J Sign Process Syst (2016) 82:391–402
DOI 10.1007/s11265-015-1021-5

Optimization of the Adaptive Computationally-Scalable Motion Estimation and Compensation for the Hardware H.264/AVC Encoder

Grzegorz Pastuszak[1] · Mariusz Jakubowski[2]

Received: 4 July 2014 / Revised: 2 April 2015 / Accepted: 28 June 2015 / Published online: 21 July 2015

Abstract The adaptive computationally-scalable motion estimation algorithm and its hardware implementation allow the H.264/AVC encoder to achieve efficiencies close to optimal in real-time conditions. Particularly, the search algorithm achieves results close to optimum even if the number of search points assigned to macroblocks is strongly limited and varies with time. The architecture implementing the algorithm developed and reported previously takes at least 674 clock cycles to interpolate and load reference area, and the number cannot be decreased without decreasing the search range. This paper proposes some optimizations of the architecture to increase the maximal throughput achieved by the motion estimation system even four times. Firstly, the chroma interpolation follows the search process, whereas the luma interpolation precedes it. Secondly, the luma interpolator computes 128 instead of 64 samples per each clock cycle. Thirdly, the number of on-chip memories keeping interpolated reference area is increased accordingly to 128. Fourthly, some modules previously working at the base frequency are redesigned to operate at the doubled clock. Since the on-chip memories do not store fractional-pel chroma samples, their joint size is reduced from 160.44 to 104.44 kB. Additional savings in the memory size are achieved by the sequential processing of two reference-picture areas for each macroblock. The architecture is verified in the real-time FPGA hardware encoder. Synthesis results show that the updated architecture can support 2160p@30fps encoding for 0.13 μm TSMC technology with a small increase in hardware resources and some losses in the compression efficiency. The efficiency is improved when processing smaller resolutions.

Keyword Video coding · Motion estimation · H.264/AVC · FPGA · Very large-scale integration (VLSI) · Architecture design

1 Introduction

The motion estimation (ME) is the most computationally-intensive part of video encoders. It allows high compression efficiencies by exploiting temporal redundancy between successive pictures. The ME aims to find the best matching between the block from the currently-coded picture and previously-coded ones. The ME algorithm must search a number of possible candidate blocks in the reference picture. Their displacement from the position of the block in the current picture is signaled by motion vectors (MVs).

The ability to adapt the search path (series of MVs) to local statistics allows compression-efficient coding with a small number of checked MVs [1–4]. Furthermore, the selection between different search strategies makes the estimation more robust for different motion activities [5, 6]. On the other hand, hardware architectures usually apply the full search (FS) due to its regularity [7–15]. This approach involves a great amount of hardware resources when the high throughput and the wide search range are required. Moreover, the number of clock cycles utilized for each macroblock is difficult to scale. The design described in [13] reduces hardware resources and has the wide search range [-128,128). However, it can densely check only MVs around the predictor, and assumed access

✉ Grzegorz Pastuszak
G.Pastuszak@ire.pw.edu.pl

1 Institute of Radioelectronics, Warsaw University of Technology, Nowowiejska 15/19, Warsaw 00-665, Poland

2 Faculty of Engineering, Academy of Business and Finance VISTULA, Stokłosy 3, Warsaw 02-787, Poland

to external memories is highly inefficient on account of short bursts and subsampling. Some architectures [16, 17] supports Diamond Search and Cross Search. Although the number of checked MVs is reduced, the resource consumption is still significant and the implemented search patters are not efficient in the case of high motion activity. Separate macroblock stages for integer-pel and fractional-pel ME used in the referenced designs force the encoder to select the best inter mode based on the simplified cost function such as Sum of Absolute (Transformed) Differences (SA(T)D). The computationally-scalable solution was proposed in [18]. The scaling is achieved by limiting the search range and skipping smaller block sizes. The number of clock cycles can vary strongly, which makes it difficult to apply in the macroblock-pipelined encoder. Moreover, the throughput is limited to 720p videos.

In our previous work [19], the adaptive computationally-scalable motion estimation architecture was proposed for H.264/AVC [20]. The architecture applies the unconventional dataflow, which removes constraints on the number, the order, and the fractional accuracy of MVs. As a consequence, it can employ different search strategies (e.g., Diamond Search and Three Step Search) to achieve near optimal results using a small number of MVs. Furthermore, the design is computationally scalable, i.e., it allows the tradeoff between the number of utilized MVs and the compression efficiency. Although the architecture can achieve near optimal results with a small number of checked MVs, the limitation on the throughput results from the interpolation and the loading of reference and interpolated samples to on-chip memories before the search process. In particular, at least 674 base-clock cycles are required for each macroblock. The throughput can be increased by using several ME processing paths. However, the hardware cost could be unacceptable.

This paper describes optimizations of the adaptive computationally-scalable ME architecture [19]. They break the previous limitation on the number of clock cycles, allowing the architecture to increase the throughput four times at a relatively small increase in hardware resources and some losses in the compression efficiency. The higher throughput is achieved by a number of optimizations. Firstly, the chroma interpolation follows the search process, whereas the luma interpolation still precedes it. Secondly, the new version applies additional pipelining of some modules to operate at the doubled clock. Thirdly, the luma interpolator computes 128 instead of 64 samples per each clock cycle. The number of on-chip memories in the compensator is increased from 64 to 128. The total capacity of on-chip memories is reduced from 160.44 to 104.44 kB despite of the fact that the coarse-level memory is increased from 32 to 64 kB to process 2160p sequences. Moreover, the support for two reference pictures does not increases the capacity due to the sequential processing of two search areas for each macroblock.

The rest of the paper is organized as follows. Section 2 reviews the previous version of the adaptive computationally-scalable ME architecture. Section 3 presents optimizations introduced in the new version. In Section 4, implementation results are provided. The paper is concluded in Section 5.

2 Previous Version of the Architecture

The architecture before optimizations can support real-time coding with quarter-pel MV accuracy and one reference picture (RP) for 1080p@30fps at 200 MHz clock. The processing with two RPs requires a higher frequency or the resolution decreased to 720p@30fps. The block diagram of the ME system is presented in Fig. 1. The system is composed of the MV generator, the compensator with the buffer for reference and original data, the coarse FS estimator, and the interpolator. Modules communicate with the encoder controller, the external memory controller, the intra predictor [21], and the residual buffer.

The system employs two-level hierarchical ME procedure. At the first stage, the coarse FS estimator performs FS on the wide search area subsampled with 16:1 ratio. To reduce the noise influence on initial MV accuracy, each sample of the coarse search area is obtained by averaging of 16 luma samples of the current and reference picture. Coarse FS is performed only for 16×16 macroblocks by using their 4×4 representations. When the coarse FS process is completed, the interpolator computes fine-search-area samples with the quarter-pel precision within the [−8, 8) range in both dimensions around the initial MV obtained from the coarse FS. The interpolator reads 40×40 reference luma samples and corresponding chroma ones from the external memory. Thus, 2400 reference samples are read for each macroblock when chroma

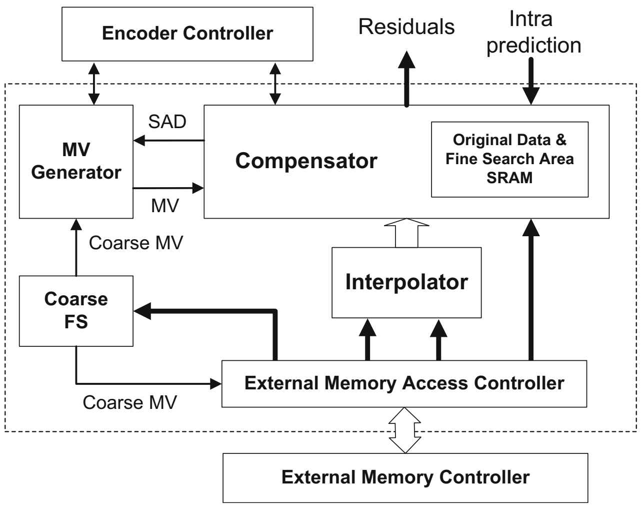


Figure 1 Motion estimation system architecture.

format is 4:2:0. In each base-clock cycle, 128 samples are loaded into the fine-search-area buffer in the compensator. As the buffer consists of 64 memory modules with the one-sample data width, writing is performed at the doubled clock rate.

The ME system applies three macroblock-level pipeline stages. The first stage is the coarse FS estimator. The second one interpolates and loads original and reference samples into the fine-search-area buffer. The third stage embeds the MV generator and the compensator that reads samples from the buffer. The MV generator follows successive steps of the Multi-Path-Search algorithm [6, 19] and determines MVs to check. Based on the MVs, the compensator computes residuals and SADs for 16×16 luma block and its partitions. SAD is fed to the MV generator selecting ME algorithm branches. The compensator also supports intra modes. Particularly, an intra prediction is first written to the buffer while coding a macroblock. Then, intra residuals are computed in the same way as for inter modes. The ME system also supports computations for all three chroma formats. When the final macroblock mode is inter, the MV generator takes final MVs for macroblock partitions, forward them to the compensator with chroma indicators, and switches to the next macroblock.

Multi-Path Search utilizes spatial correlations between MVs of neighboring macroblocks and fast search strategies. At the beginning, it checks some MVs inferred from neighbouring macroblocks (including median prediction) and zero-MV. In the second phase, either Diamond Search or Three Step Search is executed when the motion activity of neighboring macroblocks is low or high, respectively. Subsequently, Three Step Search is performed around each MV analyzed at the first phase. The subpixel estimation around the best MV found so far is performed at the third phase. After that, fast full search is executed until the number of clock cycles assigned to a macroblock is not utilized. However, the search can also be terminated earlier. The evaluation of successive MVs is performed for 16×16 luma blocks. Up to eight best results are buffered and forwarded to the reconstruction loop and the rate-distortion analysis. The final macroblock mode can include block contributions computed for different MVs.

3 Optimizations

In the ME architecture described in Section 2, the main problem is the limited throughput of the interpolator and the write speed to the buffer in the compensator. In particular, the output stage of the interpolator and the write stage of the compensator operate at the doubled frequency. In spite of this enhancement, the number of base-clock cycles taken to transfer integer-pel and interpolated samples from the interpolator to the buffer is at least 674 for one macroblock. Additional cycles are taken to write intra predictions. The direct increase of the frequency is not possible as critical paths are located at the interface between the two modules. Therefore, to decrease the number of clock cycles, the architecture is redesigned.

Following subsections describe optimization details introduced at the ME system level and in three modules: the coarse estimator, the interpolator, and the compensator. The MV generator remains almost unchanged compared to the previous version. Since the adaptation of the memory access controller to the higher frequency is straightforward, its description is omitted.

3.1 ME System Level

In terms of the ME system, two main modifications are introduced in the architecture. The first modification consists in the use of the separate chroma interpolator embedded in the processing path of the compensator. As a consequence, the interpolator preceding the compensator operates only on luma samples. The number of clock cycles utilized by the luma interpolation is decreased three times compared to the case when all components are processed sequentially. However, not-interpolated chroma samples still have to be written into the buffer with an alternative way. They are transferred through the same path as original samples. Particularly, the path allows the parallelism of eight samples per clock cycle. Higher throughputs are possible provided that a wider bandwidth to the external memory is available.

The second modification introduced to the ME architecture consists in the doubled parallelism at the interface between the luma interpolator and the compensator: from 64 to 128. To balance the increased throughput of the interface, the working frequency of preceding modules (the luma interpolator and the memory access controller) is doubled. Also, the coarse estimator operates at the increased frequency providing the same results with a smaller number of base-clock cycles.

The two design modifications described above lead to the reduction of the minimal number of clock cycles required to process one macroblock. In particular, the number of doubled-clock cycles taken to write to the buffer in the compensator is 322 (390 for 4:2:2) and includes:

- interpolated luma samples written in 156 cycles (4 stripes×(32 stripe length+7-sample extension/stripe)),
- original samples written in 64 cycles (2 stripes×16 columns for luma and 2 components×2 stripes×8 columns for chroma 4:2:2),
- reference chroma samples written in 102 and 170 cycles for 4:2:0 and 4:2:2 formats, respectively (2 components× 17 stripe length×3 (or 5) stripes).

The interpolator produces four stripes from five input stripes due to the vertical extension. Valid samples are released after the first stripe is processed. Particularly, they start to appear with the delay of 46 cycles (40-cycles for the stripe processing and 6 cycles due to the pipeline). The delay does not negatively affect the throughput since other original pixels and reference chroma samples are written meantime.

All writes are performed with stripes of the eight-sample height, as shown in Fig. 2. The number of doubled-clock cycles is less than 400. Therefore, the frequency of 200 (400) MHz for the base clock (the doubled clock) is sufficient to satisfy requirements on write cycles while encoding 2160p@30fps video. The writing of intra modes is performed in parallel with original and reference chroma samples. Hence, no additional cycles are required.

In the computationally-scalable architecture, the number of checked motion vectors is limited by the number of clock cycles assigned to a macroblock. The evaluation reported in [19] proved that Multi Path Search achieves the compression efficiency close to the optimal for about 50 checked MVs corresponding to 200 base-clock cycles. Additional clock cycles are taken to compute residuals for intra predictions. For example, four intra 16×16 and nine 8×8 modes require 16 and 36 clock cycles, respectively. Four chroma predictions computed for a macroblock coded with the intra mode take 32 cycles. Even when skipping intra 4×4 predictions and neglecting the delay of the reconstruction loop, the total number of clock cycles exceeds 200 available for the 2160p@30fps video. On the other hand, 1080p@60fps can be supported

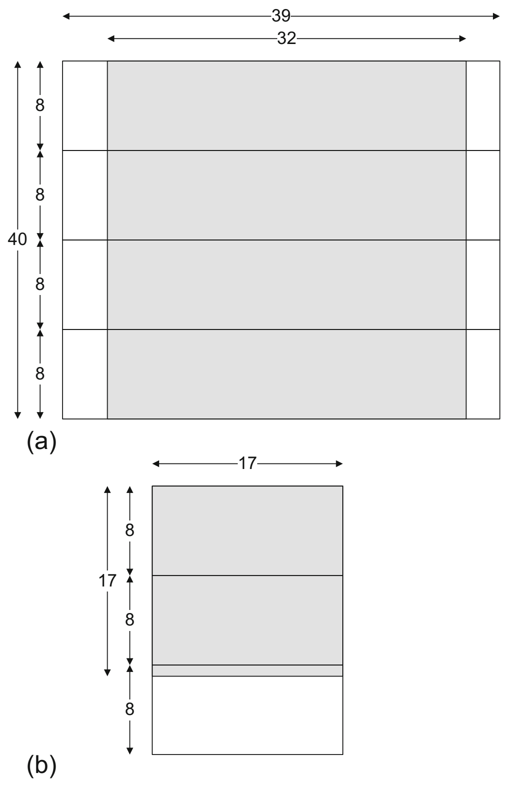


Figure 2 Division into stripes for reference samples: **a** luma; **b** chroma.

since 400 clock cycles are available. The support for 2160p@30fps requires the computation scaling, i.e., less MVs and intra modes can be checked. The scaling can be sufficient if a part of intra 8×8 modes is checked, plane modes are skipped, and the MV number is decreased to 40. This limitation involves losses in the compression efficiency.

Due to the delay between the generation of a MV and obtaining the corresponding SAD (see Subsection 3.4), the generation continuity is interrupted between successive steps of the search algorithm. Particularly, SAD for all SPs in one step must be obtained to find the best search centre for the following step. This dependence introduces time slots when inter predictions are not processed at particular stages of the pipeline. The slots are utilized to process intra modes, for both luma and chroma. When the macroblock mode selected for luma is inter, corresponding MVs are used to compute chroma residuals. If the mode decision involves a significant delay, the number of checked MVs should be decreased.

The previous version of the architecture increases the size of on-chip memories to support two RPs. The new version does not require the increase due to a modified dataflow. When two reference pictures are used, their fine search areas are fetched and checked sequentially. While the motion estimation and compensation is performed for the first reference picture, the search area for the second is fetched from external memories, interpolated, and written to buffers in the compensator. When all these operations are finished, the second reference picture is checked, while data for the next macroblock are fetched. The sequential processing for two RP decreases the maximal throughput by half compared to the case for one RP.

3.2 Coarse Estimator

The dataflow of the coarse FS module is depicted in Fig. 3. The module embeds 16 memories each of which keeps one coarse sample per each 4×4 block. Their capacity is selected to keep 16 macroblock lines. Two lines are dedicated to the current-picture data whereas the remaining to the RP data. 10 and 4 lines are assigned to the first and the second RP, respectively. At the beginning of each inter-picture coding, the memories are initialized with reference coarse-picture lines until the associated subspace is filled with up to 14 macroblock lines. Then, one current-picture line is read in. When one current-picture line is analyzed, the following is read in according to the ping-pong scheme. Similarly, the RP lines are exchanged when they are outside the top boundary of the coarse search area. If the coarse data are loaded, FS is started for successive macroblocks.

At the beginning of the macroblock processing, the current-picture macroblock representation is loaded to registers. Then, the search engine reads reference representations for successive search points (SPs). They correspond to actual MVs with components being the multiplication of four. 16

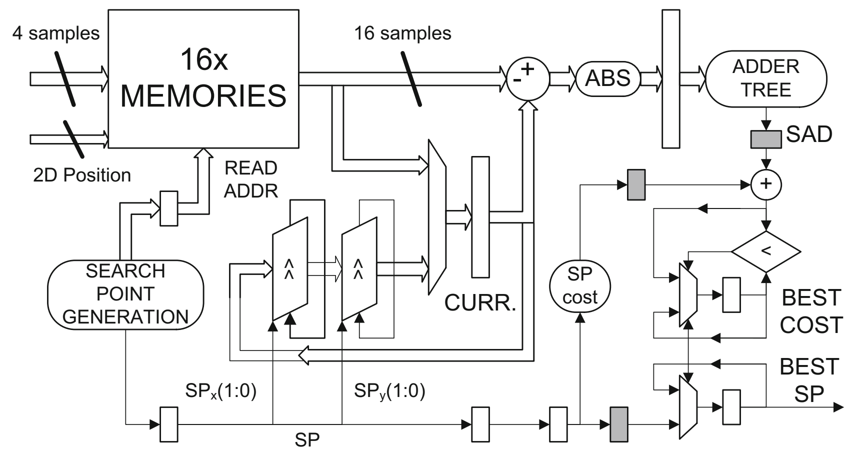


Figure 3 Architecture of the coarse FS estimator. Additional registers are colored grey.

read reference samples are subtracted from corresponding original values, and absolute values are forwarded to the adder tree. The addition result, which is the coarse-level SAD, is increased by the SP cost reflecting estimated rate of MV components. Then, such a total cost is compared with the currently minimal one. If the new cost is smaller, it becomes the currently minimal one. If two RPs are used, the processing for the second follows that for the first. Two separate coarse-level MVs are obtained for one macroblock.

As analyzed in Subsection 3.1, the minimal number of doubled-clock cycles assigned to a macroblock for one RP is 400. This number is sufficient to check coarse SPs in the range of [−10; 10]×[−9; 9], which corresponds to the range of [−40; 40]×[−36; 36] at the fine level. If more clock cycles are available, the range can be extended. For example, the coarse-level range of [−14; 14]×[−13; 13] is achieved when the number of cycles is doubled. The impact of the range limitation on the compression efficiency is noticeable only for sequences with a high motion activity. However, losses in the compression efficiency are below 0.01 dB (tests for 1080p sequences used in Subsection 4.2). Since the range of the second RP is limited vertically (+/−4), the operation at the minimal number of clock cycles assigned to each RP does not affect the search result.

Generally, the dataflow of the new version of the coarse estimator remains the same as in the previous version. However, there are two main differences which shorten critical paths. Firstly, an additional pipeline stage is inserted. Additional registers are coloured gray in Fig. 3. Secondly, the SP generation is simplified. Particularly, the generation is performed always using the ring search pattern without skipping SPs falling outside the available search range (e.g., picture/slice boundaries). Although comparators checking the search range are still used, they are removed from the generation subcircuit. If a SP falls outside, it is marked as invalid and is not taken into account at the final stage to select the best SP. Since skipping of SPs requires much less clock cycles than the interpolation for a macroblock, the modification has no impact on the result of the coarse estimation.

3.3 Luma Interpolator

The architecture of the new version of the interpolator is depicted in Fig. 4. Labels assigned to fractional positions are explained in Fig. 5. The module accepts the column of eight luma samples in a clock cycle. As a consequence, 128 samples are produced in each clock cycle (16-times more than at the input). Computations of sub-pel positions for chroma components are shifted to the following macroblock-level stage in the compensator. As a consequence, the interpolator supporting only the luma component is simplified. Particularly, two fractional bits are removed from each register keeping a sample. The two bits were indispensible to represent interpolated chroma at odd integer positions.

The interpolator embeds the memory for the vertical extension of processed columns (the convolution involves the extension). The memory works as the 40-cycle delay register (DLY), which corresponds to the width of the input reference area. The data width is adjusted to match six samples. Samples read from the memory (previous/upper eight-sample row) and from the interpolator input registers (current eight-sample row) are forwarded to integer-pel pipeline registers (G) and vertical interpolators. Due to the extension (two references above and three below), eight vertical interpolators refer to 13 of these samples to compute the column of eight half-pel samples. As the vertical extension is no longer necessary (except the one-sample extension at the bottom), the number of following pipeline registers is reduced. The pipeline carries integer-pel and half-pel samples arranged into nine- or eight- sample columns. Horizontal half-pel interpolations refer to samples at successive stages. The interpolator embeds eight or nine filter cores for each of three half-pel positions (horizontal, vertical, and horizontal-vertical). The computation of quarter-pel samples is performed at the second-last stage. In particular, quarter-pel samples are obtained by the addition of relevant integer and half-pel samples.

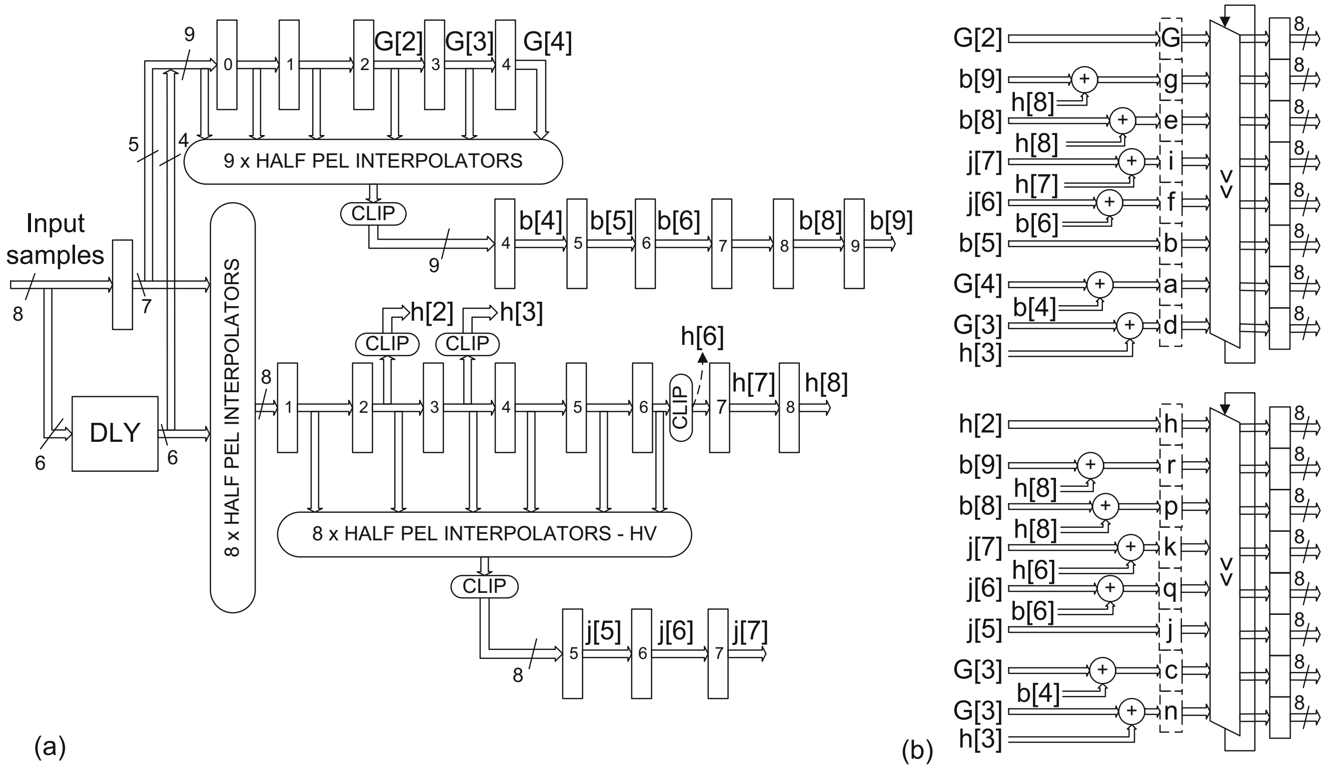


Figure 4 Architecture of the interpolator: **a** pipeline with half-pel filters; **b** output stage with quarter-pel filters.

Compared to the previous version, all stages operate at the doubled clock and the output stage is extended to process 128 samples. To enable higher frequencies, two main modifications are introduced. Firstly, reconfiguration multiplexers for chroma are removed. Secondly, half-pel filter cores are pipelined using two stages as shown in Fig. 6. In horizontal and vertical filters (b and h paths in Fig. 6a), the pipelining introduce the delay of one clock cycle. On the other hand, the second-level filter assigned to j positions (see Fig. 6b) does not introduce an additional delay. This stems from the fact that the value kept in the intermediate register is computed one cycle earlier. As a consequence, timing dependencies between half-pel paths (b, h, and j) remain unchanged compared to the previous version of the architecture. If the j path was delayed by the filter, the remaining paths would be extended by appending an additional register stage. The number of pipeline stages of the interpolator is the same as in the previous version. Although vertical and horizontal half-pel filters increase the delay by one cycle, one register stage used to interpolate odd chroma positions at the input is removed. Since the module operates at the doubled clock, its latency is decreased by half.

G_{00}	a_{00}	b_{00}	c_{00}	G_{10}	a_{10}	b_{10}	c_{10}	G_{20}
d_{00}	e_{00}	f_{00}	g_{00}	d_{10}	e_{10}	f_{10}	g_{10}	d_{20}
h_{00}	i_{00}	j_{00}	k_{00}	h_{10}	i_{10}	j_{10}	k_{10}	h_{20}
n_{00}	p_{00}	q_{00}	r_{00}	n_{10}	p_{10}	q_{10}	r_{10}	n_{20}
G_{01}	a_{01}	b_{01}	c_{01}	G_{11}	a_{11}	b_{11}	c_{11}	G_{21}
d_{01}	e_{01}	f_{01}	g_{01}	d_{11}	e_{11}	f_{11}	g_{11}	d_{21}
h_{01}	i_{01}	j_{01}	k_{01}	h_{11}	i_{11}	j_{11}	k_{11}	h_{21}
n_{01}	p_{01}	q_{01}	r_{01}	n_{11}	p_{11}	q_{11}	r_{11}	n_{21}
G_{02}	a_{02}	b_{02}	c_{02}	G_{12}	a_{12}	b_{12}	c_{12}	G_{22}

Figure 5 Fractional position labels and indices in the RP domain.

The filter cores do not embed rounding adders. Instead, three input adders in the vertical interpolators have the carry input at the least significant bit set to logic one. This is equivalent to adding 0.5 to each input argument.

In the previous version, the output stage embeds multiplexers to compute two fractional-pel positions in each base-clock cycle. As the new version applies the doubled clock to all stages, the two fractional-pel positions are computed simultaneously. Therefore, the number of samples computed in one clock cycle increases from 64 to 128. Samples computed in even and odd cycles in the previous version are now assigned to separate 8×8 output blocks (see Fig. 4b). Each block consists of eight eight-sample columns, and each column corresponds to one fractional-pel position. Figure 7 shows which samples appear at the output interface in some cycles. In successive clock cycles, eight-sample columns within blocks are rotated to provide different fractional-pel positions to each output column. Apart from the block parallelism, the order of released data is the same as in the previous version.

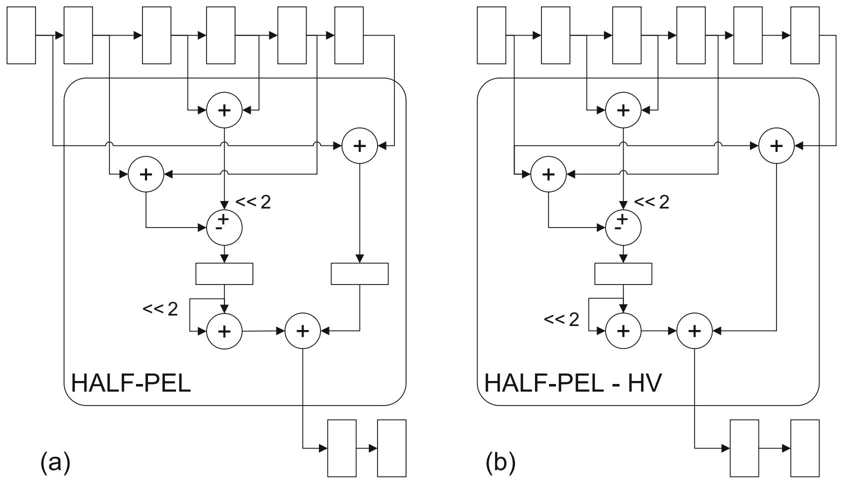


Figure 6 Half-pel filter cores for b/h (**a**) and j (**b**) positions.

3.4 Compensator

The architecture of the compensator is shown in Fig. 8. The module employs seven pipeline stages and processes one 8×8 block per clock cycle. First three stages work at doubled clock frequency. The first stage is composed of 128 memories able to store two fractional-pel luma subspaces in the range of [−8, 8). Additionally, the memories store not-interpolated reference chroma samples and intra predictions. The subspaces are switched between write and read ports in the ping-pong arrangement for inter modes. Each memory stores every eighth sample both in the horizontal and vertical dimension.

a_{00}	d_{10}	G_{20}	▨	▨	▨	▨	▨
a_{01}	d_{11}	G_{21}	▨	▨	▨	▨	▨
a_{02}	d_{12}	G_{22}	▨	▨	▨	▨	▨
a_{03}	d_{13}	G_{23}	▨	▨	▨	▨	▨
a_{04}	d_{14}	G_{24}	▨	▨	▨	▨	▨
a_{05}	d_{15}	G_{25}	▨	▨	▨	▨	▨
a_{06}	d_{16}	G_{26}	▨	▨	▨	▨	▨
a_{07}	d_{17}	G_{27}	▨	▨	▨	▨	▨

c_{00}	n_{10}	h_{20}	▨	▨	▨	▨	▨
c_{01}	n_{11}	h_{21}	▨	▨	▨	▨	▨
c_{02}	n_{12}	h_{22}	▨	▨	▨	▨	▨
c_{03}	n_{13}	h_{23}	▨	▨	▨	▨	▨
c_{04}	n_{14}	h_{24}	▨	▨	▨	▨	▨
c_{05}	n_{15}	h_{25}	▨	▨	▨	▨	▨
c_{06}	n_{16}	h_{26}	▨	▨	▨	▨	▨
c_{07}	n_{17}	h_{27}	▨	▨	▨	▨	▨

(a)

d_{80}	G_{90}	g_{20}	e_{30}	i_{40}	f_{50}	b_{60}	a_{70}
d_{81}	G_{91}	g_{21}	e_{31}	i_{41}	f_{51}	b_{61}	a_{71}
d_{82}	G_{92}	g_{22}	e_{32}	i_{42}	f_{52}	b_{62}	a_{72}
d_{83}	G_{93}	g_{23}	e_{33}	i_{43}	f_{53}	b_{63}	a_{73}
d_{84}	G_{94}	g_{24}	e_{34}	i_{44}	f_{54}	b_{64}	a_{74}
d_{85}	G_{95}	g_{25}	e_{35}	i_{45}	f_{55}	b_{65}	a_{75}
d_{86}	G_{96}	g_{26}	e_{36}	i_{46}	f_{56}	b_{66}	a_{76}
d_{87}	G_{97}	g_{27}	e_{37}	i_{47}	f_{57}	b_{67}	a_{77}

n_{80}	h_{90}	r_{20}	p_{30}	k_{40}	q_{50}	j_{60}	c_{70}
n_{81}	h_{91}	r_{21}	p_{31}	k_{41}	q_{51}	j_{61}	c_{71}
n_{82}	h_{92}	r_{22}	p_{32}	k_{42}	q_{52}	j_{62}	c_{72}
n_{83}	h_{93}	r_{23}	p_{33}	k_{43}	q_{53}	j_{63}	c_{73}
n_{84}	h_{94}	r_{24}	p_{34}	k_{44}	q_{54}	j_{64}	c_{74}
n_{85}	h_{95}	r_{25}	p_{35}	k_{45}	q_{55}	j_{65}	c_{75}
n_{86}	h_{96}	r_{26}	p_{36}	k_{46}	q_{56}	j_{66}	c_{76}
n_{87}	h_{97}	r_{27}	p_{37}	k_{47}	q_{57}	j_{67}	c_{77}

(b)

Figure 7 Location of samples at the interpolator output interface in the third (**a**) and the ninth (**b**) cycle; *Dark boxes* correspond to invalid data.

Reference and original data are read from the memories in the alternating way. The third stage shifts cyclically reference samples between positions in both dimensions to support all MVs. An example of this operation is illustrated in Fig. 9. The residuals are computed in the following stage. They are output through an 8×8 sample interface at the fifth stage that is clocked with the main clock. The following stages compute SADs for 8×8 blocks (ABS and adder tree) and accumulate them for 16×16 luma predictions (ACC). The accumulated SADs are used by the MV generator to determine the best MV at a given processing step. Eight best 16×16 modes (intra/inter, different MV/directions) are collected based on accumulated SADs in the rank list. If a mode is forwarded to the rate-distortion analysis, it is removed from the rank list.

The joint memory capacity in the compensator is reduced in the new version of the architecture since interpolated chroma is not stored. In particular, each interpolated component requires 32 kB. In the previous version, the capacity of 32 kB is also used to store intra predictions for four different QP (4×4 and 8×8 intra modes) and original samples for two macroblocks (ping-pong exchange). In the new version, additional reductions are achieved by limiting the processing to one QP for partitioned intra modes and skipping the 4:4:4 format. As a consequence, 8 kB is sufficient to store intra modes, original pixels, and reference chroma samples. Finally, the memory capacity in the compensator is reduced from 128 to 40 kB.

The new version of the compensator embeds 128 memory modules instead of 64. Although the number of memory modules is doubled, the modification has no impact on their joint capacity. Particularly, the address space of each module is decreased by half. The modification is performed to support the increased number of samples received from the interpolator. Memories are assigned to two groups, each of which consists of 64 modules. Two 8×8 sample blocks received from the interpolator are written to separate groups.

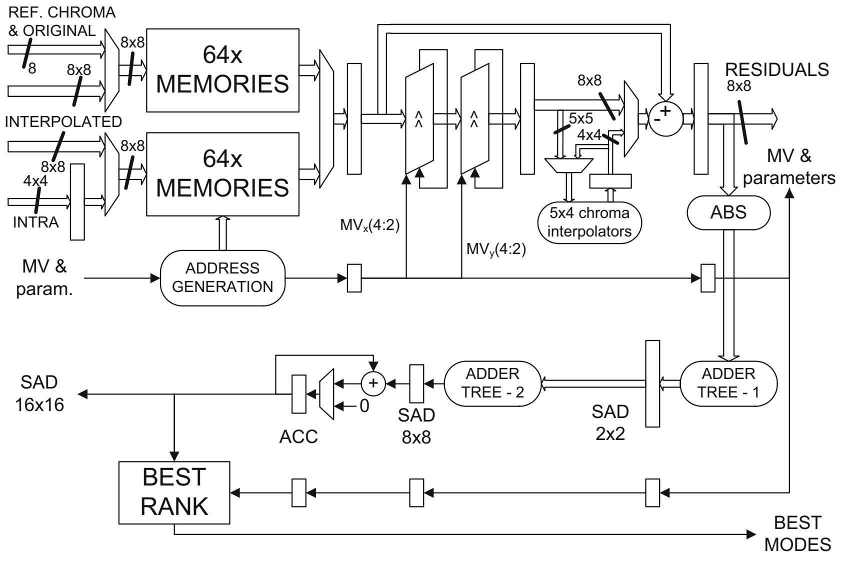


Figure 8 Compensator architecture.

The memory division into two groups allows the simplification of the write stage. In the previous version of the architecture, the write stage shares access between three input interfaces used to carry reference/interpolated, original, and intra-predicted samples. In the new version, original/chroma and intra-predicted samples are written into separate memory groups. This way multiplexing is simpler, and original and intra-predicted samples can be written in parallel.

When reference/interpolated samples are written to memories, the compensator receives a sequence of different MVs from the MV generator. For each MV, 8×8 block is read from one group of memories in dependence on the fractional position of the MV.

Although the computation of residuals and SADs is performed similarly as in the previous version of the architecture, inter chroma predictions are computed in the array of dedicated interpolators, as shown in Fig. 8. The array consists of 5×4 elementary chroma interpolators depicted in Fig. 10. The interpolation for a 4×4 output block is performed in successive two clock cycles at the doubled clock. The first and the second cycle are assigned to the horizontal and the vertical processing, respectively. After the horizontal phase, the transposed result is fed back. The transposed result of the vertical phase is used to compute residuals in the main processing path.

As a consequence of the two-phase interpolation, inter chroma predictions are obtained with the two-cycle delay. The delay involves the modification of the processing order in the main processing path. When the interpolated chroma block is selected to compute residuals, reference samples are taken to perform the interpolation for the following chroma block. Original chroma samples are read from memories with the two-cycle delay to keep the data consistency in the pipeline. If two RP are used, chroma processing is performed for each of them after the corresponding luma search.

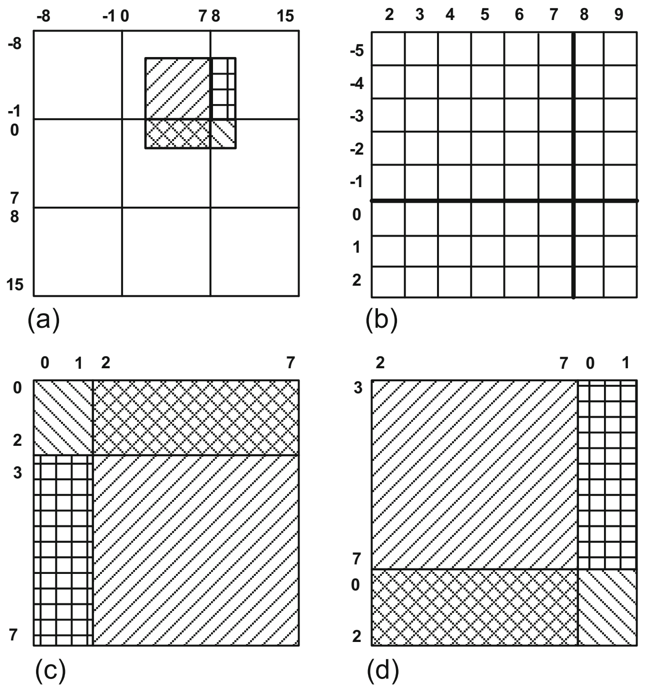


Figure 9 Sample arrangement of the read 8×8 block for MV=(2,−5): **a** location within search area; **b** block samples with their search-area indices; **c** block samples read from memories with 2D memory indices; **d** block samples after the rotation with 2D memory indices.

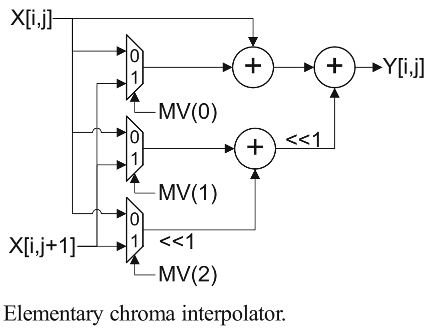


Figure 10 Elementary chroma interpolator.

4 Implementation Results

4.1 Synthesis Results

All modules of the ME system are described using VHDL. The design is validated through the comparison with results produced by the previous version of the architecture. The synthesis is performed with the Altera Quartus II software, targeted for Arria II GX FPGA devices. The ME system is integrated with other parts of the hardware video encoder [22], and the whole encoder is verified in real-time conditions with the Arria II GX device. The design can work at the base clock of 100 MHz for the speed grade equal to 5. All redesigned modules (except the chroma interpolation) operate at the doubled clock of 200 MHz. Evaluations in hardware conditions show that the minimal number of base-clock cycles taken for each macroblock is about 200. This number includes cycles utilized for access to external DDR2 memories (64 bits at 200 MHz) for one RP. The achieved throughput enables 1080p@60fps encoding. When two reference pictures are used, the minimal number of base-clock cycles is doubled. The bottleneck of the ME system is the interpolation and loading of reference and interpolated samples to on-chip memories before the search process.

The design is also synthesized with Synopsys Design Compiler using TSMC 0.13 μm standard cell library. This technology allows frequencies increased to 200 and 400 MHz for the base and doubled clock, respectively. Table 1 shows the resource consumption for each module of the ME system before (ver. 1) and after (ver. 2) the optimization. The results are provided for FPGA and ASIC technologies. The memory resources are not taken into account. However, the new version supports one or two reference pictures with the reduced memory size of 104.44 kB. As can be seen, the ME system consumes 5.8 and 5 % more logic for ASIC and FPGA, respectively. The increase is most apparent for the compensator, where the chroma interpolator is incorporated. Although the interpolator preceding the compensator is simplified to process only the luma component, the modifications introduced to the output stage increase the complexity. These two optimizations have the opposite impact on hardware resources. Their strength depends on the technology. The number of ALUTs is decreased for the FPGA implementation, whereas the number of gates is increased for the ASIC technology.

Table 1 Synthesis Results for FPGA Arria II GX and TSMC 0.13 μm

Module		TSMC 0.13 μm [gate]		Arria II GX [ALUT]	
		Ver. 1	Ver. 2	Ver. 1	Ver. 2
Compensator	1×RP	53,475	57,482	7309	8191
	2×RP	53,831	57,855	7377	8263
MV generator	1×RP	8291	8301	2162	2165
	2×RP	10,423	10,434	2696	2699
Interpolator		67,751	70,798	6800	6659
Memory access control		3674	4207	611	715
Coarse FS		7216	7865	1316	1376
Total	1×RP	140,407	148,653	18,198	19,106
	2×RP	142,895	151,159	18,800	19,712

4.2 Compression Efficiency

The optimized ME system integrated in the hardware encoder is evaluated in terms of compression efficiency for three mode configurations specified in Table 2. The configurations correspond to different limitations on the number of base-clock cycles available for each macroblock. Limitations on the number of clock cycles inferred from the intra prediction are described in [21]. Due to the delay of the rate-distortion-based mode decision required to generate chroma predictions (about 20 cycles), the configurations allow less SPs/MVs than it stems from the number of clock cycles available for one macroblock. Six 1080p sequences are evaluated [23]. QP is equal to 22, 27, 32, and 37. 51 frames are coded, where only the first is intra. The entropy mode is CAVLC. The RD-optimized mode decision is used (the hardware encoder supports it). The search range in the reference JM17.0 software is set to (−64, 63)×(−64, 63), two reference frame and all intra modes are used.

The evaluation results are summarized in Table 3 in terms of Bjontegaard Delta (Δ) Rate and PSNR [24]. PSNR is calculated as the average of luma (2/3) and chroma components (2×1/6). For the slowest configuration (800 cycles per macroblock), the compression efficiency of the inter-frame coding is lower by 4.14 % (−0.15 dB) compared to the JM.17.0 software. The losses are mainly caused by the inaccuracy of the coarse estimation stage. On the other hand, intra frames are coded with negligible losses. When the temporal prediction fails (e.g., Riverbed), the losses are slight due to the strong impact of intra-coded macroblocks. The losses introduced in the second configuration are mainly caused by the skipping of the intra 4×4 prediction. The impact of the second RP is negligible (even negative for Bluesky and Station2). The fastest configuration (200 cycles per macroblock) introduces additional losses (3.23 % and 0.11 dB compared to the second

Table 2 Mode configurations for different numbers of clock cycles per Macroblock

Number of clock cycles available per macroblock	200	400	800
SP/MV number	36	60	2×60
RP number	1	1	2
Intra 16×16 & chroma	DC, vertical, and horizontal	All	All
Intra 8×8	DC, vertical, and horizontal	On	On
Intra 4×4	Off	Off	On

Table 3 RD performance of the ME system for different numbers of clock cycles per Macroblock vs. JM.17.0

Sequence	Number of clock cycles per macroblock					
	800		400		200	
	ΔRate [%]	ΔPSNR [dB]	ΔRate [%]	ΔPSNR [dB]	ΔRate [%]	ΔPSNR [dB]
Bluesky	7.02	−0.29	9.41	−0.33	10.66	−0.43
Pedestrian area	2.94	−0.11	5.70	−0.17	11.12	−0.34
Riverbed	0.42	−0.02	2.00	−0.07	8.40	−0.32
Station2	2.78	−0.12	4.93	−0.14	5.49	−0.17
Sunflower	6.52	−0.21	7.69	−0.26	7.39	−0.24
Tractor	5.13	−0.18	5.83	−0.20	8.33	−0.28
Average	4.14	−0.15	5.79	−0.19	8.56	−0.30

configuration). They are mainly caused by the limitation of the intra prediction to three 16×16 modes and three 8×8 modes. This limitation is responsible for the quality drop of 0.08 dB (2.9 % in rate), on average. Additional evaluations for the fastest configuration using all intra modes are performed. Compared to the fastest configuration, the compression efficiency is decreased by 1.32 % (0.05 dB), on average. The evaluations show that the impact of the decreased number of SPs/MVs on the compression efficiency is much smaller than the exclusion of intra modes.

4.3 Comparison

The comparison with other architectures described in the literature is presented in Table 4. All architectures support variable-size blocks and hierarchical search, and they are synthesized with either 0.18 or 0.13 μm technology. Their clock frequencies are in the range from 100 to 200 MHz. However, the optimizations introduced to the new version of the proposed architecture increase clock frequencies of some modules to 400 MHz. The optimizations allow the highest throughput and the support for 2160p@30fps. The new and old versions of the proposed architecture implement the combination of the hierarchical search and the adaptive Multi-Path Search allowing more compression-efficient coding when considering 1080p videos. The proposed architecture has also several other advantages over referenced designs. Firstly, the architecture enables scalable and adaptive computations with the ability to apply different search strategies on the fine level. Secondly, the design is suitable for the RD analysis of a number of MVs and partition modes since the fractional-accuracy motion estimation is performed at the same macroblock stage as the RD-based mode decision (+0.5 dB compared to SA(T)D-based mode decision). Thirdly, actual MV predictors are used to estimate costs of various MVs at the fine level, whereas the mode of the left macroblock is not available in other designs due to the macroblock-oriented pipeline (neglected quality losses). Fourthly, the dedicated check of the skip mode (conditioned by the actual MV predictor) improves significantly the compression efficiency compared to other designs (+1.5 dB). Fifthly, the compensation for intra and chroma (4:2:2 and 4:2:0 formats) modes are supported. In the referenced designs, these operations are performed outside the motion estimation and compensation system.

The both versions of the proposed architecture require the lowest gate count compared to other designs. The resource

Table 4 Comparison of motion estimation architectures

Design	Ver. 1 [18]	Ver. 2	Warrington [12]	Lin [13]	Liu [14]	Yin [15]	Zhang [16]
Technology [μm]	TSMC 0.13	TSMC 0.13	TSMC 0.18	TSMC 0.13	TSMC 0.18	SMIC 0.18	0.18
Clock freq. [MHz]	200	200 & 400	155	129	200	200	117
Gate count [k]	143	169	556	283	689	260	238
Memory [kB]	160.44/265.44	104.44	n.a.	8.54	80.8	176.5	n.a.
Algorithm	Hierarchical (coarse FS+MPS)	Hierarchical (coarse FS+MPS)	Hierarchical (2 levels)	Hierarchical (3 levels)	Hierarchical (2 levels)	Hierarchical (3 levels)	DS + Cross Search
Video format	1080p/720p@30fps	2160p@30/15fps	1080p@30fps	1080p@60fps	1080p@30fps	1080p@30fps	720p@30fps
Search range	H: −64, 63 V: −64, 63	H: −64, 63 V: −64, 63	H: −64, 63 V: −64, 63	H: −128, 127 V: −128, 127	H: −96, 95 V: −64, 63	H: −128, 127 V: −96, 95	H: −96, 95 V: −96, 95
Block sizes	16×16–8×8	16×16–8×8	16×16–4×4	16×16–4×4	16×16–8×8	16×16–8×8	16×16–8×8
MV accuracy	¼-pixel	¼-pixel	1-pixel	¼-pixel	¼-pixel	¼-pixel	¼-pixel
Number of RPs	1/2	1/2	5	1	1	2	1

consumption makes the proposed architecture more suitable for FPGA than other designs since FPGA devices usually embed much more memories with respect to logic resources. However, the architecture can also be implemented in ASIC at the cost of more complex placement and routing. The memory cost of the two versions of the proposed architecture is higher compared to two other designs [12, 13]. The architecture having the smallest memory cost [13] does not take into account the cost of buffers for current/original macroblock. Moreover, access to the external DDR memory is highly inefficient on account of short bursts and subsampling, and the design is limited to one reference picture. In the new version of the proposed architecture, 40 kB are needed to store two interpolated luma search areas, not-interpolated chroma samples, original samples, and intra predictions. The capacity of 64 kB is indispensible to store 16 coarse-level macroblock lines if the support for 2160p videos is required. The capacity can be reduced to 32 kB for 1080p resolutions.

5 Conclusion

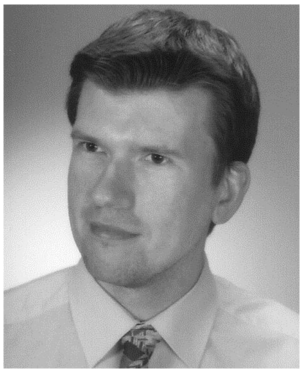
The architecture supporting the adaptive computationally-scalable ME is optimized. The throughput of the coarse estimator, the memory access controller, the interpolator, and the write stage in the compensator is doubled by increasing the clock frequency and the parallel processing. Since the chroma interpolation follows the search process, the minimal number of clock cycles assigned to one macroblock is additionally decreased by half. Moreover, the memory size is reduced from 160.44 to 104.44 kB. The new version of the ME system consumes 5.8 and 5 % more logic for ASIC and FPGA, respectively. The implementation in the medium-cost FPGA (Arria II GX) allows 1080p@60fps. The ASIC implementation can support 2160p@30fps. The results prove that the optimized architecture significantly improves the hardware efficiency. The proposed design techniques can be applied to architectures developed for H.265/HEVC [25].

References

1. Koga, T., Iinuma, K., Hirano, A., Iijima, Y., & Ishiguro, T. (1981). Motion compensated interframe coding for video conferencing. In *Proc. Nat. Telecom. Conf.* (pp. C9.6.1–C9.6.5).
2. Zhu, S., & Ma, K. K. (1997). A new diamond search algorithm for fast block matching motion estimation. In *Int. Conf. on Information, Communications and Signal Processing (ICICS '97)* (pp. 292–296).
3. Lam, C. W., Po, L. M., & Cheung, C. H. (2004). A novel kite-cross-diamond search algorithm for fast block matching motion estimation. In *IEEE Int. Symp. on Circuits and Systems (ISCAS '04)*, 3, 729–732.
4. Liu, L. K., & Feig, E. (1996). A block-based gradient descent search algorithm for block motion estimation in video coding. *IEEE Transactions on Circuits and Systems for Video Technology, 6*(4), 419–422.
5. Chen, C.-Y., Huang, Y.-W., Lee, C.-L., & Chen, L.-G. (2006). One-pass computation-aware motion estimation with adaptive search strategy. *IEEE Transactions on Multimedia, 8*(8), 698–706.
6. Jakubowski, M., & Pastuszak, G. (2009). An adaptive computation-aware algorithm for multi-frame variable block-size motion estimation in H.264/AVC. *International Conference on Signal Processing and Multimedia Applications (SIGMAP '09)* (pp. 122–125).
7. Zhou, D., Zhou, J., He, G., & Goto, S. (2014). A 1.59 Gpixel/s motion estimation processor with −211 to +211 search range for UHDTV video encoder. *IEEE Journal of Solid-State Circuits, 49*(4), 827–837.
8. Ruiz, G. A., & Michell, J. A. (2011). An efficient VLSI processor chip for variable block size integer motion estimation in H.264/AVC. *Signal Processing-Image Communication, 26*(6), 289–303.
9. Hsieh, J.-H., & Chang, T.-S. (2013). Algorithm and architecture design of bandwidth-oriented motion estimation for real-time mobile video applications. *IEEE Transactions on Very Large Scale Integration (VLSI) Systems, 21*(1), 33–42.
10. Ding, L.-F., Chen, W.-Y., Tsung, P.-K., Chuang, T.-D., Hsiao, P.-H., Chen, Y.-H., Chiu, H.-K., Chien, S.-Y., & Chen, L.-G. (2010). A 212 MPixels/s 4096×2160 p multiview video encoder chip for 3D/quad full HDTV applications. *IEEE Journal of Solid State Circuit (JSSC), 45*(1), 46–58.
11. Byun, J., Jung, Y., & Kim, J. (2013). Design of integer motion estimator of HEVC for asymmetric motion-partitioning mode and 4K-UHD. *Electronics Letters, 49*(18), 1142–1143.
12. Warrington, S., Sudharsanan, S., & Chan, W.-Y. (2007). Architecture for multiple reference frame variable block size motion estimation. *IEEE International Symposium on Circuits and Systems*, 2007. ISCAS 2007, 2894–2897.
13. Lin, Y.-K., Lin, C.-C., Kuo, T.-Y., & Chang, T.-S. (2008). A hardware-efficient H.264/AVC motion-estimation design for high-definition video. *IEEE Transactions on Circuits and Systems I, 55*(6), 1526–1535.
14. Liu, Z., Song, Y., Shao, M., Li, S., Li, L., Ishiwata, S., Nakagawa, M., Goto, S., & Ikenaga, T. (2009). HDTV1080p H.264/AVC encoder chip design and performance analysis. *IEEE Journal of Solid-State Circuits, 44*(2), 594–608.
15. Yin, H., Jia, H., Qi, H., Ji, X., Xie, X., & Gao, W. (2010). A hardware-efficient multi-resolution block matching algorithm and its VLSI architecture for high definition MPEG-like video encoders. *IEEE Transactions on Circuits and Systems for Video Technology, 20*(9), 1242–1254.
16. Zhang, L., & Gao, W. (2007). Reusable architecture and complexity-controllable algorithm for the integer/fractional motion estimation of H.264. *IEEE Transactions on Consumer Electronics, 53*(2), 749–756.
17. Porto, M., Bampi, S., Altermann, J., Costa, E., & Agostini, L. (2011). A real time and power efficient HDTV motion estimation architecture using adder-compressor. IEEE Second Latin American Symposium on Circuits and Systems pp. 1–4.
18. Rhee, C. E., Jung, J.-S., & Lee, H.-J. (2010). A real-time H.264/AVC encoder with complexity-aware time allocation. *IEEE Transactions on Circuits and Systems for Video Technology, 20*(12), 1848–1862.
19. Pastuszak, G., & Jakubowski, M. (2013). Adaptive computationally-scalable motion estimation for the hardware H.264/AVC encoder. *IEEE Transactions on Circuits and Systems for Video Technology, 23*(5), 802–812.
20. ITU-T Recommendation H.264 and ISO/IEC 14496-10 MPEG-4 Part 10, Advanced Video Coding (AVC) (2005).

21. Roszkowski, M., & Pastuszak, G. (2014). Intra prediction for the hardware H.264/AVC high profile encoder. *Journal of Signal Processing Systems, 76*(1), 11–17.
22. Pastuszak, G. (2015). Architecture Design of the H.264/AVC Encoder based on Rate-Distortion Optimization. *IEEE Transactions on Circuits and Systems for Video Technology*, doi: 10.1109/TCSVT.2015.2402911.
23. Xiph.org: Test media (2011). Available on-line at <http://media.xiph.org/video/derf/>.
24. Bjontegaard, G., Calculation of average PSNR differences between RD-curves. ITU-T VCEG-M33, VCEG 13th Meeting.
25. ITU-T Recommendation H.265 and ISO/IEC 23008-2 MPEG-H Part 2, High Efficiency Video Coding (HEVC) (2013).

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.



Mariusz Jakubowski received the M.S. degree in electronics and the Ph.D. degree in computer science from the Warsaw University of Technology, Warsaw, Poland, in 2001 and 2012, respectively. From 2002 to 2008, he was a research assistant at Telecommunications Research Institute (currently Bumar Electronics), Warsaw, Poland. From 2006 to 2012 he was a Research Assistant at the Institute of Radioelectronics Warsaw University of Technology. Currently, he is with the Faculty of Engineering, Academy of Business and Finance VISTULA, Warsaw, Poland. His research interests include video coding technology, motion estimation and associated hardware architectures.