A Low Power Versatile Video Coding (VVC) Fractional Interpolation Hardware

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

***Abstract*—Fractional interpolation in Versatile Video Coding (VVC) standard has much higher computational complexity than fractional interpolation in previous video compression standards. In this paper, a low power VVC fractional interpolation hardware is designed and implemented using Verilog HDL. The proposed hardware is the first VVC fractional interpolation hardware in the literature. It interpolates necessary fractional pixels for 1/16 pixel accuracy for all prediction unit sizes. The proposed VVC fractional interpolation hardware, in the worst case, can process 40 full HD (1920x1080) frames per second. It has up to 17% less power consumption than original VVC fractional interpolation hardware.**

Keywords—VVC, Fractional Interpolation, Hardware Implementation, FPGA, Low Power

I. INTRODUCTION

ITU and ISO are developing a new international video compression standard called Versatile Video Coding (VVC) [1]-[6]. VVC will have higher compression efficiency than High Efficiency Video Coding (HEVC) standard at the expense of much higher computational complexity [7]-[11].

HEVC standard uses 3 different 8-tap FIR filters for fractional interpolations (FI) and provides 1/4 fractional pixel accuracy. However, VVC standard uses 15 different 8-tap FIR filters for fractional interpolations and provides 1/16 fractional pixel accuracy. Therefore, VVC fractional interpolation has much higher computational complexity than HEVC fractional interpolation.

In this paper, a low power VVC fractional interpolation hardware for all prediction unit (PU) sizes is proposed. The proposed hardware interpolates all necessary fractional pixels for an 8x8 PU. For larger PU sizes, the PU is divided into 8x8 blocks, and the blocks are interpolated separately.

The proposed hardware calculates a common offset for 15 different FIR filter equations using the same input pixels in order to reduce number of constant coefficient multiplications necessary for fractional interpolation. It also calculates common sub-expressions in different FIR filter equations once and uses the results in necessary equations. Hcub multiplierless constant multiplication (MCM) algorithm [12] is also used in the proposed hardware in order to reduce number and size of the adders.

The proposed VVC fractional interpolation hardware is implemented in Verilog HDL. The Verilog RTL code is verified to work at 200 MHz in a Xilinx Virtex 7 FPGA. The proposed VVC fractional interpolation hardware, in the worst case, can process 40 full HD (1920x1080) frames per second. It has up to 17% less power consumption than original VVC fractional interpolation hardware.

The proposed hardware is the first VVC fractional interpolation hardware in the literature. Several HEVC fractional interpolation hardware implementations are proposed in the literature [13]-[15]. In [13], common sub-expressions in FIR filters are calculated once and used in all equations. It also uses Hcub MCM algorithm to implement constant multiplications. The implementation in [14] uses coarse-grained reconfigurable datapaths to implement filter equations. A high-throughput FI hardware is proposed for HEVC encoder in [15]. In Section III, VVC fractional interpolation hardware proposed in this paper is compared with them.

The rest of the paper is organized as follows. In Section II, VVC fractional interpolation algorithm is explained. In Section III, the proposed low power VVC fractional interpolation hardware is presented, and its implementation results are given. Finally, Section IV presents the conclusions.

II. VVC FRACTIONAL INTERPOLATION ALGORITHM

VVC standard uses 15 different 8-tap FIR filters for fractional pixel interpolations. The coefficients of these 15 FIR filters are shown in Table I. A_{-3} – A_4 show input pixels for a filter where sub-indices represent the indices of coefficients. The F_5 8-tap FIR filter equation is shown in (1) as an example.

$$F_5 = (-A_{-3} + 4 * A_{-2} - 11 * A_{-1} + 52 * A_0 + 26 * A_1 - 8 * A_2 + 3 * A_3 - A_4) \gg 6 \quad (1)$$

Integer pixels, fractional pixels and FIR filters used to interpolate these fractional pixels are shown in Fig. 1. There are 255 fractional (half and quarter) pixels for one integer pixel. There are 15 half-pixels between two neighboring horizontal integer pixels called horizontal half-pixels. There are 15 half-pixels between two neighboring vertical integer pixels called vertical half-pixels. These 15 horizontal and 15 vertical half-pixels are interpolated from nearest integer pixels in horizontal and vertical directions, respectively, using 15 different 8-tap FIR filters. There are 15x15=225 quarter-pixels between 15 horizontal and 15 vertical half-pixels. These quarter-pixels are interpolated from nearest horizontal half-pixels using 15 different 8-tap FIR filters.

Table II shows the number of addition and shift operations required for interpolating fractional pixels by using the filters F_1 to F_8. Since the filters F_9 to F_{15} are symmetric of the filters F_1 to F_7, the number of addition and shift operations required for F_9 to F_{15} are the same as F_1 to F_7. The number of addition and shift operations required for interpolating fractional pixels in HEVC by using F_1 and F_2 are also shown in Table II. Since F_3 is symmetric of F_1, the number of addition and shift operations required for F_3 are the same as F_1. The number of addition and shift operations shows that VVC FI has much higher computational complexity than HEVC FI.

978-1-5386-8237-1/18/$31.00 ©2018 IEEE

TABLE I. VVC FI FILTER COEFFICIENTS

Filters	Coefficients							
	A_{-3}	A_{-2}	A_{-1}	A_0	A_1	A_2	A_3	A_4
1	0	1	-3	63	4	-2	1	0
2	-1	2	-5	62	8	-3	1	0
3	-1	3	-8	60	13	-4	1	0
4	-1	4	-10	58	17	-5	1	0
5	-1	4	-11	52	26	-8	3	-1
6	-1	3	-9	47	31	-10	4	-1
7	-1	4	-11	45	34	-10	4	-1
8	-1	4	-11	40	40	-11	4	-1
9	-1	4	-10	34	45	-11	4	-1
10	-1	4	-10	31	47	-9	3	-1
11	-1	3	-8	26	52	-11	4	-1
12	0	1	-5	17	58	-10	4	-1
13	0	1	-4	13	60	-8	3	-1
14	0	1	-3	8	62	-5	2	-1
15	0	1	-2	4	63	-3	1	0

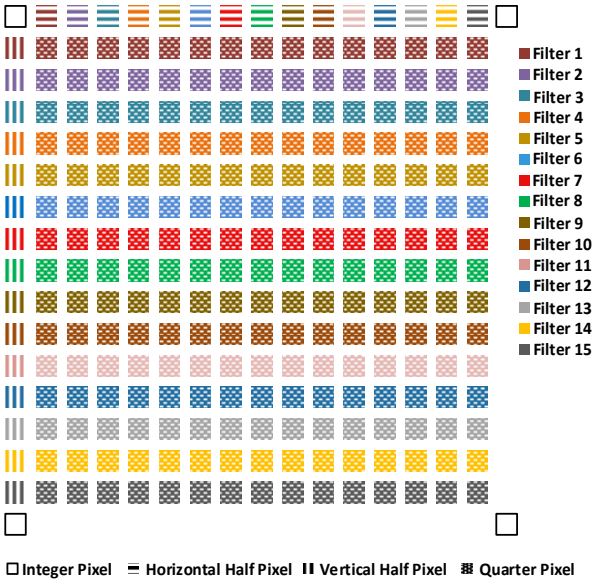

Fig. 1. Integer, half and quarter pixels.

The total number of addition and shift operations required for interpolating all fractional pixels for an 8x8 PU in HEVC and VVC are shown in Table III. Since VVC fractional interpolation calculates more fractional pixels than HEVC fractional interpolation, it has much higher computational complexity than HEVC fractional interpolation.

III. PROPOSED VVC FRACTIONAL INTERPOLATION HARDWARE

The proposed VVC fractional interpolation hardware for all PU sizes is shown in Fig. 2. The proposed hardware interpolates all fractional pixels for luma component of a PU using integer or horizontal half-pixels. The proposed hardware interpolates all necessary fractional pixels for an 8x8 PU. For larger PU sizes, the PU is divided into 8x8 blocks, and the blocks are interpolated separately. For example, a 16x16 PU is divided into four 8x8 blocks and each 8x8 block is interpolated separately.

TABLE II. ADDITION AND SHIFT AMOUNTS FOR FI FILTERS

Filters		Addition	Shift
VVC	1	7	4
	2	9	6
	3	10	7
	4	11	8
	5	14	11
	6	13	8
	7	14	11
	8	13	10
HEVC	1	11	8
	2	13	10

TABLE III. ADDITION AND SHIFT AMOUNTS FOR 8X8 PU

	Interpolated Pixels	Addition	Shift
VVC	1128	13912	10528
HEVC	17160	227656	171600

In the proposed hardware, 8x15 fractional pixels are interpolated in parallel using 15 different FIR filters in each clock cycle. The proposed hardware uses 15 pixels, integer pixels or horizontal half-pixels, to interpolate 8x15 fractional pixels in each clock cycle. The proposed hardware calculates a common offset, as shown in Table IV, for 15 different FIR filter equations in order to reduce number of constant coefficient multiplications necessary for fractional interpolation. Offset values are calculated in Offset datapath using input pixels as shown in (2).

Since common offset value is calculated, each FIR filter equation should be calculated using the filter coefficients in Table IV. Then, the resulting value should be added with common offset value. The F_5 filter equation with offset value is shown in (3) as an example. Since filters F_1 to F_7 are symmetric of filters F_9 to F_{15}, only the coefficients for filters F_1 to F_8 are shown in Table IV for simplicity.

$$offset = (-A_{-3} + 4 * A_{-2} - 8 * A_{-1} + 32 * A_0 + 32 * A_1 - 8 * A_2 + 4 * A_3 - A_4) \quad (2)$$

$$F_5 = (-3 * A_{-1} + 20 * A_0 - 6 * A_1 - A_3 + offset) \gg 6 \quad (3)$$

Each one of 15 input pixels, integer pixel or horizontal half-pixel, should be multiplied with multiple constant coefficients as explained in [13]. Table V shows constant coefficient multiplications necessary for each pixel when FIR filter equations are calculated with and without using common offset value. In Table V, A_{-6} to A_8 show 15 input pixels for filters where sub-indices represent the indices of coefficients. As shown in Table V, since constant coefficients of input pixels (A_{-4}, A_6) and (A_{-3} … A_5) are different, two different datapaths, M1 and M2, are used. When common offset value is used, number of calculated products in M1 is reduced from 4 to 2 and number of calculated products in M2 is reduced from 12 to 7. M1, M2 and Offset datapaths are shown in Fig. 3. Pixels A_{-6}, A_{-5}, A_7 and A_8 are used to calculate common sub-expressions in different equations.

Multiplications with constant coefficients are performed using adders and shifters in M1 and M2 datapaths. In order to reduce number and size of the adders, Hcub MCM algorithm

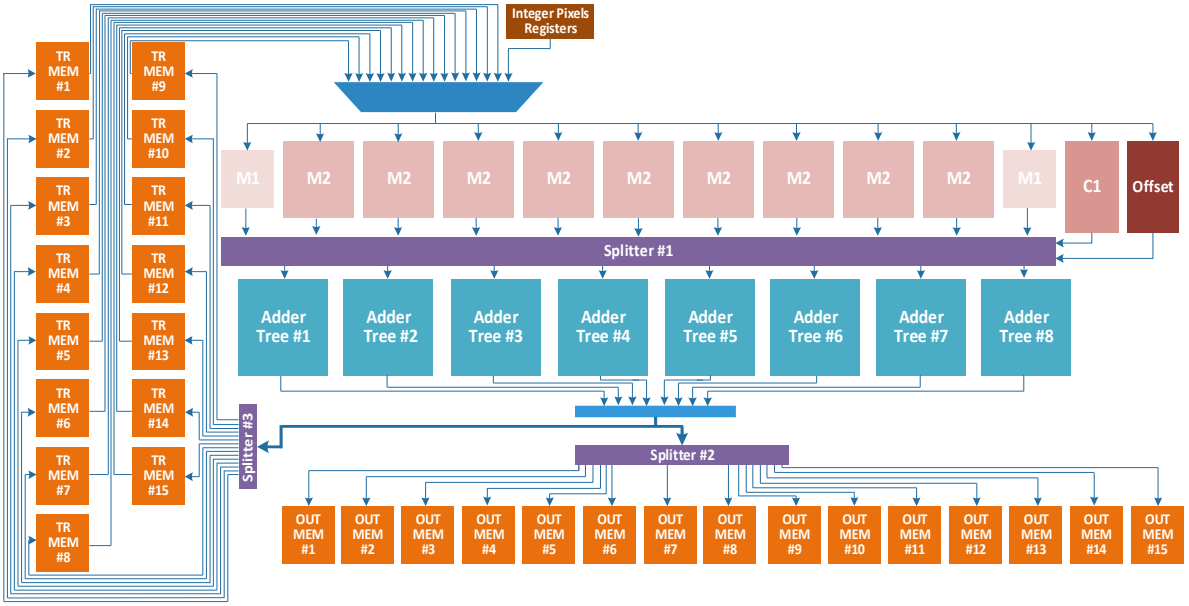

Fig. 2. Proposed VVC fractional interpolation hardware.

TABLE IV. VVC FI FILTER COEFFICIENTS WITH OFFSET

Filters	Coefficients							
	A_{-3}	A_{-2}	A_{-1}	A_0	A_1	A_2	A_3	A_4
Offset	-1	4	-8	32	32	-8	4	-1
1	1	-3	5	31	-28	6	-3	1
2	0	-2	3	30	-24	5	-3	1
3	0	-1	0	28	-19	4	-3	1
4	0	0	-2	26	-15	3	-3	1
5	0	0	-3	20	-6	0	-1	0
6	0	-1	-1	15	-1	-2	0	0
7	0	0	-3	13	2	-2	0	0
8	0	0	-3	8	8	-3	0	0

TABLE V. CONSTANT COEFFICIENTS

	Input Pixel	Constant Coefficients	Datapath	Calculated Products
Without OFFSET	A_{-4}, A_6	1,2,3,4,5,8,9,10,11	M1	3,5,9,11
	A_{-3} ... A_5	1,2,3,4,5,8,9,10, 11,13,17,26,31, 34,40,45,47,52, 58,60,62,63	M2	3,5,9,11,13,1 5,17,29,31, 45,47,63
With OFFSET	A_{-4}, A_6	1,2,3,4,5,6	M1	3,5
	A_{-3} … A_5	1,2,3,4,5,6,8,13, 15,19,20,24,26, 28,30,31	M2	3,5,7,13,15, 19,31

[12] is used. It minimizes number and size of the adders in a multiplier block which multiplies a single input with multiple constants using addition and shift operations. M1 datapath takes pixel *A* as input and calculates *3xA* and *5xA* using adders and shifters. M2 datapath takes pixel *A* as input and calculates *3xA*, *5xA*, *7xA*, *13xA*, *15xA*, *19xA* and *31xA* using adders and shifters. Offset datapath calculates eight common offset values using adders and shifters. Since 8x15 fractional pixels are calculated in parallel, eight common offset values are calculated in Offset datapath. One offset value is used for calculating 15 fractional pixels. After constant coefficient multiplications and common offset calculations are performed, fractional pixels are calculated using adder trees.

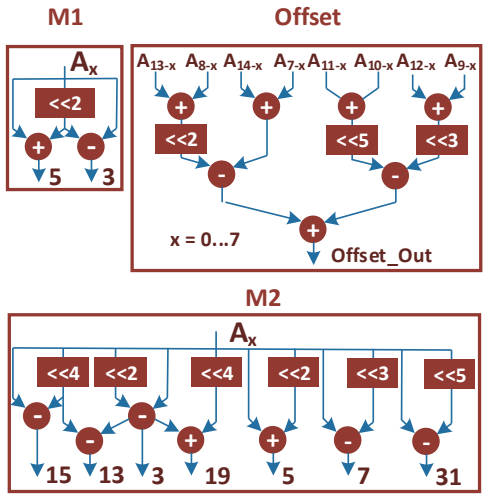

Fig. 3. M1, M2 and Offset datapaths.

As shown in Table IV, there are common sub-expressions in different equations. The expression (A_{-3} - 3∗A_{-2}) is common for FIR filters 1, 12, 13, 14 and 15. The expression (A_4 - 3∗A_3) is common for FIR filters 1, 2, 3, 4 and 15. These common sub-expressions in different equations are calculated once in C1 datapath, and the results are used in necessary equations.

As shown in Fig. 4, 30 block RAMs (BRAM) are used in the proposed hardware. 15 BRAMs are used as output memories to store fractional pixels. 15 BRAMs are used as a transpose memory to store horizontal half-pixels necessary for interpolating quarter-pixels. Each BRAM address can store eight pixels. Horizontal half-pixels are interpolated in 15 clock cycles. In each clock cycle, 8x15 horizontal half-pixels are interpolated and each 8 horizontal half-pixels are stored in 15 different BRAMs as shown in Fig. 4.

The transpose memory uses a rotating addressing scheme and the boxes with the same colors show the horizontal half-pixels stored in the same clock cycle. After all horizontal half-pixels are stored in the transpose memory in 15 clock cycles,

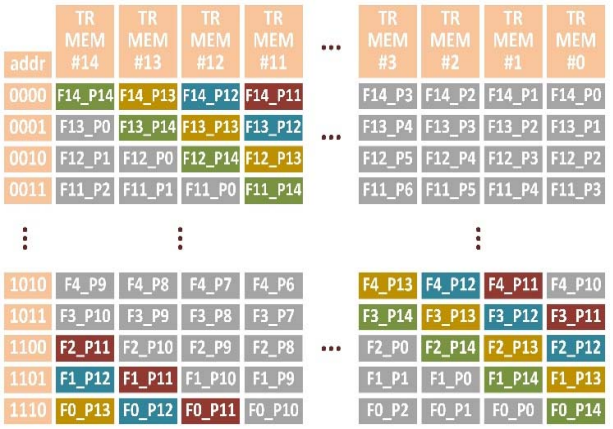

Fig. 4. Proposed transpose memory.

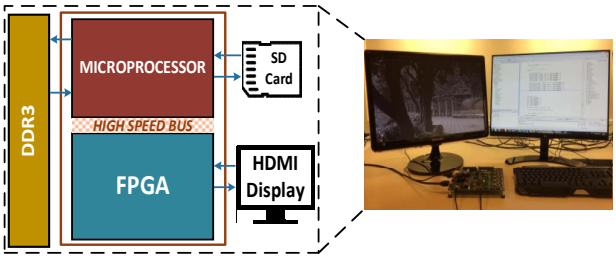

Fig. 5. Implementation of proposed VVC fractional interpolation hardware on an FPGA board.

15 pixels necessary for interpolating quarter-pixels can always be read in one clock cycle from 15 different BRAMs.

Since 255 fractional pixels should be interpolated for each integer pixel, 64x255 fractional pixels should be interpolated for an 8x8 PU. 8x7x15 extra horizontal half-pixels are necessary for interpolating quarter-pixels.

First, 8x15x15 horizontal half-pixels necessary for interpolating quarter-pixels are interpolated in 15 clock cycles, and stored in the transpose memory. Then, 8x8x15 vertical half-pixels are interpolated in 8 clock cycles. Finally, 8x8x255 quarter-pixels are interpolated in 8x15 clock cycles using horizontal half-pixels. There are four pipeline stages in the proposed hardware. Therefore, the proposed hardware interpolates the fractional pixels for an 8x8 PU in 147 clock cycles.

In this paper, an original VVC fractional interpolation hardware is also designed for comparison. This hardware implements 15 different FIR filter equations separately.

The original and proposed VVC fractional interpolation hardware for all PU sizes are implemented using Verilog HDL. The Verilog RTL codes are verified with RTL simulations. RTL simulation results matched the results of a software implementation of VVC fractional interpolation algorithm. The Verilog RTL codes are synthesized and mapped to a Xilinx VC7VX330T-3FFG1157 FPGA using Xilinx ISE 14.7. The FPGA implementations are verified with post place and route simulations. As shown in Fig. 5, FPGA implementations are also verified to work correctly on an FPGA board which includes an FPGA, dual-core microprocessor, 1 GB DRAM and interfaces such as UART and HDMI.

TABLE VI. IMPLEMENTATION RESULTS

	Original		Proposed	
Tech.	Virtex 7 FPGA	90 nm ASIC	Virtex 7 FPGA	90 nm ASIC
Slice/Gate Count	5205	64.2 K	3718	37.6 K
DFF	6408	---	3461	---
LUT	16334	---	11599	---
Memory	18 KB	18 KB	18 KB	18 KB
Max. Freq. (MHz)	208	333	200	435
Frames per Second	42 1920x1080	67 1920x1080	40 1920x1080	88 1920x1080

TABLE VII. POWER CONSUMPTION RESULTS

	Original		Proposed	
Video	Tennis	Kimono	Tennis	Kimono
Clock (mW)	52.39	52.39	27.43	27.43
Signal (mW)	162.02	218.08	144.41	193.96
Logic (mW)	141.24	194.2	109.63	151.56
BRAM (mW)	93.88	95.15	93.83	94.98
Total (mW)	449.53	559.82	375.3	467.93
Power Reduction	---	---	16.51 %	16.41 %

FPGA implementation of the original VVC fractional interpolation hardware uses 16334 LUTs, 6408 DFFs and 30 BRAMs. It can work at 208 MHz, and it can process 42 full HD (1920x1080) frames per second. FPGA implementation of the proposed VVC fractional interpolation hardware uses 11599 LUTs, 3461 DFFs and 30 BRAMs. It can work at 200 MHz, and it can process 40 full HD frames per second.

Verilog RTL codes of the original and proposed VVC fractional interpolation hardware are also synthesized to TSMC 90 nm standard cell library, and the resulting netlists are placed and routed. ASIC implementations of the original and proposed hardware use 64.2K and 37.6K gates, respectively, based on NAND (2x1) gate area excluding on-chip memory. ASIC implementations of the original and proposed hardware can work at 333 and 435 MHz, respectively, and they can process 67 and 88 full HD frames per second, respectively. The implementation results are shown in Table VI.

Power consumptions of the original and proposed hardware are estimated using Xilinx XPower Analyzer tool. Post place and route timing simulations are performed for Tennis and Kimono (1920x1080) video frames at 100 MHz [16]. The signal activities of timing simulations are stored in VCD files, and they are used for estimating the power consumptions of FPGA implementations. The power consumption results for one frame of each video are shown in Table VII. The proposed VVC fractional interpolation hardware has up to 17% less power consumption than original VVC fractional interpolation hardware.

TABLE VIII. HARDWARE COMPARISON

	[13]	[14]	[15]	Proposed
Tech.	90 nm	150 nm	90 nm	90 nm
Gate Count	28.5 K	16.6 K	224 K	37.6 K
Memory	---	1.2 KB	---	18 KB
Max. Freq. (MHz)	200	312	333	435
Frames per Second	30 3840x2160	30 3840x2160	30 1920x1080	88 1920x1080
Standard	HEVC	HEVC	HEVC	VVC

Comparison of the proposed VVC fractional interpolation hardware with HEVC fractional interpolation hardware in the literature is shown in Table VIII. Since VVC fractional interpolation has much higher computational complexity than HEVC fractional interpolation, the proposed hardware has larger area and lower performance than HEVC fractional interpolation hardware.

IV. CONCLUSION

In this paper, a low power VVC fractional interpolation hardware for all PU sizes is proposed. It is the first VVC fractional interpolation hardware in the literature. The proposed VVC fractional interpolation hardware can process 40 full HD (1920x1080) frames per second on a Xilinx Virtex 7 FPGA. It has up to 17% less power consumption than original VVC fractional interpolation hardware on the same FPGA.

REFERENCES

[1] J. Chen, Y. Chen, M. Karczewiz, X. Li, H. Liu, L. Zhang, and 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, and J. Boyce, "Algorithm Description of Joint Exploration Model 7," JVET-G1001, July 2017.

[3] H. Azgin, A. C. Mert, E. Kalali, and I. Hamzaoglu, "Reconfigurable intra prediction hardware for future video coding," IEEE Trans. on Consumer Electronics, vol. 63, no. 4., pp. 419-425, Nov. 2017.

[4] A. C. Mert, E. Kalali, and I. Hamzaoglu, "High performance 2D transform hardware for future video coding," IEEE Trans. on Consumer Electronics, vol. 63, no. 2., pp. 117-125, May. 2017.

[5] A. C. Mert, E. Kalali, and I. Hamzaoglu, "An FPGA implementation of future video coding 2D transform," IEEE Int. Conf. on Consumer Electronics – Berlin (ICCE-Berlin), pp. 31-36, Sep. 2017.

[6] M. J. Garrido, F. Pescador, M. Chavarrias, P. J. Lobo, and C. Sanz, "A high performance FPGA-based architecture for the future video coding adaptive multiple core transform," IEEE Trans. on Consumer Electronics, vol. 64, no. 1., pp. 53-60, Feb. 2018.

[7] J. Vanne, M. Viitanen, T.D. Hämäläinen, and 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.

[8] E. Kalali, Y. Adibelli, and I. Hamzaoglu, "A high performance and low energy intra prediction hardware for high efficiency video coding," Int. Conf. on Field Programmable Logic and Applications (FPL), pp. 719-722, Aug. 2012.

[9] E. Kalali, E. Ozcan, O. M. Yalcinkaya, and I. Hamzaoglu, "A low energy HEVC inverse transform," IEEE Trans. on Consumer Electronics, vol. 60, no. 4, pp. 754-761, Nov. 2014.

[10] E. Kalali, A. C. Mert, and 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.

[11] B. Stabernack, J. Möller, J. Hahlbeck, and J. Brandenburg, "Demonstrating and FPGA implementation of a full HD real-time HEVC decoder with memory optimizations for range extensions support," Int. Conference on Design and Architectures for Signal and Image Processing (DASIP), Sep. 2015.

[12] Y. Voronenko and M. Püschel, "Multiplierless constant multiplication," ACM Trans. on Algorithms, vol. 3, no. 2, May. 2007.

[13] E. Kalali and I. Hamzaoglu, "A low energy HEVC sub-pixel interpolation hardware," IEEE Int. Conference on Image Processing (ICIP), pp. 1218-1222, Oct.2014.

[14] C. M. Diniz, M. Shafique, S. Bampi, and J. Henkel, "High throughput interpolation hardware architecture with coarse-grained reconfigurable datapaths for HEVC," IEEE Int. Conference on Image Processing (ICIP), pp. 2091-2095, Sep. 2013.

[15] G. Pastuszak and M. Trochimiuk, "Architecture design and efficiency evaluation for the high-throughput interpolation in the HEVC encoder," Euromicro Int. Conference on Digital System Design, Sep. 2013.

[16] F. Bossen, "Common test conditions and software reference configurations," JCTVC-I1100, May 2012.