August 2020
Master's Degree Thesis

# A study on coding complexity reduction of Low Frequency Non-Separable Transform (LFNST) for Versatile Video Codec (VVC)

Graduate School of Chosun University

Department of Information and Communication Engineering

Ankit Kumar

# A study on coding complexity reduction of Low Frequency Non-Separable Transform (LFNST) for Versatile Video Codec (VVC)

# VVC 비디오 코덱의 2차변환 LFNST의 부호화 복잡도 감소에 관한 연구

August 28, 2020

Graduate School of Chosun University

Department of Information and Communication Engineering

Ankit Kumar

# A study on coding complexity reduction of secondary Low Frequency Non-Separable Transform (LFNST) for Versatile Video Codec (VVC)

Advisor: Prof. Bumshik Lee

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

May 2020

Graduate School of Chosun University

Department of Information and Communication Engineering

Ankit Kumar

This is to certify that the master's thesis of

Ankit Kumar

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

| | | |
|---|---|---|
| **Committee Chairperson** | **Prof. Dong-you Choi** | |
| **Committee Member** | **Prof. Jae-Young Pyun** | |
| **Committee Member** | **Prof. Bumshik Lee** | |

June 2020

Graduate School of Chosun University

# Table of Contents

# List of Figures

# List of Tables

# Acronyms

| | |
|---|---|
| VVC | Versatile Video Codec |
| DCT | Discrete Cosine Transform |
| DST | Discrete Sine Transform |
| LFNST | Low frequency Non-Separable 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 |
| TU | Transform Unit |
| CU | Coding Unit |
| TB | Transform Block |
| CB | Coding Block |

## Abstract

# A study on coding complexity reduction of secondary Low Frequency Non-Separable Transform (LFNST) for Versatile Video Codec (VVC)

Ankit Kumar
Advisor: Prof. Bumshik Lee,
Department of Information and Communication Engineering
Graduate School of Chosun University

The transform is an essential module of the codec where the transformation of the residual block in pixel domain is converted to the coefficient block in frequency domain. In the standardization of Versatile Video Codec (VVC), the secondary transform is recently introduced as Low Frequency Non-Separable Transform (LFNST), to achieve better compression efficiency for the highly dynamic images with textures and edges in comparison to the primary transform. The secondary transform in VVC uses two sets of kernels of size 16×16 and 16×48 depending on the transform unit (TU) block size and indexing based on the intra prediction mode of the TU block. Although the secondary transform has a greater impact for the better coding efficiency and compression, its implementation is very complex and requires a lot of mathematical computations. Also, the storage of the secondary

transform kernels is a major concern as it takes 8KB of memory in VVC. In this thesis, a novel coding complexity reduction method is proposed that uses only 16×48 transform kernel for all TU block size, which reduces the overhead of selecting the transform kernel set depending upon the TU size and saves 2KB of memory while storing the transform kernels. Apparently, the indexing of the kernel based on the intra prediction mode is kept same for all TU's. The proposed method gives maximum of 16 non-zero output residue coefficients in the bitstream, by zeroing the primary transform coefficients reducing the mathematical computation complexity of the LFNST and quantization process. The proposed method also provides a novel approach to derive the sub-sampled kernel from 16×48 kernel to apply for the lower TU blocks of dimension 4×N and N×4 (N>=4). The experiments are performed in VTM-5.0 reference software under the common test conditions (CTC). The results on the Bjontegaard delta bitrate (BDBR) of the proposed method show no significant loss for All Intra (AI) and Random Access (RA) configuration. However, for AI configuration, it showed a gain of 0.02% for U-chroma with 94% encoding time and 98% decoding time, whereas for RA configuration, a benefit of 0.01% for V-chroma is obtained with 97% encoding time. Overall, the proposed method shows a substantial reduction by 6% in the encoding time.

**Index terms**- VVC, LFNST, BDBR, CTC, coding block, transform block

## 요 약

# VVC 비디오 코덱의 2차변환 LFNST의 부호화 복잡도 감소에 관한 연구

안킷 쿠마르
지도교수: 이범식
조선대학교대학원,
정보통신공학과

비디오코덱에서 변환은 픽셀 영역의 잔차 신호를 주파수 영역의 변환계수로 변환시키는 중요 툴 중의 하나이다. VVC (Versatile Video Codec) 비디오 표준 코덱에서, 1 차 변환 실행 후 1 차 변환 계수에 대해 다시 변환을 수행하는 2 차 변환이 채택되었으며, 2 차 변환은 텍스처 및 엣지가있는 동적인 이미지의 압축 효율을 높이기 위해 저주파 비 분리형 변환 변환 방법 (LFNST)을 사용한다. VVC 의 2 차 변환은 변환 단위 (TU) 블록 크기 및 TU 블록의 인트라 예측 모드에 따라 다른 커널이 할당되고, 16×16 및 16×48 크기의 두 커널 조합를 사용하여 비분리 변환과정을 수행한다. VVC 에서 2 차 변환은 큰 압축 효율 향상을 가져다 주었지만 그 복잡도가 크고 많은 계산량을 필요로 하고 변환 커널이 인트라모드와 변환 크기에 따라 매우 다양하게 정의된

커널을 사용하므로, 많은 양의 커널 데이터를 저장해야 한다는 단점이 존재한다. 본 학위 논문에서는 이러한 단점을 해결하기 위해 모든 TU 블록 크기에 16×48 변환 커널 만 사용하는 새로운 코딩 복잡도 감소 방법이 제안한다. 제안 방법을 사용하면 인트라 예측 모드에 기반한 커널의 인덱싱은 모든 TU 에 대해 동일하게 유지하면서 TU 크기에 따라 변환 커널 세트를 선택하는 오버헤드를 줄이고 변환을 저장하는 동안 2KB 의 메모리를 절약할 수 있다. 또한 제안된 방법은 1 차 변환 계수를 0 으로 처리함으로써 LFNST 및 양자화 프로세스의 계산 복잡성을 감소시켜 비트스트림에서 최대 16 개의 출력 잔차 계수를 제공한다. 제안된 방법은 크기가 4xN 및 Nx4 (N> = 4) 인 TU 블록에 적용하기 위해 16x48 커널에서 서브 샘플링 커널을 유도하는 방법도 포함된다. 제안된 방법은 공통 테스트 조건 (CTC)을 이용하여 VTM-5.0 참조 소프트웨어에서 수행되었고 복잡도 감소 및 커널 수의 감소에도 불구하고 BDBR (Bjontegaard delta bitrate) 결과는 AI (All Intra) 및 RA (Random Access) 조건에서 손실을 보이지 않았다. AI 경우 인코딩 시간이 94 %이고 디코딩 시간이 98 % 인 U-chroma 의 경우 0.02 %의 이득을 얻었으며, RA 의 경우 97 %의 인코딩 시간을 가지므로 V-chroma 의 경우 0.01 %의 이득을 보여주었다.

# 1. Introduction

## 1.1 Background

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

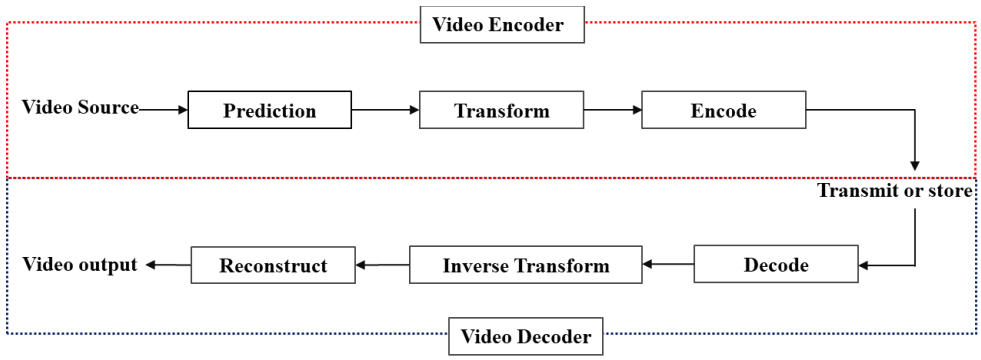


Figure 1-1. A basic block of the video codec

In video coding, initially the video source undergoes through the prediction module at the block level. The prediction coding refers to predicting the similarity of the picture region in a particular picture or frame. The codec has two kinds of prediction i.e., Inter prediction [2] and Intra prediction [3]. If the prediction is made from the previously coded block, from the current frame, it is said to be intra prediction. Whereas, if the prediction is made from other frames that has already been coded, it is referred as inter prediction. Finally, subtraction of the prediction from the current block is performed to obtain the predicted residual. After that, the transform and quantization process [4] is

performed. In transform, basic transform kernels such as DCT-2 [5], DST-7 [6], DCT-8 [7], DST-4, and DCT-4 [8] are used, where a block with the predicted residues is transformed using integer operations [9-10]. As a next step, the output of the transform process i.e., the non-zero coefficients are quantized, where each coefficient is quantized by quantization step size. The quantization reduces the precision of the transform coefficients according to a quantization parameter (QP). Higher QP values are likely to make coefficients zero, resulting in high compression at the expense of lower image or video quality. On the other hand, lower QP values are likely to have more non-zero coefficients resulting in better-decoded video or image quality but lower compression. After these steps, the encoding process is done where several 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 can then be stored or transmitted.

After all those encoding procedures, 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 is applied to re-create each block of residual data which combined 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 [14], H.262 [15], H.263 [16], H.264/Advanced Video Coding (AVC) [17], H.265/High Efficiency Video Coding (HEVC) [18]. Recently, the Versatile Video Coding (VVC) [19] is being standardized, whose main objective is to provide a significant improvement in compression performance over the existing HEVC standard [18] and also aid the deployment of higher-quality video services and emerging applications such as 360° omnidirectional immersive multimedia and HDR video [24].

## 1.2 Versatile Video Codec (VVC)

### 1.2.1 Structure of VVC

VVC is a codec which is designed by the collaborative team of ITU-T and ISO/IEC experts known as the Joint Video Experts Team (JVET). The codec is a successor to the HEVC and is designed to meet the upcoming need in the domain of video conferencing, OTT streaming, mobile telephony, and many other fields. The VVC codec is more efficient as it is targeted to provide approximately 50% bitrate reduction to the HEVC codec and high efficiency for compression. The codec is said to be versatile because it is targeted to address all of the video needs from low resolution and low bitrates to high resolution and high bitrates. The advantage of VVC over other prior codecs can be measured in terms of better bitrate gain, compression efficiency and addition of coding tools for different modules describes below. The overall block diagram of the encoder is illustrated in Fig. 1-2.

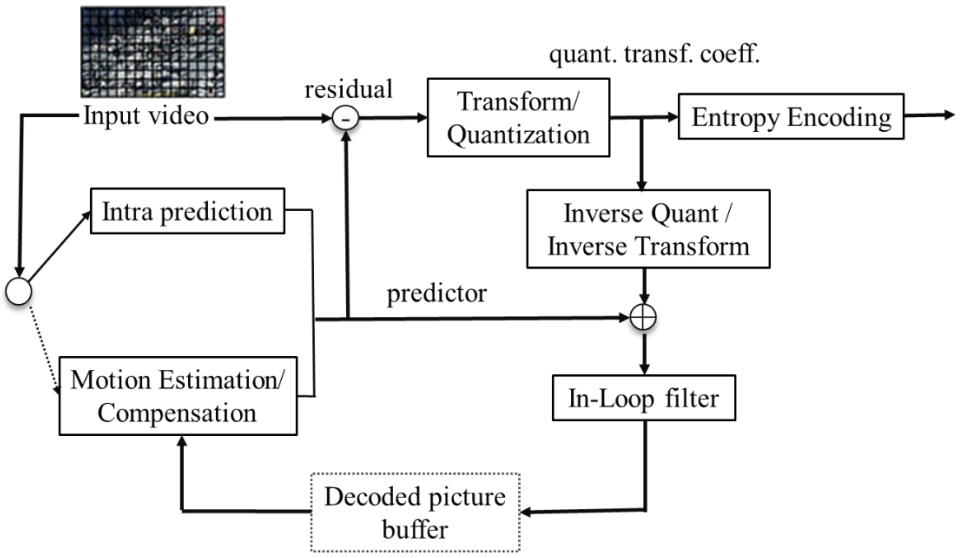


Figure 1-2. General block diagram of the VVC encoder.

The overall encoder side of the VVC shows the similar modules and coding techniques as the prior standards of codecs. Firstly, the input video is passed to the encoder, and the partitioning is performed on the coding units (CUs) based on the Multiple Tree Type (MTT) [27] partitioning which also supports the binary tree (BT) [1], ternary tree (TT) [1] and quad tree (QT) [1] structure for partitioning the block in to multiple CU's. The CU is a coding block of a luma sample and two corresponding coding blocks of chroma samples of a picture. The partitioned block is sent for the prediction module which can be either intra prediction or inter prediction and the output residual is sent for transformation in form of transform unit (TU). The TU is a transform block of luma samples and two corresponding transform blocks of chroma samples of a picture and syntax structures used to transform the transform block samples. The transform module has two sets of transform i.e., primary transform and secondary transform. The primary transform uses the kernels like DCT-2 [5], DST-7 [6], DCT-8 [7], DST-4, and DCT-4 [8] where a block with predicted

residuals is transformed using the integer operations and the output of primary transform residuals are sent for the secondary transform. The secondary transform commonly known as Low Frequency Non-Separable Transform (LFNST) uses the set of transform kernels of dimension 16×16 and 16×48, for matrix multiplication with the output residues of primary transform and gives the non-zero secondary transform residual coefficients. The output residual of secondary transform is sent for quantization. The output bitstreams are then entropy encoded and the compressed data is gained which is sent to the decoder to follow the reverse order of the modules used at the encoder to decode the input video.

A lot of new coding tools for the intra prediction and transformation has been introduced in VVC. For intra prediction, we use Intra Sub-Partitioning (ISP) [2], Matrix Intra Prediction (MIP) [3], Cross Component Linear Model (CCLM) [26], whereas the transform module has introduced the secondary transform coding tools i.e., Low Frequency Non-Separable Transform (LFNST) which is further discussed in the further sections. As, the main goal of the VVC is to attain more bitrate gain and compression efficiency than the other standard codecs, the standardization work for VVC is still on-going.

### 1.2.2 The Overall Structure of Prediction

In VVC, the decision to whether code a picture area using inter picture or intra picture prediction is made at the CU level. The prediction coding refers to predicting the similarity of the picture region in a particular picture or frame. A prediction unit (PU) partitioning structure has its root at the CU level. Depending on the basic prediction type decision, the luma and chroma CB's are further split in different block sizes and predicted from luma and chroma

prediction blocks (PBs). VVC uses two kinds of prediction i.e., inter prediction and intra prediction.

1. Intra prediction

A regular pattern in textures of natural images can be represented by geometric models. This feature is used in intra prediction [2] methods in order to remove redundancies in the spatial domain. In intra prediction, the prediction is made from the previously decoded boundary samples from the spatially neighboring TBs are used to form the prediction signal. The prediction block (PB) ranges from the dimension 4×4 to 64×64 and there are altogether 67 intra modes used for the prediction. Out of the 67 intra modes, we have 65 angular modes ranging from 2 to 66, whereas mode 0 and 1 are referred as Planar and DC mode, respectively. The DC mode prediction uses the average value of reference sample for prediction and the Planar mode prediction uses average values of two linear predictions using four corner reference samples to prevent discontinuity along the block boundaries.

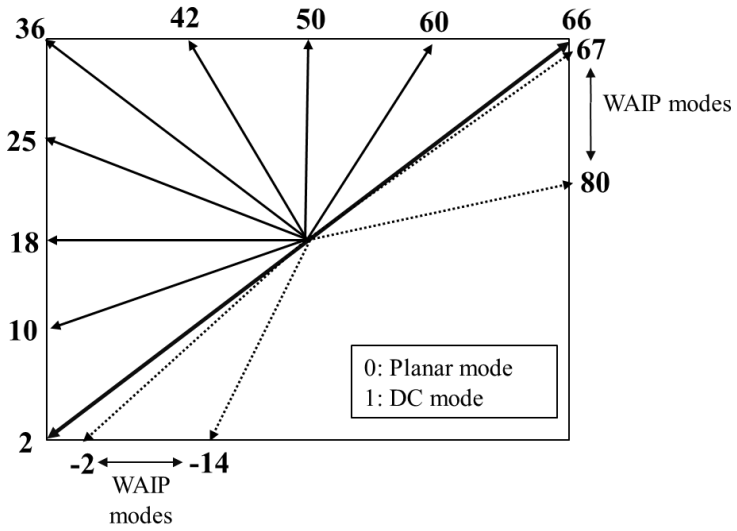


Figure 1-3. Intra prediction angular and WAIP modes in VVC.

All the angular modes reside by the left diagonal line of the square block as shown in Fig. 1-3. The concept of Wide Angle Intra Prediction (WAIP) [2] is also introduced in VVC, for the angular modes that are calculated beyond the left diagonal angles which ranges from mode -2 to -14 to the left reference boundary sample and from mode 67 to 80 to the top reference boundary sample only for the rectangle CU's. The objective of the intra prediction is to find the best model parameters that maximizes the similarity to the original block. Some of the newly added main intra coding tools are described in detail below:

❖ Intra Sub-Partition (ISP)

The newly added coding tool for intra prediction in VVC, is ISP [33], where the CU is divided into 2 or 4 partitions, either horizontally or vertically depending on the block size. It partitions the higher block sizes into smaller blocks which are easy for processing and gives efficient coding gain. The partitioning of the CU block can be inferred from the Table. 1 and visual illustration of the partitioning of $W \times H$ CU block in horizontal and vertical direction can be seen in Fig. 1-4.

Table 1. Number of sub-partitions based on block size

| **Block size** | **Number of sub-partitions** |
|---|---|
| $4 \times 4$ | Not divided |
| $4 \times 8$ and $8 \times 4$ | 2 |
| All other cases | 4 |

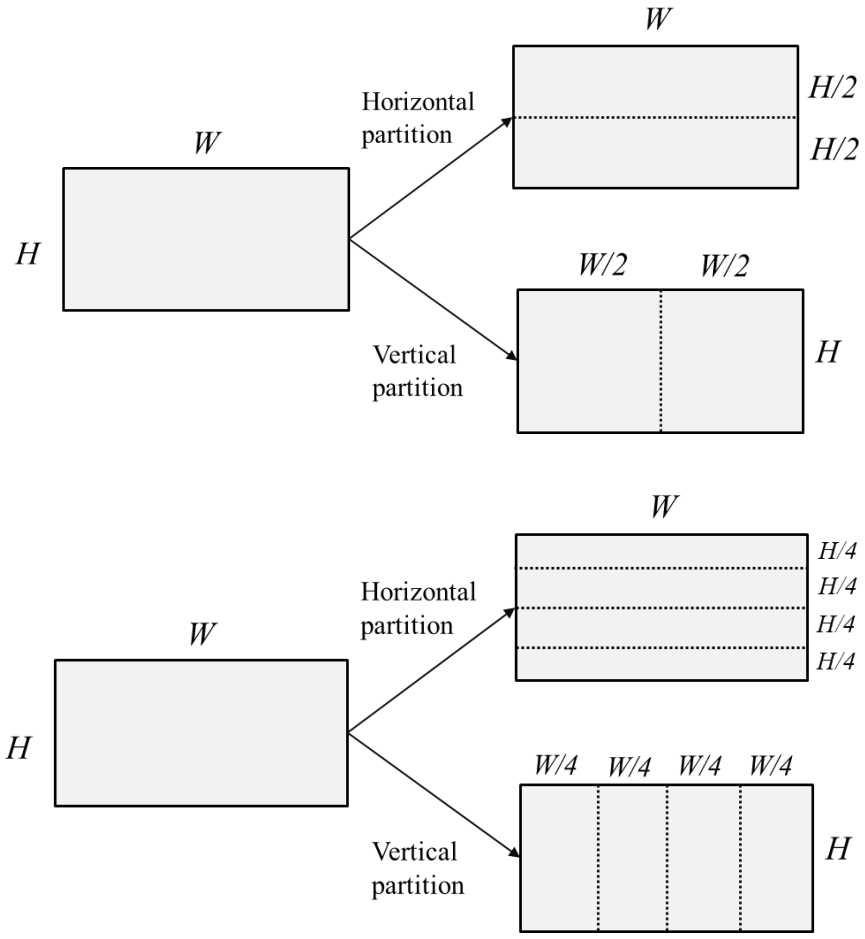


Figure 1-4. ISP horizontal and vertical (2 and 4) partitioning of $W \times H$ CU block.

❖ Matrix Intra Prediction (MIP)

MIP [34] is also the newly added intra coding tool in VVC, where the original CU block is down sampled into sub-CU ($bdry_{red}$) by averaging the top and left boundary samples, followed by matrix vector multiplication and addition of offset given in equation (1).

$$Bdry_{red} = A_k \cdot bdry_{red} + b_k \,, \tag{1}$$

The obtained result $Bdry_{red}$ i.e., the result after the matrix vector multiplication, is up sampled by linear interpolation first in horizontal direction followed by vertical direction. $A_k$ represents the MIP matrix and $b_k$ represents the offset vector used based on intra mode 'k'. MIP has three set of matrices and offset vectors for the matrix vector multiplication which is obtained based on the intra mode of the CU. The set $S_0$ has 18 matrices of dimension 16×4 and 18 offset vector of size 16 and is used for CU blocks of dimension 4×4. The set $S_1$ has 10 matrices of dimension 16×8 and 10 offset vector of size 16 and is used for CU blocks of dimension 8×4 and 4×8. The last set $S_2$ has 6 matrices of dimension 64×8 and 6 offset vector of size 64 used for all other block's sizes greater than 4×8 and 8×4 CU block.

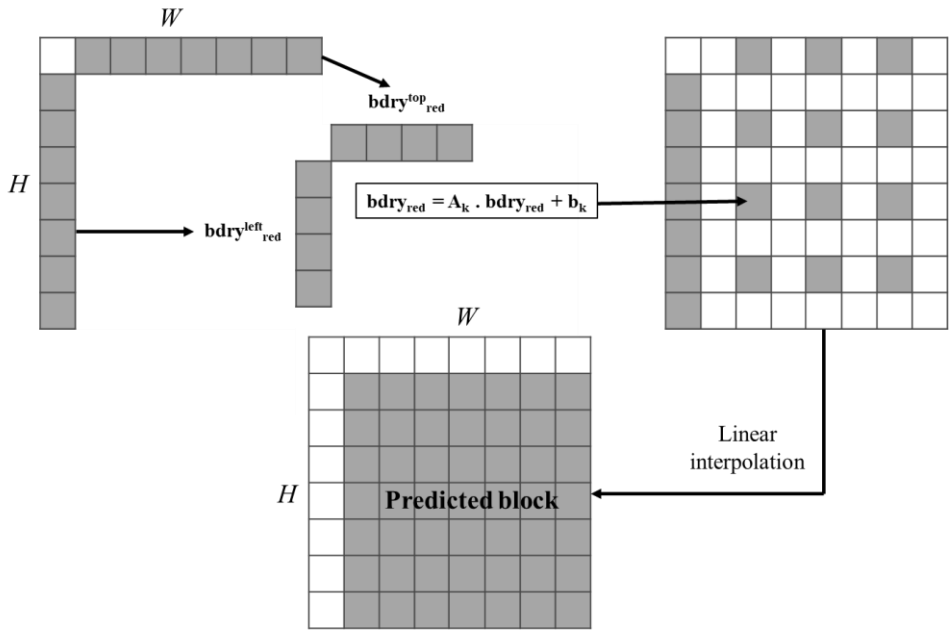


Figure 1-5. Design of MIP in VVC for $W \times H$ CU block.

❖ Cross Component Linear Model (CCLM)

Another important intra coding tool to calculate the prediction mode of chroma block is given by Cross Component Linear Model (CCLM) [26]. CCLM predicts the chroma sample values using the co-located reconstructed luma samples with a local linear regression model. The CCLM includes CCLM_L, CCLM_T and CCLM_TL mode, where the left, top and top-left samples are used to determine the chroma prediction mode using the linear model based on the linear model parameters $\alpha$ and $\beta$. For example, in the CCLM_T mode, only the top template or the top reference sample is used to calculate the linear model parameters. To get more reference samples, the number of reference sample is taken twice the original size (2W) as shown in Fig. 1-6.

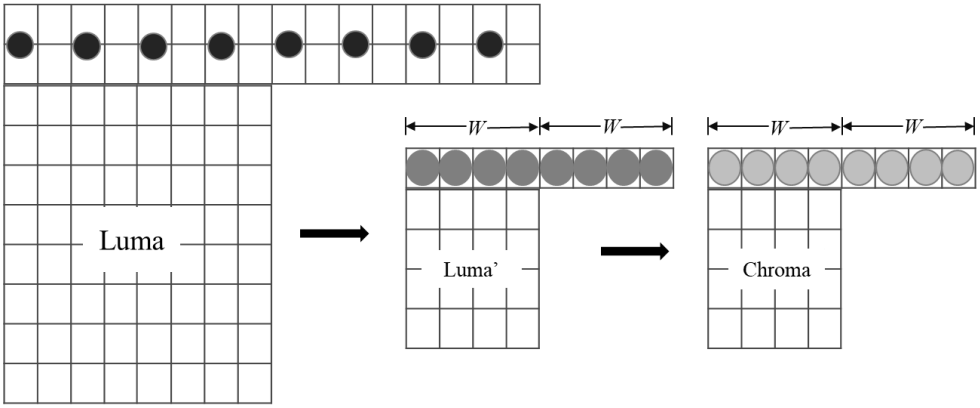


Figure 1-6. Illustration of CCLM_T mode in VVC.

In the current VVC, a simplified line fitting method is being used for the CCLM. Initially, the luma sample is down sampled to corresponding luma and chroma sample depending upon either of the CCLM modes as shown in Fig. 1-6. CCLM uses a six-tap down sampling filter while down sampling the original luma sample. This requires two rows of neighbouring luma sample above the current block and three columns of neighbouring luma sample left

of the current block. Then the model parameters are calculated as shown in equation (2)-(3) based on the co-ordinates representing the maximum and minimum luma values in Fig. 1-7.

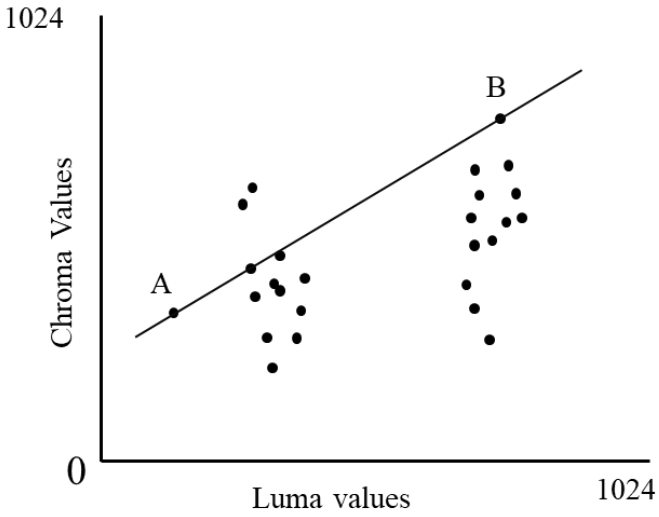


Figure 1-7. Illustration of straight line between minimum and maximum luma value.

The linear model parameters are calculated as -

$$\alpha = \frac{y_B - y_A}{x_B - x_A} \tag{2}$$

$$\beta = y_A - \alpha x_A \,, \tag{3}$$

2. Inter prediction

The inter prediction [27] performs the prediction to remove the dependencies in the temporal domain i.e., the prediction is made based on the previously coded picture not same picture frame. The objects and regions of the picture taken from the video source approximately contains similar pixels, though slightly displaces. The core idea of the inter prediction is to model the

displacement of these regions/objects restricted to rectangular blocks. A technique known as Motion Estimation (ME) [27] is used for the inter prediction which takes two inputs as the original block to model and a reference frame shifted in time. Motion estimation finds a motion vector (MV) [1] among a set of candidates by maximizing the similarity between the original block and the displaces block taken in reference.

### 1.2.3 The Overall Structure of the Transform

VVC uses the transform coding of the prediction error residual in a similar manner as in prior codec standards. The residual block is partitioned into multiple square or rectangle transform blocks (TBs) and is further computed by the core transform i.e., primary transform and further computed by secondary transform i.e., LFNST and the output bitstreams are sent for quantization.

The primary two-dimensional transforms are computed by applying one dimensional transform in the horizontal and vertical directions. The elements of the core transform matrices are derived by approximating the scaled Discrete Cosine Transform (DCT) [7] and Discrete Sine Transform (DST) [8] basis functions, considering the necessary dynamic range for transform computation, maximizing the precision and maintaining the orthogonality of the matrices which are specified as integer values [1]. For primary transform, basic transform kernels such as DCT-2, DST-7, DCT-8, DST-4, and DCT-4 are used and are referred as Multiple Transform Set (MTS). These different kernels are useful because of their different energy distribution properties, although memory storage to store different block size kernel is always one of

the major concerns. In VVC, the use of MTS transform kernel based on prediction and block size is shown in Table. 2.

Table 2. MTS transform kernel based on the prediction and block size.

| Prediction | Prediction tools | Block size | MTS transform kernel |
|---|---|---|---|
| Intra | ISP (Intra Sub-Partition) | 4×4 up to 16×16 | DCT-2, DST-7 |
| | MIP (Matrix Intra Prediction) | 4×4 up to 64×64 | DCT-2, DST-7, DCT-8 |
| | Normal Intra | 4×4 up to 64×64 | DST-7, DCT-8 |
| Inter | SBT (Sub-Block Transform) | 4×4 up to 64×64 | DCT-2, DST-7, DCT-8 (depends on SBT position) |

In general, a DCT and DST is a Fourier related transform similar to the 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 no difference in the residual 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 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 as they show the similar signal behavior.

The secondary transform is a higher order transform and is applied between forward primary transform and quantization (at encoder) and between de-quantization and inverse primary transform (at decoder side). The secondary transform i.e., LFNST has its own set of transform kernels which is multiplied with the output residual coefficients of the primary transform. There are two sets of secondary transform kernel of dimension 16×16 and 16×48 which takes 8KB of memory storage. The selection of the transform kernel is dependent on the intra prediction mode of TB as shown in Table. 3. As shown in Fig. 1-8, 4×4 (or 8×8) secondary transform is performed depends on block size. For example, 4×4 secondary transform is applied for small blocks (i.e., min (width, height) < 8) and 8×8 secondary transform is applied for larger blocks (i.e., min (width, height) > 4) per 8×8 block.

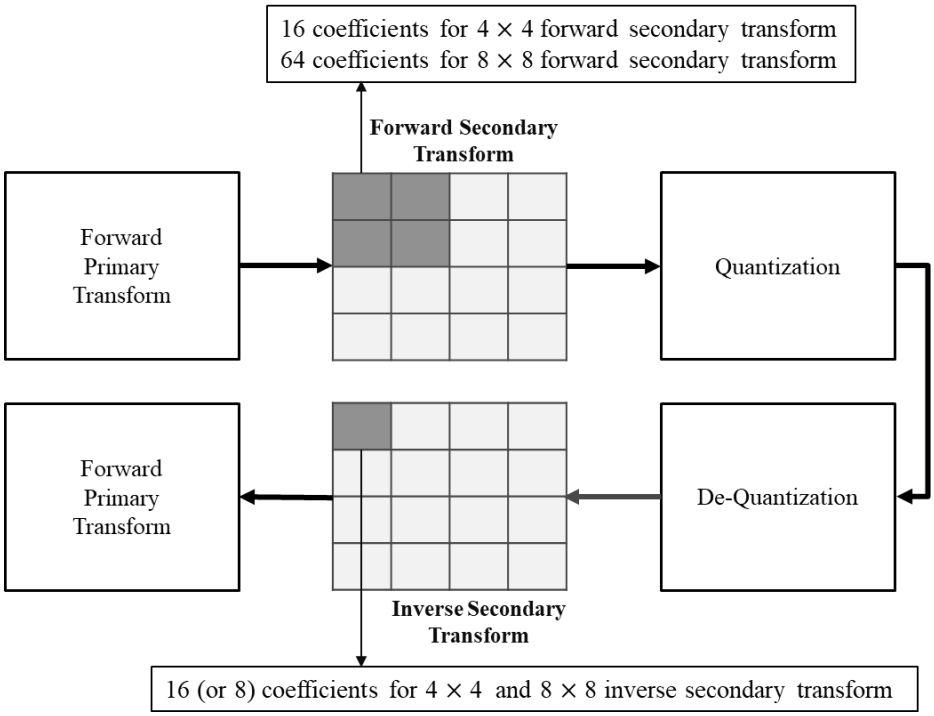


Figure 1-8. The secondary transform working design model in VVC.

The major proposed concept of this thesis is to focus on the memory storage along with the implementation of the efficient design of the secondary transform with reduced complexity. In this thesis, the proposed method reduces the computation complexity along with saving of memory by 2KB, as the LFNST kernels require 6KB of memory storage instead of 8KB.

## 1.3 Objective

The secondary transform is regarded as the prime component during the process of compression and decompression in video coding; the goals and ideas for this research set are as following:

- Derive the secondary transform kernels indexed, based on the intra prediction mode.

- Reduce the transform kernel storage from 8KB to 6KB.
- Exhibit the characteristics of different transform block computation with respect to the secondary transform kernels based on the intra prediction modes
- Use only one secondary transform kernel of size 16×48 for transform blocks of order 4×*N* and *N*×4 where *N*>=4 to show the efficient reduced complexity of operations.

## 1.4 Motivation

In the standardization of VVC, the non-separable secondary transform is widely used as compared to the separable discrete cosine transform (DCT) and the separable discrete sine transform (DST) for the video compression. Although DCT is an efficient approximation of optimal separable primary transform, it is not flexible for the highly dynamic images with textures and edges. The secondary transform is only applied to the intra-coded blocks with intra and inter slices. Recently, in the undergoing of the standardization of VVC, two secondary transform kernels were used for the VVC Test Model (VTM) i.e. 16×16 kernels for 4×4, 4×*N*, *N*×4 (*N*>4) TU and 16×48 kernel for 8×8, *N*×8, 8×*N* (*N*>8) TU. Apparently, the selection of this transform kernel is based on the indexing of the transform set with respect to the intra prediction mode of the transform block, as shown in Table. 3. Also, for the 4×4 transform blocks, the sub-sampled kernel is derived from the 16×16 kernel to get the maximum of 8 coefficients in the residue. The prime consequence of the existing design is the memory storage of the secondary transform kernels, which takes 8KB of memory to store all kernels along with the complexity and latency reduction issues.

Several proposals were presented to solve the memory and complexity reduction issue. These proposals tried to use a single kernel but could not reduce the complexity with the mathematical computation of the LFNST process. Some proposals tried to only reduce the complexity giving a limiting number of output residue coefficients in the bitstream for quantization, still keeping the memory storage of the kernel same as 8KB for either 4×4 LFNST or 8×8 LFNST design.

Hence, to overcome the issues, a simpler LFNST design with a single secondary transform kernel of 16×48 is used, and sub-sampled kernels are used for the smaller transform blocks i.e., 4×4, 4×*N*, *N*×4 (*N*>4) applied with 4×4 LFNST. For 8×8 LFNST for the higher transform block size i.e., 8×8, *N*×8, 8×*N* (*N*>8), the kernel 16×48 remains the same. Also, the proposed method reduces the output residue coefficients to a maximum of 16 non-zero coefficients, zeroing all the primary transform coefficients in the bitstream, which speeds up the encoder reducing the complexity and latency issue in the VTM. Since the secondary transform is applied post-primary transform on the transform blocks, it totally supports the fast algorithm implementation of DCT-2 and DST-7 transform kernels of the primary transform [25].

## 1.5 Thesis Layout

The rest of the thesis is organized as follows: Section 2 exhibits the related work. Section 3 describes the proposed method i.e., the usage of the single secondary transform kernel of size 16×48 to calculate the residual of the transform blocks of size 4×*N* and *N*×4 where *N*>=4 computed after undergoing primary transform along with zeroing concept. Section 4 provides all the experimental results, followed by the conclusion of this thesis in Section 5.

# 2. Related Works

## 2.1 Introduction to LFNST

In the different trends of standardization, the non-separable secondary transform has been evolved efficiently along with the primary transform. Although DCT is an efficient approximation of optimal separable transform, it's adaptivity with the fixed core is limited for the highly dynamic images with textures and arbitrarily detected edges. Hence non-separable transform can achieve better compression efficiency over the video/image with directional texture patterns.

The Low Frequency Non-Separable Transform (LFNST) is currently used in the VVC model, which has two sets of transform kernel matrix of 16×16 and 16×48 dimensions. Memory storage has become one of the major problems for the codecs recently while applying the fast algorithm along with the less memory storage to transform kernels has become a challenging task to be performed.

The application of a non-separable transform i.e., LFNST is described as follows using input as an example. To apply the non-separable transform, for the 4×4 input block *X* which is represented below in equation (4).

$$\mathbf{X} = \begin{bmatrix} x_{00} & x_{01} & x_{02} & x_{03} \\ x_{10} & x_{11} & x_{12} & x_{13} \\ x_{20} & x_{21} & x_{22} & x_{23} \\ x_{30} & x_{31} & x_{32} & x_{33} \end{bmatrix} \tag{4}$$

*X* is first represented as a one-dimensional vector **x** as (5)

$$\mathbf{X} = [x_{00} \quad x_{01} \quad x_{02} \quad x_{03} \quad x_{10} \quad x_{11} \quad x_{12} \quad x_{13} \quad x_{20} \quad x_{21} \quad x_{22} \quad x_{23} \quad x_{30} \quad x_{31} \quad x_{32} \quad x_{33}]^T \tag{5}$$

The non-separable secondary transform is calculated by (6)

$$\mathbf{F} = \mathbf{T}\mathbf{x}\,, \tag{6}$$

where **F** indicates the transform coefficient vector, and **T** is a 16×16 transform matrix. The 16×1 coefficient vector **F** is subsequently re-organized as 4×4 block using the scanning order for that block (horizontal, vertical, or diagonal). The coefficients with a smaller index will be placed with the smaller scanning index in the 4×4 coefficient block. The mapping from the intra prediction mode to the transform set is pre-defined. For each transform set, the selected non-separable secondary transform candidate is further specified by the explicitly signaled secondary transform index. The index is signaled in a bitstream once per intra coding unit after transform coefficients. The visual illustration of the forward and inverse secondary transform is given in Fig. 2-1.

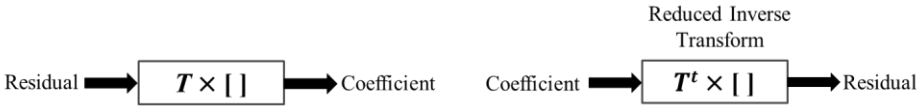


Figure 2-1. Block diagram of forward and inverse secondary transform in VVC.

An inverse secondary transform i.e., LFNST which was earlier known as Reduced Secondary Transform (RST) [35] is conditionally applied when the following two conditions are satisfied:

1. Block size is greater than or equal to the given threshold ($W$>=4 && $H$>=4)
2. The transform skip mode flag is equal to 0.

Furthermore, LFNST is applied for intra-coded CU in both intra and inter slices, and for both luma and chroma. If a dual-tree is enabled, LFNST indices for luma and chroma are signaled separately. For inter slice (the dual-tree is

disabled), a single LFNST index is signaled and used for both luma and chroma. The LFNST is also applied to the TU's which is a Matrix Intra Prediction (MIP) coded block along with the normal intra predicted block with general intra prediction and wide-angle modes. Since the secondary transform kernel is selected from the transform kernel set, determined by the intra prediction modes, the mapping of the intra prediction modes and transform set index is shown below in Table. 3.

Table 3. Mapping of the LFNST kernel set index with the intra prediction mode

| **Intra prediction mode** | **LFNST kernel set index** |
|---|---|
| Intra_pred_mode < 0 | 1 |
| 0 <= Intra_pred_mode <= 1 | 0 |
| 2 <= Intra_pred_mode <= 12 | 1 |
| 13 <= Intra_pred_mode <= 23 | 2 |
| 24 <= Intra_pred_mode <= 44 | 3 |
| 45 <= Intra_pred_mode <= 55 | 2 |
| 56 <= Intra_pred_mode <= 80 | 1 |
| 81 <= Intra_pred_mode <= 83 | 0 |

The mode 81 to 83 determines the CCLM [26] modes. Whenever if any of the three CCLM modes is indicated, the index is set to 0, the same as for the Planar and DC mode.

In the stage of standardization, there are few related works for the reduction of the complexity for LFNST as standard contributions by some participants of the standardization which is mentioned in the next section.

## 2.2 Existing Methods

### 2.2.1 A Memory Reduction Approach for LFNST

This contribution [22] proposes to unify the LFNST calculation and attempts to reduce the memory cost by using a single 16×48 kernel for LFNST. For the transform blocks of size 4×*N* and *N*×4, they considered as part of 8×*N*, and *N*×8 blocks with only the overlapping part with 16×48 kernel basis is used. Currently, in the VTM, two different kernels are used for LFNST i.e., 16×16 and 16×48 kernels, but this proposal proposes to use the later kernel for all block sizes. The proposed method provides the following benefits: saving the memory storage for the kernels, i.e., 2KB, simplifying the computation of LFNST, and the spec text is much simplified. This proposed method gives 0.05%, 0.01% luma BD-rate changes for All Intra (AI) and Random Access (RA) configurations, respectively, with 3% overall encoding time saving for AI, with simulations over VTM-5.0 under CTC conditions.

Eventually, for forward LFNST when we apply 16×48 kernel for 4×*N* and *N*×4 block, only the overlapping part with basis is used further for the secondary transform. The illustration for applying 16×48 LFNST kernel for 4×4, 4×8, 8×4 and 16×4 transform blocks (TB) are shown in the shaded region below in Fig. 2-2.

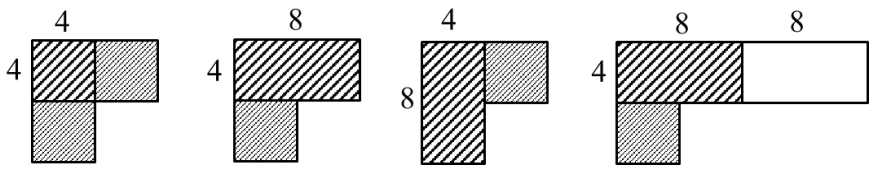


Figure 2-2. Using the single 16×48 LFNST kernel for 4×4, 8×4, 4×8 and 16×4 transform block of lower dimensions (left to right).

Consecutively, for the inverse LFNST, either 8 or 16 coefficients are given as input, and 48 coefficients samples are gained at the output, which forms the L-shaped as mentioned above in Fig. 2-2. The coefficients present only inside the block region need to be calculated, and the remaining samples are not calculated. For 4×4 transform unit, only 8 coefficients from the top left 4×4 region is calculated along with 16×48 kernel as output. Also, to maintain the worst-case multiplication per coefficient intact, they calculate only 8 coefficients for 4×8 and 8×4 transform units

### 2.2.2 A Limiting Coefficients Approach for LFNST

This contribution [21] proposed several variations to reduce the complexity of the LFNST process and the simplification. The contribution was presented with four different aspects where each aspect with their individual simulation results and detailed analysis is also presented. The aspects were:

a) A maximum of 16 LFNST coefficients as the output residue is targeted to obtain, whereas before 32 LFNST coefficients were obtained from the two top left adjacent 4×4 blocks in 4×*N* or *N*×4 TU's (*N*>4).
b) Additional zeroing of the primary transform coefficients to be done when LFNST is applied to 4×4 TU block.

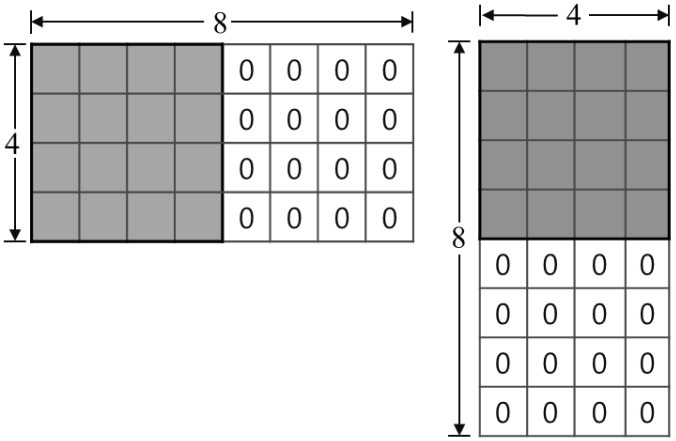


Figure 2-3. The additional zero-out of primary transform coefficients when 4×4 LFNST is applied.

c) Additional zeroing of the primary transform coefficients to be done when LFNST is applied to 8×8 TU block.

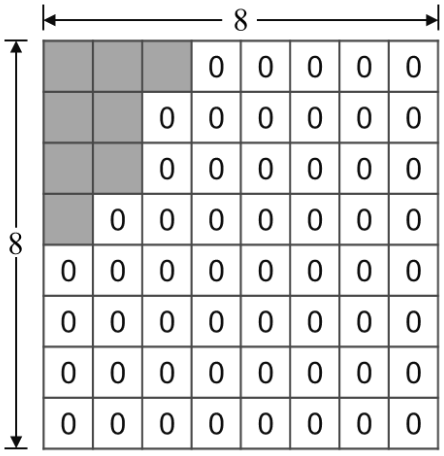


Figure 2-4. Additional zeroing out of the primary transform coefficients for the when 8×8 LFNST is applied for 8×8 TU block.

d) Changing of the region where non-zero residual transform coefficients are taken in consideration.

All the above-mentioned aspects were simulated all together in VTM-5.0 under the CTC condition with 0.6% BD-rate over All Intra (AI) with 90% EncT, 0.02% BD-rate over Random Access (RA) with 97% EncT. It is seen that zeroing the 8×8 LFNST has more impact on the complexity reduction rather than the 4×4 LFNST zeroing. The comparison in the Section 4.2 gives more insight about the results of this contribution. This proposal also reduced the processing latency in the two sets of transform kernels, which helps for the real-time encoder optimization. Although it was still questionable if the normative changes made over VTM was beneficial for further standardization.

### 2.2.3 A Limiting Coefficients and Zeroing Approach for LFNST

In this proposal [20], they propose to restrict all the coefficients outside the first sub-group coefficients to be non-significant. Since the secondary transform is applied to the top-left subset of the residual of the primary transform, in this proposal the remaining non-significant primary transform coefficients are made to be zero when LFNST is applied. The proposal allows condition on the signaling of the LFNST index on the last significant position, which helps to avoid the extra scanning of the coefficients present in the current design, which was needed just to check for the significant coefficients at a particular specific position.

Table 4. The total number of LFNST kernel used with respect to the transform unit (TU) size along with memory requirement.

| **Transform blocks** | **LFNST Kernels** |
|---|---|
| 4×4 | 16×8 |
| 4×*N*, *N*×4; *N*>8 | 16×16 |
| 8×8 | 48×8 |
| *M*×*N*; (*M*, *N*)>8 | 48×16 |
| LFNST candidates | 2 |
| Memory requirement | 8KB |

The proposal maintains the handling of the worst-case scenario (multiplication per pixel) by restricting the secondary transform for 4×4 and 8×8 transform blocks to 8×16 and 8×48 transform kernels. Total multiplications per coefficient for transform units of $M \times N$ with zeroing is given as:

$$Muls\_per\_coeff_{directMatrixMultiply} = \frac{m}{M}n + m \quad (7)$$

$$Muls\_per\_coeff_{halfButterFly} = \frac{m}{M}\left(\frac{n}{2} + \log_2 \frac{n}{2}\right) + \left(\frac{m}{2} + \log_2 \frac{m}{2}\right), \quad (8)$$

where $Muls\_per\_coeff_{directMatrixMultiply}$ indicates the direct matrix multiplication of the entire TU with the entire LFNST kernel, whereas $Muls\_per\_coeff_{halfButterFly}$ indicates the multiplication of the LFNST kernels only to the significant coefficients required by secondary transform zeroing out the primary transform coefficient following the half-butterfly scanning order.

For the mentioned block size case, last significant position of the scanning must be less than 8, while for other blocks it must be less than 16 when LFNST

is applied. For 4×*N* and *N*×4 TU with *N*>8, the proposal restriction exhibits the applicability of LFNST only once to the top left 4×4 region as shown in Fig. 2-5.

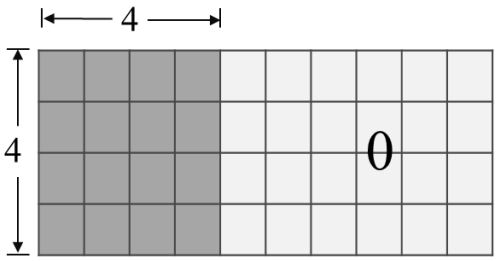


Figure 2-5. Applicability of LFNST to the first subblock for 4×*N* and *N*×4 TU block with *N*>8 having a maximum 16 coefficients.

As seen from the figure mentioned above, all the primary transform coefficients present on the right side of the transform block lightly shaded is deliberately zeroed out. With this approach, the quantization of the coefficients is significantly simplified when LFNST is tested over VTM reference software. Also, the rate-distortion optimized quantization is carried out to be maximum for the first 16 coefficients, following the scanning order, and remaining coefficients are made to be zero.

Based on the test results, there is a gain for the luma chrominance (U) with 0.02% with 10% of the reduced encoding time (EncT) for All Intra (AI). Similarly, there is 0.01% gain for the chroma component (V) with 3% reduction in the encoding time (EncT) for Random Access (RA) over VTM-5.0.

## 2.3 Problems of the Related Works

Several proposals were presented to solve the memory and complexity reduction issue. These proposals tried to use a single kernel but could not reduce the complexity with the mathematical computation of the LFNST process. Some proposals tried to only reduce the complexity giving a limiting number of output residue coefficients in the bitstream for quantization, still keeping the memory storage of the kernel same as 8KB for either 4×4 LFNST or 8×8 LFNST design. Since the secondary transform is applied post-primary transform on the transform blocks, the related proposals totally supported the fast algorithm implementation of DCT-2 and DST-7 transform kernels of the primary transform [25], but not for the efficient complexity reduction. Further, in thesis an efficient LFNST design is presented in section 3, which targets for better coding complexity reduction of the secondary transform.

All the related works presented above, either dealt with the memory reduction saving 2KB of memory or limiting the non-zero coefficients of the output residue of the secondary transform i.e., 4×4 LFNST or 8×8 LFNST coefficients. The approaches mentioned does not have a combined ground for the complexity reduction including both the memory reduction and complexity reduction of the LFNST process. Since, optimization of the codec is preferred for any coding tools, if a faster and efficient computational LFNST design is constructed, the encoder would have more significant impact on the compression of the video source.

# 3. Proposed Method

In this thesis, the complexity reduction of the secondary transform, commonly known as LFNST for the VVC is proposed. The secondary transform LFNST kernels used in this research are derived from the non-CTC images (DIV2K Dataset). Earlier, in the VTM-5.0 the reference software of the VVC, two sets of LFNST kernels i.e., 16×16 and 16×48 matrices were used for different transform units as shown in Table. 5 below, but this research proposes to use only a single 16×48 transform kernel for all TU's as shown in Table. 6, which also saves the 2KB of the storage memory of the kernel matrices.

Table 5. LFNST kernels used in VTM-5.0

| **LFNST kernel size** | **Applicable TU size** | **Memory cost** |
|---|---|---|
| 16×16 | 4×*N* and *N*×4, *N*>=4 | 2KB |
| 16×48 | 8×*N* and *N*×8, *N*>=8 | 6KB |

Table 6. Proposed LFNST kernel for VTM-5.0

| **LFNST kernel size** | **Applicable TU size** | **Memory cost** |
|---|---|---|
| 16×48 | 4×*N* and *N*×4, *N*>=4 | 6KB |

The LFNST transform kernel, is selected from the set of kernels based on the intra prediction mode of the CU of the block predicted during the intra-prediction, a step before the transformations of the TU's. The 16×48 LFNST kernel has different sub-sets of kernels of same size which is selected by indexing the intra prediction modes. The indexing of the intra prediction

modes with the LFNST kernel sets is kept unchanged and is shown in the Table. 7 below:

Table 7. Specification of the LFNST kernel set index with the intra prediction mode of the TU's.

| **Intra prediction mode** | **LFNST kernel set index** |
|---|---|
| Intra_pred_mode < 0 | 1 |
| 0 <= Intra_pred_mode <= 1 | 0 |
| 2 <= Intra_pred_mode <= 12 | 1 |
| 13 <= Intra_pred_mode <= 23 | 2 |
| 24 <= Intra_pred_mode <= 44 | 3 |
| 45 <= Intra_pred_mode <= 55 | 2 |
| 56 <= Intra_pred_mode <= 80 | 1 |
| 81 <= Intra_pred_mode <= 83 | 0 |

The 'Intra_pred_mode' mentioned in the Table. 7 indicates the intra prediction mode of the coding units, which includes the following specified modes:

- Mode 0: Planar mode
- Mode 1: DC mode
- Mode 2-66: Directional modes
- Mode 67-80: Wide Angle Intra Prediction (WAIP) [54] modes
- Mode 81-83: Cross-Component Linear Model (CCLM) [51] modes

## 3.1 The Key Feature of the Proposed Method

The representation of the proposed method in this research can be looked more vividly with the overall block diagram of the operation of the secondary transform i.e., LFNST in the VVC as:

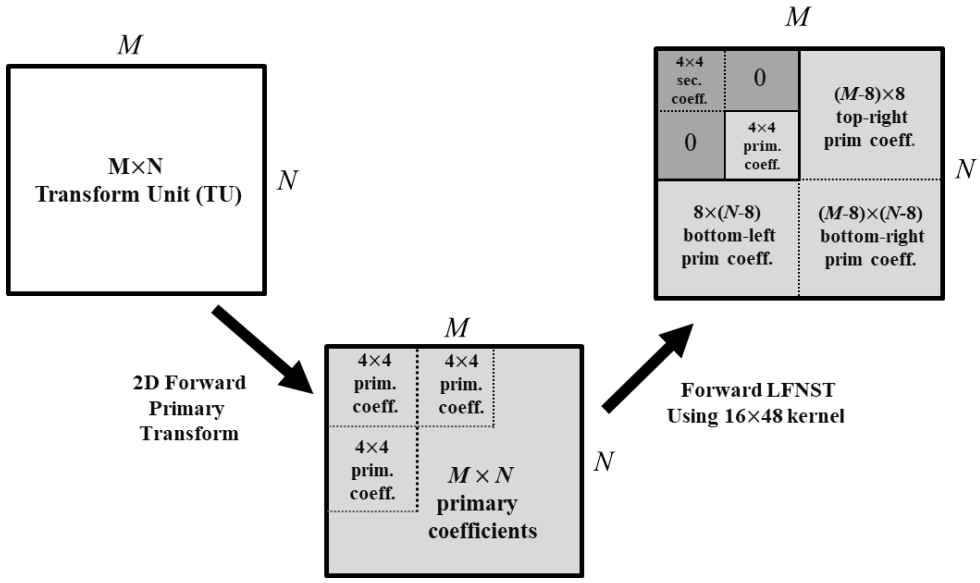


Figure 3-1. Pictorial representation of the block diagram of the LFNST computation in VVC.

The Fig. 3-1 describes the computation of the proposed LFNST in the codec, where the TU of dimension $M \times N$, post prediction stage is passed through the two dimensional forward primary transformation part, and the residue of the primary transform is passed through for the computation of the secondary transform using the single 16×48 matrix kernel irrespective of the block size i.e., 4×$N$ to $N$×4 for $N$>=4. The computation of the secondary transform for a TU block resulting in the 16 secondary transform coefficients located at the top left part of the residual TU block is computed as:

- For 16×16 TU block,

$$[16 \times 48 \;\; LFNST\ kernel] \times [48 \times 1 \;\; prim\ coeff] = [16 \times 1 \;\; \text{sec.}\, coeff] \qquad (9)$$

The TU block size (16×16) is divided into three 4×4 block size at the top-left position. Later, three 4×4 (one dimensional 48×1 matrix) block is

multiplied by the 16×48 secondary transform kernel and the resulting 16×1 secondary transformed coefficients can be obtained which is placed at the top left portion of the resulting TU and remaining adjacent 4×4 TU blocks with 16 coefficients each at the right and bottom of the top left block is made to be zeroed. The remaining ($M$-8) × 8 top right TU block, ($M$-8) × ($N$-8) bottom right TU block and 8 × ($N$-8) bottom left TU block remains unchanged having the primary transform coefficients.

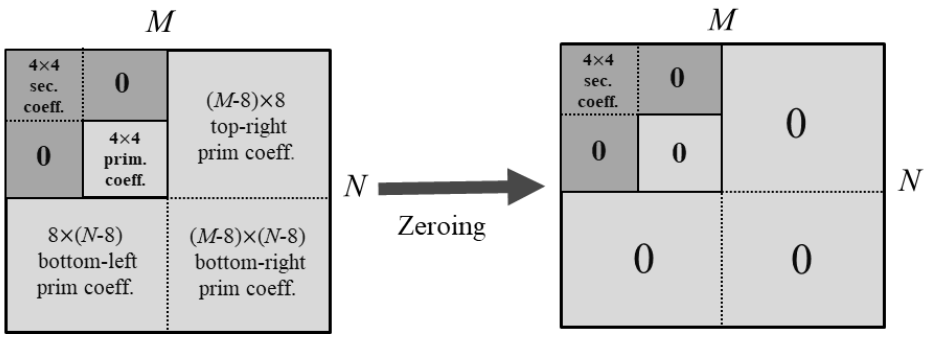


Figure 3-2. Pictorial representation of the additional zeroing of the primary transform coefficients.

In this research, we additionally zero out the remaining TU's having the primary transform coefficients so that only the top left secondary transform residue is taken into account for the further quantization process. Since the coefficients located in the top left part of the TU is more homogeneous, it more efficient to take those coefficients into account for the further coding process.

The secondary forward transform i.e., LFNST is applied to only the intra coded transform block (TB), which has both inter slice and intra slice coded in intra prediction. For the encoder, the primary forward transform is performed first, then followed by the secondary transform followed by quantization and CABAC bit coding. Whereas, for the decoder, CABAC bit decoding and

inverse quantization is first followed by a secondary inverse transform then primary inverse transform at last. The design specification for the encoder side for different sets of TU blocks is shown below:

1. Case 1: TU of 4×4 block

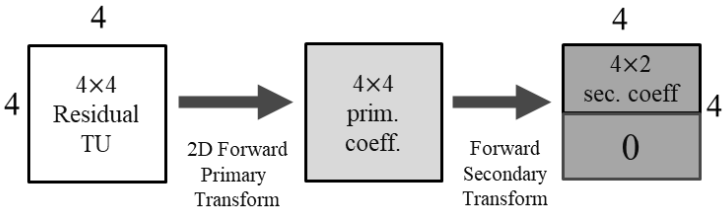


Figure 3-3. LFNST design for 4×4 TU block with max 8 non-zero bitstream coefficients.

Fig. 3-3 illustrates the proposed secondary transform for 4×4 TU block. Firstly, the TU is computed with forward primary transform, resulting in the primary transform residue, then it is multiplied with the forward secondary transform i.e., LFNST derived 16×16 kernel from 16×48 original kernel as explained in section 3.2. The resulting 16×1 matrix in equation (10), gives the secondary transform residue, and the right half of the resulting coefficients are zeroed out to give 8×1 coefficients. Thus, maximum 8 non-zero bitstream coefficients placed in the upper half of the TU.

$$[16 \times 16 \ \ LFNST\ kernel] \times [16 \times 1 \ \ prim\ coeff] = [16 \times 1 \ \ \sec coeff] \qquad (10)$$

2. Case 2: TU of 4×8 and 8×4 block

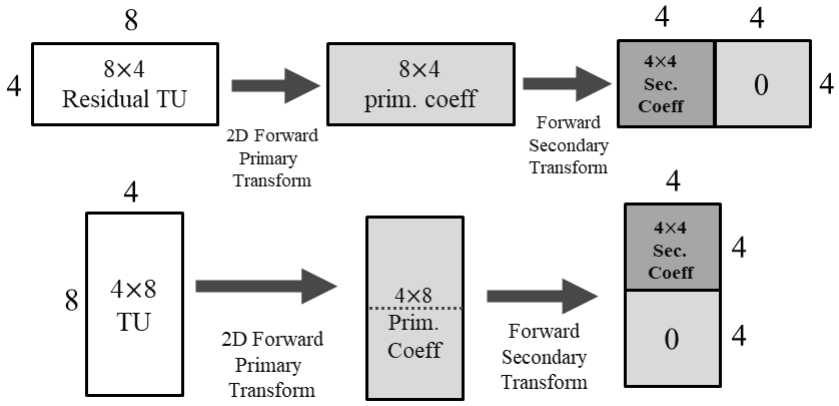


Figure 3-4. LFNST design for 4×8 and 8×4 TU block with max 16 secondary transform bitstream coefficients.

Fig. 3-4 illustrates the proposed secondary transform for 8×4 and 4×8 TU blocks. The TU block size 8×4 (is divided into two 4×4 block size. For first 4×4 (one dimensional 16×1 matrix) block is multiplied by the 16×16 secondary transform kernel derived from the 16×48 kernel as shown in Section 3.2 and the resulting 16×1 secondary transformed coefficients can be obtained as shown in equation (11). The obtained coefficients are placed in the right part of the 4×4 matrix, and the second pair 4×4 block with primary transform coefficients is zeroed out. Similar process is done for the first 4×8 transform block, both resulting in maximum 16 non-zero coefficients in the bitstream.

$$[16 \times 16 \;\; LFNST\ kernel] \times [16 \times 1 \;\; prim\ coeff] = [16 \times 1 \;\; \mathrm{sec}\ coeff] \qquad (11)$$

3. Case 3: TU of 4×*N* and *N*×4 (with *N*>=16) block

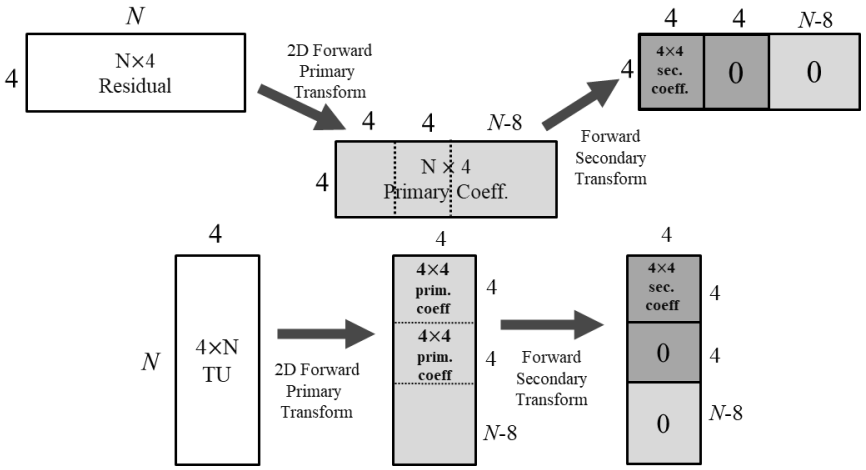


Figure 3-5. LFNST design for 4×*N* and *N*×4 TU block with max 16 secondary transform bitstream coefficients.

Fig. 3-5 illustrates the proposed secondary transform for TU blocks of *N*×4 and 4×*N*. For example, we take TU block size (16×4), which is divided into two 4×4 block size and remaining 8×4 block size. For first 4×4 (one dimensional 16×1 matrix) block is multiplied by the 16×16 secondary transform kernel derived from the 16×48 kernel as mentioned in Section 3.2 and the resulting 16×1 secondary transformed coefficients can be obtained as shown in equation (12) - (13). The obtained coefficients are placed in the leftmost part of the 4×4 matrix. A similar computation is done for the second first 4×4 residual block size, and after obtaining the secondary transform residual coefficients, we zero out the second 4×4 block along with the remaining primary transform coefficient in the rightmost 8×4 TU block. For the 16×4 TU residual block, a similar process is done for both first and second 4×4 residual block size along with zeroing of the primary transform coefficients. This design results in a maximum of 16 non-zero coefficients in the bitstream.

$$[16 \times 16\ \ LFNST\ kernel] \times [16 \times 1\ \ prim\ coeff1] = [16 \times 1\ \ \mathrm{sec}\ coeff1] \quad (12)$$

$$[16 \times 16\ \ LFNST\ kernel] \times [16 \times 1\ \ prim\ coeff2] = [16 \times 1\ \ \mathrm{sec}\ coeff2] \quad (13)$$

4. Case 4: TU of 8×8 block

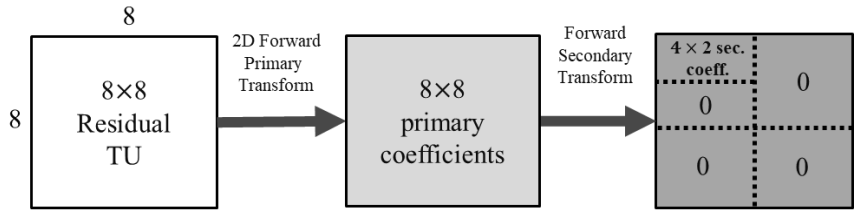


Figure 3-6. LFNST design for 8×8 TU block with max 8 secondary transform bitstream coefficients

Fig. 3-6 illustrates the proposed secondary transform for 8×8 TU blocks. The given TU block is divided into four 4×4 blocks. The top left, top right and bottom left 4×4 block (one dimensional 48×1 matrix) is multiplied by the 8×48 secondary transform kernel and the resulting 8×1 secondary transformed coefficients can be obtained as shown in equation (14). The obtained coefficients are placed in the top left, top right, and bottom left part of the TU block. The 8×8 TU block is the exceptional case where only eight non-zero coefficients are taken in bitstreams, and all remaining secondary and primary transform coefficients are zeroed out.

$$[16 \times 48\ \ LFNST\ kernel] \times [48 \times 1\ \ prim\ coeff] = [16 \times 1\ \ \mathrm{sec}\ coeff] \quad (14)$$

5. Case 5: TU= 8×16, 16×8 and $N \times N$ (with $N$>=16) blocks

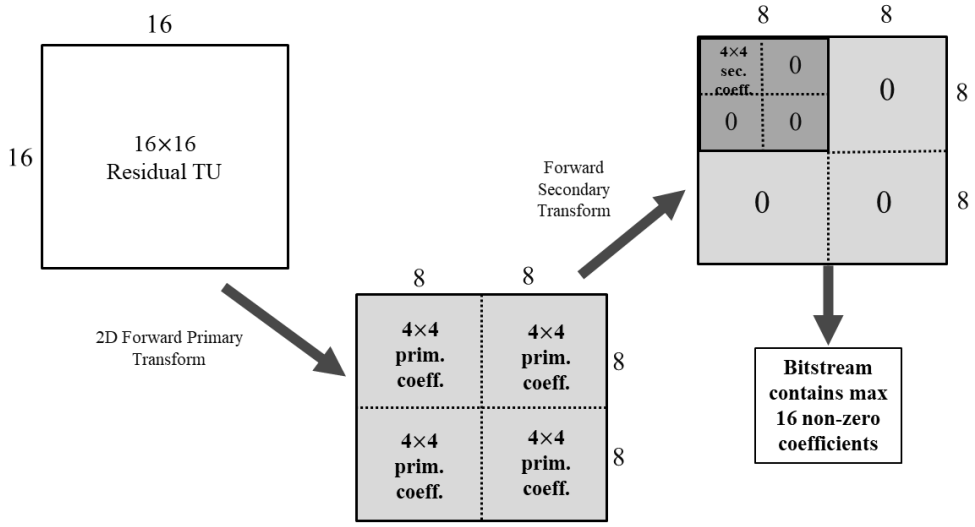


Figure 3-7. LFNST design for 16×16 TU block with max 16 secondary transform bitstream non-zero coefficients.

Fig. 3-7 illustrates the proposed secondary transform for 16×16 TU block. The given TU block is divided into four 8×8 blocks after the primary transform is applied. Further, the top left 8×8 block is further divided into four 4×4 blocks. The top left, top right and bottom left 8×8 block (one dimensional 48×1 matrix) is multiplied by the 8×48 secondary transform kernel and the resulting 16×1 secondary transformed coefficients can be obtained as shown in equation (15). The obtained coefficients are placed in the top left part of the TU block. Total 16 non-zero coefficients are taken in bitstreams, and all remaining secondary and primary transform blocks zeroed out later.

$$[16 \times 48 \;\; LFNST\ kernel] \times [48 \times 1 \;\; prim\ coeff] = [16 \times 1 \;\; \mathrm{sec}\ coeff] \quad (15)$$

## 3.2 Zeroing and Grouping Concept

Since, it is proposed to use a single 16×48 LFNST kernel for all block size i.e., 4×*N* and *N*×4 (with *N*>=4), the computation of the matrix multiplication is not possible for the TU blocks less than 8×8 block size because [16×1 prim coeff] cannot be multiplied with [16×48 LFNST kernel] to give [8×1 sec coeff]. To deal with this problem, the zeroing concept has been introduced, where for the smaller transform unit blocks, a sub-kernel is derived zeroing some of the 16×48 kernel matrix elements, in order to make it compatible for the matrix multiplication for the secondary transform process.

Firstly, the 16×48 LFNST kernel is zeroed out for the rightmost part of the kernel, leaving the non-zero 16×16 kernel present in the left part of the original kernel as the resultant, illustrated in Fig. 3-8.

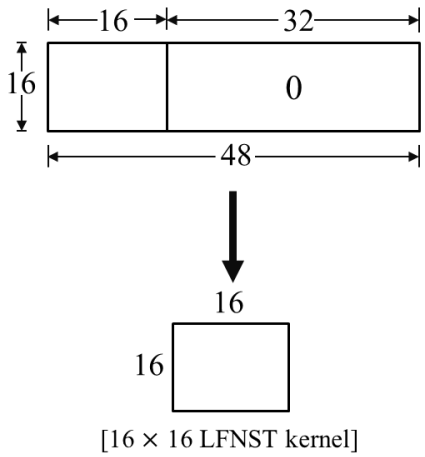


Figure 3-8. 16×16 LFNST sub kernel derivation by zeroing the rightmost part of 16×48 LFNST kernel elements.

Secondly, after analyzing the obtained kernel, the desired sub-sampled 16×16 kernel is derived by grouping the even and odd rows of the resultant

non-zero 16×16 kernel derived above from Fig. 3-8. It is also seen that for the particular LFNST candidate kernel set index i.e., 0, 1, 2 and 3 from Table. 3, which individually has further two indexed kernels (0 and 1), we can group the even and odd rows of particular index kernel and interchange amongst each other for the individual set index kernel i.e., the grouping of the odd and even rows of 16×48 kernel with set index 0 and index 0 i.e., $[R_{i,j}]_{0,0}$ is computed to derive the 16×16 sub-sampled kernel having set index 0 with index 1 i.e., $[R'_{i,j}]_{0,1}$ as shown in equation (17) and vice versa. Similarly, the sub-sampled 16×16 LFNST kernel with set index 1, 2 and 3 is derived by grouping the odd and even rows, and interchanging amongst the indexed kernel of particular set index of the non-zero resultant 16×16 kernel derived by zeroing the 16×48 LFNST kernel as shown in Fig. 3-8.

For grouping, the sign change of the kernel elements at the particular column index of the respective even and odd rows to obtain the sub-sampled 16×16 kernel to be applied for the lower TU blocks i.e. 4×$N$ or $N$×4 ($N$>=4). To derive the even rows, we pick the even indexed rows of the resultant non-zero 16×16 kernel and multiply with -1 to the row elements of the particular indexed columns whose column index is divisible by 2 or is the 9$^{th}$ column element. The derivation of the sub-sampled 16×16 LFNST kernel is obtained from (16) - (23).

$$[R'_{i,j}]_{0,0} = [R_{i,j}]_{0,1} \tag{16}$$

$$[R'_{i,j}]_{0,1} = [R_{i,j}]_{0,0} \tag{17}$$

$$[R'_{i,j}]_{2,0} = [R_{i,j}]_{2,1} \tag{18}$$

$$[R'_{i,j}]_{2,1} = [R_{i,j}]_{2,0} \tag{19}$$

$$[R'_{i,j}]_{2,0} = [R_{i,j}]_{2,1} \tag{20}$$

$$[R'_{i,j}]_{2,1} = [R_{i,j}]_{2,0} \tag{21}$$

$$[R'_{i,j}]_{3,0} = [R_{i,j}]_{3,1} \tag{22}$$

$$[R'_{i,j}]_{3,1} = [R_{i,j}]_{3,0}\,, \tag{23}$$

where, $R'_{i,j}$ is the sub-sampled 16×16 kernel, $R_{i,j}$ is the non-zero resultant 16×16 kernel obtained after zeroing 16×48 LFNST kernel, '$i$' is the even row index (0, 2, 4, ..14) and '$j$' is the column index of the kernel, respectively. The derivation of the $R_{i,j}$ negating the column indexed element for the even rows is mathematically derived by (24).

$$R_{i,j} = \begin{cases} (-1) \times R_{i,j}, & \text{if } j\%2 = 0 \,||\, j = 9, \\ R_{i,j}, & \text{otherwise,} \end{cases} \tag{24}$$

Similarly, the grouping of the odd row elements of the sub-sampled 16×16 kernel is derived from the non-zero resultant 16×16 kernel which is obtained by zeroing the 16×48 kernel using the equations (16) – (23), where the value of $R_{i,j}$ is calculated by changing the sign of the 4$^{th}$, 6$^{th}$, 8$^{th}$, 11$^{th}$, 13$^{th}$ and 15$^{th}$ column indexed kernel elements by multiplying with -1, for the individual odd indexed row kernel element respectively. The mathematical derivation of $R_{i,j}$ is obtained from equation (25).

$$R_{i,j} = \begin{cases} (-1) \times R_{i,j}, & \text{if } j \in \{4,6,8,11,13,15\}, \\ R_{i,j}, & \text{otherwise.} \end{cases} \tag{25}$$

The sub-sampled 16×16 LFNST kernel with grouped even and odd rows is illustrated in Fig. 3-9, where the highlighted kernel elements refers to the element applied with negation as per the equation (24) and (25).

| | | | | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Row 0 | 108 | -44 | -15 | 1 | -44 | 19 | 7 | -1 | -11 | 6 | 2 | -1 | 0 | -1 | -1 | 0 |
| Row 1 | -40 | -97 | 56 | 12 | -11 | 29 | -12 | -3 | 18 | 18 | -15 | -3 | -1 | -3 | 2 | 1 |
| Row 2 | 25 | -31 | -1 | 7 | 100 | -16 | -29 | 1 | -54 | 21 | 14 | -4 | -7 | 2 | 4 | 0 |
| Row 3 | -32 | -39 | -92 | 51 | -6 | -16 | 36 | -8 | 3 | 22 | 18 | -15 | 4 | 1 | -5 | 2 |

Figure 3-9. Sub-sampled 16×16 kernel for LFNST set index=0 and index =0 with grouped even and odd rows.

For the 4×4 TU block, it requires a 16×16 LFNST kernel for the computation of the secondary transform, resulting in maximum 8 non-zero secondary transform residue coefficients in the bitstreams as shown in equation (10). The 16×16 sub-sampled kernel from the 16×48 kernel is derived, as shown in Fig. 3-10.

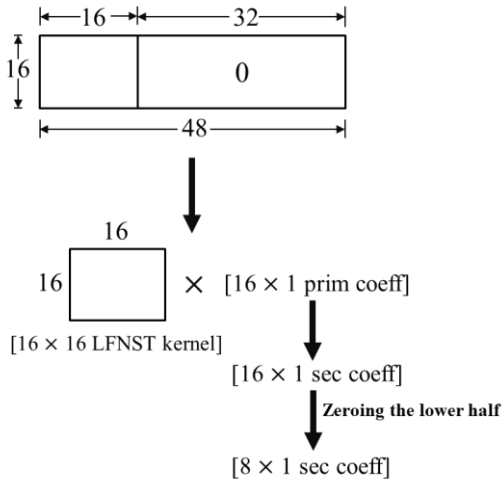


Figure 3-10. 16×16 LFNST applied to 4×4 TU block resulting in 8 residual coefficients as output.

For 4×*N* and *N*×4 (with *N*>=4) TU block, the same 16×16 sub-sampled kernel is derived from the 16×48 kernel as derived for the 4×4 block in Fig. 3-8, for the matrix multiplication of the TU and kernel, which results in the maximum of max 16 non-zero secondary transform residue coefficients in the

bitstream, zeroing out the remaining secondary and primary transform coefficients except for the top left corner as shown in Fig. 3-3. Another possibility is to derive the 16×32 sub kernel from the 16×48 kernel, but that would increase the complexity of the operation with increased multiplications and additional zeroing of the secondary transform residue coefficients to make it a maximum of 16 coefficients in the bitstream.

## 3.3 Proposed syntax for VVC standardization

As, we know the secondary transform LFNST is applied after the TU block undergo the primary transformation. Eventually, after the intra prediction process, the transformation process starts with the primary transform along with the initialization of the parameters of secondary transform. The following parsing steps are performed to initialize the parameters during the general transformation process for the scaled transform coefficients.

- **General transformation process**:

Input:

- A luma location (xTby, yTbY) specifying top left sample of current luma block
- nTbW: width of current TU
- nTbH: height of current TU
- cIdx: color component of current block
- d[x][y]: an array of dimension nTbW× nTbH, where x=0 to nTbW-1, y= 0 to nTbH-1

Design process:

– Check if LFNST index i.e. lfnst_idx[xTby][ yTby] is TRUE and (nTbW, nTbH) >= 4
    - Fetch the intra prediction mode (predModeintra) of TU block.
    - Determine output coefficients size (nLfnstoutSize) to 48 if (width,height) >= 8 else 16.
    - Determine total number of 4×4 blocks taken in consideration for LFSNT (log2LfnstSize): (width, height)>=8) ? 3 :2
    - Calculate the nLfnstSize by 1 << log2LfnstSize
    - Calculate the total output non zero ecoefficents of secondary transform (nonZeroSize) by : ( (width,width) == 4 || (width,width) == 8 ) ? 8 : 16

Output:

– An array r[x][y] with (nTbW) × (nTbH) elements of primary transform residual samples with ‘x’ rnages from 0 to nTbW-1 and ‘y’ ranges from 0 to nTbH-1.

- **LFNST transformation process**:

Input:

- The variable nonZeroSize calculated in the general transform process.
- nTrS: transform output length
- A matrix x[i] with a list of scaled non-zero transform coefficients where ‘i’ ranges between 0 to nonZeroSize-1
- The intra prediction mode (predModeIntra) of TU determined above to select the transform set.

- lfnstIdx: determines the LFNST index for transform selection in the selected LFNST set.

Design process:

- The LFNST kernel matrix is fetched after referring the look-up table where mapping of the intra prediction mode(predModeintra) and LFNST transform set index is done as shown in Table 5.
    - After the mapping of the index is done, the LFNST transform set index (lfnstTrSetIdx) is set which is used to fetch the kernel matrix
    - The low frequency transformation kernel matrix is fetched based on the lfnstTrSetIdx, and and nTrS derived above from the pool of secondary kernel with dimension of [nTrS]×[nonZeroSize] which is either 16×16 or 16×48 kernel matrix.
- The output secondary transform residue coefficient is mathematically computed as given in equation (26),

$$y[i] = \text{Clip3}\left(\text{CoefMin}, \text{Coeffmax}, \left(\left(\sum_{j=0}^{\text{nonZeroSize}-1} \text{lowFreqTransMatrix[i][j]} \times \text{x[j]} + 64\right) \gg 7\right)\right) \quad (26)$$

  Where, matrix ‘y[i]’ represents the output residue of secondary transform
- Zero out the remaining primary transform coefficients in the final residue.

Output:

- The matrix y[i] with output of non-zero secondary transform coefficients where ‘i’ ranges from 0 to nTrS (transform output length), which is sent to the bitstream.

# 4. Experimental Results and Discussions

## 4.1 Experimental Results of the Proposed Method

The simulation of the proposed method is conducted on the VVC reference software VTM-5.0 [23] under the CTC conditions [24]. The PC for the simulation has Centos 7 Linux operating system with intel ®Xeon ®Silver 4114 CPU @ 2.20 GHz processor with 220 cores and 156 GB of RAM. The simulation results are shown in Table. 8-9.

Table 8. Proposed method simulation result for All Intra configuration

| **Sequences** | **All Intra Main 10 over VTM5.0** | | | | |
|---|---|---|---|---|---|
| | Y | U | V | EncT | DecT |
| Class A1 | 0.01% | 0.03% | 0.04% | 94% | 101% |
| Class A2 | 0.01% | 0.02% | -0.03% | 93% | 100% |
| Class B | 0.04% | -0.05% | -0.01% | 93% | 99% |
| Class C | 0.03% | -0.04% | 0.04% | 95% | 95% |
| Class E | 0.06% | -0.04% | 0.02% | 94% | 96% |
| Overall | 0.03% | -0.02% | 0.01% | 94% | 98% |
| Class D | 0.00% | 0.05% | 0.09% | 94% | 101% |
| Class F | 0.01% | -0.02% | 0.02% | 96% | 97% |

Table 9. Proposed method simulation result for Random Access configuration

| **Sequences** | **Random Access Main 10 over VTM5.0** | | | | |
|---|---|---|---|---|---|
| | Y | U | V | EncT | DecT |
| Class A1 | 0.03% | 0.04% | 0.08% | 97% | 100% |
| Class A2 | -0.01% | 0.08% | 0.05% | 98% | 100% |
| Class B | 0.03% | 0.03% | -0.13% | 98% | 98% |
| Class C | 0.03% | -0.09% | 0.03% | 98% | 99% |
| Overall | 0.02% | 0.01% | -0.01% | 97% | 99% |
| Class D | 0.06% | 0.23% | 0.11% | 98% | 102% |
| Class F | 0.06% | 0.10% | 0.05% | 97% | 101% |

Table. 8 shows the simulation result for the proposed method on VTM-5.0 anchor under CTC condition for AI configuration. There is a gain of 0.2% for U-chroma, whereas negligible loss for the Y-luma and V-chroma component. The gain for the classes B-F is substantial for the chrominance (U). There is a substantial reduction in encoding time by 6% and by 2% in the decoding time. The overall result has a gain of 0.02% gain for chrominance with 94% encoding time and 98%

Table. 9 illustrates the simulation result for the proposed method for the RA configuration. There is a gain of 0.01% for the chrominance (V) with a maximum gain of 0.13% for class B. the performance for the luminance is good for the higher-class resolution images i.e., Class A1 and A2. There is also a 3% reduction in the encoding time. The overall result for the RA configuration is a gain of 0.01% for chrominance and loss of 0.02% for luminance with 97% encoding time and 99% decoding time.

The detailed experimental result for the different test sequences are as following:

Table 10. Simulation results of AI configuration

| | Test Sequences | BD-rate (piecewise cubic) | | |
|---|---|---|---|---|
| | | Y | U | V |
| Class A1 (4K) | *Tango2* | -0.01% | -0.04% | 0.17% |
| | *FoodMarket4* | 0.01% | 0.09% | -0.03% |
| | *Campfire* | 0.01% | 0.05% | -0.01% |
| Class A2 (4K) | *CatRobot1* | 0.02% | 0.02% | -0.05% |
| | *DaylightRoad2* | 0.00% | 0.02% | -0.03% |
| | *ParkRunning3* | 0.00% | 0.03% | -0.01% |
| Class B (1080p) | *MarketPlace* | 0.01% | -0.02% | -0.16% |
| | *RitualDance* | 0.01% | -0.19% | -0.04% |
| | *Cactus* | 0.03% | -0.05% | -0.06% |
| | *BasketballDrive* | 0.06% | -0.18% | 0.06% |
| | *BQTerrace* | 0.09% | 0.19% | 0.19% |
| Class C (WVGA) | *BasketballDrill* | 0.09% | -0.17% | -0.08% |
| | *BQMall* | 0.04% | -0.04% | 0.10% |
| | *PartyScene* | 0.00% | 0.03% | 0.05% |
| | *RaceHorses* | 0.00% | 0.01% | 0.08% |
| Class D (WQVGA) | *BasketballPass* | -0.03% | 0.11% | -0.06% |
| | *BQSquare* | 0.00% | 0.17% | -0.08% |
| | *BlowingBubbles* | 0.03% | -0.12% | 0.10% |
| | *RaceHorses* | -0.01% | 0.05% | 0.39% |
| Class E (720p) | *FourPeople* | -0.01% | -0.08% | -0.03% |
| | *Johny* | 0.20% | -0.02% | 0.17% |
| | *KristenAndSara* | -0.01% | -0.01% | -0.09% |
| Average Class A1 | | 0.01% | 0.03% | 0.04% |
| Average Class A2 | | 0.01% | 0.02% | -0.03% |
| Average Class B | | 0.04% | -0.05% | -0.01% |
| Average Class C | | 0.03% | -0.04% | 0.04% |
| Average Class D | | 0.00% | 0.05% | 0.09% |
| Average Class E | | 0.06% | -0.04% | 0.02% |
| Overall Average | | 0.03% | -0.02% | 0.01% |

Based on Table. 10, no significant loss in found in AI for higher resolution classes for luminance, but some gain can be observed in Class A1, A2, and B for the chrominance (V) and some observatory gain for class B and C for chrominance (U).

Table 11. Simulation result of RA configuration

| | Test Sequences | BD-rate (piecewise cubic) | | |
|---|---|---|---|---|
| | | Y | U | V |
| Class A1 (4K) | *Tango2* | 0.01% | -0.21% | -0.01% |
| | *FoodMarket4* | 0.02% | 0.05% | 0.30% |
| | *Campfire* | 0.06% | 0.28% | -0.05% |
| Class A2 (4K) | *CatRobot1* | -0.03% | -0.03% | 0.01% |
| | *DaylightRoad2* | 0.02% | 0.21% | 0.10% |
| | *ParkRunning3* | -0.02% | 0.06% | -0.04% |
| Class B (1080p) | *MarketPlace* | -0.02% | 0.11% | 0.00% |
| | *RitualDance* | 0.00% | -0.15% | -0.24% |
| | *Cactus* | -0.01% | 0.19% | -0.30% |
| | *BasketballDrive* | 0.03% | -0.17% | 0.02% |
| | *BQTerrace* | 0.14% | 0.18% | -0.14% |
| Class C (WVGA) | *BasketballDrill* | 0.08% | -0.25% | 0.09% |
| | *BQMall* | 0.03% | -0.15% | 0.04% |
| | *PartyScene* | 0.02% | -0.01% | -0.09% |
| | *RaceHorses* | -0.01% | 0.05% | -0.28% |
| Class D (WQVGA) | *BasketballPass* | 0.04% | -0.27% | -0.13% |
| | *BQSquare* | 0.12% | 0.10% | -0.31% |
| | *BlowingBubbles* | 0.08% | 0.10% | 0.10% |
| | *RaceHorses* | 0.00% | 0.20% | 0.19% |
| Average Class A1 | | 0.03% | 0.04% | 0.08% |
| Average Class A2 | | -0.01% | 0.08% | 0.05% |
| Average Class B | | 0.03% | 0.03% | -0.13% |
| Average Class C | | 0.03% | -0.09% | 0.03% |
| Average Class D | | 0.06% | 0.23% | 0.11% |
| Overall Average | | 0.02% | 0.01% | -0.01% |

Based on Table. 11, no significant loss is found for the luminance for the higher resolution classes A1 and A2, whereas some substantial gain is found for both chrominance U and V for some lower resolution test sequences of class B, C and D. From the above results, we have seen the gain for AI configuration is 0.02% and 0.01% for RA configuration with a significant reduction of encoding time of 6% and 3% for respective configurations.

## 4.2 Comparison with the state-of-the-art Methods

While comparing the proposed obtained simulation result with the memory reduction method of related work, the simulation result of contribution in [22] is shown in Table. 12. Analyzing the results, the bit rate of our proposed method is better along with the encoding time, although proposal in [22] also reduces the encoding time by 3%. Although the luminance BD rate for RA configuration is better in their proposal, they use the entire 16×48 kernel matrix-forming L-shaped output residue during the LFNST matrix multiplication operation, which doesn't help to get a substantial reduction in the encoding time as achieved by our proposal.

Table 12. Simulation results of proposal in [22].

| **Sequences** | **All Intra Main 10 over VTM5.0** | | | | |
|---|---|---|---|---|---|
| | Y | U | V | EncT | DecT |
| Class A1 | -0.02% | 0.14% | 0.02% | 98% | 101% |
| Class A2 | -0.01% | 0.16% | 0.15% | 97% | 100% |
| Class B | 0.06% | 0.01% | 0.03% | 97% | 99% |
| Class C | 0.08% | 0.07% | 0.12% | 96% | 95% |
| Class E | 0.14% | -0.14% | -0.13% | 98% | 97% |
| Overall | 0.05% | 0.04% | 0.04% | 97% | 98% |
| Class D | 0.12% | 0.05% | 0.24% | 97% | 101% |
| Class F | 0.01% | -0.02% | 0.27% | 98% | 98% |

| Sequences | Random Access Main 10 over VTM5.0 | | | | |
|---|---|---|---|---|---|
| | Y | U | V | EncT | DecT |
| Class A1 | -0.01% | 0.03% | -0.05% | 99% | 100% |
| Class A2 | 0.00% | 0.15% | 0.05% | 99% | 100% |
| Class B | 0.01% | 0.08% | -0.01% | 99% | 98% |
| Class C | 0.01% | 0.16% | 0.01% | 100% | 99% |
| Overall | 0.01% | 0.11% | 0.00% | 99% | 99% |
| Class D | 0.07% | 0.36% | 0.13% | 100% | 102% |
| Class F | 0.00% | 0.28% | 0.10% | 100% | 101% |

The simulation result of the limiting coefficients method in [21] shows a significant reduction in the encoding time as they target more for the complexity reduction based on the zeroing of the primary transform for the 4×4 LFNST and 8×8 LFNST. As mentioned in section 2.2, they proposed the tests in four parts, and the significant gain was given by allowing maximum of 16 non-zero coefficients in the residual and zeroing of primary transform coefficients for 8×8 LFNST. Although the encoding time for this proposal is high and the proposed method in this thesis, but the optimal BD loss for the luminance (Y) and chrominance (V) is more in our proposal. Also, the loss for the higher-class resolution test sequences in Class A1 and A2 is higher in their proposal for AI configuration.

Table 13. Simulation results of proposal in [21].

| Sequences | All Intra Main 10 over VTM5.0 | | | | |
|---|---|---|---|---|---|
| | Y | U | V | EncT | DecT |
| Class A1 | 0.00% | 0.11% | 0.01% | 91% | 97% |
| Class A2 | 0.00% | 0.10% | 0.11% | 88% | 99% |
| Class B | 0.08% | -0.06% | 0.04% | 90% | 100% |
| Class C | 0.10% | -0.12% | 0.05% | 88% | 100% |
| Class E | 0.10% | -0.06% | 0.03% | 91% | 101% |
| Overall | 0.06% | -0.02% | 0.05% | 90% | 99% |
| Class D | 0.10% | 0.14% | -0.12% | 89% | 101% |
| Class F | 0.02% | 0.09% | 0.09% | 94% | 99% |

| Sequences | Random Access Main 10 over VTM5.0 | | | | |
|---|---|---|---|---|---|
| | Y | U | V | EncT | DecT |
| Class A1 | 0.00% | -0.17% | 0.12% | 97% | 98% |
| Class A2 | 0.07% | 0.16% | 0.05% | 98% | 98% |
| Class B | 0.04% | -0.15% | -0.17% | 99% | 100% |
| Class C | -0.03% | -0.30% | -0.29% | 100% | 103% |
| Overall | 0.02% | -0.13% | -0.10% | 99% | 100% |
| Class D | 0.03% | 0.07% | 0.04% | 100% | 101% |
| Class F | -0.01% | -0.08% | 0.00% | 99% | 101% |

Analyzing the results based on Table. 14, the gain for the chrominance (U) is same as the proposed method. Although the proposal in [20] have significant 10% reduction in the encoding time, but it still uses two sets of LFNST kernel with no memory reduction. The gain for the RA configuration is like the proposed method with 2% reduction in the proposed method. Also, the losses for the higher resolution test sequences can be found for the Class A1 and A2. The overall result based on Table. 12, is showing less significant loss for the RA configuration with 0.02% for luma, 0.01% for U-chroma and gain of 0.01% for V-chroma, respectively.

Table 14. Simulation of proposal in [20].

| Sequences | All Intra Main 10 over VTM5.0 | | | | |
|---|---|---|---|---|---|
| | Y | U | V | EncT | DecT |
| Class A1 | 0.00% | 0.11% | 0.01% | 91% | 97% |
| Class A2 | 0.00% | 0.10% | 0.11% | 88% | 99% |
| Class B | 0.08% | -0.06% | 0.04% | 90% | 100% |
| Class C | 0.10% | -0.12% | 0.05% | 88% | 100% |
| Class E | 0.10% | -0.06% | 0.03% | 91% | 101% |
| Overall | 0.06% | -0.02% | 0.05% | 90% | 99% |
| Class D | 0.10% | 0.14% | -0.12% | 89% | 101% |
| Class F | 0.02% | 0.09% | 0.09% | 94% | 99% |

| Sequences | Random Access Main 10 over VTM5.0 | | | | |
|---|---|---|---|---|---|
| | Y | U | V | EncT | DecT |
| Class A1 | 0.03% | 0.04% | 0.08% | 97% | 100% |
| Class A2 | -0.01% | 0.08% | 0.05% | 98% | 98% |
| Class B | 0.03% | 0.03% | -0.13% | 98% | 102% |
| Class C | 0.03% | -0.09% | 0.03% | 98% | 99% |
| Overall | 0.02% | 0.01% | -0.01% | 97% | 100% |
| Class D | 0.06% | 0.23% | 0.11% | 98% | 101% |
| Class F | 0.06% | 0.10% | 0.05% | 97% | 100% |

# 5. Conclusions

In this thesis, the reduction in the complexity of the secondary transform i.e., LFNST is proposed. Since the memory consumption to store the transform kernels is always a major issue in video coding standardization of the VVC, this thesis deals with the usage of less memory to store the secondary transform kernels along with reducing the computational complexity. The proposed method uses a single 16×48 secondary transform kernel for the LFNST computation which requires only 6KB of memory instead of two kernels that required 8KB of storage. For the smaller blocks of size 4×4, $N$×4 and 4×$N$ ($N$>4), a sub-sampled kernel is derived using the 16×48 kernel from the zeroing and grouping concept. For the remaining larger block sizes 8×8, $N$×8, 8×$N$ ($N$>8), the entire 16×48 kernel is used. Also, the zeroing of the primary transform coefficients to obtain max 16 non-zero secondary transform residue coefficients in the bitstream reduces the complexity of the encoder by reducing the encoder runtime substantially. The proposed method in this thesis gives comparable results with the VTM-5.0 anchor with negligible loss and reduced encoding time by 6%, which signifies that the proposed LFNST design is of significant use in the VVC model.

# 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, July 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.

[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, "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 Transactions on Pattern Analysis and Machine Intelligence, 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] J.J. Rissanen, G.G. Langdon (March 1979), "Arithmetic coding" (PDF) IBM Journal of Research and Development. 23 (2): 149–162. doi:10.1147/rd.232.0149.

[14] D. Marpe, H. Schwarz, and T. Wiegand, "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 5 (VTM 5), [JVET-N1002]," JVET of ITU-T SG 16 WP 3 and ISO/IEC JTC 1/SC 29/WG 11, 14th Meeting: Geneva, CH, Mar. 2019.

[20] M. Siekmann, H. Schwarz, D. Marpe, "CE6-2.1: Simplification of Low Frequency Non-Separable Transform," JVET of ITU-T SG 16 WP 3 and ISO/IEC JTC 1/SC 29/WG 11, 15th Meeting: Gothenburg, SE, July 2019.

[21] M. Koo, J. Nam, J. Lim, "Non-CE6: LFNST simplification based on the methods proposed in CE6-2.1a," JVET of ITU-T SG 16 WP 3 and ISO/IEC JTC 1/SC 29/WG 11, 15th Meeting: Gothenburg, SE, July 2019.

[22] X. Zhao, X. Li, S. Liu, “CE6-related: Unified LFNST using block size independent kernel,” JVET of ITU-T SG 16 WP 3 and ISO/IEC JTC 1/SC 29/WG 11, 15th Meeting: Gothenburg, SE, July 2019.

[23] JVET VTM software — JVET. https://vcgit.hhi.fraunhofer.de/jvet/VVCSoftware_VTM/tree/VTM-5.0.

[24] 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, Oct. 2018.

[25] 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, Jan. 2019.

[26] Z. Zhang, R. Yu, “Non-CE6: On LFNST transform set selection for a CCLM coded block,” Joint Video Experts Team (JVET) of ITU-T SG 16 WP 3 and ISO/IEC JTC 1/SC 29/WG 11 15th Meeting: Gothenburg, SE, 3-12 July 2019.

[27] S. Park, Je-won. Kang, “Fast Affine Motion Estimation for Versatile Video Codec (VVC) Encoding,” IEEE Access, volume 7, 2019.

[28] D. Ruiz, G. Fernandez-Escribano, J. Luis Martinez, P Cuenca, “A unified architecture for fast HEVC intra-prediction coding,” Journal of Real-Time Image Processing, 2017.

[29] J. Nilson, “Inter-Picture Prediction for Video Compression using Low Pass and High Pass Filters,” Uppsala University, September 2017.

[30] H. Yang, L. Shen, X. Dong, Q. Ding, P. An, G. Jiang, “Low Complexity CTU Partition Structure Decision and Fast Intra Mode Decision for Versatile Video Codec,” IEEE Transactions on Circuits and Systems for Video Technology, 2019.

[31] J. Kang, C. Kim, “On DCT Coefficient Distribution in Video coding Using Quad-tree Structured Partition,” Proceedings of 2014 APSIPA Annual Summit and Conference, Siem Reap, city of Angkor Wat, Cambodia, December 9-12, 2014.

[32] F. Racape, “CE3-related: Wide-angle intra prediction for non-square blocks,” Joint Video Experts Team (JVET) of ITU-T SG 16 WP 3 and

ISO/IEC JTC 1/SC 29/WG 11 JVET-K500 15th Meeting: Gothenburg, SE, 10-18 July 2018.

[33] Santiago De-Luxan-Hernandez, “CE3: Intra Sub-Partitions Coding modes,” Joint Video Experts Team (JVET) of ITU-T SG 16 WP 3 and ISO/IEC JTC 1/SC 29/WG 11 JVET-M0102 13th Meeting: Marrakech, MA, 9-18 Jan. 2019.

[34] J. Pfaff, B. Stallenberger, M. Schafer, P. Merkle, P. Helle, T. Hinz, H. Schwarz, D. Marpe, T. Wiegand, “CE3: Affine linear weighted intra prediction (CE3-4.1, CE3-4.2),” Joint Video Experts Team (JVET) of ITU-T SG 16 WP 3 and ISO/IEC JTC 1/SC 29/WG 11, JVET-N0217, 14th Meeting: Geneva, SW, 19-27 March 2019.

[35] A. Said, X. Zhao, “Description of core Experiment 6 (CE6): Transform and transform signalling,” Joint Video Experts Team (JVET) of ITU-T SG 16 WP 3 and ISO/IEC JTC 1/SC 29/WG 11, JVET-K1026, 11th Meeting: Ljubljana, SI, 11-18 July 2018.

[36] H. Samet, “The quadtree and related hierarchical data structures,” Computer Survey, vol. 16, no. 2, pp. 187-260, Jun 1984.

[37] S. Wenger, “H.264/AVC over IP,” IEEE Transactions on Circuits and System for Video Technology, vol. 13, no. 7, pp. 645-656, Jul. 2003.

[38] K.McCann, “Summary of Evaluation of TMuC Tools in TEI2,” ITU-T/ISO/IEC Joint Collaborative Team on Video Coding (JCT-VC) document JCTVC-C225, Oct. 2010.

[39] F. Bossen, B. Bross, K. Suhring, and D. Flynn, “HEVC complexity and implementation analysis,” IEEE Transactions on Circuits and System for Video Technology, vol. 22, no. 12, pp. 1684-1695, Dec. 2012.

[40] J. -R Ohm, G. J. Sullivan, H. Schwarz, T. K. Tan, T. Wiegand, “Comparison of the coding efficiency of video coding standards - Including High Efficiency Video Coding (HEVC),” IEEE Transactions on Circuits and System for Video Technology, vol. 22, no. 12, pp. 1668-1683, Dec. 2012.

[41] G. J. Sullivan, R. L. Baker, “Rate-distortion optimized motion compensation for video compression using fixed or variable size blocks,” in Proc. GLOBECOM', Dec. 1991, pp. 85-90.

[42] T. Wiegand, M. Lightstone, D. Mukherjee, T. G. Campbell, and S. K. Mitra, “Rate-distortion optimized mode selection for very low bitrate video coding and emerging H.263 standard,” IEEE Transactions on Circuits and System for Video Technology, vol. 6, no. 4,pp. 182-190, Apr. 1996.

[43] S. De-Luxan-Hernandez, H. Schwarz, D. Marpe, and T. Wiegand, “Line-Based Intra Prediction for Next Generation Video Coding,” 25th IEEE International Conference on Image Processing (ICIP), Oct. 2018, pp. 221–225.

[44] S. De-Luxan-Hernandez, H. Schwarz, D. Marpe, and T. Wiegand, “Fast Line-Based Intra Prediction for Video Coding,” IEEE International Symposium on Multimedia (ISM), Dec. 2018, pp. 135–138.

[45] S. De-Luxan-Hernandez, V. George, J. Ma, T. Nguyen, H. Schwarz, D. Marpe, and T. Wiegand, “An Intra Subpartition Coding Mode For VVC,” in 2019 26th IEEE International Conference on Image Processing (ICIP), Oct. 2019, pp. 1203–1207.

[46] F. Racape, G. Rath, F. Urban, L. Zhao, S. Liu, X. Zhao, X. Li, A. Filippov, V. Rufitsky, and J. Chen, “CE3- related: Wide-angle intra prediction for non-square blocks,” Document JVET-K0500 of JVET, Ljubljana, SI, July 2018.

[47] Y. Lin, M. Mao, S. Song, J. Zheng, J. An, and C. Zhu, “Prediction dependent transform for intra and inter frame coding,” Document JVET-J0064, San Diego, US, April 2018.

[48] B. Bross, J. Chen, and S. Liu (Editors), “Versatile Video Coding (Draft 3),” Document JVET-L1001, Macao, CN, Oct. 2018.

[49] Xiaoran Cao, Changcai Lai, Yunfei Wang, Lingzhi Liu, Jianhua Zheng, and Yun He, “Short Distance Intra Coding Scheme for High Efficiency Video Coding,” IEEE Transactions on Image Processing, vol. 22, no. 2, pp. 790–801, Feb. 2013.

[50] X. Ma, H. Yang, and J. Chen, “CE3: Tests of cross-component linear model in BMS,” Joint Video Exploration Team (JVET) of ITU-T SG 16 WP 3 and ISO/IEC JTC 1/SC 29/WG 11, JVET-K0190, Ljubljana, SL, July 2018.

[51] G. Laroche, J. Taquet, C. Gisquet, and P. Onno, “CE3-5.1: On cross-component linear model simplification,” Joint Video Exploration Team

(JVET) of ITU-T SG 16 WP 3 and ISO/IEC JTC 1/SC 29/WG 11, JVET-L0191, Macau, CN, Oct. 2018.

[52] J. Lainema, “CE3: Wide-angle intra prediction,” Joint Video Exploration Team (JVET) of ITU-T SG 16 WP 3 and ISO/IEC JTC 1/SC 29/WG 11, JVET-K0046, Ljubljana, SL, Jul. 2018.

[53] L. Zhao, X. Zhao, S. Liu, and X. Li, “CE3-related: Unification of angular intra prediction for square and non-square blocks,” Joint Video Exploration Team (JVET) of ITU-T SG 16 WP 3 and ISO/IEC JTC 1/SC 29/WG 11, JVET-L0279, Macau, CN, Oct. 2018.

[54] A. Kumar, S. Shrestha, B. Lee, Y. Lee and J. Park, “Non-CE3: WAIP restriction for square sub-CU blocks partitioned by ISP,” Joint Video Exploration Team (JVET) of ITU-T SG 16 WP 3 and ISO/IEC JTC 1/SC 29/WG 11, 15th Meeting, JVET-O0632, Gothenburg, SE, 3-12 July 2019.

[55] N. Choi, Y. Piao, K. Choi, and C. Kim “CE3.3 related: Intra 67 modes coding with 3MPM,” Joint Video Exploration Team (JVET) of ITU-T SG 16 WP 3 and ISO/IEC JTC 1/SC 29/WG 11, JVET-K0529, Ljubljana, SL, Jul. 2018.

[56] X. Zhao, V. Seregin, A. Said, K. Zhang, H. E. Egilmez, and M. Karczewicz, “Low Complexity Intra Prediction Refinements for Video Coding,” in IEEE Picture Coding Symposium (PCS), pp. 139–143, San Fransisco, USA, Jun. 2018.

[57] L. Li, J. Heo, J. Choi, J. Choi, S. Yoo, S. Kim, and J. Lim, “CE3-6.2.1: Extended MPM list,” Joint Video Exploration Team (JVET) of ITU-T SG 16 WP 3 and ISO/IEC JTC 1/SC 29/WG 11, JVET-L0165, Macao, CN, Oct. 2018.

[58] A. Filippov, V. Rufitskiy, J. Chen, G. Van der Auwera, A. K. Ramasubramonian, V. Seregin, T. Hsieh and M. Karczewicz, “CE3: A combination of tests 3.1.2 and 3.1.4 for intra reference sample interpolation filter,” Joint Video Exploration Team (JVET) of ITU-T SG 16 WP 3 and ISO/IEC JTC 1/SC 29/WG 11, JVET-L0628, Macao, CN, Oct. 2018.

[59] S. H. Park, T. Dong, and E. S. Jang, ‘‘Low complexity reference frame selection in QTBT structure for JVET future video coding,’’ in Proc. IWAIT, Chiang Mai, Thailand, Jan. 2018, pp. 1–4.

[60] Ultra Video Group. Laboratory of Pervasive Computing at Tampere University of Technology. Accessed: Nov. 9, 2018. [Online]. Available: http://ultravideo.cs.tut.fi/

[61] S. Shrestha, A. Kumar, B. Lee, Y. Lee and J. Park, "Non-CE6: Reducing secondary transform kernel for specific residual block size," Joint Video Exploration Team (JVET) of ITU-T SG 16 WP 3 and ISO/IEC JTC 1/SC 29/WG 11, 15th Meeting, JVET-O0642, Gothenburg, SE, 3-12 July 2019.

[62] S. Shrestha, A. Kumar, B. Lee, Y. Lee and J. Park, "Non-CE6: Simplification of LFNST LUT," Joint Video Exploration Team (JVET) of ITU-T SG 16 WP 3 and ISO/IEC JTC 1/SC 29/WG 11, 16th Meeting, JVET-P0313, Geneva, SW, Sept. 2019.

[63] S. Shrestha, A. Kumar, B. Lee, Y. Lee and J. Park, "Non-CE3: Simplification of harmonization of LFNST and MIP," Joint Video Exploration Team (JVET) of ITU-T SG 16 WP 3 and ISO/IEC JTC 1/SC 29/WG 11, 16th Meeting, JVET-P0503, Geneva, SW, Sept. 2019.

[64] A. Kumar, S. Shrestha, B. Lee, Y. Lee and J. Park, "Non-CE3: LFNST restriction based on MIP," Joint Video Exploration Team (JVET) of ITU-T SG 16 WP 3 and ISO/IEC JTC 1/SC 29/WG 11, 16th Meeting, JVET-P0313, Geneva, SW, Sept. 2019.

[65] J. Han, A. Saxena, V. Melkote, and K. Rose, "Jointly optimized spatial prediction and block transform for video and image coding," IEEE Transactions on Image Processing, vol. 21, no. 4, pp. 1874-1884, Apr. 2012.

# Acknowledgement

I would like to express my deepest gratitude to my advisor Prof. Bumshik Lee for his invaluable support, suggestions, encouragement, and precious guidance throughout my research work. I am very fortunate to have a supervisor who constantly encouraged and monitored my work and responded to my queries so promptly during my graduate studies. I am thankful to all professors of Information and Communication Engineering department for providing a suitable platform to make this research success.

I would like to extend my gratitude to the committee members, Prof. Dong-you Choi and Prof. Jae-Young Pyun, for their insightful guidance and suggestions.

I would like to show my appreciation to all my fellow lab mates of Multimedia Information Processing Lab, for their extensive professional and personal guidance and their valuable suggestions regarding thesis work.

As ever, I would like to express my profound gratitude to the role model of my life, my parents and family, whose love and guidance are always with me in whatever I pursue. I shall be grateful forever for your constant love and support.