A PIPELINED SYSTOLIC ARCHITECTURE FOR THE HARDWARE ORIENTED REGION BASED MOTION ESTIMATION ALGORITHM

Andrea Fermo and Giovanni L. Sicuranza
Università degli studi di Trieste
DEEI
Via Valerio 10, 34127 Trieste, Italy
fermo@ipl.univ.trieste.it

Vojko Pahor
Telit Mobile Terminals S.p.A.
V.le Stazione di Prosecco 5/B
34014 Sgonico, Trieste, Italy
vojko.pahor@telital.com

ABSTRACT

Motion estimation is a fundamental step for high quality, low bandwidth video compression. Recently the MPEG-4 group has proposed some low complexity algorithms. They have almost the same performance in term of PSNR of the Full Search algorithm, but at the same time the complexity is dramatically reduced. However it is difficult to realize these algorithms with conventional hardware structures. For this reason we presented a Hardware Oriented Region Based algorithm (HORB) with similar performances, but that can be implemented with a simple hardware structure. Here we present an architecture tailored to meet the design constraint of the HORB algorithm (and at the same time capable of realizing the Full Search algorithm).

1 INTRODUCTION

Mobile video communication and in particular videophone will be enabled by the third generation mobile networks and terminals. The videophone will require a digital video encoder integrated on the mobile device. The digital video encoder must meet two fundamental constraints: low bit rate (because of the limited bandwidth of the radio channel) and low power (in order to save the battery life of the device). Both of them can be accomplished by accurately designing the motion estimation step. On one hand, it may take up to 80% of the total computational power of the encoder and, on the other hand, it has to guarantee the achievement of a good visual quality. Full Search (FS) technique is capable of good performances in terms of video quality and bit rate efficiency, but it leads to a high computational complexity that is not compatible with a real-time low-power application. Recently, some new low complexity algorithms have been proposed [1], [2]. They have almost the same performance in terms of PSNR of FS algorithm, but at the same time their complexity is dramatically reduced. However, it is difficult to implement these algorithms with conventional hardware structures. On the contrary, to meet the intensive computation demands, developing custom hardware gives superior performances to general purpose microprocessor implementation. For this reason we have presented [3] a hardware oriented algorithm that has very good performances, and is realizable with a simple hardware structure. Through a joint optimization of the motion estimation algorithm and the related architecture, the power consumption can be further lowered. For this reason, here we present an architecture tailored to meet the design constraint of the HORB algorithm.
In Sections 2 and 3 we briefly review known motion estimation algorithms and their hardware architectures. In Sections 4 and 5 we present our algorithm and the architecture we propose to implement this algorithm. In Section 6 we present some simulation results and finally in Section 7 our conclusions.

2 MOTION ESTIMATION ALGORITHMS

The motion estimation task is accomplished by subdividing the current frame *X* into *M* macroblocks (MBx) and by minimizing the Sum of Absolute Differences (SAD) between the pixels of each MBx with those of the MBy belonging to the previous frame *Y*. The SAD of the m^{th} MBx of size 16×16 located at (i_0, j_0) in the current frame, compared to a MBy located at displacement of (h, k) relative to MBx is defined as:

$$SAD(h,k) = \sum_{i=0}^{15} SAD(h,k)_{line}(i) \tag{1}$$

$$SAD(h,k)_{line}(i) = \sum_{j=0}^{15} |X_{curr}(i_0+i, j_0+j) - \\ -Y_{prev}(i_0+i+h, j_0+j+k)| \tag{2}$$

Usually the investigation is made over a bound limited search area (usually $[-16, 15]$). At every point (h, k) of this area corresponds a motion vector (MV) that indicates the displacement with respect to the actual MB. The FS method exhaustively evaluates all possible points within a search range in the previous image frame in order to find the MBy with the minimum SAD value ($SAD_{min}(m)$). Although yielding an optimal performance, this method has an enormous computational complexity, which could take up to 80% of the total computational power of the encoder. For this reason some low complexity algorithms have been proposed in literature, such as Three Step Search (TSS) [4], Hierarchical Search [5], Block-Based Gradient Descent Search (BBGS) [6], but they lead to a visual quality significantly lower than FS.

Recently some new algorithms had been proposed in literature, that lower significantly the computational complexity even keeping the good visual quality. In particular, Advanced Predictive Diamond Zonal Search (APDZS) [1] starts the search in the center of the search area and then moves along a spiral path, stopping the search when the result is good enough, by implementing a threshold test after every SAD calculation. In Motion Vector Field Adaptive Fast Motion Estimation (MVFAST) [2] instead, the search center is chosen based on some local motion activity measures and then a local search is performed around the search center to obtain the motion vector for the current MB. The implementation of these two algorithms however is difficult because of their lack of data regularity and because of the complexity of the control structure.

3 HARDWARE ARCHITECTURES

To achieve high throughput, a high number of operations must be carried out simultaneously. This can be reached through massive use of parallel processing and pipelining. Array processors are computational networks with distributed data storage and distributed Processing Elements (PEs). Systolic arrays are array processors with synchronous clocking and synchronous control signals. Systolic arrays for a given algorithm can be derived by the methodology proposed by Kung [7]. In [8] the fundamentals of 1D and 2D systolic arrays are presented. In [9] the reuse of data is enlarged by the use of a large set of shift registers in order to limit the *I/O* bandwidth. In [10] this result is achieved with a more sophisticated scheme that prevents the pipeline stalls. [11] presents an architecture that eliminates unnecessary computations by suspending the activity of processing elements during the computation of distortion values, as soon these values become greater than the minimum distortion calculated so far. Such ideas are the basis of the architecture described in this paper.

4 HORB ALGORITHM

3	3	3	3	3
3	2	2	2	3
3	2	1	2	3
3	2	2	2	3
3	3	3	3	3

Figure 1: Subdivision of the search area in 25 regions (the number represents the group to which each region belongs).

We have proposed in [3] a Hardware Oriented Region Based algorithm (HORB) based on the same assumptions that have led to APDZS. Nevertheless we introduced some important modifications in order to make it easily realizable by a systolic array approach, without affecting the complexity reduction in terms of SAD points calculated.

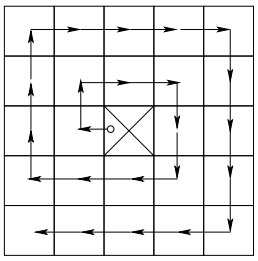

Figure 2: Search path among the regions.

4.1 No-Motion Detection

First of all, a pixel by pixel comparison is made between the actual MBx and the central MBy of the search area. The comparison is made considering only the first 5 Most Significant Bits (MSB) of every pixel in order to filter out the noise. During this comparison the pixels matching are counted. If the matching pixels are more than 70%, the MBx is chosen to be a stationary block and the search is halted.

4.2 Core of the HORB algorithm

In HORB algorithm, the entire search area is divided in 25 search regions of three groups as shown in Fig. 1. In every region (starting from the center region) an exhaustive search is made as a full search on a limited search area. For the first region, the algorithm equation is the same as in Eqs. (1,2), with $h, k = -3, ... + 3$. Once finished the search over the first region, the minimum SAD found is compared to a threshold, and it is decided whether to continue or to stop the search. In the former case, the region of group 2 that is closest to the location of the minimum SAD found in region 1 is selected (2). Then another search is made on this region, and on the regions of the same group, following the path of Fig. 2 until the threshold test condition is true. Finally, if the test always fails, the regions of the third group are considered. The main difference between this algorithm and the zonal searches (APDZS) is that instead of a pixel by pixel spiral search that leads to an irregular data flow, we employ a region by region spiral search that can be seen like many FSs over small regions. For this reason this algorithm can be implemented by a systolic array approach. The key for a good trade-off between computational saving and the quality of estimation is the determination of the threshold value. The proposed threshold calculation is different from the APDZS approach: here the threshold value depends on the distance from the center region:

$$T = SADav * (10 - i)/5 \tag{3}$$

where $i = 1, 2, 3$ indicates the group to which the region belongs and $SADav$ is the average $SADmin$ obtained in all the previous MBxs.

4.3 Partial SAD Criterion

Finally, another criterion is considered: it consists on the introduction of another threshold test after every $SAD_{line}(i)$

calculation. If the partial SAD is too high, the calculation is halted. We have proposed a threshold:

$$T_2 = SADav * (n + 5)/8 \quad (4)$$

where n is the number of rows already calculated.
Using this criterion the complexity significantly decreases.

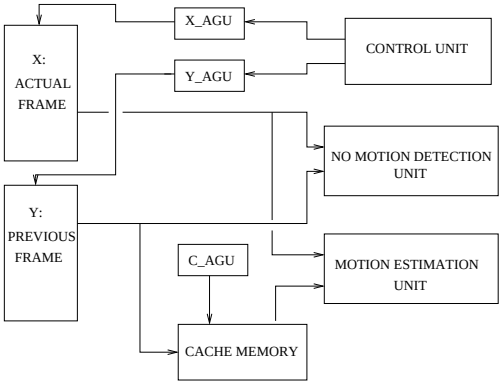

Figure 3: Scheme of the processor.

5 PROPOSED ARCHITECTURE

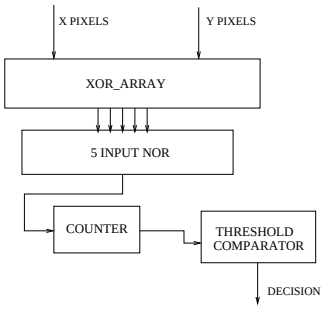

Figure 4: Scheme of the No Motion Detection Unit.

In Fig. 3 we can see the general scheme of the processor and the two distinct $AddressGenerationUnits$ (AGUs) that address the pixels related to the current frame and the previous one.
First the $No - MotionDetectionUnits$ (NMDU) module (see Fig. 4) is activated and then if the motion is decided to be on, the $MotionEstimationUnit$ (MEU) module is enabled.
HORB algorithm can be implemented by any of the systolic array architectures that have been proposed recently. However, the implementation of a FS over a window $w \times w$ is more efficient as the size w of the window is larger because there is greater reuse of consecutive pixels. A linear array of N Processing Elements (PEs) takes N cycles to be filled using only input data, and then there are w useful cycles, until the array must be refilled with a new line. The efficiency decreases as w become lower. In order to avoid the pipeline stalls, a more complex design must be considered [10]. For this reason we have developed a linear architecture tailored

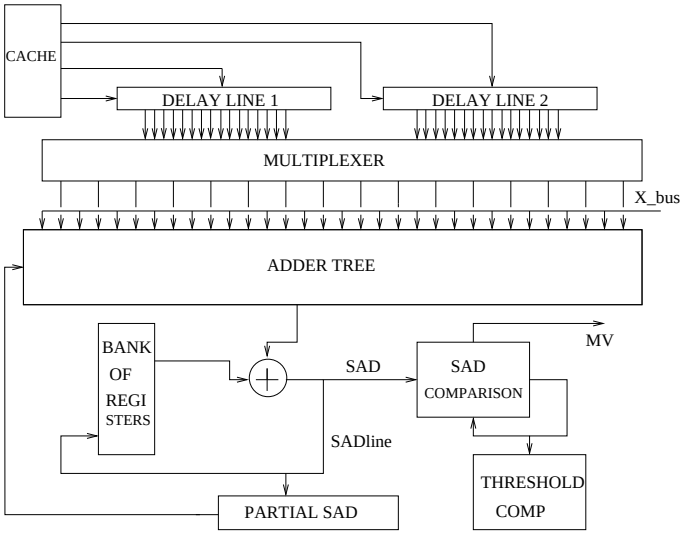

Figure 5: Scheme of the architecture proposed.

to meet the constraints of the HORB algorithm. The data related to the previous frame (Y) are entered into an internal cache memory, and then handled by another AGU that control the dataflow to the MEU.

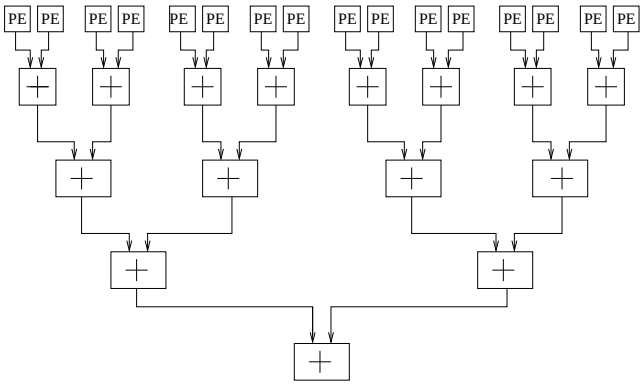

Figure 6: Scheme of the adder tree.

This module (see Fig. 5) consists of an adder tree fed by two delay lines, 16 taps long, in which Y pixels are entered serially. In each delay line a different line is entered. It takes 8 cycles to fill the pipe, because of the two inputs to each delay line. When a pipe is full, the data are entered into the 16 PEs, where the absolute differences between pixels are computed, and the results are summed by the adder tree (see Fig. 6). Therefore, a single $SAD(h,k)_{line}$ can be calculated on every cycle, due to the pipelining of the different stages of the adder tree. Then another pixel is entered into the pipe and all other data are shifted to the right. So, at every cycle the same data are used to calculate another partial $SAD(h,k)_{line}$ related to a neighbouring MV with different k, and it takes 7 cycles to calculate all the $SAD(-3,k)_{line}(1)$ with $k = -3, ... + 3$. Then the $SAD(-2,k)_{line}(1)$ is considered: the related data are already loaded into the second delay line that can be filled

	PSNR(FS)	PSNR(HORB)	Av. Power (%)
FOREMAN	30.30	30.18	15.6
MISS AMERICA	41.39	41.25	10.1
CARPHONE	30.53	30.50	12.2

Table 1: Simulation results for 3 different QCIF (15 Hz) video sequences coded at 48 Kbit/s.

while the other pipe is feeding the data into the calculation module.

Every $SAD(h,k)_{line}(1)$ obtained is stored in the bank of registers, and tested with the threshold (by the Partial SAD unit), and if the value is too high, the next computations related to that h,k are disabled as in [11]. It is worth noting that while in [11] the entire SAD is calculated for a line of the search area, before considering the other lines (thus having to reload completely the X data), we calculate first the $SAD_{line}(i)$ of all the pixels of the region before considering the $SAD_{line}(i+1)$, thus X data are loaded only once and the power consumption is reduced. This can be done because we use a Partial SAD Criterion quite different from that used by [11].

Once finished the calculation of all the Partial $SAD(h,k)_{line}(1)$ for $h=-3,...+3$ and $k=-3,..+3$, the X data, that have been stored in the meantime, through the dedicated *X bus* in the X_{next} registers (see Fig. 7), pass to the X_{act} registers and the processor is immediately able to calculate the $SAD(h,k)_{line}(2)$. These results are summed to those stored in the bank of registers, an so on.

In case of a pure FS algorithm (without the Partial SAD Criterion), a modified architecture has to be considered, without the Partial SAD module and the data are fed into the two delay lines in different order. As a consequence in this case we first calculate $SAD(-15,k)_{line}(1)$ for all k, then $SAD(-14,k)_{line}(1)$, and so on.

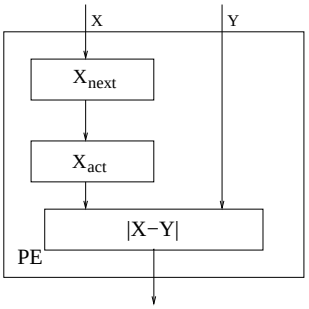

Figure 7: Scheme of the Processing Element (PE).

6 EXPERIMENTAL RESULT

We have modelled our architecture in VHDL language. It can realize both FS and HORB algorithms. The minimum clock frequency necessary to implement the algorithms in QCIF sequences at 15 fps is of 26 MHz. CIF sequences need 100 MHz. The MVs found with this module are then entered to a software video coder. We have performed the simulation for both these algorithm considering three different video sequences. In case of FS we have chosen the modified architecture. We have evaluated the power consumption as in [11], for both FS and HORB. In Table 1 we present the results in terms of PSNR and of percentage of power consumption referred to FS. As we can see, the video quality is almost the same, while the power consumption is extremely reduced by the use of our algorithm.

7 CONCLUSIONS

In this paper we have presented an architecture for low power real time motion estimation tailored to our HORB algorithm. The joint analysis of the HORB algorithm and the proposed architecture outlines the significant reduction of power consumption achieved.

References

[1] A.M Tourapis and O.C. Au, "New Result on Zonal Based Motion estimation Algorithms-Advanced Predictive Diamond Zonal Search", *Proc. of Int. Conf. of Circuits and Systems ICCS-01*, pp.512-516, Sidney, May 2001.

[2] P. I. Hosur and K.K. Ma, "Motion Vector Field Adaptive Fast Motion Estimation," *Second Int. Conf. on Information, Communications and Signal Processing (ICICS 99)*, pp.384-388, Singapore , 7-10 Dec. 1999.

[3] A. Fermo, V. Pahor and G. Sicuranza, "Hardware-Oriented Region Based Algorithm for Low Power Motion Estimation", *Proc. of ISPA-01*, pp.283-287, Pula, Croatia, June 2001.

[4] R. Li, B. Aeng and M.L.Liou,"A New Three Step Search for Block Motion Estimation", *IEEE Trans. Circuits Syst. Video Technol.*, vol. 4, pp. 438-442, Aug. 1994.

[5] K.M.Uz, M.Vetterli and D. LeGall, "Interpolative Multiresolution Coding of Advanced Television with Compatible Subchannels", *IEEE Trans. Circuits Syst. Video Technol.* vol. 1, pp.86-99, Mar 1991.

[6] T. Oscal and C. Chen, "Motion Estimation using a One Dimensional Gradient Descent Search", *IEEE Trans. Circuits Syst. Video Technol.*, vol. 10 no.4, pp 608-616, June 2000.

[7] S. Y. Kung, "VLSI Array processors", *Prentice Hall*, 1988.

[8] T. Komareck and P. Pirsch, "Array Architectures for Block Matching Algorithms", *IEEE Trans. Circuits and Syst*, vol. 36, pp.1301-1308, Oct. 1989.

[9] C.H. Hsieh and T.P. Lin, "VLSI Architecture for Block Matching Motion estimation Algorithm", *IEEE Trans. Circuits Syst. Video Technol.* vol. 2, pp.169-175, June 1992.

[10] H. Yeo and Y.H.Hu, "A Novel Modular Systolic Array Architecture for Full Search Block Matching Motion Estimation", *IEEE Trans. Circuits Syst. Video Technol.*, vol. 5, N. 5, pp.407-416, Oct. 1995.

[11] L. Sousa and N. Roma, "Low Power Array Architectures for Motion Estimation", *Proc. Int. Workshop on Multimedia Signal Processing*, pp.247-251, Copenhagen, Sept. 1999.