February 2020
Master's Degree Thesis

A study on analytical transform kernel derivation for Versatile Video Coding (VVC)

Graduate School of Chosun University

Department of Information and Communication Engineering

Sandeep Shrestha

A study on analytical transform kernel derivation for Versatile Video Coding (VVC)

February 25, 2020

Graduate School of Chosun University

Department of Information and Communication Engineering

Sandeep Shrestha

A study on analytical transform kernel derivation for Versatile Video Coding (VVC)

Advisor: Prof. Bumshik Lee

A thesis submitted in partial fulfillment of the requirements for a master's degree in engineering

November 2019

Graduate School of Chosun University

Department of Information and Communication Engineering

Sandeep Shrestha

This is to certify that the master's thesis of

Sandeep Shrestha

has been approved by examining committee for the thesis requirement for the master's degree in Engineering.

Committee Chairperson	**Prof. Jae-Young Pyun**
Committee Member	**Prof. Tai-Won Um**
Committee Member	**Prof. Bumshik Lee**

November 2019

Graduate School of Chosun University

Table of contents

List of figures

List of tables

Acronyms

VVC	Versatile Video Coding
DCT	Discrete Cosine Transform
DST	Discrete Sine Transform
CTC	Common Test Conditions
VTM	VVC Test Model
EncT	Encoding Time
DecT	Decoding Time
AI	All Intra
RA	Random Access
LDB	Low-Delay B
QP	Quantization Parameter
CABAC	Context-Adaptive Binary Arithmetic Coding
AVC	Advanced Video Coding
HEVC	High-Efficiency Video Coding
HDR	High Dynamic Range
ROM	Read Only Memory
COT	Compound Orthonormal Transform
AMTs	Adaptive Multiple Transforms
TAF	Transform Adjustment Filter

한글요약

Versatile Video Coding(VVC) 비디오 코딩을 위한변환 커널 유도 방법에 관한 연구

산딥 쉬레스따
지도교수: 이범식
조선대학교대학원,
정보통신공학과

Versatile Video Coding (VVC)의 표준화에서는 DCT-2 (Discrete Cosine Transform-2), DST-7 (Discrete Sine Transform-7), DCT-8 (Discrete Cosine Transform-8)이 주변환커널로 사용되고 있다. 그러나 이들 변환 커널과 함께 DST-4 (Discrete Sine Transform-4)와 DCT-4 (Discrete Cosine Transform-4)도 역시 그 압축 성능과 구현성 복잡도 이득으로 인해 표준화 과정 중 주 변환 커널로 고려되어왔다. 구현 복잡도의 이점으로 인해 DST-4와 DCT-4는 DST-7과 DCT-8 변환 커널의 대안으로 사용될 수 있다. 그러나 이러한 다양한 종류의 변환 커널과 크기는를 정수 계수로 저장하기 위해선느 많은 메모리가 필요하며 이 것은 VVC 비디오 코덱 중요한 복잡도 문제가 될 수 있다. 이러한 문제를 해결하기 위하여 본 학위 논문에서는 unified sparse 행렬의 개념을 제안하고, 이 행렬을 이용하여 서로 다른 크기의 변환 커널 매트릭스(DCT-2, DST-7, DCT-8, DST-4, DCT-4)를 간단한 수학적 연산을 통해 얻을 수 있다. 제안하는 unified sparse matrix는 unified-DST-3 행렬과 grouped-DST-7 행렬의 두 부분으로 구성된다. unified-DST-3 행렬은 서로 다른 크기의 변환 커널 DCT-2, DST-4 및 DCT-4 변환 커널을 유도 사용되며, gruped-DST-7 행렬은 다양한 크기의 DST-7 및 DCT-8

변환 커널을 도출하는 데 사용된다. grouped-DST-7 행렬에서 32 픽셀 크기의 DST-7에 대해 두 가지 접근방식(Approach 1과 Approach 2)을 이용할 수 있다. 실험은 공통 테스트 조건(CTC)에 따라 VTM-3.0 참조 소프트웨어에서 수행되었고 제안하는 방법의 BDBR 결과는 AI와 RA 조건에 대해 압축 성능에 영향을 주지 않았고으나 기존의 방법에 비해 79.85%의 메모리 저감 효과를 얻을 수 있었다.

Abstract

A study on analytical transform kernel derivation for Versatile Video Coding (VVC)

Sandeep Shrestha
Advisor: Prof. Bumshik Lee
Department of Information and Communication Engineering
Graduate School of Chosun University

In the standardization of Versatile Video Codec (VVC), DCT-2 (Discrete Cosine Transform-2), DST-7 (Discrete Sine Transform-7), and DCT-8 (Discrete Cosine Transform-8) are being regarded as the major transform kernels. However, along with the defined transform kernels, DST-4 (Discrete Sine Transform-4) and DCT-4(Discrete Cosine Transform-4) are also counted as the basic transform kernels. Usually, DST-4 and DCT-4 can be used as the replacement of the DST-7 and DCT-8 transform kernels respectively for their effectiveness in smaller resolution sequences. While storing all those transform kernels, memory usage is regarded as the major issue. To deal with this scenario, a common sparse unified matrix concept is introduced in this thesis paper from which any point transform kernel matrix can be obtained i.e. DCT-2, DST-7, DCT-8, DST-4 and DCT-4 after some mathematical operations. The defined common sparse unified matrix is composed up of two parts: unified DST-3 matrix and grouped DST-7 matrix. Unified

DST-3 matrix is used to derive different block size DCT-2, DST-4 and DCT-4 transform kernels whereas the grouped DST-7 matrix is used to derive different block size DST-7 and DCT-8 transform kernels. In the grouped DST-7 matrix concept, two approaches (approach 1 and approach 2) has been introduced for the 32-point DST-7 to make 32-point DST-7 transform kernel compatible with the proposed algorithm. Similarly, the tuning of one element of the certain row of the DST-7 transform kernel affects other rows of DST-7 transform kernel is also shown in the proposed grouping concept of the DST-7 transform kernel matrix. The transform kernels used for simulation are DCT-2, DST-7 and DCT-8. The simulation is conducted in VTM-3.0 reference software under the common test condition (CTC). The BDBR results of the proposed methods showed no significant loss for AI and RA. However, for LDB, approach 1 showed a gain of 0.21% for V-chroma and less significant loss of 0.09% for U-chroma whereas approach 2 showed a gain of 0.05% for V-chroma and loss of 0.30% for U-chroma. Similarly, the proposed method has no significant impact on the encoding and decoding time which shows that there is no complexity issue in the proposed method even with 79.85% memory reduction.

Index Terms— VVC, DCT-2, DST-7, DCT-8, DST-3, DST-4, DCT-4, common sparse unified matrix, unified DST-3 matrix, grouped DST-7 matrix, unit element matrices, permutation matrix.

1. Introduction

1.1 Background

Generally, a video codec is a device or software that is used to compress or decompress a digital media file, such as a video or image. The word “codec” is composed of 2 parts: encode and decode. The encoder undergoes the compression (encoding) function whereas the decoder performs the decompression (decoding) function [1]. The basic block diagram that helps to understand the basic workflow of video codec is shown in Fig. 1-1.

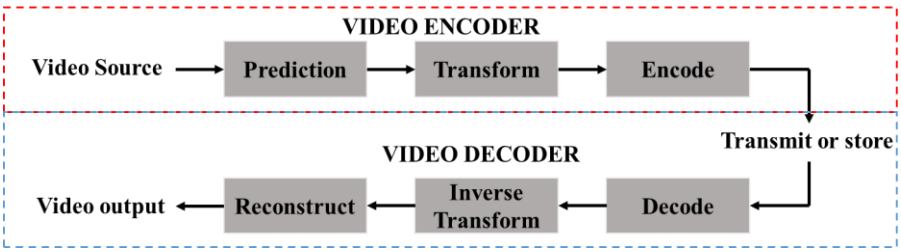


Figure 1-1. A basic block diagram of the video codec

In video coding, initially, video source undergoes through various prediction in the block level or macroblock level where prediction is done based on previously-coded are, either from the current frame (Intra prediction) [2] or from other frames that have already been coded and transmitted (Inter prediction) [3]. Finally, subtraction of the prediction from the current block or macroblock is done to obtain the predicted residual. After that, the transform and quantization process [4] is performed. In transform, basic transform kernels [5-8] such as DCT-2, DST-7, DCT-7, DST-4, and DCT-4 are used where a block of those residual samples are transformed using integer transform [9-10]. The transform produces outputs of a set of coefficients, each

of which is a weighting value for a standard basis pattern. The output of the transform i.e. a block of transformed coefficients is quantized where each coefficient is divided by an integer value, called quantization step size. The quantization reduces the precision of the transform coefficients according to a quantization parameter (QP). Setting QP to high value means that more coefficients are set to zero which results in high compression at the expense of poor decoded image or video quality. On the other hand, setting QP to a low value means more non-zero coefficients resulting in better-decoded video or image quality but lower compression. After these steps encoding process is done where a number of values are encoded to form the compressed bitstream. The values and parameter (syntax elements) are converted into binary codes using variable length coding [11] and/or arithmetic coding [12-13] and/or Context-adaptive binary arithmetic coding (CABAC) [14-15]. The encoded bitstream then can be stored or transmitted.

After all those encoding procedure, decoding is done from each of the syntax elements and extracted the information described above (quantized transform coefficients, prediction information, etc.). The inverse process is performed on the decoder side for each step mentioned above in the encoder. The quantized transform coefficients are re-scaled where each coefficient is multiplied by an integer value to restore its original scale. Similarly, an inverse transform combines the standard basis patterns i.e. inverse (DCT-2, DST-7, DCT-8, DST-4, and DCT-4) to re-create each block of residual data which combined together to form a residual macroblock. Finally, the decoder adds the prediction to the decoded residual to reconstruct a decoded different macroblock which can then be displayed as part of a video frame.

In the video coding, various standardization works have been carried out with their own specialization in each standardization. Some of the standardizations that have been carried out are H.261, H.262, H.263 [16], H.264/Advanced Video Coding (AVC) [17], H.265/High Efficiency Video Coding (HEVC) [18]. Recently the process of standardization of Versatile Video Coding (VVC) [19] is undergoing whose main objective is to provide a significant improvement in compression performance over the existing "High Efficiency Video Coding (HEVC)" standard and also aid the deployment of higher-quality video services and emerging applications such as 360° omnidirectional immersive multimedia and high-dynamic-range (HDR) video.

In the different trends of standardization, different block size transform kernels are used in the compression of video sources. In the previous video compression standards, different block size DCT-2 is regarded as the major transform kernel due to it's subsampling property for smaller sizes from its larger sized kernel [18]. Recently in the different proposals submitted for standardization of VVC, other transform kernels like DST-7, DCT-8, DST-4 and DCT-4 are also being used due to its energy distribution properties. With the introduction of the different block size transform kernels like DCT-2, DST-7, DCT-8, DST-4, and DCT-4, memory storage has become one of the major problems. On the other hand, applying the fast algorithm along with the less memory storage to transform kernels has become a challenging task to be performed.

The major proposed concept of this thesis is to focus on the memory storage along with the implementation of the fast algorithm to the different block size transform kernels. In this thesis, a common sparse unified matrix has been introduced with less memory usage based on which any block size transform

kernels i.e. DCT-2, DST-7, DCT-8, DST-4 and DCT-4 can be achieved after some mathematical computation. The proposed method only stores 1648 elements instead of 8180 elements each of 8-bit precision.

The used transform kernel elements are derived on the basis of following mathematical equations:

DCT-2
$$\left[C_N^{II}\right]_{n,k} = \sqrt{\frac{2}{N}}\varepsilon_k \cos\left(\frac{\pi(2n+1)k}{2N}\right) \quad (1)$$
where, $\varepsilon_k = \begin{cases} \frac{1}{\sqrt{2}} & k = 0 \\ 1 & \text{otherwise} \end{cases}$

DCT-4
$$\left[C_N^{IV}\right]_{n,k} = \sqrt{\frac{2}{N}} \cos\left(\frac{\pi(2n+1)(2k+1)}{4N}\right) \quad (2)$$

DCT-8
$$\left[C_N^{VIII}\right]_{n,k} = \frac{2}{\sqrt{2(N-1)}} \cos\left(\frac{\pi(2n+1)(2k+1)}{4N-2}\right) \quad (3)$$

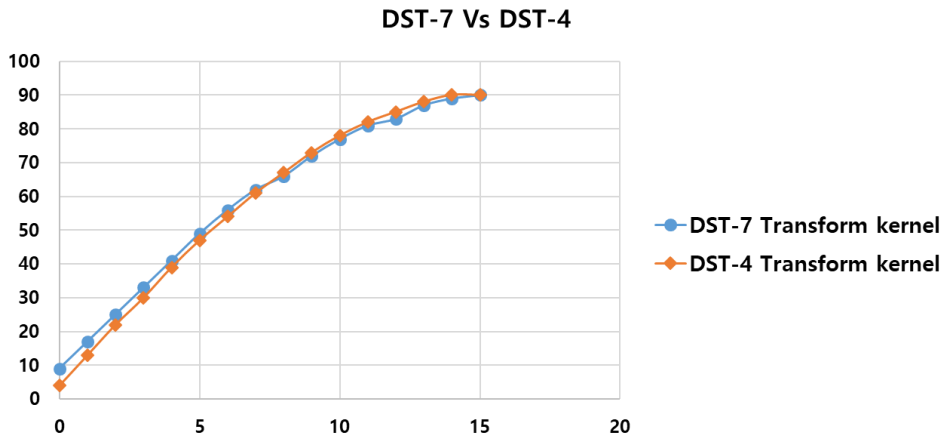
DST-4
$$\left[S_N^{II}\right]_{n,k} = \sqrt{\frac{2}{2N}}\varepsilon_k \sin\left(\frac{\pi(2n+1)(k+1)}{2N}\right) \quad (4)$$
where, $\varepsilon_k = \begin{cases} \frac{1}{\sqrt{2}} & k = N-1 \\ 1 & \text{otherwise} \end{cases}$

DST-7
$$\left[S_N^{VII}\right]_{n,k} = \frac{2}{\sqrt{2N+1}} \sin\left(\frac{\pi(2k+1)(n+1)}{2N+1}\right) \quad (5)$$

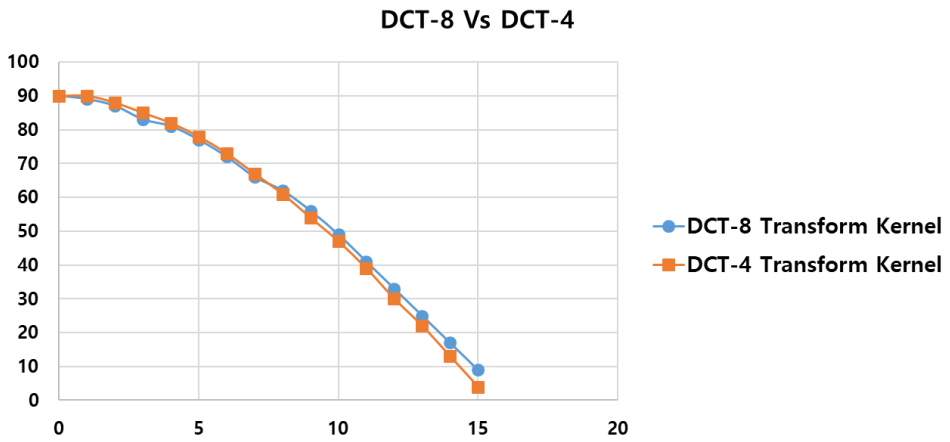
In general, a DCT and a DST is a Fourier related transform similar to discrete Fourier Transform (DFT). DCT is the real part and DST is the imaginary part of the DFT. There are eight variants of DCT and DST. The most common variant of discrete cosine transform is type-II DCT also known as DCT-2. The DCT has a strong “energy compaction” property with lossy compression as the signal information is concentrated in a few low-frequency components. It is effective only when there is not much difference in the residuals after

prediction that is why DCT-2 is mainly effective for the inter prediction. Regarding the usage of DST-7 and DCT-8, it is also regarded as lossy compression with strong “energy compaction” property as that of DCT-2. They produce more coding gains for boundary and intra prediction.

DST-4 and DCT-4 can be used as the replacement of the DST-7 and DCT-8 transform kernels respectively as they show the similarly signal behavior. The signal behaviors of respective signal are shown in Fig. 1-2.



(a) DST-7 Vs DST-4



(b) DCT-8 Vs DCT-4

Figure 1-2. DST-7 Vs DST-4 and DCT-8 Vs DCT-4

1.2 Objectives

As transform is regarded as one of the key component while undergoing compression and decompression in video coding, the goals and ideas for this research set are as following:

- Derive all the different point transform kernels i.e. DCT-2, DST-7, DCT-8, DST-4, and DCT-4 from a common sparse unified matrix.
- Reduce the (static) memory from 32768 bytes to 13168 bytes.
- Exhibit the relationship between different row elements in DST-7/DCT-8 transform kernel matrix i.e. tuning of one element of the row may affect other rows that depend on the tuned element of the row from the grouping concept of DST-7 introduced in this paper.

- Exhibit how the proposed concept supports the fast algorithm of DST-7 transform kernel.
- Use only three transform kernels DCT-2, DST-7 and DCT-8 although DCT-4 and DST-4 transform kernels can also be derived from the proposed method

1.3 Motivation

In the standardization of Versatile Video Codec (VVC), DCT-2, DST-7, and DCT-8 are being regarded as the major transform kernels. Previously, in the different video codec standards [17-18], different block-sized DCT-2 and small block-sized DST-7 transform kernels were used due to energy coefficient distribution properties for different residuals obtained after intra and inter prediction of the different frames. Recently, in the undergoing standardization of VVC, different block size transform kernels like DCT-2, DST-7, DCT-8, DST-4 and DCT-4 have been presented. It was presented as the option that DST-4 can be used instead of DST-7 and DCT-4 can be used instead of DCT-8 because of the similar energy coefficient distribution of these two transform kernels. On the other hand, the transform kernels DST-4 and DCT-4 can be achieved from the sub-sampling of even and odd rows of DCT-2 transform kernel that showed effectiveness only in smaller resolution test sequences. However, the use of different block sizes DST-7 and DCT-8 transform kernel which is also known as MTS, showed its effectiveness in different resolution test sequences. The use of MTS transform kernels based on prediction and block size is shown in Table. 1.

Table 1. MTS transform kernels based on prediction and block size

	Prediction	Prediction tools	Block Size	Transform kernel
MTS	Intra	ISP (Intra Sub-Partition)	4×4 upto 16×16	DCT-2 , DST-7
		MIP (Matrix Intra Prediction	4×4 upto 64×64	DCT-2, DST-7, DCT-8
		Normal Intra	4×4 upto 64×64	DST-7, DCT-8
	Inter	SBT (Sub-block Transform)	4×4 upto 64×64	(DCT-2, DST-7, DCT-8) depends on SBT position

As DST-7 and DCT-8 transform kernels cannot be achieved from sub-sampling or any other methods from the DCT-2 transform kernels, more efficient kernel storage and management scheme are needed. The consequence is that ROM memory size gets increases as it has to store 8180 elements each of 8-bit precision.

Several proposals were presented to solve the memory storage issue. These proposals tried to solve by replacing small block size DST-7 and DCT-8 transform kernel with small block size DST-4 and DCT-4 transform kernel respectively and to obtain large block size DST-7 and DCT-8 transform kernel, it was proposed to replace some rows of DCT-2 transform kernels with rows of DST-7 and DCT-8 transform kernels. Consequently, five transform kernels were used i.e. DCT-2 along with DST-4 and DCT-4 for small block size, DST-7 and DCT-8 for large block size. The proposed method presented failed to give larger block size DCT-2 transform kernel as some rows of DST-7 transform kernels were injected into larger block size DCT-2 transform kernel.

Similarly, different adjustment matrices and unified matrix concepts were also presented but none of them provided the required transform kernels i.e. DCT-2, DST-7 and DCT-8 for solving memory issues.

Hence, to overcome this issue, a common sparse unified matrix is proposed that stores only 1648 elements instead of 8180 elements (including different block size DCT-2, DST-7 and DCT-8) each of 8-bit precision in this thesis. The proposed method uses three transform kernels i.e. DCT-2, DST-7 and DCT-8 although it can also generate the different block size DST-4 and DCT-4 transform kernels by storing the same number of elements in memory along with the fast algorithm implementation of DCT-2 and DST-7 transform kernels.

1.4 Thesis Layout

The rest of this thesis is organized as follows: Section 2 provides the related works. Section 3 describes the proposed method i.e. common sparse unified matrix and its composition to derive different block sizes DCT-2, DCT-4, DST-4, DST-7, and DCT-8 transform kernels. Section 4 provides all the experimental results and finally, section 5 presents the conclusions of the thesis.

2. Related Works

For the standardization of VVC, different contributions related to memory were presented [21]. Some of the related proposals presented for the standardization of VTM-3.0 software are as following:

2.1 Compound Orthonormal Transform (COT)

This contribution [22] reports the use of DST-4/DCT-4 to replace DST-7/DCT-8 for 4-point and 8-point transforms used in MTS (multiple transform set) [21] and also embeds 16-point and 32-point DST-7/DCT-8 into 64-point DCT-2. It was proposed that all the transform types used in VVC can be extracted from one single 64×64 matrix so that 33% transform core storage can be saved when compared to VTM-3.0. In the proposed COT, 2-point, 4-point, 8-point, 16-point and 32-point DCT-2 can be extracted from the even rows whereas 4-point, 8-point DST-4/DCT-4 and 16-point, and 32-point DST-7/DCT-8 can be extracted from the odd rows.

As 16-point and 32-point DST-7/DCT-8 transform kernels are embedded in 64-point DCT-2 primary transform kernel, which prevents the extracting of the 64-point DCT-2 transform kernel. Consequently, the fast algorithm of the 64-point DCT-2 transform kernel [23] cannot be used. Similarly, the number of transform kernel matrix is also increased to 5 i.e. DCT-2, DST-7, DCT-8, DST-4 and DCT-4 whereas it was aimed to use only three transform kernels.

2.2 Unified Matrix for Transform

This contribution [24] proposed to use three types of transform kernels i.e. DCT-2, DST-7 and DCT-8 by sampling from the large size transform kernel matrix which is termed as the unified matrix. All these transform kernels are mathematical sampled using the relations as:

DCT-2 can be derived by sampling a subset of coefficients in the matrix as:

$$\mathbf{DCT2_{N\times N}[l,k]} = \mathbf{matrix_{64\times 64}[l, 2(6 - \log_2 N) * k]}, \quad (6)$$

where $\mathbf{DCT2_{N\times N}[l,k]}$ represents the l-th element of the k-th basic vector of DCT-2 for N × N matrix and N is a size of matrix from 64 to 2 and $\mathbf{matrix_{64\times 64}[l,k]}$ represents the unified matrix from which sub-sampling is done.

DCT-8 can be derived by sampling a subset of coefficients in the matrix as:

$$\mathbf{DCT8_{N\times N}[l,k]} = \mathbf{matrix_{64\times 64}[l, 2(6 - \log_2 N) * k + 2(5 - \log_2 N)]}, \quad (7)$$

where $\mathbf{DCT8_{N\times N}[l,k]}$ represents the l-th element of the k-th basic vector of DCT-8 for N × N matrix and N is a size of matrix from 32 to 4 and $\mathbf{matrix_{64\times 64}[l,k]}$ represents the unified matrix from which sub-sampling is done.

DST-7 can be derived by sampling a subset of coefficients in the matrix as:

$$\mathbf{DST7_{N\times N}[l,k]} = (-1)^{k} * \mathbf{DCT8\_N \times N[N - l - 1, k]} \quad (8)$$

$$\mathbf{DST7_{N\times N}[l,k]} = (-1)^{k} * \mathbf{matrix_{64\times 64}[N - l - 1, 2(6 - \log_2 N) * k + 2(5 - \log_2 N)]}, \quad (9)$$

where $\mathbf{DST7_{N\times N}[l,k]}$ represents the l-th element of the k-th basic vector of DST-7 for $N \times N$ matrix and N is a size of matrix from 4 to 32 and $\mathbf{matrix_{64\times 64}[l,k]}$ represents the unified matrix from which sub-sampling is done.

Based on the test results of the proposed method for all Intra (AI), no gain for Class A1 and Class A2 but consists of some gain for the lower classes has been shown. For random access (RA) and low delay B (LDB), the overall result shows no gain. While making an analysis based on the simulation results, losses in RA and LDB and losses in AI's class A1 and A2 might be due to the mismatch of the DST-7 and DCT-8 transform kernels with the original transform signal values.

2.3 Adjustment Stages

This proposal [25] is based on JEM7 [26], where five types of discrete cosine and sine transforms (DCTs and DSTs) are employed for primary separable transformation of residual blocks as in Fig. 2-1.

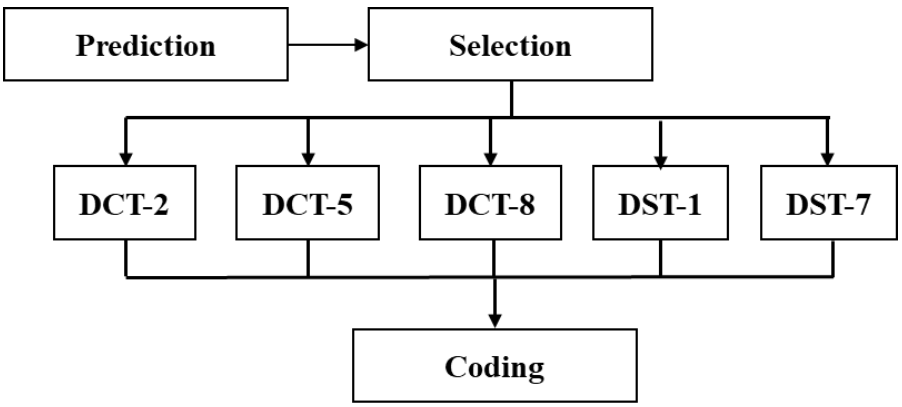


Figure 2-1. JEM-7.1 Adaptive Multiple Transform

It was proposed that these five types of discrete cosine and sine transforms can be approximated by applying different adjustment stages as shown in Fig. 2-2.

The adjustment stages are defined by sparse block-band orthogonal matrices which are proposed to be easy to compute and very similar to filtering with the small number of taps. The experiment performed on “4-tap” sparse block-band matrices.

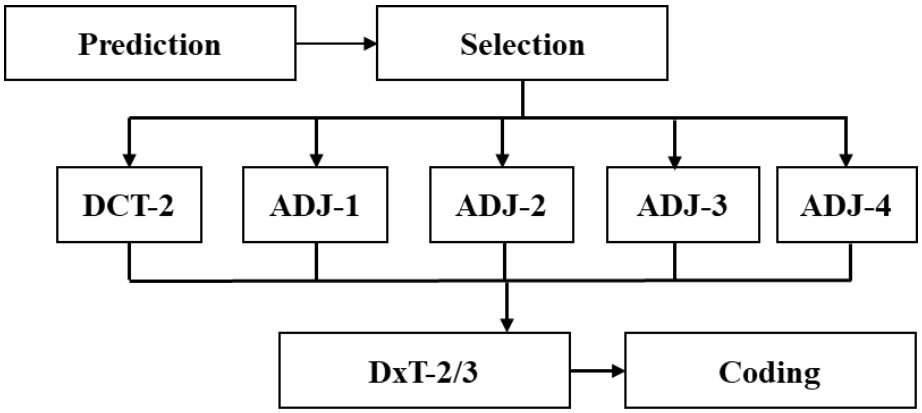


Figure 2-2. JEM-7.1 Adjustment Stages [25]

Although the proposed method tried to save memory by not storing DCT-5, DCT-8, DST-1 and DST-7 using different adjustment stages, the number of multiplication increases and the normalized values of the different adjustment stages are different, hence the transform kernels may not exactly be matched. Based on the experimental results, there is a less significant loss in luminance but the noticeable loss in the chrominance of the higher test sequences which might be due to mismatch of the proposed transform signal.

2.4 Transform Adjustment Filter (TAF)

This proposal [27] is the extended version of 2.3. This contribution reports a method of complexity reduction of the DST-7/DCT-8 kernels using MTS. The adjustment stages are given the name of TAF (Transform Adjustment Filter) which is a sparse matrix as shown in Fig. 2-3.

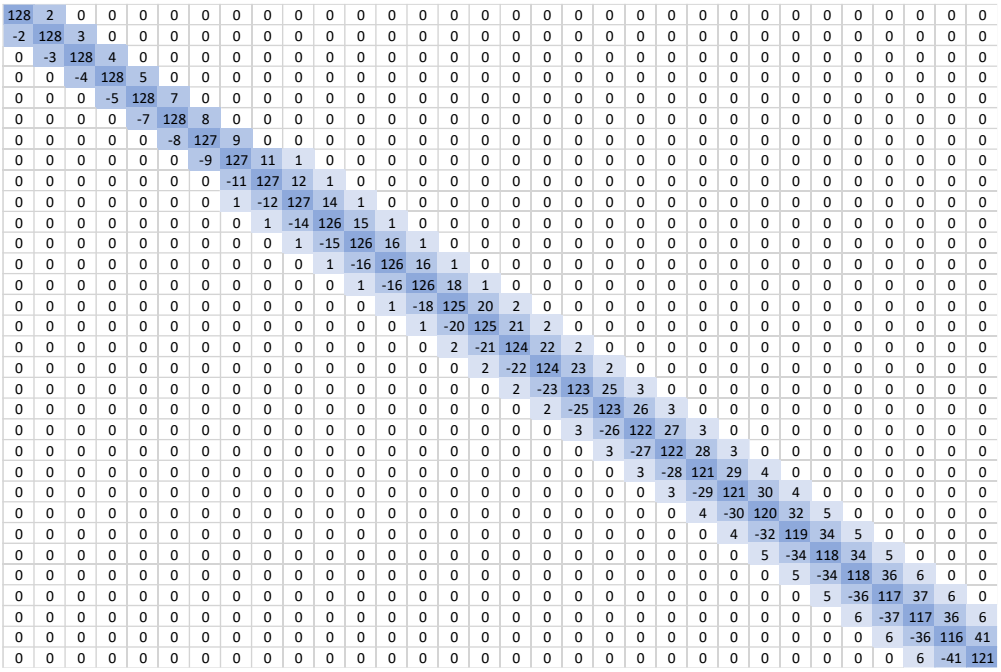


Figure 3-2. JEM-7.1 Transform Adjustment Filter (TAF)

The size M forward DST-7 ($\mathbf{S}_7$) is approximated as:

$$\mathbf{S}_7 = \mathbf{S} \cdot \mathbf{C}_2^t \cdot \mathbf{F} \cdot \mathbf{A}\,, \tag{10}$$

where $\mathbf{S}$ and $\mathbf{F}$ are the sign matrix and flip matrix and $\mathbf{C}_2^{\mathrm{t}}$ is the transpose of the DCT-2 matrix and $\mathbf{A}$ is a sparse adjustment matrix.

$$\mathbf{S}_{\mathbf{k},\mathbf{n}} = \begin{cases} 0, & \text{if } \mathrm{n!} = \mathrm{k} \\ (-1)^{\mathrm{n}}, & \text{otherwise} \end{cases} \tag{11}$$

$$\mathbf{F_{k,n}} = \begin{cases} 1, & \text{if } n = M - 1 - k \\ 0, & \text{otherwise} \end{cases} \tag{12}$$

DCT-8 ($\mathbf{C}_8$) transform kernel of size M is derived as (13):

$$\mathbf{C}_8 = \mathbf{C}_2^{t} \cdot \mathbf{F} \cdot \mathbf{A} \cdot \mathbf{F} \tag{13}$$

Consequently, the inverse DST-7 and DCT-8 are derived using the relation (14) and (15)

$$\mathbf{iS_7} = \mathbf{A^t} \cdot \mathbf{F}.\mathbf{C_2} \cdot \mathbf{S} \tag{14}$$

$$\mathbf{iC_8} = \mathbf{F} \cdot \mathbf{A^t} \cdot \mathbf{F} \cdot \mathbf{C_2}\ , \tag{15}$$

where $\mathbf{iS_7}$ and $\mathbf{iC_8}$ refers to inverse DST-7 and inverse DCT-8 respectively.

The number of adjustment stage i.e. TAF has been reduced in comparison to section 2.3 but the number of multiplication increase as additional multiplication with the TAF has to be performed. This contribution reduces memory usage only by storing the different point DCT-2 transform kernels i.e. no need to store the different point DST-7/DCT-8 transform kernels. The normalized value of the TAF is not the same which is the major reason that the derived transform kernels are not well-matched with the original DST-7/DCT-8 transform kernel. Consequently, higher loss can be seen in the higher test sequences classes in spite of the less significant loss in the overall simulation result.

3. Proposed Method

In this research, an analytical transform kernel derivation method using a common sparse unified matrix is proposed. The common sparse unified matrix is composed of two parts

1. A unified DST-3 matrix to derive DCT-2, DST-4 and DCT-4 transform kernels
2. A grouped DST-7 matrix to derive DST-7 and DCT-8 transform kernels

3.1 Common Sparse Unified Matrix

The common sparse unified matrix is the overall core matrix from which any point of DCT-2, DST-7, DCT-8, DST-4, and DCT-4 transform kernels can be derived.

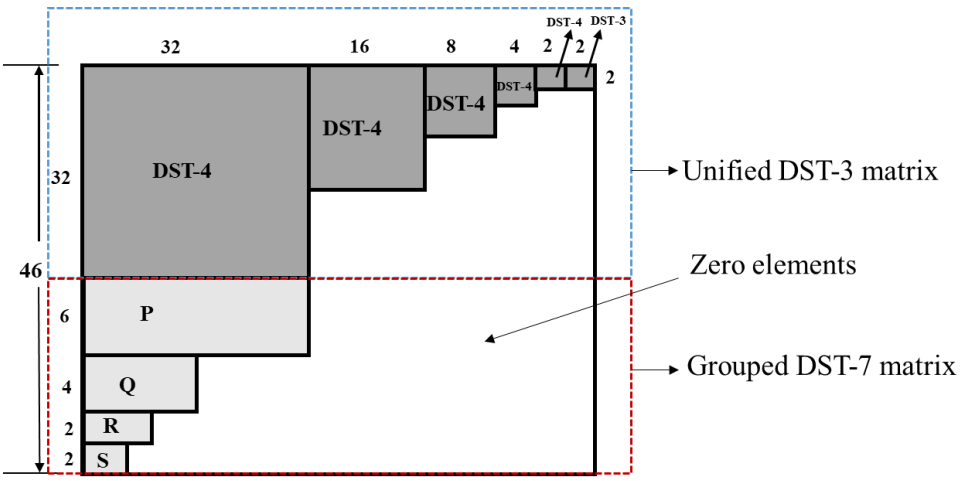


Figure 3-1. Common sparse unified matrix

Fig. 3-1. shows the basic configuration of the common sparse unified matrix. This matrix comprises 32-point, 16-point, 8-point, 4-point, and 2-point DST-4 transform kernels along with a 2-point DST-3 transform kernel. Similarly, it also comprises of 6 × 32 block size **P** matrix, 4 × 16 block size **Q** matrix, 2 × 8 block size **R** matrix and 2 × 4 block size **S** matrix. In short, the common sparse unified matrix is composed of two parts

1. A unified DST-3 matrix to derive the DCT-2 transform kernel
2. A grouped DST-7 matrix to derive the DST-7/DCT-8 transform kern els

3.2 Unified DST-3 Matrix (U)

A unified DST-3 matrix also named as **U** matrix is the matrix that is used to derive different point DCT-2 transform kernel. The unified matrix (**U**) comprises of 32-point DST-4, 16-point DST-4, 8-point DST-4, 4-point DST-4, 2-point DST-4, and 2 point DST-3 transform kernels and the remaining transform kernel elements are depicted as zero values. The structure unified DST-3 matrix is shown in Fig. 3-2.

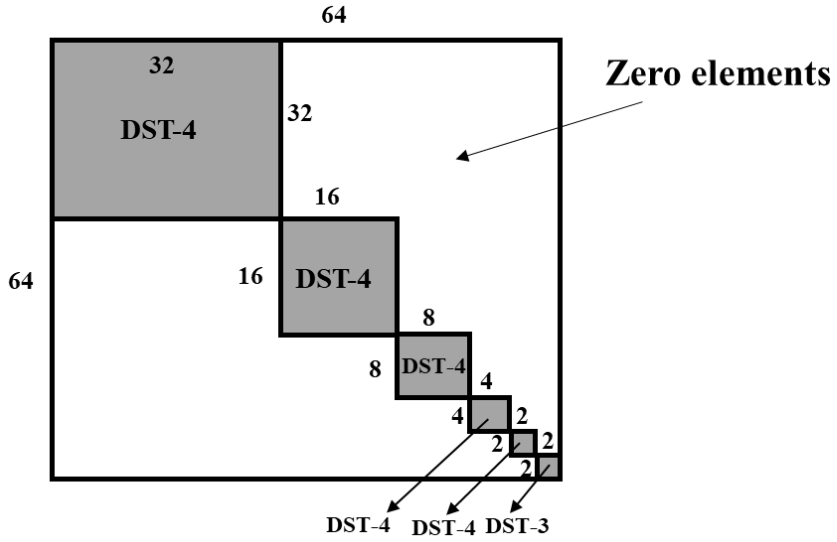


Figure 3-2. Structure of 64-point unified DST-3 (**U**) matrix

Since the **U** matrix contains a number of zero elements except DST-4 elements, the data size to be stored can significantly be reduced. Thus, the defined sparse unified DST-3 matrix stores only 1368 elements each of 8-bit precision. The overall size of the common sparse unified matrix is 10944 bytes.

3.2.1 DST-3 and DCT-2 Relationship

There exist the relationship between DST-3 and DCT-2 [25] transform kernels which can be expressed as

$$\mathbf{C_2} = \mathbf{F} \times \mathbf{S_3} \times \mathbf{S}\,, \tag{16}$$

where $\mathbf{C_2}$ and $\mathbf{S_3}$ are DCT-2 and DST-3 transform kernel respectively and **F** and **S** are a flipping and a sign change matrices, respectively and defined as

$$\mathbf{F}_{m,n} = \begin{cases} 1, & \text{if } n = N - 1 - m, \\ 0, & \text{otherwise,} \end{cases} \tag{17}$$

$$\mathbf{S}_{m,n} = \begin{cases} (-1)^m, & \text{if } n = m, \\ 0, & \text{otherwise.} \end{cases} \tag{18}$$

3.2.2 The Proposed Derivation Method of the DCT-2 Kernel

In order to retrieve any block size of the DCT-2 kernel, it is necessary to use any block size DST-3 transform kernel matrix as (16). In Fig. 3-3, at the right bottom part of the unified DST-3 matrix (**U**), there exists a 2×2 DST-3 transform kernel matrix. Using relation (16), and selected 2×2 DST-3 transform kernel matrix, 2-points DCT-2 transform kernel can be derived. For deriving other points of the DCT-2 kernel, different point DST-3 transform kernels are needed. Thus to derive different point DST-3 transform kernels from the unified DST-3 matrix, 64-point unit-element matrices i.e. **A**, **B**, **C**, **D**, **E** is multiplied with unified DST-3 matrix (**U)** which is shown in Fig. 3-3.

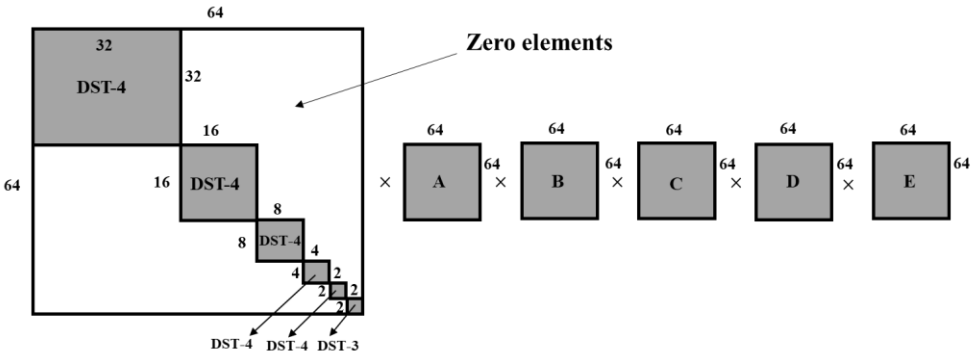


Figure 3-3. Decomposition of 64-point DST-3 transform matrix kernel

Unit-element matrices **A, B, C, D, and E** are the matrices which comprise of either 1 or -1 or 0 values. **U** matrix is multiplied with these unit-element matrices in sequential order. In each multiplication with unit-element matrices in sequential order gives a large block size DST-3 transform kernel matrix. Based on Fig. 3-3, the relationship between sparse unified DST-3 transform kernel matrix and five unit-element matrix can be shown as:

$$\mathbf{S}_{3,64} = \mathbf{UABCDE}\,, \tag{19}$$

where $\mathbf{S}_{3,64}$is the 64-point DST-3 transform kernel matrix and **U** represents the unified DST-3 matrix and **A**, **B**, **C**, **D,** and **E** represents 64-point unit-element matrices composed of -1, 0 and 1 and is defined as (20) – (24)

$$\mathbf{A_{m,n}} = \begin{cases} 1\,, & \text{if } (m = n,\ \forall\ 0 < (m,n) < 60)\ || \\ & \quad (m = n,\ \forall\ 60 < (m,n) < 62)\ || \\ & \quad \begin{pmatrix} m + n = 123\ \forall\ (61 < m < 64)\ \text{and} \\ (60 < n < 62) \end{pmatrix}\ || \\ & \quad \begin{pmatrix} i - j = -2,\ \forall\ (60 < m < 62)\ \text{and} \\ (61 < n < 64) \end{pmatrix} \\ -1\,, & \text{if } (m + n = 125,\ \forall\ 61 < (m,n) < 64 \\ 0\,, & \text{elsewhere} \end{cases} \tag{20}$$

$$\mathbf{B_{m,n}} = \begin{cases} 1\,, & \text{if } (m = n,\ \forall\ 0 < (m,n) < 56)\ \ || \\ & \quad (m = n,\ \forall\ 55 < (m,n) < 60)\ || \\ & \quad \begin{pmatrix} m + n = 119\,, \forall\ (59 < m < 64)\ \text{and} \\ (\ 55 < n < 60) \end{pmatrix} || \\ & \quad \begin{pmatrix} i - j = -4,\ \forall\ (55 < m < 60)\ \text{and} \\ (59 < n < 60) \end{pmatrix} \\ -1\,, & \text{if } (m + n = 123,\ \forall\ 59 < (m,n) < 64 \\ 0\,, & \text{elsewhere} \end{cases} \tag{21}$$

$$\mathbf{C_{m,n}} = \begin{cases} 1, & \text{if } (m = n,\ \forall\ 0 < (m,n) < 48)\ || \\ & (m = n,\ \forall\ 47 < (m,n) < 56)\ || \\ & \begin{pmatrix} m + n = 111\ , \forall\ (55 < m < 64) \text{ and} \\ (\ 47 < n < 56) \end{pmatrix} || \\ & \begin{pmatrix} i - j = -8,\ \forall\ (47 < m < 56) \text{ and} \\ (55 < n < 64) \end{pmatrix} \\ -1, & \text{if } (m + n = 119,\ \forall\ 55 < (m,n) < 64 \\ 0, & \text{elsewhere} \end{cases} \tag{22}$$

$$\mathbf{D_{m,n}} = \begin{cases} 1, & \text{if } (m = n,\ \forall\ 0 < (m,n) < 32)\ || \\ & (m = n,\ \forall\ 31 < (m,n) < 48)\ || \\ & \begin{pmatrix} m + n = 95\ , \forall\ (47 < m < 64) \text{ and} \\ (\ 31 < n < 48) \end{pmatrix} || \\ & \begin{pmatrix} i - j = -16,\ \forall\ (31 < m < 48) \text{ and} \\ (47 < n < 64) \end{pmatrix} \\ -1, & \text{if } (m + n = 111,\ \forall\ 47 < (m,n) < 64 \\ 0, & \text{elsewhere} \end{cases} \tag{23}$$

$$\mathbf{E_{m,n}} = \begin{cases} 1, & \text{if } (m = n,\ \forall\ 0 < (m,n) < 32)\ || \\ & \begin{pmatrix} m + n = 63\ , \forall\ (31 < m < 64) \text{ and} \\ (\ 0 < n < 32) \end{pmatrix}\ || \\ & \begin{pmatrix} i - j = -32,\ \forall\ (0 < m < 32) \text{ and} \\ (31 < n < 64) \end{pmatrix} \\ -1, & \text{if } (m + n = 95,\ \forall\ 31 < (m,n) < 64 \\ 0, & \text{elsewhere} \end{cases} \tag{24}$$

From the unit-element matrices different point DST-3 transform kernels can be derived as:

$$\mathbf{S}_{3,4} = \mathrm{T}[\mathbf{U}_{64} \times \mathbf{A}_{64}]_4 \tag{25}$$

$$\mathbf{S}_{3,8} = \mathrm{T}[\mathbf{U}_{64} \times \mathbf{M}_{64}]_8 \tag{26}$$

$$\mathbf{S}_{3,16} = \mathrm{T}[\mathbf{U}_{64} \times \mathbf{N}_{64}]_{16} \tag{27}$$

$$\mathbf{S}_{3,32} = \mathrm{T}[\mathbf{U}_{64} \times \mathbf{O}_{64}]_{32} \tag{28}$$

$$\mathbf{S}_{3,64} = \mathbf{U}_{64} \times \mathbf{P}_{64}\ , \tag{29}$$

where $\boldsymbol{S}_{3,N}$ is DST-3 transform kernels with N×N block selection of the right-bottom part of the matrix obtained after multiplication and $\mathrm{T}[\,]_N$ represents the function that takes N×N block of the right-bottom part of the matrix,**U** indicates the unified DST-3 matrix as in Fig. 3-1 and **M**, **N**, **O,** and **P** can be obtained using unit-element matrices as (30).

$$\begin{aligned} \mathbf{M} &= \mathbf{A} \times \mathbf{B} \\ \mathbf{N} &= \mathbf{A} \times \mathbf{B} \times \mathbf{C} \\ \mathbf{O} &= \mathbf{A} \times \mathbf{B} \times \mathbf{C} \times \mathbf{D} \\ \mathbf{P} &= \mathbf{A} \times \mathbf{B} \times \mathbf{C} \times \mathbf{D} \times \mathbf{E} \end{aligned} \tag{30}$$

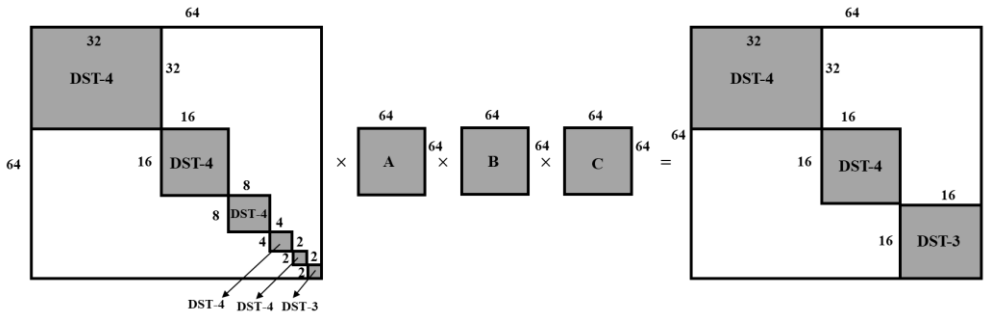


Figure 3-4. Proposed method to obtain 16-point DST-3

Fig. 3-4 represents the proposed method of obtaining DCT-2 using (27). The unified DST-3 matrix **U** is multiplied with **A**, **B,** and **C** unit-element matrix in a sequential manner, which gives the result as 32-point DST-4, 16-point DST-4, and 16-point DST-3 kernel matrices. From the generated matrix, 16×16 block size DST-3 matrices are selected which is the right bottom part of the

generated matrix. Using the relation (16), it is easy to derive the 16-point DCT-2 kernel matrix from the 16-point DST-3 matrix. Similar sequential multiplication is performed to derive different block size DST-3 transform kernel and from the different points DST-3 transform kernel, DCT-2 transform kernels can be obtained using (16).

3.3 The Proposed Grouped DST-7 Transform Kernel

In the different block size DST-7 transform kernel, some of the special rows are selected and stored. The selected stored rows are used to generate the group of elements of respective block size DST-7 transform kernel with the help of permutation matrix (**G**). Permutation matrix (**G**) is composed up of 1, -1 and 0 elements which is multiplied with the selected stored rows to generate different group depending on different row elements based on different block sizes. The rows in the particular group are dependent on each previous rows sequentially which is explained in section 3.3.1. The total elements that has to be stored to generate different block size DST-7 transform kernel is 400 elements each of 8-bit precision. The **P**, **Q**, **R**, and **S** elements as shown in Fig. 3-1 are the stored rows to implement the grouping concept of DST-7 transform kernel. After the generation of different block size DST-7 transform kernel, respective block size DCT-8 transform kernel is derived using the relation between DST-7 transform kernel, flipping matrix (**F**) and sign change matrix (**S**).

3.3.1 Grouping Concept

In order to describe the grouping concept of DST-7 in the proposed method, 32-point DST-7 $[\mathbf{S}_7]_{32}$ is taken as an example. For the 32-point DST-7, only six rows i.e. $[\mathbf{S}_{7,\mathbf{0}}]_{32}$, $[\mathbf{S}_{7,\mathbf{1}}]_{32}$, $[\mathbf{S}_{7,\mathbf{2}}]_{32}$, $[\mathbf{S}_{7,\mathbf{3}}]_{32}$, $[\mathbf{S}_{7,\mathbf{5}}]_{32}$, and $[\mathbf{S}_{7,\mathbf{6}}]_{32}$ rows are stored, which are also regarded as the specifically selected rows. [Note$[\mathbf{S}_{\mathbf{M},\mathbf{N}}]_{\mathbf{O}}$*: where M, N, and O are transform kernel, row number and block size respectively*]. Using these specific selected rows and permutation matrix (**G**), all elements of the 32-point DST-7 can be derived and complete DST-7 transform kernels are obtained as shown in Table. 2. Since only size of rows are used for deriving complete 32-points DST-7, a significantly large amount of data size can be saved for deriving the kernel.

Table 2. Generation of 32-point DST-7 transform kernel

Rows	**G (Permutation matrix)**				
$[\mathbf{S}_{7,\mathbf{0}}]_{32}$	$[\mathbf{S}_{7,\mathbf{31}}]_{32}$	$[\mathbf{S}_{7,\mathbf{30}}]_{32}$	$[\mathbf{S}_{7,\mathbf{28}}]_{32}$	$[\mathbf{S}_{7,\mathbf{24}}]_{32}$	$[\mathbf{S}_{7,\mathbf{16}}]_{32}$
$[\mathbf{S}_{7,\mathbf{1}}]_{32}$	$[\mathbf{S}_{7,\mathbf{29}}]_{32}$	$[\mathbf{S}_{7,\mathbf{26}}]_{32}$	$[\mathbf{S}_{7,\mathbf{20}}]_{32}$	$[\mathbf{S}_{7,\mathbf{8}}]_{32}$	$[\mathbf{S}_{7,\mathbf{15}}]_{32}$
$[\mathbf{S}_{7,\mathbf{2}}]_{32}$	$[\mathbf{S}_{7,\mathbf{27}}]_{32}$	$[\mathbf{S}_{7,\mathbf{22}}]_{32}$	$[\mathbf{S}_{7,\mathbf{12}}]_{32}$	$[\mathbf{S}_{7,\mathbf{7}}]_{32}$	$[\mathbf{S}_{7,\mathbf{17}}]_{32}$
$[\mathbf{S}_{7,\mathbf{3}}]_{32}$	$[\mathbf{S}_{7,\mathbf{25}}]_{32}$	$[\mathbf{S}_{7,\mathbf{18}}]_{32}$	$[\mathbf{S}_{7,\mathbf{4}}]_{32}$	$[\mathbf{S}_{7,\mathbf{23}}]_{32}$	$[\mathbf{S}_{7,\mathbf{14}}]_{32}$
$[\mathbf{S}_{7,\mathbf{5}}]_{32}$	$[\mathbf{S}_{7,\mathbf{21}}]_{32}$	$[\mathbf{S}_{7,\mathbf{10}}]_{32}$	$[\mathbf{S}_{7,\mathbf{11}}]_{32}$	$[\mathbf{S}_{7,\mathbf{9}}]_{32}$	$[\mathbf{S}_{7,\mathbf{13}}]_{32}$
$[\mathbf{S}_{7,\mathbf{6}}]_{32}$	$[\mathbf{S}_{7,\mathbf{19}}]_{32}$				

As shown in Table. 2, the rows column indicates the specific selected rows of 32-point DST-7. The elements that can be derived from the multiplication of the specific selected rows and the permutation matrix (**G**) are given row-wise. Each selected specific rows while storing are independent of each other i.e. for 32-point DST-7, the selected specific $[\mathbf{S}_{7,0}]_{32}$, $[\mathbf{S}_{7,1}]_{32}$, $[\mathbf{S}_{7,\mathbf{2}}]_{32}$, $[\mathbf{S}_{7,3}]_{32}$, $[\mathbf{S}_{7,5}]_{32}$, and $[\mathbf{S}_{7,6}]_{32}$ rows are independent of each other. Based on

the selected specific stored row, determining dependent rows can be explained in more detail in Fig. 3-5 and Fig. 3-6.

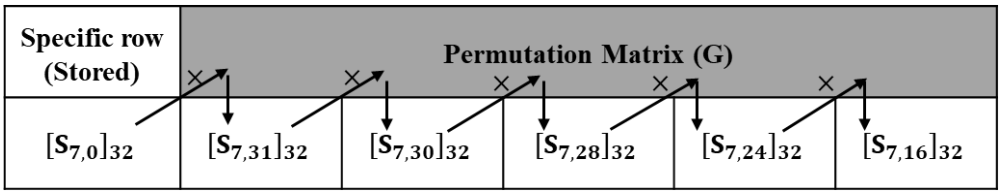


Figure 3-5. Generation of $[\mathbf{S}_{7,0}]_{32}$ dependent grouped row elements of DST-7 using permutation matrix

$$\mathbf{R_0} \times \mathbf{G} = \mathbf{R_{31}} \Rightarrow \mathbf{R_{31}} \times \mathbf{G} = \mathbf{R_{30}} \Rightarrow \mathbf{R_{30}} \times \mathbf{G} = \mathbf{R_{28}} \Rightarrow \mathbf{R_{28}} \times \mathbf{G} = \mathbf{R_{24}} \Rightarrow \mathbf{R_{24}} \times \mathbf{G} = \mathbf{R_{16}}$$

Figure 3-6. General multiplication of specific stored row and permutation matrix (**G**)

The $[\mathbf{S}_{7,0}]_{32}$ is considered as one of the basic row. Based on a $[\mathbf{S}_{7,0}]_{32}$ all the remaining rows i.e. $[\mathbf{S}_{7,31}]_{32}$, $[\mathbf{S}_{7,30}]_{32}$, $[\mathbf{S}_{7,28}]_{32}$, $[\mathbf{S}_{7,24}]_{32}$, and $[\mathbf{S}_{7,16}]_{32}$ row elements are generated. Based on Fig. 3-5 and Fig. 3-6, the following steps are used to generate the remaining rows

1. The $[\mathbf{S}_{7,31}]_{32}$ row is generated by multiplication of $[\mathbf{S}_{7,0}]_{32}$ row of DST-7 transform kernel and **G** matrix
2. The $[\mathbf{S}_{7,30}]_{32}$ row is generated by multiplying $[\mathbf{S}_{7,31}]_{32}$ row obtained from step 1. and **G**
3. In each row, the first element of the previous row should be equal to the last element of the resultant row obtained after multiplying with the **G** matrix without considering sign values as shown in Fig. 3-7.
4. The process is repeated until the last element of the $[\mathbf{S}_{7,0}]_{32}$ row matches with the first-element of the $[\mathbf{S}_{7,16}]_{32}$ row without considering the sign values as shown in Fig. 3-7.
5. If the first element generated row is negative, the sign values of each

element of the respective row is altered.

0	4	9	13	17	21	26	30	34	38	42	45	50	53	56	60	63	66	68	72	74	77	78	80	82	84	85	86	88	88	89	90	90
31	9	-17	26	-34	42	-50	56	-63	68	-74	78	-82	85	-88	89	-90	90	-88	86	-84	80	-77	72	-66	60	-53	45	-38	30	-21	13	-4
30	17	-34	50	-63	74	-82	88	-90	88	-84	77	-66	53	-38	21	-4	-13	30	-45	60	-72	80	-86	90	-89	85	-78	68	-56	42	-26	9
28	34	-63	82	-90	84	-66	38	-4	-30	60	-80	90	-85	68	-42	9	26	-56	78	-89	86	-72	45	-13	-21	53	-77	88	-88	74	-50	17
24	63	-90	66	-4	-60	90	-68	9	56	-89	72	-13	-53	88	-74	17	50	-88	77	-21	-45	86	-78	26	42	-85	80	-30	-38	84	-82	34
16	90	-4	-90	9	89	-13	-88	17	88	-21	-86	26	85	-30	-84	34	82	-38	-80	42	78	-45	-77	50	74	-53	-72	56	68	-60	-66	63

1	13	26	38	50	60	68	77	82	86	89	90	88	85	80	74	66	56	45	34	21	9	-4	-17	-30	-42	-53	-63	-72	-78	-84	-88	-90
29	26	-50	68	-82	89	-88	80	-66	45	-21	-4	30	-53	72	-84	90	-88	78	-63	42	-17	-9	34	-56	74	-85	90	-86	77	-60	38	-13
26	50	-82	88	-66	21	30	-72	90	-78	42	9	-56	85	-86	60	-13	-38	77	-90	74	-34	-17	63	-88	84	-53	4	45	-80	89	-68	26
20	82	-66	-30	90	-42	-56	86	-13	-77	74	17	-88	53	45	-89	26	68	-80	-4	84	-63	-34	90	-38	-60	85	-9	-78	72	21	-88	50
8	66	90	56	-13	-74	-88	-45	26	80	84	34	-38	-85	-78	-21	50	88	72	9	-60	-90	-63	4	68	89	53	-17	-77	-86	-42	30	82
15	90	13	-88	-26	84	38	-78	-50	72	60	-63	-68	53	77	-42	-82	30	86	-17	-89	4	90	9	-88	-21	85	34	-80	-45	74	56	-66

2	21	42	60	74	84	89	89	84	74	60	42	21	0	-21	-42	-60	-74	-84	-89	-89	-84	-74	-60	-42	-21	0	21	42	60	74	84	89
27	42	-74	89	-84	60	-21	-21	60	-84	89	-74	42	0	-42	74	-89	84	-60	21	21	-60	84	-89	74	-42	0	42	-74	89	-84	60	-21
22	74	-84	21	60	-89	42	42	-89	60	21	-84	74	0	-74	84	-21	-60	89	-42	-42	89	-60	-21	84	-74	0	74	-84	21	60	-89	42
12	84	60	-42	-89	-21	74	74	-21	-89	-42	60	84	0	-84	-60	42	89	21	-74	-74	21	89	42	-60	-84	0	84	60	-42	-89	-21	74
7	60	89	74	21	-42	-84	-84	-42	21	74	89	60	0	-60	-89	-74	-21	42	84	84	42	-21	-74	-89	-60	0	60	89	74	21	-42	-84
17	89	-21	-84	42	74	-60	-60	74	42	-84	-21	89	0	-89	21	84	-42	-74	60	60	-74	-42	84	21	-89	0	89	-21	-84	42	74	-60

3	30	56	77	88	89	80	63	38	9	-21	-50	-72	-85	-90	-84	-68	-45	-17	13	42	66	82	90	86	74	53	26	-4	-34	-60	-78	-88
25	56	-88	80	-38	-21	72	-90	68	-17	-42	82	-86	53	4	-60	88	-78	34	26	-74	90	-66	13	45	-84	85	-50	-9	63	-89	77	-30
18	88	-38	-72	68	42	-86	-4	88	-34	-74	66	45	-85	-9	89	-30	-77	63	50	-84	-13	90	-26	-78	60	53	-82	-17	90	-21	-80	56
4	38	68	86	88	74	45	9	-30	-63	-84	-90	-78	-53	-17	21	56	80	90	82	60	26	-13	-50	-77	-89	-85	-66	-34	4	42	72	88
23	68	-88	45	30	-84	78	-17	-56	90	-60	-13	77	-85	34	42	-88	72	-4	-66	89	-50	-26	82	-80	21	53	-90	63	9	-74	86	-38
14	88	30	-78	-56	60	77	-34	-88	4	89	26	-80	-53	63	74	-38	-86	9	90	21	-82	-50	66	72	-42	-85	13	90	17	-84	-45	68

5	45	78	90	77	42	-4	-50	-80	-90	-74	-38	9	53	82	89	72	34	-13	-56	-84	-88	-68	-30	17	60	85	88	66	26	-21	-63	-86
21	78	-77	-4	80	-74	-9	82	-72	-13	84	-68	-17	85	-66	-21	86	-63	-26	88	-60	-30	88	-56	-34	89	-53	-38	90	-50	-42	90	-45
10	77	80	9	-72	-84	-17	66	86	26	-60	-88	-34	53	90	42	-45	-90	-50	38	89	56	-30	-88	-63	21	85	68	-13	-82	-74	4	78
11	80	72	-17	-86	-60	34	90	45	-50	-89	-30	63	85	13	-74	-78	4	82	68	-21	-88	-56	38	90	42	-53	-88	-26	66	84	9	-77
9	72	86	34	-45	-89	-63	13	78	82	21	-56	-90	-53	26	84	77	9	-66	-88	-42	38	88	68	-4	-74	-85	-30	50	90	60	-17	-80
13	86	45	-63	-78	21	90	26	-77	-66	42	88	4	-85	-50	60	80	-17	-90	-30	74	68	-38	-88	-9	84	53	-56	-82	13	89	34	-72

6	53	85	85	53	0	-53	-85	-85	-53	0	53	85	85	53	0	-53	-85	-85	-53	0	53	85	85	53	0	-53	-85	-85	-53	0	53	85
19	85	-53	-53	85	0	-85	53	53	-85	0	85	-53	-53	85	0	-85	53	53	-85	0	85	-53	-53	85	0	-85	53	53	-85	0	85	-53

Figure 3-7. Grouped 32-point DST-7 transform kernel

Fig. 3-7 shows all the derived 32-point DST-7 transform kernels where the first row of each individual group is regarded as the stored specific row. After multiplication of the $[\mathbf{S}_{7,0}]_{32}$ row and **G**, the $[\mathbf{S}_{7,31}]_{32}$ row is obtained. In the $[\mathbf{S}_{7,31}]_{32}$ row, the first element of the $[\mathbf{S}_{7,0}]_{32}$ row i.e. 4 is equal to the last element of the $[\mathbf{S}_{7,31}]_{32}$ row neglecting the sign values. Similarly, the first

element of $[\mathbf{S}_{7,31}]_{32}$ row is equal to the last element of the $[\mathbf{S}_{7,30}]_{32}$row i.e. 9. The process is repeated until the last element of the $[\mathbf{S}_{7,0}]_{32}$ row matches with the first element of the $[\mathbf{S}_{7,16}]_{32}$ row i.e. 90 in the first group of 32-point DST-7 transform kernel elements.

Similarly, in the $[\mathbf{S}_{7,6}]_{32}$ row group of 32-point DST-7, the $[\mathbf{S}_{7,19}]_{32}$ row is obtained after multiplying with the permutation matrix (**G**). In this group, the first element of the $[\mathbf{S}_{7,6}]_{32}$ row is equal to the last element of the $[\mathbf{S}_{7,19}]_{32}$ row i.e. *53* and the last element of the $[\mathbf{S}_{7,6}]_{32}$ row is equal to the first element of the $[\mathbf{S}_{7,19}]_{32}$ row i.e. 85. Hence, only two elements are grouped.

Based on the steps explained above, all the DST-7 transform kernel row elements are obtained.

3.3.2 Specific Rows of Different Point DST-7

The generation of 16-point DST-7 transform kernel based on the proposed grouping concept is shown in the Table. 3. The specific selected rows are $[\mathbf{S}_{7,0}]_{16}$, $[\mathbf{S}_{7,1}]_{16}$, $[\mathbf{S}_{7,2}]_{16}$, and $[\mathbf{S}_{7,5}]_{16}$. Using the specific selected rows and permutation matrix (**G**), all the elements of 16-point DST-7 transform kernels are derived.

Table 3. Generation of 16-point DST-7 Transform kernel

Rows	Permutation matrix (**G**)			
$[\mathbf{S}_{7,0}]_{16}$	$[\mathbf{S}_{7,15}]_{16}$	$[\mathbf{S}_{7,14}]_{16}$	$[\mathbf{S}_{7,12}]_{16}$	$[\mathbf{S}_{7,8}]_{16}$
$[\mathbf{S}_{7,1}]_{16}$	$[\mathbf{S}_{7,13}]_{16}$	$[\mathbf{S}_{7,10}]_{16}$	$[\mathbf{S}_{7,4}]_{16}$	$[\mathbf{S}_{7,7}]_{16}$
$[\mathbf{S}_{7,2}]_{16}$	$[\mathbf{S}_{7,11}]_{16}$	$[\mathbf{S}_{7,6}]_{16}$	$[\mathbf{S}_{7,3}]_{16}$	$[\mathbf{S}_{7,9}]_{16}$
$[\mathbf{S}_{7,5}]_{16}$				

For the generation of 8-point DST-7 transform kernel, the selection specific rows are $[\mathbf{S}_{7,0}]_8$ and $[\mathbf{S}_{7,1}]_8$. The specific stored rows, permutation matrix, and rows depending on the specific stored are shown in Table. 4.

Table 4. Generation of 8-point DST-7 transform kernel

Rows	Permutation matrix (**G**)		
$[\mathbf{S}_{7,0}]_8$	$[\mathbf{S}_{7,7}]_8$	$[\mathbf{S}_{7,6}]_8$	$[\mathbf{S}_{7,4}]_8$
$[\mathbf{S}_{7,1}]_8$	$[\mathbf{S}_{7,5}]_8$	$[\mathbf{S}_{7,2}]_8$	$[\mathbf{S}_{7,3}]_8$

Similarly, the generation of 4-point DST-7 transform kernel with the help of permutation matrix (**G**) and specific stored rows i.e. $[\mathbf{S}_{7,0}]_4$ and $[\mathbf{S}_{7,1}]_4$ are shown in Table. 5.

Table 5. Generation of 4-point DST-7 transform kernel

Rows	Permutation matrix (**G**)	
$[\mathbf{S}_{7,0}]_4$	$[\mathbf{S}_{7,3}]_4$	$[\mathbf{S}_{7,2}]_4$
$[\mathbf{S}_{7,1}]_4$		

3.3.3 Derivation of DCT-8

As from the proposed grouped DST-7 matrix, different point DST-7 transform kernel elements can be easily obtained. Based on the obtained DST-7 transform kernel, DCT-8 [25] of various block size can be achieved using the relation

$$\mathbf{C}_8 = \mathbf{S} \times \mathbf{S_7} \times \mathbf{F}\ , \tag{31}$$

where $\mathbf{C}_8$ and $\mathbf{S}_7$ are the DCT-8 and DST-7 transform kernels, respectively and **S** and **F** are the sign-changing and flipping matrices, respectively and defined as

$$\mathbf{F_{m,n}} = \begin{cases} 1, & \text{if } n = N - 1 - m, \\ 0, & \text{otherwise}, \end{cases} \tag{32}$$

$$\mathbf{S_{m,n}} = \begin{cases} (-1)^m, & \text{if } n = m, \\ 0, & \text{otherwise}. \end{cases} \tag{33}$$

3.4 Permutation Matrix

The permutation matrix,(**G)** is used to derive all the elements of DST-7 transform kernel. It comprises of either 1 or -1 or 0 elements. This matrix multiplies with the specific stored rows of DST-7 which finally gives all the elements of the particular block size DST-7 transform kernel.

3.4.1 DST-7 Transform Kernel Pattern

Grouping concept is used in the derivation of the DST-7 transform kernel which is explained in section 3.3. Let's consider a certain group of elements of 32-point DST-7 transform kernel as shown in Fig. 3-8 where the$[\mathbf{S}_{7,0}]_{32}$ row is regarded as the specific stored row.

0	4	9	13	17	21	26	30	34	38	42	45	50	53	56	60	63	66	68	72	74	77	78	80	82	84	85	86	88	88	89	90	90
31	9	-17	26	-34	42	-50	56	-63	68	-74	78	-82	85	-88	89	-90	90	-88	86	-84	80	-77	72	-66	60	-53	45	-38	30	-21	13	-4
30	17	-34	50	-63	74	-82	88	-90	88	-84	77	-66	53	-38	21	-4	-13	30	-45	60	-72	80	-86	90	-89	85	-78	68	-56	42	-26	9
28	34	-63	82	-90	84	-66	38	-4	-30	60	-80	90	-85	68	-42	9	26	-56	78	-89	86	-72	45	-13	-21	53	-77	88	-88	74	-50	17
24	63	-90	66	-4	-60	90	-68	9	56	-89	72	-13	-53	88	-74	17	50	-88	77	-21	-45	86	-78	26	42	-85	80	-30	-38	84	-82	34
16	90	-4	-90	9	89	-13	-88	17	88	-21	-86	26	85	-30	-84	34	82	-38	-80	42	78	-45	-77	50	74	-53	-72	56	68	-60	-66	63

Figure 3-8. Grouped 32-point DST-7 transform kernel ($[\mathbf{S}_{7,0}]_{32}$ as the specific stored row)

If analysis is made based on Fig. 3-8, the specific elements of the respective rows are repeated at a certain pattern without considering the sign change. Based on $[\mathbf{S}_{7,0}]_{32}$ row elements, $[\mathbf{S}_{7,31}]_{32}$ row elements are derived. If a clear study on $[\mathbf{S}_{7,0}]_{32}$ and $[\mathbf{S}_{7,31}]_{32}$ row elements is done, the $[\mathbf{S}_{7,31}]_{32}$ row elements are repeated at the interval of one element gap of the $[\mathbf{S}_{7,0}]_{32}$ row i.e. 9, 17, 26, etc. elements of $[\mathbf{S}_{7,0}]_{32}$ row the 1-st, 3-rd, 5-th column respectively repeated in $[\mathbf{S}_{7,31}]_{32}$ row in 0-th, 1-st, 2-nd etc. column elements respectively without considering sign change. Similarly, the $[\mathbf{S}_{7,31}]_{32}$ row and $[\mathbf{S}_{7,30}]_{32}$ row and remaining rows depending on the previous row follows a similar pattern. This pattern exhibited by different grouped elements of DST-7 transform kernel is defined in the matrix model which is known as the permutation matrix. Hence, the permutation matrix is nothing but the replica of the pattern depicted by different groups of DST-7 transform kernel as shown in Fig. 3-9.

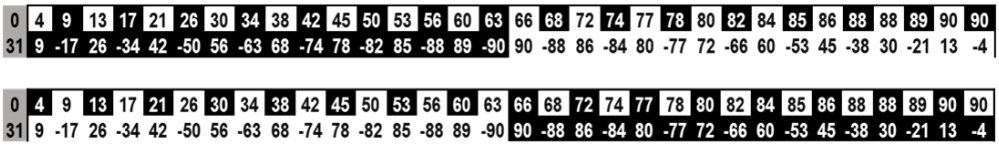


Figure 3-9. Pattern matching between $[\mathbf{S}_{7,0}]_{32}$ row and $[\mathbf{S}_{7,31}]_{32}$ row of 32-point DST-7 transform kernel

This pattern behavior shows that the elements under the same group are directly or indirectly dependent on each other which means a change of one element under the certain grouped matrix affects the other elements under the same group.

3.4.2 Algorithm for Generation of Permutation Matrix (G)

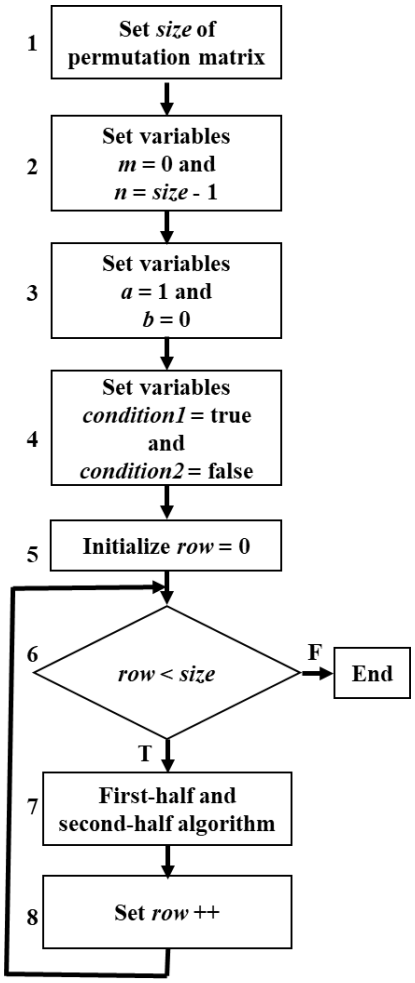


Figure 3-10. Algorithm for the generation of the permutation matrix (**G** matrix)

The following steps are used for the generation of the permutation matrix based on Fig. 3-10.

1. Initially, the *size* of the permutation matrix is defined. Based on the *size* defined, after generation of the permutation matrix (**G**) multiplication with the specific stored DST-7 transform kernel elements are done. Sizes defined are either 4-point, 8-point, 16-point,

32-point.

2. Variable *m* and *n* are set to 0 and *size*-1 so that the initial size of the permutation matrix can be set from 0.

3. Two variables *a* and *b* are initially set to 1 and 0 respectively in order to apply the condition either the row and column equal to *a* or *b* in order to set the value in the respective location as -1 or 1 or 0 while computing the values of permutation matrix.

4. Variable *condition1* and *condition2* are set to true and false respectively in order to set different logic for the first-half part and second-half part of the matrix which plays a crucial role in setting elements to -1 or 1 or 0 in the permutation matrix as shown in Fig. 3-12.

5. Initially, *row* is initialized to 0.

6. If the *row* is less than *size* defined in step 1 further processing is done else the whole operation is ended.

7. Different column operations are done based on *condition1* and *condition2* so that first-half and second-half of the matrix can be separated which finally allows us to use our first-half algorithm for the first-half part of the matrix and second-half algorithm in the second half of the matrix that allows us to set values of the permutation matrix as -1 or 1 or 0 as shown in the Fig. 3-11 and Fig. 3-12.

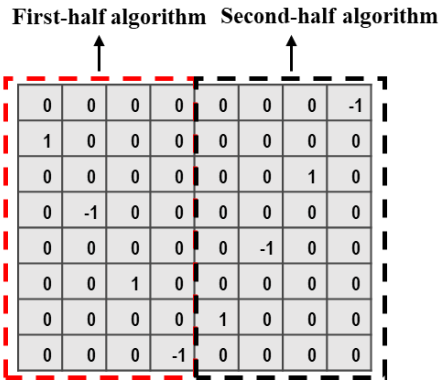


Figure 3-11. 8-point permutation matrix

The first-half and second-half algorithms are explained in section 3.4.3 and 3.4.4 respectively.

8. Variable *row* is incremented by 1 and loop back to step 6.

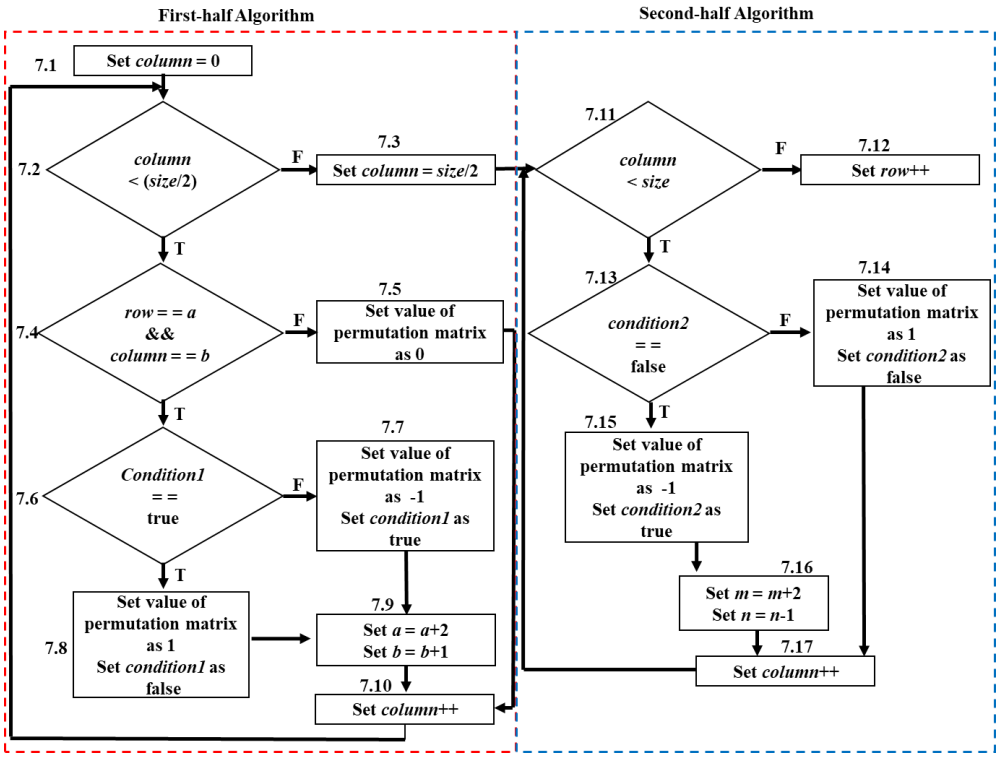


Figure 3-12. First-half and second-half algorithm to obtain permutation matrix

3.4.3 First-half Algorithm

7.1. Based on Fig. 3-12, initially the variable *column* is set to 0 i.e. in order to allow the counting of row and column of the matrix starting from [0][0] i.e. [row][column]=[0][0]

7.2. Check if the *column* is less than half of the *size* defined in step 7.1.

7.3. If the condition defined in step 7.2 is false, then the value of the variable *column* is set to half of the *size* defined in step 1 which allows setting values of the second half of the permutation matrix as -1 or 1 or 0.

7.4. If the condition defined in step 7.2 holds true, then further checking is done to set the permutation matrix value as 0 or not by using the condition as the *row* is equal to the variable *a* and *column* variable define in step 7.1 equal to variable *b* defined in step 7.3.

7.5. If the condition defined in step 7.4 is false then for the first-half of the permutation matrix at the particular location the element value is set to 0.

7.6. If the condition defined in step 7.4 is true then further checking is done i.e. if the *condition1* defined is true or false which helps in setting the value as -1 or 1.

7.7. If the condition defined in step 7.6 holds false the value of the permutation matrix at the particular location is set as -1 and the variable *condition1* is set to true.

7.8. If the condition defined in step 7.6 is true, then the permutation matrix at the particular location is set to 1 and the variable *condition1* is set to false.

7.9. Then for setting values i.e. -1 or 1 or 0 in further first-half of the permutation matrix the variable, *a* is set as *a*+2 and *b* as *b*+1.

7.10. For computing values in the further column, the variable *column* is increased by 1 and further processing is repeated from step 7.2.

3.4.4 Second-half Algorithm

7.11. After setting the value of the *column* as half of the *size* defined in step 7.3, the condition is applied to check if the variable *column* is less than *size* or not.

7.12. If the condition defined in step 7.11 holds false then the variable *row* is increased by 1.

7.13. If the condition defined in step 7.11 holds true then the *condition2* variable is checked for true or false.

7.14. If the condition defined in step 7.13 is false then the element value at the particular location of the permutation matrix is set as 1 and variable *condition2* is set to false.

7.15. If the condition defined in step 7.13 is true then the element value at the particular location of the permutation matrix is set as -1 and variable *condition2* is set to true.

7.16. Then the variable *m* is set to *m*+2 and variable *n* to *n*-1.

7.17. Finally, the variable *column* is incremented by 1 and step 7.11 is repeated.

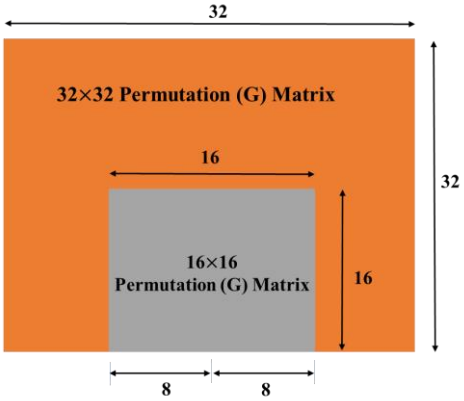
3.4.5 Alternate Method

On the basis of the first-half and second-half algorithm explained, any block size permutation matrix can easily be obtained. For the generation of the permutation matrix from the first-half and second-half algorithm, the size has to be defined each time as 4 or 8 or 16 or 32 as input based on the requirement. The same result for the different small block size permutation matrix can also

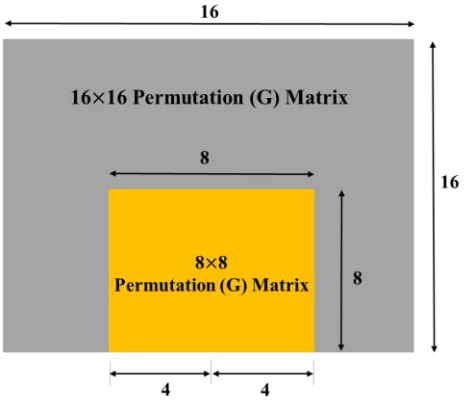
be achieved from the larger size permutation matrix by sub-sampling of larger block size permutation matrix.

Fig. 3-13 shows the selection of a smaller block size permutation matrix from the larger block size permutation matrix. Let us consider the size of the permutation matrix as 32-point. As the size of the permutation matrix is set to 32, from the first-half and second-half algorithm the 32-point permutation matrix can be obtained. From the obtained 32-point permutation matrix, the bottom middle right and left equal half part of the smaller block size generated from the larger block is usually selected as shown in Fig. 3-13 (a), (b) and (c). Fig 3-13 (a) shows the generation of 16-point permutation matrix from a 32-point permutation matrix. At the bottom middle right (16/2 = 8-point) and left (16/2 = 8-point) equal half part of the smaller block size which forms a 16-point smaller permutation matrix from the larger 32-point permutation matrix. Similarly, Fig 3-13 (b) shows the selection of the 8-point permutation matrix from a 16-point or 32-point permutation matrix and Fig. 3-13 (c) explains the selection of the 4-point permutation matrix from an 8-point or 16-point or 32-point permutation matrix.

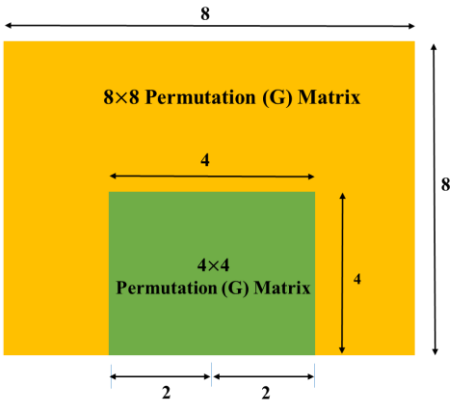
The advantage of this selection method is that there is no need to set the size of the permutation matrix as input and execute first-half and second-half algorithm again and again based on our block size requirement. The required block size permutation matrix is selected from the larger block size permutation matrix.



(a) Generation of 16-point permutation matrix from 32-point permutation matrix



(b) Generation of 8-point permutation matrix from 16-point or 32-point permutation matrix



(c) Generation of 4-point permutation matrix from 8-point or 16-point or 32-point permutation matrix

Figure 3-13. Selection of smaller block size permutation matrix from the larger block size permutation matrix

3.5 Exceptional Cases

As the procedure to derive all the elements of the DST-7 transform kernel is already been mentioned in section 3.3.1, this section deals with the exception encountered while deriving the elements of 32-point DST-7 transform kernel.

3.5.1 Exceptional Condition in Proposed Grouping Concept

As the steps for deriving the elements of DST-7 transform kernel already been mentioned which tells that in each row the first element of the previous row should be equal to the last element of the resultant row obtained after multiplication with permutation matrix (**G**) without considering sign values and the process is repeated until the last element of the first specific stored row matches with the first-element of the generated row.

Using these steps, the exception can be viewed in the $[\mathbf{S}_{7,3}]_{32}$ row group where the $[\mathbf{S}_{7,3}]_{32}$ row is considered as the specific stored row as shown in Fig. 3-14.

3	30	56	77	88	89	80	63	38	9	-21	-50	-72	-85	-90	-84	-68	-45	-17	13	42	66	82	90	86	74	53	26	-4	-34	-60	-78	-88
25	56	-88	80	-38	-21	72	-90	68	-17	-42	82	-86	53	4	-60	88	-78	34	26	-74	90	-66	13	45	-84	85	-50	-9	63	-89	77	-30
18	88	-38	-72	68	42	-86	-4	88	-34	-74	66	45	-85	-9	89	-30	-77	63	50	-84	-13	90	-26	-78	60	53	-82	-17	90	-21	-80	56
4	38	68	86	88	74	45	9	-30	-63	-84	-90	-78	-53	-17	21	56	80	90	82	60	26	-13	-50	-77	-89	-85	-66	-34	4	42	72	88
23	68	-88	45	30	-84	78	-17	-56	90	-60	-13	77	-85	34	42	-88	72	-4	-66	89	-50	-26	82	-80	21	53	-90	63	9	-74	86	-38
14	88	30	-78	-56	60	77	-34	-88	4	89	26	-80	-53	63	74	-38	-86	9	90	21	-82	-50	66	72	-42	-85	13	90	17	-84	-45	68

Figure 3-14. $[\mathbf{S}_{7,3}]_{32}$ row group of 32-point DST-7 transform kernel

Based on the combination of the $[\mathbf{S}_{7,3}]_{32}$ row element and permutation matrix (**G**), other rows like $[\mathbf{S}_{7,25}]_{32}$, $[\mathbf{S}_{7,18}]_{32}$, $[\mathbf{S}_{7,4}]_{32}$, $[\mathbf{S}_{7,23}]_{32}$, and $[\mathbf{S}_{7,14}]_{32}$ rows should be obtained. The first element of the $[\mathbf{S}_{7,3}]_{32}$ row is

well-matched with the last element of the $[\mathbf{S}_{7,25}]_{32}$ row regardless of the sign value and the first element of the $[\mathbf{S}_{7,25}]_{32}$ row is matched with the last element of the $[\mathbf{S}_{7,18}]_{32}$ row and based on the step mentioned above, the process ends up as the last element of the $[\mathbf{S}_{7,3}]_{32}$ row gets matched with the first element of the $[\mathbf{S}_{7,18}]_{32}$ row. The consequence is that the remaining rows i.e. $[\mathbf{S}_{7,4}]_{32}$, $[\mathbf{S}_{7,23}]_{32}$, and $[\mathbf{S}_{7,14}]_{32}$ rows cannot be obtained.

In order to eliminate the exception which is only seen in the $[\mathbf{S}_{7,3}]_{32}$ row group, two approaches have been defined in section 3.5.2 and section 3.5.3.

3.5.2 Approach 1

This approach deals with the modification/tuning of the $[\mathbf{S}_{7,(3,31)}]_{32}$ i.e. 88 has been changed to 89 regardless of sign change as shown in Fig. 3-15. [Note: $[\mathbf{S}_{M,(N,P)}]_O$ *where M, N, P, and O are DST-7 transform kernel, row, column and size of the matrix respectively*].

3	30	56	77	88	89	80	63	38	9	-21	-50	-72	-85	-90	-84	-68	-45	-17	13	42	66	82	90	86	74	53	26	-4	-34	-60	-78	**-89**
25	56	-88	80	-38	-21	72	-90	68	-17	-42	82	-86	53	4	-60	**89**	-78	34	26	-74	90	-66	13	45	-84	85	-50	-9	63	-89	77	-30
18	88	-38	-72	68	42	-86	-4	**89**	-34	-74	66	45	-85	-9	89	-30	-77	63	50	-84	-13	90	-26	-78	60	53	-82	-17	90	-21	-80	56
4	38	68	86	**89**	74	45	9	-30	-63	-84	-90	-78	-53	-17	21	56	80	90	82	60	26	-13	-50	-77	-89	-85	-66	-34	4	42	72	88
23	68	**-89**	45	30	-84	78	-17	-56	90	-60	-13	77	-85	34	42	-88	72	-4	-66	89	-50	-26	82	-80	21	53	-90	63	9	-74	86	-38
14	**89**	30	-78	-56	60	77	-34	-88	4	89	26	-80	-53	63	74	-38	-86	9	90	21	-82	-50	66	72	-42	-85	13	90	17	-84	-45	68

Figure 3-15. Approach 1 for deriving 32-point DST-7 transform kernel
[Highlighted is the tuned value]

In Fig. 3-15, based on the proposed concept, the effect of tuning can be seen in other row elements as well, which is directly or indirectly dependent on the $[\mathbf{S}_{7,3}]_{32}$ row stored. The consequence of the changed value is that the first

element of the $[\mathbf{S}_{7,3}]_{32}$ row matches with the last element of the $[\mathbf{S}_{7,25}]_{32}$ row i.e. 30. Similarly, the first of the $[\mathbf{S}_{7,25}]_{32}$ row matches with the last element of the $[\mathbf{S}_{7,18}]_{32}$ row i.e. 56. Following the steps of the derivation of DST-7 transform kernel, the first element of the $[\mathbf{S}_{7,18}]_{32}$ row matches with the last element of the $[\mathbf{S}_{7,4}]_{32}$ row i.e. 88. Similarly, the first element of the $[\mathbf{S}_{7,4}]_{32}$ row matches with the last element of the $[\mathbf{S}_{7,23}]_{32}$ row i.e. 38. Finally, the first element of the $[\mathbf{S}_{7,23}]_{32}$ row matches with the last element of the $[\mathbf{S}_{7,14}]_{32}$ row i.e. 68 and consequently, the last element of the stored specific row i.e. $[\mathbf{S}_{7,3}]_{32}$ row's last element 89 matches with the first element of the $[\mathbf{S}_{7,14}]_{32}$ row regardless of the sign change.

Hence, approach 1 deals with the exception encountered during the generation of 32-point DST-7 transform kernel.

3.5.3 Approach 2

As approach 1, approach 2 also deals with the modification/tuning of one of the elements of 32-point DST-7 transform kernel which is shown in Fig. 3-16.

3	30	56	77	**89**	89	80	63	38	9	-21	-50	-72	-85	-90	-84	-68	-45	-17	13	42	66	82	90	86	74	53	26	-4	-34	-60	-78	-88
25	56	**-89**	80	-38	-21	72	-90	68	-17	-42	82	-86	53	4	-60	88	-78	34	26	-74	90	-66	13	45	-84	85	-50	-9	63	-89	77	-30
18	**89**	-38	-72	68	42	-86	-4	88	-34	-74	66	45	-85	-9	89	-30	-77	63	50	-84	-13	90	-26	-78	60	53	-82	-17	90	-21	-80	56
4	38	68	86	88	74	45	9	-30	-63	-84	-90	-78	-53	-17	21	56	80	90	82	60	26	-13	-50	-77	-89	-85	-66	-34	4	42	72	**89**
23	68	-88	45	30	-84	78	-17	-56	90	-60	-13	77	-85	34	42	**-89**	72	-4	-66	89	-50	-26	82	-80	21	53	-90	63	9	-74	86	-38
14	88	30	-78	-56	60	77	-34	**-89**	4	89	26	-80	-53	63	74	-38	-86	9	90	21	-82	-50	66	72	-42	-85	13	90	17	-84	-45	68

Figure 3-16. Approach 2 for deriving 32-point DST-7 transform kernel
[Highlighted is the tuned values]

This approach deals with the modification of the $[\mathbf{S}_{7,(3,3)}]_{32}$ element value of the 32-point DST-7 transform kernel i.e. from "88" to "89" element value. The consequence of the tuned value is that the first element of the $[\mathbf{S}_{7,3}]_{32}$ row matches with the last-element of the $[\mathbf{S}_{7,25}]_{32}$ row i.e. 30 regardless of sign value. Similarly, the first element of the $[\mathbf{S}_{7,25}]_{32}$ row matches with the last element of the $[\mathbf{S}_{7,18}]_{32}$ row i.e. 56. Following steps mentioned while the generation of the DST-7 transform kernel, the first element of the $[\mathbf{S}_{7,18}]_{32}$ row matches with the last element of the $[\mathbf{S}_{7,4}]_{32}$ row i.e. 89. Similarly, the first element of the $[\mathbf{S}_{7,4}]_{32}$ row matches with the last element of the $[\mathbf{S}_{7,23}]_{32}$ row i.e. 38. The first element of the $[\mathbf{S}_{7,23}]_{32}$ row matches with the last element of the $[\mathbf{S}_{7,14}]_{32}$ row i.e. 68. Finally, the $[\mathbf{S}_{7,3}]_{32}$ group matrix is closed as the last element of the $[\mathbf{S}_{7,3}]_{32}$ row matches with the first element of the $[\mathbf{S}_{7,14}]_{32}$ row i.e. 88.

Hence, Approach 2 also deals with the exception encountered during the generation of 32-point DST-7 transform kernel.

Finally based on these two approaches using the relation as mentioned in (16), respective two tuned DCT-8 transform kernel can be derived.

3.6 DST-7 Transform Kernel Fast Algorithm

The fast algorithm [29] of DST-7 transform kernel has already been adopted. The details about the fast algorithm is described in section 3.6.1 based on the different features of DST-7 transform kernel. The proposed 16-point and 32-point DST-7 transform kernel well satisfy the features which are explained in section 3.6.2.

3.6.1 DST-7/DCT-8 Fast Algorithm

The fast algorithm of DST-7/DCT-8 is based on three features for an N-point which is described as:

Feature #1: In some basis vectors which contain N distinct numbers without considering the sign changes, it is observed that the sum of several (2 or 3) numbers equals to the sum of another several (1 or 2) numbers.

Feature #2: Replicate patterns with symmetric or anti-symmetric characteristics that can be observed in some basis vectors.

Feature #3: There is another one or two basis vector(s) which only contain(s) very few (1 or 2) distinct number(s) without considering the sign changes

To explain the fast methods utilizing these features, an example from 16-point DST-7 transform can be explained. Before explaining all those features let us consider the elements of DST-7 transform kernel as

```
{ a  b  c  d  e  f  g  h  i  j  k  l  m  n  o  p }
{ c  f  i  l  o  o  l  i  f  c  0 -c -f -i -l -o }
{ e  j  o  m  h  c -b -g -l -p -k -f -a  d  i  n }
{ g  n  l  e -b -i -p -j -c  d  k  o  h  a -f -m }
{ i  o  f -c -l -l -c  f  o  i  0 -i -o -f  c  l }
{ k  k  0 -k -k  0  k  k  0 -k -k  0  k  k  0 -k }
{ m  g -f -n -a  l  h -e -o -b  k  i -d -p -c  j }
{ o  c -l -f  i  i -f -l  c  o  0 -o -c  l  f -i }
{ p -a -o  b  n -c -m  d  l -e -k  f  j -g -i  h }
{ n -e -i  j  d -o  a  m -f -h  k  c -p  b  l -g }
{ l -i -c  o -f -f  o -c -i  l  0 -l  i  c -o  f }
{ j -m  c  g -p  f  d -n  i  a -k  l -b -h  o -e }
{ h -p  i -a -g  o -j  b  f -n  k -c -e  m -l  d }
{ f -l  o -i  c  c -i  o -l  f  0 -f  l -o  i -c }
{ d -h  l -p  m -i  e -a -c  g -k  o -n  j -f  b }
{ b -d  f -h  j -l  n -p  o -m  k -i  g -e  c -a }
```

where `{a,b,c,d,e,f,g,h,i,j,k,l,m,n,o,p}` are the N unique numbers which are specified by the formulation of DST-7.

Example of **Feature #1**

It is noted that the element values have the following characteristics.

```
a + j = l
b + i = m
```

$$c + h = n$$

$$d + g = o$$

$$e + f = p$$

Therefore, to calculate $y0$, instead of doing the following vector-by-vector multiplication:

$$y0 = a \cdot x0 + b \cdot x1 + \ldots + p \cdot x15$$

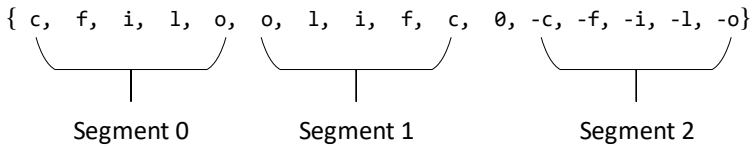
which requires 16 multiplications. The following alternative implementation can be done to derive the same results:

$$y0 = a \cdot (x0+x11) + b \cdot (x1+x12) + \ldots + j \cdot (x9+x11) + k \cdot x10$$

which requires 10 multiplications. It is observed that the number of multiplications is reduced.

Example of **Feature #2**

The second basis vector is



which can be divided into three segments, and they are replicated with sign changes or flipped versions of each other. Based on Feature #2, when calculating $y1$, instead of doing the following vector-by-vector multiplication

$$y1 = c \cdot x0 + f \cdot x1 + \ldots - o \cdot x15$$

which requires 16 multiplications. The following alternative implementation can be used to derive the same results.

```
y1 = c·(x0+x9-x11) + f·(x1+x8-x12) + i·(x2+x7-x13) + l·(x3+x6-x14) + o·(x4+x5-x15)
```

which only requires 5 multiplications. It is reported the number of multiplications is reduced.

Example of **Feature #3**

It is observed that the 6th basis vector-only contains 1 distinct number without considering the sign changes as shown below.

```
{k, k, 0, -k, -k, 0, k, k, 0, -k, -k, 0, k, k, 0, -k}
```

Instead of taking the vector-by-vector multiplication of

```
y5 = k·x0 + k·x1 + … - k·x15
```

which requires 11 multiplications, `y5` can also be achieved by the following operations:

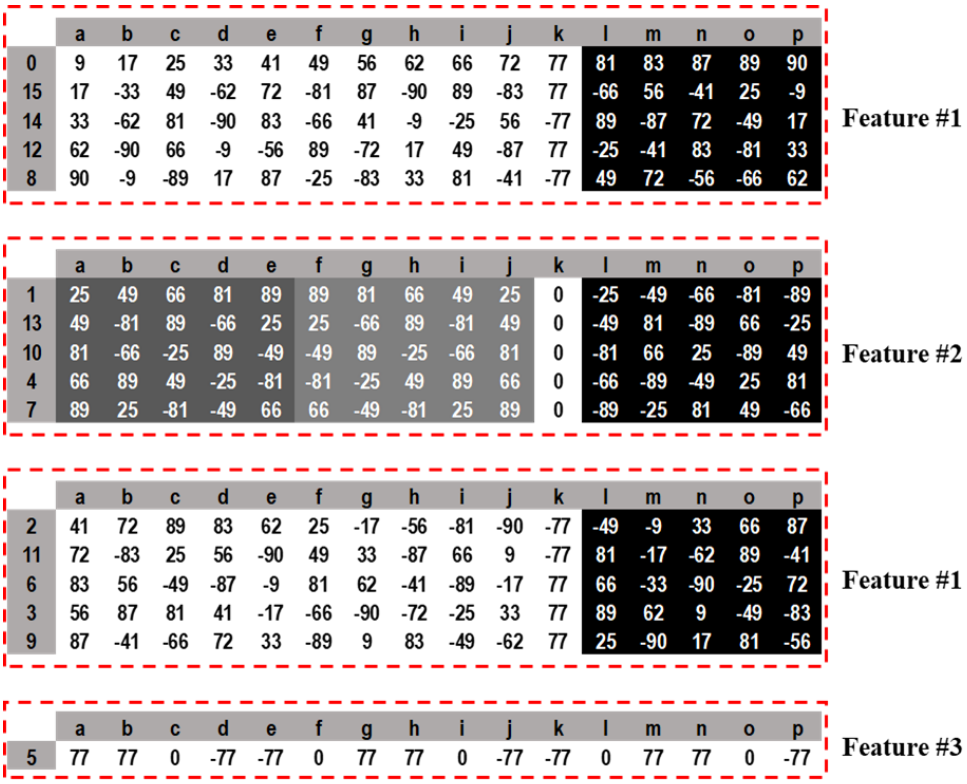
```
y5 = k·(x0+x1- … -x15)
```

which only requires 1 multiplication

3.6.2 Fast Algorithm Implementation on The Proposed DST-7

The major drawback of the fast algorithm of DST-7 transform kernel is that in different rows different features have to be applied. It becomes difficult to apply the features row-wise as there should be match in each row that it follows a certain feature or not. For the larger DST-7 transform kernel implementing this feature becomes really tedious and time-consuming.

The proposed grouping concept introduced in this thesis solves the problem by implementing a particular feature in certain row group elements. The feature applied to one row is applied to all the row group elements that are dependent on that row. This grouping concept really becomes beneficial for the large block size DST-7 transform kernel as individual selection of features for each row is not need to be performed.



	a	b	c	d	e	f	g	h	i	j	k	l	m	n	o	p	
0	9	17	25	33	41	49	56	62	66	72	77	81	83	87	89	90	
15	17	-33	49	-62	72	-81	87	-90	89	-83	77	-66	56	-41	25	-9	
14	33	-62	81	-90	83	-66	41	-9	-25	56	-77	89	-87	72	-49	17	Feature #1
12	62	-90	66	-9	-56	89	-72	17	49	-87	77	-25	-41	83	-81	33	
8	90	-9	-89	17	87	-25	-83	33	81	-41	-77	49	72	-56	-66	62	

	a	b	c	d	e	f	g	h	i	j	k	l	m	n	o	p	
1	25	49	66	81	89	89	81	66	49	25	0	-25	-49	-66	-81	-89	
13	49	-81	89	-66	25	25	-66	89	-81	49	0	-49	81	-89	66	-25	
10	81	-66	-25	89	-49	-49	89	-25	-66	81	0	-81	66	25	-89	49	Feature #2
4	66	89	49	-25	-81	-81	-25	49	89	66	0	-66	-89	-49	25	81	
7	89	25	-81	-49	66	66	-49	-81	25	89	0	-89	-25	81	49	-66	

	a	b	c	d	e	f	g	h	i	j	k	l	m	n	o	p	
2	41	72	89	83	62	25	-17	-56	-81	-90	-77	-49	-9	33	66	87	
11	72	-83	25	56	-90	49	33	-87	66	9	-77	81	-17	-62	89	-41	
6	83	56	-49	-87	-9	81	62	-41	-89	-17	77	66	-33	-90	-25	72	Feature #1
3	56	87	81	41	-17	-66	-90	-72	-25	33	77	89	62	9	-49	-83	
9	87	-41	-66	72	33	-89	9	83	-49	-62	77	25	-90	17	81	-56	

	a	b	c	d	e	f	g	h	i	j	k	l	m	n	o	p	
5	77	77	0	-77	-77	0	77	77	0	-77	-77	0	77	77	0	-77	Feature #3

Figure 3-17. Implementation of features in derived 16-point grouped DST-7 transform kernel

Fig. 3-17. shows the grouping concept applied to the 16-point DST-7 transform kernel. After applying the grouping concept in the DST-7 transform kernel, the respective groups only follows the specific feature described in section 7.1. This makes easy to separate which feature has to be used in a particular row of the transform kernel.

Feature 1

Feature 1 is supported for the groups whose specific stored rows are $[\mathbf{S}_{7,0}]_{16}$ and $[\mathbf{S}_{7,2}]_{16}$. This feature is applied to all the rows which directly or indirectly depend on the $[\mathbf{S}_{7,0}]_{16}$ row and $[\mathbf{S}_{7,2}]_{16}$ row.

Feature 2

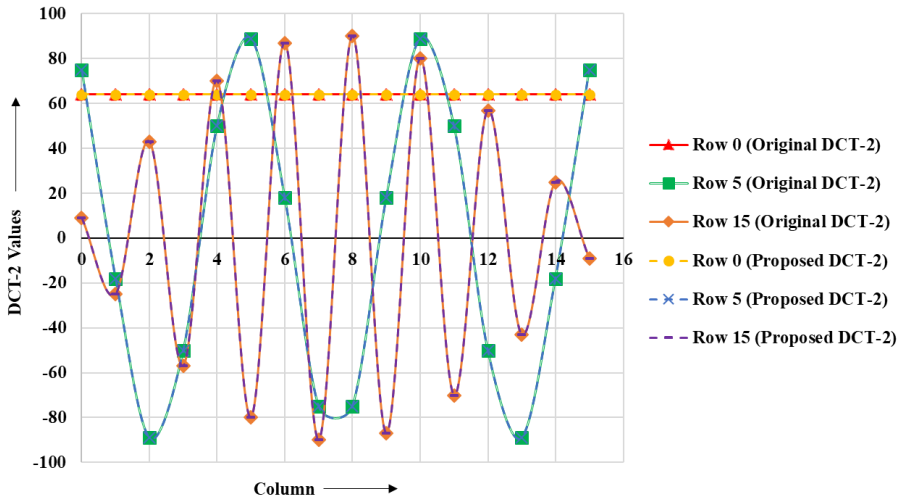
Feature 2 is supported for the group whose specific stored row is $[\mathbf{S}_{7,1}]_{16}$ row. This feature is applied to all the rows which directly or indirectly depend on the $[\mathbf{S}_{7,1}]_{16}$ row.

Feature 3

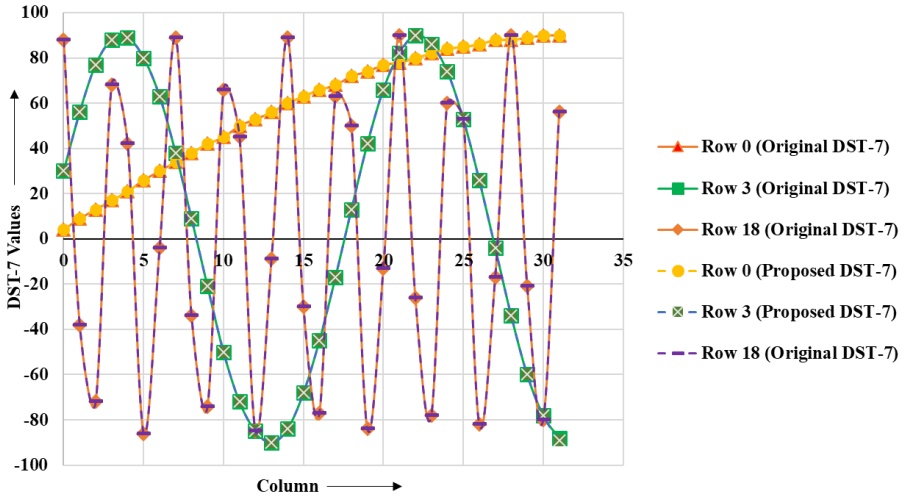
Feature 3 is supported for the group whose specific stored row is the $[\mathbf{S}_{7,5}]_{16}$ row. As no other rows depends on this specific stored row hence, only in the $[\mathbf{S}_{7,5}]_{16}$ row this feature is applied.

4. Simulation Results and Discussions

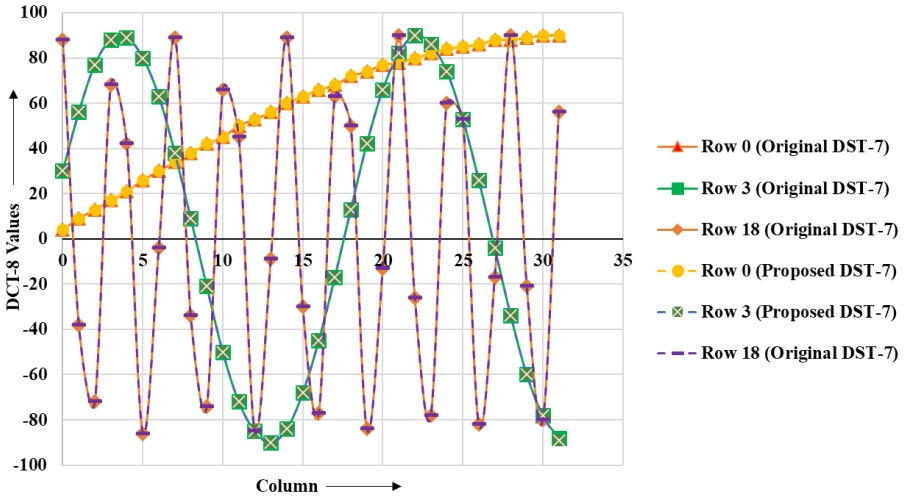
4.1 Simulation Results of Proposed methods



(a) Original DCT-2 Vs Proposed DCT-2 (16-point Approach 1 and Approach 2)

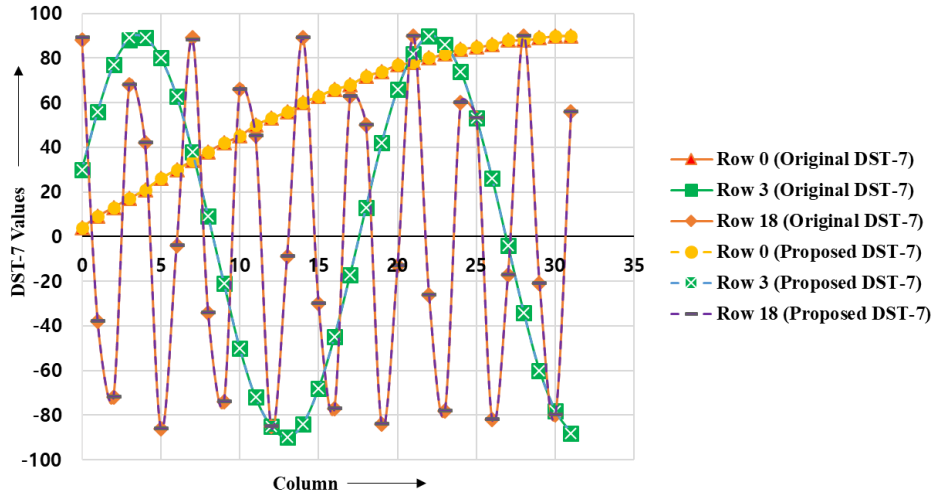


(b) Original DST-7 Vs Proposed DST-7 (32-point Approach 1)

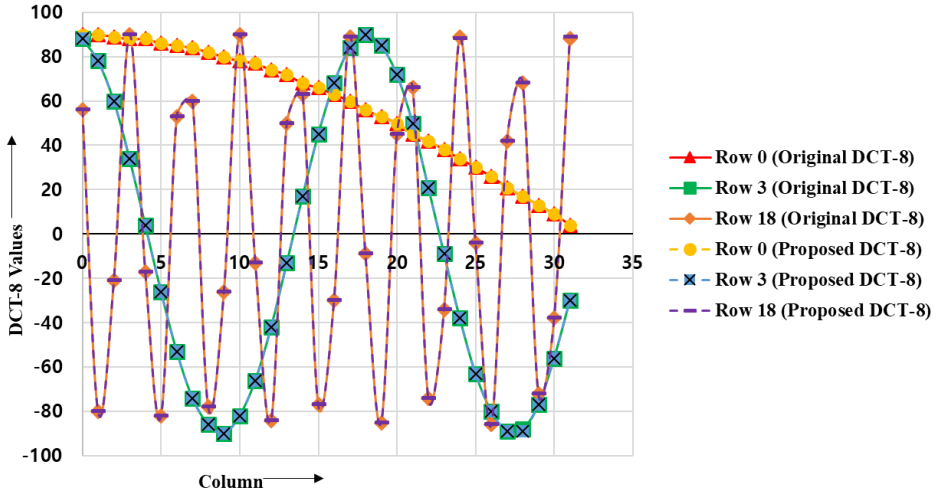


(c) Original DCT-8 Vs Proposed DCT-8 (32-point Approach 1)

Figure 4-1. Comparison of the proposed and original transform kernels (Approach 1)



(a) Original DST-7 Vs Proposed DST-7 (32-point Approach 2)



(b) Original DCT-8 Vs Proposed DCT-8 (32-point Approach 2)

Figure 4-2. Comparison of the proposed and original transform kernels (Approach 2)

In order to verify the derived transform kernels obtained from the proposed method, the original transform kernels obtained from VTM-3.0 [30] reference software is matched with the proposed transform kernels. Fig. 4-1. and Fig. 4-2. shows the comparison of similarities between original transform kernel (DCT-2, DST-7 and DCT-8) obtained from VTM-3.0 [30] reference software and the proposed derived transform kernels (DCT-2, DST-7 (approach 1 and 2) and DCT-8 (approach 1 and 2)). Based on figures, it can be seen that the derived transform kernels are closely overlapped with the original transform kernels. This indicates that the original transform kernels are well approximated by the proposed transform kernels and are more likely to have identical coding gains with the original transform kernels.

Similarly, the simulation of the proposed method is conducted on the VVC reference software VTM-3.0[30] using the CTC condition [31]. The PC for

simulation has Centos 7 OS with intel ®Xeon ® Silver 4114 CPU @ 2.20 GHz processor with 20 cores and 156 GB of RAM. The simulation results of approach 1 and approach 2 are shown in Table.6 and Table.10 respectively

Table 6. Overall simulation results of approach 1

	All Intra Main10 of Approach 1				
	Over VTM-3.0				
	Y	U	V	EncT	DecT
Class A1	0.00%	-0.08%	0.00%	100%	100%
Class A2	0.00%	-0.04%	0.00%	100%	100%
Class B	0.00%	0.02%	0.06%	100%	100%
Class C	0.00%	-0.04%	-0.07%	100%	100%
Class E	0.00%	0.04%	0.00%	100%	100%
Overall	0.00%	-0.02%	0.00%	100%	100%
Class D	0.00%	0.00%	0.10%	102%	100%

	Random access Main10 of Approach 1				
	Over VTM-3.0				
	Y	U	V	EncT	DecT
Class A1	0.01%	0.07%	0.06%	101%	100%
Class A2	-0.04%	0.08%	0.08%	101%	100%
Class B	0.00%	0.18%	0.01%	100%	100%
Class C	0.00%	0.01%	0.10%	101%	100%
Overall	-0.01%	0.09%	0.06%	101%	100%
Class D	-0.01%	-0.20%	-0.15%	101%	100%

	Low delay B Main10 of Approach 1				
	Over VTM-3.0				
	Y	U	V	EncT	DecT
Class B	-0.01%	-0.29%	-0.42%	100%	100%
Class C	0.06%	0.21%	0.10%	101%	100%
Class E	-0.02%	0.54%	-0.51%	101%	100%
Overall	0.01%	0.08%	-0.27%	101%	100%
Class D	-0.03%	0.08%	-0.21%	101%	100%

Table. 6. shows the simulation results of Approach 1 based on the VTM-3.0 anchor under CTC condition. For AI configuration, there is no loss in luminance whereas there is 0.08% chroma (U) gain in class A1 and negligible loss i.e. 0.10% in class D in chroma (V). The overall results for 0.00% in luminance and a 0.02% gain in chrominance (U).

For RA configuration, the maximum gain for luminance is 0.04% for class A2 and a negligible loss of 0.01% for class A1. Similarly, for chrominance (U), the maximum RA gain is 0.20% for class D and a negligible loss of 0.18% for class B. The overall result for RA configuration is a gain of 0.01% gain for luminance and loss of 0.09% i.e. maximum negligible loss for chrominance.

For LDB configuration, the maximum gain for luminance is 0.03% for class D and loss of 0.06% for class C. The chrominance gain (V) is 0.51% for class E and loss is 0.10% for class C. Similarly, chrominance (U) gain of 0.29% for class B and loss is 0.54% for class E. The overall result for luminance is negligible 0.01% loss and negligible loss of 0.08% in U chrominance and in V chrominance, there is a gain of 0.27%.

The detail experimental result of approach 1 using different test sequences are as following

Table 7. Simulation results of Approach 1 (AI)

	Test Sequences	BD-rate (piecewise cubic)		
		Y	U	V
Class A1 (4K)	*Tango2*	-0.01%	-0.31%	0.08%
	FoodMarket4	0.01%	-0.01%	0.04%
	Campfire	-0.01%	0.08%	-0.10%
Class A2 (4K)	*CatRobot1*	-0.01%	-0.01%	0.01%
	DaylightRoad2	0.00%	-0.10%	0.02%
	ParkRunning3	0.01%	0.00%	-0.03%
Class B (1080p)	*MarketPlace*	0.02%	-0.13%	0.00%
	RitualDance	0.00%	0.07%	0.10%
	Cactus	0.01%	0.01%	0.05%
	BasketballDrive	-0.01%	-0.02%	0.07%
	BQTerrace	0.00%	0.15%	0.09%
Class C (WVGA)	*BasketballDrill*	0.01%	-0.18%	-0.12%
	BQMall	-0.01%	0.15%	-0.06%
	PartyScene	0.00%	-0.04%	-0.10%
	RaceHorses	0.01%	-0.09%	0.00%
Class D (WQVGA)	*BasketballPass*	0.01%	-0.17%	-0.08%
	BQSquare	0.00%	0.23%	0.28%
	BlowingBubbles	0.00%	-0.03%	0.03%
	RaceHorses	0.00%	-0.03%	0.18%
Class E (720p)	*FourPeople*	0.00%	-0.05%	-0.01%
	Johnny	-0.02%	0.18%	-0.07%
	KristenAndSara	0.02%	-0.02%	0.08%

Based on Table. 7 approach 1, no significant loss is found in AI for luminance and for chrominance, some gain is observed in the higher resolution classes but some negligible losses are observed in lower resolution classes.

Table 8. Simulation results of Approach 1 (RA)

	Test Sequences	BD-rate (piecewise cubic)		
		Y	U	V
Class A1 (4K)	*Tango2*	0.04%	0.14%	0.01%
	FoodMarket4	0.00%	0.11%	0.17%
	Campfire	0.00%	-0.04%	0.01%
Class A2(4K)	*CatRobot1*	-0.04%	0.10%	0.17%
	DaylightRoad2	-0.04%	0.19%	0.07%
	ParkRunning3	-0.03%	-0.06%	0.01%
Class B(1080p)	*MarketPlace*	-0.02%	0.28%	-0.13%
	RitualDance	-0.01%	-0.02%	0.00%
	Cactus	-0.03%	-0.04%	0.14%
	BasketballDrive	0.05%	-0.01%	-0.14%
	BQTerrace	0.01%	0.69%	0.17%
Class C(WVGA)	*BasketballDrill*	-0.01%	0.13%	0.15%
	BQMall	-0.02%	-0.12%	0.18%
	PartyScene	0.02%	0.18%	-0.18%
	RaceHorses	0.00%	-0.15%	0.24%
Class D(WQVGA)	*BasketballPass*	-0.03%	-0.09%	-0.17%
	BQSquare	-0.07%	-0.95%	-0.24%
	BlowingBubbles	-0.01%	0.01%	0.30%
	RaceHorses	0.08%	0.23%	-0.47%

Based on Table. 8 approach 1, no significant loss is seen in RA as well for both luminance and chrominance. Higher-resolution shows some gain whereas less significant loss is shown in the lower resolution test sequences.

Table 9. Simulation results of Approach 1 (LDB)

	Test Sequences	BD-rate (piecewise cubic)		
		Y	U	V
Class B(1080p)	*MarketPlace*	0.05%	-0.04%	-0.35%
	RitualDance	0.03%	-0.11%	-0.27%
	Cactus	-0.10%	-0.45%	-0.06%
	BasketballDrive	0.01%	-0.16%	-0.13%
	BQTerrace	-0.05%	-0.68%	-1.28%
Class C(WVGA)	*BasketballDrill*	0.04%	0.55%	0.10%
	BQMall	0.16%	0.61%	0.11%
	PartyScene	0.04%	-0.22%	-0.08%
	RaceHorses	-0.01%	-0.10%	0.28%
Class D(WQVGA)	*BasketballPass*	-0.09%	0.36%	-0.43%
	BQSquare	-0.02%	0.55%	0.59%
	BlowingBubbles	0.00%	-0.43%	-0.83%
	RaceHorses	-0.02%	-0.16%	-0.17%
ClassE(720p)	*FourPeople*	0.17%	-0.77%	-0.30%
	Johnny	-0.22%	2.81%	-0.22%
	KristenAndSara	0.00%	-0.43%	-1.00%

Table. 9 shows the detailed simulation results of LDB which shows mostly gains in luminance and chrominance in lower resolution test sequences.

From the above results, it is seen that approach 1 results are very close to the VTM-3.0 anchor. This signifies that the transform kernel obtained from the proposed method is significantly close to the original transform kernel.

Table 10. Overall simulation results of Approach 2

	All Intra Main10 of Approach 2				
	Over VTM-3.0				
	Y	U	V	EncT	DecT
Class A1	0.00%	-0.11%	0.02%	100%	100%
Class A2	0.00%	-0.01%	0.02%	99%	100%
Class B	0.00%	0.01%	0.07%	100%	100%
Class C	0.00%	-0.06%	-0.10%	100%	100%
Class E	0.00%	0.01%	0.03%	100%	100%
Overall	0.00%	-0.03%	0.01%	100%	100%
Class D	0.00%	0.16%	-0.06%	102%	100%

	Random access Main10 of Approach 2				
	Over VTM-3.0				
	Y	U	V	EncT	DecT
Class A1	0.00%	-0.07%	0.07%	100%	100%
Class A2	-0.02%	0.14%	0.02%	100%	100%
Class B	-0.01%	0.26%	0.06%	101%	100%
Class C	0.02%	-0.03%	0.04%	100%	100%
Overall	0.00%	0.09%	0.05%	100%	100%
Class D	0.00%	0.10%	0.07%	101%	100%

	Low delay B Main10 of Approach 2				
	Over VTM-3.0				
	Y	U	V	EncT	DecT
Class B	0.02%	-0.16%	0.23%	96%	100%
Class C	0.02%	0.34%	0.19%	101%	100%
Class E	0.00%	0.99%	-0.84%	101%	100%
Overall	0.01%	0.30%	-0.05%	99%	100%
Class D	0.00%	0.48%	0.15%	101%	100%

Table. 10 shows the simulation results of approach 2 which is based on VTM-3.0 anchor under CTC condition. For the AI configuration, there is no change in luminance. In the luminance, the maximum gain is 0.11% for class A1 and loss is 0.16% for class D. The overall result shows that there is no change in luminance whereas there is a gain of 0.03% and loss of 0.01% i.e. regarded as negligible in chrominance.

For the RA configuration, 0.02% is the maximum luminance gain for class A2, 0.02% is the maximum luminance loss i.e. for class C. Regarding chrominance, the maximum gain is 0.07% for class A1, and maximum loss is 0.26% for class B. The overall results show that there is neither gain nor loss in luminance and a very negligible loss of 0.09% in chrominance.

For the LDB configuration, there is negligible loss of 0.02% for luminance in class B and C. Similarly, the maximum gain for chrominance is 0.84% in class E and loss of 0.99% in class E. The overall results show that there is negligible loss of 0.01% in luminance and loss of 0.30% and gain of 0.05% in chrominance.

The detail experimental result of approach 2 using different test sequences are as following

Table 11. Simulation results of Approach 2 (AI)

	Test Sequences	BD-rate (piecewise cubic)		
		Y	U	V
Class A1 (4K)	*Tango2*	0.00%	-0.45%	0.04%
	FoodMarket4	0.01%	0.07%	0.11%
	Campfire	0.00%	0.04%	-0.09%
Class A2(4K)	*CatRobot1*	0.00%	-0.06%	0.06%
	DaylightRoad2	-0.01%	0.01%	0.00%
	ParkRunning3	0.00%	0.01%	-0.01%
Class B(1080p)	*MarketPlace*	-0.01%	-0.05%	0.08%
	RitualDance	0.01%	0.00%	0.06%
	Cactus	0.01%	0.01%	0.12%
	BasketballDrive	-0.01%	0.05%	-0.05%
	BQTerrace	0.00%	0.04%	0.12%
Class C(WVGA)	*BasketballDrill*	0.02%	0.02%	-0.02%
	BQMall	-0.01%	-0.06%	-0.02%
	PartyScene	0.00%	-0.01%	-0.12%
	RaceHorses	0.00%	-0.19%	-0.24%
Class D(WQVGA)	*BasketballPass*	0.05%	0.08%	-0.13%
	BQSquare	-0.01%	0.20%	0.07%
	BlowingBubbles	-0.01%	0.07%	-0.06%
	RaceHorses	-0.03%	0.29%	-0.11%
ClassE(720p)	*FourPeople*	0.00%	0.07%	0.01%
	Johnny	-0.03%	-0.09%	-0.12%
	KristenAndSara	0.02%	0.06%	0.19%

Table. 11 shows the detail simulation result of approach 2 where no significant loss is found in AI for luminance and for chrominance, some gain is observed in the higher resolution classes. The highest gain observed for

chrominance is 0.45% for Class A1 Tango2 test sequence and the highest loss observed for chrominance is 0.29% for the Class D RaceHorses test sequence.

Table 12. Simulation results of Approach 2 (RA)

	Test Sequences	BD-rate (piecewise cubic)		
		Y	U	V
Class A1 (4K)	*Tango2*	0.00%	-0.21%	-0.06%
	FoodMarket4	0.01%	0.02%	0.16%
	Campfire	-0.01%	-0.02%	0.10%
Class A2(4K)	*CatRobot1*	-0.02%	0.19%	0.17%
	DaylightRoad2	-0.01%	0.32%	-0.08%
	ParkRunning3	-0.03%	-0.07%	-0.03%
Class B(1080p)	*MarketPlace*	-0.04%	0.33%	0.10%
	RitualDance	0.01%	0.13%	0.12%
	Cactus	-0.04%	-0.10%	-0.15%
	BasketballDrive	0.01%	-0.11%	-0.09%
	BQTerrace	0.01%	1.03%	0.31%
Class C(WVGA)	*BasketballDrill*	0.01%	-0.11%	0.13%
	BQMall	0.00%	0.23%	-0.21%
	PartyScene	0.01%	0.04%	0.16%
	RaceHorses	0.06%	-0.27%	0.10%
Class D(WQVGA)	*BasketballPass*	-0.02%	0.47%	-0.10%
	BQSquare	-0.10%	-0.19%	-0.40%
	BlowingBubbles	-0.02%	0.14%	0.58%
	RaceHorses	0.15%	-0.03%	0.20%

Table. 12 shows the detail simulation result of approach 2 where no significant loss is found in RA for luminance but for chrominance, 1.03% loss is observed in class B BQTerrace test sequence and 0.27% gain in class C RaceHorses test sequence. In spite of high loss in RA, the overall result for all test sequences is showing negligible loss for chrominance.

Table 13. Simulation results of Approach 2 (LDB)

	Test Sequences	BD-rate (piecewise cubic)		
		Y	U	V
Class B(1080p)	*MarketPlace*	0.05%	0.02%	0.22%
	RitualDance	0.04%	-0.10%	-0.03%
	Cactus	-0.06%	-0.52%	0.28%
	BasketballDrive	0.02%	-0.42%	-0.12%
	BQTerrace	0.04%	0.24%	0.79%
Class C(WVGA)	*BasketballDrill*	-0.01%	0.67%	0.66%
	BQMall	0.06%	0.56%	0.34%
	PartyScene	0.03%	0.08%	-0.14%
	RaceHorses	-0.01%	0.04%	-0.10%
Class D(WQVGA)	*BasketballPass*	0.02%	0.88%	-0.53%
	BQSquare	-0.10%	0.46%	0.73%
	BlowingBubbles	0.06%	0.65%	0.40%
	RaceHorses	0.00%	-0.07%	0.00%
ClassE(720p)	*FourPeople*	0.14%	0.14%	-0.07%
	Johnny	-0.13%	1.74%	-1.28%
	KristenAndSara	-0.02%	1.09%	-1.18%

Table. 13 shows the LDB detail simulation result of approach 2. The maximum gain for luminance is 0.13% for class E Johny test sequence and maximum loss is 0.14% for class E FourPeople test sequence. Similarly, for chrominance, the maximum loss is 1.74% for Class E Johny test sequence and gain is 1.28 for V chrominance. The maximum gain and loss are observed in class E test sequences. The overall result is also showing some losses of 0.30% in chrominance but a negligible loss of 0.01% in luminance.

4.2 Approach 1 and Approach 2 Comparison

Comparing the approach 1 and approach 2, both comprises of negligible losses based on the VTM-3.0 anchor. Although approach 1 and approach 2 results are close to each other but base on the results of LDB, approach 1 is better than approach 2. In approach 2, there exist a loss of 0.30% in U chroma whereas there is a negligible loss of only 0.08% in U chroma of LDB in approach 1. Similarly, the LDB gain of V chroma of approach 1 is higher than approach 2 i.e. a 0.27% gain in comparison to a 0.05% gain. While analyzing the overall result and detail result of different test sequences, it can be said that approach 1 is better than approach 2 as approach 1 provides small gain over VTM-3.0.

4.3 Comparison Between Proposed Method and COT

While comparing the proposed obtained results with the COT [22], the simulation results of COT are shown in the Table. 14. Analyzing the results, although the results of COT are better than the proposed, 64-point DCT-2 cannot be achieved, as kernel elements of DST-4 and DST-7 are embedded in 64-point DCT-2. This is the main reason why better results for MTS turned off cannot be achieved where only different point DCT-2 transform kernel is used. Similarly, the memory that has to be stored in COT is 32768 bytes (64-point) whereas the proposed method stores only 13168 bytes memory and after some computation, the same transform kernels of different sizes can be achieved.

Table 14. Simulation results of COT

	All Intra Main10				
	Over VTM-3.0				
	Y	U	V	EncT	DecT
Class A1	-0.01%	-0.01%	0.09%	101%	100%
Class A2	-0.05%	-0.05%	-0.01%	102%	100%
Class B	-0.14%	-0.13%	-0.12%	102%	98%
Class C	-0.29%	-0.21%	-0.31%	101%	99%
Class E	-0.34%	-0.28%	-0.13%	98%	98%
Overall	-0.17%	-0.14%	-0.11%	101%	99%
Class D	-0.31%	-0.32%	-0.31%	104%	104%

	Random access Main10				
	Over VTM-3.0				
	Y	U	V	EncT	DecT
Class A1	0.00%	0.02%	0.05%	101%	99%
Class A2	-0.02%	0.17%	0.09%	99%	99%
Class B	-0.07%	0.06%	0.01%	101%	99%
Class C	-0.11%	-0.05%	0.04%	101%	100%
Overall	-0.06%	0.05%	0.04%	101%	99%
Class D	-0.14%	-0.44%	-0.34%	101%	101%

	Low delay B Main10				
	Over VTM-3.0				
	Y	U	V	EncT	DecT
Class B	-0.02%	-0.26%	0.08%	100%	100%
Class C	0.01%	0.44%	0.17%	99%	96%
Class E	0.10%	1.06%	0.16%	102%	98%
Overall	0.02%	0.30%	0.13%	100%	98%
Class D	0.02%	0.20%	0.08%	104%	102%

4.4 Comparison Between Proposed Method and Unified Matrix

Based on the results from Table. 15, some gain is shown in the AI configuration. However gain is seen in the overall result of AI configuration, higher classes i.e. A1 and A2 are showing high losses. Similarly, loss is seen in the RA and LDB configuration as well. The losses in class A1 and A2 along with RA and LDB losses are due to the mismatching of proposed DST-7 and DCT-8 with the original DST-7 and DCT-8 transform kernels. Similarly, the memory needed to store here in this proposal is 32768 bytes (64-points) whereas the proposed method in this paper stores only 13168 bytes. The simulation results proposed in [24] are shown in Table. 15.

Table 15. Simulation results of Unified matrix

	All Intra Main10				
	Over VTM-3.0				
	Y	U	V	EncT	DecT
Class A1	0.20%	0.16%	0.30%	101%	100%
Class A2	0.15%	0.09%	0.09%	101%	100%
Class B	-0.02%	-0.15%	-0.14%	101%	99%
Class C	-0.30%	-0.35%	-0.41%	102%	100%
Class E	-0.25%	-0.35%	-0.29%	100%	98%
Overall	-0.06%	-0.14%	-0.12%	101%	99%
Class D	-0.35%	-0.36%	-0.47%	103%	102%

	Random access Main10				
	Over VTM-3.0				
	Y	U	V	EncT	DecT
Class A1	0.18%	0.26%	0.32%	101%	101%
Class A2	0.14%	0.19%	0.31%	99%	99%
Class B	0.07%	0.17%	-0.10%	100%	100%
Class C	-0.07%	-0.04%	0.05%	98%	96%
Overall	0.07%	0.14%	0.11%	99%	99%
Class D	-0.14%	-0.45%	-0.26%	98%	96%

	Low delay B Main10				
	Over VTM-3.0				
	Y	U	V	EncT	DecT
Class B	0.17%	0.02%	0.33%	98%	96%
Class C	0.15%	0.34%	0.39%	98%	98%
Class E	0.19%	0.81%	-0.36%	96%	88%
Overall	0.17%	0.32%	0.18%	97%	95%
Class D	0.06%	0.05%	-0.17%	98%	97%

4.5 Proposed Method and Adjustment Stages Comparison

Based on the results [25] from Table. 16, the overall result of AI showed negligible luminance loss of 0.03% and U chrominance loss of 0.05% and V chrominance loss of 0.11% with an Encoding time of 104%. In the higher test sequences, there is high loss in the chrominance whereas there is few loss in the luminance for all the test sequences. Similarly, for RA there is no gain i.e. luma loss of 0.04% and U and V chroma loss of 0.06% and 0.13% respectively.

Table 16. Simulation results of Adaptive Multiple Transforms (AMTs)

	All Intra Main10				
	Over VTM-3.0				
	Y	U	V	EncT	DecT
Class A1	0.06%	0.18%	0.19%	101%	97%
Class A2	0.06%	0.03%	0.10%	102%	97%
Class B	0.03%	0.09%	0.13%	102%	98%
Class C	0.01%	-0.01%	0.11%	105%	105%
Class E	0.03%	-0.06%	0.11%	109%	106%
Overall	0.03%	0.05%	0.11%	104%	101%
Class D	0.01%	0.05%	-0.01%	107%	109%

	Random Access Main10				
	Over VTM-3.0				
	Y	U	V	EncT	DecT
Class A1	0.03%	0.06%	0.14%	101%	99%
Class A2	0.04%	0.12%	0.22%	100%	100%
Class B	0.05%	-0.13%	0.05%	101%	99%
Class C	0.00%	0.04%	0.03%	99%	98%
Overall	0.04%	0.06%	0.13%	100%	99%
Class D	0.05%	0.26%	0.24%	100%	100%

4.6 Proposed Method and TAF Comparison

Based on the simulation results [27] shown in Table. 17, no AI gain is seen in the luminance and gain of 0.11% is seen in class D of U chrominance and in V chrominance small gain of 0.03% is seen in class C. The overall result is showing less significant loss. For the higher test sequence classes, there is showing high loss. Similarly, for RA higher test sequences are showing the

high loss and the overall result is a 0.06% loss for luma and 0.11% loss for both U and V chroma. For the LDB, the overall result is showing loss i.e. 0.03% loss of luma and 0.18% and 0.16% loss in U and V chroma respectively. The loss is seen due to the mismatch of the proposed DST-7/DCT-8 transform kernel signal with the original transform kernel signals.

Table 17. Simulation results using Transform Adjustment Filter (TAF)

	All Intra Main10				
	Over VTM-3.0				
	Y	U	V	EncT	DecT
Class A1	0.10%	0.12%	0.13%	95%	84%
Class A2	0.14%	0.08%	0.03%	97%	88%
Class B	0.07%	0.03%	0.10%	96%	88%
Class C	0.01%	0.07%	-0.03%	97%	93%
Class E	0.06%	0.04%	0.11%	96%	89%
Overall	0.07%	0.06%	0.06%	96%	88%
Class D	0.04%	-0.11%	0.08%	97%	96%
Class F	0.03%	0.04%	0.09%	97%	94%

	Random Access Main10				
	Over VTM-3.0				
	Y	U	V	EncT	DecT
Class A1	0.09%	0.11%	0.08%	99%	98%
Class A2	0.05%	0.20%	0.24%	99%	98%
Class B	0.05%	0.25%	0.12%	99%	98%
Class C	0.06%	-0.12%	0.04%	99%	100%
Overall	0.06%	0.11%	0.11%	99%	99%
Class D	-0.02%	-0.45%	-0.51%	99%	100%
Class F	0.01%	-0.09%	0.14%	99	98%

	Low delay B Main10				
	Over VTM-3.0				
	Y	U	V	EncT	DecT
Class B	0.05%	-0.17%	0.12%	100%	101%
Class C	0.01%	-0.01%	0.21%	100%	100%
Class E	0.02%	1.00%	0.16%	100%	99%
Overall	0.03%	0.18%	0.16%	100%	100%
Class D	-0.04%	0.21%	-0.17%	100%	100%
Class F	0.14%	0.36%	-0.60%	100%	99%

5. Conclusion

In this thesis, a kernel derivation method for the VVC video codec is proposed. Since the memory consumption to store different block size transform kernels is a major issue in video coding standardization of VVC, this thesis deals with storing fewer elements of transform kernels and deriving all the transform kernels based on that stored transform kernel elements. The proposed method uses a common unified sparse matrix which is derived from the DST-3 matrix and some specific rows derived from the selected rows from the DST-7 transform kernel. Similarly, to make compatible with the proposed method for DST-7 transform kernel, two approaches (Approach 1 and Approach 2) have been introduced which signifies the DST-7 transform kernel closely matches the proposed DST-7 transform kernel. The proposed method also supports the fast algorithm of DST-7 transform kernel which got adopted [21]. The proposed method in this thesis gives comparable results with the VTM-3.0 anchor with negligible loss which signifies that the proposed transform kernels are significantly close to the original signal in the VTM-3.0 anchor.

Acknowledgment

I would like to express my deepest gratitude to my advisor, Prof. Bumshik Lee, for his support, patience, and encouragement throughout my graduate study. His technical and editorial advice and words of encouragement and guideless were essential to complete this research successfully.

I am expressing my deep gratitude to my parents, brother and sisters for their love, understanding, supports, and encouragement during the period of this research.

Finally my deepest thanks to my fellow lab mates for their support throughout the course of my research.

References

[1] Thomas Wiegand, Gary J. Sullivan, Senior Member, IEEE, Gisle Bjøntegaard, and Ajay Luthra, Senior Member, IEEE, 'Overview of the H.264/AVC Video Coding Standard', IEEE TRANSACTIONS ON CIRCUITS AND SYSTEMS FOR VIDEO TECHNOLOGY, vol. 13, Jul. 2003.

[2] Kai Zhang, Li Zhang, Wei-Jung Chien, Marta Karczewicz, 'Intra-Prediction Mode Propagation for Video Coding', IEEE JOURNAL ON EMERGING AND SELECTED TOPICS IN CIRCUITS AND SYSTEMS, vol. 9, Mar. 2019.

[3] Jean Bégaint, 'Towards novel inter-prediction methods for image and video compression', Ph.D. thesis, University of Rennes 1, Comue University Britian Loire, Research unit: INRIA Rennes - Brittany Atlantic and Technicolor R&I, November 29, 2018. [Online]. Available: https://hal.archives-ouvertes.fr/tel-01960088/document

[4] V. K. Goyal, 'Theoretical foundations of transform coding', IEEE Signal Processing Magazine, vol. 18, no. 5, pp. 9-21, Sept. 2001

[5] Abedi, M.; Sun, B.; Zheng, Z., 'A Sinusoidal-Hyperbolic Family of Transforms With Potential Applications in Compressive Sensing', IEEE Transactions on Image Processing. 28 (7): 3571–3583, Jul. 2019.

[6] Anil K. Jain, 'A Sinusoidal Family of Unitary Transforms', IEEE trans. pattern anal. mach. intell, vol. PAMI-1, pp. 356-365, Oct. 1979.

[7] WK Cham, PhD, 'Development of Integer Cosine Transforms by the Principle of Dyadic Symmetry', Proc. Inst. Elect. Eng., pt. 1, vol. 136, no. 4, pp. 276-282, Aug. 1989.

[8] G. Strang, 'The Discrete Cosine Transform', SIAM Review, vol. 41 no. 1, pp. 135- 147, Mar. 1999.

[9] Iain Richardson, 'White Paper: 4x4 Transform and Quantization in H.264/AVC',2010.

[10] Sadiqullah Khan and Gulistan Raja. (2004), 'Integer cosine transform and Its application in Image/Video Compression', ICSEA, Conference proceeding Islamabad, pp.189-193.

[11] Richardson I.E. G . (2002), 'White Paper H.264/MPEG-4 Part 10: Variable Length Coding'.

[12] Press, WH; Teukolsky, SA; Vetterling, WT; Flannery, BP (2007). 'Section 22.6. Arithmetic Coding'. Numerical Recipes: The Art of Scientific Computing (3rd ed.). New York: Cambridge University Press. ISBN 978-0-521-88068-8.

[13] Rissanen, J.J.; Langdon G.G., Jr (March 1979). 'Arithmetic coding' (PDF). IBM Journal of Research and Development. 23 (2): 149–162. doi:10.1147/rd.232.0149V. K. Goyal, 'Theoretical foundations of transform coding', IEEE Signal Processing Magazine, vol. 18, no. 5, pp. 9-21, Sept. 2001

[14] Marpe, D., Schwarz, H., and Wiegand, T., 'Context-Based Adaptive Binary Arithmetic Coding in the H.264/AVC Video Compression Standard', IEEE Trans. Circuits and Systems for Video Technology, Vol. 13, No. 7, pp. 620–636, July 2003.

[15] 'Context-Based Adaptive Binary Arithmetic Coding (CABAC) ', Fraunhofer Heinrich Hertz Institute. Retrieved 13 July 2019

[16] S. Vetrivel, K. Suba, G.A Thisha, 'An Overview of H.26x Series And its Applications', International Journal of Engineering Science and Technology, Vol. 2(9), 2010, 4622-4631.

[17] T. Wiegand, G.J. Sullivan, G. Bjontegaard, A. Luthra, 'Overview of the H.264/AVC video coding standard', IEEE Transactions on Circuits and Systems for Video Technology, vol. 13, No. 7. (2003), pp. 560-576.

[18] Gary J. Sullivan, Jens-Rainer Ohm, Woo-Jin Han, Thomas Wiegand, 'Overview of the High Efficiency Video Coding (HEVC) Standard', IEEE, pp. 1649–1668, Sep. 2012

[19] J. Chen and Y. Ye, S. H. Kim, 'Algorithm description for Versatile Video Coding and Test Model 3 (VTM 3),[JVET-L1002]', JVET of ITU-T SG 16 WP 3 and ISO/IEC JTC 1/SC 29/WG 11, 12th Meeting: Macao, CN, Oct. 2018.

[20] Rao, K. R.; Yip, P., 'Discrete Cosine Transform', IEEE Transactions on Computers. Boston: Academic Press, 1990.

[21] Gary J. Sullivan , Jens-Rainer Ohm, 'Meeting Report of the 13th Meeting of the Joint Video Experts Team (JVET)', Joint Video Experts Team (JVET) of ITU-T SG 16 WP 3 and ISO/IEC JTC 1/SC 29/WG 11, 13th Meeting: Marrakech, MA, 9–18 Jan. 2019.

[22] X. Zhao, X. Li, S. Liu, 'CE6: Compound Orthonormal Transform JVET-M0200', Joint Video Experts Team (JVET) of ITU-T SG 16 WP 3 and ISO/IEC JTC 1/SC 29/WG 11 13th Meeting: Marrakech, MA, Jan. 2019.

[23] Feig, E.; Winograd, S., 'Fast algorithms for the discrete cosine transform', IEEE Transactions on Signal Processing. 40 (9), pp. 2174–2193., Sep. 1992.

[24] K. Choi, M. Park, M.W. Park, W. Choi, 'CE6: Unified matrix for transform JVET-M0200', Joint Video Experts Team (JVET) of ITU-T SG 16 WP 3 and ISO/IEC JTC 1/SC 29/WG 11 13th Meeting: Marrakech, MA, Jan. 2019.

[25] A. Said, H. Egilmez, V. Seregin, M. Karczewicz, 'Complexity Reduction for Adaptive Multiple Transforms (AMTs) using Adjustment Stages JVET-J0066', Joint Video Experts Team (JVET) of ITU-T SG 16 WP 3 and ISO/IEC JTC 1/SC 29/WG 11 10th Meeting: San Diego, US, Apr. 2018.

[26] JVET JEM software--https://jvet.hhi.fraunhofer.de/svn/svn_HMJEMSoftware/

[27] Pierrick Philippe, 'CE6: MTS simplification with TAF', Joint Video Experts Team (JVET) of ITU-T SG 16 WP 3 and ISO/IEC JTC 1/SC 29/WG 11 13th Meeting: Marrakech, MA, Jan. 2019.

[28] Zhaobin Zhang, Xin Zhao, Xiang Li, Li Li, Member, IEEE, Yi Luo, Shan Liu, and Zhu Li, Senior Member, IEEE, 'Fast DST-7/DCT-8 with Dual Implementation Support for Versatile Video Coding', IEEE TRANSACTIONS ON CIRCUITS AND SYSTEMS FOR VIDEO TECHNOLOGY, no. under review.

[29] Xin Zhao, Xiang Li, Yi Luo, Shan Liu, 'CE6: Fast DST-7/DCT-8 with dual implementation support JVET-M0497', Joint Video Experts Team (JVET) of ITU-T SG 16 WP 3 and ISO/IEC JTC 1/SC 29/WG 11 13th Meeting: Marrakech, MA.

[30] JVET VTM software — JVET. https://vcgit.hhi.fraunhofer.de/jvet/VVCSoftware_VTM/tree/VTM-3.0.

[31] P. Hanhart, J. Boyce, K. Choi, J.-L. Lin, 'JVET common test conditions and evaluation procedures for 360° video'. Joint Video Exploration Team (JVET) of ITU-T SG 16 WP 3 and ISO/IEC JTC 1/SC 29/WG 11 12th Meeting: Macau, CN.