A VLSI Scalable Processor Array for Motion Estimation

P. Baglietto, M. Maresca, A. Migliaro and M. Migliardi

DIST - University of Genoa
via Opera Pia 13 - 16145 Genova
tel +39-10-353.2983 fax +39-10-353.2948
e-mail prp@dist.unige.it

Abstract - In this paper we describe a parallel architecture for motion estimation based on the Full Search Block Matching Algorithm. The distinctive characteristic of the proposed architecture is its suitability to be implemented both in a high performance dedicated device for embedded systems (e.g. an ASIC) and on mesh connected SIMD massively parallel computers. The paper describes the first of these options in detail.

Keywords: MPEG, Motion Estimation, Processor Array, VLSI.

1 Introduction

Motion estimation is one of the issues in full motion video compression [1][2]. Motion estimation allows to reduce the amount of information needed to represent a frame in a video sequence by eliminating the temporal redundancy due to the fact that a frame is usually very similar to the previous ones.

A video coder implements motion estimation by encoding the current frame with respect to one of the preceding frames in the video sequence taken as a reference frame. The current frame is represented as a combination of the estimation function to be applied to the reference frame and of the error resulting from the application of such a function to the reference frame. The frame resulting from the application of the estimation function to the reference frame, called estimated frame, should be as similar as possible to the current frame and the error is the difference between the estimated frame and the current frame.

The computation of the estimation function is called motion estimation. The motion estimation in the MPEG and H.261 standards is computed following the Block Matching Algorithm (BMA) which consists of dividing the current frame into fixed size blocks of pixels and for each block finding the most similar block of pixels in the reference frame. As a consequence, each block of the current frame is estimated by means of a motion vector representing the difference between the position of the block in the current frame and the position of the corresponding block in the reference frame. The set of motion vectors associated to the blocks of the current frame are the estimation function.

As the amount of computation requested to identify the motion vectors is large and the degree of parallelism that can be exploited appears to be high, special purpose architectures for motion estimation have been proposed and are currently available [3][4][5].

In this paper we describe a metodology that allows to efficiently implement the Full Search BMA (FSBMA) in parallel both in a high performance dedicated device for embedded systems (tipically ASIC) and on mesh connected SIMD massively parallel computers (e.g. the MasPar MP-1).

The FSBMA for motion estimation presents three levels of potential parallelism:

1. at the frame level the computation of the motion vector of each block is totally independent from the computation of the motion vectors of the other blocks;
2. at the block level the computation of the error associated to each possible motion vector is totally independent from the computation of the errors associated to other possible motion vectors;
3. at the error level all the differences to be computed (see next section for details) can be carried out in parallel;

The implementation described in this paper, at the ASIC level, exploits the parallelism at the block level while an implementation on SIMD massively parallel computers may in addition exploit the parallelism at the frame level to reach higher performance. Parallelism at the error level cannot be easily exploited, especially in ASICs, as it requires the use of parallel adders, and gives less performance improvements compared to the other two types of parallelism. The available ICs for motion estimation prevalently exploit parallelism at the block level [3][4]. The VLSI implementation of our architecture follows the same technique and improves the performance of these ICs.

The paper is organized as follows. In the next section we present the BMA algorithm and a parallelization technique suitable for implementation both in a SIMD massively parallle computer and in VLSI. In section 3 we present an abstract parallel architecture supporting such a parallelisation technique and in section 4 we describe a VLSI ASIC implementing such an abstract architecture and discuss its performance. Finally we give some concluding remarks.

2 The parallel algorithm for full block matching

The BMA estimates the amount of motion on a block by block basis. In a tipical BMA, a frame on $N \times M$ pixels is divided into blocks of $n \times n$ pixels (in MPEG and H.261 standard $n=16$). Each block of pixels of the frame, called reference block, is compared with a set of blocks of a previous frame, called candidate blocks, and the candidate block most similar to the reference block is identified. The blocks in the set are not taken from the whole frame, but, on the contrary, they are taken from a search area of size $(n+2p)\times(n+2p)$ around the position of the reference block, where p is the maximum displacement (see fig. 1). The motion vector of a reference block corresponds to the displacement of the best matching candidate block.

Among the possible search methods, the Full Search BMA, which searches all possible candidate blocks in a search area, is the optimal solution and has the lowest control overhead. This last feature makes the FSMBA easy to be implemented in a SIMD processor array and highly suitable for VLSI implementation.

Among the possible matching criteria the Mean Absolute Difference (MAD) is usually preferred for VLSI implemetation because it requires the repetition of simple operations. The MAD is defined as follows:

$$MAD(u, v) = \sum_{i=1}^{n} \sum_{j=1}^{n} \left| S(i+u, j+v) - R(i, j) \right|$$

where R(i, j) is a pixel of the reference block and S(i+u, j+v) is the corresponding pixel of the candidate block displaced of (u, v), which is also the corresponding motion vector. The motion vector associated with the reference block is the motion vector corresponding to the least MAD. According to this definition of the FBMA, in each search area there are $(2p+1)^2$ candidate blocks and the computation of the MAD of a candidate block requires n^2 subtractions, n^2 absolute value operations and n^2 accumulations. A total of $3n^2 \times (2p+1)^2$ operations for each reference block.

In order to be able to perform the motion estimation in real-time, a parallel architecture is required. The architecture we propose in the paper is based on the parallelization of the MAD calculation for a single reference block over the $(2p+1)^2$ possible displacements, taking advantage of the fact that the computation of the MADs corresponding to the $(2p+1)^2$ displacements are independent from each other. The same technique can be implemented on a massively parallel computer taking advantage also of the parallelization over the blocks, as the MADs of all the blocks can be calculated in parallel.

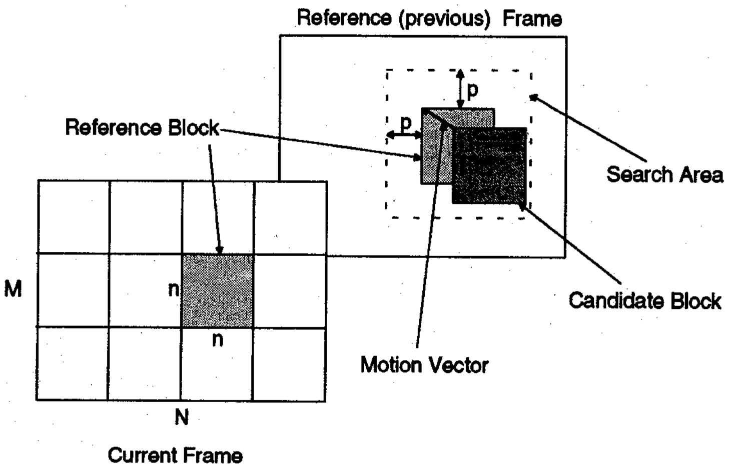

Figure 1 - Block matching for motion estimation.

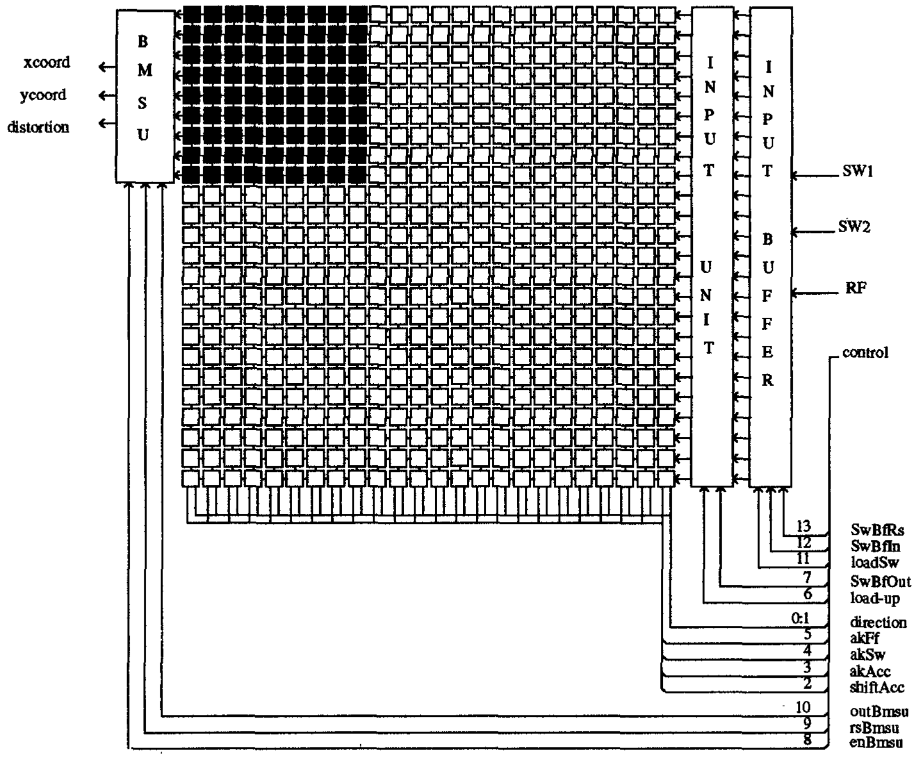

Figure 2 - Processor array architecture.

3 The processor array architecture

The parallel architecture based on the technique described in the previous section is basically a mesh of Processing Elements (PEs) and of Memory Elements (MEs) which are respectively represented as black and white boxes in figure 2. The PEs compute the MADs for each possible motion vector and the MEs to store all the pixels of the search area including those pixels which are not used at a given processing step.

A single PE is composed of a 8-bit register to store the current pixel of the search area, a 8-bit adder, a 1-complement unit for absolute value computation, a 16-bit adder for accumulation of the result in a 16-bit register. At each clock cycle the three following operations are executed:

- the value of a pixel of the reference block is broadcast to all the PEs;
- all the PEs simultaneously compute the absolute value of the difference between the pixel of the reference block received and the pixel of the search area stored in their 8-bit register; then they accumulate the result;
- the mesh of registers (which include the 8-bit registers inside each PE) shift its content along one of the four directions (NEWS).

The pixels of the search area (contained in the 8-bit registers, i.e. the ME and the 8-bits registers of the PEs) shift over the mesh of PEs according a snake shaped path, while the values of the pixels of the reference block are sequentially broadcast to all the PEs. At each clock cycle the $(2p+1)^2$ PEs compute the difference between a pixel of the reference block and all the pixels of the search area onto which the reference block pixel can be mappedi n parallel. Then they compute the absolute values of the results and accumulate them in the 16-bit registers.

A total of $n^2+4p(n\text{-}1)\text{-}1$ registers are needed to store the pixels of the search area in addition to the $(2p+1)^2$ PEs. A total of n^2 clock cycles are required to compute the MADs while a few more clock cycles are needed to extract the MADs from the PEs, to compare them and to initialize the mesh with a new search area.

In addition to the processor and register array, the architecture is composed of:

- a Best Match Selection Unit compares all the MADs computed by the PEs and extracts the minimum among them along with the associated motion vector;
- an Input Buffer Unit which allows to overlap the input of the next search area with the computation of the MADs for the current search area;
- an Input Unit which distributes the pixels stored in the Input Buffer according to the pattern required by the current step of computation.

4 The VLSI implementation

The VLSI implementation of the architecture defined in the previous section has been realized with the Epoch Silicon Compiler using a standard cell technology. The characteristics of the chip are:

Technology:	0.5 μ, 3 metals, 3V	clock speed:	50 MHz
Package:	PLCC 84 pins	block rate	183000 blocks/sec
Core area:	8.8×7.3 mm.	on chip memory	1062 Bytes
# of PEs:	81	# of MOPS:	12150

The chip was designed for the special case of blocks of size n=16 (as reccomended by the MPEG and H.261 standards) and a search area of size p=4. As a consequence the ASIC hosts a mesh of 576 elements (81 PEs and 495 MEs) and is able to deliver the motion vector associated to a reference block every 273 clock cycles (256 cycles for MADs computation, 16 cycles for best match selection and initialization with the next search area pixels, 1 cycle for input buffer reset). However the chip can be used with blocks and search areas of different size.

The chip can operate at a maximum clock frequency of 50 MHz which corresponds to a maximum block rate of 183 Kblock/sec and to a maximum pixel rate of 46,9 Mpixels/sec.

The use of this chip as a part of a board, or more in general of an embedded system, is very flexible as can be seen from the folowing table which lists the number of chips and the clock speed required to process various video streams at two level of accuracy (video streams at a rate of 30 frame/sec., blocks of size n=16, search area of size p=4 and p=8):

Frame size	# of blocks	Search Area 24×24 pixels	Search Area 32×32 pixels
640×480	36.000	1 ASIC at 10 MHz	1 ASIC at 30 Mhz with time sharing
800×600	57.000	1 ASIC at 16 MHz	1 ASIC at 48 Mhz with time sharing
1280×1024	153.600	1 ASIC at 42 MHz	4 ASIC at 42 Mhz in parallel
1500×1024	180.480	1 ASIC at 50 MHz	4 ASIC at 50 Mhz in parallel

5 Concluding Remarks

We have presented a general approach to the problem of the parallelization of the Full Search Block Matching Algorithm for motion estimation suitable to be adopted both in SIMD massively parallel computers and in ASIC. In particular we have discussed an ASIC which can be used in embedded systems to compress video sequences according the MPEG and H.261 standards since the degree of parallelism exploited allows to obtain real time performance even at a low clock rate. The paper demonstrates that the VLSI implementation of the proposed approach is feasible and convenient using available technology. Finally the proposed approach can be used both in emebdded systems and in general purpose computers since its performance is fully scalable and proportional to the size of the parallel system used.

References

[1] *Video codec for audio visual services at p× 64 kb/s*, CCITT Reccomendation H.261, 1990.

[2] *Coding of moving pictures and associated audio*, Committee Draft of standard ISO11172: ISO/MPEG/90/176, Dec. 1991.

[3] P. Ruetz, P. Tong, D. Bailey, D. Luthi and P. Ang, *A high performance full-motion video compression chip set*, IEEE Trans. on Circuits and systems for video technology, Vol. 2, N. 2, June 1992, pp. 111-122.

[4] H. Fujiwara, M. Liou, M. Sun, K. Yang, M. Maruyama, K. Shomura and K. Ohyama, *An all-ASIC implementation of a low bit-rate video codec*, IEEE Trans. on Circuits and systems for video technology, Vol. 2, N. 2, June 1992, pp. 123-134.

[5] C. Hsieh and T. Lin, *VLSI architecture for block-matching motion estimation algorithm*, IEEE Trans. on Circuits and systems for video technology, Vol. 2, N. 2, June 1992, pp. 169-175.