Algorithm and architecture design of the motion estimation for the H.265/HEVC 4K-UHD encoder

Grzegorz Pastuszak · Maciej Trochimiuk

Received: 20 January 2015 / Accepted: 22 June 2015 / Published online: 2 July 2015
© The Author(s) 2015. This article is published with open access at Springerlink.com

Abstract This paper presents the algorithm and the architecture of the high-throughput motion estimation system for the H.265/HEVC encoder. The design allows the processing of 2160p@30fps videos at the clock frequency of 400 MHz. The architecture embeds two parallel processing paths for the integer-pel and the fractional-pel motion estimation. The paths share the same memories. Access conflicts are avoided by the use of dual-port modules and register buffers for reused samples. In each clock cycle, the integer-pel and the fractional-pel path can evaluate one and four motion vectors for an 8×8 luma block, respectively. A separate interpolator for chroma additionally increases the throughput. The integer-pel path supports test zone search for 8×8 prediction blocks. The motion estimation for larger blocks is performed by the utilization of results of the 8×8 search. The search for rectangular PUs is performed only at the fractional-pel level and reuses partial costs computed for square PUs. As a consequence, a significant amount of computation is saved. 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 implemented algorithm is verified in the HM16 software. If 2160p@30fps videos are encoded with the low-delay configuration, BD-PSNR and BD-rate are equal to -0.026 dB and 1.64 %, respectively.

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

1 Introduction

Research and standardization efforts in video coding led to the specification of the H.265/HEVC standard [1, 2] in 2013. At the same quality of the reconstructed video, the standard provides an improvement in compression efficiency of about 35–50 % compared to its predecessor H.264/AVC [3]. However, the better compression efficiency is achieved at the price of increased computational complexity. Although the general structure of the encoder and the decoder remains the same, there are many changes in the algorithm. Instead of 16×16 -pixel macroblocks, the new standard applies coding tree units (CTUs), which can be up to 64×64 pixels in size. Each CTU can be recursively split into square coding units (CUs) with the minimal size of 8×8 pixels. Each $2N \times 2N$ CU can be partitioned into predictions units (PUs). N can be equal to 4, 8, 16, or 32. There are eight allowable partition shapes: two square shapes ($2N \times 2N$ and $N \times N$), two symmetric rectangular shapes ($N \times 2N$ and $2N \times N$), and four asymmetric rectangular shapes ($2N \times 3N/2$, $2N \times N/2$, $3N/2 \times 2N$, and $N/2 \times 2N$). Each inter PU has a separate motion vector (MV). Similar to H.264/AVC, the H.265/HEVC allows quarter pixel accuracy MVs. There are new interpolation schemes to compute fractional-pel positions. In particular, 7-tap and 8-tap filters are used for the luma interpolation of half-pel and quarter-pel positions, respectively. Chroma samples are computed using 4-tap filters. Although design and implementation of digital filters is a thoroughly explored

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

issue, high-throughput video encoders require some effort to obtain efficient hardware solutions.

With the exception of our previous designs [4, 5], architectures for the motion estimation (ME) consist of two parts assigned to the integer-pel and fractional-pel search [6–15]. This approach requires 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 only supports 16×16 blocks, limiting the compression efficiency.

Some high-throughput interpolators have been proposed in 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 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 designed for H.265/HEVC have been described in literature [12–15]. They achieve throughputs suitable for video resolutions from 1080p to 4320p. All the designs neglect the interpolation for merge modes. Three designs [12–14] are based on the assumption that the size of prediction units is selected at the integer-pel ME. 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); however, it consumes a large amount of hardware resources. Generally, a compression-efficient and high-throughput implementation requires more hardware resources and increases power consumption. Therefore, there is a need for solutions in which these parameters are optimized.

This study presents the high-throughput ME architecture dedicated to the H.265/HEVC encoder. Similar to the previous works [4, 5], the architecture can check one integer-pel motion vector for an 8×8 block in each clock cycle. As an arbitral order of motion vectors is allowed, the architecture supports the test zone search (TZS) algorithm used in the HM software. A significant amount of computation is saved for 16×16 and 32×32 prediction blocks by the exploitation of results of the 8×8 -block search. In the case of rectangular PUs, only MVs checked for the fractional-pel ME of $2N \times 2N$ PUs are evaluated, which additionally reduces the complexity at small quality losses.

The present study has four novel contributions at the architecture level. Firstly, the use of dual-port memories and register buffers for reused data allows shared and conflict-free access from two processing paths corresponding to the integer-pel and the fractional-pel search. Secondly, the extension of the interpolation to 9×9 blocks allows the evaluation of four 8×8 fractional-pel blocks at a small increase in the resource cost. Thirdly, the architecture enables the two-dimensional continuous interpolation of 9×9 blocks with reconfigurable and dedicated filter cores. Fourthly, the separate chroma interpolator additionally increases the throughput and design flexibility.

The rest of the paper are 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 H.265/HEVC motion estimation system. The applied scheduling is described in Sect. 4. Section 5 presents the motion estimation algorithm executed by the proposed architecture. Section 6 provides implementation results. Finally, the paper is concluded in Sect. 7.

2 Design for adaptive motion estimation

The adaptive computationally scalable motion estimation algorithm allows video encoders to achieve close to optimal efficiencies 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 for the H.265/HEVC encoder [4] is depicted in Fig. 1. The

Fig. 1 Architecture of the adaptive motion estimation system for the H.265/HEVC encoder [4]

architecture embeds the interpolator and the motion vector generator. The remaining elements build the compensator. The design allows the adaptive search for both integer-pel and fractional-pel positions. The fractional-pel search is performed around some MVs selected at the integer-pel stage. However, the integer-pel estimation is interrupted whenever data are submitted or released from the interpolator because of the sharing of the same memories and the residual computation. As a consequence, less clock cycles can be utilized at both stages of the motion estimation. Higher throughputs are achieved when interpolated pixels are stored in the memories. On the other hand, the interpolation before writing into the memories involves a large memory cost [5]. This disadvantage is of particular importance in the case of the H.265/HEVC encoder, in which the processing of 16×16 -pixel macroblocks is replaced by CTUs with sizes up to 64×64 pixels.

In order to process blocks of 8×8 samples, the interpolator embeds 64 reconfigurable filters [4]. 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 number of filters corresponds to the size of blocks processed in the main path. Although the interpolation parallelism is high, the throughput is limited by the reading of 8×8 blocks at the input. In particular, two and four 8×8 blocks must be read to obtain the 1D and 2D interpolation of three fractional-pel positions for one block, respectively. More clock cycles are utilized when the interpolation is performed in two dimensions. If 100 cycles are available for each 8×8 -pixel block (2160p@30fps), the interpolation around two integer-pel MVs can be performed for luma. Particularly, one 1D luma interpolation with the cross pattern takes 16 cycles, whereas 2D interpolation for nine positions takes 27 cycles. Two corresponding chroma blocks are interpolated in 10 cycles for one position. Totally, 96 out of 100 cycles are utilized for the luma and chroma interpolation. 54 cycles are available for the integer-pel search interleaved with memory reads for the fractional-pel estimation. Although the throughput is significantly improved compared to other designs [12–14], it is still insufficient to evaluate the greater number of PU sizes. Additional interpolations are indispensable to support merge modes.

The compensator embeds 64 memory modules to store 256×160 reference pixels (luma and chroma) [4]. Data access is based on ‘moving windows,’ as shown in Fig. 2. The memory space is divided into four subspaces each of which are 64×160 pixels in size. 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 CTU is in progress. When the motion estimation for the next CTU is started, the windows are moved right by 64 pixels (the width of one subspace). The

Fig. 2 The assignment of the write and read ports to memory subspaces in the ‘moving windows’ scheme [4]

subspace assigned to the write port is filled with reference pixels which become 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 $(-64, 63) \times (-48, 47)$. Each of 64 memories keeps every eighth integer-pel sample both in the horizontal and vertical dimension. Generally, samples for a given MV can belong to four adjacent 8×8 blocks (see Fig. 3a, b). Since MVs vary in the whole search range, the 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 spatial consistency (see Fig. 3c, d).

3 New architecture

In the architecture described in the previous section [4], the integer-pel and the fractional-pel search share the same processing path with the interleaved processing. As a consequence, the number of clock cycles assigned to the integer-pel estimation is decreased almost by half, which has a negative impact on the compression efficiency. The main bottleneck is introduced by the memory read port able to provide one 8×8 block in each clock cycle. In order to

Fig. 3 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

resolve the problem, the new architecture incorporates dual-port memory modules instead of two-port ones. The main advantage of dual ports is that they can operate in either the read or the write mode. In the architecture, the first port is assigned to the integer-pel path, whereas the second is used as the input to the interpolator. The interpolator incorporates the register buffer at the inputs stage to reuse samples from the second path. Since the interpolator does not read data in each clock cycle, some cycles can still be utilized to write the reference pixels for the following CTUs. The same approach is applied to the memory storing original samples.

The new architecture of the motion estimation system is depicted in Fig. 4. The architecture embeds two processing paths (integer-pel and fractional-pel) which read data from original sample and reference sample memories through separate ports. The integer-pel path operates as the loop where the search process can be adapted according to the result obtained for previous MVs. Particularly, the MV generator supports the TZ search algorithm for 8×8 blocks. The module embeds two finite state machines which determine MVs for the integer-pel and the fractional-pel path. The fractional-pel path includes an interpolator for luma samples. Although the path cannot be fed

in each clock cycle due to the memory writing, the continuous processing is allowed by the buffering and the reusing of reference data in the interpolator.

The luma interpolator can release 9×9 blocks in successive clock cycles. Each one includes four overlapping 8×8 blocks, as shown in Fig. 5. The common area for all the blocks has 7×7 samples. The four 8×8 blocks correspond to fractional accuracy MVs whose horizontal and/or vertical components differ by 1. As a consequence, it is possible to simultaneously check four fractional-pel MVs around an integer-pel MV. The computation of 9×9 blocks increases the parallelism of the fractional-pel path four times at a relatively small increase of the hardware cost. With the assumption that the interpolator embeds separate horizontal and vertical filters, their number increases from 64 to 72 and 81, respectively. Four 8×8 candidate blocks distinguished within the 9×9 block are subtracted from original samples. Four results are used to compute sum of absolute differences (SAD). SADs increased by corresponding MV bit costs are compared one another. Based on the comparisons, the rank list is built. The rank is used to preselect the group of four candidate MVs worth analyzing with the rate-distortion criterion. Actually, the rank list is used for square PUs. For each

Fig. 4 Architecture of the motion estimation system with two processing paths corresponding to the interpolation and the integer-pel search

Fig. 5 Locations of input and output blocks for the 2D interpolation. 9×9 block released from the luma interpolator includes four overlapping 8×8 blocks

rectangular partition, only one best candidate MV is selected.

9×9 blocks released from the interpolator must be written to a buffer to wait for the end of the preselection process. Some predictions corresponding to preselected MV candidates should be kept until they are forwarded to the reconstruction loop and the rate-distortion optimization. The buffer is outside the ME system and will be used to integrate with other encoder modules [17].

The new architecture of the luma interpolator is depicted in Fig. 6. The architecture incorporates separate stages for horizontal and vertical interpolation. Four successive 8×8 input blocks are written into the input ring buffer composed of four 16×4 register groups. The ring can reuse the four written blocks. When the blocks are reused, reference memories can be written with new data. One group of registers provides samples to 36 horizontal filters. The filters interpolate one 9×4 block in each clock cycle. Each horizontal filter can be reconfigured to support quarter-pel and half-pel interpolations. Additionally, the filters can be configured as the bypass path when the horizontal MV component is integer. Actually, filter cores do not support the interpolation for $3/4$ positions. This interpolation is obtained by the horizontal transposition of samples at the input and the output. This method is also applied in the chroma interpolator.

The horizontal interpolator computes the 9×16 sample array in four clock cycles and then forwards it to the vertical stage. The vertical interpolator can be implemented as the 9×9 array of reconfigurable filters which determine a 9×9 block for one fractional-pel MV in each clock cycle. However, the hardware cost of 81 reconfigurable filters is significant. To save resources, the vertical stage incorporates 54 dedicated and nine reconfigurable filters. Each of three fractional-pel interpolations ($1/4$, $1/2$, and $3/4$) is performed with 18 dedicated filters. Separate bypass paths transfer 18 samples not interpolated vertically. Each bypass path includes the rounding adder and the range limiter. Nine reconfigurable filters perform all interpolations for the

Fig. 6 Architecture of the luma interpolator. *FRB* first ring buffer, *SRB* second ring buffer, *HR* horizontal register

most right column in three cycles. The fourth cycle is utilized to transfer nine samples through the bypass path. The remaining eight columns are horizontally rotated between registers feeding dedicated filters and the bypass path. In particular, the register content is moved by two columns in each clock cycle. Each register column is assigned to one of the three groups of dedicated filters or to the bypass path. As a consequence, the 9×9 blocks released from the vertical stage consists of samples interpolated for four fractional-pel MVs. Thus, SADs must be accumulated in parallel for 16 fractional-pel MVs in four clock cycles. One multiplexer at the interpolator output is used to restore locations of four 2×9 sample groups. Another multiplexer vertically transposes positions in the most right column if the result for the $3/4$ interpolation is released.

Inter-chroma predictions are obtained with MVs inferred from luma for a given CU size. Therefore, the required throughput of the chroma interpolation is much smaller than that of luma. The chroma interpolator embeds 12 reconfigurable filter cores assigned to two processing stages, as shown in Fig. 7. Eight cores interpolate chroma samples horizontally, whereas the vertical stage incorporates the remaining four cores. The chroma interpolator can provide four samples in each clock cycle. The inter-chroma predictor embedding the interpolator is designed as a separate processing path with dedicated 16 memories. The chroma predictor applies similar dataflow as in the case of the luma path. However, the prediction is determined only for one MV.

Figures 8, 9, 10, 11 show architectures of particular filter cores incorporated to the design. All the filters are

implemented with two pipeline stages not shown in the figures for the clarity. Particularly, one stage includes one or two layers of adders/subtractors. The architecture of one reconfigurable filter used at the horizontal stage of the luma interpolator is depicted in Fig. 8. The filter supports half-pel (H) and quarter-pel (Q) interpolation. The reconfiguration between the two types of the interpolation is performed with seven multiplexers. Additional three multiplexers allow the transfer of one sample when the horizontal interpolation is not used. In this case, inputs indicated as I (integer) in Fig. 8 are selected. The half of 10 two-input multiplexers have one input fixed to zero. As a consequence, the multiplexers are reduced to AND gates. Similar reductions are achieved for five multiplexers in the chroma filter.

The design of reconfigurable filters is well suited to FPGA devices since multiplexers are embedded in the same logic cell as the following adder/subtractor. The luma and chroma filter cores embed 12 and 10 adders/subtractors, respectively. The previous architecture required 22 adders/subtractors for the filter supporting both luma and chroma and 17 for luma. Therefore, the significant reduction of resources is achieved when the filter is limited to the luma processing.

Figures 10 and 11 depict architectures of dedicated filters used at the vertical stage for the half-pel and quarter-pel interpolation, respectively. The half-pel filter embeds 10 adders/subtractors whereas the quarter-pel filter consumes one more. Dedicated filters embed the rounding adder in the tree. The output multiplexer accomplishes the clipping (CLIP) of the final result to avoid overflow and underflow.

Fig. 7 Architecture of the chroma inter predictor **Fig. 8** Architecture of the reconfigurable luma filter **Fig. 10** Architecture of the dedicated half-pel luma filter **Fig. 9** Architecture of the reconfigurable chroma filter

4 Scheduling

The ME system is pipelined based on 32×32 units, as shown in Fig. 12. Four parts of 64×64 PUs are processed only for the merge mode in relevant time intervals. To

Fig. 11 Architecture of the dedicated quarter-pel luma filter

support 2160p@30fps video at 400 MHz, the number of clock cycles assigned to one unit is 1600. It means that 100 cycles are assigned to one 8×8 input block at each stage. The integer-pel ME allocates most cycles to the 8×8 search. The best MVs found for 8×8 blocks are used for the 16×16 search, in particular, four MVs are evaluated for one 16×16 block. Similarly, the best four MVs found at the 16×16 search are evaluated for one 32×32 block. Since eight MVs are assigned to larger PUs, the 8×8

Fig. 12 Scheduling of the motion estimation

search can utilize 92 MVs. For lower resolutions, more MVs are checked.

The fractional-pel ME needs 16 clock cycles to evaluate 64 MVs around one integer-pel MV. Thus, the search can be performed around six MVs for a given 8×8 block. However, some cycles are required to interpolate MVs identified for the merge mode, in particular, four cycles are utilized to obtain the 8×8 interpolation for one MV. If the merge MV falls in the range of the regular fractional ME, no additional cycles are required. It is assumed that 48 cycles are allocated to regular fractional ME around three integer-pel MVs (8×8 , 16×16 , and 32×32 PUs). The remaining 52 cycles are utilized to process 13 merge mode candidates determined for different CU divisions. The regular fractional-pel ME for a given PU is skipped if its range matches that for a larger PU. Saved cycles are utilized to evaluate more merge mode MVs. Since the availability of most of merge MVs depends on the mode decision for preceding CUs/PUs, merge mode candidates are evaluated at the same stage as the reconstruction loop and the CU/PU mode decision.

Interpolation filters specified in H.265/HEVC refer up to eight luma samples located in row/column at neighboring pixel positions. Therefore, the 2D interpolation of one sample requires access to the 8×8 reference block. If four blocks are accessed, the output can be extended to the 9×9 block. Provided that 8×8 blocks appear at the interpolator input, four cycles are taken to load the input registers. The location of the blocks can be identified by specific MVs, as shown in Fig. 5. For convenience, the following description will refer to motion vector differences (MVDs) relative to the integer-pel position around which the fractional-pel search is executed. If two horizontally adjacent 8×8 blocks are obtained for MVDs equal to $(-4, 0)$ and $(4, 0)$, the interpolator can compute MVDs equal to $(1/4, 0)$, $(1/2, 0)$, $(3/4, 0)$, $(-1/4, 0)$, $(-1/2, 0)$, and $(-3/4, 0)$. The same rule applies to the vertical processing. Four reference blocks required for the 2D interpolation have the following MVD: $(-4, -4)$, $(4, -4)$, $(-4, 4)$, and $(4, 4)$.

The luma interpolator can provide one 9×9 block in each clock cycle. As discussed in Sect. 3, the 9×9 block includes four overlapping 8×8 blocks. Thus, blocks for all 64 fractional-pel MVD around an integer-pel MV can

be released in 16 successive cycles. The interpolation process is divided into four phases. In each phase, the interpolator computes the block for MVDs having the same horizontal fraction. Figure 13 shows the pattern used to generate fractional-pel samples around the integer-pel MV located in the middle. The pattern is regular and independent on the cost obtained for particular MVDs. Moreover, the pattern extends the fractional-pel search to MVDs whose horizontal or vertical component is equal to -1 . Since the search is full, the compression efficiency is improved by 0.03 dB, on average. In the case of the merge mode, the design must interpolate samples with one of four phases used for the full search. Although the 8×8 block should be interpolated only for one MVD, one phase provides 16 MVDs. This is utilized to evaluate more MVDs and merge candidates (if they fall in the range).

The timing diagram of 2D luma interpolation is depicted in Fig. 14. To perform the interpolation around one integer-pel MV, four reference blocks are read from the input. The figure shows MVD corresponding to the blocks. In the timing diagram, there are some clock cycles when reference blocks are not read from memories. Such periods are utilized to write new data.

Fig. 13 Fractional-pel search pattern. *Squares, triangles, crosses, and diamonds* indicate the first, the second, the third, and the fourth phase, respectively

Fig. 14 Timing diagram of the 2D luma interpolation

To perform the luma interpolation, four 8×8 reference blocks are taken from the input and written to the first ring buffer. The buffer consists of four register groups (FRB[0]–FRB[3]), each of which keeps four 16 sample rows. In each clock cycle, the rows are vertically rotated between register groups. Row indices are indicated in Fig. 14. Each reference block is simultaneously written to two register groups. Since each row is composed of samples taken from two reference blocks, two groups are half-filled with new samples in one cycle. Due to the rotation, the first/third block is written to FRB[0] and FRB[1], whereas the second/fourth block is written to FRB[1] and FRB[2]. If the 3/4 interpolation is performed, samples written to FRB[3] registers are horizontally transposed. The FRB[3] registers feed horizontal filters. The filtering result is obtained with the delay of two clock cycles. Horizontally interpolated samples corresponding to four rows are written to horizontal registers (HR) in each clock cycle. Every fourth clock cycle, 12 rows kept in HR and four rows available at filter outputs are forwarded to the second ring buffer (SRB). The buffer feeds 63 vertical filters and 18 bypass paths. The SRB is composed of nine columns. Eight of them are horizontally rotated by two positions in each clock cycle. Each two of six columns are assigned to a group of 18 dedicated filters supporting one particular type of the interpolation (either 1/2, 1/4, or 3/4). Two columns are assigned to 18 bypass paths. Similar to the horizontal stage, the filtering result is obtained with the delay of two clock cycles. The rotation in the second ring buffer allows the processing of eight

columns with each filter type. On the other hand, multiplexers are required at outputs to restore appropriate locations of columns in the 9×9 block. One of nine columns is not rotated, and it feeds nine reconfigurable filters. The filters are reconfigured in each clock cycle to support one particular type of the interpolation. For the 3/4 interpolation, samples kept in SRB[4] are vertically transposed.

5 Search strategy

The proposed ME architecture can check an 8×8 prediction for one integer-pel MV and four fractional-pel MVs in each clock cycle. In practice, the number of evaluated MVs is limited and depends on the clock frequency and the video resolution. If the motion estimation operates at the frequency of 400 MHz and processes 2160p@30fps videos, the number of integer-pel MVs per each 8×8 block in the original image is about 100. This number should be allocated to all evaluated PUs corresponding to the block. Taking into account wider search ranges required for the 2160p@30fps resolution, numbers of MVs allocated to particular PUs can be too small to achieve a high compression efficiency. In the case of the fractional-pel ME, the available number of clock cycles can also limit the efficiency. Other limitations stem from the encoder dataflow, which introduces the delay between the ME and the final mode decision (based on the rate-distortion optimization). The delay causes some MV predictions to be

unknown at the ME. Thus, costs of evaluated MVs cannot be estimated reliably. Moreover, the determination of predictions for merge modes must follow the mode decision for preceding blocks.

Taking into account the limitations described above, the proposed search strategy introduces the following simplifications to the motion estimation algorithm applied in the HM software:

- The search range is set to $(-64, 63) \times (-64, 63)$.
- Test zone search is performed only for 8×8 PUs. It is interrupted when the number of checked MVs achieves the limit specified for a given resolution. The limit corresponds to the number of clock cycles assigned in the hardware architecture (e.g., 92 for 2160p@30fps). If some 8×8 PUs within the 32×32 unit do not utilize all allowable cycles, the remaining cycles are added to continue the interrupted search. This reallocation makes losses in the compression efficiency negligible.
- The integer-pel motion estimation for 16×16 PUs is performed by utilizing results from the 8×8 search. Four MV candidates are taken from MVs found for 8×8 blocks included in a given PU.
- The integer-pel motion estimation for 32×32 PUs is performed by utilizing results from the 16×16 search. MV candidates are determined according to the rule applied in the 16×16 search.
- Rectangular PUs are evaluated within the range of the fractional-pel estimation corresponding $2N \times 2N$ PUs. Although this simplification significantly reduces the ME complexity, it has a small impact on the average compression efficiency (0.3 %).
- MV costs are estimated based on results of the 8×8 search if a neighbor belongs to the same CTU. In this case, MV differences are computed with the assumption that neighbors are 8×8 blocks. In the remaining cases, actual MV predictors are taken from adjacent CTUs.
- Only merge mode candidates are evaluated for 64×64 PUs and their rectangular partitions. The exclusion of the 64×64 search decreases the compression efficiency by 0.8 % (−0.02 dB), on average.
- At least three merge mode candidates are evaluated for each PU if the video resolution is 2160p@30fps. More candidates can be processed if any of the three following conditions are true: First, merge MVs fall in the range of the fractional-pel search for the same or a larger PU. Second, fractional-pel search for a given PU matches that for a larger PU. Third, the resolution is lower than 2160p@30fps. The conditions stem from the scheduling and allow a better utilization of available clock cycles. In particular, more merge modes are

evaluated to avoid the redundant processing and/or no-operation cycles.

- The final MV is not selected with sum of absolute transformed differences (SATD) used in the HM software. Instead, candidate MVs are selected based on SAD at the fractional-pel stage. Four candidates are selected for square PUs. The remaining (rectangular) PUs have one candidate MV. It is assumed that corresponding predictions are used in the mode selection based on the rate-distortion analysis. This approach decreases the compression efficiency by 0.3 % compared to the use of SATD.

The reuse of results of the 8×8 search saves a significant amount of computations. Particularly, eight integer-pel MVs are evaluated for larger PUs including a given 8×8 block. Moreover, MVs for the larger PUs are reused for smaller ones.

To estimate the efficiency achievable with the architecture, the reference model HM16 is used with the low-delay configuration defined in Common Test Conditions [18] and one reference frame. The software is modified according to the simplifications described above. Sequences assigned to different video classes are coded. Apart from classes A–E specified in Common Test Conditions, a separate group of 2160p (4K) sequences is also evaluated. The sequences are taken from two video repositories [19, 20]. The first [19] includes six sequences: Bosphorus, Jockey, Honey Bee, Shake and Dry, Ready Steady Go, and Yacht Ride. The second group [20] includes Crowd Run, Ducks Take Off, In To Tree, and Park Joy. Sequences in the first and the second group are originally captured at 120 and 50 fps, respectively. To provide reliable results, their frame rate is decreased to 30 and 25 fps by coding every fourth and second frame, respectively. Evaluation results are summarized in Table 1 in terms of Bjontegaard Measures [21]. As can be seen, losses in the compression efficiency are relatively small. The largest loss is obtained for small-resolution sequences (classes C and D). It is caused by the impact of the smallest PUs, which are less frequently selected compared to the HM software.

Table 1 Evaluation results for BD-PSNR and BD-Rate

Class	Resolution	BD-PSNR [dB]	BD-rate (%)
A	2560 × 1600	−0.033	1.45
B	1920 × 1080	−0.031	1.56
C	832 × 480	−0.103	2.92
D	416 × 240	−0.148	4.15
E	1280 × 720	−0.069	2.98
4 K	3840 × 2160	−0.026	1.64
Average		−0.068	2.45

6 Implementation results

The architecture of the motion estimation system is specified in VHDL and verified with the modified HM16 reference model [2]. Apart from memory mapping, the VHDL description is independent on the technology selected. The synthesis is performed for FPGA and ASIC technologies using the Altera Quartus II software (ver. 13.1) and Synopsys Design Compiler (ver. 2013.03-SP5-4), respectively. Particularly, FPGA synthesis is performed for Aria II GX FPGA devices (speed grade 5), 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 luma interpolator, which embeds 45 reconfigurable and 54 dedicated filters. The reconfigurable filter at the horizontal stage needs 157 ALUTs and 1517 gates for FPGA and ASIC technologies, respectively. At the vertical stage, resources consumed by each of nine reconfigurable filters are increased to 272 ALUTs and 3049 gates, respectively. The increase stems from the greater number of bits used to represent inputs and intermediate results. Dedicated filters consume less resource (116–166 ALUTs or 1747–2080 gates). The previous version of the interpolator [4] embeds 64 reconfigurable filters, each of which requires much more resources compared to filters used in the new design. For example, the fully featured filter in the previous design consumes 488 ALUTs or 4524 gates. The new architecture reduces the resource consumption mainly by the incorporation of dedicated filters for luma and separate chroma filters.

For the ASIC technology, the design can operate at the frequency of 400 MHz. This performance enables the encoder to allocate about 100 clock cycles per each 8×8 block if the resolution is 2160p@30fps. The estimated power consumption of the ASIC implementation is equal to 293 mW. The high power consumption is caused by memories keeping reference and original pixels. The FPGA implementation can operate at 200 MHz. As a consequence, the throughput is decreased by half.

The luma and chroma paths incorporate 64 dual-port and 16 two-port memory modules, respectively. The modules store reference pixels. Each module in the luma path is 0.75

kB in size. In the case of the chroma path, the size is 1.5 kB. The joint capacity of 72 kB allows the search range of $(-64, 63) \times (-64, 63)$ for both luma and chroma. Wider ranges are possible at the cost of the increased memory size. The original luma samples are stored in a separate dual-port memory with a capacity of 4 kB. This capacity is sufficient to keep samples for one CTU. Since the ME system is pipelined based on 32×32 units, the assignment of memory subspaces is swapped between four processing stages (the writing, the integer-pel ME, the fractional-pel ME, and the merge mode evaluation).

Byun et al. [10] presented the H.265/HEVC integer-pel full search architecture supporting all prediction unit sizes with 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 (422.7 k gates and 76 kB memories). Moreover, the search range is wider $[(-64, 63) \times (-64, 63)]$. The low-power integer-pel design was proposed by Sanchez et al. [11]. Its resource consumption is relatively low (50 k gates and 82 kbit 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, 4. The proposed architecture has the highest throughput of 260 fractional-pel samples per clock cycle ($4 \text{ blocks} \times 64 \text{ luma samples} + 4 \text{ chroma samples}$). Taking into account the luma and chroma interpolation, the previous architecture [4] consumes slightly more resources. Since the proposed one significantly increases the throughput, the parallelism-to-resource ratio is several times higher. The ratio is also much higher compared to other designs. Moreover, the proposed design achieves this ratio for the 2D interpolation, whereas the ratio for most others takes into account the 1D parallelism. The maximal working frequency of the proposed architecture is the highest within the ASIC comparison. Although three implementations [12–14] require less resource, they offer much lower parallelism. As a consequence, declared throughputs are achieved when the size of the prediction unit is selected prior to the interpolation. One design [15]

Table 2 Resource consumption for FPGA and ASIC technologies

Module	Arria II GX (ALUT)	TSMC 90 nm (gate)	Memory (kB)	Power (mW)
MV generator	3598 (7.71 %)	27,836 (6.59 %)	–	3.1
Luma predictor	3412 (7.32 %)	28,982 (6.86 %)	48	164.7
Luma interpolator	24,202 (51.89 %)	240,072 (56.80 %)	–	30.2
Cost estimator	12,143 (26.03 %)	97,124 (22.98 %)	4	27.1
Chroma predictor	546 (1.17 %)	3264 (0.77 %)	24	65.4
Chroma interpolator	2742 (5.88 %)	25,386 (6.00 %)	–	2.5
Total	46,643	422,664	76	293.0

Table 3 Comparison with other FPGA architectures

Design	Afonso [12]	Pastuszak [4]	This study
Technology	Stratix III	Arria II GX	Arria II GX
Clock (MHz)	403	200	200
Resources (ALUT)	4077 + 16547	28,757	26,944
Parallelism (sample/clock)	27 (1D)	64 (1D)	260 (2D)
1000 × parallelism/resources	1.31	2.26	9.64
Throughput	2160p@30fps	1080p@60fps	1080p@60fps
Dynamic power (mW)	379	182	171
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 study
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	277,074	265,458
Parallelism (sample/clock)	12 (1D)	8 (1D)	16 × 12 (2D)	64 (1D)	260 (2D)
1000 × parallelism/resources	0.3972	0.2462	0.1623	0.2310	0.9794
Throughput	2160p@30fps	2160p@60fps	4320p@30fps	2160p@30fps	2160p@30fps
Power (Mw)	–	–	198.6	11.4	30.2
Features	Luma and chroma	Luma	Luma	Luma and chroma	Luma and chroma

supports 4320p@30fps video with the interpolation for three PU 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, and implemented filters are specified in Working Draft 3. Since one of the architectures [13] requires additional memories and the control logic, its actual design efficiency (parallelism/resources) is lower. Compared to the previous version of the interpolator [4], the power consumption of the new one is higher for the ASIC technology. It stems from the continuous processing at all pipeline stages, which increases the switching activity of the circuit.

Most referenced designs support only the luma interpolation [12, 14, 15]. 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. However, the power consumption would be increased. Moreover, the gain in the frequency would not compensate the increased latency of the deeply pipelined processing path composed of the luma predictor, the interpolator, and the cost estimator. The latency of the path affects timing constraints corresponding to the final mode decision and the availability of corresponding MVs. Thus, it would be difficult to determine merge mode candidates and MV costs for the highest throughput.

Although the hardware cost of the interpolator is decreased compared to the previous one [4], the proposed ME system is more complex. Particularly, the compensator in the previous architecture consumes 42.5 k gates, whereas the inter luma/chroma predictor and the cost estimator in the new one require 129.4 k gates. There are two main reasons of the increase. First, separate processing paths for the integer-pel and the fractional-pel are used. Second, four costs are simultaneously evaluated in the fractional-pel path. Since most logic resources are contributed by interpolators (265.5 k gates), the increased complexity in the remaining modules is relatively small in terms of the whole ME system. The throughput is increased by the factor of 1.85 (100/54) and 3.1 (100/32) for the integer-pel and fractional-pel processing, respectively.

7 Conclusion

The ME architecture is developed for the H.265/HEVC encoder. The design embeds two parallel processing paths for the integer-pel and the fractional-pel motion estimation. The paths share the same dual-port memories. Internal buffers and the scheduling allow the writing of reference samples through the port assigned to the fractional-pel path. The architecture supports TZS for 8×8 prediction blocks. The motion estimation for larger blocks is performed by utilizing results of the 8×8 search. The search for rectangular PUs is performed only at the fractional-pel level

and reuses partial costs computed for $2N \times 2N$ PUs. The design achieves the best ratio of the throughput to hardware resources compared to other designs. The design can check about 100 integer-pel MVs for each 8×8 input block when encoding 2160p@30fps video at the 400 MHz. Within future works, the proposed ME system will be integrated with the intra encoder [17] to support inter modes.

Acknowledgments This research was supported in part by PL-Grid Infrastructure.

Open Access This article is distributed under the terms of the Creative Commons Attribution 4.0 International License (<http://creativecommons.org/licenses/by/4.0/>), which permits unrestricted use, distribution, and reproduction in any medium, provided you give appropriate credit to the original author(s) and the source, provide a link to the Creative Commons license, and indicate if changes were made.

References

- ITU-T Recommendation H.265 and ISO/IEC 23008-2 MPEG-H Part 2, High efficiency video coding (HEVC) (2013)
- HEVC software repository—HM-16.0 reference model. <https://hevc.hhi.fraunhofer.de/trac/hevc/browser/tags/HM-16.0> (2015). Accessed 29 June 2015
- ITU-T Rec. H.264 and ISO/IEC 14496-10 MPEG-4 Part 10, Advanced video coding (AVC) (2005)
- Pastuszak, G., Trochimiuk, M.: Architecture design of the high-throughput compensator and interpolator for the H.265/HEVC encoder. *J. Real Time Image Process.* Online first articles (2014)
- 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)
- 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)
- 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)
- 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)
- Oktem, S., Hamzaoglu, I.: An efficient hardware architecture for quarter-pixel accurate H.264 motion estimation. In: *10th Euromicro Conference on Digital System Design*, pp. 1142–1143 (2007)
- Byun, J., Jung, Y., Kim, J.: Design of integer motion estimator of HEVC for asymmetric motion-partitioning mode and 4K-UHD. *Electron. Lett.* **49**(18), 1142–1143 (2013)
- 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)
- 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)
- 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)
- 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)
- 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)
- 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)
- Pastuszak, G., Abramowski, A.: Algorithm and architecture design of the H.265/HEVC intra encoder. *IEEE Transactions on Circuits and Systems for Video Technology*, pp. 1 (2015). doi:10.1109/TCSVT.2015.2428571
- Bossen, F.: Common test conditions and software configurations, JCT-VC-L1100. JCT-VC, Geneva (2013)
- Ultra video group, test sequences: (online). <http://ultravideo.cs.tut.fi/#testsequences> (2015). Accessed 29 June 2015
- Xiph.org: test media, <http://media.xiph.org/video/derf/> (2011). Accessed 29 June 2015
- Bjontegaard, G.: Calculation of average PSNR differences between RD-Curves. In: *ITU-T VCEG-M33, VCEG 13th Meeting*

video/audio processing and compression, high-performance digital ICs.

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/

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.