IEEE IEEE CTSoc CONSUMER TECHNOLOGY SOCIETY | IEEE Transactions on Consumer Electronics

Novel HEVC and VVC Fractional Interpolation Hardware

Journal:	*Transactions on Consumer Electronics*
Manuscript ID	TCE-2023-01-0005
Manuscript Type:	Original Article
Date Submitted by the Author:	01-Jan-2023
Complete List of Authors:	Mahdavi, Hossein; Sabanci University, Electronics Engineering Hamzaoglu, Ilker; Sabanci University, Electronics Engineering
Keywords:	hevc, vvc, fractional interpolation, hardware

Novel HEVC and VVC Fractional Interpolation Hardware

Hossein Mahdavi and Ilker Hamzaoglu, *Senior Member, IEEE*

***Abstract*— Fractional interpolation (FI) is a computationally complex algorithm used in the high efficiency video coding (HEVC) and versatile video coding (VVC) video encoder and decoder. In this paper, a novel technique is proposed for implementing HEVC FI (HFI) and VVC FI (VFI). The proposed technique decomposes the coefficients of FIR filters such that the number of additions is reduced. In this paper, HFI hardware, VFI hardware and approximate VFI hardware are designed and implemented using the proposed technique. The proposed HFI hardware has higher performance, less area, and less power consumption than the best HFI hardware in the literature. The proposed VFI hardware has higher performance, less area, and less power consumption than the best VFI hardware in the literature. The proposed approximate VFI hardware have the same performance, less area, and less power consumption than the best approximate VFI hardware in the literature.**

Keywords— HEVC, VVC, Fractional Interpolation, Hardware.

I. Introduction

ITU and ISO developed the international video compression standards HEVC [1]-[5] and VVC [6]-[11]. VVC has higher compression efficiency than HEVC at the cost of higher computational complexity [12]-[14].

In HFI, 1 8-tap and 2 7-tap finite impulse response (FIR) filters are used. In HFI, 3 horizontal half pixels (HHPs), 3 vertical half pixels (VHPs), and 9 quarter pixels (QPs) are interpolated for every integer pixel (IP). In VFI, 8 7-tap and 7 8-tap FIR filters are used. In VFI, 15 HHPs, 15 VHPs, and 225 QPs are interpolated for every IP. Hence, VFI has higher computational complexity than HFI.

Several HFI hardware are proposed in the literature [15]-[20]. The hardware proposed in [15] uses a new data reuse technique and a highly parallel architecture to improve throughput. The hardware proposed in [16] has 2 parallel datapaths for IP and fractional pixel (FP) motion estimation which share the same memories. The hardware proposed in [17] uses a reconfigurable datapath which can process different filter types. The hardware proposed in [18] uses the Hcub multiplierless constant multiplication (MCM) algorithm. It calculates common sub-expressions in filter equations only once. Hence, the number and size of the adders, and adder tree depth are reduced. The hardware proposed in [19] uses memory-based constant multiplication. The multiplications of an input pixel with multiple constant coefficients of FIR filters are pre-calculated and stored in memory. A high-level synthesis (HLS) based hardware implementation of HFI is proposed in [20]. It has higher performance than manual HFI hardware implementation at the expense of much larger area.

Several VFI hardware are proposed in the literature [21]-[25]. The hardware proposed in [21] implements 15 FIR filters in parallel. It calculates 255 FPs for every IP. The hardware proposed in [22] can be used only for motion compensation. It calculates 1 FP for every IP.

The approximate VFI hardware proposed in [23] and [24] include 14 3-tap and 1 4-tap FIR filters. They have less area and power consumption than the exact VFI hardware at the expense of a very small quality loss. The approximate VFI hardware proposed in [24] uses common offset values and Hcub MCM algorithm. An HLS based hardware implementation of VFI is proposed in [25]. It has higher performance than manual VFI hardware implementation at the expense of much larger area.

We proposed a novel technique for implementing HFI in [26]. It reduces the number of additions by decomposing the coefficients of the FIR filters. An HFI hardware using the proposed technique is proposed in [26].

This paper is extended version of [26]. In this paper, we also apply the proposed technique to exact and approximate VFI algorithms. We propose exact VFI hardware and approximate VFI hardware using the proposed technique.

The proposed FI hardware are implemented using Verilog HDL. The proposed HFI hardware has higher performance, less area, and less power consumption than the best HFI hardware in the literature. It can process 50 quad full HD (3840×2160) video frames per second (fps). The proposed VFI hardware has higher performance, less area, and less power consumption than the best VFI hardware in the literature. It can process 48 full HD (1920×1080) video fps. The proposed approximate VFI hardware have the same performance, less area, and less power consumption than the best approximate VFI hardware in the literature. They can process 49 and 52 full HD (1920×1080) video fps.

II. HEVC and VVC Fractional Interpolation

In HFI, 1 8-tap and 2 7-tap FIR filters are used. These 3 FIR filters type A, type B, type C are shown in (1), (2), (3), respectively. Fig. 1 shows IPs "$A_{x,y}$", HHPs "$a_{x,y}$, $b_{x,y}$, $c_{x,y}$", VHPs "$d_{x,y}$, $h_{x,y}$, $n_{x,y}$", and QPs "$e_{x,y}$, $f_{x,y}$, $g_{x,y}$, $i_{x,y}$, $j_{x,y}$, $k_{x,y}$, $p_{x,y}$, $q_{x,y}$, $r_{x,y}$" in a prediction unit (PU). The nearest IPs in horizontal direction are used for interpolating HHPs (a, b, c)

This research was supported in part by the Scientific and Technological Research Council of Turkey (TUBITAK) under the contract 118E134.

H. Mahdavi and I. Hamzaoglu are with Faculty of Engineering and Natural Sciences, Sabanci University, 34956 Tuzla, Istanbul, Turkey
(e-mail: {hmahdavi, hamzaoglu}@sabanciuniv.edu}).

and the nearest IPs in vertical direction are used for interpolating VHPs (d, h, n). The QPs are interpolated from the nearest HHPs.

In VFI, 8 7-tap and 7 8-tap FIR filters are used. Table I shows the coefficients of these FIR filters. A_{-3} to A_4 represent the input pixels. For example, F_7 FIR filter is shown in (4). The coefficients of filters F_9 to F_{15} are not shown in the table since they are symmetric with the filters F_7 to F_1.

$$F_7 = \begin{pmatrix} -A_{-3} + 4 \times A_{-2} - 11 \times A_{-1} + 45 \times A_0 \\ +34 \times A_1 - 10 \times A_2 + 4 \times A_3 - A_4 \end{pmatrix} >> 6 \quad (4)$$

In VFI, there are 15 HHPs between 2 neighboring horizontal IPs and 15 VHPs between 2 neighboring vertical IPs. The half pixels are interpolated from nearest IPs using 15 FIR filters. The QPs are interpolated from nearest HHPs using 15 FIR filters.

A. *VVC FI Hardware Proposed in [21]*

The exact VFI hardware proposed in [21] uses Hcub MCM algorithm and calculates a common offset for all the equations of 15 FIR filters. It also calculates common sub-expressions once and uses their results in different equations. The coefficients of offset and filters in this hardware are shown in Table II. Since the coefficients of filters F_9 to F_{15} are symmetric with the coefficients of filters F_7 to F_1, they are not shown in the table.

B. *Approximate VVC FI Filters Proposed in [24]*

Small coefficients of VFI FIR filters have less effect on the filter result. Due to spatial correlation, neighboring pixels have similar values. In [24], approximate VFI F1 FIR filters are proposed by assuming that the pixels multiplied with smaller coefficients are similar. In [24], approximate VFI F2 FIR filters are proposed by substituting most of the coefficients in F1 with the closest 2^n values. Hence, most of the multiplications of F2 FIR filters are implemented using only shift operations. In [24], MCMF1 and MCMF2 hardware are also proposed for implementing the approximate VFI F1 and F2 FIR filters, respectively.

Both MCMF1 and MCMF2 hardware use MCM algorithm and calculate 3 common offsets (Off_1, Off_2, Off_3). The coefficients of offsets and filters in MCMF1 and MCMF2 hardware are shown in Table III and Table IV, respectively.

In Table III, the coefficients of filters $F1OF_9$ to $F1OF_{15}$ are not shown since they are symmetric with the coefficients of $F1OF_7$ to $F1OF_1$. The filters $F1OF_9$ to $F1OF_{15}$ require the second offset (Off_2). In Table IV, F2 FIR filters $F2OF_4$, $F2OF_6$, $F2OF_9$, $F2OF_{11}$ are the same as $F2OF_5$, $F2OF_7$, $F2OF_{10}$, $F2OF_{12}$, respectively. Hence, in MCMF2 hardware, only filters $F2OF_4$, $F2OF_6$, $F2OF_9$, $F2OF_{11}$ are calculated, and their results are also used for filters $F2OF_5$, $F2OF_7$, $F2OF_{10}$, $F2OF_{12}$, respectively.

III. Proposed Hardware

A. *Proposed HEVC FI Hardware*

In the proposed HFI hardware, FIR filter coefficients are decomposed to other coefficients. Although this increases the number of coefficients, it also increases the number of common sub-expressions which in turn reduces the number of additions. Decompositions of the coefficients in type A, type B, and type C FIR filters are shown in equations (5), (6), and (7), respectively. Common sub-expressions are shown with the same color.

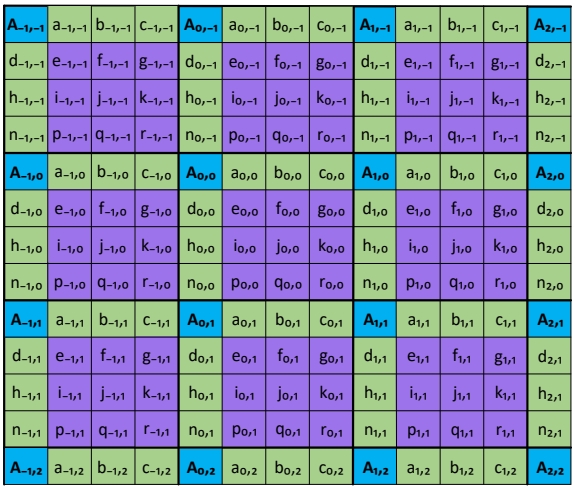

Fig. 1. Integer pixels, HHPs, VHPs, QPs in HEVC FI.

TABLE I
VVC FI Filters

	A_{-3}	A_{-2}	A_{-1}	A_0	A_1	A_2	A_3	A_4
F_1	0	1	-3	63	4	-2	1	0
F_2	-1	2	-5	62	8	-3	1	0
F_3	-1	3	-8	60	13	-4	1	0
F_4	-1	4	-10	58	17	-5	1	0
F_5	-1	4	-11	52	26	-8	3	-1
F_6	-1	3	-9	47	31	-10	4	-1
F_7	-1	4	-11	45	34	-10	4	-1
F_8	-1	4	-11	40	40	-11	4	-1

TABLE II
VVC FI Filters With Offset

	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
F_1	1	-3	5	31	-28	6	-3	1
F_2	0	-2	3	30	-24	5	-3	1
F_3	0	-1	0	28	-19	4	-3	1
F_4	0	0	-2	26	-15	3	-3	1
F_5	0	0	-3	20	-6	0	-1	0
F_6	0	-1	-1	15	-1	-2	0	0
F_7	0	0	-3	13	2	-2	0	0
F_8	0	0	-3	8	8	-3	0	0

$$a_{0,0} = \left(-A_{-3,0} + 4 \times A_{-2,0} - 10 \times A_{-1,0} + 58 \times A_{0,0} + 17 \times A_{1,0} - 5 \times A_{2,0} + A_{3,0}\right) >> 6 \quad (1)$$

$$b_{0,0} = \left(-A_{-3,0} + 4 \times A_{-2,0} - 11 \times A_{-1,0} + 40 \times A_{0,0} + 40 \times A_{1,0} - 11 \times A_{2,0} + 4 \times A_{3,0} - A_{4,0}\right) >> 6 \quad (2)$$

$$c_{0,0} = \left(A_{-2,0} - 5 \times A_{-1,0} + 17 \times A_{0,0} + 58 \times A_{1,0} - 10 \times A_{2,0} + 4 \times A_{3,0} - A_{4,0}\right) >> 6 \quad (3)$$

The proposed HFI hardware interpolates 24 FPs in each clock cycle (CC) by calculating 8 type A, 8 type B, and 8 type C FIR filters in parallel. In addition to common sub-expressions shown in (5), (6), (7), there are more common sub-expressions in these 24 FIR filter equations with decomposed coefficients.

Some common sub-expressions are negated versions of each other. For example, negated version of the sub-expression "$A_{-2,0}$ -4×$A_{-1,0}$" in (7), i.e., "-$A_{-2,0}$ +4×$A_{-1,0}$", exists in the FIR filter equations of $a_{1,0}$ and $b_{1,0}$. Hence, it can be calculated only once, and its result can be used in FIR filter equations of $a_{1,0}$ and $b_{1,0}$ by negating it. Also, the negated version of the sub-expression "-4×$A_{2,0}$ +$A_{3,0}$" in (5) exists in the FIR filter equations of $b_{-1,0}$ and $c_{-1,0}$. The sub-expression "-$A_{1,0}$ -$A_{2,0}$" in (5) exists in the FIR filter equation of $c_{2,0}$.

Table V shows all the common sub-expressions in the proposed HFI hardware and number of adders used to implement them. The proposed HFI hardware uses 54 adders to calculate common sub-expressions. It uses 8 adders to calculate the 8 sub-expressions that are not common in the FIR filters such as "-$A_{-1,0}$ -$A_{2,0}$" in (6), and it uses 80 adders in adder trees. Hence, it uses 142 adders.

HFI hardware proposed in [18] uses 176 adders. Hence, the proposed HFI hardware uses 19.3% less adders than the one in [18]. Since the HFI hardware proposed in [19] is memory-based hardware, the number of adders used in it is not compared with the number of adders used in the proposed hardware.

Fig. 2 shows the proposed HFI hardware for all PU sizes. The splitters represent interconnects in the proposed hardware. They are used to simplify the figure. Fig. 3 shows the proposed datapaths for implementing all sub-expressions including common sub-expressions shown in Table V. They are shown as Sub-Expressions block in Fig. 2.

The proposed hardware interpolates all FPs (HHPs, VHPs, QPs) for luma component of an 8×8 PU. The larger PU sizes are decomposed into 8×8 blocks and these 8×8 blocks are interpolated separately.

3×8×15 HHPs are interpolated in 15 CCs and stored into transpose memories. 3×8×8 VHPs are interpolated in 8 CCs. 9×8×8 QPs are interpolated in 24 CCs using the HHPs in transpose memories. There are 3 pipeline stages in the proposed hardware. Hence, all FPs for an 8×8 PU are interpolated in 50 CCs.

B. *Proposed VVC FI Hardware*

In the proposed VFI hardware, FIR filter coefficients are decomposed to other coefficients in the forms of powers of 2 as shown in Table VI. Although this increases the number of coefficients, it also increases the number of common sub-expressions which in turn reduces the number of additions. The proposed VFI hardware also uses the common offsets proposed in [21].

In the proposed VFI hardware, 8×15 FPs are interpolated in parallel in each CC using 15 IPs or 15 HHPs and 8 sets of 15 FIR filters with decomposed coefficients. There are more common sub-expressions in these 8×15 FIR filters.

TABLE III
APPROXIMATE F1 FIR FILTERS WITH OFFSET USED IN MCMF1

		A_{-1}	A_0	A_1	A_2	Required Offset
Offsets	Off_1	-8	64	8	0	
	Off_2	0	8	64	-8	
	Off_3	-8	8	8	-8	
F1 FIR Filters with Offset	$F1OF_1$	6	-1	-5	0	Off_1
	$F1OF_2$	4	-2	-2	0	Off_1
	$F1OF_3$	2	-4	2	0	Off_1
	$F1OF_4$	1	-6	5	0	Off_1
	$F1OF_5$	0	-12	12	0	Off_1
	$F1OF_6$	1	-17	16	0	Off_1
	$F1OF_7$	0	-19	19	0	Off_1
	$F1OF_8$	0	32	32	0	Off_3

TABLE IV
APPROXIMATE F2 FIR FILTERS WITH OFFSET USED IN MCMF2

		A_{-1}	A_0	A_1	A_2	Required Offset
Offsets	Off_1	-8	64	8	0	
	Off_2	0	8	64	-8	
	Off_3	-8	8	8	-8	
F2 FIR Filters with Offset	$F2OF_1$	6	0	-6	0	Off_1
	$F2OF_2$	4	0	-4	0	Off_1
	$F2OF_3$	0	0	0	0	Off_1
	$F2OF_4$	0	-8	8	0	Off_1
	$F2OF_6$	0	-24	24	0	Off_1
	$F2OF_8$	0	32	32	0	Off_3
	$F2OF_9$	0	24	-24	0	Off_2
	$F2OF_{11}$	0	8	-8	0	Off_2
	$F2OF_{13}$	0	0	0	0	Off_2
	$F2OF_{14}$	0	-4	0	4	Off_2
	$F2OF_{15}$	0	-6	0	6	Off_2

$$a_{0,0} = \left(-A_{-3,0} + 4 \times A_{-2,0} - 10 \times A_{-1,0} + 40 \times A_{0,0} + 18 \times A_{0,0} + 18 \times A_{1,0} - A_{1,0} - A_{2,0} - 4 \times A_{2,0} + A_{3,0}\right) >> 6 \quad (5)$$

$$b_{0,0} = \left(-A_{-3,0} + 4 \times A_{-2,0} - A_{-1,0} - 10 \times A_{-1,0} + 40 \times A_{0,0} + 40 \times A_{1,0} - 10 \times A_{2,0} - A_{2,0} + 4 \times A_{3,0} - A_{4,0}\right) >> 6 \quad (6)$$

$$c_{0,0} = \left(A_{-2,0} - 4 \times A_{-1,0} - A_{-1,0} - A_{0,0} + 18 \times A_{0,0} + 18 \times A_{1,0} + 40 \times A_{1,0} - 10 \times A_{2,0} + 4 \times A_{3,0} - A_{4,0}\right) >> 6 \quad (7)$$

TABLE V
COMMON SUB-EXPRESSIONS IN THE PROPOSED HEVC FI HARDWARE

General form	Sub-expressions	Adders
$-A_{x-1,0} + 4 \times A_{x,0}$	$-A_{-6,0} + 4 \times A_{-5,0}$, $-A_{-5,0} + 4 \times A_{-4,0}$, … , $-A_{3,0} + 4 \times A_{4,0}$	10
$-10 \times A_{x-1,0} + 40 \times A_{x,0}$	$-10 \times A_{-4,0} + 40 \times A_{-3,0}$, $-10 \times A_{-3,0} + 40 \times A_{-2,0}$, … , $-10 \times A_{3,0} + 40 \times A_{4,0}$	8
$4 \times A_{x-1,0} - A_{x,0}$	$4 \times A_{-2,0} - A_{-1,0}$, $4 \times A_{-1,0} - A_{0,0}$, … , $4 \times A_{7,0} - A_{8,0}$	10
$40 \times A_{x-1,0} - 10 \times A_{x,0}$	$40 \times A_{-2,0} - 10 \times A_{-1,0}$, $40 \times A_{-1,0} - 10 \times A_{0,0}$, … , $40 \times A_{5,0} - 10 \times A_{6,0}$	8
$A_{x-1,0} + A_{x,0}$	$A_{-4,0} + A_{-3,0}$, $A_{-3,0} + A_{-2,0}$, … , $A_{5,0} + A_{6,0}$	10
$18 \times A_{x-1,0} + 18 \times A_{x,0}$	$18 \times A_{-3,0} + 18 \times A_{-2,0}$, $18 \times A_{-2,0} + 18 \times A_{-1,0}$, …, $18 \times A_{4,0} + 18 \times A_{5,0}$	8
Total adders		54

Table VII shows all the common sub-expressions in the proposed VFI hardware, and the number of adders used to implement them. In each row, for each common sub-expression general form, all specific sub-expressions, which are negated or shifted versions of each other, are shown. For example, in the second row, "$8\times A_{-2}$ $-2\times A_{-1}$" and "$32\times A_{-2}$ - $8\times A_{-1}$" are obtained by shifting "$4\times A_{-2}$ $-A_{-1}$" 1 bit and 3 bits to the left, respectively.

In addition to the 94 adders used to calculate common sub-expressions shown in Table VII, the proposed VFI hardware uses 8×3 adders for calculating the offsets (Off_1, Off_2, Off_3) of all the 8 sets of 15 FIR filters, and it uses 8×58 adders in adder trees. Hence, it uses 94 +8×3 +8×58 = 582 adders. The VFI hardware proposed in [21] uses 633 adders. Hence, the proposed VFI hardware uses 8% less adders than the one proposed in [21].

Fig. 4 shows the proposed VFI hardware for all PU sizes. The splitters represent interconnects in the proposed hardware. TR MEM and OUT MEM are the transpose memories and output memories, respectively. Sub-Expressions block calculates all common sub-expressions shown in Table VII. Fig. 5 shows the proposed datapaths used in the Sub-Expressions block. Common sub-expressions "$-A_{x-3}$ $-A_{x-2}$ $-A_{x-1}$ $-A_x$" and "$-4\times A_{x-2}$ $+16\times A_{x-1}$ $+4\times A_{x-1}$ $-4\times A_x$" are calculated using other common sub-expressions. Hence, only 1 extra adder is used for implementing them.

8×15×15 HHPs are interpolated in 15 CCs and stored in transpose memories. 8×8×15 VHPs are interpolated in 8 CCs. 8×8×225 QPs are interpolated in 8×15 CCs using HHPs. There are 4 pipeline stages in the proposed hardware. Hence, all the FPs for an 8×8 PU are interpolated in 147 CCs.

C. *Proposed Approximate VVC FI Hardware DCF1*

The proposed DCF1 approximate VFI hardware implements the approximate F1 VFI FIR filters. In DCF1 hardware, the coefficients of F1 FIR filters are decomposed into other coefficients in the forms of powers of 2 as shown in Table VIII. Although this increases the number of coefficients, it also increases the number of common sub-expressions which in turn reduces the number of additions. The coefficients of common sub-expressions are shown with the same color in the table.

In DCF1 hardware, 8×15 FPs are interpolated in parallel in each CC using 15 IPs or 15 HHPs and 8 sets of 15 F1 FIR filters with decomposed coefficients. There are more common sub-expressions in these 8×15 FIR filters. Table IX shows all the common sub-expressions in DCF1 hardware, and the number of adders used to implement them.

DCF1 hardware uses 68 adders for calculating all common sub-expressions. It uses 8×3 adders to calculate the offsets (Off_1, Off_2, Off_3) of all the 8×15 FIR filters, and it uses 8×21 adders in adder trees. Hence, 68 +8×3 +8×21 = 260 adders are used in DCF1 hardware. The MCMF1 hardware proposed in [24], which implements the approximate F1 FIR filters, uses 341 adders; 38 adders in Hcub MCM blocks, 62 adders for implementing common sub-expressions, 33 adders to calculate the offsets (Off_1, Off_2, Off_3), and 208 adders in adder trees. Hence, DCF1 uses 23.75% less adders than the MCMF1.

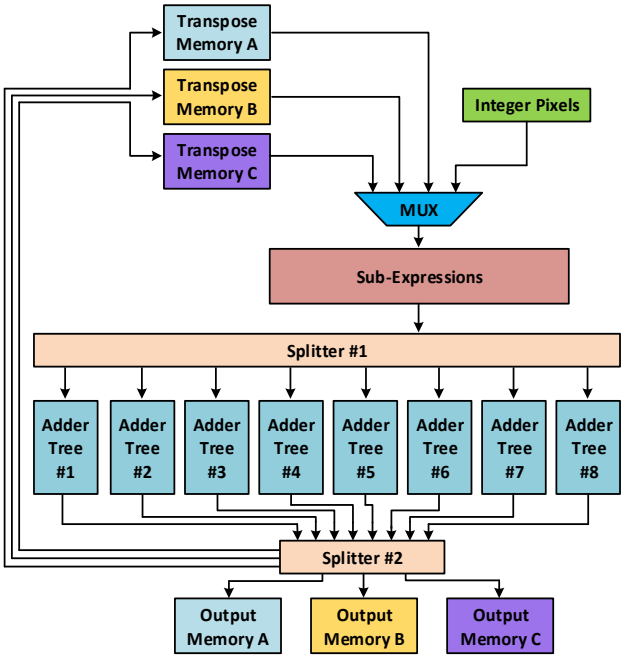

Fig. 2. Proposed HEVC FI hardware.

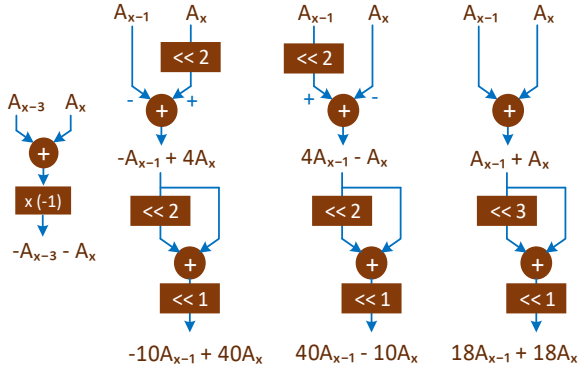

Fig. 3. Sub-expression datapaths in the proposed HEVC FI hardware.

TABLE VI
DECOMPOSED COEFFICIENTS IN THE PROPOSED VVC FI HARDWARE

	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
F_1	1	1-4	4+1	-1+32	-32+4	2+4	-4+1	1
F_2	0	-2	2+1	-2+32	-32+8	1+4	-4+1	1
F_3	0	-1	0	32-4	-4-16+1	4	-4+1	1
F_4	0	0	-2	2+8+16	-16+1	-1+4	-4+1	1
F_5	0	0	1-4	16+4	-4-2	0	-1	0
F_6	0	-1	-1	16-1	-1	-2	0	0
F_7	0	0	-1-2	8+4+1	2	-2	0	0
F_8	0	0	-1-2	8	8	-2-1	0	0
F_9	0	0	-2	2	1+4+8	-2-1	0	0
F_{10}	0	0	-2	-1	-1+16	-1	-1	0
F_{11}	0	-1	0	-2-4	16+4	-4+1	0	0
F_{12}	1	1-4	4-1	1-16	16+8+2	-2	0	0
F_{13}	1	1-4	4	1-16-4	-4+32	0	-1	0
F_{14}	1	1-4	4+1	8-32	32-2	1+2	-2	0
F_{15}	1	1-4	4+2	4-32	32-1	1+4	-4+1	1

TABLE VII
COMMON SUB-EXPRESSIONS IN THE PROPOSED VVC FI HARDWARE

General form	Sub-expressions	Adders
$-A_{x-2}+4\times A_{x-1}$	$A_{-6}-4\times A_{-5}$, $A_{-5}-4\times A_{-4}$, … , $A_4-4\times A_5$ $-2\times A_{-4}+8\times A_{-3}$, $-2\times A_{-3}+8\times A_{-2}$, … , $-2\times A_4+8\times A_5$ $-4\times A_{-4}+16\times A_{-3}$, $-4\times A_{-3}+16\times A_{-2}$, … , $-4\times A_4+16\times A_5$ $-8\times A_{-4}+32\times A_{-3}$, $-8\times A_{-3}+32\times A_{-2}$, … , $-8\times A_3+32\times A_4$	11
$4\times A_{x-1}-A_x$	$4\times A_{-3}-A_{-2}$, $4\times A_{-2}-A_{-1}$, … , $4\times A_7-A_8$ $8\times A_{-3}-2\times A_{-2}$, $8\times A_{-2}-2\times A_{-1}$, …, $8\times A_5-2\times A_6$ $32\times A_{-2}-8\times A_{-1}$, $32\times A_{-2}-8\times A_{-1}$, …, $32\times A_5-8\times A_6$	11
$A_{x-2}+A_{x-1}$	$A_{-6}+A_{-5}$, $A_{-5}+A_{-4}$, … , A_7+A_8 $-4\times A_{-3}-4\times A_{-2}$, $-4\times A_{-2}-4\times A_{-1}$, … , $-4\times A_4-4\times A_5$	14
$A_{x-1}-A_x$	$A_{-2}-A_{-1}$, $A_{-1}-A_0$, … , A_5-A_6 $-A_{-4}+A_{-3}$, $-A_{-3}+A_{-2}$, … , $-A_3+A_4$ $-2\times A_{-5}+2\times A_{-4}$, $-2\times A_{-4}+2\times A_{-3}$, … , $-2\times A_3+2\times A_4$ $2\times A_{-2}-2\times A_{-1}$, $2\times A_{-1}-2\times A_0$, … , $2\times A_6-2\times A_7$ $-4\times A_{-5}+4\times A_{-4}$, $-4\times A_{-4}+4\times A_{-3}$, … , $-4\times A_2+4\times A_3$ $4\times A_{-3}-4\times A_{-2}$, $4\times A_{-2}-4\times A_{-1}$, … , $4\times A_6-4\times A_7$ $16\times A_{-3}-16\times A_{-2}$, $16\times A_{-2}-16\times A_{-1}$, … , $16\times A_4-16\times A_5$ $-16\times A_{-3}+16\times A_{-2}$, $-16\times A_{-2}+16\times A_{-1}$, … , $-16\times A_4+16\times A_5$ $32\times A_{-3}-32\times A_{-2}$, $32\times A_{-2}-32\times A_{-1}$, … , $32\times A_4-32\times A_5$ $-32\times A_{-3}+32\times A_{-2}$, $-32\times A_{-2}+32\times A_{-1}$, … , $-32\times A_4+32\times A_5$	12
$A_{x-3}+A_x$	$A_{-4}+A_{-1}$, $A_{-3}+A_0$, … , A_3+A_6 $-A_{-4}-A_{-1}$, $-A_{-3}-A_0$, … , $-A_3-A_6$	8
$A_{x-3}-A_x$	$A_{-3}-A_0$, $A_{-2}-A_1$, … , A_4-A_7 $-A_{-5}+A_{-2}$, $-A_{-4}+A_{-1}$, … , $-A_2+A_5$	10
$A_{x-4}-A_x$	$A_{-4}-A_0$, $A_{-3}-A_1$, … , A_3-A_7 $-A_{-5}+A_{-1}$, $-A_{-4}+A_0$, … , $-A_2+A_6$	9
$-A_{x-3}-A_{x-2}-A_{x-1}-A_x$	$-A_{-5}-A_{-4}-A_{-3}-A_{-2}$, $-A_{-4}-A_{-3}-A_{-2}-A_{-1}$, … , $-A_4-A_5-A_6-A_7$	10
$-4\times A_{x-2}+16\times A_{x-1}+4\times A_{x-1}-4\times A_x$	$-4\times A_{-4}+16\times A_{-3}+4\times A_{-3}-4\times A_{-2}$, … , $-4\times A_4+16\times A_5+4\times A_5-4\times A_6$	9
Total adders		94

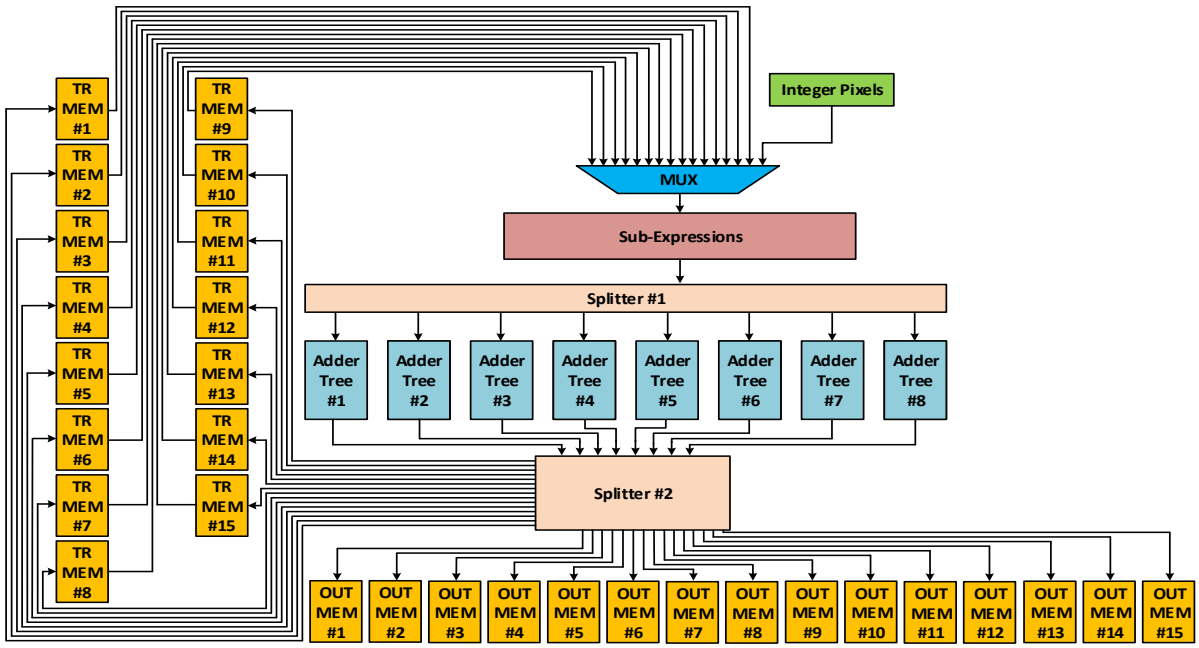

Fig. 4. Proposed VVC FI hardware.

Block diagram of DCF1 hardware is similar to block diagram of the proposed VFI hardware shown in Fig. 4. Their Sub-Expressions and Adder Tree blocks are different. Fig. 6 shows the proposed datapaths used in the Sub-Expressions block of DCF1 hardware. Some common sub-expressions are calculated using the results of other common sub-expressions.

Interpolation order and number of pipeline stages in DCF1 hardware are the same as the proposed VFI hardware. Hence, DCF1 hardware interpolates all the FPs for an 8×8 PU in 147 CCs.

D. Proposed Approximate VVC FI Hardware DCF2

The proposed DCF2 approximate VFI hardware implements the approximate F2 VFI FIR filters. In DCF2 hardware, the coefficients of F2 FIR filters are decomposed into other coefficients in the forms of powers of 2 as shown in Table X. Although this increases the number of coefficients, it also increases the number of common sub-expressions which in turn reduces the number of additions. The coefficients of common sub-expressions are shown with the same color in the table.

The coefficients are decomposed in such a way that the adder sizes are also reduced. For example, for $F2OF_4$ FIR filter, instead of implementing $[(8\times A_1 - 8\times A_0) + Off_1] >> 6$ in Table IV where $Off_1 = -8\times A_{-1} + 64\times A_0 + 8\times A_1$, we implement $[(A_1 - A_0) + Off_1] >> 3$ where $Off_1 = -A_{-1} + 8\times A_0 + A_1$.

F2 FIR filters $F2OF_4$, $F2OF_6$, $F2OF_9$, $F2OF_{11}$ are the same as $F2OF_5$, $F2OF_7$, $F2OF_{10}$, $F2OF_{12}$, respectively. Hence, in DCF2 hardware, only FIR filters $F2OF_4$, $F2OF_6$, $F2OF_9$, $F2OF_{11}$ are calculated, and their results are also used for $F2OF_5$, $F2OF_7$, $F2OF_{10}$, $F2OF_{12}$, respectively.

In DCF2 hardware, 8×15 FPs are interpolated in parallel in each CC using 15 IPs or 15 HHPs and 8 sets of 11 F2 FIR filters with decomposed coefficients. There are more common sub-expressions in these 8×11 FIR filters. Table XI shows all the common sub-expressions in DCF2 hardware, and the number of adders used to implement them.

DCF2 hardware uses 42 adders to calculate all the sub-expressions. It uses 8×3 adders to calculate the offsets (Off_1, Off_2, Off_3) of all the 8 sets of 11 FIR filters, and it uses 8×9 adders in adder trees. Hence, 42 + 8×3 +8×9 = 138 adders are used in DCF2 hardware. The MCMF2 hardware proposed in [24], which implements the approximate F2 FIR filters, uses 158 adders; 11 adders in Hcub MCM blocks, 42 adders for implementing common sub-expressions, 33 adders to calculate the offsets (Off_1, Off_2, Off_3), and 72 adders in adder trees. Hence, DCF2 hardware uses 12.65% less adders than the MCMF2 hardware.

Block diagram of DCF2 hardware is similar to the block diagram of the proposed VFI hardware shown in Fig. 4. Their Sub-Expressions block, Adder Tree block, and the numbers of OUT MEMs are different. DCF2 hardware uses 11 OUT MEMs because the F2 FIR filters $F2OF_5$, $F2OF_7$, $F2OF_{10}$, $F2OF_{12}$ are the same as other F2 FIR filters so their results are not calculated and stored. Fig. 7 shows the proposed datapaths used in the Sub-Expressions block of DCF2 hardware. Some common sub-expressions are calculated using the results of other common sub-expressions.

Interpolation order and number of pipeline stages in DCF2 hardware are the same as the proposed VFI hardware. Hence, DCF2 hardware interpolates all the FPs for an 8×8 PU in 147 CCs.

IV. Implementation Results

All the proposed HFI and VFI hardware are implemented using Verilog HDL. Verilog RTL codes of all the proposed FI hardware are synthesized, placed and routed to a 28 nm FPGA. To have a fair comparison, Verilog RTL codes of the FI hardware proposed in [18], [19], [21], and [24] are synthesized, placed and routed to the same 28 nm FPGA. The FPGA implementations are verified with post place and route simulations.

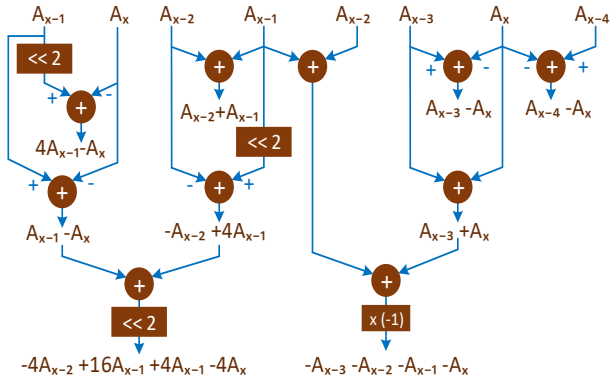

Fig. 5. Sub-expression datapaths in the proposed VVC FI hardware.

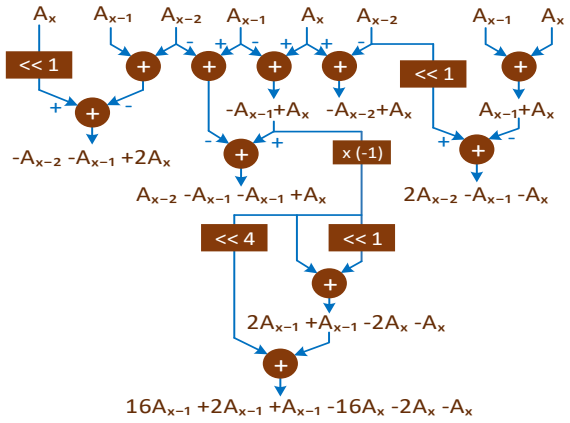

Fig. 6. Sub-expression datapaths in DCF1 hardware.

TABLE VIII
APPROXIMATE F1 FIR FILTERS WITH OFFSET USED IN DCF1 HARDWARE

		A_{-1}	A_0	A_1	A_2	Required Offset
Offsets	Off_1	-8	64	8	0	
	Off_2	0	8	64	-8	
	Off_3	-8	8	8	-8	
FIR Filters with Offset Used in DCF1	$F1OF_1$	2+4	-1	-1-4	0	Off_1
	$F1OF_2$	4	-2	-2	0	Off_1
	$F1OF_3$	2	-2-2	2	0	Off_1
	$F1OF_4$	1	-1-1-4	4+1	0	Off_1
	$F1OF_5$	0	-8-4	4+8	0	Off_1
	$F1OF_6$	1	-1-16	16	0	Off_1
	$F1OF_7$	0	-1-2-16	16+2+1	0	Off_1
	$F1OF_8$	0	32	32	0	Off_3
	$F1OF_9$	0	1+2+16	-16-2-1	0	Off_2
	$F1OF_{10}$	0	16	-16-1	1	Off_2
	$F1OF_{11}$	0	4+8	-8-4	0	Off_2
	$F1OF_{12}$	0	1+4	-4-1-1	1	Off_2
	$F1OF_{13}$	0	2	-2-2	2	Off_2
	$F1OF_{14}$	0	-2	-2	4	Off_2
	$F1OF_{15}$	0	-4-1	-1	4+2	Off_2

TABLE IX
COMMON SUB-EXPRESSIONS IN DCF1 HARDWARE

General form	Sub-expressions	Adders
$-A_{x-1}+A_x$	$-A_{-3}+A_{-2}, -A_{-2}+A_{-1}, \dots, -A_5+A_6$ $-2\times A_{-3}+2\times A_{-2}, -2\times A_{-2}+2\times A_{-1}, \dots, -2\times A_5+2\times A_6$ $-4\times A_{-3}+4\times A_{-2}, -4\times A_{-2}+4\times A_{-1}, \dots, -4\times A_4+4\times A_5$ $-8\times A_{-3}+8\times A_{-2}, -8\times A_{-2}+8\times A_{-1}, \dots, -8\times A_4+8\times A_5$ $-16\times A_{-3}+16\times A_{-2}, -16\times A_{-2}+16\times A_{-1}, \dots, -16\times A_4+16\times A_5$ $A_{-4}-A_{-3}, A_{-3}-A_{-2}, \dots, A_4-A_5$ $2\times A_{-4}-2\times A_{-3}, 2\times A_{-3}-2\times A_{-2}, \dots, 2\times A_4-2\times A_5$ $4\times A_{-3}-4\times A_{-2}, 4\times A_{-2}-4\times A_{-1}, \dots, 4\times A_4-4\times A_5$ $8\times A_{-3}-8\times A_{-2}, 8\times A_{-2}-8\times A_{-1}, \dots, 8\times A_4-8\times A_5$ $16\times A_{-3}-16\times A_{-2}, 16\times A_{-2}-16\times A_{-1}, \dots, 16\times A_4-16\times A_5$	10
$-A_{x-2}+A_x$	$-4\times A_{-3}+4\times A_{-1}, -4\times A_{-2}+4\times A_0, \dots, -4\times A_4+4\times A_6$ $-8\times A_{-4}+8\times A_{-2}, -8\times A_{-3}+8\times A_{-1}, \dots, -8\times A_3+8\times A_5$ $4\times A_{-4}-4\times A_{-2}, 4\times A_{-3}-4\times A_{-1}, \dots, 4\times A_3-4\times A_5$ $8\times A_{-3}-8\times A_{-1}, 8\times A_{-2}-8\times A_0, \dots, 8\times A_4-8\times A_6$	9
$A_{x-1}+A_x$	$-A_{-3}-A_{-2}, -A_{-2}-A_{-1}, \dots, -A_4-A_5$ $-2\times A_{-3}-2\times A_{-2}, -2\times A_{-2}-2\times A_{-1}, \dots, -2\times A_4-2\times A_5$ $32\times A_{-3}+32\times A_{-2}, 32\times A_{-2}+32\times A_{-1}, \dots, 32\times A_4+32\times A_5$	8
$A_{x-2}-A_{x-1}-A_{x-1}+A_x$	$A_{-4}-A_{-3}-A_{-3}+A_{-2}, A_{-3}-A_{-2}-A_{-1}+A_0, \dots, A_3-A_4-A_5+A_6$	9
$2\times A_{x-2}-A_{x-1}-A_x$	$2\times A_{-4}-A_{-3}-A_{-2}, 2\times A_{-3}-A_{-2}-A_{-1}, \dots, 2\times A_3-A_4-A_5$ $4\times A_{-4}-2\times A_{-3}-2\times A_{-2}, 4\times A_{-3}-2\times A_{-2}-2\times A_{-1}, \dots, 4\times A_3-2\times A_4-2\times A_5$	8
$-A_{x-2}-A_{x-1}+2\times A_x$	$-A_{-3}-A_{-2}+2\times A_{-1}, -A_{-2}-A_{-1}+2\times A_0, \dots, -A_4-A_5+2\times A_6$ $-2\times A_{-3}-2\times A_{-2}+4\times A_{-1}, -2\times A_{-2}-2\times A_{-1}+4\times A_0, \dots, -2\times A_4-2\times A_5+4\times A_6$	8
$2\times A_{x-1}+A_{x-1}-2\times A_x-A_x$	$2\times A_{-3}+A_{-3}-2\times A_{-2}-A_{-2}, \dots, 2\times A_4+A_4-2\times A_5-A_5$ $-2\times A_{-3}-A_{-3}+2\times A_{-2}+A_{-2}, \dots, -2\times A_4-A_4+2\times A_5+A_5$ $8\times A_{-3}+4\times A_{-3}-8\times A_{-2}-4\times A_{-2}, \dots, 8\times A_4+4\times A_4-8\times A_5-4\times A_5$ $-8\times A_{-3}-4\times A_{-3}+8\times A_{-2}+4\times A_{-2}, \dots, -8\times A_4-4\times A_4+8\times A_5+4\times A_5$	8
$16\times A_{x-1}+2\times A_{x-1}+A_{x-1}-16\times A_x-2\times A_x-A_x$	$16\times A_{-3}+2\times A_{-3}+A_{-3}-16\times A_{-2}-2\times A_{-2}-A_{-2}, \dots,$ $16\times A_4+2\times A_4+A_4-16\times A_5-2\times A_5-A_5$	8
Total adders		68

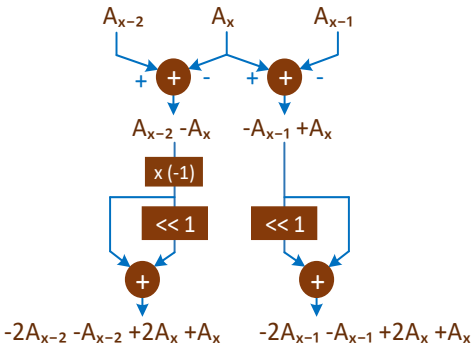

Fig. 7. Sub-expression datapaths in DCF2 hardware.

The implementation results of HFI hardware are shown in Table XII. The power consumption results are shown in Table XII and Table XIII. The results shown as "---" have not been reported in the corresponding papers. The proposed HFI hardware has less area and less power consumption than the HFI hardware in the literature. The proposed HFI hardware has higher performance than the manual HFI hardware implementations proposed in [15]-[19]. Although the HFI HLS implementation proposed in [20] has higher performance than the proposed HFI hardware, it uses more than 10 times LUTs.

The implementation and power consumption results of VFI hardware on the same 28 nm FPGA are shown in Table XIV and Table XV, respectively. Power consumptions of all FPGA implementations are estimated using a gate level power estimation tool for 1 frame of 1920×1080 Tennis (T) video and 1920×1080 Kimono (K) video at 100 MHz [27].

The proposed exact VFI hardware has higher performance, less area, and up to 4.78% less power consumption than the best exact manual VFI hardware [21]. Although the VFI HLS implementation proposed in [25] has higher performance than the proposed VFI hardware, its area is 4 times larger. The proposed approximate VFI hardware DCF1 and DCF2 have the same performance, less area, and up to 8.92% and 8.50% less power consumption than the approximate VFI hardware MCMF1 and MCMF2 proposed in [24], respectively.

TABLE X
APPROXIMATE F2 FIR FILTERS WITH OFFSET USED IN DCF2 HARDWARE

		A_{-1}	A_0	A_1	A_2	Required Offset	Required Final Shift
Offsets	Off_1	-1	8	1	0		
	Off_2	0	1	8	-1		
	Off_3	-1	1	1	-1		
FIR Filters with Offset Used in DCF2	$F2OF_1$	2+1	0	-2-1	0	Off_1 >>2	>>5
	$F2OF_2$	1	0	-1	0	Off_1 >>1	>>4
	$F2OF_3$	0	0	0	0	Off_1	>>3
	$F2OF_4$	0	-1	1	0	Off_1	>>3
	$F2OF_6$	0	-2-1	2+1	0	Off_1	>>3
	$F2OF_8$	0	4	4	0	Off_3	>>3
	$F2OF_9$	0	2+1	-2-1	0	Off_2	>>3
	$F2OF_{11}$	0	1	-1	0	Off_2	>>3
	$F2OF_{13}$	0	0	0	0	Off_2	>>3
	$F2OF_{14}$	0	-1	0	1	Off_2 >>1	>>4
	$F2OF_{15}$	0	-2-1	0	2+1	Off_2 >>2	>>5

TABLE XI
COMMON SUB-EXPRESSIONS IN DCF2 HARDWARE

General form	Sub-expressions	Adders
$A_{x-2} - A_x$	$A_{-4} - A_{-2}$, $A_{-3} - A_{-1}$, … , $A_4 - A_6$ $2 \times A_{-4} - 2 \times A_{-2}$, $2 \times A_{-3} - 2 \times A_{-1}$, … , $2 \times A_4 - 2 \times A_6$ $-A_{-4} + A_{-2}$, $-A_{-3} + A_{-1}$, … , $-A_4 + A_6$	9
$-A_{x-1} + A_x$	$-A_{-3} + A_{-2}$, $-A_{-2} + A_{-1}$, … , $-A_4 + A_5$ $-2 \times A_{-3} + 2 \times A_{-2}$, $-2 \times A_{-2} + 2 \times A_{-1}$, … , $-2 \times A_4 + 2 \times A_5$ $A_{-3} - A_{-2}$, $A_{-2} - A_{-1}$, … , $A_4 - A_5$ $2 \times A_{-3} - 2 \times A_{-2}$, $2 \times A_{-2} - 2 \times A_{-1}$, … , $2 \times A_4 - 2 \times A_5$	8
$-2 \times A_{x-1} - A_{x-1} + 2 \times A_x + A_x$	$-2 \times A_{-3} - A_{x-3} + 2 \times A_{-2} + A_{-2}$, … , $-2 \times A_4 - A_4 + 2 \times A_5 + A_5$ $2 \times A_{-3} + A_{x-3} - 2 \times A_{-2} - A_{-2}$, … , $2 \times A_4 + A_4 - 2 \times A_5 - A_5$	8
$-2 \times A_{x-2} - A_{x-2} + 2 \times A_x + A_x$	$-2 \times A_{-4} - A_{-4} + 2 \times A_{-2} + A_{-2}$, … , $-2 * A_4 - A_4 + 2 * A_6 + A_6$ $2 \times A_{-4} + A_{-4} - 2 \times A_{-2} - A_{-2}$, … , $2 \times A_4 + A_4 - 2 \times A_6 - A_6$	9
Total adders		34

TABLE XII
IMPLEMENTATION RESULTS OF HEVC FI HARDWARE

	[15]	[16]	[17]	[18]	[19]	[20]	Proposed
FPGA (nm)	65	40	65	28	28	40	28
Slices	---	---	---	1349	1370	---	1196
FFs	---	---	2550	3892	3909	---	3747
LUTs	28486	24202	5017	2863	3345	27100	2510
36K BRAM	---	---	2	3	3.5	---	3
Max. Freq. (MHz)	120	200	283	230	244	313	323
Frames per Second	---	60 1920×1080	30 2560×1600	36 3840×2160	37 3840×2160	191 3840×2160	50 3840×2160
Power Consumption (mW)	---	171	89	196	210	---	83

TABLE XIV
IMPLEMENTATION RESULTS OF VVC FI HARDWARE

	VVC FI - MCM [21]	[25]	Proposed Exact VVC FI	MCMF1 [24]	Proposed DCF1	MCMF2 [24]	Proposed DCF2
Exact (E) / Approximate (A)	E	E	E	A	A	A	A
Slices	3121	15319	2970	2047	1934	2001	1851
FFs	3589	37450	4167	3394	3089	2326	2272
LUTs	10731	39047	9951	7112	6467	6725	6493
36K BRAM	30	30	30	30	30	26	26
Max. Freq. (MHz)	219	150	230	237.5	237.5	246.9	249.3
Clock Cycles (8×8 PU)	147	74	147	147	147	147	147
Frames per Second	46 1920×1080	62 1920×1080	48 1920×1080	49 1920×1080	49 1920×1080	51 1920×1080	52 1920×1080

TABLE XV
POWER CONSUMPTION RESULTS OF VVC FI HARDWARE (MW)

	VVC FI - MCM [21]		Proposed Exact VVC FI		MCMF1 [24]		Proposed DCF1		MCMF2 [24]		Proposed DCF2	
Video	T	K	T	K	T	K	T	K	T	K	T	K
Clock	25	25	27	27	22	22	20	20	17	16	17	17
Signal	172	238	174	238	60	83	53	73	65	87	59	76
Logic	203	288	185	253	58	82	48	65	56	77	47	62
BRAM	137	138	137	138	134	138	134	138	111	114	111	114
Total	537	689	523	656	274	325	255	296	249	294	234	269

V. CONCLUSIONS

In this paper, a novel technique is proposed for implementing HFI and VFI. Exact HFI hardware, exact VFI hardware, and approximate VFI hardware are designed and implemented using the proposed technique. The proposed exact HFI and exact VFI hardware have higher performance, less area, and less power consumption than the best exact HFI and exact VFI hardware, respectively. The proposed approximate VFI hardware have the same performance, less area, and less power consumption than the best approximate VFI hardware.

TABLE XIII
POWER CONSUMPTION RESULTS OF HEVC FI HARDWARE (MW)

	[18]		[19]		Proposed	
Video	T	K	T	K	T	K
Clock	11	11	9	9	11	11
Signal	143	226	144	225	35	51
Logic	26	39	25	37	21	31
BRAM	16	16	32	33	16	16
Total	196	292	210	304	83	109

REFERENCES

[1] B. Bross, W. J. Han, J. R. Ohm, G. J. Sullivan, Y. K. Wang, and T. Wiegand, "High Efficiency Video Coding (HEVC) Text Specification Draft 10," *JCTVC-L1003*, Feb. 2013.

[2] K. Singh and S. R. Ahamed, "Low power motion estimation algorithm and architecture of HEVC/H.265 for consumer applications," *IEEE Trans. on Consumer Electronics*, vol. 64, no. 3, pp. 267-275, Aug. 2018.

[3] Z. Pan, J. Lei, Y. Zhang, and F. L. Wang, "Adaptive fractional-pixel motion estimation skipped algorithm for efficient HEVC motion estimation," *ACM Trans. on Multimedia Computing, Communications, and Applications (TOMM)*, vol. 14, no. 1, pp. 1-19, Jan. 2018.

[4] A. Singhadia, M. Mamillapalli, and I. Chakrabarti, "Hardware-efficient 2D-DCT/IDCT architecture for portable HEVC-compliant devices," *IEEE Trans. on Consumer Electronics*, vol. 66, no. 3, pp. 203-212, Jul. 2020.

[5] S. Gogoi and R. Peesapati, "Design and Implementation of an Efficient Multi-Pattern Motion Estimation Search Algorithm for HEVC/H. 265," *IEEE Trans. on Consumer Electronics*, vol. 67, no. 4, pp. 319-328, Nov. 2021.

[6] 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.

[7] J. Chen, E. Alshina, G. J. Sullivan, J. R. Ohm, and J. Boyce, "Algorithm description of joint exploration model 7," *JVET-G1001*, Jul. 2017.

[8] 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.

[9] 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.

[10] B. Bross, Y. K. Wang, Y. Ye, S. Liu, J. Chen, G. J. Sullivan, and J. R. Ohm, "Overview of the versatile video coding (VVC) standard and its applications," *IEEE Trans. on Circuits and Systems for Video Technology*, vol. 31, no. 10, pp. 3736-3764, Oct. 2021.

[11] A. Wieckowski, J. Brandenburg, B. Bross, and D. Marpe, "VVC Search Space Analysis including an Open, Optimized Implementation," *IEEE Trans. on Consumer Electronics*, Feb. 2022.

[12] Y. Fan, Y. Zeng, H. Sun, J. Katto, and X. Zeng, "A pipelined 2D transform architecture supporting mixed block sizes for the VVC standard," *IEEE Trans. on Circuits and Systems for Video Technology*, vol. 30, no. 9, pp. 3289-3295, Aug. 2019.

[13] W. Hamidouche, T. Biatek, M. Abdoli, E. Francois, F. Pescador, M. Radosavljevic, D. Menard, and M. Raulet, "Versatile video coding standard: A review from coding tools to consumers deployment," *IEEE Consumer Electronics Magazine*, Jan. 2022.

[14] M. Viitanen, J. Sainio, A. Mercat, A. Lemmetti, and J. Vanne, "From HEVC to VVC: the First Development Steps of a Practical Intra Video Encoder," *IEEE Trans. on Consumer Electronics*, Jan. 2022.

[15] C. Y. Lung and C. A. Shen, "A high-throughput interpolator for fractional motion estimation in high efficient video coding (HEVC) systems," in *IEEE Asia Pacific Conf. on Circuits and Systems (APCCAS)*, pp. 268-271, Nov. 2014.

[16] G. Pastuszak and M. Trochimiuk, "Algorithm and architecture design of the motion estimation for the H. 265/HEVC 4K-UHD encoder," *Journal of Real-Time Image Processing*, vol. 12, no. 2, pp. 517-529, Aug. 2016.

[17] C. M. Diniz, M. Shafique, S. Bampi, and 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.

[18] E. Kalali and I. Hamzaoglu, "A low energy HEVC sub-pixel interpolation hardware," in *IEEE Int. Conf. on Image Process.*, pp. 1218-1222, Oct. 2014.

[19] A. C. Mert, E. Kalali, and I. Hamzaoglu, "An HEVC fractional interpolation hardware using memory based constant multiplication," in *IEEE Int. Conf. on Consumer Electronics (ICCE)*, Jan. 2018.

[20] P. Sjövall, M. Rasinen, A. Lemmetti, and J. Vanne, "High-Level Synthesis Implementation of an Accurate HEVC Interpolation Filter on an FPGA," in *IEEE Nordic Circuits and Systems Conf. (NorCAS)*, pp. 1-7, Oct. 2021.

[21] A. C. Mert, E. Kalali, and I. Hamzaoglu, "A low power versatile video coding (VVC) fractional interpolation hardware," in *Conf. on Design and Architecture for Signal and Image Process. (DASIP)*, Oct. 2018.

[22] H. Azgin, A. C. Mert, E. Kalali, and I. Hamzaoglu, "A reconfigurable fractional interpolation hardware for VVC motion compensation," in *Euromicro Conf. on Digital System Design*, Oct. 2018.

[23] H. Azgin, E. Kalali, and I. Hamzaoglu, "An approximate versatile video coding fractional interpolation hardware," in *IEEE Int. Conf. on Consumer Electronics (ICCE)*, Jan. 2020.

[24] H. Mahdavi, H. Azgin, and I. Hamzaoglu, "Approximate versatile video coding fractional interpolation filters and their hardware implementations," *IEEE Trans. on Consumer Electronics*, vol. 67, no. 3, Aug. 2021.

[25] I. Hamzaoglu, H. Mahdavi, and E. Taskin, "FPGA Implementations of VVC Fractional Interpolation Using High-Level Synthesis," in *IEEE Int. Conf. on Consumer Electronics (ICCE)*, Jan. 2022.

[26] H. Mahdavi and I. Hamzaoglu, "An efficient HEVC fractional interpolation hardware," in *IEEE Int. Conf. on Consumer Electronics (ICCE)*, Jan. 2021.

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

Hossein Mahdavi received B.S. degree in Electrical and Electronics Engineering from Azad University, Iran, in 2006, and M.S. degree in Digital Electronics Engineering from Shahid Beheshti University, Tehran, Iran, in 2017. He is currently a Ph.D. student in Electronics Engineering at Sabanci University, Istanbul, Turkey. His research interests include digital VLSI design for video processing and coding.

Ilker Hamzaoglu (M'00-SM'12) received B.S. and M.S. degrees in Computer Engineering from Bogazici University, Istanbul, Turkey in 1991 and 1993, respectively. He received Ph.D. degree in Computer Science from University of Illinois at Urbana-Champaign, IL, USA in 1999. He worked as a Senior and Principle Staff Engineer at Motorola Inc. in Schaumburg, IL, USA between August 1999 and August 2003. He started working as an Assistant Professor at Sabanci University, Istanbul, Turkey in September 2003, where he is currently working as an Associate Professor. His research interests include low power digital hardware design for video processing and compression, System-on-Chip (SoC) ASIC and FPGA design, approximate computing, high level synthesis.