Programmable modified systolic array for fast one- and two-dimensional convolutions

L. Borghesi, E. Giuliano, and G. Musso

Elettronica San Giorgio Elsag S.p.A., Genova, Italy

F. Cabiati and P. Ottonello

Dipartimento di Fisica, Università di Genova, Genova, Italy

Received December 17, 1985; accepted May 14, 1986

The computational regularity of convolution can be exploited for efficient design of high-throughput architecture able considerably to decrease the calculation times with respect to those typical of full-software solutions. We describe a fast coprocessor able to perform one- and two-dimensional convolutions, the latter case being applicable provided that the convolving bidimensional filter is separable. The arithmetic unit is based on a semisystolic architecture whose high modularity feature is also useful for an efficient very-large-scale-integration custom implementation. The main novelty here consists in some architectural features permitting convolutions with filters having dimensions greater then the number of cells in the semisystolic array either by paralleling a suitable number of identical arrays or by iterative calculations with a single array. Details concerning the arithmetic unit are given together with a general description of the coprocessor. The overall performances of the actual prototype are also presented.

1. INTRODUCTION

Image-based techniques currently represent a significant field of research and application in a broad range of technical and scientific domains. Computer vision,[1,2] robot vision, remote sensing,[3] acoustical imaging, and tomography[4] are only a few of the more relevant fields in which data acquired from the external world are topologically organized as images.

For these fields of application two main classes of computing processes may be defined. The first is the case of image-formation processes (acoustical imaging, synthetic aperture radar for remote sensing, etc.) in which sparse data are processed to organize them in a conventional image form. The second is the case in which data acquired already represent an image of the external world and the computing target consists of extracting, describing, and understanding the information contained in the image. In both cases a large number of image transformation techniques are based on convolution algorithms,[1] which, in spite of their extremely useful mathematical properties, are too slow to permit real applications on general-purpose computers.

In particular, in the computer and robot vision fields a considerable amount of image processing is required in early stages for filtering spatial frequencies to enhance image contrast or to detect pictorial features necessary for further, more abstract, computation levels.[5–8]

In these fields the required accuracy in transforming images needs large-sized convolution masks, and, at the same time, operating conditions impose a short computation time for each acquired video frame. Many methods and algorithms useful for speeding up the calculations of convolutions have been proposed[9–12] and implemented both by standard components and by very-large-scale integration (VLSI).[13,14] A simple and effective solution for both of the above constraints is achieved by limiting the class of bidimensional masks to the set of separable ones.[15] This limitation does not appear too rigorous in practical, usual cases. In fact, also from a theoretical standpoint, separability is a weak limitation, because any square mask may be decomposed in an additive sequence of separable masks (i.e., by using Taylor expansion).

The purpose of this paper is to show the implementation of a fast one-dimensional–two-dimensional (1D–2D) convolver that takes advantage of separability. In fact, a 2D convolution is carried out by two consecutive 1D convolutions along the two orthogonal directions.

The whole system is based on special-purpose hardware built around an arithmetic unit (A.U.) that has been designed starting from an eight-cell linear semisystolic array.

This architectural solution, besides being particularly suited for VLSI implementation because of its simple and regular data and control flow, presents, as pointed out by Ersoy,[16] some advantages over the well-known systolic approach.[17,18]

Moreover, since in a semisystolic array the input data do not propagate from cell to cell [see Fig. 1(a)], this structure lends itself well to the proposed improvement, which allows a single array to implement filters with a number of coefficients larger than the number of cells in the array. Obviously, for this goal to be attained, suitable control of the flow of data and filter coefficients together with some local storage must also be provided.

Details of the A.U. module are given in Section 3, and in Section 4 more attention is devoted to the complete coprocessor which, connected as a direct-memory-access (DMA) peripheral to a VAX 11/780 computer (Digital Equipment Corporation), reduces the calculation times

0740-3232/86/091561-05$02.00

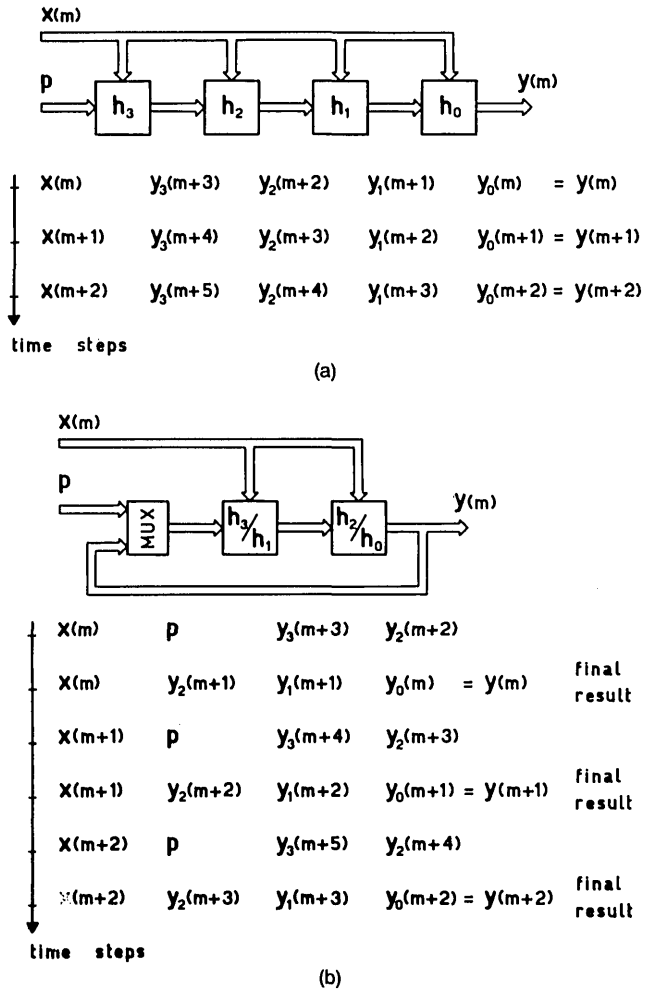

Fig. 1. (a) A four-cell semisystolic array directly implements a four-coefficient convolving filter; one final result is available at every time step. (b) Only two semisystolic cells can implement the same filter because of iterative operations; one final result is available every other time step. In both figures the time-step table makes evident the growth of final results. The input p is a biasing number used for rounding purposes.

about 2 orders of magnitude. In Section 5, finally, some experimental results obtained with a working prototype are presented.

2. BACKGROUND

Image processing and feature extraction generally represent the first stages in an artificial-vision process, in which some level of understanding of a real scene is obtained from information present in one or several bidimensional scene images.

These processing procedures, belonging to the early vision area,[2] usually are intensive in terms of computations to be done on large rates of acquired data. On the other hand, one cannot oversimplify the feature-extraction procedures without losing a great deal of the information present in the image or getting detection accuracies that are too poor, and often not sufficient, to build up a consistent image description.

In order to increase the computation speed, with no loss of information, the important class of separable 2D convolutions, in which the convolving mask is obtained by the product of two unidimensional masks, can be considered. Indeed, each 2D separable convolution can be executed by means of two cascaded 1D convolutions:

$$\begin{aligned} g(x,y) * I(x,y) &= [g_1(x)g_2(y)] * I(x,y) \\ &= g_2(y) * [g_1(x) * I(x,y)] \\ &= g_1(x) * [g_2(y) * I(x,y)], \end{aligned} \tag{1}$$

where $g(x,y) = g_1(x)g_2(y)$ defines a separable 2D convolutive operator and $*$ is a 1D or 2D convolution symbol.

In doing computations as described in Eq. (1) in the discrete domain, the number of multiplications required for computing a transformed pixel is $2n$ instead of n^2, where n is the dimension of the square discrete convolution matrix.

As an example, $2n$ may be considered the "difference of Gaussian filter" described by Marr and Hildreth,[6] which plays a central role in early stages of biological and artificial vision systems, particularly in edge-extraction processes.[19]

The basic idea of the difference of Gaussian filter is closely related to the following formulas[20]:

$$\nabla^2[G(x,y) * I(x,y)] = [\nabla^2 G(x,y)] * I(x,y), \tag{2}$$

where

$$G(x,y) = k \exp\left[-\frac{4(x^2+y^2)}{w^2}\right]$$

represents the filter's impulse response and ∇^2 represents the Laplacian operator.

In the computation model described by Eq. (2), the Gaussian filter performs a smoothing process on the image, and the Laplacian operator, owing to its second-order-derivative content, transforms smoothed brightness edges into zero-crossing points in the transformed image.

Moreover, taking into account the separability properties of the $(\nabla^2 G)$ operator ($\nabla^2 G$ is a sum of bidimensional functions obtained from the product of unidimensional functions in x and y), this model is suitable for fast implementation.

As a possible use of edges it may be worth citing the generalized Hough transform method as applied to 2D shape recognition independent of translation, rotation, and a scaling factor.[21] In this case the recognition process is based on a matching procedure among all edges detected in the image, and shape prototypes are described in terms of their visible boundaries.

Performances of such a recognizer depend strongly on the accuracy in estimating edge location and orientation by a spatial operator, which, as a consequence, must be generally defined by using large convolution masks.

3. ARITHMETIC UNIT DESCRIPTION

The semisystolic cell that is actually realized consists of an 8 × 8 multiplier, a 20-bit adder, a small RAM for the storage of filter coefficients, and some registers holding intermediate results.

As we have already said, the A.U. basically performs 1D digital convolution, giving the output sequence

$$\begin{aligned} y(m) &= \sum_{i=0}^{n-1} h(i)x(m-i) \\ &= h(n-1)x(m-n+1) + \ldots + h(0)x(m), \end{aligned}$$

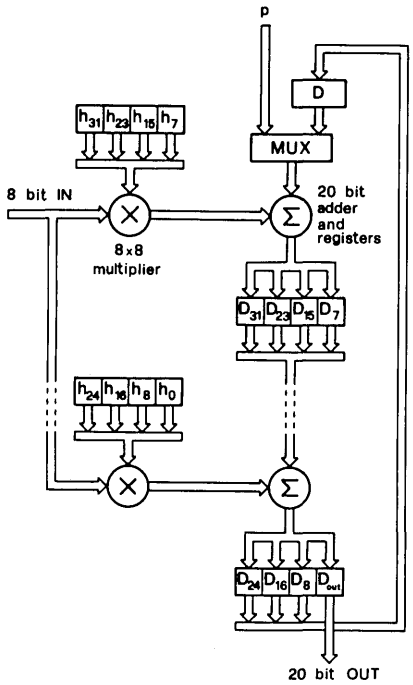

Fig. 2. Schematic diagram of the arithmetic module. Convolutions with a filter size larger than the number of multiplying stages are performed by recirculating partial results inside the module. MUX, 20-bit digital multiplexer.

where, as usual, $h(i)$ are the filter coefficients and $x(m-i)$ are the terms of the input sequence.

By defining

$$y_k(m) = \sum_{i=k}^{n-1} h(i)x(m-i)$$

$$= h(n-1)x(m-n+1) + \ldots + h(k)x(m-k),$$

which is the partial result that is available k time steps before the corresponding final result $y(m)$, and by posing $y_n(m) = 0$, the relation

$$y_k(m) = y_{k+1}(m) + h(k)x(m-k)$$

is obtained.

This recursive formula gives the different outputs of each cell and is employed in Fig. 1(a) to show the growing of each final result along the cells, at different time steps, for the simple case of a four-cell array and a four-coefficient filter.

The flow of data and intermediate results is simple to follow: in fact, at every time step an input term $x(m)$ is broadcast to all the cells: in each cell parallel calculations are performed and a partial result is supplied to the next cell at the end of every cycle (a cycle is defined as the time interval between two consecutive time steps).

The output of the last cell in the array (after an initial delay of n time steps) is a final result, i.e., a term of the output sequence $y(m)$.

In Fig. 1(b) the same result, namely, a convolution of the sequence $x(m)$ with a four-coefficient filter, is obtained by using only two cells in the iterative (two-cycle) mode of operation. The corresponding time-step table makes comprehension of the operating principle straightforward.

In Fig. 2 the actual eight-cell recirculating array is shown, which can implement filters with as many as 32 coefficients through as many as four iterations inside the module.

The option of partial-results iterations is added to a modified finite-impulse-response (FIR) structure (also employed in a TRW device[22] that is easily obtainable from the FIR canonical architecture).

The complete modularity feature of this modified structure is fully preserved in the recirculating A.U. module of Fig. 2, where a product and an addition are the only two consecutive operations always to be performed; in particular, no additional consecutive operations are required for larger filter size.

As an example of the iterative mode of operation, we describe the four-cycle sequence permitting the calculation of a convolution with a 32-coefficient filter. The process starts with a programmed number (p in Fig. 2) set by the approximating circuit (see below) at one input of the first adder and with coefficients $h_{31}, h_{30}, \ldots, h_{24}$ (first column) at one input of the multipliers. Eight terms are calculated during the first cycle. At the next time step the partial result, stored in the D_{24} register, is put back at the input of the first adder; the second cycle begins, and eight other terms, calculated with coefficients stored in the second column, are added.

The process is repeated, recirculating the content of registers D_{16} and D_8; then the fourth cycle ends with a final result (i.e., an addition on 32 terms) available at the register D_{out}.

The input and the output of the adder chain in the module can be used as cascading input and output when several (as many as four) modules are connected in order to reduce the execution time. In the first module additions start again from a programmed number, while in the following ones the input for the first adder is fed by the output of the upstream module.

It is worth noting that the possibility of recirculating partial results significantly improves the A.U. versatility with respect to the common FIR-modified architecture.[22] In fact, for the implementation of a convolver (or in general of a concurrent system for signal processing) one may choose between the two extreme possibilities: N cascaded A.U. modules to obtain the $8N$-coefficient fastest convolver or a single A.U. for an N-times slower convolver. A final remark about the A.U. module concerns the dynamics of the convolutional filter masks. During the loading operation, the control logic can direct the storing of several (as many as 16) different sets of coefficients into the A.U. RAM's. It is then feasible to program the control logic in order to switch from one set to another at any time without interrupting the ongoing convolution process; thus the A.U. can be considered a device with selectable masks.[13]

4. GENERAL DESCRIPTION OF THE COPROCESSOR

A prototype that uses medium- and large-scale integrated circuits and operates in connection with a host computer (see Fig. 3) has been built; it is used mainly for application developments and performance evaluations in its specific field, i.e., early vision processing. This connection permits DMA-controlled downloading of the image (512 × 512 pixels, 256 gray levels) to the coprocessor video memory (V.M.) plus additional data providing both filter coefficients

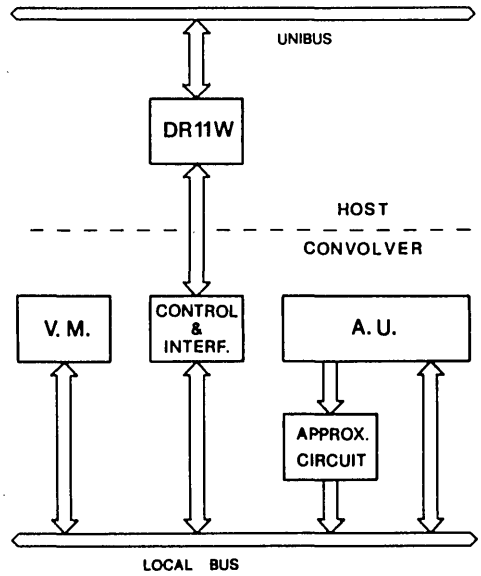

Fig. 3. Block diagram of the convolver. The different block functions are described in the text. DR11W, DMA interface.

(stored in the small RAM's of each cell) and information on the sequence of transforms to be performed.

After the loading operation the convolver goes into a waiting state: the calculation starts on the arrival of a "go" command from the keyboard or by program. Data are picked up in sequence from the V.M. and processed without any approximation through 8 × 8 multipliers and 20-bit adders.

All final results may be stored as 8-bit words in the same memory from which input data flow toward the A.U. or in a different memory module when available. In the first case neither item of information is lost, nor does overlap of image occur because a result is available when the corresponding pixel in the original image is no longer required for the calculation. When a greater precision is required, a second operating mode can be selected, allowing each result to be transferred immediately (as a 16-bit word) to the computer with no local storage.

The overall processing time, once the image is stored in the V.M., is linearly dependent on the size of the filter and on the hardware configuration of the coprocessor. In its minimum configuration, the coprocessor can perform 1D convolution of a 512 × 512 pixel image with an eight-coefficient filter in 35 msec (obviously a complete 2D convolution takes 70 msec). This figure directly derives from the processing power of the basic eight-cell A.U., which reaches 128 million operations per second.

Going back to the blocks of Fig. 3, some comments about the V.M. module and the approximating circuit are useful. When bidimensional images are processed and in general when data must be read or written according to a known sequence, efficient memory arrangements are possible, permitting the exchange of data at a rate higher than a single-memory integrated circuit could perform.[23] The V.M. module must have enough capacity to store a whole image sequence. In the present case a 256-Kbyte memory is formed by four independent 64-Kbyte banks, each equipped with one three-state output register and two input registers, which are alternately enabled to hold each result.

The addressing and control logic common to the four memory banks can manage the exchange of two words between the memory and the A.U. during each 125-nsec-long cycle, in spite of the longer access time (150 nsec) of the high-density static RAM's (HM6264P-15) employed. This result is obtained only because of the complete independence of the four memory banks, which makes concurrent operations possible; in fact, the control logic can cyclically sort the image pixels into the different banks. More precisely, if (i, j) are the coordinates of either an original or a transformed pixel, the value of the pixel will be stored in the first, second, third or fourth bank, respectively, according to whether

$$|i + j|_{\text{mod } 4} = 0, 1, 2, 3.$$

In this way input registers of each bank access the local data bus over four cycles for a writing operation. This also happens for the output register because the control logic again imposes cyclical access to the video information (memory reading): in this case data are read row by row or column by column according to the operator's programmed selection.

Because the use of 20-bit adders and registers inside the A.U. eliminates the possibility of any overflow, it is possible and advisable to exploit the maximum available resolution (20 bits) in making intermediate calculations.

On the other hand, every 1D convolution final result must be reduced to an 8-bit word both for local storage in the 8-bit-deep V.M. and for the second 1D convolution that must be done, again by means of the 8-bit A.U., to complete a 2D convolution. To this end a fully combinatory programmable circuit is provided, permitting the extraction of an 8-bit slice from wherever it is located in the 20-bit string (see Fig. 4). Rounding off of the extracted slice is obtainable by biasing (see input p in Fig. 2) the convolution sum with the 20-bit word 00 . . . 010 . . . 00, where the only 1-valued bit is the one in the jth position starting from the least-significant-bit position.

The number j is set by the user, who, after selecting the filter coefficients, must also decide where the 8-bit slice must begin for the best use of the available output range.

The same approximating circuit also determines the most significant bits that have been discarded by introducing a saturating logic (a number outside the range −128–127 is replaced by the nearest of the two extreme numbers of the interval).

As a final comment it is worth noting that the proposed linear A.U. can also be employed for the calculation of a large class of inner-product functions used in signal and image processing. More specifically, one A.U. module supported by a suitable video memory, which is the core of the convolver in its minimum configuration, can be a building block for a real-time vision system. This is true also because, since the input rate of a memory organized in independent banks

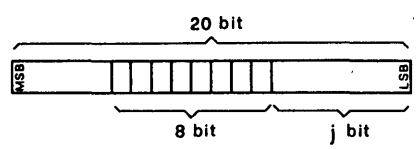

Fig. 4. The 20-bit final result string from which a rounded 8-bit word is extracted. MSB, most significant bit, LSB, least significant bit.

is compatible with most TV standards, direct-image acquisition from a TV camera is possible.

5. FINAL CONSIDERATIONS

The main advantage of this coprocessor is its overall flexibility, which permits the design of optimized hardware configurations suitable for the requirements of real-time systems as well as having a programming flexibility that is particularly useful in development environments.

With the aid of the software driver the activation of the coprocessor is simple, with macroinstructions whose arguments are the filter size, its coefficients, the final result format, and the sequential order of the two 1D convolutions needed in the 2D cases. Within certain constraints it is also possible to perform many sequential 1D convolutions in an iterative mode, obtaining filters with an equivalent size even greater than 32.

The overall precision level provided by the present architecture has been demonstrated to be adequate for most applications in the machine-vision field and is also useful if compared with floating-point software implementations.

REFERENCES

1. E. L. Hall, *Computer Image Processing and Recognition* (Academic, New York, 1979).
2. D. H. Ballard and C. M. Brown, *Computer Vision* (Prentice-Hall, Englewood Cliffs, N.J., 1982).
3. R. M. Haralick, "Automatic remote sensor image processing," in *Digital Picture Analysis*, A. Rosenfeld, ed. (Springer-Verlag, Berlin, 1976).
4. Proc. IEEE **71**, 291–431 (1983), special issue on computerized tomography.
5. P. R. Cohen and E. A. Feigenbaum, *The Handbook of Artificial Intelligence*, Vol. III (Pitman, London, 1983).
6. D. Marr and H. Hildreth, "Theory of edge detection," Artificial Intelligence Laboratory memo no. 518 (Massachusetts Institute of Technology, Cambridge, Mass., 1979).
7. R. C. Gonzales and P. Wintz, *Digital Image Processing* (Addison Wesley, Reading, Mass., 1977).
8. O. J. Braddick and A. C. Sleigh, *Physical and Biological Processing of Images* (Springer-Verlag, Berlin, 1983).
9. R. C. Agarwal and J. W. Cooley, "New algorithms for digital convolution," IEEE Trans. Acoust. Speech Signal Process. **ASSP-25**, 392–410 (1977).
10. Y. C. Jenq, "Digital convolution algorithm for pipelining multiprocessor systems," IEEE Trans. Comput. **C-30**, 966–973 (1981).
11. E. Danielsson, "Serial/parallel convolvers," IEEE Trans. Comput. **C-33**, 652–667 (1984).
12. J. F. Canny, "Finding edges and lines in images," Thesis (Massachusetts Institute of Technology, Cambridge, Mass., 1983).
13. H. T. Kung and R. L. Picard, "One-dimensional systolic array for multidimensional convolution and resampling," in *VLSI for Pattern Recognition and Image Processing*, K.-S. Fu, ed. (Springer-Verlag, Berlin, 1984).
14. G. R. Nudd, "Concurrent systems for image analysis," in *VLSI for Pattern Recognition and Image Processing*, K.-S. Fu, ed. (Springer-Verlag, Berlin, 1984).
15. H. K. Nishihara and N. G. Larson, in *Proceedings of the ARPA Image Understanding Workshop*, L. Baumann, ed. (Science Applications, Bellingham, Wash., 1981), pp. 114–120.
16. O. Ersoy, "Semisystolic array implementation of circular, skew circular and linear convolutions," IEEE Trans. Comput. **C-34**, 190–196 (1985).
17. S. Y. Kung, M. J. Whitehouse, and T. Kailath, *VLSI and Modern Signal Processing* (Prentice-Hall, Englewood Cliffs, N.J., 1985).
18. A. V. Kulkarni and D. W. L. Yen, "Systolic processing and an implementation for signal and image processing," IEEE Trans. Comput. **C-31**, 1000–1009 (1982).
19. W. E. Grimson, "Computational experiments with a feature based stereo algorithm," Artificial Intelligence Laboratory memo no. 762 (Massachusetts Institute of Technology, Cambridge, Mass., 1984).
20. H. K. Nishihara, "Hidden information in early visual processing," memo no. 360-17, Artificial Intelligence Laboratory (Massachusetts Institute of Technology, Cambridge, Mass., 1982).
21. E. Caianiello and G. Musso, *Cybernetic Systems: Recognition, Learning, Self-Organization* (Wiley, London, 1984).
22. TRW VLSI Data Book, 337–349 (TRW, Cleveland, Ohio, 1984).
23. D. C. Van Voorhis and T. H. Morrin, "Memory systems for image processing," IEEE Trans. Comput. **C-27**, 113–125 (1978).