Customisable Core-Based Architectures for Real-Time Motion Estimation on FPGAs

Nuno Roma, Tiago Dias, and Leonel Sousa

Instituto Superior Técnico / INESC-ID
Dept. of Electrical and Computer Engineering
Rua Alves Redol, 9 - 1000-029 Lisboa - PORTUGAL
Nuno.Roma@inesc-id.pt, tdias@sips.inesc-id.pt, las@inesc-id.pt
http://sips.inesc-id.pt

Abstract. This paper proposes new core-based architectures for motion estimation that are customisable for different coding parameters and hardware resources. These new cores are derived from an efficient and fully parameterisable 2-D single array systolic structure for full-search block-matching motion estimation and inherit its configurability properties in what concerns the macroblock dimension, the search area and parallelism level. The proposed architectures require significantly fewer hardware resources, by reducing the spatial and pixel resolutions rather than restricting the set of considered candidate motion vectors. Low-cost and low-power regular architectures suitable for field programmable logic implementation are obtained without compromising the quality of the coded video sequences. Experimental results show that despite the significant complexity level presented by motion estimation processors, it is still possible to implement fast and low-cost versions of the original core-based architecture using general purpose FPGA devices.

1 Introduction

Motion estimation is a fundamental operation in motion-compensated video coding [1], in order to efficiently exploit the temporal redundancy between successive frames. Among the several possible approaches, block-matching is the most used in practice. In this strategy, the current frame is divided into equal sized $N \times N$ pixel blocks that are displaced within a $(N+2p-1) \times (N+2p-1)$ search window defined in the previous frame. The motion vector is determined by looking for the best matched block of this search window. The Sum of the Absolute Differences (SAD) is the matching criteria that is usually used by most systems, due to its efficiency and simplicity.

Although the Full-Search Block-Matching (FSBM) method exhaustively considers all possible candidate blocks, which guarantees the optimal solution, it requires a lot of computational resources. In fact, FSBM motion estimation can consume up to 80% of the total computational power required by a video encoder. Hence, most of the fast block-matching motion estimation algorithms that have been proposed over the last years restrict the search space by a given search pattern, providing suboptimal solutions (e.g. [2, 3]). However most of these algorithms apply non-regular processing and

require complex control schemes, which make their hardware implementation difficult and rather inefficient.

Hence, the main objective of this paper is to propose efficient core-based VLSI array architectures based on FSBM to be implemented in Field Programmable Logic (FPL) devices. These architectures are based on a highly efficient core that was recently developed by the authors of this work, that combines both pipelining and parallel processing techniques to design powerful motion estimators based on multiple array architectures and using Application Specific Integrated Circuits (ASIC) [4]. In this paper, this original core architecture is used to derive simpler structures with reduced hardware requirements. The complexity of these architectures is reduced by decreasing the precision of the pixel values or/and the spatial resolutions in the current frame, while maintaining the original resolution in the search space. The pixel precision is configured by defining the number of bits used to represent the input data and by masking or truncating the corresponding Least Significant Bits (LSBs). On the other hand, spacial resolution is adjusted by sub-sampling the blocks of the current frame. By doing so, while the best candidate block in the previous frame is exhaustively searched, the SAD of each candidate block is computed by using only sparse pixels. Considering a typical setup with 16×16 pixels block, by applying $2:1$ or $4:1$ alternate sub-sampling schemes the number of considered pixels decreases by $1/4$ and $1/16$, respectively.

The efficiency of the proposed structures were evaluated by implementing these customisable core-based architectures in Field Programmable Gate Arrays (FPGA). It is shown that the amount of hardware required by these architectures when sub-sampling and truncation techniques are applied is considerable reduced, which allows the usage of a common framework for designing a wide range of motion estimation processors with different characteristics. Moreover, experimental results obtained with benchmark video sequences show that the application of those techniques does not introduce a significant degradation in the quality of the coded video sequences. These reduced hardware requirements architectures fit well in actual FPGAs, being a real alternative to those fast motion estimation techniques that require non-regular processing.

This paper is organised as follows. The original FSBM core architecture is presented in section 2. Section 3 proposes reduced hardware architectures to implement motion estimators on FPL devices. Experimental results obtained with FPGAs are presented in section 4 and section 5 concludes the paper.

2 FSBM Core-Based Architectures

Most motion estimation architectures that have been proposed in the last few years for hardware implementation are based on the optimum FSBM algorithm. The main reason for this is not only related to the better performance levels that it generally provides, but it is mainly due to the regularity properties that it also offers. In fact, not only does it conduct to much more efficient hardware structures, but it also provides the usage of significantly simpler control units, which is always a fundamental factor towards a real-time operation based on hardware structures.

Several FSBM structures have been proposed over the last few years (e.g.: [5, 6, 4]). Very recently, it was presented in [4] a new class of parameterisable hardware archi-

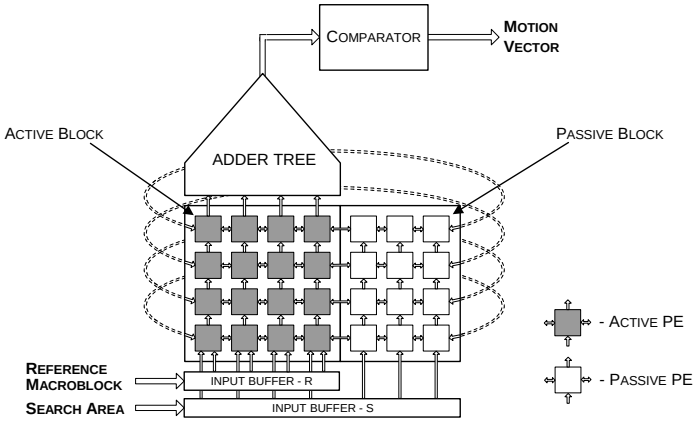


Fig. 1. Processor array proposed in [4] based on an innovative cylindrical structure and adopting the zig-zag processing scheme proposed by Vos [6] ($N = 4$, $p = 2$).

tectures that is characterised by offering minimum latency, maximum throughput and a full and efficient utilisation of the hardware resources. This last characteristic is a fundamental requisite in any FPL system, due to the limited amount of hardware resources. To achieve such performance levels, a peculiar and innovative processing scheme based on a cylindrical hardware structure and on the zig-zag processing sequence proposed by Vos [6] was adopted (see fig. 1). With such a scheme, not only was it possible to minimise the processing time, but it also provided the ability to prevent the usage of some hardware structures (the so called *passive processor elements (PEs)*) that do not carry useful information at some clock cycles.

Moreover, besides this set of performance and implementation characteristics, one other important feature of this class of processors is concerned with its scalable and configurable architecture, making it possible to easily adapt the processor configuration to fulfil the requisites of a given video coder. By adjusting a small set of implementation parameters, processing structures with distinct performance and hardware requirements are obtained, providing the ability to adjust the required hardware resources to the target implementation technology. As an example, while high performance processors requiring more resources are more suited for implementations using technologies such as ASIC or Sea-of-Gates, those low-cost processors that are meant to be implemented in FPL devices with limited hardware resources should use configurations requiring reduced amount of hardware.

3 Reduced Hardware Architectures

Despite the set of configurable properties offered by FSBM architectures and, in particular, by the class of processors proposed in [4], FPL devices often do not provide enough hardware resources to implement such processors. In those cases, the solution that is often adopted is the usage of processing structures based on sub-optimal motion estimation algorithms, that provide faster processing times and require reduced amounts

of hardware. Three different categories of sub-optimal motion estimation algorithms have been proposed:

- *Reduction of the set of considered candidate motion vectors*, where the search procedure in the previous frame is restricted to a search pattern within the search window, by using hierarchical search strategies [2, 3];
- *Decimation at the pixel level*, where the considered similarity measure is computed by using only a subset of the $N \times N$ pixels of each reference macroblock [7–9];
- *Reduction of the precision of the pixel values*, where the similarity measure is computed by truncation the LSBs of the input values to reduce the hardware resources required by the used arithmetic units [10, 9].

The main drawback of these solutions is a corresponding increase of the prediction error that inevitable arises from using a less accurate estimation. This tradeoff usually leads to a difficult and non-trivial relationship between the final picture quality and the prediction accuracy that can not be assumed to be linear. In general, a larger prediction error will lead to higher bit rates, which will conduct to the usage of greater quantisation step sizes to compensate this increase, thus affecting the quality of the decoded images.

Up until now only a few VLSI architectures have been proposed to implement fast motion estimation algorithms, by restricting the search positions according to a given search pattern [9]. In general, they imply the usage of non-regular processing structures and require higher control overheads, which difficults their design using efficient systolic structures. Consequently, they have been extensively used in software applications, where such restrictions do not usually apply so strictly.

The set of architectures that are proposed in this paper try to combine the advantages offered by the regular and efficient FSBM structures proposed in [4] with the several strategies to reduce the amount of hardware resources that are offered by sub-optimal motion estimation algorithms. By doing so, it will be possible to implement processors based on this new class of fast motion estimation processors in any FPL device, even in FPGAs with limited resources. To achieve such objective, the original FSBM architecture will be adapted to apply two of the three decimation categories referred above: the decimation at the pixel level and the reduction of the precision of the pixel values.

3.1 Decimation at the pixel level

By applying the decimation at the pixel level, the image data of the current frame is subsampled in the orthogonal directions, by considering alternate pixels in each direction. In fact, this scheme corresponds to using a lower resolution version of the reference frame in the search procedure, that is carried out within the previous full-resolution frame. As an example, in a $2:1$ sub-sampling scheme just one in each pair of consecutive pixels in each direction is considered, giving rise to decimated images with $1/4$ of the area of the original image. In the general case, the SAD similarity measure for a configuration using a $2^S:1$ sub-sampling in each direction is given by (considering N a power of 2):

$$SAD(l,c) = \sum_{i=0}^{\frac{N}{2^S}-1} \sum_{j=0}^{\frac{N}{2^S}-1} \left| x_t(i.2^S, j.2^S) - x_{t-1}(l+i.2^S, c+j.2^S) \right| \qquad (1)$$

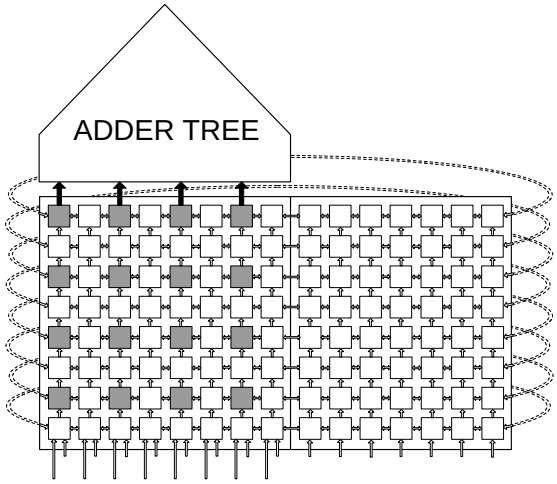


Fig. 2. Modified processing array to carry out a $2:1$ decimation function in the computation of the SAD similarity measure using the architecture proposed in [4] ($N = 8$, $p = 4$).

The FSBM circuit proposed in [4] can be easily adapted to carry out this type of sub-sampling. In fact, considering that the computation of the SAD similarity measure is performed in the active block of the processor, the decimation can be implemented by replacing the corresponding set of active PEs by passive PEs. By doing so, only the pixels with coordinates $(i.2^S, j.2^S)$ will be considered. In fig. 2 it is presented the block diagram of the modified processing array.

Since the amount of hardware resources required by passive PEs is considerable lower than that required by the active PEs, significant amounts of hardware can be saved. Moreover, by adopting the sub-sampling pattern presented in fig. 2, extra amounts of resources can be saved, since the number of inputs of the adder tree block is also reduced by the same sub-sampling factor (S).

3.2 Reduction of the precision of the pixel values

As it was previously referred, one other strategy to decrease the amount of hardware required by FSBM processors is to reduce the bit resolution of the pixel values considered in the computation of the SAD similarity function by truncating the LSBs. By adopting this strategy alone $(S = 0)$ or in conjunction with the previously described sub-sampling method $(0 < S < \log_2 N)$, the SAD similarity measure is given by:

$$SAD(l,c) = \sum_{i=0}^{\frac{N}{2^S}-1} \sum_{j=0}^{\frac{N}{2^S}-1} \left| \left\lfloor \frac{x_t(i.2^S, j.2^S)}{2^T} \right\rfloor - \left\lfloor \frac{x_{t-1}(l+i.2^S, c+j.2^S)}{2^T} \right\rfloor \right| \quad (2)$$

$$\equiv \sum_{i=0}^{\frac{N}{2^S}-1} \sum_{j=0}^{\frac{N}{2^S}-1} \left| x_t(i.2^S, j.2^S)_{7:T} - x_{t-1}(l+i.2^S, c+j.2^S)_{7:T} \right| \quad (3)$$

where T is the number of truncated bits and $x_t(i,j)_{7:T}$ are the $(8-T)$ most significant bits of a pixel value of the t^{th} frame.

The adaptation of the original FSBM architecture proposed in [4] to apply this bit truncation scheme is straightforward. In fact, it is only necessary to reduce the operands width of the several arithmetic units implemented in the active PEs, in the adder tree block and in the comparator circuit. Such modification potentially increases the maximum frequency of the pipeline and will significantly reduce the amount of required hardware, thus providing the conditions that will make it possible to implement the motion estimation processors in FPL devices.

4 Implementation and Experimental Results

The proposed customisable core-based architectures for motion estimation were completely described using IEEE-VHDL language. Both behavioural and structural parameterisable descriptions were carried out by making extensive use of "`generic`" type configuration inputs. These descriptions were used to synthesise several different setups of the proposed core in a general purposed VIRTEX XCV3200E-7 FPGA using the Xilinx Synthesis Tool from ISE 5.2.1. The considered set of configurations assumed each macroblock composed by 16×16 pixels ($N = 16$) and a maximum displacement in each direction of the search area of $p = 16$ pixels. These configurations were thoroughly tested using sub-sampling factors (S) varying between 0 and 2 and a number of truncated bits (T) of 0, 2 and 4. The experimental results of these implementations are presented in tables 1 and 3.

The proposed core-based architectures can provide significant savings of the required hardware resources. From the set of configurations presented in table 1, one can observe that reduction factors of about 75% can be obtained by using a sub-sampling factor $S = 2$ and by truncating the 4 LSBs. However, this relation should not be assumed to be linear. By considering only the pixel level decimation mechanism $(T = 0)$, it can be shown that a reduction of about 38% is obtained by using a $2:1$ sub-sampling factor $(S = 1)$, while a $4:1$ decimation will provide a reduction of about 51%. The same observation can be done by considering only the reduction of the precision of the pixel values $(S = 0)$. While using 6 representation bits $(T = 2)$ a reduction of the number of CLB slices of about 31% is obtained (a reduction of about 23% of the number of Look-up Tables (LUTs)), if only 4 representation bits are considered ($T = 4$) it provides a reduction of about 51% of the CLB slices (a reduction of about 47% of the number of LUTs). In table 2 it is presented the set of FPGA devices that should be used by each configuration, in order to maximize the efficiency of the hardware resources used by each processor.

In table 3 it is presented the variation of the maximum operating frequency of the considered configurations with the number of truncated bits (T). Contrary to what could be expected, the reduction of the operands width of the several arithmetic units does not significantly influence the processors performance. This fact can be explained if one takes into account the synthesis mechanism that is used by this family of FPGAs to synthesise and map the logic circuits using built in fast carry logic and LUTs.

	T					
	0		2		4	
S	CLB Slices	LUTs	CLB Slices	LUTs	CLB Slices	LUTs
0	90.7%	30.7%	62.6%	23.5%	43.8%	16.3%
1	56.2%	19.0%	38.2%	14.6%	26.6%	10.7%
2	44.7%	14.8%	30.1%	11.4%	21.0%	7.8%

Table 1. Percentage of the CLB slices and LUTs that are required to implement each configuration of the proposed core-based architecture for fast motion estimation in a VIRTEX XCV3200E-7 FPGA $(N = 16; p = 16)$.

	T		
S	0	2	4
0	XCV3200	XCV2600	XCV1600
1	XCV2000	XCV1600	XCV1000
2	XCV1600	XCV1000	XCV600

Table 2. Alternative FPGA devices to implement each of the considered configurations $(N = 16; p = 16)$.

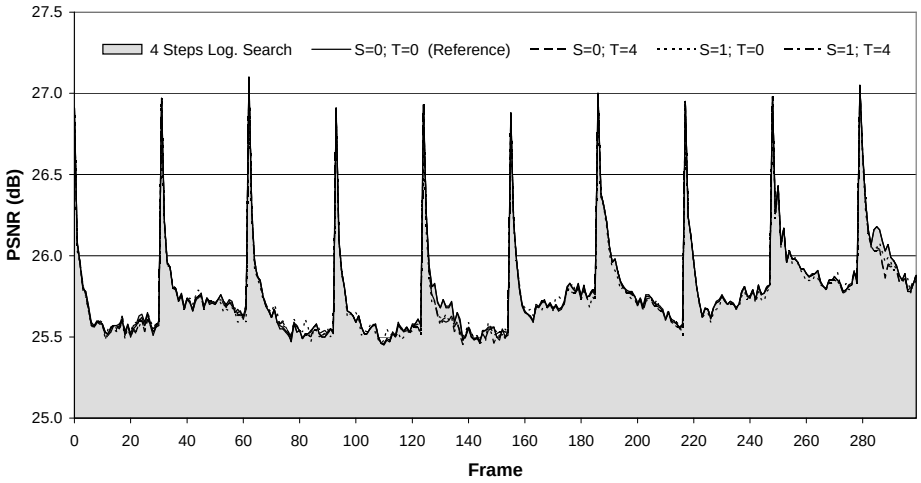
T		
0	2	4
76.1 MHz	77.8 MHz	79.8 MHz

Table 3. Variation of the maximum operating frequency with the number of truncated bits (T) $(N = 16; p = 16)$.

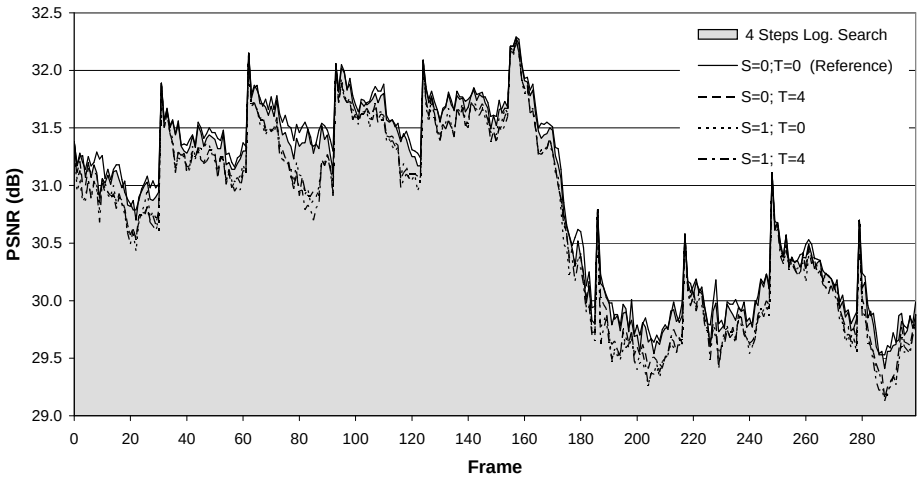
To assess and evaluate the efficiency of the synthesised processors in an implementation based on FPL devices, they were embedded as motion estimation co-processors in the H.263 video codec provided by Telenor R&D [11], by transferring the estimated motion vectors to the video coder. Peak signal-to-noise ratio (PSNR) and bit-rate measures were used to evaluate the performance of each architecture. These results were also compared with those obtained with a sub-optimal 4-steps logarithmic search algorithm [1], implemented in software.

The first 300 frames of several QCIF benchmark video sequences with different spatial detail and amount of movement were coded in interframe mode, by considering a GOP length of 30 frames and an intermediate quantisation step size of $\Delta = 30$ to keep the quantisation error as constant as possible.

In fig. 3 it is presented the obtained PSNR values for the *carphone* and *mobile* video sequences, characterised by the presence of high amount of movement and high spatial detail, respectively. Several different setups in what concerns the number of truncated bits (T) and the sub-sampling factor (S) for the decimation at the pixel level were considered. For comparison purposes, it was also presented the PSNR values obtained with the 4-steps logarithmic search algorithm. As it can be seen, the PSNR value for the INTER type frames of the *mobile* sequence is almost constant ($25-26$ dB). Hence, one can conclude that the degradation introduced by using the reduced hardware architectures is negligible: less than 0.15 dB when the 4 LSBs are truncated and the sub-sampling factor is $2:1$. Contrasting, the PSNR behaviour for the *carphone* sequence varies significantly along the time. The main reason for this fact is the amount of movement that is also varying. Even so, the reduced hardware architectures proposed in this paper only introduce a slightly degradation in the quality of the coded frames, when compared



(a) Mobile.



(b) Carphone.

Fig. 3. Comparison between the PSNR measure obtained for three different setups of the proposed reduced hardware architecture, for the original reference configuration and for the 4-steps logarithmic search algorithm.

with the performances of both the reference FSBM architecture ($S = T = 0$) and of the 4-steps logarithmic search algorithm. A maximum reduction of about 0.5 dB is observed in a few frames when the PSNR value obtained with the original (reference) architecture is greater than 30 dB.

As it was previously referred, these video quality results were obtained with a constant quantisation step size. The observed small decrease of the PSNR can be explained by the slight increase of the quantisation error, as a consequence of the inherent increase of the prediction differences in the motion compensated block, obtained with these sub-optimal matching processors. The obtained bit-rate required to store or transfer each

	T		
S	0	2	4
0	**31.9 kbps**	-0.8%	+3.4%
1	+1.7%	+3.0%	+3.8%
2	+3.9%	+3.8%	+3.8%
Four-step logarithmic search			+0.3%

(a) Miss America.

	T		
S	0	2	4
0	**49.9 kbps**	+0.0%	+2.7%
1	+4.5%	+4.1%	+8.0%
2	+10.7%	+8.4%	+8.6%
Four-step logarithmic search			+0.8%

(b) Silent.

	T		
S	0	2	4
0	**271.4 kbps**	+0.0%	+0.3%
1	+1.1%	+0.3%	+0.5%
2	+6.7%	+0.6%	+0.5%
Four-step logarithmic search			-0.1%

(c) Mobile.

	T		
S	0	2	4
0	**79.3 kbps**	-0.8%	+4.8%
1	+7.3%	+5.8%	+11.6%
2	+14.5%	+11.6%	+12.0%
Four-step logarithmic search			+1.7%

(d) Carphone.

Table 4. Variation of the output bit rate to encode the considered video sequences by using different setups of the proposed core-base architecture and the 4-steps logarithmic search algorithm.

coded video sequence is presented in table 4 for the two sequences considered above and for two more sequences (*Miss America* and *Silent*). The relative values presented in table 4 reveal the increment of the average bit-rate when the number of truncated bits and the sub-sampling factor increase. This increment reaches its maximum value of 12% for the *carphone* sequence and for a processor setup with only 4 representation bits and a pixel decimation of $2:1$, due to the presence of a lot of movement. Nevertheless, the obtained values for the other three considered video sequences with less amount of movement are quite smaller and are similar to those obtained with the 4-steps logarithmic search algorithm.

Consequently, one can conclude that the configuration using a $2:1$ decimation $(S=1)$ and using 4 representation bits $(T=4)$ presents the best compromise between hardware cost (a reduction of about 70%) and video quality. Moreover, this configuration could be implemented by using the lower-cost XCV1000 FPGA, which only has about 40% of the total number of system gates provided by the XCV3200 FPGA.

5 Conclusion

In this paper, new customisable core-based architectures are proposed to implement real-time motion estimation processors on FPGAs. The base core of these architectures is a new 2-D array structure for FSBM motion estimation that leads to an efficient usage of the hardware resources, which is a fundamental requisite in FPL systems. The proposed core-based architectures consist on a wide range of processing structures based on FSBM with different hardware requirements. The reduction of the amount

of required hardware is achieved by applying decimation at the pixel and quantisation levels, but still searching all candidate blocks of a given search area.

The proposed core-based architectures were implemented on FPGAs devices from Xilinx and their performance were evaluated by including the motion estimaton processors on a complete video encoding system. Experimental results were obtained by sub-sampling the block of the current frame with $2:1$ and $4:1$ decimation factors and by truncating 2 or 4 LSBs of the representation. From the obtained results it can be concluded that a significant reduction of the hardware resources is achieved with these architectures. Moreover, the quality of the coded video is not compromised and the corresponding bit-rate is not significantly increased. One can also conclude from the results that the configuration using a $2:1$ decimation $(S = 1)$ and using 4 representation bits $(T = 4)$ presents the best compromise between hardware cost (a reduction of about 70%) and video quality.

Acknowledgements

This work has been supported by the POSI program and the *Portuguese Foundation for Science and for Technology* (FCT) under the research project *Configurable and Optimized Processing Structures for Motion Estimation* (COSME) POSI/CHS/40877/2001.

References

1. Bhaskaran, V., Konstantinides, K.: Image and Video Compression Standards: Algorithms and Architectures. 2nd edn. Kluwer Academic Publishers (1997)
2. Koga, T., Iinuma, K., Hirano, A., Iijima, Y., Ishiguro, T.: Motion-compensated interframe coding for video conferencing. In: Proc. Nat. Telecomm. Conference, New Orleans, LA (1981) G5.3.1–G5.3.5
3. Jain, J.R., Jain, A.K.: Displacement measurement and its application in interframe image coding. IEEE Transactions on Communications **COM-29** (1981) 1799–1808
4. Roma, N., Sousa, L.: Efficient and configurable full search block matching processors. IEEE Transactions on Circuits and Systems for Video Technology **12** (2002) 1160–1167
5. Ooi, Y.: Motion estimation system design. In Parhi, K.K., Nishitani, T., eds.: Digital Signal Processing for Multimedia Systems. Marcel Dekker, Inc (1999) 299–327
6. Vos, L., Stegherr, M.: Parameterizable VLSI architectures for the full-search block-matching algorithm. IEEE Transactions on Circuits and Systems **36** (1989) 1309–1316
7. Liu, B., Zaccarin, A.: New fast algorithms for the estimation of block matching vectors. IEEE Transactions on Circuits and Systems for Video Technology **3** (1993) 148–157
8. Ogura, E., Ikenaga, Y., Iida, Y., Hosoya, Y., Takashima, M., Yamash, K.: A cost effective motion estimation processor LSI using a simple and efficient algorithm. In: Proceedings of International Conference on Consumer Electronics - ICCE. (1995) 248–249
9. Lee, S., Kim, J.M., , Chae, S.I.: New motion estimation algorithm using adaptively-quantized low bit resolution image and its vlsi architecture for MPEG2 video coding. IEEE Transactions on Circuits and Systems for Video Technology **8** (1998) 734–744
10. He, Z.L., Chan, K.K., Tsui, C.Y., Liou, M.L.: Low power motion estimation design using adaptative pixel truncation. In: Proceedings of the 1997 international symposium on Low power electronics and design, Monterey - USA (1997) 167–171
11. Telenor Research Norway: TMN (H.263) encoder/decoder, version 2.0. (1996)