An FPGA Design for the Two-Band Fast Discrete Hartley Transform

Labros Pyrgas and Paris Kitsos
Digital IC dEsign and Systems Laboratory (DICES Lab)
Computer and Informatics Engineering Department
Technological Educational Institute of Western Greece
Antirrion, Greece
e-mail: lpyrgas@hotmail.com, pkitsos@ieee.org

Athanassios N. Skodras
Electrical and Computer Engineering Department
University of Patras
Patras, Greece
E-mail: skodras@upatras.gr

***Abstract*—The discrete Hartley transform finds numerous applications in signal and image processing. An efficient Field Programmable Gate Array implementation for the 64-point Two-Band Fast Discrete Hartley Transform is proposed in this communication. The architecture requires 57 clock cycles to compute the 64-point Two-Band Fast Discrete Hartley Transform and reaches a rate of up to 103.82 million samples per second at a 92 MHz clock frequency. The architecture has been implemented using VHDL and realized on a Cyclone IV FPGA of Altera.**

Keywords— Two-Band Fast Discrete Hartley Transform; Field Programmable Gate Array architecture (FPGA); Digital Signal Processing, VHDL.

I. INTRODUCTION

The discrete Hartley transform finds applications in signal and image processing, pattern recognition, seismic wave modeling, biomedical image compression, etc. The importance of the discrete Hartley transform is due to the fact that (a) it is a real transform, i.e. its computation does not involve complex arithmetic, (b) similarly to the discrete Fourier transform, there exist algorithms for its fast computation, and (c) its kernel is involutory, i.e. the same kernel (except a scaling factor) is used for the forward and inverse transforms. The continuous Hartley transform was first introduced by R. V. L. Hartley in 1942 [1]. The discrete Hartley transform (DHT) and the decimation-in-time (DIT) fast Hartley transform (FHT) were proposed by R. N. Bracewell in 1983 and 1984, respectively [2, 3]. In 1985 Meckelburg and Lipka proposed the decimation-in-frequency (DIF) FHT algorithm [4].

A number of FPGA implementations for the DHT have already been proposed [5, 6] with very good performance. More fast DHT implementations for signal and image processing applications continued to appear [7-10]. For example, in [7], a new fast DHT radix-2^2 algorithm is presented that is suitable for the pipeline implementation of the DHT; additionally, a method for reducing the number of multiplications and additions is given. In [8], an efficient method for the computation of the discrete Hartley transform of type II. Finally, in [9] and [10] other interesting implementations of the fast DHT are presented.

Recently, a new approach for the computation of the radix-2 fast Hartley transform (FHT) was introduced [11]. This algorithm, named Two-Band Fast DHT, is based on a two-band decomposition of the input data that results in a very regular structure, avoids the input or output data shuffling, and requires slightly less multiplications than the existing approaches, at the expense of an increased number of additions. An efficient FPGA design of the approach presented in [11] is proposed in this paper.

The paper is organized as follows: In Section II the Two-Band Fast DHT algorithm is briefly described. The proposed architecture and its hardware implementation are presented in detail in Section III. The synthesis and FPGA implementation results are shown in Section IV. Finally Section V concludes the paper.

II. TWO-BAND FAST DHT ALGORITHM

The type II discrete Hartley transform (DHT) of the N-point real-valued data x_n, $n=0,1,2,\ldots, N-1$ is defined as

$$H_N(k)=\sum_{n=0}^{N-1}x(n)cas(\frac{2\pi}{N}nk),\ k=0, 1,2, \ldots, N-1 \quad (1)$$

where $cas(\cdot)=cos(\cdot)+sin(\cdot)$. The transform is linear and its coefficients $H_N(k)$ are real numbers. The decomposition of each pair of input data $x(2n)$, $x(2n+1)$ into low-band values $x_L(n)$ and high-band values $x_H(n)$ we get:

$$x_L(n)=\frac{1}{2}[x(2n)+x(2n+1)] \quad (2a)$$

$$x_H(n)=\frac{1}{2}[x(2n)-x(2n+1)] \quad (2b)$$

or

$$x(2n)=x_L(n)+x_H(n) \quad (3a)$$

$$x(2n+1)=x_L(n)-x_H(n) \quad (3b)$$

The method then proceeds by applying the DHT core on the summations and the differences on the above low-band and high-band values of adjacent samples. Output coefficients are calculated by the following DHT pair:

$$H_N(k)=H_L(k)+[\cos\vartheta H_H(k)+\sin\vartheta H_H(\tfrac{N}{2}-k)] \quad (4)$$

$$H_N(k+\frac{N}{2})=H_L(k)-[\cos\vartheta H_H(k)+\sin\vartheta H_H(\tfrac{N}{2}-k)] \quad (5)$$

where

$$\vartheta = 2\pi k / N \tag{6}$$

$$H_L(k) = H^L_{N/2}(k) + H^H_{N/2}(k) \tag{7}$$

$$H_H(k) = H^L_{N/2}(k) - H^H_{N/2}(k) \tag{8}$$

$$H_H(-k) = H^L_{N/2}(-k) - H^H_{N/2}(-k) \tag{9}$$

and

$$H^L_{N/2}(k) = \sum_{n=0}^{N/2-1} x_L(n) cas(\frac{2\pi k}{N/2} n) \tag{10}$$

$$H^H_{N/2}(k) = \sum_{n=0}^{N/2-1} x_H(n) cas(\frac{2\pi k}{N/2} n) \tag{11}$$

$H^L_{N/2}(k)$, $H^H_{N/2}(k)$ are the N/2-point DHT's of $x_L(n)$ and $x_H(n)$, respectively.

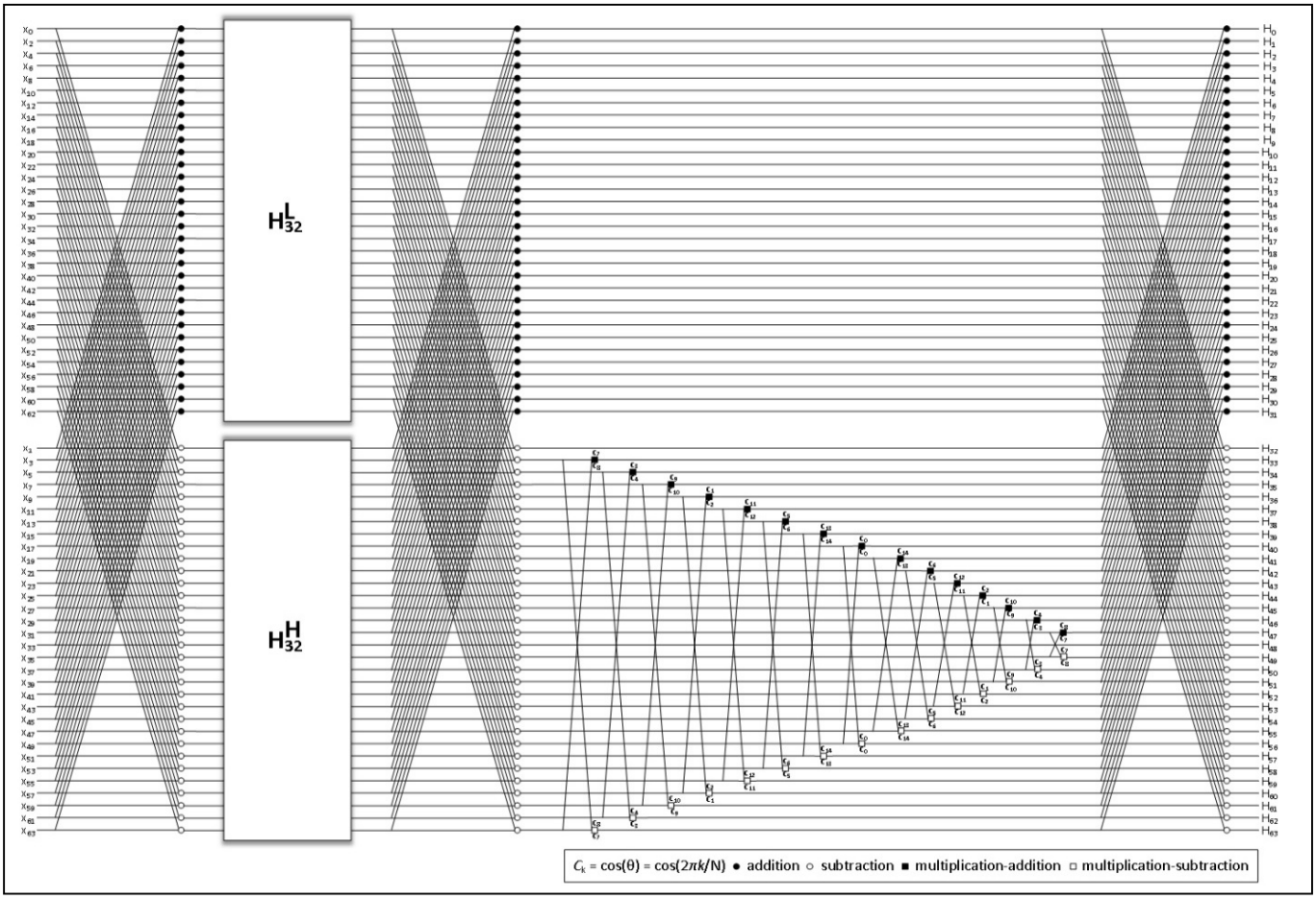

Fig. 1. Flow graph for computation of 64-point two-band fast DHT

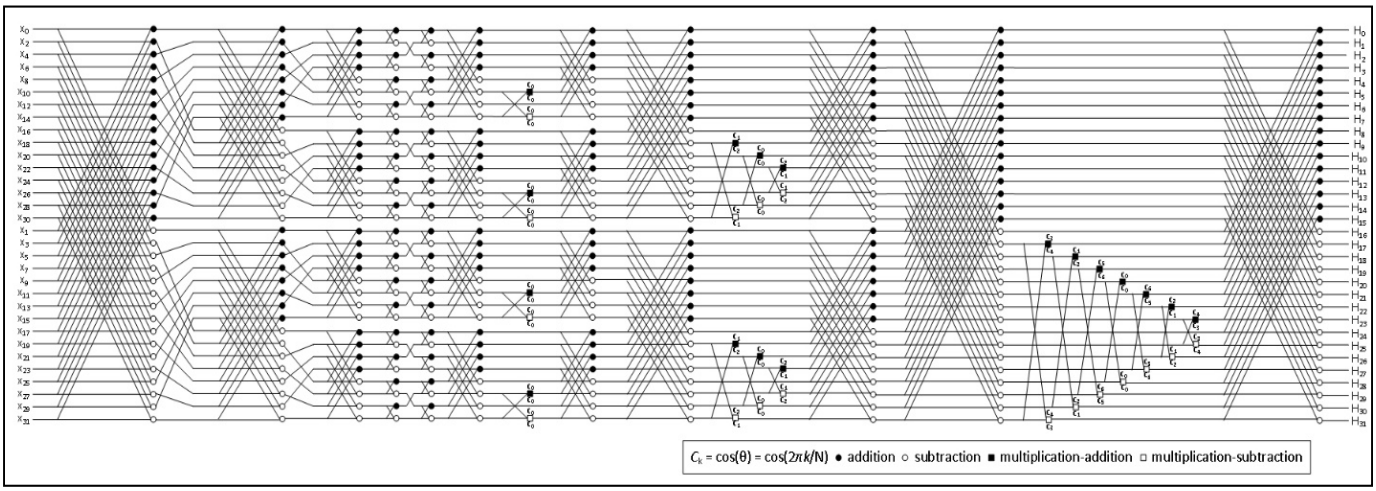

Fig. 2. Flow graph for computation of 32-point two-band fast DHT

Based on these, the 64-point Two-Band Fast DHT is composed of two 32-point Two-Band Fast DHTs, combined with 60 multiplications by twiddle factors. The 64-point Two-Band Fast DHT is shown in Fig. 1, while the 32-point Two-Band Fast DHT is shown if Fig. 2.

III. Hardware Architecture

The hardware architecture of the 64-point Two-Band Fast DHT is shown in Fig.3. The design has 64 8-bit data inputs and 64 14-bit outputs. During the first clock cycle, the input data are driven to the adders/subtractors (ADD/SUB) block. In the next stage, the adders and subtractors perform the calculations for the first butterflies. In order to keep full accuracy, the outputs of the butterflies should be 14-bit long. The results are then stored in the appropriate registers depending on the signals from the control unit block. In the second stage the data are driven to the adders/subtractors block or the multipliers (MULTIPLIERS) block in order to be processed. This process continues until all necessary additions and multiplications are executed. In the final stage the final results from the adders/subtractors block are driven to the output.

The register (REGISTER) file consists of 64 13-bit registers in order to store the data in full accuracy. A limited number of multiplexers are used to reorder and drive the inputs to the right registers during the writing state and the selected data to the correct outputs during the reading state. The regular and modular structure of the architecture results in multiplexing the same registers avoiding the area and delay overhead present in irregular architectures.

The MUX_IN_REG multiplexer drives the input data (x0, x1…x63) to the adders/subtractors block during the first clock cycle. In all the subsequent clock cycles the multiplexer forwards the data coming from the register file to the adders/subtractors block.

The MUX_MULT multiplexer's inputs are the outputs from the adder/subtractors block and the register file. For the first set of multiplications in each stage the multiplexer forwards six values directly from the output of the adders/subtractors block while in all the subsequent sets of multiplications for that stage the multiplexer forwards the data coming from the register file.

The MUX_DIV's inputs are the outputs from the adder/subtractors block and the multipliers block. Based on the selection signals from the control unit it drives one of the inputs to the register file. It can also right shift the data, in order to perform the necessary divisions. A division by 8 is required for getting a correct final output. In order to maintain maximum precision and the storing registers as small as possible, a split into 4 divisions by 2 is performed.

The adders/subtractors block performs the butterflies and consists of 8 units of 8 adders/subtractors in parallel to a total of 64 adders/subtractors with a butterfly structure that takes advantage of the high design symmetry. Fig. 4 illustrates the butterfly structure of one of the eight subunits.

The multipliers block consists of 3 units of 4 signed multipliers, 2 adders/subtractors each to a total of 12 13-bit multipliers, 6 adders/subtractors (3 each) and a 15 13-bit registers ROM as shown in Fig. 5. The multipliers block can perform 12 multiplications at a time but due to the symmetric architecture all multiplications are by twiddle factors so only 6 data points and 6 coefficients are used each time.

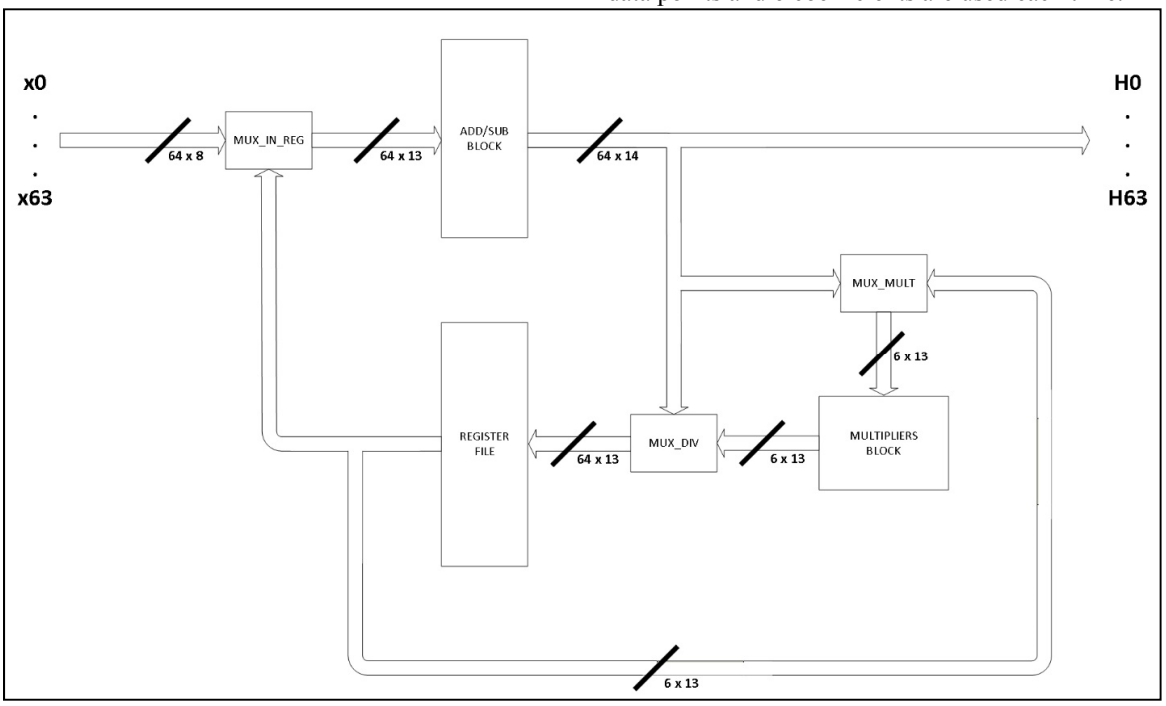

Fig. 3. The 64-point Two-Band Fast DHT architecture

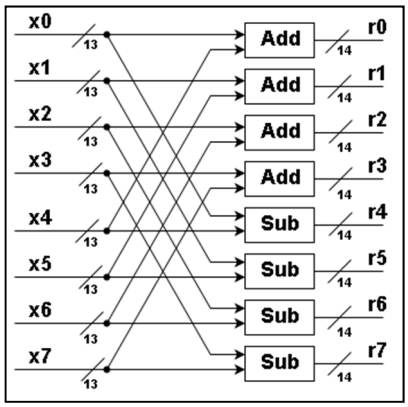

Fig. 4. The 8-point adder/subtractor subunit

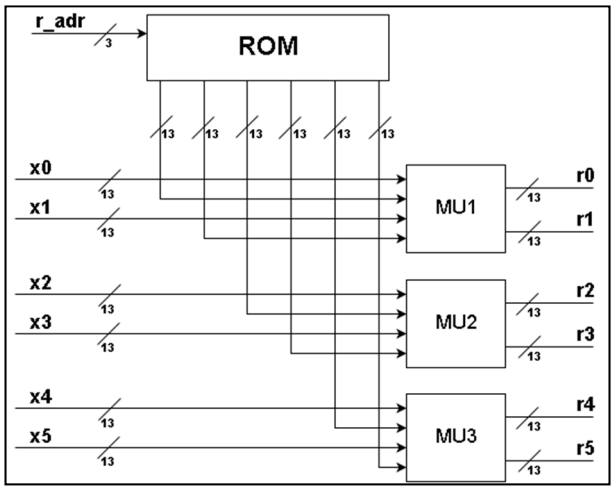

Fig. 5. The multipliers block

The ROM contains the 15 coefficients (Table I) necessary for the multiplication process. The initial decimal coefficients are multiplied by 4096 and then are store as integers during the initialization of the ROM to reduce the complexity of the multiplications. Accuracy degradation is minimal.

The architecture of one of the multiplication units is shown in Fig. 6. Each unit takes as input two data points and two coefficients. Only the [24…12] bits of the initial result are sent to the output, which corresponds to a division by 4096 to reverse for the initial scaling of the coefficients.

TABLE I. MULTIPLICATION COEFFICIENTS

Coefficient	Decimal	ROM contents
c_0	0.707107	0101101010000
c_1	0.923880	0111011001000
c_2	0.382683	0011000011111
c_3	0.980785	0111110110001
c_4	0.195090	0001100011111
c_5	0.831470	0110101001110
c_6	0.555570	0100011100100
c_7	0.995185	0111111101100
c_8	0.098017	0000110010001
c_9	0.956940	0111101010000
c_{10}	0.290285	0010010100101
c_{11}	0.881921	0111000011100
c_{12}	0.471397	0011110001011
c_{13}	0.773010	0110001011110
c_{14}	0.634393	0101000100110

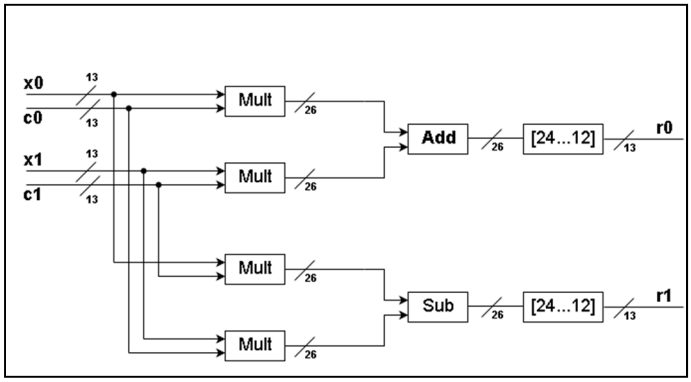

Fig. 6. The MULT subunit

IV. SYNTHESIS RESULTS

Initially, a behavioral model was developed that simulated the behavior of the 64-point Two-Band Fast DHT algorithm. This model was used in order to verify the correct functionality of the proposed architectures of the 64-point Two-Band Fast DHT using ModelSim. The proposed architectures have been captured using VHDL. The VHDL codes are implemented and synthesized by means of the Altera Quartus tool. The target FPGA device was Cyclone IV EP4CE15E22C6. The measurements are focused on the design throughput and the consumed FPGA area resources. Table II presents the synthesis results of the proposed architecture.

The 64-point Two-Band Fast DHT architecture consumes 10742 Logic Elements (LEs), a figure that compares favorably with [12] given the scale of the two computations. It achieves a clock frequency of up to 92 MHz at a temperature of 85° C. It needs 57 clock cycles to process 64 points with a bit rate equal to 103.82 million samples per second. It uses only 12 multipliers which is equal to the number of multipliers that [12] uses for 16 points and 4 less than the implementation in [13] for a 32-point proposes.

The performance can be further improved by mapping the multipliers to the available DSP elements of the device. With DSP mapping the 64-point Two-Band fast DHT architecture consumes 8560 LEs and achieves a clock frequency of up to 105.45 at the temperature of 85° C. It also needs 57 clock cycles to process the 64 points which corresponds to a bit rate equal to 118.40 million samples per second.

TABLE II. SYNTHESIS RESULTS

FPGA Device	Cyclone IV E EP4CE15E22C6			
FPGA Resources	64-point Two-Band Fast DHT (DSP mapping)		64-point Two-Band Fast DHT	
# Total Registers	4352		2472	
# Total LEs	8560		10742	
Embedded Multiplier 9-bit elements	24 (for 12 13x13 bit multipliers)		0	
Operating Conditions (1200mV)	T = 0ºC	T = 85ºC	T = 0ºC	T = 85ºC
Freq (MHz)	119.10	105.45	102.98	92.47
Sample rate (MSamples / sec)	133.72	118.40	115.62	103.82

V. CONCLUSIOS

An FPGA architecture for the 64-point Two-Band Fast DHT computation is presented in this paper. The proposed design is implemented on a highly parallel architecture of low hardware complexity due its simplified and symmetric structure. The synthesis results prove that the architecture is the preferred choice for applications of low-area and high sample rate requirements.

REFERENCES

[1] R. V. L. Hartley, 'A more symmetrical Fourier analysis applied to transmission problems', Proc. IRE, 1942, 30, (3), pp. 144–150.

[2] R. N. Bracewell, 'Discrete Hartley transform', J. Opt. Soc. Am., 1983,73, (12), pp. 1832–1835.

[3] R. N. Bracewell, 'The fast Hartley transform', Proc. IEEE, 1984, 72, (8), pp. 1010–1018.

[4] H. J. Meckelburg and D. Lipka, "Fast Hartley transform algorithm," Electronics Letters, vol. 21, no. 8, pp. 311-313, Apr. 1985.

[5] A. Amira, "An FPGA-based parameterisable system for discrete Hartley transforms implementation", Proc. Int. Conf. Image Process. (ICIP'03), 2, (2003): 567-570.

[6] A. Amira and A. Bouridane, "An FPGA implementation of discrete Hartley transforms", Proc. 7th Int. Symp. Signal Process. Appl., 1, (2003): 625-628.

[7] M. T. Hamood, S. Boussakta, 'New radix-based FHT algorithm for computing the discrete Hartley transform'. IEEE Int. Conf. on Acoustics, Speech and Signal Proc. (ICASSP), 22–27 May 2011, pp. 1581–1584.

[8] D. F. Chiper, 'Fast radix-2 algorithm for the discrete Hartley transform of type II', IEEE Signal Proc Lett., 2011, 18, (11), pp. 687–689.

[9] P. Yu, H. Hua, 'A new fast recursive algorithm for computing discrete Hartley transform and its implementation', Signal Process., 2001, 81, (10), pp. 2235–2241.

[10] P. Saha, P. Bhattacharyya, D. Kumar and A. Dandapat, "ASIC implementation of high speed processor for computing fast Hartley transformation" Proc. Int. Conf. on Advanced Electronic Systems (ICAES), 2013, pp. 334-336.

[11] A. N. Skodras, M. F. Aburdene and A.K. Nandi, "Two-band fast Hartley transform", Electronics Letters 8th January 2015 Vol. 51 No. 1 pp. 57–59.

[12] R. C. de Oliveira, M. Hélio de Oliveira, R. M. de Souza, J. P. Edval Santos, 'A flexible implementation of a matrix Laurent series-based 16-point fast Fourier and Hartley transforms', arXiv preprint arXiv: 1502.05880. , 2015.

[13] M. Tamilselvi, M. P. Gomathi, A. Lakshminarayanan, 'Resourceful fast DHT algorithm for VLSI implementation by split radix algorithm', International Journal of Research in Engineering and Technology (IJRET), Volume 2014, 3, (14), pp. 44–48.