# Efficient Dedicated Hardware Design System for the VVC Low-Frequency Non-Separable Transform

Jones Goebel, Luciano Agostini, Bruno Zatt and Marcelo Porto

Video Technology Research Group (ViTech) - Group of Architectures and Integrated Circuits (GACI)
Graduate Program in Computing (PPGC) - Federal University of Pelotas (UFPel) – Pelotas, Brazil
{jwgoebel, agostini, zatt, porto}@inf.ufpel.edu.br

***Abstract*— This paper proposes a dedicated hardware architecture for the Low-Frequency Non-Separable Transform (LFNST) of the Versatile Video Coding (H.266/VVC) standard. The VVC defines two stages of transformation, where the first stage uses traditional transform types (e.g. DCT-II, DCT-VII and DST-VII), while the secondary transform stage applies the LFNST. The LFNST is used to transform the coefficients that were transformed by the DCT-II in the primary transform, but only those from the residues that came from the intra prediction. The developed LFNST system design exploits the Clock Crossing Domain technique to extract the best relation between performance and area/power. Consequently, the design operates with two clock domains, where the core operates at a four times higher frequency than the primary transform. The ASIC synthesis results for a TSMC 40nm standard-cells library indicate that our design can process UHD 4K videos at 120 frames per second while using an area of 69.68 Kgates and with a power dissipation of 40.46 mW. When compared with related works, our design presented the lowest power dissipation and energy consumption per sample.**

***Index Terms*— Video Coding, VVC, LFNST, hardware architecture, Clock Crossing Domain.**

## I. INTRODUCTION

Digital video became omnipresent in our daily lives driven by the popularization of digital systems, especially handheld battery-powered devices such as smartphones. Those devices can be integrated with the most advanced technology, such as Super AMOLED display, 5G technology, and many dedicated System-on-a-Chips (SoCs) to process and handle all data from the user. Consequently, those devices also must be embedded with the most efficient video encoders/decoders in the market to provide the user with better video compression and real-time performance with low energy consumption when performing video encoding and/or decoding.

However, digital video has two important aspects that must be considered, and they are related to storage and transmission: (i) the prohibitive number of bits needed to represent digital videos without proper compression and (ii) the increasing demand for higher resolution videos, such as UHD 4K (3840x2160 pixels) and 8K (7680x 4320 pixels) also with higher frame rates, such as 60 and 120 frames per second (fps). Those aspects encourage the development of new video encoders that must be more efficient than their antecessors, in other words, the new video encoding must improve the compression (reduce the number of necessary bits to represent the video) and maintain the visual quality of the encoded video when compared to its antecessors.

It is possible to find a variety of video encoders in the market nowadays, where each one has unique features and tools to address the required demand of the users. For instance, some examples of preview video encoders are Advanced Video Coding (H.264/AVC) [1], High Efficiency Video Coding (H.265/HEVC) [2] and Google VP9/VP8 [3], which coexist nowadays with state-of-the-art encoders, as the AOMedia Video 1 (AV1) [4] and the Versatile Video Coding (H.266/VVC) [5].

All highlighted video encoders share the same basic steps to perform video processing, like: intra and inter prediction, transform, quantization, entropy, and filters [2]. Each one of these steps is responsible to perform important operations in the video encoder contributing to reduce a lot of data redundancies present in the video content. But the used tools inside of each one of these steps can be very different among the encoders, providing different results of coding efficiency and with very different computational cost demands.

Among all those video encoding steps, the transform has the role of changing the residues (came from the difference of the original block and the blocks predicted by the intra or inter prediction steps) from the spatial to the frequency domain, preparing the coefficients for the quantization module, where those frequencies which are less relevant to the human visual system can be attenuated or discarded, allowing very high compression rates. In this way, the transform has changed in many ways over the video encoders generations to reach an ever-growing demand for higher video encoding efficiency. For instance, the H.26x family uses the Discrete Cosine Transform II (DCT-II) as the main transform, but it is possible to observe some modifications with the use of transform step over time. For example, H.263 [6] defines only a floating-point precision DCT-II in its transform step. The H.264/AVC, defines the utilization of the DCT-II to process the residues with only two block sizes (4x4 and 8x8), but in an integer approximation. Furthermore, it was the first to define a secondary transform stage (two transform steps) with the Hadamard transform (sizes of 2x2 and 4x4) being applied to the DC coefficients from the DCT-II [1], [7].

The H.265/HEVC defines only one transform stage, however, with the addition of a second transform type beyond the DCT-II, the Discrete Sine Transform VII (DST-VII). Besides, HEVC defines four squared transform block sizes to process the residues: four block sizes for DCT-II (32x32, 16x16, 8x8 and 4x4) and only one block size for the DST-VII (4x4) [2], [8]. The DST-VII is only allowed to be applied for residual blocks that were predicted by intra prediction and both transform matrices (DCT-II and DST-VII) are defined with integer precision.

10.29292/jics.v18i1.668

Another family of video encoders is VP9/VP8 [3], which defines the utilization of DCT-II and ADST (Asymmetric Discrete Sine Transform), both in integer precision. Besides, the DCT-II and ADST are based on a fast transform approach [3], [9]. Moreover, DCT-II supports four squared blocks (32x32, 16x16, 8x8, and 4x4) and the ADST supports three block sizes (the 32x32 size is not allowed in the ADST). The VP9/VP8 also was the first encoder to allow the combinations of different transform types to compose the 2D transform, which means that different transforms can be applied in the vertical and horizontal directions.

The novel generation of video encoders, like AV1 and VVC, brings a remarkable improvement in the transform step, with a lot of important novelties. For instance, the AV1 encoder (based on the VP9/VP8) can support squared and rectangular block sizes, ranging from 64x64 down to 4x4 samples. The AV1 also defines the use of another two transform types, beyond DCT-II and ADST already used in VP9/VP8: FlipADST (Flipped Asymmetric Discrete Sine Transform) and IDTX (Identity Transform). AV1 also allows the combinations of different 1D transforms to compose the 2D transform, as already supported in VP9/VP8.

The VVC supports 39 different block sizes ranging from 4x4 up to 64x64, including squared and rectangular blocks. The VVC defines two transform stages, where the primary transform stage is based on DCT-II, DCT-VIII and DST-VII, and the secondary transform stage is based on the Low-Frequency Non-Separable Transform (LFNST) [10]. The LFNST is applied only for residual blocks predicted by the intra prediction and which have used the DCT-II as the primary transform. The VVC also allows the combination of different 1D transforms to compose the 2D transform.

This work proposes a parallel and energy-efficient hardware system design for the secondary transform stage of VVC, which attends to the real-time constraints of performance and energy/power required by the high-definition video applications, which could not be reached by software implementation, especially for battery-powered devices. Thereby, the developed design exploits the use of two clocks domains, where the LFNST core operates at a clock frequency that is four times higher than the other domain, which receives and delivers the samples to the core. The proposed hardware system design was described in Verilog and synthesized for an ASIC with a 45nm TSMC standard cell library using the Genus compiler and validated using Incisive tool integrated with MATLAB (co-simulation). The design was evaluated considering the processing of Ultra-High-Definition videos with 4098×2160 pixels (UHD 4K) at 120 and 60 frames per second (fps).

The paper is organized as follows. Section II presents the background of VVC but focuses on the LFNST. The following Section III presents the related works in the literature. Section IV presents a brief analysis of the transform stage (primary transform and LFNST) used in the VVC. Section V details the developed system design to the secondary transform stage. The results and the comparison with related works are presented in Section VI. Finally, Section VII concludes this paper.

## II. VCC BACKGROUND

The VVC was released in 2020 and it was developed by the Joint Video Exploration Team (JVET), a collaboration of the ITU-T Video Coding Experts Group (VCEG) and the ISO/IEC Moving Pictures Expert Group (MPEG). In this way, the VVC is the successor of H.265/HEVC and has the goal of doubling the compression rate while keeping the same objective quality of the encoded video when compared with the H.265/HEVC [11]. The VVC has a reference software, called the VVC Test Model (VTM) [12], which implements all encoding tools of the VVC standard and should be used to generate the results of the bit rate and objective quality of the VVC encoder. The VTM should be also used to generate results for the validation of the proposed dedicated hardware system.

Previous experiments have indicated that the VVC requires a computational effort up to 31 times higher than the H.265/HEVC [11]. But the improvement of the existing tools and the addition of the new ones brought to the VVC the desired improvement in terms of encoding efficiency.

The VVC introduced improvements in all steps in the video encoding process, including the block-partitioning structure. Each frame of the video being encoded is partitioned in blocks before the predictions and transform steps. The initial block partition is called CTU (Coding Tree Unit), and the largest CTU supported by VVC is 128x128. The partitioning stage of the VVC introduced a new block partition called advanced quadtree with multitype tree (QT+MTT), where the initial block is partitioned firstly in a quaternary tree and, over the result of the quaternary split (in the tree leaves), the multitype partition is performed (binary and ternary splits). Fig. 1 depicts all three types of partitions supported by VVC with their variations: quaternary, vertical binary, horizontal binary, vertical ternary, and horizontal ternary splitting. Each leave in this partition tree is called Coding Unit (CU) which is the basic block used by the encoding tools.

The main novelty at the inter prediction is the Affine Motion Estimation (AME), which allows the VVC the capability to predict non-translational movements, such as rotation, zoom in and out, perspective shifts and many others. Another novelty is the Geometric Partitioning Mode (GPM), which allows motion compensation using nonrectangular partitions. Other novel tools also can be highlighted like the extended merge prediction, Adaptive Motion Vector Resolution (AMVR), Symmetric MVD (SMVD), Bi-prediction with CU-level Weights (BCW), Bi-directional Optical Flow (BDOF), among others [5].

The intra prediction novelties include the increased number of prediction modes, where 67 prediction modes are supported, including the DC, Planar and 33 directional

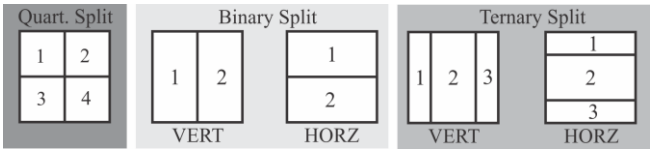

Fig. 1. The quaternary, binary and ternary splitting supported by the VVC.

modes. The directional modes can be extended to 93 angles, by using the tool Wide-angle intra prediction (WAIP). Another improvement is the utilization of more than one line of reference samples from the neighbor blocks, which is called of Multiple Reference Line (MRL) [5]. The Intra Sub-Partition (ISP) mode is also a novelty, which divides the luminance CU into two or four sub-partitions vertically or/and horizontally, depending on the block size. Besides, the VVC also allows the combinations of inter and intra prediction using weighted averaging, which is called Combined inter-/intra-picture prediction (CIIP).

The VVC in-loop filter step also presents some innovations, like the inclusion of the Adaptive Loop Filter (ALF) [13], besides the Deblocking Filter (DBF) and the Sample Adaptive Offset (SAO) presented at H.265/HEVC. The entropy encoder used in VVC is the Context Adaptive Binary Arithmetic Coder (CABAC), as in the H.265/HEVC, but with some innovations, such as improved coefficient coding and high-precision multi-hypothesis probability estimation [5].

The transform step processes the residues that came from the differences between the predicted block (intra and inter prediction) and the original block being encoded. The VVC inherited from the H.265/HEVC standard many tools and structures, like the Transform Unit (TU), the definition of the transform matrices in integer precision, and many others. However, it is possible to find many brand-new innovations in the transform step of VVC. The transform in VVC has support for larger block sizes (e.g. TU size of 64x64) and rectangular blocks (e.g. 32x64, 4x16, etc.). Besides, the VVC divides the transform step into primary and secondary transform, where at the primary transform the traditional DCT-II, DCT-VIII and DST-VII are applied.

The primary transform of the VVC also introduced the Multiple Transform Selection (MTS), and the Sub-Block Transform (SBT) [14]. The MTS allows the primary transform to evaluate different combinations of transforms in horizontal and vertical directions (to compose the 2D transform). Therefore, the transform has seven different combinations of transform to evaluate the residual blocks (e.g. DCT-II x DCT-II, DSTVII x DST-VII, DCT-VIII x DCT-VIII, DCT-VIII x DST-VII, DST-VII x DCT-VIII, DCT-II x DSTVII and DST-VII x DCT-II). The SBT is based on the Spatially Varying Transform (SVT) tool proposed in the H.264/AVC [15], and it is a new subpartition that can be applied over the residual block from the inter prediction, where only a part of the TU. The SBT was developed considering that the residual energy from the inter prediction has a different distribution in the predicted block and, in some cases, the residual energy can be in the sides of the block. In this way, only part of the residual block is encoded while the other part is discarded. The SBT defines four different modes varying the transform shapes, transform sizes, block localization, and used transform types [14].

At the secondary transform, the VVC applies the Low-Frequency Non-Separable Transform (LFNST) [10]. As the LFNST is the focus of this work, it will be detailed in the following subsection.

### *A. The Low-Frequency Non-Separable Transform (LFNST)*

The LFNST is a secondary transform stage applied to process the residual blocks predicted by the intra prediction and which was also processed by the DCT-II x DCT-II in the primary transform. On the encoder side, it is applied between the primary transform and the quantization stages, and on the other hand, on the decoder side, it is applied between the inverse quantization and the inverse primary transform stages. Furthermore, the LFNST is defined with only two configurations to process the primary coefficients: LFNST-4x4 and LFNST-8x8 [14].

The LFNST operates in a different manner when compared with the primary transform, where only part of primary coefficients is used to generate the secondary coefficients. In this way, the LFNST operates over the Region Of Interest (ROI) that corresponds to the top-left low-frequency region of the primary transformed coefficient [10]. The ROI is defined for each primary transform coefficient, independently of the used TU block size (that can be from 4x4 up to 64x64 samples). However, it can vary according to the transform size used and correspond to an area with 8x8 or 4x4 coefficients of primary transform according to LFNST-8x8 and LFNST-4x4 configurations. Hence, it is possible to conclude that the remainder of TU block that does not fit inside the ROI is discarded by the tool.

The LFNST-8x8 is used to process larger TU block sizes (height >= 8 and width >= 8) and the LFNST-4x4 is applied for TU block sizes that have at least height or width equal to four (height=4 and/or width=4) [10]. Fig. 2(a) depicts the ROI of LFNST-8x8 configuration and as can be observed, the LFNST-8x8 is applied over an 8x8 block of primary coefficients but using only 48 coefficients from this block. In other words, the LFNST-8x8 considers only three 4x4 blocks inside its ROI, which corresponds to the three 4x4 blocks “a”, “b”, and “c” in Fig. 2(a). On the other hand, the LFNST-4x4 uses only 16 coefficients from the primary transform, which corresponds to a 4x4 block of coefficients from the primary transform. In this case, the ROI is a single top-left 4x4 block, as the block “a” depicted in Fig. 2(a).

Another difference observed in the secondary transform of the VVC is that it is implemented only with a single stage of matrix multiplication. The LFNST considers mapping the NxM TU to one R one-dimensional vector (vectorization process), where the one-dimensional vector (R) can be of 16 or 48 samples according to the LFNST configuration, and N

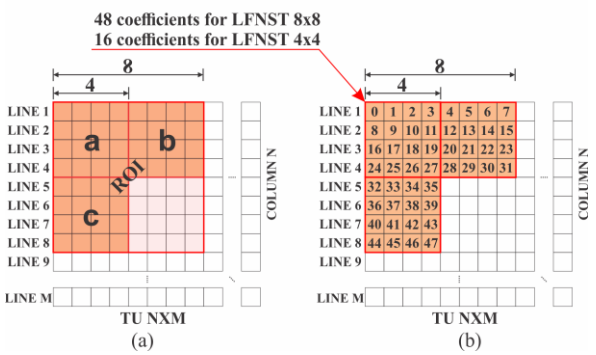


Fig. 2 Illustration of the ROI used in the LFNST (a) and the mapping performed by the LFNST-8x8 over the primary coefficients (b).

and M are the sizes of the TU. Fig 2(b) depicts the mapping performed over the primary coefficients to the LFNST-8x8. Moreover, the one-dimensional vector (R) is multiplied by the transform Kernel (K), where the equation of transformation is calculated as F = K * R. Then, the resultant vector F is reorganized back into a 4x4 or 8x8 block using a scanning order according to the block orientation (horizontal or vertical).

The LFNST defines specialized transform kernel matrices according to intra prediction mode applied and LFNST configuration (4x4 or 8x8). In total, the LFNST defines 16 transform kernels, eight transform kernels for the LFNST-4x4 (with matrices of 16x16 positions) and eight for the LFNST-8x8 (with matrices of 16x48 positions). The transform kernels are arranged into four transform sets for each LFNST configuration and, for each of these four transform sets, there are two transform kernels (two configurations x four sets x two = 16 specialized transform kernels). Table I presents an example of a kernel used in VVC for LFNST-4x4.

As already mentioned, the transform kernel is selected among four LFNST sets accordingly with the lfnstSetIdx parameter used in the VTM (ranging from 0 up to 3), which is a function of the used intra-prediction mode (intraPredMode), as presented in Table II. The VTM uses a special variable to select the transform kernel inside each lfnstSetIdx parameter, which is called *index*. Where the index variable is applied for the whole CU and ranges from 0 up to 2. When the index is 0, then the LFNST is not applied. In other cases, values 1 and 2 select a respective set of transform kernels. In other words, the lfnstSetIdx is used to select one of four sets, where for each set two transform kernels can be assigned, and the index selects the transform kernel (totaling eight transform kernels per each LFNST configuration – 4x4 and 8x8).

The last difference observed in the secondary transform when compared with the primary transform is that only part of the secondary coefficients is encoded and packaged into the bitstream. For instance, if the TU size is 8x8 or 4x4, only the first eight coefficients of the secondary transform are encoded, otherwise, 16 coefficients of the secondary transform are encoded. The LFNST at the decoder side receives only part of the secondary coefficients and reconstructs the block considering the other coefficients as zero.

Table I. Example of LFNST-4x4 Transform Kernel.

| | | | | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| *119* | *-30* | *-22* | *-3* | *-23* | *-2* | *3* | *2* | *-16* | *3* | *6* | *0* | *-3* | *2* | *1* | *0* |
| *-27* | *-101* | *31* | *17* | *-47* | *2* | *22* | *3* | *19* | *30* | *-7* | *-9* | *5* | *3* | *-5* | *-1* |
| *0* | *58* | *22* | *-15* | *-102* | *2* | *38* | *2* | *10* | *-13* | *-5* | *4* | *17* | *-1* | *-9* | *0* |
| *23* | *4* | *66* | *-11* | *22* | *89* | *-2* | *-26* | *13* | *-8* | *-38* | *-1* | *-9* | *-20* | *-2* | *8* |
| *-19* | *-5* | *-89* | *2* | *-26* | *76* | *-11* | *-17* | *20* | *13* | *18* | *-4* | *1* | *-15* | *3* | *5* |
| *-10* | *-1* | *-1* | *6* | *23* | *25* | *87* | *-7* | *-74* | *4* | *39* | *-5* | *0* | *-1* | *-20* | *-1* |
| *-17* | *-28* | *12* | *-8* | *-32* | *14* | *-53* | *-6* | *-68* | *-67* | *17* | *29* | *2* | *6* | *25* | *4* |
| *-4* | *-24* | *-23* | *1* | *17* | *-7* | *52* | *9* | *50* | *-92* | *-15* | *27* | *-15* | *-10* | *-6* | *3* |
| *-6* | *-17* | *-2* | *-111* | *7* | *-17* | *8* | *-42* | *9* | *18* | *16* | *25* | *-4* | *2* | *-1* | *11* |
| *9* | *5* | *35* | *0* | *6* | *21* | *-9* | *34* | *44* | *-3* | *102* | *11* | *-7* | *13* | *11* | *-20* |
| *4* | *-5* | *-5* | *-10* | *15* | *19* | *-2* | *6* | *6* | *-12* | *-13* | *6* | *95* | *69* | *-29* | *-24* |
| *-6* | *-4* | *-9* | *-39* | *1* | *22* | *0* | *102* | *-19* | *19* | *-32* | *30* | *-16* | *-14* | *-8* | *-23* |
| *4* | *-4* | *7* | *8* | *4* | *-13* | *-18* | *5* | *0* | *0* | *21* | *22* | *58* | *-88* | *-54* | *28* |
| *-4* | *-7* | *0* | *-24* | *-7* | *0* | *-25* | *3* | *-3* | *-30* | *8* | *-76* | *-34* | *4* | *-80* | *-26* |
| *0* | *6* | *0* | *30* | *-6* | *1* | *-13* | *-23* | *1* | *20* | *-2* | *80* | *-44* | *37* | *-68* | *1* |
| *0* | *0* | *-1* | *5* | *-1* | *-7* | *1* | *-34* | *-2* | *3* | *-6* | *19* | *5* | *-38* | *11* | *-115* |

Table II. The LFNST set selection.

| ***intraPredMode*** | ***lfnstSetIdx*** |
|---|---|
| *intraPredMode < 0* | 1 |
| *0 <= intraPredMode <= 1* | 0 |
| *2 <= intraPredMode <= 12* | 1 |
| *13 <= intraPredMode <= 23* | 2 |
| *24 <= intraPredMode <= 44* | 3 |
| *45 <= intraPredMode <= 55* | 2 |
| *56 <= intraPredMode <= 80* | 1 |

## III. Related Works

To the best of author's knowledge, there are three related works in the literature proposing architectural solutions for the LFNST of VVC standard, where two of them are previous works of the authors of this paper.

The first related work [16] presents a solution to the whole transform stage of VVC, but it also presents results for the LFNST transform. The work in [16] presents two implementations of LFNST, where the main difference between them is the parallelism (called throughput of two or four samples per cycle). In this way, the architecture versions of two and four samples were implemented with 32 and 64 multipliers, respectively. Besides, both implementations store the LFNST matrices into a ROM with a size of 4kB, which can contribute to better area utilization and power results. The designs in [16] were synthesized to a 28nm TSMC standard-cell library targeting the operation frequency of 600 MHz, which provides the processing of UHD 4K videos at 60fps for the version with the maximum parallelism (four samples per cycle).

The previous work [17] presents a dedicated hardware design for the LFNST that can process UHD 4K videos at 60fps with a frequency of 186.62 MHz synthesized to a 40nm standard-cell library. The design in [17] uses 128 multipliers and receives up to eight samples per clock cycle. The work [18] is the base of development of this work and it also exploits two clock domains with the utilization of 32 multipliers in the main processing unit but exploiting a lower level of parallelism. The proposed design in [18] was synthesized to a 40nm standard-cell library targeting the processing of UHD 4K videos at 60fps.

## IV. LFNST Analyses in the VVC Reference Software

The LFNST was introduced to provide a new stage of transformation over the primary transform coefficients for the residues from the intra prediction. So, to evaluate the coder efficiency impact of the LFNST tool over UHD 4K videos, a set of experiments was performed in the reference software of the VVC, the VTM [12].

The experiments were performed using version 18.0 of VTM and considering the random-access configuration. Six video sequences with UHD 4K resolutions were used in this experiment: Tango2@60fps, FoodMarket4@60fps, DaylightRoad2@60fps, CatRobot@60fps, ParkRunning3@50fps and Campfire@32fps. One second of each video sequence was encoded considering four Quantization

Parameters (QP): 22, 27, 32 and 37. Two executions were done for each set of videos and QPs: one with the LFNST turned on and the other with the LFNST turned off.

The experiment results are presented in Table III. The coding-efficiency metric used is BD-rate, which represents the percentage of variation in the bit rate when comparing videos encoded with two encoders keeping the same objective quality [2]. In that sense, negative results of BD-Rate indicate improvements in coding efficiency. Table III also presented average results for each sequence (considering the four evaluated QPs) of the Total Samples Processed in the Transform step (TSP-T), the Total Samples Processed by the DCT-II with residues from the Intra prediction (TSP-DI), and Total Samples Processed by the LFNST (TSP-L). For TSP-DI and for TSP-L, the values in parentheses are the percentage of the TSP-T value. The average results for all sequences are also presented in the last line of Table III. As one can observe in Table III, the LFNST provides BD-Rate reductions from 0.2% to 1.81%, and an average reduction of 1.23% in comparison with the baseline result (LFNST tool disabled). Those results indicate that the secondary transform stage brings a significant improvement to VVC, despite only one part of the primary coefficients (ROI) being used in the secondary transform to generate the secondary coefficients.

The average TSP-DI and TSP-L represent 10.8% and 2.6% of TSP-T, respectively. Then, the difference observed in the percentage of TSP-DI and TSP-L (10.8% vs. 2.6%) is related to the ROI, where the LFNST uses only up to three top-left 4x4 blocks to perform the transformation and the reminder primary coefficients are discarded. It is important to mention that, even though these BD-rate reductions seem to be small, it is obtained when working only over 2.6% of the total samples processed by the transforms module. Moreover, the LFNST is only one of the many tools introduced by the VVC, which combined can provide those significant coding efficiency gains over the HEVC standard.

Table III. BD-RATE and Total Samples Processed (Billion).

| Video | BD-rate (%) | QP | TSP-T (x10$^9$ samples) | TSP-DI (x10$^9$ samples) | TSP-L (x10$^9$ samples) |
|---|---|---|---|---|---|
| Tango2 | -1.60 | 22 | 1200 | 153 (12.8%) | 36 (3.0%) |
| | | 27 | 580 | 68 (11.7%) | 12 (2.1%) |
| | | 32 | 313 | 35 (11.2%) | 5 (1.5%) |
| | | 37 | 165 | 19 (11.3%) | 2 (1.2%) |
| FoodMarket4 | -1.81 | 22 | 961 | 136 (14.2%) | 20 (2.1%) |
| | | 27 | 449 | 61 (13.6%) | 6 (1.4%) |
| | | 32 | 203 | 27 (13.6%) | 2 (1.0%) |
| | | 37 | 93 | 12 (12.9%) | 1 (0.7%) |
| Campfire | -1.58 | 22 | 806 | 98 (12.1%) | 37 (4.6%) |
| | | 27 | 588 | 74 (12.6%) | 21 (3.6%) |
| | | 32 | 451 | 56 (12.5%) | 13 (2.8%) |
| | | 37 | 317 | 43 (13.4%) | 7 (2.3%) |
| CatRobot | -1.13 | 22 | 671 | 62 (9.2%) | 19 (2.8%) |
| | | 27 | 374 | 31 (8.4%) | 8 (2.1%) |
| | | 32 | 227 | 19 (8.5%) | 4 (1.8%) |
| | | 37 | 135 | 12 (9.2%) | 2 (1.7%) |
| ParkRunning3 | -0.28 | 22 | 1006 | 67 (6.7%) | 29 (2.9%) |
| | | 27 | 657 | 48 (7.4%) | 18 (2.7%) |
| | | 32 | 471 | 37 (7.9%) | 11 (2.3%) |
| | | 37 | 320 | 29 (8.9%) | 7 (2.1%) |
| DayLightRoad2 | -1.01 | 22 | 1096 | 123 (11.2%) | 35 (3.2%) |
| | | 27 | 547 | 54 (9.9%) | 12 (2.3%) |
| | | 32 | 303 | 27 (9.0%) | 5 (1.7%) |
| | | 37 | 166 | 15 (9.0%) | 2 (1.5%) |
| **Average** | **-1.23** | - | **504.1** | **54.5 (10.8%)** | **13.1 (2.6%)** |

## V. Developed LFNST System Design

The developed LFNST System Design is an evolution of the previous work [18], where a new hardware approach is adopted for the enhancement of parallel processing. The proposed LFSNT System Design has the capability to process up to two LFNST-4x4 in parallel. Besides, the LFNST System Design must be compatible with the primary transform, where it must process the primary coefficients at the same rate that it receives the primary coefficients. In other words, System Design must adjust itself to process one line of primary coefficients per clock cycle, since the primary transform is usually designed to process the residual block one line per clock cycle, as can be noticed in many works in the literature like [19] and [20]. However, it is also desirable to provide the primary transform with more flexible processing with parallelism of up to two LFNST-4x4, mainly for small TU sizes (e.g. 4x4). More flexibility (capability to process two LFNST-4x4 in parallel) is required to provide to LFNST system design the parallelism to reach the same throughput when it is processing LFNST 8x8.

The design also must support all 16 transform kernels, arranged between eight kernels of 16x48 matrix defined by the LFNST-8x8 and eight kernels of 16x16 matrix defined by the LFNST-4x4. Besides, the proposed architecture exploits two clock domains, where the clock domain A is responsible to drive the input and output of the System Design and the clock domain B (which operates at a clock four times higher than in clock domain A) is responsible for the processing. Fig. 3 presents a top-level diagram of the developed LFNST System Design which is composed of the LFNST core between the two synchronizers barrier: Input Synchronizer (IS) and Output Synchronizer (OS).

The LFNST Core operates at the clock domain B and receives and sends the samples from/to clock domain A. Consequently, the LFNST Core must keep the same performance as the input interface, performing at a clock that is four times higher than in domain A. In this way, the LFNST Core receives up to eight primary coefficients (one line of an 8x8 block, two lines of 4x4 blocks or one line of one 4x4 block) per clock cycle with more five configuration parameters (C. Inf in Fig. 3): height, width, IsOneLFNST-4x4, lfnstSetIdx, and index. The IS is responsible to send up to eight input samples from clock domain A line by line of primary transform block to clock domain B. Then, the sam-

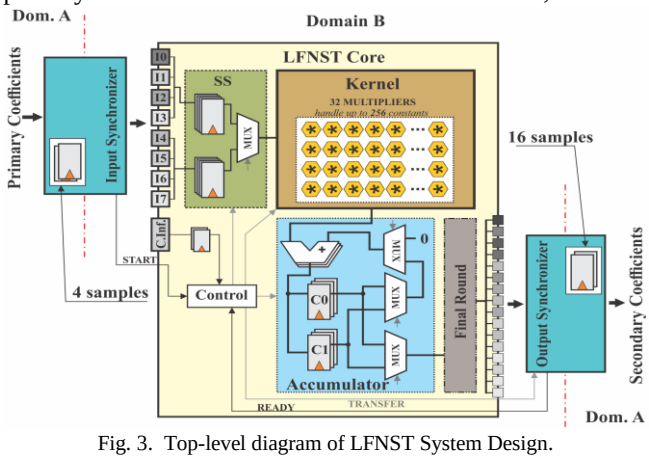


Fig. 3. Top-level diagram of LFNST System Design.

ples are processed by the LFNST Core, working in the clock domain B, and then it makes the secondary transformed coefficients available to the OS.

Therefore, it is important to notice that the design can receive up to eight input samples, but the arrangement of samples can be different according to the LFNST configuration. For instance, if the design is processing in the LFNST-8x8 configuration it receives eight input samples per clock cycle. However, if it is processing in the LFNST-4x4 configuration, two possible behaviors are allowed: (i) it processes only one LFNST-4x4, where it receives only four valid inputs per clock cycle from the primary transform or (ii) it processes two LFNST-4x4 in parallel, where the design receives eight input samples per clock cycle since the first four input samples (I0, I1, I2 and I3) are from the first LFNST-4x4 and the last four input samples (I4, I5, I6 and I7) are from the second LFNST-4x4 block.

On the other hand, the OS has the role of sending back the processed secondary coefficients from clock domain B to clock domain A. In this way, it can send back to clock domain A, those eight processed secondary coefficients if the TU block size is 8x8 or 4x4, otherwise, it can send back 16 transformed secondary coefficients. The OS also was designed to perform two sequential data transferences if the core is processing two LFNST-4x4 in parallel.

Finally, the LFNST System Design supports all those behaviors due to the IS and OS have the capacity to store four samples and 16 samples, respectively. The IS uses this storage capability when it receives only one LFNST-4x4 (e.g. to arrange into packages of eight samples) and the OS when the LFNST Core is delivering two LFNST-4x4 (e.g. to coordinate the output data to the correct sequence order). Subsection IV.C discusses more details about the requirement for additional storage in the IS and OS.

### *A. The LFNST Core*

The LFNST core is organized into five units: Sample Selector (SS), Kernel, Accumulator, Final Round (FR), and Control. The main unit present in the LFNST Core is the Kernel, which has the function of performing the multiplication of the input samples by the LFNST transform kernel, according to the ROI that is being processed. The Kernel is designed to select a subgroup of 4x8 constants and then these constants are multiplied by the input samples. In other words, the Kernel receives only four input samples and delivers eight partially processed samples per clock cycle.

Fig. 4 depicts the intern organization of the Kernel block, where it is composed of 32 multipliers and their respective Constant Generator (CG). The multipliers are arranged into a 4x8 matrix at the MULTIPLIER stage, where each of four input samples is multiplied by its respective column (eight multipliers), resulting in sub-partial results. Those sub-partial results of each line of four multipliers are added at the ADDER TREES stage, then generating eight partial outputs. The CG observed in the MULTIPLIER stage is responsible for the generation of the constant used to perform the multiplication to the input samples according to the ROI and the LFNST kernel. The CG is implemented using a single tree multiplexer controlled by the Control unit and each CG handles up to 256 constants.

The Kernel receives the input samples from the SS unit, which has the role of storing and selecting the input samples according to the used LFNST configuration. Then, the eight partially processed samples processed by the Kernel are stored and accumulated by the Accumulator. The Accumulator was designed to handle the processing of two LFNST-4x4 in parallel, in other words, the Accumulator has the capability to accumulate the two partial multiplier sample results from different LFNST blocks into the C0 and C1 accumulators. Basically, the Kernel performs the multiplication in different cycles for the first four input samples and the last four input samples, where the first partial multiplier result is accumulated into C0, and the last partial multiplier result is accumulated into C1.

The last step in the data processing of the LFNST Core is the FR, which performs the round operation to generate the secondary samples at the output. The LFNST Control coordinates the selection of input samples in the SS, selects the constants for the correct transform kernel to perform the multiplication and controls the Accumulator to accumulate the partially processed samples. Besides, the Control unit also coordinates the transference of secondary coefficients from the clock domain B (LFNST Core) to clock domain A through the Output Synchronizer using the signals TRANSFER (indicates when the transference starts) and READY (indicates when the transference is completed). Therefore, the Control unit also requires the input signal START to indicate when valid input samples are available to be processed.

### *B. Input and Output synchronizers*

The proposed design exploits two clock domains and the Synchronizers are responsible to transfer the samples to the LFNST Core input and output. In other words, the Input Synchronizer (IS) must transfer the samples from the low-frequency domain (A) to the high-frequency domain (B) and the Output Synchronizer (OS) is responsible to transfer the processed samples from a high-frequency domain (B) to the low-frequency domain (A). Besides, the IS and OS are an important part of the design due to critical communication that those synchronizers have over the LFNST Core, which needs to process the one line of the primary coefficients at cycle (clock domain A) to avoid stalls in the primary transform and to send to the output of the core the processed

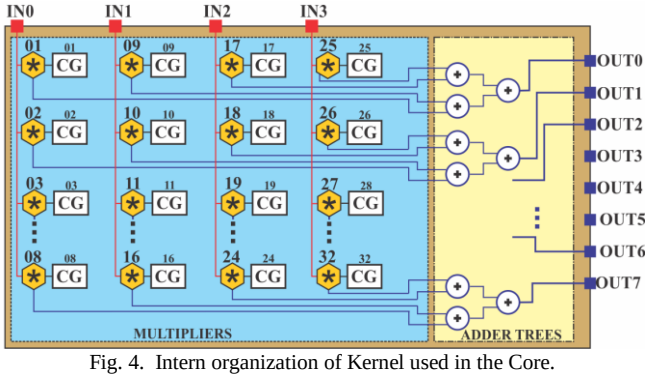

Fig. 4. Intern organization of Kernel used in the Core.

samples before the core finishes the processing of the subsequent ROI (four cycles of clock domain A or 16 cycles clock of clock domain B).

The main concern about the Clock Crossing Domain (CCD) is the register in the destiny since the register can enter a metastable state if the hold and setup times are not respected. In this way, the signal from the origin must be stable at the necessary time for the destiny domain to sample the signal and avoid the metastable state. In the literature, it is possible to find some approaches that mitigate (not eliminate) the occurrence of metastable states, such as synchronizers.

The simplest synchronizer is to register one-bit signal using two or three Flip-Flop (FF) synchronizers [21]. However, this solution is not appropriate for multi-bit signal data. So, the alternative solution is to use a load-trigger synchronizer based on two or three FF synchronizers (one-bit signal enables a load of the register at destiny) or to use a modified FIFO to operate in the CCD [22]. The utilization of synchronizes has some disadvantages like the impossibility, in some cases, to estimate the correct latency for the complete data transaction (vary with a minor change in frequency, the clock edges, variation in the manufacturing process and/or temperature); and the additional registers used by the data path increase the cost in area and power.

Fig. 5 presents the two versions of synchronizers used in the LFNST System Design: a simplified version of a load-trigger synchronizer (the reference control signal is directly transferred from one domain to the other, without feedback) and a load-trigger synchronizer using a complete handshake (the reference control signal is transferred from one domain to the other, but with feedback to the original domain to inform that this transaction was concluded). It is important to observe that both synchronizer versions are implemented with two FF synchronizers. Moreover, Fig. 6 demonstrates the timing diagram of both synchronizers operating together with the LFNST Core (considering a simplified behavior of the LFNST Core to allow this analysis).

For this timing analysis, it is important to consider two statements to better understand the synchronizer operation and to facilitate the estimation of the worst delay observed

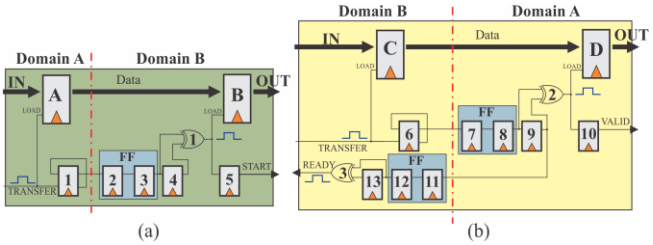


Fig. 5. Synchronizers: (a) simplified load-trigger synchronizer and (b) load-trigger synchronizer with a complete handshake.

in each synchronizer: (i) the positive edge of both clocks will be lagged by half of the fast clock domain to emulate the setup and hold times in the registers and (ii) only the second positive clock edge will generate the valid transition between the domains. The second statement is needed to guarantee that if the first attempt to sample the signal fails, the second sample will occur without any fail.

The first synchronizer version (Fig. 5(a)) is used in the IS. As one can observe in Fig. 6, the synchronizer receives a pulse at the TRANSFER signal to indicate when the data IN will be loaded in register A and to start the transference of data through the clock domains. The transaction will end with the generation of another pulse (after elapsing some cycles of domain B) at XOR_1 which will load the data into the register B and propagate the pulse to the START output signal. With the generation of this signal, the LFNST Core acknowledges that valid data is available in the input to be processed. In this process of data transference through the clock domains, the simplified load-trigger synchronizers require about one cycle of clock domain A (precisely 7/8 clock of clock domain A) to complete a data transaction. In this way, the IS needs only one simplified load-trigger synchronizer because it attends to the requirement of primary transform to deliver the primary coefficients to the core in one clock of domain A.

The second synchronizer version (Fig. 5(b)) is used in the OS and it is implemented with a complete handshake to ensure that the data was safely transmitted to the clock domain A (READY signal generated by the XOR_3). In other words, the LFNST Core requires feedback with the confirmation that the data was transferred. Observing the behavior of this synchronizer in Fig. 6 at first transference (after the

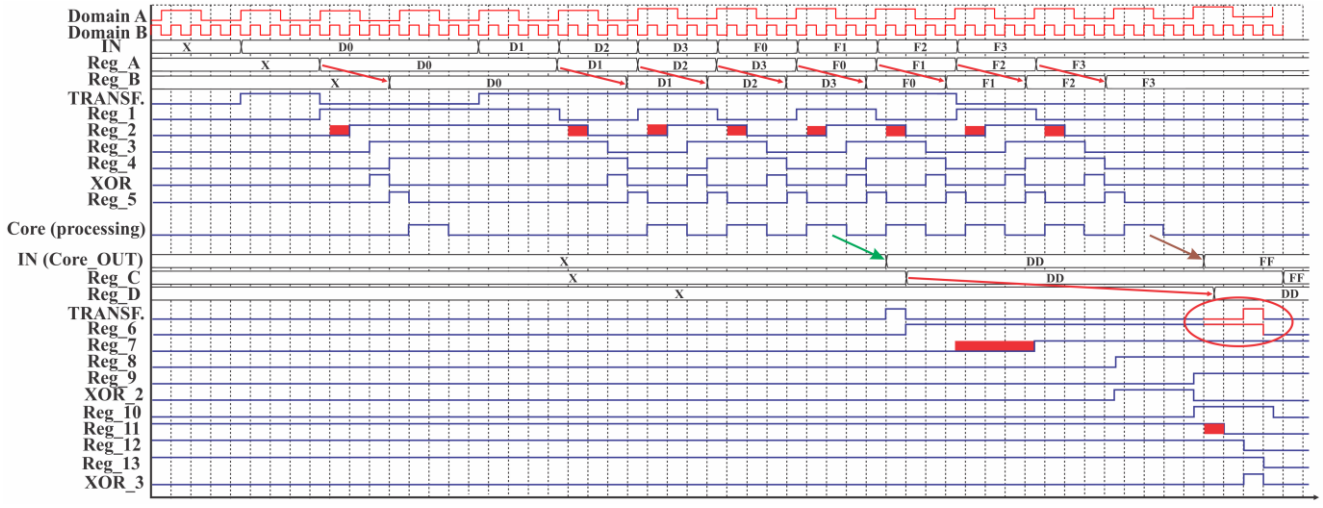


Fig. 6. Timing diagram of both synchronizers operating together with the LFSNT Core.

data is available from the LFNST Core and it is indicated by the green arrow in Fig. 6), the complete data transference will approximately occur at each four clock cycles of domain A and the LFNST Core will receive the transference confirmation (feedback signal) after 18 clock cycles of domain B. It is important to observe that in the timing diagram of the synchronizers (Fig. 6) the red rectangles represent possible metastable states that can happen in the first register in the FF (Reg_2, Reg_7 and Reg_11).

Considering the behavior of this second version of synchronizer in the communication with the LFNST Core, it is possible to observe in Fig. 6 that the READY signal (XOR_3 signal in Fig. 6) is delayed two cycles in domain B (highlighted in red ellipse Fig. 6) when the next processed block is available in the LFNST Core output (indicated by the brown arrow in Fig. 6) to be sent back to domain A (using the signal TRANSFERS). Thus, the utilization of only one synchronizer with a complete handshake in the OS will cause a stall in the primary transform, since the LFNST Core needs to wait for the transference confirmation.

As the LFNST transform must not cause any stall in the primary transform, the OS must have at least two load-trigger synchronizers with a complete handshake. However, the OS also must handle the simultaneous transference of two processed blocks by the LFNST Core (when processing the LFNST-4x4), then the OS must be implemented with four load-trigger synchronizers with a complete handshake, where two are the master used to transfer the processed blocks when the core is not processing the second LFNST-4x4, with two slave synchronizers (additional synchronizers) used when the core is processing the second LFNST-4x4. The slave synchronizers operate in pairs with the master synchronizers.

Finally, it is essential to mention that it is almost impossible to present all possible delays and metastable state occurrences because they can vary by many factors (e.g. frequency, the clock edges, etc.). However, the data transference can occur early as presented and discussed in Fig. 6. Hence, this scenario does not cause any data loss or problem to the LFNST Core, since it can identify these occurrences and can start the processing of block $n$+1 before the next block ($n$+2) arrives at the core.

### *C. LFNST System Temporal Diagram*

The LFNST Core was designed to process one line with eight samples of one ROI in four cycles of the clock domain B. Besides, the whole system must be synchronized with the primary transform and process the samples at the same rate that the primary transform delivers the primary coefficients. Fig. 7 presents a high-level timing diagram of the LFNST System Design considering some hypothetical ROI sequences (8x8, one 4x4, two 4x4, one 4x4, 8x8, and 8x8). It is important to mention that the proposed design can handle the processing of two LFNST-4x4 in parallel, but it also can process only one LFNST-4x4. In other words, the application can configure the LFNST System Design to operate by processing one or two LFNST-4x4.

The first line in Fig. 7 (Prim. T. output) shows the primary transforms outputs, considering the respective TU size and the processing of one line of the same TU or one line of two TUs per cycle. Observing the processing of ROI 8x8, the proposed system can deliver the processed secondary coefficients into 15 cycles after entering the first line of the ROI (clock domain A), where the LFNST Core uses the equivalent of eight cycles to process the primary coefficients (clock domain A). It is important to observe that those reminder seven clock cycles are due to the IS and OS, considering the worst case to transfer the data from one domain to another, as mentioned in subsection IV.B. In other words, the LFNST Core at clock domain A requires only eight clocks to process the ROI 8x8 and four clock cycles to process the ROI 4x4 (for one or two LFNST-4x4).

It is possible to observe in the timing diagram that LFNST Core produces some gaps or stalls (highlighted in red ellipses in Fig. 7) in the pipeline when processing some ROIs. This behavior means that the LFNST Core does not have valid input samples to process, so the ROI process cannot be finished at this time. The main cause of those gaps is related to the difference in the samples available in the ROI per cycle (line) and the parallelism level of the IS, which was designed to deliver eight samples per transfer. There are some cases where the ROI available does not have eight valid primary coefficients in the same cycle. For instance, when the LFNST System Design is processing one ROI 4x4 (one LFNST-4x4) and for the fifth to the eighth line of ROI 8X8. In those cases, the IS uses the additional storage of four samples to arrange the primary coefficients into packages of eight samples to send a complete line (eight samples) to the LFNST Core.

However, the LFNST Core was designed to handle the processing of two ROI 4x4, which can provide the maximum utilization of the Core to process the input samples

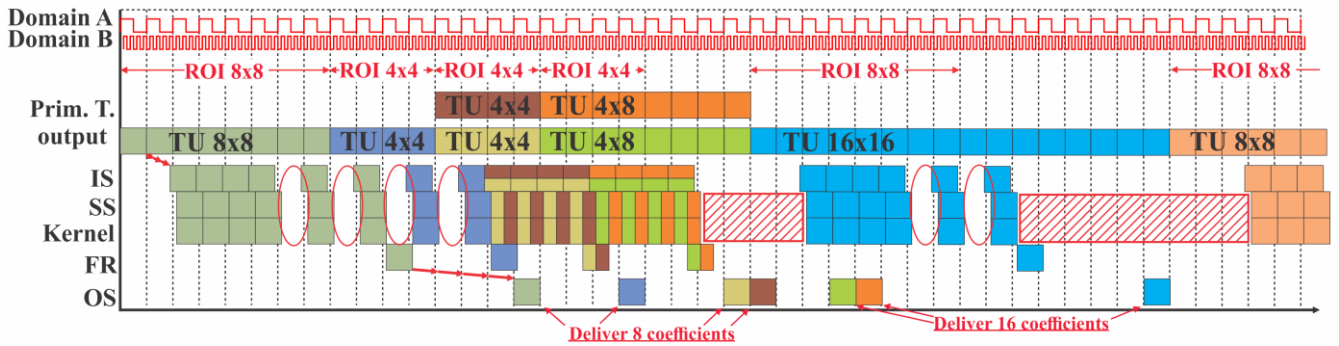


Fig. 7. Timing diagram for the proposed LFNST System Design.

when the Core is processing the LFNST-4x4. Thus, the LFNST Core does not produce any stall when processing two ROI 4x4, as can be noticed when it is processing only one ROI 4x4. Consequently, the LFNST Core generates two processed secondary coefficients packages to be sent to the Core outputs, between two clock cycles (clock domain B). The first package uses the master synchronizer and the second uses the slave synchronizer. In this way, the synchronizers in the OS do not guarantee the correct order sequence (the second package can arrive before the first package, or both packages can arrive at the same time). Then, the OS has the additional storage of 16 samples to store the data from the slave synchronizers and make it available only after the package from the master synchronizers is available at the output of the LFNST System Design.

The proposed design is also conditioned to extra gaps, which happens when the primary transforms are processing TUs sizes different from the 4x4 or 8x8 (out of the ROI). This means that the primary transform has much more data to process than the LFNST and the crosshatched red rectangular in Fig. 7 represents this occurrence in the pipeline. Therefore, this occurrence is observed with constancy and the analyses performed in Section III indicate that on average the primary transform processes 4.16 times more samples than the LFNST.

## VI. Results and Comparison

The proposed LFNST System Design was described in Verilog HDL and validated using the Incisive tool with MATLAB co-simulation. The validation considers the extraction of input vectors from the LFNST function in the VTM software. Those input vectors are used in two moments in the verification/validation of design, to create a MATLAB model at the initial stage and validate the HDL at the final stage. Furthermore, the HDL verification uses the integration of the Incisive tool with MATLAB (co-simulation) to accelerate the validation process (e.g. does not require HDL testbench). The ASIC syntheses were performed using the Genus compiler at version 20.1 and the ASIC synthesis results were obtained with the TSMC 40nm standard-cells library with 0.9V, 25ºC and Low-Voltage Threshold. The gate count considers the number of 2-input NANDs (0.9408 μm2) and the power results were estimated using the default input switching activity (20%).

The operation frequencies were defined considering the processing in the clock domain A and then extended to the clock domain B, which is four times higher than the clock domain A. Besides, the LFNST System is targeting the processing of UHD 4K videos with 4098×2160 (*W* and *H*) pixels with a color subsampling of 4:2:0 (*S*) and considering two different frame rates, 120 and 60 frames per second (fps) (*R*). Equation (1) presents the relation of video parameters (considering *W*, *H*, *S*, and *R*) for the target operation frequency at clock domain A to reach the targeted throughput.

Equation (1) also presents the variable TU_size which is the size of the primary transform TU and the variable *ROI_cycles*, which is the number of cycles required by the system to process this TU size. With the definition of the frequency at clock domain A ($f_{domain\ A}$) it is possible to estimate the frequency at clock domain B ($f_{domain\ B}$), knowing that the frequency at clock domain B is four times higher than the frequency at clock domain A.

$$F_{domain\ A} = (W \times H \times R \times S) / (TU\_size \times ROI\_cycles) \quad (1)$$

The variable *ROI_cycles* is defined to be eight and four cycles to the processing of one LFNST-8x8 and one LFNST-4x4, respectively. However, since the design can process two LFNST-4x4 in parallel, the *ROI_cycles* of LFNST-4x4 are also eight cycles. In this way, the worst-case scenario is when the LFNST System is processing a LFNST-8x8 from a TU size of 8x8, or when it is processing four LFNST-4x4 from four TU sizes of 4x4 (TU sizes in both cases has the same number of samples). It is important to observe that the LFNST System Design can process one or two LFNST-4x4 in parallel. However, the parallelism (one or two ROI 4x4) used in the design can be dynamically configured by the application (encoder or top-level system that uses the LFNST System Design).

Considering the processing scenario of two LFNST-4x4 and the targeted throughput of 4K@120fps applied to Equation 1, the LFNST System Design requires an operation frequency of 186.62MHz for the clock domain A and 746.49MHz for the clock domain B. Moreover, when targeting the process of 4K@60fps, the System requires a frequency of 93.31MHz and 373.24MHz for the clock domains A and B, respectively.

Table IV presents the ASIC synthesis results for the LFNST System Design considering the process of 4K@120fps and 4K@60fps videos. The syntheses results indicate that the whole LFNST System Design has an area utilization of 69.68 Kgates with a power dissipation of 40.46 mW when processing 4K@120fps. In this scenario, the LFNST Core represents about 75.6% (52.71 vs. 69.68 Kgates) of the total area of the LFNST System Design, while in terms of power dissipation, it represents about 84.4% (34.14 vs. 40.46 mW) of the system. On the other hand, the Kernel represents 82.1% of the area and 83.6% of the dissipated power of LFNST Core, respectively. Moreover, both synchronizers (IS + OS) have 21.3% of the total area.

The OS uses about 6.2 times more area than the IS since it is implemented with four synchronizers, each one with a complete handshake that transfers 16 samples in parallel and

Table IV. Synthesis result targeting UHD 4K.

| Architecture | 120fps | | | 60fps | | |
|---|---|---|---|---|---|---|
| | Area (kgates) | Leak. (mW) | Total Power (mW) | Area (kgates) | Leak. (mW) | Total Power (mW) |
| **IS** | 1.95 | 0.05 | 0.21 | 1.95 | 0.05 | 0.13 |
| **OS** | 12.94 | 0.35 | 3.16 | 12.94 | 0.35 | 1.75 |
| **LFNST Core** | 52.71 | 2.98 | 34.14 | 44.47 | 2.34 | 19.39 |
| ***Kernel** | 43.25 | 2.60 | 28.54 | 35.48 | 1.97 | 16.58 |
| **LFNST System Design** | 69.68 | 3.59 | 40.46 | 61.44 | 2.95 | 22.85 |

* The Kernel results are computed in the LFNST Core results.

Table V. Synthesis result comparisons targeting UHD 4K.

| Solutions | Two samples [16] | Four samples [16] | [17] | [18] | LFNST System Design | |
|---|---|---|---|---|---|---|
| | | | | | Target 1 | Target 2 |
| **External Memory** | 4k bytes | 4k bytes | - | - | - | - |
| **Multiplications Supported** | 32 | 64 | 128 | 32 | 32 | 32 |
| **Throughput** | 4k@30 | 4k@60 | 4k@60 | 4k@60 | 4k@120 | 4k@60 |
| **Frequency (MHz)** | 600 | 600 | 189.62 | 186.62 and 746.48* | 186.62 and 746.48* | 93.31 and 373.24* |
| **Area (kgates)** | 74.59 | 109.21 | 99.13 | 57.3 | 69.68 | 61.44 |
| **Technology (nm)** | 28 | 28 | 40 | 40 | 40 | 40 |
| **Total Power (mW)** | - | - | 38.5 | 32.22 | 40.46 | 22.85 |
| **Energy (nJ) (Power/samples)** | - | - | 0.052 | 0.043 | 0.027 | 0.031 |

* Internal clock domains.

has also the capacity to store 16 samples. On the other hand, the IS was implemented only with one simplified synchronizer that transfers eight samples in parallel and has the capacity to store four samples. The LFNST System Design, when running for 4K@60fps has an area reduction of 11.8% (61.44 vs. 69.68 Kgates) and a power reduction of 43.5% (22.85 vs. 40.46 mW) when compared to the results for 4K@120fps.

### *A. Comparison with related works*

Table V presents the compilation of the results of all related works previously presented with the comparison with our LFNST System Design for the two target throughputs. It is important to highlight that the proposed LFNST System Design is the only one to reach the throughput of UHD 4k@120fps, so the discussions will be focused on the results of the UHD 4k@60fps throughput. The related work [16] with a parallelism of four samples per cycle requires 47.7% more area than our work (109.21 vs. 61.44 Kgates) targeting the same throughput (4K@60fps). It is important to highlight again that the related work [16] needs an additional 4kB of internal ROM besides its total logic area. Furthermore, the related work [16] did not present power dissipation results.

Comparing this work against our previous work [17], it is possible to observe that [17] uses an area of 99.13 Kgates, which is about 38.01% higher than our design targeting UHD 4K@60fps. When comparing this work with our second related work [18], this design targeting UHD 4K@60fps presents an area utilization of 7.2% higher. However, our proposed LFNST System Design presents the lowest power dissipation (29.1% lower than [18]) when considering the same throughput of UHD 4K@60fps.

When considering the energy (power/samples processed) of each solution, our previous work [18] demands 0.043nJ per sample processed, while the proposed LFNST System Design demand only 0.031nJ. It means that our proposed System can reduce about 29.1% (0.043nJ to 0.031nJ) of the energy to process each sample at the LFNST transform. It can also be noticed that our solution can be even more efficient when working at the target throughput of 4K@120fps, where only 0.027nJ per sample are consumed, which represents a reduction of 11.5% in comparison with our results for the target throughput of 4K@60fps, and 37.2% when compared to our previews work [18].

## VII. CONCLUSION

This paper presented a dedicated hardware design for the secondary transform of the VVC standard. The proposed LFNST System Design exploits two clock domains using load-trigger synchronizers and can process up to two LFNST-4x4 in parallel. The synthesis results indicate that the proposed solution can reach the processing of UHD 4K at 120 frames per second with an area utilization of 69.68 Kgates and a power dissipation of 40.46 mW. In addition, the proposed design is the most energy efficient among the related works, presenting the lowest energy consumption per sample (0.027nJ).

As future work is planned the implementation of the whole VVC transform stage, combining the primary transform with the developed LFNST System Design. A physical synthesis of this system is also planned, to allow more accurate synchronizers latency inside the complete transform stage and also to extract even more precise results for the area and power/energy.

## ACKNOWLEDGEMENTS

This work is partly financed by the Coordenação de Aperfeiçoamento de Pessoal de Nível Superior - Brasil (CAPES) Finance Code 001. It was also supported by the Conselho Nacional de Pesquisas (CNPq) and Fundação de Amparo à Pesquisa do Estado do Rio Grande do Sul (FAPERGS) Brazilian agencies, and by the Brazilian Microelectronics Society (SBMicro) through the Programa de Apoio a Projeto de Circuitos Integrados em Universidades (APCI).

## REFERENCES

[1] T. Wiegand, G. J. Sullivan, G. Bjontegaard, and A. Luthra, "Overview of the H.264/AVC video coding standard," *IEEE Trans. Circuits Syst. Video Technol.*, vol. 13, no. 7, pp. 560–576, 2003, doi: 10.1109/TCSVT.2003.815165.

[2] Y. F. Shen, J. T. Li, Z. M. Zhu, and Y. D. Zhang, High efficiency video coding, vol. 36, no. 11. 2013. doi: 10.3724/SP.J.1016.2013.02340.

[3] D. Mukherjee et al., "A Technical Overview of VP9—The Latest Open-Source Video Codec," *SMPTE Motion Imaging Journal*, vol. 124, no. 1. pp. 44–54, 2015. doi: 10.5594/j18499.

[4] J. Han et al., "A Technical Overview of AV1," Proc. IEEE, pp. 1–28, 2021, doi: 10.1109/JPROC.2021.3058584.

[5] B. Bross, J. Chen, J. R. Ohm, G. J. Sullivan, and Y. K. Wang, “Developments in International Video Coding Standardization After AVC, With an Overview of Versatile Video Coding (VVC),” Proc. IEEE, no. July, pp. 1–31, 2021, doi: 10.1109/JPROC.2020.3043399.

[6] INTERNATIONAL TELECOMMUNICATION UNION, “ITU-T H.263 : Video coding for low bit rate communication,” ITU-T SG 16, 2005.

[7] INTERNATIONAL TELECOMMUNICATION UNION, “Recommendation ITU-T H.264: Advanced Video Coding,” Int. Telecommun. Union, 2002.

[8] G. J. Sullivan, J. R. Ohm, W. J. Han, and T. Wiegand, “Overview of the high efficiency video coding (HEVC) standard,” *IEEE Trans. Circuits Syst. Video Technol.*, vol. 22, no. 12, pp. 1649–1668, 2012, doi: 10.1109/TCSVT.2012.2221191.

[9] WebM, “VP9 Bitstream & Decoding Process Specification - v0.6,” 2016. https://www.webmproject.org/vp9/ (accessed Dec. 15, 2022).

[10] M. Koo and S. Kim, “Low Frequency Non - Separable Transform (LFNST),” *2019 Pict. Coding Symp.*, pp. 1–5, 2019.

[11] F. Pakdaman, M. A. Adelimanesh, M. Gabbouj, and M. R. Hashemi, “Complexity Analysis Of Next-Generation VVC Encoding And Decoding,” in *2020 IEEE International Conference on Image Processing (ICIP)*, 2020, pp. 3134–3138. doi: 10.1109/ICIP40778.2020.9190983.

[12] “jvet / VVCSoftware_VTM · GitLab.” https://vcgit.hhi.fraunhofer.de/jvet/VVCSoftware_VTM (accessed Nov. 07, 2022).

[13] M. Karczewicz et al., “VVC In-Loop Filters,” *IEEE Trans. Circuits Syst. Video Technol.*, vol. 31, no. 10, pp. 3907–3925, 2021, doi: 10.1109/TCSVT.2021.3072297.

[14] X. Zhao et al., “Transform Coding in the VVC Standard,” *IEEE Trans. Circuits Syst. Video Technol.*, pp. 1–1, 2021, doi: 10.1109/tcsvt.2021.3087706.

[15] C. Zhang, K. Ugur, J. Lainema, A. Hallapuro, and M. Gabbouj, “Video coding using spatially varying transform,” *IEEE Trans. Circuits Syst. Video Technol.*, vol. 21, no. 2, pp. 127–140, 2011, doi: 10.1109/TCSVT.2011.2105595.

[16] I. Farhat, W. Hamidouche, A. Grill, D. Ménard, and O. Déforges, “Lightweight Hardware Transform Design for the Versatile Video Coding 4K ASIC Decoders,” *IEEE Trans. Consum. Electron.*, vol. 67, no. 4, pp. 329–340, 2021, doi: 10.1109/TCE.2021.3126549.

[17] J. Goebel, V. Costa, L. Agostini, B. Zatt, and M. Porto, “A High-Throughput Design for the H.266/VVC Low-Frequency Non-Separable Transform,” in *2022 IEEE International Symposium on Circuits and Systems (ISCAS)*, 2022, pp. 1798–1802. doi: 10.1109/ISCAS48785.2022.9937597.

[18] J. Goebel, L. Agostini, B. Zatt, and M. Porto, “Low-Frequency Non-Separable Transform Hardware System Design for the VVC Encoder,” in *2022 35th SBC/SBMicro/IEEE/ACM Symposium on Integrated Circuits and Systems Design (SBCCI)*, 2022, pp. 1–6. doi: 10.1109/SBCCI55532.2022.9893228.

[19] L. A. Braatz, A. C. S. Beck, B. Zatt, L. V. Agostini, D. M. Palomino, and M. S. Porto, “A new hardware friendly 2D-DCT HEVC compliant algorithm and its high throughput and low power hardware design,” *2019 26th IEEE Int. Conf. Electron. Circuits Syst. ICECS 2019*, no. 1, pp. 654–657, 2019, doi: 10.1109/ICECS46596.2019.8965063.

[20] Y. Fan, Y. Zeng, H. Sun, J. Katto, and X. Zeng, “A Pipelined 2D Transform Architecture Supporting Mixed Block Sizes for the VVC Standard,” *IEEE Trans. Circuits Syst. Video Technol.*, vol. 30, no. 9, pp. 3289–3295, 2020, doi: 10.1109/TCSVT.2019.2934752.

[21] C. E. Cummings, “Clock Domain Crossing ( CDC ) Design & Verification Techniques Using SystemVerilog,” no. Cdc, pp. 1–56, 2008.

[22] A. B. Chong, “Clock Domain Crossing Verification Challenges,” *in 2021 2nd International Conference on Electronics, Communications and Information Technology (CECIT)*, 2021, pp. 383–387. doi: 10.1109/CECIT53797.2021.00075.