Research Article

Resourceful Fast Discrete Hartley Transform to Replace Discrete Fourier Transform with Implementation of DHT Algorithm for VLSI Architecture

Deepti Gautam [1], Anshuman Singh[2]

[1]Noida Institute of Engineering & Technology, Greater Noida, India
[2]Noida Institute of Engineering & Technology, Greater Noida, India

Article History: Received: 10 January 2021; Revised: 12 February 2021; Accepted: 27 March 2021; Published online: 28 April 2021

Abstract: Discrete Fourier Transform (DFT) is used in a range of digital signal processing applications, such as signal and image compression systems, filter banks, signal representation or harmonic analysis. The Discrete Hartley Transform (DHT) can be used to effectively replace the DFT when the input sequence is real. This paper introduces and applies the Discrete Hartley Transform (DHT) algorithm, which is well suited to the VLSI architecture. The purpose of using the DHT algorithm is to reduce the complexity of the VLSI architecture. With the introduction of VLSI architectures, 100,000 transistors can be mounted on a single chip. For manufacturers, the verification of these circuits on breadboards is not feasible. Computer-aided design methods have now come into being, but programmers also had to make connections manually at the gate level. Subsequently, Hardware Description Language (HDL) such as Verilog and VHDL came into being. With the integration of the DHT algorithm and its execution, Verilog authors have worked to reduce the complexity of the VLSI circuits in this article. With reduced complexity, the power consumption is reduced and thus the area is therefore reduced. DHT is an algorithm for the Radix-2. It is especially useful for image and signal processing. The DHT algorithm for length N=8 is introduced here.

Keywords: Discrete Hartley Transform (DHT), Discrete Fourier Transform (DFT), DHT Domain Processing.

1. Introduction

Discrete Hartley Transform is a Radix-2 algorithm and is especially useful in DSP applications such as image compression, signal compression, filter banks[1] signal representation and harmonic analysis, etc.[2]. The key goal behind the use of DHT is to reduce the complexity of VLSI hardware architecture [3]. While DHT computing has become an intensive hardware complexity, it is reduced. The explanation behind this is that the forward and inverse of DHT is absolutely the same except for the scaling factor. Or DHT is the opposite of its own. The same hardware will also be used for both forward and inverse measurements. In other terms, we can assume that the same number of adders, multipliers, etc. are used by both forward and inverse implementations.

DHT is somewhat similar to the Discrete Fourier Transform (DFT) or DHT is derived from DFT. The DFT includes both actual and imaginary parts, while the DHT only has real parts. The calculation of imaginary parts is negligible and therefore complex arithmetic is avoided. In other words, DHT is used as a replacement of DFT when the sequence is real [4] [5].

In other terms, we can assume that the same number of adders, multipliers, etc. are used by both forward and inverse implementations.

DHT is somewhat similar to the Discrete Fourier Transform (DFT) or DHT is derived from DFT. The DFT includes both actual and imaginary parts, while the DHT only has real parts.

There are also many spilt radix algorithms available for computing. These can be implemented with even low arithmetic cost [10]-[11]. The disadvantage with the classical spilt-radix algorithm is its irregular computational structure.

Delays in VLSI architecture are mainly introduced by multipliers, as they consume a large portion of the chip area. Hence, to implement multipliers, memory-based solutions are being taken into consideration. Implementation of multipliers with look-up table-based solutions, it is required that one operand should be constant [12]. It then becomes possible to store complete result in partial ROM. And then the memory words are reduced from 2^{2L} to 2^{L}
[1].

With DHT algorithm, VLSI architecture can be implemented based on high parallelism. Moreover, by sharing the multipliers by same constant and using sub-expression sharing technique, hardware complexity gets reduced [13][14]

2. 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.

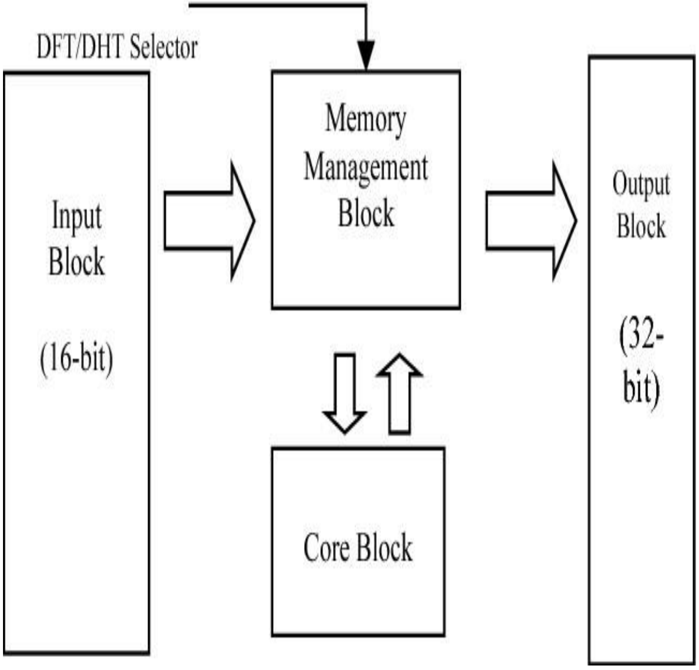


Figure 1: DHT Architecture.

□ **Forward and Inverse DHT**

Let N ≥ 4 be a power of 2. Since, it's a Radix-2 algorithm. For any given real input sequence such as ***{ y(i) } i = 0,1,2…..N -- 1 }***

DHT (N) is defined by

$$\boldsymbol{Y(k) = DHT(N)\{x(i)\}\sum_{i=0}^{N-1} y(i).cos\left[\frac{2ki\pi}{N}\right]\ldots\ldots\ldots\ldots(1)}$$

For k = 0, 1, 2, 3, 4……..N-1

Inverse relation is defined as: $\boldsymbol{Y(n)\sum_{k=0}^{N-1} X(k)\left(cos\left[\frac{2ki\pi}{N}\right] + sin\left[\frac{2ki\pi}{N}\right]\right)}$ ………………..*(2)*

For, n=0,1…………N-1

Also, cas (y) = cos (y) + sin(y)

□ **DHT algorithm for length L=8**

Here, algorithm is depicted for length 8.

$$\mathbf{Y(0)} = [(\mathbf{y(0)} + \mathbf{y(4)}) + \mathbf{y(2)} + \mathbf{y(6)})] + [(\mathbf{y(1)} + \mathbf{y(5)}) + \mathbf{y(3)} + \mathbf{y(7)})]$$

$$\mathbf{Y(2)} = [(\mathbf{y(0)} + \mathbf{y(4)}) - \mathbf{y(2)} + \mathbf{y(6)})] + [(\mathbf{y(1)} + \mathbf{y(5)}) - \mathbf{y(3)} + \mathbf{y(7)})]\ \mathbf{Y(4)} = [(\mathbf{y(0)} + \mathbf{y(4)}) + \mathbf{y(2)} + \mathbf{y(6)})] - [(\mathbf{y(1)} + \mathbf{y(5)}) + \mathbf{y(3)} + \mathbf{y(7)})]$$

$$\mathbf{Y(6)} = [(\mathbf{y(0)} + \mathbf{y(4)}) - \mathbf{y(2)} + \mathbf{y(6)})] - [(\mathbf{y(1)} + \mathbf{y(5)}) - (\mathbf{y(3)} + \mathbf{y(7)})]$$

$$\mathbf{Y(1)} = [\mathbf{y(0)} - \mathbf{y(4)}] + [\mathbf{y(2)} - \mathbf{y(6)}] + \mathbf{c}[\mathbf{y(1)} - \mathbf{y(5)}]$$

$$\mathbf{Y(3)} = [\mathbf{y(0)} - \mathbf{y(4)}] - [\mathbf{y(2)} - \mathbf{y(6)}] + \mathbf{c}[\mathbf{y(3)} - \mathbf{y(7)}]$$

$$\mathbf{Y(5)} = [\mathbf{y(0)} - \mathbf{y(4)}] + [\mathbf{y(2)} - \mathbf{y(6)}] - \mathbf{c}[\mathbf{y(1)} - \mathbf{y(5)}]$$

$$\mathbf{Y(7)} = [\mathbf{y(0)} - \mathbf{y(4)}] - [\mathbf{y(2)} - \mathbf{y(6)}] - \mathbf{c}[\mathbf{y(3)} - \mathbf{y(7)}]$$

With, c =√2

It can further reduce the number of multipliers since we are multiplying with the same constant 'c'. For the same purpose we can use same multiplier.

Therefore, Radix−8 FHT algorithm divides an N–point DHT given by (1) into four $N/8$–point DHTs defined by equations below and one $N/2$–point DHT defined by (4). The next step in deriving the algorithm, is by applying further the Radix−4 and Radix−2 FHT algorithms to the odd- and even-indexed samples of (4) respectively. Therefore, we get the following decompositions

i. For even-indexed samples-

$\boldsymbol{N}$
$\_\boldsymbol{-1}$
$\boldsymbol{4} \qquad \boldsymbol{N} \qquad \boldsymbol{N} \qquad \boldsymbol{3N}$
$Y\ (4k)\sum \quad \boldsymbol{x(n)} + \boldsymbol{x\ (n} + \_) + \boldsymbol{x\ (n} + \_) + \boldsymbol{x(n} + \_\_)\ (\boldsymbol{cos}[\boldsymbol{2ki\pi}]$ ……………….***(3)***
$\boldsymbol{n}=\boldsymbol{0} \qquad \boldsymbol{4} \qquad \boldsymbol{2} \qquad \boldsymbol{4}$

ii. For odd-indexed samples

$\boldsymbol{N}$
$\overline{8}^{-1}$
$Y \quad (8k+2) \quad \sum \ ^{8}\ \boldsymbol{x(n)} + \boldsymbol{x}\,(\frac{N}{8} - \boldsymbol{n}) \qquad - \boldsymbol{x(n} + \_{}^{N}) - \boldsymbol{x}({}^{3N} - \boldsymbol{n}) + \boldsymbol{x(n} +\!\_{}^{N}) + \boldsymbol{x}({}^{3N} - \boldsymbol{n}) - \boldsymbol{x\ (n} + {}^{3N}) -$
$\boldsymbol{n}=\boldsymbol{0} \quad \boldsymbol{4} \qquad \boldsymbol{8} \quad \boldsymbol{2} \qquad \boldsymbol{8}$
$\boldsymbol{4} \qquad -$
$\underline{\boldsymbol{x}}({}^{7N} - \boldsymbol{n})(\boldsymbol{cos}[\boldsymbol{2ki\pi}])\ \boldsymbol{x(N-n)} + \boldsymbol{x\ (n} + {}^{N}) - \boldsymbol{x}\ ({}^{N} - \boldsymbol{n}) - \boldsymbol{x(n} + {}^{3N}) + \boldsymbol{x}({}^{N} - \boldsymbol{n}) + \boldsymbol{x\ (n} + \_\_{}^{5N}) -$
$\boldsymbol{8} \qquad \boldsymbol{8} \quad \boldsymbol{4} \qquad \boldsymbol{8} \quad \boldsymbol{4} \qquad \boldsymbol{8}$
$-\boldsymbol{x}({}^{3N} - \boldsymbol{n}) - \boldsymbol{x(n} + {}^{7N})(\boldsymbol{sin}[\boldsymbol{2ki\pi}]) \qquad \boldsymbol{cos}[\boldsymbol{8ki\pi}])$ ……………….***(4)***
$\boldsymbol{4} \qquad \boldsymbol{8}$

$\boldsymbol{N}$
$\boldsymbol{-1}$
$Y \quad (8k+6) \quad \sum \ ^{\overline{8}}\ \boldsymbol{x(n)} + \boldsymbol{x}\,(\frac{N}{8} - \boldsymbol{n}) \qquad - \boldsymbol{x(n} + \_{}^{N}) - \boldsymbol{x}({}^{3N} - \boldsymbol{n}) + \boldsymbol{x(n} +\!\_{}^{N}) + \boldsymbol{x}({}^{3N} - \boldsymbol{n}) - \boldsymbol{x\ (n} + {}^{3N}) -$
$\boldsymbol{n}=\boldsymbol{0} \quad \boldsymbol{4} \qquad \boldsymbol{8} \quad \boldsymbol{2} \qquad \boldsymbol{8}$
$\boldsymbol{4} \qquad -$
$\underline{\boldsymbol{x}}({}^{7N} - \boldsymbol{n})(\boldsymbol{cos}[\boldsymbol{6ki\pi}])\ \boldsymbol{x(N-n)} + \boldsymbol{x\ (n} + {}^{N}) - \boldsymbol{x}\ ({}^{N} - \boldsymbol{n}) - \boldsymbol{x(n} + {}^{3N}) + \boldsymbol{x}({}^{N} - \boldsymbol{n}) + \boldsymbol{x\ (n} + \_\_{}^{5N}) -$
$\boldsymbol{8} \qquad \boldsymbol{8} \quad \boldsymbol{4} \qquad \boldsymbol{8} \quad \boldsymbol{4} \qquad \boldsymbol{8}$
$-\boldsymbol{x}({}^{3N} - \boldsymbol{n}) - \boldsymbol{x(n} + {}^{7N})(\boldsymbol{sin}[\boldsymbol{6ki\pi}]) \qquad \boldsymbol{cos}[\boldsymbol{8ki\pi}])$ ……………….***(5)***
$\boldsymbol{4} \qquad \boldsymbol{8}$

Finally, the preferred algorithm (radix−2/4/8 FHT) can be obtained by applying radix−2 FHT algorithm into (3), resulting in the remaining decompositions:

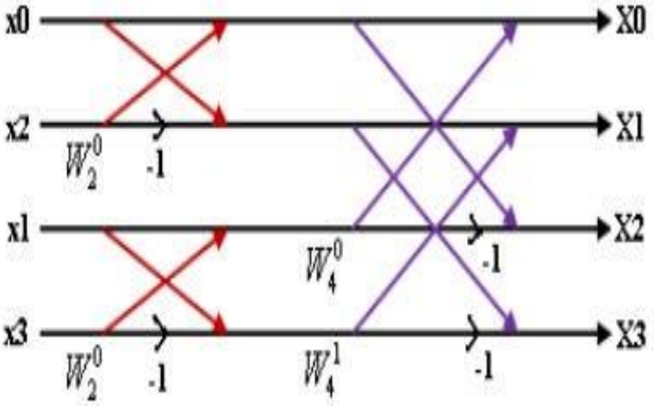


Figure 2: Radix 2 FFT

The Figure 2 shows that 4 point Radix 2 FFT butterfly diagram to compute the DFT points. The radix 2 algorithms are the simplest FFT algorithms. The decimation-in-time (DIT) radix 2 FFT recursively partitions a DFT into two half-length DFTs of the even-indexed and odd-indexed time samples. The outputs of these shorter FFTs are reused to compute many outputs, thus greatly reducing the total computational cost.

$\boldsymbol{N}$
$\_\boldsymbol{-1} \qquad {\scriptstyle N} \qquad {\scriptstyle N} \qquad {\scriptstyle 3N} \qquad {\scriptstyle N} \qquad {\scriptstyle 5N} \qquad {\scriptstyle 3N} \qquad {\scriptstyle 7N}$
$Y \quad (8k)\sum_{n=0}{}^{8}\ \boldsymbol{x(n)} + \boldsymbol{x\ (n} + \_{}_{8}) + \boldsymbol{x\ (n} + \_{}_{4}) + \boldsymbol{x\ (n} + \_\_{}_{8}\,) + \boldsymbol{x\ (n} + {}_{2}) + \boldsymbol{x\ (n} + {}_{8}\,) - \boldsymbol{x\ (n} + {}_{4}\,) + \boldsymbol{x\ (n} + {}_{8}\,)$
$\boldsymbol{cos}[\boldsymbol{8ki\pi}])$ ……………….***(6)***
$\boldsymbol{N}$
$\sum^{\overline{8}} \quad \boldsymbol{x(n)} - \qquad \boldsymbol{-1} \qquad \boldsymbol{N} \qquad \boldsymbol{N} \qquad \boldsymbol{3N} \qquad \boldsymbol{N} \qquad \boldsymbol{5N}$
$\boldsymbol{3N} \qquad \boldsymbol{7N}$

$\boldsymbol{Y\ (8k{+}4)\ _{n=0}\ x\ (n + \_8) + x(n + \_4) - (n + \_\_8) + x\ (n + {}_2) - x\ (n + {}_8) - x(n + {}_4) - x(n + {}_8)}$
$\boldsymbol{cos[8ki\pi])}$ ***……………….(7)***

Performing the computation in-place using double butterfly with 16-points provides an in-place butterfly of the proposed radix−2/4/8 algorithm.

Therefore, the odd harmonic components are calculated from u_o after some adjustment terms. Evaluating the multiplicative complexity. The above proceed can iterative be applied since that N is a power of two. We arrive then to the block diagram shown in Figure 2 below.

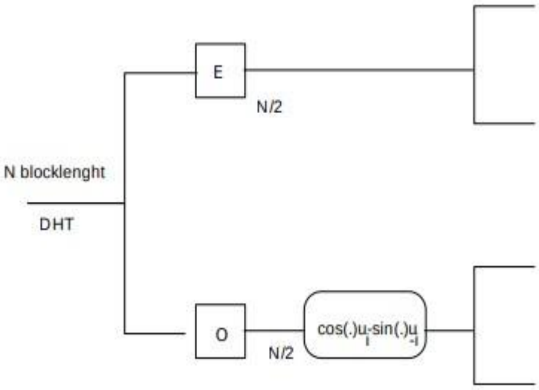


Figure 3: A step of the Fast Discrete Hartley Transform based on evenodd decomposition.

Since that each step reduces the block length to a half, there is log2N steps. Those of the adjustments, i.e., 2N multiplications give the number of multiplications in each step. Therefore, we need 2Nlog2N multiplications the same as the classical FFT.

3. Hardware flow diagram

The equations (3) – (7) can be implemented in hardware by considering the flow diagram depicted in Figure 1. The block simply consists of few ADDERs' and two MULTIPLIER blocks for multiplication with 'c'.

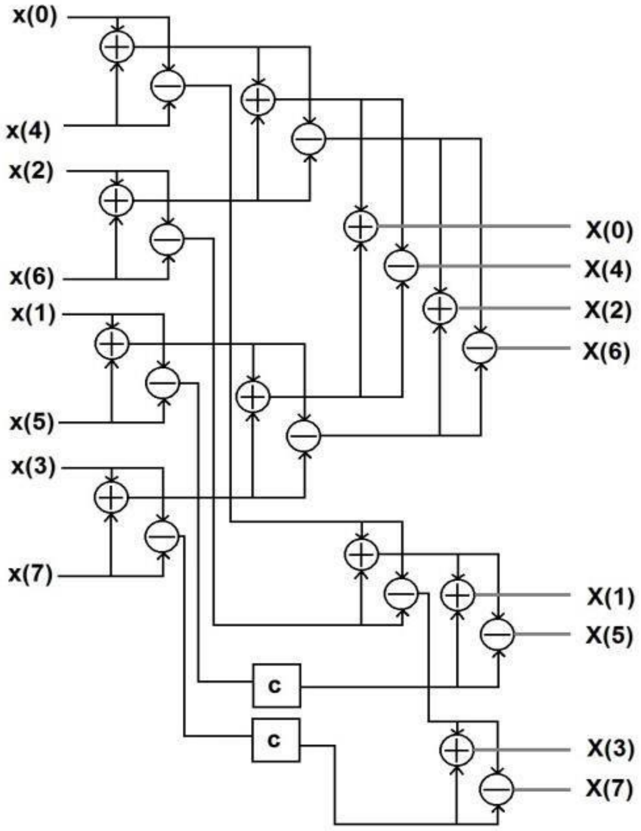


Figure 4: Hardware Flow Diagram.

In the above figure 4 the hardware flow diagram of the DHT is shown with the 2N multiplications.

1. RTL Flow Diagram

The Figure 2 shows the synthesis of the block diagram in the form of an RTL, synthesized using XILINX ISE 14.7. All the blocks are simple adders and sub tractors.

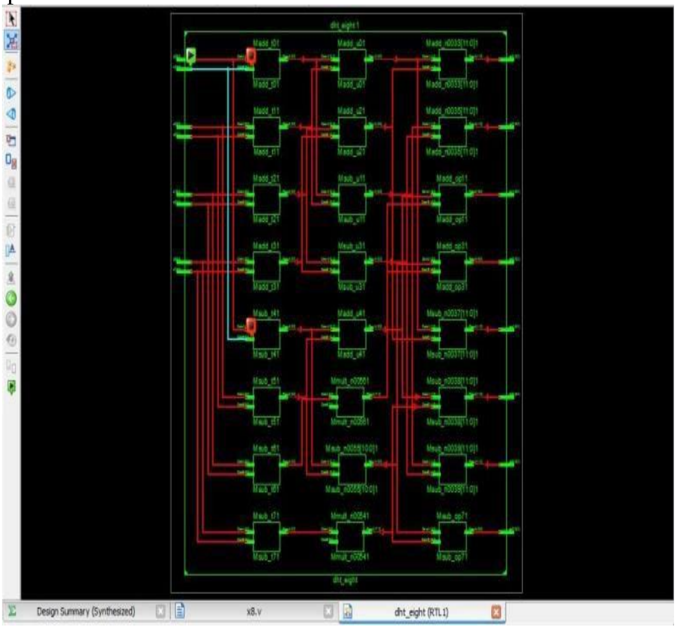


Figure 5: RTL Flow Diagram.

The RTL shown in Figure 5 can also be represented as:

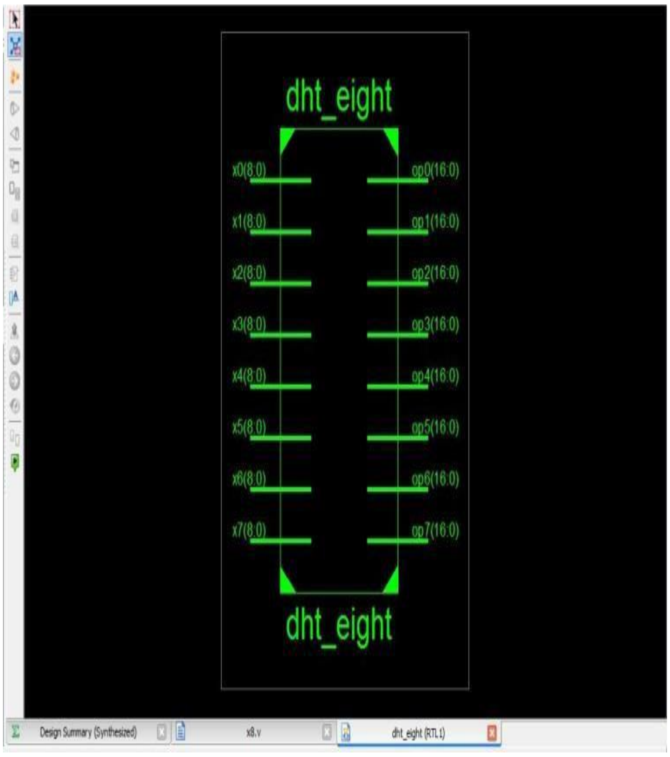


Figure 6: RTL Block Diagram

4. Literature review

Shirali Parsaietal.[1], Discrete Hartley Transform (DHT) is one of the transformations used to convert data from a time - frequency domain to a fourier transform just by using true values. DHT can be used for highly modular as well as parallel data analysis in VLSI applications. We have proposed a new algorithm for 2N duration DHT computation, where N=3 and 4. We also added a multiplier to update to a simple multiplication used in conventional DHT. This paper provides a comparison between the standard DHT algorithm as well as the proposed DHT algorithm in terms of time and area.

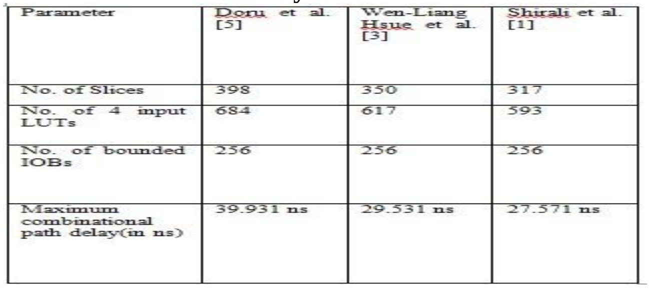
Doru Florin Chiper [2], Efficient algorithms for forward as well as inverse relationship discreet Hartley transforms have been developed over the last 30 years (DHTs). They are similar to the Fast Fourier Transform (FFT) algorithms used to measure the discreet Fourier Transform (DFT). Any of these methods are designed to minimise the complexity of calculations and/or the number of operations. A new approach to estimating the fast Hartley Transform (FHT) radix-2 has been implemented. The proposed algorithm, based on a two-band decomposition of input data, has a very normal form, prevents input or output data from shuffling, requires considerably less multiplication than the existing approaches, but increases the number of additions.

Michela Svaluto Moreolo [3], Real transformations require less computational complexity and much less physical memory than complex transformations. But the discreet fractional transformations of Fourier as well as Hartley are complex transformations. In this paper, we suggest reality-preserving the fractional variants of the discreet transformations of Fourier, Hartley, Generalized Fourier and Generalized Hartley. All the suggested discrete fractional transformations have as many parameters as possible and are therefore very robust. The proposed real discrete fractional transformations have arbitrary proprietary converters and have only two distinct proprietary values 1 and 1. The properties as well as relations of potential real discrete fractional transformations are analysed. In addition, we propose their alternative reality-preserving fractionalizations based on diagonal-like matrices to further boost their flexibility for true, conventionally discreet Hartley and generalised discreet Hartley transformations. Proposed real transition shave all the positive properties needed to be discreet fractional transformations. Finally, because the proposed new transformations have random outputs and several parameters, they are all suitable for data protection applications such as image encryption and watermarking.

Chin-Kuo Jao, Syu-Siang Long [4], In this brief, we assume the quantum-dot cellular automation (QCA) of the discreet Hadamard transform (DHT). The study of a full-parallel solution based on an efficient multibit extension to QCA is discussed for the first time. We're seeing that this contributes to a wide region as well as a delay. We then suggest a bit-serial pipelined architecture for QCA-based DHT. The proposed architecture is based on a new one-bit adder-subtractor needing only six majority gates and a feedback lock requiring only one majority gate and minimal wiring. The method results in a reduction in the area-delay-cycle product of 74 per cent and 91 per cent (over a fullparallel solution) for 4 and 8 word-lengths, respectively. The findings of the QCADesigner simulations are also discussed.

Saad Bouguezel, M. Omair Ahmad [5], A modern, very large-scale integration (VLSI) algorithm for the 2Nlength discrete transformation of Hartley (DHT) that can be efficiently implemented on a highly modular and parallel VLSI architecture with a standard structure is introduced. The DHT algorithm can be easily divided into many parallel sections that can be performed at the same time. In addition, the proposed algorithm is well adapted for sub-expression sharing approaches that can be used to greatly minimise the hardware complexity of the extremely parallel VLSI implementation. Using the benefits of the proposed algorithm and the fact that we can easily share multipliers with the same constant, the number of multipliers has been substantially decreased in such a way that the number of multipliers is very small relative to the current algorithms. Or at any time, constant multipliers can be easily applied in VLSI.

Table 1: Summary of Literature Review.



Parameter	Doru et al. [5]	Wen-Liang Hsue et al. [3]	Shirali et al. [1]
No. of Slices	398	350	317
No. of 4 input LUTs	684	617	593
No. of bounded IOBs	256	256	256
Maximum combinational path delay(in ns)	39.931 ns	29.531 ns	27.571 ns

In the above table 1 the summary of the literature review is given on that basis our proposed work is carried forward.

5. Simulation & results

The comparison of DHT algorithm with existing algorithms is presented in Table 2, this has been observed that architectures based on DHT algorithm used less number of multipliers and adders. Hence, the complexity of VLSI architecture gets reduced.

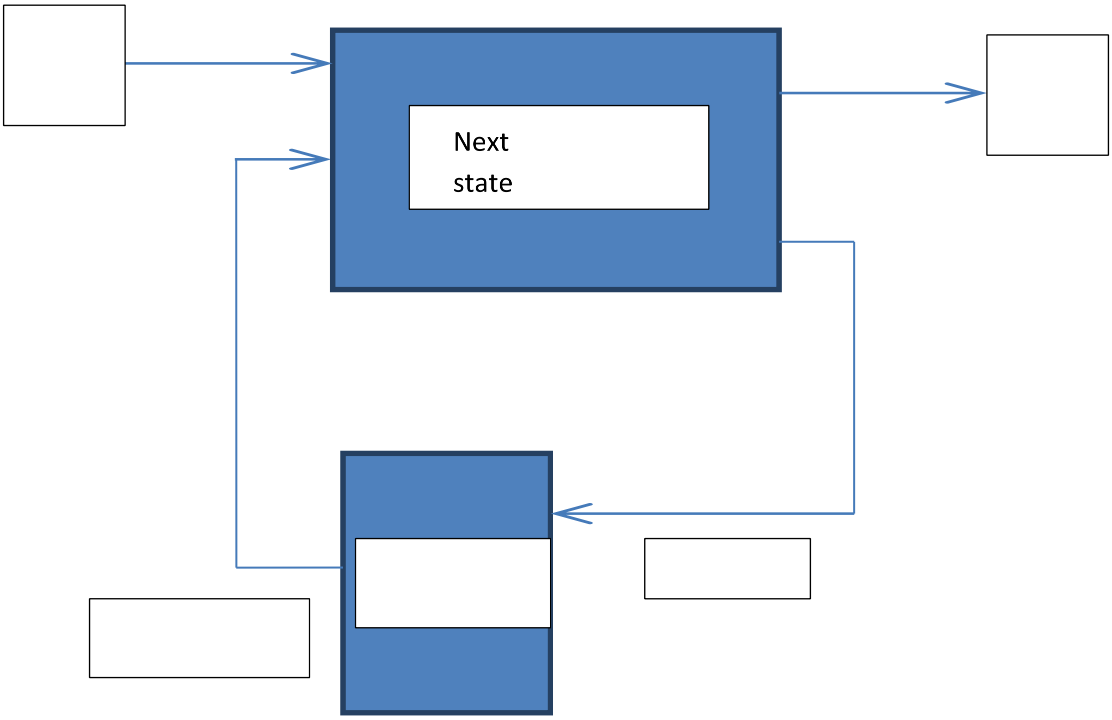


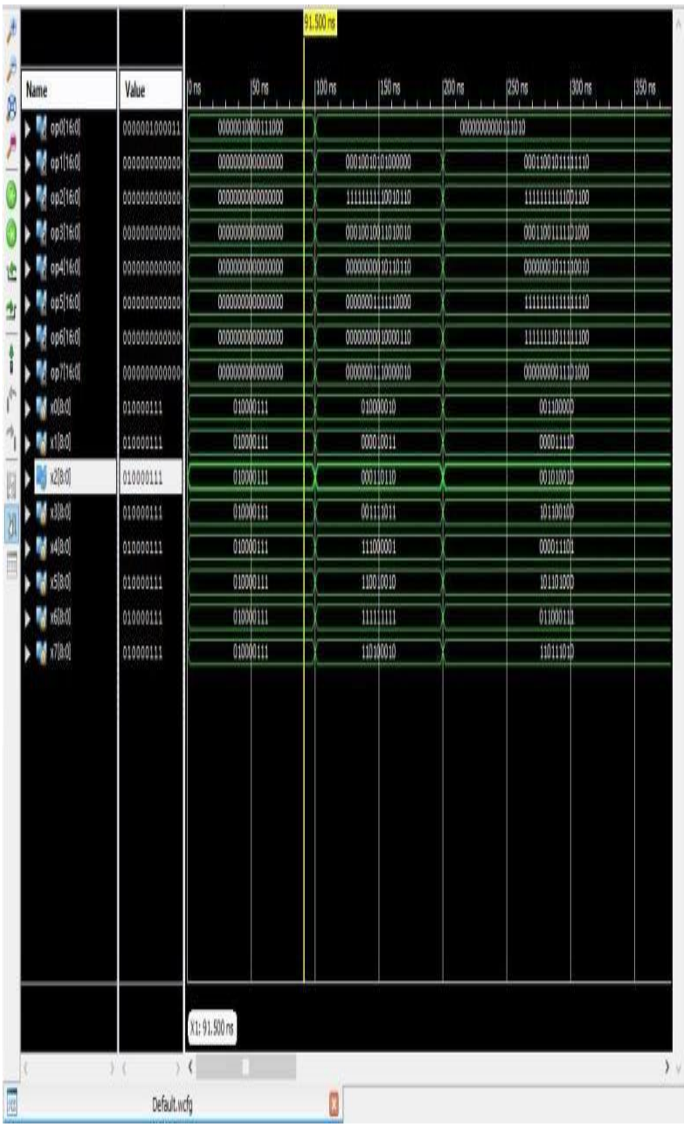
Figure 7: Block Diagram of VLSI [11].

In the above figure 7 the VLSI architecture has been shown.

Table 2 Comparison with other Algorithms

	Radix 2 [13]		**Radix 2 [13]***		**Radix 2 [11]****		**DHT**	
N	MUL	ADD	MUL	ADD	MUL	ADD	MUL	ADD
8	-	-	-	-	4	26	2	16

MUL-Multiplier ADD-Adder=



Name	Value
op0[16:0]	0000001000011
op1[16:0]	0000000000000
op2[16:0]	0000000000000
op3[16:0]	0000000000000
op4[16:0]	0000000000000
op5[16:0]	0000000000000
op6[16:0]	0000000000000
op7[16:0]	0000000000000
x0[8:0]	010000111
x1[8:0]	010000111
x2[8:0]	010000111
x3[8:0]	010000111
x4[8:0]	010000111
x5[8:0]	010000111
x6[8:0]	010000111
x7[8:0]	010000111

Figure 8: Time-slot representation.

In the above figure 8 the representation of the time-slot is given.

6. Conclusion

In this paper, a DHT algorithm for highly parallel and regular VLSI structure is presented for the length N = 8. The power consumption remains a major concern while dealing with VLSI circuits. The application of DHT algorithm leads to reduced hardware complexity, and hence reduced power consumption. The DHT algorithm implements sub-sharing expression technique with sharing of multipliers which have same constants. The DHT algorithm is mainly used in signal processing and image compression applications.

7. Future scope

The new algorithm can enhance the discovery efficiency and coverage. It is useful for service discovery of mobile application.

References

A. Shirali Parsaiand Swapnil Jain, "VHDL Implementation of Discrete Hartley Transform Using Urdhwa Multiplier", 978-1-4673-9542-7/15/$31.00 ©2015 IEEE.

B. Doru Florin Chiper, Senior Member, IEEE "Radix-2 Fast Algorithm for Computing Discrete Hartley Transform of Type III" IEEE TRANSACTIONS ON CIRCUITS AND SYSTEMS—II: EXPRESS BRIEFS, VOL. 59, NO. 5, MAY 2012

C. Michela Svaluto Moreolo, Raül Muñoz, and Gabriel Junyent "Novel Power Efficient Optical OFDM Based on Hartley Transform for Intensity-Modulated Direct-Detection Systems "JOURNAL OF LIGHTWAVE TECHNOLOGY, VOL. 28, NO. 5, MARCH 1, 2010

D. Chin-Kuo Jao, Syu-Siang Long, and Muh-Tian Shiue, Member, IEEE “DHT-Based OFDM System for Passband Transmission Over Frequency-Selective Channel “IEEE SIGNAL PROCESSING LETTERS, VOL. 17, NO. 8, AUGUST 2010

E. Saad Bouguezel, M. Omair Ahmad, Fellow, IEEE, and M. N. S. Swamy, Fellow, IEEE “New Parametric Discrete Fourier and Hartley Transforms, and Algorithms for Fast Computation “IEEE TRANSACTIONS ON CIRCUITS AND SYSTEMS—I: REGULAR PAPERS, VOL. 58, NO. 3, MARCH 2011

F. Liang Tao and Hon Keung Kwan, Senior Member, IEEE “Multirate-Based Fast Parallel Algorithms for 2-D DHT-Based Real-Valued Discrete Gabor Transform “IEEE TRANSACTIONS ON IMAGE PROCESSING, VOL. 21, NO. 7, JULY 2012

G. Pramod Kumar Meher, Senior Member, IEEE, Thambipillai Srikanthan, Senior Member, IEEE, and Jagdish C. Patra, Member, IEEE “Scalable and Modular Memory-Based Systolic Architectures for Discrete Hartley Transform

H. IEEE TRANSACTIONS ON CIRCUITS AND SYSTEMS—I: REGULAR PAPERS, VOL. 53, NO. 5, MAY

I. 2006

J. G. Eason, B. Noble, and I.N. Sneddon, “On certain integrals of Lipschitz-Hankel type involving products of Bessel functions,” Phil. Trans. Roy. Soc. London, vol. A247, pp. 529-551, April 1955. (references)

K. J. Clerk Maxwell, A Treatise on Electricity and Magnetism, 3rd ed., vol. 2. Oxford: Clarendon, 1892, pp.68-73.

L. I.S. Jacobs and C.P. Bean, “Fine particles, thin films and exchange anisotropy,” in Magnetism, vol. III, G.T. Rado and H. Suhl, Eds. New York: Academic, 1963, pp. 271-350.

M. R. Nicole, “Title of paper with only first word capitalized,” J. Name Stand. Abbrev., in press.

N. Y. Yorozu, M. Hirano, K. Oka, and Y. Tagawa, “Electron spectroscopy studies on magneto-optical media and plastic substrate interface,” IEEE Transl. J. Magn. Japan, vol. 2, pp. 740-741, August 1987 [Digests 9th Annual Conf. Magnetics Japan, p. 301, 1982].

O. M. Young, The Technical Writer’s Handbook. Mill Valley, CA: Uni