Automatic Parallelization of Full 2-D Block Matching for Real Time Motion Compensation and Mapping into Special Purpose Architectures

Nectarios Koziris, George Papakonstantinou and Panayotis Tsanakas

National Technical University of Athens
Dept. of Electrical and Computer Engineering
Computer Science Division
Zografou Campus, Zografou 15773, Greece

e-mail: papakon@cs.ece.ntua.gr
tel: +301 - 7722494
fax: +301 - 7722496

Abstract—The most important issue in video encoding is motion compensation within a frame sequence. Block matching techniques are used by various algorithms [5], [6] to estimate the motion into succesive frames. Full search 2-D block matching is the most widely used algorithm for video encoding in all standards (H261, MPEG1-2 , HTDV etc). It provides with the best SNR, since it uses exhaustive search to find the best, in terms of MAD or MSE matching candidate block. Its main disadvantage is that it requires a lot of computations which should be performed for every candidate frame. Consequently, real time application of the 2-DFS algorithm for video compression, requires the use of parallel architectures, to achieve large amount of computations. This paper presents the application of automatic loop parallelization techniques to provide with scalable or fixed systolic arrays, with optimal performance, in terms of total computation time. We transform the BMA algorithm into an equivalent form, in order to apply the automatic parallelization method. Starting from the dependence index space, we map the matching algorithm to 2-D or 3-D systolic arrays and we propose an alternative mapping for fixed size architectures.

I. BLOCK MATCHING MOTION ESTIMATION CONCEPTS:

Block-matching motion estimation/compensation is used to remove the temporal redundancy within a frame sequence, thus resulting into significant bit-rate reductions for digital image encoding methods. Nevertheless it requires a large amount of computation and heavy memory bandwidth.

In the block-matching techniques, the candidate frame for encoding is partitioned into quadratic blocks of nxn pixels. These blocks, resulting from the segmentation of the current and previous frames, are called *current* and *previous* block, respectively. For each current block, the best matching *previous* block is sought within a search area surrounding the previous block:

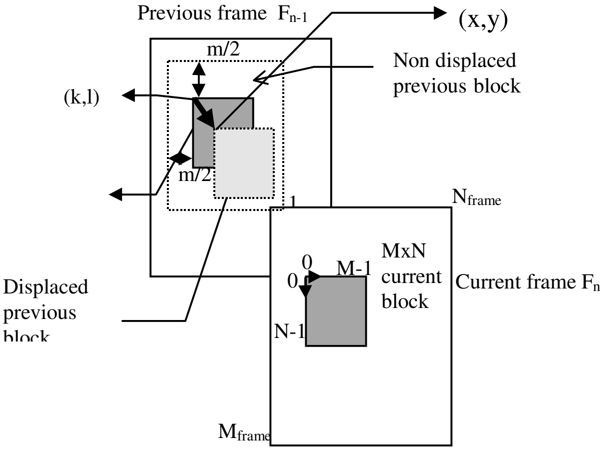


Figure 1. Block Matching Concepts

The search is limited to a maximum displacement m/2 in both directions around the position of the reference block (search area). The candidate blocks are $(m+1)^2$.

The 2DFS algorithm, is the following:

$$\mathbf{MAD_{(k,l)}(x,y) = S(x,y)} = \sum_{i=0}^{M-1}\sum_{j=0}^{N-1}\left|F_n(k+i,l+j) - F_{n-1}(k+x+i,l+y+j)\ \right|$$

$$\mathbf{v(k,l) = (x,y)\ |\ \min_{-m/2 < x,y < m/2} MAD_{(k,l)}(x,y)}$$

In order to apply the automatic methodology of [7], we transform the above summation into a four dimensional loop, where the dependencies between the different loop iterations/ statements are better shown. This transformation reveals all possible parallelism and, therefore facilitates the optimal parallelization of the BMA algorithm:

```
For x = 0 ... m
      For y = 0... m
           For i = 0 to M-1
             For j = 0 to N-1
                MAD(k,l)(x,y,i,j)=MAD(k,l)(x,y,i,j-1)+ |Fn(k+i,l+j)-Fn-1(k+x+i,l+y+j)|
             end j
            MAD(k,l)(x,y,i,N-1) = MAD(k,l)(x,y,i,N-1)+ MAD(k,l)(x,y,i-1,N-1)
         end i
         MAD(k,l)(x,y,M-1,N-1)= MAD(k,l)(x,y,M-1,N-1)+ MAD(k,l)(x,y-1,M-1,N-1)
      end y
      MAD(k,l)(x,m,M-1,N-1)= MAD(k,l)(x,y,M-1,N-1)+ MAD(k,l)(x-1,m,M-1,N-1)
end x
```

It can be easily noticed that the computation of MAD's for all candidate blocks in the previous frame F_{n-1}, is a four dimensional nested loop over indices y,x,i,j. In order to exploit all the inherent parallelisation of this nested for loop, we will apply methods for automatic parallelization of loops used in parallel compilers [3], [11], [13]. These techniques will subsequently map the parallel algorithm into a systolic structure according to the methodology and the respective tool presented in [7] .

II. AUTOMATIC TECHNIQUES FOR MAPPING THE FULL SEARCH BLOCK MATCHING ALGORITHM ONTO SYSTOLIC ARRAYS :

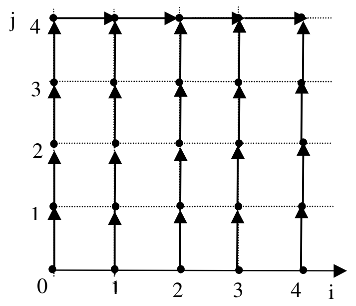
Systematic methods for parallelization of for-loops were proposed by Moldovan [11], Shang [13] and Andronikos et al. in [1]. These methods are based onto the decomposition of the algorithm into basic modules containing single assignment statements. Since the nested loop is assumed to have depth n, every indexed variable in any assignment statement is n-dimensional. Since n indices are used, the algoritmh is defined over an n-D index space. Every point inside this integer index space corresponds to an instance $(i_1,...,i_n)$ of the nested loop. This simple algorithmic model can be graphically displayed by an n-D index space graph of computation nodes and data dependence arcs. From such a representation of the computation space, various architectures can be derived.

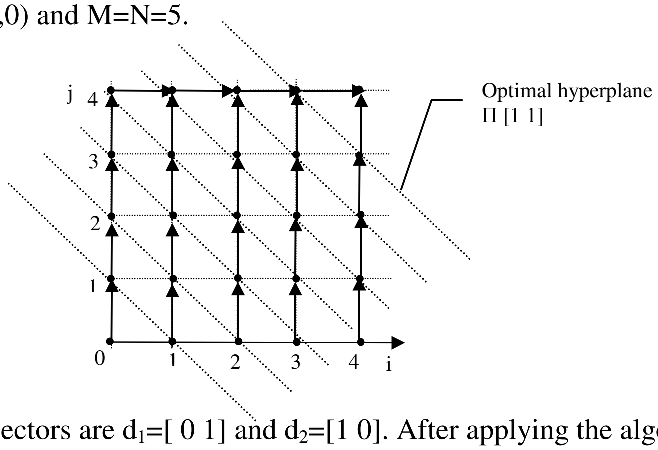
a) n-D Index Spaces:

The simplest and most straightforward implementation is the assignment of every computation node of the n-D index space to a distinct PE (Processing Element). This leads to optimal time execution of the nested loop, but requires too many processors, leading to poor processor utilization. It is equivalent to using a distinct processor for every loop instance. A more efficient design with higher processor utilization can be achieved if each PE executes the operations of multiple processor nodes.

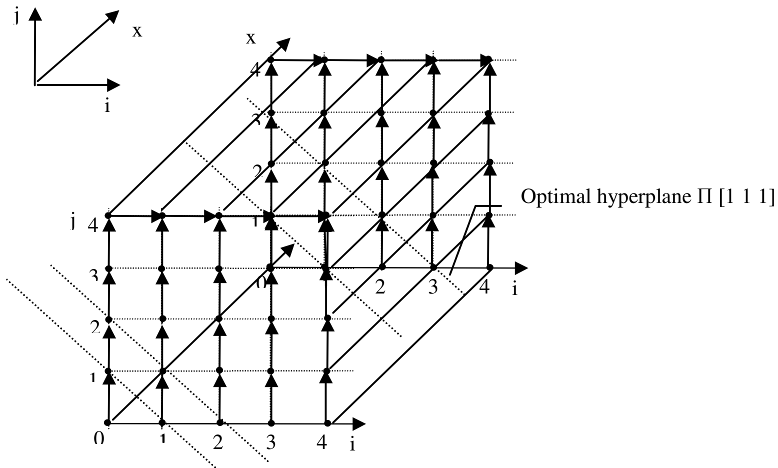
The BMA (Block-Matching Algorithm) is defined over a four-dimensional index space due to its four indexes x,y,i,j. We are proposing various decomposition techniques according to the total number of cells and the dimension of the target systolic array. It is obvious that the BMA nested loop can be decomposed into two parts, which are defined over two-dimensional index spaces. The first one is spawn by the indexes i and j and consists of the addition of the sum MAD(x,y) . In the rest, which is defined over x and y, the minimum search and the selection of the displacement vector is performed.

*note:*we consider for simplicity block (0,0) and M=N=5.





In the 2-D index space the dependence vectors are d_1=[0 1] and d_2=[1 0]. After applying the algorithm proposed in [2], the optimal time hyperplane is Π [1 1].



In the 3-D index space the dependence vectors are d_1=[0 1 0] , d_2=[1 0 0] and d_3=[0 0 1]. After applying the algorithm proposed in [2], the optimal time hyperplane is Π [1 1 1].

b)Resulting Architectures:

As the authors propose in [1], the resulting architectures are n-1 dimensional, if the initial nested loop has n dimensions. In section IIa, two representative index spaces were presented, along with the corresponding array architectures, by applying the method of [11], [7] are 1-D and 2-D. Note that the proposed block architecures are optimal in terms of total parallel execution time, since the automatic technique chooses the time hyperplane Π for each index space which minimizes $t_{parallel}=\min\left\lceil \frac{\max \Pi j_1 - \min \Pi j_2 + 1}{disp\pi} \right\rceil$ where j_1,j_2 are points of the index space

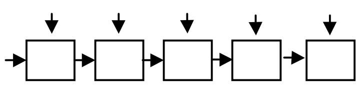


Fig 2. The proposed Systolic Array implementing i,j loop

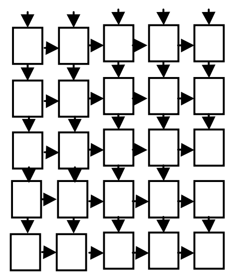


Fig 3. The proposed Systolic Array implementing i,j,y loop

In Fig. 2 , the architecture derived from the inner i,j nested loops is shown. If we want to serially perform the outer x,y loops we can simply add another cell, which succesively performs the x,y additions, and a minimum cell, as follows in fig 4:

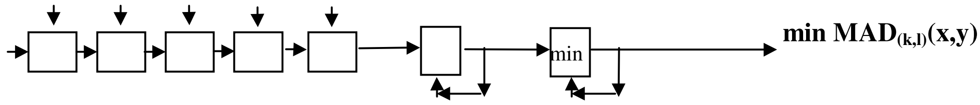


Fig 4. The proposed Systolic Array implementing i,j loop in parallel and x,y outer loops sequentially

In Fig. 3, the architecture is derived from the inner i,j,y nested loops is shown. If we want to serially perform the outer x loop we can simply add another cell, which succesively performs the x additions, and respective minimum cells, as follows in fig 5:

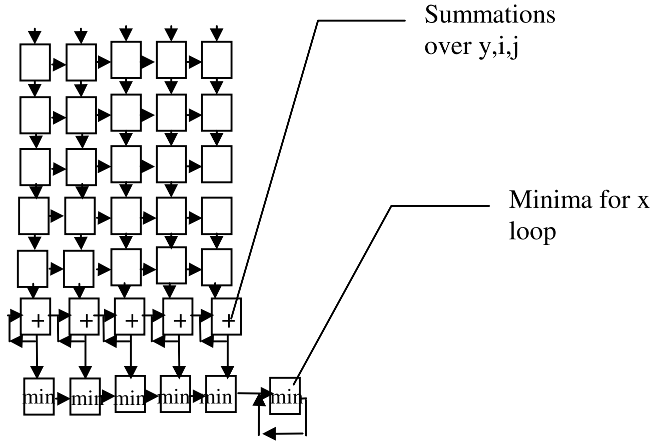


Fig 5. The proposed Systolic Array implementing i,j,y loop in parallel and x outer loop sequentially

Finally, a 3-D array which can perform all loops in parallel, as quickly as possible, can be derived by extending the above techniques. Mathematic perfomance evaluation proves the efficiency and real time response of the proposed architectures.

III. CONCLUSIONS

In this paper we have proposed systematic methods for mapping the BMA algorithm into special purpose parallel architectures. We transformed the BMA into an equivalent form and we applied automatic loop parallelization techniques. The resulting architectures are optimal both in time and space requirements. This method gives us the possibility to produce automatically special purpose hardware for real time motion compensation applications.

REFERENCES:

[1] T. Andronikos, N. Koziris, Z. Tsiatsoulis, G. Papakonstantinou and P. Tsanakas, "Lower Time and Processor Bounds for Efficient Mapping of Uniform Dependence Algorithms into Systolic Arrays, " to appear in the *Journal of Parallel Algorithms and Applications, vol. 12, 1-2,1996*

[2] T. Andronikos, N. Koziris, Z. Tsiatsoulis, G. Papakonstantinou and P. Tsanakas, "Lower Time Processor Bounds for Uniform Dependence Algorithms," First ECPD International Conference on Advanced Robotics and Intelligent Automation, Jan. 1995.

[3] A. Darte and Y. Robert, "Constructive Methods for Scheduling Uniform Loop Nests", *IEEE Trans. Parallel Distrib. Syst*., vol. 5, no. 8, pp. 814-822, Aug. 1994.

[4] M.J. Chen, L.G, Chen, T.Z Chiueh, "One - Dimensional Full Search Motion Estimation Algorithm For Video Coding", IEEE Trans. on Circuits and Systems for Video Technology, vol. 4, no. 5, pp. 504-509, Oct. 1994.

[5] H. Gharavi and M. Mills, "Blockmatching Motion Estimation Algorithms - New Results", IEEE Trans. on Circuits and Systems, vol 37, no. 5, pp. 649-651, May 1990.

[6] H.M. Jong, L.G. Chen, T.D Chiueh, "Parallel Architectures for 3-Step Hierarchical Search Block-Matching Algorithm", IEEE Trans. on Circuits and Systems for Video Technology, vol. 4, no. 4, pp. 407-415, Aug. 1994.

[7] N. Koziris, G. Papakonstantinou and P. Tsanakas, "Automatic Loop Mapping and Partitioning into Systolic Architectures, " *5th Panhellenic Conference on Informatics*, Athens 1995.

[8] T. Komarek and P. Pirsch, "Array Architecures for Block Matching Algorithms", IEEE Trans. Circuits and Systems, vol. 36, no. 10, pp. 1301-1308, Oct. 1989

[9] P.-Z. Lee and Z.M. Kedem, "Synthesizing Linear Array Algorithms from Nested For Loop Algorithms," *IEEE Trans. Comp*., vol. 37, no. 12, pp. 1578-1598, Dec. 1988.

[10] B. Liu and A. Zaccarin, "New Fast Algorithms for the Estimation of Block Motion Vectors" , IEEE Trans. on Circuits and Systems for Video Technology, vol. 3, no. 2, pp. 148-157, Apr. 1993.

[11] D.I. Moldovan and J.A.B. Fortes, "Partitioning and Mapping Algorithms into Fixed Size Systolic Arrays," *IEEE Trans. Comput*., vol C-35, no 1, pp. 1-11, Jan. 1986.

[12] W. Shang and J.A.B. Fortes, "Time optimal linear schedules for algorithms with uniform dependencies," *IEEE Trans. Comput*., vol. 40, no. 6, pp. 723-742, June 1991.

[13] L. de Vos and M. Stegherr, "Parameterizable VLSI Architectures for the Full - Search Block - Matching Algorithm", IEEE Trans. on Circuits and Systems, vol 36, no. 10, pp. 1309-1316, Oct. 1989.

[14] K.M. Yang, M.T Sun, L. Wu, "A Family of VLSI Designs for the Motion Compensation Block-Matching Algorithm", IEEE Trans. on Circuits and Systems, vol. 36, no. 10, pp. 1317-1325, Oct 1989.