An HEVC Fractional Interpolation Hardware Using Memory Based Constant Multiplication

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*—Fractional interpolation is one of the most computationally intensive parts of High Efficiency Video Coding (HEVC) video encoder and decoder. In this paper, an HEVC fractional interpolation hardware using memory based constant multiplication is proposed. The proposed hardware uses memory based constant multiplication technique for implementing multiplication with constant coefficients. The proposed memory based constant multiplication hardware stores pre-computed products of an input pixel with multiple constant coefficients in memory. Several optimizations are proposed to reduce memory size. The proposed HEVC fractional interpolation hardware, in the worst case, can process 35 quad full HD (3840x2160) video frames per second. It has up to 31% less energy consumption than original HEVC fractional interpolation hardware.**

Keywords*—HEVC, Fractional Interpolation, Hardware Implementation, Memory Based Multiplication.

I. Introduction

Joint collaborative team on video coding (JCT-VC) recently developed a new video compression standard called High Efficiency Video Coding (HEVC) [1]-[5]. HEVC provides 50% better coding efficiency than H.264 video compression standard [6]. Fractional (half-pixel and quarter-pixel) interpolation is one of the most computationally intensive parts of HEVC video encoder and decoder. Fractional interpolation accounts for 25% and 50% of the HEVC encoder and decoder complexity, respectively [6].

In H.264 standard, a 6-tap FIR filter is used for half-pixel interpolation and bilinear filter is used for quarter-pixel interpolation [7, 8]. In HEVC standard, three different 8-tap FIR filters are used for half-pixel and quarter-pixel interpolations. Block sizes from 4x4 to 16x16 are used in H.264 standard. However, in HEVC standard, prediction unit (PU) sizes can be from 4x8/8x4 to 64x64. Therefore, HEVC fractional interpolation is more complex than H.264 fractional interpolation.

Memory based constant multiplication is an efficient computation technique [9, 10]. A memory based constant multiplication hardware stores pre-computed product values for an input word into memory and necessary product value is read from the memory using input word as the address.

In this paper, an HEVC fractional interpolation hardware using memory based constant multiplication for all PU sizes is designed and implemented using Verilog HDL. The proposed hardware uses memory based constant multiplication technique for implementing multiplication with constant coefficients. The proposed memory based constant multiplication hardware stores pre-computed products of an input pixel with multiple constant coefficients in memory. Several optimizations are proposed to reduce memory size.

Several HEVC fractional interpolation hardware are proposed in the literature [11]-[15]. In Section IV, they are compared with the HEVC fractional interpolation hardware proposed in this paper. They do not use memory based constant multiplication technique.

In [11], three different 8-tap FIR filters are implemented using a reconfigurable datapath. It can calculate one FIR filter output at a time. Therefore, it can only be used for motion compensation. The proposed hardware in [12] uses Hcub multiplierless constant multiplication (MCM) algorithm for multiplication with constant coefficients. In [13]-[15], the proposed hardware use adders and shifters for FIR filter implementation.

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

II. HEVC Fractional Interpolation Algorithm

In HEVC, three different 8-tap FIR filters are used for both half-pixel and quarter-pixel interpolations. These three FIR filters type A, type B and type C are shown in (1), (2), and (3), respectively. The *shift1* value is determined based on bit depth of the pixel.

Integer pixels ($A_{x,y}$), half pixels ($a_{x,y}$, $b_{x,y}$, $c_{x,y}$, $d_{x,y}$, $h_{x,y}$, $n_{x,y}$) and quarter pixels ($e_{x,y}$, $f_{x,y}$, $g_{x,y}$, $i_{x,y}$, $j_{x,y}$, $k_{x,y}$, $p_{x,y}$, $q_{x,y}$, $r_{x,y}$) in a PU are shown in Fig. 1. The type A, type B and type C FIR filter equations for 8 half-pixels are shown in Fig. 2.

The half pixels a, b, c are interpolated from nearest integer pixels in horizontal direction, and the half-pixels d, h, n are interpolated from nearest integer pixels in vertical direction. The quarter pixels e, f, g are interpolated from the nearest half pixels a, b, c, respectively, in vertical direction using type A filter. The quarter pixels i, j, k are interpolated similarly using type B filter, and the quarter pixels p, q, r are interpolated similarly using type C filter.

HEVC fractional interpolation algorithm used in HEVC encoder calculates all the fractional pixels necessary for fractional motion estimation operation.

978-1-5386-3025-9/18/$31.00 ©2018 IEEE

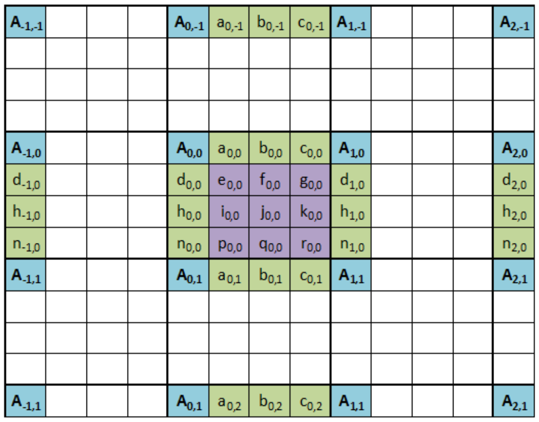
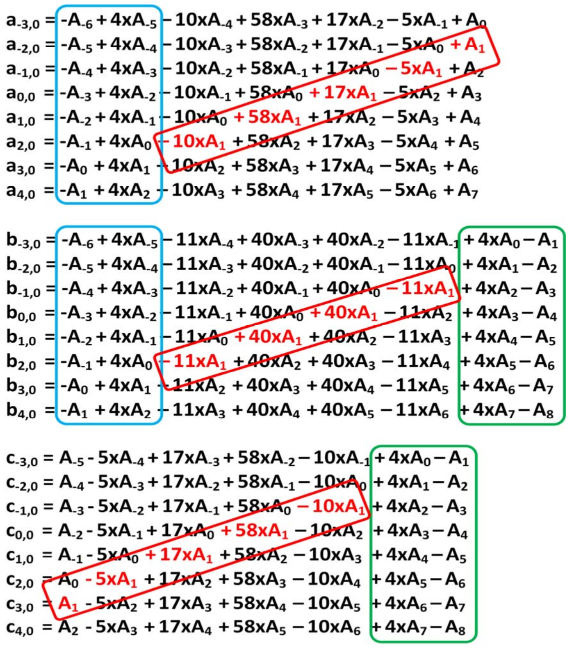

A-1,-1
A0,-1 a0,-1 b0,-1 c0,-1 A1,-1
A2,-1
A-1,0
A0,0 a0,0 b0,0 c0,0 A1,0
A2,0
d-1,0 d0,0 e0,0 f0,0 g0,0 d1,0 d2,0
h-1,0 h0,0 i0,0 j0,0 k0,0 h1,0 h2,0
n-1,0 n0,0 p0,0 q0,0 r0,0 n1,0 n2,0
A-1,1 A0,1 a0,1 b0,1 c0,1 A1,1 A2,1
A-1,1 A0,1 a0,2 b0,2 c0,2 A1,1 A2,1

Fig. 1. Integer, half and quarter pixels.



$$
\begin{aligned}
a_{-3,0} &= -A_{-6}+4xA_{-5}-10xA_{-4}+58xA_{-3}+17xA_{-2}-5xA_{-1}+A_{0}\\
a_{-2,0} &= -A_{-5}+4xA_{-4}-10xA_{-3}+58xA_{-2}+17xA_{-1}-5xA_{0}+A_{1}\\
a_{-1,0} &= -A_{-4}+4xA_{-3}-10xA_{-2}+58xA_{-1}+17xA_{0}-5xA_{1}+A_{2}\\
a_{0,0} &= -A_{-3}+4xA_{-2}-10xA_{-1}+58xA_{0}+17xA_{1}-5xA_{2}+A_{3}\\
a_{1,0} &= -A_{-2}+4xA_{-1}-10xA_{0}+58xA_{1}+17xA_{2}-5xA_{3}+A_{4}\\
a_{2,0} &= -A_{-1}+4xA_{0}-10xA_{1}+58xA_{2}+17xA_{3}-5xA_{4}+A_{5}\\
a_{3,0} &= -A_{0}+4xA_{1}-10xA_{2}+58xA_{3}+17xA_{4}-5xA_{5}+A_{6}\\
a_{4,0} &= -A_{1}+4xA_{2}-10xA_{3}+58xA_{4}+17xA_{5}-5xA_{6}+A_{7}
\end{aligned}
$$

$$
\begin{aligned}
b_{-3,0} &= -A_{-6}+4xA_{-5}-11xA_{-4}+40xA_{-3}+40xA_{-2}-11xA_{-1}+4xA_{0}-A_{1}\\
b_{-2,0} &= -A_{-5}+4xA_{-4}-11xA_{-3}+40xA_{-2}+40xA_{-1}-11xA_{0}+4xA_{1}-A_{2}\\
b_{-1,0} &= -A_{-4}+4xA_{-3}-11xA_{-2}+40xA_{-1}+40xA_{0}-11xA_{1}+4xA_{2}-A_{3}\\
b_{0,0} &= -A_{-3}+4xA_{-2}-11xA_{-1}+40xA_{0}+40xA_{1}-11xA_{2}+4xA_{3}-A_{4}\\
b_{1,0} &= -A_{-2}+4xA_{-1}-11xA_{0}+40xA_{1}+40xA_{2}-11xA_{3}+4xA_{4}-A_{5}\\
b_{2,0} &= -A_{-1}+4xA_{0}-11xA_{1}+40xA_{2}+40xA_{3}-11xA_{4}+4xA_{5}-A_{6}\\
b_{3,0} &= -A_{0}+4xA_{1}-11xA_{2}+40xA_{3}+40xA_{4}-11xA_{5}+4xA_{6}-A_{7}\\
b_{4,0} &= -A_{1}+4xA_{2}-11xA_{3}+40xA_{4}+40xA_{5}-11xA_{6}+4xA_{7}-A_{8}
\end{aligned}
$$

$$
\begin{aligned}
c_{-3,0} &= A_{-5}-5xA_{-4}+17xA_{-3}+58xA_{-2}-10xA_{-1}+4xA_{0}-A_{1}\\
c_{-2,0} &= A_{-4}-5xA_{-3}+17xA_{-2}+58xA_{-1}-10xA_{0}+4xA_{1}-A_{2}\\
c_{-1,0} &= A_{-3}-5xA_{-2}+17xA_{-1}+58xA_{0}-10xA_{1}+4xA_{2}-A_{3}\\
c_{0,0} &= A_{-2}-5xA_{-1}+17xA_{0}+58xA_{1}-10xA_{2}+4xA_{3}-A_{4}\\
c_{1,0} &= A_{-1}-5xA_{0}+17xA_{1}+58xA_{2}-10xA_{3}+4xA_{4}-A_{5}\\
c_{2,0} &= A_{0}-5xA_{1}+17xA_{2}+58xA_{3}-10xA_{4}+4xA_{5}-A_{6}\\
c_{3,0} &= A_{1}-5xA_{2}+17xA_{3}+58xA_{4}-10xA_{5}+4xA_{6}-A_{7}\\
c_{4,0} &= A_{2}-5xA_{3}+17xA_{4}+58xA_{5}-10xA_{6}+4xA_{7}-A_{8}
\end{aligned}
$$

Fig. 2. Type A, Type B and Type C FIR Filters

$$a_{0,0} = (-A_{-3,0} + 4 * A_{-2,0} - 10 * A_{-1,0} + 58 * A_{0,0} + 17 * A_{1,0} - 5 * A_{2,0} + A_{3,0}) \gg shift1 \quad (1)$$

$$b_{0,0} = (-A_{-3,0} + 4 * A_{-2,0} - 11 * A_{-1,0} + 40 * A_{0,0} + 40 * A_{1,0} - 11 * A_{2,0} + 4 * A_{3,0} - A_{4,0}) \gg shift1 \quad (2)$$

$$c_{0,0} = (A_{-2,0} - 5 * A_{-1,0} + 17 * A_{0,0} + 58 * A_{1,0} - 10 * A_{2,0} + 4 * A_{3,0} - A_{4,0}) \gg shift1 \quad (3)$$

III. Proposed HEVC Fractional Interpolation Hardware

The proposed HEVC fractional interpolation hardware for all PU sizes is shown in Fig. 3. The proposed hardware interpolates all the fractional pixels (half-pixels and quarter-pixels) for the luma component of a PU using integer or half-pixels. The proposed hardware is designed for 8x8 PU size and it produces necessary fractional pixels for an 8x8 PU. For other 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.

In the proposed hardware, 8x3 fractional pixels are interpolated in parallel using type A, type B and type C FIR filters. In the proposed hardware, common sub-expression calculation method proposed in [12] is used. As shown in Fig. 2, there are common sub-expressions in different filter type equations. Common sub-expressions in type A and type B filters are shown in blue boxes. Common sub-expressions in type B and type C filters are shown in green boxes. In the proposed hardware, common sub-expressions in different equations are calculated once, and the results are used in all the equations. The common sub-expressions are calculated in CSE datapath using adders and shifters.

Three on-chip transpose memories are used to store half-pixels necessary for interpolating quarter-pixels. The half-pixels are interpolated using integer pixels and the interpolated a, b and c half-pixels are stored in the transpose memories A, B and C, respectively. These on-chip memories reduce the required off-chip memory bandwidth and power consumption.

Each input pixel should be multiplied with multiple constant coefficients shown as red boxes in Fig. 2. Table I shows constant coefficient multiplications necessary for each input pixel. In the proposed hardware, constant coefficient multiplications are implemented using memory based constant multiplication technique. As shown in Table I, since constant coefficients of input pixels (A_{-4}, A_6) and (A_{-3} … A_5) are different, two different memories, MEM1 and MEM2, are used to store pre-computed products of an input pixel with multiple constant coefficients.

Input pixels (A_{-4}, A_6) need to be multiplied with constant coefficients 1, -5, -10 and -11. In the proposed hardware, MEM1 stores two product values *5xA* and *-11xA* for input pixel *A*. The product value *10xA* is obtained from *5xA* using shift operation. Input pixels (A_{-3} … A_5) need to be multiplied with constant coefficients 1, -5, -10, -11, 17, 40 and 58. In the proposed hardware, MEM2 stores four product values *5xA, -11xA, 17xA and 29xA* for input pixel *A*. Product values *10xA* and *40xA* are obtained from *5xA* using shift operation. After constant coefficient multiplications are performed by memory based constant multiplication technique, fractional pixels are calculated using adder trees.

TABLE I. Constant Coefficients

Input Pixel	Necessary Coefficients	Hardware	Stored Products
A_{-5}	1	---	---
A_{-4}	1,-5,-10,-11	MEM1	5,-11
A_{-3}	1,-5,-10,-11,17,40,58		5,-11,17,29
A_{-2}	1,-5,-10,-11,17,40,58		5,-11,17,29
A_{-1}	1,-5,-10,-11,17,40,58		5,-11,17,29
A_0	1,-5,-10,-11,17,40,58		5,-11,17,29
A_1	1,-5,-10,-11,17,40,58	MEM2	5,-11,17,29
A_2	1,-5,-10,-11,17,40,58		5,-11,17,29
A_3	1,-5,-10,-11,17,40,58		5,-11,17,29
A_4	1,-5,-10,-11,17,40,58		5,-11,17,29
A_5	1,-5,-10,-11,17,40,58		5,-11,17,29
A_6	1,-5,-10,-11	MEM1	5,-11
A_7	1	---	---

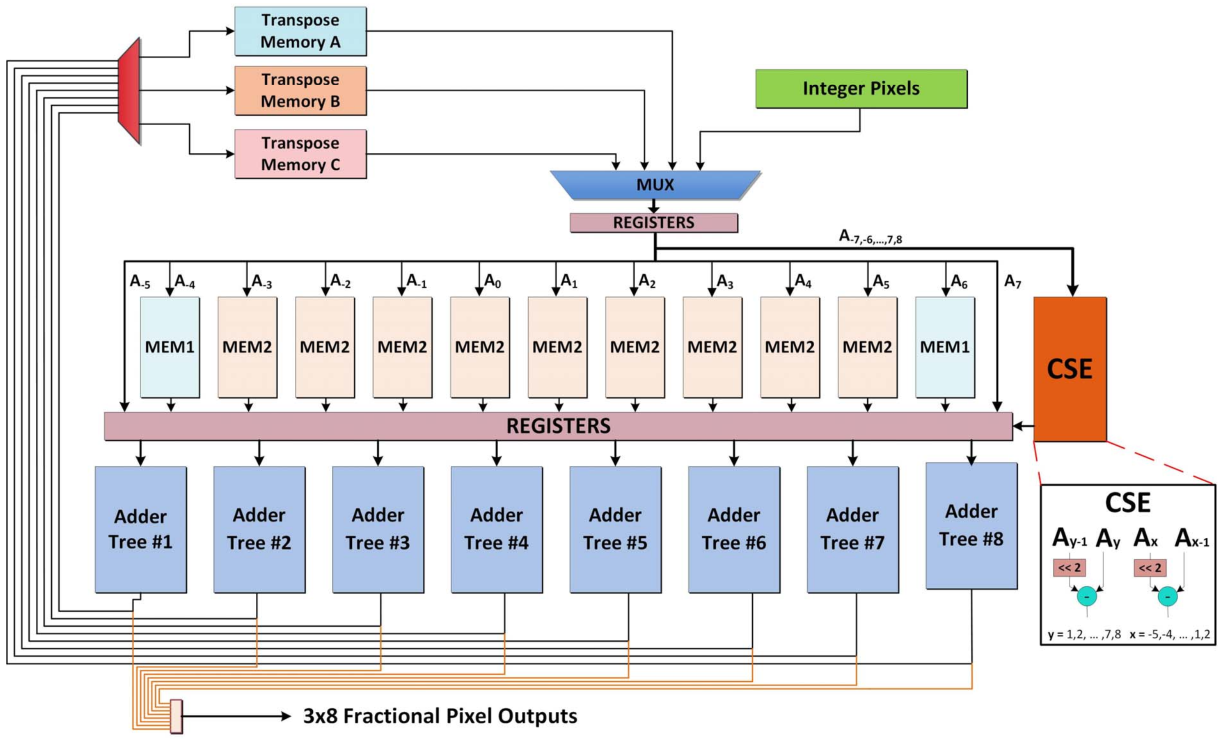


Fig. 3. Proposed HEVC Fractional Interpolation Hardware

8-bit unsigned input pixel A is used as the address of MEM1 and MEM2 memories. MEM1 stores 2 product values, *5xA* and *-11xA*, in each address. MEM2 stores 4 product values, *5xA*, *-11xA*, *17xA* and *29xA*, in each address. Since address ports of MEM1 and MEM2 are 8-bits, MEM1 and MEM2 store 2^8x2 and 2^8x4 product values, respectively.

Multiplications of an input pixel A with constant coefficients 5, -11, 17 and 29 using additions and shifts are shown in (4-7) and Fig. 4. Products of an 8-bit unsigned input pixel with constant coefficients 5, -11, 17 and 29 are 11-bits, 13-bits, 13-bits and 13-bits, respectively. Therefore, MEM1 and MEM2 should store 11+13=24 and 11+13+13+13=50 bits in each address, respectively.

$$5xA = (A \ll 2) + A \quad (4)$$

$$-11xA = 5xA + ((A' + 1) \ll 4) \quad (5)$$

$$17xA = (A \ll 4) + A \quad (6)$$

$$29xA = (A \ll 4) + (A \ll 3) + 5xA \quad (7)$$

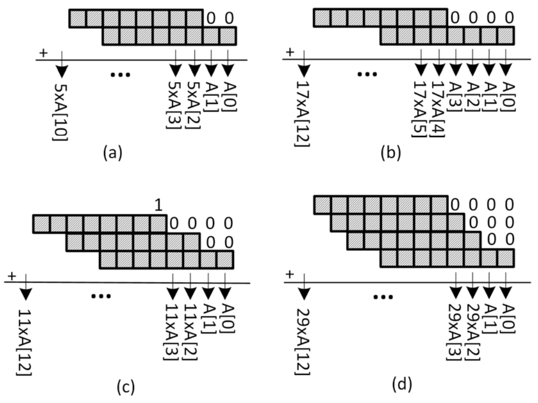


Fig. 4. Multiplication operations (a) 5xA (b) 17xA (c) -11xA (d) 29xA

As shown in Fig. 4, least significant 2-bits of *5xA*, *-11xA* and *29xA*, and least significant 4-bits of *17xA* are equal to the bits of input pixel A. Therefore, these bits of the products do not need to be stored in memories. This optimization saves 2+2=4 bits and 2+2+2+4=10 bits in each address of MEM1 and MEM2, respectively. Also, least significant third bit of *Ax5* is equal to the least significant third bit of *-11xA* and *29xA*, and the least significant fourth bit of *5xA* is equal to the least significant fourth bit of *-11xA*. Therefore, only least significant third and fourth bits of *5xA* need to be stored in memories and they should be used for *5xA*, *-11xA* and *29xA*. This optimization saves 2 bits and 2+1=3 bits in each address of MEM1 and MEM2, respectively.

Using these optimizations, number of bits in each address of MEM1 is reduced from 24 to 18 and number of bits in each address of MEM2 is reduced from 50 to 37. The proposed memories, MEM1 and MEM2, are shown in Fig. 5.

Since 15 fractional pixels should be interpolated for each integer pixel, 64x15 fractional pixels should be interpolated for an 8x8 PU. 8x7 extra a, b, c half-pixels are necessary for the interpolation of quarter-pixels.

First, 8x15 a, b and c half-pixels necessary for interpolating quarter-pixels are interpolated in 15 clock cycles, and stored in the transpose memories A, B and C, respectively. Then, 8x8 d, h, n half-pixels are interpolated in 8 clock cycles. Finally, 9x8x8 quarter-pixels are interpolated in 8x3 clock cycles using a, b and c half-pixels. There are three pipeline stages in the proposed hardware. Therefore, the proposed hardware, in the worst case, interpolates the fractional pixels for an 8x8 PU in 50 clock cycles.

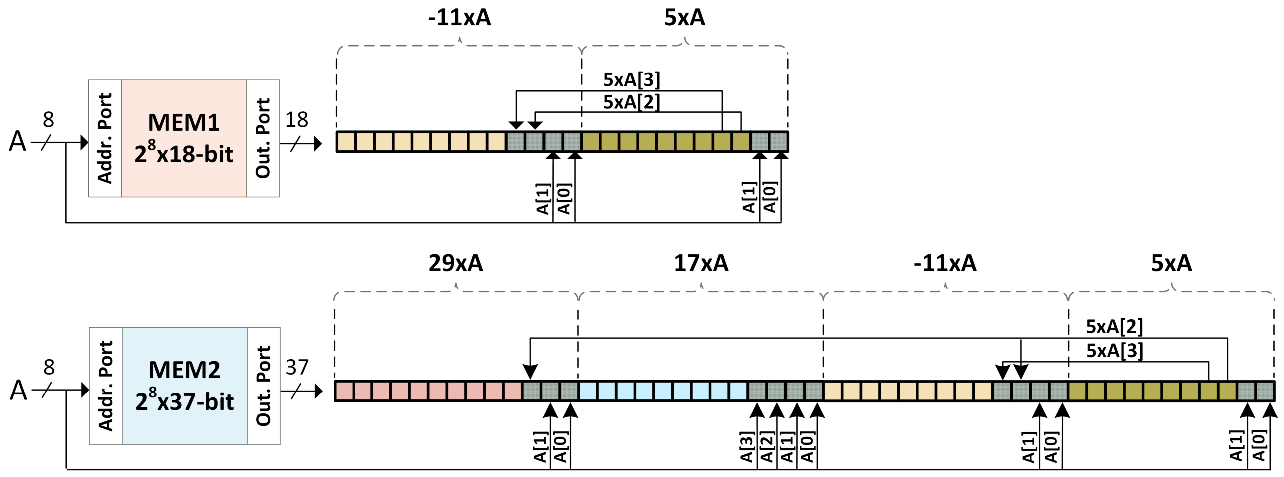


Fig. 5. MEM1 and MEM2 memories

TABLE II. IMPLEMENTATION RESULTS

	FIHW_ORG	FIHW_MCM	FIHW_MEM
FPGA	Xilinx Virtex6	Xilinx Virtex6	Xilinx Virtex6
DFFs	3207	3833	3815
LUTs	3752	3370	3806
Slices	1848	1208	1498
Max. Freq. (MHz)	154	200	233
Fps	23 QFHD	30 QFHD	35 QFHD

IV. IMPLEMENTATION RESULTS

The proposed HEVC fractional interpolation hardware using memory based constant multiplication (FIHW_MEM) for all PU sizes is implemented using Verilog HDL. The Verilog RTL code is verified with RTL simulations. RTL simulation results matched the results of a software implementation of HEVC fractional interpolation algorithm.

The Verilog RTL code is synthesized and mapped to a Xilinx XC6VLX130T FF1156 FPGA with speed grade 3 using Xilinx ISE 14.7. FIHW_MEM FPGA implementation is verified to work at 233 MHz by post place and route simulations. Therefore, it can process 35 quad full HD (3840x2160) video frames per second. It uses 3806 LUTs, 3815 DFFs and 1498 Slices.

In this paper, two HEVC fractional interpolation hardware implementations are used for energy consumption comparison. The first one (FIHW_ORG) is the original hardware proposed in [12]. It computes type A, B and C filters separately. The second one (FIHW_MCM) is the MCM hardware proposed in [12]. It computes multiplications with constant coefficients using Hcub multiplierless constant multiplication (MCM) algorithm.

Verilog RTL codes of these two HEVC fractional interpolation hardware are synthesized and mapped to a Xilinx XC6VLX130T FF1156 FPGA with speed grade 3 using Xilinx ISE 14.7. FPGA implementation of FIHW_ORG uses 3752 LUTs, 3207 DFFs and 1848 Slices. FPGA implementation of FIHW_MCM uses 3370 LUTs, 3833 DFFs and 1543 Slices. FPGA implementations of FIHW_ORG and FIHW_MCM are verified to work at 154 and 200 MHz, respectively, by post place and route simulations. Therefore, FPGA implementations of FIHW_ORG and FIHW_MCM can process 23 and 30 quad full HD (3840x2160) video frames per second, respectively. The implementation results are shown in Table II.

Power consumptions of all FPGA implementations are estimated using Xilinx XPower Analyzer tool. Post place & route timing simulations are performed for Tennis, Kimono, Park Scene (1920x1080) video frames at 100 MHz [16] and signal activities are stored in VCD files. These VCD files are used for estimating power consumptions of the FPGA implementations. As shown in Fig. 6, the proposed FIHW_MEM has up to 31% and 12.3% less energy consumption than FIHW_ORG and FIHW_MCM, respectively.

Comparison of the proposed HEVC fractional interpolation hardware with the HEVC fractional interpolation hardware in the literature is shown in Table III. The proposed HEVC fractional interpolation hardware has higher throughput than [11]-[14]. Only hardware in [15] has higher throughput than the proposed hardware at the expense of more area. The hardware in [11] has less area than the proposed hardware. However, it can only be used for motion compensation.

V. CONCLUSION

In this paper, an HEVC fractional interpolation hardware using memory-based constant multiplication for all PU sizes is proposed. The proposed hardware is verified to work at 233 MHz in a Xilinx Virtex 6 FPGA. The FPGA implementation, in the worst case, can process 35 quad full HD (3840x2160) video frames per second. It has up to 31% less energy consumption than original HEVC fractional interpolation hardware.

ACKNOWLEDGMENT

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

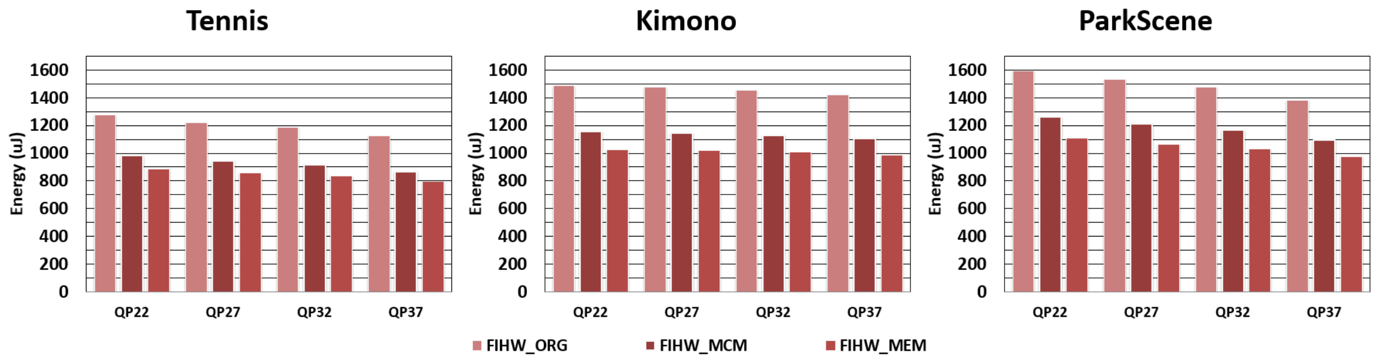


Fig. 6. Energy consumptions of FIHW_ORG, FIHW_MCM and FIHW_MEM

TABLE III. HARDWARE COMPARISON

	[11]	[12]	[13]	[14]	[15]	FIHW_MEM
FPGA	Xilinx Virtex 6	Xilinx Virtex 6	Arria II GX	Xilinx Virtex 5	Stratix III	Xilinx Virtex 6
Slices	---	---	---	2181	---	1498
LUTs	3005	3929	18831	5017	7701	3806
Block RAMs	2	6	---	2	---	---
Max. Freq. (MHz)	100	200	200	283	278	233
Frames per second	64 2560x1600	30 3840x2160	60 1920x1080	30 2560x1600	60 3840x2160	35 3840x2160

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, 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, May 2016.

[5] 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.

[6] 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.

[7] S. Yalcin, I. Hamzaoglu, "A High Performance Hardware Architecture for Half-Pixel Accurate H.264 Motion Estimation", *14th IFIP Int. Conference on VLSI-SoC*, Oct. 2006.

[8] S. Oktem, I. Hamzaoglu, "An Efficient Hardware Architecture for Quarter-Pixel Accurate H.264 Motion Estimation", *10th Euromicro Conference on Digital System Design*, Aug. 2007.

[9] P. K. Meher, "LUT Optimization for Memory-Based Computation", *IEEE Transactions on Circuits and Systems-II: Express Briefs*, vol. 57, no. 4, Apr. 2010.

[10] P. K. Meher, "New Approach to Look-Up-Table Design and Memory-Based Realization of FIR Digital Filter", *IEEE Transactions on Circuits and Systems-I: Regular Papers*, vol. 57, no. 3, Mar. 2010.

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

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

[13] G. Pastuszak, M. Trochimiuk, "Architecture Design and Efficiency Evaluation for the High-Throughput Interpolation in the HEVC Encoder", *16th Euromicro Conference on Digital System Design*, Sept. 2013.

[14] C. M. Diniz, M. Shafique, S. Bampi, J. Henkel, "A Reconfigurable Hardware Architecture for Fractional Pixel Interpolation in High Efficiency Video Coding," *IEEE Trans. on Computer-Aided Design of Integrated Circuits and Systems*, vol. 34, no. 2, pp. 238-251, Feb. 2015.

[15] H. Maich, C. Afonso, D. Franco, B. Zatt, M. Porto, L. Agostini, "High throughput hardware design for the HEVC Fractional Motion Estmation Interpolation Unit", *IEEE 20th International Conference on Electronics, Circuits, and Systems*, May 2014.

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