Published in Proceedings of the Picture Coding Symposium (PCS), 91, 1994
which should be used for any reference to this work

BIT-SERIAL PARALLEL PROCESSING VLSI ARCHITECTURE FOR A BLOCK MATCHING MOTION ESTIMATION ALGORITHM

J. Bracamonte, I. Defilippis, M. Ansorge, F. Pellandini
Institut de Microtechnique, Université de Neuchâtel, Rue de Tivoli 28, CH-2003 Neuchâtel, Switzerland

I. INTRODUCTION

Motion estimation is a key element in video signal interframe coding systems. Among the different techniques proposed for the estimation of movement of pixels between two successive frames, full-search block-matching algorithms (FSBMA) are more attractive due to their low mathematical complexity. The FSBMA based on the mean absolute difference (MAD) criterion, is preferred from a hardware implementation point of view, because it does not require any multiplication or division [1]. Furthermore, its matching performance is comparable to the one achieved with other more complex matching criteria.

A bit-serial parallel processing VLSI architecture of the FSBMA based on the MAD matching criterion is proposed in this paper. It is well suited for the estimation of motion in low bit-rate applications as teleconferencing and videophone systems. The complexity of the architecture is minimal and it features a high degree of regularity; reducing in consequence, the time and effort required for designing the layout, which is estimated to take a few man-week units, even if it is fully-customized. The basic building blocks of the architecture are shift registers and bit serial units (adders, subtracters, and one comparator), which favors bit serial data processing and a high pipeline efficiency. In terms of hardware this has two consequences. First, the capacitive load downstream of each node is minimal and thus the different modules can run at high frequencies [2] (several hundreds of MHz). Second, the full pipeline characteristic of the blocks performing both the arithmetical operations and movement of data, allows an efficient use of hardware of nearly 100% during a frame cycle[2]. The architecture is completely independent of the size of the search area associated with full search matching algorithms. By programming the control unit and the address generator we can select the size of the search area. Finally, this architecture is fully compatible with a 2D-DCT chip that has been previously developed, integrated and tested by the authors of this paper. Local memory, conversion registers and control signals can be shared by these two main components of video data compression.

II. THE BLOCK MATCHING ALGORITHM

Fig. 1 shows the principle of FSBMA. A reference block (RB) of size NxN pixels on the current frame is matched with $(2p+1)(2p+1)$ candidate blocks (CB), where p is the maximum possible estimated displacement of each RB. Each CB of NxN pixels itself, is chosen from a window or search area (SA) of $(2p+N)(2p+N)$ pixels, on the previous (predicted) frame. The MAD matching criterion is given by

$$MAD(i,j) = \frac{1}{N\text{x}N} \sum_{x=1}^{N} \sum_{y=1}^{N} \left| A(x,y) - B(x+i, y+j) \right|$$

for $-p \le i, j \le p$. $A(x,y)$ represents the intensity of a pixel with coordinates (x,y) on a RB, and $B(x+i, y+j)$ corresponds to the intensity of a pixel with coordinates $(x+i, y+j)$ on the SA.

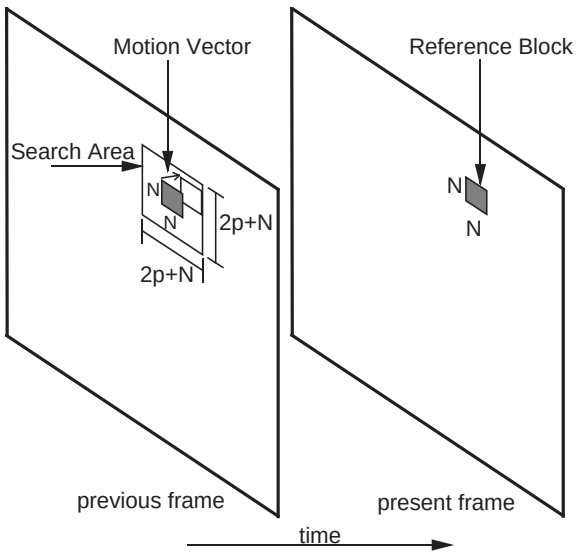

Figure 1 Block Matching Algorithm.

A simple example of the block matching algorithm with parameters $N = 3$ and $p = 3$ is shown in Fig. 2. The search area (SA) of 9x9 pixels is shown in Fig. 2(a), corresponding to the reference block (RB) of 3x3 pixels shown in Fig. 2(c). There are a total of 7x7=49 CBs in the search area to be matched with the RB.

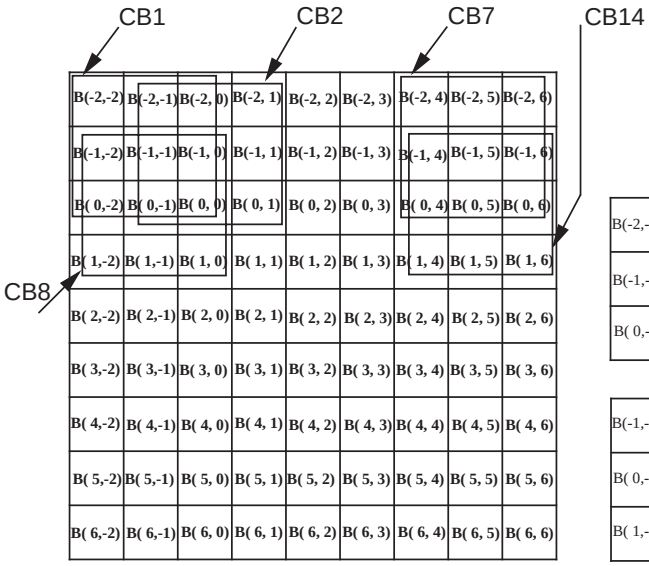

Fig. 2(a) Search Area (SA).

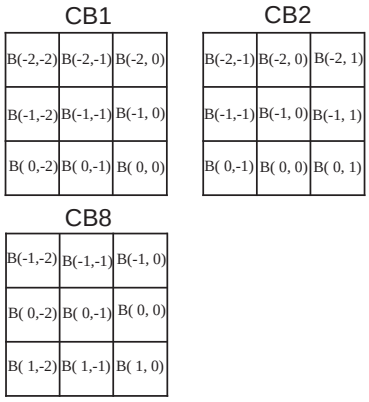

Fig. 2(b) Candidate Blocks CB1, CB2, CB8.

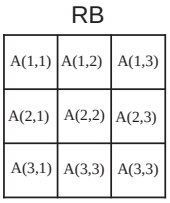

Fig. 2(c) Reference Block (RB)

Figure 2 Example of a Block Matching Algorithm with N = 3 and p = 3.

III. IMPLEMENTATION OF THE ALGORITHM

One of the main concerns regarding the hardware implementation of FSBMA, is input bandwidth. Due to the exhaustive search nature of the algorithm, a huge amount of information must be transferred, to be processed, in a relatively short period of time. Complexity and cost are, as well, of paramount importance. Bandwidth and complexity are treated directly in this paper. Cost is implicitly addressed.

An important optimization can be made regarding the input bandwidth and the pin counts, if we take advantage of the multiple pixels that are common to different CBs. That is, if we use a pixel more than once (which is the case for all the pixels on the SA, except for the four pixels on the corners) we can read it, from the previous frame memory, only once and then pipeline it or store it temporarily in local memory. For example, in Fig. 2 we note that the pixels in the 2nd and 3rd columns of CB1, correspond to the 1st and 2nd columns of CB2 respectively. Thus, after CB1 has been loaded and matched, to form CB2 we have to read only 3 additional pixels (i.e., the 3rd column of CB2: B(-2,-1), B(-1,-1) and B(0,1)) and make two column shifts from the previous CB1. In general, no additional memory is required, when a CB uses data from the neighbor, horizontally adjacent to its left. Because simultaneous to the broadcasting of bits to the module that implements the sum of absolute differences (ADs) we can pipeline them, internally in the local memory unit, to the new columns that these bits are going to form in the next CB to be processed.

In the preceding description, we reduce the input bandwidth by taking advantage of common data between two horizontal adjacent CBs. Further reduction can be obtained by also taking advantage of data common to two vertically adjacent CBs. For example, to form CB8 (and in general all the leftmost CBs, i.e., CB15, CB22,...,CB43) we only need to read three additional pixels (B(1,-2), B(1,-1) and B(1,0)) to form the 3rd row. Finally, to form CB9 (CB16, CB23, etc.) and all the CBs horizontally to its right, we only need to read one additional pixel, i.e., B(1,1). The latter, will be the most common case, meaning that to form a new CB, the majority of time, we will only need to read one additional pixel.

To conclude this section, one final elaboration is made regarding the pseudo carriage return we have chosen to scan the SA (e.g., in Fig. 2, we process CB8 right after CB7, instead of CB14 which is adjacent to CB7). By following this procedure, we simplify the number of scanning movements to 2, i.e., scan 7 times to the right and carriage return. In hardware, this implies a reduction in the complexity of data movement, the complexity of the control signals, the number of wires to run, and the number of inputs to the multiplexers of the registers containing the current CB (the C registers in Fig. 4). The alternative option (matching CB14 after CB7) implies 4 different scanning movements, i.e., scan 7 times to the right, at rightmost position scan once to the bottom, scan 7 times to the left and at leftmost position scan once to the bottom.

IV. ARCHITECTURE

The proposed architecture is shown in Fig. 3. At each clock pulse, a new pixel is read by the Conversion Register Bank (CRB) from the predicted frame memory. The CRB module is a pipelined converter: parallel input (1 pixel=k bits per clock pulse), serial output (k bits per clock pulse, corresponding to k different pixels). The operation of this module is explained in [3] and in [4]. A CRB module with k=16, has been successfully integrated, by the authors of this paper, in a 2D-DCT chip which corresponds to an updated version of the DCT circuit presented in [3].

The pipelined Candidate Block Memory module (CBM) is shown in Fig. 4. The C registers contain the NxN pixels of the current CB being matched. The wires identified with an arrow pointing to the left margin of the page, perform the pipelined column shifting. The R registers provide both the row shifting (for the 1st and 2nd columns) and the delay of data common to two leftmost CBs, e.g., CB1 and CB8; CB8 and CB15, and so on. The two blocks of six shift registers at the bottom of the figure, perform the row shifting of the 3rd column and contribute to the respective delay of 7 matching cycles, between data used for example by the 3rd columns of CB1 and CB8, CB2 and CB9, CB3 and CB10 and so on.

The block diagram of the Reference Block Memory (RBM) is shown in Fig. 5. Each pixel of the RB is self-looped 49 times, the time required to match the 49 CBs on the SA.

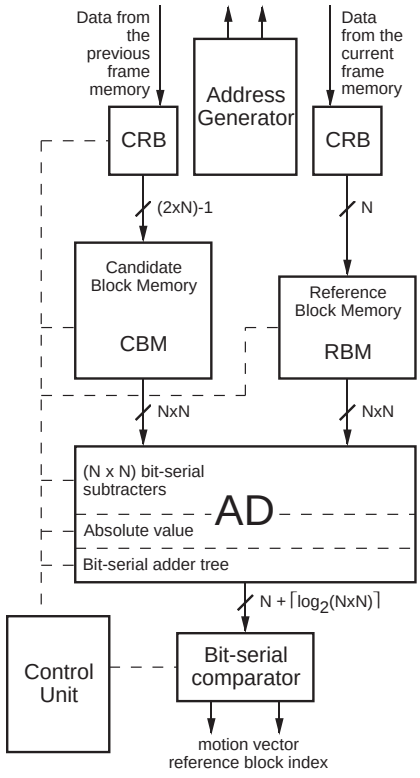

Figure 3 Architecture.

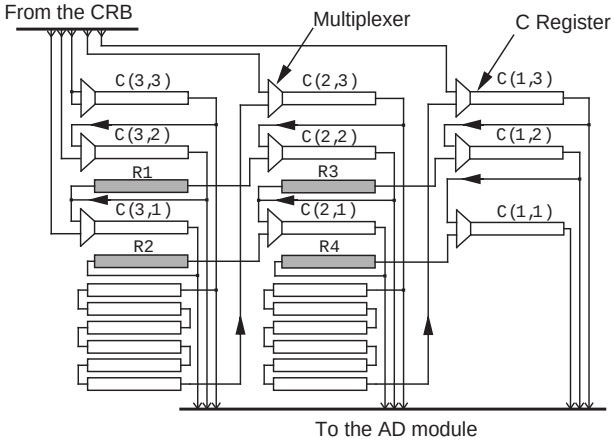

Figure 4 Candidate Block Memory (CBM).

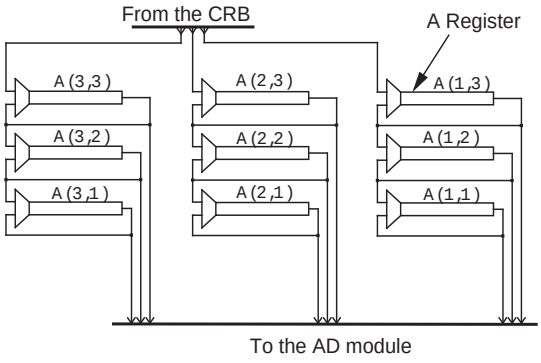

Figure 5 Reference Block Memory (RBM).

The wires providing the self-looping of the A registers, are also used to transfer data from one register to its neighbor one floor lower (e.g., from A(3,3) to A(3,2)). This is made to reduce the pin counts (when the CRBs are off chip, so they can also be used by the 2D-DCT circuit). At the beginning of a new RB, the RBM block can be completely filled by means of only 3 input wires. The same principle is used to fill the CBM module in Fig. 4.

The Absolute Difference (AD) module corresponds to a set of NxN serial subtracters, some random logic to perform the absolute value and an adder tree which outputs a scalar i.e., the sum of all the ADs of the current CB. The wordlength of this binary number, assuming the coding of the pixels as straight PCM with 8 bits/pixel, is given by: $8+\lceil \log_2(N\text{x}N) \rceil$ bits, where $\lceil x \rceil$ denotes the largest integer greater than or equal to x. Thus for $N=8$, each MAD result will be coded with 14 bits. Fourteen bits will be in consequence, for $N=8$, the wordlength of all the registers in the CBM, RBM and all the modules downstream. Every 14 clock pulses, a new match or comparison will be calculated.

V. PERFORMANCE AND FIGURES OF MERIT

Let us take an example of a video telephone application, where each frame is represented in common intermediate format (CIF): 288 lines and 352 pixels/line; a frame frequency of 10Hz; a reference block RB of 8x8 pixels; and a maximum displacement p of 8 pixels. The number of RBs per frame is (288x352)/(8x8)=1584. The MAD of each RB, is obtained after having made $(2p+1)(2p+1)=289$ matches. As has been pointed out before, for N=8, a new match is made every 14 clock pulses. To obtain the motion vector of each RB we need: 289x14= 4,046 clock pulses; plus 8x14=112 pulse cycles to fill the pipeline of the CMB (only once per RB), which gives a total of 4,158 clock pulses. The current frame is thus calculated in: 4,158 x1584= 6,586,272 clock pulses; plus 196 pulses required to fill the pipeline of the CRB (only once per frame). At 10 frames/s, a clock rate of approximately 65 MHz is required.

As described in section I, the basic building blocks of the architecture are shift registers bit serial adders and subtracters. The capacitive load each node has to charge or discharge is minimum, so the architecture is as fast as the technology allows it to be[2]. The layout (1.2µm CMOS CMN12 process from VLSI Technology Inc.) of the shift registers has been simulated using the Quicksim and Spice of the Compass environment, both for pseudo 2-phase [5] and for true single phase [6] clocking strategies. With both simulators we obtained a maximal functional frequency of about 180MHz for the pseudo 2-phase clocking, and about 215MHz using true single phase clocking (in fact even higher functional frequencies were obtained with Spice, which is to be expected, due to the worst-case/pessimistic characteristic of Quicksim). For the bit-serial adders and subtracters, we expect functional frequencies, in the same order of magnitude. In all the above cases, we obtained operating frequencies of about 3 times higher than that required for this video telephone application. The total memory requirement (CBM and RMB) for this application is of 4046 bits. Thus, based on our experience, we estimate that a single small-area chip would satisfy all the computational and memory requirements.

ACKNOWLEDGEMENTS

This work was supported by the Swiss National Found for Scientific Research (FNRS) under Grant 20-33997.92

REFERENCES

[1] H.G.Musmann, P.Pirsch, H.Grallert: "Advances in Picture Coding"; *Proc. IEEE,* Vol. 73, No. 4, pp. 523-548, April 1985.

[2] P. Denyer, D. Renshaw: *"VLSI Signal Processing: A Bit-Serial Approach",* Addison-Wesley, VLSI System Series, 1985.

[3] U. Sjöstrom, I. Defilippis, M. Ansorge, F. Pellandini: "A Discrete Cosine Transform Chip for Real Time Video Applications", *Proc. IEEE Int. Symp. Circuits and Systems, ISCAS'90,* Vol. 2, pp 1620-1623, New Orleans, LA, USA, May 1990.

[4] I. Defilippis, U. Sjöstrom, M. Ansorge, F. Pellandini: "Optimal Architecture and Time Scheduling of a Distributed Arithmetic Based Discrete Cosine Transform Chip", *Proc. European Signal Processing Conference EUSIPCO'90,* Vol. 3, pp.1579-1582, Barcelona, Spain, September 1990.

[5] N. Weste, K. Eshraghian: "*Principles of CMOS VLSI Design: A Systems Perspective",* Addison-Wesley, VLSI System Series, 1985.

[6] J. Yuan, C. Svensson: "High-Speed CMOS Circuit Technique", *IEEE J. of Solid-State Circuits*, Vol. SC-24, No. 1, February 1989, pp. 62-70.