater than both these two numbers.

The right hand side (RHS) of the product can be obtained by simply multiplying the numbers of the Column 2 (6*12 = 72). The left hand side (LHS) of the product can be found by cross subtracting the second number of Column 2 from the first number of Column 1 or vice versa, i.e., 94 - 12 = 82 or 88 - 6 = 82. The final result is obtained by concatenating RHS and LHS (Answer = 8272).

The proposed Nikhilam sutra architecture is shown in Fig. 1.5 and is based on the above illustration of the sutra.

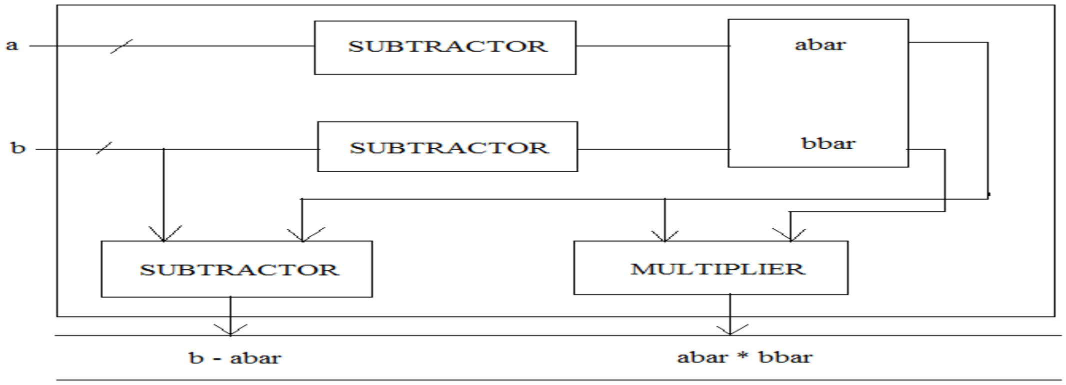


Fig. 1.5. Proposed architecture of Nikhilam sutra multiplier

It is known from literature that Urdhva based multiplier is expected to work faster for small inputs and Nikhilam sutra based multiplier for large inputs. Hence, Integrated Vedic Multiplier Architecture is proposed in this paper, which is capable of selecting the better multiplier sutra based on the inputs given. The proposed Integrated Vedic Multiplier Architecture is shown in Fig. 1.6.

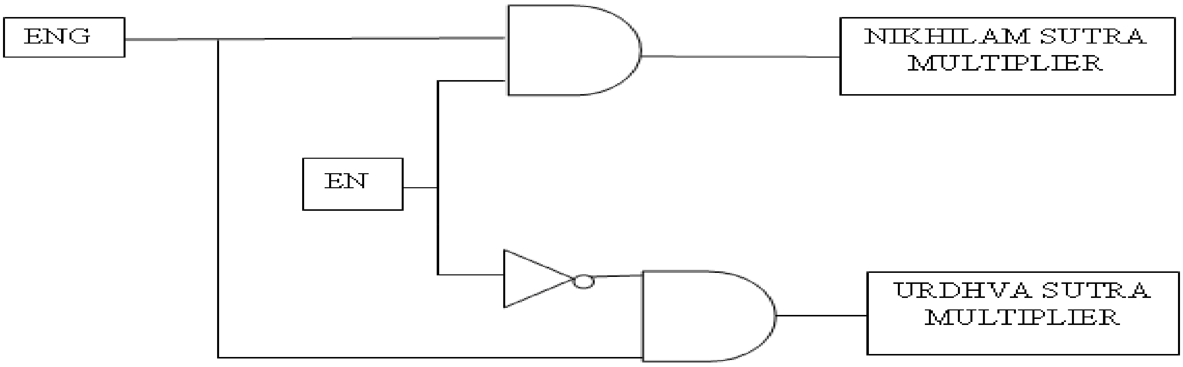
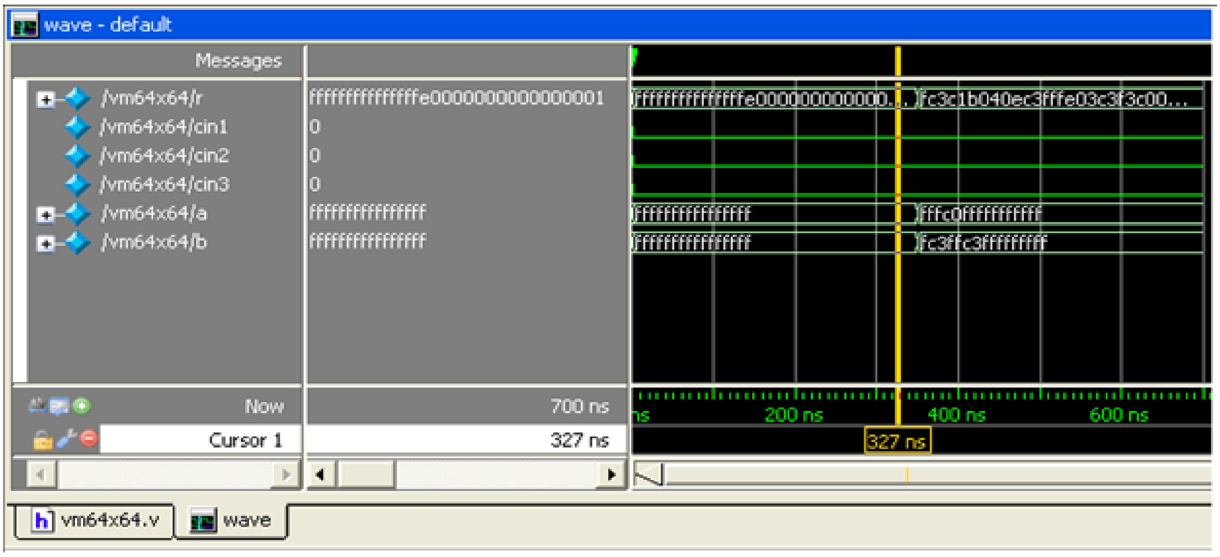


Fig. 1.6. Proposed Integrated Vedic multiplier architecture

The concept is that, the initial conditions are set at the start (say) at around 20% from the nearest base as the Nikhilam limit. If the inputs lie inside Urdhva limit, Urdhava based multiplier will perform the multiplication and if the inputs lie inside Nikhilam limit, Nikhilam based multiplier will perform the multiplication. This is extended for all higher order cases. This proposed architecture is aimed at achieving faster results. Also, when one multiplier is ‘ON’, the other is ‘OFF’. This accounts for low power consumption of the proposed architecture.

IV. RESULT

A. Simulation Result

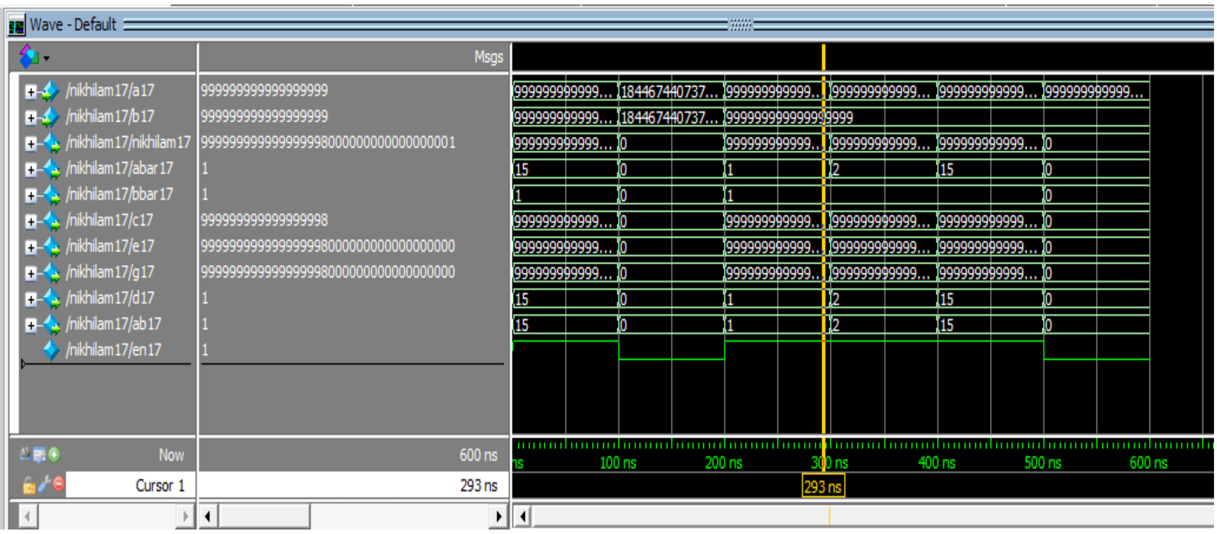

wave - default

```
Messages
/vm64x64/r      fffffffffffffffe0000000000000001
/vm64x64/cin1   0
/vm64x64/cin2   0
/vm64x64/cin3   0
/vm64x64/a      ffffffffffffffff
/vm64x64/b      ffffffffffffffff

Now             700 ns
Cursor 1        327 ns
```

Fig. 1.7. Simulation result of 64x64 Urdhva multiplier

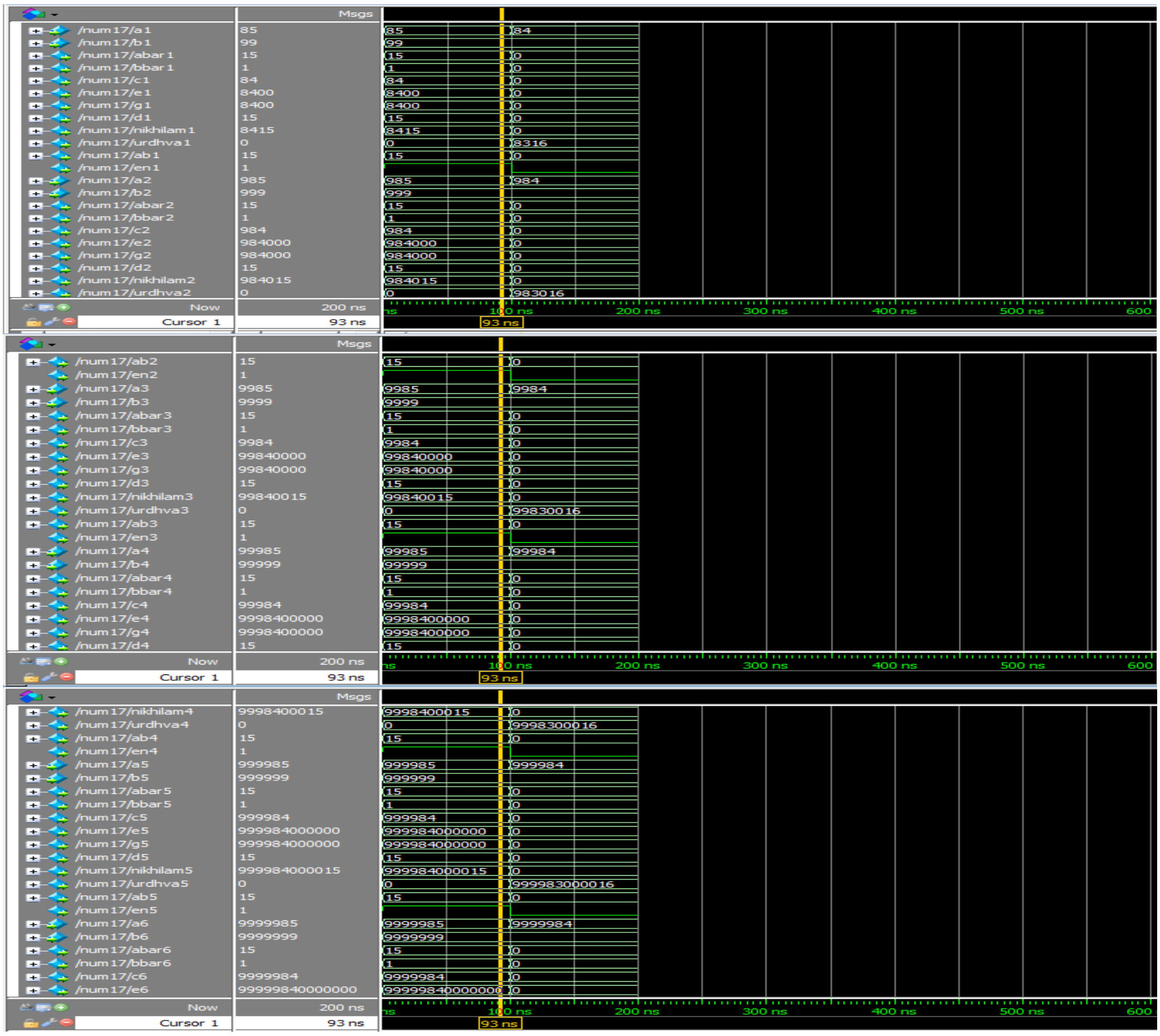
Here ‘a’ and ‘b’ are the two 64 bit inputs (hexadecimal) and ‘r’ is the output which results in a 128 bit binary number. Corresponding hexadecimal values are shown in the output.



```
Wave - Default
                                  Msgs
/nikhilam17/a17         9999999999999999999
/nikhilam17/b17         9999999999999999999
/nikhilam17/nikhilam17  99999999999999999980000000000000000001
/nikhilam17/abar17      1
/nikhilam17/bbar17      1
/nikhilam17/c17         9999999999999999998
/nikhilam17/e17         99999999999999999980000000000000000000
/nikhilam17/g17         99999999999999999980000000000000000000
/nikhilam17/d17         1
/nikhilam17/ab17        1
/nikhilam17/en17        1
Now                     600 ns
Cursor 1                293 ns
```

Fig. 1.8. Simulation result of Proposed 64x64 Nikhilam multiplier

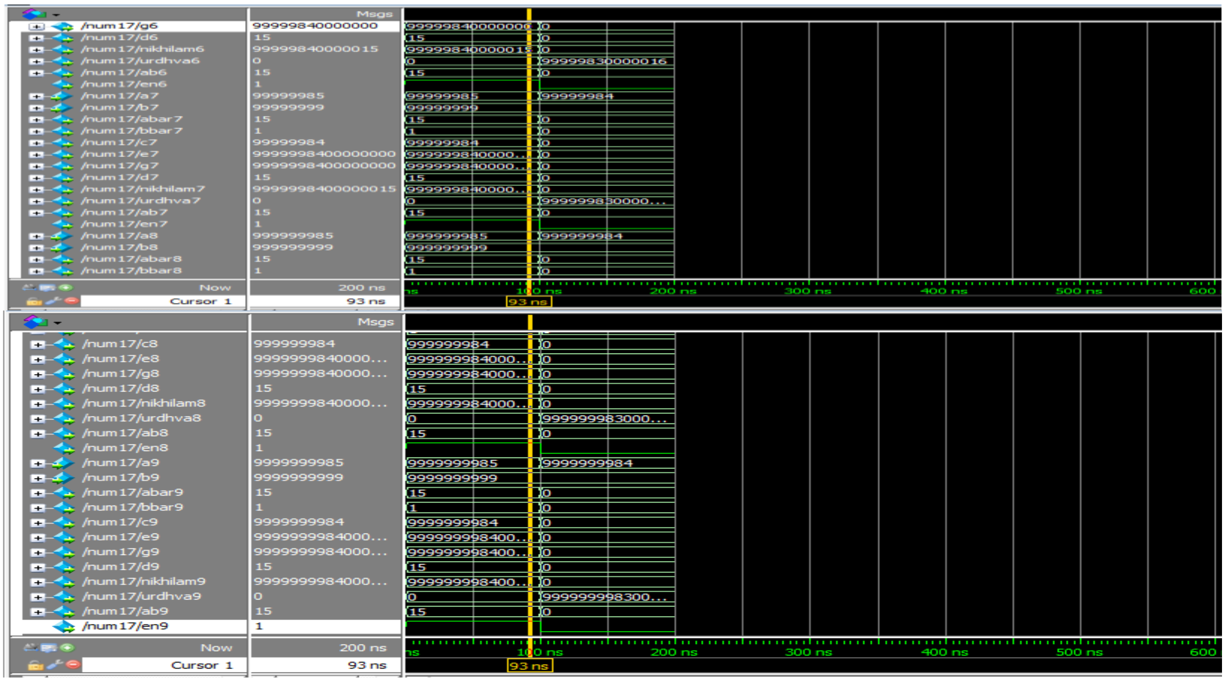
Here if both ‘a’ and ‘b’ are within the permissible limits of the Nikhilam condition, ‘en’ becomes ‘1’ and if ‘a’ and ‘b’ are outside Nikhilam limit, ‘en’ becomes ‘0’ and Nikhilam stops functioning (i.e.) “OFF”.



```
                          Msgs
/num17/a1           85
/num17/b1           99
/num17/abar1        15
/num17/bbar1        1
/num17/c1           84
/num17/e1           8400
/num17/g1           8400
/num17/d1           15
/num17/nikhilam1    8415
/num17/urdhva1      0
/num17/ab1          15
/num17/en1          1
/num17/a2           985
/num17/b2           999
/num17/abar2        15
/num17/bbar2        1
/num17/c2           984
/num17/e2           984000
/num17/g2           984000
/num17/d2           15
/num17/nikhilam2    984015
/num17/urdhva2      0
Now                 200 ns
Cursor 1            93 ns

                          Msgs
/num17/ab2          15
/num17/en2          1
/num17/a3           9985
/num17/b3           9999
/num17/abar3        15
/num17/bbar3        1
/num17/c3           9984
/num17/e3           99840000
/num17/g3           99840000
/num17/d3           15
/num17/nikhilam3    99840015
/num17/urdhva3      0
/num17/ab3          15
/num17/en3          1
/num17/a4           99985
/num17/b4           99999
/num17/abar4        15
/num17/bbar4        1
/num17/c4           99984
/num17/e4           9998400000
/num17/g4           9998400000
/num17/d4           15
Now                 200 ns
Cursor 1            93 ns

                          Msgs
/num17/nikhilam4    9998400015
/num17/urdhva4      0
/num17/ab4          15
/num17/en4          1
/num17/a5           999985
/num17/b5           999999
/num17/abar5        15
/num17/bbar5        1
/num17/c5           999984
/num17/e5           999984000000
/num17/g5           999984000000
/num17/d5           15
/num17/nikhilam5    999984000015
/num17/urdhva5      0
/num17/ab5          15
/num17/en5          1
/num17/a6           9999985
/num17/b6           9999999
/num17/abar6        15
/num17/bbar6        1
/num17/c6           9999984
/num17/e6           99999840000000
Now                 200 ns
Cursor 1            93 ns
```



```
                      Msgs
/num17/g6             99999840000000
/num17/d6             15
/num17/nikhilam6      9999984000001 5
/num17/urdhva6        0
/num17/ab6            15
/num17/en6            1
/num17/a7             99999985
/num17/b7             99999999
/num17/abar7          15
/num17/bbar7          1
/num17/c7             99999984
/num17/e7             9999998400000000
/num17/g7             9999998400000000
/num17/d7             15
/num17/nikhilam7      999999840000001 5
/num17/urdhva7        0
/num17/ab7            15
/num17/en7            1
/num17/a8             999999985
/num17/b8             999999999
/num17/abar8          15
/num17/bbar8          1
              Now     200 ns
         Cursor 1     93 ns

                      Msgs
/num17/c8             999999984
/num17/e8             9999999840000...
/num17/g8             9999999840000...
/num17/d8             15
/num17/nikhilam8      9999999840000...
/num17/urdhva8        0
/num17/ab8            15
/num17/en8            1
/num17/a9             9999999985
/num17/b9             9999999999
/num17/abar9          15
/num17/bbar9          1
/num17/c9             9999999984
/num17/e9             99999999984000...
/num17/g9             99999999984000...
/num17/d9             15
/num17/nikhilam9      99999999984000...
/num17/urdhva9        0
/num17/ab9            15
/num17/en9            1
              Now     200 ns
         Cursor 1     93 ns
```

Fig. 1.9. Simulation result of Proposed 64x64 Integrated Multiplier Architecture

Fig.1.9. shows the output of the proposed Integrated Multiplier Architecture. This clearly shows that based on the conditions, only one multiplier sutra performs the multiplication at any given time. When the inputs are large and close to the base such as 100, 1000, 10000, etc., Nikhilam sutra does the multiplication and saves time. When any other normal input is given (i.e.) when the inputs are small, Nikhilam sutra stops working and Urdhva multiplier performs the multiplication. Thus this proposed paper becomes an Integrated Vedic Multiplier Architecture which ensures that the best multiplier is performing the multiplication and saves time depending on the given inputs.

B. Speed Analysis Report

The designs of 8x8 bits, 16x16 bits, 32x32 bits and 64x64 bits Vedic multiplier have been implemented on *Xilinx ISE 11 series* for a series of ten multiplicands each, which fall in the Nikhilam range. It is therefore seen from Table.1.that, on an average, in case of 8 bit multipliers, Urdhva performs better than Nikhilam because of the small size of the multiplicands. However, as the size of the multiplicands increase, Nikhilam performs much faster than Urdhva & achieves an increase of 135.04% (more than twice as fast) in speed for 64 bit multiplicands. The speed analysis is as shown in Table.1.

Size / Type	8x8	16x16	32x32	64x64
Urdhva	13.455 ns	25.083 ns	44.667 ns	77.132 ns
Nikhilam	18.699 ns	20.094 ns	24.075 ns	32.816 ns
% increase in speed	Urdhva faster by 38.97%	Nikhilam faster by 24.82%	Nikhilam faster by 85.53%	Nikhilam faster by 135.04%

Table.1. Synthesis Report

V. CONCLUSION AND FUTURE SCOPE

The time taken for multiplication operation is reduced by employing the Vedic algorithms. Here integrated Vedic multiplier architecture is proposed for further reduction in time. Depending on the inputs, the better sutra is selected by the architecture itself. Future work includes the integration of the divider block, multiply and accumulate (MAC) unit, thereby making it into a Vedic Arithmetic and Logical unit (ALU).

REFERENCES

[1] Prabir Saha, Arindham Banerjee, Partha Battacharyya, Anup Dhandapat, “High speed design of complex multiplier using Vedic mathematics”, *Proceedings of the 2011 IEEE students technology symposium, IIT Kharagpur, pp. 237-241, Jan.* 2011.

[2] Purushottam D. Chidgupkar and Mangesh T. Karad, “The Implementation of Vedic Algorithms in Digital Signal Processing”, *UICEE, Global J. of Engng. Educ., Vol.8, No.2, pp. 153-158, 2004.*

[3] Himanshu Thapliyal and Hamid R. Arabnia, “A Time-Area- Power Efficient Multiplier and Square Architecture Based On Ancient Indian Vedic Mathematics”, *Department of Computer Science, The University of Georgia, 415 Graduate Studies Research Center Athens, Georgia 30602-7404, U.S.A.*

[4] E. Abu-Shama, M. B. Maaz, M. A. Bayoumi, “A Fast and Low Power Multiplier Architecture”, *The Center for Advanced Computer Studies, The University of Southwestern Louisiana Lafayette, LA 70504.*

[5] Harpreet Singh Dhillon and Abhijit Mitra, “A Reduced- Bit Multiplication Algorithm for Digital Arithmetics”, *International Journal of Computational and Mathematical Sciences ,2008.*

[6] Shamim Akhter, “VHDL Implementation of Fast NXN Multiplier Based on Vedic Mathematics”, *Jaypee Institute of Information Technology University, Noida, 201307 UP, INDIA, 2007 IEEE.*

[7] Himanshu Thapliyal, Saurabh Kotiyal and M. B Srinivas, “Design and Analysis of A Novel Parallel Square and Cube Architecture Based On Ancient Indian Vedic Mathematics”, *Centre for VLSI and Embedded System Technologies, International Institute of Information Technology, Hyderabad, 500019, India, 2005 IEEE.*

[8] Jagadguru Swami Sri Bharati Krishna Tirthji Maharaja,“Vedic Mathematics*”, Motilal Banarsidas, Varanasi, India, 1986.*

[9] Himanshu Thapliyal and M.B Srinivas, “VLSI Implementation of RSA Encryption System Using Ancient Indian Vedic Mathematics”, *Center for VLSI and Embedded System Technologies, International Institute of Information Technology Hyderabad-500019, India.*

[10] Abhijeet Kumar, Dilip Kumar, Siddhi, “Hardware Implementation of 16*16 bit Multiplier and Square using Vedic Mathematics”, *Design Engineer, CDAC, Mohali.*

ABOUT THE AUTHORS

Ramachandran.S received his Bachelor of Engineering in Electronics and Communication Engineering from Anna University, Chennai in 2009. He is currently Pursuing Master of Technology in VLSI Design in Amrita University, Bangalore.

His main research interests include VLSI design, Digital design.

Kirti.S.Pande was born in Maharashtra, India on March 16, 1980. She graduated from Nagpur University in 2001. She completed post graduation from Amrita Vishwa VidyaPeetham in 2010. She Worked as a lecturer at various universities like Nanded University and Visveshwaraya Technological University. Presently she is working as an Asst. Professor at Amrita Vishwa VidyaPeetham and has been engaged with research in VLSI design, Processor based design and FSM based Digital design.