Journal of Systems Architecture 59 (2013) 870–877

Contents lists available at SciVerse ScienceDirect

Journal of Systems Architecture

journal homepage: www.elsevier.com/locate/sysarc

Efficient smart-camera accelerator: A configurable motion estimator dedicated to video codec

Wajdi Elhamzi [a,b], Julien Dubois [b,*], Johel Miteran [b], Mohamed Atri [a], Barthelemy Heyrman [b], Dominique Ginhac [b]

[a] University of Monastir, Laboratory of EμE, Faculty of Sciences of Monastir, Tunisia
[b] University of Burgundy, Laboratory Le2i, UMR CNRS 6306, 21000 Dijon, France

ARTICLE INFO

Article history:
Available online 16 May 2013

Keywords:
Configurable motion estimation
Smart camera accelerator
Fractional Motion Estimation
FPGA

ABSTRACT

Smart cameras are used in a large range of applications. Usually the smart cameras transmit the video or/and extracted information from the video scene, frequently on compressed format to fit with the application requirements. An efficient hardware accelerator that can be adapted and provide the required coding performances according to the events detected in the video, the available network bandwidth or user requirements, is therefore a key element for smart camera solutions. We propose in this paper to focus on a key part of the compression system: motion estimation. We have developed a flexible hardware implementation of the motion estimator based on FPGA component, fully compatible with H.264, which enables the integer motion search, the fractional search and variable block size to be selected and adjusted. The main contributions of this paper are the definition of an architecture allowing flexibility and some new hardware optimizations of the architecture of the motion estimation allowing the improvement of the performances (computing time or hardware resources) compared to the state of the art. The paper describes the design and proposes a comparison with state-of-art architectures. The obtained FPGA based architecture can process integer motion estimation on 720×576 video streams at 67 fps using full search strategy, and sub-pel refinement up to 650 KMacroblocks/s.

1. Introduction

The smart camera is a label which refers to cameras that have not only the availability to grab images but also to process them in real-time. Hence, some hardware accelerators based on a combination of FPGA, specific circuits (ASIC) and processors are developed and embedded inside of the camera to reach real-time [1,2]. Generally, the processing enables useful information to be extracted from the image and therefore requires a low output bandwidth [3]. Nevertheless, some smart cameras are still used to transmit the original video stream (or a part of it) as well as the results obtained from the embedded processing. For this configuration, the original video stream is frequently compressed to reduce the required data-bandwidth. The compression is therefore embedded in the smart camera using a dedicated accelerator. The adjustment of coding performance that we propose in this paper is different than state-of-art approaches. The recent MPEG Scalable Video coding standard (MPEG SVC) [4] aims to encode a high-quality video bit-stream which contains a subset of bit-streams. Parts of this subset are dropped to match with the network capacities used to transfer the video. At reception level, the terminal received a video stream adjusted to its features (i.e. resolution, frame rate). Nevertheless, whatever the selected configuration, a high-quality bit-stream is encoded which represents a high processing charge. Our approach is to adjust the trade-off between the video coding performances and processing time at the codec level. At this level, some standards (i.e. H.264) offer several configurations (named profiles) and parameters which enable the coding performances to be adjusted. The flexibility is easily obtained with the software implementation however it is still a challenge to design a hardware accelerator which supports several profiles and parameter modifications.

For instance, Del Bue et al. [5] propose a software implementation based on a DSP component which enables the compression to be adjusted according to the information extracted in real-time from the video scene. Del Bue et al. propose a smart camera which enables several regions of interest to be detected in the scene and the coding performances to be adjusted for each of them. The system task is to transmit to its base station high quality foreground data, while trading-off the quality of the background data. Using a MPEG-4 Simple profile codec, different texture quantifications are applied on the foreground data and the background data. The

* Corresponding author.
E-mail address: jdubois@u-bourgogne.fr (J. Dubois).

1383-7621/$ - see front matter © 2013 Elsevier B.V. All rights reserved.
http://dx.doi.org/10.1016/j.sysarc.2013.05.005

quantification control is efficient for achieving the video coding adjustment, meanwhile the processing time is fixed. We propose a similar approach which focuses on the main part of the coding process (for almost all the standard): the motion estimation (ME) [6]. The motion estimation is well known to be the most computation-intensive stage of video coding process. Any improvement on this stage has therefore impact on the whole video codec's performances. Focusing on the motion estimation adaptation enables adjustments to be proposed on processing time as well as on the bit-rate and the PSNR. The processing time tuning represents a new contribution compared to Del Bue's approach. This approach enables the best trade-off between coding performances and the processing time to be defined by the user. Based on our experience on H.264 standard, we propose to adjust the motion estimation performances using three key features: (i) the format of input data (i.e. the size of the blocks to match), (ii) the integer search method, (iii) the optional fractional search. These adjustments allows the user to fit the application's constraints (image spatial resolution, frame-rate, bit-rate, PSNR).

The motion hardware accelerators presented in the literature do not consider the three key features simultaneously. The integer search stage is compulsory, contrary to the Variable Block Size Motion Estimation (VBSME) and the Fractional Motion Estimation (FME) refinement. Both these optional phases enable PSNR to be significantly increased as discussed in Section 2. Thus, the proposed architecture has been designed to support in particular these two optional phases. Nevertheless the optional phases, in particular FME, increase significantly the processing time. Indeed, the FME represents 45% of the inter-prediction processing time. Hence, the proposed architecture has been designed to reach high-performances on the FME phase. The proposed architecture enables higher throughput in term of Macro-Blocks per second to be reached compared to state of the art architecture's performances [7,8].

The selection of the integer search method enables the application requirements (bit-rate, etc.) to be reached as well as the processing time. However, the hardware accelerators focusing on the integer search [9,10], propose efficient, but fixed architectures which support either full-search or one reduced search strategy. Consequently for these solutions, the integer search method cannot be modified. Our contribution is to propose an architecture which overcomes this limitation, allowing the user to select the integer search strategy.

It is still nowadays a challenge to define such an efficient hardware accelerator which supports such flexibility as well as high coding performances. Moreover, this accelerator should not require the whole processing resources of the system as other image processing must be embedded inside of the smart camera. Therefore, we propose a motion estimation accelerator, fully compatible with H.264, which supports several configurations of the three key features previously mentioned. The main contributions of this paper are the definition of an architecture allowing flexibility and some new hardware optimizations of the architecture of the motion estimation allowing to improve the performance (computing time or hardware resources) compared to the state of the art. The paper is organized as follows. In Section 2, we analyze how the search strategy, the optional use of Fractional Motion Estimation (FME) and the Variable Block Size Motion Estimation (VBSME) allow the coding performances to be adjusted. In Section 3, the motion estimator architecture and the associated hardware implementation, based on a FPGA component, are presented. We describe the two main units of the architecture: a generic Integer Motion Estimation architecture which supports user-friendly search strategy selection and the FME architecture. Finally, the conclusion and future work are discussed in Section 4.

2. Key features to adjust motion estimation performances

We propose to focus on features of the motion estimation which represents the key stage of many of the standards (i.e. the powerful H.264) as well as the most expensive task in term of processing time.

The main idea in using the ME stage is to predict the next frame from the previous one, and then to code the prediction error. As the motion is not generally homogeneous on the whole frame, and also in order to reduce the processing complexity, the image is split into macro-blocks. The estimation is then operated on each macro-block which is searched into the reference frame using usually a Block Matching Algorithm (BMA). The matching criterion is usually done by the Sum of Absolute Differences (SAD). The most accurate vector corresponds to the position which causes SAD to be minimum. At this stage, the optimum corresponds to an integer position.

Many search algorithms have been developed to propose an alternative to the exhaustive Full Search approach (FS) where each position of the macro-block in the search window is considered. These optimized approaches aim to converge to the best matching motion vector without considering all possible positions. This number of points per block to be checked (NSP) is then reduced, as the algorithm uses predefined search patterns or previously predicted motion vectors to guide the process. Using these algorithms impacts the global coding performances (i.e. bit-rate, image quality and processing time). The improvement of the bit-rate or the image quality is achieved by finding the best possible motion vectors. Meanwhile, reducing the total search time is achieved by selecting the proper fast motion estimation method. Nevertheless, the bit-rate and the image quality can be decreased compared to the FS approach. For all these algorithms, several search phases are required to converge to the most accurate vector. For instance, square-shaped or hexagon-shaped or diamond-shaped search patterns with different sizes are employed in several fast motion algorithms such as, Three-Step-Search (TSS) [11], the four step search (4SS) [12], the hexagon-based search (HEXBS) [13], the Diamond Search (DS) [14] and the block-based gradient descent search (BBGDS) [15] algorithms. The aforementioned fast search algorithms are evaluated in [16] by considering the output PSNR and the processing time. Table 1 summarizes the results of this study. The study is done considering several videos with different resolution or motion speeds. Performances depend on the video content. Globally, all the Reduced Search algorithms (RS) achieve a significant speed-up compared to FS while maintaining high PSNR as presented in the Table 1. The user can select a reduced search strategy to decrease the processing time nevertheless the PSNR may decrease for high-motion speed or high texture variation. The largest decrease obtained in this study is 1.13 dB. The results are confirmed by other studies. For instance, the enhanced efficient DS algorithm, named Modified Diamond Search (MDS), is proposed and compared with other fast approaches and FS method in [17]. This algorithm achieves significant speed-up compared to FS. Indeed, on average the processing time decreases by 99% with a negligible degradation in both PSNR and bit-rate except for high-speed motion and/or high texture variation where PSNR can decrease by 0.6 dB.

Table 1
The impacts of the search strategy on the PSNR and the processing time.

$PSNR_{FS} - PSNR_{RS}$ Average, Min, Max	Processing $Time_{FS}$/ Processing $Time_{RS}$ Average, Min, Max
0.5 dB, 0.02 dB, 1.13 dB	98.08, 53.69, 214.75

The selection of the integer search algorithm can therefore enable the coding performances, as well as the processing time, to be adjusted by the user to match with application's constraints.

The second major possible adjustment concerns the type of precision chosen for position estimation. Indeed, the motion of blocks usually does not match exactly in the integer positions. So, to find best matches, fractional position accuracy can be used. If the best motion vector is a fractional position, an interpolation is needed to predict the current block. According to [18,7], Fractional Motion Estimation (FME) upgrades on average the rate distortion efficiency by +4 dB in PSNR and requests 45% of the inter-prediction processing time as shown in Table 2. In [8,18], authors evaluated FME in H.264 using several sequences and have shown that using half or quarter-pel increases image quality.

Another refinement included in recent standards allows the improvement of the estimation performance: the VBSME [9]. This method is based on the BMA, combined with a dynamic selection of the blocks size. The VBSME is carried out in both IME and FME phases. In H.264, VP8 and other video codec, a 16×16 sized macro-block can be further partitioned into 16×8, 8×16, 8×8, 8×4, 4×8 and 4×4 sub-blocks. When all sub-blocks are in uniform motion, all sub-block motion vectors will be the same as the motion vector for the entire macro-block. Nevertheless, when sub-blocks partitions are moving in different directions, sub-block motion vectors can differ significantly from each other and from the motion vector of the macro-block. Consequently the ME unit must be able to generate a separate motion vector for each of the sub-blocks. The advantages of a large block size are (i) simplicity and (ii) the limited number of vectors that must be encoded and transmitted. However, in areas of complex spatial structures and motion, better performances can be achieved with smaller block sizes.

Since the image quality, bit-rate and global codec performances depend on the application and the video content, it is useful to propose a flexible hardware architecture allowing the user to chose the algorithm embedded in the smart camera, and eventually to adapt dynamically the algorithm depending on the events detected in the scene.

3. Configurable architecture for motion estimation

3.1. Overview of the proposed architecture

The presented architecture aims to propose a flexible solution to adjust the video stream transferred by the smart camera. The accelerator is able to support several search strategies at IME stage and different configurations for FME stage. The Variable Size Block is available for each of these two stages. The global architecture is depicted in Fig. 1.

We propose in this paper an improved version of the architecture which basic principles have been presented in [19]. We have modified the global architecture and improved significantly the performances of the FME to reach high-level compression rate that fit with smart camera applications. For IME and FME stages, the best matching is performed using the 16×16-pixels reference macro-block and a region extracted from the corresponding search window. Processing one matching in one cycle is not, from our point of view, a realistic solution for implementation reasons (this requires hardware resources and data-bandwidth). A trade-off between the hardware resources and the processing time is obtained by performing, in one cycle, the comparison between a macro-block row with the corresponding row extracted in the search window. Therefore one key feature of our architecture is the embedded cache memories design which provides one row of macro-block and one row of the search window at each system's clock cycle.

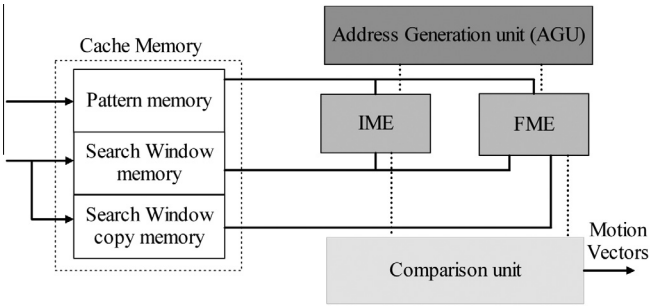


Fig. 1. The top-level view of the proposed motion estimation architecture.

After the best integer motion vector is estimated, the Fractional Motion Estimation accuracy can start. The half-pel refinements of the surrounding eight half-search positions are computed, and then the quarter-pel refinements of eight quarter-search positions surrounding the best half-search position are computed. The list of matching positions to be considered is performed by the Address Generator Unit (AGU) and transferred to the IME and FME modules. The address generation is regular for FME. The list of addresses is processed using the results obtained during the integer estimation. Eight addresses are therefore systematically determined for each sub-block. For IME the address generation depends of the selected search algorithm. Using a full search strategy, the motion detection process is completely regular. Only one generation phase is required, as all possible positions of the pattern in the search window are scanned contrary to fast search approaches. All the fast search strategies are intended to converge gradually in several phases to the right motion vector. For each phase, the next set of positions is defined using the results of the previous list of matching. For all configurations, the address generation can be described with two simple schedulers. The one in charge of the FME address generation uses a simple and fixed strategy, as described previously. The scheduler in charge of the IME phase can be modified by the user according to the selected strategy. For IME, two fixed size 16-pixels rows are respectively extracted from the reference macro-block and from the search window. The IME architecture presented in Section 3.2 supports VBS and all possible sub-blocks are processed in parallel. For FME, an interpolation phase is required, using a set of six-tap filters and bilinear filters respectively for half-pel and quarter-pel refinements. Therefore, the region to be extracted is slightly larger than the block width. The pixel number extracted also depends on the selected mode (16×16, 16×8, 8×8, 8×4, 4×4, etc.). For block's width equal to 16 pixels, 8 pixels and 4 pixels respectively, 22, 14 and 10 pixels should be extracted. The sub-block matching is generally estimated sequentially. Yang et al. have proposed in [7] an architecture which enables two sub-blocks (4×4 or 4×8) to be processed in parallel. This principle has been used for our architecture, therefore the cache memories structure has been modified compared to [19]: a copy of the search window has been introduced to enable two simultaneous accesses to two independent regions of this window. As discussed in Section 3.3, this data redundancy associated to a modification of the interpolation unit enables two sub-blocks (4×4 or 4×8 or 8×4 or 8×8 or 8×16) to be processed in parallel which allows a significant performance improvement compared to state-of-art to be obtained.

Table 2
The impacts of the FME on the PSNR and the processing time.

$PSNR_{FME} - PSNR_{FS}$ Average, Min, Max	Processing $Time_{FME}$/ (Processing $Time_{FS}$ + Processing $Time_{FME}$)
4 dB, 2 dB, 6 dB	0.45

3.2. Integer Motion Estimator supporting variable block size

The architecture of the Processing Unit is a key point of the integer motion estimation, in terms of hardware resources and processing time. Several architectures have been proposed in the literature, some implementing Fixed Block Size Motion Estimation (FBSME) based on the FS algorithm, and some implementing VBSME, as the Propagate Partial SAD [9,20], the SAD Tree [21], and the Parallel Sub-Tree [22]. Due to data dependency of full search motion estimation, 1D and 2D systolic arrays are generally used for efficient implementation of VBSME. One of the first 1D-systolic PEs-array implementations of VBSME was presented by Yap and McCanny [9], and later improved upon by Song et al. [20] and Fatemi et al. [23]. Two dimension array architectures have also been proposed for high-end application domains, such as HDTV [10]. All these structures support only FS strategy. Other related works support DS strategy, but the originality of the unified PU architecture that we propose is to support several strategies as well as VBSME.

The IME phase is highly regular therefore the proposed architecture is based on the Propagate Partial SAD architecture used in [9,20]. Four kinds of operators are therefore required: absolute difference, adder, accumulator and comparators. The 16 differences are added with a six-stage pipelined structure. The 16 accumulators, which are included in this structure, enable all 40 sub-blocks defined in VBSME to be processed. The described structure enables a matching to be processed sequentially, row-by-row, in 16 cycles. This architecture is fixed for any search strategy without any processing time overhead. Indeed, the cache memory which enables the macro-block and the search window to be stored, has been designed to provide one macro-block row and one search window row (selected according the considered address) at each clock cycle without any latency between any random address. Consequently two random matching can be performed without any latency. Therefore there is not a penalty in term of processing time between two matching using reduced search instead of FS. The only constraint is having to include an embedded cache memory which is present in most of the motion estimation accelerators. We have implemented two search strategies to provide system flexibility: Full Search (FS) and Diamond Search (DS). The implementation has been done on a Virtex6 FPGA target (6vlx240tff784-3). The proposed architecture can be considered as a low-cost implementation of a motion estimator. Table 3 shows the hardware resources requested for our FPGA based implementation and a comparison between our IME architecture and previously published ASIC VBSME processors [9,20,23].

All these selected architectures enable a matching to be processed row-by-row. Therefore the number of PEs, as well as the data bandwidth, is reduced compared to the resources required by the solutions which process a matching in only one step. For all these lost-cost architectures, the 41 possible motion vectors are carried out by a common pipelined structure of 16 PEs which enables a matching to be processed in 16 cycles. A 16×16 search range can therefore be processed in 4096 cycles. The architecture described in [23] is original as a 32×32 searching range can be performed. High-clock frequency can be reached due to a pipelined structure and a pixel truncation technique. Nevertheless the process still requires 26624 cycles due to the large searching range and the higher number of sub-blocks to be performed. The originality of our approach is to support different search strategies. Therefore the main challenge was to provide the input data without latency even for two matchings with non-consecutive accesses in the cache memory. Our implementation frequency reaches 438 MHz that overcomes all mentioned designs, as we use more recent technology (40 nm). Hence a matching is processed within 36.5 ns. Using a 16×16 search window, 720×576 video streams can be processed at 67 fps in a FS mode. Meanwhile, using a DS method and considering a realistic average range of 15–30 matchings per macro-block, a 1080 HD video stream can therefore respectively be processed from 223 down to 111 fps. The IME implementation results demonstrate that very low-cost architecture can be proposed. Moreover, it obtains real-time performances for high resolution images using fast search strategies. Meanwhile the FME is a complex task and is time consuming, therefore in order to obtain high-performances at the system level, the corresponding implementation should be optimized.

3.3. High-speed Fractional Motion Estimator

The FME stage enables sub-pel accuracy to be performed therefore the considered search region should be interpolated. As the half-pel and quarter-pel refinements are generally processed sequentially, the interpolation is done with two successive filtering operations. The half-pel and quarter-pel estimation differ mainly by the interpolation stage. The half-pel refinement requires 6-tap separable FIR filters which is more complex than the quarter-pel processing which is performed using only bilinear filters. Each half-pel value is calculated using six adjacent pixels horizontally or vertically [18,7]. Once half-pel samples are available, the pixel values at quarter-pel locations are processed with basic bilinear weighting of the values at half-pel and integer-pel positions. Fig. 2 depicts the overall block diagram of the proposed architecture of FME. It consists of two interpolation based units, eight processors units, 16 memory units and two comparator units.

In each refinement stage, eight candidates surrounding the refinement center are evaluated simultaneously using the processor units. Sixteen classes of interpolated pixels are defined in [24]. In order to optimize the matching process with the eight processors, each class is stored individually. The proposed architecture requires therefore four memory banks for half-pel processing and 12 banks for the quarter-pel refinement. Each bank is implemented with dual-port memory embedded into the FPGA component. Only two memory blocks are required per class. Therefore, the used hardware resources are still low and suitable for FPGA implementation. For instance, the 32 memory blocks represent less than 8% of Virtex 6vlx240 FPGA's memory blocks. Two comparator units

Table 3
Comparison of VBSME architectures performances. The required hardware resources of the proposed architecture are: 1168 slice registers, 1281 slice LUT and 1 BRAM.

Ref.	Yap's	Song's	Fatemi's	Proposed architecture
Year	2004	2006	2009	2012
Tech (μm)	TSMC 0.13	TSMC 0.18	TSMC 0.18	Virtex6 0.04
Searching range	16 × 16	16 × 16	32 × 32	16 × 16
Latency	4096	4096	26624	4096
Gate count	61 k	51.7 k	44 k	NA
Freq (MHz)	294	266	316	438
Video	720 × 576@45 fps	352 × 288@25 fps	352 × 288@30 fps	352 × 288@67 fps
Search strategies	FS	FS	FS	FS/DS

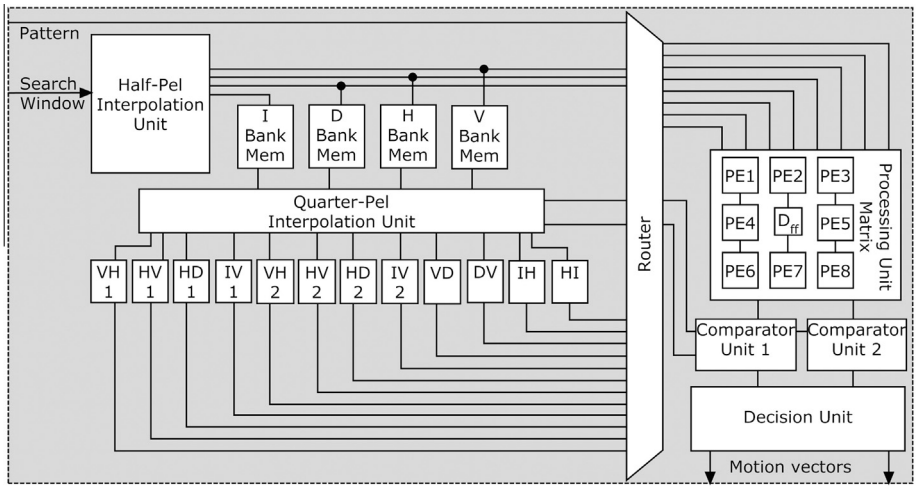


Fig. 2. Overall FME architecture.

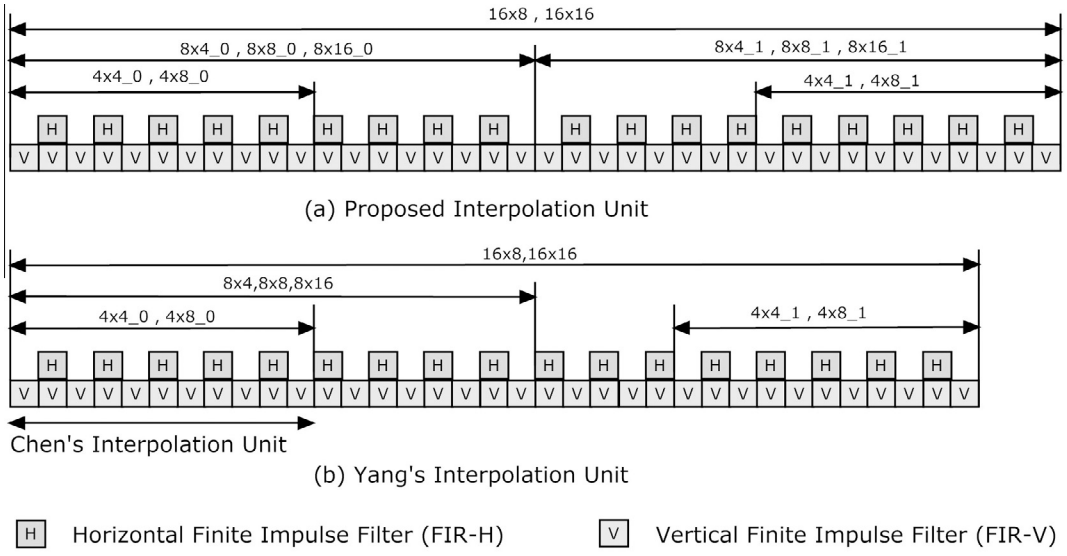


Fig. 3. Half-pel interpolation units.

are implemented to enable two sub-blocks to be estimated simultaneously.

Half-pel interpolation is the most time consuming step of FME. Many hardware architectures have been proposed to accelerate the computation of the FME algorithm [18,7,25] and have been implemented in Application Specific Integrated Circuit (ASIC) technology. In our design, we propose two versions of a modular interpolation unit. They provide a trade-off between the processing time, the redundant interpolation and hardware utilization. Parallel sets of 6-taps FIR horizontal (FIR-H) and vertical (FIR-V) process the integer input data. In [18], Chen proposes an architecture based on 4-pixels interpolation unit with nine 4×4 processing units (PUs). In each refinement stage, nine candidates around the refinement center are evaluated simultaneously. All blocks are decomposed into 4×4 sub-blocks during processing. However, decomposing large blocks into 4×4 blocks brings redundant fractional pixel interpolation. This redundancy problem appears in the overlapping area of the adjacent interpolation window. To overcome this problem, Yang [7] proposes a new architecture based on 16-pixels interpolation unit with nine 16×16 processing units which removes all the redundant pixel area. Moreover, this design adopts a short-latency 16-pixels wide interpolation to increase throughput. All block sizes are processed by 16×16 processing units. This architecture enables 4×8 and 4×4 blocks to be processed in parallel. Hence the memory bandwidth, required for reading in parallel the reference pixels of search window memory, becomes very large as mentioned previously. Ta proposes in [25] an 8-pixels interpolation unit and decomposes all larger blocks into 8×4 blocks. The redundancy is totally removed except for the 16-pixels wide block. All these architectures use a sequential approach: the half-pel is processed followed by the quarter-pel refinement. The same processing unit is used for both sub-pel refinements.

To improve the performances, we propose two new approaches. Both solutions are based on a 16-pixels wide interpolation unit. The first architecture, we proposed in [19], use a pipeline stage be-

Table 4
Comparison hardware resources.

	Chen's	Yang's	Ta's	Proposed architecture	
				Version 1	Version 2
Nbr of FIR-H	5	17	9	17	18
Nbr of FIR-V	11	35	19	35	38
Nbr of PUs	9	9	6	16	8

Table 5
Comparison of number of cycles requested for different subblocks.

Sub-block types block number		Chen's		Yang's		Ta's		Proposed architecture			
								Version 1		Version 2	
		Cycles/block	Total cycles	Cycles/block	Total cycles	Cycles/block	Total cycles	Cycles/block	Total cycles	Cycles/block	Total cycles
16 × 16	1	22 × 4	88	22 × 1	22	22 × 2	44	22 × 1	22	22	22
16 × 8	2	14 × 4	112	14 × 1	28	14 × 2	56	14 × 1	28	14	28
8 × 16	2	22 × 2	88	22 × 1	44	22 × 2	44	22 × 1	44	22 ÷ 2	22
8 × 8	4	14 × 2	112	14 × 1	56	14 × 1	56	14 × 1	56	14 ÷ 2	28
8 × 4	8	10 × 2	160	10 × 1	80	10 × 1	80	10 × 1	80	10 ÷ 2	40
4 × 8	8	14 × 1	112	14 ÷ 2	56	14 × 1	112	14 × 1	56	14 ÷ 2	28
4 × 4	16	10 × 1	160	10 ÷ 2	80	10 × 1	160	10 × 1	160	10 ÷ 2	40
Processing 1/2 1/4		Sequentially		Sequentially		Sequentially		Pipeline		Sequentially	
Latency		NA		29		NA		29		29	
Half-Pel	41	832		366		552		502		276	
Quarter-Pel	41	1664		790		1104		553		610	

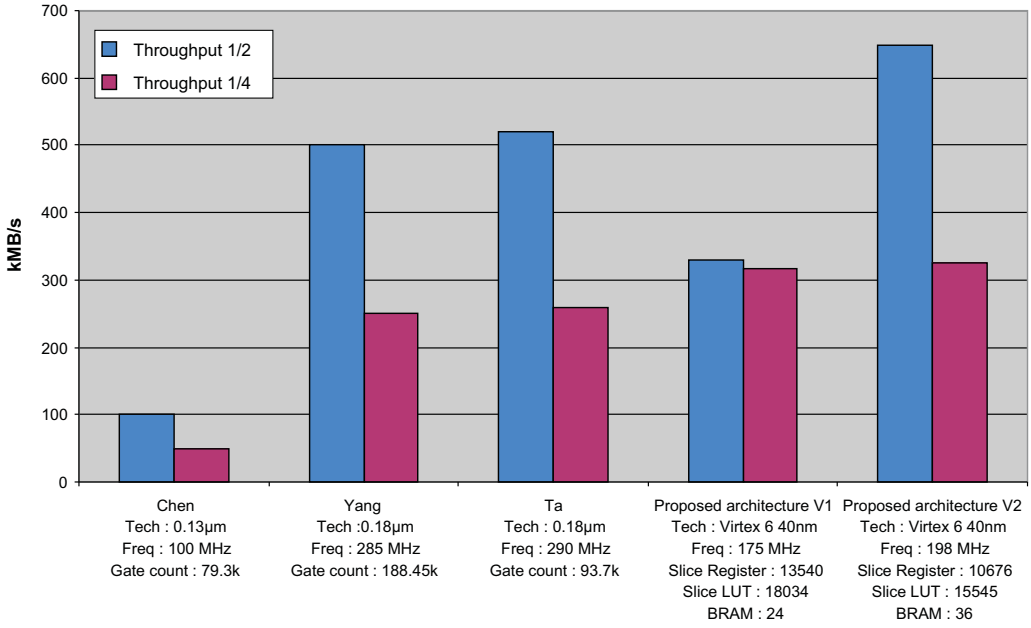


Fig. 4. Comparison of the FME architectures throughput performances.

tween half-pel and quarter-pel accelerators. Each of them requests eight processing units. High-performances are obtained for quarter-pel refinement nevertheless this pipelined architecture is obviously less performant in half-pel. Meanwhile the half-pel accuracy is sufficient for many smart camera applications. We propose a second solution based on the sub-blocks parallel processing which aims to optimize the half-pel processing, keeping high global performances. The proposed architecture processes a maximum number of two sub-blocks in parallel. This version aims to support any sub-block sizes therefore the interpolation unit must be modified. As depicted in Fig. 3, two concatenated 8-pixels interpolation units operate in parallel. This architecture consumes 28×2 FIRs (9×2 FIR-H and 19×2 FIR-V) to process 28-pixels integer input data. The number of required pixels is therefore 28 which is 14 pixels twice.

Table 4 regroups the number of horizontal, vertical FIR's and processing units required for the various half-pel interpolation implementations. Table 5 presents the number of cycles requested respectively for each kind of sub-block as well as the global performances of the different architectures. Chen's low-cost interpolation unit requires only 16 FIRs and 9 PUs. Each sub-block is processed sequentially. When block size is wider than 4×4, they are processed in several stages. Therefore the required number of cycles for sub-pel refinement is higher than other implementations. Yang's design has a 22 input integer pixel, which requires 52 FIRs and 18 PUs. It enables two 4×4 or 4×8 sub-blocks to be performed simultaneously. Half-pel and quarter-pel refinement is achieved, for all sub-blocks, respectively in 366 and 790 clock cycles. Ta's architecture represents a trade-off between complexity and performances compared to these two previous architectures. Our pipeline version, noted version 1 in Tables 4 and 5, enables high-performances to be obtained for quarter-pel processing: the motion estimation can be performed in 553 cycles for all sub-blocks that improves Yang's performances by 30%. In this version, the interpolation unit is based on Yang's. The pipelined architecture requires a double number of processors.

Compared to Yang's architecture, the second solution with the interpolation modification, noted version 2 in the two Tables, only requires two extra filters meanwhile improves the processing performances for both sub-pel refinements. The proposed architecture is able to perform simultaneously two sub-blocks either 8×16, 8×8, 8×4, 4×8 or 4×4. Hence, half-pel and quarter-pel modes are respectively performed for all 41 sub-blocks in 276 and 610 cycles. We saved respectively about 25% and 23% of processing time in comparison to Yang's architecture.

Fig. 4 shows the results of FME architecture implementation, in terms of hardware resources, system frequency and throughput. The state-of-art designs are based on similar technology (0.18 μm) and our implementation uses the 40 nm technology available on Virtex 6 FPGA (6vlx240tff784-3). We select FPGA technology to offer a high-flexibility in the implementation of other processing inside of the smart camera (such as event detection). Moreover the dynamic or partial reconfiguration available on FPGA component could represent an interesting solution when selecting the FME architecture in function of the application. Meanwhile, the comparison of the hardware resources is difficult as the architectures use different technologies, nevertheless frequency and throughput can be compared. The implementation of the version 1 and 2 enables a half-pel refinement to be processed respectively at 329 and 649 KMacroBlocks/s (KMB/s), and quarter-pel at 316 and 325 KMB/s. The implementation of the first version has slightly lower frequency than the second version. Therefore, the performances of implementation of the second version is higher for both configurations, with a significant improvement of 25% compared to state-of-art performances.

4. Conclusion

The adjustment of processing and/or communication, according to the events happening in the video scene or some environment modifications (as a network bandwidth reduction), represents a major challenge for a new generation of smart cameras. Therefore the configurable motion estimator represents a key element in a video codec which enables such flexibility to be achieved. We have implemented a motion estimator on a FPGA which is a current target for smart camera design. The proposed accelerator enables the integer search strategy to be adjusted and the optional VBSME and sub-pel refinements to be processed. Moreover, the current implementation enables high-speed performances to be reached. Hence for IME, 1080 HD video streams can be processed up to 200 fps using fast search strategy. Due to a novel FME architecture, the same video streams can be processed with half-pel and quarter-pel refinements respectively at frame rate of 41 and 81 fps (respectively around 330 and 650 KMB/s) which represents a significant improvement (25%) compared with the state-of-art. This solution can therefore represent an efficient solution for many video coding applications. Current investigations aim to consider dynamic reconfiguration to adjust on-line FME architecture.

References

[1] F. Dias Real, F. Berry, Smart Cameras: Technologies and Applications, Smart Cameras, Springer, New York, 2009. pp. 35–50.
[2] R. Mosqueron, J. Dubois, M. Mattavelli, D. Mauvilet, Smart camera based on embedded HW/SW coprocessor, EURASIP Journal on Embedded Systems (2008) 1–13.
[3] R. Mosqueron, J. Dubois, M. Paindavoine, High-speed smart camera with high resolution, EURASIP Journal on Embedded Systems (2007) 1–15.
[4] H. Schwarz, D. Marpe, T. Wiegand, Overview of the scalable video coding extension of the H.264/AVC Standard, IEEE Transactions on Circuits Systems and Video Technology 17 (9) (2007) 1103–1119.
[5] A. Del Bue, D. Comaniciu, V. Ramesh, C. Regazzoni, Smart cameras with real-time video object generation, in: Proceedings of International Conference on Image Processing (ICIP), vol. 3, 2002, pp. III-429–III-432.
[6] T. Wiegand, G.J. Sullivan, G. Bjontegaard, A. Luthra, Overview of the H.264/AVC video coding standard, IEEE Transactions on Circuits Systems and Video Technology 13 (7) (2003) 560–576.
[7] C. Yang, S. Goto, T. Ikenaga, High performance VLSI architecture of fractional motion estimation in H.264 for HDTV, in: Proc. IEEE ISCAS, Greece, 2006, pp. 2605–2608.
[8] Y.H. Chen, T.C. Chen, S.Y. Chien, Y.W. Huang, L.G. Chen, VLSI Architecture design of fractional motion estimation for H.264/AVC, Journal of Signal Processing Systems 53 (3) (2008) 335–347.
[9] Y.Ya. Swee, J.V. McCanny, A VLSI architecture for variable block size video motion estimation, IEEE Transactions on Circuits and Systems for Video Technology 51 (7) (2004) 384–389.
[10] C.Y. Chen, S.Y. Chien, Y.W. Huang, T.C. Chen, T.C. Wang, L.G. Chen, Analysis and architecture design of variable block size motion estimation for H.264/AVC, IEEE Transactions on Circuits and Systems-I: Regular Papers 53 (3) (2006) 578–593.
[11] T. Koga, K. Ilinuma, A. Hirano, Y. Iijima, T. Ishiguro, Motion compensated interframe coding for video conferencing, in: Proc. Nat. Telecom. Conf., USA, 1981, pp. G5.3.1–G5.3.5.
[12] L.M. Po, W.C. Ma, A novel four-step search algorithm for fast block motion estimation, IEEE Transactions on Circuits and Systems for Video Technology 6 (3) (1996) 313–317.
[13] C. Zhu, X. Lin, L.P. Chau, Hexagon-based search pattern for fast block motion estimation, IEEE Transactions on Circuits and Systems for Video Technology 12 (5) (2002) 349–355.
[14] S. Zhu, K.K. Ma, A new diamond search algorithm for fast block matching motion estimation, IEEE Transactions on Image Process 9 (2) (2000) 287–290.
[15] L. Liu, E. Feig, 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) (1996) 419–422.
[16] Y.G. Lee, J.B. Ra, Fast motion estimation robust to random motions based on a distance prediction, IEEE Transactions on Circuits and Systems for Video Technology 16 (7) (2006) 869–875.
[17] Y. Ismail, J. McNeelly, M. Shaaban, M. Bayoumi, Enhanced efficient diamond search algorithm for fast block motion estimation, in: Proc. IEEE ISCAS, Taipei, 2009, pp. 3198–3201.
[18] T.C. Chen, Y.W. Huang, L.G. Chen, Fully utilized and reusable architecture for fractional motion estimation of H.264/AVC, in: Proc. IEEE ICASSP, 2004, pp. 9–12.
[19] W. Elhamzi, J. Dubois, J. Miteran, M. Atri, R. Tourki, Hardware implementation of a configurable motion estimator for adjusting the video coding, in: Proceedings on Advanced Concepts for Intelligent Vision Systems (ACIVS2012), Czech Rep., 2012.
[20] Y. Song, Z. Liu, T. Ikenaga, S. Goto, A VLSI architecture for variable block size motion estimation in H.264/AVC with low cost memory organization, IEICE Transactions on Fundamentals E89 (12) (2006) 3594–3601.
[21] T.C. Chen, S.Y. Chien, Y.W. Huang, C.H. Tsai, C.Y. Chen, T.W. Chen, L.G. Chen, Analysis and architecture design of an HDTV720p 30 Frames/s H.264/AVC Encoder, IEEE Transactions on Circuits and Systems for Video Technology 16 (6) (2006) 673–688.
[22] Z. Liu, Y. Song, Z. Liu, T. Ikenaga, S. Goto, A fine-grain scalable and low memory cost variable block size motion estimation architecture for H.264/AVC, IEICE Transactions on Fundamentals E89-C (12) (2006) 1928–1936.
[23] M.R.H. Fatemi, H.F. Ates, R. Salleh, A bit-serial sum of absolute difference accelerator for variable block size motion estimation of H.264, in: Proc. Conference on Innovative in Intelligent Systems and Industrial Applications, 2009, pp. 1–4. .
[24] G.A. Ruiz, J.A. Michell, An efficient VLSI architecture of fractional motion estimation in H.264 for HDTV, Journal of Signal Processing Systems 62 (3) (2010) 443–457.
[25] N.T. Ta, J.R. Choi, High performance fractional motion estimation in H.264/AVC based on one-step algorithm and 8×4 element block processing, Signal Processing: Image Communication 26 (2011) 85–92.

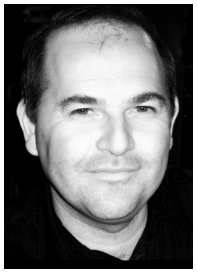
Wajdi Elhamzi is PhD student at the University of Burgundy (France) and Monastir (Tunisia) since 2009. He obtained his Master degree in electronics in 2008 from Monastir University. He is involved in FPGA based hardware implementation of image processing algorithm, and currently focus on motion estimation for real-time video coding.

Julien Dubois is associated professor at the University of Burgundy since 2003. He is a member of the Laboratory Le2i (UMR CNRS 6063). His research interests include real-time implementation, smart camera, hardware design based on data-flow modeling, motion estimation and image compression. In 2001, he received PhD in Electronics from the University Jean Monnet of Saint Etienne (France) and joined EPFL based in Lausanne (Switzerland) as a project leader to develop a co-processor, based on FPGA, for a new CMOS camera.

Johel Miteran received the PhD degree in image processing from the University of Burgundy, Dijon, France in 1994. Since 1996, he has been an assistant professor and since 2006 he has been professor at Le2i, University of Burgundy. He is now engaged in research on classification algorithms, face recognition, access control problem and real time implementation of these algorithms on software and hardware architecture.

Mohamed Atri born in 1971, received his PhD degree in Micro-electronics from the Science Faculty of Monastir in 2001. He is currently a member of the Laboratory of Electronics & Micro-electronics. His research includes Circuit and System Design, Image processing, Network Communication, IPs and SoCs.

Barthélémy Heyrman received the PhD degree in electronics and image processing from Burgundy University, France, in 2005. He is currently an Associate Professor at the University of Burgundy, France, and a member of LE2I UMR CNRS 6306 (Laboratory of Electronic, Computing and Imaging Sciences). His main research topics are system-on-chip smart camera and embedded image processing chips.



Dominique Ginhac received the PhD degree in electronics and image processing from Clermont-Ferrand University, France, in 1999. He is currently a full Professor at the University of Burgundy, France, and member of LE2I UMR CNRS 6306 (Laboratory of Electronic, Computing and Imaging Sciences). His main research topics are image acquisition and embedded image processing on CMOS VLSI chips.