# An ASIP Approach for Adaptive AVC Motion Estimation

Svetislav Momcilovic, Nuno Roma and Leonel Sousa
INESC-ID/IST, TULisbon
Rua Alves Redol 9, 1000-029, Lisboa, PORTUGAL
Email:{Svetislav.Momcilovic,Nuno.Roma,Leonel.Sousa}@inesc-id.pt

***Abstract*— A new algorithm and an adapted hardware architecture of an ASIP are proposed in this paper. When compared with other hardware ASIP implementations, this architecture significantly speeds up the motion estimation procedure and substantially decreases the memory requirements. Moreover, it also makes use of significantly fewer memory accesses, still maintaining its coding quality performances in what concerns both the obtained bit rate and PSNR. As a consequence, the proposed algorithm proves to be specially adequate to be implemented in most embedded systems with restricted computational and power resources that are often adopted by portable and battery supplied devices.**

## I. Introduction

Motion Estimation (ME) is one of the most important and computationally expensive parts of a video encoder and it is used to reduce the temporal redundancy between adjacent frames in video sequences. One of the most adopted ME techniques is Block Matching (BM). This technique divides each frame in square shaped blocks, called Macroblocks (MBs), and treats them as basic ME units.

Although the newly introduced techniques proposed by the H.264/AVC coding standard (such as Rate-Distortion Optimization (RDO), variable MB sizes and multiple reference frames), highly improve the coding quality, they also drastically increase the involved computational load. This makes the implementation of ME algorithm even more important, since it may represent up to 80% of the whole set of computations.

The ME algorithms developed for the H.264/AVC coding standard are mainly based on two approaches. One of them computes the best matching candidate by guiding the search procedure using predefined search patterns. One of such fast algorithms is the Hybrid Unsymmetrical-cross Multi-Hexagon-grid Search (UMHexagonS) [1], which makes use of a complex pattern structure based on four different search patterns. Although such fast algorithms are somewhat suitable for hardware implementations, they still require a significant amount of computations. As a consequence, fast ME algorithms are usually combined with adaptive schemes, making use of prediction and early stopping techniques. These techniques try to predict the starting point that should be used by the search procedure and skip unnecessary search points as soon as the matching result is good enough. An adapted variant of the Enhanced Predictive Zonal Search (EPZS) [2] is usually adopted in the H.264/AVC. This algorithm has shown to be very robust, by using a significant number of predictors and a very complex pattern structure. However, its memory and computational requirements are quite high, which makes it difficult to be implemented in real time.

In contrast, in [3] it was recently proposed an alternative low memory and computational reduced adaptive ME algorithm that still keeps the simplicity of pattern based algorithms and the effectiveness of adaptive approaches.

Meanwhile, most of the hardware implementations of ME algorithms that have been presented for the AVC coding standard are core oriented to full search or UMHexagonS. Furthermore, not only are such proposals focused to a specific algorithm or coding standard, but it is often not known the coding and quality performances of such hardware architectures. They also often do not provide the possibility to change the algorithm parameters in order to adjust the tradeoff between frequency and memory requirements on the one side and coding quality on the other.

In this paper, a new Application Specific Instruction Set Processor (ASIP) is presented. This ASIP not only provides high adaptiveness in what concerns the coding parameters, but it also offers the possibility to effectively implement different adaptive algorithms. This processor is an adapted version of the H.263 oriented ASIP proposed in [4]. It provides high quality performances, even with a slight quality decrease, when compared with the full EPZS algorithm, whose complexity tends to go greatly beyond the real time coding capability offered by any available architecture (when 5 reference frames and all block sizes are considered). The operating frequencies required by the proposed processor for high encoding quality are not higher than 200-250 MHz.

The rest of the paper is organized as follows. In section II the proposed adaptive ME algorithm is formulated. Section III presents the architecture of the developed ASIP. Section IV presents the experimental results and section V concludes the paper.

## II. Proposed Algorithm

In Fig. 1 it is depicted the proposed adaptive ME algorithm [3]. This algorithm not only exploits the spatial and temporal redundancies through prediction techniques, but it also applies adaptive search patterns and early-stopping thresholding.

The proposed adaptive ME algorithm is the result of the combined effect of three different techniques:

1-4244-1001-0/07/$25.00 ©2007 IEEE.

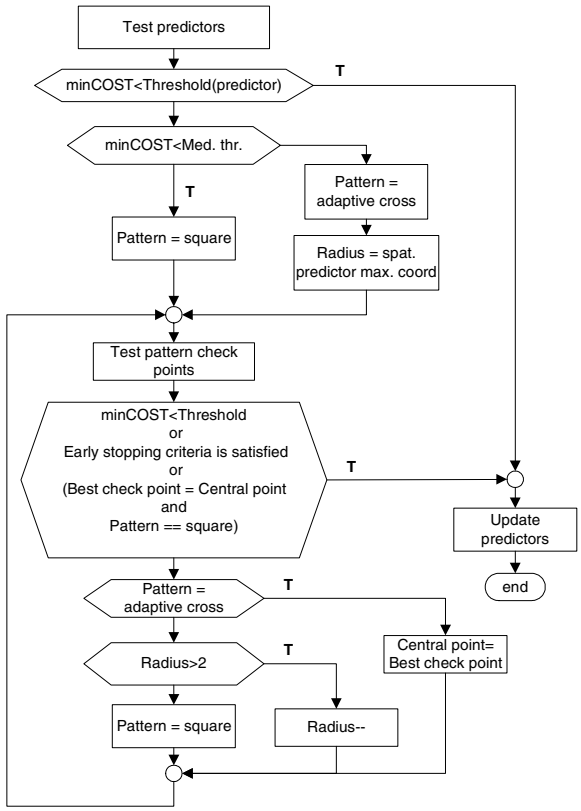


Fig. 1. Proposed adaptive ME algorithm.

*i) Prediction*, whose main purpose is the selection of the best starting search point, in order to accelerate the search procedure for the best matching MB. The set of predictors used by the proposed algorithm are chosen from a restricted set of often used predictors, in order to keep the memory requirements as low as possible. Among these predictors, the most influent set are the spatial predictors, namely, the left, the upper and the upper-left neighbor MB Motion Vectors (MVs), as well as their median value and zero predictor. This set is still enhanced with spatial predictors that are computed from ME within the sub-blocks. These MVs correspond to the best ME cost values within the same $16 \times 16$ pixels area. If the computed minimum cost was lower than a pre-defined threshold, then the predictors are updated and the search process is finished.

*ii) Adaptive search patterns*, in order to further optimize the search procedure. The proposed algorithm makes use of two patterns that apply different strategies. The first one is based on an adaptive cross pattern with a simple modification, so that the displacement of each candidate MB is multiplied by an adaptive radius that is obtained as the biggest value among all the predictor coordinates. The second pattern, which may either be applied alone or in conjunction with the first one, depending on the radius value, is a simple $3 \times 3$ square pattern. With the described strategy, if the minimal distortion measure, obtained by examining all the considered predictors, is less than the threshold that was defined for the median predictor, only one square pattern is applied. Otherwise, the adaptive cross pattern is used first, by decreasing its radius in each performed search step until a radius equal to 2 pixels is reached. At such instant, the search pattern is switched to a square.

*iii) Threshold*, whose aim is to stop the search as soon as the obtained result is good enough and it is not useful to prosecute with it. The threshold value is based on the best ME cost obtained within the same $16 \times 16$ pixels area. For the first MB within this area, it is used the value from the previous $16 \times 16$ pixels area, while for the median predictor, a specific increased threshold is adopted.

As it will be shown in the following, the combination of these techniques will provide a substantial reduction of the involved computations and memory accesses, without introducing any significant quality loss in the output video stream.

## III. ASIP INSTRUCTION-SET AND MICRO-ARCHITECTURE

### A. Instruction Set

The Instruction Set Architecture (ISA) of the proposed ASIP was designed to meet the set of requirements usually presented by adaptive ME algorithms and was specially optimized to be used in portable and mobile platforms, where power consumption and implementation area are mandatory constraints. Consequently, it is based on a register-register architecture and is composed by only eight instructions. These instructions were carefully chosen among the set of operations that are more often used in standard ME algorithms. This register-register approach was adopted due to its simplicity and efficiency, allowing the design of simpler and less hardware consuming circuits. The register file consists of sixteen 16-bit General Purpose Registers (GPRs) and eight Special Purpose Registers (SPRs).

The instructions provided by the proposed ISA are grouped in four different categories, as it can be seen from Table I, and were obtained as the result of an analysis of the execution of several different ME algorithms:

*i) Control instruction*: the jump instruction `J` directly changes the value of the program counter, providing the capability to instantaneously change the control-flow of a program.

*ii) Arithmetic instructions*: provide the standard arithmetic instructions, namely `ADD`, `SUB` and `DIV2` instructions, that are mainly used to calculate the memory addresses and to compare the ME cost.

TABLE I

INSTRUCTION-SET ARCHITECTURE OF THE PROPOSED ASIP.

| **Instr.** | **15 13** | **12** | **11 8** | **7 5** | **4 3** | **0** |
|---|---|---|---|---|---|---|
| LD | 000 | t | - | | | |
| J | 001 | cc | - | #addr | | |
| MOVR | 010 | Rd | | - | Rs | |
| MOVC | 011 | t | Rd | #const | | |
| SAD16 | 100 | - | Rd | Rs1 | Rs2 | |
| DIV2 | 101 | - | Rd | Rs | - | |
| ADD | 110 | - | Rd | Rs1 | Rs2 | |
| SUB | 111 | - | Rd | Rs1 | Rs2 | |

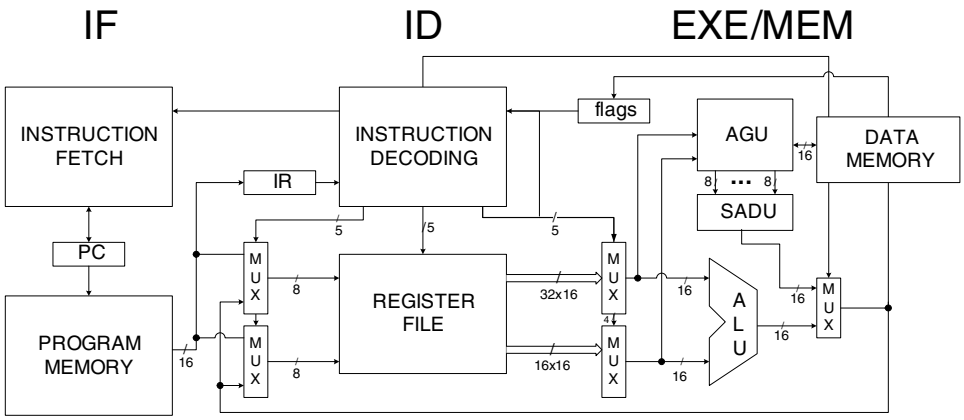


Fig. 2. Architecture of the proposed ASIP.

*iii) Graphics instruction*: the `SAD16` instruction allows the computation of the Sum of Absolute Differences (SAD) distortion measure between a MB and a candidate block. This operation computes the SAD value considering two sets of 16 pixels and accumulates the result in a GPR. Therefore, the computation of the SAD value for a given candidate requires the execution of several consecutive `SAD16` operations. To further improve the algorithm efficiency and to reduce the program size, both the horizontal and vertical coordinates of the set of pixels of the candidate block under processing are also updated with the execution of this operation. An additional contribution of this instruction to improve the processing efficiency of an AVC encoder is oriented to the computation of the required pixel addresses. Hence, the whole set of pixels belonging to a $4 \times 4$ MB are processed in a single `SAD16` operation, while 8 `SAD16` instructions are required for $16 \times 8$ or $8 \times 16$ pixels blocks, etc.

*iv) Memory data transfer instruction*: the processor comprises a small and fast local memory that stores the pixels of a MB and of its corresponding search area, and includes a memory data transfer operation, `LD`, that loads the data into this local memory. This loading of the data concerning a MB and the corresponding search area is performed independently. As a result, the `LD` operation has a 1-bit control field to specify the type of image area to be loaded into the local memory.

The encoding of the instructions into binary representation was performed using 16-bit and a fixed format. For each instruction it is specified an opcode and up to three operands, depending on the instruction category.

### B. Micro-architecture

The proposed ISA is supported by a specially designed micro-architecture, following strict power and area driven policies to support its implementation in portable and mobile platforms. This micro-architecture presents a modular structure and it is composed by simple and efficient units to optimize the data processing, as it can be seen from Fig. 2.

*1) Control unit:* The control unit is characterized by its low complexity, due to the adopted fixed instruction encoding format and a careful selection of the opcodes for each instruction. This leads to the implementation of a very simple and fast hardwired decoding unit, which enables almost all instructions to complete in just one clock cycle. To minimize the power consumption, the switching activity at the function unit level is strictly controlled, by inhibiting input updates to functional units whose outputs are not required for a given operation. Moreover, the operating frequency is also adjusted according to the programmed algorithm and the current available energy level.

*2) Datapath:* For the most complex and specific operations, such as the `LD` and `SAD16` instructions, the datapath also includes specialized units to improve the efficiency of these operations: the Address Generation Unit (AGU) and the SAD Unit (SADU).

The `LD` operation is executed by a dedicated AGU optimized for ME, which is capable of fetching all the pixels corresponding to a MB and its entire search area. To maximize the efficiency of the data processing, this unit can work in parallel with the remaining functional units of the micro-architecture. Using such feature, programs can be optimized by rescheduling the `LD` instructions to allow data fetching from memory to occur simultaneously with the execution of other parts of the program that do not depend on this data. This not only significantly reduces the memory traffic to the external memory, but also provides a considerable reduction in the power consumption of the video encoding system.

The SADU can execute the `SAD16` operation in up to sixteen clock cycles and it uses the Arithmetic and Logic Unit (ALU) to update the coordinates of the line of pixels of the candidate block. The number of clock cycles required for the computation of a SAD value is imposed by the type of architecture of this unit, which depends on the power consumption and implementation area constraints specified at design time.

## IV. Experimental results

To access the performance of the proposed processor, an accurate cycle based simulator of the proposed ASIP was first realized, preceding its actual hardware implementation [4].

TABLE II

FREQUENCY REQUIREMENTS OF THE PROPOSED ESTIMATOR FOR REAL-TIME CODING OF DIFFERENT CIF VIDEO SEQUENCES [MHZ].

| MB sizes | 16x16, 8x8, 4x4 | | | not smaller then 8x8 | | | all MB sizes | | |
|---|---|---|---|---|---|---|---|---|---|
| ref. frames | 2 | 3 | 5 | 2 | 3 | 5 | 1 | 2 | 5 |
| Foreman | 200 | 261 | 359 | 134 | 175 | 237 | 223 | 337 | 602 |
| Mobile | 194 | 257 | 347 | 125 | 167 | 223 | 217 | 330 | 602 |
| Bus | 167 | 217 | 293 | 90 | 118 | 163 | 184 | 287 | 520 |

TABLE III

PSNR COMPARISON OF THE PROPOSED ESTIMATOR AND THE EPZS SOFTWARE IMPLEMENTATION [DB].

| sequences | MB sizes | 16x16, 8x8, 4x4 | | | not smaller then 8x8 | | | all MB sizes | | |
|---|---|---|---|---|---|---|---|---|---|---|
| | ref. frames | 2 | 3 | 5 | 2 | 3 | 5 | 1 | 2 | 5 |
| Foreman | Prop. | 36.67 | 36.70 | 36.69 | 36.68 | 36.70 | 36.68 | 36.62 | 36.73 | 36.74 |
| | EPZS | 36.98 | 37.00 | 37.03 | 36.99 | 37.01 | 37.04 | 36.93 | 37.05 | 37.09 |
| Mobile | Prop. | 34.15 | 34.12 | 34.18 | 34.08 | 34.09 | 34.04 | 33.92 | 34.17 | 34.19 |
| | EPZS | 34.20 | 34.28 | 34.30 | 34.16 | 34.24 | 34.27 | 34.03 | 34.24 | 34.37 |
| Bus | Prop. | 35.02 | 35.03 | 35.02 | 35.02 | 35.01 | 34.99 | 34.98 | 35.03 | 35.01 |
| | EPZS | 35.13 | 35.12 | 35.12 | 35.11 | 35.12 | 35.12 | 35.06 | 35.19 | 35.21 |

In this paper, the simulator was integrated within the JVT reference software of an H.264/AVC video encoder [5].

In Table II it is presented the set of results regarding the frequency requirements for real-time encoding using the proposed algorithm. The encoder was tested for a wide set of video sequences and encoding parameters, considering a fixed quantization factor and 30 fps. The presented values denote worst case results, obtained for the most demanding test sequences, and can be regarded as maximal encoding requirements. The advantages of this ASIP approach were extensively exploited and further proved, by changing the estimator parameters and obtaining the corresponding results. From the presented results, it can be seen that with frequencies not higher than 200MHz, and by carefully choosing the encoder parameters, high quality coding can be obtained.

For comparison purposes, in Table III it is presented the Peak Signal Noise Ratio (PSNR) quality measures obtained with the proposed estimator and with the EPZS algorithm. The slight coding quality degradation that can be observed may be explained by the restrictions imposed by the hardware implementation, leading to the usage of simplified arithmetic operations and restricted predictor sets, with the aim of decreasing the memory requirements.

In Table IV it is presented the comparative results concerning the memory requirements for the considered algorithms ($w$ and $h$ denote the number of MBs in each direction and $r$ represents the number of reference frames). It was not considered the amount of memory required to store the MVs (2 bytes each) and for the ME cost values (3 bytes each). Similarly, the memory required for the search area was also not considered, because hardware and software concepts are significantly different. In what concerns the EPZS algorithm, it was considered the amount of memory required to apply the temporal predictors, as well as the spatial predictors, whose exact size depends on the number of reference frames. From this table it can be seen that the proposed algorithm significantly reduces the required memory space, making it specially suited to be implemented by hardware motion estimators. As an example, for the QCIF format the usual homologous temporal and spatial predictors occupy almost 10 times more memory than those used by the proposed algorithm. In fact, the memory requirements of the EPZS algorithm greatly supersedes the restrictions of most hardware implementations, and proves the need for the usage of hardware oriented adaptive algorithms, as the one that is proposed here.

TABLE IV

REQUIRED MEMORY SPACE.

| Algorithm | EPZS | Proposed |
|---|---|---|
| Mem [Bytes] | $2w(h \cdot r + 1)$ | $15 + 2w$ |

## V. CONCLUSIONS

A new adaptive ME algorithm was proposed. This algorithm significantly speeds up the ME procedure and substantially decreases the memory requirements, when compared with other fast search approaches, such as the H.264/AVC oriented EPZS algorithm, still providing similar coding quality and bit-rate performances. As a consequence, it proved to be specially adequate to be implemented in most embedded systems with restricted computational and power resources that are often adopted by portable and battery supplied devices.

## REFERENCES


[1] "Fast integer pel and fractional pel motion estimation for AVC," in *Joint Video Team (JVT) of ISO/IEC MPEG and ITU-T VCEG, JVT-F016*, Dec. 2002.

[2] H.-Y. Tourapis and A. Tourapis, "Fast motion estimation within the H.264 codec," in *Int. Conf. on Multimedia and Expo, (ICME'03)*, vol. 3, July 2003, pp. 517–520.

[3] S. Momcilovic, N. Roma, and L. Sousa, "Adaptive motion estimation algorithm for H.264/AVC," in *15th International Conference on Digital Signal Processing (DSP'07)*. Cardiff, Wales - UK: IEEE, July 2007.

[4] T. Dias, S. Momcilovic, N. Roma, and L. Sousa, "Adaptive motion estimator for autonomous video devices," *EURASIP Journal on Embedded Systems, special issue on Embedded Systems for Portable and Mobile Video Platforms*, 2007.

[5] *JVT Reference Software unofficial version 12.0*, http://iphome.hhi.de/suehring/tml/download.