

Hardware Acceleration of Approximate Transform Module for the Versatile Video Coding Standard

Ahmed Kammoun, Wassim Hamidouche, Pierrick Philippe, Fatma Belghith, Nouri Massmoudi, Jean-François Nezan

▶ **To cite this version:**

Ahmed Kammoun, Wassim Hamidouche, Pierrick Philippe, Fatma Belghith, Nouri Massmoudi, et al.. Hardware Acceleration of Approximate Transform Module for the Versatile Video Coding Standard. European Signal Processing Conference (EUSIPCO 2019), Sep 2019, Coruña, Spain. hal-02175144

HAL Id: hal-02175144
https://hal.science/hal-02175144

Submitted on 5 Jul 2019

HAL is a multi-disciplinary open access archive for the deposit and dissemination of scientific research documents, whether they are published or not. The documents may come from teaching and research institutions in France or abroad, or from public or private research centers.

L'archive ouverte pluridisciplinaire **HAL**, est destinée au dépôt et à la diffusion de documents scientifiques de niveau recherche, publiés ou non, émanant des établissements d'enseignement et de recherche français ou étrangers, des laboratoires publics ou privés.

Hardware Acceleration of Approximate Transform Module for the Versatile Video Coding Standard

Ahmed Kammoun[⋆,†], Wassim Hamidouche[⋆], Pierrick Philippe[‡], Fatma Belghith[†]
Nouri Massmoudi[†] and Jean-François Nezan[⋆]
[⋆] Univ Rennes, INSA Rennes,CNRS, IETR - UMR 6164, France
Email: firstname.lastname@insa-rennes.fr
[‡] b<>com, E-mail: pierrick.philippe@b-com.com
[†] Univ Sfax, ENIS, LETI-LR99ES37, Tunisia, E-mail: Nouri.Masmoudi@enis.rnu.tn

***Abstract*—Versatile Video Coding (VVC) is the next generation video coding standard expected by the end of 2020. VVC introduces several new coding tools that enable better coding performance compared to the High Efficiency Video Coding (HEVC) standard. The Multiple Transform Selection (MTS) concept, as introduced in VVC, relies on three trigonometrical transforms, and at the encoder side, selects the couple of horizontal and vertical transforms that maximises the Rate-Distortion cost. However, the new Discrete Sine Transform (DST)-VII and Discrete Cosine Transform (DCT)-VIII do not have fast computing algorithms and rely on matrix multiplication, which requires high hardware resources especially for large block sizes.**

This paper tackles the hardware implementation of an approximation of MTS module. This approximation consists in applying adjustment stages, based on sparse block-band matrices, to a variants of DCT-II family mainly DCT-II and its inverse. Therefore, an efficient 2D 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. A unified 2D implementation of 16 and 32-point forward-inverse DCT-II, approximate DST-VII and DCT-VIII is also presented. The synthesis results show that the design is able to sustain 2K and 4K videos at 377 and 94 frames per second, respectively, while using only 18% of Alms, 40% of registers and 34% of Digital Signal Processing (DSP) blocks of the Arria10 SoC platform.

***Index Terms*—Versatile Video Coding, Hardware implementation, Approximation, DCT-II, DST-VII and DCT-VIII.**

I. Introduction

The future video coding standard named VVC is expected by the end of 2020 [1]. The latest draft version provides around 30% coding gain with respect to HEVC [2]. This coding gain is achieved at the expanse of higher computational complexity [3], [4]. The MTS process is one of the key coding tools that have been introduced in the VVC standard [5]. The MTS consists in testing, at the encoder side, different transform types and select the one that provides the best rate distortion performance [6], [7]. In the early stage of VVC standardization, this concept included five DCT/DST transform types [5]: DST-I, DCT-II, DCT-V, DST-VII and DCT-VIII. Statistical analysis showed that DCT-II, DST-VII and DCT-VIII are the most used in transform process and brought more than 90% of the coding gain [8]. Therefore, regarding complexity and coding efficiency, DCT-V and DST-I are no longer considered in VVC.

Hardware implementations are meant to reduce the computational complexity and provide some acceleration to process such modules. However, supporting multiple transforms has several consequences related to memory and logic resources allocation. Therefore, providing high performance design under the hardware constraints of the target device would be a crucial issue. The evolution of the Field-Programmable Gate Array (FPGA) chips [9]–[11], equipped with soft and hard improvements, have encouraged researchers to adopt the implementation of this new transform approach in the objective to provide efficient implementations. Works in [12]–[15] have investigated the implementation of the MTS including the five transform types. They propose different hardware architectures with several limitations. *Mert et al.* [12] propose a 2D implementation including all transform types for 4×4 and 8×8 sizes supporting 2D process using adders and shifts instead of multiplication operations. Although this work presents 2D hardware implementation of all transform types, it only supports 4×4 and 8×8 block sizes. However, the transform at larger block sizes (16×16 and 32×32) are more complex and would require higher resources. In [13], *Garrido et al.* have proposed a pipelined 1D hardware implementation for all block sizes from 4×4 to 32×32. Although the work supports all block sizes, it only deals with 1D design. The transform process consists in 2D operations which could normally be more complex. Moreover, this design does not consider asymmetric block size combinations. In [14], *Kammoun et al.* present a multiplierless implementation of MTS transform module restricted only to 4×4 block size. Later, their work in [15] proposes a unified and 2D hardware implementation using the Intellectual Property (IP) Cores multipliers [16] with the DSPs of the Arria 10 FPGA device. This design supports all block sizes and 2D process with good speedup performance, while it requires high logic utilization compared to solutions proposed in [12], [13].

Several new contributions have been propped by the Joint Video Experts Team (JVET) to overcome the complexity/resources allocations issues of these new transforms [17]–[20]. These solutions are based mainly on approximations aiming to reduce the computational complexity and required

logic resources. Works in [17], [18] and [19], [20] propose an approximations of transforms involved in the MTS. The idea consists in using the DCT-II transform to approximate the other considered transform types (DST-VII and DCT-VIII). In fact, both contributions present the same principle except that the proposal in [20] offers less complexity with practically the same coding performance according to Bjøntegaard Delta Rate (BD-BR) metric. The DCT-II is selected as the core transform since it offers symmetry and recursion properties with practical decomposition in butterfly structures [21]–[23], and has been well studied and optimized for previous hybrid video coding standards.

This work will focus on DCT-II, DST-VII and DCT-VIII forward and inverse implementation through the approximation approach. The approximation is based on the adjustment stage applied on DCT-II and IDCT transforms. The contributions of this paper are summarized in two points: 1) Propose an efficient unified and pipelined architecture of both DCT-II and Inverse DCT-II core transform supporting 4, 8, 16 and 32 block sizes with low computational complexity and logic resource allocation. 2) Propose a 2D implementation of Approximate forward and inverse DST-VII and DCT-VIII design through adjustments stages. The proposed unified architecture enables a high performance efficiency in terms of processed frame per second while using a moderate hardware and logic resources of the FPGA target device.

The rest of the paper is organized as follows. Section II presents the principle of the VVC transform approximation. Section III details 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 IV. Finally, Section V concludes this paper.

II. Multiple stage Approximation Approach

In order to reduce the computational complexity and the resource allocation of the transform block, the approximation approach originally proposed in [17] presents an efficient alternative that approximates several DCT/DST types. It consists in applying adjustment stages of low complexity to DCT-II family transforms with practically no loss in coding performance. The relations between these DCT-II variants transforms can be expressed as follow:

$$C_3 = C_2^T, \quad S_2 = \Lambda \cdot C_2 \cdot \Gamma, \quad S_3 = \Gamma \cdot C_2^T \cdot \Lambda \tag{1}$$

where C_2 is the matrix of DCT_{II} coefficients and matrices Λ and Γ are defined as follows:

$$\Lambda_{i,j} = \begin{cases} 1, & \textit{if } j = N-1-i \\ 0, & \text{otherwise} \end{cases}, \Gamma_{i,j} = \begin{cases} (-1)^i, & \textit{if } j = i \\ 0, & \text{otherwise} \end{cases} \tag{2}$$

In fact, matrices Λ and Γ can be interpreted by vector reflection and sign changes, respectively, which are computationally trivial. Using the transforms of (1), different types of DCTs and DSTs can be approximated by applying adjustment stages (pre-processing and post-processing) to the DCT-II family transforms. The approximation of the DST-VII is computed as follows:

$$\hat{S}_7 = \Gamma \cdot C_2^T \cdot \Lambda \cdot A \tag{3}$$

where Λ and Γ are defined in (2) and A is the adjustment matrix. The objective is to find the appropriate A that minimizes the weighted least-squares error between the DST-VII S_7 and its approximated version $\hat{S}_7$:

$$E(A) = \sum_{i=1}^{N} \omega_i \sum_{j=1}^{N} \left(S_{7\,i,j} - \hat{S}_{7\,i,j} \right)^2 \tag{4}$$

where ω_i, $1 \leq i \leq N$ is a weight vector of size N which might account for the relative importance of the components frequency. When the ω_i is constant equal to 1, the error function corresponds to the squared Frobenius norm. An important property of orthogonality has to be taken in consideration for the adjustment matrix A. The second constraint on the adjustment stages is to be sparse block-band matrix, which can be computed at lower complexity with a number of taps $\theta < N$. The conducted experimentsshow that adjustment stages using "4 to 6-tap" sparse block-band matrices provide a good trade-off between coding gain, complexity and memory usage. For this work, coefficients of 5 tap sparse block-band adjustment matrices are used ($\theta = 5$). The coefficients of A matrix are generated by a genetic algorithm as the solution of the approximation mathematical problem in Equation (4). This method allows to approximate transform types using only DCT-II implementation and sparse block-band matrices as adjustment stages with low additional complexity.Therefore, the overall complexity is significantly lower than using full DCT-VII/DST-VII transforms as in the original design [15].

III. 2D hardware implementation of transform module

As expressed in (3), the approximation is ensured by applying adjustment stages to either the forward DCT-II or its inverse with some changes that are computationally trivial performed through permutation and sign change matrices Λ and Γ. In the following we will detail the main transform core implementation and then how to expand it to support 2D forward and inverse DST-VII and DCT-VIII implementations.

A. Unified Forward and Inverse DCT-II Core Transform

The DCT-II and IDCT-II 32-point kernels are computed as given in the following equations:

$$C_{32} = P_{32} \cdot \begin{pmatrix} C_{16} & 0 \\ 0 & O_{16} \end{pmatrix} \cdot \begin{pmatrix} I_{16} & J_{16} \\ -J_{16} & I_{16} \end{pmatrix} \tag{5}$$

$$C_{32}^T = \begin{pmatrix} I_{16} & -J_{16} \\ J_{16} & I_{16} \end{pmatrix} \cdot \begin{pmatrix} C_{16}^T & 0 \\ 0 & O'_{16} \end{pmatrix} \cdot P_{32} \tag{6}$$

where P_{32} is a permutation matrix to reorder the output data in appropriate from, C_{16} is the half size DCT matrix, O_{16} is a matrix of size 16×16 consisting in odd rows of the first 16 columns of the DCT matrix. I_{16} and J_{16} are, respectively, the identity and the cross-identity (reflection) matrices of size

16×16. Finally, O'_{16} is a matrix of size 16×16 consisting of odd rows of the first 16 columns of the IDCT matrix. Comparing O_{16} and O'_{16}, it can be noticed that for i from 0 to 15, O_{16} i^{th} column has the same coefficients as the 15- i^{th} column of O'_{16} but in inverse order. Subsequently, the O'_{16} matrix can be implemented using the same architecture of O_{16}. This can be achieved with computationally trivial steps, by inverting the inputs and outputs orders. As a result, we propose a unified architecture design that embeds forward and inverse DCT sharing the same 32×32 odd part of the DCT matrix, which is the most consuming in terms of multiplication operations. C_{16} and C_{16}^T are the half size DCT-II and IDCT-II, respectively. They include lower modules order (C_8, C_4) benefiting from recursion property. 32-point DCT implementation (or IDCT-II) requires 340 multiplication operations. The proposed architecture of the unified DCT-II and IDCT-II requires only 424 multiplication operations (256 plus 84 multiplication operations for each C_{16} and C_{16}^T). Reusing the same architecture of O_{16} in a unified DCT-IDCT scheme allows to preserve 256 multiplication operations and then enables considerable reduction in logic resource. Fig 1 illustrates the proposed architecture of the unified DCT-IDCT core transform. From (5) and (6) and benefiting from

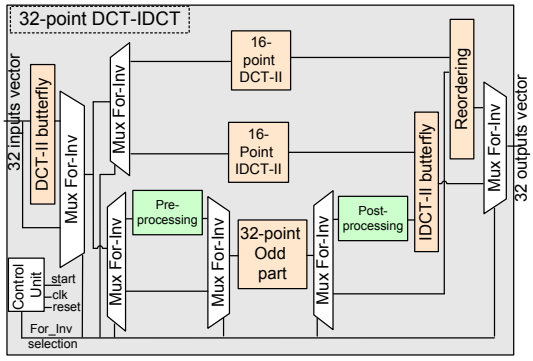


Fig. 1. Proposed architecture of unified 32-point DCT-II and IDCT-II core transform

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. In the case of inverse DCT-II, 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 complexity as explained in (1). The obtained results, associated with C_{16}^T implementation (16-point IDCT-II) outputs go through IDCT butterfly stage in order to provide the final IDCT 32-point outputs. In the case of forward DCT-II, 32-point odd part is computed as O_{16}. Then, the obtained results, with C_{16} implementation (16-point DCT-II), 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.

B. Proposed 2D Implementation of VVC Transform Approximation

In this work we consider 16 and 32 approximation orders as they are the most complex cases. 16 and 32-point adjustments matrices of DST-VII are 5 taps sparse block-band matrices generated and used in (3). They are placed and used as a pre-processing stage in the forward transform process, and a post-processing stage (after transposition) in the inverse one. Fig 2 presents the proposed architecture for the DST-VII approximation in forward ("0" selection path) and inverse ("1" selection path) configurations. Table I gives the computational

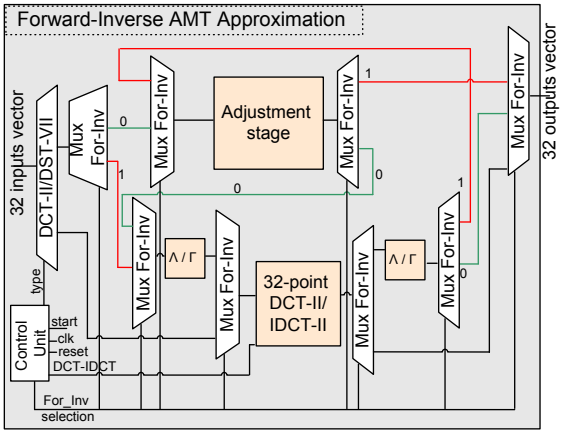


Fig. 2. Proposed architecture of approximate forward-inverse transform design

complexity in number of multiplications required for the proposed implementation of the DCT-II and DST-VII, compared to the original (ie. butterfly structures with recursion property for DCT-II and explicit matrices multiplications for DST-VII [15]). The performance refers to both forward and inverse transforms. 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 operations.

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 used multipliers. DCT-VIII is obtained easily using DST-VII architecture according to (7) with no additional resource requirements

$$C_8 = \Gamma \cdot S_7 \cdot \Lambda \tag{7}$$

The proposed 2D circuit is able to compute efficient approximation of DCT-II, DST-VII and DCT-VIII using a unified 1D forward-inverse DCT-II core transform and adjustment stages (sparse block-band matrices with maximum 5 coefficients per row) with low additional computational complexity. Moreover, it is unified for both 16 and 32 block sizes and reconfigurable to perform either forward or inverse transform processes. Input and output First In First Out (FIFO) memory blocks are added in both ends of the design to store and display input and output vectors. For this, a control unit according to a

TABLE I
COMPARISON OF PROPOSED APPROXIMATION IMPLEMENTATION COMPLEXITY IN NUMBER OF MULTIPLIERS WITH RESPECT TO THE ANCHOR [15]

	16-point				32-point			
	For-Inv DCT-II		For-Inv DST-VII		For-Inv DCT-II		For-Inv DST-VII	
	Anchor	Proposed	Anchor	Proposed	Anchor	Proposed	Anchor	Proposed
Multipliers	168	168	512	58	680	424	2048	114

state machine is defined. It is responsible for assigning the appropriate signals and blocks, and controlling reconfiguration aspects. In addition, it manages all the different steps of 2D process.

IV. EXPERIMENTAL AND SYNTHESIS RESULTS

A. Experimental Setup

The proposed 2D transform design is implemented using the Verilog HDL description language. The architectures of 1D and 2D processes of different orders have been tested with simulation and synthesis software tools [24], [25] under Arria 10 FPGA device [10]. Test bench files were used to validate the output results. The coding performance of the approximate DST-VII and DCT-VIII transforms are assessed under the Common Test Condition with the Benchmark Set (BMS) (including the 5 MTS transforms) software.

B. Rate Distortion Coding Performance

Using 5 tap sparse-band matrices for adjustment stages, Table II gives the BD-BR coding performance of the approximate DST-VII and DCT-VIII transforms in All Intra (AI) and Random Access (RA) configurations. These results can

TABLE II
BD-BR CODING PERFORMANCE OF APPROXIMATE VVC TRANSFORM OVER BMS SOFTWARE FOR AI AND RA CONFIGURATIONS

Cla.	All Intra			Random Access		
	Y	U	V	Y	U	V
A1	0.04%	0.05%	0.10%	0.02%	-0.10%	0.13%
A2	0.12%	0.00%	0.03%	0.03%	0.06%	-0.09%
B	0.06%	0.02%	0.03%	0.04%	0.10%	0.10%
C	0.04%	0.10%	0.03%	0.01%	-0.07%	0.00%
E	0.04%	-0.01%	-0.02%	–	–	–
Av.	**0.06%**	0.03%	0.03%	**0.03%**	0.00%	0.04%

only support the effectiveness of VVC transform approximation method as they show almost no loss in BD-BR coding performance with respect to the accurate MTS implementation (no approximation) in both AI and RA coding configurations.

C. Synthesis Results of the Proposed Approximate Transforms

In order to evaluate the implementation of approximate DCT-II, DST-VII and DCT-VIII transform types, we can first have an idea about their explicit implementation in the original MTS design. They have, each, its own implementation using the associate kernel (matrix). Synthesis results of pipelined DCT-II and DST-VII implementations are presented in Table III for 16 and 32-point. Results show that it provides good performance in terms of processed frames per second up to 135 and 361 of 4K videos for 16 and 32-point modules,

TABLE III
SYNTHESIS RESULTS OF THE 1D 16 AND 32-POINT DCT-II AND DST-VII [15]

	16-point		32-point	
	DCT-II	DST-VII	DCT-II	DST-VII
Alms	2428	5981	11231	22794
Registers	14041	50135	76711	186418
DSPs	84	186	276	681
Frequency	401 MHz		268 MHz	
Cycles	61		61	
Fps (2K)	541		1440	
Fps (4K)	135		361	

respectively. It can also be noticed that 32-point module implementation requires about 3x hardware resource than 16-point one. Moreover, it is worth noting that for 32-point implementation, internal architectures are slightly modified in a way to reduce logic utilization by more than half and also the required clock cycles to compute 32×32 block (61 for 32-point) [15]. Otherwise, logic resource would be 6x or more and then exceed the target device range. In addition, it could be further reduced by sacrificing the pipeline but that eventually would affect the coding speed in terms of processed frames per second (more execution time + lower operational frequency). Furthermore, information given in Table III refers only to requirements for forward transform configuration. This is only to have an idea on how complex and high consuming multiple transform types implementation is. On the other hand, the proposed implementation of the approximation method, aims to maintain the desirable high performance while keeping minimal logic utilization. It should be noted that the same hardware architecture implementation and pipeline process of work in [15] are used for all synthesis. Table IV presents the synthesis results of the proposed DCT-IDCT core transform (first part). This latter, configured to operate as Forward

TABLE IV
SYNTHESIS RESULTS OF THE UNIFIED 32-POINT DCT CORE TRANSFORM AND THE PROPOSED ARCHITECTURE FOR 32-POINT FORWARD-INVERSE DCT-II,DST-VII AND DCT-VIII APPROXIMATION

	DCT-II / IDCT-II		Approximation design	
	16-point	32-point	16-point	32-point
Alms	25271		33327	
Registers	90116		112037	
DSPs	408		580	
Frequency	318 MHz		332 MHz	
Cycles	53	97	63	110
Fps (2K)	493	1079	433	993
Fps (4K)	123	297	108	248

or Inverse DCT (as explained in section III), will be used

easily in DST-VII and inverse DST-VII implementation using adjustment stages with low additional computational complexity. The second part (right) of Table IV gives the synthesis results of the 1D DST-VII approximation implementation. It 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 V. The low computational complexity introduced by adjustment stages will have an impressive impact on the design performance. In fact, associated with the DCT- core transform,

TABLE V
SYNTHESIS RESULTS OF THE 2D 32-POINT FORWARD-INVERSE APPROXIMATION DESIGN FOR DCT-II, DST-VII AND DCT-VIII

2D process	16-point	32-point
Alms	45422 (18%)	
Registers	136003 (40%)	
DSPs	580 (34%)	
Frequency	257 MHz	
Cycles	130	224
Fps (2K)	162	377
Fps (4K)	40	94

the unified design is able to compute 2D forward and inverse approximation for DCT-II, DST-VII and DCT-VIII transform types supporting 16 and 32 sizes. It requires only 18% of Adaptive Logic Modules (Alms), 40% of registers and 34% of DSP blocks offered by the target device. Moreover, it is able to sustain 2K and 4K video processing at 377 and 94 frames per second, respectively.

V. CONCLUSION

In this paper we have proposed an efficient 2D hardware implementation of approximate VVC transform process. The approximation methodology consists in applying low cost adjustment stages to a DCT-II variant in order to approximate the other transform types. The proposed 32-point 1D architecture allows to process 4K videos at 248 frames per seconds. It used a reconfigurable and pipelined DCT-II core transform to compute forward and inverse DCT-II sharing the most logic consuming part. A unified 2D implementation design is also provided. It 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 377 and 94 frames per second, respectively.

REFERENCES

[1] Joint Call for Proposal on Video Compression with Capability beyond HEVC, MPEG document N17195, Joint Video Exploration Team (JVET) of ITU-T VCEG (Q6/16) and ISO/IEC MPEG (JTC 1/SC 29/WG 11), Oct. 2017.

[2] G. J. Sullivan, J. Ohm, W. Han, and T. Wiegand, Overview of the high efficiency video coding (hevc) standard, IEEE Transactions on Circuits and Systems for Video Technology, vol. 22, no. 12, pp. 16491668, Dec 2012.

[3] H. Schwarz, C. Rudat, M. Siekmann, B. Bross, D. Marpe, and T.Wiegand, Coding Efficiency / Complexity Analysis of JEM 1.0 coding tools for the Random Access Configuration, in Document JVET-B0044 3rd 2nd JVET Meeting: San Diego, CA, USA, February 2016.

[4] E. Alshina, A. Alshin, K. Choi, and M. Park, Performance of JEM 1 tools analysis, in Document JVET-B0044 3rd 2nd JVET Meeting: San Diego, CA, USA, February 2016.

[5] Algorithm Description of Joint Exploration Test Model 7(JEM7), MPEG document N17055, Joint Video Exploration Team (JVET) of ITU-T VCEG (Q6/16) and ISO/IEC MPEG (JTC 1/SC 29/WG 11), July 2017.

[6] 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. 25142525, May 2018.

[7] X. Zhao, J. Chen, M. Karczewicz, L. Zhang, X. Li, and W-J. Chien, Enhanced multiple transform for video coding, Data Compression Conference (DCC), pp. 7382, March 2016.

[8] A. Jallouli, F. Belghith, M. A. Ben Ayed, W. Hamidouche, J. Nezan, and N. Masmoudi, Statistical analysis of post-hevc encoded videos, in 2017 IEEE International Workshop on Signal Processing Systems (SiPS), Oct 2017, pp. 16.

[9] Intel-2016, Cyclon-v-device-overview, [Onine]: https://www.altera.com/documentation/sam1403480548153.html.

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

[11] Intel-2017, Intel-stratix-10-gx/sx-device-overview, [Onine]: https://www.altera.com/documentation/joc1442261161666.html.

[12] A.Can. 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.

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

[14] A. Kammoun, S. Ben 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. 16.

[15] 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.

[16] 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.

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

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

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

[20] V. Lorcy and P. Philippe, CE6: Further simplification of AMT with adjustment stages (Test CE6.1.6b), in Document JVETL0135- v1 12th JVET Meeting: Macao, CN, October 2018.

[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. 788793.

[22] P.K. Meher, S.Y. Park, B.K. Mohanty, K.S. Lim, and C. Yeo, Efficient integer dct architectures for hevc, IEEE Transactions on Circuits and Systems for Video Technology, vol. 24, no. 1, pp. 168178, Jan 2014.

[23] A. Ahmed and M.U. Shahid, N Point DCT VLSI Architecture for Emerging HEVC Standard, VLSI Design, pp. 113, 2012.

[24] Intel-FPGA-download-center, [Onine]: https://www.altera.com/downloads/downloadcenter. html.

[25] Mentor-modelsim-functional-verification-tool-web, [Onine]:https://www.mentor.com/products/fv/modelsim.