PARAMETERIZABLE HARDWARE ARCHITECTURES FOR AUTOMATIC SYNTHESIS OF MOTION ESTIMATION PROCESSORS

Nuno Roma **Leonel Sousa**
Nuno.Roma@inesc-id.pt *las@inesc-id.pt*

Instituto Superior Técnico / INESC-ID
Department of Electrical Engineering
Rua Alves Redol, No. 9 - 1000-029 Lisboa - PORTUGAL
Tel. +351 21 3100300 - Fax: +351 21 3145843

Abstract - A new class of fully parameterizable multiple array architectures for motion estimation (ME) in video sequences based on the Full Search Block Matching (FSBM) 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 processors according to the setup parameters, the processing time and the circuit area specified limits. With this purpose, a software configuration tool has been implemented to determine the set of possible configurations which fulfill the requisites of the video coder, providing the ability to automatically generate the VHDL description of the selected configuration. The implementation of a single array processor configuration on a single-chip is presented. Experimental results evidence the ability to estimate motion vectors in real-time with this configuration.

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], exploring 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.

As a consequence of the huge amount of computations required by ME, a great research effort has been made to develop efficient dedicated structures and specialized processors [2]. Due to its regular processing scheme and simple control structures, FSBM algorithms, using the sum of absolute differences (SAD) matching criteria, have been the most used in VLSI implementations, providing optimal estimation results and leading to fast and efficient processing structures. One of the first discussions about FSBM architectures and their classification was presented by Komarek and Pirsch [3]. Komarek presented the characteristics of a set of 1-D and

2-D arrays, obtained by reducing the dimension of the original dependence graph using traditional index projection, time scheduling and graph folding techniques [4]. Their main difference is the explored processing concurrency, implying the usage of different structures and different number of Processor Element (PE)s.

Among all these systolic structures, the AB2 2-D architecture proposed by Vos [5, 6] has been regarded as one of the most efficient structures [7]. Its peculiar processing scheme provides it with a short processing time and a limited amount of hardware resources. However, it still has some non-explored features, which can be used to significantly improve its efficiency and parallelism levels. Therefore, this architecture was selected as the basis for the presented research and it will be shown that significant improvements in what concerns its hardware requirements can be obtained, since the amount of memory used to store the search area data can be substantially reduced to achieve a full utilization of the hardware resources. Moreover, the proposed new single array architecture will be extended to multiple array architectures, by exploring the parallelism and lack of data dependencies provided by the ME procedure. A new class of parameterizable array architectures was 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. An integrated circuit based on the proposed class of architectures has been developed using a standard cell library of a CMOS - $0.25\ \mu\mathrm{m}$ technology process. Experimental results evidence the possibility to estimate motion vectors in 4CIF video sequences at a rate upto 16 frames/s with this implemented configuration.

NEW-AB2 SINGLE ARRAY ARCHITECTURE

As it was referred, the proposed new single array architecture, designated by *New-AB2*, presents some significant improvements in what concerns the utilization of the hardware resources. Due to the similarities between the processing schemes of this architecture and the architecture proposed by Vos, its description will be done by contrasting its optimized characteristics with those presented by Vos and Stegherr [5, 6].

Processor structure

The diagram shown in fig. 1 illustrates the main differences between the architecture proposed by Vos, represented with solid and dotted style lines ( ), and the *New-AB2* architecture, represented with solid and dot-dashed style lines ( ). Like in other AB2 structures, each pixel of the reference macroblock (MB) 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 running-data registers for the displacement and storage of search area pixels. Both the reference MB and the search area pixels are transfered into the processor through two vertical input

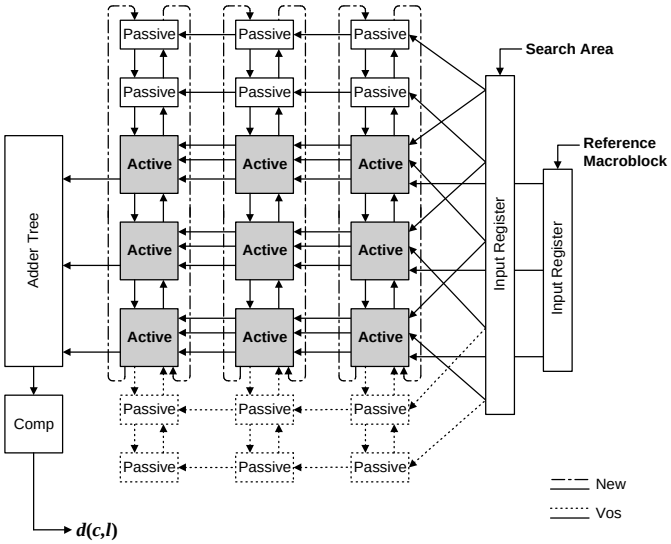


Figure 1: AB2 processor array for FSBM-ME, considering each reference MB with $N \times N$ pixels $(N = 3)$ and the search area composed by $(2p + N) \times (2p + N)$ pixels $(p = 1)$.

register chains, with length N and $2p + N$, respectively.

Within the PE array, search area pixels can be displaced in 3 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 MBs 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 two adjacent lines of the search area. These extra cycles are often required by other architectures to displace search area data inside the array [3, 8].

The processing scheme of Vos' architecture can be represented, in a simplified way, by the sequence of states shown in fig. 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 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.

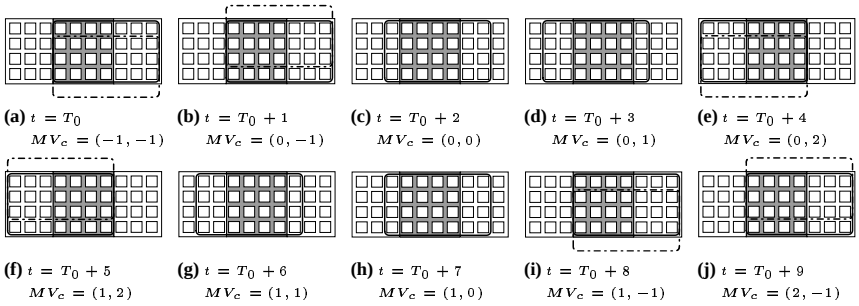


Figure 2: Zig-zag data flow of search area pixels on Vos architecture ($N = 4$, $p = 2$).

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 $N \times (N + 2p - 1)$ pixels, half of the total amount of passive PEs 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 fig. 3. 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.

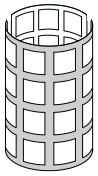


Figure 3: Proposed architecture disposed over a cylindrical surface.

A simplified block diagram of the proposed structure is presented in fig. 4. The cylindrical structure of fig. 3 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 fig. 1. The processing scheme of the proposed *new-AB2* architecture is shown in fig. 6, using the same setup of fig. 2. Contrasting with

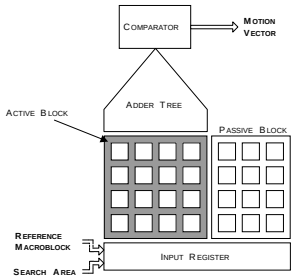


Figure 4: Simplified diagram of the proposed *new-AB2* structure.

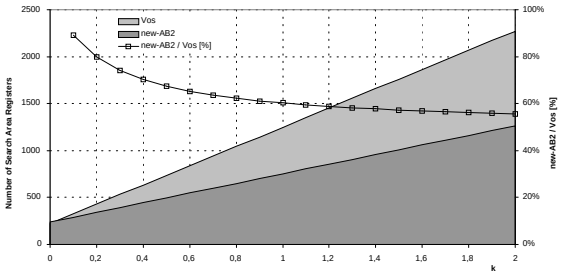


Figure 5: Relation between the required number of displacement registers in Vos architecture and in the *new-AB2* architecture.

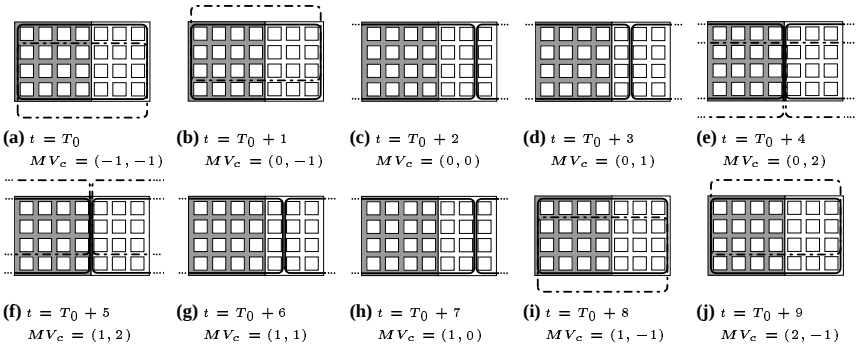


Figure 6: Zig-zag data flow of search pixels on the *New-AB2* architecture ($N = 4, p = 2$).

Vos' architecture, this structure does not require the usage of passive PEs without useful data at any moment. The chart presented in fig. 5 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 $\square$ 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 $k = 2$ ($p = 2N$).

NEW CLASS OF MULTIPLE ARRAY ARCHITECTURES

The ME processor can be further speeded up by computing several similarity measures in parallel through the usage of multiple active blocks (hereafter designated by "cores"): since both the passive and the active PEs perform the same displacement task, a $N \times N$ passive PE array can be replaced by an active PE array. Nevertheless, a processor composed by C processing cores will require C adder tree blocks to compute the similarity values of the candidate MBs being computed 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. 4), 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 $(N-1) \times h$ 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 left column of the first active block, thus obtaining the previously described cylindrical structure (see fig. 3).

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 reduce the active block width to only ℓ PEs ($1 < \ell \leq N$), thus adapting the processor structure to the characteristics of the implementation technology (see fig. 7). It can be proved that in 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 MBs defined in a $(2p+N-1)^2$ pixel search window, the effective value of the number of candidate

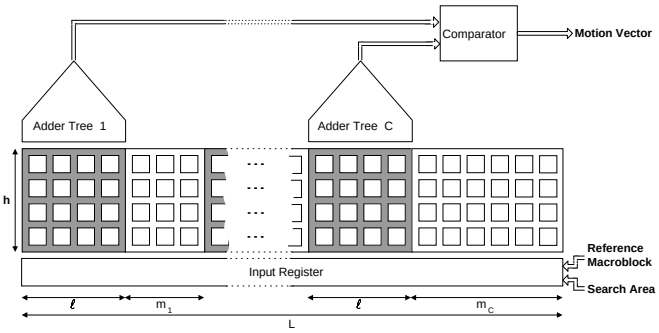


Figure 7: FSBM architecture with C active blocks.

MBs in each line of the search area ($\hat{p}$) is given by $\hat{p} = C \left\lfloor \frac{2p}{C} \right\rfloor$, with a maximum deviation from the initial value given by $0 \leq 2p - \hat{p} \leq C - 1$ [9]. Moreover, the effective search area is composed by $L \times L = [\hat{p} + (N-1)]^2$ pixels and the number of passive PEs between each of the C active blocks is given by:

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

The dimension of the processor array can be further reduced if only h rows of PEs ($0 < h \leq N$) are used. Hence, the number of clock cycles required by each of these structures to process a reference MB is given by (see [9]):

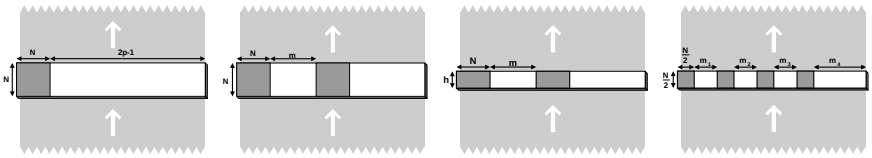
$$T = \left(\left\lceil \frac{N}{h} \right\rceil \times \left\lceil \frac{N}{\ell} \right\rceil \right) \times \hat{p} \times \left\lfloor \frac{\hat{p}}{C} \right\rfloor \quad (2)$$

In fact, considering active blocks with $\ell < N$ or $h < N$ implies the usage of processing schemes where each similarity measure is only obtained after several complete excursions of the processing cores over the corresponding search area, in order to process each of the $\left(\left\lceil \frac{N}{h} \right\rceil \times \left\lceil \frac{N}{\ell} \right\rceil\right)$ fractions of the reference MB. During these excursions, search area data is displaced in both directions in the processing array. Since the processing of each excursion requires $\left\lfloor \frac{\hat{p}}{C} \right\rfloor$ clock cycles, one can conclude that $\left(\left\lceil \frac{N}{h} \right\rceil \times \left\lceil \frac{N}{\ell} \right\rceil\right) \times \left\lfloor \frac{\hat{p}}{C} \right\rfloor$ clock cycles are needed to process each line of the search area. Considering a total of $\hat{p}$ lines of candidate MBs, one can easily obtain eq. 2.

Typical structures

By adjusting the small set of implementation parameters (h, ℓ and C), processing structures with distinct performance and hardware requirements are obtained. The corresponding number of clock cycles required to process one reference MB can be obtained by substituting these values in eq. 2.

The type I structure shown in fig. 8(a) makes use of a single active block ($C = 1$) and corresponds to the single array structure previously described. The



(a) Type I structure **(b)** Type II structure **(c)** Type III structure **(d)** Type IV structure

Figure 8: Single and multi-processor structures

corresponding number of clock cycles required to process one reference MB is $T_I = \hat{p}^2 = (2p)^2$.

Contrasting with type I structure, the type II structure shown in fig. 8(b) makes use of two active blocks ($C = 2$), thus increasing the amount of required hardware by a factor of 2. Therefore, the number of clock cycles required to process one reference MB is halved: $T_{II} = \hat{p}^2/2 = \frac{1}{2}(2p)^2$.

In the type III structure shown in fig. 8(c), the amount of required hardware has been reduced by using only $h = N/2$ rows of PEs. Two active blocks ($C = 2$) are used in this structure, leading to the same number of clock cycles as type I structure: $T_{III} = \hat{p}^2$. This fact can be easily understood if one realizes that although the number of PEs used by this processor array is only half of the required by type I 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 characteristic can provide significant advantages when the target implementation has limited hardware resources available (such as FPGA devices).

The type IV structure, illustrated in fig. 8(d), is characterized by using the same hardware resources than type III structure, but with $C = 4$ and $h = \ell = \frac{N}{2}$. The required number of clock cycles is $T_{IV} = \hat{p}^2$.

In the previous analysis, the extra clock cycles required between the processing of consecutive reference MBs to transfer or remove unused or already processed search data from the array was not taken into account. These extra clock cycles can be avoided if a transparent transfer mechanism based on the pre-fetch layer described in [9] is used. In table 1 it is summarized the number of clock cycles and the number of registers required to process a reference MB by each of the described new structures using this transparent transfer mode, and by the architecture proposed by Vos. In a typical implementation, with $h = \ell = 16$ and $C = 1$ ($N = p = 16$),

Structure	Processing time	Number of registers required
New-AB2 I	$(2p)^2$	$6N^2 + 3N(2p-1)$
New-AB2 II	$\frac{1}{2}(2p)^2$	$9N^2 + 3N(2p-1)$
New-AB2 III	$(2p)^2$	$\frac{13}{2}N^2 + \frac{3}{2}N(2p-1)$
New-AB2 IV	$(2p)^2$	$\frac{21}{2}N^2 + \frac{3}{2}N(2p-1)$
Vos [5]	$(2p)^2 + N(N+2p-1)$	$6N^2 + 2N(2p-1)$

Table 1: Required time and number of registers to process one reference MB.

the pre-fetch layer approach represents an increase in the number of registers of only 19.6%. This increase of the hardware resources can be easily accepted if the achieved speed up (1.734) is taken into account.

MOTION ESTIMATION PROCESSORS

Efficient ME processors based on the previously described class can only be obtained if a careful design of the processing elements as well as the several peripheral blocks is carried out. Moreover, a flexible and parameterizable description of the processor in the synthesis language must be also performed in order to achieve the desired configurable characteristics.

Processor element

The internal structure of the active PE is shown in fig. 9. A PE is composed by four main blocks: the search area transfer circuit (A), the reference MB load circuit (B), the absolute difference arithmetic unit (C) and the accumulator circuit (D). Since passive PEs are only responsible for the displacement of search area pixels, their internal structure is similar to the search area transfer circuit of the active PEs (block A). The search pixel used in the computation of the similarity measure is selected from the set of four pixels supplied by the three PEs located below, on the left and on the right of the considered PE and by a register from the transparent layer (S2 input), and is transferred to an internal standing-data register. In contrast, the reference pixel is only transferred from one of the $\left(\lceil \frac{N}{h} \rceil \times \lceil \frac{N}{l} \rceil\right)$ running-data registers to the standing-data register when the processing of a new reference MB fraction begins. The SAD similarity measure is computed in the absolute difference arithmetic unit and the obtained result is added with the partial sum obtained from the lower level neighbor PE (see fig. 4). These partial sums are accumulated in columns, giving rise to N partial results of the similarity function at the outputs of the N active PEs located in the upper margin of the array. These results are subsequently accumulated by a $log_2 N$ level adder-tree, to obtain the similarity value of the candidate MB under consideration. This value is then fed to the comparator, which retains the similarity measure and the displacement vector corresponding to the best matching candidate MB.

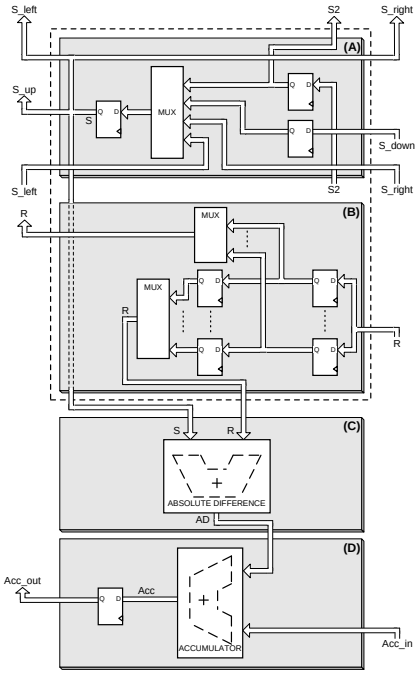


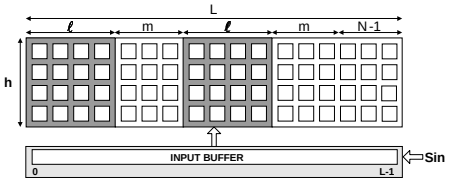
Figure 9: Active PE internal structure.

Input buffers

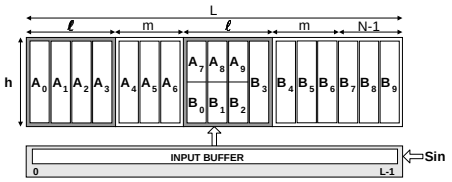
Reference and search area pixels are loaded into the processor array through two serial-in-parallel-out (SIPO) input buffers, whose function is to read the pixels of a given line serially and transfer them in parallel to the array.

In particular, the search area input buffer reads fractions of the previous image, composed by the set of L pixels of each line of the search area, and transfers them in parallel to the array as soon as all the $\left(\left\lceil\frac{N}{h}\right\rceil \times \left\lceil\frac{N}{\ell}\right\rceil\right)$ fractions of the reference MB have been processed. Its implementation is carried out by means of L cascaded registers with their outputs connected to the processor array (see fig. 10(a)).

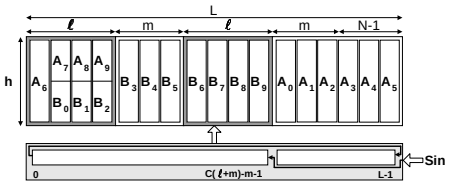
However, with the introduction of the new processing scheme based on the processor cylindrical structure, some measures have to be taken in order to provide the correct data to each PE. Fig. 10(a) shows an example of a processor array with $C = 2$, $N = 4$ and $p = 7$. To simplify the explanation, the processor array was represented schematically in fig. 10(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 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 array is the same as in the input image. Consequently, it is only necessary to transfer them in parallel to the array.



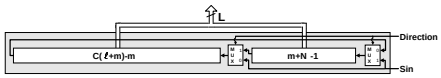
(a) Considered implementation.



(b) Correct alignment of the pixels in the input buffer with those in the array.



(c) Misalignment of the pixels in the input buffer with those in the processing array.



(d) Input buffer and alignment circuit of search area pixels.

Figure 10: 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 distribution of the pixels in the processing array. In fact, their alignment can only be guaranteed if the connection of the L registers is done as it is illustrated in fig. 10(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. 10(d) it is presented the search area input buffer and its alignment circuit.

This circuit is characterized by a variable connection topology of the two smaller input registers that depends on the direction of the flow of the 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 positions of the processing array.

Processor description

A configuration software tool has been developed to determine the set of possible configurations of the proposed class of architectures which fulfill the requisites of the video coder. This configuration tool, whose graphical interface is illustrated in fig. 11, receives the following set of parameters: the image resolution and the frame rate; the dimension of the reference MB (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 pre-fetch layer [9] is to be used. These parameters are used to compute the required number of PEs to be implemented in each line and column of the active blocks (h, ℓ) and the number of processing cores that will compute in parallel the SAD similarity measure (C). 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 technology (input file). As soon as the selection of the desired configuration has been carried out, this tool outputs the complete description of the FSBM processor using IEEE-VHDL description language.

EXPERIMENTAL RESULTS

The desired efficiency levels of the proposed class of processors can only be achieved if its implementation is carried out with a careful study of the blocks that affect more significantly its overall performance, trying to minimize their 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 were considered and carried out in circuits C and D of fig. 12, in order to obtain the shortest critical path as possible.

One of the considered approaches consisted in the usage of a growing bit-vector operand strategy: while the absolute difference arithmetic unit performs its operation on 8-bit input values, the accumulation circuit operates on 12-bit operands, required to accommodate the sum of all partial values of a given column. Following the same strategy, 16-bit operands were considered in the adder-tree structures, thus providing the required dynamic range. To further minimize the processing time, both the active PE and the adder tree circuits were implemented using carry-save adder topologies. However, an incrementer circuit has been used in the computation of the most significant bits, in order to avoid unnecessary wide bit-vector operands, resulting from the 1-bit growth between each stage of the carry-save accumulation circuits. Moreover, Sklansky prefix-adder structures [10] have been extensively used

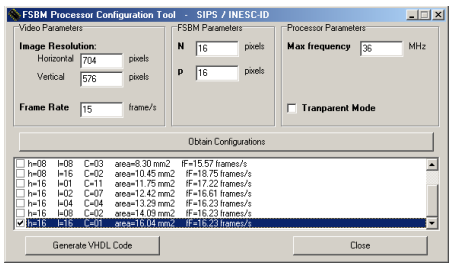


Figure 11: FSBM processor configuration tool.

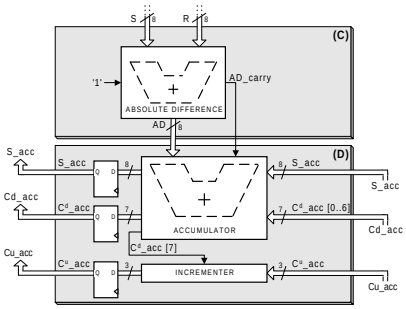


Figure 12: Internal structure of blocks C and D of the active processor element.

due to their significant advantages in what concerns the minimization of the critical path.

To achieve the required characteristics in what concerns the processor configurability, the description of the several blocks of the proposed class of processors was carried out using a fully parameterizable IEEE-VHDL description, making extensive use of '`generic`' configuration inputs and using a fully structural description of the main processing structures.

A FSBM chip based on the proposed class of architectures was designed with Synopsys and Cadence design tools, using a $0.25\mu m$ CMOS technology process. The implemented configuration is composed 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 used by ITU-T H.26x and ISO MPEG standards. A complete list of the processor characteristics is presented in table 2. The chip is able to deliver 28.6GOPs at $36.5MHz$ 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 performs one addition and the comparator unit computes one comparison. As an example, with these results one can conclude that this implemented processor is able to perform motion estimation on 4CIF (704×576) image sequences at a rate up to 16 frames per second.

Algorithm	FSBM
Block size*	16×16
Search range*	$-15, +16$

*- configurable

(a) Algorithm parameters

Technology	CMOS - $0.25\mu m$ UMC
Supply voltage	$2.5V$ / $3.3V$
Active PE area	$32,184.1\mu m^2$
Die area	$16.07mm^2$
Max. frequency	$36.5MHz$
Max. resolution	4CIF/16 fps

(b) Chip characteristics

Table 2: Implemented processor characteristics.

CONCLUSIONS

A class of fully parameterizable multiple array architectures for ME in video sequences has been proposed in this paper. This class makes use of the FSBM algorithm and is based on a new efficient AB2 single array architecture with minimum latency, maximum throughput and full utilization of the hardware resources. This class provides the possibility to process in parallel several candidate MBs, through the usage of multiple and independent processing cores that compute the similarity measures. A software configuration tool has been 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. Processors based on this class of architectures are able to perform ME according to the existing ITU-T H.26x and ISO MPEG standards with configurable search ranges and video quality tradeoffs.

References

[1] Vasudev Bhaskaran and Konstantinos Konstantinides, *Image and Video Compression Standards: Algorithms and Architectures*, Kluwer Academic Publishers, second edition, June 1997.

[2] Y. Ooi, *Digital Signal Processing for Multimedia Systems*, chapter 12 - Motion estimation system design, pp. 299–327, 1999.

[3] T. Komarek and P. Pirsch, "Array architectures for block matching algorithms," *IEEE Transactions on Circuits and Systems*, volume 36, no. 10, pp. 1301–1308, October 1989.

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

[5] L. Vos and M. Stegherr, "Parameterizable VLSI architectures for the full-search block-matching algorithm," *IEEE Transactions on Circuits and Systems*, volume 36, no. 10, pp. 1309–1316, October 1989.

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

[7] Keshab K. Parhi and Takao Nishitani (eds.), *Digital Signal Processing for Multimedia Systems*, Marcel Dekker, Inc., 1999.

[8] C. H. Hsieh and T. P. Lin, "VLSI architecture for block matching motion estimation algorithm," *IEEE Transactions on Circuits and Systems for Video Technology*, volume 2, no. 2, pp. 169–175, June 1992.

[9] Nuno Roma and Leonel Sousa, "Implementation aspects of MESA processor," Technical Report RT/001/2001, INESC-ID – Lisboa, January 2001.

[10] J. Sklansky, "Conditional sum addition logic," *IRE Transactions on Electronic Computers*, volume EC-9, no. 6, pp. 226–231, June 1960.