Available online at www.elixirpublishers.com (Elixir International Journal)

Advanced Engineering Informatics

Elixir Adv. Engg. Info. 39 (2011) 4634-4638

ISSN: 2229-712X

# High speed DCT design using Vedic mathematics

N.J.R. Muniraj[1] and N.Senathipathi[2]

[1]Tejaa Shakthi Innovation Centre, Tejaa Shakthi Institute of Technology, Coimbatore

[2]PA College of Engineering, Pollach.

**ARTICLE INFO**

**Article history:**

Received: 19 July 2011;

Received in revised form:

20 September 2011;

Accepted: 26 September 2011;

**Keywords**

Vedic mathematics,

Urdhva Tiryagbhyam,

Faster approach.

**ABSTRACT**

This paper proposes a novel method of designing discrete cosine transform using Vedic mathematics. Multipliers are fundamental and area intensive component in the architecture of any DSP system. In many circumstances, there are situations where the complexity and delay of the whole circuit increases because of inefficient multipliers. So it is necessary to reduce such complexity of Multipliers for efficient DSP architecture. The design of Vedic Multiplier is aimed to create such. The design is based on the Sutras of ancient Indian Vedic mathematics which was rediscovered from Vedas between 1911 and 1918 .The whole of Vedic mathematics is based on 16 sutras and manifests unified structure of mathematics. The DCT algorithm is based on 'Urdhva-Tiryak' sutra .It is expected that the Vedic architecture reduces the space and time hence the complexity of the multiplier when implemented in digital domain. The algorithm is implemented using verilog HDL and is tested and verified. It is found to reduce the space and increases the speed by when compared to conventional dct using array multiplier. The multiplier can be substituted for conventional multipliers in various applications. The exploration of Vedic algorithms in Digital Signal Processing may prove to be extremely advantageous. Hence it can be applied for discrete cosine transform application.



## Introduction

With the rapid progress of VLSI technologies, many processors based on audio and image signal processing, have been developed recently. These processors need faster addition and multiplications which are of extreme importance in processing of images and signals. So there is always a constant need for new algorithms and efficient hardware to implement them. This can be possible by use of multiplication based on Vedic Algorithm [4].

## Vedic mathematics

Vedic Mathematics is the name given to the ancient system of Indian Mathematics. Vedic mathematics was rediscovered from the ancient Indian scriptures between 1911 and 1918 by Sri Bharati Krisna Tirthaji (1884- 1960), a scholar of Sanskrit, mathematics, history and philosophy. He studied these ancient texts for years and, after careful investigation, was able to reconstruct a series of mathematical formulae called Sutras.

Bharati Krishna Tirthaji, who was also the former Shankaracharya of Puri, India, delved into the ancient Vedic texts and establishing the techniques of this system in his pioneering work, Vedic Mathematics (1965), which is considered the starting point for all work on Vedic Mathematics [5].

## The Vedic Multiplication Method

The multiplication using Vedic mathematics [5] consists of two sutras

i)Nikilam Sutra

ii)Urdhva –Tiryak sutra

Here Urdhva –Tiryak sutra only discussed.

## The Urdhva-Tiryak Sutra

The second sutra named 'Urdhva Tiryagbhyam' is the general formula applicable to all cases of multiplication and will also be found very useful. The formula itself is very short and terse, consisting of only one compound word and means "vertically and crosswise".

The multiplication of two or three digit numbers utilizing conventional mathematical methods needs no explanation.

Alternatively, the Vedic method is illustrated for two digit number. Suppose we have to multiply 12 by 13.

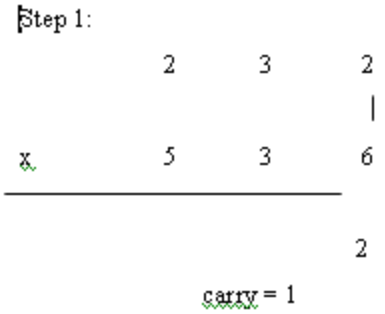
i) We multiply the left-hand-most digit 1 of the multiplicand vertically by the left hand most digit 1 of the multiplier, get their product a and set it down as the left hand most part of the answer

ii) We then multiply 1 and 3, and 1and 2 cross-wise, add the two, get 5 as the sum and set it down as the middle part of the answer; and

iii) We multiply 2 and 3 vertically, get 6 as their product and put it down as the last the right-hand –most part of the answer

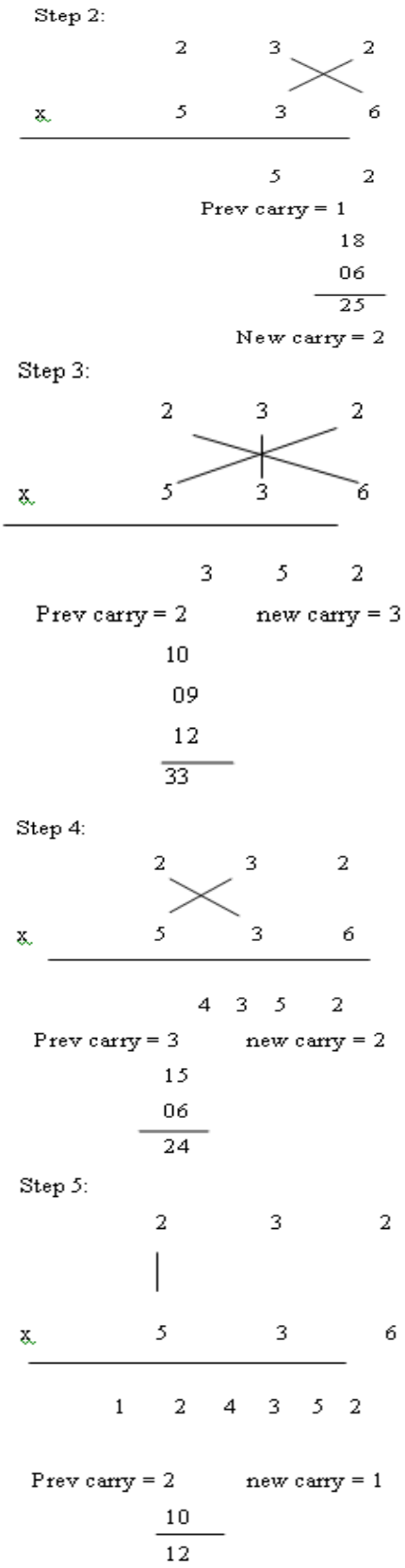
Thus 12x13=156

The following example shown below illustrates multiplication in the presence of carry. Suppose we want to multiply 232x536




Tele:

E-mail addresses: njrmuniraj@yahoo.com




i) The digits on the two sides of line are multiplied and the result is added in the previous carry.

ii) When more than one line is in the step, all the results are added with the previous carry and the process is thus continued. In this case initially, the previous carry is equal to Zero

iii) A Unit place digit of addition result is one of the digits in the answer; this is derived from full multiplication, while the remaining digits act as a carry.

iv) If the numbers of the digits are not same in the multiplier and multiplicand, then the biggest number has to be counted .The smaller number should be prep ended with 0s so that both numbers will be of same digits.

**Vedic multiplication using urdhva tiryagbhyam**

The Vedic multiplication deals with implementing a fast carry method of addition of the intermediaries generated during Vedic multiplication. All the inputs and outputs of any full adder or half adder do not contribute equally to the delay. Therefore, we sort them in a way which favors the use of fast inputs and outputs in the paths which are critical to the speed. We assign slow inputs to the signal paths which belong to the domain where an increase in delay is tolerable. In the discussed multiplier we examine the entire multiplication path i.e. all the signals that are entering the adders, which thus leads to global rather than local optimization. This has been dealt with Wang et al (1995).

The multiplication problem can be formulated as two step architecture, the first one being the formation of the partial products and the second one being the addition of the formed intermediaries In contrast to the classical method of multiplication using array multiplier, this method formulates the intermediaries using the cross products (Urdhva-Tiryagbhyam) .Once the intermediaries (referred to as the Rij's and Cij's) are formed, then fast carry method is used to calculate the final product.

A large number of methods are available for fast addition of the partial products. The method followed is a novel technique, which combines the Vedic mathematics and the critical path speed optimization. The simulations showing the speed of proposed multiplier in comparison to the standard array multipliers have been done in VHDL[1] and the results for the chosen specific device are shown later in this paper. This technique of Vedic multiplication though uses larger gate count but at the same time helps to increase in speed when compared to the array multiplier. Hence, it is beneficial in places where speed is important. The multiplication uses two techniques

i) A Novel technique

ii) Faster approach

The first step of the algorithm is the formation of the intermediaries (Rij's and Cij's) using the above method of digit representation and computation.

**Formation of Intermediaries**

The basic underlying steps in multiplication by Vedic mathematics have already been stated .The algebraic principle involved is as follows. The process of ascent and descent i.e. going forward with the digits on the upper row and coming backwards with the digits on the lower row is being followed. This is a general principle that can be applied to any length of digits. The basic methodology behind it is calculation of the digits having equal weights in one go, and thus reducing the time involved in operation.

The process of multiplication as shown in the Table .1 can be carried out from left to right or from right to left. This is general principle that can be applied to any place value system, including binary system.

The algorithm implemented is the general algorithm called Urdhva Tiryakbhyam which means vertically and cross wise.Urdhva means vertically up-down. Tiryakbhyam means left to right or vice versa

$P_0 = a_0 x b_0$

$P_1 = a_1 x b_0 + a_0 x b_1$

$P_2=a_2x\ b_0+a_0\ xb_2+a_1\ x\ b_1$
$P_3 = a_0 x\ b_3+a_3\ b_0+a_1xb_2+a_2\ x\ b_1$
$P_4= a_3\ x\ b_1+b_3\ x\ a_1+a_2\ xb_2$
$P_5 =a_3\ x\ b_2+\ a_2\ xb_3$
$P_6 = a_3\ xb_3$
$P_7, P_8$ = Carry Generated

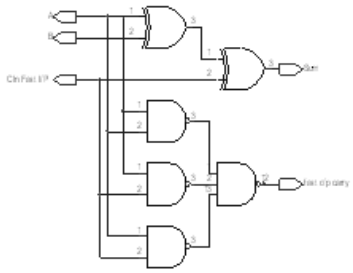


**Fig 1. Full adder**

**Realization of The Product**

Once the intermediaries are formed we form the fast carry adder. It is known that the delay from an input to an output in Full adder is not same. This delay is even dependent in a particular transition (0-1,1-0).It is also possible to come up with different realizations of a full adder where a specific signal path is favored with respect to the others is designed in such a way that a signal propagation of the final path takes a minimal amount of time. This is sometimes done even at the expense of other possible signal paths .For example a ripple carry adder is designed so that the carry in to carryout delay is minimized. In the case of parallel multiplier, the design objective would be to minimize the delay from the inputs to the sum, of the full adder which has direct effect on the critical path.[6]

In this example shown in the Fig.1, the delay from input A or B to the sum is equal to two equivalent XOR delays. The delay for the path from Cin to the output sum is equal to one XOR equivalent delay. We define Cin as a fast input. For this case, the propagation delay from A or B to the output sum is twice as long as the propagation delays from input Cin to the sum output. Considering the delay at the output sum, in this particular technology delay from input A or B to sum output is equivalent o two XOR delays. However, delay from inputs (A, B or Cin) to the out put carry is equivalent to one XOR delay. We define carry as a fast output .The value of those delays varies with technology and particular circuit implementation.

**Vertical optimization**

The advantage of proper interconnection of fast inputs and fast outputs is explained below .The Fig.2 below shows an optimized 4:2 cell which is a result of applying our delay model and properly interconnecting fast inputs and fast outputs with the objective of minimizing the critical path of the 4:2 compressors. The delay through a 4: 2 compressor level is equivalent to three XOR gate delays regardless of the path. The use of the 4:2 compressor permits the reduction of the vertical critical path while the path involving the carry propagation, that we call horizontal path, is not changed [7].

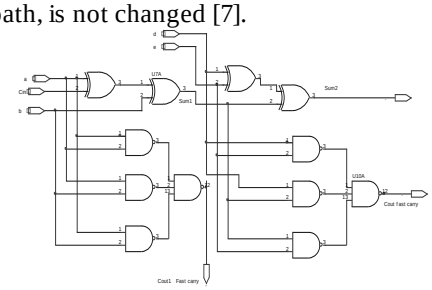


**Fig 2. Vertical Optimization**

**Horizontal Optimization**

The basic idea of this method is to make proper connection globally so that the delay throughout each path is approximately the same. The Non Optimized interconnection is shown in the Fig.3 where the connection to the next stage is done without considering the long and short delay paths.

The long delay path originating from previous compressor should be connected to the short delay path of the next one, and so on. In general, this is not always possible since each output function has its unique characteristics and requires specific logic cells in its path. It is feasible to apply this idea to apply this idea to the partial product array using full adders(FA) since all of the partial product bits are logically the same and, therefore ,interchangeable. In this fact carries generated from the nth column are fed into the fast inputs of the (n+1) th column thereby resulting in the horizontal optimization as shown in the Fig.4. The cout at bit position n in the figure is connected to the fast input i.e cin of the next stage, thereby resulting in a critical path of 3 equivalent XOR gate delays. If however this cout is connected to the slow input i.e 'b' of the next stage then the critical path increases by 1 equivalent XOR gate.

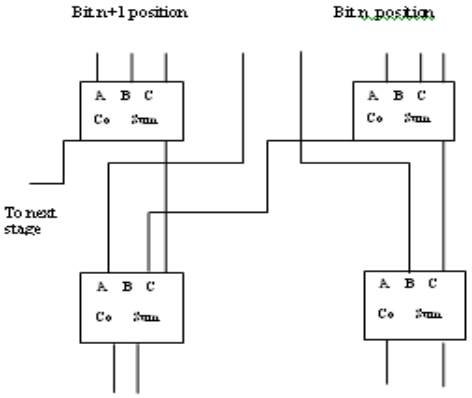



**Fig.3 Non Optimized interconnection**

Thus we optimize both the vertical as well as the horizontal paths with respect to time, resulting into a three dimensional optimization process.

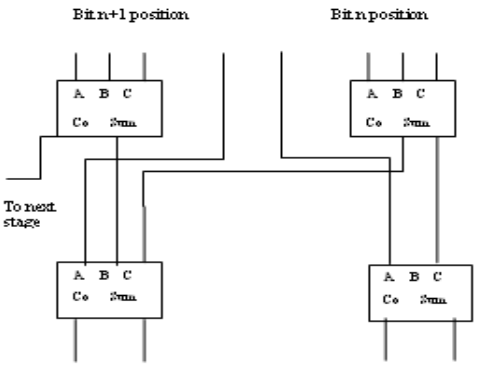



**Fig .4Optimized interconnection**

**The Discrete Cosine Transform (DCT)**

The Discrete Cosine Transform is similar to the Fourier Transform in that it transforms a signal from the spatial or time domain to the frequency domain, as in if preparing an image for compression.

The Discrete Cosine Transform (DCT) is highly suitable for transform coding of images because it attempts to decorrelate the image data.

After decorrelation each transform coefficient can be encoded independently without losing compression efficiency

**The one-dimensional DCT**

The general equation for a 1D (N data items) DCT is defined by the following equation:

$F(u) = (2/N)^{1/2} \Sigma \delta(i)[\pi.u/2.N \ (2i + 1)] f(i)$

and the corresponding inverse 1D DCT transform is simple F-1(u)

Where

$\delta(i) = 1/2$ for $\xi = 0$ , otherwise 1

**DCT design**

The one dimensional DCT matrix is shown below [6]

$$\begin{bmatrix} Y(0) \\ Y(1) \\ Y(2) \\ Y(3) \\ Y(4) \\ Y(5) \\ Y(6) \\ Y(7) \end{bmatrix} = \begin{bmatrix} C_4 & C_4 & C_4 & C_4 & C_4 & C_4 & C_4 & C_4 \\ C_1 & C_3 & C_5 & C_7 & -C_7 & -C_5 & -C_3 & -C_1 \\ C_2 & C_6 & -C_6 & -C_2 & -C_2 & -C_6 & C_6 & C_2 \\ C_3 & -C_7 & -C_1 & -C_5 & -C_5 & C_1 & C_7 & -C_3 \\ C_4 & -C_4 & -C_4 & C_4 & C_4 & -C_4 & -C_4 & C_4 \\ C_5 & -C_1 & C_7 & C_3 & -C_3 & -C_7 & C_1 & -C_5 \\ C_6 & -C_2 & C_2 & -C_6 & -C_6 & C_2 & -C_2 & C_6 \\ C_7 & -C_5 & C_3 & -C_1 & C_1 & -C_3 & C_5 & -C_7 \end{bmatrix} \cdot \begin{bmatrix} X(0) \\ X(1) \\ X(2) \\ X(3) \\ X(4) \\ X(5) \\ X(6) \\ X(7) \end{bmatrix}$$

**DCT Coefficient**

The DCT coefficient values are calculated and the values are in fractional form. The obtained fractional values are converted into two's complement binary format in order to apply Common Subexpression Elimination method. The Table 3 shows the two's complement coefficient value for DCT, using 8s7 format

**Results And Discussions**

The Vedic Multiplier is implemented on DCT structure (Virtex 2pxc2vp50ff1152).From the results we can prove that the Vedic Multiplier works efficient than conventional Array Multiplier [2]. The total space occupied is reduced to 50 % and the Propagational delay has also drastically reduced, which increase the speed by 43%. Compared to conventional mathematical methods, these are computationally faster and easy to perform.

The Synthesis and Timing report generated by different Multipliers is shown and compared below,

**Conclusion And Future Work**

Vedic mathematical methods which are derived from ancient systems of computations, now made available to everyone through the great work of Jagadguru Swami Sri Bharati Krishna Tirthaji Maharaja, who published the book. The multiplication sutra of Vedic Mathematics is tested and verified for two sutras. it is found that urdhva tiryak sutra is better and this has improved the performance such as speed compared to the array multiplier it works it proves to be better .It increases the speed by and reduces the space occupied. Therefore this approach is extremely beneficial in discrete cosine transform processing applications.

There is an overwhelming need to explore Vedic algorithms in detail so as to verify its applicability in digital image processing applications, image compression techniques different domains of engineering. The other sutras of Vedic Mathematics for division, squaring, integration etc., can be explored and can be applied in different domains of engineering and other fields.

**References**

1. Bhasker.J (2001), 'VHDL primer', BS Publications,Hyderabad,III Edition
2. Douglas A.Pucknell and Kamran Eshraghian (2001), ' Basic VLSI Design' Prentice Hall of India private Ltd., New Delhi,pp240-253.
3. M.Potkonjak. M.B.Srivastava. And A.P.Chandrakasan. (1996) 'Multiple Constant Multiplications: Efficient and Versatile Framework and Algorithms for Exploring Common Subexpression Elimination' IEEE Transactions on Computer Aided Design of Integrated Circuits and Systems
4. Design and implementation of high speed low power VLSI chip for the dct transform, Professor A.Doboli
5. Implementation of Vedic Algorithms by Prof.Puroshottam,D.chidgupkar,prof.mangeshT.karad

**Authors Biography**

N.J.R.Muniraj is presently working as a Principal of Tejaa Shakthi Institute of Technology, Coimbatore. He has more than 22 years of teaching and five years of industrial experience. He has presented more than 40 National and International papers and published fifteen international journal papers. His research area includes VLSI Signal Processing, Neural Networks, Image Processing and MEMS. He is also heading the Tejaa Shakthi Innovation centre. He expresses his sincere thanks to his chairman Mr.T.N.P.Muthu Natarajan and the secretary Ms.A.Tharalakshmi for their support and encouragement.

**Table.1Architecture of Vedic Multiplier**

| Multiplier | | $a_3$ | | $a_2$ | | $a_1$ | | $a_0$ | |
|---|---|---|---|---|---|---|---|---|---|
| Multiplicand | | $b_3$ | | $b_2$ | | $b_1$ | | bo | |
| $P_8$ | $P_7$ | $P_6$ | $P_5$ | $P_4$ | $P_3$ | $P_2$ | $P_1$ | $P_0$ | PRODUCT |

**Table.3 DCT coefficient**

| Coefficient Name | Coefficient Value | Two's Complement Value |
|---|---|---|
| C1 | 0.4904 | 00111110 |
| C2 | 0.4619 | 00111011 |
| C3 | 0.4157 | 00110101 |
| C4 | 0.3536 | 00101101 |
| C5 | 0.2778 | 00100011 |
| C6 | 0.1913 | 00011000 |
| C7 | 0.0975 | 00001100 |
| -C1 | -0.4904 | 11000010 |
| -C2 | -0.4619 | 11000101 |
| -C3 | -0.4157 | 11001011 |
| -C4 | -0.3536 | 11010011 |
| -C5 | -0.2778 | 11011101 |
| -C6 | -0.1913 | 11101000 |
| -C7 | -0.0975 | 11110100 |

**Comparison**

**Table.2 FPGA utilization**

| configuration | 8 Bit Array Multiplier | 8 Bit Urdhva-Tiryak Vedic Multiplier | Dct using Array multipler | Dct using Vedic multiplier |
|---|---|---|---|---|
| Number of Slices Utilized | 95 | 53 | 2483/23616 | 1109/23626 |
| Number of 4 input LUTs Utilized | 165 | 93 | 4610/47232 | 2133/47232 |
| Number of bonded IOBs Utilized | 32 | 32 | 192/692 | 192/692 |

**Table.3 Timing Report**

| *Delay* | 8 Bit Array Multiplier | 8 Bit Urdhva-Tiryak Vedic Multiplier | Dct using Array mult | Dct using Vedic mult |
|---|---|---|---|---|
| Maximum Combinational Path Delay | 95.21ns | 33.751ns | 105.6 ns | 89.5 ns |