JETIR.ORG ISSN: 2349-5162 | ESTD Year : 2014 | Monthly Issue
JOURNAL OF EMERGING TECHNOLOGIES AND INNOVATIVE RESEARCH (JETIR)
An International Scholarly Open Access, Peer-reviewed, Refereed Journal

Area-Delay Efficient based Discrete Hartley Transform using CBL Adder

Anoop Mishra [1], Prof. Swati Gupta[2]
M. Tech. Scholar, Department of Electronics and Communication Engineering, Vidhyapeeth Institute of Science and Technology, Bhopal, M.P.[1]
Head of Dept., Department of Electronics and Communication Engineering, Vidhyapeeth Institute of Science and Technology, Bhopal, M.P.[2]

Abstract: A discrete Hartley transform (DHT) algorithm can be efficiently implemented on a highly modular and parallel architecture having a regular structure is consisting of multiplier and adder. Discrete Hartley Transform (DHT) is one of the transform used for converting data in time domain into frequency domain using only real values. We have proposed a new algorithm for calculating DHT of length 2^N, where N=3 and 4. We have implemented Urdhwa multiplier based on compressor as an improvement in place of simple multiplication used in conventional DHT. This paper gives a comparison between conventional DHT algorithm and proposed DHT algorithm in terms of delays and area.

Index Terms – **Discrete Hartley Transform (DHT), Urdhwa Multiplier, 4:2 Compressor, 7:2 compressor and Xilinx Vertex family**

I. Introduction

Signal processing involves analyzing, extraction of information and synthesis of the signal. It depends on the type of signal and the nature of information carried by the signal. In general, it is a mathematical and a graphical tool for operations on signal either for data compression, data transmission, elimination of noise, filtering, or smoothing and identification etc. The signal could be sound, image, a time varying sensed data or control signal, to mention a few. The mathematical representation of a signal, if function of time, is known as time domain representation. The same signal can also be expressed as a sum of sine and cosine functions of frequencies and this representation of signal is called the frequency domain representation. The method of converting a time domain signal into the frequency domain is known as transformation. A time domain signal depicts how the signal changes with time, whereas the frequency domain representation gives information regarding the various frequency components present in the signal. Frequency domain representation is commonly used for visualizing the real world signal. Most of the natural signals are analog in nature and can be processed directly by appropriate analog systems. However, with the advances in integrated circuits (IC) technology and computer technology, the digital signal processing (DSP) techniques are preferred on account of a number of advantages such as flexibility, accuracy and speed in designing discrete time systems and the use of computers for analysis.
Digital signal processing techniques find applications in a variety of systems such as in speech processing, data transmission, image processing, instrumentation, bio-medical engineering, seismology, oil exploration, detection of nuclear explosion, besides in the processing of signals received from the outer space, to mention a few [1-2]. Discrete transforms have multifarious diverse applications in digital signal processing and other areas of science, engineering and technology. The representation of signals by these transforms leads to convenient solution of the problems and often they provide greater insight into the physical phenomena. In the digital domain various types of discrete orthogonal transforms, such as discrete-time Fourier transform (DTFT), z transform (ZT), discrete cosine transform (DCT), discrete sine transform (DST), discrete Hartley transform (DHT), Walsh transform (WT), discrete Hadamard transform (DHDT), and slant transform (ST) are mostly used. These transforms have played a key role in signal processing for a number of years, and therefore, transform coding continues to be a topic of interest in theoretical as well as for applied work in this field. One dimensional transforms, in general, have central role in developing applications such as speech coding, least mean squares error (LMSE) filters, digital filter banks (Trans-multiplexers) (TDM /FDM), ceptstral analysis, speech recognition, digital storage media (CD-ROM, DAT , optical Disk , magnetic hard disk, laser disk , magneto-optical disk), Speech encryption etc. In the present research work, we have focused our attention on the discrete orthogonal transforms. These transforms are used for data compression of natural signals like audio signal so as to reduce the bandwidth requirement for transmission. Different coding schemes used for compression are known as Transform coding. Transform coding exploits the fact that for a signal, large amount of energy is concentrated in a small fraction of transform coefficients.

II. URDHWA MULTIPLIER

Multipliers are the blocks in DSP processors which requires large amount of storage and processing complexity. Vedic mathematics has well known an ancient method for mathematical operations which are easy to use Urdhwa triyambakam is a method used for multiplication calculations. It takes the advantage of vertical and crosswise calculations. In this technique multiplication of terms is obtained by using simple AND logical operations, full adders and half adders.

In this paper a 8 bit Urdhwa multiplier is used. For this purpose of addition, we have been used various designs of adders and compressors. Here, these adders and compressors are deployed in such a way so as to provide optimized design of multiplier in terms of delay. Four half adders, two full adders, five 7: 2 compressor and ten 4:2 compressors has been used.

We have presented three designs for Urdhwa multiplier. These three designs vary in terms of 4:2 compressor designs. First design uses the simple 4:2 compressor which uses two full adders. Second design uses four XOR gates and two multiplexers (2×1). Third design of 4:2 compressors is two XOR-XNOR gates and four multiplexers (2×1). First design of 4:2 compressors is simplest one, but more delay. Next design has lesser delay as compared to the first design at the cost of higher complexity. But the last which is our proposed design has least amount of delay but has greatest complexity and occupies maximum space among all of them.

We have designed 7:2 compressor using different designs of 4:2 compressor thus we got three variations of 7:2 compressor.

- 4:2 COMPRESSOR

To add binary numbers with minimal carry propagation we use compressor adder instead of other adder. Compressor is a digital modern circuit which is used for high speed with minimum gates requires designing technique. This compressor becomes the essential tool for fast multiplication adding technique by keeping an eye on fast processor and lesser area.

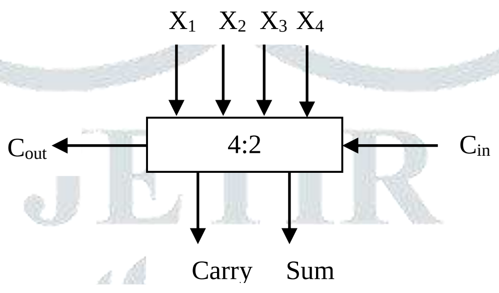


Figure 1: Block Diagram of 4:2 Compressors

4:2 compressors are capable of adding 4 bits and one carry, in turn producing a 3 bit output. The 4-2 compressor has 4 inputs X_1, X_2, X_3 and X_4 and 2 outputs Sum and Carry along with a Carry-in (Cin) and a Carry-out (Cout) as shown in Figure 1. The input Cin is the output from the previous lower significant compressor.

The Cout is the output to the compressor in the next significant stage. The critical path is smaller in comparison with an equivalent circuit to add 5 bits using full adders and half adders. Similar to the 3-2 compressor the 4-2 compressor is governed by the basic equation the standard implementation of the 4-2 compressor is done using 2 Full Adder cells as shown in Figure 2(a). When the individual full Adders are broken into their constituent XOR blocks, it can be observed that the overall delay is equal to 4*XOR.

$$X_1 + X_2 + X_3 + X_4 + C_{in} = sum + 2*(Carry + C_{out})$$

$$Sum = X_1 \oplus X_2 \oplus X_3 \oplus X_4 \oplus C_{in}$$

$$C_{out} = (X_1 \oplus X_2).X_3 + (X_1 \oplus X_2).X_1$$

$$C_{arry} = U.C_{in} + (\overline{X_1 \oplus X_2 \oplus X_3 \oplus X_4}).X_4$$

Where

$$U = X_1 \oplus X_2 \oplus X_3 \oplus X_4$$

The standard implementation of the 4-2 compressor is done using 2 Full Adder cells as shown in Figure 2(a). When the individual full Adders are broken into their constituent XOR blocks, it can be observed that the overall delay is equal to 4*XOR. The block diagram in Figure 2(b) shows the existing architecture for the implementation of the 4-2 compressor with a delay of 3*XOR. The equations governing the outputs in the existing architecture are shown below

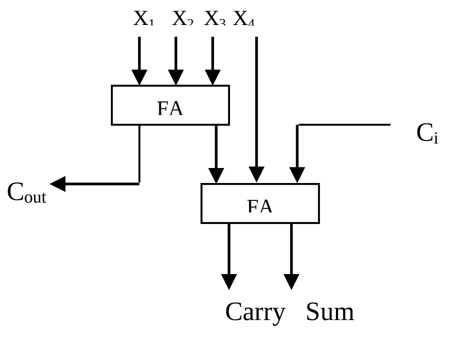


Figure 2(a): 4:2 Compressors using Full Adder

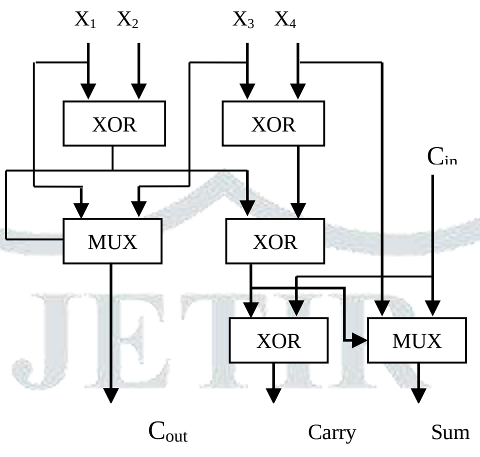


Figure 2(b): 4:2 Compressors using XOR Gate

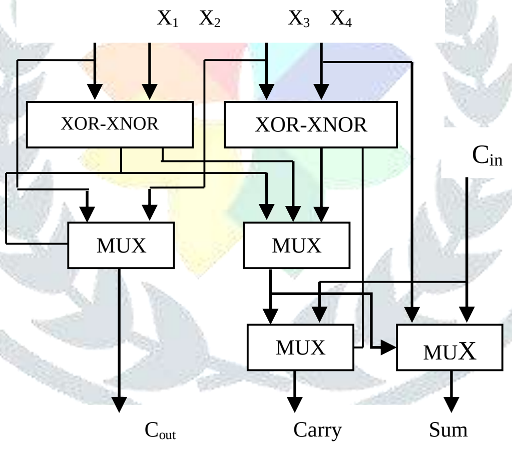


Figure 2(c): 4:2 Compressors using XOR-XNOR Gate

Thus replacing some XOR blocks with multiplexer's results in a significant improvement in delay. Also the MUX block at the SUM output gets the select bit before the inputs arrive and thus the transistors are already switched by the time they arrive. This minimizes the delay to a considerable extent. This is shown in Figure 2(c).
The equations governing the outputs in the proposed architecture are shown below

$$Sum = (X_1 \oplus X_2).X_3 + (X_1 \oplus X_2).(X_3 \oplus X_1).C_{in}$$

$$C_{out} = (X_1 \oplus X_2).X_3 + (X_1 \oplus X_2).X_1$$

All the designs have been captured by VHDL and the functionality is verified by RTL and gate level simulation shown in table 1. Designs are implemented on a Xilinx Spartan 3 FPGA using VHDL as the RTL language with the help of Xilinx ISE design suite 14.1.

TABLE 1: Device utilization summary (Vertex-4) of 4:2 Compressors using full adder, 4:2 Compressor using XOR Gate and 4:2 Compressor using XOR-XNOR Gate

Design	No. of Slices	No. of 4 input LUTs	MCPD (ns)
4:2 Compressor	3	6	6.257
4:2 Compressor using XOR gate	2	4	5.663
4:2 Compressor using XOR-XNOR gate	2	3	5.280

- 7:2 COMPRESSOR

Similar to its 4:2 compressor counterpart, the 7:2 compressors as shown in Figure 3, is capable of adding 7 bits of input and 2 carry's from the previous stages, at a time. In our implementation, we have designed a novel 7:2 compressor utilizing two 4:2 compressors, two full adders and one half adders. We have designed 7:2 compressor using different designs of 4:2 compressor thus we got three variations of 7:2 compressor.

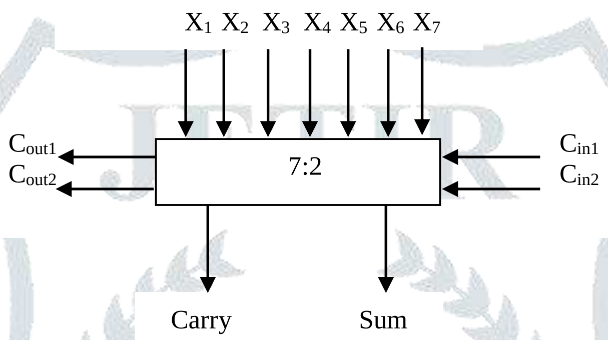


Figure 3: Block Diagram of 7:2 Compressors

III. DISCRETE HARTLEY TRANSFORM

Discrete Hartley Transform is abbreviated for DHT and this transform was proposed by R. V. L. Hartley in 1942. DHT is the analogous to Fast Fourier transform which provides the only real value at any cost. The main difference from the DFT is that it transforms the real inputs to real outputs with no intrinsic involvement of complex value. DFT can be used to compute the DHT, and vice versa.

Real Inputs
DHT
Real Outputs

Figure 4: Block Diagram of DHT

In other words, Discrete Hartley transform is used to convert real values into real ones. It requires decomposition of data into stages using butterfly similar to FFT. But the butterfly used in DHT is quite different in terms of coefficients or multipliers. With the increase in number of DHT sequence length the number of coefficients is also increased simultaneously.
Let $N \geq 4$ be a power of two. For any real input sequence $\{x(i) : i = 0,1,2,..................N-1\}$

$$X(k) = DHT(N)\{x((i)\}$$

$$= \sum_{i=0}^{N-1} x(i).cas[2ki\pi / N] \qquad for\, k = 0,1..........N-1$$

Where $cas(x) = \cos(x) + \sin(x)$

- ALGORITHM FOR 16 POINT DHT-

We present an implementation of fast DHT algorithm for a length N=1. There are six stages required to complete the butterfly design of N=16 length DHT. These stages include summing stages and coefficient multiplying stages. Before first stage the data sequence are arranged in bit reversed pattern by using any method like permutation. Then in the first stage the pairs of bit reversed patterns are added to form eight terms. In the second stage, one third of the terms are again added and subtracted to form further three terms.
First two stages do not include any multiplication. Remaining terms are multiplied by the first coefficient. In the next stage again two new coefficients are introduced which is multiplied by the lower half of the third stage. In each stage multiplying of coefficients stage precedes its summing stage. After coefficient multiplication it is preceded by its summing stage to form the

common terms used in the final stage. Last stage includes only summing of terms. Finally we get the transformed data sequence in order and do not need any permutation.

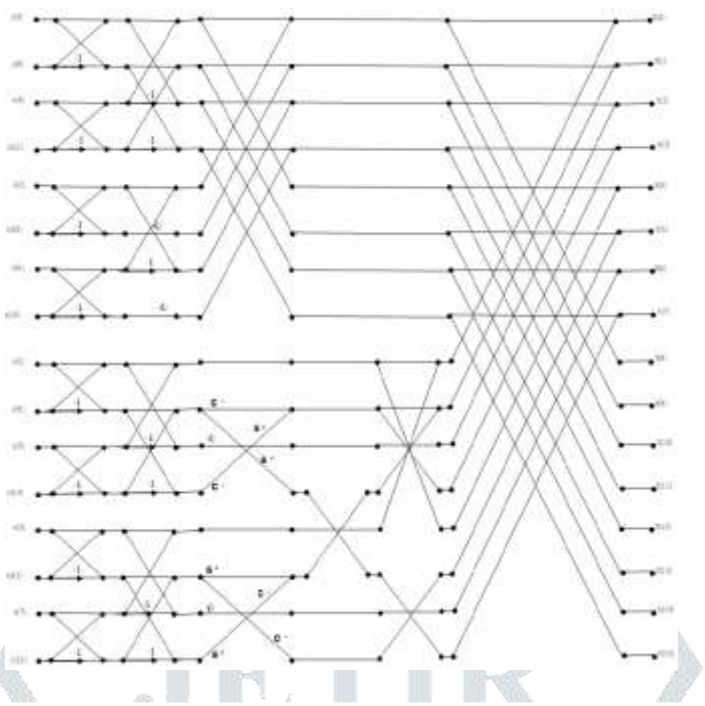


Figure 5: 16-point DHT Butterfly

- Mathematical calculation

$$X(0) = (x(0)+x(8)) + (x(4)+x(12)) + (x(2)+x(10))$$
$$+(x(6)+x(14)) + (x(1)+x(9)) + (x(5)+x(13)) +$$
$$(x(3)+x(11)) + (x(7)+x(15))$$
$$X(1) = (x(0)+x(8)) + (x(4)-x(12)) + c_1(x(2)-x(10))$$
$$+(x(5)+x(13)) + c_3\{(x(3)-x(11)) + (x(7)-x(15))\} +$$
$$c_2\{(x(3)-x(11)) - (x(7)-x(15))\}$$
$$X(2) = (x(0)+x(8)) - \{(x(4)+x(12)) + (x(6)+x(14)\}$$
$$+c_1\{(x(1)+x(9)) - (x(5)+x(13))\} + (x(2)+x(10))$$
$$(x(3)+x(11)) + (x(7)+x(15))$$
$$X(3) = (x(0)-x(8)) - (x(4)-x(12)) + c_1(x(6)-x(14))$$
$$+(x(5)+x(13)) + c_3\{(x(3)+x(11)) + (x(7)-x(15))\} +$$
$$c_2\{(x(3)-x(11)) - (x(7)-x(15))\}$$
$$X(4) = (x(0)+x(8)) + (x(4)+x(12)) + (x(2)+x(10))$$
$$+(x(6)+x(14)) + (x(1)+x(9)) + (x(5)+x(13)) -$$
$$\{(x(3)+x(11)) + (x(7)+x(15))\}$$
$$X(5) = (x(0)+x(8)) + (x(4)-x(12)) + c_1(x(2)-x(10))$$
$$+(x(5)+x(13)) + c_3\{(x(3)-x(11)) + (x(7)-x(15))\} -$$
$$c_2\{(x(5)-x(13)) - (x(1)-x(9))\}$$
$$X(6) = (x(0)+x(8)) + (x(6)+x(14)) - \{(x(2)+x(10))$$
$$+(x(4)+x(12))\} + c_1\{(x(3)-x(11)) + (x(7)-x(15))\}$$
$$X(7) = (x(0)+x(8)) - (x(4)-x(12)) + c_1(x(6)-$$
$$x(14)) - c_3\{(x(5)+x(13)) - (x(1)-x(9))\} + c_2\{(x(5)$$
$$-x(13)) + (x(1)-x(9))\}$$

$$X(8) = (x(0)+x(8)) + (x(4)+x(12)) + (x(2)+x(10)) + (x(6)+x(14)) - (x(1)+x(9)) - (x(5)+x(13)) - (x(3)+x(11)) - (x(7)+x(15))$$

$$X(9) = (x(0)-x(8)) + (x(4)-x(12)) + c_1(x(2)-x(10)) + (x(5)+x(13)) - c_3\{(x(3)-x(11)) + (x(7)-x(15))\} + c_2\{(x(3)-x(11)) - (x(7)-x(15))\}$$

$$X(10) = (x(0)+x(8)) - \{(x(4)+x(12)) + (x(6)+x(14)\} - c_1\{(x(1)+x(9)) - (x(5)+x(13))\}$$

$$X(11) = (x(0)-x(8)) - (x(4)-x(12)) + c_1(x(6)-x(14)) + c_3\{(x(3)-x(11)) + (x(7)-x(15))\} + c_2\{(x(3)-x(11)) - (x(7)-x(15))\}$$

$$X(12) = (x(0)+x(8)) + (x(4)+x(12)) + (x(2)+x(10)) + (x(6)+x(14)) - \{(x(1)+x(9)) + (x(5)+x(13))\} + (x(3)+x(11)) + (x(7)+x(15))$$

$$X(13) = (x(0)-x(8)) + (x(4)-x(12)) - c_1(x(2)-x(10)) - c_3\{(x(3)-x(11)) + (x(7)-x(15))\} + c_2\{(x(1)-x(9)) - (x(5)-x(13))\}$$

$$X(14) = (x(0)+x(8)) - (x(4)+x(12)) - (x(2)+x(10)) + c_1(x(3)+x(11)) - (x(7)+x(15))$$

$$X(15) = (x(0)-x(8)) - (x(4)-x(12)) - c_1(x(6)-x(14)) - c_3\{(x(13)-x(5)) - (x(1)-x(9))\} + c_2\{(x(1)-x(9)) - (x(5)-x(13))\}$$

IV. PROPOSED METHODOLOGY

Here, the DHT of length N=16 point requires 67 additions and 12 multiplications. We have used a different technique called Urdhwa tiryakbhyam of ancient Vedic times for multiplication. This multiplication technique reduces the delay and complexity. It converts multiplication into simple logical AND operation using associated circuits of full, half adders and AND gate. Further, the area is reduced by using various compressors for adding the partial products of multiplication.

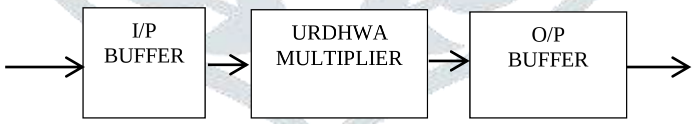


Figure 6: Proposed Architecture

V. SIMULATION RESULTS

In this paper, we have designed three architectures of DHT transform. First architecture is designed using simple Urdhwa multiplier. In simple Urdhwa multiplier, we have used 4:2 compressor designed by using full adder and half adder i.e. the basic design of 4:2 compressor. Second architecture is designed using modified Urdhwa multiplier. In modified Urdhwa multiplier, we have used 4:2 compressor designed by using XOR gates and multiplexer.

Third architecture is designed using modified Urdhwa multiplier. In modified Urdhwa multiplier, we have used 4:2 compressor designed by using XNOR gates and multiplexer.

All three architecture are designed by using Xilinx 14.2i.The simulated results are displayed in table 2.In the simulation results proposed DHT gives the best performance.

TABLE II: Comparisons Result for 8-point Discrete Hartley Transform (DHT) with word length 16.

Parameter	**Previous Technique**	**Modified DHT**	**Proposed DHT**
No. of Slices	601	553	548
No. of 4 input LUTs	1125	1032	1029
No. of bounded IOBs	256	256	256
Maximum combinational path delay(in ns)	29.62	24.97	21.96

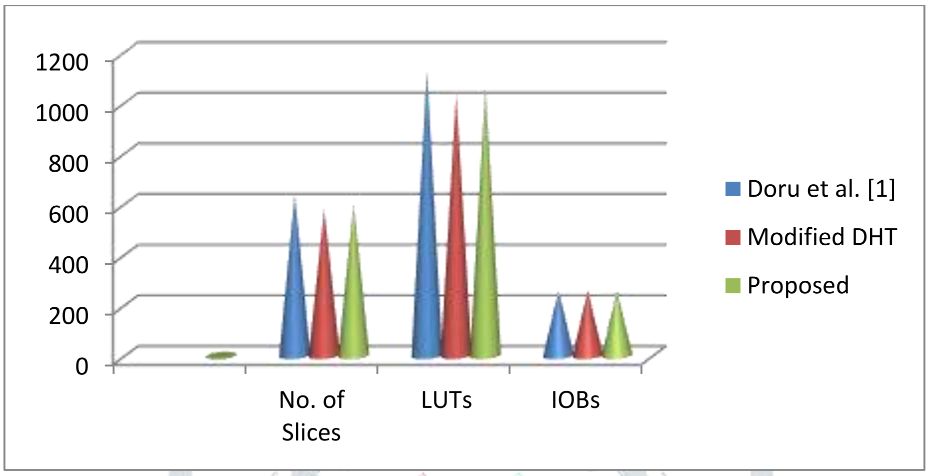
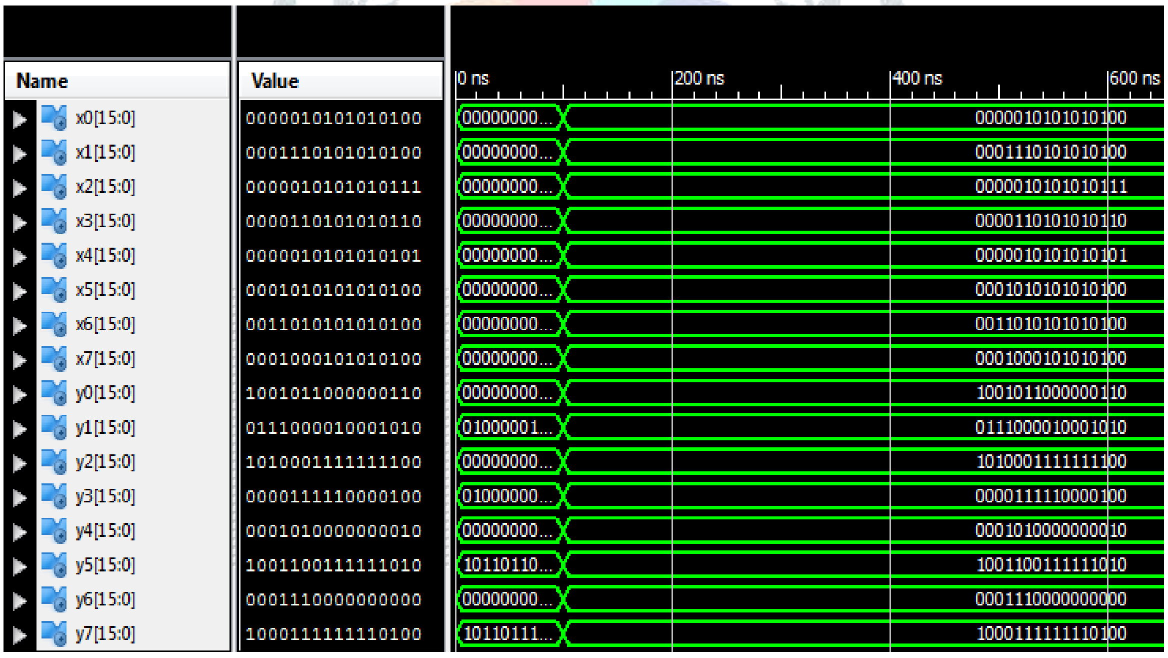


Figure 7: Show the bar graph of 8-bit DHT with word length 16



```
Name        Value               0 ns            200 ns    400 ns    600 ns
x0[15:0]    0000010101010100    00000000...     0000010101010100
x1[15:0]    0001110101010100    00000000...     0001110101010100
x2[15:0]    0000010101010111    00000000...     0000010101010111
x3[15:0]    0000110101010110    00000000...     0000110101010110
x4[15:0]    0000010101010101    00000000...     0000010101010101
x5[15:0]    0001010101010100    00000000...     0001010101010100
x6[15:0]    0011010101010100    00000000...     0011010101010100
x7[15:0]    0001000101010100    00000000...     0001000101010100
y0[15:0]    1001011000000110    00000000...     1001011000000110
y1[15:0]    0111000010001010    01000001...     0111000010001010
y2[15:0]    1010001111111100    00000000...     1010001111111100
y3[15:0]    0000111110000100    01000000...     0000111110000100
y4[15:0]    0001010000000010    00000000...     0001010000000010
y5[15:0]    1001100111111010    10110110...     1001100111111010
y6[15:0]    0001110000000000    00000000...     0001110000000000
y7[15:0]    1000111111110100    10110111...     1000111111110100
```

Figure 8: Output Waveform of 8-bit DHT with word length 16

VI. CONCLUSION

DHT is a new transform used for real value to real value conversion. Urdhwa Triyambakam is an ancient technique for multiplication. DHT is used in various fields such as image processing, space science, scientific applications etc. Delay provided and area required by hardware are the two key factors which are need to be consider. Here we present DHT with 8×8 Urdhwa multiplier by using full adder, OR and XNOR gates in different types of compressors.

Among all three compressors, compressor provides the least amount of delay. Also, it takes least number of slices i.e. Occupy least area among all three compressors.

REFERENCES

[1] Nikhil Advaith Gudala and Trond Ytterdal, "Implementation of High Speed and Low Power Carry Select Adder with BEC", International Midwest Symposium on Circuits and Systems IEEE 2021.

[2] N. Hamed S. Timarchi "Low-Power and Fast Full Adder by Exploring New XOR and XNOR Gates" IEEE Transactions on Very Large Scale Integration (VLSI) Systems 2018.

[3] Low Power Approximate Adders for General Computing Using Differential Transmission Gate" on 28.03.2018 National Conference on Innovations in Communication and Computing NCICC‟ 18 SNS College of Technology.

[4] Omid Akbari, Mehdi Kamal, Ali Afzali-Kusha, and Massoud Pedram, "Dual-Quality 4:2 Compressors for Utilizing in Dynamic Accuracy Configurable Multipliers", IEEE Transactions on Very Large Scale Integration (VLSI) Systems, Vol. 34, Issue 7, 2017.

[5] Doru Florin Chiper, *Senior Member, IEEE, "*A Novel VLSI DHT Algorithm for a Highly Modular and Parallel Architecture", IEEE TRANSACTIONS ON CIRCUITS AND SYSTEMS—II: EXPRESS BRIEFS, VOL. 60, NO. 5, MAY 2013.

[6] Sushma R. Huddar and Sudhir Rao, Kalpana M., "Novel High Speed Vedic Mathematics Multiplier using Compressors ", 978-1 - 4673-5090-7/13/$31.00 ©2013 IEEE.

[7] S. S. Kerur, Prakash Narchi, Jayashree C N, Harish M Kittur and Girish V A, "Implementation of Vedic multiplier for Digital Signal Processing", International Conference on VLSI, Communication & Instrumentation (ICVCI) 2011, Proceedings published by International Joural of Computer Applications® (IJCA), pp.1 -6.

[8] Himanshu Thapaliyal 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, India.

[9] Jagadguru Swami Sri Bharati Krishna Tirthaji Maharaja, "Vedic Mathematics: Sixteen simple Mathematical Formulae from the Veda", Delhi (2011).

[10] Sumit Vaidya and Depak Dandekar. "Delay-power perfor-mance comparison of multipliers in VLSI circuit design". International Journal of Computer Networks & Communications (IJCNC), Vol.2, No.4, July 2010.

[11] P. D. Chidgupkar and M. T. Karad, "The Implementation of Vedic Algorithms in Digital Signal Procesing", Global J. of Eng. Edu, Vol.8, No.2, 204, UICEE Published in Australia.

[12] Asmita Haveliya, "Design and Simulation of 32-Point FFT Using Radix-2 Algorithm for FPGA Implementation", Second International Conference on Advanced Computing & Communication Technologies IEEE 2012.

[13] S. Correa, L. C. Freitas, A. Klautau and J. C. W. A. Costa, "VHDL Implementation of a Flexible and Synthesizable FFT Processor", IEEE Latin America Transactions, Vol. 10, No. 1, Jan. 2012.