AT&T BELL LABORATORIES TECHNICAL JOURNAL
Vol. 63, No. 2, February 1984
Printed in U.S.A.

# Moment Calculations by Digital Filters

By Z. L. BUDRIKIS* and M. HATAMIAN†

(Manuscript received May 25, 1983)

We present a simple recursive algorithm for computing moments of two-dimensional integer arrays. It uses only additions and can be implemented for high-speed and real-time computation at video rates. We describe Complementary Metal-Oxide Semiconductor (CMOS), Very Large-Scale Integrated (VLSI) implementation of the algorithm in a single chip that can calculate the 16 moments $\mu^{i,j}$ $(i, j = 0, 1, 2, 3)$ (i.e., up to the sixth-order moment) on $512 \times 512$ array of 8-bit integers in real time (at video rate). Such a chip can have potential applications in image processing, graphics, and robotics. The basic building block of the system is a single-pole digital filter that is implemented by recursive addition. The complexities involved in designing the chip, as well as its area, are significantly reduced by taking advantage of the fact that the column samples of the data array can be processed at a much slower rate than the row samples. An estimate of the chip area obtained from the layout design of the individual cells is given.

## I. INTRODUCTION

Moments are familiar from statistics and mechanics, and are finding applications in other areas, among them video processing.[1,2] The mean, or first moment, is a particularly robust estimator of the 'center' of a distribution. Similarly, the centroid is a good single indication of the location of an extended object. We want to use it for pinpointing the position of a light pen.[3]

A possible brake on calculating moments, particularly in real-time applications, may well have been the computational burden that they

* University of Western Australia. † AT&T Bell Laboratories.

appear to entail. It would seem that the calculation calls for exponentiations and multiplications, as well as additions. However, given real-world measurements that are finite in precision, range, and number, and are at regular intervals, the calculation of moments can involve much less effort than at first seems necessary.

One scheme has already been reported[4] in which the multiplying coefficients are generated without exponentiation and in which the multiplications are distributed over the individual bits of the measurements and are therefore accomplished by single AND gates. An even simpler approach is possible, requiring less computation by an order of magnitude, and is reported here.

Our approach comes about from recognizing that Infinite Impulse Response (IIR) digital filters with impulse responses $h(n) = u(n)$, or $nu(n)$ or $n^2u(n)$ or $n^iu(n)$, with $u(n)$ the unit step sequences, will calculate respectively the zeroth-, first-, second-, and $i$th-order moments of their input sequence. It is a straightforward matter to extend this to arrays in the plane and higher dimensionality, since the computations along the different dimensions are separable from each other.

The poles of the required filters can be very easily realized by first-order sections, and these are nothing more than recursions with unity feedback. Thus, if only the poles, and not zeros, of these filters are realized, the computation between successive sample points is just one addition and can be instrumented to take place in real time on picture data from standard television. The basic building block of the system, then, will be a first-order stage, and the resulting filter will have a highly regular and recursive structure that is extremely attractive for Very Large-Scale Integrated (VLSI) implementation.

As will be seen in Section III, the consequence of discarding the zeros of these filters is that the output of, say, the $m$th stage, rather than being the $m$th-order moment itself, is a linear combination of first through $m$th-order moments. This, however, does not create any major difficulty because the coefficients of this linear combination are exactly known, and therefore starting from zeroth-order moment, at each stage the effect of previous moments can be removed by subtraction in a post-processing step.

We describe a design for a custom Integrated Circuit (IC) in Complementary Metal-Oxide Semiconductor (CMOS) technology that can calculate the 16 moments, $\mu^{i,j}$ $(i, j = 0, 1, 2, 3)$, of a $512 \times 512$, 8 bits/pixel image in real time (i.e., at conventional video rate).

## II. THE TASK

Given the array $x(n, m)(n = 0, 1, \cdots N, m = 0, 1, \cdots M)$ of integer values, the moments about $m = 0$, $n = 0$ are

$$\mu_0^{i,j} = \sum_{n=0}^{N} \sum_{m=0}^{M} n^i m^j x(n, m)$$

$$= \sum_{n=0}^{N} n^i \left( \sum_{m=0}^{M} m^j x(n, m) \right). \tag{1}$$

Only the innermost computations, corresponding to along line in television, have to be at speed. The outer computations are at lower rate by a factor of $M$. The choice of $n = 0$, $m = 0$ as the pivot for the moments is arbitrary. Well-known expressions relate the moments about one pivot to those about another. The pivot point about which we chose to do the computations is $n = N$, $m = M$, i.e.,

$$\mu^{i,j} = \sum_{n=0}^{N} (N - n)^i \sum_{m=0}^{M} (M - m)^j x(n, m). \tag{2}$$

## III. THE ALGORITHM

We first consider the one-dimensional case. Consider a sequence $x(n)$ $n = 0, 1, \cdots N$; if this sequence is applied to the input of a digital filter with impulse response $h(n) = n^i u(n)$, then from the well-known convolution theorem the output will be

$$y(n) = \sum_{k=0}^{N} x(k)h(n - k)$$

$$= \sum_{k=0}^{N} x(k)(n - k)^i. \tag{3}$$

If this output is evaluated at $n = N$, then we will have

$$y(N) = \sum_{k=0}^{N} x(k)(N - k)^i, \tag{4}$$

which is the $i$th-order moment of $x(n)$ about the point $n = N$. Figure 1 shows the filter for $i = 0$. This is a single-pole filter with the transfer function $1/z - 1$ in $z$-transform domain. Its implementation is trivial, just an adder with one sample delay and a feedback, or basically an accumulator. For higher-order moments ($i = 1, 2, \cdots$) the filter will have both poles and zeros. In general, for the $i$th-order moment we have $i$ zeros and a pole of order $i + 1$ located at $z = 1$. The first and

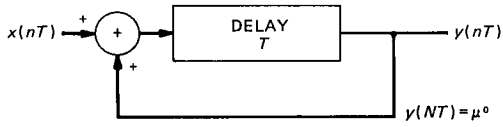


Fig. 1—Single-pole filter for generating zeroth-order moment.

Table I—Impulse response and transfer function of moment filters

| Moment Order, $i$ | Impulse Response, $h(n)$ | Transfer Function, $H(z)$ | All-Pole Transfer Function, $\hat{H}(z)$ | All-Pole Impulse Response, $\hat{h}(n)$ |
|---|---|---|---|---|
| 0 | $u(n)$ | $\frac{1}{z-1}$ | $\frac{1}{z-1}$ | $u(n)$ |
| 1 | $nu(n)$ | $\frac{z}{(z-1)^2}$ | $\frac{1}{(z-1)^2}$ | $(n\text{-}1)u(n\text{-}1)$ |
| 2 | $n^2u(n)$ | $\frac{z(z+1)}{(z-1)^3}$ | $\frac{1}{(z-1)^3}$ | $1/2\,(n-2)^2u(n-2)$ $+ 1/2\,(n-2)u(n-2)$ |
| 3 | $n^3u(n)$ | $\frac{z(z^2+4z+1)}{(z-1)^4}$ | $\frac{1}{(z-1)^4}$ | $1/6\,(n-3)^3u(n-3)$ $+ 1/2\,(n-3)^2u(n-3)$ $+ 1/3\,(n-3)u(n-3)$ |

second columns of Table I, respectively, show the impulse response and transfer function of the moment filters for $i = 0, 1, 2, 3$. If we discard the zeros of the transfer function, then for the $i$th-order moment we obtain the all-pole filter

$$\hat{H}_i(z) = \frac{1}{(z-1)^{i+1}}. \tag{5}$$

This filter can be very easily implemented by cascading $i + 1$ first-order stages similar to the one shown in Fig. 1. Obviously, the output of the filter without zeros will no longer be just the $i$th-order moment of its input signal except for $i = 0$. We examine this next.

Starting from $i = 0$ with the $z$-transform pair,

$$\hat{H}_0(z) = \frac{1}{z-1} \Leftrightarrow \hat{h}_0(n) = u(n),$$

and, using the differentiation property of the $z$ transform, we can progressively derive the impulse responses $\hat{h}_i(n)$ corresponding to $\hat{H}_i(z) = 1/(z-1)^{i+1}$ for all values of $i$. The results for $i = 0, 1, 2, 3$ are shown in the fourth column of Table I. To see the effect of zero elimination, let us consider the case for $i = 3$ as an example. From Table I we have:

$$\hat{h}_3(n) = \frac{1}{6}(n-3)^3u(n-3) + \frac{1}{2}(n-3)^2u(n-3) + \frac{1}{3}(n-3)u(n-3). \tag{6}$$

Convolving this impulse response with the input $x(n)$ and evaluating the output at $n = N + 3$ will result in:

$$y_3(N+3) = \frac{1}{6}\sum_{k=0}^{N} x(k)(N-k)^3 + \frac{1}{2}\sum_{k=0}^{N} x(k)(N-k)^2 + \frac{1}{3}\sum_{k=0}^{N} x(k)(N-k)$$

$$= \frac{1}{6}\mu^3 + \frac{1}{2}\mu^2 + \frac{1}{3}\mu^1, \tag{7}$$

which is a linear combination of first-, second-, and third-order moments. Similarly, for an arbitrary value of $i$, the output of the filter evaluated at $n = N + i$ will be a linear combination of first through $i$th-order moments, i.e.:

$$y_i(N+i) = \sum_{\ell=1}^{i} k_\ell \mu^\ell. \tag{8}$$

Conversely, the moments $\mu^i$ $(i = 0, 1, \cdots)$ can be expressed as a linear combination of the filter outputs $y_i(N+i)$; in matrix notation,

$$\vec{\mu} = \mathbf{C}\vec{Y}, \tag{9}$$

where

$$\vec{\mu} = \begin{bmatrix} \mu^0 \\ \mu^1 \\ \mu^2 \\ \vdots \end{bmatrix} \qquad \vec{Y} = \begin{bmatrix} y_0(N) \\ y_1(N+1) \\ y_2(N+2) \\ \vdots \end{bmatrix}.$$

The coefficient matrix $\mathbf{C}$ can be easily derived from the impulse responses $\hat{h}_i(n)$. For $i = 5$, this matrix is:

$$\mathbf{C} = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & -1 & 2 & 0 & 0 & 0 \\ 0 & 1 & -6 & 6 & 0 & 0 \\ 0 & -1 & 14 & -36 & 24 & 0 \\ 0 & 1 & -30 & 150 & -240 & 120 \end{bmatrix}. \tag{10}$$

Figure 2 shows the complete filter structure for generating zeroth- through third-order moments. The delay introduced on the output of each stage is necessary for synchronizing the output before performing the matrix operation.

So far in this section we have been discussing the moment calculation problem for a one-dimensional sequence $x(n)$; extending the algorithm to two or higher dimensions is a straightforward matter. This is true because the computations along the different directions are separable. Consider an array of integers $x(n, m)$ $(n = 0, 1, \cdots N, m = 0, 1, \cdots M)$, which can, for example, be the digitized samples

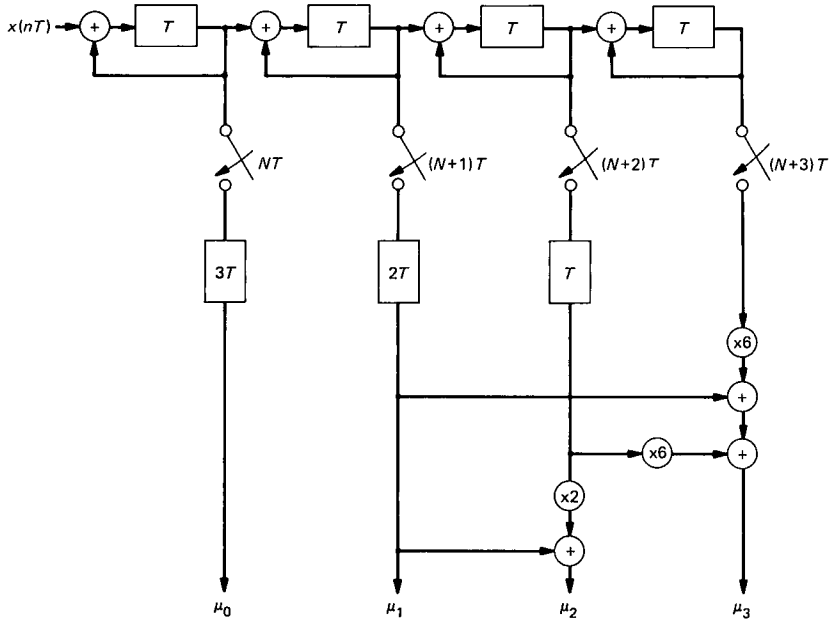


Fig. 2—Moment filter for generating zeroth- through third-order moments.

of a video image. As mentioned in Section II, the moments $\mu^{i,j}$ [i.e., $(i+j)$th order] of this array can be computed from

$$\mu^{i,j} = \sum_{n=0}^{N} (N-n)^i \sum_{m=0}^{M} (M-m)^j x(n, m)$$

$$= \sum_{n=0}^{N} (N-n)^i y_j(n). \tag{11}$$

The inner summation, $y_j(n)$, represents the $j$th-order moment of the $n$th row ("along-line moment") which, as discussed previously, can be computed by filtering the samples on that row with a filter with the impulse response $m^j u(m)$ (row filter) and evaluating the output at $m = M$.

Now $\mu^{i,j}$ is equal to the $i$th-order moment of the sequence $y_j(n)$ and can be computed using a filter with impulse response $n^i u(n)$ (column filter). Figure 3 shows the filter block diagram for generating $\mu^{i,j}$. The input to the column filter implementing eq. (11) is updated every $M$ samples of the input signal, corresponding to one row of the array $x(n, m)$; the row filter is also reset at this time. If the array $x(n, m)$ represents the digitized samples of a video image, then for real-time operation, the sampling rate will be equal to the pixel frequency for the row filter and line frequency for the column filter. As we will see

in Section IV, this fact greatly reduces the complexity of implementation.

The column filters can be realized with single-pole stages in exactly the same manner as was previously discussed in this section. There are two alternatives. One is to use the filter structure shown in Fig. 2, which includes the dematrixing operation, for the column filters as well and make it a building block for implementing the two-dimensional filter. The block diagram of Fig. 4 illustrates this approach. In this case the outputs of the two-dimensional filter will be the true moments $\mu^{i,j}$. The other alternative is to use the original single-pole stage shown in Fig. 1 as the building block for constructing the two-dimensional filter. In this case the outputs of the filter will be a linear combination of the moments; to obtain the actual moments $\mu^{i,j}$, a dematrixing operation has to be done at the end of the process (i.e., after the last row of the input array has passed through the filter). Figure 5 shows the complete two-dimensional structure implementing this latter filter configuration. From the implementation point of view this structure is more attractive than the former realization. (More on this in the following section.) The relationship between the array $x(n, m)$ in Fig. 5 and the input signal is determined by the sampling strategy. For example, if $f(t)$ is a sequentially scanned signal to be filtered, then

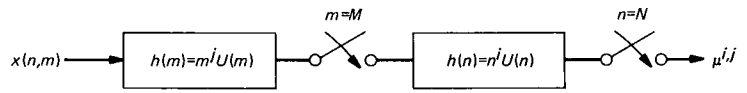


Fig. 3—Block diagram of the filter for generating $\mu^{ij}$.

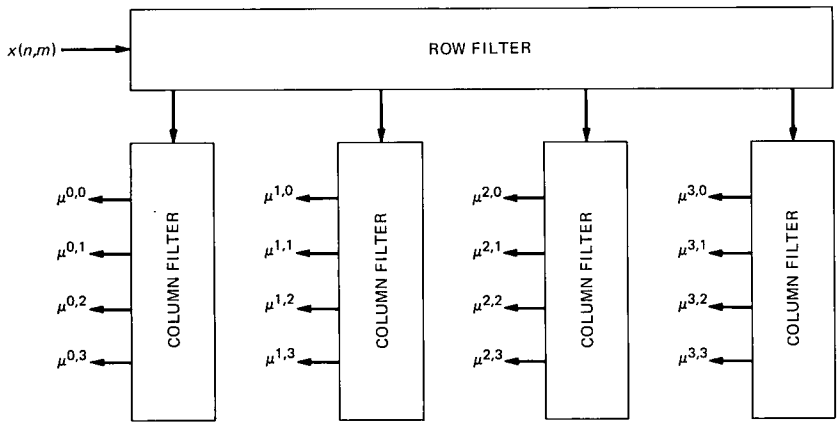


Fig. 4—Moment filter for generating $\mu^{ij}$ $(i, j = 0, 1, 2, 3)$ using the filter shown in Fig. 2 as a building block.

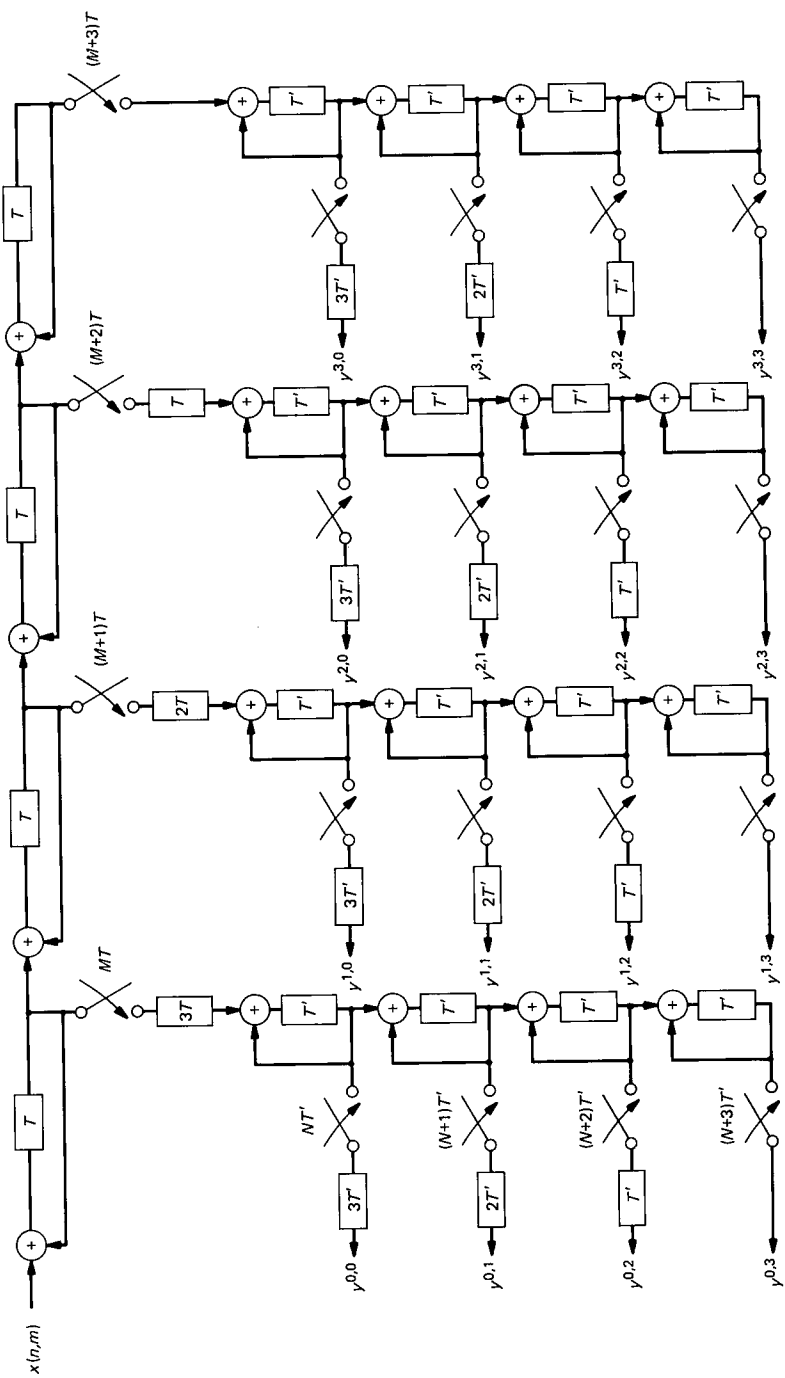


Fig. 5—Moment filter for generating $\mu^{ij}$ $(i, j = 0, 1, 2, 3)$ using the single-pole stage as a building block; $T' = MT$.

$$x(n, m) = f(mT + nT'),$$

where $T$ is the pixel period and $T'$ is the line period.

## IV. VLSI IMPLEMENTATION

In this section we describe VLSI design for implementation of our two-dimensional filtering algorithm, which can calculate the moments $\mu^{i,j}$ ($i = 0, 1, 2, 3, j = 0, 1, 2, 3$) of a 512 × 512, 8 bits/pixel image in real time (i.e., at conventional video rate). We consider a maximum chip size of 6 mm × 6 mm. To meet the speed requirement the picture element (pel) processing time should be less than 125 ns.

We examine the two filter structures shown in Figs. 4 and 5 for possible VLSI implementation. The boxes labeled 'row filter' and 'column filter' in Fig. 4 could be made identical, and each could be realized by one chip. Then the two-dimensional filter would be constructed by putting five of these chips together. This may at first seem a reasonable approach, but in fact, it is not: the speed requirements are set by the calculations that are performed in the 'row filter', while the size of the accumulators is dictated by the 'column filters'. The result would be very large accumulators (up to 64 bits) operating at the high rate of at least 8 MHz. Moreover, that implementation would preclude any possibility of having all calculations done on a single chip.

With the structure of Fig. 5, all mismatches of speed and size can be avoided. What is more, the whole system can be put on a single chip with an area occupancy only slightly greater than needed for a single filter of Fig. 4. The only shortcoming of the system of Fig. 5 compared to that of Fig. 4 is that it is the poles-only realization and requires external dematrixing. That, however, is only minor, for those calculations involve only a few multiplications and additions, and can be readily performed by the host processor that would use the computed moments.

The basic building block of the filter in Fig. 5 is an accumulator functioning as a single-pole digital filter. The maximum word lengths that may appear as outputs and hence determine minimum accumulator sizes increase with order. For 8-bit input data samples in a 512 × 512 array, the accumulations along the row section may progressively reach 17, 25, 33, and 41 bits, and those in the last column section, progressively 48, 55, 62, and 68 bits.

To achieve the required speed in the row section, we have decided to use look-ahead carry adders. The column sections could be implemented with simple ripple carry adders occupying much less area and obviously making it possible to accommodate all circuits on the available chip area. In fact, we do even better than that by using serial

additions. Conversion to serial streams when passing data from the top row into the columns is in any case indicated by wiring considerations.

If given parallel operations in both row and columns, there would be inordinate numbers of wires that would have to pass down vertically, presenting a difficult routing problem. To alleviate this, we decided to use a parallel-to-serial convertor on the output of each stage of the row filter and transfer serially the outputs to the column heads. Note that the along-row and down-to-column transfers take place at different times, the latter only once every line during the horizontal-blanking interval, and that the horizontal connections remain in parallel without any sacrifice in speed. To achieve the dual feed-out, the output registers of the along-row accumulators are designed as parallel in parallel/serial out shift registers.

Once given serial streams into the columns, there is no point in converting back to parallel operation or using ripple carry adders: the accumulations can be done equally well serially, using for each stage a one-bit full adder and a shift register. The block diagram of Fig. 6 illustrates this final design. The upper portion of this figure shows the four stages of the row filter. The lower portion is divided into four sections that correspond to the four column filters and receive their input from the serial output of the shift registers of the row-filter accumulators.

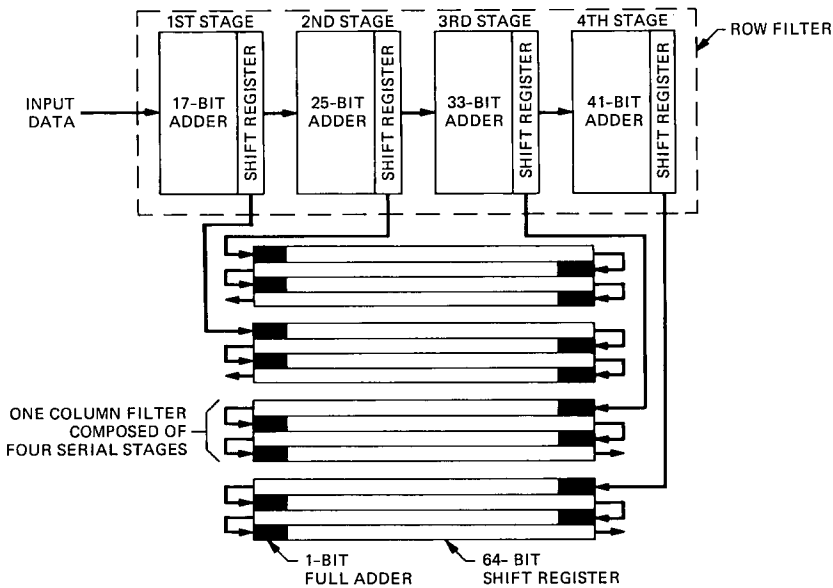


Fig. 6—Block diagram of a VLSI design for single-chip implementation of the two-dimensional moment-generating filter.

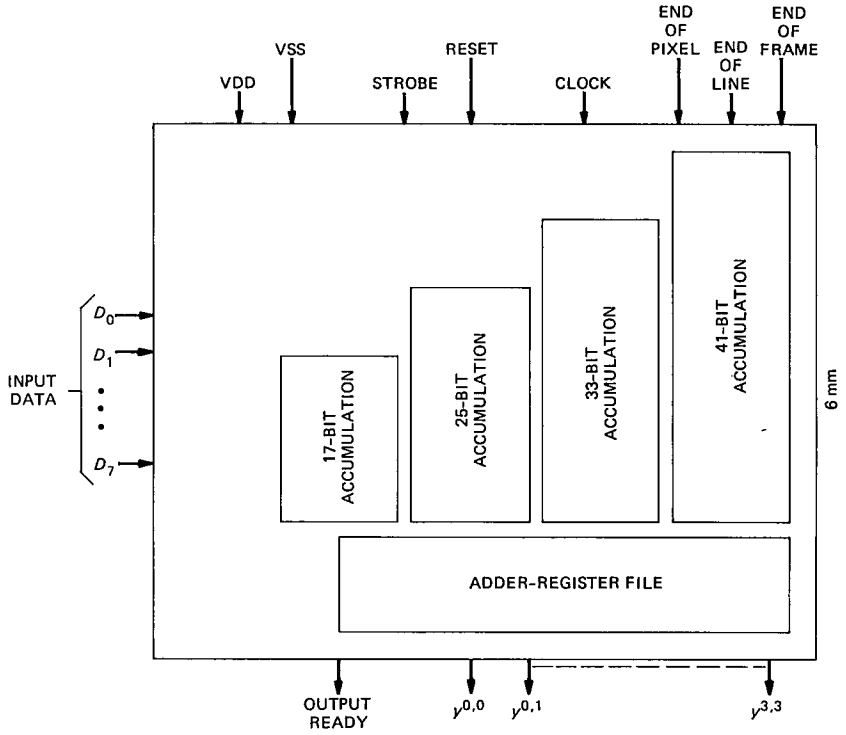


Fig. 7—Floor plan and input/output pins for the design shown in Fig. 6.

Each section consists of four stages stacked on top of each other, each containing a one-bit full adder (the dark area) and a 64-bit shift register composing a 64-bit serial accumulator. The feedback path for the accumulator is not shown in the figure.

The design of the chip is not complete yet, but a conservative estimate of the size has been obtained from the design of the individual cells. The design is for CMOS technology with a two-micrometer design rule. Figure 7 shows the result of the size estimate in a floor plan block diagram drawn in proportion to the 6 mm × 6 mm boundary. Each section of the floor plan is drawn 15-percent larger in both dimensions than the layout size estimated from individual cell designs. As can be seen from Fig. 7, there is still plenty of silicon area left for adding the timing and control section and, if desired, one or two more column stages for generating higher-order moments. The anticipated input/output pins for this chip are also shown in Fig. 7. The chip operation is synchronized with the input image data through three synch pulses: End of Pixel (EOP), End of Line (EOL), and End of Frame (EOF). After the output data ready signal is asserted, the host processor should read the 16 outputs, $y^{0,0}, y^{0,1}, \cdots y^{3,3}$, in serial format.

## V. CONCLUSION

We have described a simple and powerful algorithm for computing moments of a two-dimensional array of integers using digital filters. The algorithm involves only addition operation and is very suitable for real-time implementation. A two-dimensional digital filter with a separable impulse response, $h(n, m) = n^i u(n) m^j u(m)$, can generate the $(i + j)$th order moment of its input. Realization of an all-pole version of such a filter results in a filter with a highly regular structure capable of producing all the moments, from zeroth to $(i + j)$th order, of a two-dimensional input data array. We have shown that the consequence of eliminating the zeros of the filter and using an all-pole version is that the outputs will be a linear combination of the moments rather than the direct moments themselves. But this does not create any difficulty because the constant coefficients of the linear combination are known and a simple dematrixing operation at the end of the process will recover the actual moments. The basic building block of our moment filter is a single-pole digital filter that can be easily implemented by an accumulator. We have proposed a VLSI design for single chip implementation of this moment calculation algorithm. The preliminary results obtained from the layout design of the individual cells for 2-micrometer CMOS technology indicate that on a single 6 mm × 6 mm chip, we can certainly implement a filter for generating the 16 moments $\mu^{i,j}$ $(i, j = 0, 1, 2, 3)$ of a 512 × 512, 8 bits/pixel image in real time.

## VI. ACKNOWLEDGMENTS

We would like to thank Neil Weste for providing the information on the size and the speed of the look-ahead adder circuits.

## REFERENCES

1 R. C. Gonzalez and P. A. Wintz, *Digital Image Processing*, Reading, MA: Addison-Wesley, 1977.
2. U. K. Hu, "Visual Pattern Recognition by Moment Invariants," IRE Trans. Info. Theory, *IT-8* (February 1962), pp. 179–87.
3. Z. L. Budrikis and A. N. Netravali, private communication.
4. R. L. Andersson, private communication.

## AUTHORS

**Zigmantas L. Budrikis,** B.Sc., 1955, and B.E. (Electrical Engineering, Hons I), 1957, University of Sydney; Ph.D., 1970, University of Western Australia; Research Laboratories of Telecom Australia, 1958–1960; Aeronautical Research Laboratories, 1961; on Faculty at University of Western Australia since 1962. In 1968 Mr. Budrikis was Visiting Lecturer at the University of California at Berkeley, and since 1972 he has been a recurrent visitor, as MTS or Consultant, at AT&T Bell Laboratories. He is interested in communications

and electromagnetics. Member, IEEE, Optical Society of America, NY Academy of Science. Fellow, IE Australia.

**Mehdi Hatamian,** B.S.E.E., 1977, Ary-Mehr University of Technology, Tehran; M.S. and Ph.D. (Electrical Engineering), University of Michigan, Ann Arbor, in 1978 and 1982, respectively; AT&T Bell Laboratories, 1982—. From 1979 to 1982 Mr. Hatamian was a Research Assistant on a NASA project working on the design and development of hardware and software for one of the Space Shuttle experiments. His research interests are in real-time signal processing, circuit design, image processing, and VLSI design. Member, IEEE, Sigma Xi.