VLSI Architecture for Motion Estimation using the Block-Matching Algorithm

César Sanz[+], Matías J. Garrido[+], Juan M. Meneses[*]

[+]Dpto. de Sistemas Electrónicos y de Control. E.U.I.T. Telecomunicación
[*]Dpto. de Ingeniería Electrónica. E.T.S.I. Telecomunicación
Technical University of Madrid.
{cesar,matias}@sec.upm.es, meneses@die.upm.es

Abstract

In this paper an architecture is described that implements motion estimation in image coding, using a block-matching algorithm and an exhaustive search method. The architecture, EST256, consists of 256 processor elements, deals with a search area of -8/+7 and performs 11 GOPS (subtraction, absolute value determination, accumulation and comparison). It is implemented with ES2 0.7 μm double-metal-layer CMOS technology. This ASIC is cascadable to deal with bigger search areas.

1. Introduction.

Nowadays, Moving Images Coding has a very promising application field: Videoconferencing, Videophoning, Digital video storage, High-Definition Television (HDTV), Digital Television and Multimedia Systems are some of the keywords in this area. In Image Coding Systems, data compression is needed for efficient management of the large amount of information. For example, a colour image with resolution of 1000 by 1000 pels (*p*icture *el*ements) will occupy 3 megabytes of storage in an uncompressed form. Data compression is especially useful for the transmission of such data through transmission channels. For instance, bit-rate ranges from 10 Mb/s for broadcast-quality video to more than 100 Mb/s for HDTV signals.

To facilitate industrial application of this technology some standards have been proposed: the Joint Photographic Experts Group (JPEG) standard for still picture compression [1]; the Consultative Committee on International Telephony and Telegraphy (CCITT) Recommendation H.261 (px64) for videoconferencing [2] and the Moving Pictures Experts Group (MPEG) [3] for full-motion image compression.

For moving images compression *hybrid coding* is used. This compression method is based on both redundancies in the data and the nonlinearities of human vision; and combines transform coding with predictive coding. It exploits the usually high spatial correlation of the images and the low sensitivity of the human eye to high spatial frequencies. For this, Transformation to the frequency-domain is applied, using the Discrete Cosine Transform (DCT) [4]; then high-frequency DCT coefficients are coded with fewer bits than the low ones. This technique achieves compression ratios from 10:1 to 50:1. On the other hand, hybrid coding also exploits the temporal redundancy of the image sequence and reduces information using prediction techniques based on motion estimation. This scheme increases the compression ratio to 200:1.

Motion Estimation is the most demanding part in the coding algorithm. For example, in an image coding/decoding system according to Rec. H.261 [5], the computational power required is approximately 1.2 GOPS [6]; and around 50% of this effort is devoted to the motion estimation. At the decoder, motion estimation is not necessary, therefore lower computational power is required.

Making a brief historical review, up to the late eighties, programmable architectures for image processing were proposed. These processors were oriented not only to image coding but to general image processing. They were improved designs of the classical DSP architecture with one or several processors working in parallel to provide bigger computational power as required for these applications. This kind of processors is reviewed in [7].

Due to the excessive size of the previous systems, some specific architectures for image coding were proposed in the early nineties. These dedicated solutions were based on a chip-set of ASICs [8-10]. However, research activity in this direction has decreased because there are two drawbacks: a long design time is required and the systems are not flexible to adaptation to changes in the standards.

Recently, there has been a trend to programmability again in image coders/decoders, but with an architectural conception very far from DSP style [11-12]. These new

ED&TC '96
0-89791-821/96 $5.00

architectures are based on a very optimised RISC core processor (with an instruction set oriented to the application) and aset of specialised processing units (coprocessors) for those tasks that require a higher computational power. The RISC core executes the functions related to the system control including management of the coprocessor activities as well as some simpler pieces of the coding algorithm. One of these coprocessors is dedicated to Motion Estimation.

In section 2 motion estimation is reviewed. The proposed architecture is explained in section 3, and section 4 shows a comparison between this realisation and some commercial chips for motion estimation. Finally, section 5 concludes the results.

2. Motion estimation.

To implement motion estimation in coding image applications, the most popular and widely used method, due to its easy implementation, is the block-matching algorithm (BMA).

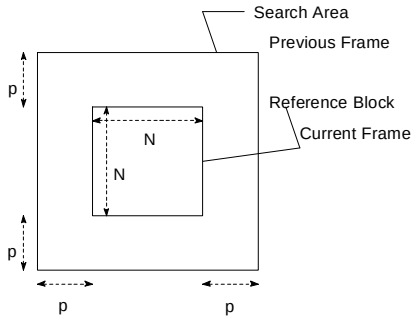

Figure 1.

The BMA divides the image in squared blocks and compares each block in the current frame (reference block) with those within a reduced area of the previous frame (search area) looking for the most similar one, as shown in figure 1. This matching procedure is made by determining the optimum of the selected cost function. We can describe the problem at three levels:

At the first level, the *searching algorithm* choice is set. The most accurate one is the exhaustive (also called full-search), consisting of the evaluation of the cost function in all and every possible locations of the reference block within the search area. The major drawback of this method is its computational cost which strongly increases as the search area does. In fact, $(2p+1)^2$ evaluations of the cost function are required, where p is the maximum displacement of the reference block within the search area in the four spatial directions.

In order to decrease the number of evaluations of the cost function, several special search strategies have been proposed. In this way reductions by, at least one order of magnitude, are possible. However, they can lead to a local rather than a global optimum. In [13] a comparative summary of the most relevant of these strategies can be found.

The choice of the *cost function* is at the second level of the problem description. Two functions are the most commonly used: the Mean Square Error (MSE) and the Minimum Mean Absolute Error (MAE). In these kind of applications, simulations show that MSE and MAE perform very similarly [14]. For this reason MAE is the most widely adopted because of its simpler computational complexity (hardware multiplier is not required). This function is presented in (1), where N is the block size, x are the pels in the reference block; and x_A are those within the search area.

$$D(i,j) = \frac{1}{N^2}\sum_{m=1}^{N}\sum_{n=1}^{N} |\, x(m,n) - x_A(m+i, n+j)| \quad (1)$$

Finally, in the third level are the *hardware architectures* capable of supporting the two previous levels. The computationalpower that these architectures must provide is extremely high. For example, for 720x576 pels[1] images, at a rate of 25 Hz, a search area of 16 pels in each direction, full-search algorithm and using MAE as the function cost, 34 GOPS are necessary; considering as operations: subtraction, absolute value determination and accumulation [15]. To undertake this intensive computation, massively parallel and intensively pipelined implementations are required. On the other hand, the large amount of data managed, mainly in the search area, demands highly efficient data-flow and memory architecture, so the bandwidth required remains attainable. For the previous example, the peak bit-rate needed is larger than 800 Mb/s.

3. The proposed architecture.

3.1. Architecture description.

The proposed architecture, EST256, is a generalisation of the one described in [16] but with a higher number of processor elements (PE) as well as better performance. There are also some additional new features: management of the image boundaries is included in the device, as well as additional hardware resources for an easier connection of several devices working in parallel to increase the search window size. Some differences in data-flow are also introduced to facilitate this parallel connection without increasing the bandwidth requirements in the previous frame memory. These aspects are described below.

1 Image resolution in MPEG-2 standard.

The implementation of the cost function expressed in (1) can be decomposed as shown in figure 2. In EST256, the **i** and **j** loops are performed in parallel and therefore it is necessary to have $(2p+1)^2$ PEs and the motion vector is computed in N^2 operation cycles.

```
for (i= -p; i ≤ p; i++){
    for (j= -p; j ≤ p; j++){
        for (m= 1; m ≤ N; m++){
            for (n= 1; n ≤ N; n++){
                D(i,j) += | x(m,n) - xA(m+i, n+j) | ;
            }
        }
    }
}
```

Figure 2. Algorithm decomposition.

The block size used is 16x16 pels according to H.261 and MPEG-2 [17] standards and the affordable search area is -8/+7. The number of PEs working concurrently is 256 and each single processor computes the cost function for one of the 256 possible locations of the reference block within the search area. The array outputs the motion vector corresponding to each reference block, 256 cycles after the last pel of the block has been entered into the array. Figure 3 shows the block diagram of the PE. The *Sub* block is a 8-bits subtractor (luminance pels have 8-bits positive values, from 0 to 255) and *Acc* is a 16-bits accumulator to avoid overflow.

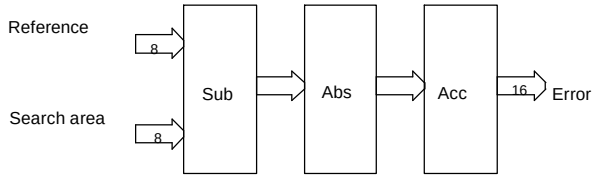

Figure 3. Basic diagram of the processor element.

Figure 4 shows the arrangement of the 256 processor array. To reduce the required bandwidth, EST256 has three 8-bits input ports. After initial latency, the Comparator block inputs one error computation in each cycle and compares it with the previous minimum, storing the lowest. The Boundaries block disables the Comparator when its input value is not valid, this condition arises for some locations of the blocks located on the top, bottom, left and right boundaries of the image. The architecture provides the minimum error value, the coordinates of the motion vector for this position and the error value for the (0,0) motion vector (without movement).

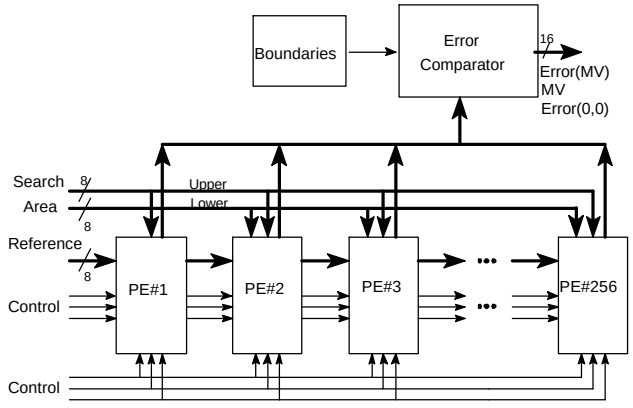

Figure 4. EST256 architecture.

3.2. Data flow.

The ordering of pels in the image is by horizontal bands, each 16-pels high, as shown in figure 5. The pels within each band are read by columns, from top to bottom, simultaneously from the three input ports available in the motion estimator. In Table 1, the sequence of operations in the PEs and the data flow is shown: the pels within the search area are used by all processors in the same cycle, while pels from the reference block are delayed one cycle per PE.

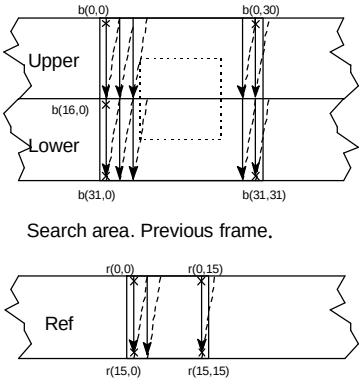

Figure 5. Pels organisation.

The PE#1 and all of them on 1+16n positions ($0 \le n \le 15$) in the array use pels only from Upper band and compute the error in the locations whose upper left corner has (0,i) coordinates, with $0 \le i \le 15$. However, the rest of PEs use data from both Upper and Lower bands and its architecture slightly differs from figure 3 including a multiplexer for appropriate band selection.

When the initial latency is elapsed (512 cycles), the circuit

Table 1

T	Ref	Upper	Lower	PE#1	PE#2	PE#3
0	r(0,0)	b(0,0)		r(0,0)-b(0,0)		
1	r(1,0)	b(1,0)		r(1,0)-b(1,0)	r(0,0)-b(1,0)	
2	r(2,0)	b(2,0)		r(2,0)-b(2,0)	r(1,0)-b(2,0)	r(0,0)-b(2,0)
3	r(3,0)	b(3,0)		r(3,0)-b(3,0)	r(2,0)-b(3,0)	r(1,0)-b(3,0)
4	r(4,0)	b(4,0)		r(4,0)-b(4,0)	r(3,0)-b(4,0)	r(2,0)-b(4,0)
...	...	...		...	...	...
15	r(15,0)	b(15,0)		r(15,0)-b(15,0)	r(14,0)-b(15,0)	r(13,0)-b(15,0)
16	r(0,1)	b(0,1)	b(16,0)	r(0,1)-b(0,1)	r(15,0)-b(16,0)	r(14,0)-b(16,0)
17	r(1,1)	b(1,1)	b(17,0)	r(1,1)-b(1,1)	r(0,1)-b(1,1)	r(15,0)-b(17,0)
...	...	...	...	...	...	...

provides new results every 256 cycles (this is the time required for inputting a new reference block into the array). This is possible because the error function computation for block *I* is concurrent with the computation for block *I+1*, (the next on the right or the first from the left of the following slice or image frame) so pels within the search area that are coincident to both *I* and *I+1* blocks, are reused.

3.3. Design Methodology.

The first step was the design of a VHDL low-level model of the processor element and the array, as well as the ancillary elements (FIFOs and multiplexers) to connect the system appropriately to the frame memory. After functional simulations had been validated and before the system was physically implemented, we observed that using automatic synthesis tools from VHDL descriptions did not provide a good solution because the high number of PEs would require the optimisation of their area. In this way, full-custom design methodologies become a better approach but the design time required is too long. Therefore we consider a semi-custom solution helped by the use of data-path compiler tools, as an intermediate scheme that provides reasonable area and speed features.

In Table 2, the features of the ASIC we have developed are shown.

Table 2

Technology	ES2 dml CMOS 0,7 μm
Chip size	10,6 mm x 12,6 mm
Data-path block size (4 PEs)	2.617 μm x 424 μm
Clock frequency	20 MHz
# equiv. transistors	604.566
# signal pads	67

Figure 6. Layout

Figure 6 shows the layout of the ASIC. The 64 dotted rectangular blocks are modules generated by the ES2 data-path compiler. The 256-to-1 16 bits wide multiplexer necessary to connect PEs outputs to Comparator input is distributed inside data-path blocks, as shown in figure 7. Moreover, each four data-path blocks row multiplexes its error outputs in the same way. This solution reduces complexity with some speed penalty. The Boundaries block, the Comparatorand the final 16-to-1 multiplexer are implemented using standard cells.

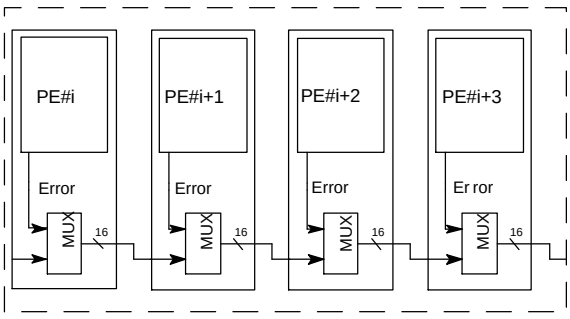

Figure 7. Multiplexing logic in data-path block.

4. Comparison with commercial ICs.

In this section, EST256 is compared to STI3220 (SGS-Thomson) [18] and L64720 (LSI Logic) [19]. Both of them can also implement full-search BMA with blocks of 16x16 pels and a -8/+7 search area, using MAE as the cost function. In Table 3 some differences are shown.

Table 3

Feature	L64720	STI3220	EST256
Frames per second	12	44	49
Frame mem acc. time	-	98 ns	98 ns
Boundaries control	external	external	integrated
Input ports (8 bits)	2+1	3+1	2+1

The number of frames per second has been calculated at maximum clock frequency considering 720x576 pel images.

The required access time to the previous frame memory has been calculated in better-case, considering only one access per clock cycle, therefore FIFO delay-lines must be used: two FIFOs of 11.264 bytes each, for STI3220 which has three input ports, and only one for EST256 (see figure 8, for a block diagram of system connection). This time has not been considered in L64720 because this device can not afford the 25 frames/s required.

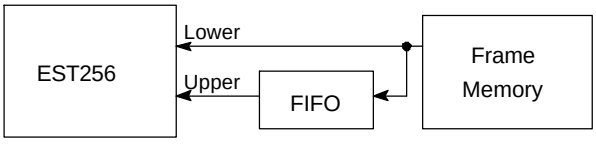

Figure 8. Previous frame memory connection diagram.

In STI3220 and L64720 the boundaries management must be controlled by external logic. However, our architecture includes the boundaries control that can be adapted to different frame size.

Finally, as the minimum search area compliant with MPEG-2 standard is -16/+15, it is necessary to use several devices, working in parallel, to keep real-time video requirements. With the three devices that we are comparing, it is possible to build systems connecting four chips to deal with this search area, but in EST256 some specific facilities has been included to simplify this task: the architecture outputs the Reference block with a 256-cycles delay by means of a dedicated output port, moreover, the output results (the minimum error, the motion vector and the error in the origin) can be randomly accessed any time during 256 cycles. In figure 9 a system block diagram is shown. As it can be seen only two external FIFOs are needed and the frame memory bandwidth remains the same.

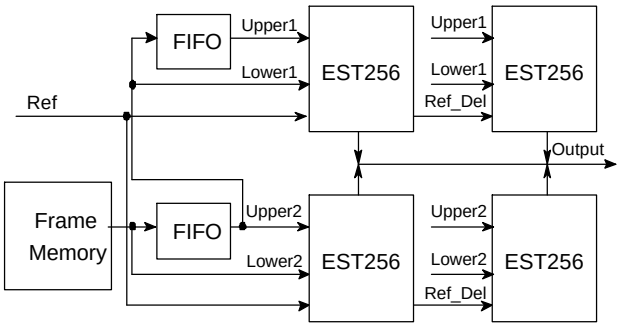

Figure 9. System block diagram for -16/+15 search area.

5. Conclusions.

In this paper we describe a specific architecture that implements motion estimation in image coding, using a full search block-matching algorithm. The proposed architecture performs 11 GOPS for the operations: subtraction, absolute value determination, accumulation and comparison. The allowable pel-rate in the ASIC is faster than the one required in the MPEG-2 standard and cascaded chips can be used to deal with bigger search areas. For example, using four devices, a -16/+15 search area (minimum search area in MPEG-2) can be afforded.

6. Acknowledgements.

This work is being supported by a grant TIC95-0791 from the *Comisión Interministerial de Ciencia y Tecnología* (CICYT) of the Spanish Government.

7. Bibliography.

[1] G.K. Wallace. "The JPEG Still Picture Compression Standard". *Communications of the ACM*. Vol. 34, nº4. April 1991.

[2] M. Liu. "Overview of the px64 Kbits/s Video Coding Standard". *Communications of the ACM*. Vol. 34,nº 4, April 1991.

[3] D. Le Gall. "MPEG: A video Compression Standard for Multimedia Applications". *Communications of the ACM*. Vol 34, nº 4, April 1991.

[4] K.R. Rao & P. Yip. "Discrete Cosine Transform. Algorithms, Advantages, Applications". Academic Press Inc. 1990.

[5] CCITT. Recommendation H.261. Dec. 1990. "Line transmission on non-telephone signals. Video codec for audiovisual services at p x 64 kbit/s".

[6] K. Guttag et al. "A single-Chip Multiprocessor For Multimedia: The MVP". *IEEE Computer Graphics and Applications*. Nov, 1992. pp 53-64.

[7] Konstantinides, V. Bhaskaran. "Monolithic Architectures for Image Processing and Compression" *IEEE Computer Graphics & Applications*. Nov 1992.

[8] H. Fujiwara et al. " An All-ASIC Implementation of Low Bit-Rate Video Decoder". *IEEE Trans. on Circuits and Systems*. Jun 1992.

[9] P.A. Ruetz et al. "A High-Performance Full-Motion Compression Chip Set". *IEEE Trans. on Circuits and Systems*. Jun, 1992.

[10] I. Tamitani et al. "An Encoder/Decoder Chip Set for the MPEG Video Standard". IEEE ICASSP-92, CS Press, Los Alamitos, Calif., 1992.

[11] D. Bursky. "Improved DSP ICs Eye New Horizons". *Electronics Design*. Nov 11, 1993.

[12] P. Pirsch, N. Demassieux, W. Gehrke. "VLSI Architectures for Video Compression-A Survey". *Proceedings of the IEEE*. Vol. 83 No 2. Feb 1995.

[13] M. Ghanbari. "The Cross-Search Algorithm for Motion Estimation". *IEEE Trans. on Communications*. Vol. 38 No 7. Jul 1990.

[14] R. Srinivasan, K.R. Rao. "Predictive coding based on efficient motion estimation". *IEEE Trans. on Communications*. Vol COM-33, Aug 1985.

[15] T. Komarek, P. Pirsch. "Array Architectures for Block Matching Algorithms". *IEEE Trans. on Circuits and Systems*. Vol 36, No 10, Oct 1989.

[16] 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, Oct 1989.

[17] ISO/IEC JTC1/SC29. Recommendation H.262. Nov. 1993. "Generic Coding of Moving Pictures and Associated Audio".

[18] "STI3220 Motion Estimation Processor". Advance data. July 1992. SGS-Thomson.

[19] "L64720 Video Motion Estimation Processor (MEP)". May 1994. LSI Logic.