Chapter 4

Approximated Transform Core Architecture for HEVC

This chapter explains the derivation of Walsh–Hadamard Transform (WHT) based DCT and IDCT matrices for HEVC. Based on that, different approximated models are proposed to reduce the hardware complexity of an HEVC encoder with very small compromise in the coding performance. Those approximate models are implemented on FPGA and ASIC platforms and the complexity of the core transform architecture for HEVC is assessed.

4.1 Introduction

HEVC can reduce the bit rate by 50% as compared to that of the H.264/AVC [11] to achieve the same visual quality. To achieve this saving, a large number of new features such as more number of intra and inter prediction modes, recursive quadtree partitioning, higher order TU have been introduced in the HEVC. All these

newly introduced features increase the encoder complexity by about 40 – 70% [71] as compared to that of H.264/AVC. On the other hand, small area and low power consumption are the primary design constraints in case of portable hand-held devices as the battery life is limited. Therefore, architectural optimization by suitable approximation is imperative to reduce the power consumption and computational complexity for HEVC.

It is observed that RDO process [72] consumes a major portion of the power and hardware resources. RD cost is calculated after splitting each coding block quadratically and recursively to obtain optimum block size. So, transform and entropy coding are recursively performed in each quadtree depth level. It means, optimizing the RDO process can significantly reduce the power consumption and complexity of the encoder. Several methods have been proposed at algorithmic level which further reduces the RDO complexity. Fast mode decision methods for intra- or inter- coding [105–107] and all zero block detection [108–110] are few of them. These optimization techniques depend on the characteristics of predicted signals and employ less complex WHT [111] to analyze it.

Some works have proposed RDO optimization by approximating forward- and inverse-DCT. Among them, the most popular approximation is integer transform [67] which is discussed in Chapter 3. In integer transform, real-valued coefficients are replaced by integers such that propoerties of DCT are mostly maintained. Integer transform invokes integer arithmatic only and therefore, reduces the computational complexity. Such type of integer transform is proposed and adopted in HEVC core transform [66]. Its hardware implementation is presented in [68] and in Chapter 3 which use MCMs [69] to obviate the need of multipliers. Integer transform is further approximated in [73, 74, 77, 78] and optimized designs are proposed. However, these approximation methods use square-wave like transform and hence, their accuracy and coding

performance degrade significantly. Fixed point approximation of DCT coefficients is used in [79] and discussed in Chapter 3 which improves hardware cost with minimal degradation in coding performance.

The HEVC encoder requires WHT to determine the sum of absolute transformed differences (SATD) in the prediction unit. Architecture proposed in [70] exploits the relationship between WHT and DCT. Such type of architecture uses pre-computed WHT results from the prediction unit to calculate DCT. However, it requires a large number of rotation units which increase the hardware cost and delay. Masera et al. [71] proposed an approximate method which dynamically skips some rotations depending on the characteristics of the input signal.

WHT based transform architecture is the most suitable for HEVC as the same architecture can be used to compute both the transform coefficients and WHT used in prediction stage. However, it involves a huge number of arithmetic operations to realize a large set of rotations. Therefore, the performance of the architecture degrades in terms of area, power and speed. In order to reduce the complexity of the rotation based architecture, an approximation algorithm for WHT based transform architectures is proposed in this chapter. The algorithm results in an approximated architecture to affirm a particular trade-off between coding accuracy and hardware complexity. Four such approximated architectures are proposed which provide a significant reduction in hardware complexity with four different coding efficiencies. Many approximation techniques have already been proposed in [47, 55, 73, 74] which are very popular for image compression. However, most of them are not devised for video compression as it demands more accuracy than image. It has been observed that for the most of the cases the coding performance of the proposed WHT based approximation schemes is better than that of the earlier reported approximation methods.

TABLE 4.1: Percentage of different size transforms used in different sequences during RDO

Sequence class	AI				LD				RA			
	4	8	16	32	4	8	16	32	4	8	16	32
A	57.3	35.4	5.6	1.5	-	-	-	-	72.5	21.4	5.3	0.8
B	55.7	36	6.6	1.5	71.7	21.8	5.5	0.8	72.6	21.2	5.1	0.75
C	61	33	4.6	1.3	72	21.7	5.4	0.8	72.6	21.4	5.2	0.78
D	60.9	33	4.9	1.2	74	20.4	4.7	0.7	74.6	20.2	4.6	0.63
E	56.5	35.4	6.4	1.6	71.7	21.6	5.7	0.9	-	-	-	-
Av.	58.3	34.6	5.6	1.4	72.4	21.4	5.3	0.8	73	21	5	0.75

We have presented the WHT based DCT and IDCT architectures in this chapter. Other contributions of this chapter are as follows:

- Statistical analysis of various UHD video sequences is performed to understand the usage of different size of transform kernel in HEVC.
- New data flow models of WHT based transform architectures (forward and inverse) are proposed.
- An algorithm is proposed for approximating transform architectures and the effect of the proposed approximation algorithm on different size of transform has been analyzed to demonstrate the trade-off between the coding performance and hardware cost.
- An approximate higher order WHT and on its basis high-frequency coefficients pruning is proposed.

4.2 Transform size statistics in HEVC

Four different sizes (i.e., 4×4, 8×8, 16×16 and 32×32) of DCT and IDCT are used in the HEVC for forward and inverse transform operations of the residual data. It

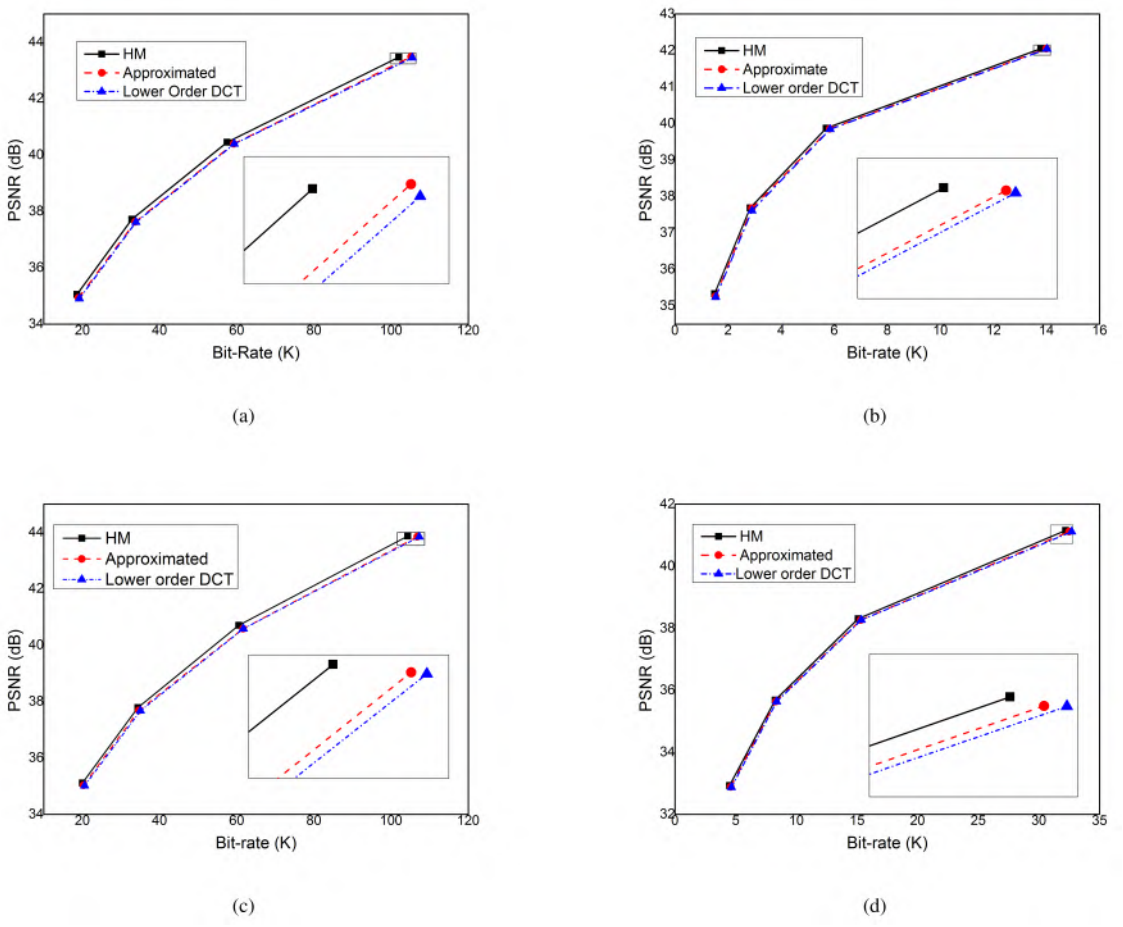


FIGURE 4.1: RD graph comparison of original DCT with higher order approximated and with only lower order (4 and 8) DCT. (a) & (c) for AI, and (b) & (d) for RA encoder configuration for Traffic (2560×1600) and PeopleOnStreet (2560×1600) sequences, respectively

has been observed that higher order transforms (i.e.,16×16 and 32×32) reduces bit rate up to 10% and this bit rate savings is more apparent for the sequences with higher resolution [66]. However, an acute increase in complexity is observed with the use of higher order transforms. Therefore, an efficient approximation of higher order transform architectures is imperative to reduce the complexity. To study the effect of approximation on coding performance, we performed different statistical analysis with HEVC test model HM-16.15 [12]. Different encoder configurations viz. AI, LD and RA are used with the test conditions and sequences described in [103].

We studied the percentage of the different size of transform operations performed in HEVC. These results are presented in Table 4.1. After averaging over different test sequences of a specific class, it is observed that for any kind of encoder configuration more than 90% of times lower order transforms are performed, and the higher order transform operations are performed less than 8% of times. Among different encoder configurations, almost the same percentage of higher order operations are used in LD and RA configuration. However, it is more in the case of AI configuration. All type of sequences use almost same percentage of different size transform operations. So, we have selected sequences of the highest resolution (i.e., Class A) for rest other statistical analysis.

It is clear from the previous discussion that the higher order transform operations increase complexity significantly. However, they are rarely used during video coding. Therefore, an approximation technique for the transform kernel can be used to reduce computational complexity. Hence, to study the effect of approximation, we have replaced higher order DCTs by WHT and observed the YUV PSNR difference with the two configurations of HM software. The first configuration is the default configuration which uses all size of DCT. In the second configuration, maximum transform size was limited to 8 (i.e., lower order DCT only). These results are plotted in Fig. 4.1. Note that, the difference in RD plots between HM default and approximated mode are more pronounced in AI than that of the RA configuration as it uses more number of higher order DCTs. It is also noticeable that although WHT is the bold approximation of the DCT, still approximated mode manages to achieve better coding performance over lower order DCT. Therefore, a little accuracy increment can produce reasonable trade-off between coding efficiency and hardware complexity. The approximation can be further extended by discarding higher frequency components of larger size DCTs. As the higher order DCTs are applied

only in the homogeneous regions, most of the energy can be preserved by calculating only few low-frequency components. Therefore, these statistics prove that the higher order DCTs can tolerate some approximations with a negligible coding loss to reduce the computational complexity. The same is true for IDCT also. So, in this chapter, we have proposed a signal flow diagram of 32-point DCT as well as IDCT and their different approximated architectures to sustain a balance between performance accuracy and hardware complexity.

4.3 Approximated architectures

In this Section, we have proposed generalized models for N^{th} order WHT based DCT and IDCT. An algorithm is presented to approximate the WHT based DCT as well as IDCT architectures, and the effect of the approximation algorithm on different size of transforms are measured.

4.3.1 Proposed DCT architectures

The advantage of WHT based DCT architecture is two folds. Firstly, it can use precomputed WHT coefficients from the prediction stage to compute DCT. Secondly, WHT based DCT architecture requires less hardware resources as compared to the other designs [112]. Designs in [71] and [70] are also WHT based DCT architectures for HEVC. However, a new approach of WHT based DCT for HEVC is proposed in this chapter which obviates the need of hardware blocks for bit reversal, gray coding and lifting scheme discussed in [70, 71]. Consequently, hardware cost and power consumption of the proposed architecture is less as compared to [71] and [70].

As suggested in [112], any N-point DCT matrix $\boldsymbol{D_N}$ can be written in terms of orthonormal WHT matrix $\hat{\boldsymbol{W}}_{\boldsymbol{N}}$ as

$$\boldsymbol{D_N} = \boldsymbol{D_N}.\hat{\boldsymbol{W}}_{\boldsymbol{N}}{}^{T}.\hat{\boldsymbol{W}}_{\boldsymbol{N}} = \boldsymbol{T_N}.\hat{\boldsymbol{W}}_{\boldsymbol{N}} = \frac{1}{\sqrt{N}}.\boldsymbol{T_N}.\boldsymbol{W_N}, \tag{4.1}$$

where

$$\boldsymbol{W_N} = \sqrt{N}.\hat{\boldsymbol{W}}_{\boldsymbol{N}} \tag{4.2}$$

and

$$\boldsymbol{T_N} = \boldsymbol{D_N}.\hat{\boldsymbol{W}}_{\boldsymbol{N}}{}^{T}. \tag{4.3}$$

The matrix $\boldsymbol{W_N}$ can be obtained by re-ordering the basis vectors of Hadamard matrix as shown in [111].

$\boldsymbol{T_N}$ is a orthonormal matrix which can be recursively decomposed to lower order matrices and finally, we obtain $\boldsymbol{T_2}$ as an identity matrix of size 2×2. Therefore, (4.1) can be written as

$$\begin{aligned} \boldsymbol{D_N} &= \frac{1}{\sqrt{N}}.\boldsymbol{P_N}{}^{T}.\begin{bmatrix} \boldsymbol{T}_{\frac{\boldsymbol{N}}{\boldsymbol{2}}} & \boldsymbol{0} \\ \boldsymbol{0} & \boldsymbol{U}_{\frac{\boldsymbol{N}}{\boldsymbol{2}}} \end{bmatrix}.\boldsymbol{P_N}.\boldsymbol{W_N} \\ &= 2^{-\frac{m}{2}}.\boldsymbol{P_N}{}^{T}\begin{bmatrix} \boldsymbol{T}_{\frac{\boldsymbol{N}}{\boldsymbol{2}}} & \boldsymbol{0} \\ \boldsymbol{0} & \boldsymbol{U}_{\frac{\boldsymbol{N}}{\boldsymbol{2}}} \end{bmatrix}\begin{bmatrix} \boldsymbol{W}_{\frac{\boldsymbol{N}}{\boldsymbol{2}}} & \boldsymbol{0} \\ \boldsymbol{0} & \boldsymbol{W}_{\frac{\boldsymbol{N}}{\boldsymbol{2}}} \end{bmatrix}\begin{bmatrix} \boldsymbol{I}_{\frac{\boldsymbol{N}}{\boldsymbol{2}}} & \boldsymbol{J}_{\frac{\boldsymbol{N}}{\boldsymbol{2}}} \\ \boldsymbol{I}_{\frac{\boldsymbol{N}}{\boldsymbol{2}}} & -\boldsymbol{J}_{\frac{\boldsymbol{N}}{\boldsymbol{2}}} \end{bmatrix}. \end{aligned} \tag{4.4}$$

Here, $\boldsymbol{I}_{\frac{\boldsymbol{N}}{\boldsymbol{2}}}$ and $\boldsymbol{J}_{\frac{\boldsymbol{N}}{\boldsymbol{2}}}$ are identity and cross-identity matrices of size $\frac{N}{2} \times \frac{N}{2}$, respectively. An example of cross-identity matrix of order four is shown in (4.5).

$$\boldsymbol{J_4} = \begin{bmatrix} 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 \\ 0 & 1 & 0 & 0 \\ 1 & 0 & 0 & 0 \end{bmatrix}. \tag{4.5}$$

$\boldsymbol{P_N}$ is a permutation matrix of size $N \times N$. For example, permutation matrix $\boldsymbol{P_8}$ can be written as

$$\boldsymbol{P_8} = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \end{bmatrix}. \tag{4.6}$$

Further, $\boldsymbol{U_{\frac{N}{2}}}$ is a matrix of size $\frac{N}{2} \times \frac{N}{2}$ and used to generate the odd DCT coefficients. This matrix can be further decomposed as

$$\boldsymbol{U_{\frac{N}{2}}} = \boldsymbol{V_{\frac{N}{2}}} . \begin{bmatrix} \boldsymbol{U_{\frac{N}{4}}} & \boldsymbol{0} \\ \boldsymbol{0} & \boldsymbol{U_{\frac{N}{4}}} \end{bmatrix}. \tag{4.7}$$

$\boldsymbol{U_1}$=1 and $\boldsymbol{V_{\frac{N}{2}}}$ is a rotational matrix of size $\frac{N}{2} \times \frac{N}{2}$ which can be written as

$$\boldsymbol{V_{\frac{N}{2}}} = \begin{bmatrix} \cos(\theta) & 0 & \cdots & \cdots & 0 & \sin(\theta) \\ 0 & \ddots & 0 & 0 & \iddots & 0 \\ \vdots & 0 & \cos(k.\theta) & \sin(k.\theta) & 0 & \vdots \\ \vdots & 0 & -\sin(k.\theta) & \cos(k.\theta) & 0 & \vdots \\ 0 & \iddots & 0 & 0 & \ddots & 0 \\ -\sin(\theta) & 0 & \cdots & \cdots & 0 & \cos(\theta) \end{bmatrix}, \tag{4.8}$$

where $\theta = \frac{\pi}{2N}$ and $k = \frac{N}{2} - 1$.

In HEVC, real-valued elements of D_N are rounded to the nearest integer after multiplication with a constant value. Therefore, integer DCT C_N can be written as

$$\boldsymbol{C_N} = \lfloor const \times \boldsymbol{D_N} \rceil, \tag{4.9}$$

where $\lfloor * \rceil$ represents rounding operation. In HEVC, $const = 2^{6+\frac{m}{2}}$, where $m = log_2 N$. The basis vectors of the integer DCT remain almost orthogonal and maintain almost equal norm. In this way, drifting error and the requirement of the floating point multiplier is eliminated.

From (4.1), (4.4) and (4.9), HEVC compliance DCT matrix $\boldsymbol{C_N}$ can be expressed as

$$\begin{aligned} \boldsymbol{C_N} &= \lfloor 2^{6+\frac{m}{2}}.\boldsymbol{D_N} \rceil = \lfloor 2^6.\boldsymbol{T_N} \rceil.\boldsymbol{W_N} \\ &= \boldsymbol{P_N}^T.\begin{bmatrix} \lfloor 2^6.\boldsymbol{T_{\frac{N}{2}}} \rceil & \boldsymbol{0} \\ \boldsymbol{0} & \lfloor 2^6.\boldsymbol{U_{\frac{N}{2}}} \rceil \end{bmatrix}.\begin{bmatrix} \boldsymbol{W_{\frac{N}{2}}} & \boldsymbol{0} \\ \boldsymbol{0} & \boldsymbol{W_{\frac{N}{2}}} \end{bmatrix}.\begin{bmatrix} \boldsymbol{I_{\frac{N}{2}}} & \boldsymbol{J_{\frac{N}{2}}} \\ \boldsymbol{I_{\frac{N}{2}}} & -\boldsymbol{J_{\frac{N}{2}}} \end{bmatrix}. \end{aligned} \tag{4.10}$$

It can be observed from (4.10) that 1D DCT of a vector $\boldsymbol{x} = [x_0, x_1, \cdots, x_{N-1}]^T$ can be computed as

$$\begin{bmatrix} y_0^N \\ y_2^N \\ \vdots \\ y_{N-2}^N \end{bmatrix} = \boldsymbol{C_{\frac{N}{2}}}.\begin{bmatrix} a_0 \\ a_1 \\ \vdots \\ a_{\frac{N}{2}-1} \end{bmatrix} \tag{4.11}$$

$$\begin{bmatrix} y_1^N \\ y_3^N \\ \vdots \\ y_{N-1}^N \end{bmatrix} = \hat{\boldsymbol{U}}_{\boldsymbol{\frac{N}{2}}}.\boldsymbol{W_{\frac{N}{2}}}.\begin{bmatrix} b_0 \\ b_1 \\ \vdots \\ b_{\frac{N}{2}-1} \end{bmatrix}, \tag{4.12}$$

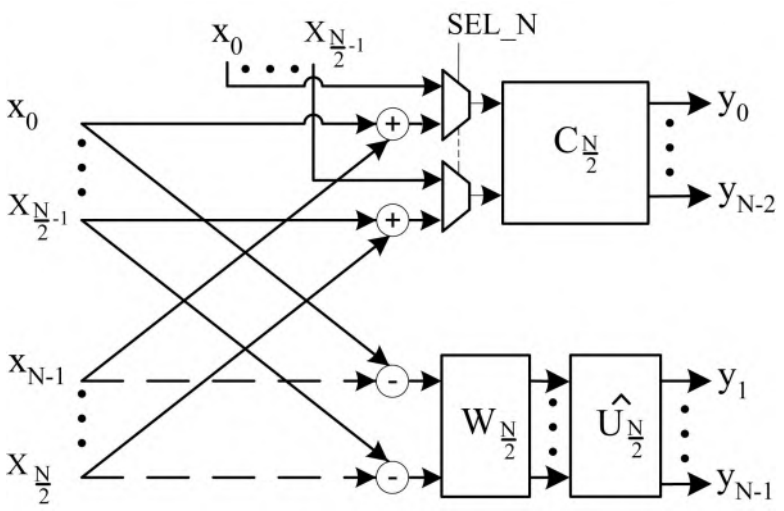


FIGURE 4.2: An N-point 1D DCT architecture

where

$$\hat{\boldsymbol{U}_{\frac{N}{2}}} = \lfloor 2^6 . \boldsymbol{U}_{\frac{N}{2}} \rceil = \left\lfloor \boldsymbol{V}_{\frac{N}{2}} . \begin{bmatrix} \hat{\boldsymbol{U}_{\frac{N}{4}}} & \boldsymbol{0} \\ \boldsymbol{0} & \hat{\boldsymbol{U}_{\frac{N}{4}}} \end{bmatrix} \right\rfloor \tag{4.13}$$

and

$$\begin{aligned} a_n &= x_n + x_{N-n-1} \\ b_n &= x_n - x_{N-n-1}. \end{aligned} \tag{4.14}$$

The vector $\boldsymbol{y} = \{y_0, y_1, \ldots, y_{N-1}\}$ represents output of the DCT operation. The generalized architecture for N-point 1D DCT is shown in Fig. 4.2. An IB unit is required to compute all a_n and b_n signals. Multiplexers are placed in the input side so that the same architecture can compute $\frac{N}{2}$-point 1D DCT also. Only $\boldsymbol{C}_{\frac{N}{2}}$ module is used during $\frac{N}{2}$-point DCT computation and the rest of the architecture remains idle. Thus, a 32-point DCT core ($\boldsymbol{C_{32}}$) consists of $\boldsymbol{C_{16}}$, $\boldsymbol{C_8}$, $\boldsymbol{C_4}$ modules and hence, one can compute 4, 8, 16, 32 -point DCT depending on the select lines 'SEL_N'. A decoder can be used to generate these select lines.

4.3.2 Proposed Approximation

The architecture proposed in Fig. 4.2 demands a large number of cascaded rotational units to compute the odd DCT coefficients. The most efficient technique to realize a rotation unit is by lifting method [51] which decompose each rotation into three steps to reduce the computational complexity. However, three multipliers and three adders are still needed to realize a single rotational unit. Therefore, the physical realization of this architecture requires large hardware resources and computational time is also high. To avoid the use of multipliers and to reduce the hardware resources, each rotational unit is approximated such that orthogonality and normality errors are minimized. To attain this, we have approximated each *cosine* as well as *sine* terms in $\boldsymbol{V_{\frac{N}{2}}}$ as below.

$$\begin{aligned} \cos(p.\theta) &= 2^{-a_p}, \ \ for \ p = 1, 3, 5, \cdots, k; \\ \sin(p.\theta) &= 2^{-b_p}, \ \ for \ p = 1, 3, 5, \cdots, k; \end{aligned} \tag{4.15}$$

where a_p and b_p are the integers. For 32-point DCT the magnitude of $\cos(p.\theta)$ and $\sin(p.\theta)$ varies from 1 to $\frac{\pi}{64}$. Hence, magnitude of a_p and b_p varies from 0 to 5. Algorithm 1 is used to approximate $\cos(p.\theta)$ and $\sin(p.\theta)$ for every value of p. It selects the best values of a_p and b_p depending on the minimum error (i.e., f_{err}) in either orthogonality or normality. Orthogonality and normality errors can be calculated as

$$\begin{aligned} Orthogonality\_err &= |\boldsymbol{D_N}.\boldsymbol{D_N}^T - \text{diag}(\text{diag}(\boldsymbol{D_N}.\boldsymbol{D_N}^T))| \\ Orthonormality\_err &= |\boldsymbol{I_N} - \text{diag}(\text{diag}(\boldsymbol{D_N}.\boldsymbol{D_N}^T))|, \end{aligned} \tag{4.16}$$

where I_N is the identity matrix of size $N \times N$. It has been observed that both the approximated matrices based on orthogonality and normality error are orthogonal. However, normality error is less for the later one. Therefore, the function f_{err} is selected to calculate the normality error in the approximated matrix. It has been

Algorithm 1 Approximation of *cosine* and *sine* terms

procedure APPROXIMATION($\boldsymbol{V_{\frac{N}{2}}}$)
  for $p = 1 : 2 : k$ **do**
    $E_{now} \leftarrow inf$;
    for $a_p (or\ b_p) = 0 : 5$ **do**
      $E_{prev} \leftarrow E_{now}$; $\boldsymbol{V_{\frac{N}{2}new}} \leftarrow \boldsymbol{V_{\frac{N}{2}}}$;
      Obtain $\boldsymbol{V_{\frac{N}{2}new}}$ by replacing $\cos(p.\theta)$ by 2^{-a_p}
      (or $\sin(p.\theta)$ by 2^{-b_p});
      Obtain $\boldsymbol{D_N}$ from $\boldsymbol{V_{\frac{N}{2}new}}$;
      $E_{now} \leftarrow f_{err}(\boldsymbol{D_N})$;
      if $E_{now} < E_{prev}$ **then**
        $\boldsymbol{V_{\frac{N}{2}}} \leftarrow \boldsymbol{V_{\frac{N}{2}new}}$

observed that the proposed algorithm produces a matrix similar to DCT which is completely orthogonal.

To investigate the impact of the proposed approximation, we have studied the coding performance for the following four cases using HM-16.0 [12] software with common test conditions as described in [103].

1. The proposed approximation method was applied to all size DCT (CASE-I).

2. The proposed approximation method was applied to all size DCT except order 4 (CASE-II).

3. The proposed approximation method was applied to all size DCT except order 4 and 8 (CASE-III).

4. The proposed approximation method was applied on DCT of order 32 only, but not on 4, 8 and 16. (CASE-IV).

TABLE 4.2: PSNR loss due to the proposed approximation on different DCT size with (1) AI (2) LD (3) RA configurations

Class	Sequences	C-I	C-II	C-III	C-IV
A	Traffic	0.272	0.149	0.063	0.031
	PeopleOnStreet	0.283	0.156	0.060	0.030
B	CrowdRun	0.235	0.151	0.062	0.025
	ParkScene	0.223	0.134	0.064	0.026
E	KristenAndSara	0.245	0.148	0.072	0.034
	FourPeople	0.205	0.115	0.055	0.023

(1)

Class	Sequences	C-I	C-II	C-III	C-IV
B	CrowdRun	0.139	0.068	0.018	0.005
	ParkScene	0.087	0.048	0.014	0.007
E	KristenAndSara	0.215	0.130	0.032	0.003
	FourPeople	0.196	0.116	0.042	0.005

(2)

Class	Sequences	C-I	C-II	C-III	C-IV
A	Traffic	0.131	0.072	0.038	0.014
	PeopleOnStreet	0.1336	0.068	0.020	0.010
B	CrowdRun	0.125	0.064	0.021	0.008
	ParkScene	0.026	0.062	0.026	0.009

(3)

TABLE 4.3: Number of rotations performed in 1D DCT

Block Size	C-I	C-II	C-III	C-IV	Reference
4x4	0	4	4	4	4
8x8	0	8	40	40	40
16x16	0	16	80	272	272
32x32	0	32	160	544	1568

For all those cases YUV-PSNR variations are calculated with respect to HEVC reference software using the standard BD-rate approach [88] with four different quantization parameters (i.e., 22, 27, 32 and 37). The YUV-PSNR variations are furnished in Table 4.2 for AI, LD and RA encoder configurations.

The proposed approximation algorithm replaces the rotation unit by a butterfly structure. Therefore, rotations are inessential with the proposed approach. The

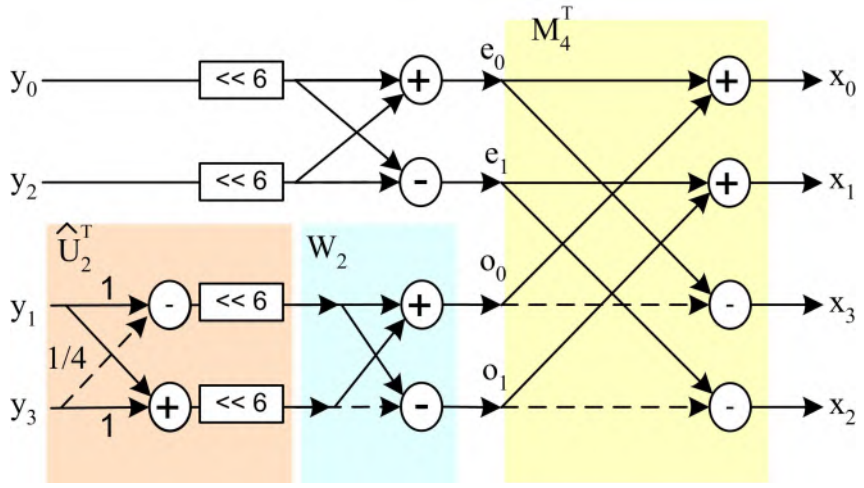


FIGURE 4.3: Architecture of (a) 4-point DCT (b) Multiplier Unit (MU)

details of the number of rotations saved for each case are furnished in Table 4.3.

It can be observed that for large size DCT, θ is very small. Therefore, the terms $\cos(\phi)$ and $\sin(\phi)$ (i.e.,$\phi = p.\theta$) which produce low frequency DCT coefficients can be approximated as $\cos(\phi) \approx 1$ and $\sin(\phi) \approx \phi$. These large size DCTs are applied in homogeneous regions where the most of the energy is concentrated in few low-frequency components. Therefore, the impact of the proposed approximation in the large size DCT is less. Consequently, the CASE-IV produces the best coding performances as reflected in Table 4.2. Note that, in CASE-IV, the performance degradation is insignificant even after the 32-point DCT has been approximated by the algorithm 1. On the other hand, least number of rotations are performed in CASE-I thereby limiting the required hardware resources to the minimum. However, CASE-II and CASE-III also produces negligible performance loss with respect to the standard integer DCT in HEVC reference software and produces a good trade-off between the coding performance and hardware resource requirement.

4.3.3 Hardware implementation of DCT architectures

Although we have implemented all four architectures, hardware design approach for only CASE-II is discussed in this section. As DCT and IDCT have almost similar structures, The hardware architecture for CASE-I is elaborated in IDCT designing part of this chapter. The same design approach can be extended to other cases as well.

4.3.3.1 4-point integer DCT

From (4.11), (4.12) and (4.14) the integer DCT kernel for 4-point DCT can be written as

$$\begin{bmatrix} y_0^4 \\ y_2^4 \end{bmatrix} = \boldsymbol{C_2}.\begin{bmatrix} a_0 \\ a_1 \end{bmatrix} = \begin{bmatrix} 64 & 64 \\ 64 & -64 \end{bmatrix}.\begin{bmatrix} a_0 \\ a_1 \end{bmatrix}$$
$$\begin{bmatrix} y_1^4 \\ y_3^4 \end{bmatrix} = \hat{\boldsymbol{U}}_{\boldsymbol{2}}.\boldsymbol{W_2}.\begin{bmatrix} b_0 \\ b_1 \end{bmatrix} \tag{4.17}$$

where,

$$\boldsymbol{W_2} = \begin{bmatrix} 1 & 1 \\ 1 & -1 \end{bmatrix} \tag{4.18}$$

and

$$\begin{aligned} a_0 &= x_0 + x_3; \quad a_1 = x_1 + x_2; \\ b_0 &= x_0 - x_3; \quad b_1 = x_1 - x_2; \end{aligned} \tag{4.19}$$

From Section 4.2, it can be observed that 4-point DCT is frequently used and its complexity is the minimum. Therefore, in CASE-II proposed approximation method is not applied for 4-point DCT. A schematic diagram of 4-point integer DCT is shown

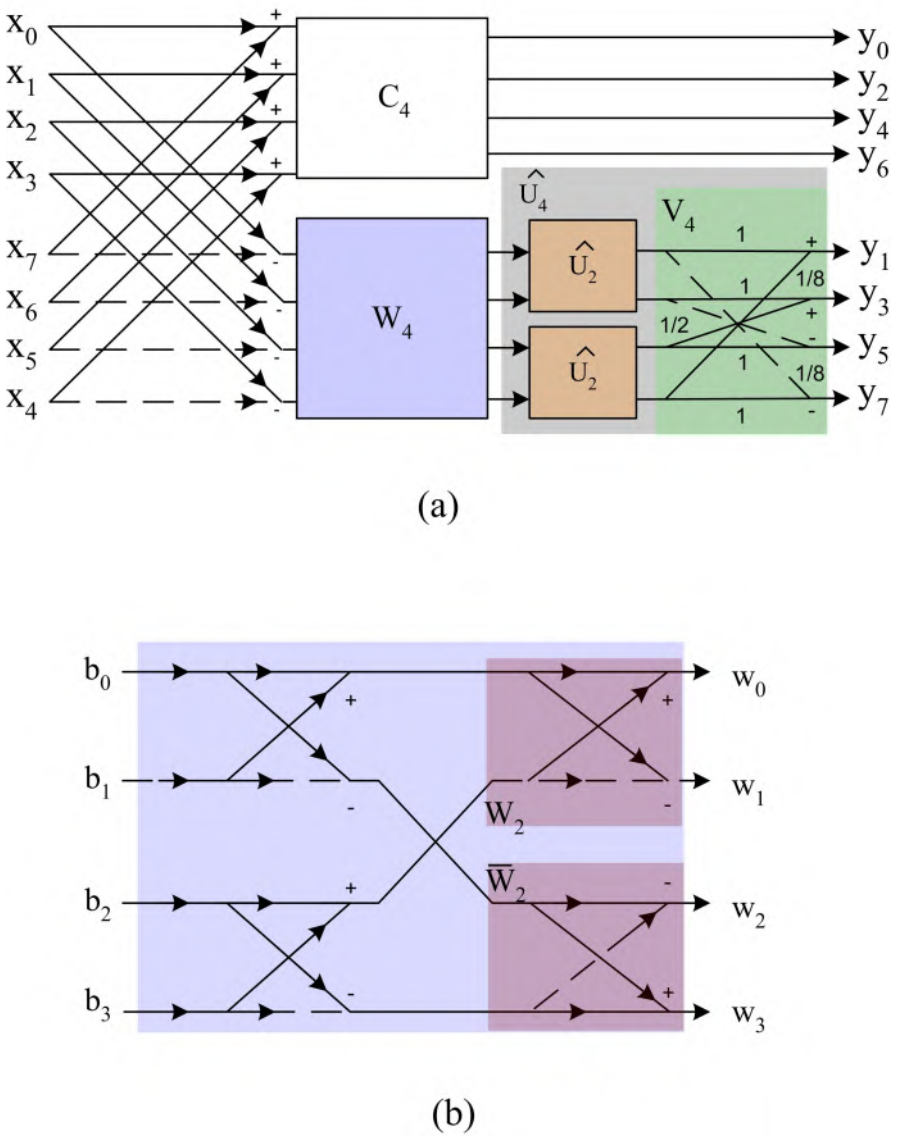


FIGURE 4.4: Architecture of (a) 8-point DCT (b) 4-point WHT ($\boldsymbol{W_4}$) using two 2-point WHT ($\boldsymbol{W_2}$)

in Fig. 4.3(a). Here, multiplications are performed by Multiplier Unit (MU) and each MU can perform two multiplications simultaneously. Shift and add operations are used to realize these MUs. We used Hcub algorithm with minimum adder tree depth to optimize shift-add structure for all the MUs. We observed that adder tree depth in all the MUs can be bounded to one with a slight hand tuning of few

coefficients. Hence, $\hat{\boldsymbol{U}}_{\boldsymbol{2}}$ is selected as

$$\hat{\boldsymbol{U}}_{\boldsymbol{2}} = \begin{bmatrix} 60 & 24 \\ -24 & 60 \end{bmatrix}. \tag{4.20}$$

Schematic model of a MU is shown in Fig. 4.3(b).

4.3.3.2 8-point integer DCT

Following the similar approach 8-point DCT coefficients of a vector $\boldsymbol{x} = [x_0, x_1, \cdots, x_7]$ can be calculated as

$$\begin{aligned} \begin{bmatrix} y_0^8 & y_2^8 & y_4^8 & y_6^8 \end{bmatrix}^T &= \boldsymbol{C_4}.\begin{bmatrix} a_0 & a_1 & a_2 & a_3 \end{bmatrix}^T \\ \begin{bmatrix} y_1^8 & y_3^8 & y_5^8 & y_7^8 \end{bmatrix}^T &= \hat{\boldsymbol{U}}_{\boldsymbol{4}}.\boldsymbol{W_4}.\begin{bmatrix} b_0 & b_1 & b_2 & b_3 \end{bmatrix}^T. \end{aligned} \tag{4.21}$$

The respective values of a_n and b_n, for $N = 8$, can be calculated from (4.14). $\boldsymbol{W_4}$ is WHT matrix of order 4 and can be represented as

$$\boldsymbol{W_4} = \begin{bmatrix} 1 & 1 & 1 & 1 \\ 1 & 1 & -1 & -1 \\ 1 & -1 & -1 & 1 \\ 1 & -1 & 1 & -1 \end{bmatrix}. \tag{4.22}$$

According to (4.13) $\hat{\boldsymbol{U}}_{\boldsymbol{4}}$ can be computed in terms of $\hat{\boldsymbol{U}}_{\boldsymbol{2}}$ as follows:

$$\hat{\boldsymbol{U}}_{\boldsymbol{4}} = \boldsymbol{V_4}.\begin{bmatrix} \hat{\boldsymbol{U}}_{\boldsymbol{2}} & \boldsymbol{0} \\ \boldsymbol{0} & \hat{\boldsymbol{U}}_{\boldsymbol{2}} \end{bmatrix}, \tag{4.23}$$

where $\hat{\boldsymbol{U}}_{\boldsymbol{2}}$ is as shown in (4.20) and $\boldsymbol{V_4}$ can be written as

$$\boldsymbol{V_4} = \begin{bmatrix} \cos(\frac{\pi}{16}) & 0 & 0 & \sin(\frac{\pi}{16}) \\ 0 & \cos(\frac{3\pi}{16}) & \sin(\frac{3\pi}{16}) & 0 \\ 0 & \sin(\frac{3\pi}{16}) & -\cos(\frac{3\pi}{16}) & 0 \\ \sin(\frac{3\pi}{16}) & 0 & 0 & -\cos(\frac{\pi}{16}) \end{bmatrix}. \tag{4.24}$$

The procedure 'Approximation' is called with original $\boldsymbol{V_4}$ as its argument to find approximated $\boldsymbol{V_4}$ as discussed in Section 4.3.2. As there are two different angles (i.e.,$\frac{\pi}{16}$ and $\frac{3\pi}{16}$) in (4.24), two iterations of *'for'* loop in Algorithm 1 are required. Approximated value of sine or cosine term of an angle is evaluated in a single iteration of the loop. For example, $\sin(\frac{\pi}{16})$ can be approximated by the near about dyadic number $\frac{1}{4}$ or $\frac{1}{8}$. However, in the first iteration the proposed procedure checks all the dyadic values from 1 to $\frac{1}{32}$ and select the best one which produces minimum orthonormality error. Hence, the approximated $\boldsymbol{V_4}$ becomes

$$\boldsymbol{V_4} = \begin{bmatrix} 1 & 0 & 0 & \frac{1}{8} \\ 0 & 1 & \frac{1}{2} & 0 \\ 0 & -\frac{1}{2} & 1 & 0 \\ -\frac{1}{8} & 0 & 0 & 1 \end{bmatrix}. \tag{4.25}$$

Here, all the elements of the matrix $\boldsymbol{V_4}$ can be realized by shift operations and hence, need of the multipliers and rotation units is eliminated. An 8-point DCT architecture is shown in Fig. 4.4(a). Multiplexers at the input side are not shown in this figure to avoid clutter. The proposed 8-point DCT architecture requires a 4-point WHT module. Any N-point WHT can be calculated as follows:

$$\boldsymbol{W_N} = \begin{bmatrix} \boldsymbol{W_{\frac{N}{2}}} & \boldsymbol{0} \\ \boldsymbol{0} & \boldsymbol{\bar{W}_{\frac{N}{2}}} \end{bmatrix} . \begin{bmatrix} 1 & 1 & 0 & \cdots & 0 & 0 \\ \vdots & \vdots & \vdots & \cdots & \vdots & \vdots \\ 0 & 0 & 0 & \cdots & 1 & 1 \\ 1 & -1 & 0 & \cdots & 0 & 0 \\ \vdots & \vdots & \vdots & \cdots & \vdots & \vdots \\ 0 & 0 & 0 & \cdots & 1 & -1 \end{bmatrix} . \tag{4.26}$$

Here, $\boldsymbol{\bar{W}_{\frac{N}{2}}}$ is obtained by flipping $\boldsymbol{W_{\frac{N}{2}}}$ vertically. Therefore, 4-point WHT can be configured using two 2-point WHT as shown in Fig. 4.4(b).

To investigate the degree of approximation, we have calculated the equivalent 8-point DCT matrix $\boldsymbol{C_8}$ using (4.10) and is presented in (4.27).

$$\boldsymbol{C_8} = \begin{bmatrix} 64 & 64 & 64 & 64 & 64 & 64 & 64 & 64 \\ 88.5 & 79.5 & 46.5 & 25.5 & -25.5 & -46.5 & -79.5 & -88.5 \\ 84 & 36 & -36 & -84 & -84 & -36 & 36 & 84 \\ 78 & -6 & -102 & -66 & 66 & 102 & 6 & -78 \\ 64 & -64 & -64 & 64 & 64 & -64 & -64 & 64 \\ 66 & -102 & 6 & 78 & -78 & -6 & 102 & -66 \\ 36 & -84 & 84 & -36 & -36 & 84 & -84 & 36 \\ 25.5 & -46.5 & 79.5 & -88.5 & 88.5 & -79.5 & 46.5 & -25.5 \end{bmatrix} . \tag{4.27}$$

This matrix, $\boldsymbol{C_8}$, is HEVC compliance as it is almost equivalent to the integer DCT matrix used in HEVC.

4.3.3.3 Higher order DCTs

Like 4- and 8- point DCT, architectures for 16- and 32- point DCT can also be designed by adopting the same procedure. A 16-point DCT architecture is shown in Fig. 4.5(a). However, from the statistical analysis presented in Section 4.2, it is

clear that retaining few low-frequency coefficients of higher order DCTs maintains good trade-off between accuracy and complexity. Therefore, hardware for higher coefficients of 32-point DCT is pruned. Hardware pruning is achieved by spatial correlation property of a homogeneous residual block. Usually a 32-point DCT is used for homogeneous block where intensity of a sample $x_i \approx x_{i+1}$. Therefore, samples

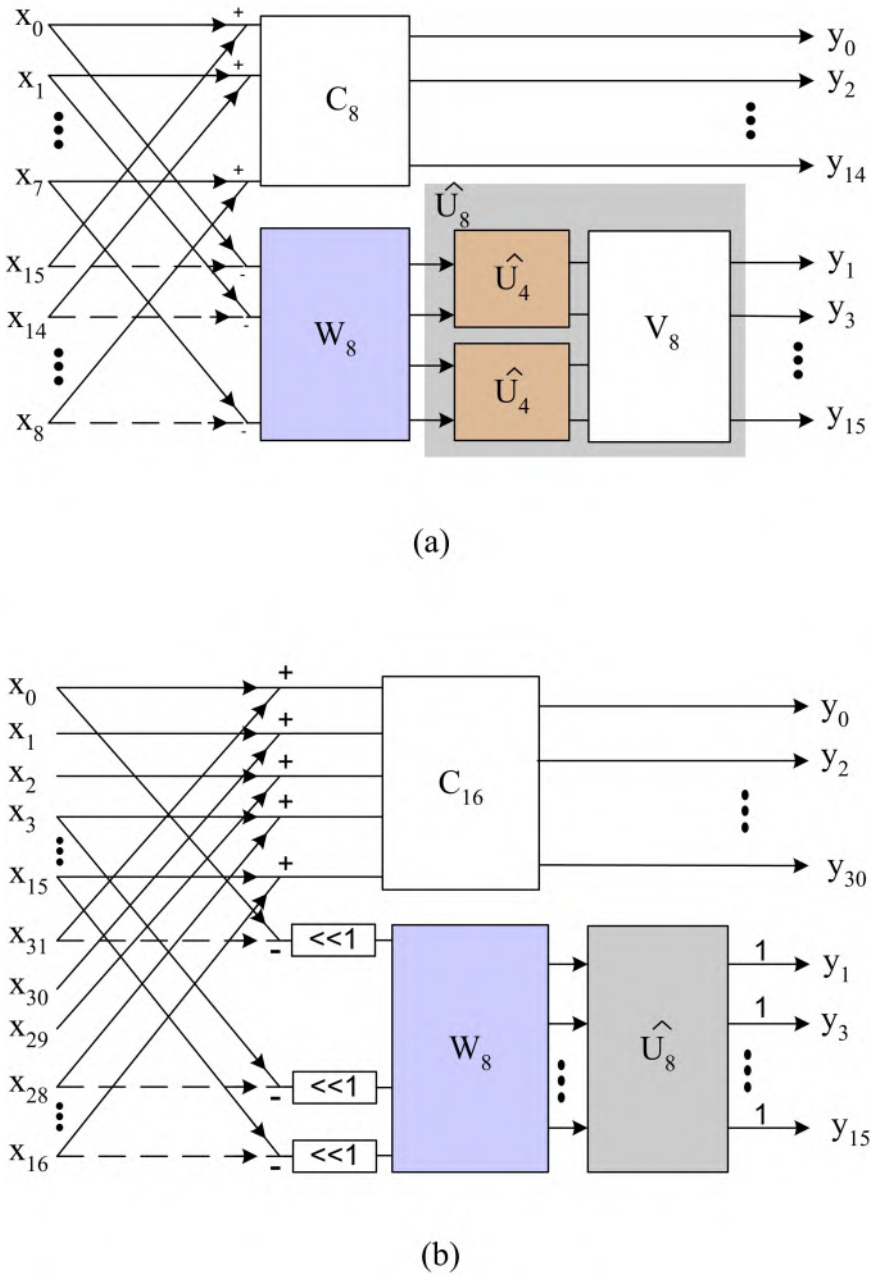


FIGURE 4.5: Architecture of (a) 16-point and (b) 32-point DCT

of a homogeneous block hold the following relations.

$$\begin{aligned}(x_i + x_{31-i}) &\approx (x_{i+1} + x_{30-i}) \\ (x_i - x_{31-i}) &\approx (x_{i+1} - x_{30-i})\end{aligned} \tag{4.28}$$

Hence, for 32-point DCT (4.26) can be written as

$$\boldsymbol{W_{16}} = \begin{bmatrix} \boldsymbol{W_8} & \boldsymbol{0} \\ \boldsymbol{0} & \bar{\boldsymbol{W}}_{\boldsymbol{8}} \end{bmatrix} . \begin{bmatrix} 2 & 0 & 0 & \cdots & 0 & 0 \\ \vdots & \vdots & \vdots & \cdots & \vdots & \vdots \\ 0 & 0 & 0 & \cdots & 0 & 2 \\ 0 & 0 & 0 & \cdots & 0 & 0 \\ \vdots & \vdots & \vdots & \cdots & \vdots & \vdots \\ 0 & 0 & 0 & \cdots & 0 & 0 \end{bmatrix}. \tag{4.29}$$

This expression provides 16 lower frequency coefficients of 32-point WHT whereas, remaining 16 high-frequency coefficients become zero. Hence, an optimized 32-point DCT module requires a 16-point DCT unit ($\boldsymbol{C_{16}}$), an 8-point WHT unit ($\boldsymbol{W_8}$) and a $\hat{\boldsymbol{U}}_{\boldsymbol{8}}$ unit as shown in Fig. 4.5(b).

4.3.4 Proposed IDCT architecture

Variable and large size transform has increased the coding efficiency, but at the cost of huge computational complexity. Therefore, optimized hardware implementation is a new research domain especially for the small hand-held and battery operated devices. Decoder optimization is of high priority for small handheld devices as they need to support HEVC decoding rather than encoding feature. Hence, IDCT optimization is very essential for such platforms where little compromise in accuracy is permitted. Resource sharing and approximation methods are tried and exploited

for the same reason in recent years. It may be feasible to share hardware resources among prediction unit, DCT and IDCT blocks in HEVC with WHT based matrix decomposition method.

Like DCT, the IDCT matrix can be decomposed as

$$\boldsymbol{D}_{\boldsymbol{N}}^{\boldsymbol{T}} = 2^{-\frac{m}{2}}.\begin{bmatrix} \boldsymbol{I}_{\frac{\boldsymbol{N}}{\boldsymbol{2}}} & \boldsymbol{I}_{\frac{\boldsymbol{N}}{\boldsymbol{2}}} \\ \boldsymbol{J}_{\frac{\boldsymbol{N}}{\boldsymbol{2}}} & -\boldsymbol{J}_{\frac{\boldsymbol{N}}{\boldsymbol{2}}} \end{bmatrix}\begin{bmatrix} \boldsymbol{W}_{\frac{\boldsymbol{N}}{\boldsymbol{2}}} & \boldsymbol{0} \\ \boldsymbol{0} & \boldsymbol{W}_{\frac{\boldsymbol{N}}{\boldsymbol{2}}} \end{bmatrix}\begin{bmatrix} \boldsymbol{T}_{\frac{\boldsymbol{N}}{\boldsymbol{2}}}^{\boldsymbol{T}} & \boldsymbol{0} \\ \boldsymbol{0} & \boldsymbol{U}_{\frac{\boldsymbol{N}}{\boldsymbol{2}}}^{\boldsymbol{T}} \end{bmatrix}\boldsymbol{P}_{\boldsymbol{N}}. \tag{4.30}$$

Similarly, the HEVC compliance IDCT matrix $\boldsymbol{C}_{\boldsymbol{N}}^{\boldsymbol{T}}$ can be expressed as

$$\begin{aligned} \boldsymbol{C}_{\boldsymbol{N}}^{\boldsymbol{T}} &= \lfloor 2^{6+\frac{m}{2}}.\boldsymbol{D}_{\boldsymbol{N}} \rceil = \boldsymbol{W}_{\boldsymbol{N}}.\lfloor 2^{6}.\boldsymbol{T}_{\boldsymbol{N}} \rceil^{T} \\ &= \begin{bmatrix} \boldsymbol{I}_{\frac{\boldsymbol{N}}{\boldsymbol{2}}} & \boldsymbol{I}_{\frac{\boldsymbol{N}}{\boldsymbol{2}}} \\ \boldsymbol{J}_{\frac{\boldsymbol{N}}{\boldsymbol{2}}} & -\boldsymbol{J}_{\frac{\boldsymbol{N}}{\boldsymbol{2}}} \end{bmatrix}.\begin{bmatrix} \boldsymbol{W}_{\frac{\boldsymbol{N}}{\boldsymbol{2}}} & \boldsymbol{0} \\ \boldsymbol{0} & \boldsymbol{W}_{\frac{\boldsymbol{N}}{\boldsymbol{2}}} \end{bmatrix}.\begin{bmatrix} \lfloor 2^{6}.\boldsymbol{T}_{\frac{\boldsymbol{N}}{\boldsymbol{2}}} \rceil^{T} & \boldsymbol{0} \\ \boldsymbol{0} & \lfloor 2^{6}.\boldsymbol{U}_{\frac{\boldsymbol{N}}{\boldsymbol{2}}} \rceil^{T} \end{bmatrix}.\boldsymbol{P}_{\boldsymbol{N}} \\ &= \begin{bmatrix} \boldsymbol{I}_{\frac{\boldsymbol{N}}{\boldsymbol{2}}} & \boldsymbol{I}_{\frac{\boldsymbol{N}}{\boldsymbol{2}}} \\ \boldsymbol{J}_{\frac{\boldsymbol{N}}{\boldsymbol{2}}} & -\boldsymbol{J}_{\frac{\boldsymbol{N}}{\boldsymbol{2}}} \end{bmatrix}.\begin{bmatrix} \boldsymbol{C}_{\frac{\boldsymbol{N}}{\boldsymbol{2}}}^{\boldsymbol{T}} & \boldsymbol{0} \\ \boldsymbol{0} & \boldsymbol{W}_{\frac{\boldsymbol{N}}{\boldsymbol{2}}}.\lfloor 2^{6}.\boldsymbol{U}_{\frac{\boldsymbol{N}}{\boldsymbol{2}}} \rceil^{T} \end{bmatrix}.\boldsymbol{P}_{\boldsymbol{N}}. \end{aligned} \tag{4.31}$$

where

$$\boldsymbol{U}_{\frac{\boldsymbol{N}}{\boldsymbol{2}}}^{\boldsymbol{T}} = \begin{bmatrix} \boldsymbol{U}_{\frac{\boldsymbol{N}}{\boldsymbol{4}}}^{\boldsymbol{T}} & \boldsymbol{0} \\ \boldsymbol{0} & \boldsymbol{U}_{\frac{\boldsymbol{N}}{\boldsymbol{4}}}^{\boldsymbol{T}} \end{bmatrix}.\boldsymbol{V}_{\frac{\boldsymbol{N}}{\boldsymbol{2}}}^{\boldsymbol{T}}. \tag{4.32}$$

The signal flow graph of N-point 1D-IDCT is shown in Fig. 4.6. The approximate algorithm discussed in Section 4.3.2 can also be applied for IDCT architecture also to curtail the use of rotation units. Therefore, hardware cost and processing time can be reduced.

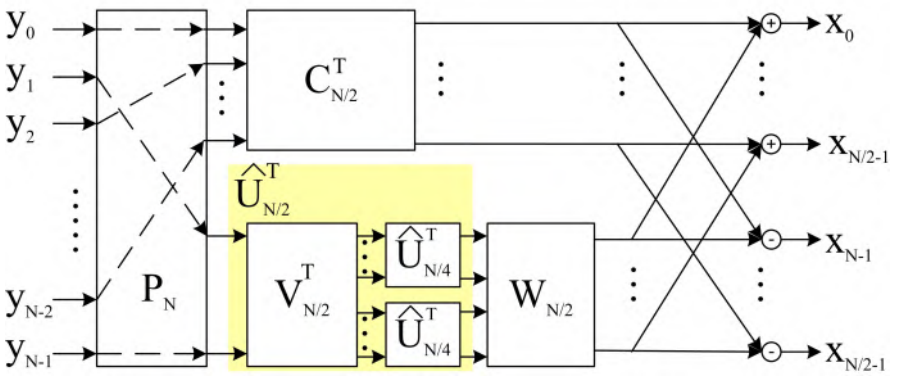


FIGURE 4.6: Signal Flow Graph of 1D IDCT (C_N^T) of order N

4.3.5 Hardware implementation of IDCT architectures

The hardware implementation of 4 and 8 -point IDCT architectures for only CASE-I have been discussed in the following subsections.

4.3.5.1 4-point IDCT

From (4.31) the 4-point IDCT can be written as

$$\boldsymbol{C_4^T} = \begin{bmatrix} \boldsymbol{I_2} & \boldsymbol{I_2} \\ \boldsymbol{J_2} & -\boldsymbol{J_2} \end{bmatrix} . \begin{bmatrix} \boldsymbol{C_2^T} & \boldsymbol{0} \\ \boldsymbol{0} & \boldsymbol{W_2}.\hat{\boldsymbol{U}}_{\boldsymbol{2}}{}^{T} \end{bmatrix} . \boldsymbol{P_4}, \tag{4.33}$$

where

$$\boldsymbol{C_2^T} = \begin{bmatrix} 64 & 64 \\ 64 & -64 \end{bmatrix} \tag{4.34}$$

and

$$\hat{\boldsymbol{U}}_{\boldsymbol{2}}{}^{T} = \lfloor 2^6.\boldsymbol{U_2} \rceil^T = \begin{bmatrix} 2^6.\boldsymbol{U_1}{}^{T} & \boldsymbol{0} \\ \boldsymbol{0} & 2^6.\boldsymbol{U_1}{}^{T} \end{bmatrix} . \boldsymbol{V_2^T}. \tag{4.35}$$

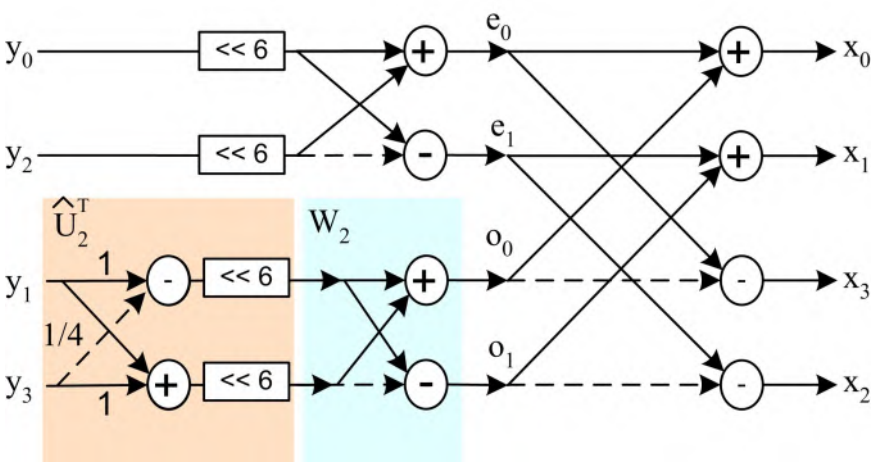


FIGURE 4.7: Architecture of 4-point IDCT (C_4^T)

The proposed approximation algorithm produces V_2 as below.

$$V_2 = \begin{bmatrix} 1 & -\frac{1}{4} \\ \frac{1}{4} & 1 \end{bmatrix}. \tag{4.36}$$

Data flow model of 4-point IDCT is shown in Fig. 4.7.

4.3.5.2 8-point IDCT

Following the similar approach 8-point IDCT can be calculated as

$$\boldsymbol{C_8^T} = \begin{bmatrix} \boldsymbol{I_4} & \boldsymbol{I_4} \\ \boldsymbol{J_4} & -\boldsymbol{J_4} \end{bmatrix}. \begin{bmatrix} \boldsymbol{C_4^T} & \boldsymbol{0} \\ \boldsymbol{0} & \boldsymbol{W_4}.\hat{\boldsymbol{U}}_{\boldsymbol{4}}{}^T \end{bmatrix}. \boldsymbol{P_8}, \tag{4.37}$$

where

$$\hat{\boldsymbol{U}}_{\boldsymbol{4}}{}^T = \lfloor 2^6.\boldsymbol{U_4} \rceil^T = \begin{bmatrix} \hat{\boldsymbol{U}}_{\boldsymbol{2}}{}^T & \boldsymbol{0} \\ \boldsymbol{0} & \hat{\boldsymbol{U}}_{\boldsymbol{2}}{}^T \end{bmatrix}. \boldsymbol{V_4^T}. \tag{4.38}$$

Here, $\boldsymbol{W_4}$ is WHT matrix of order 4 and $\boldsymbol{V_4}$ is obtained from (4.25) by the proposed approximation algorithm as discussed in Section 4.3.2 to avoid the use of rotational

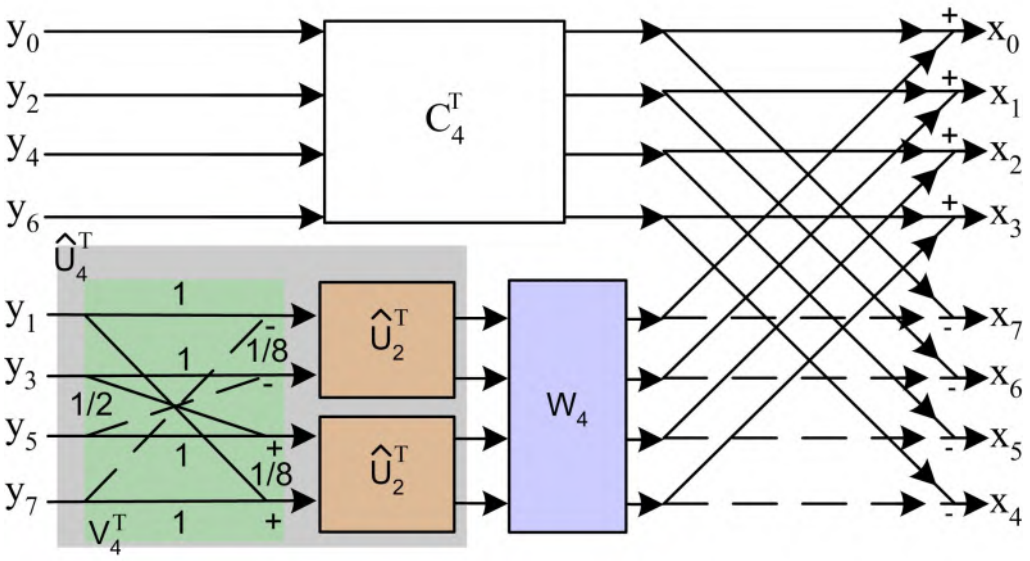


FIGURE 4.8: Architecture of 8-point IDCT (C_8^T)

units. As discussed before, all the elements of the matrix $\boldsymbol{V_4}$ can be realized by right shift operations. Therefore, the need of the multipliers is eliminated. An 8-point IDCT architecture is shown in Fig. 4.8. The proposed design approach can be extended to higher order IDCT modules too.

4.4 Results and Discussion

Approximation reduces the hardware complexity at the cost of reduced coding efficiency. In this section, the coding efficiency as well as hardware efficiency of different approximation techniques are compared. The proposed method is also compared with other existing designs and results are discussed in this section.

TABLE 4.4: Accuracy comparison of 32-point DCT

	MSE ($\times 10^{-2}$)	Coding Gain	Efficiency (%)
DCT	0	9.77	81.7
HM [12]	0.0016	9.77	81.4
JAM [73]	9.95	10.41	59.5
Scalable [74]	10.78	10.1	59.7
Proposed Methods			
CASE-I	4.06	8.95	63.8
CASE-II	1.79	9.44	72.0
CASE-III	1.39	9.44	76.4
CASE-IV	0.92	9.06	79.1

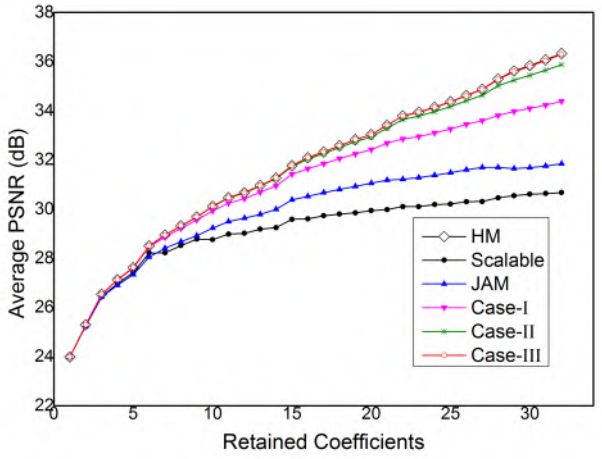
4.4.1 Coding efficiency

The proposed approximation technique produces new DCT and IDCT matrices. In order to evaluate the accuracy of the proposed approximated matrices, we computed the transform related measures like mean square error (MSE), coding gain and efficiency as defined in [20]. As the IDCT matrix produced is transpose of DCT matrix, we performed the computation for DCT matrix only. These parameters are compared with that of the original DCT matrix and the integer DCT used in the reference software. We have also computed those parameters for other two approximated architectures, i.e., architecture proposed by Jridi, Alfalou, Meher (JAM) [73] and the scalable architecture [74]. These results are presented in Table 4.4. To the best of our knowledge, these architectures are also designed to cater the requirements of HEVC. However, they use square wave approximations of different size DCT matrices. Therefore, MSE is very high and at the same time, efficiency is very low as compared to that of the proposed method. It can be observed from Table 4.4 that although the accuracy of integer DCT in [12] is higher, the proximity measures of the proposed matrices are comparable. The MSE is the lowest for Case-IV and the highest for Case-I, as expected. However, the MSE and efficiency measures of the matrix in Case-I is better than that of the JAM [73] and scalable [74] matrices.

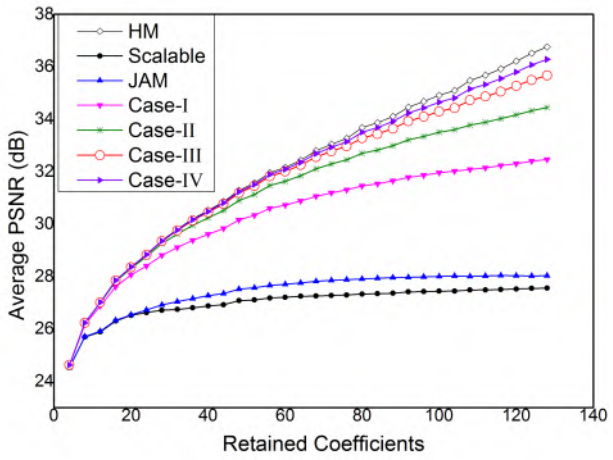
4.4.2 Image coding performance

To further evaluate the energy compaction ability, image compression performance of the proposed approximation methods is measured. Like JPEG, block-based 2-D transform is applied on standard grayscale images (Miscellaneous) of bit depth 8 and size 512×512 obtained from public image bank [113]. The performance was also measured with the block size of 16×16 and 32×32. Thereafter, for each block only initial r-coefficients are retained to reconstruct the images according to zigzag sequence. For the block size of 8×8, we have chosen the minimum and the maximum value of r as r_{min} = 1 and r_{max} = 32, respectively. These values are 4 times and 16 times higher for the block size of 16×16 and 32×32, respectively. Average PSNR vs. retained coefficients curves for different block sizes are shown in Fig. 4.9.

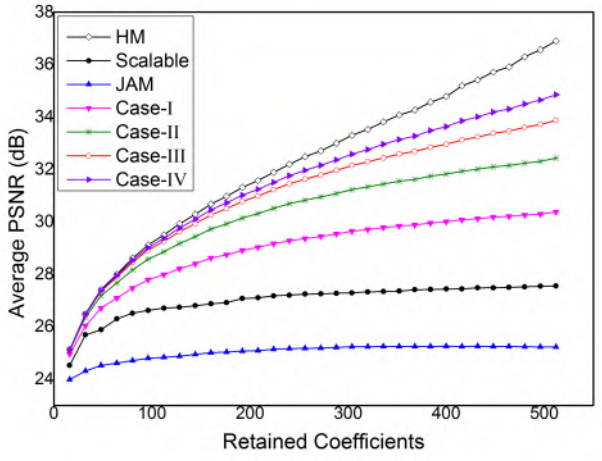
The same experiment is performed on some of the existing methods and their plots are also presented in Fig. 4.9. As the coefficients of DCT and IDCT are same, the experiments with DCT matrix are performed. A conclusion regarding performance of the proposed IDCT matrix can also be drawn from the same set of experiments. Hence, for all the cases inverse transform is calculated by original IDCT with required normalization adjustment. This will include the error due to encoder-decoder coefficient mismatch in the coding performance. Note that, approximated 8-point DCT matrices for Case-III and Case-IV are the same and therefore, have similar image coding performance with block size of 8×8. Hence, the performance of Case-IV is not plotted for the block size of 8×8 in Fig. 4.9(a). It is clear from the figure that for the smaller size block, image coding performance of the proposed approximation scheme is almost similar to that of the performance of the integer DCT in HM reference software [12] and it is better than all other existing approximation schemes. On the other hand, for large block size, the approximated methods follow the performance of the integer DCT for small number of retained coefficients.



(a)



(b)



(c)

FIGURE 4.9: PSNR vs Retained coefficients curves for block size of (a) 8×8 (b) 16×16 and (c) 32×32

As discussed in the previous section, larger size DCTs are applied on homogeneous regions where maximum energy is concentrated in few low-frequency components. Hence, preserving few coefficients is more than sufficient to compress large size block.

4.4.3 Video coding performance

The video coding performance of different DCT architectures is discussed in Section 4.3.2. In this section video coding performance of IDCT architecture for CASE-I is discussed and performance of the remaining cases can be predicted.

The proposed integer IDCT model is integrated in the HEVC reference software [12] and the impact of the proposed approximation is measured. Thereafter, the coding performance is compared with that of the integer IDCT matrix available in the reference software. All the experiments are performed using HM-16.15 with common test conditions as described in [103]. Input sequences from five different classes are used. These are Traffic 2560×1600, PeopleOnStreet 2560×1600, CrowdRun 1920×1080, ParkScene 1920×1080, BasketballDrillText 832×480, BQMall 832×480, BasketballPass 416×240, RaceHorses 416×240, KristenAndSara 1280×720 and FourPeople 1280×720. The standard BD-rate approach is followed [88] to calculate PSNR variations with four different quantization parameters (i.e., 22, 27, 32 and 37). The comparison results in terms of combined PSNR (i.e., YUV-PSNR) variation for AI, LD and RA encoder configurations are presented in Table 4.5. Note that the negative PSNR difference represents coding loss compared to that of the reference software. Table 4.5 shows that the maximum PSNR variation due to this approximation method is less than 0.5 dB. It can be observed that the proposed approximation method works better for rotation unit having small angle of rotations

TABLE 4.5: PSNR variation with respect to the reference algorithm

Class	Sequences	AI	LD	RA
A (WQXGA))	Traffic	-0.350	-	-0.175
	PeopleOnStreet	-0.354	-	-0.192
B (1080P)	CrowdRun	-0.250	-0.133	-0.122
	ParkScene	-0.261	-0.099	-0.126
C (WVGA)	BasketballDrill	-0.179	-0.081	-0.306
	BQMall	-0.481	-0.423	-0.445
D (240i)	BasketballPass	-0.221	-0.130	-0.124
	Racehorses	-0.390	-0.339	-0.466
E (720p)	KristenAndSara	-0.304	-0.227	-
	FourPeople	-0.269	-0.205	-

and therefore, for the large size of IDCT. Hence, a little increase in the accuracy of the lower order rotation matrices can significantly improve coding performance.

4.4.4 Implementation results

4.4.4.1 ASIC implementation

Verilog RTL coding is done for all the approximate DCT architectures as well as the 32-point generalized architecture without any approximation as discussed in Section 4.3. Thereafter, the source codes for all the architectures are synthesized by Synopsys Design Compiler using 90 nm standard cell library [104] to compute the gain in hardware implementation. Detailed synthesis report of these designs and complexity comparison is presented in Table 4.6. Here, gate count has been estimated by normalizing total area with respect to the area of 2-input NAND gate. During row transform, 16 bits input bus width is suggested in [66]. So, all the proposed architectures are implemented with 16 bits input size. For all the cases, registers are inserted at the input as well as at the output stages to constrain the clock period at 4 ns. However, power has been calculated at the fixed clock frequency of 100 MHz.

The proposed approximation method reduces the hardware complexity and it results in power and area reduction. Case-I of the proposed architecture consumes 82% less power, whereas Case-IV consumes 43% less power. In VLSI design area-delay product is an important aspect to compare different architectures. The area-delay product is calculated by multiplying number of gates with processing time. It is observed from Table 4.6 that area-delay product for all the proposed approximated DCT architectures is at least 43% less than that of the architecture without any approximation. It is worth noting that architectures for Case -II and -III maintain good trade-off between coding performance and hardware cost.

We have further compared the implementation results as well as PSNR variation with some of the lifting based [70] [71], MCM based [68] [80], truncation scheme [77] and real-valued[79] DCT architectures. Those designs are intended for HEVC core transform and support 4-, 8-, 16-, 32- order transforms. However, few of them are approximated architectures intended to reduce hardware complexity and power consumption. The PSNR variation for the same sequence (i.e., Traffic) was picked in AI encoder configuration for fair comparison. These comparison results are presented in Table 4.7. Note that, the proposed architectures in Case-III and Case-IV significantly reduce the gate count as well as power consumption with negligible sacrifice in accuracy. PSNR variation of the Case-II is also comparable to that of [71]. But, the gate count is approximately one-third as compared to that of the MCM based, truncation scheme and lifting based approach. The architecture in [79] uses a new set of real-valued coefficients to reduce intermediate data depth as well as overall gate count of the DCT core. But, it does not use any approximation technique to minimize complexity.

Table 4.6: Complexity comparison of different approximated 1D DCT architectures

Architectures		Area (Gate)	Time (ns)	Power (mW)	Area-Delay product ($\times 10^{-4}$)	Area-Delay product reduction	Power Reduction
Without approximation (Original)		111.23 K	4	16.83	4.45	-	-
Approximated	Case-I	20.22 K	4	4.00	0.81	82%	76%
	Case-II	37.85 K	4	5.64	1.51	66%	66%
	Case-III	51.41 K	4	6.87	2.06	54%	59%
	Case-IV	63.40K	4	7.83	2.54	43%	53%

Table 4.7: Comparison of 1D DCT architectures implemented on CMOS technology

		Tech.	Area (Gates)	Freq. (MHz)	Throughput (Gsps)	Power (mW)	PSNR var. (dB)	Architecture type
Lifting based	[71]*	90 nm	163 K	250	3.212	15.5	0.1	Approximated
	[70]	90 nm	144 K	150	0.56	-	-	Without Approx.
MCM based	[68]$	90 nm	131 K	187	2.992	23	-	Without Approx.
	[80]†	65 nm	115.8 K	476	8.01	6	-	Without Approx.
Truncation scheme [77]		90 nm	102 K	187	-	12.33	0.005	Approximated
Real-Valued [79]		90 nm	88.6 K	256.4	4.1	16.2	0.003	Without Approx.
Proposed	Original	90 nm	111.2 K	250	4	16.83	-	Without Approx.
	Case-I		20.2 K			4	0.27	Approximated
	Case-II		37.9 K			5.64	0.15	Approximated
	Case-III		51.4 K			6.87	0.06	Approximated
	Case-IV		63.4 K			7.83	0.03	Approximated

*Unfolded Architecture, $Pruned Architecture, †1-ASU Architecture

In this chapter, we have proposed a new approximation technique using which hardware complexity of the DCT core reduces significantly. Hence, the gate count of each of the proposed approximated architectures is smaller as compared to that of the real-valued DCT architecture [79]. Further, operating frequency of the proposed architectures is comparable to all the architectures implemented using 90 nm technology. Performance of [80] is higher as it is a pipelined architecture and uses register after each addition operation. Additionally, it has been implemented on advanced technology node. This comparison results prove that the proposed method of approximation produces better result than that of the other existing approximation techniques and most of the other architectures implemented without any approximation, as listed in Table 4.7.

We have computed the complexity of the proposed approximated architectures to measure the effect of the approximation technique. Only adders and shifters are used to design all the proposed architectures. But, wire shifting, which does not require any logic resources, is used in these designs. Therefore, the complexity of all the approximated architectures has been computed in terms of number of adders. Module-wise summary of the adders required in all the architectures is presented in Table 4.8. This table shows that the requirement of the adders increases from Case-I to Case-IV, as expected. It is so because the approximation has been applied for all size of DCTs in Case-I, whereas only 32-point DCT has been approximated in Case IV. The complexity comparison of the proposed approximated architectures with that of the MCM based, lifting based and real-valued architectures is presented in Table 4.9. The multiplexers count in each case is also mentioned in this table. As the number of inputs for a particular size of DCT remains the same, the multiplexer count is the same for all the cases. The proposed method of approximation significantly reduces the hardware complexity of the DCT core which is evident from

TABLE 4.8: Adders count in proposed approximated architectures

Modules	Case-I				Case-II				Case-III				Case-IV			
	N=4	N=8	N=16	N=32	N=4	N=8	N=16	N=32	N=4	N=8	N=16	N=32	N=4	N=8	N=16	N=32
IB	4	8	16	24	4	8	16	24	4	8	16	24	4	8	16	24
$\boldsymbol{C}_{\frac{N}{2}}$	2	10	34	98	2	14	46	126	2	14	54	150	2	14	54	166
$\boldsymbol{W}_{\frac{N}{2}}$	2	8	24	24	2	8	24	24	2	8	24	24	2	8	24	24
$\hat{\boldsymbol{U}}_{\frac{N}{2}}$	2	8	24	24	6	16	40	40	6	24	56	56	6	24	72	72
Total	10	34	98	170	14	46	126	214	14	54	150	254	14	54	166	286

TABLE 4.9: Complexity of 1D DCT in different design methodologies

N	MCM based [68]		Lifting based [70]			Real-valued [79]		Proposed							
								Case-I		Case-II		Case-III		Case-IV	
	A	SH	M	A	SH	A	SH	A	MUX	A	MUX	A	MUX	A	MUX
4	14	10	3	11	0	14	10	10	0	14	0	14	0	14	0
8	50	30	15	39	0	54	38	34	68	46	68	54	68	54	68
16	186	86	49	115	2	182	126	98	208	126	208	150	208	166	208
32	682	278	139	307	8	614	334	170	492	214	492	254	492	286	492

M=No. of multipliers; A=No. of adders/subtractor; SH=No. of shifters; MUX= No. of multiplexers

TABLE 4.10: Synthesis results and complexity comparison for 1D DCT architectures implemented on FPGA

Architecture	Size	LUT	SLICE	Time (ns)	Power (mW)	Area-Delay product	Area-Delay product reduction	Power reduction	Mul	Add	Shift
Reference	4	300	84	5.724	10	1717.2	-	-	4	8	2
	8	1341	400	6.801	50	9120.1	-	-	22	28	4
	16	4944	1448	6.84	180	41232.9	-	-	86	100	8
	32	18341	5240	9.808	711	179888.5	-	-	342	372	16
Proposed	4	258	80	4.388	8	1132	34%	20%	-	14	10
	8	900	266	5.996	33	5396.4	41%	34%	-	46	22
	16	2594	748	7.926	107	20560	50%	41%	-	126	54
	32	4507	1301	9.3	210	41915.1	77%	70%	-	214	96

TABLE 4.11: Comparison with other DCT architectures implemented on FPGA platform

	[73]	[74]	[78]	[81]	[82]	Prop.
DCT size	32	32	32	8	32	32
i/p bit-lenght	8	8	8	8	9	8
Tech. (nm)	45	45	45	28	28	28
LUT	1656	1760	1776	21568	5600	2752
DSP Block	-	-	-	-	128	0
Reg/FF	664	-	-	7309	-	0
Speed (MHz)	136	422	422	279	177	125
Through. (per clock)	32	32	32	8	4	32
PSNR var.	0.61	0.34	0.22	-	-	0.15
Transform	1D	1D	1D	1D	2D	1D

Table 4.9.

All the proposed architectures can process a maximum of 32 samples per clock at 250 MHz operating frequency. Therefore, using the proposed architecture, any folded DCT model can calculate 2D DCT of 4 Giga samples per second (Gsps). Hence, for the best case, it can process 321 fps and 80 fps of 4K (3840×2160) and 8K (7680×4320) UHD video, respectively in 4:2:0 YUV format.

4.4.4.2 FPGA Implementation

In order to compare the proposed hardware architecture with the integer DCT matrix of the HEVC reference software [12], the source code of 4, 8, 16 and 32 -point approximate DCT architectures (Case-II) are synthesized using Xilinx Vivado 2016.2 tool and are implemented on Virtex-7 FPGA. We have also implemented the integer DCT architecture published in HEVC reference software [12] on the same platform. Detailed synthesis report of these two designs and their complexity comparison in terms of number of multipliers, adders/subtractors and shifters is presented in Table 4.10. For all the cases, here too, the input bus width is of 16 bits. Registers are inserted at the input as well as at the output stage to measure the maximum frequency of operation. Vector less power calculation is done at the fixed clock frequency of 50 MHz. Since static power, IO power and clock power are not directly related to the complexity of the design; those are excluded during power calculation on FPGA platform. For a given architecture, processing time is calculated by subtracting positive time slack from the constrained time.

The proposed approximation method reduces hardware complexity significantly as

compared to that of the reference architecture. The proposed architecture is multiplier-less and Case-II uses merely 58% of the total adders as compared to that of the reference architecture. The power and area (i.e., LUTs) requirements for the proposed architecture reflect the complexity reduction. The proposed architecture consumes 70% less power, whereas number of LUTs required is less than one-fourth to that of the reference design. At the same time, processing time has also shortened. It can be observed from Table 4.10 that area-delay product for all the proposed N-point DCT architectures is less and 77% reduction with respect to the reference architecture is achieved for 32-point DCT. Note that, the area-delay product is calculated by multiplying number of LUTs with processing time.

To further evaluate the performance of the proposed approximated architectures, we compared the implementation results of the design in Case-II with some of the existing designs. However, the input size of the proposed design is selected to be 8 bits for a fair comparison. These comparison results are presented in Table 4.11. FPGA of same technology node is used in case of designs in [81], [82] and the proposed one. But, the proposed architecture requires less number of hardware resources than those two designs. Although the speed of the proposed architecture is less, its throughput is four and eight times higher as compared to that of the [81] and [82], respectively. Therefore, the proposed architecture can process more number of samples per second as compared to those two architectures. The scalable architecture [74] and the architecture in [78] are pipelined architectures. They use registers after each stage of addition operations to improve operating frequency. The JAM [73], Scalable [74] and the architecture in [78] use square wave approximation of DCT. As a result, the number of LUTs required for those architectures is less. However, the PSNR comparison shows that the accuracy of those architectures is lower as compared to that of the proposed architecture. Additionally, the WHT

architecture in the proposed design can be shared with the prediction unit of the codec. It reduces overall processing time and hardware cost as the precomputed WHT coefficients from the prediction stage can be used to compute DCT. Hence, the proposed architecture provides a good trade-off between the accuracy and hardware cost. Minimum 94 MHz operating frequency is required to process 8K UHD at 30 frames/s in 4:2:0 YUV format [68]. Therefore, the FPGA implementation of the proposed architecture (Case-II) can support 8K video in real-time.

4.5 Summary

This chapter explains the derivation of the approximated forward and inverse core transform matrices. These alterations are made with the intention to reduce the implementation complexity of an HEVC encoder bearing only a minimal tolerance on the coding performance. A new algorithm is proposed to reduce complexity of the matrices by replacing it by dyadic values which are easy to implement in hardware and incur less complexity. The algorithm results minimum approximation errors and it is applied on Walsh–Hadamard Transform (WHT) based DCT and IDCT architectures. To achieve that, two new generalized models of HEVC compliance DCT and IDCT kernel have been proposed using WHT based matrix decomposition method. Thereafter, a few approximated architecture are implemented to provide different trade-offs between coding accuracy and hardware complexity.