Low Complexity HEVC Sub-Pixel Motion Estimation Technique and Its Hardware Implementation

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*—In this paper, a low complexity High Efficiency Video Coding (HEVC) sub-pixel motion estimation (SPME) technique is proposed. The proposed technique reduces the computational complexity of HEVC SPME significantly at the expense of slight quality loss by calculating the sum of absolute difference (SAD) values of sub-pixel search locations using the SAD values of neighboring integer pixel search locations. In this paper, an efficient HEVC SPME hardware implementing the proposed technique for all prediction unit (PU) sizes is also designed and implemented using Verilog HDL. The proposed hardware, in the worst case, can process 38 Quad Full HD (3840x2160) video frames per second.**

Keywords—HEVC, Sub-Pixel Motion Estimation, Hardware Implementation, FPGA.

I. INTRODUCTION

A new international video compression standard called High Efficiency Video Coding (HEVC) is recently developed [1]-[3]. It has 50% better video compression efficiency than H.264 standard. In order to increase the performance of integer pixel motion estimation, sub-pixel motion estimation (SPME), which provides sub-pixel accurate motion vector (MV) refinement, is performed. HEVC uses SPME same as H.264. However, HEVC SPME has higher computational complexity than H.264 SPME. HEVC standard uses three different 8-tap FIR filters for sub-pixel interpolation and up to 64x64 prediction unit (PU) sizes [4]. SPME is heavily used in an HEVC encoder [5]. It accounts for up to 49% of total encoding time of HEVC video encoder.

In this paper, a low complexity HEVC SPME technique for all PU sizes is proposed. The proposed technique interpolates the sum of absolute difference (SAD) values of sub-pixel search locations using the SAD values of neighboring integer pixel search locations. In this paper, an efficient HEVC SPME hardware implementing the proposed technique for all PU sizes is also designed and implemented using Verilog HDL. In order to reduce number and size of adders in this hardware, Hcub multiplierless constant multiplication (MCM) algorithm is used [6]. The proposed hardware finishes SPME for a PU in 6 clock cycles. It, in the worst case, can process 38 Quad Full HD (QFHD) (3840x2160) video frames per second.

Several HEVC SPME hardware are proposed in the literature [7]-[9]. In [7], SPME hardware searches all possible 48 sub-pixel search locations. However, it only supports square shaped PU sizes. In [8], SPME hardware supports all PU sizes but 8x4, 4x8 and 8x8. It uses bilinear filter for quarter-pixel interpolation. Also, it searches 12 sub-pixel search locations. In [9], SPME hardware supports all PU sizes but it uses a scalable search pattern. HEVC SPME hardware proposed in this paper is compared with these HEVC SPME hardware in Section V.

The rest of this paper is organized as follows. In Section II, HEVC SPME algorithm is explained. In Section III, the proposed HEVC SPME technique is explained. In Section IV, the proposed HEVC SPME hardware including the proposed technique is explained. The implementation results are given in Section V. Section VI presents the conclusion.

II. HEVC SPME ALGORITHM

After integer pixel motion estimation is performed for a PU, SPME is performed for the same PU to obtain sub-pixel accurate MV. In HEVC reference software video encoder (HM) [10], SPME is performed in two stages. As shown in Fig. 1, 8 sub-pixel search locations around the best integer pixel search location are searched in the first stage. 8 sub-pixel search locations around the best sub-pixel search location of the first stage are searched in the second stage. HEVC SPME first interpolates the necessary sub-pixels for sub-pixel search locations using three different 8-tap FIR filters. In Fig. 1, half-pixels a, b, c and d, h, n are interpolated using the nearest integer pixels in horizontal and vertical directions, respectively. Quarter-pixels e, i, p and f, j, q and g, k, r are interpolated using the nearest a and b and c half-pixels, respectively. HEVC SPME then calculates the SAD values for each sub-pixel search location, and determines the best sub-pixel search location with the minimum SAD value.

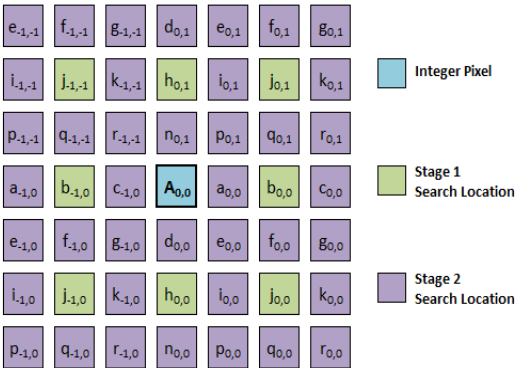
$e_{-1,-1}$ $f_{-1,-1}$ $g_{-1,-1}$ $d_{0,1}$ $e_{0,1}$ $f_{0,1}$ $g_{0,1}$
$i_{-1,-1}$ $j_{-1,-1}$ $k_{-1,-1}$ $h_{0,1}$ $i_{0,1}$ $j_{0,1}$ $k_{0,1}$
$p_{-1,-1}$ $q_{-1,-1}$ $r_{-1,-1}$ $n_{0,1}$ $p_{0,1}$ $q_{0,1}$ $r_{0,1}$
$a_{-1,0}$ $b_{-1,0}$ $c_{-1,0}$ $A_{0,0}$ $a_{0,0}$ $b_{0,0}$ $c_{0,0}$
$e_{-1,0}$ $f_{-1,0}$ $g_{-1,0}$ $d_{0,0}$ $e_{0,0}$ $f_{0,0}$ $g_{0,0}$
$i_{-1,0}$ $j_{-1,0}$ $k_{-1,0}$ $h_{0,0}$ $i_{0,0}$ $j_{0,0}$ $k_{0,0}$
$p_{-1,0}$ $q_{-1,0}$ $r_{-1,0}$ $n_{0,0}$ $p_{0,0}$ $q_{0,0}$ $r_{0,0}$
Integer Pixel
Stage 1 Search Location
Stage 2 Search Location

Fig. 1. Sub-pixel Search Locations

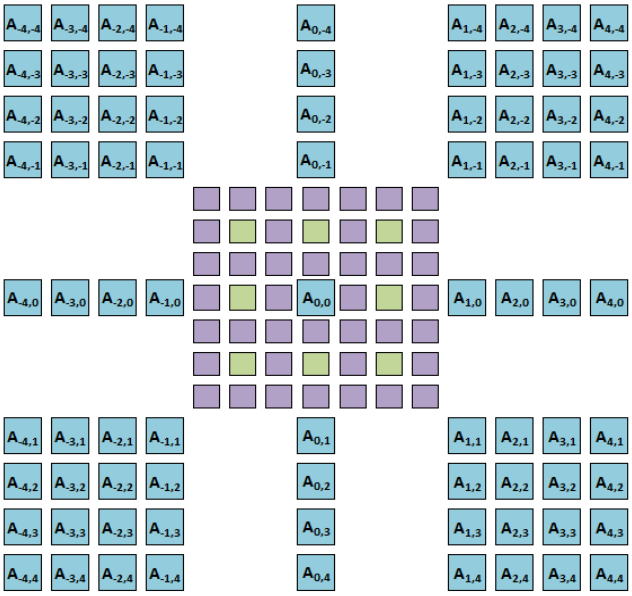
$A_{-4,-4}$ $A_{-3,-4}$ $A_{-2,-4}$ $A_{-1,-4}$ $A_{0,-4}$ $A_{1,-4}$ $A_{2,-4}$ $A_{3,-4}$ $A_{4,-4}$
$A_{-4,-3}$ $A_{-3,-3}$ $A_{-2,-3}$ $A_{-1,-3}$ $A_{0,-3}$ $A_{1,-3}$ $A_{2,-3}$ $A_{3,-3}$ $A_{4,-3}$
$A_{-4,-2}$ $A_{-3,-2}$ $A_{-2,-2}$ $A_{-1,-2}$ $A_{0,-2}$ $A_{1,-2}$ $A_{2,-2}$ $A_{3,-2}$ $A_{4,-2}$
$A_{-4,-1}$ $A_{-3,-1}$ $A_{-2,-1}$ $A_{-1,-1}$ $A_{0,-1}$ $A_{1,-1}$ $A_{2,-1}$ $A_{3,-1}$ $A_{4,-1}$
$A_{-4,0}$ $A_{-3,0}$ $A_{-2,0}$ $A_{-1,0}$ $A_{0,0}$ $A_{1,0}$ $A_{2,0}$ $A_{3,0}$ $A_{4,0}$
$A_{-4,1}$ $A_{-3,1}$ $A_{-2,1}$ $A_{-1,1}$ $A_{0,1}$ $A_{1,1}$ $A_{2,1}$ $A_{3,1}$ $A_{4,1}$
$A_{-4,2}$ $A_{-3,2}$ $A_{-2,2}$ $A_{-1,2}$ $A_{0,2}$ $A_{1,2}$ $A_{2,2}$ $A_{3,2}$ $A_{4,2}$
$A_{-4,3}$ $A_{-3,3}$ $A_{-2,3}$ $A_{-1,3}$ $A_{0,3}$ $A_{1,3}$ $A_{2,3}$ $A_{3,3}$ $A_{4,3}$
$A_{-4,4}$ $A_{-3,4}$ $A_{-2,4}$ $A_{-1,4}$ $A_{0,4}$ $A_{1,4}$ $A_{2,4}$ $A_{3,4}$ $A_{4,4}$

Fig. 2. 9x9 Integer Pixels

TABLE I. COMPUTATION AMOUNT FOR SQUARE-SHAPED PU SIZES

	Original HEVC SPME				Proposed
PU Sizes	**8x8**	**16x16**	**32x32**	**64x64**	**All**
Number of Interpolations	**1377**	**4641**	**16929**	**64545**	**100**
Number of Abs. Diff.	**1024**	**4096**	**16384**	**65536**	**0**

TABLE II. PSNR AND SSIM RESULTS

	Frame	ΔPSNR (dB)	SSIM
Class B (1920x1080)	**Tennis**	**-0.847**	**0.975**
	Kimono	**-0.225**	**0.982**
	Basketball D.	**-0.015**	**0.970**
	Park Scene	**-0.313**	**0.974**

III. PROPOSED HEVC SPME TECHNIQUE

The proposed HEVC SPME technique interpolates SAD values of sub-pixel search locations using the SAD values of neighboring integer pixel search locations. As shown in Fig. 2, the proposed technique uses SAD values of the best integer pixel search location, $A_{0,0}$, and its neighboring 80 integer pixel search locations, a 9x9 SAD block, for directly interpolating SAD values of 48 sub-pixel search locations using HEVC sub-pixel interpolation FIR filters. SAD values of half-pixel search locations are interpolated using the SAD values of nearest integer pixel search locations. SAD values of quarter-pixel search locations are interpolated using the SAD values of a, b, c half-pixel search locations.

The proposed technique performs SPME in two stages, same as HEVC reference software video encoder (HM) [10]. However, it performs SPME without interpolating a sub-pixel and calculating an absolute difference (AD). Table I shows the number of interpolation and AD operations required for performing HEVC SPME for one square-shaped PU. Since the proposed technique only interpolates SAD values of sub-pixel search locations, number of interpolation operations is significantly reduced and AD operation is not required.

The proposed HEVC SPME technique is implemented in MATLAB. As shown in Table II, MATLAB simulation results show that it slightly decreases PSNR and achieves good structural similarity index (SSIM) results.

IV. PROPOSED HEVC SPME HARDWARE

The proposed HEVC SPME hardware for all PU sizes is shown in Fig. 3. It takes 9x9 20-bit SAD values of 9x9 integer pixel search locations as input into Integer SAD buffer. Three buffers are used to store the SAD values of sub-pixel search locations. These on-chip buffers reduce the required off-chip memory bandwidth and power consumption.

The proposed hardware has three interpolation units. Each interpolation unit takes 9 SAD values as input and interpolates 20-bit SAD values of 3x2=6 sub-pixel search locations in each clock cycle. It interpolates 2 SAD values using type A, 2 SAD values using type B and 2 SAD values using type C FIR filter equations. As shown in Fig. 4, common expressions are calculated in type A, type B and type C FIR filter equations and same integer pixel is multiplied with different constant coefficients in type A, type B and type C FIR filter equations. Therefore, in an interpolation unit, common expressions in different equations are calculated once, and the result is used in all the equations.

Multiplications in FIR filter equations are performed using only adders and shifters. In the proposed hardware, Hcub MCM algorithm is used to reduce number and size of the adders, and to minimize adder tree depth [11]. Hcub algorithm tries to minimize number of adders, their bit size and adder tree depth in a multiplier block, which multiplies a single input with multiple constants. A multiplier block hardware has only one input, and it outputs results of multiplications with all the constants. Hcub algorithm determines necessary shift and addition operations in a multiplier block.

As shown in Table III, since different constant coefficients are used in FIR filter equations, three different multiplier blocks are used. Common 1 (C1) datapath calculates the common sub-expressions in the equations shown in the blue boxes in Fig. 4. Multiplier 1 (M1), Multiplier 2 (M2), and Multiplier 3 (M3) datapaths calculate the multiplications with multiple constant coefficients for different set of coefficients. For example, M2 datapath calculates the multiplications for A_1 written with red color in Fig. 4.

Comparator unit compares the SAD values of sub-pixel search locations, and determines the best sub-pixel search location with minimum SAD value. It uses three 20-bit comparators and performs comparison in 6 clock cycles.

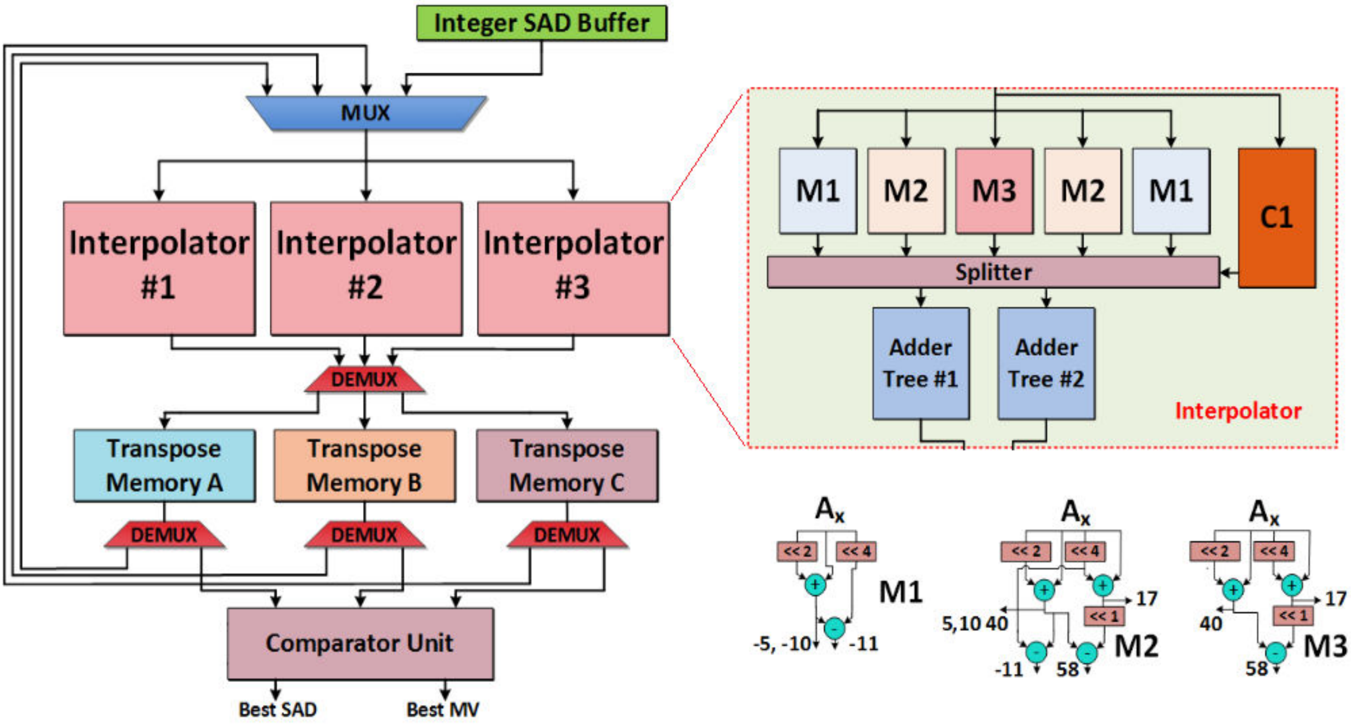

Fig. 3. Proposed HEVC Sub-Pixel Motion Estimation Hardware

$$a_{-1,0} = -A_{-4} + 4xA_{-3} - 10xA_{-2} + 58xA_{-1} + 17xA_0 - 5xA_1 + A_2$$
$$a_{0,0} = -A_{-3} + 4xA_{-2} - 10xA_{-1} + 58xA_0 + 17xA_1 - 5xA_2 + A_3$$

$$b_{-1,0} = -A_{-4} + 4xA_{-3} - 11xA_{-2} + 40xA_{-1} + 40xA_0 - 11xA_1 + 4xA_2 - A_3$$
$$b_{0,0} = -A_{-3} + 4xA_{-2} - 11xA_{-1} + 40xA_0 + 40xA_1 - 11xA_2 + 4xA_3 - A_4$$

$$c_{-1,0} = A_{-3} - 5xA_{-2} + 17xA_{-1} + 58xA_0 - 10xA_1 + 4xA_2 - A_3$$
$$c_{0,0} = A_{-2} - 5xA_{-1} + 17xA_0 + 58xA_1 - 10xA_2 + 4xA_3 - A_4$$

Fig. 4. Type A, Type B and Type C FIR Filters

TABLE III. CONSTANT COEFFICIENTS

Input SADs	Coefficients	Datapath
A_{-4}	-1	
A_{-3}	-1, 4	C1
A_{-2}	4, -5, -10, -11	M1
A_{-1}	-5, -10, -11, 17, 40, 58	M2
A_0	17, 58, 40	M3
A_1	-5, -10, -11, 17, 40, 58	M2
A_2	4, -5, -10, -11	M1
A_3	-1, 4	
A_4	-1	C1

SAD values of 48 sub-pixel search locations should be interpolated. First, 9x2 SAD values of a, b, c half-pixel search locations necessary for interpolating SAD values of quarter-pixel search locations are interpolated using SAD values of integer pixel search locations in 3 clock cycles. Then, 2x1 SAD values of d, h, n half-pixel search locations are interpolated using SAD values of integer pixel search locations in 1 clock cycle. Finally, 2x2 SAD values of quarter-pixel search locations are interpolated using SAD values of a, b, c half-pixel search locations in 2 clock cycles.

Because of the input data loading and pipelining, the proposed hardware starts producing outputs after 12 clock cycles. It then continues producing outputs at every 6 clock cycles without any stall. Therefore, it finishes SPME for a PU in 6 clock cycles.

V. IMPLEMENTATION RESULTS

The proposed HEVC SPME hardware for all PU sizes including the proposed technique is implemented using Verilog HDL. The Verilog RTL implementation is verified with RTL simulations. RTL simulation results matched the results of MATLAB implementation of HEVC SPME including the proposed technique.

The Verilog RTL code is synthesized and mapped to a XC6VLX365T Xilinx Virtex 6 FPGA with speed grade 3. The FPGA implementation is verified with post place & route simulations. The FPGA implementation uses 5200 LUTs, 1814 Slices and 3794 DFFs. The FPGA implementation works at 142 MHz. It can process 19 QFHD (3840x2160) video frames per second.

Power consumption of the FPGA implementation is estimated using Xilinx XPower Analyzer tool. Post place & route timing simulations are performed for Tennis, Kimono, BQ Terrace and Basketball Drive class B videos (one frame from each video) at 100 MHz [12] and signal activities are stored in VCD files. These VCD files are used for estimating power consumption of the FPGA implementation. These power consumption results are shown in Table IV.

TABLE IV. POWER CONSUMPTION RESULTS

	Tennis	**Kimono**	**BQ Terr.**	**Basketball D.**
Clock (mW)	33	33	33	33
Logic (mW)	68	79	78	67
Signal (mW)	143	168	163	139
Total Power (mW)	244	280	274	239

TABLE V. HARDWARE COMPARISON

	[7]	**[8]**	**[9]**	**Proposed**	
Tech.	65 nm	65 nm	Xilinx Virtex6	90 nm	Xilinx Virtex6
Gate/Slice Count	249.1 K	1183 K	130306	26 K	1814
Max Freq. (MHz)	396.8	188	200	280	142
Power Dissip. (mW)	48.67	198.6	----	28	280
Supported PU sizes	Square Shaped	All but 8x8,8x4 and 4x8	All	All	All
Fps	60 QFHD	30 QFHD	32 QFHD	38 QFHD	19 QFHD
Fps * (Normalized)	6 QFHD	15 QFHD	32 QFHD	38 QFHD	19 QFHD

*: Frames per second when hardware processes all PU sizes

In order to compare the proposed HEVC SPME hardware with the HEVC SPME hardware in the literature, the Verilog RTL code is also synthesized to a 90 nm standard cell library and resulting netlist is placed and routed. The resulting ASIC implementation works at 280 MHz. It can process 38 QFHD (3840x2160) video frames per second. Gate count of the ASIC implementation is calculated as 26K according to NAND (2x1) gate area excluding on-chip memory.

The comparison of the proposed HEVC SPME hardware with the HEVC SPME hardware in the literature is shown in Table V. The proposed hardware implements HEVC SPME for all PU sizes and it is the only hardware that implements the two stages SPME performed in HEVC reference software video encoder (HM) [10]. It is faster, and it has smaller area and lower power consumption than the other HEVC SPME hardware. HEVC SPME hardware proposed in [9] is faster than FPGA implementation of the proposed hardware. However, it has 70 times larger area than FPGA implementation of the proposed hardware.

VI. CONCLUSION

In this paper, a low complexity HEVC SPME technique is proposed. The proposed technique reduced the computational complexity of HEVC SPME significantly at the expense of slight quality loss. In this paper, an efficient HEVC SPME hardware implementing the proposed technique for all PU sizes is also designed and implemented using Verilog HDL. The proposed hardware, in the worst case, can process 38 QFHD (3840x2160) video frames per second.

ACKNOWLEDGMENT

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

REFERENCES

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

[2] 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.

[3] E. Kalali, E. Ozcan, O. M. Yalcinkaya, I. Hamzaoglu, “A Low Energy HEVC Inverse DCT Hardware”, IEEE Int. Conference on Consumer Electronics – Berlin, Sept. 2013.

[4] E. Kalali, Y. Adibelli, I. Hamzaoglu, “A Reconfigurable HEVC Sub-Pixel Interpolation Hardware”, IEEE Int. Conference on Consumer Electronics - Berlin, Sept. 2013.

[5] J. Vanne, M. Viitanen, T.D. Hämäläinen, 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.

[6] Y. Voronenko, M. Püschel, “Multiplierless Constant Multiple Multiplication”, ACM Trans. on Algorithms, vol. 3, no. 2, May 2007.

[7] V. Afonso, H. Maich, L. Audibert, B. Zatt, M. Porto, L. Agostini, “Memory-Aware and High-Throughput Hardware Design for the HEVC Fractional Motion Estimation”, Symposium on Integrated Circuits and System Design, 2013.

[8] G. He, D. Zhou, Y. Li, Z. Chen, T. Zhang, S. Goto, “High-Throughput Power-Efficient VLSI Architecture of Fractional Motion Estimation for Ultra-HD HEVC Video Encoding”, IEEE Trans. on VLSI Systems, vol.23, no.12, pp.3138-3142, March 2015.

[9] D. Ding, X. Ye, S. Wang, “1/2 and 1/4 Pixel Paralleled FME with A Scalable Search Pattern for HEVC Ultra-HD Encoding”, IEEE Int. Conf. on Communication Technology, pp.278-281, Oct. 2015.

[10] K. McCann, B. Bross, W.J. Han, I.K. Kim, K. Sugimoto, G. J. Sullivan, “High Efficiency Video Coding (HEVC) Test Model (HM) 15 Encoder Description”, JCTVC-Q1002, June 2014.

[11] E. Kalali, I. Hamzaoglu, “A low energy HEVC sub-pixel interpolation hardware,” IEEE Int. Conference on Image Processing, pp. 1218-1222, Oct. 2014.

[12] F. Bossen, “Common test conditions and software reference configurations”, JCTVC-I1100, May 2012.