Journal of Real-Time Image Processing (2022) 19:1081–1090
https://doi.org/10.1007/s11554-022-01250-y

ORIGINAL RESEARCH PAPER

A high-performance two-dimensional transform architecture of variable block sizes for the VVC standard

Sonda Ben Jdidia[1] · Fatma Belghith[1] · Nouri Masmoudi[1]

Received: 25 March 2022 / Accepted: 17 August 2022 / Published online: 1 September 2022
© The Author(s), under exclusive licence to Springer-Verlag GmbH Germany, part of Springer Nature 2022

Abstract
The versatile video coding standard H.266/VVC release has been accompanied with various new contributions to improve the coding efficiency beyond the high-efficiency video coding (HEVC), particularly in the transformation process. The adaptive multiple transform (AMT) is one of the new tools that was introduced in the transform module. It involves five transform types from the discrete cosine transform/discrete sine transform families with larger block sizes. The DCT-II has a fast computing algorithm, while the DST-VII relies on a complex matrix multiplication. This has led to an additional computational complexity. The approximation of the DST-VII can be used for the transform optimization. At the hardware level, this method can provide a gain in power consumption, logic resources use and speed. In this paper, a unifed two-dimensional transform architecture that enables exact and approximate DST-VII computation of sizes $8 \times 8, 8 \times 16, 8 \times 32, 16 \times 8, 16 \times 16, 16 \times 32, 32 \times 8, 32 \times 16$ and 32×32 is proposed. The exact transform computation can be processed using either multipliers or the MCM algorithm, while the approximate transform computation is based on additions and bit-shifting operations. All the designs are implemented under the Arria 10 FPGA device. The synthesis results show that the proposed design implementing the approximate transform matrices is the most efficient method with only 4% of area consumption. It reduces the logic utilization by more than 65% compared to the multipliers-based exact transform design, while about 53% of hardware cost saving is obtained when compared to the MCM-based computation. Furthermore, the approximate-based 2D transform architecture can operate at 78 MHz allowing a real-time coding for 2K and 4K videos at 100 and 25 frames/s, respectively.

Keywords Versatile video coding · Discrete sine transform · Approximate computing · Hardware implementation · FPGA

1 Introduction

All along the past years, the multimedia world has witnessed a real revolution through the emergence of a myriad of applications where digital videos are ubiquitous. This revolution is accompanied by users' interest in immersive and high-quality video contents and by the platforms supporting video sharing such as social networks, messaging platforms and video streaming services.

As a result, the amount of data exchanged, especially on the Internet, continues to increase. A forecast done by Cisco indicates that, by 2022, IP video traffic will represent 82% of the global internet traffic [1]. In addition, Comcast estimates that the current COVID-19 crisis has increased the Voice over Internet Protocol (VoIP) and video conferencing by 210-285 % and other video consumption by 20-40% compared to the pre-pandemic period [2]. This tremendous growth is mainly explained by pervasive connectivity and the proliferation of advanced multimedia solutions that support emerging formats such as 4K/8K Ultra-High-Definition (UHD) video and 360° omnidirectional video. This leads to major challenges for communication networks that have limited bandwidths and storage capacity, making it necessary to have more efficient video compression standards that keep the quality unchanged.

To handle the excessive increase in video consumption, the ISO/IEC (International Standardization Organization/International Electro-technical Commission) Motion Picture Experts Group (MPEG) and ITU-T (International Telecommunication Union-Telecommunication) Video Coding Experts Group (VCEG) have developed different

✉ Sonda Ben Jdidia
sonda.benjdidia@enis.tn

[1] Université de Sfax, ENIS, LETI, 3038 Sfax, Tunisia

video coding standards. The latest standards, Advanced Video Coding (AVC)/H.264 [3] and High-Efficiency Video Coding (HEVC)/ H.265 [4], were released in 2003 and 2013, respectively.

However, despite the crucial improvements provided by the HEVC standard, a more efficient video compression standard is required especially for high-quality video contents. Therefore, VCEG and MPEG have joined again and formed the Joint Video Exploration Team (JVET) in October 2015 to study new coding tools beyond the capabilities of the HEVC. A first version of the Versatile Video Coding standard was published in July 2020 as H.266/VVC [5].

The VVC standard has improved the coding efficiency while increasing its complexity from 7.5× to 34× compared to the HEVC. This has revealed a critical need for more effective methods of video compression.

The introduction of the Adaptive Multiple Transform (AMT) which incorporates new types of transforms with sizes ranging from 4×4 up to 128×128 has led to an additional complexity in the VVC transform module. This has led researchers to study new transforms offering a compromise between performance and arithmetic complexity. The transform approximation has proved to be an efficient approach, since it simplifies the calculations of existing transforms rather than inventing new ones and provides approximate results that have practical significance at a considerably lower arithmetic and computational cost [6].

Only a few works in the literature have dealt with the hardware implementation of the VVC transforms. Some of which have targeted the optimization of the transforms using the approximation approach. This study proposes a unified two-dimensional architecture that provides exact and approximate DST-VII transform calculations. It embeds all the square and rectangular transform block sizes ($8\times8, 8\times16, 8\times32, 16\times8, 16\times16, 16\times32, 32\times8, 32\times16$ $8\times8, 8\times16, 8\times32, 16\times8, 16\times16, 16\times32, 32\times8, 32\times16$ and 32×32). The approximate transform design achieves a compromise between the hardware resources' consumption and the operational frequency. It is able to perform 2Kp100 video coding.

The remainder of this paper is organized as follows. Section II depicts the principle of the transform coding in the VVC as well as some related works which have dealt with efficient hardware implementation of either exact or approximate transforms. Section III describes the proposed hardware implementation of the 2D DST-VII transform as well as the different modules that it includes. Section IV presents the synthesis results of the proposed hardware implementations. Finally, section V concludes the paper.

2 VVC transform

2.1 VVC transform review

The discrete cosine/sine transform proves to be a basic algorithm for compression owing to its signal decorrelation and energy concentration properties. The DCT and DST transforms are widely used in video compression applications. In the VVC, the AMT approach notably includes three features:

- Multiple core transforms, namely, DCT-II, DCT-V, DCT-VIII, DST-I, and DST-VII, whose basis functions are shown in Table 1. The coefficients of the transform matrices are scaled and rounded into integer numbers while maintaining the important transform properties.
- Either similar or different transform types can be used in the 1D and the 2D transforms.
- Either symmetric or asymmetric transform units are supported which is not the case in the HEVC.

The AMT approach is performed on inter- and intra-coded residual blocks. It is controlled by a CU level flag which indicates whether the AMT is applied or not. When the CU level flag value is 0, the residual coding is ensured by the DCT-II/DST-VII transform like that of the HEVC. Otherwise, two other flags are added to mention the horizontal and

Table 1 Basis functions of DCT-II/V/VIII and DST-I/VII transforms

Transform type	Basis function $T_i(j), i,j = 0, 1, ..., N-1$
DCT-II	$T_i(j) = \omega_0 \cdot \sqrt{\frac{2}{N}} \cos\left(\frac{\pi.i.(2j+1)}{2N}\right), where\ \omega_0 = \begin{cases} \sqrt{\frac{2}{N}}, & i=0 \\ 1, & i \neq 0 \end{cases}$
DCT-V	$T_i(j) = \omega_0.\omega_1 \sqrt{\frac{2}{2N-1}} \cos\left(\frac{2\pi.i.j}{2N-1}\right), where\ \omega_0 = \begin{cases} \sqrt{\frac{2}{N}}, & i=0 \\ 1, & i \neq 0 \end{cases}, \omega_1 = \begin{cases} \sqrt{\frac{2}{N}}, & j=0 \\ 1, & j \neq 0 \end{cases}$
DCT-VIII	$T_i(j) = \sqrt{\frac{4}{2N+1}} \cos(\frac{\pi.(2i+1).(2j+1)}{4N+2})$
DST-I	$T_i(j) = \sqrt{\frac{2}{N+1}} \sin(\frac{\pi.(i+1).(j+1)}{N+1})$
DST-VII	$T_i(j) = \sqrt{\frac{4}{2N+1}} \sin(\frac{\pi.(2i+1).(j+1)}{2N+1})$

vertical transforms to be used. For intra-coded Coding Unit (CU), a transform selection is carried out depending on the Intra-Prediction Mode (IPM) selected among the 67 given modes [7]. For each mode, a transform set is chosen from three sets as defined in Table 2. The two transform candidates within the selected set are tested for both horizontal and vertical transforms. Afterwards, the optimal transform which meets the lowest rate distortion cost is chosen [8]. As for the inter-coded CU, the DST-VII and DCT-VII transforms are used for all inter modes and for both horizontal and vertical transforms.

The reference software JEM7.1 [9] adopts integer transforms in the transformation process. However, a new concept called Multiple Transform Selection (MTS) is adopted in the final version of the reference software, named VVC Test Model (VTM), with only three transform kernels (DCT-II, DST-VII, and DCT-VIII).

The 2D transform is computed by performing the 1D transform in the horizontal and vertical directions, respectively. The 1D horizontal transform is computed according to Eq. (1)

$$Y_{int} = T_H \cdot X^T, \tag{1}$$

where T_H is the $N \times N$ matrix of the horizontal transform which can be one of the transform types represented in Table 1, X is the $(1 \times N)$ input row vector, and Y_{int} is the $(N \times 1)$ intermediate output column vector. After performing the 1D horizontal transform, the 1D vertical transform is computed according to Eq. (2)

$$Y = T_V \cdot Y_{int}^T, \tag{2}$$

where T_V is the $N \times N$ matrix of the vertical transform which can be of the same or different type as the horizontal transform and Y is the $(N \times 1)$ output column vector. Equation 3 summarizes the 2D transform computation

$$Y = T_V \cdot (T_H \cdot X^T)^T. \tag{3}$$

Since the transform types are independent, it is actually hard to design a unified architecture that allows the 2D computation of all the five transform types together. Consequently, it is recommended to propose an efficient design

Table 2 Pre-defined transform candidate subsets

Transform Set	Transform candidates
0	DST-VII, DCT-VIII
1	DST-VII, DST-I
2	DST-VII, DCT-V

for each transform type with a low hardware cost. This can be achieved by efficiently implementing either the exact or the approximate version of the transform. A fairly relevant approximation of the DST-VII transform was proposed in [10] which targets the transforms of sizes 8, 16, and 32. The hardware architectures of both exact and approximate 1D transforms were designed. This paper focuses on the 2D implementation of the DST-VII transform in its exact and approximate versions based on the 1D architecture of [10].

2.2 Related works

Within the transform hardware implementation context, efficient transform computation algorithms represent a major controversial topic, subject of intensive studies. Based on a literature review, the transform computation' relating approaches tend to be categorized under the exact and approximate methods. Concerning the present work, the focus of interest is primarily laid on investigating a unifed 2D architecture that enable both exact and approximate transform computations. To our knowledge, no previously elaborated work has suggested to design a unified architecture that involves two transform computation methods.

2.2.1 Hardware implementation of exact transforms

Within the context of efficient exact transform computation algorithms, Mert et al. [11] proposed two methods of the AMT 2D hardware implementation including all the transform types for sizes 4×4 and 8×8. The first method uses separate datapaths for each 1D transform and the second method uses one reconfigurable datapath for the 1D column/row transform. Implementation results show that the reconfigurable hardware consumes less logic resources and more energy than the baseline hardware. The proposed methods allow a 2D transform computation of all the transform types, but do not support transform sizes larger than 8, which are more complex and require more logic resources. Authors in [12] suggested a hardware implementation of the AMT of size 8. The transforms are computed using either Altera multipliers (LPM) or addition and bit-shifting operations. Synthesis results show that the design based on addition and bit-shifting operations is more efficient in terms of hardware resources consumption and operational frequency. Garrido et al. [13] proposed a hardware architecture for the five transforms of the AMT allowing a $4 \times 4, 8 \times 8, 16 \times 16$ or 32×32 1D transform. The proposed architecture is implemented on three FPGA chips. The synthesis results show that the design can process real-time 4K UHD formats with a low hardware resources' consumption. However, this architecture only concerns the 1D transform applied to symmetric block size combinations, whereas including the 2D transform as well as considering asymmetric block may be more complex.

The authors extended their work in [14] to consider the 2D transform design. The architecture is composed of two 1D AMT processors and a transpose memory to store the 1D intermediate results. The design is synthesized for a low-end, a middle-end and a high-end FPGA devices. Synthesis results reveal that the logic utilization is almost the same in all cases, while the operational clock frequency differs from one case to another and depends on the FPGA technology. Authors in [15] proposed an implementation of size 4 AMT transforms using additions and shifts instead of multiplications. The transforms are described in two versions : with or without use of the state machine. The synthesis results show that both methods, supporting all the transform types, require less than 3% of the resources available in the FPGA device and provide 285 MHz and 318 MHz, respectively, as the maximum operational frequency. This work was extended in [16] to a 2D hardware implementation supporting square and rectangular block sizes. The proposed design requires about 53% of logic resources and 93% of DSPs available in the FPGA device. Fan et al. [17] designed a 32×32 block-based transform architecture that allows a 2D DST-VII and DCT-VIII transform computation of all sizes. The proposed ASIC-based design needs a small logic utilization and reaches an operational frequency of 250 MHz. Sau et al. [18] introduced two different implementations of the AMT. The first is a standalone design which permits the computation of one single transform. The second is a reconfigurable design which allows the computation of multiple transforms. The different architectures, performing 4×4 2D transform, are evaluated and compared in terms of hardware resources consumption and maximum operational frequency. Yibo et al. [19] adopted a unified transform architecture that computes 1D DST-VII and DCT-VIII transforms of all sizes through a Reduced Adder Graph (RAG-n) algorithm to design the Shift-Addition Units (SAU) . A comparison with the multiplication-based designs show that the SAU-based design results in area and power savings.

2.2.2 Hardware implementation of approximate transforms

Some relevant works revealed efficient computation of the approximate transforms instead of the exact ones. In this context, Kammoun et al. [20] developed a hardware design of forward and inverse 2D approximate transforms of sizes $4 \times 4, 8 \times 8, 16 \times 16$ and 32×32 meant to consume only 9% of the total hardware area and only 15% of the DSP blocks. Yet, it does not support rectangular block sizes which may result in additional area consumption. Authors in [21] presented a hardware implementation of 8-point exact and approximate DST-VII transform. The hardware synthesis results show that the approximate design presents 70% of hardware resources saving with regard to the exact transform implementation. Ben Jdidia et al. [10] proposed a hardware description of 8- and 16-point exact transforms using multipliers and the Multiple Constant Multiplication (MCM) algorithm as well as the approximate transforms using addition and bit-shifting operations. The synthesis results reveal that the proposed approximations provide a significant gain in terms of hardware resources and power consumption. The work in [22] depicts the implementation of the approximate DST-VII and DCT-VIII 1D transforms of sizes ranging from 4 to 32. The proposed hardware design requires a reduced hardware cost.

3 2D DST-VII transform hardware implementation

A two-dimensional transform applied to an $M \times N$ transform block is processed sequentially as follows :

1. The M-point 1D transform is performed on the transform block input columns.
2. The results are rounded by applying the appropriate shift value according to the transform size.
3. The rounded results are transposed.
4. The N-point 1D transform is performed on the columns of the transposed matrix.
5. The results are rounded by applying the appropriate shift value according to the transform size to get the final output results.

In this section, an architecture is described to compute the 2D DST-VII transform for the blocks of sizes up to 32×32. Figure 1 sums up the proposed architecture of the 2D transform. The interface of the 2D DST-VII architecture is outlined in Table 3. The transform computation is launched with a pulse in the X_{av} input signal. The 1D and 2D transform sizes are specified by sel_v and sel_h input signals, respectively. The 16-bit input data $X0..X7$ are fed to the design to compute the outputs $Y0..Y7$ which are validated by the Y_{av} output signal.

In the proposed design, three 1D transforms are instantiated, one for each size. They are reused for the second 1D transform. Duplicating the 1D transforms can be needed to pipeline all the computation, at the cost of a higher area usage. The proposed architecture is based on the 1D DST-VII design of [10] to compute either the exact or the approximate version of the DST-VII transform. Since the design is unified for sizes 8, 16 and 32, then, according to the "sel_v" signal which defines the transform size to be used in the 1D transform, the input column is given in one cycle when the transform size is 8, two cycles when the transform size is 16 (first half first) and four cycles when the transform size is 32 (first quarter first). The "transform index" signal takes '0'

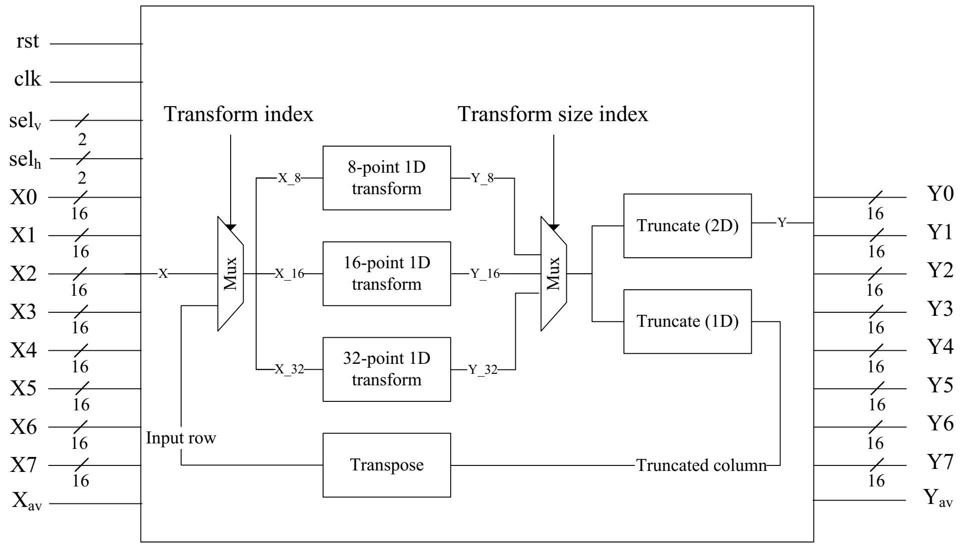

Fig. 1 The proposed 2D transform architecture

Table 3 2D interface description design

Signal	I/O	Bits	Description
clk	I	1	System clock
reset	I	1	Asynchronous system reset, active low
sel_v	I	2	1D transform size :
			0 : size 8, 1 : size 16, 2 : size 32
sel_h	I	2	2D transform size :
			0 : size 8, 1 : size 16, 2 : size 32
X_{av}	I	1	Inputs are available : positive pulse
X0 .. X7	I	128	Input vector, 8×16–bit inputs
Y0 .. Y7	O	128	Output vector, 8×16–bit outputs
Y_{av}	O	1	Outputs are available : positive pulse

during the first 1D transform and '1' during the second 1D transform. After performing the 1D transform, the outputs are rounded using the shift values defined in the VVC reference software. These values depend on the transform size and whether it is the 1D or 2D transform. The truncated columns are fed to the transpose module. The result will serve as input to the second 1D transform. The number of inputs is specified according to the "sel_h" signal which defines the transform size to be used in the second 1D transform. Once the 2D outputs are obtained and shifted, the results which are the final output rows are given in one cycle when the transform size is 8, two cycles when the transform size is 16 (first half first) and four cycles when the transform size is 32 (first quarter first). The signal Y_{av} indicates that the outputs are available. The different modules of the proposed architecture are detailed in the sections below.

An example of a 2D transform computation for a 16×32 transform block is outlined in Fig. 2. When the input signal value sel_v is set to "01", it indicates that the size of the first 1D transform is equal to 16. When the input signal value sel_h is equal to "10", the size of the second 1D transform is equal to 32. The 32 input columns are read from the input file; each column is read every two clock cycles, because the input column is composed of 16 coefficients and only 8 inputs can be read in every cycle. These inputs are registered in the signal X_16, which is a vector of 16 inputs, as shown in Fig. 1. Once all the values of the signal X_16 are read, the computation of the 1D transform of size 16 can be launched to generate the intermediate outputs, which are stored in the signal Y_16. The truncate and the transposition modules are processed combinatorially. The result of the transposition stage will serve as input for the second 1D transform, which is a transform of size 32. The same process is used to compute the output values, which are stored in the signal Y_32. These values are shifted to get the final output results which are given row by row, each row every four clock cycles.

3.1 The 1D transform computation

The N-point 1D transform architecture is depicted in Fig. 3. It allows the computation of the 1D N-point transform ($N \in \{8, 16, 32\}$) using one of the methods : the exact transform using either multipliers or the MCM algorithm and the approximate transform using adders and bit-shifting operations. One of the three methods is chosen according to the control signal "computation method". For a given input column vector X_N which is an $N \times 16$–bit vector, one of the mentioned methods is used to compute the ouput Y_N whose width depends on the computation method. The three methods are well explained in [10].

The transform module is basically a matrix multiplication. The transform computation performed using multipliers requires a lot of hardware resources. The MCM algorithm is used in transform computation, because the input number is always multiplied by the same transform matrix

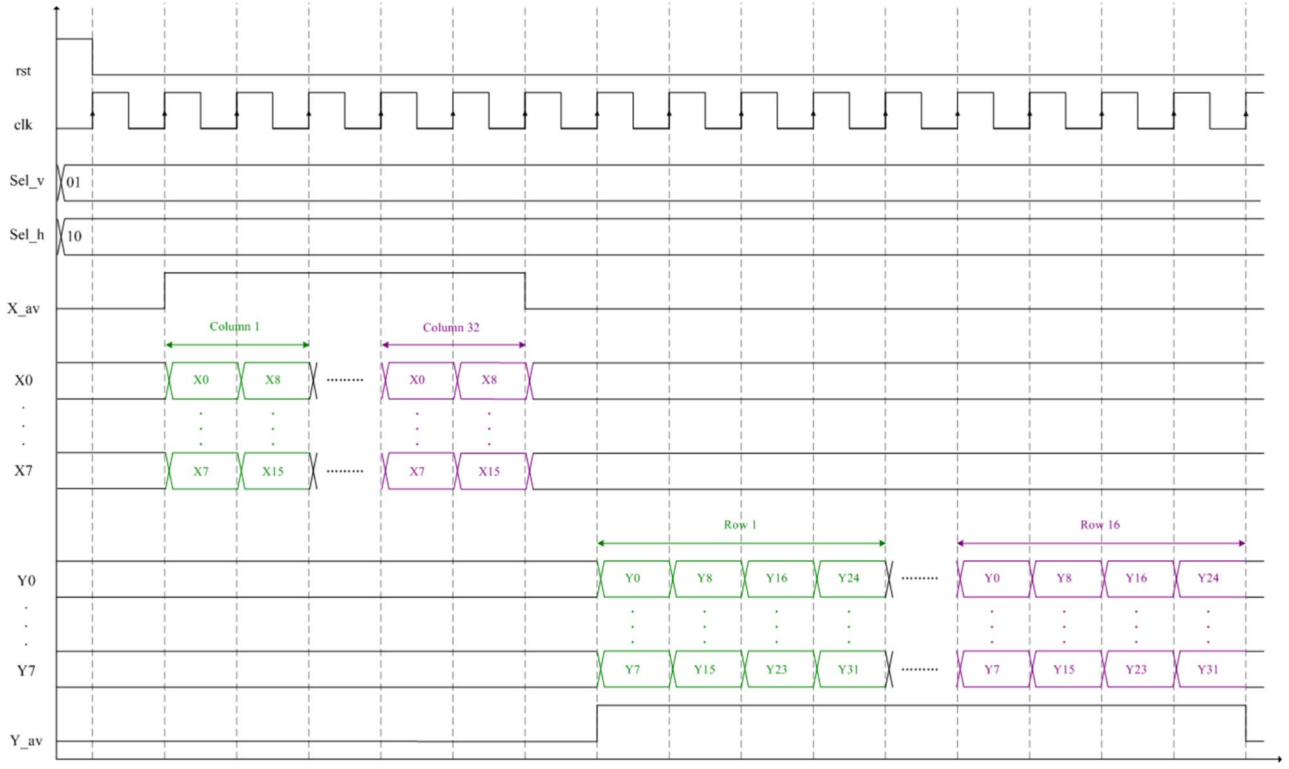

Fig. 2 Timing chronogram for the 2D transform of a 16 × 32 block size

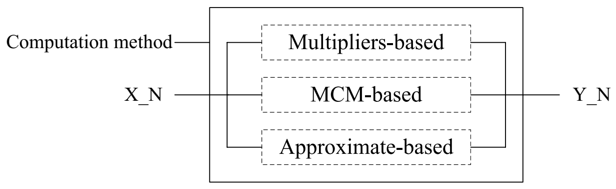

Fig. 3 The N-point 1D transform architecture

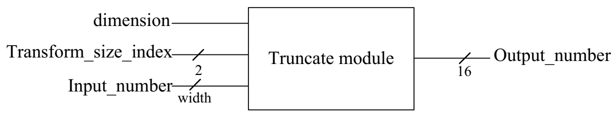

Fig. 4 The truncate module

coefficients. Another alternative which drastically reduces the hardware resources consumption is to consider approximate transforms instead of exact ones. These lightweight transforms can be efficiently implemented by only adopting adders and bit-shifting operations.

3.2 The truncate module

Figure 4 illustrates the truncate module which is used to scale the outputs of the 1D column DST-VII and the 1D row DST-VII to 16 bits. The truncate (1D) module shifts the 1D column outputs right by 6, 7 and 8 bits when the transform size is 8, 16 and 32, respectively. The truncate (2D) module shifts the 1D row outputs right by 11, 12 and 13 bits when the transform size is 8, 16 and 32, respectively. Selecting whether it is the 1D or the 2D transform is done using the "dimension" input signal. The transform size is selected according to the "transform_size_index" signal.

3.3 The transpose module

The intermediate 1D results are temporarily stored in a matrix of registers. To transpose the result of the first 1D transform, each transform column is written in a matrix of registers, and it is read row by row. Figure 5 illustrates the transpose module. This block design stores the 1D output column vectors of size *N* after applying the appropriate shift value.

4 Experimental results

4.1 Experimental setup

The 2D transform design is described using the VHDL hardware description language. The 2D transform process of the

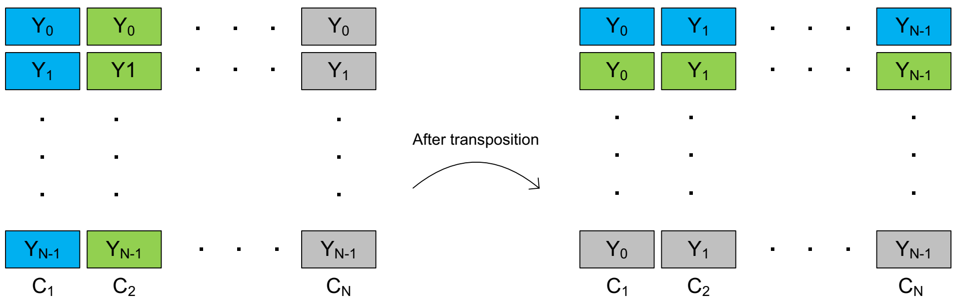

Fig. 5 The transpose module

Table 4 Synthesis results of the 2D DST-VII transform design

	Exact transform		Approximate transform
	Multipliers-based	MCM-based	
Pins	264 (27%)	264 (27%)	264 (27%)
ALMs	50876 (12%)	37527 (9%)	17631 (4%)
Registers	17313	17313	17313
DSPs	0	0	0
Frequency (MHz)	69	63	78

different block sizes is tested with simulation and synthesis software tools [23, 24], under the Arria 10 FPGA device [25]. Self-checking testbench files are used to verify the output results. The test process is carried out as follows. A set of pseudorandom inputs and their corresponding outputs are generated using a python script for the different block sizes. Then, the testbench file reads the input file and compares the simulation results with those obtained using the reference software implementation. The different tests cover all the transform unit sizes from 8×8 to 32×32 including symmetric and asymmetric blocks.

4.2 Results and analysis

The synthesis results of the 2D transform design are displayed in Table 4. The design uses 264 pins which represent 27% of the total pins available in the FPGA device. Pin counts regroups the clock, reset, sel_v, sel_h, X_{av} and the 8 16-bit input signals as well as Y_{av} and the 8 16-bit output signals. For the exact 2D transform computation using multipliers, only 12% of the ALMs and 1% of the registers are required. The design does not use any DSP blocks and reaches an operational frequency up to 69 MHz. The MCM-based design is more efficient in terms of logic utilization. It uses 9% of the ALMs. The operational frequency achieved by the MCM-based transform design is of 63 MHz. The computation of the 2D approximate transform is the most efficient implementation in terms of area consumption. In comparison to the multipliers-based computation, the 2D approximate transform computation requires approximately 2× less logic resources. Compared to the MCM-based computation, about 53% of hardware cost saving is obtained.

Table 5 Performance of the 2D transform design

Computation method	Transform block size	Cycles	2K fps	4K fps
Multipliers	8×8	17	84	21
	8×16	33	86	22
	16×16	65	87	22
	16×32	129	88	22
	32×32	257	88	22
MCM	8×8	17	76	19
	8×16	33	79	20
	16×16	65	80	20
	16×32	129	80	20
	32×32	257	81	20
Approximated	8×8	17	94	24
	8×16	33	97	24
	16×16	65	99	25
	16×32	129	100	25
	32×32	257	100	25

These synthesis results of the 2D transform design have the same tendencies as for the 1D transform design (presented in [10]) in terms of logic utilization. The operational frequency is improved compared to the exact transform computation methods. It reaches 78 MHz.

The performances of the 2D transform design of the different block sizes are evaluated in Table 5. The number of cycles depends on the transform block size which is being computed. It varies from 17 cycles in the case of an 8×8 block size to 257 cycles in the case of a 32×32 block size. The throughput in frames per seconds (fps) is calculated

Table 6 Comparison of different 2D hardware transform designs for the VVC standard

Design	Mert et al.[11]	Kammoun et al.[16]	Garrido et al.[14]	Kammoun et al.[20]	Proposed
Technology	40 nm FPGA	ME 20 nm FPGA	ME 20nm FPGA	ME 20 nm FPGA	ME 20 nm FPGA
ALMs	5292	133017	1312	36766	17631
Registers	–	274902	3654	–	17313
DSPs	–	1561	32	738	0
Frequency (MHz)	167	147	458	228	78
Frames/sec	3840x2160p30	1920x1080p50	3840x2610p18	3840x2610p96	1920x1080p100
Transform unit	4, 8	4, 8, 16, 32	4, 8, 16, 32	4, 8, 16, 32	8, 16, 32
	sym	sym + asym	sym	sym	sym + asym
Transform type	AMT	AMT	MTS	MTS	DST-VII
Dimension	2D	2D	2D	2D	2D

using Eq. (4) for the different 2D block size combinations. *Freq* represents the operational frequency used to compute an $M \times N$ block size over C_{Cycles} clock cycles, *Res* defines the video resolutions 2K (1920×1080) and 4K (3840×2160), and 1.5 is a factor assigned to the image color sampling in 4:2:0

$$fps = \frac{\text{Freq} \times M \times N}{C_{\text{Cycles}} \times Res \times 1.5}. \tag{4}$$

The proposed approximate-based transform achieves a high frame rate performance. It can be noticed that the larger the transform unit is, the better the throughputs are. These results are obtained assuming the same block size for all the transforms. Whereas, in real video applications, each frame is encoded using different transform block sizes. Consequently, in this regard, the proposed 2D design may have a better performance. The approximate-based 2D transform architecture has a more efficient coding performance in terms of frames per second compared to the exact transform design computed using either the multipliers or the MCM algorithm. In fact, it enables a real-time coding for 2K and 4K videos at 100 and 25 frames/s, respectively.

Table 6 displays a comparison between different 2D hardware designs for the VVC standard. The approximate-based 2D transform architecture ensures a good compromise between the logic utilization and the operational frequency. Compared to the design proposed by Mert et al.[11], the proposed design consumes more hardware resources, because it supports 8-, 16- and 32-point transform modules and enables all the possible combinations of block sizes which can be symmetric or asymmetric. The design proposed by Kammoun et al.[16] supports all the AMT types and sizes, but it consumes important logic resources in terms of ALMs and registers. The design proposed by Garrido et al.[14] is the most efficient one regarding area. The hardware architecture proposed by Kammoun et al.[20] is more efficient than their previous work presented in [16] in terms of logic utilization. However, the proposed design does not consider the asymmetric transform block sizes and requires more logic resources.

5 Conclusion

In this work, a unified two-dimensional architecture of the exact and the approximate versions of the DST-VII transform is proposed. It allows the computation of the square and rectangular transform block sizes ($8 \times 8, 8 \times 16, 8 \times 32, 16 \times 8, 16 \times 16, 16 \times 32, 32 \times 8, 32 \times 16$ and 32×32).

Actually, the proposed 2D transform mainly consists of the 1D transform module, the truncate module, and the transpose module. Three variants can be used to compute the 1D transform including the multipliers-based exact transform, the MCM-based exact transform, and the approximate transform with adders and bit-shifting operations.

All the designs are implemented under the Arria 10 FPGA device. The proposed architecture turns out to remarkably outperform the already existing designs as it ensures a good compromise between the logic utilization and the operational frequency. The proposed designs highlight the efficiency of the approximated transforms through synthesis results. The proposed computation of the 2D approximate transform is the most efficient implementation with only 4% of area consumption. Compared to the multipliers-based computation, the 2D approximate transform requires nearly 2× less logic resources. Compared to the MCM-based computation, about 53% of hardware cost saving is obtained. More than that, the approximate-based 2D transform architecture can operate at 78 MHz enabling a real-time coding for 2K and 4K videos at 100 and 25 frames/s, respectively.

To the best of our knowledge, this is the first 2D unified transform design that allows the computation of the exact transform using two different methods as well as the approximate transforms. The proposed architecture takes into account all the symmetric and asymmetric transform units.

As future works, the proposed design can be extended to a unified one that supports the computation of the DST-VII and the DCT-VIII transforms of different lengths. This is because the DCT-VIII can be easily computed from the DST-VII transform. This hardware sharing will not involve any additional hardware resources. Furthermore, even though the proposed architecture is designed for the encoder, it can be extended for the decoder side by applying a transposition to the transform matrices.

References

1. Cisco Visual Networking Index: Forecast and Trends 2017-2022. http://web.archive.org/web/20181213105003/https:/www.cisco.com/c/en/us/solutions/collateral/service-provider/visual-networking-index-vni/white-paper-c11-741490.pdf (2018). Accessed June 2021
2. Comcast Corporation: COVID-19 network update. https://corporate.comcast.com/covid-19/network/may-20-2020 (2021). Accessed June 2021
3. Wiegand, T., Sullivan, G.J., Bjontegaard, G., et al.: Overview of the h. 264/avc video coding standard. IEEE Trans. Circuits Syst. Video Technol. **13**(7), 560–576 (2003)
4. Sullivan, G.J., Ohm, J.-R., Han, W.-J., et al.: Overview of the high efficiency video coding (hevc) standard. IEEE Trans. Circuits Syst. Video Technol. **22**(12), 1649–1668 (2012)
5. International Telecommunication Union (ITU): New versatile video coding standard to enable next-generation video compression.' https://www.itu.int/en/mediacentre/Pages/pr13-2020-New-Versatile-Video-coding-standard-video-compression.aspx (2021). Accessed Apr 2021
6. Kammoun, A., Hamidouche, W., Philipp, P., Belghith, F., Massmoudi, N., Nezan, J.-F.: Hardware acceleration of approximate transform module for the versatile video coding standard. In: 2019 27th European Signal Processing Conference (EUSIPCO), pp. 1–5 (2019). https://doi.org/10.23919/EUSIPCO.2019.8902594
7. Schwarz, H., Rudat, C., Siekmann, M., et al.: Coding efficiency/complexity analysis of jem 1.0 coding tools for the random access configuration. In: Document JVET-B0044 3rd 2nd JVET Meeting (2016)
8. Chen, J., Alshina, E., Sullivan, G. J., et al.: Algorithm description of joint exploration test model 7 (jem 7). In: Joint Video Exploration Team (JVET) of ITU-T SG (2017)
9. Joint-Video-Exploration-Team: JEVT software repository. https://jvet.hhi.fraunhofer.de/svn/svn_HMJEMSoftware/branches/HM-16.6-JEM-7.1-dev/ (2019). Accessed March 2019
10. Ben Jdidia, S., Belghith, F., Sallem, A., et al.: Hardware implementation of PSO-based approximate DST transform for VVC standard. J. Real-Time Image Process. **19**(1), 87–101 (2022)
11. Mert, A.C., Kalali, E., Hamzaoglu, I.: High performance 2d transform hardware for future video coding. IEEE Trans. Consum. Electron. **63**(2), 117–125 (2017)
12. Ben Jdidia, S., Kammoun, A., Belghith, F., et al.: Hardware implementation of 1-d 8-point adaptive multiple transform in post-hevc standard. In: 18th International Conference on Sciences and Techniques of Automatic Control and Computer Engineering (STA), IEEE, pp. 146–151 (2017)
13. Garrido, M.J., Pescador, F., Chavarrias, M., et al.: A high performance FPGA-based architecture for the future video coding adaptive multiple core transform. IEEE Trans. Consum. Electron. **64**(1), 53–60 (2018)
14. Garrido, M.J., Pescador, F., Chavarrías, M., et al.: A 2-D multiple transform processor for the versatile video coding standard. IEEE Trans. Consum. Electron. **65**(3), 274–283 (2019)
15. Kammoun, A., Ben Jdidia, S., Belghith, F., et al.: 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), IEEE, pp. 1–6 (2018)
16. Kammoun, A., Hamidouche, W., Belghith, F., et al.: Hardware design and implementation of adaptive multiple transforms for the versatile video coding standard. IEEE Trans. Consum. Electron. **64**(4), 424–432 (2018)
17. Fan, Y., Zeng, Y., Sun, H., et al.: A pipelined 2d transform architecture supporting mixed block sizes for the VVC standard. IEEE Trans. Circuits Syst. Video Technol. **30**(9), 3289–3295 (2019)
18. Sau, C., Ligas, D., Fanni, T., et al.: Reconfigurable adaptive multiple transform hardware solutions for versatile video coding. IEEE Access **7**, 153258–153268 (2019)
19. Yibo, F., Jiro, K., Heming, S., et al.: A minimal adder-oriented 1d dst-vii/dct-viii hardware implementation for vvc standard. In: 2019 32nd IEEE International System-on-Chip Conference (SOCC), IEEE, pp. 176–180 (2019)
20. Kammoun, A., Hamidouche, W., Philippe, P., et al.: Forward-inverse 2D hardware implementation of approximate transform core for the VVC standard. IEEE Trans. Circuits Syst. Video Technol. **30**(11), 4340–4354 (2019)
21. Ben Jdidia, S., Belghith, F., Jridi, M., et al.: A multicriteria optimization of the discrete sine transform for versatile video coding standard. Signal Image Video Process. **16**(2), 329–337 (2022)
22. Zeng, Y., Sun, H., Katto, J., et al.: Approximated reconfigurable transform architecture for vvc. In: 2021 IEEE International Symposium on Circuits and Systems (ISCAS), IEEE, pp. 1–5 (2021)
23. Mentor-modelsim: Functional-verification-tool-web. https://www.mentor.com/products/fv/modelsim (2019)
24. Intel-FPGA: Download-center. https://www.altera.com/downloads/downloadcenter. html (2019)
25. Intel/Altera: Intel-arria-10-device-overview. https://www.altera.com/documentation/sam1403480274650.html (2018)

Publisher's Note Springer Nature remains neutral with regard to jurisdictional claims in published maps and institutional affiliations.

Springer Nature or its licensor holds exclusive rights to this article under a publishing agreement with the author(s) or other rightsholder(s); author self-archiving of the accepted manuscript version of this article is solely governed by the terms of such publishing agreement and applicable law.

Sonda Ben Jdidia was born in Sfax, Tunisia, in 1993. She received her Electrical Engineering degree and her Ph.D degree from the National Engineering School of Sfax (ENIS), Tunisia, in 2017 and 2021 respectively. She is currently a member of research team at Laboratory of Electronics and Information Technology LETI-ENIS, University of Sfax. Her research interests include video coding and compression, potential video coding standards and codecs, FPGA hardware implementation.

Fatma Belghith was born in 1988, in Sfax, Tunisia. She received her degree in Electrical Engineering and her Ph.D degrees from the National School of Engineering (ENIS), Sfax, Tunisia, in 2012 and 2016 respectively. She is currently a member of research team at Laboratory of Electronics and Information Technology LETI-ENIS, University of Sfax. She is an Assistant Professor at Faculty of Science and Technology of Sidi Bouzid, Tunisia. Her current research interests include video coding with emphasis on HEVC standard and beyond, hardware implementation using FPGA and embedded systems technology

Nouri Masmoudi was born in Sfax, Tunisia, in 1955. He received electrical engineering degree from the Faculty of Sciences and Techniques - Sfax, Tunisia, in 1982, the DEA degree from the National Institute of Applied Sciences-Lyon and University Claude Bernard-Lyon, France in 1984. From 1986 to 1990, he prepared his thesis at the laboratory of Power Electronics (LEP) at the National School Engineering of Sfax (ENIS). He received his PhD degree from the National School Engineering of Tunis (ENIT), Tunisia in 1990. From 1990 to 2000, he was an assistant professor at the electrical engineering department -ENIS. Since 2000, he has been an associate professor and head of the group ‘Circuits and Systems’ in the Laboratory of Electronics and Information Technology. Since 2003, He is responsible for the Electronic Master Program at ENIS. His research activities have been devoted to several topics: Design, Telecommunication, Embedded systems and Information technology. Video Coding (Motion Estimation, Mode Decision, H.264 Standard), Image Processing (Wavelet Image Compression, Subband Image Coding, Image Interpolation, Denoising).