Journal of Scientific & Industrial Research
Vol. 81, July 2022, pp. 748-753

Parallel Hardware Implementation of Walsh Hadamard Transform

Pulak Mazumder[1]*, Soumyadeep Chandra[2], Sekhar Rana[3], Mainak Mukhopadhyay[4] & Mrinal Kanti Naskar[2]

[1]Department of ECE, Regent Education and Research Foundation, Kolkata 700 121, West Bengal, India

[2]Department of ETCE, Jadavpur University, Kolkata 700 032, West Bengal, India

[3]Department of Electronics and Communication Engineering, MCKV Institute of Engineering, Howrah 711 104, West Bengal, India

[4]Department of ECE, Birla Institute of Technology, Mesra, Ranchi 835 215, Jharkhand, India

Received 18 February 2021; revised 15 June 2022; accepted 20 June 2022

The Walsh Hadamard Transform is a powerful notion in digital signal processing. This paper explains the construction of parallel hardware architecture using the mathematical concept of Kronecker product based approach to Walsh Hadamard Transform and its simulation using Verilog. This architecture is simulated here using Field Programmable Gate Array (FPGA) technology in Verilog Spartan 3e platform. Furthermore, this paper illustrates the fast algorithm and parallel computational result of both one-dimensional and two-dimensional transforms using the Kronecker product.This algorithm can be used to implement a systolic array based dedicated hardware for computation of the transform. Our proposed hardware design for the Walsh Hadamard Transform will be used in various digital signal processing applications. The systematic derivation of parallel architecture design using the concept of Kronecker product and stride permutation would depict the real time processing rather than conventional way and reducing time complexity using minimal resources is a challenging task.

Keywords: FPGA, Kronecker product, Systolic architecture, Verilog

Introduction

The Walsh Hadamard Transform (WHT) is a mathematical construct that finds wide applications in the fields of digital signal processing, data compression, and encryption.[1] It also finds application in quantum computer information processing and it is more often called Hadamard gate in this context. The transform is particularly useful in feature extraction for pattern recognition and digital image processing because of its easy implementation using simple arithmetic stages.[2] Also, the binary nature of the Walsh functions and the Hadamard matrix allow easy semiconductor-based computer implementation.[3]

The usage of the Kronecker product allows the de-implementation of WHT using an algorithm that is both recursive and parallel. For a one-dimensional input of size 2α, the algorithm completes in α clock cycles. Moreover, for a two-dimensional input of size $(2\alpha \times 2\alpha)$, the same is complete in 2α clock cycles. Thus, using this approach can significantly reduce the number of clock cycles.[4]

Kronecker product allows the decomposition of the WHT matrix into simpler arithmetic stages comprising of homogeneous recursive array block. It thus provides a parallel computation of systolic based array implementation of WHT for synchronous evaluation.[5] Decomposing the one-dimensional input into pairs and applying them to the arithmatic stages allows for parallel execution.[6] The output obtained is stride-permuted and applied back to the same arithmetic stages, continuing the execution recursively. For a two-dimensional input, we design an algorithm that works on the column-major representation of the input.[7] This representation is one-dimensional, allowing the computation similar to the above. As discussed in our previous work[8]**,** the step-by-step development beginning from Granata's paper.[4] from the theoretical approach of expressing the DSP algorithm using the Kronecker product gives insight into developing the parallel hardware architecture.

Although the previous work had focused on modeling the FFT algorithm[3,9], which threw light onto the plethora of algorithms involving recursion that can have a similar implementation, the major constraint remains the implementation of this derived architecture for practical. This had been greatly aided

*Authors for Correspondence
E-mail: pulak.mazumder@gmail.com

by Field-programmable Gate arrays which have experienced pleasant favoritism from researchers with laboratory constraints. Our case resolves around a simple purpose. We implemented a very well-known transform technique from the area of DSP using the Kronecker product, and as a result, we were able to achieve a parallelism technique for the algorithms. This has helped us to propose a new architecture for an Application Specific Integrated Circuit (ASIC) dedicated to this purpose. Our proposed WHT algorithm's computational result is faster than basic WHT and implemented parallel too. As a result, our proposed architecture is faster than our previous work[8] and reduces the time and space complexity.

This paper is constructed as follows. The Materials and Method section gives a brief description of the basic concept of the Kronecker product and its properties. We use these properties to develop formulae for evaluating WHT of the 1D and 2D inputs recursive and parallel in sub-sections respectively. Then in next sub-section, we propose our systolic array-based WHT architecture based on these recursive formulae. In the following sub-section, the paper also contains algorithms for the architecture proposed. And last, the paper contains a section dedicated to the mathematical and time complexity analysis of the algorithms in Results and Discussion section.

Materials and Methods

Basic Kronecker Properties

Let $A_{n1,n2}$, and $B_{m1,m2}$ be two arbitrary matrices of dimension n1 by n2 and m1 by m2, respectively. Let F be a field such as R or C. For any matrices, $A = [a_{i,j}] \in F_{m\times n}\ and\ B \in F_{p\times q}$, their Kronecker product[4] (i.e., the direct product or tensor product) denoted by $C = A \otimes B$, is defined as

$A \otimes B = [a_{i,j}B]$of dimension $(n1 \times m1) \times (n2 \times m2)$

$$C = \left(A_{n1,n2} \otimes B_{m1,m2}\right) \qquad \dots (1)$$

$Thus, A \otimes B \in F_{(mp)\times(nq)}$

Mathematical Model: 1-D Transform

The WHT performs an orthogonal, symmetric, involution, linear operation on a set X_N of $N = (2\alpha)$ real numbers. For 1-D WHTs, the set of numbers is represented as a vector $X_{N\times1}$, and the transform[10] is given by:

$$Y_{N\times1} = W_N \cdot X_{N\times1} \qquad \dots (2)$$

W_N, the Hadamard matrix for $N = (2\alpha)$, can be recursively defined as follows:

$$W_N = \begin{bmatrix} W_{\frac{N}{2}} & W_{\frac{N}{2}} \\ W_{\frac{N}{2}} & -W_{\frac{N}{2}} \end{bmatrix} \qquad \dots (3)$$

Alternatively, $W_{2\alpha}$ can be defined using the Kronecker product as follows,

$$W_{2^\alpha} = W_2 \otimes W_{2^{\alpha-1}} \qquad \dots (4)$$

By solving the recurrence for $W_{2\alpha}$ using the method of substitution, we can get,

$$W_{2^\alpha} = (W_2 \otimes W_2 \otimes W_2 \dots \otimes W_2 \otimes W_2) \qquad \dots (5)$$

Now the Product rule implies:

$$A_{N_1} \otimes \dots \otimes A_{N_t} = \prod_{k=1}^{t}\left(I_{N_{k-1}} \otimes A_{N_k} \otimes I_{\frac{N}{N_k}}\right) \qquad \dots (6)$$

The above generalized product rule can be modified specifically for the WHT as follows:

$$W_{2^\alpha} = \prod_{i=0}^{\alpha-1}\left(I_{2^i} \otimes W_2 \otimes I_{2^{\alpha-i-1}}\right) \qquad \dots (7)$$

Further, using the commutation theorem, product rule, and the properties of stride permutationmatrices, we can obtain:

$$W_{2^\alpha} = \prod_{i=0}^{\alpha-1} P_{2^\alpha,2}(I_{2^{\alpha-1}} \otimes W_2) \qquad \dots (8)$$

Therefore, putting Eq. (8) in the definition of WHT stated in Eq. (2), we get

$$Y_{N\times1} = \prod_{i=0}^{\alpha-1} P_{2^\alpha,2}(I_{2^{\alpha-1}} \otimes W_2)X_{N\times1} \qquad \dots (9)$$

The above formulation[8] gives an efficient algorithm for computing the Hadamard matrix where $(I_{2^{\alpha-1}} \otimes W_2)X_{N\times1}$ is a parallel operation and $P_{2\alpha,2}$ is a 2α- point stride 2 permutation matrix.

However, using the recursive Kronecker Product property of W_N, as shown in Eq. (5), allows parallel execution of lower-order 2-point WHT transform block for generating higher-order WHT transformation on input X_N.

4-point Transform:

The block diagram of section-wise implementation of Fast 4-point WHT is given in Fig. 1.

A 4 point WHT transform matrix, using the Kronecker product rule, is represented by:

$$W_4 = \begin{bmatrix} 1 & 1 & 1 & 1 \\ 1 & -1 & 1 & -1 \\ 1 & 1 & -1 & -1 \\ 1 & -1 & -1 & 1 \end{bmatrix};\ W_2 = \begin{bmatrix} 1 & 1 \\ 1 & -1 \end{bmatrix}$$

$$or, W_4 = W_2 \otimes W_2 = \begin{bmatrix} W_2 & W_2 \\ W_2 & -W_2 \end{bmatrix}; \qquad \dots (10)$$

Let us consider a matrix A_n of dimension $n \times n$ such that:

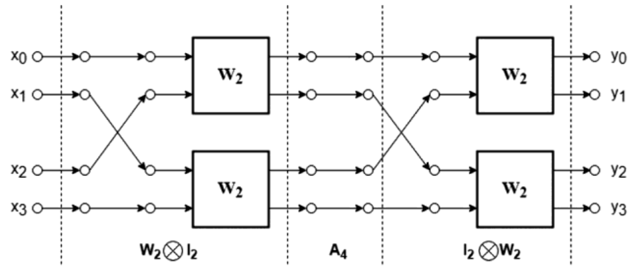

Fig. 1 — Block Diagram of Section-wise Implementation of Fast 4-point WHT

$$W_4 = \begin{bmatrix} W_2 & A_2W_2 \\ W_2 & -A_2W_2 \end{bmatrix} = \begin{bmatrix} I_2 & A_2 \\ I_2 & -A_2 \end{bmatrix}\begin{bmatrix} W_2 & 0 \\ 0 & W_2 \end{bmatrix};$$

$$or, W_4 = \begin{bmatrix} I_2 & I_2 \\ I_2 & -I_2 \end{bmatrix}\begin{bmatrix} I_2 & 0 \\ 0 & A_2 \end{bmatrix}\begin{bmatrix} W_2 & 0 \\ 0 & W_2 \end{bmatrix}; \quad \dots (11)$$

Thus, on comparing Eq. (10) with equation Eq. (11), we get that W_4 can be represented in the Kronecker product as:

$$W_4 = (W_2 \otimes I_2)\begin{bmatrix} I_2 & 0 \\ 0 & A_2 \end{bmatrix}(I_2 \otimes W_2)$$

$$where, \begin{bmatrix} I_2 & 0 \\ 0 & A_2 \end{bmatrix} = A_4 = I_4 = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

Thus, a 4-point WHT can be represented as:

$$W_4 = (W_2 \otimes I_2)(I_4)(I_2 \otimes W_2) \quad \dots (12)$$

8-point Transform

Similarly, from Fig. 2, using the recursive nature of the Kronecker product, the equation representing an 8-point WHT transform is:

$$W_8 = \begin{bmatrix} I_4 & I_4 \\ I_4 & -I_4 \end{bmatrix}\begin{bmatrix} I_4 & 0 \\ 0 & A_4 \end{bmatrix}\begin{bmatrix} W_4 & 0 \\ 0 & W_4 \end{bmatrix};$$

$$W_8 = (W_2 \otimes I_4)\begin{bmatrix} I_4 & 0 \\ 0 & A_4 \end{bmatrix}(I_2 \otimes W_4) \quad \dots (13)$$

$$\text{where,} \begin{bmatrix} I_4 & 0 \\ 0 & A_4 \end{bmatrix} = A_8 = I_8$$

Thus, the reduced equation is:

$$W_8 = (W_2 \otimes I_4)(I_8)(I_2 \otimes W_4) \quad \dots (14)$$

The Generalised Equation for 1-D Fast WHT

From the above examples, we can generalise the equation for representing a 1D N-point WHT as:

$$W_N = \left(W_2 \otimes I_{\frac{N}{2}}\right)\begin{bmatrix} I_{\frac{N}{2}} & 0 \\ 0 & A_{\frac{N}{2}} \end{bmatrix}\left(I_2 \otimes W_{\frac{N}{2}}\right)$$

$$W_N = \left(W_2 \otimes I_{\frac{N}{2}}\right)(A_N)\left(I_2 \otimes W_{\frac{N}{2}}\right) \quad \dots (15)$$

$$where\ A_N = diag(1) = I_N$$

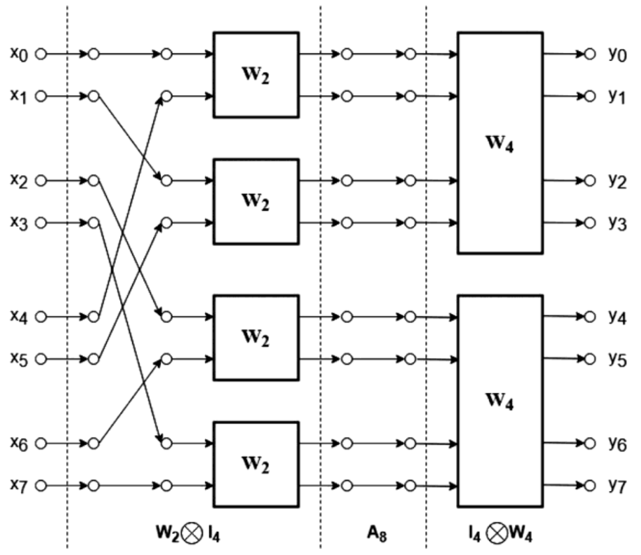

Fig. 2 — Block Diagram for Section-wise Implementation of Fast 8-point WHT

Mathematical Model: 2-D Transformation

The WHT of a square matrix X_N of order N is given by

$$Y_{N\times 1} = W_N \times X_N \times W_N^T \quad \dots (16)$$

Now, we can observe from Eq. (3) and Eq. (8), that the transpose of W_N is the same matrix as W_N^T

$$W_N = W_N^T = \begin{bmatrix} W_{\frac{N}{2}} & W_{\frac{N}{2}} \\ W_{\frac{N}{2}} & -W_{\frac{N}{2}} \end{bmatrix} \quad \dots (17)$$

$$or, W_N = W_N^T = \prod_{i=0}^{\alpha-1} P_{2^\alpha,2}(I_{2^{\alpha-1}} \otimes W_2) \quad \dots (18)$$

If we represent the matrix $X_N$as a column matrix $X_C$of order $2^{2\alpha} \times 1$, we can rewrite Eq. (16) for obtaining 2-D WHT as follows.

$$Y_C = (W_N \otimes W_N^T) \times X_C \quad \dots (19)$$

where, Y_C is a column matrix representation for Y_N.

Now for $N = 2^\alpha$, in Eq. (19), we can write:

$$W_N \otimes W_N^T = W_{2^\alpha} \otimes W_{2^\alpha} = W_{2^{2\alpha}} = W_{N^2} \quad \dots (20)$$

Therefore, we can present the following formulation for the computation of 2-D WHT using Eq. (18) and (19).

$$Y_C = \prod_{i=0}^{2^{\alpha-1}} P_{2^{2\alpha},2}(I_{2^{2\alpha-1}} \otimes W_2)X_C \quad \dots (21)$$

The above formulation gives an efficient algorithm for computing the Hadamard matrix where $(I_{2^{2\alpha-1}} \otimes$ $W2XC$ is a parallel operationand P2α,2 is a 2α- point stride 2 permutation matrix.

However, using properties of Kronecker product, we know from Eq. (15) that in 1D transform W_N *or* W_{2^α} can be represented as:

$$W_N = W_{2^\alpha} = \left(W_2 \otimes I_{\frac{N}{2}}\right)(A_N)\left(I_2 \otimes W_{\frac{N}{2}}\right)$$

Substituting this in Eq. (20), where $N = 2^\alpha$, we get:

$$W_{N^2} = \left(W_2 \otimes I_{\frac{N^2}{2}}\right)(A_{N^2})\left(I_2 \otimes W_{\frac{N^2}{2}}\right) \quad \dots (22)$$

Proposed Architecture

The Generalised WHT equation has two parts:

1. The parallel computation of inputs to recursive W_2 blocks.
2. The multiplication of outputs generated from the W_2 transform block in part 1 generates output.

We present a hardware architecture where each iteration in Eq. (15) is completed in one circuit clock cycle. In part (1), addition and subtraction for computation of recursive 2-point WHT transform blocks are done in a positive half cycle.[11] In part(2), the transformed outputs are multiplied to generate output Y_N for N-point W_N transform in the corresponding negative half cycle.

This hardware implementation requires the following devices: N Arithmetic Units (AU) that consist of an adder and a subtractor for part (1) operation and N/2 Multipliers for part (2) operation. The outputs of multipliers, i.e., Q1 to Q7, are again loaded into X1 to X7 and fed back to the AUs in the next clock cycle for computation of the next iteration in Eq. (15). In Fig. 3 flowchart representation of the algorithm used for computation of (a) 1-D and (b) 2-D Walsh Hadamard Transform is depicted.

Algorithms

Algorithm for 1D transform

Algorithm 1: Algorithm describing the computation of 1D WHT Transform

Input: X_n: N point (2^α) input to 1D Transform
Output: Y_n: N point output of 1D Transform

```
1 $X_n$ ← input of size $n$
2 $for\ \alpha \leftarrow 1\ times\ do$
3 $For\ every\ rising\ edge\ of\ the\ circuit\ clock$
4 $Y_n^T \leftarrow AU(X_n);$
5 $For\ the\ corresponding\ falling\ edge\ of\ the\ clock$
6 $X_n \leftarrow X_n \leftarrow Multipliers(Y_n^T);$
7 end
```

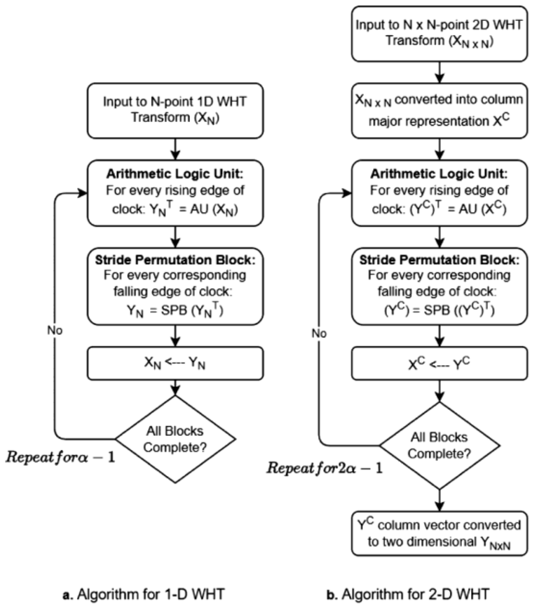
Input to N-point 1D WHT Transform (X_N)
Arithmetic Logic Unit: For every rising edge of clock: Y_N^T = AU (X_N)
Stride Permutation Block: For every corresponding falling edge of clock: Y_N = SPB (Y_N^T)
X_N <--- Y_N
All Blocks Complete?
No
Repeat for α − 1
a. Algorithm for 1-D WHT
Input to N x N-point 2D WHT Transform (X_N x N)
X_N x N converted into column major representation X^C
Arithmetic Logic Unit: For every rising edge of clock: (Y^C)^T = AU (X^C)
Stride Permutation Block: For every corresponding falling edge of clock: (Y^C) = SPB ((Y^C)^T)
X^C <--- Y^C
All Blocks Complete?
No
Repeat for 2α − 1
Y^C column vector converted to two dimensional Y_NxN
b. Algorithm for 2-D WHT

Fig. 3 — Flowchart representation of the algorithm used for computation of (a) 1-D and (b) 2-D Walsh Hadamard Transform

Algorithm for 2D transform

Algorithm 2: Algorithm describing the computation of 2D WHT Transform

Input: $X_{n\times n}$: N × N point input to 2D Transform
Output: $Y_{n\times n}$: N × N point output to 2D Transform

```
1 $X_{n\times n}$ ← input of size $n \times n$
2 Convert $X_{n\times n}$ into a column major representation $X_C$
3 $for\ 2\alpha \leftarrow 1\ times\ do$
4 $For\ every\ rising\ edge\ of\ the\ circuit\ clock$
5 $Y_C^T \leftarrow AU(X_C);$
6 $For\ the\ corresponding\ falling\ edge\ of\ the\ clock$
7 $X_C \leftarrow X_C \leftarrow Multipliers(Y_C^T);$
8 end
9 Convert column matrix $Y_C\ to\ 2D\ matrix\ Y_{n\times n}$
```

Results and Discussion

The Generalized Recursive Equation for Calculation of Operations in Fast WHT

S_N is the number of additions required for fast implementation of N-point WHT.

$$S_N = \left(\frac{N}{2} \times 2\right) + \left(2 \times S_{\frac{N}{2}}\right) \quad \dots (23)$$

P_N is the number of multiplications required for fast implementation of N-point WHT.

$$P_N = \left(\frac{N}{2}\right) + \left(2 \times P_{\frac{N}{2}}\right) \quad \dots (24)$$

The comparison based on number of calculation is represented in Table 1.

Table 1 — Number of Calculations

Comparison of operations of WHT

N	Previous work[14]			Proposed			Saved (%)
	P_N	S_N	Net	P_N	S_N	Net	Net
4-point WHT	6	10	16	4	8	12	25
8-point WHT	18	26	44	12	24	36	18.18
16-point WHT	42	68	110	38	64	102	7.27

Time Complexity Analysis

The Computational Complexity of Previous Works on WHT

The Walsh Hadamard Transform can be regarded as built out of multidimensional Discrete Fourier Transforms (DFTs) of size $2 \times 2 \times \ldots \times 2$. The Walsh Hadamard Matrix W_N is a $N \times N$ matrix where $N = 2^m$ real numbers.

The naive implementation of WHT of order $N = 2^m$ would have a computational complexity of $\mathcal{O}(N^2)$.[10] However, even using the fast Hadamard transform algorithm[12,13], the Hadamard transform can be computed in $\mathcal{O}(N \log_2 N)$.

1-D Transform

The expression for the time complexity can be obtained from the algorithm itself. Lines 4 and 6 of algorithm 1 get executed in 1 clock cycle, which we can take as a unit of time, say t. Therefore, the total time for computation:

$$T = \big(1 + (\alpha - 1)\big)t = \alpha t \qquad \ldots (25)$$

As $\alpha = \log_2 N$, the complexity of algorithm 1 is given by $\mathcal{O}(\log_2 N)$.

2-D Transform

Similarly, using the algorithm for 2D transform the total time for computation.

$$T = \big(1 + (2\alpha - 1)\big)t = 2\alpha t = (2\log_2 N)t \qquad \ldots (26)$$

As the input is a $N \times N$, total input size, $M = N \times N = N^2$. Therefore,

$$T = (2\log_2 N)t = \left(2\log_2 M^{\frac{1}{2}}\right)t = (\log_2 M)t \qquad \ldots (27)$$

Thus, the asymptotic representation for the time complexity of algorithm 2, can be given by $\mathcal{O}(\log_2 M)$.

Hardware Implementation

The proposed architecture has been verified using Verilog and tested using the 'Spartan 3e development kit'. The following diagrammatic schematic architecture shown in Figs 4 and 5 is implemented on Verilog to generate the following statistics. The

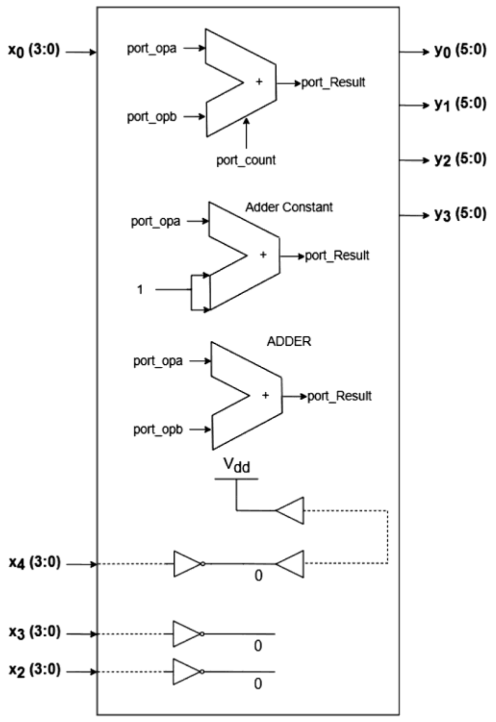

Fig. 4 — Schematic architecture of 4-point WHT

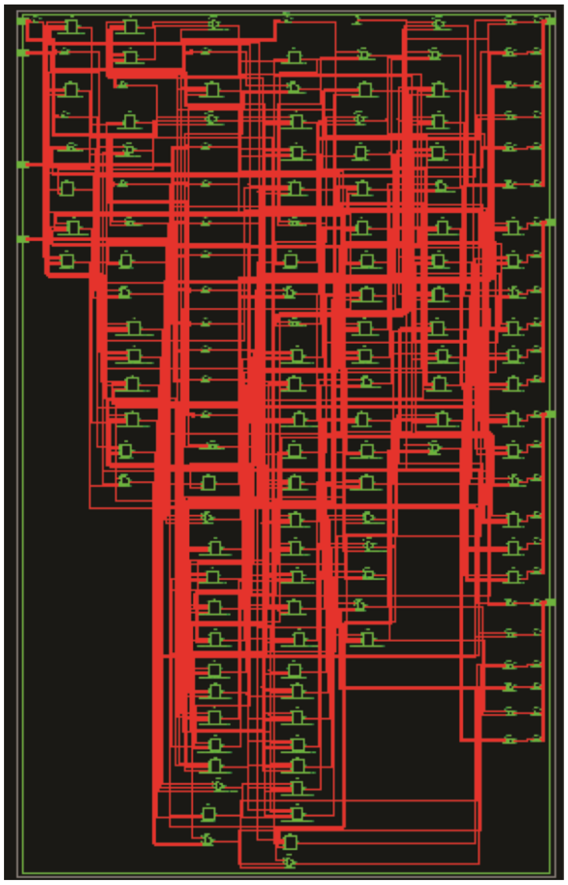

Fig. 5 — Technology schematic view of WHT implementation

Table 2 — HDL Synthesis Report

Macro Statistics

Components	Number
4-bit Adder carry out	2
5-bit Adder carry out	1
6-bit Adder carry out	14
Total Adder/Subtractors	17

Table 3 — Device Utilisation Summary

Logic Utilization	Used	Available	Utilisation
Number of Slices	45	960	4%
Number of 4 input LUTs	81	1920	4%
Number of bonded IOBs	40	66	60%

Table 4 — Time taken for computation of WHT with a different sequence length

Sequence length	Proposed
4-point WHT	14.508ns (12.261ns logic, 2.247ns route, 84.5% logic, 15.5% route)
8-point WHT	14.666ns (11.184ns logic, 3.482ns route, 76.3% logic, 23.7% route)

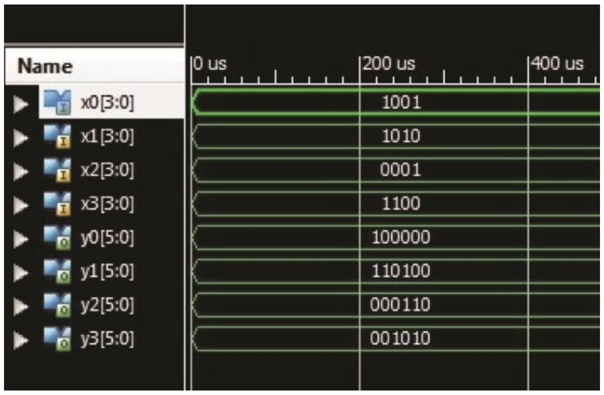

Fig. 6 — Timing diagram of 4 point WHT hardware implementation

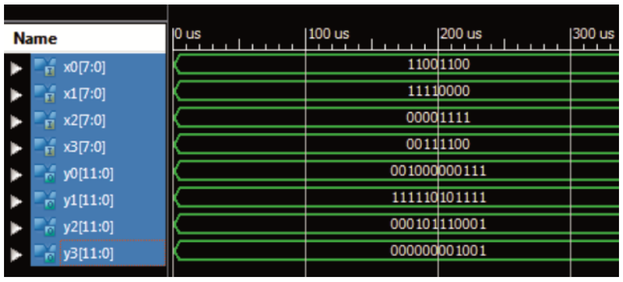

Fig. 7 — Timing diagram of 8 point WHT hardware implementation

complete HDL Synthesis Report Statistics is provided in Table 2. The Devise Utilisation Report is shown in Table 3.

The computational results are formulated in Figs 6 and 7. The corresponding runtime of the proposed WHTs are shown in Table 4.

Conclusions

The mathematical model developed to design the proposed architecture depicts a new dimension for Walsh Hadamard Transform. The dimensional transformation using the Kronecker product for fast implementation can give an alternate substitution in image processing. Walsh-Hadamard is a combination of two algorithms and it is an efficient method of image compression and signal filtering. It is because these algorithms work on the matrix basis. Since signals and images are a combination of matrices, this method finds it easy to handle them efficiently and fast. The results obtained from the simulation study have been quite promising. The simulation results using Verilog clearly confirms on the efficiency of the proposed method as compared with conventional technique. The next step would be to implement a higher-order Walsh Hadamard transform tested through the FPGA development kit.

References

1 Crochiere R E & RabinerL R, *Multirate Digital Signal Processing, Englewood Cliffs* (NJ Prentice-Hall) 1983.

2 Wang Z, Harmonic analysis with a real frequency function, I Aperiodic case, II Periodic and bounded cases and III Data sequence, *Appl Math Comput*, **9** (1981) 53–73.

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

4 Granata J, Conner M & Tolimieri R, Recursive fast algorithm and the role of the tensor product, *IEEE Trans Signal Process*, **40(12)** (1992) 2921–2930.

5 Bi G & Chen Y Q, Fast DHT Algorithms for Length N = q × 2m, IEEE Trans. on Signal Processing, **47**(1999) 900–903.

6 Tolimieri R, An M & Lu C, *Algorithms for Discrete Fourier Transform and Convolution* (Springer-Verlag) 1989.

7 Johnson J R, Johnson R W, Rodriguez D & Tolimieri R, A methodology for designing, modifying, and implementing Fourier transform algorithms on various architectures, *Circuits Syst Signal Pro*, **9(4)** (1990) 449–500.

8 Mazumder P, Middya R & Naskar M K, Hardware implementation of fast recursive walsh-hadamard transform, *Int J Comput Sci Eng*, **7(1)** (2019) 28–32.

9 Milder P A, Franchetti F, Hoe, J C & Püschel M, *Discrete Fourier Transform Compiler: From Mathematical Representation to Efficient Hardware*, CSSI Technical Report #CSSI-07-01, (Carnegie Mellon University) 2007

10 Chiper D F, Radix-2 fast algorithm for computing discrete hartley transform of type III, *IEEE Trans Circuits Syst II: Express Br*, **59(5)** (2012) 297–301.

11 Chiper D F, A Novel VLSI DHT Algorithm for a highly modular and parallel architecture, *IEEE Trans Circuits Syst II: Express Br*, **60(5)** (2013) 282–286.

12 BracewellR N, The Fast Hartley Transform, *Proc IEEE*, **72(8)** (1984) 1010–1018.

13 Fino, Bernard J & Ralph AlgaziV, Unified matrix treatment of the fast Walsh-Hadamard transform, *IEEE Trans Comput*, **11** (1976) 1142–1146.