A New Efficient VLSI Architecture for Full Search Block Matching Motion Estimation

Nuno Roma and Leonel Sousa
Instituto Superior Técnico / INESC-ID, Lisboa, Portugal

Abstract: A new efficient *type I* architecture for motion estimation in video sequences based on the Full-Search Block-Matching (FSBM) algorithm is proposed in this paper. This architecture presents minimum latency, maximum throughput and full utilization of the hardware resources, combining both pipelining and parallel processing techniques. The implementation of an array processor for motion estimation in a single-chip using 0.25 μm CMOS technology is presented. Experimental results show that this processor is able to estimate motion vectors in 4CIF video sequences at a rate of 16 frames/s.

Key words: Motion Estimation, Block Matching, Array Architectures, Specialized Processors.

1. INTRODUCTION

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

Motion estimation requires a huge amount of computations. Consequently, a great research effort has been made to develop efficient dedicated structures and specialized processors [6]. Due to their regular processing scheme and simple control structures, FSBM algorithms have

been the most widely used in VLSI implementations, providing optimal estimation results and leading to fast and efficient processing structures. Moreover, the sum of absolute differences (SAD) matching criteria has been extensively applied in these processors, due to its simplicity and satisfactory results.

Several different structures have been proposed over the last few years [4][9][3][2]. Most of them are 2D or 1D arrays derived from the Dependence Graph (DG) of the FSBM algorithm. However, a comparative analysis of these architectures shows that none of them provides a maximum and constant throughput or a full utilization of the hardware resources.

The main goal of the research presented in this paper is the analysis and development of a new and efficient array architecture for motion estimation in video sequences based on the FSBM algorithm. This new architecture uses the *AB2 type* architecture proposed by Vos [9] and its peculiar processing scheme as the basis for the present research. In fact, it will be shown that Vos' architecture can be significantly improved in what concerns both the latency and the hardware requirements. The amount of memory used to store the search area data can be substantially reduced through a full utilization of the hardware resources. Moreover, the time wasted to fill the processor pipeline whenever a new reference block and search area are needed can be avoided, by introducing in the architecture an extra layer with pre-fetch registers.

The proposed architecture was described using fully parameterizable IEEE-VHDL code and its functionality was thoroughly tested. Fast arithmetic units for addition and absolute difference computation were also designed based on prefix-adder and binary adder-tree structures. An integrated circuit for motion estimation was developed, by making use of the proposed architecture and using a standard cell library of a CMOS - 0.25 μm technology process. Experimental results show that the implemented circuit is able to estimate motion vectors in 4CIF video sequences at a rate of 16 frames/s.

2. FSBM ARCHITECTURES

In this section, the efficiency of the main systolic architectures proposed over the last few years for the FSBM algorithm is compared. Those architectures can be regarded as regular arrays of Processor Elements (PEs), where each PE computes the SAD similarity measure. The number of PEs that composes the array defines the concurrency level of the estimation process, which is usually dependent on the used performance versus circuit area trade-off.

```
(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 do                {N × N reference macroblock}
         for v = 0 to N 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)}
```

Figure 1. FSBM algorithm using the SAD similarity function.

The FSBM algorithm can be described using the four nested loops presented in *Figure 1*: the inner (*u*,*v*) loops perform the matching calculation for a given candidate macroblock, while the outer (*c*,*l*) loops are responsible for the displacements inside the search area, to test all the considered candidate macroblocks. The specific characteristics of a given FSBM architecture are defined by the considered configuration and by the set of loops that are executed in parallel. Assuming, for example, that the index *l* in the algorithm of *Figure 1* is set to a fixed value and that each PE performs the primitive operation *SAD*, a 3D DG can be derived [5]. Systolic structures can be derived by applying the usual operations to project the DG and obtain array structures defined in lower dimensional spaces: index projection, time scheduling and graph folding [5]. Architectures are usually classified according to the set of projections performed, giving rise to 1D structures if multi-projection techniques are applied. Their execution time is dependent on the specific arrangement of data supply and on the number of projections performed in the retiming procedure.

One of the first discussions about FSBM architectures classification was presented by Komarek and Pirsch [4]. They discussed the characteristics of a set of 2D and 1D arrays, obtained by reducing the dimension of the original DG using traditional index projection, time scheduling and graph folding techniques [5]. The main difference between these arrays is the exploited processing concurrency, implying the usage of different structures and different number of PEs (#*PE*). The so called *type I - AB2* bidimensional structure requires $\#PE=N^2$, while the *AS2 type* array uses $\#PE=N\times(2p+1)$. By projecting the DG twice, one-dimensional arrays are obtained, such as the *AB1* structure, with $\#PE=N$, and the *AS1* structure, with $\#PE=2p+1$.

Vos and Stegherr [9] proposed an improved version of the *type I - AB2* two-dimensional structure, which presents some significant advantages in

what concerns the processing time. They also proposed another structure that was obtained by reversing the processing order of the four loops of the FSBM algorithm: by indexing the inner loops with the variable pair (c,l), all candidate macroblocks are considered whenever a different pixel of the reference macroblock is read from the frame memory at a given clock cycle. Another *type I* architecture was proposed by Hsieh [3], with some improvements in what concerns the transfer of data into the processing circuit. In his proposal, the candidate macroblock pixels are supplied as a series of one-dimensional data through a set of delay elements. Chang [2] 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 two dimension (u,v) projection (slice) in the (c,l) two-dimensional space (tiling). Under certain particular conditions, Chang's model provides a hardware utilization rate very close to the optimal (100%). However, this is only possible if multiple data input lines are used.

Array processors can be classified according to their performance level, which is related to the required number of clock cycles (T) to estimate the motion vectors. This last measure is usually the most important figure of merit used to compare architectures intended to work in real time. The values of #*PE* and T of the referred architectures are presented in *Table 1*. For comparison purposes, it was also considered the limit situation corresponding to a processor array with a single PE, which was designated by *SinglePE* architecture. It is worth noting that, in practice, the real values of T can be significantly greater than those presented in *Table 1*. Frequently, extra clock cycles are necessary to fill the pipeline and dummy results are often computed to preserve a regular data flow.

The circuit area (A*) and the processing time (T*) of the considered architectures were estimated by parameterizing the set of expressions presented in *Table 1* in terms of $k = p/N$. The obtained results are presented in *Figure 2* and *Figure 3*, respectively, using logarithmic scales to accommodate the large range of values.

Table 1. 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

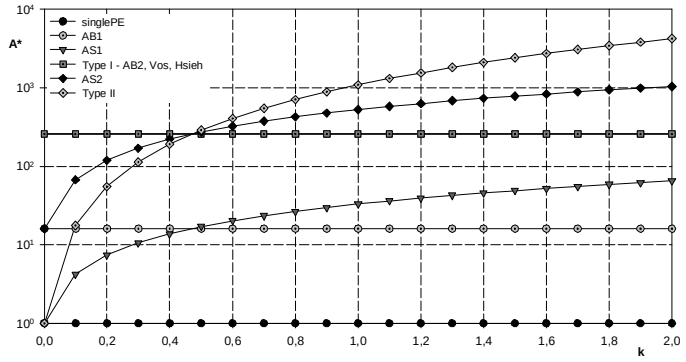


Figure 2. Circuit area (A*) in function of $k=p/N$ (N=16).

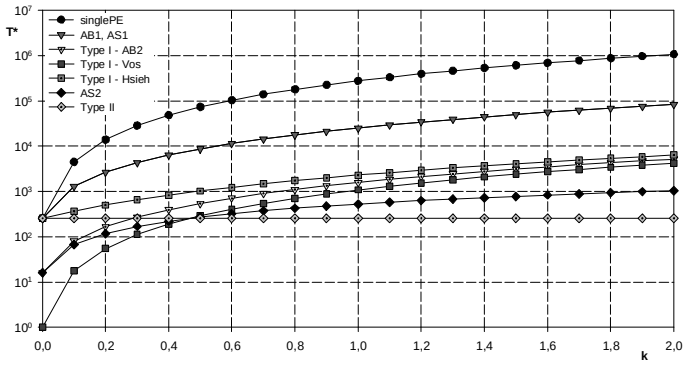


Figure 3. Processing time (T*) in function of $k=p/N$ (N=16).

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 formers offer advantages for greater search areas. In what concerns the processing time, while for most architectures it increases with the search window size, it remains constant for the *type II* structure. This result was already expected, since one PE is used to compute the similarity measure of each candidate macroblock.

Among all these array structures, the *type I* architecture proposed by Vos and Stegherr [9] was recognized as being one of the most efficient structures [7]. Its main advantages are the short processing time and the limited amount of required hardware resources, when compared with the other bidimensional structures. However, this architecture still has some non-exploited features, which can be used to significantly improve its efficiency in terms of hardware requirements and parallelism level. In the next section, a new efficient array architecture is proposed.

3. A NEW ARRAY ARCHITECTURE

The proposed architecture, designated by *"New-AB2"*, is based on *Vos* architecture but presents some significant improvements in two different aspects: *i)* processor structure; *ii)* data transfer. Due to the similarities between the processing schemes of these two architectures, the description of the proposed structure is done by contrasting its optimized characteristics with those presented by Vos and Stegherr [9]. Therefore, references to *Vos* architecture will be done whenever it shows to be convenient.

3.1 Processor Structure

The diagram shown in *Figure 4* illustrates the main differences between the architecture proposed by Vos, represented using solid and dotted style lines (——), and the *New-AB2* architecture, represented with solid and dot-dashed style lines (—·—).

Like other *type I* bidimensional 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 *Figure 4*). Each passive PE is composed by running-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

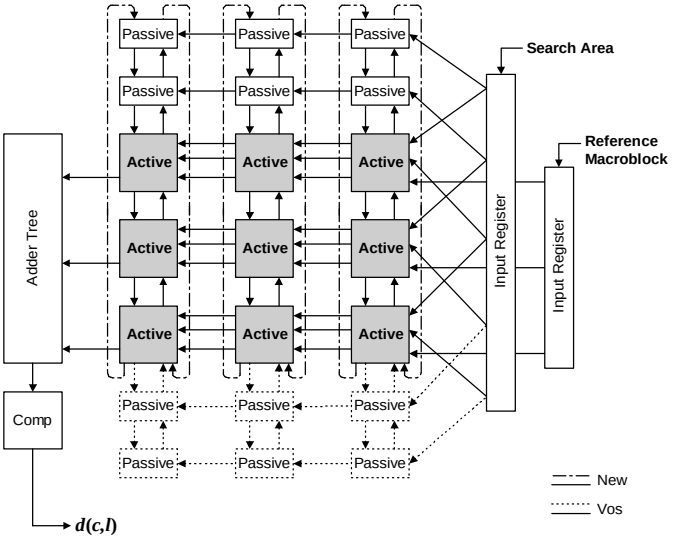


Figure 4. Type I processor array for FSBM motion estimation, considering each reference macroblock with $N \times N$ pixels ($N = 3$) and the search area composed by $(N+2p) \times (N+2p)$ pixels ($p = 1$).

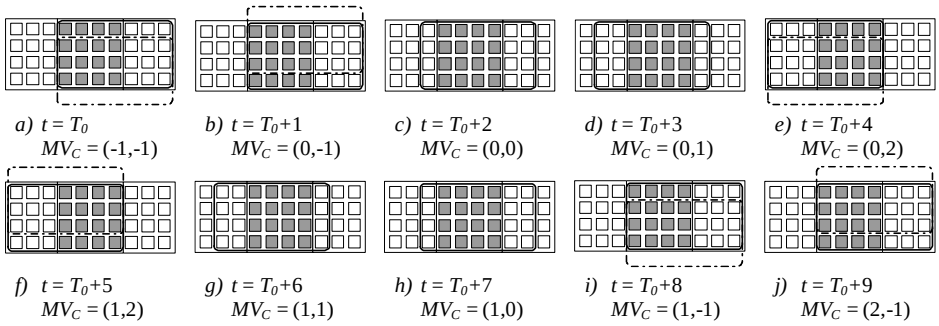
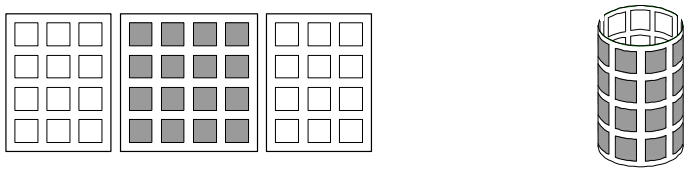


Figure 5. Zig-zag data flow of search area pixels in *Vos* architecture (N=4, p=2).

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 next $2p+1$ clock cycles, search area data is shifted downwards one position per cycle. Meanwhile, the pixels corresponding to different candidate macroblocks are transferred through the several active PEs, which provide one SAD 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 next $2p+1$ 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, preventing the need for dummy clock cycles between two adjacent lines of the search area. These extra cycles are often required by other architectures to displace search area data inside the array [4][3].

The processing scheme of *Vos* architecture can be represented in a simplified way by the sequence of states shown in *Figure 5*. The fraction of the search area being processed by the structure 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 search fractions leaving the array, corresponding to the start of the search procedure in a new row of candidate macroblocks.



a) *b)*

Figure 6. Rearrangement of the processor array: (a) - planar processor; (b) - the pair of passive blocks is superimposed by disposing the processor over a cylindrical surface.

From *Figure 5* it is possible to realize that in an array composed by N^2 active PEs and by $2\times N(2p\text{-}1)$ passive PEs, used to process search fractions with $N\times(N+2p\text{-}1)$ pixels, half of the total amount of passive PEs, $N\times(2p\text{-}1)$, are not being used. 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 *Figure 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, it is worth noting that the zig-zag processing scheme can still be applied to this modified structure, preserving the properties of *Vos* architecture but keeping all PEs busy at any instant.

A simplified block diagram of the proposed *new-AB2* structure is presented in *Figure 7*. The cylindrical structure of *Figure 6(b)* is obtained by connecting the passive PEs located on the right margin of the passive block with the active PEs of the left margin of the active block, as it was shown in *Figure 4*. The processing scheme of this architecture is shown in *Figure 8* for the same setup of *Figure 5*.

Contrasting with the architecture proposed by Vos, this structure does not require the usage of passive PEs not carrying useful data at some clock

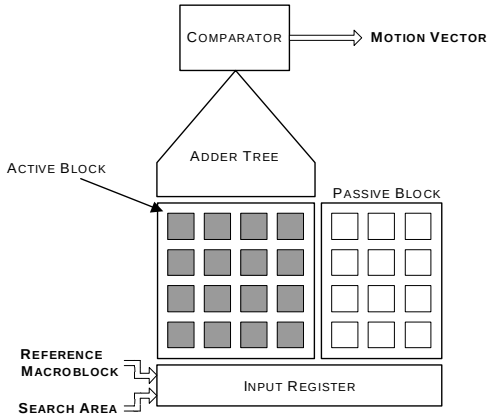


Figure 7. Simplified diagram of the proposed *new-AB2* structure.

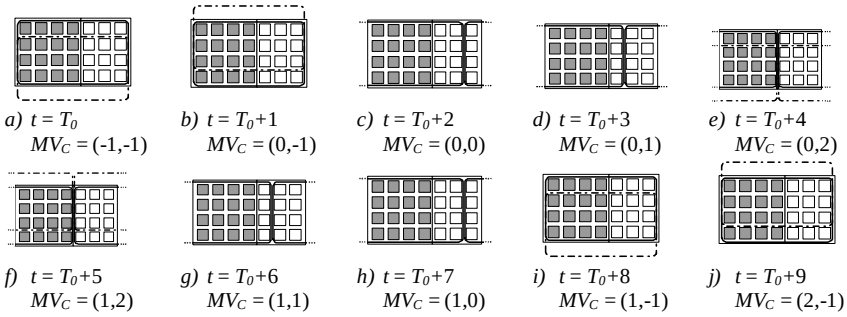


Figure 8. Zig-zag data flow of search area pixels in the proposed *new-AB2* architecture (N=4, p=2).

cycles. Moreover, the zig-zag processing scheme of *Vos* architecture is preserved, thus maintaining its recognized efficiency properties. However, while *Vos* architecture requires $[N+2\times(2p-1)]\times N$ registers, in the proposed architecture only $[N+(2p-1)]\times N$ registers are necessary. The chart presented in *Figure 9* shows the variation of the number of registers required by both structures to perform the displacement of search area pixels, by considering N=16 and $k=p/N$. The line-chart represented with the □ marks shows the relation between the number of registers required by both architectures. This relation is about 60% for k=1 ($p=N$) and 55% for k=2 ($p=2N$).

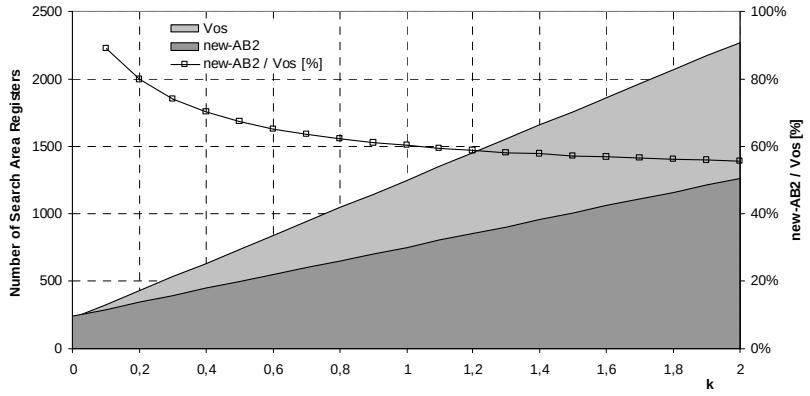


Figure 9. Relation between the required number of displacement registers in *Vos* architecture and in the *new-AB2* architecture.

3.2 Data Transfer

Many motion estimation architectures require extra clock cycles between the processing of adjacent lines of the search area to displace the search data inside the array [4][3]. Moreover, additional clock cycles are often spared in many architectures between the processing of consecutive reference

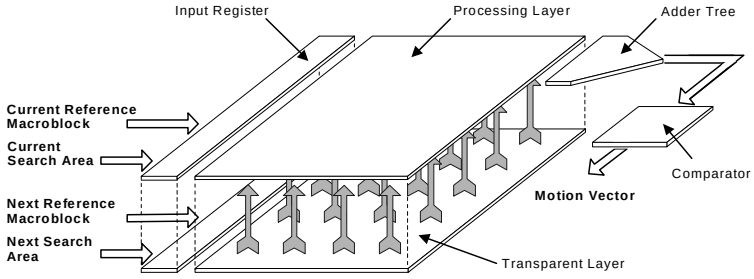


Figure 10. Prefetch layer used to load the internal registers in transparent mode.

macroblocks, to insert and remove unused or already processed data from the array [9][3]. In both situations, these extra clock cycles often lead to the loss of a significant amount of time.

Although the zig-zag processing scheme proposed by Vos provides the means to avoid the time loss associated with the displacement of search data inside the array, it does not prevent from sparing extra clock cycles between the processing of consecutive reference macroblocks. To attenuate this problem, Vos proposed the usage of running-data registers to store the pixels corresponding to the next reference macroblock while the current reference macroblock was being processed in the so called standing-data registers. When this processing is concluded, the next reference macroblock can then be instantaneously transferred from the running-data registers to the standing-data registers.

However, this transfer mechanism is not sufficient to eliminate the need for extra clock cycles, since search area data still has to be loaded into the array. To circumvent this limitation, a new data transfer method based on the usage of an additional pre-fetch layer (see *Figure 10*) is now proposed. With such a structure, it is now possible to pre-load both the reference and part of the search area 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 preload the reference macroblock data like *Vos* structure does (by using the so-called running-data registers), but it also enables a simultaneous prefetching of the search area data, making it possible to compute a new SAD similarity value in every clock cycle.

It is worth noting that this improved transfer scheme does not imply any increase of the data input bandwidth. In fact, during the processing of a given reference macroblock, it is now necessary to load $(N+2p-1)\times(2p-1)$ pixels corresponding to the current search area, $(N+2p-1)\times N$ pixels corresponding to the next search area and N^2 pixels of the next reference macroblock, which is exactly the same amount of data that would be

required if no transparent layer was used. However, this efficiency improvement implies the usage of some more registers: with the proposed *new-AB2* structure $2\times[N+(2p-1)]\times N$ registers are required, while with the original *Vos* architecture $[N+2\times(2p-1)]\times N$ registers are used, thus leading to an increase of N^2 registers. In a typical implementation, with N=16 and k=1 (p=N=16), it represents an increase of only 20.5%, which can be easily tolerated if the achieved performance gains are taken into account.

4. EXPERIMENTAL RESULTS

The desired efficiency level of the proposed processor can only be achieved if its implementation is carried out in conjunction with a careful study of the blocks that more significantly affect its overall performance, trying to minimize its processing critical path. The most time consuming operations performed by the processor are those involved in the computation of the SAD similarity measure: addition, subtraction and absolute value computation. Consequently, gate level optimizations must be considered in order to obtain the shortest critical path as possible [8].

The description of the several blocks that compose the processor was carried out using IEEE-VHDL description language. To achieve the required characteristics in what concerns the processor configurability, this description was focused on easily obtaining fully parameterizable VHDL code, by making extensive use of `'generic'` type configuration inputs. Furthermore, in order to achieve the required optimization levels of the several processing structures, a fully structural description of the main blocks of the processor was carried out by using the most elementary logic operations provided by the implementation library.

A FSBM chip based on the proposed architecture was designed with Synopsys synthesis tools and Cadence design tools, using a standard cell library based on a 0.25 μm CMOS technology process. The implemented processor is composed by 16×16 active PEs (N=16), with a search range from -15 to +16 pixels (p=16). Since the active PE is the most important module, a careful optimization procedure was carried out in terms of area and speed. 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 processor are presented in *Table 2* and *Table 3*.

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. 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 computes one comparison.

Table 2. Algorithm characteristics.

Algorithm	FSBM
Block size*	16 × 16
Search range*	-15, +16
Max. resolution*	4CIF 16 frames/s

* - configurable

Table 3. Chip characteristics.

Process	0.25 μm CMOS – 1P5M
Supply voltage	2.5V / 3.3V
Die size	4 × 4 mm
Active PE area	32,184.2 μm^2
Max. frequency	36.5 MHz
Pin count	56

5. CONCLUSIONS

A new efficient *type I* architecture for motion estimation in video sequences was proposed in this paper. This architecture presents minimum latency, maximum throughput and full utilization of the hardware resources. These optimized characteristics were achieved through the development of a new processing scheme for the processor array and through the introduction of an extra pre-fetch layer to avoid the need for extra clock cycles to transfer the data between the processor and the video coding system. Experimental results proved that this architecture can be used to implement the existing ITU-T H.26x and ISO MPEG video coding standards, with configurable search ranges and video quality tradeoffs. The implemented processor is able to estimate motion vectors in 4CIF video sequences at a rate of 16 frames/s.

REFERENCES

[1] V. Bhaskaran and K. Konstantinides., *Image and Video Compression Standards: Algorithms and Architectures*, Kluwer Academic Publishers, 2nd edition, June 1997.

[2] 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*, 5(4):332-343, August. 1995.

[3] C. H. Hsieh and T. P. Lin., VLSI Architecture for Block Matching Motion Estimation Algorithm, *IEEE Transactions on Circuits and Systems for Video Technology*, 2(2):169-175, June 1992.

[4] T. Komarek and P. Pirsch, Array Architectures for Block Matching Algorithms, *IEEE Transactions on Circuits and Systems*, 36(10):1301-1308, October. 1989.

[5] S. Y. Kung, *VLSI Array Processors*, Prentice Hall, 1988.

[6] Y. Ooi, *Motion Estimation System Design*, in K. Parhi and T. Nishitani (eds.): *Digital Signal Processing for Multimedia Systems*, Marcel Dekker Inc., chap. 12: 299-327, 1999.

[7] K. K. Parhi and T. Nishitani, editors, *Digital Signal Processing for Multimedia Systems*, Marcel Dekker, Inc., 1999.

[8] N. Roma, L. Sousa, Implementation Aspects of MESA Processor, Technical Report, INESC-ID, RT/001/2001, January, 2001.

[9] L. Vos and M. Stegherr, Parameterizable VLSI Architectures for the Full-Search Block-Matching Algorithm, *IEEE Transactions on Circuits and Systems*, 36(10):1309-1316, October 1989.