

Architecture design of the high-throughput compensator and interpolator for the H.265/HEVC encoder

Grzegorz Pastuszak · Maciej Trochimiuk

Received: 13 December 2013 / Accepted: 29 March 2014 / Published online: 16 April 2014
© The Author(s) 2014. This article is published with open access at Springerlink.com

Abstract This paper presents the architecture of the high-throughput compensator and the interpolator used in the motion estimation of the H.265/HEVC encoder. The architecture can process 8×8 blocks in each clock cycle. The design allows the random order of checked coding blocks and motion vectors. This feature makes the architecture suitable for different search algorithms. The interpolator embeds 64 multiplierless reconfigurable filter cores to support computations for different fractional-pel positions. Synthesis results show that the design can operate at 200 and 400 MHz when implemented in FPGA Arria II and TSMC 90 nm, respectively. The computational scalability enables the proposed architecture to trade the throughput for the compression efficiency. If 2160p@30fps video is encoded, the design clocked at 400 MHz can check about 100 motion vectors for 8×8 blocks.

Keywords Video coding · Interpolation · Motion estimation · H.265/HEVC · FPGA · Very large-scale integration (VLSI)

1 Introduction

The latest research and standardization efforts in video coding have led to the specification of the H.265/HEVC standard [1, 2]. In terms of the rate-distortion efficiency, it

provides the improvement of about 35–50 % over its predecessor H.264/AVC [3] at the price of increased computational complexity. Although the general structure of encoder and decoder remains the same, there are many changes in the algorithm of each module. In the case of the subpixel motion estimation and compensation, the size of prediction blocks can be selected from $8 \times 4/4 \times 8$ to 64×64 . Also, there are new interpolation schemes to compute fractional-pel positions. Compared to H.264/AVC, the new interpolation provides the improvement of inter coding by 0.144 dB, on average [4]. H.264/AVC and H.265/HEVC standards employ fractional-sample interpolation of reference pictures with the quarter-pixel accuracy. The H.264/AVC standard uses six-tap luma filtering of half-pel positions followed by linear interpolation for quarter-pel positions. Chroma samples are computed by weighed interpolation of four closest integer-pel samples. In H.265/HEVC, seven-tap and eight-tap filters are used for the luma interpolation of half-pel and quarter-pel positions, respectively. Chroma samples are computed using four-tap filters. Filter coefficients are provided in Table 1.

Design and implementation of digital filters is a thoroughly explored issue, especially for small-tap filters. Moreover, digital signal processors are well suited for high-speed filtering since their architectures are adjusted to vector and matrix computations using many parallel multiply-and-accumulate units. Nevertheless, computational resources of processors are often not sufficient to support compression-efficient encoding, especially for high resolutions. Therefore, dedicated hardware accelerators are necessary.

Except one design [5], architectures for the motion estimation consist of two parts assigned to the integer-pel and fractional-pel search [6–15]. This approach requires

G. Pastuszak (✉) · M. Trochimiuk
Institute of Radioelectronics, Warsaw University of Technology,
Nowowiejska 15/19, 00-665 Warsaw, Poland
e-mail: G.Pastuszak@ire.pw.edu.pl

M. Trochimiuk
e-mail: M.Trochimiuk@ire.pw.edu.pl

Table 1 Filter coefficients in the H.265/HEVC interpolator

Filter type	Reference sample index							
	-3	-2	-1	0	1	2	3	4
Luma 1/4	-1	4	-10	58	17	-5	1	
Luma 1/2	-1	4	-11	40	40	-11	4	-1
Luma 3/4		1	-5	17	58	-10	4	-1
Chroma 1/8			-2	58	10	-2		
Chroma 1/4			-4	54	16	-2		
Chroma 3/8			-6	46	28	-4		
Chroma 1/2			-4	36	36	-4		
Chroma 5/8			-4	28	46	-6		
Chroma 3/4			-2	16	54	-4		
Chroma 7/8			-2	10	58	-2		

separate reference-pixel buffers for each part. The integer-pel part usually applies the hierarchical strategies to extend the search range, which involves quality losses. Most architectures use non-adaptive search patterns and their resource consumption is large [6–10]. The architecture supporting Multipoint Diamond Search proposed in [11] requires less resource. However, it supports only 16×16 blocks, limiting the compression efficiency.

Some high-throughput interpolators have been proposed in the literature for H.264/AVC [5–9]. Their scheduling assumes two successive steps, one for the half-pel interpolation and another for the quarter-pel interpolation. This approach is natural in terms of the specification of quarter-pel computations which refer to the results of half-pel computations. This dataflow cannot be applied directly in H.265/HEVC since quarter-pel samples are computed using separate filters. In particular, more filters are needed in the second step. Furthermore, the hardware cost increases due to a larger number of filter taps and much higher throughputs required (more partitioning modes). Some interpolator architectures have been described in publications [12–15]. They achieve throughputs suitable for video resolutions from 1080 to 4320p. Their scheduling assumes that interpolation is performed only around one point selected at the integer-pel stage for a given prediction unit. This limits the search flexibility and scalability. Moreover, three designs [12–14] are based on the assumption that the size of prediction units is selected. If the processing of more sizes has to be performed, the throughput is decreased accordingly. One design [15] supports three prediction block sizes (16×16 , 32×32 , and 64×64). On the other hand, it consumes a large amount of hardware resources. In general, obtaining a compression-efficient and high-throughput implementation requires more hardware resources and increases power consumption. Therefore, there is a need for solutions optimizing these parameters.

This work presents the high-throughput architecture of the compensator and the interpolator dedicated to the H.265/HEVC encoder. Similarly to the previous work dedicated to H.264/AVC [5], the modules can check one motion vector for an 8×8 block in each clock cycle. This feature makes the architecture suitable for different fast search algorithms as the order of motion vectors does not affect the architecture. Moreover, the computational scalability enables the tradeoff between the throughput for the compression efficiency. The present work has three novel contributions. Firstly, the fine-level search range is extended, whereas the memory cost is reduced. Secondly, the interpolation for a given prediction block can be performed in parallel with the integer-pel search. Thirdly, the proposed reconfigurable filter core for H.265/HEVC can process both luma and chroma.

The proposed dataflow reduces the size of on-chip memories 16 times while preserving the random order of checked coding blocks and fractional-accuracy motion vectors. The interpolator embeds 64 multiplierless reconfigurable filter cores to support computations for different fractional-pel positions.

The rest of the paper is organized as follows: Sect. 2 reviews previous developments on the hardware design of the adaptive motion estimation. Section 3 describes the new architecture of the motion estimation system for H.265/HEVC. Scheduling of the interpolation is presented in Sect. 4. Section 5 concentrates on the design of the reconfigurable filter core used in the interpolator. Section 6 provides implementations results. Finally, the paper is concluded in Sect. 7.

2 Design for adaptive motion estimation

The adaptive computationally scalable motion estimation algorithm allows the H.264/AVC encoders to achieve efficiencies close to optimal in real-time conditions [16]. The algorithm can employ different search strategies to adapt to local motion activity, and the number of checked search points is set by the encoder controller for each macroblock. The algorithm can achieve results close to optimum even if the number of search points assigned to macroblocks is strongly limited and varies with time. The block diagram of the architecture supporting the adaptive computationally scalable motion estimation at the fine level [5] is depicted in Fig. 1. For clarity, the coarse-level engine is omitted. Note that the coarse level in the hierarchical motion estimation enables a wider search range at the lower accuracy of motion vectors. The architecture includes the interpolator and the motion vector generator. The remaining elements build the compensator. The design allows the adaptive search for both integer-pel and

Fig. 1 Architecture of the adaptive motion estimation system

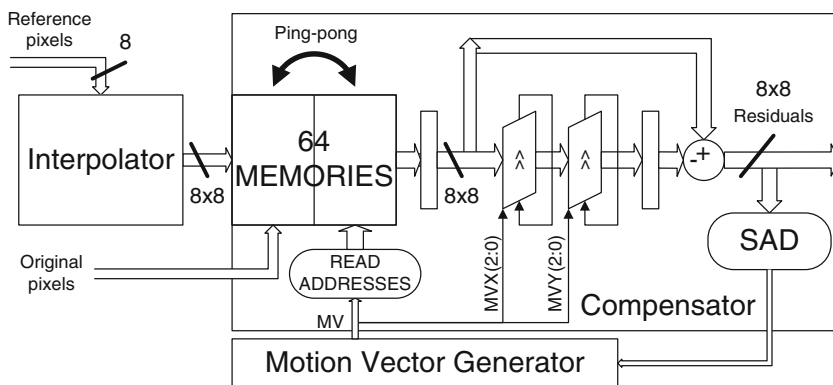
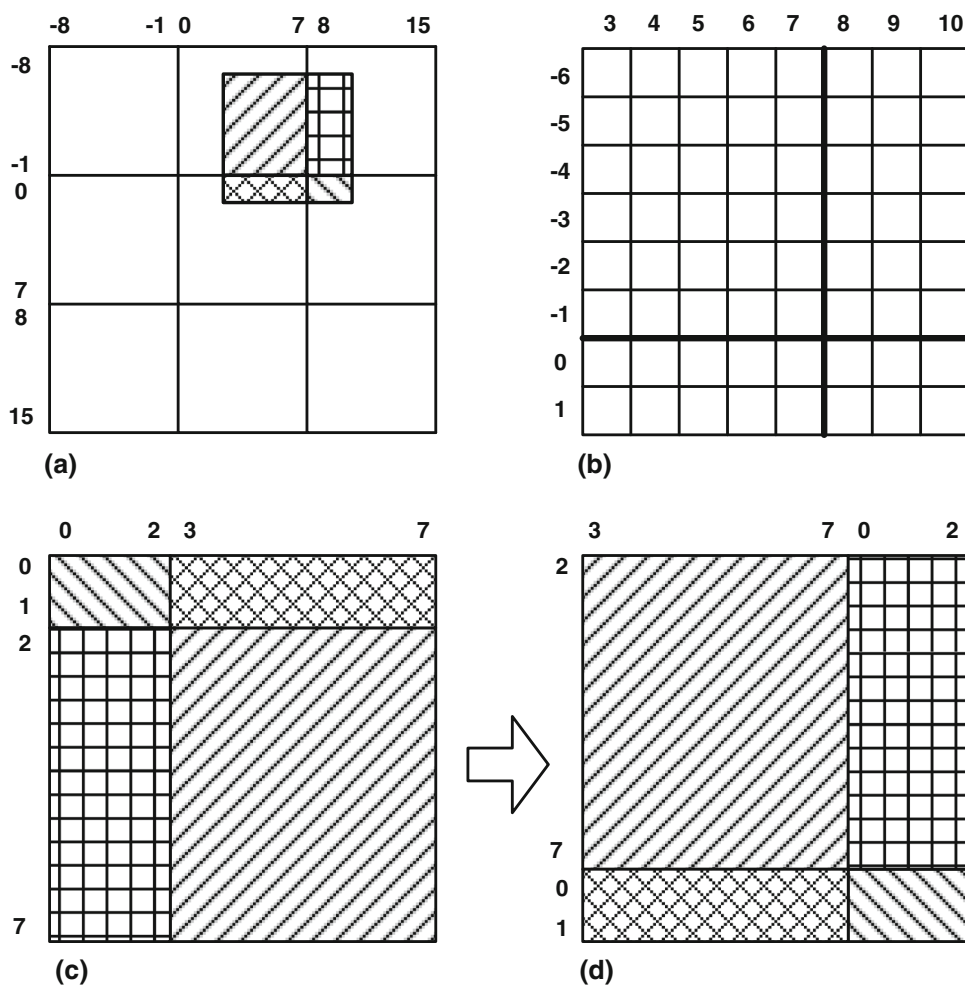


Fig. 2 Sample arrangement of the 8×8 block for MV = (3, -6): **a** location within search area; **b** block samples with their search-area indices; **c** block samples read from memories with 2D memory indices; **d** block samples after the rotation with 2D memory indices



fractional-pel positions as the interpolated pixels are loaded to 64 on-chip memories along with integer-pel ones. Particularly, the interpolator processes the reference area prior to the search algorithm performed by the motion vector generator and the compensator. Each memory keeps every eighth sample both in the horizontal and vertical dimension, and each fraction-pel sample is stored in the same

memory as its closest top-left integer neighbor. In general, samples for a given MV can belong to four adjacent 8×8 blocks (see Fig. 2a, b). Since MVs vary in the whole search range, data control is enhanced. Firstly, integer parts of read address coordinates are incremented for memories keeping samples located at the bottom and right sides of the 8×8 search-area grid lines. Secondly, two shifters rotate

samples between block positions in the horizontal and vertical dimension to restore their space consistency (see Fig. 2c, d).

Original pixels and intra predictions are buffered in the same memories as the interpolated reference area. As a consequence, the joint size of the memories is significant even when the search range is small. For example, each of the 64 memory modules needs 16k bits when using one reference frame and the search range of $[-8, 7]$ in both dimensions. In the case of the H.265/HEVC encoder, the memory capacity would be much greater since the processing on 16×16 -pixel macroblocks is replaced by coding tree units with sizes up to 64×64 pixels. Taking into account the wider search range of $[-32, 31]$ in both dimensions, the memory capacity could be increased 16 times. If the architecture is restricted to the luma component, the increase can be reduced to four. However, such a design still would be inefficient in terms of silicon area and power consumption. The interpolator developed for the considered H.265/HEVC architecture is about three to four times more complex [4] than that used in H.264/AVC. This increase stems from the greater number and order of interpolation filters applied.

3 New architecture

The dataflow is modified to reduce the memory capacity in the H.265/HEVC adaptive ME system described in the previous section. Instead of storing interpolated pixels in the memories, the architecture can compute fractional-pel samples after reading integer-pel ones from the memories. For a given search range, this approach decreases the memory space assigned to luma samples 16 times. The interpolated area of each chroma component occupies the same memory space as luma one regardless of the chroma subsampling format. If the interpolation is performed after reading (as in the modified architecture), the reduction of the memory space for chroma is greater than for luma. Particularly, the 4:2:0 video format enables the 64-fold reduction. The lower memory cost allows wider search ranges, subsequently removing the need for the hierarchical search.

The proposed architecture embeds 64 memory modules to store the 256×160 reference pixels (luma and chroma). Instead of the ping-pong scheme, data access is based on ‘moving windows’, as shown in Fig. 3. The memory space is divided into four subspaces each of which has the size of 64×160 pixels. One of the subspaces is assigned to the write port, whereas the remaining ones to the read port. The assignment is fixed until the processing for a given coding tree unit is in progress. When the motion estimation for the next coding tree unit is started, the windows are moved

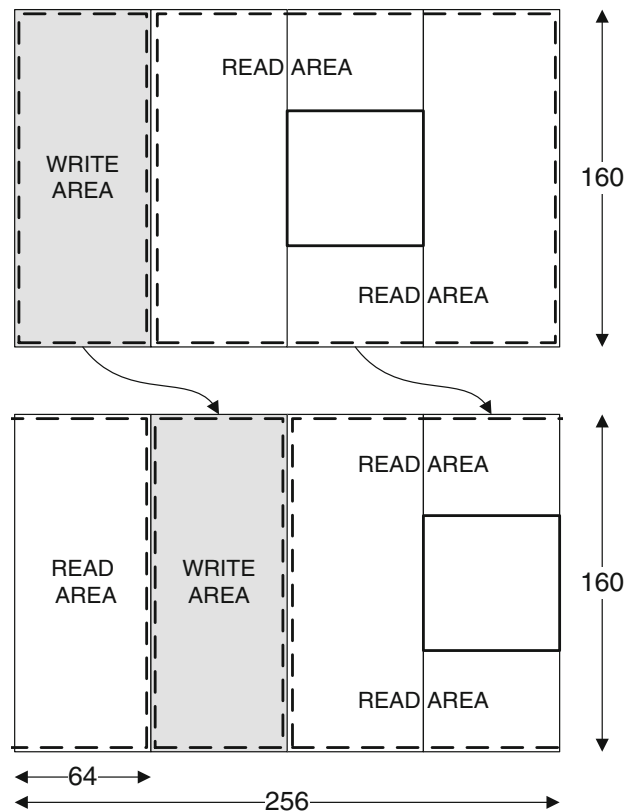


Fig. 3 The assignment of the write and read ports to memory subspaces in the ‘moving windows’ scheme

right by 64 pixels (the width of one subspace). The subspace assigned to the write port is filled with reference pixels which become the part of the search area for the next three coding tree units. In the meantime, the read port is used to access the search area. The size of the 192×160 -pixel area assigned to the read port enables the search range of $(-64, 63) \times (-48, 47)$.

In the adaptive computationally scalable motion estimation system applied for the H.264/AVC [5], original and reference 8×8 blocks are read alternately from the same memories. This dataflow is inconvenient for high-resolution videos since the throughput is limited by memory access. In the proposed architecture, separate memories for original and reference pixels allow a doubled access rate. If MVs are checked for the same 8×8 blocks continuously, the memory with original samples should remain unchanged. This gives the opportunity to reduce power consumption.

The interpolation using reference samples read from memories has a significant impact on the architecture of the adaptive ME system, which is depicted in Fig. 4. The main path of the compensator bypasses the interpolator and can support the integer-pel motion estimation with the throughput of one 8×8 block per clock cycle. For fractional-pel luma, two or four 8×8 blocks

Fig. 4 Architecture of the adaptive motion estimation system with the interpolation on samples read from memories

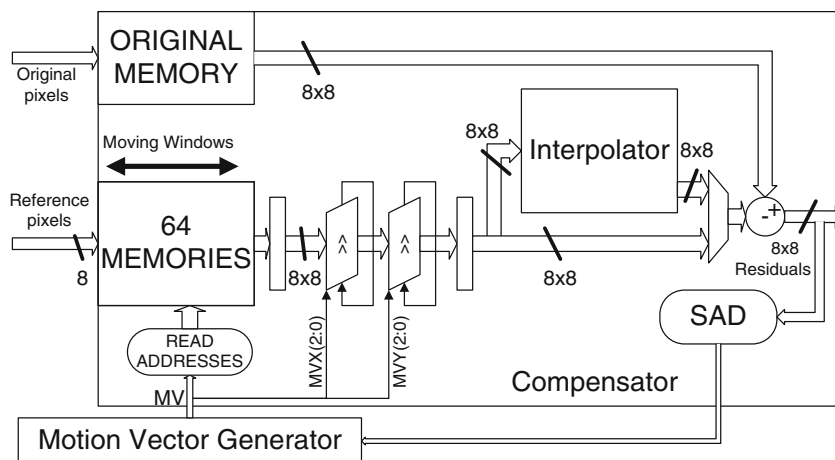
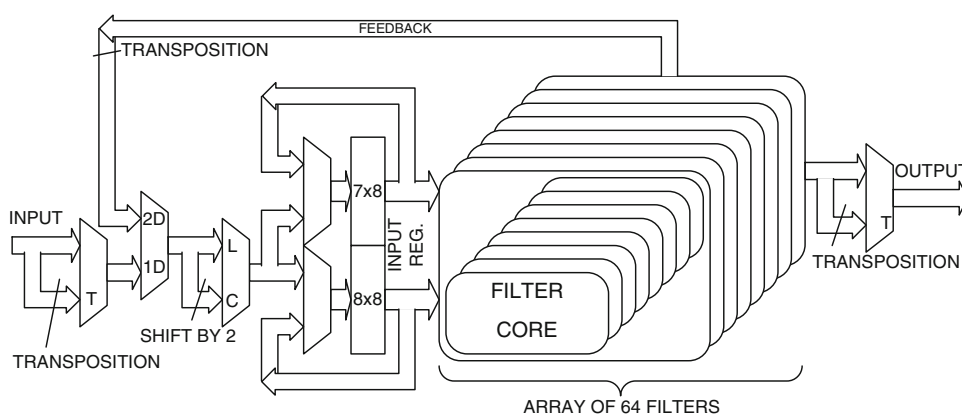


Fig. 5 Interpolator architecture. Each connection carries an 8x8 sample block



are forwarded to the interpolator in successive clock cycles. The interpolated blocks appear at the interpolator output after a number of clock cycles. The delay is dependent on the interpolation type, as described in the following section.

The general architecture of the interpolator is depicted in Fig. 5. To process blocks of 8x8 samples, the interpolator embeds 64 reconfigurable filters. The reconfiguration allows the computation of four fractional-pel positions (e.g., 0, 1/4, 1/2, and 3/4) for both luma and chroma samples. The filters take samples from 15x8 input registers. Register control is accomplished by a set of multiplexers. Firstly, data kept in input registers remain unchanged if the interpolation is performed for either horizontally or vertically neighboring fractional samples in consecutive clock cycles. Secondly, chroma samples are written to the input register through shifting by two horizontal/vertical positions. Thirdly, if a motion vector has both components with non-zero fractions, the feedback loop is selected as the input to registers after the horizontal processing. Fourthly, 8x8 blocks at the input and output interfaces can be transposed to select between the horizontal and vertical interpolation.

4 Scheduling

Interpolation filters specified in H.265/HEVC refer up to eight samples located in row/column at neighboring pixel positions. Apart from the corresponding integer-pel position, three preceding and four following samples are accessed. Thus, the 1D horizontal and vertical interpolations can be accomplished with access to the 15x8 and 8x15 areas, respectively. To access the area in the same clock cycle, the interpolator architecture embeds 15x8 input 16-bit registers. Provided that 8x8 blocks appear at the interpolator input, two cycles are taken to load the registers. The blocks are taken from the main datapath in the compensator, and their location can be identified by specific motion vectors, as shown in Fig. 6. For the convenience, the following description will refer to motion vector differences (MVDs) from the integer-pel position around which the fractional-pel search is executed. If the compensator provides two adjacent 8x8 blocks obtained for MVDs equal to (-3, 0) and (5, 0), the interpolator can compute MVDs equal to (0, 1/4), (0, 1/2), and (0, 3/4). The two accessed blocks allow the horizontal extension required by filters. Decrementing the horizontal MV

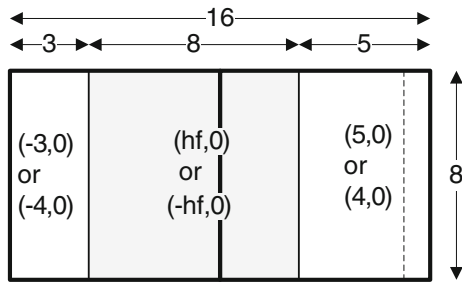


Fig. 6 Locations of input and output blocks for the 1D horizontal interpolation. ‘*hf*’ denotes any of fractions 1/2, 1/4, and 3/4. The dashed line indicates the 15×8 area

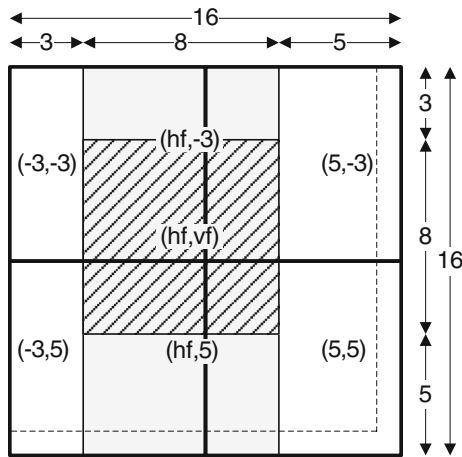


Fig. 7 Locations of input and output blocks for the 2D interpolation. ‘*hf*’ and ‘*vf*’ denote any of fractions 1/2, 1/4, and 3/4. Dashed lines indicate the 15×15 area

components of input blocks [i.e., (−4, 0) and (4, 0)] allows the computation for negative fractional-pel positions. The 1D vertical interpolation involves two modifications compared to the horizontal one. Firstly, components of MVDs are exchanged, i.e., vertically adjacent blocks are accessed. Secondly, both 8×8 input blocks and the interpolation result are transposed.

The 2D interpolation of an 8×8 luma block requires access to the 15×15 area due to the extension by three and four positions in both dimensions. Therefore, the compensator must provide four adjacent reference 8×8 blocks, as illustrated in Fig. 7. MVDs shown in the figure allow the interpolation for fractional-pel positions located on the bottom-right side of the integer-pel search centre (positive fractions). The computation for other sides is possible by decrementing one or both MVD components. The horizontal interpolation is executed separately for upper and lower blocks for a selected horizontal fraction (*hf*). Two vertically adjacent 8×8 blocks are computed and used in the following vertical phase. As the wiring between input

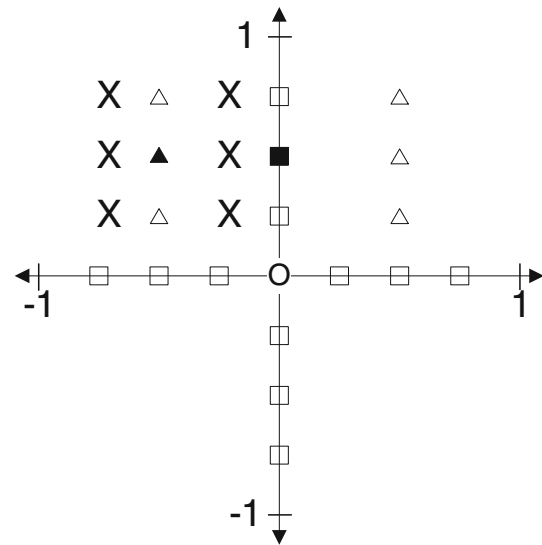


Fig. 8 Example of the fractional-pel search pattern. Squares, triangles, and crosses indicate the first, the second, and the third phase, respectively. Black elements indicate the best points found in corresponding phases

registers and the filter array is fixed to support the horizontal processing, the transposition is applied before (the feedback) and after (final result) the vertical interpolation.

The example of the fractional-pel search pattern is shown in Fig. 8. In the first phase, only 1D processing is used. The remaining phases involve the 2D interpolation. The timing diagram of 1D and 2D interpolations is depicted in Figs. 9 and 10, respectively. In the first case, the architecture computes 12 fractional-pel positions forming the cross pattern (squares in Fig. 8). Results appear at the interpolator output three clock cycles after the second 8×8 block at the input. The delay stems from three pipeline/register stages embedded in the datapath. Based on two 8×8 input blocks, three fractional-pel positions are computed in successive clock cycles. When blocks for the three positions are released, next two input blocks appear at the input. In this case, residuals are computed for blocks released from the interpolator output rather than from the integer-pel datapath of the compensator. Therefore, providing input blocks to the interpolator involves two additional clock cycles only at the beginning of the 1D interpolation. Although input blocks are determined for the interpolator, they can also be utilized to check corresponding integer-pel positions.

Figure 10 depicts the 2D interpolation for luma and chroma. The architecture performs the horizontal and vertical processing in two consecutive phases. In the first phase for luma, four adjacent 8×8 blocks included in the 16×16 area are loaded into the 15×8 input register in four clock cycles. When each of the two block pairs is written, the horizontal interpolation is started. 1D results appear at

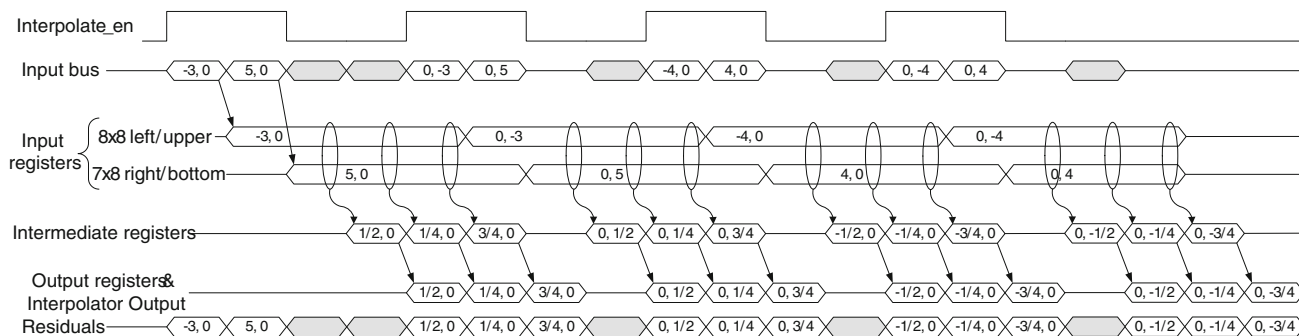


Fig. 9 Timing diagram of successive 1D interpolations

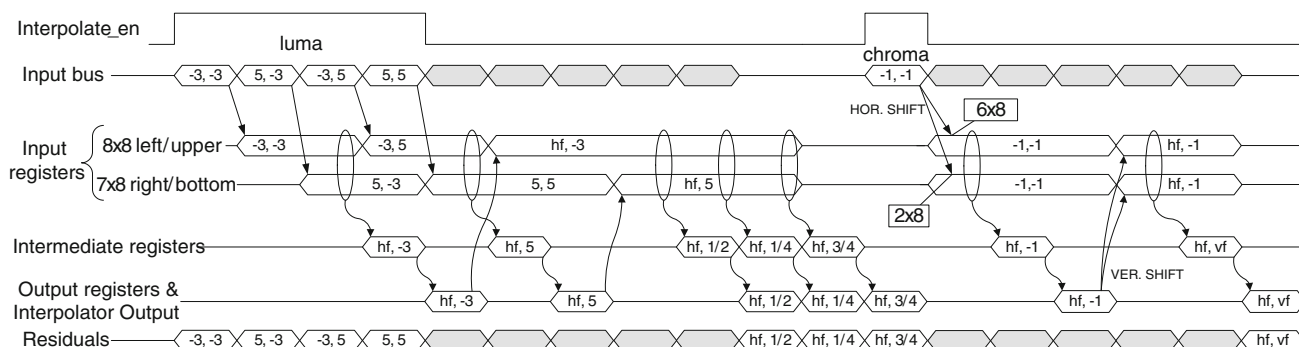


Fig. 10 Timing diagram of the 2D interpolation

the filter output with the two-cycle delay. They are fed back to input registers. The vertical interpolation starts when both results are written into input registers. 8×8 predictions appear at the interpolator output with a two-cycle delay. Three vertically neighboring fractional-pel positions can be computed in successive clock cycles.

The interpolation for chroma (4:2:0) differs from that for luma. Only one 8×8 is received at the beginning. It is sufficient to compute the 4×4 fractional-pel output. The input block is obtained for the MVD equal to $(-1, -1)$ and must be written to registers with a shift by two horizontal positions. This operation adjusts the input domain to active filter taps, as access to two left register columns is required only for luma. The analogous operation is accomplished while feeding the registers before the vertical processing. Only four columns of the filter array are used due to the smaller size of the output block. Moreover, the bottom row of filters is also not utilized. Therefore, the architecture employs only 4×7 filters supporting the chroma interpolation.

The 1D and 2D interpolation introduces the delay between receiving of input blocks and releasing of fractional-pel results. In the meantime, data appearing in the main path of the compensator bypass the interpolator. Such data are in grey in Figs. 9 and 10. Since the integer-pel path in the compensator does not communicate with the

interpolator, corresponding clock cycles can be utilized to check some integer-accuracy motion vectors for other prediction blocks. Such interleaved processing requires the two-thread motion vector generator.

5 Reconfigurable filter structure

Interpolation filters specified in H.265/HEVC have seven or eight taps as shown in Table 1. The hardware architecture can map each tap into a separate multiplier with results summed by three-level adder tree (see Fig. 11). If the design must work at high frequencies, the filter structure should be pipelined. Successive pipeline stages should be balanced in terms of critical path delays. The most obvious approach is to place registers after the multiplications and between particular tree stages. As signal delays in carry chains of cascaded arithmetic circuits overlap to a large extent, the number of stages can be limited and balanced with the stage having the longest path (e.g., multiplication).

FPGA devices usually have DSP units dedicated to filter implementations. Such units embed multipliers, adder trees, and internal pipeline registers allowing the operation at high frequencies. Such resources enable the straightforward implementation of the fractional sample

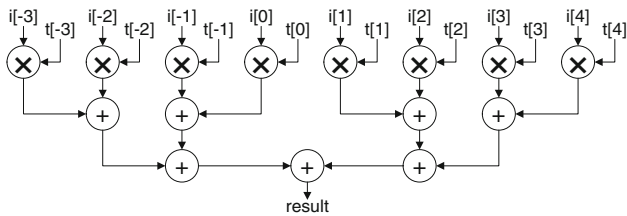


Fig. 11 Straightforward architecture of the 8-tap interpolation filter

interpolation. Apart from high speed, implementations based on DSP units utilize few general-purpose resources such as logic elements. However, the number of DSP units in FPGA devices is limited, and they can be utilized in other modules of the H.265/HEVC encoder, i.e., the forward/inverse transform and the (de)quantization. Therefore, the implementation based on regular logic elements can better fit FPGA resources.

In ASIC implementations, the incorporation of regular multipliers is inefficient when coefficients are constant or limited to a narrow set of values. A better approach is to design adder/subtractor trees with dataflow reconfigured by multiplexers. This approach can also be used in FPGA devices when there is insufficient availability of DSP units. The multiplexers can change the actual filter coefficients by shifting and switching between the intermediate branches of the tree. For example, it is possible to add samples corresponding to the same coefficient values at the first stage (the property of symmetrical filters). In general, there are many possible filter implementations based on the adder/subtractor tree. However, preferred solutions should minimize the amount of utilized resources. The design of particular filters exploits some well-known optimization techniques. Firstly, filter coefficients which are equal to a power of two do not require multiplication circuits as hardwired-shifting can yield the result. Secondly, multiplications are equivalent to additions/subtractions of the same sample up-shifted by different position numbers. Thirdly, an input sample or intermediate results can be directed to different adder/subtractor nodes to balance the tree. Fourthly, the reconfiguration of the filter coefficients can be realized by multiplexing. Fifthly, when two filters have the same coefficient values with the inverted order, it is possible to share resources by changing the assignment of the input samples with multiplexers.

Figure 12 depicts the proposed structure of the reconfigurable filter based on the adder/subtractor tree. The full-featured filter embeds 22 adders/subtractors at two pipeline stages. If only luma is supported, the number of adders/subtractors is decreased to 17. The dataflow is reconfigured by five types of multiplexers:

- NI multiplexers select between non-inverted and inverted order of input samples. This selection allows the same dataflow for filter pairs with the mirrored order of coefficients. In particular, the I input is active when processing element implements the 3/4 luma filter or the 5/8, 6/8, or 7/8 chroma filters. Otherwise, the N input is selected.
- OE multiplexers are utilized only for chroma to select between odd (O) and even (E) numerators of fractional-pel positions. The O input is selected when the circuit is configured as the 1/8, 3/8, 5/8, or 7/8 filters. Otherwise, the E input is selected.
- HQ multiplexers select between half- and quarter-pel configurations. The H input is selected when the circuit is configured as the half-pel luma filter or the 3/8, 4/8, or 5/8 chroma filters. Otherwise, the Q input is selected.
- LC multiplexers select between luma (L) and chroma (C) interpolation modes.
- The SAT multiplexer accomplishes the saturation of the final result to avoid overflow and underflow.

Interpolation filters in H.265/HEVC involve the increase of the output bit accuracy to avoid underflow and overflow. In extreme cases, the output requires additional seven bits compared to the input accuracy. Additionally, the sign bit must be taken into account. Therefore, 16 bits are needed to represent samples after the 1D interpolation before the rounding operation. In the second phase of the 2D interpolation on 16-bit signed data, the output accuracy is increased by seven bits to 23 bits. Consecutive stages of the adder/subtractor tree gradually increase the range to 23 bits at the 2D output. To keep the same output range for the 1D and 2D processing, eight-bit integer-pel samples are written to input registers before the interpolation with shifting up by six bit positions.

6 Implementation results

The architecture of the compensator and the interpolator is specified in VHDL and verified with the HM 13 reference model [2]. The synthesis is performed for FPGA and ASIC technologies using the Altera Quartus II software (ver. 12.0) and Synopsys Design Compiler (ver. 2012.06-SP5), respectively. Particularly, FPGA synthesis is performed for Aria II GX FPGA devices (speed grade 4), whereas TSMC 90 nm is selected as the ASIC technology. Implementation results are summarized in Table 2. As can be seen, the main contribution to the resource consumption is from the interpolator, which embeds 64 reconfigurable filter cores. The full-featured filter needs 488 ALUTs and 4524 gates for FPGA and ASIC technologies, respectively. If only luma is supported, the resources are decreased significantly

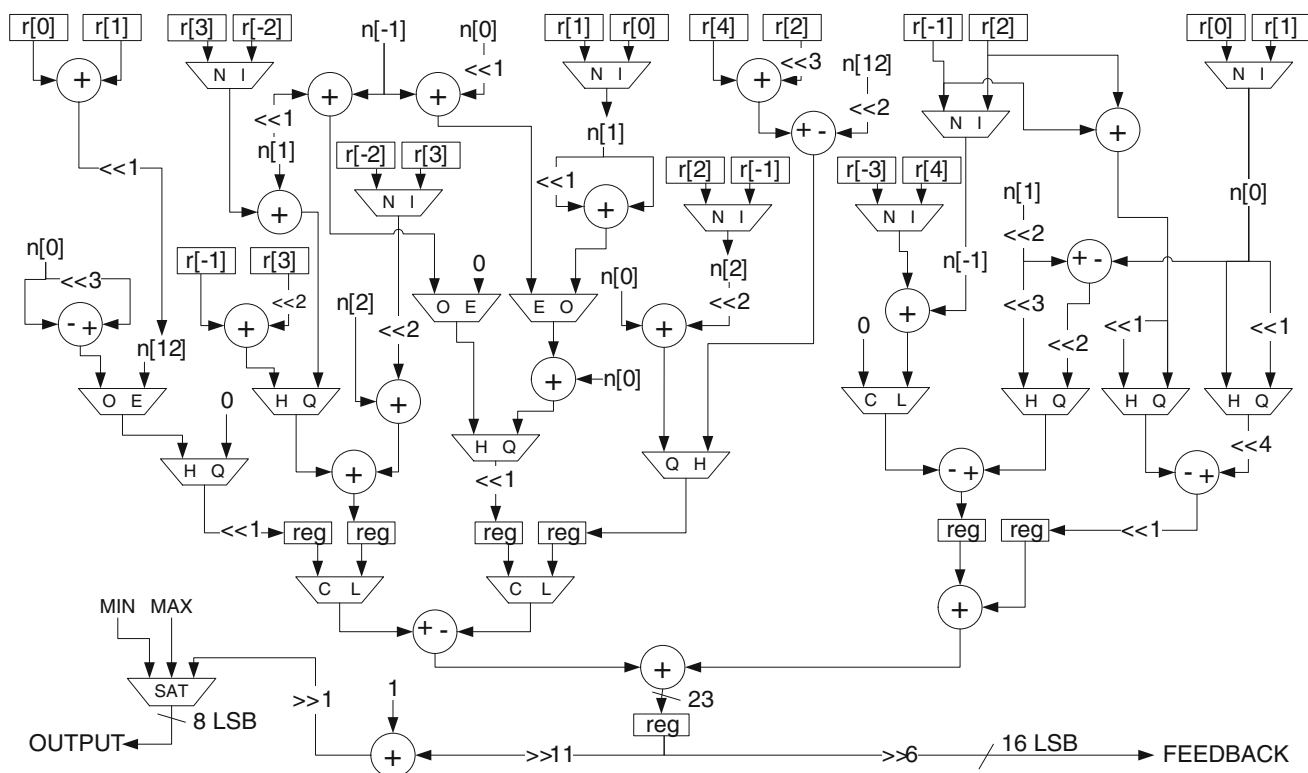


Fig. 12 Architecture of the reconfigurable filter. Indices of input registers ($r[x]$) are relative and depend on the position in the filter array

Table 2 Synthesis results

Technology	Units	Module	
		Compensator	Interpolator
Arria II GX	Logic (ALUT)	4 247	28 757
	Clock (MHz)	200	200
TSMC 90 nm	Logic (gate)	42 519	277 074
	Clock (MHz)	400	400
	Memory (kB)	64 + 12	–

to 325 ALUTs and 3452 gates. The joint resource consumption for the compensator and the interpolator is close to that for the whole motion estimation system since the motion vector generator is expected to utilize less logic [5]. For the ASIC technology, the design can operate at the frequency of 400 MHz. This performance enables the encoder to allocate about 400 and 100 clock cycles per each 8×8 block for 1080p@30fps and 2160p@30fps, respectively. One motion vector (integer-pel or fractional-pel) can be checked almost in each clock cycle. The compression efficiency of this approach will depend on the search algorithm accomplished by the motion vector generator. The estimated power consumption of the ASIC implementation is equal to 11.4 and 244.2 mW for the interpolator and the compensator, respectively. The high

power consumption of the compensator is caused by memories keeping reference pixels.

The compensator incorporates 64 two-port memory modules to store reference pixels. Each module has the size of 1kB. This capacity allows the search range of $(-64, 63) \times (-48, 48)$ for both luma and chroma. Wider ranges are possible at the cost of the increased memory size. The original pixels are stored in a separate memory of a capacity of 12 kB.

Byun et al. [10] presented the H.265/HEVC integer-pel full search architecture supporting all prediction unit sizes at the range of $(-32, 31) \times (-32, 31)$. The design consumes 3.56 M gates and 23 kB memories. The hardware cost of the motion estimation system described in this paper is much smaller even if the motion vector generator would be several times more complex than that developed for H.264/AVC [5]. Moreover, the search range is wider. Low-power integer-pel design was proposed by Sanchez et al. [11]. Its resource consumption is relatively low (50k gates and 82k bit memories). However, it supports only 16×16 blocks and a narrow search range, which does not exploit the compression potential of H.265/HEVC.

The proposed architecture for H.265/HEVC interpolator is compared with other designs in Tables 3 and 4. The tables include implementation results for our previous architecture [4] generating fractional-pel samples before

Table 3 Comparison with other FPGA architectures

Design	Afonso [12]	Pastuszak [4]	This work
Technology	Stratix III	Arria II GX	Arria II GX
Clock (MHz)	403	200	200
Resources (ALUT)	4077 + 16547	18 831	28 757
Parallelism (sample/clock)	27	64	64
1,000 × parallelism/resources	1.31	3.40	2.26
Features	Luma	Luma and chroma	Luma and chroma

Table 4 Comparison with other ASIC architectures

Design	Diniz [13]	Guo [14]	He [15]	Pastuszak [4]	This work
Technology (nm)	TSMC 150	SMIC 90	65	TSMC 90	TSMC 90
Clock (MHz)	312	250	188	400	400
Resources (gate)	30 209	32 496	1 183 000	224 094	277 074
Parallelism (sample/clock)	12	8	16×12	64	64
1,000 × parallelism/resources	0.3972	0.2462	0.1623	0.2856	0.2310
Throughput	2160p@30fps	2160p@60fps	4320p@30fps	1080p@30fps	2160p@30fps
Features	Luma and chroma	Luma	Luma	Luma and chroma	Luma and chroma

storing them into 64 on-chip memories. Although the proposed architecture needs more resources in the interpolator, it significantly reduces the size of the memories in the compensator, as described in Sect. 3. The maximal working frequency of the proposed architecture is the highest within the ASIC comparison. Three implementations [12–14] offer lower parallelism. As a consequence, declared throughputs are achieved when the size of the prediction unit is selected prior to the interpolation. One design supports 4320p@30fps video with the interpolation for three prediction unit sizes (64×64, 32×32, and 16×16). On the other hand, it consumes a large amount of resources, and its design efficiency is the lowest. Moreover, adapted simplifications involve quality losses. Only one of the architectures [13] has a significantly better design efficiency (parallelism/resources). However, it requires additional memories and the control logic. Most referred designs support only the luma interpolation [12, 14, 15]. One ASIC design [15] implements filters specified in Working Draft 3. The FPGA implementation proposed by Afonso et al. [12] achieves a high frequency due to deep pipelining and the better device. The proposed architecture can also be modified to operate at higher frequencies by the insertion of registers. This modification would not increase the logic resources since at least one flip-flop is embedded in each ALUT.

The proposed architecture can support many search algorithms owing to the flexibility in the order of checked motion vectors. Therefore, the compression efficiency

depends on the selected algorithm and the number of clock cycles assigned to particular coding tree units. To estimate the efficiency achievable with the architecture, the reference model HM13 is used for the low-delay configuration defined in Common Test Conditions. Five 1080p sequences [19] are evaluated with 50 frames and TZ Search. The software is modified to model a limited number of checked integer-pel motion vectors. The limitation is set to 50 or 100 for each 8×8 prediction block. The first corresponds to the half of clock cycles allocated in the case of the 2160p video coding. The remaining cycles can be utilized for the subpixel estimation, chroma, and additional estimation for larger prediction units. For larger prediction blocks, only common motion vectors checked for four smaller prediction blocks are taken into account. The reference scenario has no limitations on the number of checked motion vectors. Evaluation results are summarized in Table 5 in terms of Bjontegaard Delta Rate [20]. Although losses in the compression efficiency are not large, there is a space to develop more robust search algorithm. In particular, the correlation between smaller prediction blocks should be utilized to obtain more predictions for larger blocks.

The real-time H.265/HEVC encoding of 1080p videos can also be obtained using the parallel processing on CPU and GPU platforms [17, 18]. On the other hand, the compression efficiency of such implementations is significantly lower compared to the HM reference software (0.7–0.92 dB).

Table 5 Evaluation results for BD-rate for different numbers of checked integer-pel MVs

Sequence	BD-rate	
	50 MVs	100 MVs
Blue sky	1.36	0.46
Ducks take off	0.32	0.04
Station2	1.65	0.26
Rush hour	4.02	0.38
Tractor	2.43	0.74
Average	1.96	0.38

7 Conclusion

The architecture of the compensator and the interpolator is developed for the H.265/HEVC adaptive motion estimation. The design performs interpolation on reference pixels read from on-chip memories. This allows much wider search ranges and reduced memory sizes. The design achieves the high utilization of hardware resources, thanks to interleaving of integer- and fractional-pel computations. The design can check about 100 motion vectors for each 8×8 block when encoding 2160p@30fps video at the 400 MHz. Future research will concentrate on the development of an efficient search strategy able to support different prediction block sizes.

Open Access This article is distributed under the terms of the Creative Commons Attribution License which permits any use, distribution, and reproduction in any medium, provided the original author(s) and the source are credited.

References

1. ITU-T Recommendation H.265 and ISO/IEC 23008-2 MPEG-H Part 2, High efficiency video coding (HEVC), (2013)
2. HEVC software repository: HM-10.0 reference model. Available: https://hevc.hhi.fraunhofer.de/svn/svn_HEVCSoftware/branches/HM-10.0-dev/
3. ITU-T Rec. H.264 and ISO/IEC 14496-10 MPEG-4 Part 10, Advanced video coding (AVC), (2005)
4. Pastuszak, G., Trochimiuk, M.: Architecture design and efficiency evaluation for the high-throughput interpolation in the hevc encoder. *Euromicro Conf. Digit. Syst. Des. (DSD)* **2013**, 423–428 (2013)
5. Pastuszak, G., Jakubowski, M.: Adaptive computationally-scalable motion estimation for the hardware H.264/AVC encoder. *IEEE Trans. Circuits Syst. Video Technol.* **23**(5), 802–812 (2013)
6. Chen, T.-C., Chien, S.-Y., Huang, Y.-W., Tsai, C.-H., Chen, C.-Y., Chen, T.-W., Chen, L.-G.: Analysis and architecture design of an HDTV720p 30 frames/s H.264/AVC encoder. *IEEE Trans. Circuits Syst. Video Technol.* **16**(6), 673–688 (2006)
7. Liu, Z., Song, Y., Shao, M., Li, S., Li, L., Ishiwata, S., Nakagawa, M., Goto, S., Ikenaga, T.: HDTV1080p H.264/AVC encoder chip design and performance analysis. *IEEE J. Solid-State Circuits* **44**(2), 594–608 (2009)
8. Yang, C., Goto, S., Ikenaga, T.: High performance VLSI architecture of fractional motion estimation in H.264 for HDTV, In: *IEEE International Symposium on Circuits and Systems (ISCAS 2006)*, pp. 21–24 (2006)
9. Oktem, S., Hamzaoglu, I.: An efficient hardware Architecture for quarter-pixel accurate H.264 motion estimation, In: *10th Euro-micro Conference on Digital System Design*, pp. 1142–1143 (2007)
10. Byun, J., Jung, Y., Kim, J.: Design of integer motion estimator of HEVC for asymmetric motion-partitioning mode and 4 K-UHD. *Electron. Lett.* **49**(18), 1142–1143 (2013)
11. Sanchez, G., Porto, M., Agostini, L.: A hardware friendly motion estimation algorithm for the emergent HEVC standard and its low power hardware design, In: *IEEE International Conference on Image Processing*, pp. 1991–1994 (2013)
12. Afonso, V., Maich, H., Agostini, L., Franco, D.: Low cost and high throughput FME interpolation for the HEVC emerging video coding standard, In: *IEEE Fourth Latin American Symposium on Circuits and Systems (LASCAS)* (2013)
13. Diniz, C. M., Shafique, M., Bampi, S., Henkel, J.: High-throughput interpolation hardware architecture with coarse-grained reconfigurable datapaths for HEVC, In: *IEEE International Conference on Image Processing*, pp. 2091–2095 (2013)
14. Guo, Z., Zhou, D., Goto, S.: An optimized MC interpolation architecture for HEVC, In: *IEEE International Conference on Acoustics, Speech and Signal Processing*, pp. 1117–1120 (2012)
15. He, G., Zhou, D., Chen, Z., Zhang, T., Goto, S.: A 995Mpixels/s 0.2nJ/pixel fractional motion estimation architecture in HEVC for Ultra-HD, In: *IEEE Asian Solid-State Circuits Conference*, pp. 301–304 (2013)
16. Jakubowski, M., Pastuszak, G.: An adaptive computation-aware algorithm for multi-frame variable block-size motion estimation in H.264/AVC, In: *International Conference on Signal Processing and Multimedia Applications (SIGMAP '09)*, pp. 122–125 (2009)
17. Wang, X., Song, L., Chen, M., Yang, J.: Paralleling variable block size motion estimation of HEVC on CPU plus GPU platform, In: *IEEE International Conference on Multimedia and Expo Workshops (ICMEW)* (2013)
18. Wang, X., Song, L., Chen, M., Yang, J.: Paralleling variable block size motion estimation of HEVC on multi-core CPU plus GPU platform, In: *IEEE International Conference on Image Processing (ICIP)*, pp. 1836–1839 (2013)
19. Xiph.org: Test media, available on-line at <http://media.xiph.org/video/derf/> (2011)
20. Bjontegaard, G.: Calculation of Average PSNR differences Between RD-Curves, In: *ITU-T VCEG-M33, VCEG 13th Meeting* (2001)

Grzegorz Pastuszak received the M.S. degree in microelectronics in 2001 and the Ph.D degree in multimedia technology in 2006, both from Warsaw University of Technologies, Warsaw, Poland. From 2001 to 2002, he was an ASIC designer in FFC, Tokyo, and Fujitsu Devices, Yokohama. Currently, he is with Institute of Radioelectronics Warsaw University of Technology. His areas of interest include VLSI architectures and algorithms, image/video/audio processing and compression, high-performance digital ICs.

Maciej Trochimiuk received the M.S. degree in radio-communication and multimedia technology from the Warsaw University of Technology in 2012, where he is currently pursuing the Ph.D degree. His current research interests include video and image processing and compression technologies, computer vision and efficient hardware implementations of the related algorithms for the embedded systems.