International Journal of Engineering Research and General Science Volume 4, Issue 2, March-April, 2016
ISSN 2091-2730

FPGA implementation of Integer DCT for HEVC

M. Murali [1], V. Sandhya [2], P. Harika[3]
[1]PG scholar in ECE, [2]Asst. Prof, [3]Asst. Prof
Department of ECE, BVC Engineering College, Odalarevu, [1]+91-9010449605
[1]murali.marlapudi@gmail.com, [2]vsandhya445@gmail.com, [3]harika.pangam@gmail.com.

Abstract--In this paper, area-efficient architectures for the implementation of integer discrete cosine transform (DCT) of different lengths to be used in High Efficiency Video Coding (HEVC) are proposed. An efficient constant matrix multiplication scheme can be used to derive parallel architectures for 1-D integer DCT of different lengths such as 4, 8, 16, and 32. Also power-efficient structures for folded and full-parallel implementations of 2-D DCT is implemented with proposed architecture. The proposed architecture with 32-point length is 29.2% and 9.2% area efficient, also results in 13.1% and 2.8% less Area-Delay product respectively when compared to basic and existing models.

Also pruning is applied to proposed architecture to improve the performance which results in 50.78% decrease in area Delay product for 32-point integer DCT.

Key Words-High Efficiency Video Coding (HEVC), integer discrete cosine transform (DCT), video coding.

I. INTRODUCTION

In digital image processing, data compression is necessary to improve efficiency in storage and transmission. Transformation is one popular technique for data compression. By first transforming correlated pixels into weakly correlated ones, and after a ranking in their energy contents, for example, and retaining only the most significant components, high compression ratio is possible [1]. Since inverse transformation is needed to reproduce the original image from the compressed data, it is important that the transform process be simple and fast. The family of orthogonal transforms [2] is well suited for this application because the inverse of an orthogonal matrix is its transpose. In real-time applications, a transform is most likely implemented using a dedicated chip. Thus the shorter bit length (integer cosine transform) ICT will lead to a simpler IC structure and shorter computation time.

II. HEVC (High Efficiency Video Coding)

The discrete cosine transform has found a wide range of applications in signal and image processing [3], [4], especially in image compression. It has become the heart of international standards in image compression such as JPEG and MPEG [5]. In some applications, the input data consists of integer vectors or integer matrices. But the output of DCT does not consist of integers. For lossless coding it would be of great interest to be able to characterize the output completely again with integers. In the JPEG-2000 proposal [6], the use of the integer DCT-II for lossless image coding is recommended. However, up to now, lossless coding schemes are hardly based on integer DCTs which have been studied in recent years [7], [8], [9], [10], [11].

HEVC is said to double the data compression ratio compared to H.264/MPEG-4 AVC at the same level of video quality. It can alternatively be used to provide substantially improved video quality at the same bit rate. The design of most video coding standards is primarily aimed at having the highest coding efficiency. Coding efficiency is the ability to encode video at the lowest possible bit rate while maintaining a certain level of video quality. There are two standard ways to measure the coding efficiency of a video coding standard, which are to use an objective metric, such as peak signal-to-noise ratio (PSNR), or to use subjective assessment of video quality. Subjective assessment of video quality is considered to be the most important way to measure a video coding standard since humans perceive video quality subjectively. The Discrete Cosine Transform (DCT) plays a vital role in video compression due to its near-optimal de-correlation efficiency and high energy compaction. HEVC supports DCT of different sizes such as 4,8,16 and 32 and it is a loss compression technique. Integer DCT integrates both loss and lossless coding scheme perfectly.

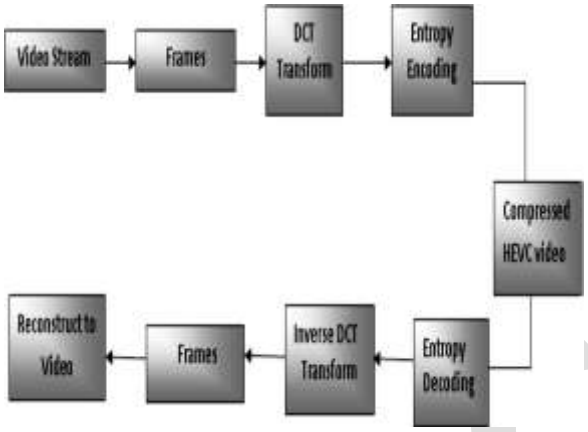

Figure1: Flow of HEVC Mechanism

III.ALGORITHM

Integer DCT can be generated by make use of Matrix. The 4-point Integer DCT generation depends on 2x2 matrices that are obtained from a 4x4 matrix as given below. The shown matrix is observable to be orthogonal symmetric. The need of symmetry in the matrix is already been stated. The 8x8 and 16x16 matrices for implementation of higher order Integer DCT operations are as shown below.

$$\begin{bmatrix} 64 & 64 & 64 & 64 \\ 83 & 36 & -36 & -83 \\ 64 & -64 & -64 & 64 \\ 36 & -83 & 83 & -36 \end{bmatrix}$$

$$\begin{bmatrix} 64 & 64 & 64 & 64 & 64 & 64 & 64 & 64 \\ 89 & 75 & 50 & 18 & -18 & -50 & -75 & -89 \\ 83 & 36 & -36 & -83 & -83 & -36 & 36 & 83 \\ 75 & -18 & -89 & -50 & 50 & 89 & 18 & -75 \\ 64 & -64 & -64 & 64 & 64 & -64 & -64 & 64 \\ 50 & -89 & 18 & 75 & -75 & -18 & 89 & -50 \\ 36 & -83 & 83 & -36 & -36 & 83 & -83 & 36 \\ 18 & -50 & 75 & -89 & 89 & -75 & 50 & -18 \end{bmatrix}$$

$$\begin{pmatrix} 64 & 64 & 64 & 64 & 64 & 64 & 64 & 64 & 64 & 64 & 64 & 64 & 64 & 64 & 64 & 64 \\ 90 & 87 & 80 & 70 & 57 & 43 & 25 & 9 & -9 & -25 & -43 & -57 & -70 & -80 & -87 & -90 \\ 89 & 75 & 50 & 18 & -18 & -50 & -75 & -89 & -89 & -75 & -50 & -18 & 18 & 50 & 75 & 89 \\ 87 & 57 & 9 & -43 & -80 & -90 & -70 & -25 & 25 & 70 & 90 & 80 & 43 & -9 & -57 & -87 \\ 83 & 36 & -36 & -83 & -83 & -36 & 36 & 83 & 83 & 36 & -36 & -83 & -83 & -36 & 36 & 83 \\ 80 & 9 & -70 & -87 & -25 & 57 & 90 & 43 & -43 & -90 & -57 & 25 & 87 & 70 & -9 & -80 \\ 75 & -18 & -89 & -50 & 50 & 89 & 18 & -75 & -75 & 18 & 89 & 50 & -50 & -89 & -18 & 75 \\ 70 & -43 & -87 & 9 & 90 & 25 & -80 & -57 & 57 & 80 & -25 & -90 & -9 & 87 & 43 & -70 \\ 64 & -64 & -64 & 64 & 64 & -64 & -64 & 64 & 64 & -64 & -64 & 64 & 64 & -64 & -64 & 64 \\ 57 & -80 & -25 & 90 & -9 & -87 & 43 & 70 & -70 & -43 & 87 & 9 & -90 & 25 & 80 & -57 \\ 50 & -89 & 18 & 75 & -75 & -18 & 89 & -50 & -50 & 89 & -18 & -75 & 75 & 18 & -89 & 50 \\ 43 & -90 & 57 & 25 & -87 & 70 & 9 & -80 & 80 & -9 & -70 & 87 & -25 & -57 & 90 & -43 \\ 36 & -83 & 83 & -36 & -36 & 83 & -83 & 36 & 36 & -83 & 83 & -36 & -36 & 83 & -83 & 36 \\ 25 & -70 & 90 & -80 & 43 & 9 & -57 & 87 & -87 & 57 & -9 & -43 & 80 & -90 & 70 & -25 \\ 18 & -50 & 75 & -89 & 89 & -75 & 50 & -18 & -18 & 50 & -75 & 89 & -89 & 75 & -50 & 18 \\ 9 & -25 & 43 & -57 & 70 & -80 & 87 & -90 & 90 & -87 & 80 & -70 & 57 & -43 & 25 & -9 \end{pmatrix}$$

The even positions and odd positions of output vector of 4-point integer DCT can be obtained by matrix multiplication with vectors [a(0) a(1)] and [b(0) b(1)] as stated below.

$$\begin{matrix} y(0) \\ y(2) \end{matrix} = \begin{bmatrix} 64 & 64 \\ 64 & -64 \end{bmatrix} \begin{matrix} a(0) \\ a(1) \end{matrix}$$

$$\begin{matrix} y(1) \\ y(3) \end{matrix} = \begin{bmatrix} 83 & 36 \\ 36 & -83 \end{bmatrix} \begin{matrix} b(0) \\ b(1) \end{matrix}$$

Here the intermediate vectors [a(0) a(1)] and [b(0) b(1)] are obtained from input vector [x(0) x(1) x(2) x(3)] as stated below

a0 <= x0 + x3; a1 <= x1 + x2;

b0 <= x0 - x3; b1 <= x1 - x2;

Finally the multiplication of the above stated matrices with intermediate vectors will result in following equations.

Y(0)=64xa(0)+64xa(1)

Y(2)=64xa(0)-64xa(1)
Y(1)=83xb(0)+36xb(1)
Y(3)=36xb(0)-83xb(1)

To solve these equations an eight 8x8 multipliers and four 16 bit adder / subtractions are needed. As already known the implementation of digital multiplier in hardware requires number of adders and consumes considerable area and consumes considerable power, eventually result in more delay causes performance degradation of overall circuit.

In-order to optimise the design metrics the implementation of multiplier must be replaced, the obvious solution is shifters. As already known a simple left shift is equivalent to multiply by two, the output vector generation equations of 4-point integer DCT stated can be obtained using shifters replaced by multipliers is given below.

Y(0)=(<<6)a(0)+(<<6)a(1)
Y(2)=(<<6)a(0)-(<<6)a(1)
Y(1)={(<<6)+(<<1)[(<<3)+1]+1}b(0)+{(<<2) [(<<3)+1]}b(1)
Y(3)={(<<2)[(<<3)+1]}b(0)-{(<<6)+(<<1)[(<<3)+1]+1}b(1)

The implementation shown may not require any multiplications but requires six, four, two and two shifters respectively of 6-bit, 4-bit, 2-bit, 1-bit wide and four 16-bit adder/subtractor. The generalised algorithm for input vector of N-bit wide for implementation of integer DCT is as given below:

$$\begin{bmatrix} y(0) \\ y(2) \\ . \\ . \\ . \\ y(N-4) \\ y(N-2) \end{bmatrix} = \mathbf{C}_{N/2} \begin{bmatrix} a(0) \\ a(1) \\ . \\ . \\ . \\ a(N/2-2) \\ a(N/2-1) \end{bmatrix}$$

$$\begin{bmatrix} y(1) \\ y(3) \\ . \\ . \\ . \\ y(N-3) \\ y(N-1) \end{bmatrix} = \mathbf{M}_{N/2} \begin{bmatrix} b(0) \\ b(1) \\ . \\ . \\ . \\ b(N/2-2) \\ b(N/2-1) \end{bmatrix}$$

$$a(i) = x(i) + x(N-i-1)$$

$$b(i) = x(i) - x(N-i-1)$$

$$m_{N/2}^{i,j} = c_N^{2i+1,j} \text{ for } 0 \le i, j \le N/2-1$$

IV.PROPOSEDARCHITECTURE

The proposed architecture implementation is based on shift and add/subtract design of existing model but implemented with an architecture where hardware resources are reused to miniature the hardware requirement so as to reduce design metrics of VLSI such as area, delay, power and cost. The following flow shows the implemented design of proposed algorithm for implementation of four point integer DCT.

Y(0)=(<<6)a(0)+(<<6)a(1)
Y(2)=(<<6)a(0)-(<<6)a(1)
Y(1)={(<<6)+**(<<1)[(<<3)+1]**+1}b(0)+(<<1){**(<<1)[(<<3)+1]**}b(1)
Y(3)=(<<1){**(<<1)[(<<3)+1]**}b(0)-{(<<6)+**(<<1)[(<<3)+1]**+1}b(1)

Here the digital block (<<1)[(<<3)+1] is reutilized while performing operations. Because of the reuse, algorithm requires shifter 6-bit, 3-bit and 1-bit wide respectively six, two and four. Also four 16-bit wide adders or subtractors are required.

TABLE1-Comparison of Hardware Resources Requirement

4-Point	Six-shifters	Three-shifters	Two-shifters	One-shifter
Existing	6	4	2	2
Proposed	6	2	0	4

TABLE1 above shows the Comparison between Existing and Proposed algorithms in terms of hardware resource requirement considering number of shifters required for each algorithm. The proposed architecture requires half the number of 3-bit shifters and hence area may be saved to lot extent. Implementation of hardware architecture for above algorithm will be appear as shown in the following Figure2.Architecture in Figure2 can be viewed as input addition/ subtraction unit, shifting unit and output addition/subtraction unit. Multiplication with constant 64 can be obtained as input vector left shifted by six bits and multiplication by 36 and 83 can be obtained as stated in the algorithm and hardware implementation is as shown in figure(b) below where 3-bit shifter and 1-bit shifter are reused for the calculation of both 83 and 36 multiplications. The implementation of 8, 16 and 32 point integer DCT architectures can also be done in the same way as shown in Figure2 for 4 point.Figure3 shows the implementation of hardware for constant multiplication for 8 point integer DCT implementation.

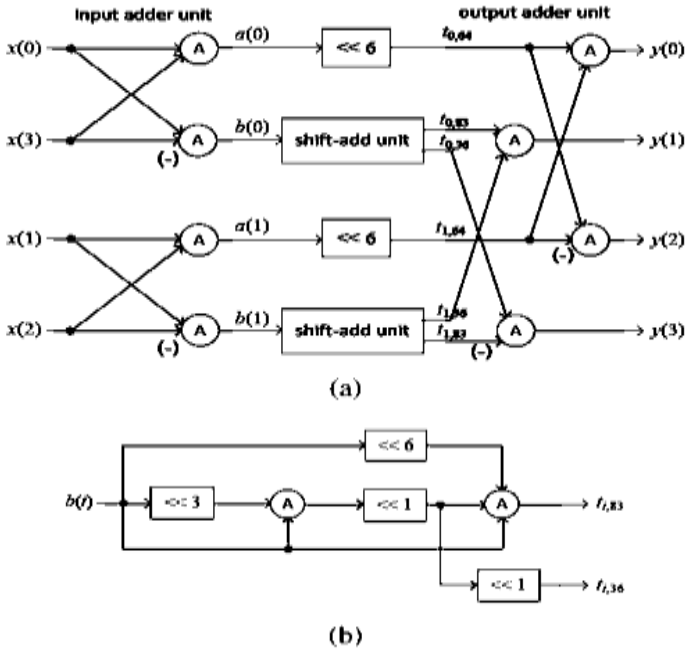

Figure2: Proposed Architecture for 4-point Integer DCT

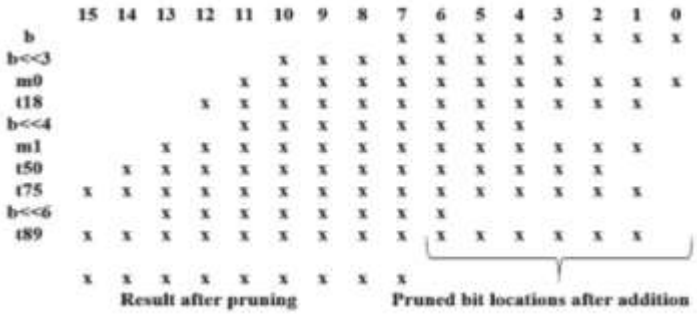

Figure3: Proposed Pruning scheme for 8-point DCT

Here one 3-bit shifter and two 1-bit shifters are reused to decrease the hardware requirement.

Image processing applications are less concern to accuracy, because of the eye persistence. Hence pruning is applied to the proposed architecture to further improve the design metrics. Figure3 shows the pruning concept applied in shift and add unit for 8-point Integer DCT, where after shift-add operation the resultant values are pruned to 9 instead of 16 to reduce the complexity of output addition unit and to reduce the transmission band width.

V.RESULTS

Integer DCT for High Efficient Video Coding implemented with varied input vectors from 4-point to 32-point and synthesized for Vertex-E FPGA. Obtained readings are tabulated in TABLE2. The implemented shift and add units with data flow from input vector to the output pad through internal hardware for calculating product of the input applied vector with Integer DCT matrix coefficients for 8-point and 16-point implementations are shown in Figure4 and Figure5 respectively. Also the implemented 2 dimensional integer DCT with dataflow is shown in figure6.The readings furnished here shows number of LUTs required, amount of Delay produced and Area-Delay Product having considerable importance in VLSI design of 4,8,16 and 32 point Integer DCT implementations using Basic algorithm, Existing algorithm and proposed algorithm.

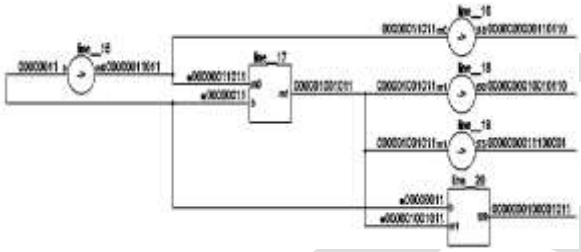

Figure4: Implemented shift and add unit for proposed 8-point architecture

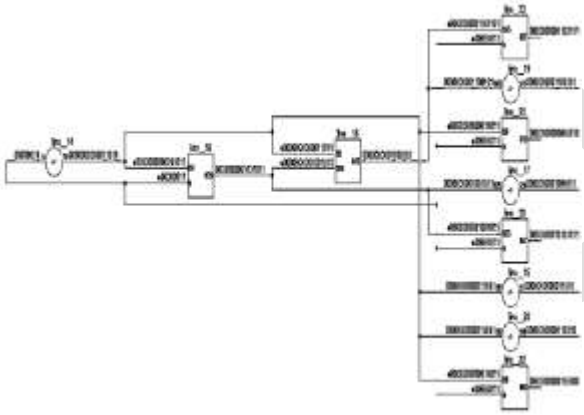

Figure5: Implemented shift and add unit for proposed 16-point architecture

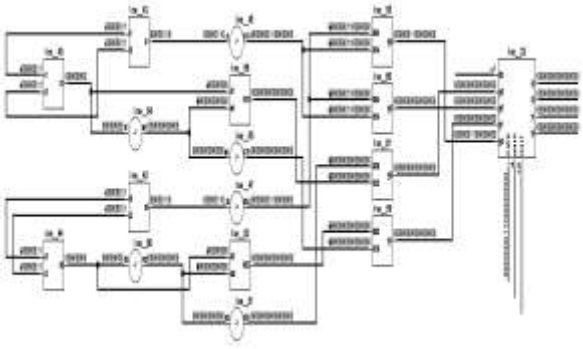

Figure6: Implemented 2D integer DCT for 4X4 input

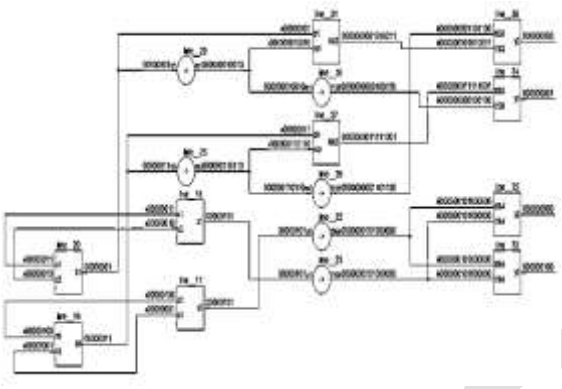

Figure7: Implemented 4-point Integer DCT with pruning.

From the readings the proposed algorithm is the best in performance in terms of area occupancy and delay produced. The proposed architecture with 32-bit length is 29.2% and 9.2% area efficient, also results in 13.1% and 2.8% less Area-Delay product respectively when compared to basic and existing models. Also implemented pruned architecture results in 50.78% decrease in area Delay product for 32-point integer DCT compared to proposed architecture. Figure8 shows the comparison of Area consumed and Delay produced among the Basic, Existing, proposed and proposed-pruned algorithms.

TABLE2 - Comparison among Basic, Existing and Proposed Architectures.

Design	N	LUT's	Delay(ns)	ADP
Basic	4	158	22.097	3.491
	8	863	24.503	21.146
	16	2927	33.937	99.333
	32	10007	47.435	474.68
Existing	4	135	20.159	2.721
	8	553	23.137	12.794
	16	2356	37.628	88.651
	32	8453	51.026	431.32
Pro-posed	4	129	20.05	2.586
	8	510	24.523	12.506
	16	1924	37.367	71.894
	32	7740	54.169	419.26
Proposed after pruning	4	114	20.05	2.285
	8	413	24.983	10.317
	16	1466	33.836	49.603
	32	5593	49.714	278.05

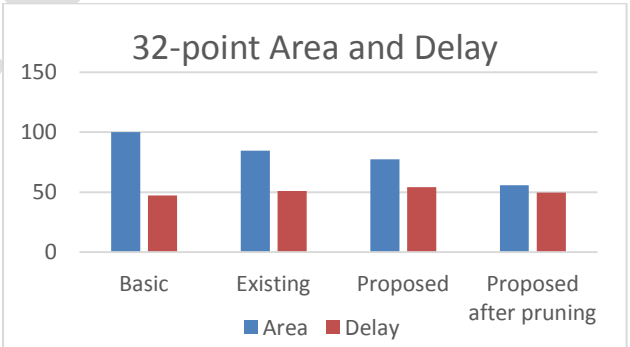

Figure8: Comparison of Area and Delay

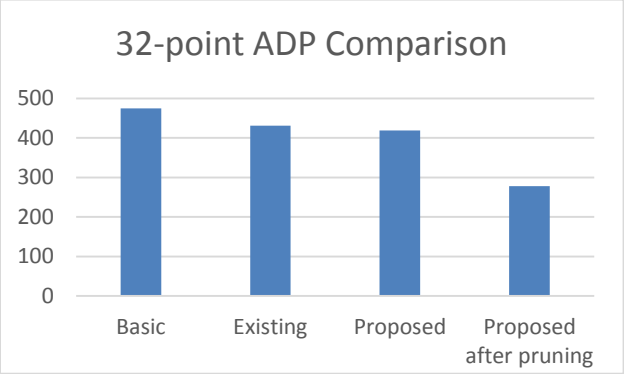

Figure9: Comparison of Area-Delay Product

Here the observed readings show that proposed algorithm occupies very less area and provides less amount of delay. Also the proposed design results in less area delay product as shown in figure9. Hence the proposed architecture optimised in terms of Design metrics. Also the proposed pruned architecture shows considerable improvement in Area-Delay-Product.

CONCLUSION

A new architecture for implementation of 4,8,16 and 32 point 1D integer DCT for HEVC is implemented and synthesized for Vertex-E FPGA. Also found that the implemented design consumes less number of LUTs and produces less amount of delay hence result in less Area Delay Product and Power Delay Product when compared to Existing and Basic methods. Also 2D integer DCT implemented in 4,8,16 and 32 point input vector. The proposed architecture with 32-bit length is 29.2% and 9.2% area efficient, also results in 13.1% and 2.8% less Area-Delay product respectively when compared to basic and existing models. Also pruning is applied to proposed architecture to improve the performance which results in 50.78% decrease in area Delay product for 32-point integer DCT. The pruned architecture for 4,8,16 and 32 point Integer DCT architectures were synthesized.

REFERENCES:

[1] A. Rosenfeld and A. Kak, Digital Picture Processing. New York: Academic. New York, 1982.
[2] A. K. Jain. "A sinusoidal family of unitary transforms." IEEE Trans. Pattern Analysis Much. Inrel., vol. 4, pp. 356-365, Oct. 1979.and Consent), JCT-VC L1003, Geneva, Switzerland, Jan. 2013.
[3] K.R. Rao, P. Yip, Discrete Cosine Transform: Algorithms, Advantages, Applications, Academic Press, Boston, 1990.
[4] G. Strang, The discrete cosine transform, SIAM Rev. 41 (1999) 135–147.
[5] V. Bhaskaran, K. Konstantinides, Images and Video Compression Standards: Algorithms and Architectures, Kluwer, Boston, 1997.
[6] M.W. Marcellin, M.J. Gormish, A. Bilgin, M.P. Boliek, An overview of JPEG-2000, in: Proc. Data Compression Conf., 2000, pp. 523–541.
[7] W.K. Cham, P.C. Yip, Integer sinusoidal transforms for image processing, Internat. J. Electron. 70 (1991) 1015–1030.
[8] L.Z. Cheng, H. Xu, Y. Luo, Integer discrete cosine transform and its fast algorithm, Electron. Lett. 37 (2001) 64–65.
[9] K. Komatsu, K. Sezaki, Reversible discrete cosine transform, in: Proc. IEEE Internat. Conf. Acoust. Speech Signal Process. 1998, pp. 1769–1772.
[10] J. Liang, T.D. Tran, Fast multiplier less approximations of the DCT: The lifting scheme, IEEE Trans. Signal Process. 49 (2001) 3032–3044.
[11] W. Philips, Lossless DCT for combined lossy/lossless image coding, in: Proc. IEEE Internat. Conf. Image Process. Vol. 3, 1998, pp. 871–875