Binary Division Algorithm and High Speed Deconvolution Algorithm (Based on Ancient Indian Vedic Mathematics)

Surabhi Jain, Mukul Pancholi, Harsh Garg, Sandeep Saini
Department of Electronics and Communication
The LNM Institute of Information Technology
Jaipur, India
Email: [ahinsa02, mukulpancholi, harsh2306.garg]@gmail.com, sandeep.saini@lnmiit.ac.in

Abstract—**The performance of any processor solely depends upon its power, area and delay. In order to get an effective processor, its power, area and delay should be less. Division is always considered to be bulky and one of the most difficult operations in arithmetic and hence all the implementations of division algorithms in VLSI architecture have higher orders of time and space complexities. Vedic Mathematics on the other hand offers a new holistic approach to mathematics. Its range extends from the most concrete values of numerical computation to the most abstract aspects of the dynamics of intelligence. In this work we have implemented an optimized binary division architecture using sutras of Vedic Mathematics which are Nikhilam Sutra and Parvartya Sutra. This work discusses about these two algorithms of division and their application for calculating deconvolution. Both the algorithms have been implemented with improved results of time delay and are with fewer complexities. The proposed division algorithm is coded in Verilog, synthesized and simulated using Xilinx ISE design suit 14.2. Simulated results for proposed Vedic divider circuit shows a reduction in delay of 19% than the conventional method.**

Keywords*—*Binary Division, Vedic Mathematics, Nikhilam, Parvartya, Deconvolution.

I. INTRODUCTION

Binary division operation is of immense importance in the field of hardware implementation of signal processing. Inherently, division operation is a sequential type of operation, thereby it is more costly in terms of computational complexity and latency (propagation delay) compared with other mathematical operations like multiplication and addition [1]. Various algorithms have been used and implemented in past in order to make division process more efficient and effective such as digit recurrence implementation methodology (restoring, non-restoring), division by convergence method (Newton-Raphson method), division by series expansion (Goldschmidt algorithm) [2]. The cost in terms of area and computational complexity of digit recurrence algorithms is low due to the large number of iterations; therefore, latency (propagation delay) becomes high [3]. While, some of the investigator rely on higher radix implementation of digit recurrence algorithm to reduce the iteration, therefore the latency becomes improved from the earlier reports, but these scheme additionally increases the hardware complexity.

In algorithmic and structural levels, a lot of division techniques had been developed in order to establish better result in terms of reduction in latency of the divider circuitry; which reduces the iteration aiming to reduction of latency but the principle behind division was same in all cases. Vedic Mathematics is the ancient system of Indian mathematics which has a unique technique of calculations based on 16 sutras or principles [4]. These principles are general in nature and can be applied in many ways and in almost any branch of mathematics. The computational methods of Vedic Mathematics have been found to be easier to learn, faster to implement than conventional methods. In this paper we report on two division algorithms and their architectures based on such ancient Indian Mathematics. "Nikhilam Navatascaramam Dasatah" (NND) [1] is a Sanskrit term indicating "all from 9 and last from 10" and "Paravartya Method" [3] is also a Sanskrit term indicating "all Transpose and apply", are adopted from Vedas; formulas are encountered to implement the division circuitry. By employing the Vedic methodology division has been implemented by multiplication and addition, therefore reduces the iteration, owing to the substantial reduction in propagation delay.

With the help of these division algorithms a fast method for computing the linear deconvolution of two finite length sequences is also introduced in this paper. Method is explained in detail in [5]. This method is similar to computing long-hand division and polynomial division.

II. VEDIC MATHEMATICS BASED DIVISION ALGORITHM

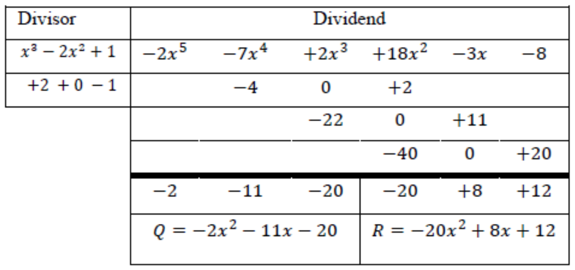
Ancient Indian Vedic Mathematics deals mainly with sixteen Sutras and their applications for carrying out tedious and cumbersome arithmetical operations, and to a very large extent, executing them mentally. Paravartya Sutra and Nikhilam Sutra are two of these sixteen Sutras used for division, in Vedic Mathematics [4].

A. Paravartya Sutra

This paper presents an algorithm to perform binary number division based on the Paravartya Sutra. Synthetic division [6], which is a short cut method for dividing a polynomial by a linear divisor of the form general division can be considered as a special case of the Paravartya Sutra [4]. The Polynomial Remainder Theorem states that if polynomial $(ax^n + bx^{n-1} + cx^{n-2} + dx^{n-3})$ is divided by (x-p) then the remainder is

R = $(ap^n + bp^{n-1} + cp^{n-2} + dp^{n-3})$

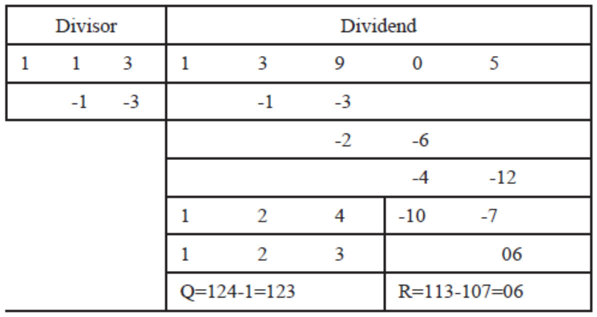
The Paravartya Sutra however, provides us with both Quotient and Remainder as illustrated in examples below.



Divisor	Dividend					
x^3-2x^2+1	$-2x^5$	$-7x^4$	$+2x^3$	$+18x^2$	$-3x$	-8
$+2\ +0\ -1$		-4	0	$+2$		
			-22	0	$+11$	
				-40	0	$+20$
	-2	-11	-20	-20	$+8$	$+12$
	$Q=-2x^2-11x-20$			$R=-20x^2+8x+12$		

Fig. 1. Example illustrating Paravartya Sutra for polynomial division

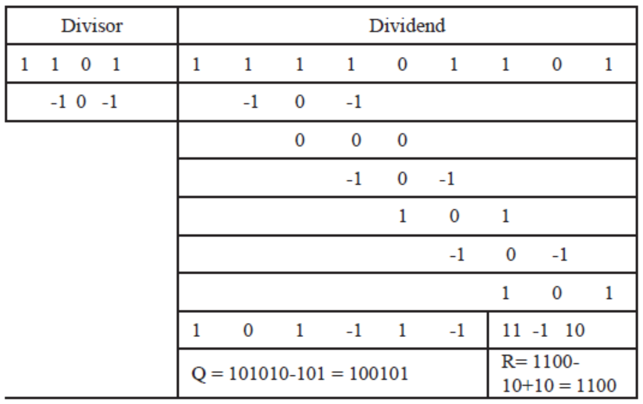
Unlike Synthetic division, the divisor in Fig. 1 is non linear. This helps illustrate the scope of the Paravartya Sutra [7]. Following the representation similar to the case of polynomial division, Fig. 2 illustrates the Paravartya Sutras application to decimal number division. We have taken 13905 as dividend and 113 as our divisor. Using regular repetitive subtraction method it takes around 124 cycles of calculations but in our proposed vedic division it takes only around 8 cycles of calculations, which is a significant reduction in computation.



Divisor	Dividend				
1 1 3	1	3	9	0	5
-1 -3		-1	-3		
			-2	-6	
				-4	-12
	1	2	4	-10	-7
	1	2	3		06
	Q=124-1=123			R=113-107=06	

Fig. 2. Application of Paravartya Sutra to Decimal Number System

Now we have extended the Sutra to perform Binary Number Division as illustrated in Fig. 3.



Divisor	Dividend								
1 1 0 1	1	1	1	1	0	1	1	0	1
-1 0 -1		-1	0	-1					
			0	0	0				
				-1	0	-1			
					1	0	1		
						-1	0	-1	
							1	0	1
	1	0	1	-1	1	-1	11	-1	10
	Q = 101010-101 = 100101						R= 1100-10+10 = 1100		

Fig. 3. Extention of Paravartya Sutra to Binary Number System

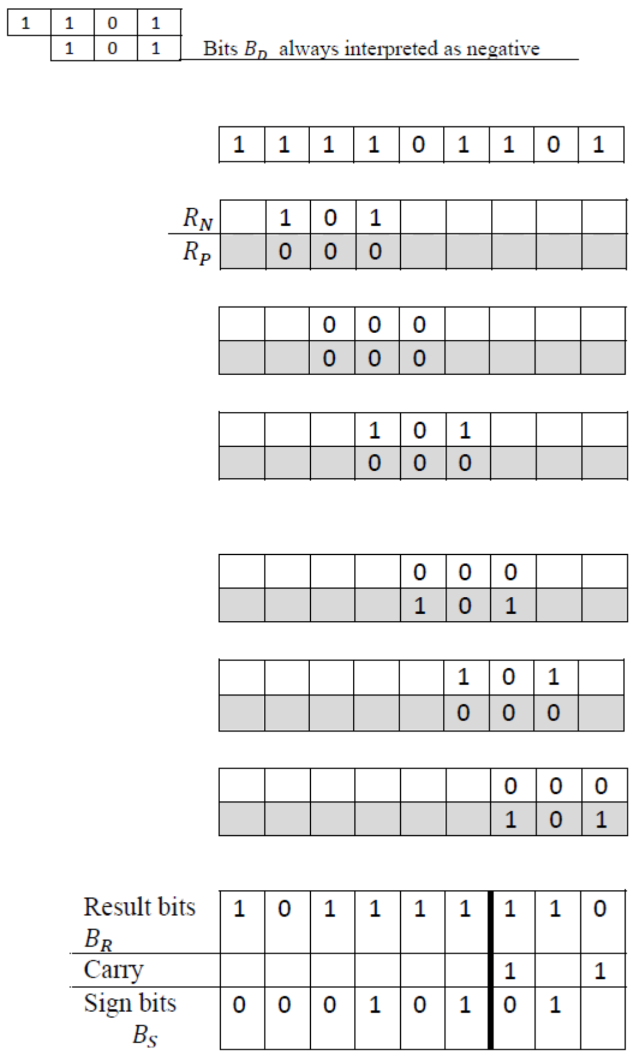
1) Hardware Implementation of Paravartya Sutra: The binary form of Paravartya Sutra is implemented using Dedicated Register Bank Structure. Fig. 4 shows one such register bank. At each stage subtract contents of RegN bit wise vertically and add contents of RegP bit wise vertically. This explains hardware representation of -1.

TABLE I. REPRESENTATION OF 0, 1,-1

	0	1	-1
RegN	0	0	1
RegP	0	1	0

TABLE II. TRUTH TABLE FOR HDL IMPLEMENTATION

Divisor Bit(B_d)	Result Bit(B_r)	Sign bit(B_s)	Interpretation
0	X	X	0
1	0	0	0
1	1	0	-1
1	1	1	1



1	1	0	1	
	1	0	1	Bits B_D always interpreted as negative

	1	1	1	1	0	1	1	0	1
R_N		1	0	1					
R_P		0	0	0					
			0	0	0				
			0	0	0				
				1	0	1			
				0	0	0			
					0	0	0		
					1	0	1		
						1	0	1	
						0	0	0	
							0	0	0
							1	0	1
Result bits B_R	1	0	1	1	1	1	1	1	0
Carry							1		1
Sign bits B_S	0	0	0	1	0	1	0	1	

Fig. 4. Expanded explanation of the register contents for the above example

The hardware can be reduced further by retaining only two of the shown registers at the Result position, one being the current register and the other being the Result Register which is continuously refreshed at every stage of computation. The Result Register will show final result when computations for all bits have been done [7].

B. Nikhilam Sutra

The Nikhilam sutra goes as follows: Nikhilam Navatascaramam Dasatah, literally meaning all from 9 And the last from 10. Nikhilam division algorithm just involves the addition of numbers which is very much different from the traditional division technique including multiplication of big numbers by the trial digit of the quotient at each step and subtract that result from dividend at each step. To illustrate the method further, we will take an example. Let us work out 1123/88.

$$\begin{array}{ll|ll}
88 & & 12 & \\
1 & 1 & 2 & 3 \\
0 & 1 & 2 & 4 \\
0 & 0 & 2 & \\
\hline
1 & 2 & 6 & 7
\end{array}$$

The algorithm can be implemented as follows:

Step 1: Assuming Dividend is equal to 1123 and Divisor is equals to 88. Considering base of operation is equals to 100. Subtract the divisor from base of operation i.e. equals to 12.

Step 2: Take the first digit (MSD) of the dividend, put down below the vertical line; here MSD is equal to 1.

Step 3: Multiply the subtraction results with MSD, put down below the dividend. Result is equal to (1x12=12). (Here individual digit multiplication has been performed).

Step 4: The product of the left-most digit of the 10's complement with the sum goes under the next digit of the numerator, while the product of the digit to its right with the sum goes under the next digit to the right and so on.

Step 5: Perform the addition of the multiplication result with dividend digits. Thus from the above chart our quotient is equal to 12 and remainder is equals to 67.

Step 6: If remainder is greater than the divisor, we divide the remainder by the divisor and add the new quotient to the original quotient and retain the new remainder as the final remainder.
The same method is extended for other numbers. Thus, in our division process by the Nikhilam formula, we perform only small single-digit multiplication; we do no subtraction and no division at all; and yet we readily obtain the required quotient and the required remainder [4].

1) Flow chart diagram of the proposed Nikhilam divider: Flow chart diagram of the proposed divider using Nikhilam formula for 8 bit by 4 bit which has been adopted from ancient Vedic Mathematics has been shown in Fig. 5. The architecture for division using 'Nikhilam' Sutra consists of four major sub-segments:- (i) Complement circuitry, (ii) Adder and (iii) Incrementer (iv) Multiplier. Assume that, A and B are dividend and divisor respectively. The 'n' bit input from divisor is fed to the complement circuitry. Complement methodology that has been used here, is the two's complement method.

The result of complement is fed to the multiplier with 4 MSB of dividend and same 4 MSB of dividend is fed to the incrementer which is initialized by '0'. The result of multiplier is fed to the adder with 4 LSB of dividend. If the result of the adder is greater than divisor then the output from the adder is again fed to the multiplier as a new dividend.

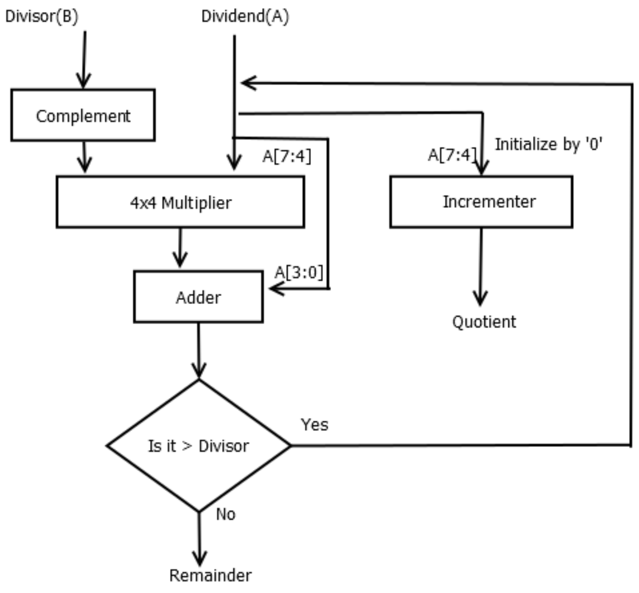


Fig. 5. Flowchart diagram of division operation using Nikhilam Sutra

The operation is repeated again until the result of the incrementer is either n/2 bit or n/2+1 bit. The output from the adder is the actual remainder and the result of the incrementer is the quotient.

III. Deconvolution using vedic maths

In general, the object of deconvolution is to find the solution of a convolution equation of the form:

$$y(n) \quad = \quad f(n) * g(n) \tag{1}$$

Usually, y is some recorded signal, and f is some signal that wish to recover, but has been convolved with some other signal g before get recorded. If one know g(n), then one can perform deterministic deconvolution. If the two sequences f(n) and g(n) are causal, then the convolution sum is given by as [8]

$$y(n) \quad = \quad \sum_{k=0}^{n} f(k)g(n-k), \quad n \geq 0 \tag{2}$$

Therefore, solving for f(n) given g(n) and y(n) results in

$$f(n) \quad = \quad \frac{y(n) - \sum_{k=0}^{n-1} f(k)g(n-k)}{g(0)}, \quad n \geq 1 \tag{3}$$

where

$$f(0) \quad = \quad \frac{y(0)}{g(0)} \tag{4}$$

where the solution requires that g(0) $\neq$ 0.

In this section, a basic recursive deconvolution method using Vedic maths for finite length sequences is computed. This recursion can be carried out in a manner similar to long division [5]. Division operation is implemented by algorithms based on Vedic Mathematics while to obtain partial products vedic multiplier is used [9]. Lets take example, let y[n] = (16 28 34 37 17 12) and g[n] = (4 5 3 4), solving for f(n) given g(n) and y(n). The sequences are set up in a fashion similar to long division, as shown below, but where no carries are performed out of a column.

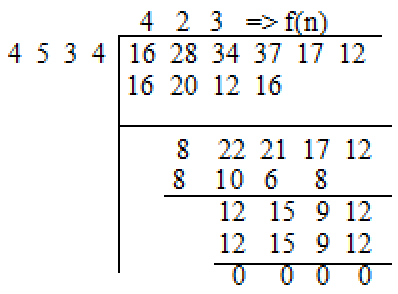


Fig. 6. Deconvolution by proposed method

By observing Fig. 6 one can easily predict, for implementing speedy deconvolution by above method, divider, multiplier and adder(to achieve subtraction in form of addition) used in design must be speedy [10]. High speed division can be carried out by selecting proper division algorithm. Division operation is implemented by using vedic divider algorithms which are Nikhilam and Paravartya and compare the result of both. From the result it is observed that when the divisor is near to base but less than the base then Nikhilam sutra is more suitable and when divisor is greater than the base then Paravartya sutra is more suitable for implementing division.

IV. Simulations and Results

The vedic division algorithm proposed in this paper is simulated and synthesised using the Xilinx Design Suit 14.2 with the device family as Virtex5 and device XC5VLX20T-2ff323. The table 3 and table 4 below show the synthesis report of the proposed work of Nikhilan and Paravartya algorithm for division using vedic maths with the logic resource utilization.

TABLE III. Device Utilization Summary for Nikhilam Sutra

Logic Utilization	Used	Available	Utilization
No. of Slice LUTs	197	12480	1%
No. of Bounded IOBs	26	172	15%

TABLE IV. Device Utilization Summary for Paravartya Sutra

Logic Utilization	Used	Available	Utilization
No. of Slices	72	12480	0%
No. of Bounded IOBs	22	172	12%

Table 5 shows delay comparison of proposed circuit of division using two algorithms of vedic mathematics (Nikhilam and Paravartya) with the division circuit implemented using Non Restoring method for 8 bit by 4 bit division. From the table it is clear that proposed Parvartya method provides 19% delay improvement than Non Restoring method.

TABLE V. Time delay for different division methods

Method	Delay
Non Restoring Algorithm [10]	17.911ns
Nikhilam Algorithm	35.787ns
Paravartya Algorithm	14.452ns

The simulated results of Nikhilam Algorithm, Paravartya Algorithm and Deconvolution are shown in the following figures:

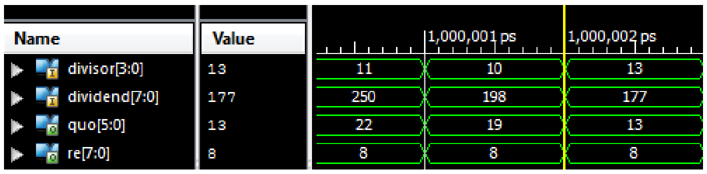

```
Name            Value   |1,000,001 ps        |1,000,002 ps
divisor[3:0]    13      11      10           13
dividend[7:0]   177     250     198          177
quo[5:0]        13      22      19           13
re[7:0]         8       8       8            8
```

Fig. 7. Simulation result of Nikhilam Sutra

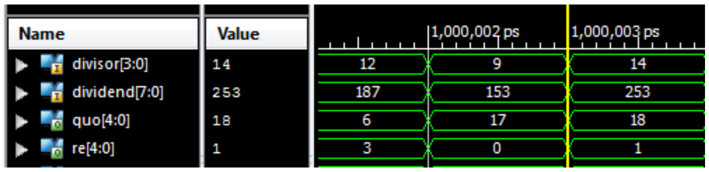

```
Name            Value   |1,000,002 ps        |1,000,003 ps
divisor[3:0]    14      12      9            14
dividend[7:0]   253     187     153          253
quo[4:0]        18      6       17           18
re[4:0]         1       3       0            1
```

Fig. 8. Simulation result of Parvartya Sutra

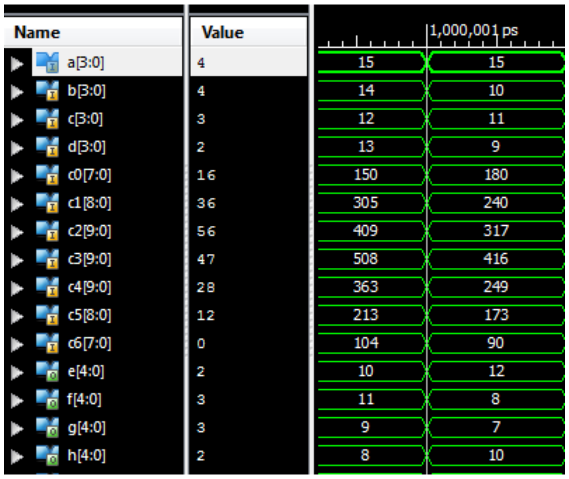

```
Name      Value   |1,000,001 ps
a[3:0]    4       15     15
b[3:0]    4       14     10
c[3:0]    3       12     11
d[3:0]    2       13     9
c0[7:0]   16      150    180
c1[8:0]   36      305    240
c2[9:0]   56      409    317
c3[9:0]   47      508    416
c4[9:0]   28      363    249
c5[8:0]   12      213    173
c6[7:0]   0       104    90
e[4:0]    2       10     12
f[4:0]    3       11     8
g[4:0]    3       9      7
h[4:0]    2       8      10
```

Fig. 9. Simulation result of Deconvolution

Table 6 shows comparison of delay for deconvolution using vedic division algorithm over deconvolution using non restoring division algorithm.

TABLE VI. Comparison of Delay for Deconvolution using Vedic Division versus Non Restoring Division

Method	Delay
Deconvolution using Non Restoring Algorithm [9]	84.262ns
Deconvolution using Paravartya Algorithm	57.352ns

From the table it is clear that proposed deconvolution method provides 31% delay improvement than deconvolution using non restoring method.

V. Conclusion

The main focus of this paper is to introduce methods for calculating the division with the help of vedic algorithms that is easy to learn and perform. From the simulated results it is observed that time delay of vedic divider architecture is reduced by approximately 19% than the conventional method. This paper also introduced a straightforward approach for performing the deconvolution. The application of vedic division in this straightforward approach of deconvolution is also introduced in this paper which has less delay of 31% than the conventional method.

References

[1] Saha, Prabir, et al. ”Vedic Divider: Novel Architecture (ASIC) for High Speed VLSI Applications.” Electronic System Design (ISED), 2011 International Symposium on. IEEE, 2011.

[2] Obermann, Stuart F., and Michael J. Flynn. ”Division algorithms and implementations.” Computers, IEEE Transactions on 46.8 (1997): 833-854.

[3] Senapati, Ratiranjan, Bandan Kumar Bhoi, and Manoranjan Pradhan. ”Novel binary divider architecture for high speed VLSI applications.” Information & Communication Technologies (ICT), 2013 IEEE Conference on. IEEE, 2013.

[4] Jagadguru Swami Sri Bharati Krishna Tirthaji Maharaja, Vedic Mathematics. Motilal Banarsidass, New Delhi, India, 1994.

[5] Pierre, John W. ”A novel method for calculating the convolution sum of two finite length sequences.” Education, IEEE Transactions on 39.1 (1996): 77-80.

[6] Fan, Lianghuo. ”A generalization of synthetic division and a general theorem of division of polynomials.” Mathematical Medley 30.1 (2003): 30-37.

[7] Joglekar, Ashish, Shaunak Vaidya, and Ajinkya Kale. ”A Novel Binary Division Algorithm Based On Vedic Mathematics And Applications To Polynomial Division.” CSC. 2008.

[8] J. G. Proakis and D. G. Manolakis, Digital Signal Processing: Principles, Algorithm, and Applications, 2nd Edition. New York Macmillan, 1992.

[9] Lomte, Rashmi K., and P. C. Bhaskar. ”High Speed Convolution and Deconvolution Using Urdhva Triyagbhyam.” VLSI (ISVLSI), 2011 IEEE Computer Society Annual Symposium on. IEEE, 2011.

[10] Lomte, Rashmi K., and P. C. Bhaskar. ”Speedy Deconvolution using Vedic Mathematics.” International Journal of Scientific and Engineering Research 2.5 (2011): 115-118.