H.264 FRACTIONAL MOTION ESTIMATION REFINEMENT: A REAL-TIME AND LOW COMPLEXITY HARDWARE SOLUTION FOR HD SEQUENCES

F URBAN[1,2], *R POULLAOUEC*[1], *JF NEZAN*[2], *O DEFORGES*[2]

(1) THOMSON RD FRANCE,
Video Compression Lab
1 av. de belle fontaine, CS 17616
35576 Cesson Sévigné CEDEX, France
fabrice.urban@thomson.net

(2) IETR/Image group Lab
UMR CNRS 6164/INSA
20 av. des Buttes de Coësmes
35043 RENNES Cedex, France
jnezan,odeforge@insa-rennes.fr

ABSTRACT

The MPEG-4 AVC/H.264 video compression standard introduces a high motion estimation complexity. Quarter-pixel accuracy and variable block size enhances compression performances, but increase computation requirements. We propose a low complexity VLSI design for variable block size fractional motion estimation of high definition video sequences. Thanks to an improved datapath a high throughput is achieved with low logic resources. A complete real-time motion estimation application has been prototyped on a heterogeneous platform comprising a DSP and a FPGA. The system achieves motion estimation of 720p sequences at 60 frames per second.

1. INTRODUCTION

Motion estimation is known to be a key operation for video compression. A highly accurate motion estimation can significantly reduce the bit-rate of a video stream, but involves a high computational complexity. For the H.264 video compression standard up to 60% of the computation load of the video encoder concerns motion estimation with multiple reference frames, variable block search and Fractional-accuracy Motion Estimation (FME).

Integer Motion Estimation (IME) has been widely studied in the past few years. Fast algorithms have been developed to reduce the computational burden without decreasing quality [1, 2, 3], and a lot of VLSI architectures have been designed [4, 5, 6].

However with FME, the complexity and memory bandwidth are highly increased. A few work on FME coprocessors can be found [7, 8, 9]. This paper presents a new design to refine a motion vector to sub-pixel accuracy. The datapath of the Processing Element (PE) is optimized for fractional-pixel accuracy. The design achieves high efficiency with limited hardware resource. It supports variable block size with no hardware modification and Lagrangian cost computation.

The design has been prototyped in a complete motion estimation application on a multicomponent platform. A Digital Signal Processor (DSP) computes IME and an FPGA refines the motion field to quarter-pixel accuracy using H.264 standard interpolation filters. The system achieves real-time motion estimation on 720p (1280x720) sequences up to 60 frames per second.

The paper is organized as follows: Section 2 introduces fractional-pixel accuracy motion estimation, section 3 proposes an optimized VLSI design, section 4 gives prototyping results on a heterogeneous platform, finally section 5 concludes.

2. FRACTIONAL-PIXEL ACCURACY MOTION ESTIMATION OVERVIEW

Increasing the precision of motion vectors enhances the compression performances of a video encoder, but introduces an extra computation cost. This section evaluates the complexity of fractional-pixel motion estimation for H.264 high definition video encoding.

In the MPEG-4/AVC H.264 standard, the quarter-pixel accuracy luminance picture is interpolated with two successive filtering stages [10]. Half-pixel samples are interpolated first using a 6-tap separable FIR filter with coefficients (1,-5, 20, 20, -5, 1). Once half-pixel samples are available, quarter pixel samples are computed by averaging two adjacent samples horizontally, vertically or diagonally.

The computation overhead introduced by sub-pixel refinement is due to two operations: the interpolation of sub-pixel samples and the evaluation of new displacements (distortion computation).

2.1 Interpolation strategy

The image resolution is enhanced, involving the use of interpolation filters, and higher memory constraints. Interpolating the full sub-pixel frame before motion estimation is usually banished because of the huge necessary memory bandwidth (For quarter-pixel FME the memory size and bandwidth are increased by a 16:1 ratio compared to IME). Instead an on-the-fly interpolation strategy is preferred. the filters are applied on a small search window around the best integer-pixel accuracy matching block (Fig. 1). Memory constraints are thus reduced. Hardware and software implementations can take advantage of high bandwidth local memories. External memory access concern only integer-pixel accuracy data.

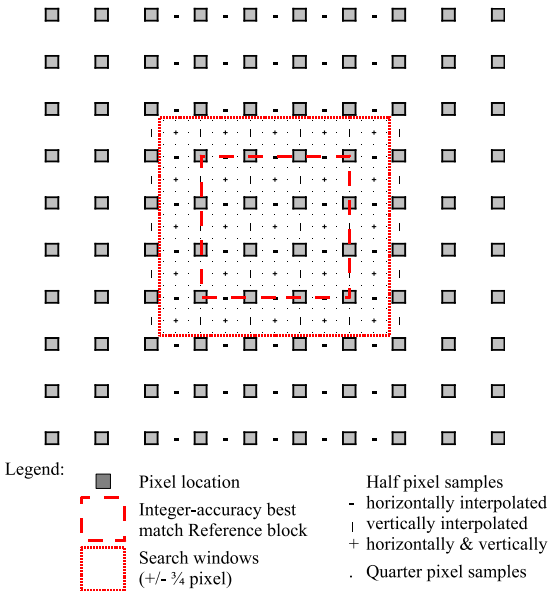


Figure 1: Sub-pixel search window and necessary data to compute quarter-pixel samples for a 4x4 block

2.2 Search strategy

The density of candidate motion vectors is increased. To limit the number of search points a two-step approach is generally preferred:

The motion is estimated at integer pixel accuracy and then refined to quarter-pixel with a limited search range (usually $r < 1$ pixel) around the integer-accuracy best match.

The FME search strategy depends on the target and implementation. For hardware implementations a restricted full search usually reduces logic control and optimizes the datapath. For software implementations it is better to reduce computations and a two step approach (logarithmic search [1]) is usually adopted.

3. PROPOSED VLSI ARCHITECTURE FOR FME

Despite the small search window of sub-pixel refinement, this step is the most computationally intensive part of motion estimation. This section describes a dedicated VLSI architecture for fractional-pixel accuracy motion vector refinement. It achieves high throughput with low memory bandwidth and low resource utilization.

3.1 Overall VLSI architecture design

To benefit from the high degree of parallelism of a VLSI implementation the algorithm must be regular. Therefore a quarter-pixel accuracy full search approach with Lagrangian optimization is adopted with a search range of $\frac{3}{4}$ pixels in each direction. 48 candidates ($(2*3+1)^2$-1) are evaluated around the IME best match. Necessary input data are the pixel-accuracy reference window, the current block and the differentially coded integer-accuracy motion vector in order to compute a Lagrangian cost for each candidate.

The sub-pixel refinement coprocessor includes (fig 2) a quarter pixel interpolation module to generate sub-pixel samples. A Processor Element (PE) matrix is in charge of computing a distortion measure for each candidate displacement. The adopted distortion measure is the Sum of Absolute Differences (SAD). The best candidate (holding the smallest SAD) is finally selected in the decision tree. The block size is parameterizable without any hardware modification in order to support multiple block size motion estimation.

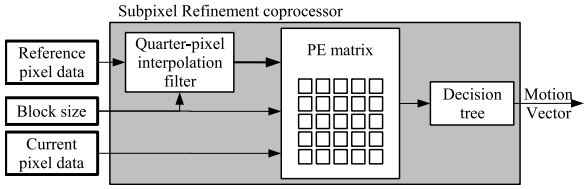


Figure 2: Overall pipeline design

The design of each module is in adequacy with the throughput of previous one. The datapath is thus optimized. The design allows high throughput with low resources utilization. External memory bandwidth is kept low thanks to efficient data reuse.

3.2 Interpolation filters

Sub-pixel data is interpolated on-the-fly to eliminate the need for data memorization. Reference window pixels are input serially in raster-scan order beginning from top-left corner of the reference window. Once the filter is initialized (depending on the filter size), 16 quarter-pixel samples are generated per input pixel, as shown in Fig. 3-left. To improve parallelism p input pixels are processed simultaneously per clock cycle, resulting in outputting $16 \times p$ quarter pixels per clock cycle.

Any interpolation filter can be used: from H.264 filters for best accuracy to bilinear filter for resources and time saving.

3.3 PE matrix

From existing dedicated architectures for IME that can be be found in previous work [4, 5, 11], designs using inter-layer parallelism [5] (or type II architecture [4]) best fit line-scan input mode and small search range and best minimize the hardware resources.

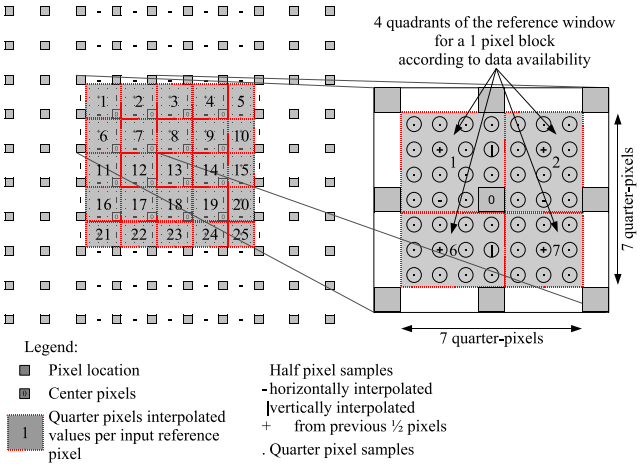


Figure 3: Scheduling of H.264 filtered sub-pixels availability for a 4x4 block (left) and distortion dependencies (right)

In existing full-search sub-pixel refinement architectures [8, 9] the interpolation filter is not included in the study. We propose here a distortion computation matrix architecture that match the interpolation filter design to reduce hardware requirements. In addition the subsequent decision tree is also reduced.

3.3.1 *Integer-pixel accuracy full search modeling*

In[4] the full search Integer Motion Estimation (IME) algorithm with a search range of +/- r for a $n \times m$ block is expressed as 4 nested loops (Alg. 1).

Algorithm 1 IME nested loops

for $k = 1..m$ (block height)
 for $l = 1..n$ (block width)
  for $\Delta i = -r..r$ (vertical search range)
   for $\Delta j = -r..r$ (horizontal search range)
    $SAD(\Delta j, \Delta i) += |x_1(k,l) - x_2(k+\Delta i, l+\Delta j)|$
   end Δj
  end Δi
 end l
end k

x_1 denotes current picture and x_2 reference picture

Inter-layer parallelism is obtained by unrolling Δi and Δj loops in hardware. The distortion measure is computed simultaneously for every search point in $(2 \times r + 1)^2$ PEs. This model results from two hypothesis: firstly the current pixel $x_1(k,l)$ is broadcast to all PEs and secondly all reference pixels from $x_2(k-r, l-r)$ to $x_2(k+r, l+r)$ are available and propagated to the PEs with registers.

3.3.2 *Inverted sub-pixel accuracy full search modeling*

For Fractional Motion Estimation (FME) images x_1 and x_2 do not have the same scale. Data density is higher for the search window then for the current block. Consequently the number of propagation register increase exponentially with the accuracy [8]. To reduce the number of registers, the technique used in [11] for IME, which consists in broadcasting reference window data to the PEs and propagating current block can be derived to FME: reference window data is spread to appropriate PEs. Alg. 1 is then inverted (let $u = k + \Delta i$ and $v = l + \Delta j$) and two loops (Δg and Δh) are added to handle fractional-pixel accuracy. The model is transformed into Alg. 2. The number of registers is then independent of the sub-pixel accuracy. Δi and Δj are integer displacements and Δg and Δh are

sub-pixel accuracy displacements with a being the accuracy factor (i.e. $a = 2$ for half pixel, $a = 4$ for quarter-pixel).

Algorithm 2 new FME nested loops

```
for u = 1 − s..m + s (block height)
 for v = 1 − s..n + s (block width)
  for Δi = −s..s (integer vertical search range)
   for Δj = −s..s (integer horizontal search range)
    for Δg = −(a − 1)..0 (fractional v. search range)
     for Δh = −(a − 1)..0 (fractional h. search range)
      SAD(Δj, Δi)+ = |x1(u − Δi, v − Δj)
            −x2(au + Δg, av + Δh)|
     end Δh
    end Δg
   end Δj
  end Δi
 end v
end u
```

with $-r \leq a\Delta i + \Delta g \leq r$ and $-r \leq a\Delta j + \Delta h \leq r$
and $1 \leq u - \Delta i \leq m$ and $1 \leq v - \Delta j \leq n$

Loops Δi, Δj, Δg and Δh are unrolled in hardware (e.g. for quarter pixel accuracy this results in a 7×7 PE matrix). As a result reference pixels from $x_2(ak - r, al - r)$ to $x_2(ak, al)$ are spread to appropriate PEs and current pixels $x_1(k,l)$ to $x_1(k - s, l - s)$ are propagated.

The inequalities $1 \leq u - \Delta i \leq m$ and $1 \leq v - \Delta j \leq n$ are ensured in hardware by propagating "Enable" signals of the PEs along with current block data. The first a^2 cost results are available after $m(n+1)$ cycles, the subsequent ones after appropriate delays. The resulting systolic architecture (Fig. 4) has many advantages.

- Propagation registers concern now the current block and their number is roughly divided by a^2.
- The subsequent comparison unit is reduced (shorter pipeline) because the distortion results of search points are not available simultaneously.
- The system works as fast as the interpolation filter can provide data because search area samples computed on the fly are used immediately.
- Interpolated data are never stored, thus saving bandwidth and memory/registers.

The proposed architecture reduces hardware requirements over previous ones while giving optimal performances. The distortion computation unit and the comparison tree are designed taking into account data dependencies with interpolation filter. The absolute differences between the quarter-pixel samples and the current pixel values are computed as soon as the data is available then accumulated. Fig 4 shows the design of the SAD computation matrix. For clarity of the figure only half pixel is represented ($a = 2$ and $r = \frac{1}{2}$) and $p = 1$. The 9 positions are partitioned in 4 quadrants because of data causality (current block propagation). Each quadrant represents the unrolling of Δg and Δh loops of Alg.2. The 4 quadrants is the result of the unrolling of Δi and Δj loops. The top left quadrant is started first, the top right quadrant is delayed by one cycle and the delay of the bottom quadrants corresponds to the computation of one line. Dummy cycles necessary to fill the search area registers are removed from the initialization step. Instead results are partitioned according to data availability and the decision tree resources are reduced. Indeed the comparison units are shared; for quarter-pixel accuracy there are at most $a^2 = 16$ values (out of 49) to compare at a time.

This architecture makes full use of the inter-layer parallelism, and in addition supports intra-level parallelism. Performances can be increased by unrolling Δl loop by a factor p. The detail of PE implementation on the fig 5 corresponds to $p = 2$. At each cycle 2 absolute differences are computed and accumulated. The performances are thus roughly doubled at the cost of a hardware resource increase (the interpolation filters and PE matrix are enlarged).

In order to consider rate-distortion optimization, the SAD accumulator is initialized by a value corresponding to the Lagrangian cost of the vector. This value is computed with very low hardware resources during the initialization of the interpolation pipeline.

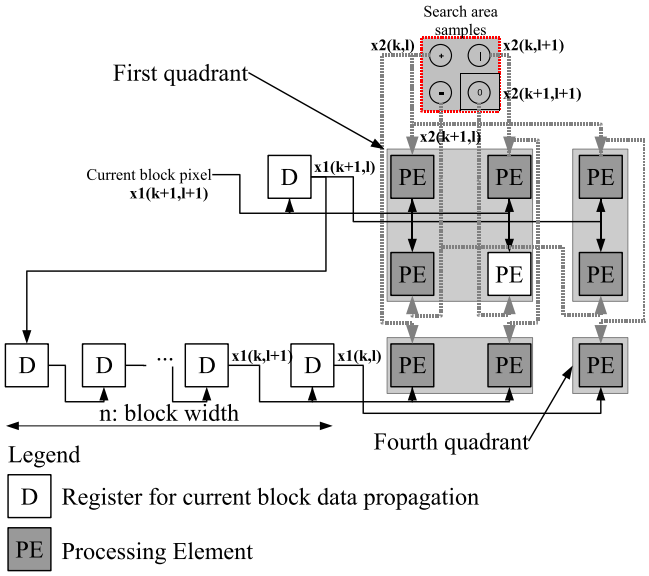


Figure 4: SAD matrix design (half pixel)

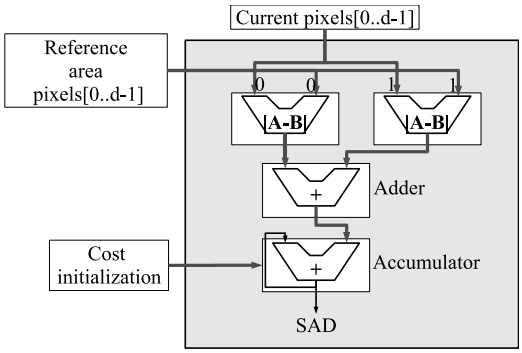


Figure 5: PE design

This architecture is compatible with a low memory bandwidth but achieves a high degree of parallelism. The use of both inter- and intra-level parallelism further enhances performances.

Hardware resources are almost independent of the block size. The block size is parameterizable with no hardware modification.

3.4 Decision tree

To obtain the optimal motion vector, costs function computed in the PE matrix are compared to each other in a dedicated comparison tree (Fig. 6). Simultaneously available values (issued from the same quadrant) are compared in a binary tree structure. The lowest cost of each quadrant is then sequentially compared to the current optimum. Finally the sub-pixel motion vector is output when the 4th quadrant's costs have been processed.

The binary tree base size is limited to a^2 thanks to the early start of distortion calculation in the PEs. Consequently its resources are shared between the 4 quadrants without delaying the result. On the contrary, because there are less values to compare the processing pipeline is shorter.

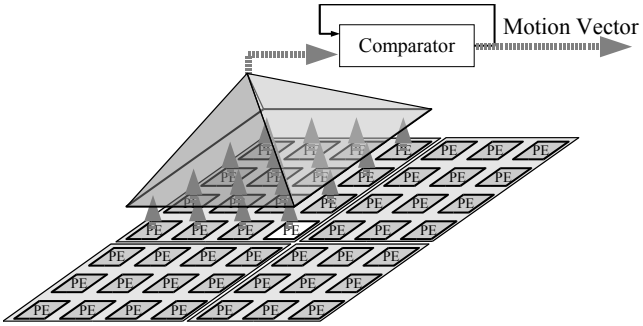
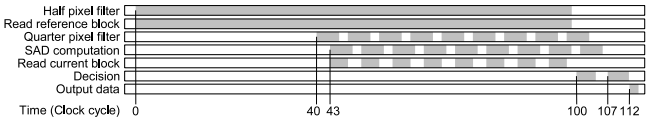
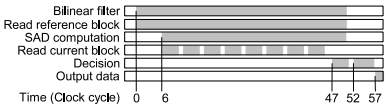


Figure 6: Decision tree



(a) H.264 filter



(b) Bilinear filter

Figure 7: Units scheduling for $p = 2$

3.5 Implementation results

Compared to previous motion estimation architectures, the data is processed as soon as it is available instead of waiting for the whole search window. Data register count and logic are thus reduced thanks to an improved datapath.

Input width	Filter type	pipeline length	Total cycles
$p = 1$ byte	H.264	13	209
	Bilinear	8	108
$p = 2$ bytes	H.264	14	112
	Bilinear	9	59
$p = 4$ bytes	H.264	14	70
	Bilinear	9	39

Table 1: FPGA implementation timings

The number of cycles necessary to refine an 8x8 motion vector to quarter-pixel accuracy is given table 1 for different implementations. The implementation of the refinement with an intra-level parallelism of 2 pixels per cycle results in 112 cycles latency with the H.264 interpolation filter and 59 cycles with the bilinear filter. The difference is due to the filter size which implies a longer initialization time for horizontal and vertical interpolation with the standard filter (Fig. 7). The system using the H.264 filter and clocked at 133MHz produces a result in 840ns for 8x8 blocks. This is fast enough to handle 720p frames at 60Hz. For better performances, intra-level parallelism can be increased.

Tab 2 gives synthesis results for the quarter pixel refinement implementation with H.264 filter and 2 pixels per cycle on a Xilinx Virtex 2 Pro FPGA. Resources can be drastically reduced with a bilinear filter, for which an intra-level parallelism of 1 pixel per cycle is sufficient to obtain comparable timing results. The maximum frequency is 150 MHz. A 32x32-bits multiplier is used to compute the Lagrangian cost. The propagation registers necessary in the interpolation filters and PE matrix are implemented in RAM blocks. Consequently the logic cost is reduced and almost independent of the block size. The design supports then variable block size with virtually no extra cost.

register Implementation	Max block size	4-input LUTs	Gate count
Block RAM (10)	8x8	6.3K	98K
Block RAM (10)	16x16	6.4K	99.5K
LUTs (1.4 Kb)	8x8	7.4K	127K
LUTs (2.5 Kb)	16x16	8.9K	162K

Table 2: Synthesis results on XC2VP20-5

4. MULTICOMPONENT ME PROTOTYPE

A complete motion estimation application has been prototyped on a heterogeneous platform composed of a TI Digital Signal Processor C6416 at 1GHz and a Virtex 2 Pro FPGA. The DSP runs the IME algorithm and the FPGA handle FME refinement. The operations have been efficiently scheduled to take advantage of the available parallelism of the platform.

4.1 IME on DSP

Many IME have been developed to find a compromise between computation complexity and motion vector accuracy [1, 12, 13, 14, 2, 15]. The well-known EPZS (Enhanced Predictive Zonal Search) [2] and hierarchical (HME) [15] algorithms have been implemented and evaluated in [3]. EPZS offers good performances both in execution time and accuracy. HME is more robust but is more computationally intensive. For the rest of this paper EPZS algorithm have been retained since it is faster and more popular for software implementation.

4.2 FPGA as a sub-pixel refinement coprocessor

The sub-pixel refinement operation being very computationally intensive, this task is performed on an FPGA where the design presented above is implemented.

The IME is based on a predictive algorithm (cf Section 4.1). Previously estimated motion vectors are necessary to predict the current motion as well as to compute the Lagrangian cost. This causes data dependencies between IME and FME which results in inevitable sequential processing. To take advantage of the parallel multicomponent architecture the motion vector of the left block is input in the IME stage at integer accuracy instead of quarter-pixel accuracy. As a result the motion estimation architecture is a 2-stage pipeline. The first stage is the IME on DSP and the second stage is the FME on FPGA (Fig. 8).

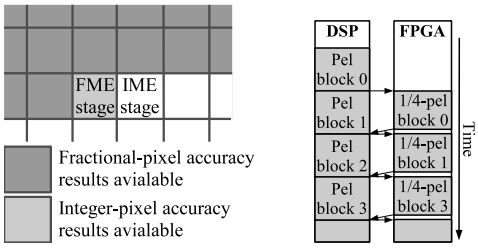


Figure 8: Block-level pipeline implementation

4.3 Timing results

The table 3 gives timing results per block and per 1280x720 high definition frame. The DSP runs the EPZS Integer motion estimation algorithm at 900 ns for a 8x8 block and 4000 ns for a 16x16 block.

The sub-pixel refinement on the FPGA takes only 842 and 1925 ns and is realized in parallel thus with no overhead.

Practical results however present an execution time of 1250 and 4600 ns for 8x8 and 16x16. The small overhead is due to the prototyping platform constraints. No external memory is connected to the FPGA thus current block and reference window must be transferred by the DSP to input buffers on the FPGA. This increases refinement operation which has to take into account data transfers.

Block size	8x8	16x16
DSP pel	900 ns 720p frame: 13 ms	4000 ns 14.4 ms
FPGA 1/4 pel refinement only w H.264 filter	842 ns (12 ms)	1925 ns (7 ms)
Total DSP + FPGA Pel + 1/4 pel	1250 ns **720p frame: 18 ms**	4600 ns **16.5 ms**

Table 3: Execution times for 8x8 and 16x16 blocks

With one 1GHz DSP and an FPGA at 133 MHz the computation of 1/4-pixel motion fields achieves 55 and 60 frames per second for 8x8 and 16x16 block size respectively.

The coprocessor achieves good acceleration results with low resources and memory bandwidth. To further improve performances, the intra-level parallelism can be increased. The throughput roughly doubles each time input bandwidth p doubles but it increases logic ressources.

Compared to previous work [7, 8, 9, 6], the proposed design achieves high throughput and low resources with no implmentation trade-off. Thanks to the use of a DSP the system offers a high degree of flexibility.

4.4 Impact of quarter-pixel motion vector refinement

In order to compare the impact of implementation trade-offs on the motion field quality the motion estimator have been implemented in an H.264 video encoder internally developed at Thomson R&D. Several 1280x720 high definition video sequences have been encoded with variable block size motion estimation from 8x8 to 16x16, one reference frame and constant quantification at different quantification steps and for different configurations of the motion estimation.

The two sequences that highlight the most FME impact are “city” and “horses”. They are low motion and high detail sequences. the quarter-pixel accuracy motion estimation reduces the bit-rate by more than 50%. Table 4 summarize relative bit-rate loss with or without sub-pixel motion refinement.

FME implementation	Bit-rate loss 720p “city”	Bit-rate loss 720p “horses”
$\frac{1}{4}$-pixel with block-level pipeline	0	0
pixel accuracy	61 %	24 %

Table 4: Implementation trade-offs and relative bit-rate loss

5. CONCLUSION

On the one hand Fractional Motion Estimation involves a high computation power, especially for high definition video and latest compression standards. One the other hand encoding performances are highly increased.

An efficient VLSI design is proposed to refine motion vectors to sub-pixel accuracy with a small search range around a previously computed integer-accuracy motion vector. The hardware resources are reduced thanks to a new PE matrix design well suited to FME. It requires low memory bandwidth and achieves high throughput. The design supports variable block size motion estimation.

A complete motion estimation application has been prototyped on a heterogeneous platform embedding a DSP and a FPGA. The system computes 720p motion fields up to 60 frames per seconds.

REFERENCES

[1] J. R. Jain and A. K. Jain, “Displacement measurement and its application in interframe coding,” *IEEE Transactions on Communications*, vol. COM-29(12), pp. 1799–1808, 1981.

[2] A. M. Tourapis, “Enhanced Predictive Zonal Search for Single and Multiple Frame Motion Estimation,” *Visual Communications and Image Processing*, pp. 1069–79, 2002.

[3] F. Urban, R. Poullaouec, J. F. Nezan, and O. Déforges, “Real-time Multi-DSP Motion Estimator for MPEG-4 AVC/H.264 High Definition Video,” *International Conference on Signals and Electronic Systems*, September 2006.

[4] L. Vos and M. Stegherr, “Parameterizable VLSI architectures for the full-search block-matching algorithm,” *IEEE Transactions on Circuits and Systems*, vol. 36 issue 10, pp. 1309–1316, 1989.

[5] C.-Y. Chen, S.-Y. Chien, Y.-W. Huang, T.-C. Chen, T.-C. Wang, and L.-G. Chen, “Analysis and architecture design of variable block-size motion estimation for H.264/AVC,” *IEEE Transactions on Circuits and Systems I*, vol. 53, no. 3, pp. 578 – 593, March 2006.

[6] K. Gaedke, M. Borsum, M. Georgi, A. Kluger, J.-P. Le Glanic, and P. Bernard, “Architecture and VLSI implementation of a programmable HD real-time motion estimator,” *IEEE International Symposium on Circuits and Systems*, May 2007.

[7] T.-C. Chen, Y.-W. Huang, and L.-G. Chen, “Fully utilized and reusable architecture for fractional motion estimation of H.264/AVC,” *Internatinal Conference on Acoustics, Speech an Signal Processing*, vol. 5, pp. 9–12, 2004.

[8] T. Dias, N. Roma, and L. Sousa, “Fully parameterizable vlsi architecture for sub-pixel motion estimation with low memory bandwidth requirements,” November 2005.

[9] C. Rahman and W. Badawy, “A quarter pel full search block motion estimation architecture for H.264/AVC,” *IEEE International Conference on Multimedia and Expo*, July 2005.

[10] I. E. Richardson, *H.264 and MPEG-4 Video Compression: Video Coding for Next-generation Multimedia*. John Wiley and Sons, 2003.

[11] H. Yeo and Y. H. Hu, “A novel modular systolic array architecture for full-search blockmatching motion estimation,” *International Conference on Acoustics, Speech, and Signal Processing*, vol. 5, pp. 3303–3306, May 1995.

[12] P. Hosur and K. Ma, “Motion Vector Field Adaptive Fast Motion Estimation,” *Second International Conference on Information, Communications and Signal Processing (ICICS ’99)*, 1999.

[13] K. Virk, N. Khan, S. Masud, F. Nasim, and S. Idris, “Low Complexity Recursive Search Based Motion Estimation Algorithm for Video Coding Applications,” in *Proceedings of 13th European Signal Processing Conference*, Antalya, Turkey, 2005.

[14] Y.-S. Chen, Y.-P. Hung, and C.-S. Fuh, “Fast Block Matching Algorithm Based on the Winner-Update Strategy,” in *IEEE Transactions on Image Processing*, vol. 10, August 2001.

[15] B. Chupeau, P. Robert, M. Pecot, and P. Guillotel, “Multiscale motion estimation,” *Workshop on Advanced Matching in Vision and Artificial Intelligence*, 5th, 6th June 1990.