An FPGA Implementation of Future Video Coding 2D Transform

Ahmet Can Mert, Ercan Kalali, Ilker Hamzaoglu
Faculty of Engineering and Natural Sciences, Sabanci University
34956 Tuzla, Istanbul, Turkey
{ahmetcanmert, ercankalali, hamzaoglu}@sabanciuniv.edu

***Abstract*— Future Video Coding (FVC) is a new international video compression standard offering much better compression efficiency than previous video compression standards at the expense of much higher computational complexity. In this paper, an FPGA implementation of FVC 2D transform is proposed. The proposed FVC 2D transform hardware can perform 2D DCT-II, DCT-V, DCT-VIII, DST-I, DST-VII operations for 4x4 and 8x8 transform units. It uses two reconfigurable datapaths for all 1D transforms. It implements multiplications with constants using DSP blocks in FPGA. The proposed FPGA implementation, in the worst case, can process 54 8K Ultra HD (7680x4320) video frames per second. The proposed FPGA implementation has up to 29% less energy consumption than the FPGA implementation of FVC 2D transform hardware in the literature.**

Keywords—Discrete Cosine Transform, Discrete Sine Transform, FVC, Hardware Implementation, FPGA.

I. Introduction

ITU and ISO standardization organizations are jointly developing a new international video compression standard called Future Video Coding (FVC) [1]-[3]. FVC will provide much better compression efficiency than the previous High Efficiency Video Coding (HEVC) video compression standard at the expense of much more computational complexity [4]-[9].

HEVC uses Discrete Cosine Transform (DCT) / Inverse Discrete Cosine Transform (IDCT). In addition, it uses Discrete Sine Transform (DST) / Inverse Discrete Sine Transform (IDST) for 4x4 intra prediction in certain cases. DCT and DST have high computational complexity, and they are heavily used in an HEVC encoder [10]. DCT and DST operations account for 11% of the computational complexity of an HEVC video encoder. They account for 25% of the computational complexity of an all intra HEVC video encoder.

HEVC uses DCT-II and DST-VII. It uses 4x4, 8x8, 16x16, 32x32 Transform Unit (TU) sizes. In order to improve the compression efficiency, FVC uses DCT-II, DCT-V, DCT-VIII, DST-I, DST-VII, and it uses 4x4, 8x8, 16x16, 32x32, 64x64 TU sizes [11], [12]. Therefore, FVC transform operations have much higher computational complexity than HEVC transform operations.

In this paper, an FPGA implementation of FVC 2D transform is proposed. The proposed hardware performs 2D DCT-II, DCT-V, DCT-VIII, DST-I, and DST-VII operations for 4x4 and 8x8 TU sizes by applying 1D transforms in vertical and horizontal directions. It processes two 4x4 TUs in parallel or one 8x8 TU. Therefore, it can calculate 8 DCT/DST coefficients per clock cycle. The proposed hardware uses one reconfigurable datapath for all 1D column transforms and one reconfigurable datapath for all 1D row transforms.

Xilinx FPGAs have built-in full-custom DSP blocks which can perform constant multiplications faster and with less energy than adders and shifters. A DSP block can be used to perform different constant multiplications by providing proper constant value to its input. Therefore, it is more efficient to implement constant multiplications using DSP blocks instead of using adders and shifters in an FPGA implementation.

Therefore, the proposed hardware implements multiplications with constants using DSP blocks in FPGA instead of using adders and shifters. It uses data gating to reduce energy consumption. The proposed FVC 2D transform hardware is implemented using Verilog HDL. The proposed FPGA implementation, in the worst case, can process 54 8K Ultra HD (7680x4320) video frames per second.

Two FVC 2D transform hardware are proposed in the literature [13]. They implement FVC 2D transform operations for 4x4 and 8x8 TU sizes by applying 1D transforms in vertical and horizontal directions. The baseline hardware uses separate datapaths for each 1D transform. The reconfigurable hardware uses two reconfigurable datapaths for all 1D transforms. Since it is more efficient to implement constant multiplications using adders and shifters instead of using multipliers in an ASIC implementation, they both implement multiplications with constants using adders and shifters. Therefore, the proposed FPGA implementation of FVC 2D transform has up to 29% and 59% less energy consumption than FPGA implementations of baseline and reconfigurable hardware, respectively.

Several HEVC 2D DCT hardware are proposed in the literature [14]-[16]. Their FPGA implementations are compared with the proposed FPGA implementation of FVC 2D transform in section IV.

The rest of the paper is organized as follows. In Section II, FVC transform algorithms are explained. In Section III, the proposed FVC 2D transform hardware is explained. The implementation results are given in Section IV. Finally, Section V presents the conclusion.

TABLE I
DCT-II, DCT-V, DCT-VIII, DST-I, DST-VII BASIS FUNCTIONS

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

II. FVC TRANSFORM ALGORITHM

Basis functions for 1D DCT-II, DCT-V, DCT-VIII, DST-I and DST-VII for an NxN block are shown in Table I, where *i, j = 0, 1, … , N-1.*

HEVC uses DCT-II and DST-VII. It uses 4x4, 8x8, 16x16, 32x32 TU sizes for DCT [16]. It also uses DST for 4x4 intra prediction in certain cases. HEVC performs 2D transform operation by applying 1D transforms in vertical and horizontal directions. The coefficients in HEVC 1D transform matrices are derived from DCT-II and DST-VII basis functions. However, integer coefficients are used for simplicity. HEVC 1D DCT-II and DST-VII matrices for 4x4 TU size are shown in (1) and (2).

In order to improve the compression efficiency, FVC uses DCT-II, DCT-V, DCT-VIII, DST-I, DST-VII, and it uses 4x4, 8x8, 16x16, 32x32, 64x64 TU sizes. FVC also performs 2D transform operation by applying 1D transforms in vertical and horizontal directions. The coefficients in the FVC 1D transform matrices are derived from DCT and DST basis functions. However, integer coefficients are used for simplicity. FVC 1D transform matrices for 4x4 TU size are shown in (3)-(7).

HEVC uses the same transform type for vertical and horizontal 1D transforms for performing a 2D transform. However, FVC may use different transform types for vertical and horizontal 1D transforms. It uses an adaptive multiple transform (AMT) scheme to determine 1D transform types. AMT is enabled or disabled for each coding unit (CU). When AMT is disabled for a CU, only DCT-II is used for this CU. When AMT is enabled for a CU, 1D transform types for vertical and horizontal directions are selected based on prediction type, intra or inter prediction, for this CU.

In FVC, as shown in Table II, three different 1D transform sets are defined. Each transform set consists of two transform types. In intra prediction, transform set is selected based on intra prediction mode. In inter prediction, transform set 2 is used for all inter prediction modes.

$$DCT-II_{4x4} = \begin{bmatrix} 64 & 64 & 64 & 64 \\ 83 & 36 & -36 & -83 \\ 64 & -64 & -64 & 64 \\ 36 & -83 & 83 & -36 \end{bmatrix} \tag{1}$$

$$DST-VII_{4x4} = \begin{bmatrix} 29 & 55 & 74 & 84 \\ 74 & 74 & 0 & -74 \\ 84 & -29 & -74 & 55 \\ 55 & -84 & 74 & -29 \end{bmatrix} \tag{2}$$

$$DCT-II_{4x4} = \begin{bmatrix} 256 & 256 & 256 & 256 \\ 334 & 139 & -139 & -334 \\ 256 & -256 & -256 & 256 \\ 139 & -334 & 334 & -139 \end{bmatrix} \tag{3}$$

$$DCT-V_{4x4} = \begin{bmatrix} 194 & 274 & 274 & 274 \\ 274 & 241 & -86 & -349 \\ 274 & -86 & -349 & 241 \\ 274 & -349 & 241 & -86 \end{bmatrix} \tag{4}$$

$$DCT-VIII_{4x4} = \begin{bmatrix} 336 & 296 & 219 & 117 \\ 296 & 0 & -296 & -296 \\ 219 & -296 & -117 & 336 \\ 117 & -296 & 336 & -219 \end{bmatrix} \tag{5}$$

$$DST-I_{4x4} = \begin{bmatrix} 190 & 308 & 308 & 190 \\ 308 & 190 & -190 & -308 \\ 308 & -190 & -190 & 308 \\ 190 & -308 & 308 & -190 \end{bmatrix} \tag{6}$$

$$DST-VII_{4x4} = \begin{bmatrix} 117 & 219 & 296 & 336 \\ 296 & 296 & 0 & -296 \\ 336 & -117 & -296 & 219 \\ 219 & -336 & 296 & -117 \end{bmatrix} \tag{7}$$

TABLE II
TRANSFORM SETS

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

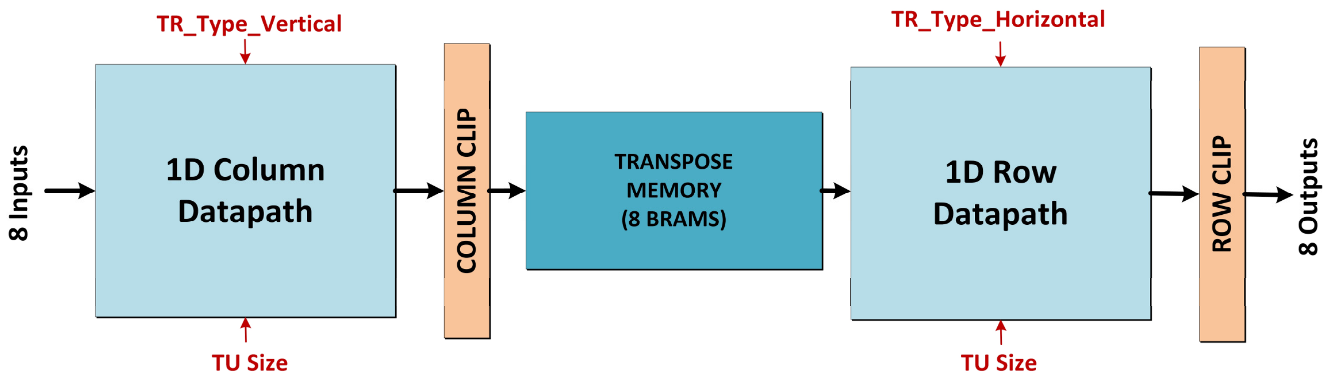

Fig. 1. Proposed FVC 2D Transform Hardware

III. Proposed FVC 2D Transform Hardware

The proposed FVC 2D transform hardware for 4x4 and 8x8 TU sizes is shown in Fig. 1. The proposed hardware performs 2D DCT/DST by first performing 1D DCT/DST on the columns of a TU, and then performing 1D DCT/DST on the rows of the TU. After 1D column DCT/DST, the resulting transformed coefficients are stored in a transpose memory, and they are used as input for 1D row DCT/DST.

The proposed hardware uses one reconfigurable datapath for implementing all 1D column DCT/DST types and one reconfigurable datapath for implementing all 1D row DCT/DST types. The proposed hardware calculates eight transformed coefficients per clock cycle for both 4x4 and 8x8 TU sizes. When the proposed hardware processes 8x8 TU size, eight inputs are eight residuals in one column of an 8x8 TU. When it processes 4x4 TU size, eight inputs are four residuals in one column of a 4x4 TU and four residuals in one column of another 4x4 TU.

An N-point 1D transform can be performed by performing two N/2-point 1D transforms with some preprocessing for FVC DCT-II and DST-I. FVC DCT-V, DCT-VIII and DST-VII do not have this property. Since the proposed hardware uses one reconfigurable datapath for all 1D transforms, N-point DCT-II and DST-I are also performed by performing one N-point DCT-II and DST-I same as DCT-V, DCT-VIII, DST-VII.

The proposed reconfigurable 1D column datapath is shown in Fig. 2. Column and row datapath have the same hardware architecture. Since each 1D DCT/DST uses different transform coefficients, different constant multiplication operations should be performed for each 1D DCT/DST. Xilinx FPGAs have built-in full-custom DSP blocks which can perform constant multiplications faster and with less energy than adders and shifters. A DSP block can be used to perform different constant multiplications by providing proper constant value to its input. Therefore, the proposed hardware implements constant multiplications using DSP blocks in FPGA instead of using adders and shifters.

For implementing constant multiplications, 8x8=64 DSP blocks are used in 1D column datapath and 8x8=64 DSP blocks are used in 1D row datapath. In the column datapath, each transform input sent to 8 DSP blocks in the same column. Each DSP block takes one transform input and one transform coefficient as input, and it performs constant multiplication. 64 and 32 DSP blocks are used for one 8x8 TU and two 4x4 TUs, respectively. Since the proposed hardware can perform 5 different DCT/DST operations for 2 different TU sizes, a multiplexer is used at the input of each DSP block to select proper transform coefficient. 1D transform type (TR_Type_ Vertical) and TU size (TU_size) are used as select signals for the multiplexers.

In order to calculate each output of 1D DCT/DST for an 8x8 TU, outputs of DSP blocks in the same row are added. 8 DSP blocks in the same row and their adder tree structure is shown in Fig. 2. 8 DSP blocks in the other rows have the same structure. In the figure, only one of them is shown for simplicity.

In order to calculate each output of 1D DCT/DST for a 4x4 TU, outputs of DSP blocks in the same row are added. Since two 4x4 TUs are processed in parallel, outputs of first 4 DSP blocks in the same row are added for the first 4x4 TU. Outputs of last 4 DSP blocks in the same row are added for the second 4x4 TU.

In order to reduce energy consumption of the proposed hardware, data gating is used for the inputs of DSP blocks in 1D column datapath and 1D row datapath. 1D DCT/DST operation for an 8x8 TU uses 64 DSP blocks. 1D DCT/DST operation for a 4x4 TU uses 16 DSP blocks. Therefore, when two 4x4 TUs are processed in parallel, the input registers of 32 DSP blocks are not updated. This prevents unnecessary switching activities in the DSP blocks and therefore reduces energy consumption.

As shown in Fig. 3, the transpose memory is implemented using 8 Block RAMs (BRAM). 4 and 8 BRAMs are used for 4x4 and 8x8 TU sizes, respectively. Since a BRAM address can store 32-bits and one transformed coefficient of 1D column DCT/DST is 16-bits, each BRAM address can store two transformed coefficients. When the proposed hardware processes 4x4 and 8x8 TU size, each BRAM address stores two and one transformed coefficients, respectively.

In the Fig. 3, the numbers in the each box show the BRAM that coefficient is stored. The results of 1D column DCT/DST are generated column by column. For 8x8 TU size, first, the coefficients in column 0 (C0) are generated in a clock cycle and stored in 8 different BRAMs. Then, the coefficients in column 1 (C1) are generated in the next clock cycle and stored in 8 different BRAMs using a rotating addressing scheme.

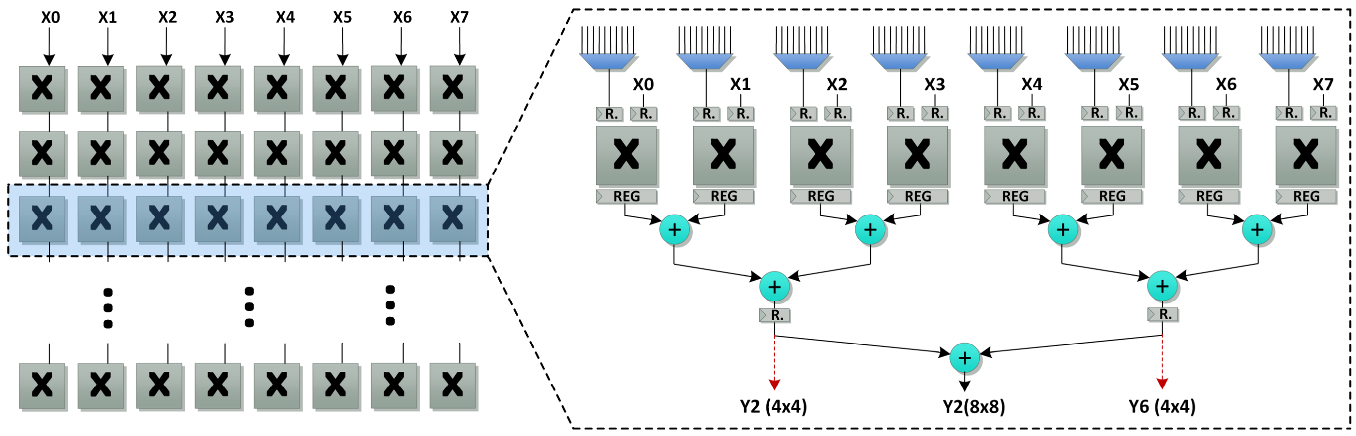

Fig. 2. 1D Column/Row Datapath

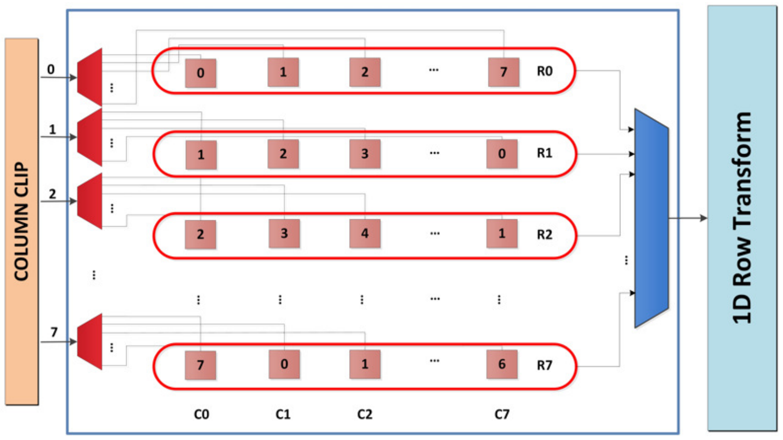

Fig. 3. Transpose Memory

This continuous until the coefficients in column 7 (C7) are generated and stored in 8 different BRAMs using the rotating addressing scheme. This ensures that the 8 coefficients necessary for 1D row DCT/DST in a clock cycle can always be read in one clock cycle from 8 different BRAMs.

Column clip and row clip hardware are used to scale the outputs of 1D column DCT/DST and 1D row DCT/DST to 16 bits, respectively. Column clip hardware shifts 1D column DCT/DST outputs right by 3 and 4 bits for 4x4 and 8x8 TU sizes, respectively. Row clip hardware shifts 1D row DCT/DST outputs right by 10 and 11 bits for 4x4 and 8x8 TU sizes, respectively.

The proposed hardware performs 1D DCT/DST for 4x4 and 8x8 TU sizes in 4 and 8 clock cycles, respectively. 1D column DCT/DST and 1D row DCT/DST operations are pipelined. While 1D row DCT/DST for current TU is performed, 1D column DCT/DST for next TU is also performed. Because of the input data loading and pipeline stages, the proposed hardware starts generating the results of 1D row DCT/DST in 16 clock cycles. It then continues generating the results row by row in every clock cycle until the end of the last TU in the video frame without any stalls.

IV. Implementation Results

The proposed FVC 2D transform hardware is implemented using Verilog HDL. The Verilog RTL codes are verified with RTL simulations. RTL simulation results matched results of FVC 2D transform implementation in Joint Exploration Test Model (JEM) 4.0 reference software encoder [2]. The Verilog RTL codes are synthesized and mapped to a Xilinx XC6VLX550T FF1759 FPGA with speed grade 2 using Xilinx ISE 14.7.

The FPGA implementation is verified to work at 222 MHz by post place and route simulations. Post place and route simulation results matched results of FVC 2D transform implementation in JEM 4.0 reference software encoder. Therefore, it can process 54 8K Ultra HD (7680x4320) video frames per second. The FPGA implementation uses 3332 LUTs, 2082 DFFs, 128 DSP Blocks and 8 BRAMs.

An HEVC 2D DCT hardware for all TU sizes is proposed in [16]. In this paper, for fair comparison, this hardware is implemented for 4x4 and 8x8 TU sizes by using DSP blocks in FPGA for implementing multiplications with constants. The FPGA implementation uses 2069 LUTs, 665 DFFs, 44 DSP Blocks and 8 BRAMs. The FPGA implementation is verified to work at 222 MHz by post place and route simulations. Therefore, it can process 54 8K Ultra HD (7680x4320) video frames per second. In this paper, this FPGA implementation is called as HEVC 2D DCT hardware.

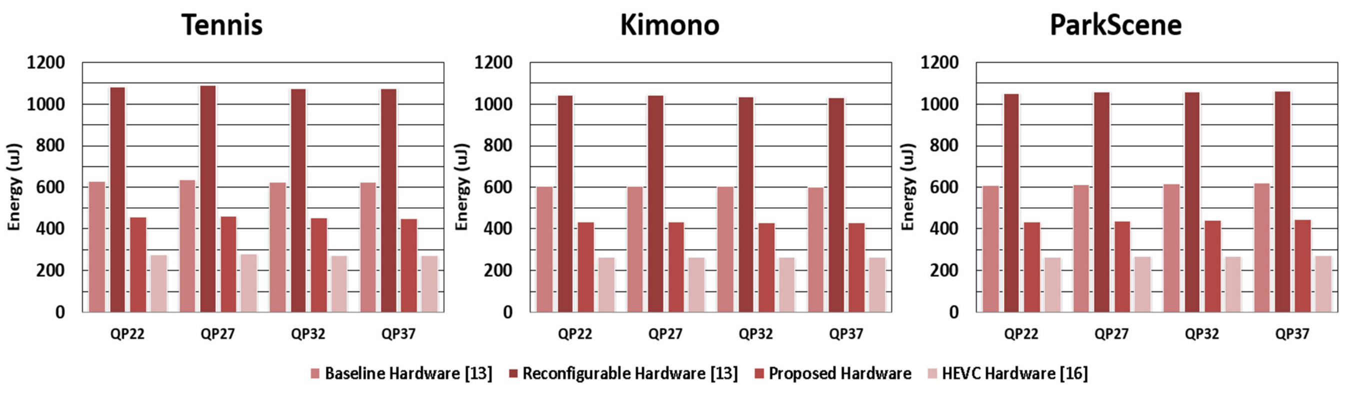

Fig. 4. Energy Consumptions of FVC 2D Transform and HEVC 2D DCT Hardware

TABLE III
MULTIPLIER, ADDER, MUX AMOUNTS IN 1D DATAPATH

	HEVC Hardware [16]	Proposed FVC Hardware
Multiplier	22	64
Adder	28	56
10-bit 2-to-1 MUX	-	342

Number of multipliers, adders and multiplexers used in 1D (column or row) datapath of the proposed FVC 2D transform hardware and the HEVC 2D DCT hardware are shown in Table III. Since FVC 2D transform operations have much higher computational complexity than HEVC 2D DCT operations, the proposed FVC reconfigurable 1D column/row datapath uses more multipliers, adders and multiplexers than the column/row datapath in the HEVC 2D DCT hardware.

Power consumptions of the FPGA implementations are estimated using Xilinx XPower Analyzer tool. Post place and route timing simulations are performed for Tennis, Kimono and Park Scene (1920x1080) videos at 100 MHz [17], and signal activities are stored in VCD files. These VCD files are used for estimating power consumptions of the FPGA implementations.

Energy consumptions of the FPGA implementations of FVC baseline and reconfigurable 2D transform hardware proposed in [13], the proposed FVC 2D transform hardware and the HEVC 2D DCT hardware for one frame of each video are shown in Fig. 4. Since FVC 2D transform operations have much higher computational complexity than HEVC 2D DCT operations, the proposed FVC 2D transform hardware consumes more energy than the HEVC 2D DCT hardware. Since the proposed FVC 2D transform hardware implements multiplications with constants using DSP blocks in FPGA instead of using adders and shifters, the proposed FPGA implementation of FVC 2D transform has up to 29% and 59% less energy consumption than FPGA implementations of FVC baseline and reconfigurable 2D transform hardware, respectively.

The proposed FPGA implementation of FVC 2D transform is compared with FPGA implementations of FVC 2D transform hardware and HEVC 2D DCT hardware in the literature [13]-[16]. The comparison is shown in Table IV. Since the FVC baseline and reconfigurable 2D transform hardware proposed in [13] implement multiplications with constants using adders and shifters, their FPGA implementations are slower and use more Slices, LUTs and DFFs than the proposed FPGA implementation of FVC 2D transform. However, they do not use any DSP blocks.

Since FVC 2D transform operations have much higher computational complexity than HEVC 2D DCT operations, the proposed FPGA implementation of FVC 2D transform is slower and uses more FPGA resources than the FPGA implementations of HEVC 2D DCT hardware proposed in [14]-[16]. Since HEVC 2D DCT hardware proposed in [14] performs DCT-II for TU sizes up to 32x32, its FPGA implementation uses more FPGA resources.

V. CONCLUSION

In this paper, an FPGA implementation of FVC 2D transform is proposed. The proposed hardware implements multiplications with constants using DSP blocks in FPGA. The proposed FPGA implementation can process 54 8K Ultra HD (7680x4320) video frames per second. The proposed FPGA implementation has up to 29% less energy consumption than the FPGA implementation of FVC 2D transform hardware in the literature.

ACKNOWLEDGEMENT

This research was supported in part by the Scientific and Technological Research Council of Turkey (TUBITAK) under the contract 115E290.

REFERENCES

[1] J. Chen, Y. Chen, M. Karczewicz, X. Li, H. Liu, L. Zhang, X. Zhao, "Coding tools investigation for next generation video coding", *ITU-T SG16 COM16–C806*, Feb. 2015.

[2] J. Chen, E. Alshina, G. J. Sullivan, J. R. Ohm, J. Boyce, "Algorithm Description of Joint Exploration Model 4", *JVET-D1001*, Oct. 2016.

[3] S. H. Park, E. S. Jang, "An Efficient Motion Estimation Method for QTBT Structure in JVET Future Video Coding", *Data Compression Conference (DCC)*, Apr. 2017.

[4] ITU-T and ISO/IEC, High Efficiency Video Coding, *ITU-T Rec. H.265 and ISO/IEC 23008-2 (HEVC)*, April 2013.

[5] E. Ozcan, Y. Adibelli, I. Hamzaoglu, "A High Performance Deblocking Filter Hardware for High Efficiency Video Coding", *IEEE Trans. on Consumer Electronics*, vol.59, no.3, pp.714-720, Aug. 2013.

[6] E. Ozcan, E. Kalali, Y. Adibelli, I. Hamzaoglu, "A Computation and Energy Reduction Technique for HEVC Intra Mode Decision", *IEEE Trans. on Consumer Electronics*, vol.60, no.4, pp.745-753, Nov. 2014.

TABLE IV
HARDWARE COMPARISON

	HEVC [14]	HEVC [15]	HEVC [16]	[13]		Proposed FVC
				FVC Baseline	FVC Reconfig.	
FPGA	Arria II GX	Xilinx Virtex7	Xilinx Virtex 6	Xilinx Virtex 6	Xilinx Virtex 6	Xilinx Virtex 6
Slices	-	-	810	7930	5292	1223
LUTs	7300	2478	2069	27144	17173	3332
DFFs	-	-	665	12309	4571	2082
DSP Blocks	128	64	44	-	-	128
Max. Freq. (MHz)	200	289	222	167	143	222
Frames per Second	-	70 3840x2160	54 7680x4320	40 7680x4320	35 3840x2160	54 7680x4320
Throughput (pixels/cycle)	-	-	8	8	8	8
Max Bit Length	25	25	25	27	27	27
Transform Unit Size	4, 8, 16, 32	4, 8	4, 8	4, 8	4, 8	4, 8
Transform Type	DCT-II	DCT-II	DCT-II	DCT-II, DCT-V, DCT-VIII, DST-I, DST-VII	DCT-II, DCT-V, DCT-VIII, DST-I, DST-VII	DCT-II, DCT-V, DCT-VIII, DST-I, DST-VII
Transform	2D	2D	2D	2D	2D	2D

[7] A. C. Mert, E. Kalali, I. Hamzaoglu, "Low Complexity HEVC Sub-Pixel Motion Estimation Technique and Its Hardware Implementation", *IEEE Int. Conference on Consumer Electronics – Berlin*, Sept. 2016.

[8] E. Kalali, Y. Adibelli, I. Hamzaoglu, "A High Performance and Low Energy Intra Prediction Hardware for High Efficiency Video Coding", *Int. Conference on Field Programmable Logic and Applications*, Aug. 2012.

[9] E. Kalali, E. Ozcan, O. M. Yalcinkaya, I Hamzaoglu, "A Low Energy HEVC Inverse Transform Hardware", *IEEE Trans. on Consumer Electronics*, vol. 60, no. 4, pp. 754-761, Nov. 2014.

[10] J. Vanne, M. Viitanen, T. D. Hamalainen, A. Hallapuro, "Comparative Rate-Distortion-Complexity Analysis of HEVC and AVC Video Codecs", *IEEE Trans. on Circuits and Systems for Video Technology*, vol. 22, no. 12, pp.1885-1898, Dec. 2012.

[11] X. Zhao, J. Chen, M. Karczewicz, L. Zhang, X. Li, W. Chien, "Enhanced Multiple Transform for Video Coding", *Data Compression Conference*, April 2016.

[12] T. Biatek, V. Lorcy, P. Castel, P. Philippe, "Low-Complexity Adaptive Multiple Transform for post-HEVC Video Coding", *Picture Coding Symposium*, Dec. 2016.

[13] A. C. Mert, E. Kalali, I Hamzaoglu, "High Performance 2D Transform Hardware for Future Video Coding", *IEEE Trans. on Consumer Electronics*, vol. 62, no. 2, May 2017.

[14] G. Pastuszak, "Hardware architecture for the H.265/HEVC discrete cosine transform", *IET Image Processing*, vol. 9, no. 6, pp. 468-477, June 2015.

[15] M. Chen, Y. Zhang, 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.

[16] E. Kalali, A. C. Mert, I Hamzaoglu, "A Computation and Energy Reduction Technique for HEVC Discrete Cosine Transform", *IEEE Trans. on Consumer Electronics*, vol. 62, no. 2, pp. 166-174, May 2016.

[17] K. Suehring, X. Li, "JVET Common Test Conditions and Software Reference Configurations", *JVET-B1010*, Feb. 2016.