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

Surabhi Jain
Electronics and Communication Department
The LNM Institute of Information Technology
Jaipur, India
Email: ahinsa02@gmail.com

Sandeep Saini
Electronics and Communication Department
The LNM Institute of Information Technology
Jaipur, India
Email: sandeep.saini@lnmiit.ac.in

***Abstract*—In Digital Signal Processing, the convolution and deconvolution with a very long sequence is ubiquitous in many application areas. The basic blocks in convolution and deconvolution implementation are multiplier and divider. They consume much of time. This paper presents a direct method of computing the discrete linear convolution, circular convolution and deconvolution. The approach is easy to learn because of the similarities to computing the multiplication of two numbers. The most significant aspect of the proposed method is the development of a multiplier and divider architecture based on Ancient Indian Vedic Mathematics sutras Urdhvatriyagbhyam and Nikhilam algorithm. The results show that the implementation of linear convolution and circular convolution using vedic mathematics is efficient in terms of area and speed compared to their implementation using conventional multiplier & divider architectures. The coding is done in VHDL. Simulation and Synthesis are performed using Xilinx ISE design suit 14.2. Simulated results for proposed 4x4 bit Vedic convolution circuit shows a reduction in delay of 88% than the conventional method and 41% than the OLA method.**

***Keywords*—Linear Convolution, Circular Convolution, Deconvolution, Vedic Mathematics, Urdhva Triyagbhyam, Nikhilam, VHDL.**

I. Introduction

With the latest advancement of VLSI technology, digital signal processing plays a pivotal role in many areas of electrical engineering. Discrete convolution is central to many applications of Digital Signal Processing and Image Processing. It is used for designing of digital filter and correlation application. However, beginners often struggle with convolution because the concept and computation requires a number of steps that are tedious and slow to perform. The most commonly taught approach is a graphical method because of the visual insight into the convolution mechanism. Graphical convolution is very systematic to compute but is also very tedious and time consuming [1]. The principal components required for implementation of convolution calculation are adder and multiplier for partial multiplication. Therefore the partial multiplication and addition are bottleneck in deciding the overall speed of the convolution implementation technique. Complexity and excess time consumption are always the major concern of engineers which motivates them to focus on more advance and simpler techniques. Pierre and John [2] have implemented a fast method for computing linear convolution, circular convolution and deconvolution. This method is similar to the multiplication of two decimal numbers and this similarity makes this method easy to learn and quick to compute. Also to compute deconvolution of two finite length sequences, a novel method is used. This method is similar to computing long-hand division and polynomial division.

As a need of proposed method, all required possible adders are studied. All these adders are synthesized using Xilinx Design Suit 14.2. Their delays and areas are compared. Adders which have the highest speed and occupy a comparatively less area, are selected for implementing convolution. Since the execution time in most DSP algorithms mainly depends upon the time required for multiplication, so there is a need of high speed multiplier. Now a days, time required in multiplication process is still the dominant factor in determining the instruction cycle time of a DSP chip [3]. Traditionally shift and add algorithm is being used for designing. However this is not suitable for VLSI implementation and also from delay point of view. Some of the important algorithms proposed in literature for VLSI implementable fast multiplication are Booth multiplier, array multiplier and Wallace tree multiplier [4]. Although these multiplication techniques have been effective over conventional "shift and add" technique but their disadvantage of time consumption has not been completely removed. Vedic Mathematics provides unique solution for this problem. The Urdhva Triyagbhyam Sutra or Vertically and Crosswise Algorithm for multiplication is discussed and then used to develop digital multiplier architecture. For division, different division algorithms are studied, by comparing drawbacks and advantages of each algorithm, Nikhilam Algorithm based on vedic mathematics is modified according to need and then used.

II. PROPOSED ALGORITHM USING VEDIC MATHS

Vedic mathematics is an ancient Vedic mathematics which provides the unique technique of mental calculation with the help of simple rules and principles. It is based on sixteen sutras which transact different branches of mathematics i.e. algebra, geometry, arithmetic etc. In this paper the algorithms of vedic mathematics are used to design multiplier and divider. With the help of these algorithms convolution, circular convolution and deconvolution are implemented.

978-1-4799-2993-1/14/$31.00 ©2014 IEEE

A. Convolution

In this section a novel multiplier architecture [5] based on Urdhva Triyagbhyam Sutra of Ancient Indian Vedic Mathematics is embedded into proposed method of convolution to improve its efficiency in terms of speed and area. This method for discrete convolution using vedic multiplication algorithm is best introduced by a basic example. For this example, let f(n) equal the finite length sequence (4 2 3) and g(n) equal the finite length sequence (4 5 3 4). The linear convolution of f(n) and g(n) is given by [1]:

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

$$y(n) = \sum_{k=-\infty}^{\infty} f(k)g(n-k) \quad (2)$$

This can be solved by several methods, resulting in the sequence y(n) = (16 28 34 37 17 12). This new approach for calculating the convolution sum is set up like multiplication where the convolution of f(n) and g(n) is performed as follows:

```
g(n):          4  5  3  4
f(n):        *    4  2  3
-------------------------
             12  15  9 12
          08  10   6  8
       16 20  12  16
-------------------------
y(n):  16 28  34  37 17 12
```

Fig. 1. Convolution by proposed method

As seen in the Fig. 2 computation of the convolution sum, the approach is similar to multiplication calculation, except carries are not performed out of a column. This first example shows the simplicity of this method and how easily the calculation can be performed. As shown below, this method can be used to check intermediate values in graphical convolution, as well as the final answer. In Fig. 1, the convolution sum is computed using graphical convolution. Fig. 1(a) shows the sequences f(n) and g(n). For each value of n, the convolution sum consists of a folding, translation, multiplication, and summation. For a given value of n, the summation is a product of the sequence f(k) and the folded and translated sequence g(n-k). The left-hand column of Fig. 1(b) shows both sequences f(k) and g(n-k) for each value of n, and the right-hand column shows the product of the two sequences $v_n(k)$ which is given by

$$v_n(k) = f(k)g(n-k) \quad (3)$$

The value of the convolution sum for each value of n is

$$f(n) * g(n) = \sum_k v_n(k) \quad (4)$$

The final answer for the graphical convolution method is shown in Fig. 1(c). This answer was verified above using the new method. The sequence $v_n(k)$, which is an intermediate answer in computing the convolution sum, may also be checked as shown below using the method presented in this paper.

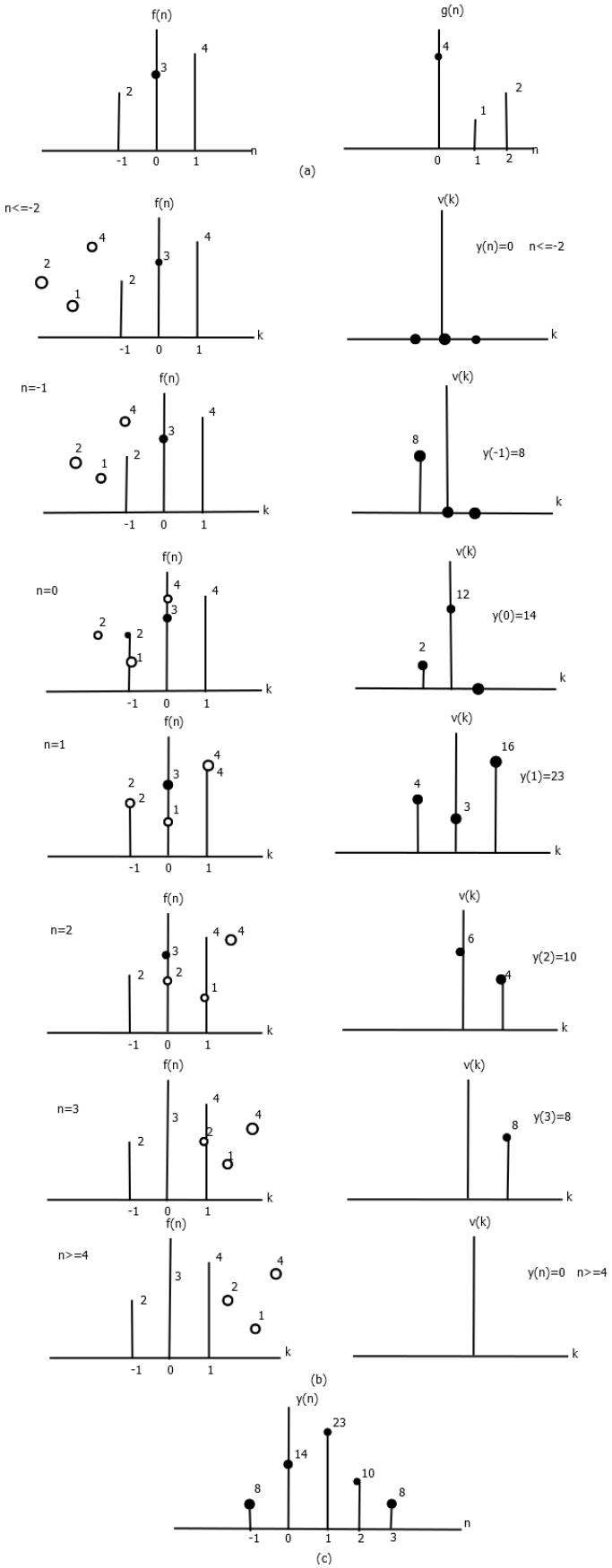

Fig. 2. The sequences f(n) and g(n), shown in (a), are graphically convolved in (b), resulting in the sequence y(n), shown in (c)

g(n):			4	1	2	
f(n):		*	2	3	4	k
v(1):			16	04	08	1
v(0):		12	03	06		0
v(-1):	08	02	04			-1
y(n):	08	14	23	10	08	
n:	-1	0	1	2	3	

Fig. 3. Verification of intermediate terms using proposed method

Above example illustrates the ease in computing the convolution sum for finite sequences using this new method.

1) Vedic Mutiplier:Urdhva Triyagbhyam: Among all available multipliers, this paper proposes a systematic design methodology for fast and area efficient digit multiplier based on Vedic Mathematics. In the proposed convolution method the multiplier architecture is based on an algorithm Urdhva Triyagbhyam (Vertical and Crosswise) of Ancient Indian Vedic Mathematics [5].

The use of Vedic Mathematics lies in the fact that it reduces the typical calculations in conventional mathematics to very simple ones. Urdhva Tiryagbhyam Sutra is a general multiplication formula applicable to all cases of multiplication [6]. Because of parallelism in generation of partial products and their summation obtained, speed is improved. In this algorithm the small block can be wisely utilized for designing bigger NxN multiplier. For higher no. of bits in input, little modification is required. Divide the no. of bit in the inputs equally in two parts. Let's analyse 4x4 multiplications, say $A_3A_2A_1A_0$ and $B_3B_2B_1B_0$. Following are the output line for the multiplication result,$S_7S_6S_5S_4S_3S_2S_1S_0$. Let's divide A and B into two parts, say $A_3A_2\&A_1A_0$ for A and $B_3B_2\&B_1B_0$ for B. Using the fundamental of Vedic multiplication, taking two bit at a time and using 2 bit multiplier block, we can have the following structure for multiplication.

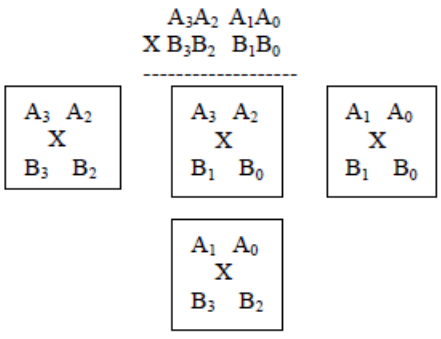

Fig. 4. Block diagram presentation for 4x4 multiplications

Each block as shown above is 2x2 multiplier. First 2x2 multiplier inputs are A_1A_0 and B_1B_0. The last block is 2x2 multiplier with inputs A_3A_2 and B_3B_2. The middle one shows two, 2x2 multiplier with inputs A_3A_2 and B_1B_0 and A_1A_0 and B_3B_2. So the final result of multiplication, which is of 8 bit, $S_7S_6S_5S_4S_3S_2S_1S_0$, can be interpreted as given below.

A_3A_2	A_3A_2	A_1A_0	A_1A_0
B_3B_2	B_1B_0	B_3B_2	B_1B_0
$S_{33}S_{32}S_{31}S_{30}$	$S_{23}S_{22}S_{21}S_{20}$	$S_{13}S_{12}S_{11}S_{10}$	$S_{03}S_{02}S_{01}S_{00}$

Assuming the output of each multiplication is as given above. For the final result, add the middle product term along with the term shown below.

$$\begin{array}{lllll} S_{33}S_{32} & S_{31} & S_{30} & 0 & 0 \quad S_{01}S_{00} \\ & S_{23} & S_{22} & S_{21} & S_{20} \\ & S_{13} & S_{12} & S_{11} & S_{10} \\ & 0 & 0 & S_{03} & S_{02} \end{array}$$

The first two outputs S_0 and S_1 are same as that of S_{00} and S_{01}. Result of addition of the middle terms by using two, 4 bit full adders will forms output line from $S_5S_4S_3S_2$. One of the full adder will be used to add $(S_{23}S_{22}S_{21}S_{20})$ and $(S_{13}S_{12}S_{11}S_{10})$ and then the second full adder is required to add the result of 1^{st} full adder with $(S_{31}S_{30}S_{03}S_{02})$. The respective sum bit of the 2^{nd} full adder will be $S_5S_4S_3S_2$. Now the carry generated during 1^{st} full adder operation and that during 2^{nd} full adder operation should be added using half adder so that the final carry and sum to be added with next stage i.e. with $S_{33}S_{32}$ to get S_7S_6. The same can be extended for input bits 8, 16, 32.

B. Circular Convolution

Circular convolution has many applications and is usually introduced to electrical engineering students in a digital signal processing. The novel method for computing linear convolution using vedic mathematics from above subsection is easily modified for circular convolution [2]. This method of computing circular convolution is best illustrated by example. Let f(n) = (2 3 1 0) and g(n) = (4 5 2 1). The circular convolution of f(n) and g(n) is given by

$$y(n) = f(n) * g(n) \quad (5)$$

$$y(n) = \sum_{k=0}^{N-1} f(k)g((n-k)modN) \quad (6)$$

y(n)= (13 13 23 23) where N is the length of the sequences. This circular convolution calculation may be performed similar to the method for linear convolution from above subsection. The multiplier architecture is implemented using vedic algorithm.

The location of the triangle of bold faced numbers is repositioned for circular convolution compared with linear convolution. The location of these numbers is due to the circular translation in circular convolution. The far left value in the circular convolution solution corresponds to y(N - 1) where N is the length of the sequence.

```
   2   3   1   0
⊗  4   5   2   1
-------------------
   2   3   1   0
   6   2   0   4
   5   0  10  15
   0   8  12   4
-------------------
  13  13  23  23

 y(3) y(2) y(1) y(0)
```

Fig. 5. Circular Convolution by proposed method

C. Deconvolution

A direct method is also presented for the deconvolution of two finite length discrete-time sequences. This deconvolution method is similar to computing long-hand division and polynomial division, just as the direct convolution method is similar to multiplication [7]. Many other deconvolution methods are available. In this section, a basic recursive deconvolution method for finite length sequences is computed. This recursion can be carried out in a manner similar to long division.

```
                4  2  3  => f(n)
4 5 3 4 | 16 28 34 37 17 12
        | 16 20 12 16
        |-------------------
        |    8  22 21 17 12
        |    8  10  6  8
        |-------------------
        |       12 15 9 12
        |       12 15 9 12
        |-------------------
        |        0  0  0  0
```

Fig. 6. Deconvolution by proposed method

Division operation is implemented by using Nikhilam algorithm based on vedic mathematics while to obtain partial products vedic multiplier is used. For instance, the first example in subsection A may be reworked, solving for f(n) given g(n) and y(n). The sequences are set up in a fashion similar to long division, as shown in Fig. 6, but where no carries are performed out of a column.

1) Vedic Divider: Nikhilam Algorithm: The Nikhilam sutra goes as follows: Nikhilam Navatascaramam Dasatah, literally meaning all from 9 And the last from 10. To illustrate the method further, we will take an example. Let us work out 123/8.

```
8      2
1 2  | 3
0 2  | 8
-------------
1 4  | 11
1 5  | 3
```

The first line consists of the denominator followed by its 10's complement (2 is 8's 10's complement). The numerator has been divided by a "|" such that there are as many digits to the right of the "|" as there are digits in the denominator. We then put a zero under the first digit of the numerator. Now add up the digits in that column of the numerator to get a sum of 1 (1 + 0 = 1). Multiply it by the 10's complement to get 2 (1 x 2 = 2). Put that under the second digit of the numerator. The sum of the digits under the second column is 4. Multiplying this by the 10's complement gives us 8. Put the 8 under the third digit of the numerator, right of the "|". Now we add up the numbers under the columns. Note that there is no carry over from the right of the "|" to the left of it. Following the rules on how to deal with a remainder greater than the denominator, we divide the remainder by the denominator 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. 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 [8].

III. SIMULATIONS AND RESULTS

The vedic convolution algorithm proposed in this paper is simulated and synthesised using the Xilinx Design Suit 14.2 with the device family as Spartan3 and device XC3S400-5fg320. The table 1 below shows the synthesis report of the proposed work for convolution using vedic maths with the logic resource utilization.

TABLE I. DEVICE UTILIZATION SUMMARY FOR CONVOLUTION

Logic Utilization	Used	Available	Utilization
No. of Slices	358	3584	9%
No. of LUTs	623	7168	8%
No. of Bounded IOBs	96	221	48%

The simulated results of convolution and circular convolution are shown below:

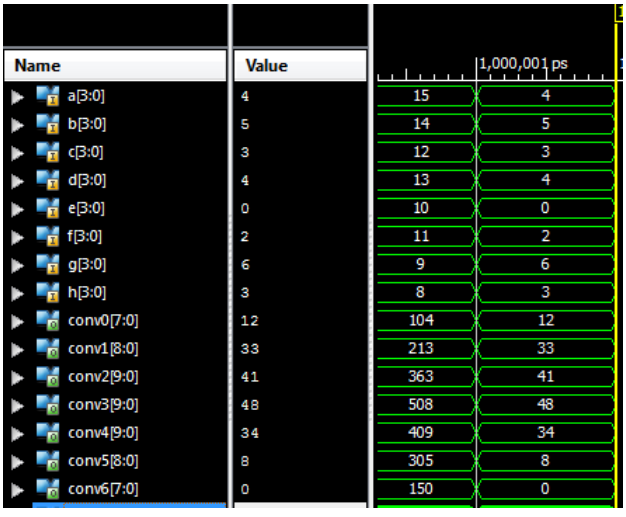

Fig. 7. Simulation results of Convolution using vedic mathematics

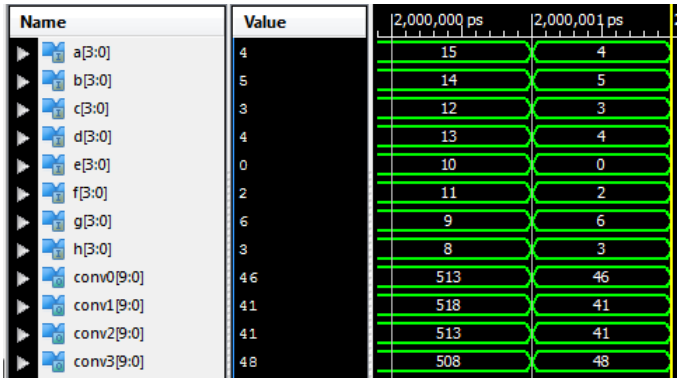

Fig. 8. Simulation results of Circular Convolution using Vedic Mathematics

TABLE II. EXECUTION TIMES FOR CONVENTIONAL METHOD, OLA METHOD USING VEDIC MATHS AND PROPOSED METHOD

Input Sequence Length	2^{16}	2^{20}
Conventional Method [7]	195.059ns	
Traditional OLA Method [9]	0.21sec	11.28sec
OLA Method using Vedic Maths [9]	36.85ns	36.92ns
Proposed Vedic Method	21.661ns	22.943ns

Table 2 shows delay improvement of proposed circuit of convolution over the circuit implemented using conventional array multiplier and overlap & add method for different input sequence length. From the table it is clear that proposed method provides 41% delay improvement than OLA method and 88% improvement than Conventional method. Table 3 shows comparison of delay for vedic circular convolution over the conventional method.

TABLE III. COMPARISON OF DELAY FOR VEDIC CIRCULAR CONVOLUTION VERSUS CONVENTIONAL CIRCULAR CONVOLUTION

Method	Delay
Conventional Method(inbuilt) [10]	62.47us
Proposed Vedic Method	21.963ns

IV. CONCLUSION

The main focus of this paper is to introduce a method for calculating the linear convolution, circular convolution and deconvolution with the help of vedic algorithms that is easy to learn and perform. The execution time and area of the proposed method for convolution using vedic multiplication algorithm is compared with that of convolution with the simple multiplication is less. From the simulated results it is observed that delay of Linear Convolution architecture is reduced by approximately 88% than the conventional method. An extension of the proposed linear convolution approach to circular convolution using vedic multiplier is also introduced which has less delay and area than the conventional method. This paper also introduced a straightforward approach to performing the deconvolution.

REFERENCES

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

[2] 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.

[3] Rudagi, J. M., Vishwanath Ambli, Vishwanath Munavalli, Ravindra Patil, and Vinaykumar Sajjan. "Design and implementation of efficient multiplier using Vedic mathematics." (2011): 162-166.

[4] Vaidya, Sumit, and Deepak Dandekar. "Delay-Power Performance Comparison of multipliers in VLSI circuit design." International Journal of Computer Networks & Communications (IJCNC) 2.4 (2010): 47-56.

[5] Akhter, Shamim. "VHDL implementation of fast NxN multiplier based on vedic mathematic." In Circuit Theory and Design, 2007. ECCTD 2007. 18th European Conference on, pp. 472-475. IEEE, 2007.

[6] Thapliyal, Himanshu, and M. B. Srinivas. "High Speed Efficient NxN Bit Parallel Hierarchical Overlay Multiplier Architecture Based on Ancient Indian Vedic Mathematics." Enformatika Trans 2 (2004): 225-228.

[7] 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.

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

[9] Hanumantharaju, M. C., et al. "A High Speed Block Convolution using Ancient Indian Vedic Mathematics." Conference on Computational Intelligence and Multimedia Applications, 2007. International Conference on. Vol. 2. IEEE, 2007.

[10] Itawadiya, Akhalesh K., et al. "Design a DSP operations using vedic mathematics." Communications and Signal Processing (ICCSP), 2013 International Conference on. IEEE, 2013.