ISSN (Print) : 2320 – 3765
ISSN (Online): 2278 – 8875

International Journal of Advanced Research in Electrical, Electronics and Instrumentation Engineering

(An ISO 3297: 2007 Certified Organization)

Vol. 3, Issue 10, October 2014

Novel DHT Algorithm Implementation Using Sharing Multipliers

G.Venkatesh, O.Sudhakar

PG student (M.Tech), GIET, Rajahmundry, India

Assistant Professor, Dept. of ECE, GIET, Rajahmundry, India

ABSTRACT: A new very large scale integration (VLSI) algorithm for a 2^N-length discrete Hartley transform (DHT) that can be efficiently implemented on a highly modular and parallel VLSI architecture having a regular structure is presented. In the proposed method multiply the multipliers by using array multiplier and to add the co-efficients by using Carry Look-ahead Adder (CLA). In the proposed system we are reducing the area by using the array multiplier and Carry Look-ahead Adder (CLA). The DHT algorithm can be efficiently split on several parallel parts that can be executed concurrently. Moreover, the proposed algorithm is well suited for the sub expression sharing techniques that can be used to significantly reduce the hardware complexity of the highly parallel VLSI implementation. Using the advantages of the proposed algorithm and the fact that we can efficiently share the multipliers with the same constant, the number of the multipliers has been significantly reduced such that the number of multipliers is very small comparing with that of the existing algorithms. Moreover, the multipliers with a constant can be efficiently implemented in VLSI.

KEYWORDS*:* Discrete Hartley transform (DHT), DHT domain processing, fast algorithms.

I. INTRODUCTION

The Discrete Fourier transform (DFT) is used in many digital signal processing applications as in signal and image compression techniques, filter banks [1], signal representation, or harmonic analysis [2]. The discrete Hartley transform (DHT) [2], [3] can be used to efficiently replace the DFT when the input sequence is real. In the literature, there are some fast algorithms for the computation of DHT [4]–[7] and some algorithms for the computation of generalized DHT [8]–[10]. There are also several split-radix algorithms for computing DHT with a low arithmetic cost. Thus, Sorensen *et al.* [11] and Malvar [12] proposed split-radix algorithms for DHT with a low arithmetic cost. Bi [13] proposed another split-radix algorithm where the odd-indexed transform outputs are computed using an indirect method. The classical split-radix algorithm is difficult to implement on VLSI due to its irregular computational structure and due to the fact that the butterflies significantly differ from stage to stage. Thus, it is necessary to derive new such algorithms that are suited for a parallel VLSI system.

There are also in the literature several fast algorithms that use a recursive strategy as those in [14] for discrete cosine transform (DCT) and that in [10] for generalized DHT. Since DHT is computationally intensive, it is necessary to derive dedicated hardware implementations using the VLSI technology. One category of VLSI implementations is represented by systolic arrays. There are many systolic array implementations of DHT [15]–[18]. Systolic array architectures are modular and regular, but they use particularly pipelining and not parallel processing to obtain a high-speed processing. In the literature, highly parallel solutions as those in [8] and [19] were also proposed. In [8], a highly parallel and modular solution for the implementation of type-III DHT based on a new VLSI algorithm is proposed. In [19], we have a highly parallel solution for the implementation of DHT based on a direct implementation of fast Hartley transform (FHT). It is worth to note that hardware implementations of FHT are rare. Multipliers in a VLSI structure consume a large portion of the chip area and introduce significant delays. This is the reason why memory-based solutions to implement multipliers have been more and more used in the literature [15], [20]–[24]. To efficiently implement multipliers with lookup-table-based solutions, it is necessary that one operand to be a constant. When one of the operands is constant, it is possible to store all the partial results in a ROM, and the number of memory words is significantly reduced from 2^{2L} to 2^L. In this brief, a new VLSI DHT algorithm that is well suited for a VLSI

10.15662/ijareeie.2014.0310036

ISSN (Print) : 2320 – 3765
ISSN (Online): 2278 – 8875

International Journal of Advanced Research in Electrical, Electronics and Instrumentation Engineering

(An ISO 3297: 2007 Certified Organization)

Vol. 3, Issue 10, October 2014

implementation on a highly parallel and modular architecture is proposed. It can be used for designing a completely novel VLSI architecture for DHT. Moreover, using subexpression sharing technique [25] and sharing the multipliers with the same constant, the hardware complexity can be significantly reduced the number of multipliers being very small, significantly less than that in [8]. In the proposed solution, we have used only multipliers with a constant that can be efficiently implemented in VLSI. The proposed solution is not only appealing by its high level of parallelism and by using a modular and regular structure but it can be also used to obtain a small hardware complexity by extensively sharing the common blocks. The rest of this brief is organized as follows. In Section II, we present a new algorithm for computing an *N*-point DHT. In Section III, we present an algorithm for a small-length DHT. In Section IV, we analyze the arithmetic cost, and in Section V, we present some examples of our algorithm. In Section VI, we present the new VLSI architecture. The conclusion is presented in Section VII

II. NEW VLSI ALGORITHM FOR DHT

Let $N \geq 4$ be a power of two. For any real input sequence $\{x(i) : i = 0, 1, \ldots, N-1\}$, the DHT($N$) is defined by

$$X(k) = \text{DHT}(N)\{x(i)\} = \sum_{i=0}^{N-1} x(i) \cdot \text{cas}[2ki\pi/N] \quad \text{for } k = 0, 1, \ldots, N-1 \tag{1}$$

where $\text{cas}(x) = \cos(x) + \sin(x)$.

We can compute a N-length DHT using a new algorithm given by the following relations:

$$X_N(k)\{x(i)\} = X_{N/2}(k)\{x(2i)\} + u(0) \cdot \sin(2k\pi/N) + [X_{N/2}(k)\{u(i)\} - u(0)/2] \cdot 2 \cdot \cos(2k\pi/N) \tag{2}$$

$$X_N(N/2+k)\{x(i)\} = X_{N/2}(k)\{x(2i)\} - u(0) \cdot \sin(2k\pi/N) - [X_{N/2}(k)\{u(i)\} - u(0)/2] \cdot 2 \cdot \cos(2k\pi/N)] \quad \text{for } k = 0, 1, \ldots, N/4-1 \tag{3}$$

$$X_N(N/2-k)\{x(i)\} = X_{N/2}(N/2-k)\{x(2i)\} + u(0) \cdot \sin(2k\pi/N - [X_{N/2}(N/2-k)\{u(i)\} - u(0)/2] \cdot 2 \cdot \cos(2k\pi/N) \tag{4}$$

$$X_N(N-k)\{x(i)\} = X_{N/2}(N/2-k)\{x(2i)\} - u(0) \cdot \sin(2k\pi/N) + [X_{N/2}(N/2-k)\{u(i)\} - u(0)/2] \cdot 2 \cdot \cos(2k\pi/N) \quad \text{for } k = 1, \ldots, N/4 \tag{5}$$

where

$$X_{N/2}(k)\{x(2i)\} = \sum_{i=0}^{N/2-1} x(2i) \cdot \text{cas}\left[2ki\frac{\pi}{N/2}\right] \tag{6}$$

$$X_{N/2}(k)\{u(i)\} = \sum_{i=0}^{N/2-1} u(i) \cdot cas\left[2ki\frac{\pi}{N/2}\right] \tag{7}$$

are DHT of length $N/2$, with $\{u(i) : i = 0, 1, \ldots, N/2-1\}$ an auxiliary input sequence given by

$$u(N/2-1) = x(N-1) \tag{8}$$

$$u(i) = x(2i+1) - u(i+1) \quad \text{for } i = N/2-2, \ldots, 1, 0. \tag{9}$$

ISSN (Print) : 2320 – 3765
ISSN (Online): 2278 – 8875

International Journal of Advanced Research in Electrical, Electronics and Instrumentation Engineering

(An ISO 3297: 2007 Certified Organization)

Vol. 3, Issue 10, October 2014

For the computation of (2)–(5), there are necessary extra $7N/4$ additions and $N/2$ multiplications, if we share the multipliers with the same constant. For the computation of the auxiliary input sequence using (8) and (9), there are necessary extra $N/2 - 1$ additions. The obtained algorithm can be used as a VLSI algorithm where the number of multipliers can be significantly reduced by sharing the multipliers with the same constant as will be shown in Section VI. The number of multipliers can be further reduced using sub expression sharing techniques and the sharing of multipliers with the same constant, as shown in Section VI.

TABLE I
COMPUTATIONAL COMPLEXITY

	Radix 2 [13]*		Radix-2 [13]**		Radix-2 [11]		Proposed	
N	M	A	M	A	M	A	M	A
8					4	26	2	16
16	10	62	10	62	20	74	12	67
32	40	168	34	174	69	194	40	205
64	118	418	98	438	196	482	112	553
128	320	1008	258	1070	516	1154	288	1393
256	806	2354	642	2518	1284	2690	704	3361

III. ALGORITHM FOR A SMALL DHT

An efficient implementation of a fast DHT algorithm closely depends on an efficient algorithm for a small DHT. We present here an efficient DHT algorithm for a length $N = 8$

$$\begin{aligned}
X(0) &= [(x(0)+x(4))+(x(2)+x(6))] \\
&\quad +[(x(1)+x(5))+(x(3)+x(7))] \\
X(2) &= [(x(0)+x(4))-(x(2)+x(6))] \\
&\quad +[(x(1)+x(5))-(x(3)+x(7))] \\
X(4) &= [(x(0)+x(4))+(x(2)+x(6))] \\
&\quad -[(x(1)+x(5))+(x(3)+x(7))] \\
X(6) &= [(x(0)+x(4))-(x(2)+x(6))] \\
&\quad -[(x(1)+x(5))-(x(3)+x(7))] \\
X(1) &= [x(0)-x(4)]+[x(2)-x(6)]+c[x(1)-x(5)] \\
X(3) &= [x(0)-x(4)]-[x(2)-x(6)]+c[x(3)-x(7)] \\
X(5) &= [x(0)-x(4)]+[x(2)-x(6)]-c[x(1)-x(5)] \\
X(7) &= [x(0)-x(4)]-[x(2)-x(6)]-c[x(3)-x(7)]
\end{aligned}$$

with $c = \sqrt{2}$.

We have MDHT(8) = 2 and ADHT(8) = 16 as defined in the following. Due to the fact that we have to multiply with the same constant "c," we can share the same multiplier, thus further reducing the number of multipliers

IV. ARITHMETIC COST

Let ADHT(N) and MDHT(N) denote the number of additions and multipliers for computing DHT(N). We have

$$M_{\mathrm{DHT}(N)} = 2M_{\mathrm{DHT}(N/2)} + (1/2)N \qquad (10)$$

$$A_{\mathrm{DHT}(N)} = 2A_{\mathrm{DHT}(N/2)} + (9/4)N - 1 \qquad (11)$$

where MDHT(8) = 2 and ADHT(8) = 16. Solving the recursions (10) and (11), we obtain

ISSN (Print) : 2320 – 3765
ISSN (Online): 2278 – 8875

International Journal of Advanced Research in Electrical, Electronics and Instrumentation Engineering

(An ISO 3297: 2007 Certified Organization)

Vol. 3, Issue 10, October 2014

$$M_{\mathrm{DHT}(N)} = \frac{1}{2}N(\log_2 N - 5) \quad (12)$$

$$A_{\mathrm{DHT}(N)} = \frac{9}{4}N\log_2 N - \frac{39}{8}N + 1. \quad (13)$$

Table I lists the required number of multiplications and additions for the proposed algorithm, the Sorensen one and Bi algorithm, where rotations are implemented with four multiplications and two additions (Radix-2 [13] *) and with three multiplications and three additions (Radix-2 [13] **). The values of *M* in the proposed algorithm are computed considering that the multipliers with the same constant are shared. The number of multipliers in Sorensen algorithm [11] is significantly greater than that in the proposed one. The number of multipliers for Bi algorithm where rotations are implemented with four multiplications and two additions is greater than the necessary number of multipliers for our algorithm and slightly smaller when the rotations are implemented with three multiplications and three additions. However, the split-radix algorithm has an irregular structure and is difficult to be implemented in hardware as opposed to our algorithm that has a regular and modular structure and can be very easily implemented in parallel as it will be shown in Section VI for a DHT of length $N = 32$. Moreover, the number of multipliers in the proposed implementation can be significantly further reduced by sharing multiplications as shown in Section IV.

V. EXAMPLE OF THE PROPOSED ALGORITHM

We shall illustrate the main features of the proposed algorithm considering a DHT of length $N = 32$.

A. DHT of Length N = 32

We first compute recursively the auxiliary input sequences

$$u^{(0)}(15) = x(31) \quad (14)$$

$$u^{(0)}(i) = x(2i+1)) - u^{(0)}(i+1) \quad \text{for } i = 0, 1, \ldots, 14 \quad (15)$$

$$v^{(0)}(7) = x(30) \quad (16)$$

$$v^{(0)}(i) = x(4i+2)) - v^{(0)}(i+1) \quad \text{for } i = 0, 1, \ldots, 6 \quad (17)$$

$$u^{(1)}(7) = u^{(0)}(15) \quad (18)$$

$$u^{(1)}(i) = u^{(0)}(2i+1)) - u^{(1)}(i+1) \quad \text{for } i = 0, 1, \ldots, 6. \quad (19)$$

Then, we have to compute in parallel (21)–(28).

These equations have been obtained by a further reformulation of the equations obtained directly from (2)–(5) in such a way that we can extensively use the technique of sub expression sharing [18] and sharing the multipliers with the same constant. Thus, the number of multipliers has been significantly reduced at only 16, a significantly lower value than the theoretical value 40 from Table I that has been obtained using (2)–(5) without using the aforementioned technique. As can be seen, the proposed VLSI algorithm has a very good potential for using hardware sharing techniques, and many sub expressions have been used in common. We can thus significantly reduce the hardware complexity of the VLSI implementation. Moreover, due to the fact that the same constant is used in several multiplications, we can use the technique of sharing the multipliers with the same constant. Having only multiplications with a constant, we can efficiently implement these multipliers in VLSI.

VI. HIGHLY PARALLEL VLSI ARCHITECTURE

In order to clearly illustrate the features and advantages of the proposed algorithm, the VLSI architecture for a DHT of length $N = 32$ is presented in Fig. 1(a) and (b). It can be seen that the proposed architecture is highly parallel and has a modular and regular structure being formed of only a few blocks: U, MUL, ADD/SUB, XCH,

ISSN (Print) : 2320 – 3765
ISSN (Online): 2278 – 8875

International Journal of Advanced Research in Electrical, Electronics and Instrumentation Engineering

(An ISO 3297: 2007 Certified Organization)

Vol. 3, Issue 10, October 2014

and a few additional adders/subtracters. The "U" blocks implement (20), XCH blocks interchange the values and are simply implemented in hardware by appropriate wiring, and MUL blocks are used to implement the shared multipliers with a constant. This block contains four multipliers with a constant. Each multiplier is shared by four input sequences that are multiplied with the same constant in an interleaved manner using multiplexers and demultiplexers controlled by two clocks. One of the advantages of this algorithm and architecture is the fact that the multiplications with the same constant are shared in the MUL blocks. Thus, the number of multipliers is significantly less than the value 40 given in Table I which has become now only 16. The final values $Y\_(k)$ of Section A and $Y0(k)$ of Section B are finally added to obtain the output sequence $Y(k)$ using an additional adder not presented in Fig. 1 for simplicity. The proposed architecture has a high throughput of 32 samples per clock and can be pipelined. It is highly parallel using a low hardware complexity structure. The multipliers with a constant in MUL blocks can be efficiently implemented in hardware using the techniques proposed in [20]–[24]. Parallel processing is one of the major ways to reduce power consumption, the high processing speed being traded off for low power using the reduction of the supply voltage value [26]. The required control structure is very simple which is another important advantage.

We define another module as

$$U_8(k)\{x_a(i)\} = X_8(k)\{x_a(i)\} - x_a(0)/2. \qquad (20)$$

VII. CHIPER: NOVEL VLSI DHT ALGORITHM FOR A HIGHLY MODULAR AND PARALLEL ARCHITECTURE

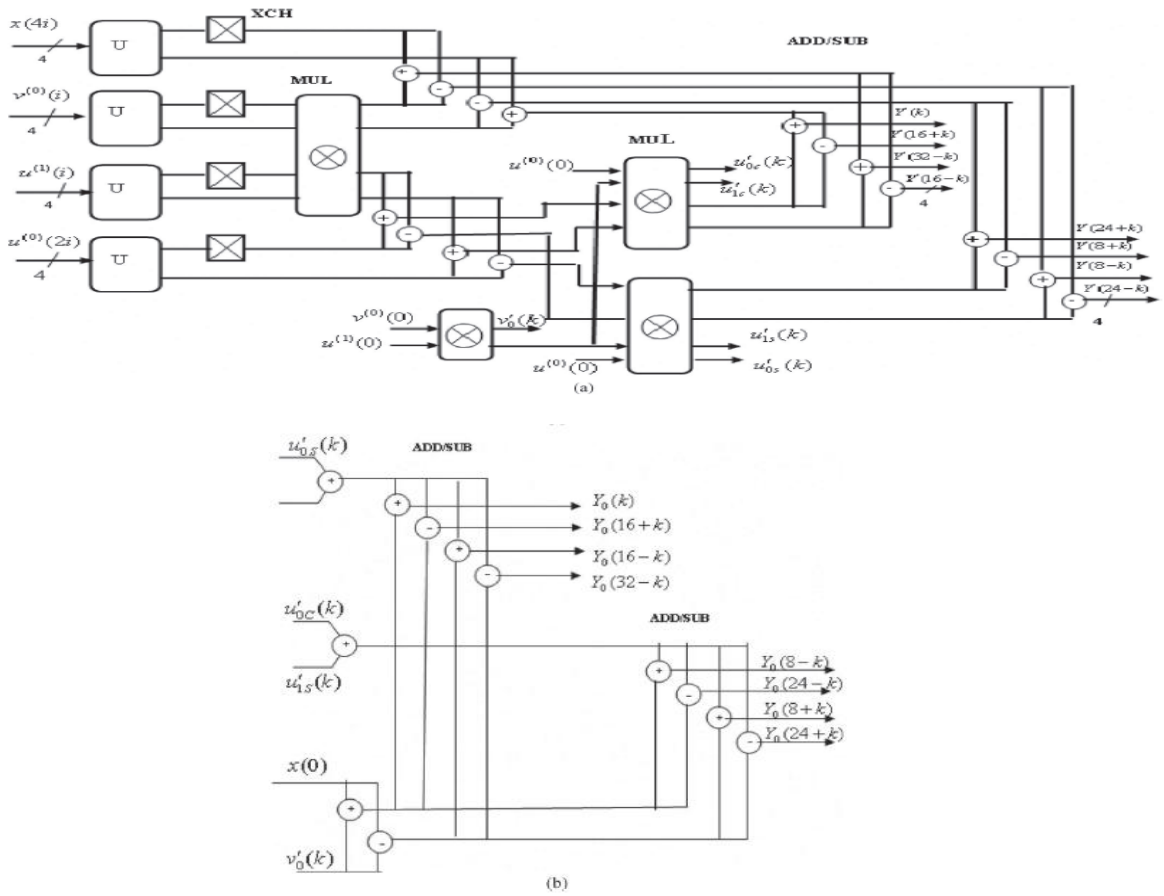


Fig.1. (a) VLSI architecture for DHT of length $N = 32$ (Section A). (b) VLSI architecture for DHT of length $N = 32$ (Section B).

10.15662/ijareeie.2014.0310036

ISSN (Print) : 2320 – 3765
ISSN (Online): 2278 – 8875

International Journal of Advanced Research in Electrical, Electronics and Instrumentation Engineering

(An ISO 3297: 2007 Certified Organization)

Vol. 3, Issue 10, October 2014

For $X_{32}(k)$, we can write the following relations:

$$\begin{aligned}
&X_{32}(k)\{x(i)\}\\
&=\left[U_8(k)\{x(4i)\}+U_8(k)\left\{v^{(0)}(i)\right\}\cdot 2\cdot\cos(2k\pi/16)\right]\\
&\quad+x(0)/2+v^{(0)}(0)\cdot\sin(2k\pi/16)+u^{(0)}(0)\cdot\sin(2k\pi/32)\\
&\quad+\left[U_8(k)\left\{u^{(0)}(2i)\right\}+U_8(k)\left\{u^{(1)}(i)\right\}\cdot 2\cdot\cos(2k\pi/16)\right]\\
&\quad\cdot 2\cdot\cos(2k\pi/32)+u^{(1)}(0)\cdot 2\sin(2k\pi/16)\cdot\cos(2k\pi/32)
\end{aligned}\tag{21}$$

$$\begin{aligned}
&X_{32}(k+8)\{x(i)\}\\
&=\left[U_8(k)\{x(4i)\}-U_8(k)\left\{v^{(0)}(i)\right\}\cdot 2\cdot\cos(2k\pi/16)\right]\\
&\quad+x(0)/2-v^{(0)}(0)\cdot\sin(2k\pi/16)+u^{(0)}(0)\cdot\cos(2k\pi/32)\\
&\quad-\left[U_8(k)\left\{u^{(0)}(2i)\right\}-U_8(k)\left\{u^{(1)}(i)\right\}\cdot 2\cdot\cos(2k\pi/16)\right]\\
&\quad\cdot 2\cdot\sin(2k\pi/32)+u^{(1)}(0)\cdot 2\sin(2k\pi/16)\cdot\sin(2k\pi/32)
\end{aligned}\tag{22}$$

$$\begin{aligned}
&X_{32}(16+k)\{x(i)\}\\
&=\left[U_8(k)\{x(4i)\}+U_8(k)\left\{v^{(0)}(i)\right\}\cdot 2\cdot\cos(2k\pi/16)\right]\\
&\quad+x(0)/2+v^{(0)}(0)\cdot\sin(2k\pi/16)-u^{(0)}(0)\cdot\sin(2k\pi/32)\\
&\quad-\left[U_8(k)\left\{u^{(0)}(2i)\right\}+U_8(k)\left\{u^{(1)}(i)\right\}\cdot 2\cdot\cos(2k\pi/16)\right]\\
&\quad\cdot 2\cdot\cos(2k\pi/32)-u^{(1)}(0)\cdot 2\sin(2k\pi/16)\cdot\cos(2k\pi/32)
\end{aligned}\tag{23}$$

$$\begin{aligned}
&X_{32}(24+k)\{x(i)\}\\
&=\left[U_8(k)\{x(4i)\}-U_8(k)\left\{v^{(0)}(i)\right\}\cdot 2\cdot\cos(2k\pi/16)\right]\\
&\quad+x(0)/2-v^{(0)}(0)\cdot\sin(2k\pi/16)-u^{(0)}(0)\cdot\cos(2k\pi/32)\\
&\quad+\left[U_8(k)\left\{u^{(0)}(2i)\right\}-U_8(k)\left\{u^{(1)}(i)\right\}\cdot 2\cdot\cos(2k\pi/16)\right]\\
&\quad\cdot 2\cdot\sin(2k\pi/32)-u^{(1)}(0)\cdot 2\sin(2k\pi/16)\cdot\sin(2k\pi/32)\\
&\qquad\text{For } k=0,1,\ldots,3
\end{aligned}\tag{24}$$

$$\begin{aligned}
&X_{32}(8-k)\{x(i)\}\\
&=\left[U_8(8-k)\{x(4i)\}\right.\\
&\qquad\left.-U_8(8-k)\left\{v^{(0)}(i)\right\}\cdot 2\cdot\cos(2k\pi/16)\right]\\
&\quad+x(0)/2+v^{(0)}(0)\cdot\sin(2k\pi/16)+u^{(0)}(0)\cdot\cos(2k\pi/32)\\
&\quad+\left[U_8(8-k)\left\{u^{(0)}(2i)\right\}\right.\\
&\qquad\left.-U_8(8-k)\left\{u^{(1)}(i)\right\}\cdot 2\cdot\cos(2k\pi/16)\right]\\
&\quad\cdot 2\cdot\sin(2k\pi/32)+u^{(1)}(0)\cdot 2\sin(2k\pi/16)\cdot\sin(2k\pi/32)
\end{aligned}\tag{25}$$

$$\begin{aligned}
&X_{32}(16-k)\{x(i)\}\\
&=\left[U_8(8-k)\{x(4i)\}\right.\\
&\qquad\left.+U_8(8-k)\left\{v^{(0)}(i)\right\}\cdot 2\cdot\cos(2k\pi/16)\right]\\
&\quad+x(0)/2-v^{(0)}(0)\cdot\sin(2k\pi/16)+u^{(0)}(0)\cdot\sin(2k\pi/32)\\
&\quad-\left[U_8(8-k)\left\{u^{(0)}(2i)\right\}\right.\\
&\qquad\left.+U_8(8-k)\left\{u^{(1)}(i)\right\}\cdot 2\cdot\cos(2k\pi/16)\right]\\
&\quad\cdot 2\cdot\cos(2k\pi/32)+u^{(1)}(0)\cdot 2\sin(2k\pi/16)\cdot\cos(2k\pi/32)
\end{aligned}\tag{26}$$

ISSN (Print) : 2320 – 3765
ISSN (Online): 2278 – 8875

International Journal of Advanced Research in Electrical, Electronics and Instrumentation Engineering

(An ISO 3297: 2007 Certified Organization)

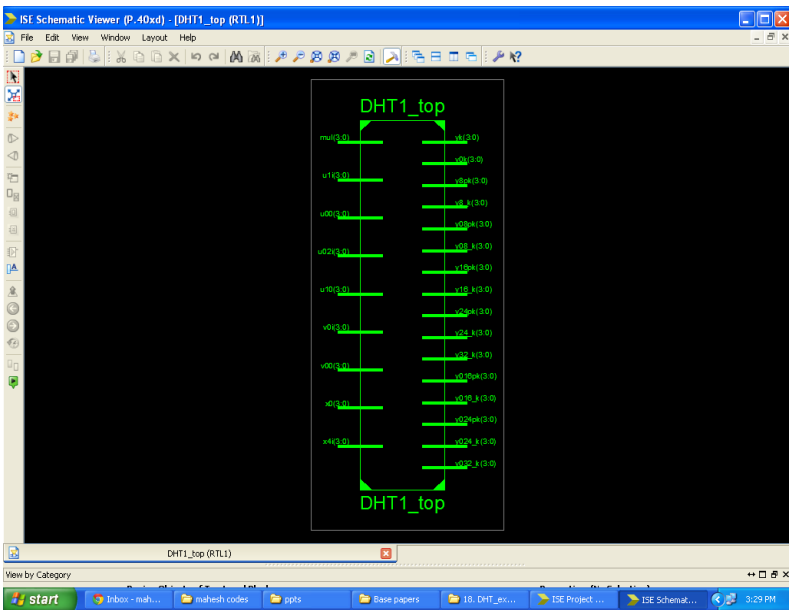
Vol. 3, Issue 10, October 2014

$$X_{32}(24-k)\{x(i)\} = \left[U_8(8-k)\{x(4i)\} - U_8(8-k)\left\{v^{(0)}(i)\right\}\cdot 2\cdot\cos(2k\pi/16)\right] + x(0)/2 + v^{(0)}(0)\cdot\sin(2k\pi/16) - u^{(0)}(0)\cdot\cos(2k\pi/32) - \left[U_8(8-k)\left\{u^{(0)}(2i)\right\} - U_8(8-k)\left\{u^{(1)}(i)\right\}\cdot 2\cdot\cos(2k\pi/16)\right]\cdot 2\cdot\sin(2k\pi/32) - u^{(1)}(0)\cdot 2\sin(2k\pi/16)\cdot\sin(2k\pi/32) \quad (27)$$

$$X_{32}(32-k)\{x(i)\} = \left[U_8(8-k)\{x(4i)\} + U_8(8-k)\left\{v^{(0)}(i)\right\}\cdot 2\cdot\cos(2k\pi/16)\right] + x(0)/2 - v^{(0)}(0)\cdot\sin(2k\pi/16) - u^{(0)}(0)\cdot\sin(2k\pi/32) + \left[U_8(8-k)\left\{u^{(0)}(2i)\right\} + U_8(8-k)\left\{u^{(1)}(i)\right\}\cdot 2\cdot\cos(2k\pi/16)\right]\cdot 2\cdot\cos(2k\pi/32) - u^{(1)}(0)\cdot 2\sin(2k\pi/16)\cdot\cos(2k\pi/32) \quad \text{for } k=1,\ldots,4. \quad (28)$$

VIII. SIMULATION RESULTS

Block diagram



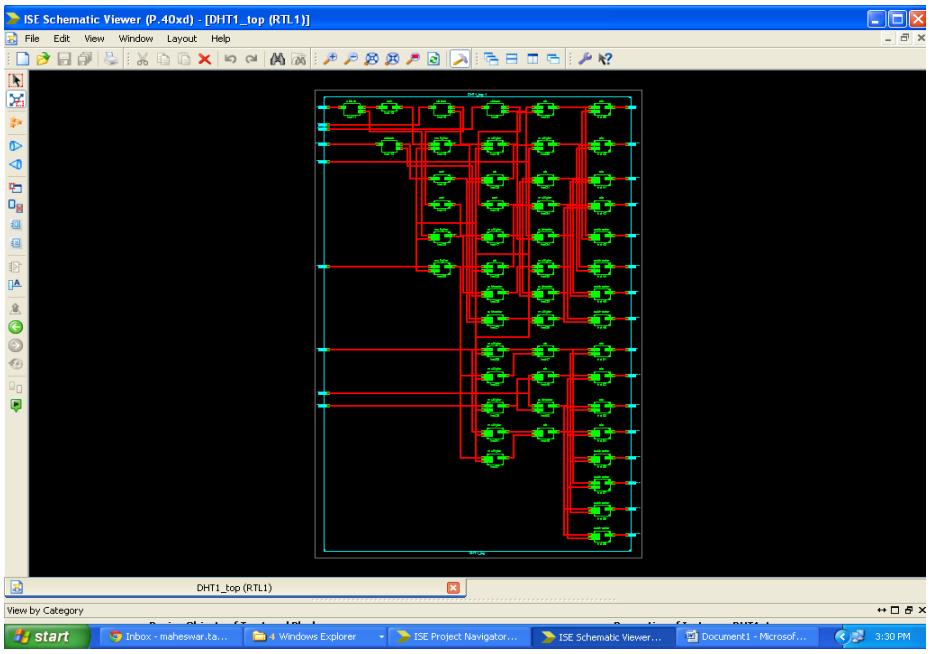
ISSN (Print) : 2320 – 3765
ISSN (Online): 2278 – 8875

International Journal of Advanced Research in Electrical, Electronics and Instrumentation Engineering

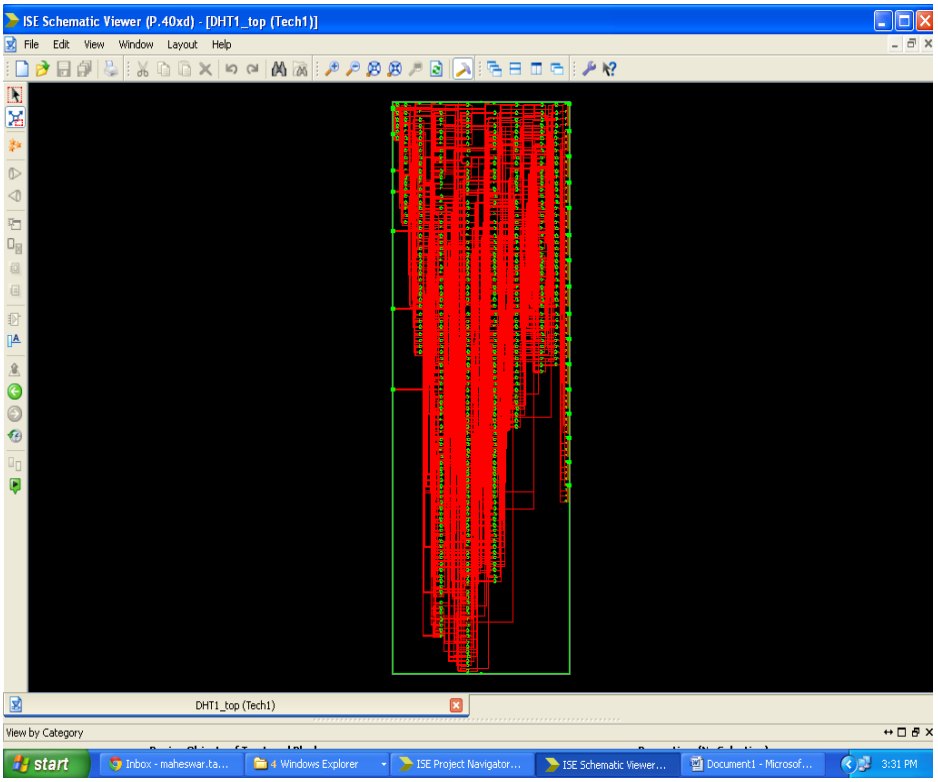
(An ISO 3297: 2007 Certified Organization)

Vol. 3, Issue 10, October 2014

RTL schematic



Technology schematic



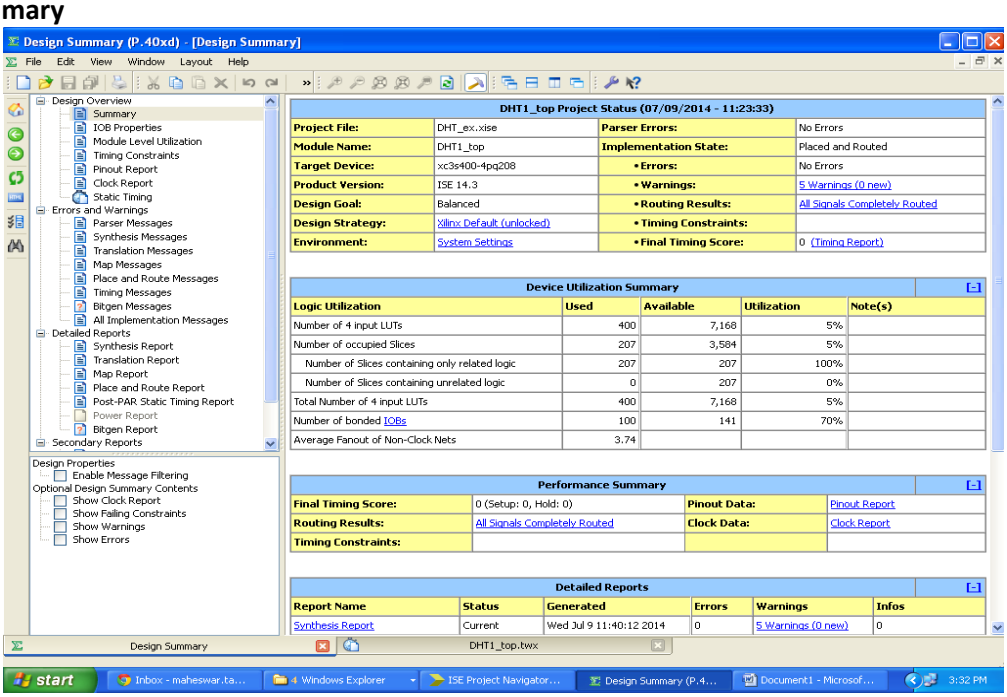
ISSN (Print) : 2320 – 3765
ISSN (Online): 2278 – 8875

International Journal of Advanced Research in Electrical, Electronics and Instrumentation Engineering

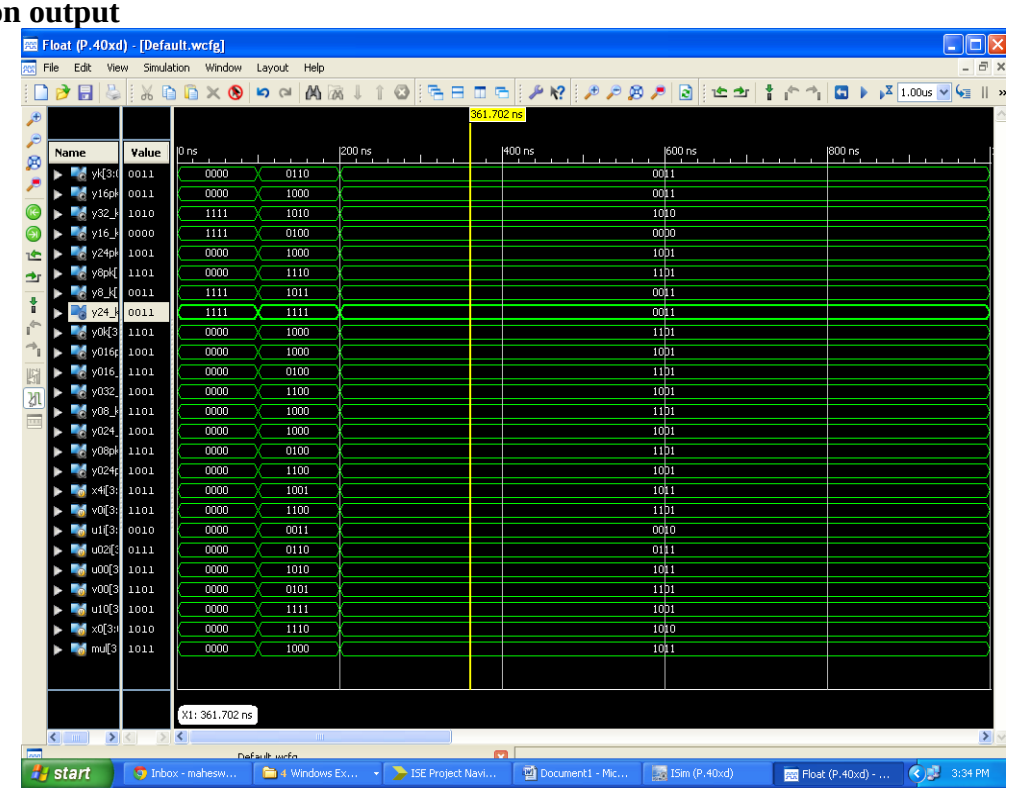
(An ISO 3297: 2007 Certified Organization)

Vol. 3, Issue 10, October 2014

Design summary



Simulation output



ISSN (Print) : 2320 – 3765
ISSN (Online): 2278 – 8875

International Journal of Advanced Research in Electrical, Electronics and Instrumentation Engineering

(An ISO 3297: 2007 Certified Organization)

Vol. 3, Issue 10, October 2014

XI. CONCLUSION

In this brief, a new highly parallel VLSI algorithm for the computation of a length-$N = 2^n$ DHT having a modular and regular structure has been presented. Moreover, this algorithm can be implemented on a highly parallel architecture having a modular and regular structure with a low hardware complexity by extensively using a sub expression sharing technique and the sharing of multipliers having the same constant.

REFERENCES

[1] R. E. Crochiere and L. R. Rabiner, Multirate Digital Signal Processing. Englewood Cliffs, NJ, USA: Prentice-Hall, 1983.
[2] Z. Wang, "Harmonic analysis with a real frequency function, I Aperiodic case, II Periodic and bounded cases, and III data sequence," Appl. Math. Comput., vol. 9, no. 1, pp. 53–73, Jul. 1981.
[3] J. Xi and J. F. Chicharo, "Computing running discrete Hartley transform and running discreteWtransforms based on the adaptive LMS algorithm," IEEE Trans. Circuits Syst. II, Analog Digit. Signal Process., vol. 44, no. 3, pp. 257–260, Mar. 1997.
[4] G. Bi, Y. Chen, and Y. Zeng, "Fast algorithms for generalized discrete Hartley transform of composite sequence length," IEEE Trans. Circuits Syst. II, Analog Digit. Signal Process., vol. 47, no. 9, pp. 893–901, Sep. 2000.
[5] S. Bouguezel, M. O. Ahmad, and M. N. S. Swamy, "New parametric discrete Fourier and Hartley transforms, and algorithms for fast computation," IEEE Trans. Circuits Syst. I, Reg. Papers, vol. 58, no. 3, pp. 562– 575, Mar. 2011.
[6] J. S. Wu, H. Z. Shu, L. Senhadji, and L. M. Luo, "Radix 3 × 3 algorithm for the 2-D discrete Hartley transform," IEEE Trans. Circuits Syst. II, Exp. Briefs, vol. 55, no. 6, pp. 566–570, Jun. 2008.
[7] S. Bouguezel, M. O. Ahmad, and M. N. S. Swamy, "A split vector-radix algorithm for the 3-D discrete Hartley transform," IEEE Trans. Circuits Syst. I, Reg. Papers, vol. 53, no. 9, pp. 1966–1976, Sep. 2006.
[8] D. F. Chiper, "Radix-2 fast algorithm for computing discrete Hartley transform of type III," IEEE Trans. Circuits Syst. II, Exp. Briefs, vol. 59, no. 5, pp. 297–301, May 2012.
[9] H. Z. Shu, J. S. Wu, C. F. Yang, and L. Senhadji, "Fast radix-3 algorithm for the generalized discrete Hartley transform of type II," IEEE Signal Process. Lett., vol. 19, no. 6, pp. 348–351, Jun. 2012.
[10] D. F. Chiper, "Fast radix-2 algorithm for the discrete Hartley transform of type II," IEEE Signal Process. Lett., vol. 18, no. 11, pp. 687–689, Nov. 2011.
[11] H. V. Sorensen, D. L. Jones, C. S. Burrus, and M. T. Heideman, "On computing the discrete Hartley transform," IEEE Trans. Acoust., Speech, Signal Process., vol. ASSP-33, no. 5, pp. 1231–1238, Oct. 1985.
[12] H. S. Malvar, Signal Processing With Lapped Transforms. Norwood, MA, USA: Artech House, 1992.
[13] G. Bi, "New split-radix algorithm for the discrete Hartley transform," IEEE Trans. Signal Process., vol. 45, no. 2, pp. 297–302, Feb. 1997.
[14] C. W. Kok, "Fast algorithm for computing discrete cosine transform," IEEE Trans. Signal Process., vol. 45, no. 3, pp. 757–760, Mar. 1997.
[15] P. K. Meher, J. C. Patra, and M. N. S. Swamy, "High throughput memorybased architecture for DHT using a new convolutional formulation," IEEE Trans. Circuits Syst. II, Exp. Briefs, vol. 54, no. 7, pp. 606–610, Jul. 2007.
[16] P. K. Meher, T. Srikanthan, and J. C. Patra, "Scalable and modular memory-based systolic array architectures for discrete Hartley transform," IEEE Trans. Circuits Syst. I, Reg. Papers, vol. 53, no. 5, pp. 1065–1077, May 2006.
[17] D. F. Chiper, M. N. S. Swamy, and M. O. Ahmad, "An efficient systolic array algorithm for the VLSI implementation of prime-length DHT," in Proc. ISSCS, Jul. 2005, vol. 1, pp. 167–169.
[18] S. B. Pan and R. H. Park, "Unified systolic array for computatiuon of DCT/DST/DHT," IEEE Trans. Circuits Syst. Video Technol., vol. 7, no. 2, pp. 413–419, Apr. 1997.
[19] A. Erickson and B. S. Fagin, "Calculating the FHT in hardware," IEEE Trans. Signal Process., vol. 40, no. 6, pp. 1341–1353, Jun. 1992.
[20] H.-R. Lee, C.-W. Jen, and C.-M. Liu, "On the design automation of the memory-based VLSI architectures for FIR filters," IEEE Trans. Consum. Electron., vol. 39, no. 3, pp. 619–629, Jun. 1993.
[21] J.-I. Guo, C.-M. Liu, and C.-W. Jen, "The efficient memory-based VLSI array design for DFT and DCT," IEEE Trans. Circuits Syst. II, Analog Digit. Signal Process., vol. 39, no. 10, pp. 723–733, Oct. 1992.
[22] P. K. Meher, "LUT optimization for memory-based computation," IEEE Trans. Circuits Syst. II, Exp. Briefs, vol. 57, no. 4, pp. 285–289, Apr. 2010.
[23] P. K. Meher, "New approach to look-up table design and memory-based realization of FIR digital filters," IEEE Trans. Circuits Syst. I, Reg. Papers, vol. 57, no. 3, pp. 592–603, Mar. 2010.
[24] D. F. Chiper and P. Ungureanu, "Novel VLSI algorithm and architecture with good quantization properties for a high-throughput area efficient systolic array implementation of DCT," EURASIP J. Adv. Signal Process., vol. 2011, no. 1, pp. 1–14, Jan. 2011.
[25] R. I. Hartley, "Subexpression sharing in filters using canonic signed digit multipliers," IEEE Trans. Circuits Syst. II, Analog Digit. Signal Process., vol. 43, no. 10, pp. 677–688, Oct. 1996.
[26] K. K. Parhi, VLSI Digital Signal Processing. New York, NY, USA: Wiley, 1999.