Forward-Inverse 2D Hardware Implementation of Approximate Transform Core for the VVC Standard

Ahmed Kammoun, Wassim Hamidouche, Pierrick Philippe, Olivier Déforges, Fatma Belghith, Nouri Masmoudi, Jean-François Nezan

Forward-Inverse 2D Hardware Implementation of Approximate Transform Core for the VVC Standard

Ahmed Kammoun, Wassim Hamidouche, Pierrick Philippe, Olivier Déforges, Fatma Belghith, Nouri Masmoudi, and Jean-François Nezan,

***Abstract*—The future video coding standard named Versatile Video Coding (VVC) is expected by the end of 2020. VVC will enable better coding efficiency than the current High Efficiency Video Coding (HEVC) standard. This coding gain is brought by several coding tools. The Multiple Transform Selection (MTS) is one of the key coding tools that have been introduced in VVC. The MTS concept relies on three transform types including Discrete Cosine Transform (DCT)-II, Discrete Sine Transform (DST)-VII and DCT-VIII. Unlike the DCT-II that has fast computing algorithms, the DST-VII and DCT-VIII rely on more complex matrix multiplication.**

In this paper an approximation approach is proposed to reduce the computational cost of the DST-VII and DCT-VIII. The approximation consists in applying adjustment stages, based on sparse block-band matrices, to a variant of DCT-II family mainly DCT-II and its inverse. Genetic algorithm is used to derive the optimal coefficients of the adjustment matrices. Moreover, an efficient hardware implementation of the forward and inverse approximate transform module is proposed. The architecture design includes a pipelined and reconfigurable forward-inverse DCT-II core transform as it is the main core for DST-VII and DCT-VIII computations. The proposed 32-point 1D architecture including low cost adjustment stages allows the processing of a video in 2K and 4K resolutions at 1095 and 273 frames per second, respectively. A unified 2D implementation of forward-inverse DCT-II, approximate DST-VII and DCT-VIII is also presented. The synthesis results show that the design is able to sustain a video in 2K and 4K resolutions at 386 and 96 frames per second, respectively, while using only 12% of Alms, 22% of registers and 30% of DSP blocks of the Arria10 SoC platform.

***Index Terms*—Versatile Video Coding, Hardware implementation, Approximation, DCT-II, Adjustment stages, FPGA.**

I. Introduction

The increasing demand on video contents coupled with the emerging video formats including 4K, 8K resolutions, High Frame Rate (HFR), High Dynamic Range (HDR) and omnidirectional videos considerably contribute to increase the traffic over Internet. Recent study conducted by Cisco in [1] has predicted that video traffic will increase from 61% of the global IP traffic in 2016 to 82% in 2021. This brings new challenges to video compression community to further enhance the coding efficiency of the High Efficiency Video Coding (HEVC) video coding standard. The Joint Video Experts Team (JVET), established by Motion Picture Experts Group (MPEG) and Video Coding Experts Group (VCEG) [2], has been developing the next generation video coding standard called Versatile Video Coding (VVC). VVC standard, expected by the end of 2020, introduces several new coding tools enabling up to $\sim 30\%$ [3] of coding gain beyond HEVC [4]. However, this coding gain comes at the expense of additional coding and decoding complexities estimated in Random Access (RA) configuration to 800% and 170% [5], respectively. These complexities increase to 2170% and 179% in All Intra (AI) coding configuration [5]. Multiple Transform Selection (MTS) is one of the new concepts introduced in VVC [6]. The earlier version of the MTS, integrated in the Joint Exploration Model (JEM), consists of five transform kernels including DCT types II, V and VIII, and DST types VII and I [7]. In VVC, the MTS relies only on three trigonometric transforms including DCT-II and VIII, and DST-VII. These latter leverage the most of coding gain achieved in the JEM by the five transform types [8]. The basis functions of DCT-II C_2, DST-VII S_7 and DCT-VIII C_8 are computed by Equations (1), (2) and (3), respectively [9], while their first four basis functions

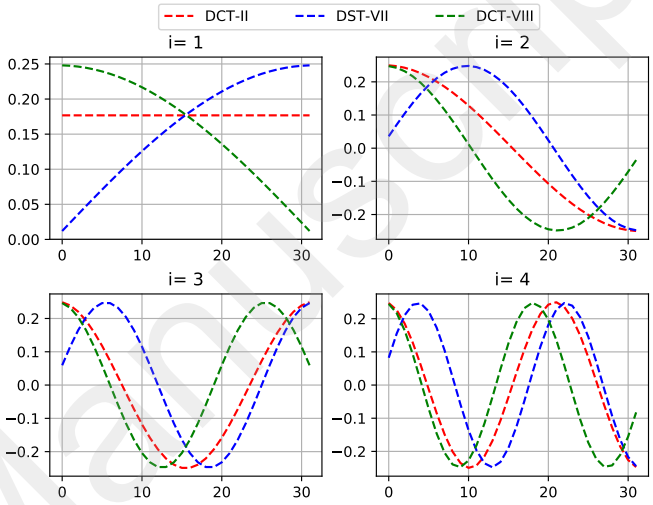

Fig. 1. Illustration of the first four basis functions of DCT-II, DCT-VIII and DST-VII

Ahmed Kammoun, Wassim Hamidouche, Olivier Déforges and Jean-François Nezan are with INSA Rennes, Institute of Electronic and Telecommunication of Rennes (IETR), CNRS - UMR 6164, VAADER team, 20 Avenue des Buttes de Coesmes, 35708 Rennes, France (E-mails: {Ahmed.Kammoun@insa-rennes.fr, Wassim.Hamidouche@insa-rennes.fr, Olivier.Deforges@insa-rennes.fr and Jean-Francois.Nezan@insa-rennes.fr)

Pierrick Philippe is with b<>com, 1219 Avenue des Champs Blancs, 35510 Cesson-Sévigné, (E-mail: pierrick.philippe@orange.com)

Ahmed Kammoun, Fatma Belghith and Nouri Masmoudi are with Univ Sfax, ENIS, Laboratory of Electronics and Information Technology (LETI), LR99ES37, Sfax Tunisia (E-mail: Nouri.Masmoudi@enis.rnu.tn)

Manuscript submitted on 4 June, 2019.

$(i = 1, 2, 3, 4)$ are drown in Fig. 1.

$$C_{2\,i,j} = \gamma_i \sqrt{\frac{2}{N}} \cos\left(\frac{\pi(i-1)(2j-1)}{2N}\right), \quad (1)$$

with $\gamma_i = \begin{cases} \sqrt{\frac{1}{2}} & i = 1, \\ 1 & i \in \{2, \ldots, N\}. \end{cases}$.

$$S_{7\,i,j} = \sqrt{\frac{4}{2N+1}} \sin\left(\frac{\pi(2i-1)j}{2N+1}\right). \quad (2)$$

$$C_{8\,i,j} = \sqrt{\frac{4}{2N+1}} \cos\left(\frac{\pi(2i-1)(2j-1)}{2(2N+1)}\right), \quad (3)$$

with $(i, j) \in \{1, 2, \ldots, N\}^2$ and N is the transform size.

Besides the usual DCT-II used in video coding standards, VVC encoder selects combinations of DCT-VIII and DST-VII, for the horizontal and vertical transforms, to optimize the Rate Distortion (RD) cost J, a trade-off between distortion D and rate R [10]

$$J = D + \lambda R. \quad (4)$$

While the DCT-II has been well studied and optimized with fast implementations [11]–[13], the DST-VII/DCT-VIII do not have efficient fast implementation algorithms [14], [15], and rely on classical matrix multiplications. In this paper we focus on approximating the DST-VII based on the inverse DCT-II and an adjustment band matrix A as initially proposed in [16]

$$\hat{S}_7 = \Gamma \cdot C_2^T \cdot \Lambda \cdot A, \quad (5)$$

where $\hat{S}_7$ is the approximation of S_7, $\Gamma \cdot C_2^T \cdot \Lambda$ is equivalent to the DST-III transform S_3, Λ and Γ matrices are computed by Equations (6) and (7), respectively.

$$\Lambda_{i,j} = \begin{cases} 1, & \textit{if } j = N + 1 - i, \\ 0, & \text{otherwise.} \end{cases} \quad (6)$$

$$\Gamma_{i,j} = \begin{cases} (-1)^{i-1}, & \textit{if } j = i, \\ 0, & \text{otherwise.} \end{cases} \quad (7)$$

The DCT-VIII C_8 can be derived from DST-VII, due to their duality property, without additional complexity involving only permutation Λ and sign change Γ matrices

$$C_8 = \Gamma \cdot S_7 \cdot \Lambda. \quad (8)$$

This paper tackles the problem of hardware implementation of the three transform types used in VVC on the target *Arria 10* Field-Programmable Gate Array (FPGA) platform. The approximation of DST-VII through adjustment band matrix A and inverse DCT-II is first modelled as a constrained integer optimisation problem. The genetic algorithm is then used to solve the problem and compute the adjustment matrices for large transform sizes $N \in \{16, 32\}$. We propose an efficient unified and pipelined hardware architecture for both forward and inverse DCT-II. This latter is used to approximate forward and inverse DST-VII and DCT-VIII along with additional adjustment stage at low computational complexity and logic resource allocation. This architecture supports a reconfigurable 2D implementation of approximate DST-VII and DCT-VIII design that can be integrated in both hardware VVC encoder and decoder. In terms of coding efficiency, the approximate DST-VII and DCT-VIII preserve the coding gain brought by the MTS. On the other hand, the proposed unified hardware architecture enables reaching a high frame rate while using a moderate hardware and logic resource of the *Arria10* FPGA device. It enables processing a video in HD and 4K resolutions at 386 and 96 frames per second (fps), respectively.

The rest of this paper is organized as follows. Section II presents the state-of-the-art of hardware implementations of DCT-II and MTS. The approximation of DST-VII, expressed as a constrained integer optimization problem, is described in Section III. Section IV presents the proposed hardware implementation of the 2D approximate transform design. The experimental and synthesis results of 1D and 2D implementations are presented and discussed in Section V. Finally, Section VI concludes the paper.

II. Related Works

A. Multiple Transform Selection in VVC

The concept of separable transforms competition has been widely investigated for HEVC [17], [18] which considers only DCT-II along with DST-VII for Intra luma blocks of size 4×4 [19], and then integrated in the JEM software [10]. This latter enables five trigonometrical transform types including DCT-II, V and VIII, and DST-I and VII. This concept enables a significant increase in coding efficiency estimated around 3% of bitrate reduction [10]. However, this coding gain comes at the expense of both memory increase, used to store the coefficients of those transforms, and complexity overhead required to test the transform candidates at the encoder side. To cope with the complexity increase, subsets of transform candidates are defined offline, which are tested depending on the prediction configurations such as the Intra prediction mode and the block size.

The MTS concept in VVC defines only three transform types including DCT-II, VIII and DST-VII. As illustrated in Fig. 2, the MTS concept selects, for Luma blocks of size lower than 64, the set of transforms that minimizes the rate distortion cost among five transform sets and the skip configuration. However, only DCT-II is considered for chroma components

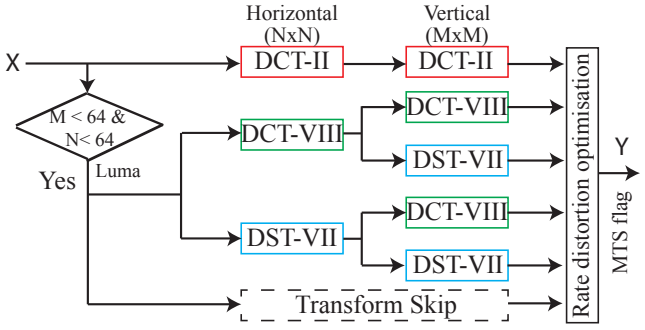

Fig. 2. The concept of 2D separable transforms selection in VVC. X is the input block of residuals, Y is the output transformed block and MTS flag is the index of the selected set of transforms

and Luma blocks of size higher than 32. The MTS solution

brings a significant coding gain of respectively 2% and 0.9% in AI and RA coding configurations [20] compared to single DCT-II transform.

B. Hardware Acceleration of Transforms

1) Hardware Implementation of DCT-II: Several DCT-II hardware implementations have been proposed in the literature, as it is the common transform used in video coding standards: Advanced Video Coding (AVC) and HEVC. Shen *et al*. [21] presented a unified Very Large Scale Integration (VLSI) architecture for 4, 8, 16, and 32 point Inverse Integer Core Transforms (IICT). Regular multipliers and hardware sharing (recursion) are applied to the 16- and 32-point IICTs. To reduce the required hardware resources, the intermediate 1D results are transposed using the Static Random Access Memory (SRAM) module. Work in [22] also leveraged the SRAMs for the 1D transpose process in their proposed 2D pipelined and unified DCT/IDCT/Hadamard architecture with reduced logic resource. Meher *et al.* [19] presented an efficient and reusable architectures for DCT-II implementation supporting different sizes using constant matrix multiplications. This architecture can be pruned to reduce the implementation complexity of both folded and full-parallel 2D DCT-II implementations with only a marginal effect on the coding performance (from 0.8% to 1% Bjøntegaard Delta Rate (BD-BR) loss in coding efficiency when both DCT-II and inverse DCT-II are pruned). Chen *et al.* [23] proposed a 2D hardware implementation of the HEVC DCT transform. The presented reconfigurable architecture supports all block sizes from 4×4 to 32×32. To reduce the logic utilization, this implementation benefits from several hardware resources, such as Digital Signal Processing (DSP) blocks, multipliers and memory blocks. The proposed architecture has been synthesized for various FPGA platforms showing that the design sustains 4Kp30 video encoding with reduced hardware cost. Ahmed *et al.* [24] proposed a dynamic N-point DCT-II hardware implementation for HEVC inverse transform of sizes 4×4, 8×8, 16×16 and 32×32. The hardware architecture is partially folded in order to save the area and improve the speed up of the design. This architecture reaches an operating frequency of 150 MHz which enables supporting real time processing of 1080p30 video.

2) Hardware Implementation of MTS: Recently, several works [25]–[29] have investigated the hardware implementation of the initial version of MTS including the five transform types. Mert *et al.* [25] proposed a 2D implementation including all transform types for 4×4 and 8×8 sizes using adders and shifts instead of multiplications. Although this work presented a 2D hardware implementation of all transform types, it only supports 4×4 and 8×8 block sizes, while the transform of larger block sizes (16×16 and 32×32) are more complex and would require more hardware resources. In [26], Garrido *et al.* proposed a pipelined 1D hardware implementation for all block sizes from 4×4 to 32×32. However, this solution only considers 1D design, while the transform process consists in 2D operations which could normally be more complex. Moreover, this design does not consider asymmetric block size combinations. This work has been then extended in [27] to support 2D design using Dual port RAMs for the transpose memory. They proposed to pipeline the 2D process placing two separate 1D processors in parallel for horizontal and vertical transforms. In [28], Kammoun *et al.* presented a multiplierless implementation of the MTS 4-point transform module. This has been extended to 2D hardware implementation of all block sizes (including rectangular ones), with using the Intellectual Property (IP) Cores multipliers [30] to leverage the DSPs blocks of the *Arria 10* platform [29]. This solution supports all transform types and enables a 2D transform process with efficient pipeline architecture. However, it requires high logic utilization compared to solutions proposed in [25], [26].

3) Approximations of Transform module: Several contributions have been proposed by the JVET to overcome the complexity/resource allocations issues of the MTS [16], [31]–[33]. These solutions have proposed to reduce the computational complexity in number of multiplications per pixel required to process the DST-VII and DCT-VIII. In fact, approximation of transform module is not new in the literature, and was widely investigated for DCT-II [34]–[41]. Jridi *et al* [34] presented a generalized approximation algorithm for the 8-point DCT-II. This solution relies on factorizing the DCT matrix into even-odd decomposition and then replacing the odd part with the even one to further reduce the operation count. The approximate 8-point DCT architecture is used to generate a reconfigurable implementation of larger block sizes based on the same principle. However, the rough approximation of the 8-point core and using it for larger sizes introduce more than 5% coding loss in terms of rate distortion performance (BD-BR). Renda *et al* [35] proposed to approximate the 8-point DCT-II transform by an exact low-complexity factorization of the 8-point DCT-II [42] to be used as core module in the generalized algorithm proposed in [34]. The 8×8 matrix multiplication is reduced to only 5 multiplications and 29 additions. The 8-point scheme is then used to generate larger transform sizes of 16 and 32. This enables a better coding performance compared to the work in [34], but it still achieves a poor rate distortion performance with an average of 4% bitrate loss. Work in [36] proposed a three processing levels to approximate the DCT-II transform. This approach consists in replacing all multiplication operations with shifts and additions, high frequency coefficient filtering and then using inexact additions to compute the DCT-II transform. Work in [37] proposed a DCT-II approximation based on Walsh Hadamard transform (WHT) followed by Givens rotations. Considering a statistical analysis, four DCT-II approximations modes are derived skipping some rotations to reduce its computational complexity at the expanse of bitrate loss up to 7.3%, 5.1% and 9.6% for AI, RA and LD configurations, respectively. Implementation results under 90 nm ASIC enabled high frame rate 8K video processing up to 64 fps. Sun *et al.* [38] proposed an approximate DCT-II design which lies in a combination of truncation schemes of Least Significant Bit (LSB) and Most Significant Bit (MSB). Moreover, quantization results are used to determine the all zero coefficient columns so as their processing is skipped to further reduce the operational count. Hardware implementation

TABLE I
COMPUTATIONAL COMPLEXITY OF DCT-II IMPLEMENTATIONS

Transforms	4-point			8-point			16-point			32-point		
	+	×	>>	+	×	>>	+	×	>>	+	×	>>
DCT-II butterfly [19], [21], [23]	8	4	–	28	20	–	100	84	–	372	340	–
DCT-II [24]	17	0	5	74	0	39	232	0	132	548	0	249
DCT-II [25]	88	0	80	784	0	608	–	–	–	–	–	–
Approximate DCT-II [39]	–	–	–	26	20	–	78	64	–	202	152	–
Approximate DCT-II [40], [41]	8	0	2	24	0	6	64	0	12	160	0	24
Forward multiplication	12	16	–	56	64	–	240	256	–	992	1024	–

results of the approximate DCT-II design show a significant power and area cost reduction with negligible bitrate losses of 0.27%, 0.05% and 0.21% for AI, RA and LD configurations, respectively. Chen *et al.* [39] presented an approximate DCT-II solution supporting block of sizes from 8 to 64. This solution relies on a factorizable structure for both even and approximate odd parts to further reduce its implementation complexity while preserving similar rate distortion performance compared to the original solution. Jridi *et al.* [40] proposed an approximation method of the HEVC-DCT-II that leverages the even-odd butterfly architecture. The matrix coefficients of the even part are replaced by two coefficient values requiring only shift operations to perform the multiplication. The second step of the approximation replaces odd part by the even one in order to reduce the computational complexity of the design, especially benefiting from the recursion property. As a result, all multiplications are removed and larger block sizes implementations are optimized. Work in [41] proposed to approximate the HEVC DCT-II transform design by using a similar approach than the one developed in [40], while the difference lay in odd part coefficients, which are approximated according to their distance with respect to the extremum two values (max and min) of the original transform.

These approximation methods of DCT-II certainly decrease the computational complexity at the expense of some coding loss in terms of image quality (Peak Signal to Noise Ratio (PSNR)) and BD-BR performance. However, unlike the HEVC, MTS involves three transform types. Therefore, in the proposed solution, the DCT-II is not approximated and used instead as the main core to approximate the DST-VII and DCT-VIII. Otherwise, the coding loss would no longer be neglected to preserve the MTS coding gain estimated between 1 to 2% in VVC. The computational complexity, in terms of number of multiplications, additions and shifts of the different DCT-II implementations are summarized in TABLE I for different block sizes $N \in \{4, 8, 16, 32\}$.

III. APPROXIMATION METHOD OF THE MTS TRANSFORMS

A. Problem Formulation

In order to reduce the computational complexity and the resource allocation of the transform block, the approximation approach originally proposed in [16] presents an efficient alternative that approximates several DCT/DST types. It consists in applying adjustment stages of low complexity to DCT-II family transforms. The relations between these DCT-II variants transform matrices are expressed as follow

$$C_3 = C_2^T, \;\; S_2 = \Lambda \cdot C_2 \cdot \Gamma, \;\; S_3 = \Gamma \cdot C_2^T \cdot \Lambda, \qquad (9)$$

where Λ and Γ are defined in Equations (6) and (7) with $N \in \{4, 8, 16, 32\}$.
In fact, Λ and Γ matrices can be interpreted by vector reflection and sign changes, respectively, which are computationally trivial. Using the transforms of Equation (9), different types of DCTs and DSTs can be approximated by applying adjustment stages (pre-processing and post-processing) to the DCT-II family transforms.

In this paper, we focus on the approximation of the DST-VII based on the inverse DCT-II and then DCT-VIII can be derived from the approximate DST-VII $\hat{S}_7$ as expressed in Equation (8). TABLE II gives the DCT-II family used to approximate forward and inverse DST-VII and DCT-VIII.

TABLE II
DCT-II VARIANTS USED TO APPROXIMATE DST-VII AND DCT-VIII

Transform type	DCT-II	DST-VII	DCT-VIII
Forward	DCT-II	DST-III	DCT-III
Inverse	DCT-III	DST-II	DCT-II

B. Approximation through Adjustment Stage

The proposed DST-VII approximation ($\hat{S}_7$) enables reduction of the DST-VII computational complexity since it only involves the DCT-II transform and a multiplication by a band matrix A. Therefore, the complexity of this approximation is equal to the complexity of the DCT-II plus the complexity related to the multiplication by the band matrix A which depends on the maximum number of non-zero coefficients by row θ. The complexity of the multiplication by the matrix A in terms of numbers of multiplications and additions are given by $\theta\, N$ and $(\theta - 1)\, N$, respectively.
The error between the DST-VII S_7 and its approximation $\hat{S}_7$ is expressed by a weighted least-squares error

$$E(A) = \sum_{i=1}^{N} \omega_i \sum_{j=1}^{N} \left(S_{7\,i,j} - \hat{S}_{7\,i,j} \right)^2, \qquad (10)$$

where ω_i, $i \in \{1, \ldots, N\}$ is a weight vector of size N which might account for the relative importance of the frequency components. When the ω_i is constant equal to 1, the error function corresponds to the squared Frobenius norm.

Orthogonality has to be taken in consideration for the adjustment matrix A. This property is required since it enables the use of transpose matrix instead of its inverse to recover the original signal without introducing losses at compression stage. The orthogonality of the adjustment matrix A can be expressed by Equation (11)

$$O(A) = ||A \cdot A^T - I||_F^2, \tag{11}$$

where I is the identity matrix and $||\cdot||_F$ stands for the Frobenius norm. The objective function of this constrained optimization problem is expressed with a Lagrangian multiplier λ as follows

$$\min_A E + \lambda\, O(A). \tag{12}$$

Equation (12) aims to minimize the trade-off between error $E(A)$ and orthogonality $O(A)$ of the approximate transform $\hat{S}_7$. The trade-off between approximation and orthogonality can be tuned by the Lagrangian parameter λ. The second constraint on the adjustment stage is to be sparse block-band matrix, which is easy to compute with small number of taps. The optimal solution of the optimization problem of Equation (12) consists in the $A^\star$ matrix that leads to the the original DST-VII S_7 expressed as follows

$$A^\star = \Lambda \cdot C_2 \cdot \Gamma \cdot S_7, \tag{13}$$

with $E(A^\star)$ and $||A^\star \cdot A^{\star\,T} - I||_2^2$ terms are both equal to zero. We applied Equation (13) to compute $A^\star$ matrix for $N = 8$ with values multiplied by 2^β (with β the bit-depth set to 7 bits) and rounded to the nearest integer $\tilde{A}_{8,8}$

$$\tilde{A}^\star_{8,8} = \begin{pmatrix} 127 & 11 & -6 & 4 & -2 & 2 & -1 & 0 \\ -10 & 125 & 20 & -10 & 6 & -4 & 2 & -1 \\ 6 & -16 & 122 & 30 & -13 & 7 & -4 & 1 \\ -4 & 10 & -22 & 118 & 39 & -15 & 7 & -2 \\ 4 & -8 & 14 & -27 & 114 & 47 & -15 & 4 \\ -3 & 7 & -11 & 18 & -32 & 110 & 53 & -10 \\ 3 & -6 & 10 & -15 & 23 & -38 & 109 & 47 \\ -2 & 4 & -7 & 10 & -15 & 22 & -39 & 118 \end{pmatrix}.$$

However, the A^* solution is not appropriate as it does not provide integer values, as required for video coding applications, and does not reveal a sparse property, leading to fewer arithmetic operations. $\tilde{A}^\star_{8,8}$ has its most significant absolute values around the diagonal and lower absolute values are located at lower-left and upper-right parts of the matrix. This property of the adjustment matrix A is stronger for adjustment matrices of higher sizes $N \in \{16, 32\}$.

In this paper, adjustment band matrix that minimizes the trade-off between error and orthogonality is sought with the constraint of A to include few integer values different from zero. This discrete constrained optimization problem is expressed as follows

$$\begin{aligned} \underset{A}{\text{minimize}} \quad & E(A) + \lambda\, O(A), \\ \text{subject to} \quad & A_{i,j} = 0, \quad \forall\, j > i + \lfloor \theta/2 \rfloor, \\ & A_{i,j} = 0, \quad \forall\, j \leq i - \lceil \theta/2 \rceil, \\ & i, j \in \{1, \ldots, N\}^2, \\ & A_{i,j} \in \mathbb{Z} \cap [\, -2^\beta + 1,\, 2^\beta \,], \\ & \lambda \in \mathbb{R}^+. \end{aligned} \tag{14}$$

It has been shown in [43] that the DST-VII is optimal in terms of energy packing for image intra-predicted residuals. Indeed, those residuals have an auto-correlation matrix which is tri-diagonal matrix R_x of size $N \times N$ expressed by Equation (15).

$$R_{x\,i,i} = b,\ R_{x\,i,i+1} = c,\ R_{x\,j-1,j} = a,\ R_{x\,N,N} = b - \alpha, \tag{15}$$

with $(a, b, c, \alpha) = (-1, 2, -1, 1)$ and $1 \leq i < N$, $1 < j \leq N$ The eigen-vectors of the matrix R_x are the basis of the DST-VII transform [9]. Therefore, for the approximation of the DST-VII, we propose to weight the relative importance of the approximation basis with the eigen-values of the of the auto-correlation matrix R_x. This gives more importance to the lower frequency range where an important part of the signal energy stands. According to [44] the eigen-values are computed as follows

$$\omega_i = b + 2\sqrt{a\,c}\,\cos\left(\frac{2\,i\,\pi}{2N+1}\right),\ \ i = 1, \ldots, N. \tag{16}$$

C. Genetic Search Algorithm

To provide an approximation of the DST-VII, the adjustment matrix, which consists of a selected number θ of integer values around the diagonal, need to be determined for a desired level of orthogonality $O(A)$ expressed in Equation (12). To solve this problem in the integer domain, continuous optimization methods such as gradient descent are not appropriate. Also, an exhaustive search would result in evaluating $\left(2^{\beta+1}+1\right)^{\theta\,N}$ combinations (β is the bit-depth set to 7 bits). Techniques such Integer Programming [45] can provide helpful techniques in that context. However, in this study, a genetic algorithm approach was preferred as it provided satisfactory results and appeared to converge well.

Genetic algorithms, are easily re-configurable to address various scenarios such that the adjustment matrix with different number of coefficients per row. Indeed, this optimization algorithm solves Equation (12) with $\theta\,N$ parameters with the same strategy. Basically, it consists in changing individual elements of the adjustment matrix in the *mutation* process. Although convergence is not guaranteed with the Genetic Algorithm approach, it appears in practice that it converges in a consistent fashion with different initialization points.
The principle of the genetic search is the following:

- From a set of N_p selected adjustment matrices, called parents, N_c children are created by individual changes in the close-to-diagonal values. One among the children's values, randomly selected, is changed by the addition of +/-1 while ensuring that the value remain in the adjustment matrix bit-depth range.
- The resulting $N_p\,N_c$ adjustment candidate matrices are evaluated with Equation (12), this can be done in parallel, e.g. using OpenMP programming interface [46].
- From the candidate matrices, $N_p - 1$ are randomly retained, and the best performing matrix that minimizes the trade-off between error and orthogonality is kept. From these N_p matrices the three steps are re-iterated until convergence of the algorithm.

As the A matrices have coefficients around the diagonal, the number of parameters depends on the the matrix size and the number of coefficients per row θ. It is in the range of $\theta\, N$, each coefficient is to be expressed on β bits to allow implementation on fixed-point devices. The convergence is measured in terms of stabilization of the algorithm, when there is no further reduction of the optimized metric after many iterations. The λ value is modified in order to provide different solutions in the approximation / orthogonality space. It is essential in video coding to provide transforms with a reconstruction level, sufficiently low, to avoid the introduction of distortion in the transform process. Subsequently, λ needs to be chosen in a way that the orthogonality measure $O(A)$ is in the range of -60 dB, the same orthogonality level than the discrete DCT-II used in VVC.

For this work, coefficients of 5 tap sparse block-band adjustment matrices $\theta = 5$ are used with an additional constraint of symmetry across the diagonal between non-zero coefficients

$$\begin{aligned} A_{j,i} &= -A_{i,j},\ \forall\, j = i+1,\ \ i \in \{1, 2, \ldots, N-1\}, \\ A_{j,i} &= A_{i,j},\ \forall\, j = i+2,\ \ i \in \{1, 2, \ldots, N-2\}. \end{aligned} \tag{17}$$

The value $\theta = 5$ is selected since it achieves a good trade-off between complexity and approximation of the original DST-VII transform. The symmetric property reduces the memory storage of the adjustment matrix, and enables a faster convergence of the Genetic algorithm.

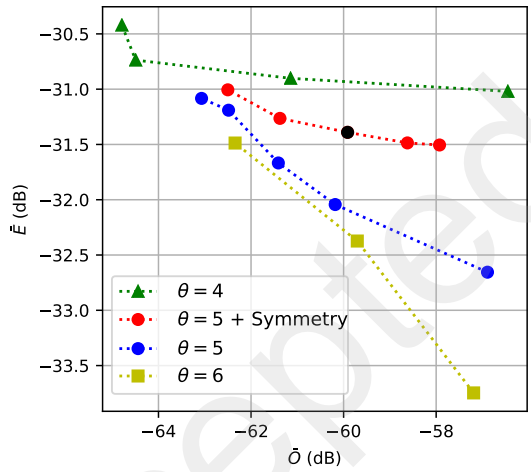

Fig. 3. Performance of approximate DST-VII transform $N = 32$.

Figure 3 illustrates the error and orthogonality performance of the proposed solution for different θ values. The configuration highlighted in black, enabling the desired level of orthogonality around -60 dB and symmetry of coefficients with $\theta = 5$, is selected for hardware implementation and its coding performance is assessed under the VVC Test Model (VTM) 3.0 software.

IV. 2D Hardware Implementation of Transform Module

As expressed in Equation (9) giving the relations between DCT-II types, the approximations of forward and inverse DST-VII are performed by applying adjustment stages to inverse DCT-II C_2^T and the forward DCT-II C_2, respectively. In the following we detail the implementation of the main DCT-II forward and inverse transforms, which are then used to approximate 2D forward and inverse DST-VII and DCT-VIII transforms.

A. Unified Forward and Inverse DCT-II Core Transform

In this section the C_2^N corresponds to the N-point DCT-II matrix with $N \in \{4, 8, 16, 32\}$. The DCT-II and IDCT-II N-point kernels are computed by Equations (18) and (19), respectively

$$C_2^N = P^N \cdot \begin{pmatrix} C_2^{N/2} & 0 \\ 0 & O^{N/2} \end{pmatrix} \cdot \begin{pmatrix} I^{N/2} & J^{N/2} \\ -J^{N/2} & I^{N/2} \end{pmatrix}, \tag{18}$$

$$[C_2^N]^T = \begin{pmatrix} I^{N/2} & -J^{N/2} \\ J^{N/2} & I^{N/2} \end{pmatrix} \cdot \begin{pmatrix} [C_2^{N/2}]^T & 0 \\ 0 & O'^{N/2} \end{pmatrix} \cdot P^N, \tag{19}$$

where P^N is a permutation matrix to reorder the output data in appropriate form, $C_2^{N/2}$ is the DCT-2 of size $N/2$, $O^{N/2}$ is a matrix of size $N/2{\times}N/2$ consisting of odd rows of the first $N/2$ columns of the C_2^N matrix. $I^{N/2}$ and $J^{N/2}$ are, respectively, the identity and the cross-identity (reflection) matrices of size $N/2{\times}N/2$. Finally, $O'^{N/2}$ is a matrix of size $N/2{\times}N/2$ consisting of odd rows of the first $N/2$ columns of the $[C_2^N]^T$ matrix.

Comparing $O^{N/2}$ and $O'^{N/2}$, we notice that for i from 1 to $N/2$, $O^{N/2}$ i^{th} column has the same coefficients than the $N/2$-i^{th} column of $O'^{N/2}$ but in inverse order. Subsequently, $O'^{N/2}$ can be implemented using the same architecture than $O^{N/2}$. This can be achieved with computationally trivial steps, by inverting the inputs and outputs orders. As a result, a unified architecture design is proposed to embed forward and inverse DCT-II sharing the same $N{\times}N$ odd part of the C_2^N matrix, which is the most complex part.

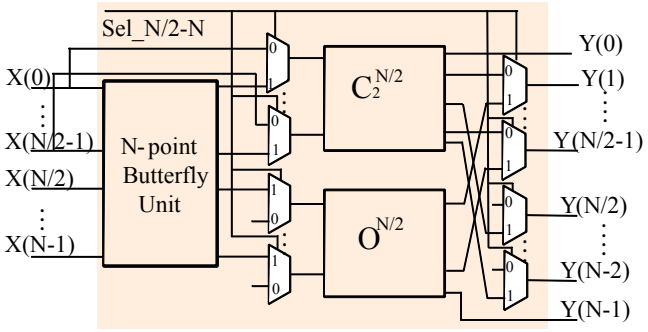

Fig. 4. Proposed architecture of recursive C_2^N implementation with N=8, 16 and 32.

Therefore, benefiting from recursion property as presented in Fig. 4, the same principle is applied for lower block sizes to deepen the hardware sharing in the unified circuit.In terms of required number of operations, the state of the art 32-point butterfly forward and inverse DCT-II implementation requires 680 multiplication operations according to TABLE I for both DCT-II and Inverse DCT-II. The proposed architecture of the unified DCT-II and IDCT-II requires only 344 multiplication

TABLE III
COMPUTATIONAL COMPLEXITY OF THE PROPOSED FORWARD AND INVERSE DCT-II AND APPROXIMATE DCT-VIII AND DST-VII IMPLEMENTATIONS

Transforms	4-point			8-point			16-point			32-point		
	+	×	>>	+	×	>>	+	×	>>	+	×	>>
DCT-II butterfly [19], [21], [23]	16	8	–	56	40	––	200	168	––	744	680	––
Forward matrix multiplication	24	32	––	112	128	––	480	512	––	1984	2048	––
Proposed DCT-II	16	8	––	36	24	––	92	88	––	332	344	––
Proposed Approx DST-VII	24	32	––	112	128	––	51	58	9	112	114	30

operations: 256 (odd part) plus 88 (even part) multiplication operations of C_2^{16}/$[C_2^{16}]^T$).

Recursion property and reusing the same architecture of different odd-part matrices in a unified DCT-II/IDCT-II scheme enable considerable reduction in logic resource and allow preserving 256, 64 and 16 multiplication operations respectively for 32, 16 and 8-point designs. TABLE III details the computational complexity of the proposed architecture design for different block sizes from 4 to 32 considering forward and inverse processes. Moreover, multiplication operations are performed using the Library of Parametrized Modules (LPM) IP Cores multipliers offered by DSP blocks of the Arria 10 FPGA device. Fig. 6 illustrates the proposed architecture of the unified DCT-II/IDCT-II core transform.

From equations (18) and (19), and benefiting from butterfly decomposition architecture, the difference between DCT-II and IDCT-II is the hierarchical application of the associate butterfly block; as a first or last stage for forward and inverse processes, respectively, depending on *Forward-Inverse* selection signal. For the IDCT $[C_2^{32}]^T$ computation, the 32-odd part is computed as O'^{16}. Trivial pre-processing and post-processing steps on its associated inputs and outputs are applied with no additional computing complexity. The obtained results of the $[C_2^{16}]^T$ implementation (16-point IDCT-II) outputs go through IDCT-II butterfly stage in order to provide the final IDCT-II 32-point outputs.
O'^{16} implementation requires 16 clock cycles where all multiplications are performed at one clock cycle using LPM multipliers, then adder trees (with two addition operations) are placed sequentially to generate the output. The pipeline installed consists in introducing assignment stages. They are based on registers and have basically two roles: storing the current results and transferring the appropriate data and intermediate signals to the next stage. These components are responsible for the pipeline operation avoiding data conflicts or loss which may occur in the next clock cycles as inputs are refreshing [29].

Fig. 5 presents a timing diagram of 1D IDCT-II computation of 32×32 input block. It details the different steps and latency required to generate 1D output results. In the case forward DCT-II (*Forward-Inverse* is equal to 0), the 32-point odd part is computed as O^{16}. Then the obtained results together with the C_2^{16} multiplication (16-point DCT-II) ones form the final outputs of 32-point DCT-II. The design is not only unified for forward and inverse DCT, but also for all block sizes from 4 to 32 through a size dependent selection process.

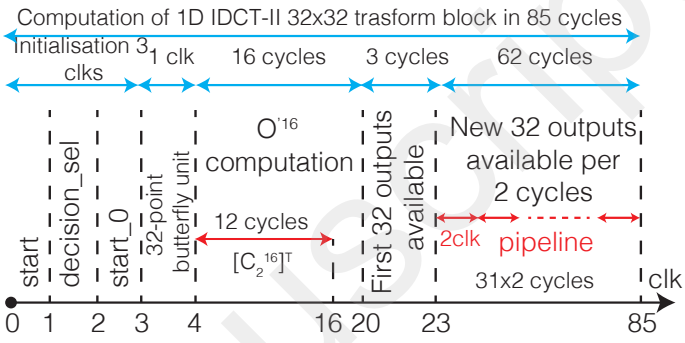

Fig. 5. Timing diagram of 1D IDCT-II 32×32 block computation

B. Hardware Architecture of Adjustment Stages

As explained in Section III-A, the approximation method is based on DCT-II architecture and diagonal-sparse orthogonal adjustment matrices with low computational complexity. These latter are generated using the genetic algorithm detailed in Section III-C.

In this work we consider 16 and 32 approximation orders as they are the most complex cases. 16 and 32-point adjustment matrices of DST-VII are 5 tap sparse block-band matrices. As illustrated in Fig. 7, the adjustment matrices are placed and used as a pre-processing stage in the forward transform process, and a post-processing stage in the inverse one.

With a maximum of 5 coefficients per row, it would require 80 and 160 multiplications for 16 and 32-point orders, respectively. However, it is worth noting that not all adjustment matrix rows include five coefficients, and coefficients with power-of-two values are implemented using shift operations, which would further reduce the number of required multipliers.

The symmetry property of the adjustment matrix A expressed by Equation (17) enables using the same coefficients to perform the multiplication by its inverse A^T in the post processing stage of the inverse DST-VII. Therefore, we can use the same implementation of the adjustment matrices in both forward and inverse transform processes and half of associated computational complexity is preserved. Then, as the approximation approach consists in using the DCT-II architecture, DST-VII implementation requires only the number of operations included by the adjustment matrices implementation over the DCT-II ones. The approximate DCT-VIII $\hat{C}_8$ is obtained easily using approximate DST-VII $\hat{S}_7$ architecture with only some changes in input and output order and signs as expressed in Equation (8). Therefore, the

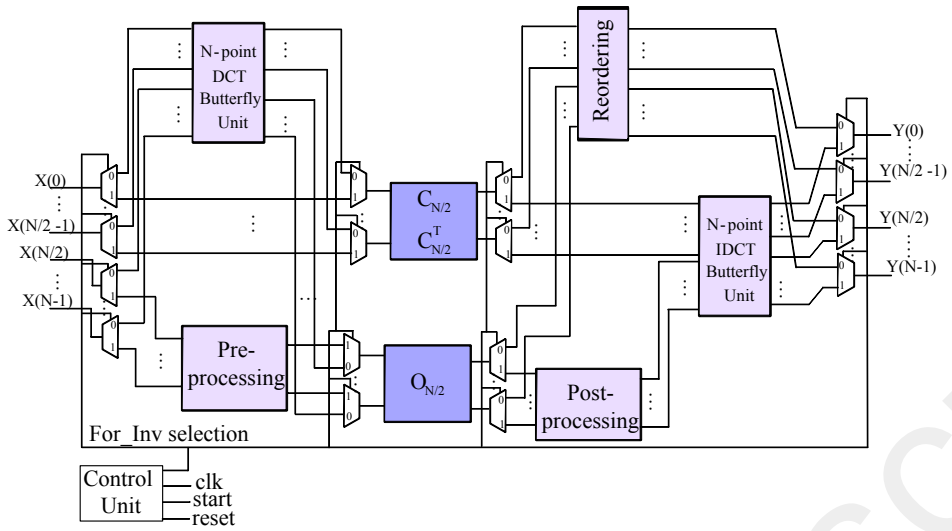

Fig. 6. Proposed architecture of unified 32-point DCT-II and IDCT-II core transforms

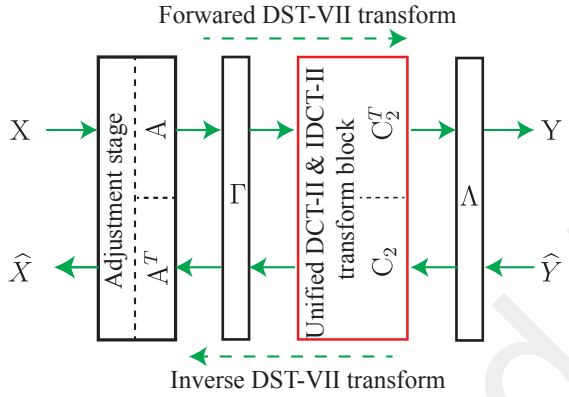

Fig. 7. 1 D approximate DST-VII transform scheme using the pre/post processing stages

approximate DCT-VIII transform requires almost no hardware resource (except one multiplexer) and does not introduce additional computational complexity. TABLE III shows the computational complexity required for the proposed DCT-II, and approximate DCT-VIII and DST-VII implementations for both forward and inverse operations. Approximation through adjustment stages is used for 16 and 32-point orders because they are the most complex cases. For 4 and 8-point DST-VII, straightforward matrix multiplication is used as presented in TABLE III.

C. Proposed 2D Implementation of VVC Transform Approximation

Using property of separable transforms, the 2D process could be computed by the row-column decomposition technique in two distinct stages. First, a 1-dimensional unit is performed for each column of the input matrix to generate the intermediate output. This unified circuit enables the computation of DCT-II, approximate DCT-VIII or DST-VII depending on the selected transform type as illustrated in Fig. 9. Once the first N intermediate 1D outputs are available, they are scaled and stored in N Dual-Port RAM (16×512 i.e 16×32×16) at a column order ($IntermOu\_0\_0$ .. $IntermOu\_0\_31$). Fig. 8 shows the structure of the transposition memory.

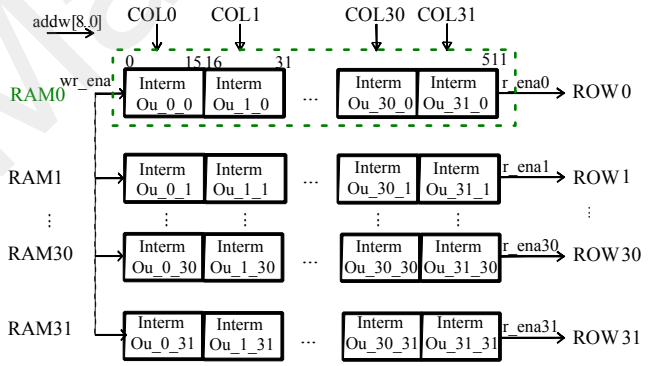

Fig. 8. Architecture of the transposition memory with 32 Dual-Port RAMs. Data can be written at a column basis and read at a row basis

After storing the outputs of the N columns, the first block memory ($RAM0$) will contain the first results computed from each column. For $N = 32$, data of $RAM0$ output port buffer will be the concatenation of all first output values from every processed column (32 values of 16 bits each: $IntermOu\_0\_0$ .. $IntermOu\_31\_0$). The advantage of using dual port RAMs enables reading the 32 values with a single reading signal. The same principle is used to secure the storage process in $RAM1$ to $RAM31$. As a result, considering 32×32 1D intermediate output matrix, assigning consecutive r_ena signals for the 32 RAMs sequentially, leads to automatically transpose the results and fed them as inputs to the same 1D architecture in order to generate the desired 2D output. The proposed 2D circuit is able to efficiently compute approximate DST-VII and DCT-VIII transforms using a unified 1D forward-inverse DCT-II core transform and adjustment stages circuit. Moreover, it is unified for both 16 and 32 block sizes and reconfigurable to perform either forward or inverse transform processes. Input and output

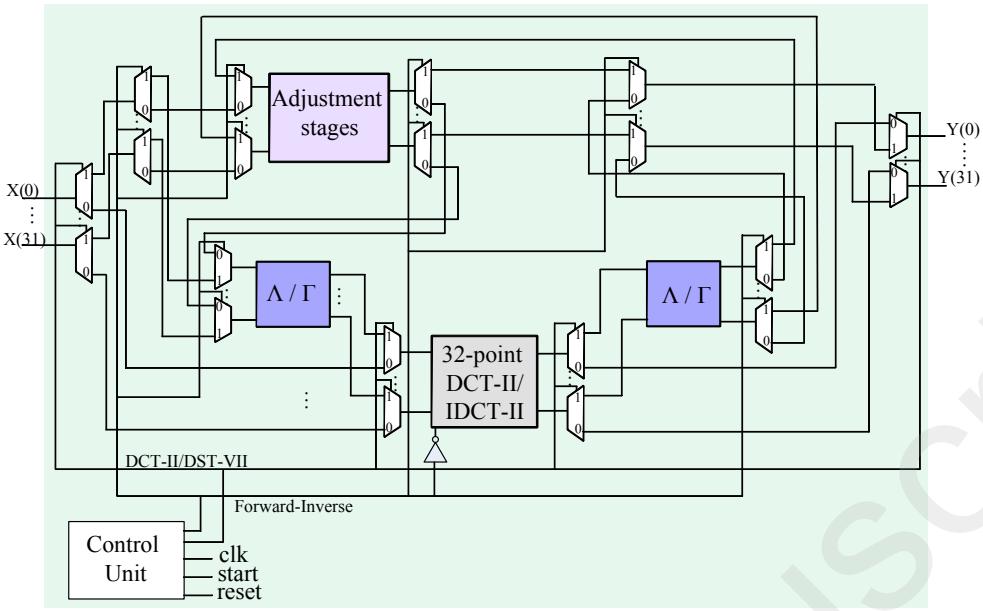

Fig. 9. Proposed unified architecture of approximate forward-inverse transform design

First In First Out (FIFO) memory blocks are added in both ends of the design, each of size 16 Kbits (16×32×16), to store and display input and output vectors. Moreover, a control unit according to a state machine is defined. It enables assigning the appropriate signals and blocks, and control reconfiguration aspects. In addition, it manages the different steps of 2D pipeline process.

V. Experimental and Synthesis Results

A. Experimental setup

The coding and complexity performance of the proposed approximate solution are investigated in this section under the VVC Common Test Condition (CTC). Those experiments are tested among mandatory video classes, where each class corresponds to a specific resolution (up to 4K video) and video content characteristic (with computer generated and visio-conference materials). The proposed approximate DST-VII and DCT-VIII have been integrated in the VTM draft 3.0 reference software [47]. The BD-BR metric is used to assess the coding performance over four bitrates between two coding configurations giving the bitrate gain/loss $(-/+)$ in percentage for similar PSNR quality. The encoder and decoder run times are also drown to assess the complexity of the proposed approximations.
On the other hand, for this work, hardware implementation of the transforms is also done using the Verilog HDL description language. The architectures of 1D and 2D processes of different orders have been tested with state of the art simulation and synthesis software tools [48], [49] under Arria 10 Systems on Chips (SoC) FPGA device [50]. Test bench files were used to validate the output results.

B. Rate Distortion Coding Performance

TABLE IV gives the coding and complexity performance is terms of both BD-BR and run time of the proposed solution. The encoding ($EncT$) and decoding ($DecT$) run times are also compared in percentage to the anchor VTM3.0 [51]. This latter uses the HEVC DCT-II up to size 64 together with DST-VII and DCT-VIII core transforms for MTS, up to size 32, implemented as matrix multiplications. From TABLE IV, it is shown that the proposed approximations of the DST-VII and DCT-VIII introduce slight coding loss of 0.15% in average for the luminance component (Y), and 0.09% for the two chrominance components (U and V) in AI coding configuration. Overall, we can conclude that the coding performance remains similar to the anchor for RA and Low Delay B (LDB) Inter coding configurations.

The encoding and decoding run times slightly decrease with the approximate DST-VII and DCT-VIII in AI configuration, while they remain constant in RA and LDB configurations. These results can only support the effectiveness of the proposed VVC transform approximation method. In fact, the gain in number of multiplications and additions enabled by the approximate transforms through adjustment matrices is low in the context of the VTM software, which includes other time consuming operations. However, this gain in number of operations as well as in memory usage has a significant impact in the context of hardware implementation on FPGA platforms with limited logic and memory resources.

C. Synthesis Results and Discussion

Since MTS approximation is based on DCT-II architecture, TABLE V presents the area consumption of some related works for 1D 32-point forward DCT-II implementation on different platforms. In this paper a unified forward and inverse DCT-II design is proposed. Thus, regarding information given in TABLE V it would consider twice the required hardware cost. In addition, for further fair evaluation, we will focus more on [29] work which provided both DCT-II and DST-VII implementations without approximation on the same FPGA

TABLE IV
PERFORMANCE (%) IN TERMS OF BD-BR AND RUN TIME COMPLEXITY OF APPROXIMATE DST-VII AND DCT-VIII

Classes	All Intra Main 10					Random Access Main 10					Low Delay B Main 10				
	Y	U	V	*EncT*	*DecT*	Y	U	V	*EncT*	*DecT*	Y	U	V	*EncT*	*DecT*
A1	0.15	0.11	0.14	93	80	0.12	0.31	0.27	99	97	–	–	–	–	–
A2	0.22	0.12	0.08	95	84	0.09	0.21	0.23	99	98	–	–	–	–	–
B	0.14	0.14	0.20	94	84	0.10	0.31	0.17	99	98	0.07	-0.22	0.11	100	101
C	0.06	0.00	-0.05	95	89	0.07	-0.06	0.35	99	100	0.06	0.15	0.35	100	100
E	0.18	0.10	0.06	94	86	–	–	–	–	–	0.06	0.81	-0.11	100	97
Average	0.15	0.09	0.09	94	85	0.09	0.19	0.25	99	98	0.06	0.16	0.14	100	100

TABLE V
AREA CONSUMPTION OF SOME 1D 32-POINT BUTTERFLY DCT-II IMPLEMENTATIONS ON DIFFERENT PLATFORMS

Transform	Dimention	Technology	Area consumption
DCT-II [40]	1D forward	Xilinx Sparta	18772 (LUT)
DCT-II [19]	1D forward	TSMC 90nm	253 (Kgate)
DCT-II [29]	1D forward	Arria 10 Soc	11231 (Alm)

target device with similar pipelining approach as used in the proposed work.

TABLE VI provides more detailed hardware synthesis results of 16 and 32-point DCT-II and DST-VII implementations proposed in [29]. Results presented in TABLE VI show that the proposed design provides good performance in terms of processed frames per second up to 135 and 361 of 4K videos for 16 and 32-point modules, respectively. It can also be noticed that 32-point module implementation requires about 3 to 5x hardware resources than 16-point one.

TABLE VI
SYNTHESIS RESULTS OF THE 1D 16 AND 32-POINT DCT-II AND DST-VII [29]

	16-point		32-point	
	DCT-II	DST-VII	DCT-II	DST-VII
Alms	2428(1%)	5981(2.5%)	11231(4.5%)	22794(9%)
Reg.	14041(4%)	50135(15%)	76711(22.5%)	186418(55%)
DSPs	84(5%)	186(11%)	276(16%)	681(40%)
Freq.	401 MHz		268 MHz	
Cycles	61		61	
2K	541 fps		1440 fps	
4K	135 fps		361 fps	

Otherwise, logic resource would be 6x or more and then exceed the target device range. This technique is used in the proposed work without affecting the computational design performance. Furthermore, information given in TABLE VI refers only to requirements for forward transform configuration. This is only to have an idea on the complexity and required resources of hardware implementation of the MTS.

On the other hand, the implementation of the approximation method, aims to maintain the desirable high performance while keeping minimal logic utilization. TABLE VII presents the synthesis results of the proposed unified forward/inverse DCT-II core transform. This latter, configured to operate as Forward or Inverse DCT-II, will be used in DST-VII and inverse DST-VII implementations using adjustment stages.

The second part (right) of TABLE VII gives the synthesis results of the 1D DST-VII approximation implementation. It

TABLE VII
SYNTHESIS RESULTS OF THE UNIFIED 1D 32-POINT DCT CORE TRANSFORM AND THE PROPOSED ARCHITECTURE OF APPROXIMATE FORWARD-INVERSE DST-VII AND DCT-VIII

	DCT-II / IDCT-II		Approximation design	
	16-point	32-point	16-point	32-point
Alms	16505 (7%)		23199 (9%)	
Registers	51862 (15%)		69226 (20%)	
DSPs	328 (20%)		500 (30%)	
Frequency	308 MHz		316 MHz	
Cycles	46	85	55	95
Frame rate (2K)	551 fps	1205 fps	472 fps	1095 fps
Frame rate (4K)	137 fps	300 fps	118 fps	273 fps

embeds the DCT-II core transform and then the additional complexity introduced by adjustment stages can be interpreted or deducted as the difference between DCT-II transform core and DST-VII approximation results. Finally, the synthesis results of the unified 2D approximation circuit are summarized in TABLE VIII. The low computational complexity introduced by adjustment stages will have a high impact on the design performance. We can notice that the larger block size is, the

TABLE VIII
SYNTHESIS RESULTS OF THE UNIFIED 2D IMPLEMENTATION DESIGN OF 32-POINT FORWARD-INVERSE DCT-II AND APPROXIMATE DST-VII AND DCT-VIII

	DCT-II / IDCT-II		App. DST-VII / DCT-VIII	
	16-point	32-point	16-point	32-point
Alms	26130 (10%)		31421 (12%)	
Registers	62109 (18%)		75553 (22.5%)	
DSPs	328 (20%)		500 (30%)	
Memory	64 Kbits (<1%)		64 Kbits (<1%)	
Frequency	225 MHz		228 MHz	
Cycles	95	175	115	196
Frame rate (2K)	194	423	163	386
Frame rate (4K)	49 fps	105 fps	41 fps	96 fps

higher frame rate performance is as long as the pipeline is going deeper with more rows to compute. Thus, the proposal is able to sustain 2K and 4K video processing at 386 and 96 frames per second, respectively. Moreover, it requires only 12% of Alms, 22% of registers and 30% of DSP blocks offered by the target device.

It should be noted that the proposed design is configured to compute one transform type at a time in both sides (encoder and decoder). At the encoder, pixels are processed many times through the rate distortion optimization process which would affect the measured throughput in fps. On the other hand,

TABLE IX
COMPARISON OF DIFFERENT 2D HARDWARE TRANSFORM DESIGNS

Solutions	[23]	[25]	[27]	[29]	Proposed
Technology	28 nm FPGA	40 nm FPGA	ME 20 nm FPGA	ME 20 nm FPGA	ME 20 nm FPGA
Area cons. (Alms)	−−	5292	3654	133017	36766
DSPs	128	−−	32	1561	738
Frequency (Mhz)	222	167	458	147	228
Throughput (fps)	3840×2160p30	3840×2160p30	3840×2160p18	1920×1080p50	3840×2160p96
Transform unit	4×4, 8×8, 16×16, 32×32	4×4, 8×8	4×4,8×8, 16×16,32×32	4×4,8×4,16×4,32×4, 4×8,8×8,16×8,32×8, 4×16,8×16,16×16,32×16, 4×32,8×32,16×32, 32×32	4×4, 8×8, 16×16, 32×32
Transform type	DCT-II	DCT-II, DST-I, DST-VII, DCT-VIII, DCT-V	DCT-II, DST-VII, DCT-VIII	DCT-II, DST-I, DST-VII, DCT-VIII, DCT-V	DCT-II, DST-VII, DCT-VIII
Dimension	2D	2D	2D	2D	2D
Process	Forward	Forward	Forward	Forward	Forward + Inverse

computing two different transform types in parallel would be an alternative way of further optimization, especially that the presented solution is a low hardware area consuming. Then, the actual throughput in the encoder could be increased.

A fair comparison with other works in literature is quite difficult. Most of works are focusing on the 2D-HEVC DCT-II. There are only few works related to MTS hardware implementation. TABLE IX summarizes the key parameters to compare the proposed unified design performance with state of the art works. Work in [25] involves 5 transform types but is restricted to only 4×4 and 8×8 block sizes reaching 30 fps for 4K video coding. Work in [27] presents also an interesting 2D implementation of MTS module regarding hardware cost. It is unified for all block sizes from 4×4 to 32×32 but is not able to support real time coding for 4K videos. Work in [29] is considered as the first 2D MTS implementation supporting 5 transform types and all block sizes (including rectangular ones) from 4 to 32. However, it is drawback is the high usage of logic resource. Finally, all these works consider only forward transform process.

On the other hand, it is worth noting that the proposed design enables both forward and inverse transform processes. In fact, associated with the DCT/ IDCT-core transform, the unified circuit is able to compute 2D forward and inverse implementation for DCT-II, approximate DST-VII and DCT-VIII transform types supporting all sizes from 4×4 to 32×32 unlike the other works presented in TABLE IX. As a result, considering this fact would require twice their results. Moreover, as it is mentioned above, the low additional hardware requirements of forward and inverse DST-VII architectures (for 4×4 and 8×8 sizes through matrix multiplication) can be noticed in area consumption and DSP blocks used for the proposed work (TABLE VIII and TABLE IX). Furthermore, the proposed solution is able to sustain 4K video processing at 96 frames per second requiring only moderate hardware cost of the target device.

VI. CONCLUSION

In this paper we have proposed the approximation method adopted for hardware implementation of forward and inverse MTS concept of VVC standard. It consists in applying low cost adjustment stages to a DCT-II variant in order to approximate DST-VII and DCT-VIII transform types. An efficient hardware implementation of approximate VVC transform process is also proposed. The 32-point 1D architecture allows the processing of 4K videos at 273 frames per seconds. It embeds a reconfigurable and pipelined DCT-II core transform to compute forward and inverse DCT-II sharing the most logic consuming part. The proposed unified 2D implementation design can compute forward and inverse DCT-II, DST-VII and DCT-VIII approximation while using only moderate hardware resource of the target device. The unified circuit is able to sustain 2K and 4K video processing at 386 and 96 frames per second, respectively.

Future works will aim to include 64 transform order for DCT-II. Moreover, rectangular block sizes would be considered with hopefully similar performance results.

ACKNOWLEDGMENT

This project has received funding from Bpifrance Financement under grant DOS0061463/00 (EFIGI FUI project)

REFERENCES

[1] CISCO, "Global_2021_Forecast_Highlights," *https://www.cisco.com/c/dam/m/en_us/solutions/service-provider/vni-forecast-highlights/pdf/Global_2021_Forecast_Highlights.pdf*, p. 6, 2016.

[2] A. Segall, V. Baroncini, J. Boyce, J. Chen, and T. Suzuki, "Joint Call for Proposals on Video Compression with Capability beyond HEVC," *JVET document H1002 (JVET-H1002 v4), 8th JVET Meeting: MACAO, China*, Oct. 2017.

[3] N. Sidaty, W. Hamidouche, O. Deforges, and P. Philippe, "Compression efficiency of the emerging video coding tools," in *2017 IEEE International Conference on Image Processing (ICIP)*, Sept 2017, pp. 2996–3000.

[4] G. J. Sullivan, J. Ohm, W. Han, and T. Wiegand, "Overview of the High Efficiency Video Coding (HEVC) Standard," *IEEE* vol. 22, no. 12, pp. 1649–1668, Dec 2012.

[5] F. Bossen, X. Li, and K. Suehring, "AHG report: Test model software development (AHG3)," in *JVET document N0003 (JVET-N0003), 14th JVET Meeting: Geneva, CH*, March 2019.

[6] J. Chen, E. Alshina, G. J. Sullivan, J. Ohm, and J. Boyce, "Algorithm Description of Joint Exploration Test Model 7(JEM7)," *document JVET G1001 (JVET-G1001 v1), 7th JVET Meeting: Torino, IT*, July 2017.

[7] X. Zhao, J. Chen, M. Karczewicz, L. Zhang, X. Li, and W. Chien, "Enhanced Multiple Transform for Video Coding," in *2016 Data Compression Conference (DCC)*, March 2016, pp. 73–82.

[8] A. Jallouli, F. Belghith, M. A. B. Ayed, W. Hamidouche, J. F. Nezan, and N. Masmoudi, "Statistical analysis of Post-HEVC encoded videos," in *2017 IEEE International Workshop on Signal Processing Systems (SiPS)*, Oct 2017, pp. 1–6.

[9] A. K. Jain, "A Sinusoidal Family of Unitary Transforms," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. PAMI-1, no. 4, pp. 356–365, Oct 1979.

[10] X. Zhao, J. Chen, M. Karczewicz, A. Said, and V. Seregin, "Joint Separable and Non-Separable Transforms for Next-Generation Video Coding," *IEEE Trans. Image Process.*, vol. 27, no. 5, pp. 2514–2525, May 2018.

[11] G. Plonka and M. Tasche, "Fast and Numerically Stable Algorithms for Discrete Cosine Transforms," *Linear Algebra and its Applications*, vol. 394, pp. 309 – 345, 2005.

[12] C. Loeffler, A. Ligtenberg, and G. S. Moschytz, "Practical fast 1-D DCT algorithms with 11 multiplications," in *International Conference on Acoustics, Speech, and Signal Processing (ICASSP),*, vol. 2, May 1989, pp. 988 – 991.

[13] M. Vetterli and H. J. Nussbaumer, "Simple FFT and DCT algorithms with reduced number of operations," *IEEE journal on Signal Processing*, vol. 6, no. 4, pp. 267 – 278, 1984.

[14] R. K. Chivukula and Y. A. Reznik, "Fast Computing of Discrete Cosine and Sine Transforms of Types VI and VII," in *Proc. SPIE*, vol. 8135, 2011, pp. 8135 – 8135 – 10.

[15] M. Puschel and J. M. F. Moura, "Algebraic Signal Processing Theory: Cooley-Tukey Type Algorithms for DCTs and DSTs," *IEEE Transactions on Signal Processing*, vol. 56, no. 4, pp. 1502 – 1521, April 2008.

[16] A. Said, H. Egilmez, V. Seregin, and M. Karczewicz, "Complexity Reduction for Adaptive Multiple Transforms (AMTs) using Adjustment Stages," in *Document JVET J0066 (JVET-J0066 v3), 10th JVET Meeting: San Diego, CA, USA*, April 2018.

[17] A. Arrufat, P. Philippe, K. Reuzé, and O. Déforges, "Low complexity transform competition for HEVC," in *2016 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, March 2016, pp. 1461–1465.

[18] T. Biatek, V. Lorcy, and P. Philippe, "Transform Competition for Temporal Prediction in Video Coding," *IEEE* pp. 1–1, 2018.

[19] P. Meher, S. Park, B. Mohanty, K. Lim, and C. Yeo, "Efficient Integer DCT Architectures for HEVC," *IEEE* vol. 24, no. 1, pp. 168–178, Jan 2014.

[20] W.-J. Chien, J. Boyce, Y.-W. Chen, R. Chernyak, K. Choi, R. Hashimoto, Y.-W. Huang, H. Jang, S. Liu, and D. Luo, "JVET AHG report: Tool reporting procedure (AHG13)," in *JVET document N0013 (JVET-N0013), 14th JVET Meeting: Geneva, CH*, March 2019.

[21] S. Shen, W. Shen, Y. Fan, and X. Zeng, "A Unified 4/8/16/32-Point Integer IDCT Architecture for Multiple Video Coding Standards," in *2012 IEEE International Conference on Multimedia and Expo*, July 2012, pp. 788–793.

[22] J. Zhu, Z. Liu, and D. Wang, "Fully pipelined dct/idct/hadamard unified transform architecture for hevc codec," in *2013 IEEE International Symposium on Circuits and Systems (ISCAS2013)*, May 2013, pp. 677–680.

[23] M.Chen, Y. Zhang, and C. Lu, "Efficient architecture of variable size HEVC 2D-DCT for FPGA platforms," *International Journal of Electronics and Communications*, vol. 73, pp. 1–8, March 2017.

[24] A. Ahmed and M. Shahid, "N Point DCT VLSI Architecture for Emerging HEVC Standard," *VLSI Design*, pp. 1–13, 2012.

[25] A. Mert, E. Kalali, and I. Hamzaoglu, "High Performance 2D Transform Hardware for Future Video Coding," *IEEE Trans. Consum. Electron.*, vol. 62, no. 2, May 2017.

[26] M. Garrido, F. Pescador, M. Chavarrias, P. Lobo, and C. Sanz, "A High Performance FPGA-based Architecture for Future Video Coding Adaptive Multiple Core Transform," *IEEE Trans. Consum. Electron.*, March 2018.

[27] M. J. Garrido, F. Pescador, M. ChavarrÃas, P. J. Lobo, and C. Sanz, "A 2-d multiple transform processor for the versatile video coding standard," *IEEE Transactions on Consumer Electronics*, vol. 65, no. 3, pp. 274–283, Aug 2019.

[28] A. Kammoun, S. B. Jdidia, F. Belghith, W. Hamidouche, J. F. Nezan, and N. Masmoudi, "An optimized hardware implementation of 4-point adaptive multiple transform design for post-HEVC," in *2018 4th International Conference on Advanced Technologies for Signal and Image Processing (ATSIP)*, March 2018, pp. 1–6.

[29] A. Kammoun, W. Hamidouche, F. Belghith, J. Nezan, and N. Masmoudi, "Hardware Design and Implementation of Adaptive Multiple Transforms for the Versatile Video Coding Standard," *IEEE Transactions on Consumer Electronics*, vol. 64, no. 4, October 2018.

[30] Intel-FPGA-Integer-Arithmetic-IP-Cores-User-Guide, "Intel 2017," [Onine]_ Available: https://www.altera.com/en-US/pdfs/literature/ug/ug-lpm-alt-mfug.pdf.

[31] A. Said, H. Egilmez, V. Seregin, M. Karczewicz, and V. Seregin, "Efficient Implementations of AMT with Transform Adjustment Stages," in *Document JVET K0272 (JVET-K0272 v2), 11th JVET Meeting: Ljubljana, SI*, July 2018.

[32] P. Philippe and V. Lorcy, "Further simplification for AMT complexity reduction (CE6.1.2)," in *Document JVET K0299 (JVET-K0299), 11th JVET Meeting: Ljubljana, SI*, July 2018.

[33] V. Lorcy and P. Philippe, "CE6: Further simplification of AMT with adjustment stages (Test CE6.1.6b)," in *Document JVET L0135 (JVET-L0135 v1), 12th JVET Meeting: Macao, CN*, October 2018.

[34] M. Jridi, A. Alfalou, and P. K. Meher, "A Generalized Algorithm and Reconfigurable Architecture for Efficient and Scalable Orthogonal Approximation of DCT," *IEEE Transactions on Circuits and Systems I: Regular Papers*, vol. 62, no. 2, pp. 449–457, Feb 2015.

[35] G. Renda, M. Masera, M. Martina, and G. Masera, "Approximate Arai DCT Architecture for HEVC," in *2017 New Generation of CAS (NGCAS)*, Sept 2017, pp. 133–136.

[36] H. A. F. Almurib, T. N. Kumar, and F. Lombardi, "Approximate DCT Image Compression Using Inexact Computing," *IEEE Transactions on Computers*, vol. 67, no. 2, pp. 149–159, Feb 2018.

[37] M. Masera, M. Martina, and G. Masera, "Adaptive approximated dct architectures for hevc," *IEEE* vol. 27, no. 12, pp. 2714–2725, Dec 2017.

[38] H. Sun, Z. Cheng, A. M. Gharehbaghi, S. Kimura, and M. Fujita, "Approximate dct design for video encoding based on novel truncation scheme," *IEEE Transactions on Circuits and Systems I: Regular Papers*, vol. 66, no. 4, pp. 1517–1530, April 2019.

[39] Z. Chen, Q. Han, and W. Cham, "Low-Complexity Order-64 Integer Cosine Transform Design and its Application in HEVC," *IEEE* vol. 28, no. 9, pp. 2407–2412, Sept 2018.

[40] M. Jridi and P. K. Meher, "Scalable Approximate DCT Architectures for Efficient HEVC-Compliant Video Coding," *IEEE* vol. 27, no. 8, pp. 1815–1825, Aug 2017.

[41] M. Jridi, A. Alfalou, and P. K. Meher, "Efficient approximate core transform and its reconfigurable architectures for HEVC," *Journal of Real-Time Image Processing*, Apr 2018. [Online]. Available: https://doi.org/10.1007/s11554-018-0768-x

[42] Y. Arai, T. Agui, and M. akajimaN, "A Fast DCT-SQ Scheme for Images," *Transactions of IEICE*, vol. E71, no. 11, pp. 1095–1097, Nov 1988.

[43] A. Saxena and F. C. Fernandes, "DCT/DST-Based Transform Coding for Intra Prediction in Image/Video Coding," *IEEE Transactions on Image Processing*, vol. 22, no. 10, pp. 3974–3981, Oct 2013.

[44] W. chyuan Yueh, "Eigenvalues of several tridiagonal matrices," in *Applied Mathematics E-notes*, 2005, pp. 5–66.

[45] L. A. Wolsey, "Integer Programming," in *Wiley, ISBN: 978-0-471-28366-9*, Sep 1998.

[46] R. Chandra, R. Menon, L. Dagum, D. Kohr, D. Maydan, and J. McDonald, "Parallel Programming in OpenMP," in *Morgan Kaufmann Publishers, ISBN: 1-55860-671-8*, 2000.

[47] B. Bross, J. Chen, and S. LiU, "Versatile Video Coding (Draft 3)," in *Document JVET L1001 (JVET-L1001 v9), 12th JVET Meeting: Macao, CN*, October 2018.

[48] Intel-FPGA-Download-Center. [Onine]:https://www.intel.com/content/www/us/en/programmable/downloads/download-center.html.

[49] "Mentor-ModelSim-Functional-Verification-Tool-web," [Onine]:https://www.mentor.com/products/fpga/verification-simulation.

[50] Intel/Altera-2017, "Intel-arria-10-device-overview," [Onine]: https://www.altera.com/documentation/sam1403480274650.html.

[51] JVET, "VVC Test Model (VTM 3.0)." https://vcgit.hhi.fraunhofer.de/jvet/VVCSoftware_VTM/tree/VTM-3.0, Dec. 2018.

Appendix A
16-point adjustment matrix with $\theta = 5$

$$
\begin{pmatrix}
128 & 4 & -1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
-4 & 128 & 8 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
1 & -8 & 127 & 10 & -1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 1 & -10 & 127 & 14 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 2 & -14 & 126 & 16 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 2 & -16 & 126 & 16 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 1 & -16 & 125 & 24 & -1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 2 & -23 & 123 & 26 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 6 & -25 & 124 & 19 & 2 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 3 & -19 & 125 & 21 & 2 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & -21 & 124 & 24 & 3 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 2 & -24 & 123 & 27 & 3 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 2 & -27 & 121 & 31 & 5 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 4 & -31 & 119 & 35 & 2 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 4 & -34 & 117 & 39 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 9 & -38 & 122
\end{pmatrix}
$$

Appendix B
32-point adjustment matrix with $\theta = 5$, columns from 1 to 16

$$
\begin{pmatrix}
128 & 2 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
-2 & 128 & 3 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & -3 & 128 & 4 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & -4 & 128 & 5 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & -5 & 128 & 7 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & -7 & 128 & 8 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & -8 & 127 & 9 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & -9 & 127 & 11 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & -11 & 127 & 12 & 1 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & -12 & 127 & 14 & 1 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & -14 & 126 & 15 & 1 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & -15 & 126 & 16 & 1 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & -16 & 126 & 16 & 1 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & -16 & 126 & 18 & 1 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & -18 & 125 & 20 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & -20 & 125 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 2 & -21 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 2 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0
\end{pmatrix}
$$

Appendix C
32-point adjustment matrix with $\theta = 5$, columns from 17 to 32

$$
\begin{pmatrix}
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
2 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
21 & 2 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
124 & 22 & 2 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
-22 & 124 & 23 & 2 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
2 & -23 & 123 & 25 & 3 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 2 & -25 & 123 & 26 & 3 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 3 & -26 & 122 & 27 & 3 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 3 & -27 & 122 & 28 & 3 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 3 & -28 & 121 & 29 & 4 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 3 & -29 & 121 & 30 & 4 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 4 & -30 & 120 & 32 & 5 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 4 & -32 & 119 & 34 & 5 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 5 & -34 & 118 & 34 & 5 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 5 & -34 & 118 & 36 & 6 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 5 & -36 & 117 & 37 & 6 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 6 & -37 & 117 & 36 & 6 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 6 & -36 & 116 & 41 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 6 & -41 & 121
\end{pmatrix}
$$