Journal of VLSI Signal Processing 34, 277–290, 2003

Automatic Synthesis of Motion Estimation Processors Based on a New Class of Hardware Architectures

NUNO ROMA AND LEONEL SOUSA
Department of Electrical & Computer Engineering, Instituto Superior Técnico/INESC-ID, Rua Alves Redol 9, 1000-029 Lisboa, Portugal

Received September 6, 2001; Revised April 8, 2002; Accepted July 22, 2002

Abstract. A new class of fully parameterizable multiple array architectures for motion estimation in video sequences based on the Full-Search Block-Matching algorithm is proposed in this paper. This class is based on a new and efficient AB2 single array architecture with minimum latency, maximum throughput and full utilization of the hardware resources. It provides the ability to configure the target processor within the boundary values imposed for the configuration parameters concerning the algorithm setup, the processing time and the circuit area. With this purpose, a software configuration tool has been implemented to determine the set of possible configurations which fulfill the requisites of a given video coder. Experimental results using both FPGA and ASIC technologies are presented. In particular, the implementation of a single array processor configuration on a single-chip is illustrated, evidencing the ability to estimate motion vectors in real-time.

Keywords: motion estimation, block matching, array architectures, specialized processors

1. Introduction

Video coding systems have been assuming an increasingly important role in several application areas tied in with digital television, videophone and videoconference. Several video compression standards have been established for these different applications [1], exploiting both the spatial and the temporal redundancies of video sequences to achieve the required compression rates. Among these techniques, motion-compensation has proved to be a fundamental method to improve interframe prediction in video coding.

Motion estimation requires a huge amount of computations [1], justifying the great research effort that has been made to develop efficient dedicated architectures and specialized processors [2–4]. Due to their regular processing schemes and simple control structures, Full-Search Block-Matching (FSBM) algorithms, using the SAD matching criteria, have been the most widely used in VLSI implementations, providing optimal estimation results and leading to fast and efficient structures. However, from a comparative analysis of the main array architectures that have been proposed over the last few years [4–8], one concludes that most of them do not provide a maximum and constant throughput or a full utilization of the hardware resources.

One of the first discussions about full-search block-matching architectures and their classification was presented by Komarek and Pirsch [5]. They presented the characteristics of several 1D and 2D arrays, obtained by reducing the dimension of the original Dependence graph (DG) [5] using traditional index projection, time scheduling and graph folding techniques [9]. Their main difference is the exploited processing concurrency, using distinct structures and different number of PEs. As an example, considering each reference macroblock with $N \times N$ pixels and the search area composed by $(2p + N) \times (2p + N)$ pixels, the so called type I—AB2 structure requires N^2 PEs, while the AS2 type array uses $N \times (2p + 1)$ processor elements. By projecting the initial DG twice, 1D arrays are obtained, such as the AB1 structure, with N PEs, or the AS1, with $2p + 1$ PEs.

Meanwhile, other architectures have been proposed [2, 4, 5–8]. Vos and Stegherr [6] proposed an improved version of the AB2 structure that presents some significant advantages in what concerns the processing time. Hsieh [7] 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 [8] proposed an alternative notation for the DG representation in order to improve the original four loop based model: instead of nodes and links, he repeatedly allocated a 2D projection (slice) in the 2D 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.

Among all these proposals, the AB2 2D architecture presented by Vos [6] has been regarded as one of the most efficient structures [2, 4]. Its peculiar processing scheme provides it with a short processing time and a limited amount of hardware requirements. However, it still has some non-exploited features, which can be used to significantly improve its efficiency and parallelism levels. A careful study of these characteristics naturally led to the selection of this architecture as the basis for the presented research. It will be shown that significant improvements in what concerns its hardware requirements can be obtained. As an example, the amount of memory used to store the search area data can be substantially reduced, thus achieving a full utilization of the hardware resources. Moreover, it will be shown that the proposed new 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.

Consequently, a new class of parameterizable array architectures will be derived, integrating the proposed single array and multiple array architectures. This class was described using fully parameterizable VHDL code and its functionality was thoroughly tested, using both FPGA and ASIC technologies. An integrated circuit based on the proposed class of architectures has been developed using a standard cell library of a CMOS—0.25 μm technology process. Experimental results evidence the possibility to estimate motion vectors in 4CIF video sequences at a rate up to 16 frames/s with this implemented configuration.

2. New-AB2 Single Array Architecture

The architecture proposed by Vos is illustrated in Fig. 1. Like in other AB2 structures, 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. 1). Each passive PE is composed by data registers for the displacement and storage of search area pixels. Both the reference macroblock and the search area pixels are transfered 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 at 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 following $2p + 1$ clock cycles, search area data is shifted upwards in a similar manner as described above, being shifted to the left after $2p + 1$ clock cycles. This zig-zag processing scheme provides fast processing capabilities, by preventing the need for dummy clock cycles between any two adjacent lines of the search area. These extra cycles are often required by other architectures to displace search data inside the array [5, 7].

The processing scheme of Vos' architecture can be represented, in a simplified way, by the sequence of states shown in Fig. 2, considering each reference macroblock with $N \times N$ pixels ($N = 4$) and the search

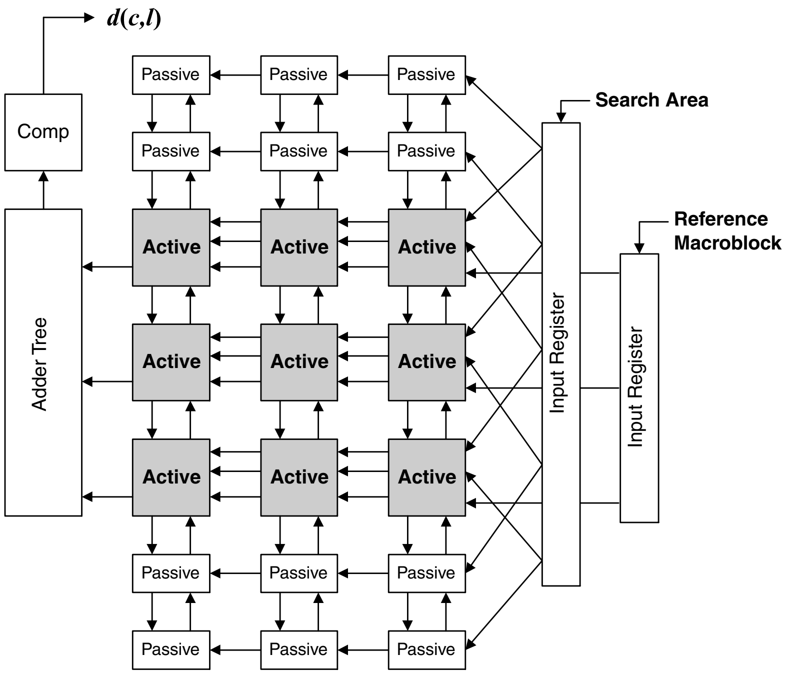


Figure 1. AB2 architecture proposed by Vos, considering each reference macroblock with $N \times N$ pixels ($N = 3$) and the search area composed by $(2p + N) \times (2p + N)$ pixels ($p = 1$).

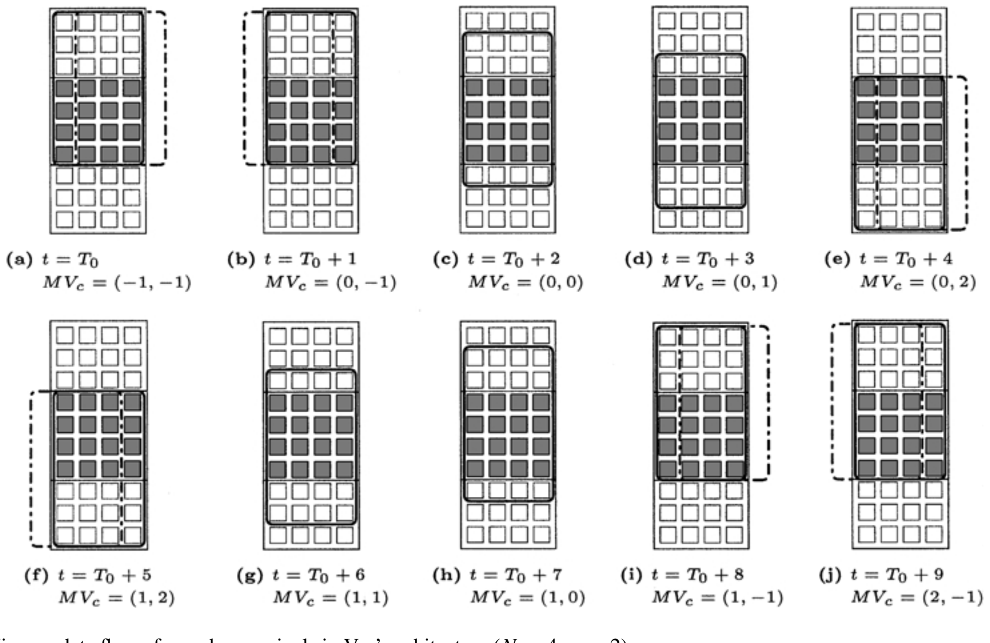

(a) $t = T_0$ $MV_c = (-1, -1)$
(b) $t = T_0 + 1$ $MV_c = (0, -1)$
(c) $t = T_0 + 2$ $MV_c = (0, 0)$
(d) $t = T_0 + 3$ $MV_c = (0, 1)$
(e) $t = T_0 + 4$ $MV_c = (0, 2)$
(f) $t = T_0 + 5$ $MV_c = (1, 2)$
(g) $t = T_0 + 6$ $MV_c = (1, 1)$
(h) $t = T_0 + 7$ $MV_c = (1, 0)$
(i) $t = T_0 + 8$ $MV_c = (1, -1)$
(j) $t = T_0 + 9$ $MV_c = (2, -1)$

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

area composed by $(N + 2p - 1) \times (N + 2p - 1)$ pixels ($p = 2$). 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 dashed-line rectangles. The rightmost dashed-line rectangles represent search area fractions entering the processor in the next clock cycle, while the leftmost dashed-line ones represent already processed search fractions leaving the array.

The processing state represented in Fig. 2(a) corresponds to the processing of candidate macroblock with coordinates $(-1, -1)$ at instant $t = T_0$. At $t = T_0 + 1$, the previously processed search fraction leaves the array and a new fraction is loaded into to the processor, starting the search procedure in the row of candidate macroblocks with coordinate $l = 0$. During this and in the following three clock cycles, the similarity measures corresponding to coordinates $(0, -1)$, $(0, 0)$, $(0, 1)$ and $(0, 2)$ are computed, displacing the search area pixels in the downward direction. At $t = T_0 + 5$, the search area previously processed leaves the array and a new fraction, corresponding to the horizontal coordinate $l = 1$, is loaded. Simultaneously, the similarity measure corresponding to coordinates $(1, 2)$ is computed. The processing of this row is concluded at $t = T_0 + 8$, after the set of candidate macroblocks with coordinates $(1, 2)$, $(1, 1)$, $(1, 0)$ e $(1, -1)$ has been processed, displacing the pixels in the upward direction. In Fig. 2(j) it is represented the beginning of the processing of a new search fraction with coordinate $l = 2$. This state, corresponding to candidate macroblock with coordinates $(2, -1)$, is similar to the one represented in Fig. 2(b), thus concluding the description of the processing scheme.

From a brief analysis of Fig. 2 one realizes 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 $N(N + 2p - 1)$ pixels, half of the total amount of passive PEs are not being used at each clock cycle. 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. 3. By doing so, since the two passive blocks are superimposed, one can naturally discard one of them, using the other to displace the search area pixels. Nevertheless, it is worth noting

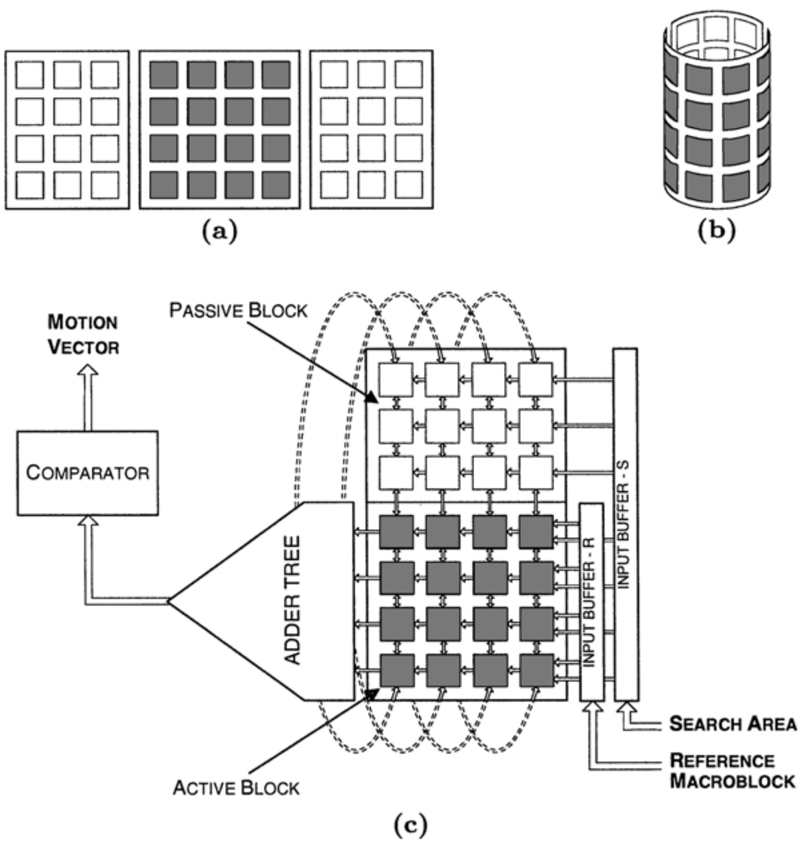


Figure 3. Rearrangement of the processor array ($N = 4$, $p = 2$): (a) planar processor proposed by Vos; (b) the pair of passive blocks are superimposed by disposing the processor over a cylindrical surface; (c) proposed new-AB2 architecture using a single passive block. To ease the representation, the diagrams shown in Fig. 3(a) and (b) were rotated by 90°.

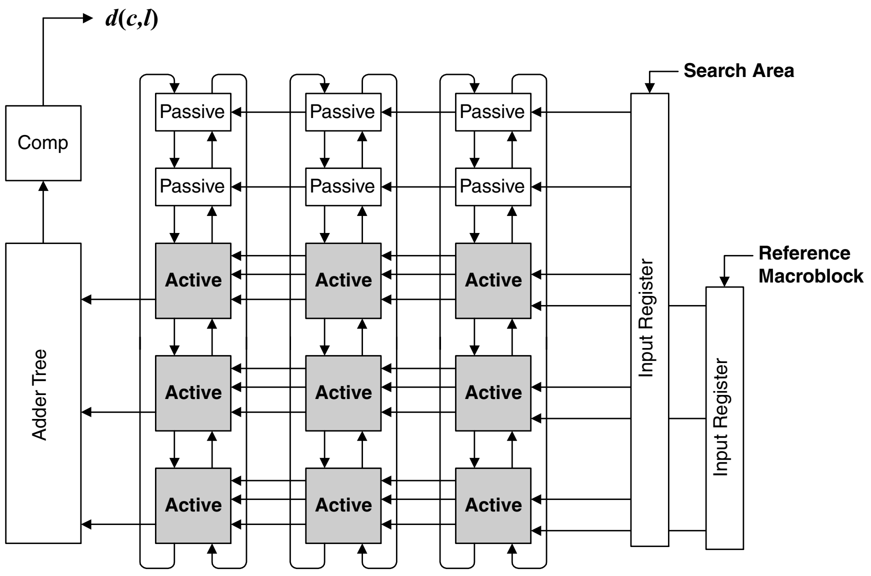


Figure 4. Proposed architecture, considering each reference macroblock with $N \times N$ pixels ($N = 3$) and the search area composed by $(2p + N) \times (2p + N)$ pixels ($p = 1$).

that the zig-zag processing scheme can still be applied to this new structure, preserving the properties of Vos' architecture but keeping all PEs busy at any instant.

A simplified block diagram of the proposed structure is presented in Fig. 3(c). The cylindrical structure of Fig. 3(b) is obtained by connecting the passive PEs located on the top margin of the passive block with the active PEs of the bottom margin of the active block. The most significant differences between this new architecture and the architecture proposed by Vos can be seen by comparing the detailed diagram illustrated in Fig. 4 with the diagram shown in Fig. 1, considering each reference macroblock with $N \times N$ pixels ($N = 3$) and the search area composed by $(2p + N) \times (2p + N)$ pixels ($p = 1$).

The processing scheme of the proposed architecture is shown in Fig. 5. As before, it has been considered a search area with $(2p + N - 1)^2$ pixels, with $N = 4$ and $p = 2$. Contrasting with Vos' architecture, this structure does not require passive PEs without useful data at any

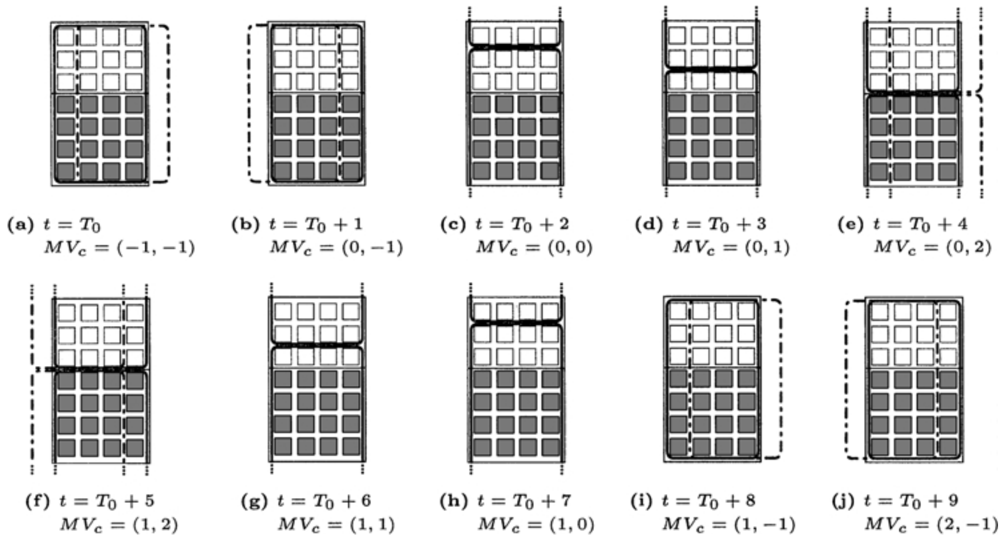

(a) $t = T_0$ $MV_c = (-1, -1)$
(b) $t = T_0 + 1$ $MV_c = (0, -1)$
(c) $t = T_0 + 2$ $MV_c = (0, 0)$
(d) $t = T_0 + 3$ $MV_c = (0, 1)$
(e) $t = T_0 + 4$ $MV_c = (0, 2)$
(f) $t = T_0 + 5$ $MV_c = (1, 2)$
(g) $t = T_0 + 6$ $MV_c = (1, 1)$
(h) $t = T_0 + 7$ $MV_c = (1, 0)$
(i) $t = T_0 + 8$ $MV_c = (1, -1)$
(j) $t = T_0 + 9$ $MV_c = (2, -1)$

Figure 5. Zig-zag data flow of search pixels in the proposed architecture ($N = 4$, $p = 2$).

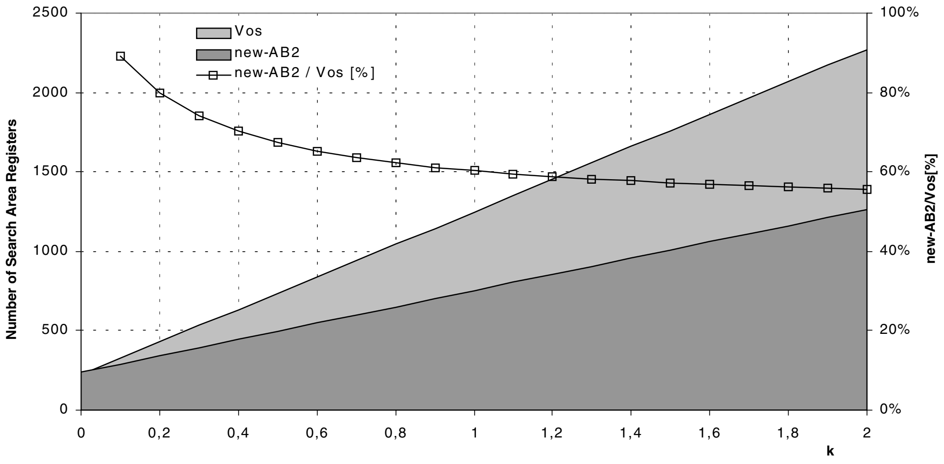


Figure 6. Relation between the required number of displacement registers in Vos architecture and in the *new-AB2* architecture.

moment. The chart presented in Fig. 6 shows the variation of the number of registers required by both structures to perform the displacement of search area pixels, considering $N = 16$ and $k = p/N$. The line-chart represented with the □ marks shows the relation between the number of registers required by the two architectures. This relation is about 60% for $k = 1 (p = N)$ and 55% for $k = 2 (p = 2N)$.

3. New Class of Multiple Array Architectures

The proposed motion estimation architecture can be further speeded up by computing several similarity measures in parallel using multiple active blocks (hereafter designated by "cores"). In fact, since both the passive and the active PEs perform the same displacement task of the search area pixels, a $N \times N$ passive PE array can be replaced by an active PE array. However, such processor, composed by C processing cores, will require C adder tree blocks to compute the similarity values of all candidate macroblocks being processed at each clock cycle, as it is shown in Fig. 7. Likewise, the former passive block, composed by $(2p - 1) \times N$ processing elements (see Fig. 3), will be fragmented into several smaller passive blocks, which are attached to the corresponding active blocks (see Fig. 7). The fraction of the last passive block composed by the set of $(N - 1)$ columns of passive 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. 3).

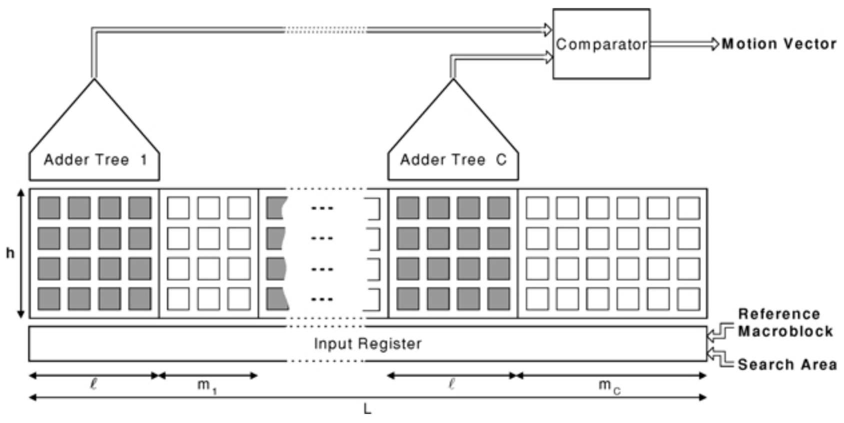


Figure 7. FSBM architecture with C active blocks.

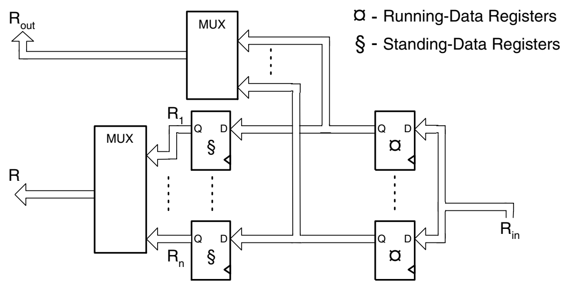


Figure 8. $\lceil \frac{N}{h} \rceil \times \lceil \frac{N}{\ell} \rceil$ reference pixels must be stored in the standing-data registers of the active PE.

The usage of several active blocks makes it also possible to split the computation of each similarity 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 ($1 \leq \ell, h \leq N$), thus adapting the processor structure to the characteristics of the implementation technology (see Fig. 7). In such schemes, each distortion measure is only obtained after multiple complete excursions of the processing cores in the corresponding search areas, in order to process each of the $\lceil \frac{N}{h} \rceil \times \lceil \frac{N}{\ell} \rceil$ fractions of the reference macroblock. Consequently, $\lceil \frac{N}{h} \rceil \times \lceil \frac{N}{\ell} \rceil$ reference pixels, corresponding to each of these excursions, will have to be stored in the standing-data registers of the PEs, as it is shown in Fig. 8. Meanwhile, search area data must be displaced in both directions in 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 $\lfloor \frac{2p}{C} \rfloor$. It can be proved [10] that this restriction will imply the usage of a slightly smaller search area, composed by only $\hat{p}$ candidate macroblocks in each row. In fact, considering 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 $(N + 2p - 1)^2$ 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 estimated 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 pair of active blocks is given by:

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

where the last block of $N-1$ PEs in m_c is the previously described *connection block.*

Hence, by adjusting a restricted set of implementation parameters (h, ℓ and C), processing structures with distinct performance levels, 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.

Considering an array processor composed by C processing cores with $\ell \times h$ active PEs and a search area with $L \times L$ pixels, the number of clock cycles (T_{new}) required to estimate the motion vector of a given reference macroblock is given by:

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

In fact, since the processing of each of the $\lceil \frac{N}{h} \rceil \times \lceil \frac{N}{\ell} \rceil$ fractions of the reference macroblock requires $\lfloor \frac{\hat{p}}{C} \rfloor$ clock cycles, $(\lceil \frac{N}{h} \rceil \times \lceil \frac{N}{\ell} \rceil) \times \lfloor \frac{\hat{p}}{C} \rfloor$ cycles are needed to process each row of the search area. Considering a total of $\hat{p}$ rows of candidate macroblocks, one can easily obtain Eq. (5).

It can be proved [10, 11] that the number of data registers required by the new proposed structure is given by:

$$R_{new} = h \times \left\{ 3[\hat{p} + (N-1)] + \ell . C . \left[2 \left(\left\lceil \frac{N}{h} \right\rceil \times \left\lceil \frac{N}{\ell} \right\rceil \right) + 1 \right] \right\} \tag{6}$$

In particular, if $h = \ell = N$ one can compare Eqs. (5) and (6) with Eqs. (7) and (8), corresponding to the number of clock cycles (T_{vos}) and to the number of registers (R_{vos}) required by the processor proposed by Vos [6]:

$$T_{vos} = (2p)^2 + N(N + 2p - 1) \tag{7}$$

$$R_{vos} = 6N^2 + 2N(2p - 1) \tag{8}$$

3.1. *Arranging the Structures by Categories*

The chart presented in Fig. 9 illustrates the relation between the set of implementation parameters (h, ℓ and C) and the number of clock cycles (T) and registers (R), normalized to the corresponding number of cycles and registers required by the architecture proposed by Vos:

$$T = \frac{T_{new}}{T_{vos}}; \quad R = \frac{R_{new}}{R_{vos}} \tag{9}$$

In order to obtain a comparison as fair as possible, all configurations considered in this section will assume the same algorithm parameters: $N = 16$ and $p = 16$. In this chart, the point with coordinates $(R, T) = (1, 1)$, corresponding to Vos' architecture, defines four different categories of processors with distinct characteristics in what concerns the processing structure and the performance levels:

- *Category I* ($R > 1$, $T < 1$). Processors of this category exhibit smaller processing times than the processor proposed by Vos, by taking advantage of greater levels of parallelism offered by the full-search block-matching algorithm. Three examples of processors of this category are illustrated in Fig. 10(a)–(c). Processor A makes use of a single active block ($C = 1$) and coincides with the single array structure previously described. The corresponding number of clock cycles required to process one reference macroblock is $T_A = \hat{p}^2 = (2p)^2$. Processor B makes use of two active blocks ($C = 2$), thus increasing the amount of required hardware by a factor of 2. The required number of clock cycles is halved: $T_B = \hat{p}^2/2 = \frac{1}{2}(2p)^2$. Processor C uses four active blocks ($C = 4$) and $h = \ell = \frac{N}{2}$. The required number of clock cycles is $T_C = \hat{p}^2 = (2p)^2$. In fact, although the number of PEs used by this processor is only half of the required by processor A, it uses exactly the same amount 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.
- *Category II* ($R < 1$, $T > 1$). Processors of this category use a smaller amount of hardware resources than the processor proposed by Vos. However, several complete excursions of the processing cores over the corresponding search area are required. One example of such processors is presented in Fig. 10(d). This processor makes use of a single active block ($C = 1$) and the amount of required hardware has

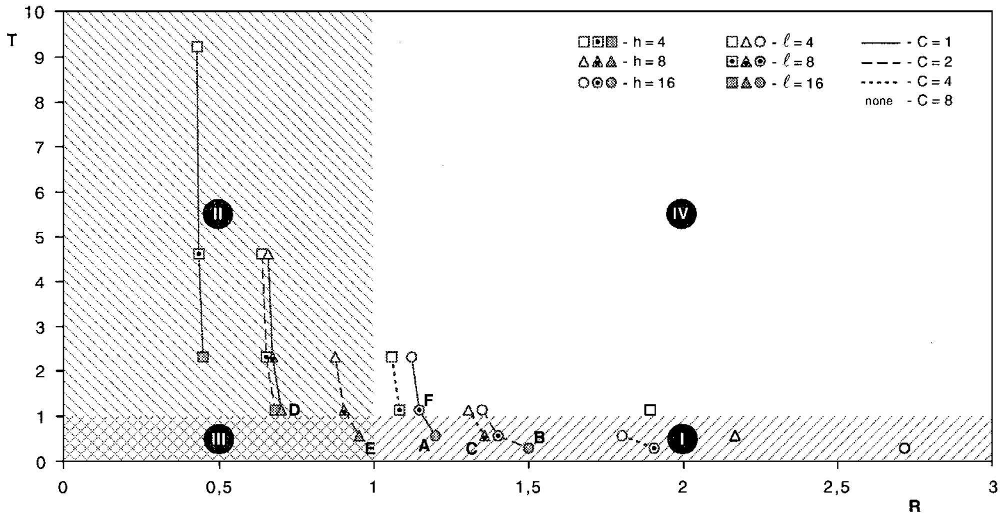


Figure 9. Relation between the implementation parameters (h, ℓ and C) and the number of cycles (T) and registers (R) required by some structures based on the proposed class of processors (h, $\ell = 4, 8, 16$; $C = 1, 2, 4, 8$; $N = p = 16$).

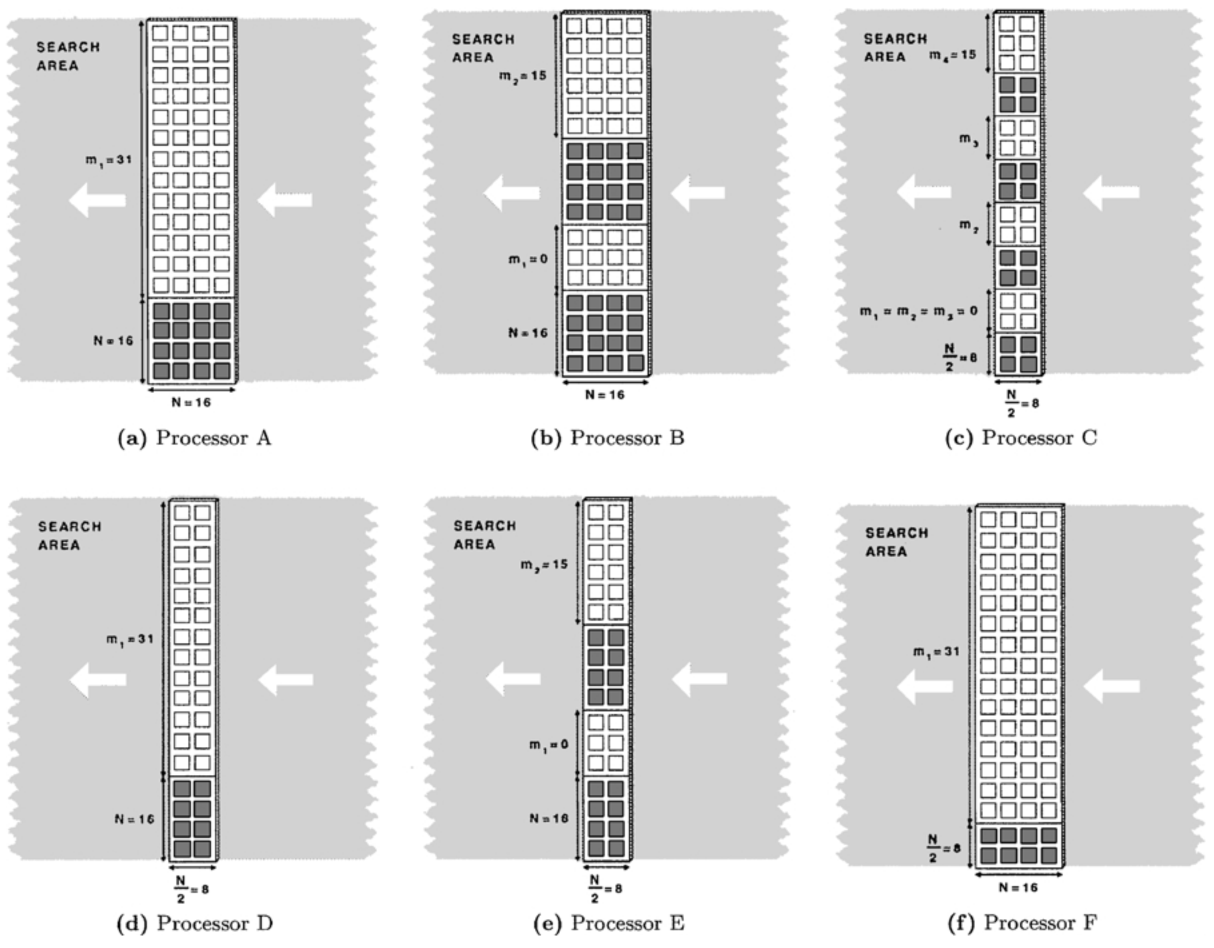


Figure 10. Examples of single and multi-array structures based on the proposed class of processors ($N = 16$; $p = 16$).

been reduced by using only $h = N/2$ rows of PEs (see Fig. 7). Consequently, the corresponding number of required clock cycles is $T_D = 2\hat{p}^2 = 2(2p)^2$.

- *Category III* ($R < 1, T < 1$). Processors of this category exhibit both smaller processing times and smaller hardware requirements than the processor proposed by Vos. One example of such processors is presented in Fig. 10(e). The amount of required hardware has been reduced by using two active blocks ($C = 2$) with only $h = N/2$ rows of PEs (see Fig. 7). The corresponding number of clock cycles is the same as processor A: $T_E = (2p)^2$.
- *Category IV* ($R > 1, T > 1$). Processors of this category present both greater processing times and greater hardware resources than the processor proposed by Vos. One example of such processors is presented in Fig. 10(f). It makes use of a single active block ($C = 1$) with $\ell = N/2$ columns of PEs (see Fig. 7). The corresponding number of clock cycles is given by $T_F = 2\,\hat{p}^2 = 2(2p)^2$.

Hence, by adjusting a small set of implementation parameters (h, ℓ and C), processing structures with distinct performance and hardware requirements are obtained, providing the ability to adjust the required hardware resources to the target implementation technology. Consequently, processors of category II (and III) are more suitable for applications where the implementation technology presents limited hardware resources, such as FPGAs. On the other hand, processors of category I (and III) are more suitable for applications where this limitation is not so strict, such as those implemented using ASIC or Sea-of-Gates technologies, and where greater performances with reduced processing times are required.

Table 1. Clock cycles and number of registers required to process one reference macroblock.

Processor	h	ℓ	C	T	R
A	N	N	1	$(2p)^2$	$6N^2 + 3N(2p-1)$
B	N	N	2	$\frac{1}{2}(2p)^2$	$9N^2 + 3N(2p-1)$
C	$\frac{N}{2}$	$\frac{N}{2}$	4	$(2p)^2$	$\frac{21}{2}N^2 + \frac{3}{2}N(2p-1)$
D	$\frac{N}{2}$	N	1	$2(2p)^2$	$4N^2 + \frac{3}{2}N(2p-1)$
E	$\frac{N}{2}$	N	2	$(2p)^2$	$\frac{13}{2}N^2 + \frac{3}{2}N(2p-1)$
F	N	$\frac{N}{2}$	1	$2(2p)^2$	$\frac{11}{2}N^2 + 3N(2p-1)$
Vos [6]	N	N	1	$(2p)^2 + N(N+2p-1)$	$6N^2 + 2N(2p-1)$

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 [11] is used. With such 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. Data stored in the so-called transparent layer is transferred to the processing layer as soon as the last candidate macroblock is processed. Therefore, not only does this structure pre-load reference macroblock data like Vos structure does (by using the so-called running-data registers), but it also enables a simultaneous pre-fetching of the search area data, making it possible to compute a new similarity measure in every clock cycle.

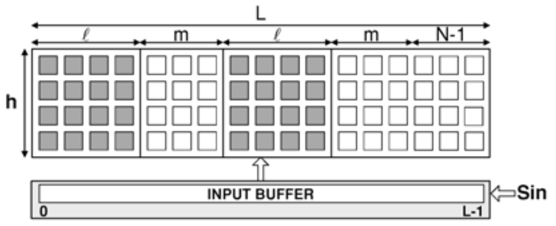
In Table 1 it is summarized the number of clock cycles and the number of registers required to process one reference macroblock by each of the processors illustrated in Fig. 10 (using the transparent transfer mode) and by the architecture proposed by Vos. In a typical implementation, with $h = \ell = 16$ and $C = 1 (N = p = 16)$, the pre-fetch layer approach represents an increase in the number of registers of only 19.6%. This increase of hardware resources can be easily tolerated if the achieved speedup (1.734) is taken into account [10].

4. Data-Flow Control

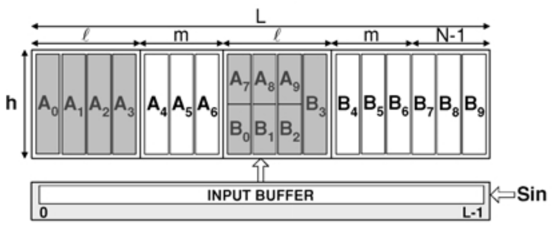
The desired data-flow between the several blocks of the processor can only be guaranteed through a careful design of the various control units that provide the set of control signals required by the several blocks of the processor. To simplify the design of these units, they were decomposed into local and simpler control circuits and synchronized through a restricted set of signals. The number of states of each controller was optimized to guaranty the tradeoff between the desired efficiency and 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 $(\lceil \frac{N}{h} \rceil \times \lceil \frac{N}{\ell} \rceil)$ 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.

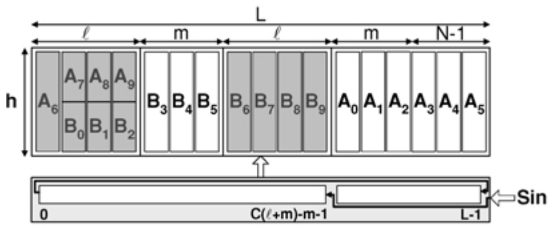
However, with the introduction of the new processing scheme based on the processor cylindrical structure, some measures had to be taken in order to provide the correct data to each PE. Figure 11(a) shows an example of a processor array with $C = 2$, $N = 4$ and $p = 7$. To simplify the explanation, the processor array was represented schematically in Fig. 11(b), where each column of the search area 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.



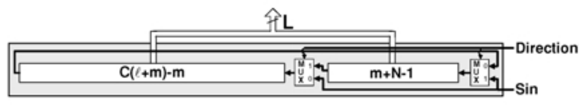
(a) Considered implementation.



(b) Correct alignment of the 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 area pixels.

Figure 11. Search area input buffer.

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 the pixels stored in the processing array. In fact, their alignment can only be guaranteed if the L registers are connected as it is illustrated in Fig. 11(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.

In Fig. 11(d) it is presented the search area input buffer and its alignment circuit. 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 the data being processed. This variable topology is accomplished by two multiplexers, making it possible to transfer in parallel the set of all L input pixels to the correct positions in the processing array.

To assure the correct function of this scheme, it is necessary to guarantee that these input buffers will never underflow under any circumstances. One possible solution is to use two different clock signals: the *read* and the *processing* clock signals. Considering a configuration using C processing cores, one realizes that L clock cycles (plus 3 extra clock cycles) are required to perform the read operation into the input buffers, whereas $\lfloor \frac{2p}{C} \rfloor$ clock cycles are needed to process the set of candidate macroblocks in each row of the search area by each active block. Assuming that the frequency of the *read* clock signal (f_{read}) is α times greater than the frequency of the *processing* clock (f_{proc}), one have:

$$t_{proc} \geq t_{read} \Leftrightarrow \left\lfloor \frac{2p}{C} \right\rfloor \geq \frac{L+3}{\alpha} \tag{10}$$

where $L = C\lfloor \frac{2p}{C} \rfloor + N - 1$. Thus,

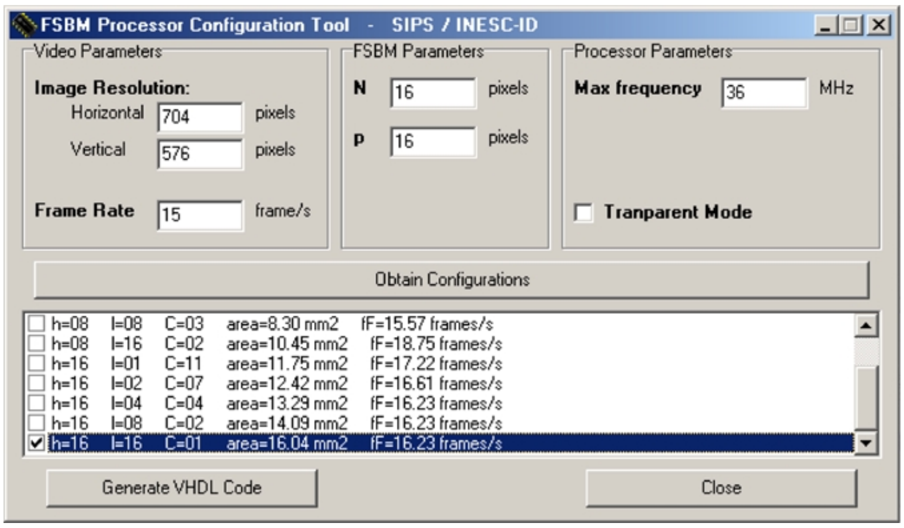
$$\alpha \geq \frac{C\lfloor \frac{2p}{C} \rfloor + N + 2}{\lfloor \frac{2p}{C} \rfloor}, \quad \alpha \in \mathbb{N} \tag{11}$$

$$\Leftrightarrow \alpha \geq \left\lceil C + \frac{N+2}{\lfloor \frac{2p}{C} \rfloor} \right\rceil \tag{12}$$

5. Automatic Synthesis

A software configuration tool has been developed to automatically determine the set of possible configurations of the proposed class of architectures which fulfill the requisites of a given video coder. This configuration tool, whose graphical interface is illustrated in Fig. 12, receives the following set of parameters: the image resolution and the required frame rate; the dimension of the reference macroblock (N) and the maximum displacement in each direction (p); the processor maximum operating frequency and a flag indicating if the transparent transfer mode based on the referred prefetch layer [10] is to be used. These parameters are then used to compute the required number of active PEs to be implemented in each line and column of the active blocks (h, ℓ) and the number of processing cores (C) that will compute, in parallel, the SAD similarity measure.

For each possible configuration, this application outputs the effective maximum frame rate (f_F) and may also supply the required semiconductor area, calculated with a priori information about the used target technology (input file). As soon as the selection of the desired



Video Parameters
Image Resolution:
Horizontal: 704 pixels
Vertical: 576 pixels
Frame Rate: 15 frame/s

FSBM Parameters
N: 16 pixels
p: 16 pixels

Processor Parameters
Max frequency: 36 MHz
☐ **Tranparent Mode**

```
☐ h=08  l=08  C=03  area=8.30 mm2   fF=15.57 frames/s
☐ h=08  l=16  C=02  area=10.45 mm2  fF=18.75 frames/s
☐ h=16  l=01  C=11  area=11.75 mm2  fF=17.22 frames/s
☐ h=16  l=02  C=07  area=12.42 mm2  fF=16.61 frames/s
☐ h=16  l=04  C=04  area=13.29 mm2  fF=16.23 frames/s
☐ h=16  l=08  C=02  area=14.09 mm2  fF=16.23 frames/s
☑ h=16  l=16  C=01  area=16.04 mm2  fF=16.23 frames/s
```

Figure 12. FSBM processor configuration tool.

configuration has been carried out by the user, this tool automatically generates the complete description of the corresponding full-search block-matching processor, using synthesizable IEEE-VHDL description language. The generated VHDL code is then synthesized using *Synopsys* synthesis tools or *Cadence* design tools, in order to implement the selected configuration in the target technology.

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, making extensive use of '`generic`' configuration inputs. Moreover, the VHDL code generated by this tool presents a fully structural description of the main processing blocks.

Hence, this configuration tool provides the means to automatically generate and synthesize any processor configuration that is able to fulfill the performance levels required by a given video coder. This processor can then be implemented using any target technology, such as ASIC or FPGA.

6. Experimental Results

One of the main objectives of the presented research was the development of a fully parameterizable and technology independent class of full-search block-matching architectures, which supports automatic synthesis of efficient motion estimation processors. As it was shown in Figs. 9 and 10, this class has the capability to provide several different and scalable structures, in order to adjust the required amount of hardware resources to the target implementation technology.

Two different technologies have been considered: Field-Programmable Gate Arrays (FPGA) and Standard Cell based ASICs. The implementation based on an FPGA makes use of a Virtex XCV600 device from Xilinx [12], while the ASIC implementation was performed using the Diplomat-25 standard cell library, based on the UMC 0.25 μm CMOS technology process from Virtual Silicon Technology Inc. [13].

Among the several blocks of the full-search block-matching processor, the PE array, composed by the active and passive PEs, is the block that uses the most significant part of the hardware resources. Moreover, the operating frequency of the processor is greatly dependent on the critical path of this block. Consequently, an early and coarse estimation of the total amount of hardware resources required by a given configuration of the proposed class of processors, as well as the corresponding maximum operating frequency, can be obtained from the implementation characteristics of the active and passive PEs. These characteristics are presented in Tables 2 and 3. As it was expected, the implementation based on the standard cell library is able to offer greater performance levels than the FPGA based implementation. However, the later provides significant advantages in what concerns its reconfigurable characteristics, which are often of great interest

Table 2. Implementation characteristics of the active and passive PEs using a Virtex XCV600 FPGA.

	Active PE	Passive PE
Number of CLB slices:	72	32
–Slice Flip-Flops:	75	43
–4 input LUTs:	73	16
Minimum period:	23.831 ns	9.481 ns

Table 3. Implementation characteristics of the active and passive PEs using a standard cell library based on a 0.25 μm CMOS technology.

	Active PE	Passive PE
Area:	32,184.1 μm^2	14,760.1 μm^2
Minimum period:	3.44 ns	0.64 ns

to easily adjust the configuration to a given target application.

It is worth noting that the maximum operating frequency obtained from the implementation of a given configuration using one of the considered technologies will naturally lead to values that are significantly lower than those obtained from the minimum periods presented in Tables 2 and 3. The reason for this reduction are the extra latency delays arisen from the routing stages of the implementation procedure.

In what concerns the FPGA implementation, the maximum operating frequency obtained for a configuration similar to processor D illustrated in Fig. 10(d), with $h = 4$, $\ell = 8$ and $C = 1$, and considering $N = p = 8$, is about 10 MHz. This processor provides the ability to perform motion estimation in QCIF (176×144) image sequences at a rate up to 7.2 fps. In what concerns the device utilization rate, the number of CLB slices required by each active and passive PE represent 1.04% and 0.46% of the total amount of CLB slices available in this device (6912). Consequently, one should expect that processor configurations with greater levels of parallelism will require the utilization of FPGA devices with greater hardware resources, such as the Virtex XCV800 or XCV1000 [12]. The number of I/O signals required by this configuration is about 50, which can be easily accommodated by any of the referred devices.

A complete full-search block-matching processor based on the proposed class of architectures was designed with *Synopsys* synthesis tools and *Cadence* design tools. The implemented configuration is composed

Table 4. Implemented processor characteristics.

(a) Algorithm parameters	
Algorithm	FSBM
Block size*	16×16
Search range*	$-15, +16$
(b) Implementation results	
Technology	CMOS—0.25 μm
Process	UMC—1P5M
Supply voltage	2.5 V/3.3 V
Active PE area	32,184.1 μm^2
Die area	16.07 mm^2
Maximum frequency	36.5 MHz
Maximum resolution	4CIF/16 fps

*Configurable.

by a single active block ($C = 1$) with 16×16 PEs ($N = 16$) and is characterized by a search range from -15 to $+16$ pixels ($p = 16$), corresponding to the set of parameters typically adopted by ITU-T H.26x and ISO MPEG standards (see Table 4(a)). The implementation results of this configuration are presented in Table 4(b). The chip is able to deliver 28.6GOPs at 36.5 MHz over typical voltage and temperature ranges. In each clock cycle, each of the N^2 active PEs computes one difference, one absolute value and one accumulation operation; each of the $2^{\log_2 N} - 1$ adder-tree PEs computes one addition and the comparator unit performs one comparison. This implemented processor is able to perform motion estimation in 4CIF (704×576) image sequences at a rate up to 16 frames per second.

Consequently, we conclude that ASIC based implementations are more suitable for motion estimation in high-resolution video sequences, while FPGA based processors may be preferable in applications requiring reconfigurable capabilities to process lower-resolution image sequences at moderate frame rates.

7. Conclusions

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 substantial improvements in the AB2 single array architecture for the full-search block-matching algorithm, as well as the development of a new processing scheme. These significant enhancements led to the minimization of its latency, the maximization of its throughput and a full utilization of the hardware resources, avoiding the need for extra clock cycles to displace search area pixels inside the array.

The new class of processors provide the capability of further speeding up the estimation of motion vectors, by computing in parallel several similarity measures using multiple and independent processing cores. Optimized implementations of these processing structures can be obtained by means of a software configuration tool, which was developed to provide the ability to automatically synthesize 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 edn., Kluwer Academic Publishers, 1997.
2. Y. Ooi, "Motion Estimation System Design," in *Digital Signal Processing for Multimedia Systems*, K.K. Parhi and T. Nishitani (Eds.), Marcel Dekker, Inc., 1999, chap. 12, pp. 299–327.
3. P. Pirsch, N. Demassieux, and W. Gehrke, "VLSI Architectures for Video Compression—A Survey," *Proceedings of the IEEE*, vol. 83, no. 2, 1995, pp. 220–246.
4. S. Kittitornkun and Y.H. Hu, "Frame-Level Pipelined Motion Estimation Array Processor," *IEEE Transactions on Circuits and Systems for Video Technology*, vol. 11, no. 2, 2001, pp. 248–251.
5. T. Komarek and P. Pirsch, "Array Architectures for Block Matching Algorithms," *IEEE Transactions on Circuits and Systems*, vol. 36, no. 10, 1989, pp. 1301–1308.
6. L. Vos and M. Stegherr, "Parameterizable VLSI Architectures for the Full-Search Block-Matching Algorithm," *IEEE Transactions on Circuits and Systems*, vol. 36, no. 10, 1989, pp. 1309–1316.
7. C.H. Hsieh and T.P. Lin, "VLSI Architecture for Block Matching Motion Estimation Algorithm," *IEEE Transactions on Circuits and Systems for Video Technology*, vol. 2, no. 2, 1992, pp. 169–175.
8. S. Chang, J.H. Hwang, and C.W. Jen, "Scalable Array Architecture Design for Full Search Block Matching," *IEEE Transactions on Circuits and Systems for Video Technology*, vol. 5, no. 4, 1995, pp. 332–343.
9. S.Y. Kung, *VLSI Array Processors*, Prentice Hall, 1988.
10. N. Roma and L. Sousa, "Implementation Aspects of MESA Processor," Technical Report RT/001/2001, INESC-ID, Lisboa, Portugal, 2001.
11. N. Roma and L. Sousa, "A New VLSI Architecture for Full Search Block Matching," in *IFIP International Conference on Very Large Scale Integration (VLSI-SoC'2001)*, Montpellier, France, 2001, pp. 213–218.
12. Xilinx, "Virtex 2.5V Field Programmable Gate Arrays—Product Specification," Xilinx Inc., 2001.
13. VST, "Diplomat-25 Standard Cell Library–0.25 μm UMC Process," Virtual Silicon Technology Inc., 1999.

Nuno Roma received his 5-year degree (licentiature) in electrical engineering from Instituto Superior Técnico (IST), Lisbon, Portugal, in 1998. In 2001 he received the M.Sc. degree in electrical engineering from IST, with a merit scholarship from the Electrical and Computer Engineering department. Currently, he is pursuing the Ph.D. degree in electrical engineering at IST and is developing his research with the Signal Processing Research Group (SIPS) at Instituto de Engenharia de Sistemas e Computadores-Investigação e Desenvolvimento (INESC-ID). His research interests include application specific architectures for signal processing, image processing, video coding/transcoding and compressed-domain video processing.
nuno.roma@inesc-id.pt

Leonel Augusto Seabra Sousa received the MS and Ph.D. degrees in Electrical and Computer Engineering from Instituto Superior Técnico (IST), Lisbon, Portugal, in 1989 and 1996, respectively. He is currently an assistant professor at the Electrical and Computer Engineering department of IST and he is one of the leaders of the Signal Processing Systems research group (SIPS) at Instituto de Engenharia de Sistemas e Computadores-Investigação e Desenvolvimento (INESC-ID). His research interests include signal processing and parallel computing, in particular efficient algorithms and specific architectures for image processing and video coding. Dr. Leonel Sousa is a member of the IEEE.
las@inesc-id.pt