Chapter 3

Transform Core in HEVC

The transform operation plays a vital role in hybrid video coding technique. HEVC uses hybrid video coding mechanism and involves variable size higher order DCT for spatial to frequency domain transformation. In this chapter, the forward and inverse transform process used in HEVC has been discussed in detail. Design of different transform architectures and their hardware complexities are analyzed. Finally, an optimized transform architecture is proposed and assessment of the proposed architecture in terms of video coding performance, as well as hardware cost, is presented.

3.1 Introduction

Transform operations are playing a vital role in the field of signal and systems for decades. Sometimes a signal becomes easier to handle in the frequency domain as compared to the spatial domain and hence, a conversion from spatial to frequency domain is performed by forward transform operation. On the other hand, the inverse transform performs frequency to spatial domain conversion to recover the original

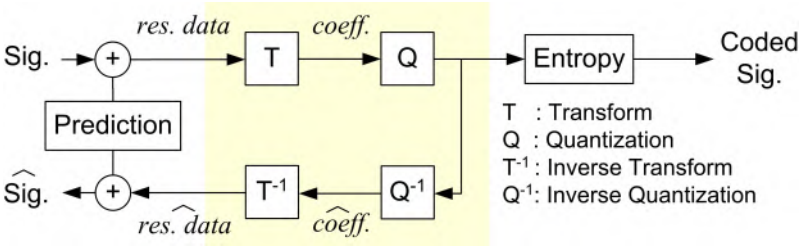

FIGURE 3.1: Typical dataflow in standard video codec

signal. Like HEVC, most of the standard video codecs follow the hybrid video coding scheme [92] which uses transform operation to achieve high compression efficiency. It has been observed that one of the reasons behind the high compression efficiency of HEVC is its large and flexible transform coding ability [13].

3.2 Transforms in HEVC

In recent years, every standard video codec employs prediction as well as transform operations, as shown in Fig. 3.1. The prediction operations remove the spatial and temporal correlations in a video sequence. A residual signal is generated by subtracting the prediction signal from the original signal. The residual signal can be further decorrelated by applying transform operation. As the HVS is not sensitive to high frequency components, those are quantized and the coefficients are entropy coded into bit-stream.

It is expected that 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 have been introduced in the HEVC. One of them is large and variable size TU. TU is a primary unit in HEVC wherein residual data are transformed and quantized. The size of the TU varies from 4×4 to 32×32 to

support variable size transforms. A transform operation is performed in each size of TU to obtain energy compaction. But, the effective size of TU is determined during RDO operation [94].

The Karhunen-Loàve Transform (KLT) [102] is the first choice for transform operation as it has the optimum energy compaction and the signal decorrelation ability [27]. Despite of optimum performance of KLT, it has some limitations and it is very hard to implement for real-time applications [92]. As a result, it is rarely used in practical applications like image and video coding algorithms. For practical and real-time applications, a transform must have fast computational algorithm so that an efficient VLSI architecture implementation is possible. Moreover, the transform must have energy compaction and the signal decorrelation ability as close to the KLT. It has been observed that a family of the sinusoidal transforms have almost similar energy compaction and decorrelation capabilities as that of the KLT [102]. This family comprises of different types (i.e., type I–VIII) DCT and DST. Although all of them are originated from sine and cosine functions, their basis vectors are different from each other. Further details of these transforms are available in [20].

The HEVC draft has adopted two different transform operations: a core transform and an alternative transform [11]. The core transform is an integer approximation of DCT type II and III, whereas alternative transform is integer approximation of DST type VI and VII. The alternative transform is applied to only intra-predicted residual block of size 4×4 for luma component. For all the remaining TUs, the core transform is applied.

3.2.1 DCT and IDCT

Since the last few decades, DCT is playing a key role in image and video compressions. Its strong energy compaction capability and decorrelation property make it popular in most of the image and video coding tools such as JPEG, MPEG, H.261, H.263, H.264/AVC and the latest HEVC. To handle the growing demand of high resolution multimedia services, efficient DCT computation and its VLSI implementation is a major research topic in the field of circuits and systems.

In all standard video coding specifications which are based on the hybrid video coding scheme, the DCT-II and DCT-III transform pair is used for transformation of the prediction error, where DCT-II is the forward and DCT-III is the corresponding inverse transform. Since this is the most commonly used DCT pair, it is often referred to as the DCT and IDCT, respectively.

In general, 2D DCT is used for energy compaction in block based video coding process, including HEVC. The equation for a 2D DCT of block size $M \times N$ is defined as [102]:

$$F(u,v) = \sum_{m=0}^{M-1} \sum_{n=0}^{N-1} \sqrt{\frac{k_m}{M}} \sqrt{\frac{k_n}{N}} \cos\left[\frac{\pi(2m+1)u}{2M}\right] \cos\left[\frac{\pi(2n+1)v}{2N}\right] f(m,n), \quad (3.1)$$

where

$$k_p = \begin{cases} \frac{1}{\sqrt{2}}, & \text{if } p = 0 \\ 1, & \text{otherwise.} \end{cases} \quad (3.2)$$

Here, $f(m,n)$ and $F(u,v)$ represent the $(m,n)^{th}$ and $(u,v)^{th}$ elements of the input and output blocks, respectively.

The hardware design for direct computation of (3.1) requires very complex architecture and may not be economical from the perspective of hardware resource requirement. Although those designs may have very high performance ability, but at the same time they demand high hardware cost, power consumption and greater design efforts.

The alternate design approach takes the advantage of the separable property of 2D DCT. Using this property of DCT, (3.1) can be written as

$$F(u,v) = \sum_{m=0}^{M-1} \sqrt{\frac{k_m}{M}} \cos\left[\frac{\pi(2m+1)u}{2M}\right] \hat{f}(m,v), \tag{3.3}$$

where

$$\hat{f}(m,v) = \sum_{n=0}^{N-1} \sqrt{\frac{k_n}{N}} \cos\left[\frac{\pi(2n+1)v}{2N}\right] f(m,n). \tag{3.4}$$

Note that, (3.3) and (3.4) represent row-wise and column-wise 1D transform of the block, respectively. It shows that the 2D DCT of a block can be calculated by performing 1D column and row transform consecutively. Two different types of architecture are possible based on the separable property of 2D DCT. These are as follows:

1. **Full-parallel Architecture**

   A full-parallel 2D DCT architecture is shown in Fig. 3.2 which uses two 1D

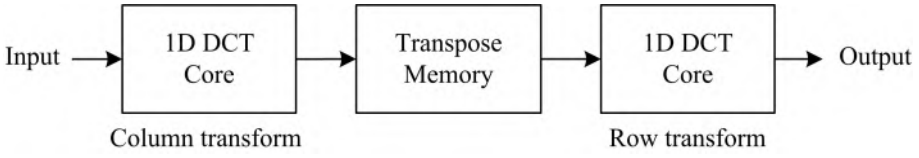

FIGURE 3.2: Block diagram of full-parallel architecture

DCT cores [68]. The first DCT core performs a column-wise transform operation of the input TU, while the second core furnishes a row-wise transformation of the intermediate results. After the column transform, the intermediate results are stored in a Transpose Memory (TM) in matrix form. Such TM block performs transposition of the block obtained after the column-transformed operation and ensures data ordering as required by the row transform operation.

2. **Folded Architecture**

   The most popular architecture for 2D DCT computation is folded architecture [68]. Fig. 3.3 shows the block diagram for a folded architecture in which a single 1D DCT core is used to compute column as well as row transform. The input data are fed to the DCT core from either external circuit or TM block during column or row transform, respectively. The folded architecture is efficient than that of the full-parallel architecture in terms of area required to implement the design. However, the throughput of the folded architecture is less as compared to full-parallel architecture.

As per the draft, all the TUs used in HEVC are square in shape [11]. For the square shaped (i.e., $M = N$) blocks, (3.3) and (3.4) become identical and can be expressed

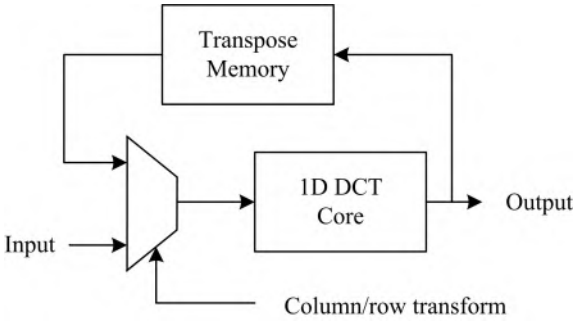

FIGURE 3.3: Block diagram of folded architecture

as

$$y_i = \sum_{j=0}^{N-1} x_j d_{i,j}^N . \tag{3.5}$$

Here, $\boldsymbol{y} = [y_0, y_1, \ldots, y_{N-1}]^T$ represents the N-point 1D DCT outputs of input samples $\boldsymbol{x} = [x_0, x_1, \ldots, x_{N-1}]^T$ and $d_{i,j}^N$ represents different DCT coefficients for $i, j = 0, 1, \ldots, N-1$. The coefficient $d_{i,j}^N$ can be defined as

$$d_{i,j}^N = \sqrt{\frac{k}{N}} \cos\left[\frac{\pi(2j+1)i}{2N}\right], \tag{3.6}$$

where $k = 1$ and 2 for $i = 0$ and $i > 0$, respectively. In terms of matrix product, (3.5) can be written as

$$\boldsymbol{y} = \boldsymbol{D_N}.\boldsymbol{x}, \tag{3.7}$$

where $\boldsymbol{D_N}$ represents N-point DCT coefficient matrix of size $N \times N$. A DCT coefficient matrix of size 4×4 is shown in (3.8).

$$\boldsymbol{D_4} = \frac{1}{\sqrt{4}} \begin{bmatrix} 1 & 1 & 1 & 1 \\ \sqrt{2}\cos(\frac{\pi}{8}) & \sqrt{2}\cos(\frac{3\pi}{8}) & -\sqrt{2}\cos(\frac{3\pi}{8}) & -\sqrt{2}\cos(\frac{\pi}{8}) \\ 1 & -1 & -1 & 1 \\ \sqrt{2}\cos(\frac{3\pi}{8}) & -\sqrt{2}\cos(\frac{\pi}{8}) & \sqrt{2}\cos(\frac{\pi}{8}) & -\sqrt{2}\cos(\frac{3\pi}{8}) \end{bmatrix} \tag{3.8}$$

The equation for 2D IDCT shares the same separable property with that of the DCT. Hence, 2D IDCT can also be computed by row-column decomposition method. Like DCT, the 1D IDCT can be computed as

$$x_j = \sum_{i=0}^{N-1} y_i d_{i,j}^N , \tag{3.9}$$

where $d_{i,j}^{N}$ is the same coefficient as defined in (3.6). Therefore, in terms of matrix product, the IDCT can be written as

$$\boldsymbol{x} = \boldsymbol{D_N}^{-1}.\boldsymbol{y} = \boldsymbol{D_N}^{T}.\boldsymbol{y}\,. \tag{3.10}$$

It is clear from (3.7) and (3.10) that DCT and IDCT matrices are the same but they are transpose of each other. Therefore, they share same set DCT matrix properties as described below [66].

1. The coefficients $\boldsymbol{d_i} = \{d_{i0}, d_{i1}, \cdots, d_{i(N-1)}\}$ in the matrix $\boldsymbol{D_N}$ form the basis vector. The basis vectors are orthonormal to each other, i.e.

$$\begin{aligned} \boldsymbol{d_i}^{T}.\boldsymbol{d_j} &= 0 \quad for \quad i \neq j; \\ &= 1 \quad otherwise. \end{aligned} \tag{3.11}$$

   Since basis vectors are orthonormal, good compression efficiency as well as energy compaction can be achieved using DCT.

2. The even basis vectors (i.e., for $i = 0, 2, \cdots, N-2$) are symmetric, whereas all the odd basis vectors (i.e., for $i = 1, 3, \cdots, N-1$) are anti-symmetric. This property is used to reduce the number of arithmetic operations performed during DCT and IDCT computation.

3. It can be observed that the coefficients of lower order DCT/IDCT matrix are the subset of the coefficients in higher order DCT/IDCT matrix. For example, coefficients in 4×4 matrix are the subset of coefficients in 8×8 matrix. Similarly, coefficients in 8×8 matrix are subset of coefficients in 16×16 matrix and so on.

Due to this property, hardware resource sharing is possible during different size of DCT computation in HEVC.

4. All the elements in DCT/IDCT matrix (D_N/D_N^T) have small number of distinct values. The N-point coefficient matrix D_N of size $N \times N$ consists of $2^m - 1$ number of unique coefficients with different magnitudes, where $m = log_2 N$.

5. Other than the symmetry and anti-symmetry properties, the coefficients of a DCT matrix follow some trigonometric relationships which can be further exploited to reduce arithmetic operations. These properties can also be utilized to implement fast algorithms such as Chen's fast factorization [38].

3.3 Integer transform in HEVC

DCT and IDCT are the most popular transform used in image and video coding standards including JPEG, MPEG-2 and H.263 due to its high energy compaction ability and efficient computational algorithm. However, it is clear from the discussion in the previous section that DCT, as well as IDCT, are real-valued transforms and very high precision is required during computation. This is the major concern during the hardware implementation of DCT and IDCT. It requires high precision floating point multipliers, which incur high hardware cost. Additionally, high precision operations can cause encoder-decoder mismatch and drifting error [66]. To minimize this drift between encoder and decoder implementations and to reduce hardware cost, HEVC has adopted integer approximation of DCT and IDCT matrices [11]. In integer transform, all the real-valued DCT coefficients are scaled by a large number and then rounded to the nearest integer. Therefore, integer transform

matrix C_N can be written as

$$\boldsymbol{C_N} = \lfloor const. \times \boldsymbol{D_N} \rceil, \tag{3.12}$$

where $\lfloor * \rceil$ represents rounding operation. This type of integer DCT has almost similar coding performance as that of the real-valued DCT and its hardware implementation requires less number of computations [66].

Prior to HEVC, H.264/AVC also used integer transforms of size 4×4 and 8×8 [67]. But, HEVC included additional larger transforms of size 16×16 and 32×32. In the case of higher order transforms, the approximation error in the frequency domain becomes considerable [20]. To address this issue, a large scaling factor is selected so that it maintains the following specifications [66]:

- Closeness to original DCT matrix: The properties and performance of the derived integer matrix must be as close as possible to that of the original DCT/IDCT matrix. The closeness of integer matrix is measured using following equation:

$$m_{ij} = |const \times d_{ij} - c_{ij}|/c_{00} \tag{3.13}$$

  The c_{ij} is $(i,j)^{th}$ element of the integer matrix $\boldsymbol{C_N}$.

- Almost orthogonal basis vectors: Orthogonal property of basis vectors is the key behind energy compaction ability. The basis vectors in the derived integer matrix must also maintain this property as far as possible. The orthogonality measure of the integer transform can be measured as:

$$o_{ij} = \boldsymbol{c_i c_j}^T / \boldsymbol{c_0 c_0}^T, \quad for \quad i \neq j. \tag{3.14}$$

Here, $\boldsymbol{c_k} = [c_{k0}, c_{k1}, \cdots, c_{k(N-1)}]$ are the basis vectors of the integer matrix for $k = 0, 1, \cdots, N-1$.

- Almost equal norm for basis vectors: DCT matrix is an orthonormal matrix. Therefore, derived basis vector must maintain near about norm so that a normalizing factor can be used to recover the signal. This property is necessary to maintain a simple quantization and dequantization methods. The norm of the derived integer matrix is measured as follows:

$$n_i = |1 - \boldsymbol{c_i}\boldsymbol{c_i}^T / \boldsymbol{c_0}\boldsymbol{c_0}^T|. \tag{3.15}$$

  Norm measure shown in (3.15) reflects the actual deviation of the derived basis vector from the equal norm property.

- The coefficients of the integer transform matrix are selected such that they can be represented using 8 bits including sign bit.

- Multipliers can be represented using 16 bits or less with no cascaded multiplications or intermediate rounding. The accumulators, if used, can be implemented using less than 32 bits.

- Coefficients of lower order DCT/IDCT matrix are the subset of the coefficients in higher order DCT/IDCT matrix. This property must be maintained by derived integer transform so that a single core can compute all size of transforms used in HEVC.

To satisfy all the aforesaid conditions, scaling factor for N-point transform is selected as:

$$const = 2^{6+\frac{m}{2}}, \quad where \quad m = log_2 N. \tag{3.16}$$

To achieve an acceptable balance between few DCT properties, hand-tuning is also performed to some of the scaled and rounded elements of the integer matrix. The detailed description of the hand-tuning method can be found in [66]. After using (3.12), (3.16) and hand-tuning operations, each DCT coefficient holds integer value. For example, the integer matrix C_4, which is a scaled, rounded and hand-tuned version of D_4 in (3.8), can be written as

$$C_4 = \begin{bmatrix} 64 & 64 & 64 & 64 \\ 83 & 36 & -36 & -83 \\ 64 & -64 & -64 & 64 \\ 36 & -83 & 83 & -36 \end{bmatrix}. \tag{3.17}$$

Note that, C_4 has only three unique coefficients, which is one of the properties of original DCT/IDCT matrix. Therefore, C_4 can also be written as

$$C_4 = \begin{bmatrix} \gamma_0 & \gamma_0 & \gamma_0 & \gamma_0 \\ \alpha_0 & \alpha_1 & -\alpha_1 & -\alpha_0 \\ \gamma_0 & -\gamma_0 & -\gamma_0 & \gamma_0 \\ \alpha_1 & -\alpha_0 & \alpha_0 & -\alpha_1 \end{bmatrix}, \tag{3.18}$$

where $\gamma_0 = 64$, $\alpha_0 = 83$ and $\alpha_1 = 36$. Similarly, C_8 has seven unique coefficients and can be written as

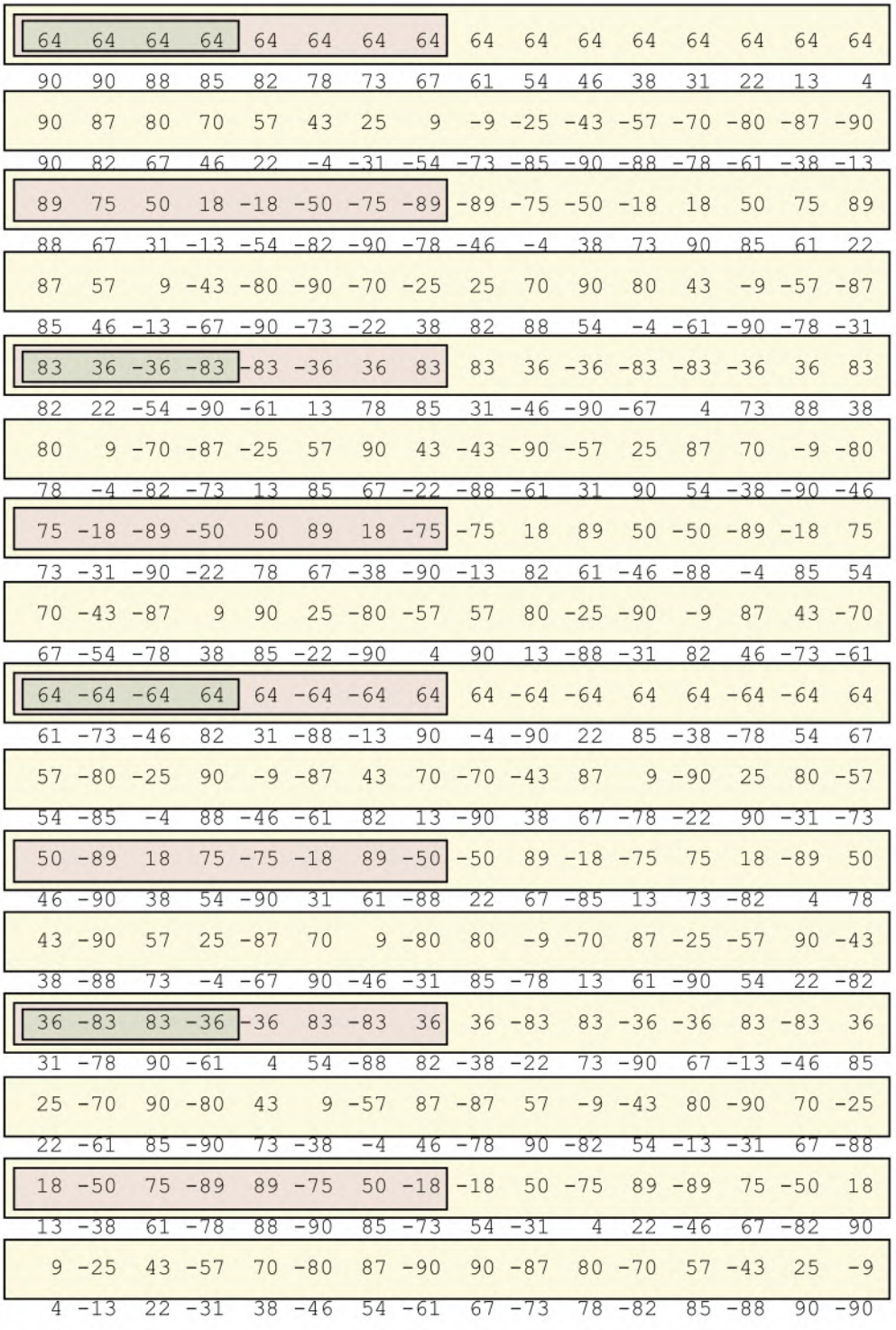

FIGURE 3.4: Left half of 32-point integer matrix (C_{32}) with embedded 16-point (C_{16}) (yellow shading), 8-point (C_8) (pink shading) and 4-point (C_4) (green shading) matrices [66]

$$C_8 = \begin{bmatrix} \gamma_0 & \gamma_0 & \gamma_0 & \gamma_0 & \gamma_0 & \gamma_0 & \gamma_0 & \gamma_0 \\ \beta_0 & \beta_1 & \beta_2 & \beta_3 & -\beta_3 & -\beta_2 & -\beta_1 & -\beta_0 \\ \alpha_0 & \alpha_1 & -\alpha_1 & -\alpha_0 & -\alpha_0 & -\alpha_1 & \alpha_1 & \alpha_0 \\ \beta_1 & -\beta_3 & -\beta_0 & -\beta_2 & \beta_2 & \beta_0 & \beta_3 & -\beta_1 \\ \gamma_0 & -\gamma_0 & -\gamma_0 & \gamma_0 & \gamma_0 & -\gamma_0 & -\gamma_0 & \gamma_0 \\ \beta_2 & -\beta_0 & \beta_3 & \beta_1 & -\beta_1 & -\beta_3 & \beta_0 & -\beta_2 \\ \alpha_1 & -\alpha_0 & \alpha_0 & -\alpha_1 & -\alpha_1 & \alpha_0 & -\alpha_0 & \alpha_1 \\ \beta_3 & -\beta_2 & \beta_1 & -\beta_0 & \beta_0 & -\beta_1 & \beta_2 & -\beta_3 \end{bmatrix}, \tag{3.19}$$

where β_0 = 89, β_1 = 75, β_2 = 50 and β_3 = 18. It is clear from (3.18) and (3.19) that the coefficients of 4-point integer transform (C_4) are embedded in 8-point integer transform (C_8). Similarly, the coefficients of 8-point are embedded in 16-point and the coefficients of 16-point are embedded in 32-point transform. Therefore, all the coefficients of 4, 8 and 16 -point transforms are embedded in a 32-point integer transform matrix, as shown in Fig. 3.4. In this figure, only left half of the 32-point integer matrix is shown as the size is very large and the missing right half of the even and odd rows can be derived by symmetry and anti-symmetry properties, respectively. All the embedded coefficients of 4-, 8-, and 16- point matrices are shown by green, pink and yellow shadings, respectively.

The coding performance of the integer transform matrix is approximately similar because it almost holds all the properties of DCT/IDCT matrix. Table 3.1 shows the closeness (m_{ij}), orthogonality (o_{ij}) and normality (n_i) measures of integer transform matrices as per (3.13), (3.14), (3.15), respectively. It is clear from Table 3.1 that the deviation from orthogonality and normality property for the derived integer transform is very less and it closely approximate the original DCT/IDCT matrix.

TABLE 3.1: Performance measures of the integer transform matrices [92]

Measures	4-point	8-point	16-point	32-point
m_{ij}	<0.0213	<0.0213	<0.0213	<0.0213
o_{ij}	0	<0.0016	<0.0029	<0.0029
n_i	<0.0009	<0.0009	<0.0009	<0.0013

3.3.1 Hardware implementation and complexity analysis

The finite precision coefficients eliminate the use of floating point multipliers in integer transform. Therefore, hardware implementation of integer transform is much efficient as compared to that of the original DCT/IDCT. The straight forward implementation of 1D integer transform performs a matrix multiplication operation as shown in (3.7). Therefore, for an N-point input vector, the number of arithmetic operations for a 1D forward/inverse transform via direct matrix multiplication is N^2 multiplications and $N(N–1)$ additions (including subtractions). For an $N \times N$ input block, the complexity of a 1D transform becomes N^3 multiplications and $N^2(N–1)$ additions/subtractions. The separable property of DCT enables a 2D transform to be implemented via two 1D transforms with a transpose operation between them. Thus, for a 2D transform of an $N \times N$ input block, the complexity is $2N^3$ multiplications and $2N^2(N–1)$ additions/subtractions. However, transform matrix can be decomposed using symmetry or anti-symmetry property of the basis vectors, which significantly reduces the number of arithmetic operations. For example, using (3.7) and (3.19), 8-point 1D forward transform can be calculated as

$$\begin{bmatrix} y_0 \\ y_1 \\ \vdots \\ y_7 \end{bmatrix} = \boldsymbol{C_8}. \begin{bmatrix} x_0 \\ x_1 \\ \vdots \\ x_7 \end{bmatrix} = \begin{bmatrix} \gamma_0 & \gamma_0 & \gamma_0 & \gamma_0 & \gamma_0 & \gamma_0 & \gamma_0 & \gamma_0 \\ \beta_0 & \beta_1 & \beta_2 & \beta_3 & -\beta_3 & -\beta_2 & -\beta_1 & -\beta_0 \\ \alpha_0 & \alpha_1 & -\alpha_1 & -\alpha_0 & -\alpha_0 & -\alpha_1 & \alpha_1 & \alpha_0 \\ \beta_1 & -\beta_3 & -\beta_0 & -\beta_2 & \beta_2 & \beta_0 & \beta_3 & -\beta_1 \\ \gamma_0 & -\gamma_0 & -\gamma_0 & \gamma_0 & \gamma_0 & -\gamma_0 & -\gamma_0 & \gamma_0 \\ \beta_2 & -\beta_0 & \beta_3 & \beta_1 & -\beta_1 & -\beta_3 & \beta_0 & -\beta_2 \\ \alpha_1 & -\alpha_0 & \alpha_0 & -\alpha_1 & -\alpha_1 & \alpha_0 & -\alpha_0 & \alpha_1 \\ \beta_3 & -\beta_2 & \beta_1 & -\beta_0 & \beta_0 & -\beta_1 & \beta_2 & -\beta_3 \end{bmatrix} \begin{bmatrix} x_0 \\ x_1 \\ \vdots \\ x_7 \end{bmatrix}. \tag{3.20}$$

Here, $[y_0, y_1, \ldots, y_7]^T$ represents the N-point 1D DCT outputs of input samples $[x_0, x_1, \ldots, x_7]^T$. Using the symmetry and anti-symmetry properties, (3.20) can be decomposed into even and odd parts as shown below

$$\begin{bmatrix} y_0 \\ y_2 \\ y_4 \\ y_6 \end{bmatrix} = \begin{bmatrix} \gamma_0 & \gamma_0 & \gamma_0 & \gamma_0 \\ \alpha_0 & \alpha_1 & -\alpha_1 & -\alpha_0 \\ \gamma_0 & -\gamma_0 & -\gamma_0 & \gamma_0 \\ \alpha_1 & -\alpha_0 & \alpha_0 & -\alpha_1 \end{bmatrix} \begin{bmatrix} x_0 + x_7 \\ x_1 + x_6 \\ x_2 + x_5 \\ x_3 + x_4 \end{bmatrix} = \boldsymbol{C_4}. \begin{bmatrix} x_0 + x_7 \\ x_1 + x_6 \\ x_2 + x_5 \\ x_3 + x_4 \end{bmatrix}; \tag{3.21}$$

$$\begin{bmatrix} y_1 \\ y_3 \\ y_5 \\ y_7 \end{bmatrix} = \begin{bmatrix} \beta_0 & \beta_1 & \beta_2 & \beta_3 \\ \beta_1 & -\beta_3 & -\beta_0 & -\beta_2 \\ \beta_2 & -\beta_0 & \beta_3 & \beta_1 \\ \beta_3 & -\beta_2 & \beta_1 & -\beta_0 \end{bmatrix} \begin{bmatrix} x_0 - x_7 \\ x_1 - x_6 \\ x_2 - x_5 \\ x_3 - x_4 \end{bmatrix} = \boldsymbol{B_4}. \begin{bmatrix} x_0 - x_7 \\ x_1 - x_6 \\ x_2 - x_5 \\ x_3 - x_4 \end{bmatrix}. \tag{3.22}$$

The odd-even decomposition method for the 8-point transform can be extended to any N-point transform as follows:

$$\begin{bmatrix} y_0^N \\ y_2^N \\ \vdots \\ y_{N-2}^N \end{bmatrix} = \boldsymbol{C}_{\frac{N}{2}} \begin{bmatrix} a_0 \\ a_1 \\ \vdots \\ a_{N/2-1} \end{bmatrix}, \tag{3.23}$$

$$\begin{bmatrix} y_1^N \\ y_3^N \\ \vdots \\ y_{N-1}^N \end{bmatrix} = \boldsymbol{B}_{\frac{N}{2}} \begin{bmatrix} b_0 \\ b_1 \\ \vdots \\ b_{N/2-1} \end{bmatrix}, \tag{3.24}$$

where

$$\begin{aligned} a_n &= x_n + x_{N-n-1}, \\ b_n &= x_n - x_{N-n-1}. \end{aligned} \tag{3.25}$$

$B_{\frac{N}{2}}$ is a matrix of size $\frac{N}{2} \times \frac{N}{2}$ for odd coefficient calculation and its coefficients at (i, j)th location can be expressed as

$$b_{i,j}^{\frac{N}{2}} = c_{2i+1,j}^N \quad \text{for} \quad 0 \le i, j \le (\frac{N}{2} - 1) \tag{3.26}$$

where $c_{2i+1,j}^N$ is $(2i+1, j)$th coefficient in matrix C_N.

The odd-even decomposition method significantly reduces the complexity of the transform operation. In general, for the two-dimensional N-point transform operation, the number of multiplications and additions (excluding the rounding operations associated with the shift operations) required can be given by [66]

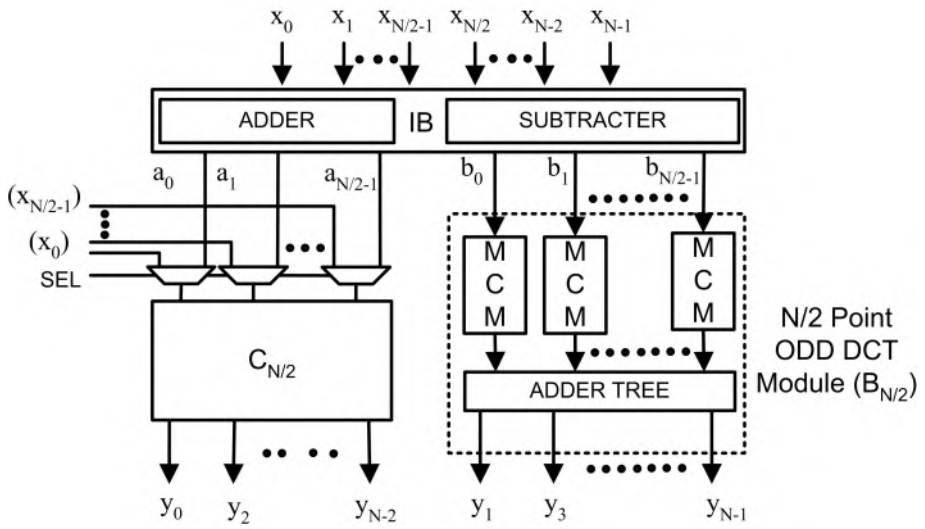

FIGURE 3.5: A generalized hardware model for N-point 1D integer transform computation

$$O_{mult} = 2N\left(1 + \sum_{k=1}^{log_2 N} 2^{2k-2}\right), \tag{3.27}$$

$$O_{add} = 2N\left(\sum_{k=1}^{log_2 N} 2^{k-1}(2^{k-1}+1)\right). \tag{3.28}$$

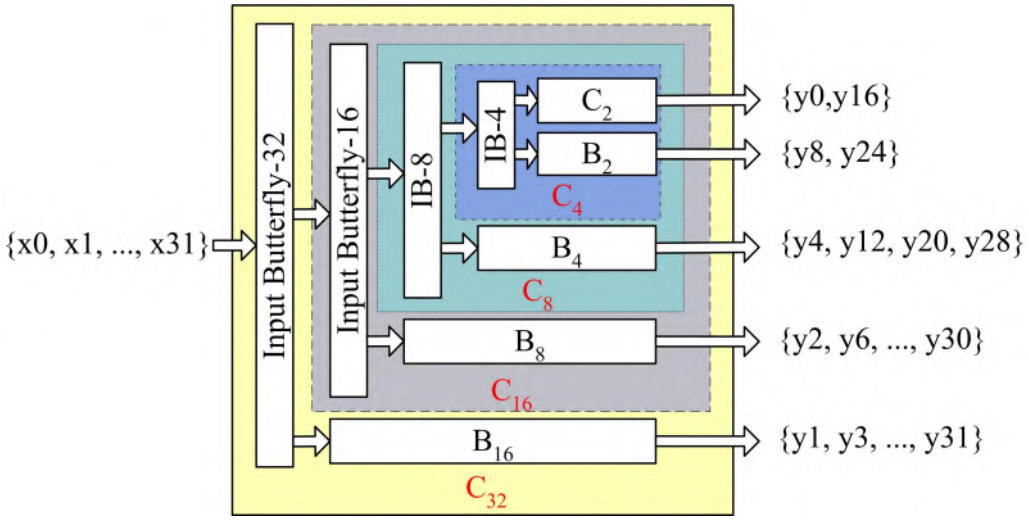

FIGURE 3.6: Hardware sharing model of forward transform used in HEVC [66]

Odd-even decomposition method is one of the most popular methods used to implement integer transform in hardware. An architecture for N-point integer transform is shown in Fig. 3.5 which uses odd-even decomposition method. Hardware realization of C_N requires an Input Butterfly (IB) unit and other two units: one to produce even coefficients ($C_{\frac{N}{2}}$) and other for odd coefficients ($B_{\frac{N}{2}}$, as shown in the dotted box). From the hardware implementation perspective, a multiplication is considered as an expensive operation because it requires a large number of physical resources and chip area, especially while implementing large algorithms such as the 2D 32×32 HEVC core transform. Hence, all the multiplications are realized by shift and add operations to avoid the use of multipliers. In integer transform computation, a single input is multiplied by different coefficients which are fixed. MCM blocks are used to realize these constant multiplications. MCM is very popular for constants multiplication and further details on MCM can be found in [69]. Such $\frac{N}{2}$ number of MCMs and output adder trees are used to realize the module $B_{\frac{N}{2}}$. The $C_{\frac{N}{2}}$ unit can also be used to compute $\frac{N}{2}$-point DCT. At this time, inputs (shown in brackets) must be fed directly to $C_{\frac{N}{2}}$ unit. Therefore, it is clear from Fig. 3.5 that the same architecture can compute either N- or $\frac{N}{2}$- point integer transform depending on the input signal 'SEL.' Thus, a single kernel of 32-point transform (C_{32}) consists of C_{16}, C_8, C_4 modules and hence, it can compute 4, 8, 16, 32 -point transforms depending on the three select lines 'SEL.' Fig. 3.6 depicts a typical hardware resource sharing model used in HEVC core transform. Multiplexers are not shown in the figure to avoid clatter.

3.3.2 Dynamic range and scaling

Finite precision integer transform has huge advantages over real-valued DCT/IDCT. It not only reduces hardware complexity and drifting error, but also a single 32-point transform core can compute transform operations of all sizes used in HEVC. Additionally, multiplications with integer values can be achieved by add and shift operations and thus, use of multipliers is eliminated during hardware realization. However, one can notice that the norm of the basis vectors varies with the transforms size. As a consequence, different quantization and de-quantization matrices are required during different size transform computation. Implementation cost of these non-flat quantization and de-quantization matrices is very high.

Another disadvantage of integer transform is its high dynamic range. As the integer transform is derived by multiplying a large constant with the DCT/IDCT matrix, it is obvious that a large number of bits are required to represent each of the outputs [11]. Therefore, size of the TM required to store these outputs as well as bit-depth of the intermediate data buses are very high. These all result increased chip area and hardware cost.

In order to maintain a reasonable trade-off between accuracy and computational complexity in the transform stage of HEVC, it was decided to limit the bit depth of the coefficients after each transform stage as 16-bit signed integers, i.e., in the range of $[-2^{15}, 2^{15}-1]$ or $[-32768, 32767]$ for any input of bit depth b [66]. Additional intermediate scaling of S_{T1}, S_{T2}, S_{IT1}, and S_{IT2} bits, as shown in Fig. 3.7, are essential to achieve this requirement.

Residual data are generated after the prediction operation. The magnitude of residual data varies from -2^b to $2^b - 1$, where b represents the bit depth of the video sequence. Clearly, $(b+1)$ bits are required to represent the residual data, including

the sign bit. At any instance, 2^m number of such residual data are fed to the column transform (DCT-I) operation. Therefore, the maximum absolute value of the output samples after the column transform will be $2^6 \times 2^m \times 2^b$ which requires $(b + m + 7)$ bits, including sign bit. It requires $S_{T1} = (b + m + 7 - b_t)$ bits truncation to store transformed data in a b_t-bit transpose memory. Similarly, $(b_t + m + 6)$ bits are required to represent a sample of output data after row transform (DCT-II) operation. Therefore, the amount of truncation required is $S_{T2} = (m + 6)$ bits to store the data in the same memory.

In the case of 2D-IDCT, S_{IT1} and S_{IT2} represent bit truncations required after the first and the second inverse transform, respectively. It requires $(b_t + 6)$ bits to represent the maximum value after the first IDCT and $S_{IT1} = 6$ bits truncation is required to store it in TM. On the other hand, these values are $(b_t + 6)$ and $S_{IT1} = b_t + 5 - b$ bits, respectively after the second IDCT.

3.4 Tranform matrix with real-valued coefficients

Integer transform increases datapath length and the circuit delay, thereby limiting the maximum speed at which a DCT block can operate. Additionally, it increases

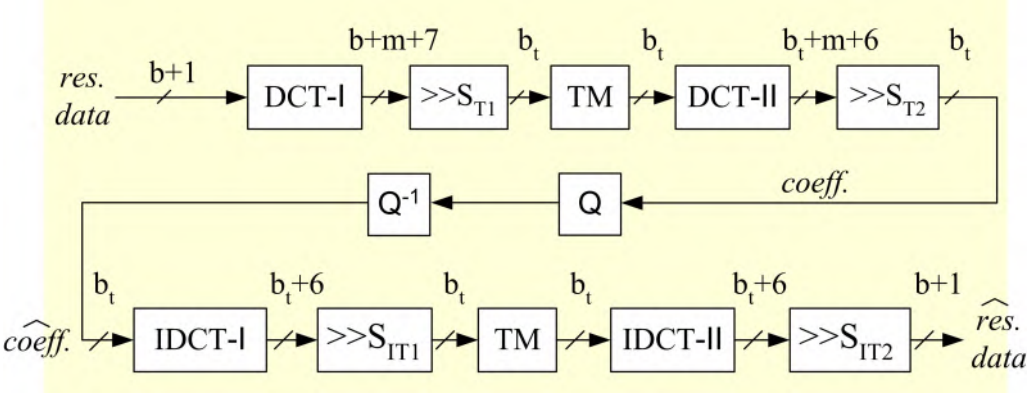

FIGURE 3.7: Datapath details of forward and inverse transforms chain

the hardware resource requirement. For example, the size of the TM used to store intermediate results after the first transform, is large which can incur greater hardware cost in terms of area and power. To maintain a reasonable trade-off between accuracy and implementation cost, outputs are truncated after each transform and stored in a TM of bit-depth 16. Consequently, there is a scope to redesign the transform kernel such that the intermediate data depth is reduced. However, maintaining DCT properties and accuracy up to a certain extent are the major challenges.

To address these issues, a transform architecture with a new set of fixed point coefficients is proposed in this section so that truncation error does not affect the performance of the transform core significantly. The magnitude of each coefficient is kept as small as possible such that intermediate data path gets optimized and operating speed of the transform block is maximized. A reasonable trade-off between computational accuracy and its hardware cost is achieved with this architecture.

3.4.1 Fixed Point Approximation of DCT Coefficients

It can be verified from (3.6) that elements of N-point DCT have $(N-1)$ distinct values and DCT computations require floating point multiplications with these values. Realizing these floating point multiplications with infinite precision and at the same time maintaining properties of DCT matrix are the main challenges. On the other hand, integer transform has finite precision, but high dynamic range. To address these issues and to make hardware implementation of DCT matrix more efficient, all the real-valued coefficients of 32-point DCT are represented in fixed-point format and are expressed as

$$d_{i,j}^{32} = \frac{a}{2^l} \approx \sqrt{\frac{k}{32}} \cos\left[\frac{\pi(2j+1)i}{64}\right], \tag{3.29}$$

where a and l are integers, $k = 1$ and 2 for $i = 0$ and $i > 0$, respectively.

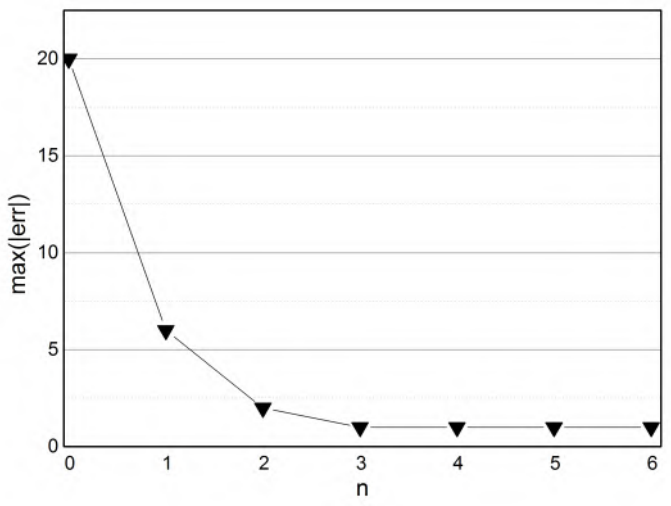

FIGURE 3.8: Maximum error versus scaling factor plot

With this approach, DCT can be performed by integer multiplications and right shift operations. However, hardware implementation of this method can produce large truncation error. To reduce the truncation error, the matrix obtained is multiplied by a scaling factor 2^n, where n is an integer. Therefore, any element of the proposed DCT matrix R_N can be written as

$$r_{i,j}^N = 2^n \times d_{i,j}^N \,, \tag{3.30}$$

where $d_{i,j}^N$ is expressed using (3.29). The magnitude of n determines the truncation errors which can be calculated as

$$\begin{aligned} &\boldsymbol{err} = \boldsymbol{x} - \hat{\boldsymbol{x}} \,, \\ &\textit{where} \quad \hat{\boldsymbol{x}} = (\boldsymbol{R_{32}}^T(\boldsymbol{R_{32}}.\boldsymbol{x})) >> 2n \,. \end{aligned} \tag{3.31}$$

The symbols $(.)^T$ and '>>' specify transpose and right shift operations, respectively. Input vector $\boldsymbol{x}$, containing a single non-zero element, is selected to produce all the frequency components at the output. However, the amplitude of this nonzero

TABLE 3.2: Comparison of coding performances

	E_{or}	E_{nr}	E_{cl}	C_g	η
Integer transform	0.0029	0.0014	0.0213	9.770	81.401
Proposed real-valued transform	0.0023	0.0024	0.0199	9.768	81.654

element is varied from the lowest to the highest value that a residual sample can take. To calculate the error, we have implemented $\boldsymbol{R_{32}}$ and $\boldsymbol{R_{32}^T}$ with different values of n in MATLAB using the hardware based approach as proposed in [68]. Fig. 3.8 depicts the maximum error which results with different values of n. This figure shows that the error is the minimum when $n = 3$ and it remains constant thereafter. Therefore, $n = 3$ is used to optimize the proposed DCT design for HEVC with least errors. Hence, 31 distinct values for 32-point DCT become

$$\begin{aligned} r_{i,0}^{32} = \{&2, 2, 127/2^6, 63/2^5, 31/2^4, 245/2^7, 15/2^3, 59/2^5, 29/2^4, 113/2^6, 55/2^5, 53/2^5, \\ &103/2^6, 99/2^6, 3/2, 181/2^7, 43/2^5, 5/2^2, 19/2^4, 71/2^6, 1, 15/2^4, 55/2^6, 25/2^5, \\ &43/2^6, 37/2^6, 31/2^6, 25/2^6, 9/2^5, 3/2^4, 3/2^5\}, \end{aligned} \tag{3.32}$$

where $i = 1, \cdots, 31$.

To verify the degree of approximation of the proposed DCT matrix, the maximum errors in the -closeness to DCT (E_{cl}), -orthogonality (E_{or}) and -norm (E_{no}) were calculated as defined in (3.13), (3.14), (3.15), respectively. The coding gain (C_g), the transform efficiency (η) were calculated as defined in [20] and the value of these parameters are compared with the integer DCT in [66] and the results are presented in Table 3.2. It shows that the proposed matrix approximately holds all the properties with minimal errors which are comparable to that of the HEVC core transform matrix [66].

A 32-point DCT is selected for error analysis because truncation error is large in the higher order DCT than that of the lower order. However, coefficient values for lower order DCTs are the subsets of (3.32) and can be written as

$$r_{i,j}^{\frac{N}{2}} = r_{2i+1,j}^{N} \times S_N, \quad \text{for} \quad 0 \leq i,j \leq (\frac{N}{2} - 1). \tag{3.33}$$

Here, scaling factor $S_N = \sqrt{\frac{32}{N}} = 2^{\frac{5-m}{2}}$ and $m = log_2 N$. For 2D DCT/IDCT this factor becomes 2^{5-m} which can be easily obtained with left shift operation on input vectors.

3.4.2 Data Flow Model of 2D DCT

The proposed data flow model of 2D -forward and -inverse transform for HEVC main profile is shown in Fig. 3.9. A single core is used for all sizes of DCT calculations. Therefore, inputs are multiplied by a scaling factor S_N^2 as shown in (3.33). This increases the accuracy of the lower order DCTs using the same hardware resources. In HEVC, 2D DCT of residual data is calculated by row and column transform consecutively. The maximum absolute value of the output samples after the first transform will be $2^{(b+\frac{11}{2})}$ where, b is the bit depth of the video. It requires $S_{T1} = (b + 7 - b_t)$ bits truncation to store transformed data in a b_t-bit TM. Similarly, after second transform the maximum value of output samples will be 2^{b_t+m-1}. Therefore, the amount of truncation required is $S_{T2} = m$ bits to store the data in the same memory.

This entails that a 15-bit transpose memory is enough to process a video with bit-depth 8 and it does not require any truncation after the first DCT operation. We have selected $S_{T1} = S_{T2} = m$, and scaling factor $S_N^2 = 2^5$ to design an uniform

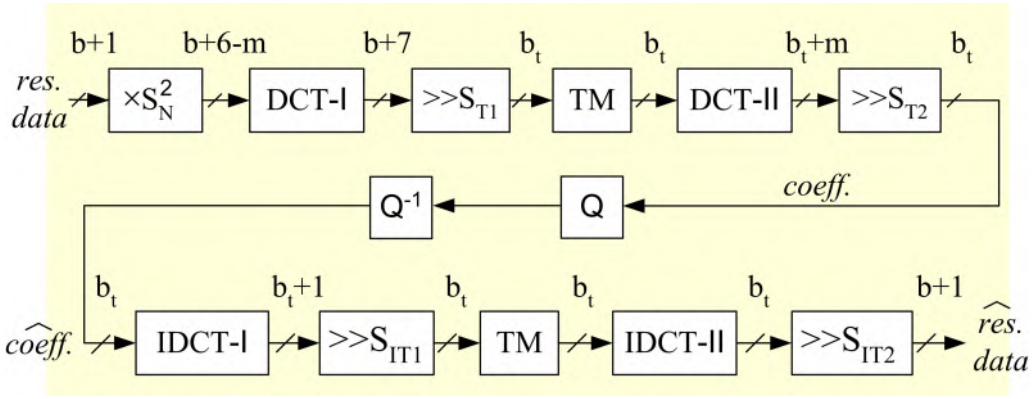

FIGURE 3.9: Datapath details for the proposed real-valued forward and inverse transforms

architecture for 8-bit video signal. Hardware cost of the folded architecture can be optimized in this manner.

In the case of 2D IDCT, S_{IT1} and S_{IT2} represent bit truncations required after the first and the second inverse transform, respectively. The maximum absolute value after the first IDCT will be $2^{(b_t-\frac{1}{2})}$ and truncation required S_{IT1} = 1 bit. On the other hand, after the second IDCT, the maximum absolute value will be 2^{b_t-1} and $S_{IT2} = b_t - b - 1$ bits truncation is required to recover the residual data. Scaling factor S_N^2 is not considered in the inverse transform as it increases the hardware cost for IDCT. It is clear from this discussion that the norm of the residual data can be preserved during forward and inverse transform by bit truncation operations.

3.4.3 Hardware implementation

The real-valued transform matrix obey symmetry and anti-symmetry properties of DCT. Hence, all the lower order DCTs can be realized by using the hardware implemented for even output coefficients of next higher order DCT. Therefore, an N-point DCT architecture can be reused to compute $\frac{N}{2}$-point DCT where, $N = 4$, 8, 16, 32. Therefore, the generalized hardware model for N-point DCT is similar

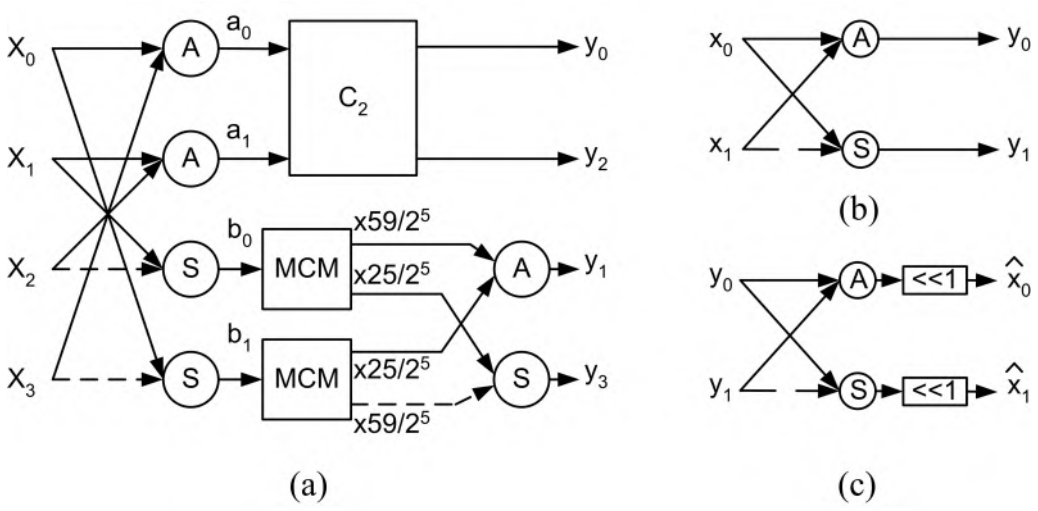

FIGURE 3.10: Architecture of (a) 4-point DCT (b) 2-point DCT (c) 2-point IDCT

to Fig. 3.5. An example of 4-point DCT architecture is discussed in the following subsection.

3.4.4 4-point DCT architecture

The proposed 4-point DCT architecture is shown in Fig. 3.10. It consists of an IB unit which produces a_0, a_1, b_0, and b_1 according to (3.25). In the next stage, a 2-point DCT architecture is used to produce even output coefficients, whereas two MCMs and output adder trees are used to produce odd output coefficients.

3.4.4.1 2-point DCT

The proposed 2-point DCT and IDCT can be expressed as

$$C_2 = K_{DCT}.\begin{bmatrix} 1 & 1 \\ 1 & -1 \end{bmatrix}, \quad C_2^T = K_{IDCT}.\begin{bmatrix} 1 & 1 \\ 1 & -1 \end{bmatrix}. \tag{3.34}$$

Note that $K_{DCT} = K_{IDCT} = \frac{181}{2^7}$ from (3.32). In order to recover the original unscaled signal after forward and inverse transform these constants must satisfy (3.35).

$$K_{DCT}.K_{IDCT} = 2\,. \tag{3.35}$$

Therefore, we selected $K_{DCT} = 1$ and $K_{IDCT} = 2$ to reduce implementation cost. The design for 2-point DCT and IDCT are shown in Fig. 3.10 (b) and (c), respectively.

3.4.4.2 MCM

The odd coefficients for 4-point DCT can be written as

$$\begin{bmatrix} y_1 \\ y_3 \end{bmatrix} = [B_2].\begin{bmatrix} b_0 \\ b_1 \end{bmatrix} = \begin{bmatrix} o_0 & o_1 \\ o_1 & -o_0 \end{bmatrix}.\begin{bmatrix} b_0 \\ b_1 \end{bmatrix}. \tag{3.36}$$

From (3.32), elements of matrix B_2 can be identified as $o_0 = \frac{59}{2^5}$ and $o_1 = \frac{25}{2^5}$. Here, fixed point multiplications are achieved with MCMs and hardware complexity is reduced. For all N, we followed the Hcub algorithm [69] to minimize the critical path of the MCMs. As a result, the maximum adder depth in any MCMs unit does not exceed two as compared to that of three in [68]. However, right- as well as left-shift operations are used to achieve desired constant multiplications. Such type of an MCM used in 4-point DCT is shown in Fig. 3.11(a).

3.4.4.3 Adder tree

Finally, outputs of the MCMs are added by adder tree unit to produce the final results. Any adder tree in N-point DCT consists of $(m-1)$ adder stages. In the proposed architecture, inputs are truncated by a single bit at every stage of adder

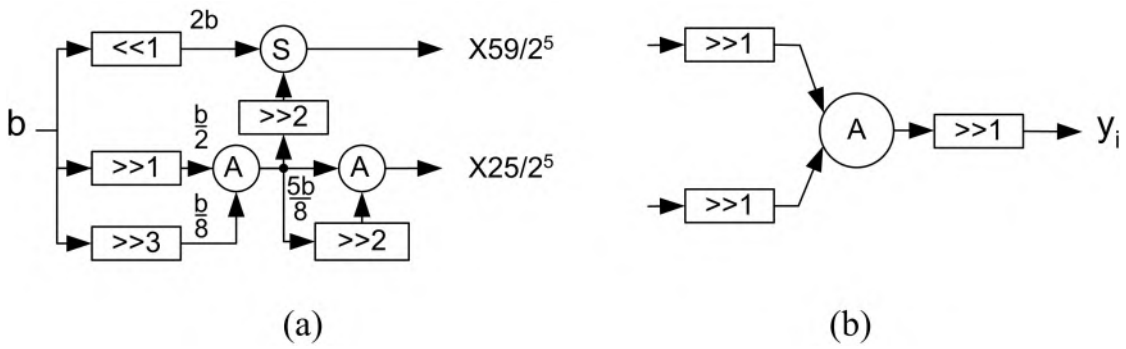

FIGURE 3.11: (a) MCM (b) Adder tree architecture for 4-point DCT

tree and final outputs are produced by truncating one more bit. Therefore, m bits truncation is achieved without any extra hardware. Fig. 3.11(b) shows an output adder tree for 4-point DCT.

3.5 Results and Discussion

3.5.1 Coding Efficiency

The coding performance of the proposed real-valued DCT matrix is compared with the integer DCT matrix of the HEVC reference software [12]. All the experiments are performed on HM-16.15 reference software with common test conditions as described in [103]. Input sequences from five different classes are used. These are Traffic 2560×1600, PeopleOnStreet 2560×1600, Kimono 1920×1080, ParkScene 1920×1080, BasketballDrillText 832×480, BQMall 832×480, BasketballPass 416×240, RaceHorses 416×240, Johnny 1280×720 and FourPeople 1280×720. The standard BD-rate approach is followed [88] to calculate PSNR variations with four different QP values (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 3.3. Note that the negative PSNR difference represents coding

TABLE 3.3: PSNR variation with respect to the reference algorithm

Class	Sequences	AI	LD	RA
A (WQXGA))	Traffic	-0.004	-	-0.000
	PeopleOnStreet	-0.003	-	0.001
B (1080P)	Kimono	-0.018	-0.008	-0.008
	ParkScene	-0.004	-0.003	-0.002
C (WVGA)	BasketballDrill	-0.004	0.005	-0.004
	BQMall	0.000	-0.003	0.009
D (240i)	BasketballPass	-0.001	-0.011	-0.007
	Racehorses	-0.000	-0.023	-0.017
E (720p)	Johnny	-0.010	0.041	-
	FourPeople	-0.004	-0.008	-

loss compared to that of the reference software. It is evident from Table 3.3 that the maximum PSNR variation with the proposed DCT matrix is less than 0.05 dB compared to that of the HEVC reference algorithm. However, the dynamic range of the proposed real-valued DCT matrix is less as compared to the integer DCT matrix. Therefore, hardware implementation of the proposed DCT matrix is less complex and has higher speed as compared to that of the reference algorithm as shown in the subsequent sub-sections. Additionally, the proposed algorithm requires an array of 15-bit memory, whereas HEVC reference algorithm needs an array of 16-bit memory to transpose intermediate data.

3.5.2 Implementation Results

3.5.2.1 FPGA implementation

The proposed 1D DCT architecture has been coded in Verilog for $N = 4, 8, 16$ and 32. It is synthesized using Xilinx Vivado 2016.2 and implemented on Virtex-7 FPGA. For comparison purpose, we have also implemented the integer DCT architecture as proposed in [68]. However, MCMs of the integer DCT architecture are re-designed with Hcub algorithm [69]. Detailed synthesis report and complexity comparison in

TABLE 3.4: Synthesis results and complexity comparison for 1D DCT architectures

Architecture	Size	LUT	SLICE	Time (ns)	Power (mW)	Area-Delay product	Area-Delay product reduction	Power reduction	Add	Shift
Integer DCT [68] + Hcub [69]	4	293	87	4.097	10	1200.1	-	-	14	10
	8	1189	352	5.643	44	6709.5	-	-	54	30
	16	4313	1213	6.84	154	29500.9	-	-	198	118
	32	13876	3970	9.642	629	133792.4	-	-	710	404
Proposed real-valued DCT	4	218	67	3.766	6	821	31.6%	40%	14	10
	8	904	271	5.504	31	4975.6	25.8%	29.5%	54	38
	16	3026	869	6.326	96	19142.5	35.1%	37.7%	182	126
	32	10256	3004	8.154	356	83627.4	37.5%	43.4%	614	334

terms of number of adders/subtractors and shifters are presented in Table 3.4. The input and output bus width of both architectures is 16 bits. Registers are inserted at the input and the output stage to compute the frequency of operation. Vector less power estimation is performed with the clock constrained at 100 MHz.

FPGA implementation of the proposed 32-point 1D DCT operates at 122.6 MHz frequency. It is clear from Table 3.4 that the hardware complexity of the proposed architecture is less as compared to the integer DCT. Consequently, FPGA implementation of the proposed architecture requires less area and power as compared to that of the integer DCT architecture [68]. The synthesis results show that area-delay product and power consumption of the proposed 32-point architecture is 37.5% and 43.4% less than that of the integer DCT architecture, respectively. It is because the multiplications in the integer DCT architecture increase intermediate data bus width. It increases intermediate processing element size as well as processing time. On the contrary, the proposed architecture uses a scaled version of real-valued DCT coefficients. So, the intermediate data bus width is small. Consequently, the intermediate processing elements size and processing time are less as compared to the integer DCT.

3.5.2.2 ASIC implementation

We have also coded a reusable architecture [68] of the proposed method to produce constant throughput irrespective of the DCT size. The source code of the reusable architecture is synthesized by Synopsys Design Compiler using 90-nm standard cell library [104] and results are compared with some of the existing DCT/IDCT architectures. These comparison results are presented in Table 3.5. Here, gate count has been estimated by normalizing total area with respect to the area of 2-input NAND gate and power has been calculated at 100 MHz. It is observed that the

Table 3.5: Comparison of ASIC implementations

	[68]	[75]	[71]	[76]	Prop.
Technology	90 nm	90 nm	90 nm	90 nm	90 nm
Gate counts	131 K	115.7 K	163 K	63.8 K	88.6 K
Max. Freq. (MHz)	187	200	250	270	256.4
Power (mW)	23.1	-	-	-	16.2
Max. Throughput (samples / clock)	32	32	25.7	8	32
FoM	45.64	55.32	39.4	33.85	92.6

proposed architecture requires 88.6K logic gates at 256.4 MHz operating frequency. To maintain a constant throughput of 32 samples/clock, the proposed architecture requires less number of gates and can operate at a higher frequency as compared to that of the designs in [68], [75] and [71]. The performance comparison shows that the designs in [76] operate at a higher frequency. However, its maximum throughput is 8 samples/clock and it depends on the size of DCT also. Therefore, it does not support 8K video with 60 FPS in 4:2:0 YUV format. But, with constant throughput (32 samples/clock) the proposed architecture can process 329 fps and 82 fps of 4K and 8K video, respectively in 4:2:0 YUV format. To determine the overall efficiency, Figure of Merit (FoM) is computed as

$$FoM = \frac{Total\ samples\ processed\ per\ second}{Total\ gate\ counts \times 10^3}. \tag{3.37}$$

On the basis of FoM, it is clear that the proposed architecture is the better than the reported ones.

3.6 Summary

The transform operations used in HEVC are introduced in this chapter as these operations serve the basis of this thesis. Particularly, the DCT as well as IDCT operations and the difficulties to implement those in hardware platform are discussed in details. The variable size integer transforms used in HEVC is discussed and its complexity analysis and hardware implementation method are reviewed in details. Finally, the disadvantage of integer transform is discussed and a new real-valued optimized architecture is proposed with detailed data-flow model. The proposed real-valued model uses a new set of fixed point coefficients which holds all the DCT/IDCT properties with minimal error and its coding efficiency is comparable to that of the HEVC integer transform. Intermediate data length and complexity of the proposed architecture is less as compared to that of the integer transform. Therefore, it reduces the area requirement and processing time when implemented on ASIC and FPGA platforms. The next chapter reveals the approximated transforms aimed to reduce complexity of HEVC execution without severe degradation in the reconstructed and decoded video quality.