Transactions Letters

Efficient and Configurable Full-Search Block-Matching Processors

Nuno Roma and Leonel Sousa

***Abstract*—Efficient VLSI architectures for motion estimation using the full-search block-matching algorithm are proposed in this paper. These structures are based on an improved and more efficient two-dimensional single-array architecture with minimum latency, maximum throughput, and full utilization of the hardware resources. This optimized architecture is extended to a class of fully parameterizable multiple array architectures that combine both pipelining and parallel processing techniques and provide the ability to configure the processors according to the setup parameters, the processing time and the circuit area specified limits. The development of a single-array processor in a single-chip based on a 0.25-μm CMOS technology process proves the practical interest of the proposed architecture for implementing real-time motion estimators.**

***Index Terms*—Array processing, motion compensation, multiprocessing systems, pipeline processing, systolic arrays, very-large-scale integration, video codecs, video coding.**

I. Introduction

MOTION COMPENSATION is a fundamental technique to obtain data compression in video coding by exploiting interframe prediction of temporal redundancies in video sequences [1]. Several block matching algorithms have been proposed. Among them, the full-search block-matching (FSBM) algorithm exhaustively compares each $N \times N$ block of the current frame with all candidate blocks of the $(N+2p)^2$ search window defined within the previously processed frame. Although this algorithm provides the optimal solution, it is the most computationally expensive.

Motion estimation requires a huge amount of computations [1], which justifies the great research effort that has been made to develop efficient dedicated architectures and specialized processors [2]–[4]. Besides the FSBM algorithm, which is extremely regular and suitable for implementations based on array structures [5], other faster block-matching algorithms have been also proposed [6]. Most of them consider only a reduced set of candidate motion vectors, simpler matching or distortion computations, or even a subset of the block motion field. However, not only do these algorithms provide suboptimal solutions, since the considered search spaces are necessarily reduced, but most of them apply nonregular processing schemes. Consequently, only a few architectures have been proposed to implement fast motion estimation algorithms, due to their less regular structures and higher control overheads [6].

The main goal of the research presented in this paper is the analysis and development of efficient array architectures and dedicated processors for motion estimation based on the FSBM algorithm. From a comparative analysis of the main array architectures that have been proposed over the last few years [7]–[10], [4], one concludes that most of them do not provide a maximum and constant throughput or a full utilization of the hardware resources. Due to its peculiar processing scheme, the AB2 architecture [7] proposed by Vos and Stegherr [8] was selected as the basis for the presented research. It is shown that this architecture can be significantly improved in what concerns the hardware requirements, by reducing the amount of memory required to store the search area, thus achieving a full utilization of the hardware resources. It is also shown that the improved single-array architecture can be extended to multiple-array architectures by exploiting the parallelism and lack of data dependencies provided by the motion-estimation algorithm. A new class of parameterizable array architectures is derived, integrating the proposed single- and multiple-array architectures.

The proposed class of architectures was described using fully parameterizable VHSIC Hardware Description Language (VHDL) code and its functionality was thoroughly tested. An integrated circuit for motion estimation was developed, by making use of this class of architectures and using a standard cell library of a CMOS 0.25-μm technology. Experimental results show that the implemented configuration is able to estimate motion vectors in 4CIF video sequences in real-time.

II. FSBM Architectures

The FSBM algorithm using the sum of absolute differences (SAD) distortion measure can be described using the four nested loops presented in Fig. 1. The specific characteristics of a given FSBM architecture are defined by the set of these loops that are executed in parallel. Assuming, for example, that the variable l in the algorithm of Fig. 1 is set to a fixed value and that each processor element (PE) performs the primitive operation SAD, a 3-D dependence graph (DG) can be derived [11], [12]. Systolic array structures are obtained by applying the usual index projection, time scheduling and graph folding [11], [12] operations to project the DG onto structures defined in lower dimensional

Manuscript received June 4, 2001; revised May 17, 2002. This work was supported in part by the POSI program and by FCT PRAXIS XXI project POSI/CHS/40877/2001. This paper was recommended by Associate Editor S. Chen.

N. Roma and L. Sousa are with the Electrical and Computer Engineering Department, Instituto Superior Técnico (IST), Universidade Técnica de Lisboa (UTL), Lisboa, Portugal, and also with the Signal Processing Research Group (SiPS), INESC-ID, 1000-029 Lisboa, Portugal (e-mail: nuno.roma@inesc-id.pt; las@inesc-id.pt).

Digital Object Identifier 10.1109/TCSVT.2002.806818

```
(x, y) ⇐ (0, 0) {motion vector initialization}
SAD (x, y) ⇐ ∞
for c = −p to p do {(2p + 1) × (2p + 1) search area}
  for l = −p to p do
    SAD (c, l) ⇐ 0 {SAD similarity measure initialization}
    for u = 0 to (N − 1) do {N × N reference macroblock}
      for v = 0 to (N − 1) do
        SAD (c, l) + = |R (u, v) − S (c + u, l + v)|
      end for
    end for
    if SAD (c, l) < SAD (x, y) then
      (x, y) = (c, l) ; SAD (x, y) = SAD (c, l)
    end if
  end for
end for
return (x, y) {motion vector = (x, y)}
```

Fig. 1. FSBM algorithm using SAD distortion measure.

spaces. FSBM architectures are usually classified according to the set of performed projections, giving rise to 1-D structures if multiprojection techniques are applied. Their execution time is dependent on the specific arrangement of the data supply and on the number of projections performed in the retiming procedure.

One of the first discussions about FSBM architectures was presented by Komarek and Pirsch [7]. They discussed the characteristics of a set of 2-D and 1-D processor arrays that were obtained by reducing the dimensions of the initial DG using the referred operations. The main difference between these arrays is the exploited processing concurrency, which implies different structures and different number of PEs (#PE) and is dependent on the desired performance versus circuit area trade-off. The so-called type I—AB2 bidimensional structure requires $\#\mathrm{PE} = N^2$, while the AS2 type array uses $\#\mathrm{PE} = N \times (2p+1)$. By projecting the initial DG twice, 1-D arrays are obtained, such as the AB1 structure, with $\#\mathrm{PE} = N$, or the AS1, with $\#\mathrm{PE} = 2p+1$.

Vos and Stegherr [8] proposed an improved version of the type I—AB2 2-D structure which presents some significant advantages in what concerns the processing time. Hsieh [9] presented a type I architecture with some improvements in what concerns the transfer of data into the processing circuit. In his proposal, the candidate macroblock is supplied as a series of one-dimensional data through a set of delay elements. Chang [10] proposed an alternative notation for the DG representation in order to improve the four loop based model: instead of nodes and links, he repeatedly allocated a 2-D (u, v) projection (slice) in the (c, l) 2-D space (tiling). If some conditions are met, Chang's model provides a hardware utilization rate very close to the optimal (100%). However, this is only possible by using multiple data-input lines. Very recently, Kittitornkun and Hu [4] also proposed a different mapping of the original DG in order to improve the throughput of the algorithm, by eliminating all idle clock cycles in the transitions between consecutive frames.

One of the most important figures of merit that is often used to compare the performance of the architectures is the required number of clock cycles (T) to estimate the motion vectors. The values of #PE and T of some of the referred architectures are presented in Table I. It is worth noting that, in practice, the real values of T can be significantly greater than those presented in Table I. In fact, extra clock cycles are frequently necessary to fill the pipeline and dummy results are often computed to preserve a regular data flow. For comparison purposes, the limit situation corresponding to a processor array with a single PE, designated by SinglePE architecture, was also considered.

TABLE I
FSBM SYSTOLIC STRUCTURES

Architecture	$\#PE$	T
SinglePE	1	$N^2 \times (2p+1)^2$
AB1	N	$N \times (2p+1) \times (2p+N)$
AS1	$2p+1$	$N \times (2p+N) \times (2p+1)$
type I - AB2	$N \times N$	$(2p+1) \times (2p+N)$
type I - Vos	$N \times N$	$(2p+1)^2$
type I - Hsieh	$N \times N$	$(2p+N)^2$
AS2	$N \times (2p+1)$	$N \times (2p+1)$
type II	$(2p+1) \times (2p+1)$	N^2

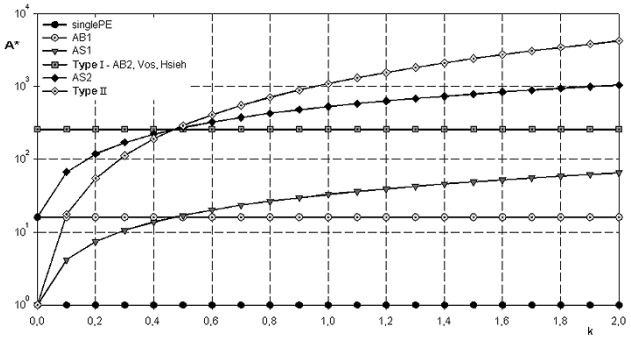

Fig. 2. Circuit area (A^*) in function of k $(N = 16)$.

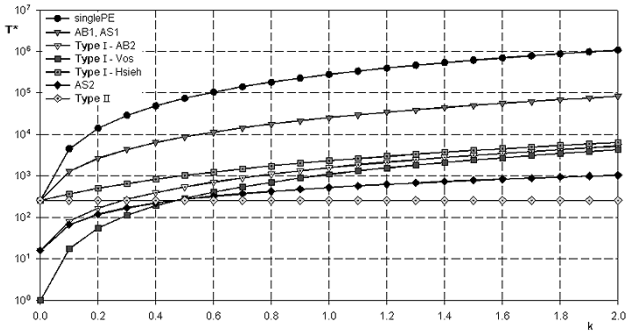

Fig. 3. Processing time (T^*) in function of k $(N = 16)$.

The circuit area (A^*) and the processing time (T^*) were estimated by parameterizing the set of expressions presented in Table I in terms of $k = (p/N)$, which represents the relation between the reference macroblock width (N) and the maximum considered displacement of the candidate macroblocks in each direction in the search area (p). The obtained results are presented in Figs. 2 and 3. In AB1 and type I architectures, the circuit area is independent of the search window size (N and N^2 processing elements, respectively), while in AS1, AS2 and type II structures, it increases significantly with the dimension of this window. Therefore, these last structures are usually advantageous for small sized search windows $(p \leq N/2)$, while the former offer advantages for greater search areas. In what concerns the processing time, while for most architectures it increases with the search window size, its value is constant for the

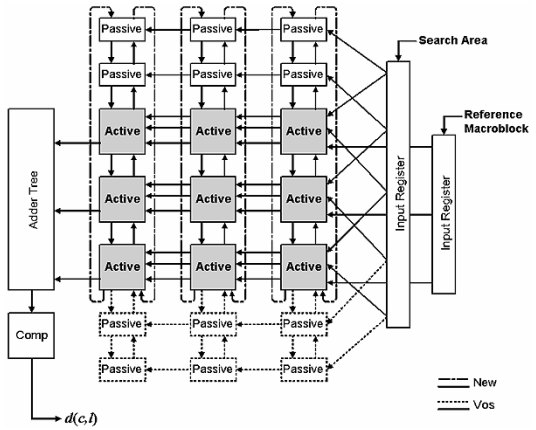

Fig. 4. Type I processor array for FSBM motion estimation, considering $N = 3$ and $p = 1$.

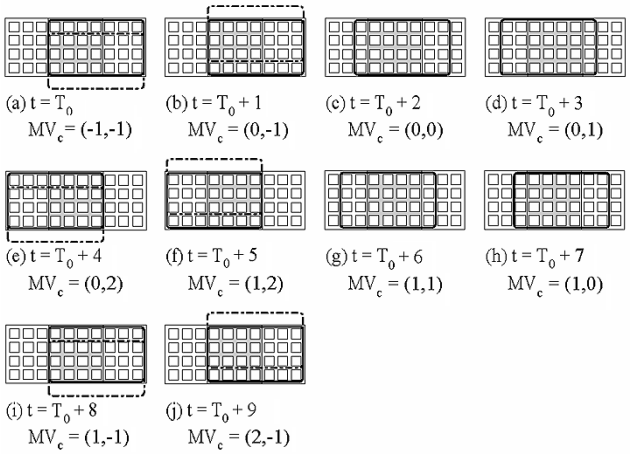

Fig. 5. Zig-zag data flow of search-area pixels in Vos' architecture ($N = 4, p = 2$).

type II structure, since one PE is used to compute the SAD measure of each candidate macroblock.

Among all these array structures, the type I architecture proposed by Vos has been recognized as being one of the most efficient structures [4], [13]. Its main advantages are the short processing time and the limited amount of hardware requirements, when compared with other bidimensional structures. However, this architecture still has some nonexploited features, which can be used to significantly improve its efficiency in terms of hardware resources and exploited parallelism level.

III. An Improved Single-Array Architecture

The proposed single-array architecture is based on Vos architecture but presents significant improvements in what concerns its structure. Due to the similarities between the processing schemes of these two architectures, the description of the proposed structure will be done by contrasting its optimized characteristics with those presented by Vos and Stegherr [8], [14]. Therefore, references to Vos architecture will be done whenever it shows to be convenient.

The diagram shown in Fig. 4 illustrates the main differences between the architecture proposed by Vos, represented using solid and dotted style lines (⋯⋯), and the proposed architecture, represented with solid and dotted-dashed style lines (—·—·—). Like any other type I bidimensional structure, each pixel of the reference macroblock is assigned to one of the N^2 PEs that compute the SAD similarity function (designated by *active PEs*). Besides this *active block*, the processor proposed by Vos is also composed by two *passive blocks* with $2p \times N$ *passive PEs*, which are appended to each side of the active block (see Fig. 4). Each passive PE is mainly composed by data registers for the displacement and storage of search-area pixels. Both the reference macroblock and the search-area pixels are transferred into the processor through two vertical input register chains, with length N and $2p + N$, respectively.

Within the PE array, search-area pixels can be displaced in three directions: upwards, downwards, and to the left. If, at a given clock cycle, one column with $2p+N$ pixels of the search area is fed into the structure through the set of $2p + N$ upper inputs, all search-area pixels within the PE array are simultaneously shifted one position to the left. During the following $2p+1$ clock cycles, search-area data is shifted downwards one position per cycle. Meanwhile, the pixels corresponding to different candidate macroblocks are processed by the several active PEs, obtaining one similarity value in each clock cycle. After $2p + 1$ shift-down operations, another left shift of the search area is performed and a new column of pixels is fed in the right side of the array. However, this column is now loaded through the $2p + N$ lower inputs. This alternation of input positions in the input register chain is repeated along the search process. During the next $2p+1$ clock cycles, search-area data is shifted upwards in a similar manner as described above, being shifted to the left after the $(2p+1)$th clock cycles. Contrasting with other architectures [7], [9], this zig-zag processing scheme provides fast processing capabilities, preventing the need for dummy clock cycles between any two adjacent rows of the search area. To ease the discussion, a 90° rotation of the processor diagram presented in Fig. 4 will be considered in the description.

The processing scheme of Vos' architecture can be represented, in a simplified way, by the sequence of states shown in Fig. 5. The fraction of the search area being processed at a given clock cycle was represented using a solid-line rectangle, whereas those leaving or entering the processor were represented using a dashed-line rectangle. The bottom dashed-line rectangles represent search-area fractions entering the processor in the next clock cycle, while the top dashed-line ones represent already processed search fractions leaving the array.

Fig. 5 evidences that in an array composed by N^2 active PEs and by $2 \times N(2p-1)$ passive PEs, used to process search fractions with $N \times (N+2p-1)$ pixels, half of the total amount of passive PEs, $N \times (2p-1)$, are not being used at any clock cyle. However, these passive PEs are required whenever search-area pixels are displaced into their registers. The proposed solution to overcome this drawback consists in disposing the Vos' planar structure over a cylindrical surface, as it is shown in Fig. 6. By doing so, since the pair of passive blocks is superimposed, one can naturally discard one of them, using the other to displace the search-area pixels. Nevertheless, the zig-zag processing scheme can still be applied to this modified structure, preserving the

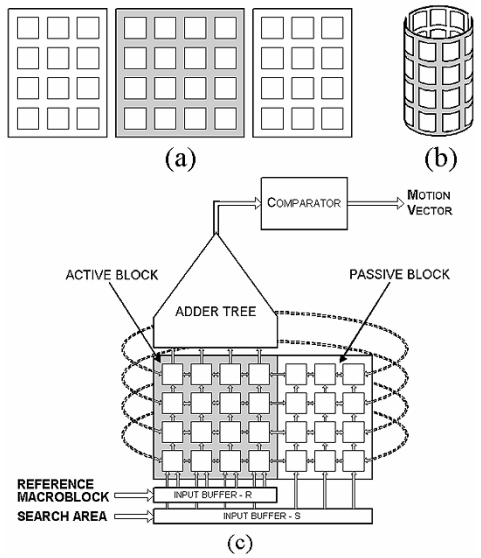

Fig. 6. Rearrangement of the processor array $(N = 4, p = 2)$. (a) Planar processor. (b) Processor disposed over a cylindrical surface. (c) Simplified block diagram of the proposed structure.

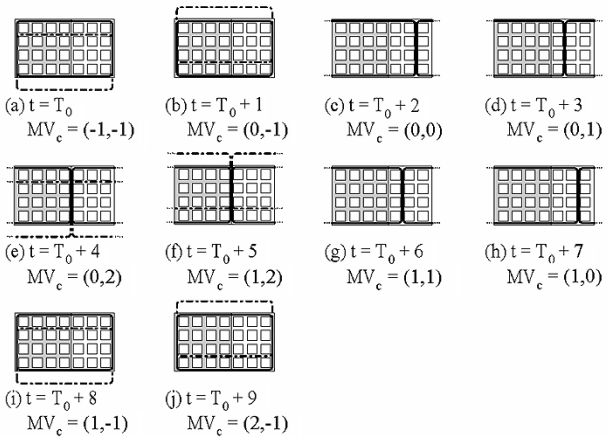

Fig. 7. Zig-zag data flow of search-area pixels in the proposed architecture $(N = 4, p = 2)$.

properties of Vos' architecture but keeping all PEs busy at any instant.

A simplified block diagram of the proposed structure is shown in Fig. 6(c). The cylindrical structure of Fig. 6(b) is obtained by connecting the passive PEs located in the right margin of the passive block with the active PEs of the left margin of the active block, as it was shown in Fig. 4. The new processing scheme of the proposed architecture is shown in Fig. 7, for the same setup of Fig. 5.

In contrast with Vos' architecture, this structure does not require passive PEs not carrying useful data at some clock cycles. Moreover, the zig-zag processing scheme of Vos' architecture is preserved, thus maintaining its recognized efficiency. However, while in Vos' architecture $[N+2\times(2p-1)]\times N$ registers are required, in the proposed architecture only $[N+(2p-1)]\times N$ registers are necessary. The chart presented in Fig. 8 illustrates the relation between the number of registers required by both structures to perform the displacement of search-area pixels. This

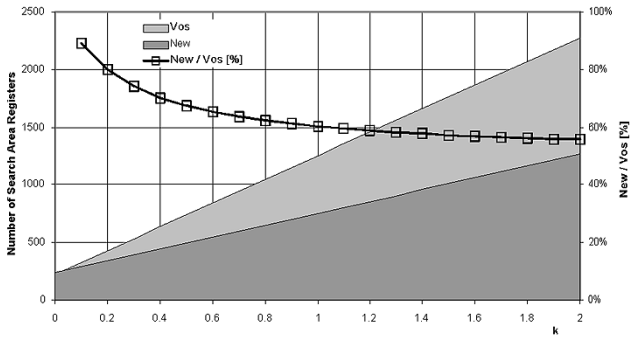

Fig. 8. Number of displacement registers required in Vos' architecture and in the proposed architecture $(N = 16, k = p/N)$.

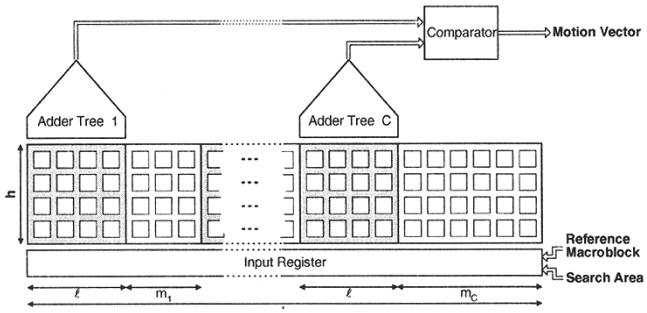

Fig. 9. FSBM architecture with C active blocks.

relation is about 60% for $k = 1$ $(p = N)$ and 55% for $k = 2$ $(p = 2N)$.

IV. New Class of Multiple-Array Architectures

The proposed motion-estimation architecture can be further sped up by computing in parallel several distortion measures using multiple active blocks (hereafter designated by "cores"): since both the passive and the active PEs perform the same displacement task of the search-area pixels, an $N \times N$ passive PE array can be replaced by an array of active PEs with the same size. However, as shown in Fig. 9, such a processor, composed of C processing cores, will require C adder-tree blocks to compute the distortion values of all candidate macroblocks being processed at each clock cycle. Likewise, the former passive block, composed of $(2p-1)\times N$ processing elements [see Fig. 6(c)], is now fragmented into several smaller passive blocks (attached to the corresponding active blocks). The fraction of the last passive block composed by the set of $(N-1)$ columns of PEs located next to the right margin of the structure is designated by *connection block*. The last column of this block is connected with the leftmost column of the first active block, thus obtaining the previously described cylindrical structure [see Fig. 6(c)].

The usage of several active blocks makes it also possible to split the computation of each distortion value into more than one active block or time unit. This feature provides the ability to use active blocks composed by only ℓ columns of PEs and h rows of PEs $(1 \le \ell, h \le N)$, thus adapting the processor structure to the characteristics of the implementation technology (see Fig. 9). In such schemes, each distortion measure is only obtained after multiple complete excursions of the processing

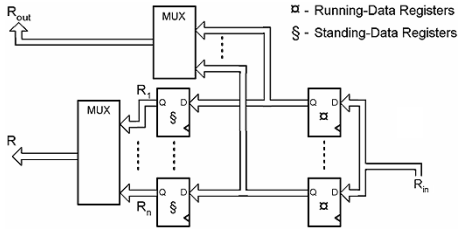

Fig. 10. $\lceil N/h \rceil \times \lceil N/\ell \rceil$ reference pixels must be stored in the standing-data registers of the active PE.

cores in the corresponding search areas, in order to process each of the $\lceil N/h \rceil \times \lceil N/\ell \rceil$ fractions of the reference macroblock. Consequently, $\lceil N/h \rceil \times \lceil N/\ell \rceil$ reference pixels, corresponding to each of these excursions, will have to be stored in the standing-data registers of the PEs, as shown in Fig. 10. Meanwhile, search-area data must be displaced in both directions of the processing array.

To maximize the effectiveness of the processor, it is assumed that all PEs process the same number of search-area pixels. Consequently, the number of candidate macroblocks processed by each of the C active blocks in a given row of the search area should be fixed to $\lfloor 2p/C \rfloor$. This restriction will imply the usage of a slightly smaller search area, composed by only $\hat{p}$ candidate macroblocks in each row. In fact, for an array processor composed by C processing cores (each one with $\ell \times h$ active PEs) and a search range initially defined in the interval $[-(p-1), p]$, corresponding to a total of $(2p)^2$ candidate macroblocks defined in a $(2p+N-1)^2$ pixel search window, the effective number of candidate macroblocks in each row of the search area $(\hat{p})$ is given by

$$\hat{p} = C \left\lfloor \frac{2p}{C} \right\rfloor \tag{1}$$

with a maximum deviation from the initial value given by

$$0 \leq 2p - \hat{p} \leq C - 1. \tag{2}$$

The number of pixels that compose the effective search area is

$$L \times L = [\hat{p} + (N-1)]^2 \tag{3}$$

and the number of passive PEs between each of the C active blocks is given by m_i, with the rightmost passive block accommodating the $(N-1)$ columns of passive PEs of the connection block

$$m_i = \begin{cases} \left\lfloor \frac{2p}{C} \right\rfloor - \ell, & 1 \leq i < C \\ \left\lfloor \frac{2p}{C} \right\rfloor - \ell + N - 1, & i = C \end{cases}. \tag{4}$$

The number of clock cycles (T) required to estimate the motion vector for a given reference macroblock in a processor composed by C processing cores is given by

$$T = \left(\left\lceil \frac{N}{h} \right\rceil \times \left\lceil \frac{N}{\ell} \right\rceil \right) \times \hat{p} \times \left\lfloor \frac{\hat{p}}{C} \right\rfloor. \tag{5}$$

In fact, the search area can be split in three different regions (A, B, and C), as illustrated in Fig. 11. While in region B, all candidate macroblocks are processed using all the $\lceil N/h \rceil \times \lceil N/\ell \rceil$ fractions of the reference macroblock, in regions A and C, not every fraction is used in each excursion of the active blocks [see Fig. 11(b) and (c)]. The number of complete excursions (E) required to process each region can be computed by analyzing Fig. 11

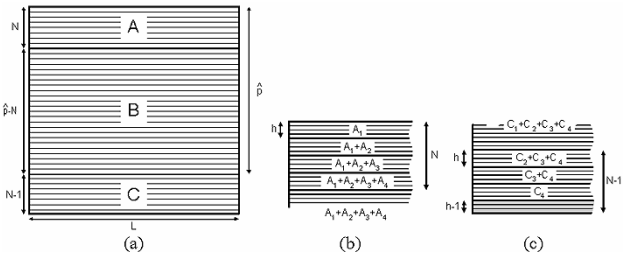

Fig. 11. Required time to estimate each motion vector. (a) Division of the search area in regions A, B and C. (b), (c) Processing of the first N and the last $(N-1)$ rows of the search area, respectively $(h = (N/4); \ell = N)$.

$$E(A) = E(A_1) + E(A_2) + \cdots + E\left(A_{\lceil \frac{N}{h} \rceil}\right) \tag{6a}$$

$$= \left\lceil \frac{N}{\ell} \right\rceil \times \left\{ N + (N-h) + \cdots + \left[N - \left(\left\lceil \frac{N}{h} \right\rceil - 1 \right) h \right] \right\} \tag{6b}$$

$$= \left(\left\lceil \frac{N}{h} \right\rceil \times \left\lceil \frac{N}{\ell} \right\rceil \right) \times \left[N - \frac{h}{2} \left(\left\lceil \frac{N}{h} \right\rceil - 1 \right) \right] \tag{6c}$$

$$E(B) = \left(\left\lceil \frac{N}{h} \right\rceil \times \left\lceil \frac{N}{\ell} \right\rceil \right) \times (\hat{p} - N) \tag{7}$$

$$E(C) = E\left(C_{\lceil \frac{N}{h} \rceil}\right) + \cdots + E(C_2) + E(C_1) \tag{8a}$$

$$= \left\lceil \frac{N}{\ell} \right\rceil \times \left\{ [(N-1) - (h-1)] + [(N-1) - (h-1) - h] + \cdots + \left[(N-1) - (h-1) - \left(\left\lceil \frac{N}{h} \right\rceil - 1 \right) h \right] \right\} \tag{8b}$$

$$= \left(\left\lceil \frac{N}{h} \right\rceil \times \left\lceil \frac{N}{\ell} \right\rceil \right) \times \left[N - \frac{h}{2} \left(\left\lceil \frac{N}{h} \right\rceil - 1 \right) - h \right] \tag{8c}$$

Thus, (5) is obtained by multiplying the sum of (6c), (7), and (8c) by the number of clock cycles required to process each horizontal excursion $(\lceil \hat{p}/C \rceil)$.

Hence, by adjusting a restricted set of implementation parameters (h, ℓ, and C), processing structures with distinct performances, as well as different hardware requirements, are obtained. This feature can be particularly interesting in implementations with limited hardware resources (such as FPGAs), providing the means to adjust the used resources to the target technology.

A. Typical Structures

In the remaining part of Section IV, some examples of different configurations obtained from the proposed class of processors will be presented. Each processor was denominated as $\mathrm{HLC}(a, b, c)$, where $a = (N/h), b = (N/\ell)$ and $c = C$.

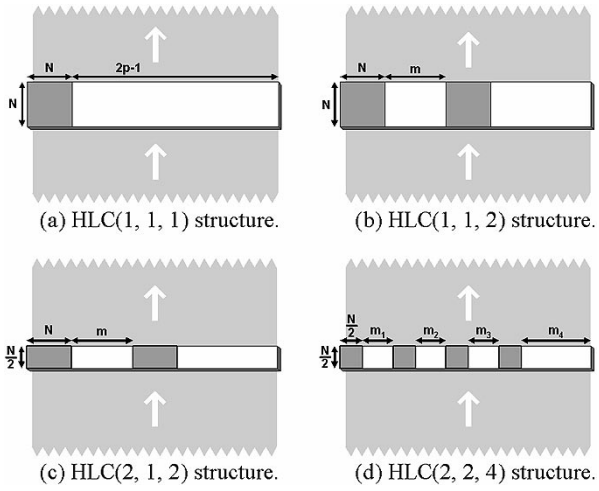

Fig. 12. Single- and multi-array structures.

The $\mathrm{HLC}(1,1,1)$ structure, shown in Fig. 12(a), makes use of a single active block $(C=1)$ and coincides with the previously presented single-array processor. The number of clock cycles required to process one reference macroblock is given by

$$T_{(1,1,1)} = \left(\left\lceil\frac{N}{h}\right\rceil \times \left\lceil\frac{N}{\ell}\right\rceil\right) \times \hat{p} \times \left\lfloor\frac{\hat{p}}{C}\right\rfloor\Bigg|_{h=\ell=N;\ C=1} = \hat{p}^2. \tag{9}$$

In contrast, the $\mathrm{HLC}(1,1,2)$ structure, shown in Fig. 12(b), makes use of two active blocks $(C=2)$, thus increasing the amount of required hardware by a factor of two. Therefore, the number of clock cycles required to process one reference macroblock is halved as follows:

$$T_{(1,1,2)} = \left(\left\lceil\frac{N}{h}\right\rceil \times \left\lceil\frac{N}{\ell}\right\rceil\right) \times \hat{p} \times \left\lfloor\frac{\hat{p}}{C}\right\rfloor\Bigg|_{h=\ell=N;\ C=2} = \frac{\hat{p}^2}{2} \tag{10}$$

Fig. 12(c) illustrates the structure $\mathrm{HLC}(2,1,2)$ where the total amount of required hardware was reduced by using only $h=N/2$ rows of PEs. Two active blocks $(C=2)$ are used in this structure

$$T_{(2,1,2)} = \left(\left\lceil\frac{N}{h}\right\rceil \times \left\lceil\frac{N}{\ell}\right\rceil\right) \times \hat{p} \times \left\lfloor\frac{\hat{p}}{C}\right\rfloor\Bigg|_{h=\frac{N}{2};\ \ell=N;\ C=2} = \hat{p}^2. \tag{11}$$

The number of clock cycles required by this structure is exactly the same used by a type I structure. This fact can be easily understood if one realizes that although the number of registers used by this processor array is only half of the required by the $\mathrm{HLC}(1,1,1)$ structure, it uses exactly the same number of active PEs. Consequently, not only does this structure provide advantages in what concerns the amount of required hardware resources, but it is also characterized by a shorter pre-fetch time of the search-area data. This feature can be significantly advantageous when the target implementation provides limited hardware resources (e.g., FPGA devices).

The structure $\mathrm{HLC}(2,2,4)$, illustrated in Fig. 12(d), is characterized by the same hardware resources than $\mathrm{HLC}(2,1,2)$ structure. However, $C=4$ active blocks were implemented using $h=\ell=(N/2)$. As before, the required number of clock cycles is also $\hat{p}^2$.

TABLE II
NUMBER OF CLOCK CYCLES REQUIRED BY THE CONSIDERED STRUCTURES TO PROCESS ONE REFERENCE MACROBLOCK

HLC Structure	Search area transfer mode	
	Transparent	Non-transparent
$(1,1,1)$	$\hat{p}^2$	$\hat{p}^2+\hat{p}N+N(N-1)$
$(1,1,2)$	$\frac{1}{2}(\hat{p})^2$	$\frac{1}{2}(\hat{p})^2+\hat{p}N+N(N-1)$
$(2,1,2)$; $(2,2,4)$	$\hat{p}^2$	$\hat{p}^2+\frac{1}{2}\hat{p}N+\frac{1}{2}N(N-1)$
nR	$\hat{p}^2+2pN$	$\hat{p}^2+2\hat{p}N+N(N-1)$

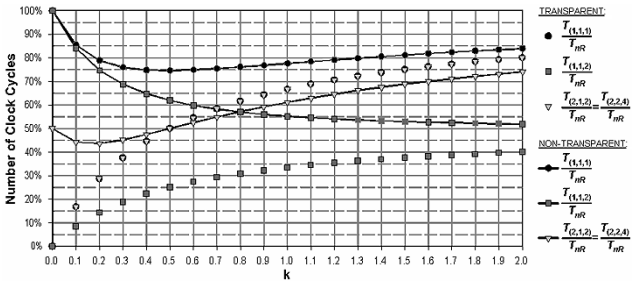

Fig. 13. Relation between the number of clock cycles required by the considered structures of the proposed class and by the traditional nonreversive structure, using both the transparent and nontransparent transfer modes $(k=(p/N))$.

For the sake of comparison of the performances of the presented structures, we also considered an architecture using a single active block $(C=1)$ and not making use of the reversive zig-zag processing scheme. In this nonreversive (nR) structure, search-area pixels are always displaced in the same direction, which presents some analogies with the architecture proposed by Hsieh [9]. Consequently, N additional clock cycles are required to process each row of candidate macroblocks in order to move the processed pixels to the opposite side of the array

$$T_{nR} = \hat{p} \times (L+1) = \hat{p} \times [\hat{p}+(N-1)+1] = \hat{p}^2+\hat{p}N. \tag{12}$$

The number of clock cycles required by each of these structures is summarized in Table II. In the previous analysis, the extra clock cycles required between the processing of consecutive reference macroblocks to transfer or remove unused or already processed search data from the array were not taken into account. These extra clock cycles can be avoided if a transparent transfer mechanism based on the pre-fetch layer described in [15] is used. With such a structure, it is possible to pre-load both the reference and part of the search data corresponding to the next reference macroblock while the current macroblock is being processed. Therefore, any idle clock cycles in the transitions between consecutive reference macroblocks can be avoided, giving rise to the values presented in the first column of Table II. Fig. 13 shows the relation between these values and the number of cycles required by the traditional nonreversive architecture. This chart evidences the significant increase of the number of clock cycles required by those structures using the nontransparent transfer mode for small search areas (small values of k). In such situations, the parcel of time corresponding to the read operation of the search data cannot be disregarded. For greater values of k,

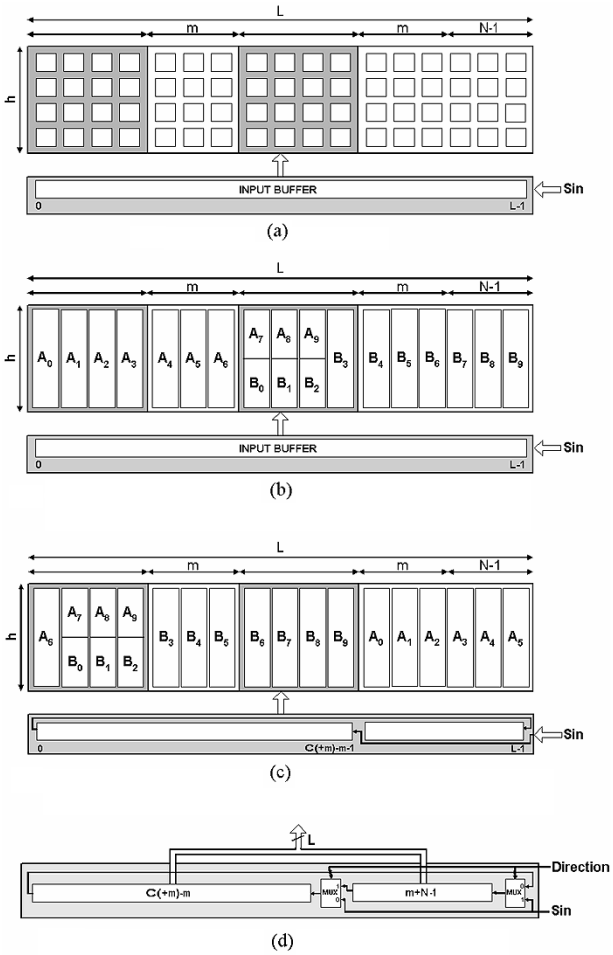

Fig. 14. Search-area input buffer. (a) Considered implementation. (b) Correct alignment of pixels in the input buffer with those in the processing array. (c) Misalignment between the pixels in the input buffer and those in the processing array. (d) Input buffer and alignment circuit of the search pixels.

these relations approach those obtained for the structures with transparent pre-fetching.

V. Data-Flow Control

The desired data-flow between the several blocks of the processor can only be guaranteed through careful design of the various control units that provide the set of signals required by the several blocks of the processor. To simplify the design of these units, they were decomposed into local and simpler control circuits and synchronized through a restricted set of signals. The number of states of each controller was optimized in order to guaranty the tradeoff between the efficiency and the flexibility levels of each circuit and the required amount of hardware resources. The main control circuits are the *central control unit* (decomposed into the main state machine, the data flow controller and the line and column counters), the *clock generator*, and the *reference and search-area input controllers*.

As an example, the search-area input controller is responsible for loading search-area pixels into the processor array by means of a serial-in-parallel-out (SIPO) input buffer. It reads fractions of the previous image, composed by the set of L pixels corresponding to each row of the search area, and transfers them in parallel to the array as soon as all the $(\lceil N/h \rceil \times \lceil N/\ell \rceil)$ fractions of the reference macroblock have been processed. Its implementation is carried out by means of L cascaded registers with their outputs connected to the processor array, as shown in Fig. 14(a).

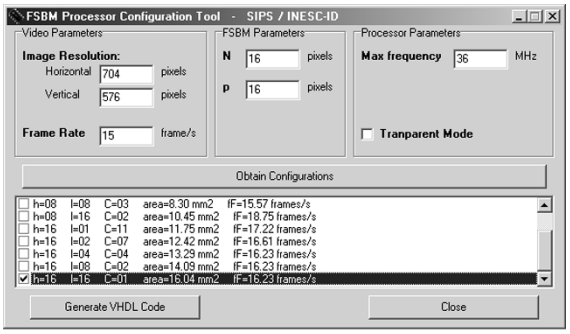

Fig. 15. FSBM processor configuration tool.

However, with the introduction of the new processing scheme based on the cylindrical structure, some measures have to be taken into account in order to provide the correct data to each PE. Fig. 14(a) shows an example of a processor array with $C = 2, N = 4$ and $p = 7$. To simplify the explanation, the processor array was schematically represented in Fig. 14(b), where each column was conveniently numbered according to the corresponding active block A or B. Since some of these columns are processed by more than one active block, they were marked with both representations. The situation illustrated in this figure corresponds to one of the two possible extreme positions of the data being processed in the array and coincides with the transfer of a new search line. In this case, the position of the pixels in the input buffer is the same as in the array. Consequently, it is only necessary to transfer them in parallel to the array.

However, in the situation corresponding to the other extreme position, the spatial distribution of the pixels loaded into the input registers is not aligned with those in the processing array. In fact, their alignment can only be guaranteed if the L registers are connected as it is illustrated in Fig. 14(c). In this case, the set of L input registers is split into two smaller shift registers: one with $C(\ell + m) - m$ registers and other with $m + N - 1$ registers.

The complete search-area input buffer and its alignment circuit is presented in Fig. 14(d). This circuit is characterized by a variable topology of the connections between the two smaller input registers, which depends on the direction of the flow of data being processed. This variable topology is accomplished by means of two multiplexers, making it possible to transfer in parallel all the L input pixels to the correct position in the processing array.

VI. Implementations and Experimental Results

A configuration software tool was developed to determine the several possible configurations of the proposed class of architectures that fulfill the requisites of a given video coder. This configuration tool, whose graphical interface is illustrated in Fig. 15, receives the following set of parameters: the image resolution and the frame rate; the dimension of the reference macroblock (N) and the maximum allowed displacement in

TABLE III
CHARACTERISTICS OF THE IMPLEMENTED CONFIGURATION

Process	0.25μm CMOS-1P5M
Supply voltage	2.5V / 3.3V
Die size	$4 \times 4\text{mm} = 16.07\text{mm}^2$
Active PE area	$32,184.1\mu\text{m}^2$
Maximum frequency	36.5MHz
Pin-Count	56

each direction (p); the processor maximum operating frequency and a flag indicating if the pre-fetching transparent transfer mode is to be used. These parameters are used to compute the number of PEs to be implemented in each line and column of each active block (ℓ, h) and the number of processing cores that will compute the several distortion measures in parallel (C). For each possible configuration, the application outputs the effective maximum frame rate (f_F) and may also supply the required semiconductor area to implement the processor, calculated with *a priori* information about the used technology. As soon as the selection of the desired configuration has been carried out, the configuration tool outputs the complete description of the FSBM processor using IEEE-VHDL description language [15].

To achieve the required characteristics in what concerns the processor performance and configurability, the description of the several blocks of the proposed class of processors was carried out using a fully parameterizable VHDL description, by making extensive use of "`generic`" type configuration inputs. Furthermore, a fully structural description of these circuits was also carried out, using only the most elementary logic operations provided by the implementation library to achieve the required optimization levels of the several processing blocks.

It was designed a FSBM integrated circuit based on the proposed class of architectures with *Synopsys* synthesis tools and *Cadence* design tools. This chip was implemented using the Diplomat-25 standard cell library, based on the UMC 0.25-μm CMOS technology process from Virtual Silicon Technology Inc. [16]. The implemented processor is composed by a single active block $(C = 1)$ with 16×16 PEs $(\ell = h = N = 16)$ and it is characterized by a search range from -15 to $+16$ pixels $(p = 16)$, corresponding to the set of parameters tipically adopted by ITU-T H.26x and ISO MPEG standards [1]. The total area of the implemented chip is about 16.07 mm^2, with a total pin-count of 56. The main characteristics of the chip are summarized in Table III. The chip is able to deliver 28.6 GOPs at 36.5 MHz over typical voltage and temperature ranges, giving rise to a total of 1.78 GOPs/mm^2. With such a configuration, it is possible to estimate motion vectors in 4CIF video sequences at a rate of 16 frame/s.

VII. CONCLUSION

A new class of fully parameterizable multiple-array architectures for motion estimation in video sequences was proposed in this paper. This class is the result of the introduction of significant improvements in the AB2 single-array architecture for the FSBM algorithm, which led to the minimization of its latency, the maximization of its throughput and a full utilization of the hardware resources.

The proposed class of processors provides the capability of further speeding up the estimation of motion vectors, by computing in parallel several distortion measures using multiple and independent processing cores. Optimized implementations of these processing structures can be obtained by means of a software configuration tool that was developed to provide the ability to automatically configure the target processors according to the setup parameters, the processing time, and the circuit area specified limits.

The obtained fully parameterizable VHDL description of this class of architectures provides the possibility to implement efficient processors that are able to perform motion estimation according to the existing ITU-T H.26x and ISO MPEG video coding standards with configurable search ranges and video quality tradeoffs. Experimental results obtained from the implementation of a single-array configuration have shown that it is possible to estimate motion vectors, for example in 4CIF video sequences, at a rate of 16 frames/s with a single chip.

REFERENCES

[1] V. Bhaskaran and K. Konstantinides, *Image and Video Compression Standards: Algorithms and Architectures*, 2nd ed. Norwell, MA: Kluwer, June 1997.

[2] Y. Ooi, "Motion estimation system design," in *Digital Signal Processing for Multimedia Systems*, K. K. Parhi and T. Nishitani, Eds. New York: Marcel Dekker, 1999, ch. 12, pp. 299–327.

[3] L. Sousa, "Applying conditional processing to design low-power array processors for motion estimation," *Proc. IEEE Int. Conf. Image Processing*, Oct. 1999.

[4] S. Kittitornkun and Y. H. Hu, "Frame-level pipelined motion estimation array processor," *IEEE Trans. Circuits Syst. Video Technol.*, vol. 11, pp. 248–251, Feb. 2001.

[5] P. Pirsch, N. Demassieux, and W. Gehrke, "VLSI architectures for video compression—A survey," *Proc. IEEE*, vol. 83, no. 2, pp. 220–246, Feb. 1995.

[6] B. Zeng, R. Li, and M. L. Liou, "Optimization of fast block motion estimation algorithms," *IEEE Trans. Circuits Syst. Video Technol.*, vol. 7, pp. 833–844, Dec. 1997.

[7] T. Komarek and P. Pirsch, "Array architectures for block matching algorithms," *IEEE Trans. Circuits Syst.*, vol. 36, pp. 1301–1308, Oct. 1989.

[8] L. Vos and M. Stegherr, "Parameterizable VLSI architectures for the full-search block-matching algorithm," *IEEE Trans. Circuits Syst.*, vol. 36, pp. 1309–1316, 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] S. Chang, J. H. Hwang, and C. W. Jen, "Scalable array architecture design for full search block matching," *IEEE Trans. Circuits Syst. Video Technol.*, vol. 5, pp. 332–343, Aug. 1995.

[11] S. Y. Kung, *VLSI Array Processors*. Englewood Cliffs, NJ: Prentice-Hall, 1988.

[12] K. K. Parhi, *VLSI Digital Signal Processing: Design and Implementation*. New York: Wiley-Interscience, 1999.

[13] K. K. Parhi and T. Nishitani, Eds., *Digital Signal Processing for Multimedia Systems*. New York: Marcel Dekker, 1999.

[14] L. Vos, M. Stegherr, and T. G. Noll, "VLSI architectures for the full-search blockmatching algorithm," in *Proc. ICASSP'89*, 1989, pp. 1687–1690.

[15] N. Roma and L. Sousa, "Implementation Aspects of MESA Processor,", Lisboa, Portugal, Tech. Rep. RT/001/2001, INESC-ID—Lisboa, Jan. 2001.

[16] VST, "Diplomat-25 Standard Cell Library—0.25 μm UMC Process," Virtual Silicon Technology Inc., Sunnyvale, CA, Dec. 1999.