Hardware Implementation of a Configurable Motion Estimator for Adjusting the Video Coding Performances

Wajdi Elhamzi, Julien Dubois, Johel Miteran,
Mohamed Atri, and Rached Tourki

University of Burgundy, Laboratory Le2i, UMR CNRS 6306,
21000 Dijon - France
University of Monastir, Laboratory of EμE
Faculty of Sciences of Monastir, Tunisia
{wajdi.elhamzi,julien.dubois,johel.miteran}@u-bourgogne.fr,
{mohamed.atri,rached.tourki}@fsm.rnu.tn

Abstract. Despite the diversity of video compression standard, the motion estimation still remains a key process which is used in most of them. Moreover, the required coding performances (bit-rate, PSNR, image spatial resolution, etc.) depend obviously of the application, the environment and the network communication. The motion estimation can therefore be adapted to fit with these performances. Meanwhile, the real time encoding is required in many applications. In order to reach this goal, we propose in this paper a hardware implementation of the motion estimator which enables the integer motion search algorithms to be modified and the fractional search and variable block size to be selected and adjusted. Hence this novel architecture, especially designed for FPGA targets, proposes high-speed processing for a configuration which supports the variable size blocks and quaterpel refinement, as described in H.264.

Keywords: Configurable motion estimation, hardware implantation, H.264, search strategy, fractional search, video coding performances.

1 Introduction

Video coding has been the subject of many research works in last decades. A large number of coding solutions have been described to fit with the diversity of the compression standards and the required coding performances, which are correlated to the constraints defined by the user or fixed by the environment (i.e. networks used for data transmission and the target receiver setup). Consequently a large number of video codec has been developed. Despite this diversity, some particular processing stages, such as motion estimation [1], are implemented in most of the proposed solutions. The motion estimation is well known to be the most computation-intensive stage of video coding process. Any improvement on this stage can therefore impact on the whole video codec's performances. From another point of view, the motion estimation configuration can be adjusted to fit the application's constraints (image spatial resolution, frame-rate, bit-rate,

© Springer-Verlag Berlin Heidelberg 2012

PSNR). For instance the motion estimation stage, described in the recent standards such as H.264 video or VP8, is highly efficient as well as highly sophisticated and complex. Meanwhile, different configurations can be defined to match with the required coding performances.

According to our analysis, the key features of motion estimation to be adjusted in current standard as H.264 are:

- The format of input data (i.e. the size of the blocks to match),
- The integer search method,
- The optional fractional search.

Software solutions can easily support any configuration nevertheless they may struggle to match the application's requirements. Indeed, for high-quality applications, the computational cost often exceeds the available resources of a standard computer. Meanwhile, to define an efficient hardware accelerator which supports such flexibility as well as high coding performances, it is still nowadays a challenge. Therefore, we propose in this paper, a motion estimation accelerator, fully compatible with H.264, which supports different configurations especially modifications on the three key features previously identified. The paper is structured as follow. In the section 2, we review the basic principles of the motion estimation. We recall and compare several integer search algorithms. Finally the impact fractional search on the coding performances (PSNR, processing-time) is presented. In section 3, we propose analysis on the integer motion estimation (IME). The common parts of these algorithms are then bringing out in order to prove that a generic structure can be proposed. The motion estimator's architecture has been designed and optimized to propose an efficient FPGA implementation in respect with the complexity and the regularity of the integer search and the fractional search. The hardware implementation results and discussion are finally presented in section 4.

2 Motion Estimation Technique

The Motion Estimation (ME) is an effective stage to detect temporal redundancies between successive frames in a video sequence. Therefore, it has become a crucial part of many video compression standards. The motion estimation aims to predict, as accurately as possible, the next frame from the current frame. The frame is split into fixed size macroblocks, currently 16x16 pixels. The prediction is processed for each macroblock of the current frame. As the motion estimation algorithm is not fixed by the video standard, many solutions have been proposed. The most popular is the Block-Matching Algorithm (BMA). In this method, the basic idea is to localize a reference block within the search area in the previous frame. A matching criterion is used to estimate similarities between any two given blocks. The applied BMA is performed using the Sum of Absolute Differences (SAD) matching criterion. This approach is known as the integer motion search as only integer displacements of the reference macroblock into the search window are performed. Some sub-pixel refinement can be processed. The Fractional Motion Estimation (FME) [2] is usually done for HalfPel and QuarterPel accuracy. The Variable Block Size Motion Estimation (VBSME) [3] is another major refinement which is included in recent standards. Indeed this approach is the use of BMA with the ability for the encoder to dynamically select the size of the blocks. The macro-block is split into smaller blocks and the estimation is performed on each sub-blocks. The VBS motion estimation can be processed at both integer and fractional levels. The motion estimation performances are therefore highly correlated not only to the selected integer search algorithm but also to the optional refinement stages VBSME and FME. Therefore the following sections discuss the impact of the integer search algorithms as well as the VBSME and the FME algorithms on the quality of image and other encoder's requirements.

2.1 Impact of the Search Strategy - Related Work

Many ME algorithms have been described in the literature. The most accurate strategy is the Full Search (FS) algorithm, which by exhaustively comparing all positions in the search window, gives the most accurate motion vector which causes SAD to be minimum. On the other hand, fast but sub-optimal algorithms compute the best matching candidate by guiding the search procedure using predefined search patterns. For instance, in Three-Step-Search (TSS) [4], the New Three Step Search (NTSS) [5] , Four Step Search (4SS) [6], the Hexagon-Based Search (HEXBS) [7] , the Diamond Search (DS) [8], the Cross-Diamond Search (CDS) [9], and the Block-Based Gradient Descent Search (BBGDS) [10] algorithms, square-shaped or hexagon-shaped or diamond-shaped search patterns with different sizes are employed. These algorithms performed well in relatively small search range and low-resolution video sequences. Improving the quality of image is achieved by finding the best possible motion vectors, which means motion vectors that will generate the smallest residual difference during the motion compensation. Reducing the total search time is achieved by selecting the proper fast motion estimation, which consists to reduce the Number of Search Points per block (NSP) to be checked. Nevertheless, the image quality can decrease compared with a FS approach. Hence, the mentioned fast search algorithms are also evaluated regarding the output PSNR. This fact is illustrated in [11]. It is noted that, for low and medium motion activity video sequences (Carphone, Foreman, and Mobile) the degradation of PSNR is slightly or negligible but the speedup is much improved. The situation is changed for the high-motion activity video sequences (Tennis Table, Football): the PSNR significantly decreases and the degradation of image quality is then visible. Otherwise, DS and BBGDS algorithms provide better PNSR performances than other fast algorithms while maintaining nearly the same search speed for the sequences Table tennis and Football. An enhanced efficient DS algorithm, named Modified Diamond Search (MDS), is proposed and compared with others fast approach and FS method in [12]. The proposed method as well as the others fast search methods DS, 4SS and N3SS achieves significant speed-up compared to FS. Hence the processing time respectively decreases of 99%, 94%, 73% and 65% for high motion video sequence (Football). A negligible degradation in both PSNR and bit-rate is observed. The low complexity of DS approach family induces that this kind of algorithms can be considered for hardware implementation. The FS algorithm is suitable to high-speed motion and/or high texture variation.

2.2 Impact of VBSME and FME - Related Work

H.264 introduces two new features to ME, the VBSME and the Sub-pixel accuracy motion estimation. The VBSME is carried out in two phases: integer motion estimation (IME) and fractional motion estimation (FME). In H.264, VP8 and others video codec, a 16x16 sized macroblock can be further partitioned into 16x8, 8x16, 8x8, 8x4, 4x8 and 4x4 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 macroblock. However, if fine grain motion exists and sub-blocks are moving in different directions, sub-block motion vectors can differ significantly from each other and from the motion vector of the macroblock. 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 performance can be achieved with the smaller block size.

Usually, the motion of blocks does not match exactly in the integer positions. So, to find best matches, fractional position accuracy is used. If the best motion vector is a fractional position, an interpolation is needed to predict the current block. According to [2] and [13], fractional motion estimation (FME) upgrades rate distortion efficiency by + 4dB in peak signal-to-noise ratio (PSNR) and requests 45% of the inter-prediction processing time. In [14], four sequences with different characteristics are used for the experiment. Foreman stands for medium motions, Soccer sequence for high motions, Mobile and Optis have complex textures. Clearly using half or quarter-pel increases image quality. The accuracy of motion compensation is in quarter-pel resolution for H.264/AVC, which can provide significantly better compression performance, especially for images with complex texture. As shown in the state of the art analysis, the three key features which are the data block sizes (defined with VBSME), the IME strategy and the optional FME have high impact on the video codec's performances. Therefore, an efficient hardware implementation of a configurable motion estimator which supports modification on these three features can be considered as a significant contribution.

3 A Flexible Motion Estimation Architecture

3.1 Overview of the Proposed Architecture

Using a full search strategy, the motion detection process is regular. All possible positions of the pattern in the search window are scanned contrary to fast search approach. All the search strategies are intended to converge to the right motion vector with a regular process and can eventually be initialized by previous information (such as the vectors previously computed). Therefore using fast or reduced search strategies enables the number of matching to be reduced, decreasing the processing time. As mentioned previously, even if the number of matching is reduced, it is possible to achieve optimal coding results using appropriate (for the video application) reduced search algorithms. Our analysis of the IME approaches, lead us to propose a structure based on the highest possible common parts between fast motion algorithm and FS. Indeed, for all algorithms, a list of matching (positions) is processed during each Integer Motion Search Phase (IMEP). As shown in Fig.1, after the reception of the list and the number of matching, each matching is processed until all positions have been considered. Finally, the best vector and, optionally, all the resulting vectors are available. Depending on the search algorithms, all the vectors may be required and transmitted to the unit in charge of the address generation. All of the resulting vectors may be used during the generation of the next eventual list of matching. The iterative phase is depicted in figure 1.

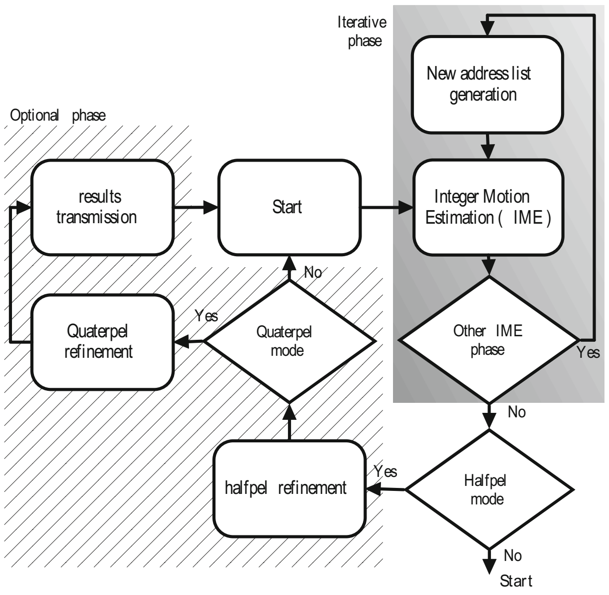


Fig. 1. Proposed algorithm

For all configurations, the address generation unit is described using a scheduler. This description could be very simple for a Full Search algorithm or more complex and irregular for more complex algorithms. The Diamond search has several phases of address generation. The full-search strategy is obviously supported in this scheme. Note that in this case, all possible addresses of the reference block in the search window are then transmitted to the operating part and therefore a unique phase is required to determine the best motion vector. In our implementation, a state machine has been used nevertheless a micro-processor (embedded in the FPGA) which represents a flexible solution as described in [15]. However, even if the state machine design may be more time consuming, higher system frequency can be achieved compared with the micro-processor based solution. Finally, depending of the user configuration, the optional fractional motion estimation (FME) can be performed with HalfPel or QuaterPel accuracy. The halfpel stage is processed systematically before the quaterpel refinement to reduce the search area and therefore memory requirement. Indeed, the subpel refinement and especially the interpolation phase are costly in terms of hardware resources [2,13,11]. Note that using a fast search for IME, induces that the FME represents the slower stage of the motion estimation. Therefore, an optimized and efficient architecture should be proposed for the FME unit. The architecture proposed is depicted Fig.2. The addresses of all matching are provided to the cache memory unit and the column extraction unit by the external address generation unit. A trade-off between the efficiency and the required resources, one matching is processed in several stages. A full parallel approach would request a large cache memory, to avoid important bottlenecks on the external memory, and extremely large processing resources for real-time implementation. We propose a trade-off between the complexity and performances with a matching done column by column.

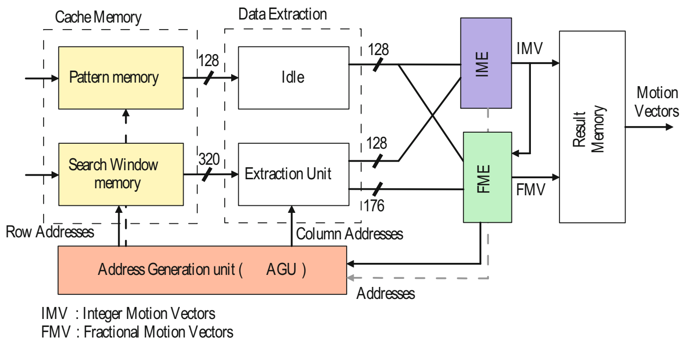


Fig. 2. The top-level view of the proposed motion estimation architecture

So as to guarantee the random access in any position of a search window the search window pixels have to be accessible to the matching engine and need to be stored in the FPGA to reduce the number of access to the external memory, so as not to exceed the available bandwidth. The cache memory permits access of two columns, one extracted from the block and the corresponding one in the search window. The architecture allows to access in one clock cycle any search window column and consecutive pattern matching evaluations do not need to be adjacent in terms of memory locations. The cache memory is obtained with dual-port memory blocks available into FPGA component. Our architecture enables an efficient random access, without any latency to be obtained. Moreover, any search-window width can be set according to the available processing. The Address Generator Unit (AGU) is charge of address generation for all configurations. The AGU allows selecting one column into the search window and one into the current block. The extraction unit enables the right amount of pixel to be selected into the search window column according to the selected matching and the motion estimation to be processed. Hence, for IME, 16 pixels (128 bits) are systematically extracted as the 16x16 macro-block and all possible sub-blocks can be processed in parallel. For FME, an interpolation phase is required. As 6 tap filters are used to interpolate the search window pixel, therefore the region to be extracted in slightly larger than the block width. The pixel number extracted also depends on the selected mode (16x16, 16x8, 8x8, 8x4, 4x4, etc.). For block's width equal to 16 pixels, 8 pixels and 4 pixels respectively, 22, 14 and 10 pixels should be extracted.

3.2 Proposed Integer Motion Estimator

The IME phase is highly regular. In our approach, each matching is operated column by column. A Processing Element (PE) is operating the comparison between a pattern pixel and the corresponding search window. The SAD matching evaluation criterion has been used in matching operator unit. It performs three operations on the input pixel stream: subtraction, absolute and accumulation. Due to regularity, 16 PEs are used in parallel as presented Fig.3.

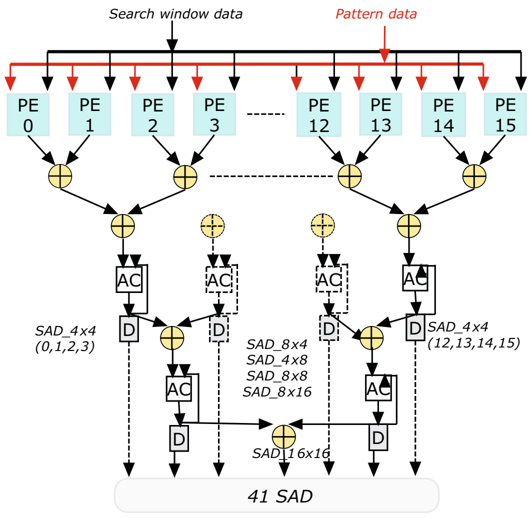


Fig. 3. IME processing Unit with VBSME supported

Moreover the architecture should support VBSME. Once again this estimation is regular as the different sub-blocks can be processed simultaneously with the 16x16 pixel macro-block. Accumulators (AC) are included inside the pipelined structure to enable the comparison to be done for all 40 sub-blocks. Such kind of efficient approach has been presented in [16]. Our version is adapted to the "two columns" process and represents therefore a low-cost approach in terms of hardware resources.

3.3 Proposed Fractional Motion Estimator

After the best integer motion vector is estimated, the fractional motion estimation accuracy can start. The HalfPel refinements of the surrounding eight half-search positions are computed, and then the QuarterPel refinements of eight quarter search positions surrounding the best half-search position are computed. In the MPEG-4/AVC H.264 standard, the QuarterPel accuracy luminance picture is interpolated with two successive filtering operations. The HalfPel refinement is more complex than the QuarterPel one and requires a 6-tap separable FIR filters with coefficients [1,-5, 20, 20, -5, 1] instead of bilinear filters. As shown in Fig.4a), each half pixel value is calculated from 6 adjacent pixels horizontally or vertically. The horizontal value h3,3 is computed from the six adjacent integer pixel samples located at horizontal direction according to the following equation:

$$h3,3 = i1,3 - 5i2,3 + 20i3,3 + 20i4,3 - 5i5,3 + i6,3. \tag{1}$$

In a similar way, the vertical half-pel value v3,3 is performed using the six adjacent pixel values located in the vertical direction as:

$$v3,3 = i3,1 - 5i3,2 + 20i3,3 + 20i3,4 - 5i3,5 + i3,6. \tag{2}$$

The diagonal half-pel value d3,3 is obtained from the six adjacent horizontal values hi or alternatively, verticals values vi,j according to:

$$d3,3 = v1,3 - 5v2,3 + 20v3,3 + 20v4,3 - 5v5,3 + v6,3. \tag{3}$$

$$d3,3 = h3,1 - 5h3,2 + 20h3,3 + 20h3,4 - 5h3,5 + h3,6. \tag{4}$$

Once half-pixel samples are available, the pixel values at QuarterPel locations are processed with basic bilinear weighting of the values at half-pel and integer-pel positions. Nevertheless, the quaterpel processing is less regular. As shown in Fig.4b), the orientation of the pixels is considered, and 12 different kinds of processing , which generate the QuaterPel positions, can be observed. Therefore we propose a novel architecture using four different memory banks for HalfPel processing and 12 banks for the QuaterPel refinement. For instance for HalfPel refinement, the original pixel named I and three kinds of interpolated pixel are stored respectively in I, H, V and D memory banks. This approach has been proposed by Ruiz [17] only for HalfPel refinement, we propose in this paper to apply it also to QuaterPel refinement. This architecture is highly efficient in term of

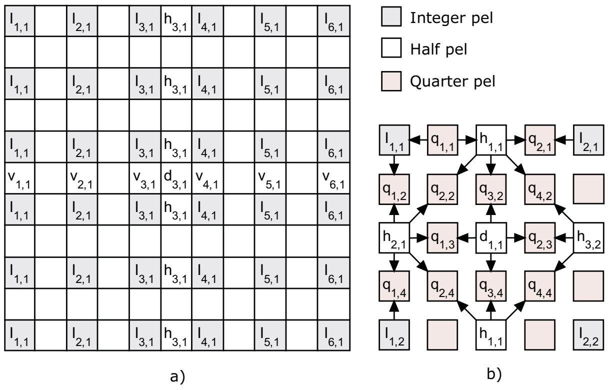


Fig. 4. a) Half-pel pixel values - (b) Quaterpel pixel values

data broadcasting therefore the processing decrease. Nevertheless the number of small cache memory is increased. Each bank is implemented with dual port memory embedded into the FPGA component. Only two memory blocks are required to store the reduced number of each class of interpolated pixels. Therefore, the used hardware resources are still low and suitable for FPGA implementation. For instance, the 32 memory blocks represent less than 8% of Virtex6 6vlx240 FPGA. The Fig.5 depicts the overall block diagram of the proposed architecture of FME. It consists of two processors used in pipeline: HalfPel and QuarterPel processors which are interpolation based units, processors units, memory unit and comparator unit. The architecture of processor and comparator unit is the same for both HalfPel and QuarterPel. In each refinement stage, eight candidates around the refinement center are evaluated simultaneously.

Our halfpel interpolation unit is based on the well-known Yang's solution [13] which processes a row 16-pixel interpolation unit. Indeed, a problem related to

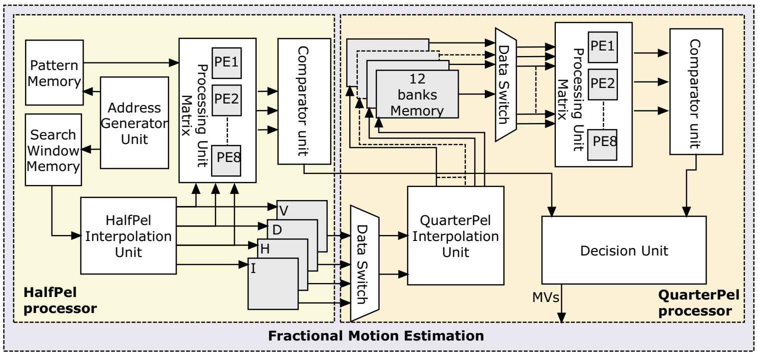


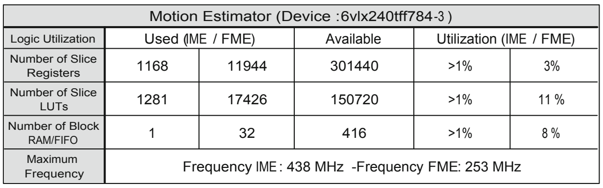
Fig. 5. Overall FME architecture

Chen's 4-pixel interpolation unit [2] is the redundant interpolating operations which appear in the overlapping area of the adjacent interpolation window. To overcome this problem, a new architecture based on 16-pixel interpolation unit with nine or eighteen 16x16 processing units is proposed by Yang's which removes all the redundancies. Our design, as Yang's, adopts a short-latency 16-pixel wide interpolator to increase throughput and eliminate redundant interpolation. Moreover, all sizes of blocks are processed by 16x16 processing units. Therefore, the hardware utilization is low when processing small size blocks (4x4 and 4x8). When 4x8 and 4x4 blocks are processed in parallel, a large memory bandwidth of search window memory is then required for reading the reference pixels in parallel. Yang's architecture enables higher processing performance to be obtained than with Chen's implementation. We used Half-pel interpolation unit proposed by Yang's. We proposed a pipeline architecture which enables Half-pel and Quarter-pel to be processed simultaneously. Consequently, the number of cycles is reduced. The processors request exactly the same scheduling in Half-pel and Quater-pel modes, therefore the performances are identical for both modes.

4 Implementation Results and Discussion

The proposed architecture can be considered as a low cost implementation of a motion estimator. The hardware ressources required for our implementation are presented in Table 1. The implementation have been done on a Virtex6 FPGA target (6vlx240tff784-3).

Table 1. Motion estimator's implementation results



Motion Estimator (Device :6vlx240tff784-3)					
Logic Utilization	Used (IME / FME)		Available	Utilization (IME / FME)	
Number of Slice Registers	1168	11944	301440	>1%	3%
Number of Slice LUTs	1281	17426	150720	>1%	11 %
Number of Block RAM/FIFO	1	32	416	>1%	8 %
Maximum Frequency	Frequency IME: 438 MHz -Frequency FME: 253 MHz				

The integer and fractional estimators are regrouped in this table. Note that only two search strategies are currently implemented: FS and DS. This flexible low-cost implementation for IME provides efficient results with a global frequency of 438 MHz. The architecture enables a matching to be processed in 16 cycles (36,5 ns) and without latency is required between two matching. With a 41x25 search window (1025 matching), 1080 HD (1920x1088) video streams can be processed at 3 fps in a FS mode. Meanwhile, using a DS method and considering a realistic average range of 15-30 matching per macro-blocks, a 1080 HD video stream can be processed between 223 and 111 fps. The same performances are obtained for HalfPel and QuaterPel mode. The Table 2, presents the number of cycle for each block size. The last row of Table 2 presents the full processing time of the 41 possible blocks in VSBME. As expected and detailed in section 3.3, the very low-cost architecture proposed by Chen is less performing than Yang's. Our architecture proposes very competitive performances and similar results to Yang's one. Comparing the two architectures, our solution reduces by two the memory size and decreases by 8 (instead of 18) the processor number for each sub-pel refinement. Our solution cannot process in parallel two 4x8 or 4x4 sub-blocks, nevertheless the pipeline structure enables Half and Quarter-Pel refinement to be processed simultaneously. Therefore, this architecture can save approximately 66% and 30% in processing time, compared with Chen's and Yang's respectively. The global time is reduced to 553 cycles .Yang's architecture has been implemented with a 0.18 μm technology. It can process 1080 HD video streams at frame rate of 30fps when running at 200 MHz. Our architecture, using the 40 nm technology available on Virtex 6 FPGA, can process this video stream at frame rate of 29 fps at 250 MHz (around 232K Macroblocks/s).

Table 2. Number of cycles required depending on sub-block size

sub-block types	block number	Chen's		Yang's		Our's	
		cycles/ block	total cycles	cycles/ block	total cycles	cycles/ block	total cycles
16x16	1	22x4	88	22	22	22	22
16x8	2	14x4	112	14	28	14	28
8x16	2	22x2	88	22	44	22	44
8x8	4	14x2	112	14	56	14	56
8x4	8	10x2	160	10	80	10	80
4x8	8	14x1	112	14 ÷ 2	56	14	112
4x4	16	10x1	160	10 ÷ 2	80	10	160
Latency		NA		29		29	
Total	41	1644		790		553	

5 Conclusion

We proposed in this paper, a flexible motion estimator which enables the integer search strategy to be adjusted and the optional VBSME and sub-pel refinements to be processed. This low-cost implementation, based on Virtex FPGA enables to reach high-speed performances. Hence for IME, 1080 HD video streams can be processed up to 200 fps. Moreover for FME mode, the same video streams can be processed at frame rate of 29 fps at 250 MHz (around 232K Macroblocks/s). Current developments aim to improve these performances, specially the sub-pel interpolation units. This solution can therefore represent an efficient adaptative solution more many video coding applications. Finally, the use of FPGA technology enables the dynamic reconfiguration to be considered. Therefore the ME accelerators could be even more scalable and can be dynamically adjusted according to the events happening in the video scene or some environment modifications (as a network bandwidth reduction).

References

1. Wiegand, T., Sullivan, G.J., Bjontegaard, G., Luthra, A.: Overview of the H.264/AVC video coding standard. IEEE Trans. on Circuits and Systems for Video Technology 13(7), 560–576 (2003)
2. Chen, T.C., Huang, Y.W., Chen, L.G.: Fully utilized and reusable architecture for fractional motion estimation of H.264/AVC. In: IEEE ICASSP, pp. 9–12 (2004)
3. Swee, Y.Y., McCanny, J.V.: A VLSI Architecture for Variable Block Size Video Motion Estimation. IEEE Trans. on Circuits and Systems for Video Technology 51(7), 384–389 (2004)
4. Koga, T., Ilinuma, K., Hirano, A., Iijima, Y., Ishiguro, T.: Motion Compensated Interframe Coding For Video Conferencing. In: Proc. Nat. Telecommun Conf., New Orleans, pp. G5.3.1–G5.3.5 (1981)
5. Li, R., Zeng, B., Liou, M.L.: A New Three-Step Search Algorithm for Fast Motion Estimation. IEEE Trans. on Circuits and Systems for Video Technology 4(4), 438–442 (1994)
6. Po, L.M., Ma, W.C.: A Novel Four-Step Search Algorithm for Fast Block Motion Estimation. IEEE Trans. on Circuits and Systems for Video Technology 6(3), 313–317 (1996)
7. Zhu, C., Lin, X., Chau, L.P.: Hexagon-Based Search Pattern for Fast Block Motion Estimation. IEEE Trans. on Circuits and Systems for Video Technology 12(5), 349–355 (2002)
8. Zhu, S., Ma, K.K.: A New Diamond Search Algorithm For Fast Block Matching Motion Estimation. IEEE Trans. on Image Process 9(2), 287–290 (2000)
9. Cheung, C., Po, L.M.: A Novel Cross-Diamond Search Algorithm for Fast Block Motion Estimation. IEEE Transactions on Circuits and Systems for Video Technology 12(12), 1168–1177 (2002)
10. Liu, L., Feig, E.: 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 (1996)
11. Lee, Y.G., Ra, J.B.: Fast Motion Estimation Robust to Random Motions Based on a Distance Prediction. IEEE Transactions on Circuits and Systems for Video Technology 16(7), 869–875 (2006)
12. Ismail, Y., McNeelly, J., Shaaban, M., Bayoumi, M.: Enhanced efficient diamond search algorithm for fast block motion estimation. In: IEEE ISCAS, Taipei, pp. 3198–3201 (2009)
13. Yang, C., Goto, S., Ikenaga, T.: High Performance VLSI Architecture of Fractional Motion Estimation in H.264 for HDTV. In: Proceedings of the IEEE ISCAS, Greece, pp. 2605–2608 (2006)
14. Chen, Y.H., Chen, T.C., Chien, S.Y., Huang, Y.W., Chen, L.G.: VLSI Architecture Design of Fractional Motion Estimation for H.264/AVC. Journal of Signal Processing Systems 53(3), 335–347 (2008)
15. Dubois, J., Mattavelli, M., Pierrefeu, L., Miteran, J.: Configurable Motion-Estimation Hardware Accelerator Module For The Mpeg-4 Reference hardware Description Platform. In: Proceedings of IEEE International Conference on Image Processing (ICIP 2005), Genova (2005)
16. Choudhury, A.R., Badawy, W.: A Quarter Pel Full Search Block Motion Estimation Architecture for H.264/AVC. In: IEEE ICME 2005 (2005)
17. Ruiz, G.A., Michell, J.A.: An Efficient VLSI Architecture of Fractional Motion Estimation in H.264 for HDTV. Journal of Signal Processing Systems 62(3), 443–457 (2010)