END UNCLASS

Semiannual Technical Summary

Multi-Dimensional Signal Processing Research Program

30 September 1983

Lincoln Laboratory

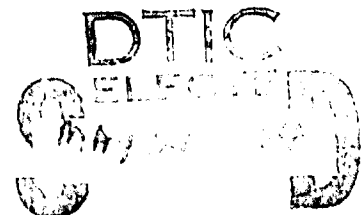### MASSACHUSETTS INSTITUTE OF TECHNOLOGY

LEXINGTON, MASSACHUSETTS

DTIC
ELECTE
MAY

84 05 21 106

The Public Affairs Office has reviewed this report, and it is releasable to the National Technical Information Service, where it will be available to the general public, including foreign nationals.

This technical report has been reviewed and is approved for publication.

FOR THE COMMANDER

Thomas J. Alpert, Major, USAF
Chief, ESD Lincoln Laboratory Project Office

MASSACHUSETTS INSTITUTE OF TECHNOLOGY
LINCOLN LABORATORY

MULTI-DIMENSIONAL SIGNAL PROCESSING RESEARCH PROGRAM

SEMIANNUAL TECHNICAL SUMMARY REPORT
TO THE
ROME AIR DEVELOPMENT CENTER

1 APRIL — 30 SEPTEMBER 1983

ISSUED 8 MARCH 1984

Approved for public release; distribution unlimited.

"Original contains color
plates: All DTIC reproduct-
ions will be in black and
white"

LEXINGTON   MASSACHUSETTS

ABSTRACT

This Semiannual Technical Summary covers the period 1 April through 30 September 1983. It describes the significant results of the Lincoln Laboratory Multi-Dimensional Signal Processing Research Program, sponsored by the Rome Air Development Center, in the areas of multiprocessor architectures for image processing and algorithms for object detection and region classification in aerial reconnaissance imagery.

TABLE OF CONTENTS

LIST OF ILLUSTRATIONS

LIST OF TABLES

MULTI-DIMENSIONAL SIGNAL RESEARCH PROGRAM

1. INTRODUCTION AND SUMMARY

The Lincoln Laboratory Multi-Dimensional Signal Processing Research Program was initiated in FY80 as a research effort directed toward the development and understanding of the theory of digital processing of multi-dimensional signals and its application to real-time image processing of aerial reconnaissance imagery. Current research projects that support this long-range goal are image modeling for target detection and multiprocessor architectures to implement image processing algorithms.

This Semiannual Technical Summary discusses our work in several areas. In Section 2, we presents a discussion of our recent efforts in improving the target detection algorithm developed in FY82. The improvements lie in modifying the algorithm to detect targets of size comparable to that of the estimation window.

In Section 3, the use of a systolic array architecture for implementing the target detection algorithm is discussed. Systolic arrays have great potential for utilizing the available parallelism in highly structured computations. A major drawback in some of these types of arrays is an inherent lack of flexibility. However, for some multi-dimensional signal processing applications, this lack of flexibility may not be important.

Section 4 contains a description of the custom-designed chip to implement the basic switching element for a butterfly interprocessor communications network. The custom chip was fabricated with NMOS technology using the DARPA silicon foundry facilities.

1

2. TARGET DETECTION

During the current reporting period, we have made several extensions to the linear predictive target detection algorithm to improve its performance on small- to medium-size targets and to extend it for detecting boundaries of larger objects. In addition, the algorithm, in slightly modified form, is suitable for segmentation of images of natural terrain by detecting boundaries between different terrain types.

2.1 ALGORITHM DESCRIPTION

The algorithm differs from the point object detection algorithm developed in FY82 as follows. Four separate one-dimensional (1-D) predictors are used to predict each point as shown in Figure 1. The prediction coefficients are independently generated for each direction using the covariance method of linear prediction. A small estimation window (typically 3×3 or 5×5 pixels) is used which does not include the test point (see Figure 1). The location of the estimation window is particularly important when the test point lies on a target border.

The use of four predictors introduces a desirable symmetry property. Further, most targets, such as tanks and trucks, have outlines which are simply connected and convex. In this case, the use of four predictors insures that most points on target borders will be predicted with coefficients generated primarily from background points. While various methods for combining the four prediction errors at each point have been tested, the simple approach of summing the four squared prediction errors yields the best results. The squared prediction errors are not normalized by the error variances. Such normalization is generally desirable but may not be feasible when dealing with larger targets. This is because larger targets may have a number of their own pixels appearing in the estimation window and background variance computations are corrupted by the presence of target pixels in the window.

The decision process has also been extended in this algorithm to take account of decisions made at neighboring pixels. If this were not done, three problems would arise. First, isolated noise pixels in background regions which sometimes exhibit high prediction errors would be erroneously classified as target pixels. Secondly, since some points along a target border may be statistically similar to the neighboring background points, these points would erroneously be classified as background. Finally, when 1-D predictors are used, the transition from one background region to another is similar to the transition from background to target. Simple thresholding of the prediction errors would erroneously cause background region boundaries to be classified as target pixels.

The foregoing considerations clearly suggest that the classification of a pixel as target or background should depend not only on the squared prediction error at that point, but also on the classification of surrounding points. A theoretical framework for the dependence can
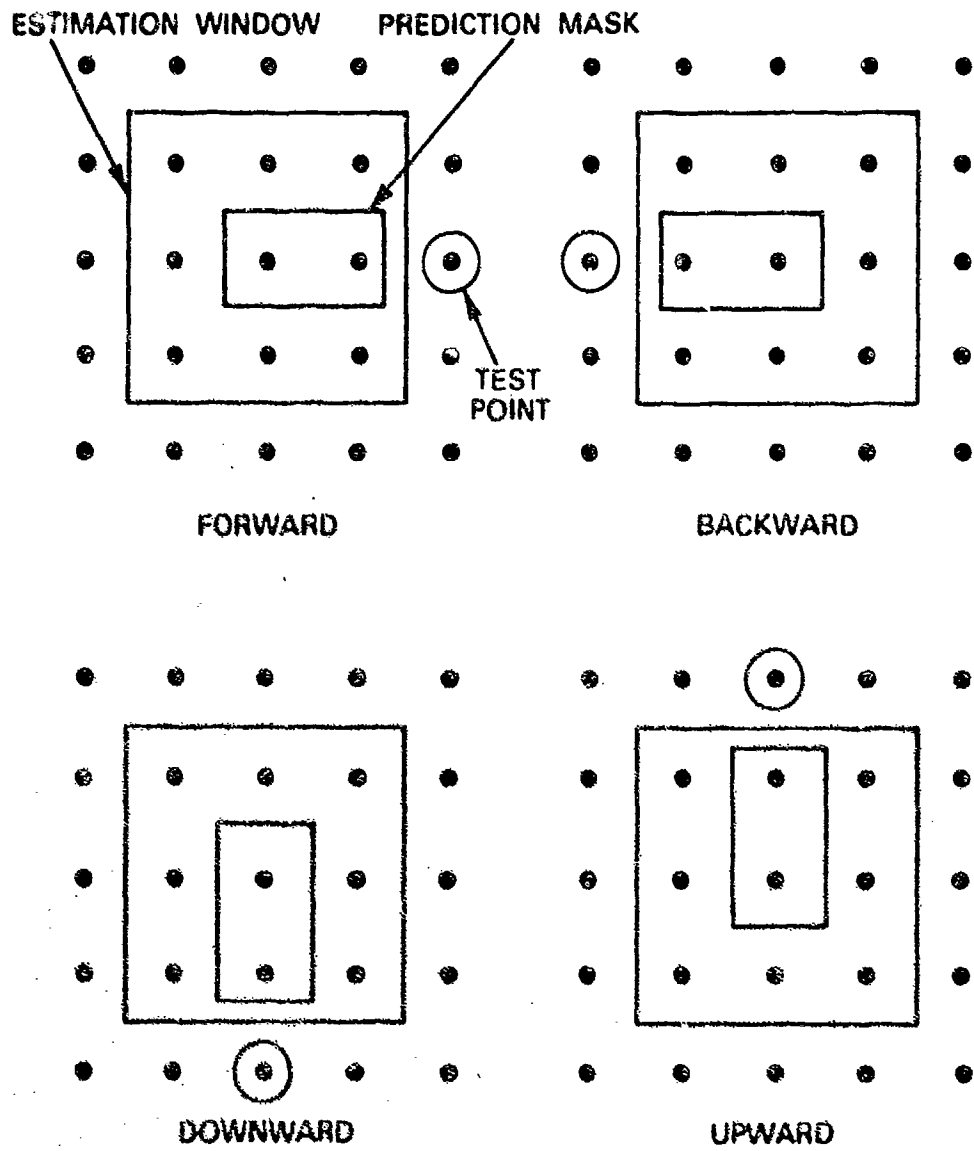
3

Figure 1. Four 1-D predictors and their estimation windows.

4

be provided by modeling the set of prior probabilities for the pixel classifications as a two-dimensional Markov process [see Reference 1]. If the prior probability is chosen to depend on a set of pixels that symmetrically surround the pixel in question, however, a circular problem arises since a point should not be declared 'target' unless some set of its neighboring points are also declared 'target'. However, this problem can be resolved in practice by iterating to a solution. That is, the thresholded prediction error is used to find an initial classification for the pixels. The pixel classifications are then updated based on the value of the prediction error and the current classification of the surrounding pixels. This process is then repeated using the new pixel classifications.

In our application of this method, we have typically chosen the Markov prior probabilities to depend on data in a 3×3 pixel window surrounding the given point. Depending on the threshold, the process converges, producing a consistent target map, after six to twelve iterations.

Two factors influence the success of the iterative algorithm. First, since targets are generally convex, a 3×3 window centered on any internal target point, or a border point, usually includes at least four other target points. Secondly, since region borders tend to be long and thin, a 3×3 window centered on a border point rarely includes more than one or two background points which may have been erroneously classified as 'target'. Thus, if the center point in the window has initially been declared 'target', the sparsity of target points in the window almost insures its reclassification as background.

2.2 EXPERIMENTAL RESULTS

This section describes the results of applying the large-target detection algorithm to aerial photographic data. Figure 2 depicts a 512×512 pixel image extracted from an aerial reconnaissance photograph. The two regions enclosed by rectangles are the subimages actually used for the tests. Both are 128×128 pixel regions which were subsampled to produce the 64×64 pixel test images shown in Figures 3a and 5a. The red overlays in Figures 3b and 5b indicate the points declared as 'targets' by the algorithm.

Figures 4 and 6 show the areas of high prediction error and high prediction error variance. In both tests, the thresholded squared errors outline the targets, or vehicles, fairly well. However, the variance is also extremely high at each target point. Normalization by the variance would tend to decrease the ability to detect these larger targets.

A variant of the target detection algorithm can also be used to detect boundaries between various textures existing in images of natural terrain and thus can be used for segmenting these images. Ordinary edge detection algorithms cannot be used for this purpose since multiple edges are found within the textures. Figure 7a shows the results of processing a photograph of trees and field with a Sobel edge detector. Figure 7b shows the result of applying the linear predictive algorithm. The difference is clearly evident.

5
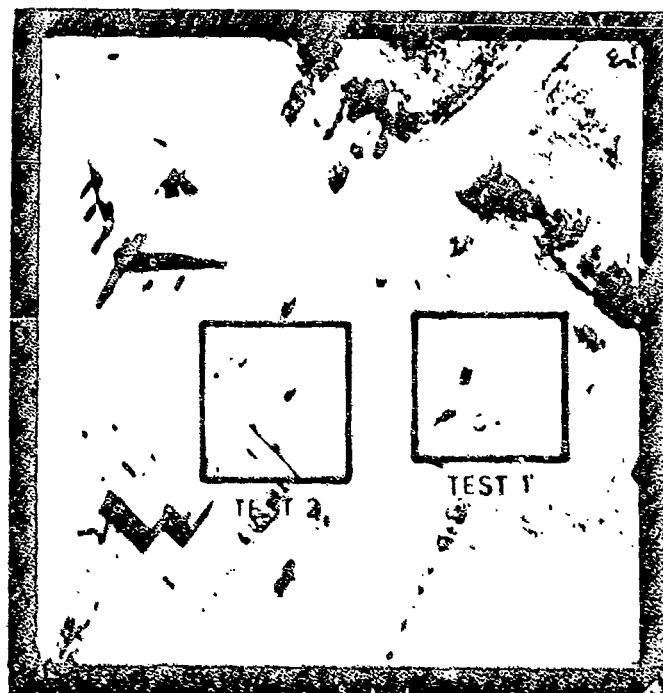
Figure 2   512 × 512 pixel aerial reconnaissance photograph

Figure 3   Test 1 (a) 64 × 64 pixel test image (b) Target detection results

Figure 4. Threshold statistics from STLP algorithm for Test 1. (a) Squared prediction errors (b) Prediction error variances.
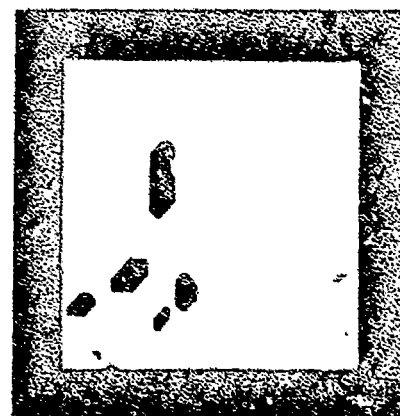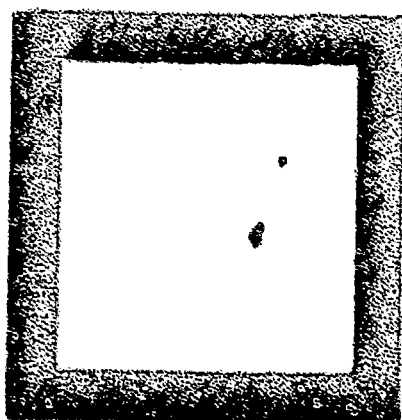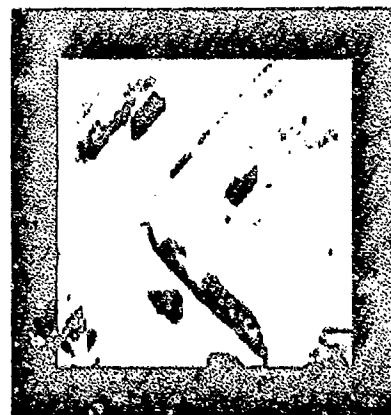
Figure 5. Test 2. (a) 64 · 64 pixel test image. (b) Target detection results

135022 S

135023 S
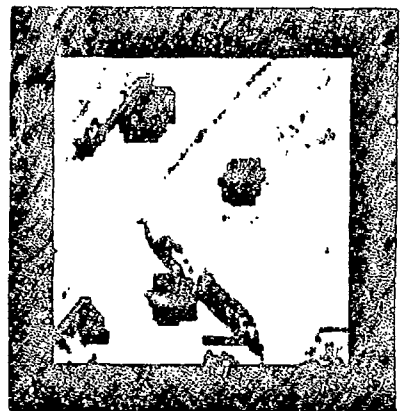
Figure 6   Threshold statistics from STLP algorithm for Test 2. (a) Squared prediction errors. (b) Prediction error variances.
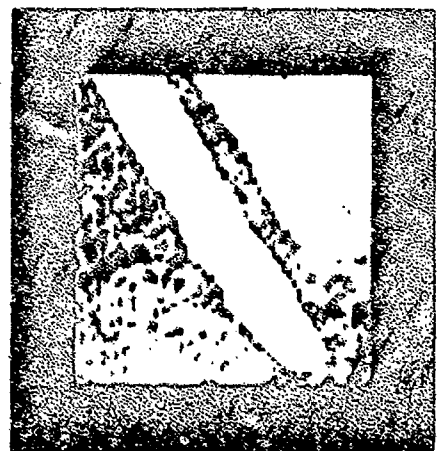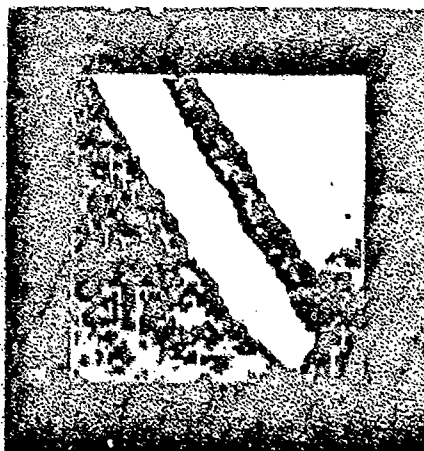
Figure 7   (a) Processing by Sobel edge detection. (b) Processing by linear predictive algorithm

11

Work is continuing to evaluate the algorithm on additional aerial photographic data and to improve deficiencies that may be discovered. Further documentation will be provided in a Master's Thesis by M. D. Richard (M.I.T. 1984), and in a paper to be presented at the 1984 International Conference on Acoustics, Speech, and Signal Processing [Reference 2].

3. SYSTOLIC ARRAY PROCESSOR DESIGN
FOR TARGET DETECTION

During the current reporting period, we have been developing an architecture for a special-purpose systolic array processor that would implement the object detection algorithm developed by us in FY82 [References 3,4]. We believe the processor could be implemented with wafer-scale integration using the Lincoln restructurable VLSI (RVLSI) technology [Reference 5] and would have throughput sufficient to provide the processing required in the MIES application (see Table I in Section 3.3.2). Details of the architecture are described in this section.

3.1 TARGET DETECTION ALGORITHMS

We begin by reviewing the steps in the small object detection algorithm and detail the mathematical computations involved. Recall that the algorithm employs 2-D linear prediction to adaptively estimate pixel intensity values from a set of neighbors (see Figure 8).
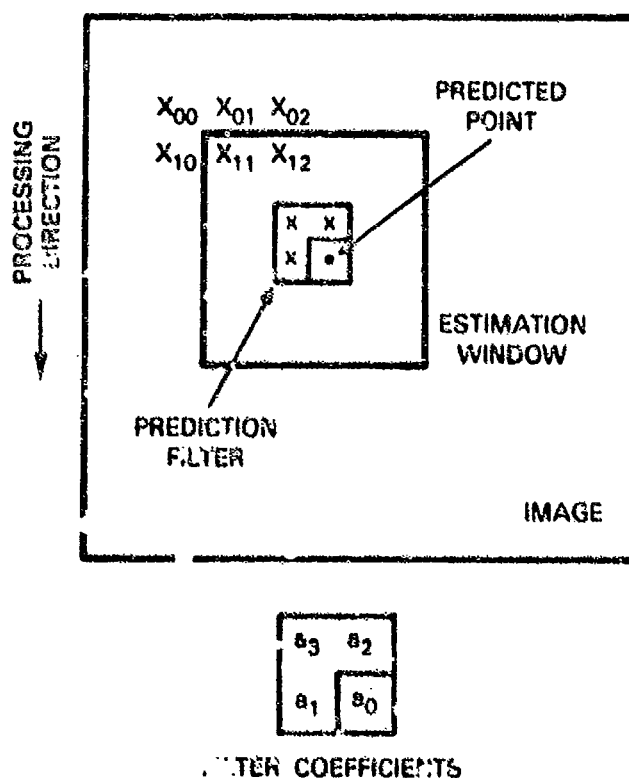
Figure 8. Linear prediction for target detection.

14

Estimates for the prediction filter coefficients are made from data in a larger estimation window centered on the point to be predicted. The image is scanned along columns and this operation is repeated at each pixel. When the predicted pixel value does not match the measured pixel value, the corresponding pixel is marked as a possible 'target'.

The computational steps of the algorithm can be summarized as follows. First, a 2-D covariance matrix is formed from data in the estimation window. Next, a system of Normal equations is solved to obtain the prediction filter coefficients $\underline{a}$ and the prediction error variance σ^2. Finally, the prediction error is computed by applying the filter to the data at the center of the window and is normalized by σ. This normalized prediction error is then compared to a threshold to perform the target detection. A postfiltering step which has optionally been applied in the original version of the algorithm has so far not been included in the design. However, this step would be straightforward to implement as a part of the processor or as an external operation applied prior to display.

The steps just described can be implemented as a connected set of systolic arrays as shown in Figure 9. Data enters the lower array and elements of the covariance matrix are computed. As these elements are computed, they flow into the next array which begins computation of the filter coefficients. Finally, the filter coefficients are applied to the delayed input data to compute the prediction error. Details of the algorithm and array structures are given in the next two sections.

3.2 COMPUTATIONAL METHODS

In the following, it will be assumed that the linear predictive filter is 2×2 pixels and the estimation window is 8×8 pixels in size. These parameter values were found to be suitable for processing of typical photographs collected by low flying reconnaissance aircraft. The image may be of any size and processing is assumed to proceed along columns.

3.2.1 Covariance Computation

Computation of the covariance matrix requires data within the estimation window and in one row above and one column to the left. Some of the required data points are labeled in Figure 8. The mean is assumed to be removed from the data prior to the following operations. This could be done in a separate pass through the image or simultaneously with computation of the covariance in a method described later. Conceptually, a large matrix S is formed from this data and partitioned as shown below.

THRESHOLDED ERRORS EXIT
(Target Detection Data)

COMPUTE
PREDICTION ERROR

$$\epsilon = \frac{|a^T \underline{X}|}{\sigma} > T$$

(Systolic Array)

COMPUTE
FILTER COEFFICIENTS
$\underline{a}$ AND ERROR
VARIANCE σ^2

(Systolic Array)

DELAY

COMPUTE
COVARIANCE
MATRIX R

(Systolic Array)

IMAGE DATA ENTERS
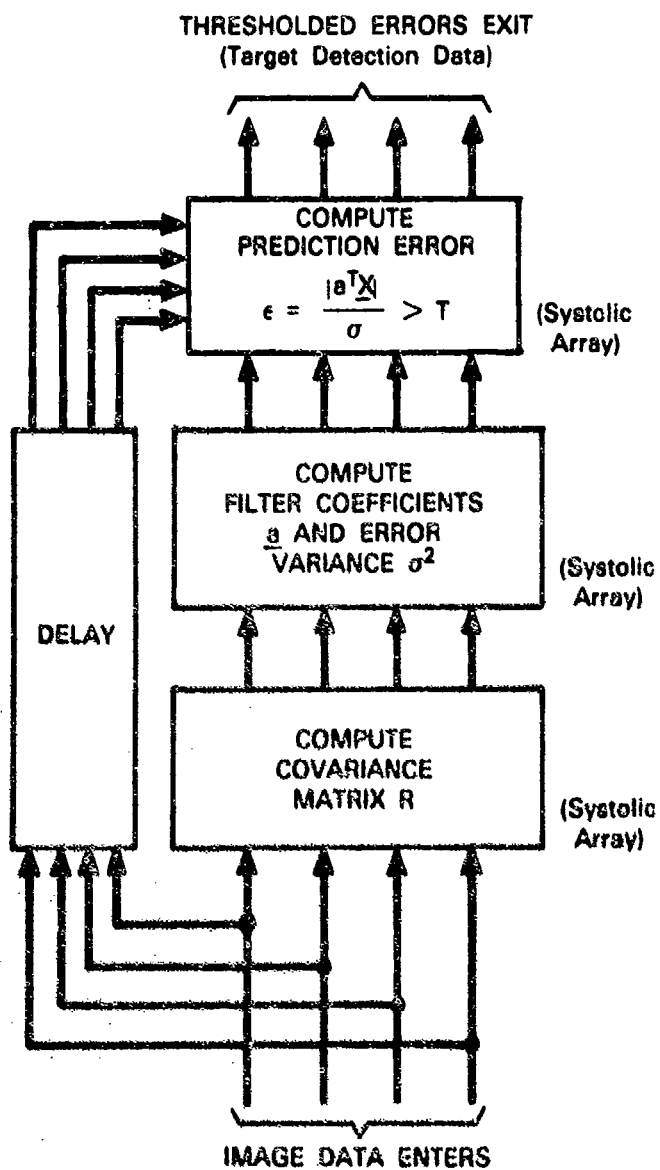
Figure 9. Flow of data and computation.

16

$$S = \begin{bmatrix} X_{00} & X_{01} & X_{10} & X_{11} \\ X_{01} & X_{02} & X_{11} & X_{12} \\ & \bullet & & \\ & \bullet & & \\ & \bullet & & \\ X_{07} & X_{08} & X_{17} & X_{18} \\ \hdashline X_{10} & X_{11} & X_{20} & X_{21} \\ X_{11} & X_{12} & X_{21} & X_{22} \\ & \bullet & & \\ & \bullet & & \\ & \bullet & & \\ X_{17} & X_{18} & X_{27} & X_{28} \\ \hdashline & & & \\ \hdashline X_{70} & X_{71} & X_{80} & X_{81} \\ X_{71} & X_{72} & X_{81} & X_{82} \\ & \bullet & & \\ & \bullet & & \\ & \bullet & & \\ X_{77} & X_{78} & X_{87} & X_{88} \end{bmatrix} \begin{matrix} \Big\} S_0 \\ \\ \\ \\ \Big\} S_1 \\ \\ \\ \\ \Big\} S_7 \end{matrix} \qquad (1)$$

Each row of S is formed by placing the filter mask at a selected point in the estimation window (say at X_{11}) and listing the points under the mask from top left to bottom right. The covariance matrix is then computed from

$$R = S^T S$$

$$= S_0^T S_0 + S_1^T S_1 + \ldots + S_7^T S_7 \quad . \qquad (2)$$

The covariance matrix expressed in this form is well suited for computation on a systolic array of simple cells that perform the scalar operation $c \leftarrow c + a \cdot b$. A first array computes the matrix products $S_i^T S_i$ in succession. These terms are then fed into another array that computes a running sum of eight terms. In this way, the structure is able to compute one entire new covariance matrix as each row of data in the estimation window is scanned.

3.2.2. Solving for the Filter Coefficients

Once the covariance matrix is computed, the filter coefficients can be obtained by solving a set of linear equations. These equations, known as the Normal equations, take the form

$$\frac{1}{N} R \underline{a} = \begin{vmatrix} 0 \\ 0 \\ 0 \\ \sigma^2 \end{vmatrix} \qquad (3)$$

where $\underline{a}$ is the vector of filter coefficients in the order $[a_3, a_2, a_1, a_0]^T$, σ^2 is the theoretical prediction error variance, and N is a normalizing factor equal to the number of terms in the estimation window that were summed to compute R (64 in our case). The Normal equations are usually written with the factor N included in the definition of R, and the matrix is usually arranged so that the terms in the two vectors of (3) appear in reversed order. The form (3) is most convenient for our computations, however, because of a relation that exists between the filter coefficients and a triangular decomposition of the covariance matrix [Reference 6]. In particular, since R can always be written as the product of a lower triangular matrix L with ones on the diagonal, and an upper triangular matrix U, (3) can be rewritten as

$$U \underline{a} = L^{-1} \begin{vmatrix} 0 \\ 0 \\ 0 \\ N\sigma^2 \end{vmatrix} = \begin{vmatrix} 0 \\ 0 \\ 0 \\ N\sigma^2 \end{vmatrix} . \qquad (4)$$

(The last equality follows from the fact that L^{-1} is also lower triangular with ones on its diagonal.) Thus, if an LU decomposition is performed on the covariance matrix, a set of normalized filter coefficients can be found by solving the triangular linear system

$$U (\underline{a}/N\sigma^2) = \begin{vmatrix} 0 \\ 0 \\ 0 \\ 1 \end{vmatrix} . \qquad (5)$$

These normalized coefficients are sufficient since the desired prediction error is normalized by σ and the remaining factor $N\sigma$ can be included in the threshold (see Figure 9). Fortunately, systolic array configurations exist both for performing the LU decomposition and for solving the triangular linear system (5). Thus these computations, like those needed to compute R, can be pipelined through simple processors.

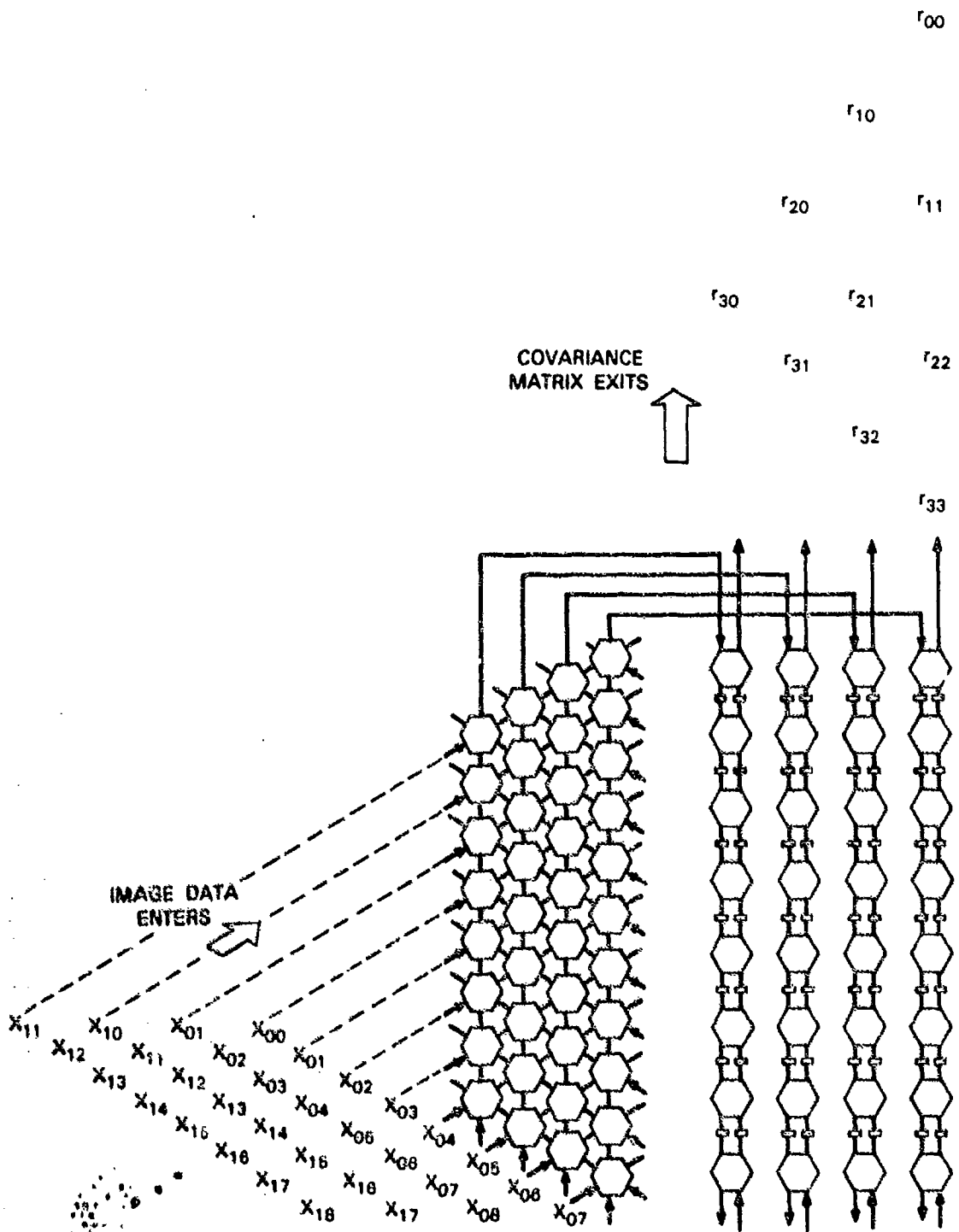r_{00}

r_{10}

r_{20} r_{11}

r_{30} r_{21}

COVARIANCE
MATRIX EXITS r_{31} r_{22}

r_{32}

r_{33}

IMAGE DATA
ENTERS

Figure 10. Systolic arrays for computation of the covariance matrix.

19

3.3 SYSTOLIC ARRAYS

3.3.1 Covariance Matrix Computation

A correlation matrix for the set of data in the estimation window can be computed on a set of systolic arrays of cells that perform a multiply-accumulate (MAC) function [References 7,8]. The covariance matrix can then be formed by removing a constant mean value from each correlation term in a manner to be described later. The set of arrays needed to compute the correlation matrix is shown in Figure 10. Data enters the sides of the hexagonally-connected array and is clocked through the cells. Terms of the complete correlation matrix exit from the top of the linearly-connected arrays. The small boxes represent unit delay registers where data enters and simply exits on the next clock cycle. Specific operation of the MAC cells is detailed in Figure 11. The hexagonally connected array at the left of Figure 10 computes a single matrix product $S_i^T S_i$. The linear arrays

$$c \leftarrow c + a \times b \qquad\qquad c \leftarrow c + a$$
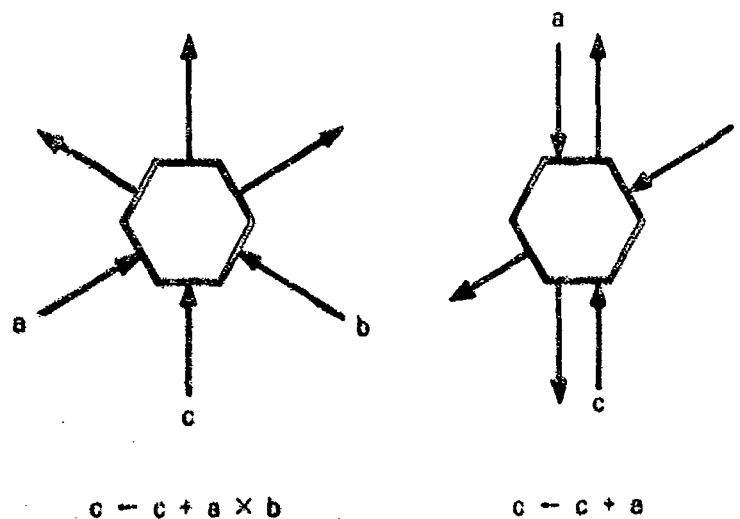
Figure 11. Multiply-accumulate (MAC) cells.

then compute the running sum of eight such products. Data provided to the array can be thought of as being produced by a set of (four) scanning devices that scan along rows of the estimation window. Each such device produces one column in the matrix S_i. Since processing of the image involves moving the estimation window along columns, scanning can proceed continuously from top to bottom of the image (see Figure 3). As a result, data enters arrays continuously.

Each column of the arrays computes terms along a diagonal of R that is separated by the clock period T. Corresponding terms in the products $S_i^T S_i$ and $S_{i+1}^T S_{i+1}$ are separated by 4T. The linear arrays operate at the same clock period and provide sums of terms sampled at every fourth term.

20

Figure 12. Arrays for computation and removal of the mean value.

A procedure for removing the mean as the data enters is shown in Figure 12. One row of data entering the hexagonal array is first passed through a linear array which computes the sum of its terms. The row sums are fed through another linear array to compute the sum of terms in the estimation window. This sum is squared, divided by N and subtracted from the correlation terms exiting the main arrays.

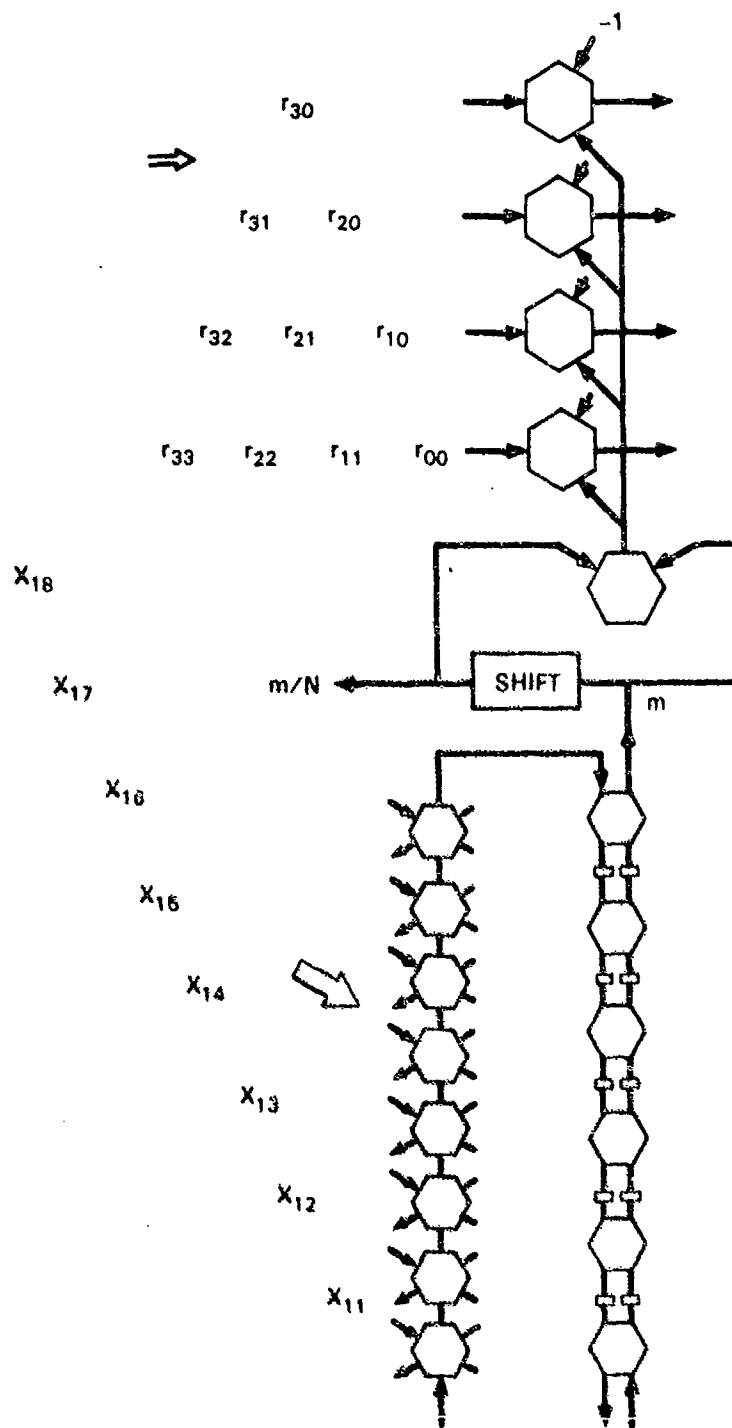The LU decomposition for solving the Normal equations can also be performed on a hexagonally connected array. Solution of the resulting triangular system is performed on a linear array. The configuration is shown in Figure 13. These arrays required two special cells detailed in Figure 14 and a set of LIFO buffer registers for temporary storage of terms resulting from the LU decomposition. Details of the array operation are given in Reference 7. Throughput of these arrays is slower than that of Figure 10 by a factor of 3. Thus, corresponding adjustments must be made in the clock rates. Alternative systolic array designs for performing LU decomposition are possible [References 9,10], but they do not match the architecture and data flow of the rest of the system as well as the configuration of Figure 13 does.
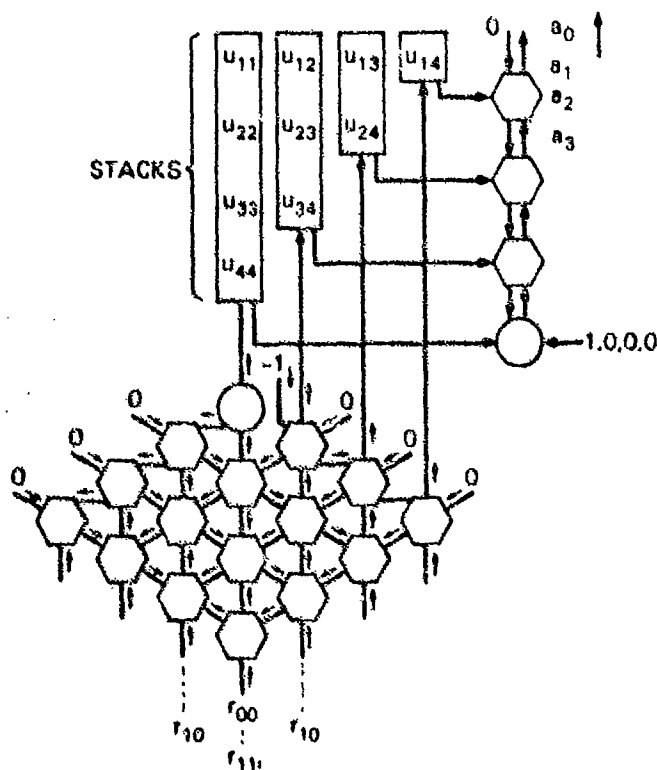

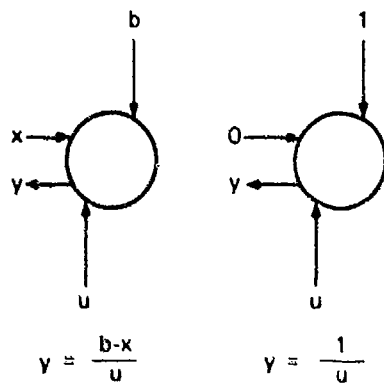Figure 13. Systolic arrays for solving Normal equations.

Figure 14. Special cells to perform division.
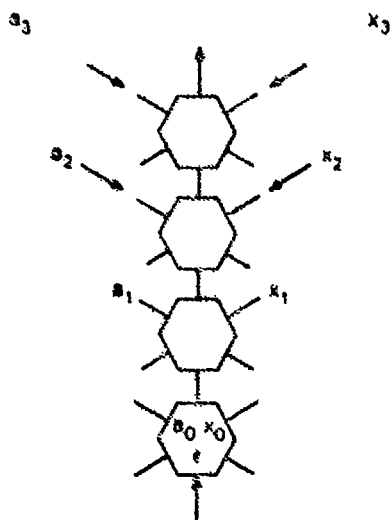
$$y = \frac{b-x}{u} \qquad y = \frac{1}{u}$$

Figure 15. Array for linear predictive filtering.

The final step in the detection algorithm is to apply the prediction coefficients to the data in the center of the estimation window to produce the normalized prediction error. This is done with the simple linear array shown in Figure 15.

3.3.2 Timing and Throughput

Figure 16 shows a time-line of events in the computation sequence for the target detection algorithm. Currently, throughput is limited by the array that performs LU decomposition. Since this array has only 1/3 the throughput of the arrays that compute the covariance, the clock rate for the covariance array has to be slowed from T to 3T. The linear array of Figure 13 that computes the solution of (5) requires two clock cycles for every three cycles of the LU decomposition array. Thus, the overall system should be provided with a clock with period T/2; this clock rate should be divided by 6 before application to the arrays of Figures 10 and 12, divided by 2 before application to the hexagonal array in Figure 14, and the linear array in Figure 15, and divided by 3 before application to the linear array in Figure 13.

The total initial delay for processing of a new column of image data is 154T. Thereafter, error residuals are computed at a rate of 12T. The initial delay can be represented as an 'overhead rate' and is listed as a percentage of the steady-state processing time in Table I. Table I also gives the throughput of the processor for various sized images computed for $T = 1\mu s$.

TABLE I		
Target Detection Algorithm Processing Rates for Various Image Sizes		
Image Size	Overhead (percent)	Processing Rate* (frames/sec)
64 × 64	22	16.7
128 × 128	11	4.6
256 × 256	6	1.2
512 × 512	3	0.3
1024 × 1024	1	0.08
*Based on $T = 1 \mu s$.		

3.3.3 VLSI Implementation Considerations

The regular structure of the arrays and the use of a large number of simple MAC cells should allow the entire processor for target detection to be implemented with current VLSI
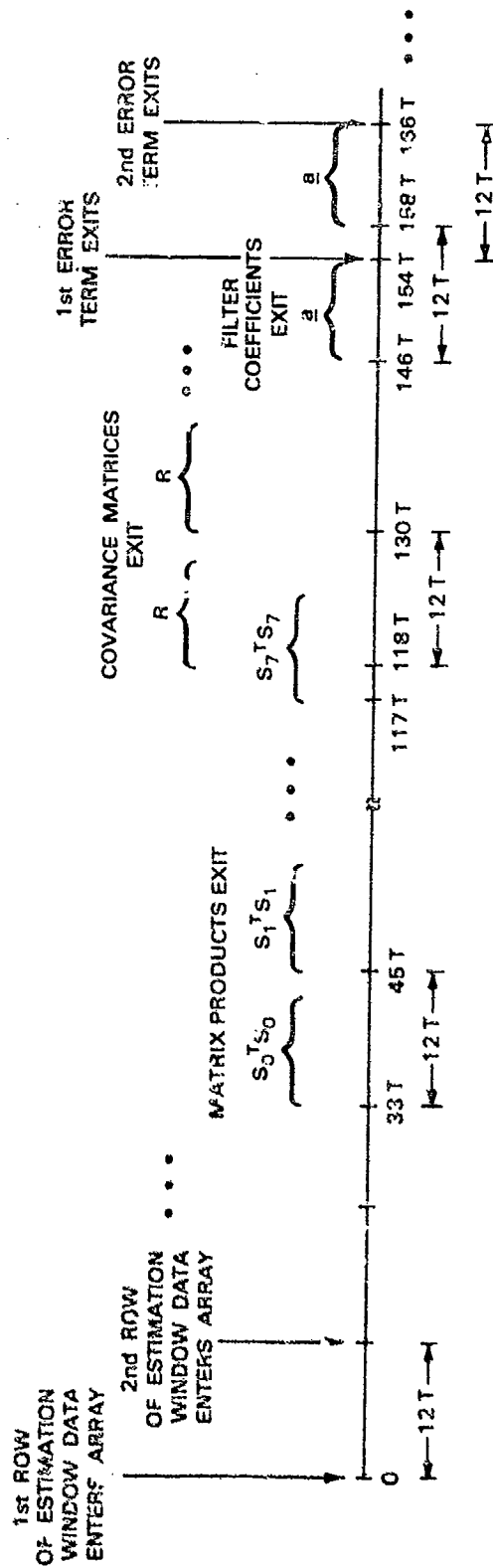
Figure 16. Time-lines for target detection algorithm.

25

wafer-scale technology. These considerations, in fact, led us to choose the present design for the processor over alternatives that use fewer but considerably more complex cells [see, e.g., Reference 10].

The target detection processor described here requires 109 MAC cells, two special-purpose divide cells, buffer storage and delay. We believe these requirements could be met using the wafer-scale restructurable VLSI technology being developed at Lincoln Laboratory. A system for performing 16-point FFTs at a data rate of 16 MHz on a 3-inch wafer has already been developed and tested [Reference 11]. This system employed 128 16-bit MAC cells similar to those required for the target detection processor. These cells, implemented with 5-μm two-level metal CMOS technology using bit serial multipliers were able to perform one 16-bit multiply-and-accumulate in one microsecond. Restructurable VLSI allows working cells within a wafer to be utilized and defective cells to be ignored in forming the intra-wafer connections that comprise the processor. The two specialized cells of Figure 13 require 16-bit fixed-point division. However, because of their small number, various designs, such as a non-restoring divider, appear to be suitable for VLSI implementation and would have the necessary throughput.

4. SINGLE-CHIP SWITCHING ELEMENT

In the previous Semiannual Technical Summary [Reference 1], we discussed the design of a custom single-chip implementation of the switching element used in a butterfly interconnection network. In this section, we shall briefly review the requirements for the network and describe the switching element which has been fabricated.

4.1 REVIEW OF THE BUTTERFLY INTERCONNECTION NETWORK

A butterfly interconnection network is useful for providing communications between N processors where N is equal to a power of two. In contrast to a crossbar switching system whose complexity grows as N^2, the butterfly interconnection network has a complexity that grows only as $(N/2) \cdot \log_2 N$. An example of a butterfly interconnection network for N = 16 is shown in Figure 17. The processors connected via the butterfly network are labeled P0 through P15, but for convenience of illustration, the processor outputs are found on the left side of the figure while the processor inputs are found on the right side of the figure.

Although the butterfly network does not possess the complete connectivity of the crossbar switch, it nevertheless supports several important classes of simultaneous communications between pairs of processors. Our experience has shown that this type of network can support the communication necessary for the efficient multiprocessor implementation of important signal processing operations such as fast Fourier transforms, convolutions, and processing pipelines. For image processing applications, efficient use of a multiprocessor system can be made in many cases simply by partitioning the image data so that each processor can work independently on its own subimages with the butterfly network providing any communications required by the processors working on neighboring subimages.

The butterfly interconnection network, shown in Figure 17, can be used to support up to 16 parallel communication channels between 16 pairs of processor outputs and inputs. (In general, a butterfly network can support up to N parallel channels, where N is the number of processors.) In addition, it will support a broadcast mode where any single processor output can transmit messages to the inputs of all the processors.

4.2 REVIEW OF THE SWITCHING ELEMENT FUNCTION

Each circle in Figure 17 represents a single butterfly switching element and its associated control logic. Each switching element can be set to one of four external states: straight, crossed, upper-broadcast, and lower-broadcast, as shown in Figure 18. We have designed a custom LSI integrated circuit that implements one two-input, two-output switching element. Each chip is capable of switching two 8-bit-wide data paths as well as the necessary control signals. These chips can be used to implement butterfly networks of any size up to and including N = 256. Each chip also has the capability of acting in 'slave'
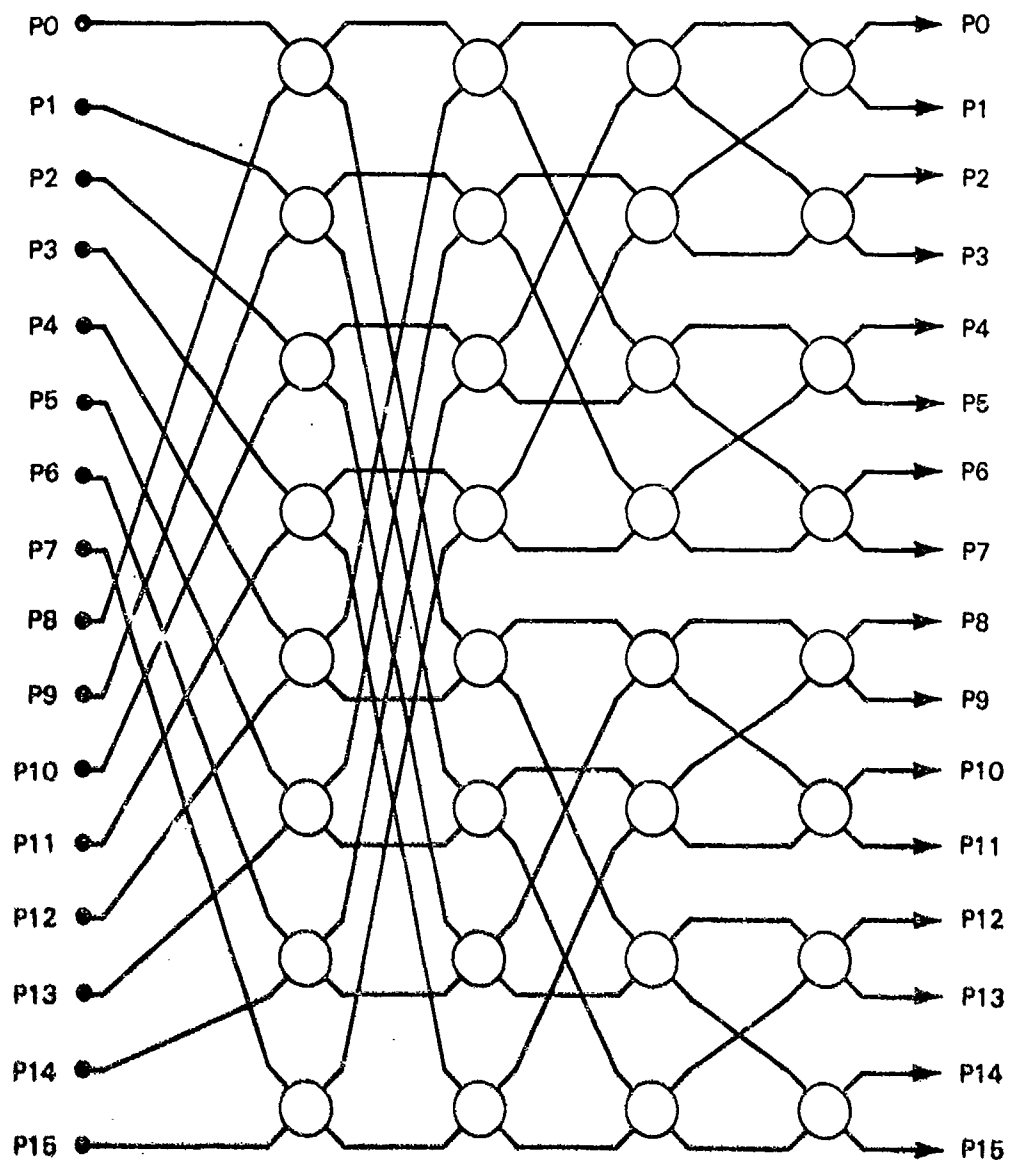
27

Figure 17. A butterfly interconnection network. For simplicity, processor outputs are indicated on the left and processor inputs are indicated on the right.

29

CROSSED

STRAIGHT
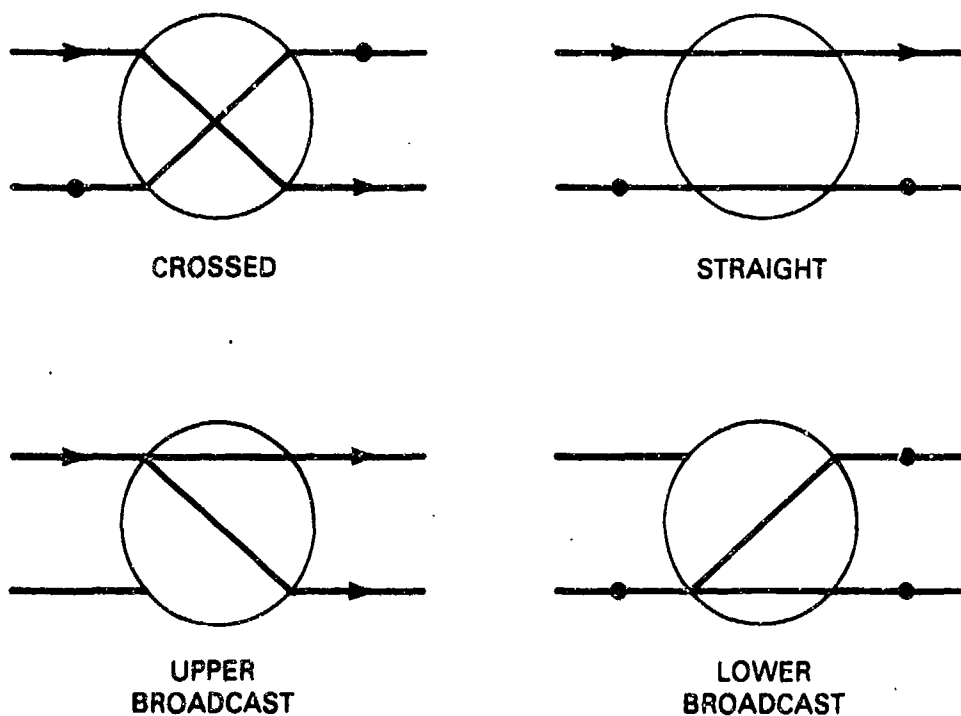
UPPER BROADCAST

LOWER BROADCAST

Figure 18. Four external states of the butterfly switching element.

mode, using the control signals of another chip (its 'master') to determine its external state. This permits two chips to implement a 16-bit-wide switch, three chips to implement a 24-bit-wide switch, and so on.

4.3 PHYSICAL DESCRIPTION OF THE SWITCHING ELEMENT CHIP

Figure 19 shows a photograph of the single-chip switching element that was designed and fabricated. It was designed using the Lincoln Integrated Circuit Language (LICL) on a VAX 11/780 computer with an AED color display terminal. LICL is a low-level computer-aided design tool that permits the designer to lay out rectangles in the various layers of the integrated circuit. The Mead-Conway design rules for NMOS were used and design rule checkers and a SPICE simulator were applied to the design to verify its operation. The chip was fabricated using the DARPA 'silicon foundry' facility administered by the Information Sciences Institute of the University of Southern California. Four-micron linewidths were used in the fabrication process. The photograph of the chip shows clearly that there is a great deal of unused silicon in this particular design. The chip complexity was not constrained by available silicon area but rather by the number of pins available on the package (64 in this case). Because of this limitation, we chose to implement 8-bit-wide data paths in our switching element.

Physically, the chip consists of two major elements. The center of the chip contains the finite-state controller consisting of a programmable logic array (PLA) and a static register. The second major element, the actual switches, are distributed around the periphery of the
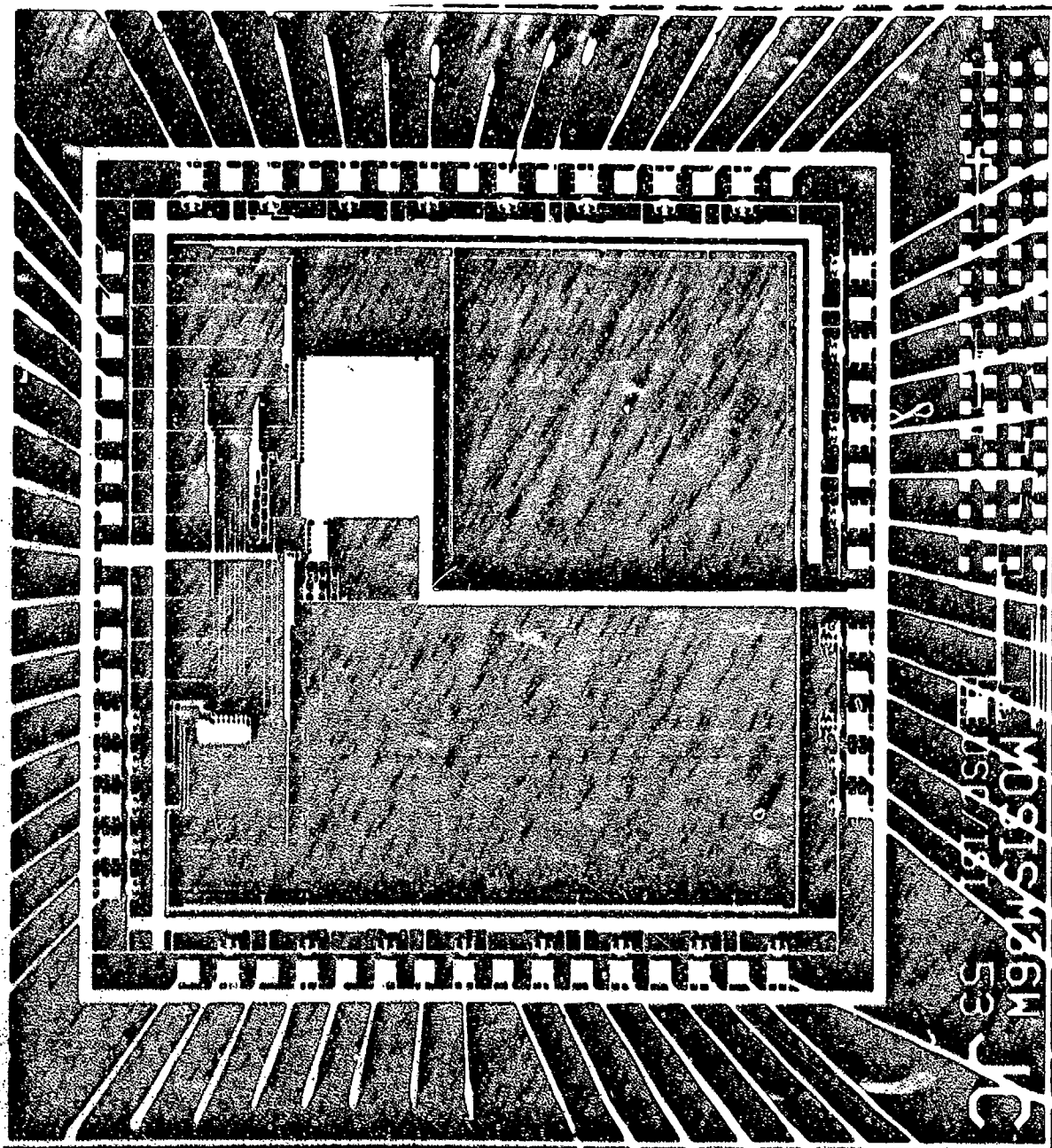
30

Figure 19. Custom integrated circuit to realize a switching element

chip next to the bonding pads. Once the switches have been set by the controller, the signals being switched travel only very short distances on the chip. Figure 20 shows a close-up of a 4-pad section of the chip's periphery. Pads 1 and 3 are input pads and are distinguished by a lack of pad-driver circuitry from the output pads 2 and 4. The switching function is embodied in the six inverters and associated pass transistors shown above the four pads. Depending on the values of two control variables determined by the finite-state controller, pad 1 is connected to pad 2 and pad 3 is connected to pad 4, or pad 1 is connected to pad 4 and pad 3 is connected to pad 2. In addition, the switch can be set to two broadcast states as indicated in Figure 18. In one state, pad 1 is connected to both pads 2 and 4, and in the other state, pad 3 connected to both pads 2 and 4. The switching function is indicated schematically in Figure 21 with the two control variables represented by 'x' and 'y'. For comparison, Figure 22 shows the color layout design for a similar (not identical) 4-pad switching unit. The colors used are the standard Mead-Conway convention. Blue represents metal, green represents diffusion, red represents polysilicon, and yellow represents ion implantation. The widest horizontal blue strip above the pads represents ground potential and the horizontal blue strip above that represents supply voltage potential (typically 5 volts). Thus, the six inverters that perform the switching function are seen to straddle the power and ground buses. The two thin horizontal blue strips above the power bus carry the two control signals, 'x' and 'y'. Inverters 1 and 4 are driven by the input pads 1 and 3, respectively, inverters 2 and 5 are driven by the control signals and perform the switching function, and inverters 3 and 6 drive the output pad drivers for pads 2 and 4, respectively. There are ten such 4-pad switching units for switching eight bits of data plus a data-strobe signal and a data-acknowledge signal. When used in the slaved configuration, the latter two switching units could be used to switch parity or error correction bits.

4.4 PRELIMINARY TEST RESULTS

The custom chips returned from the DARPA 'silicon foundry' were tested using a chip-tester and were verified to be logically correct. The finite-state controller worked in all the appropriate modes. The time to reconfigure the switching element ranged between 100 and 200 nanoseconds. Once the switch was set, data could be pumped through the chip at a 20 MHz rate in point-to-point mode and at a 15-MHz rate in broadcast mode. The chip uses a 5-volt supply voltage and is TTL-compatible. It drew 60 milliamps and dissipated 300 milliwatts.

A second design was submitted for fabrication in mid-September. The new design reduced some of the parasitic capacitances to permit even higher signaling rates, and removed access to some of the test signals so that other system-level communications signals could be brought out to the pins.

33

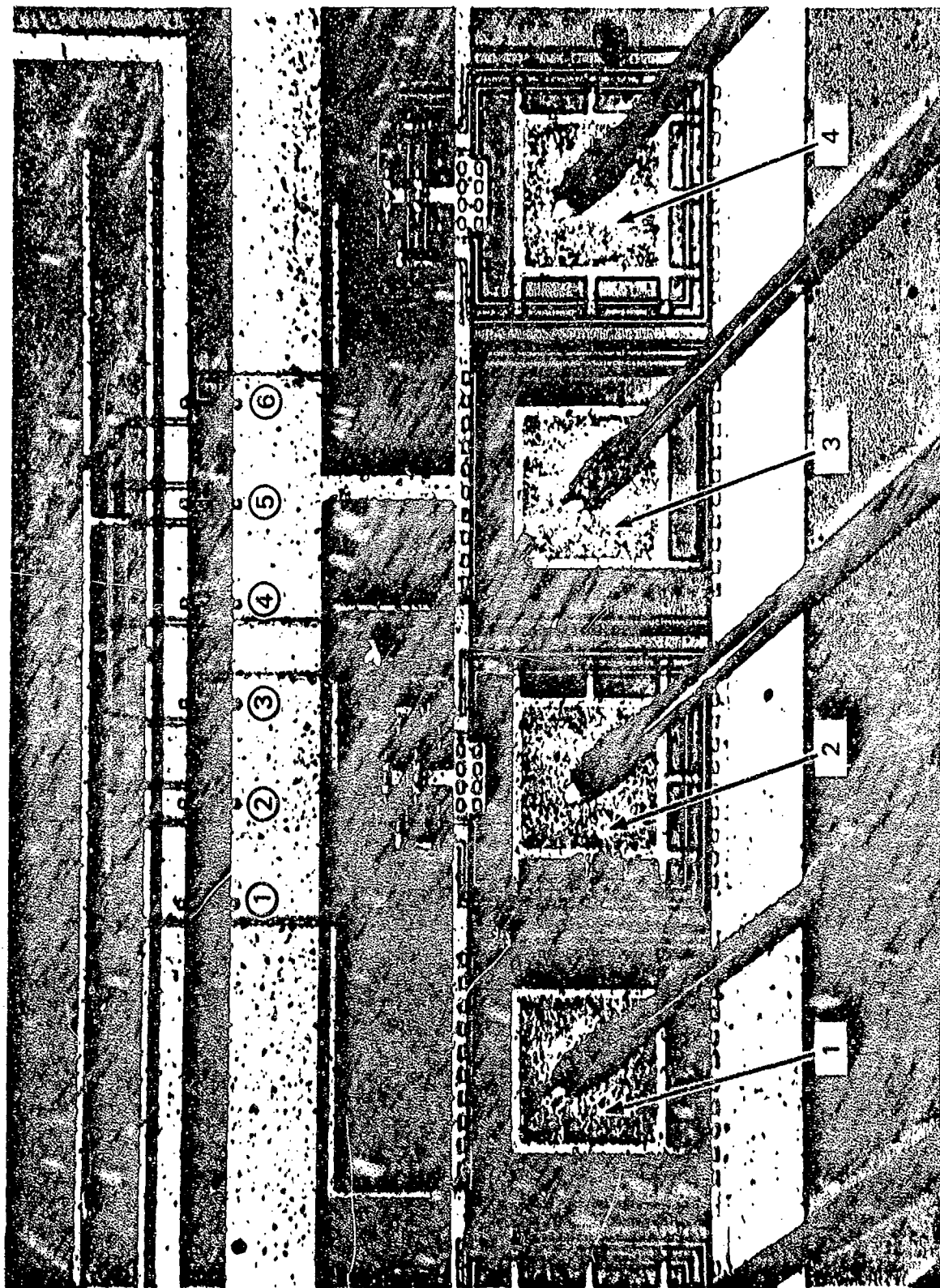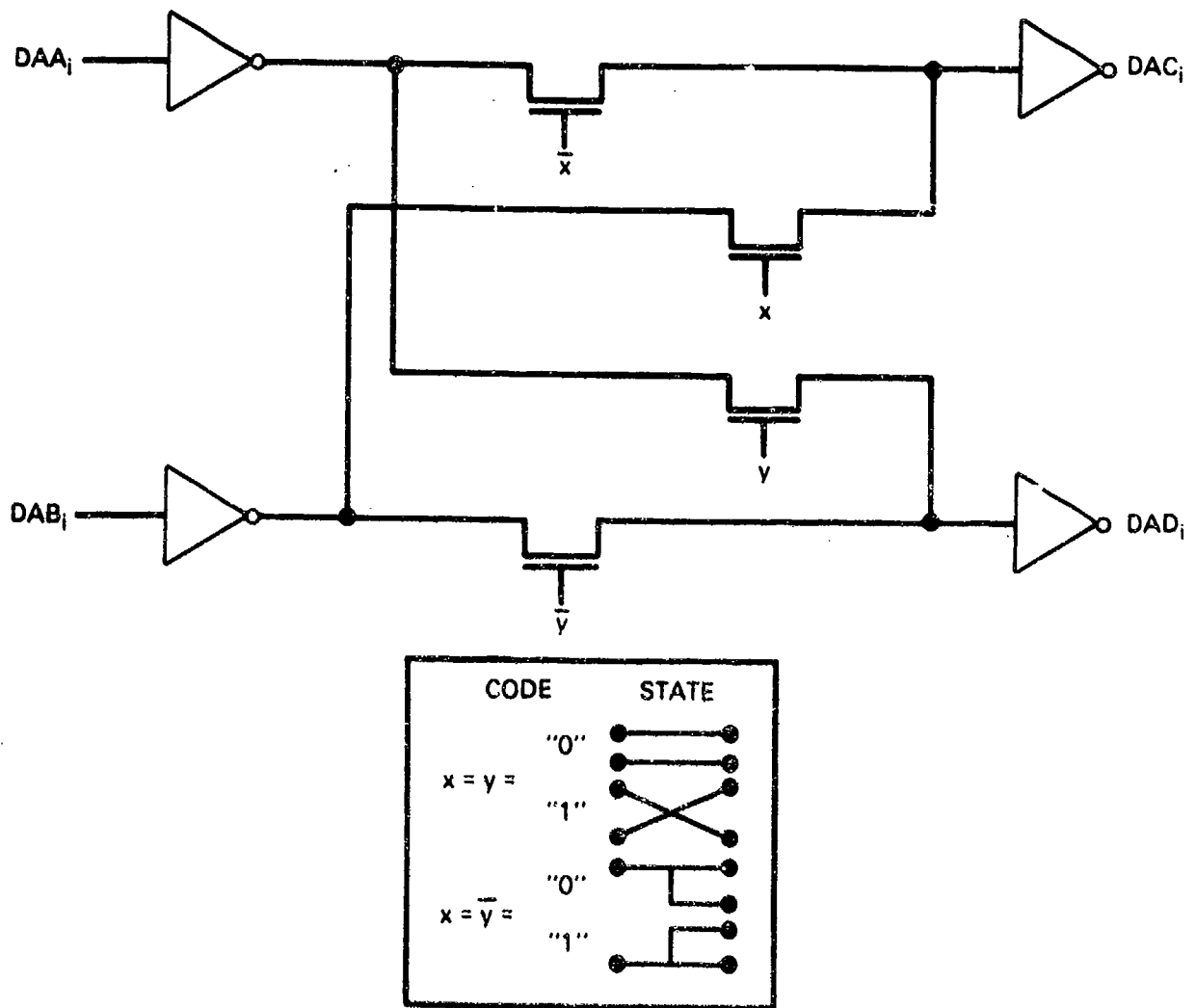Figure 20. A close-up photograph of a 4-pad section of the switching element chip

135028 S

35

Figure 21. Switching function schematic diagram.
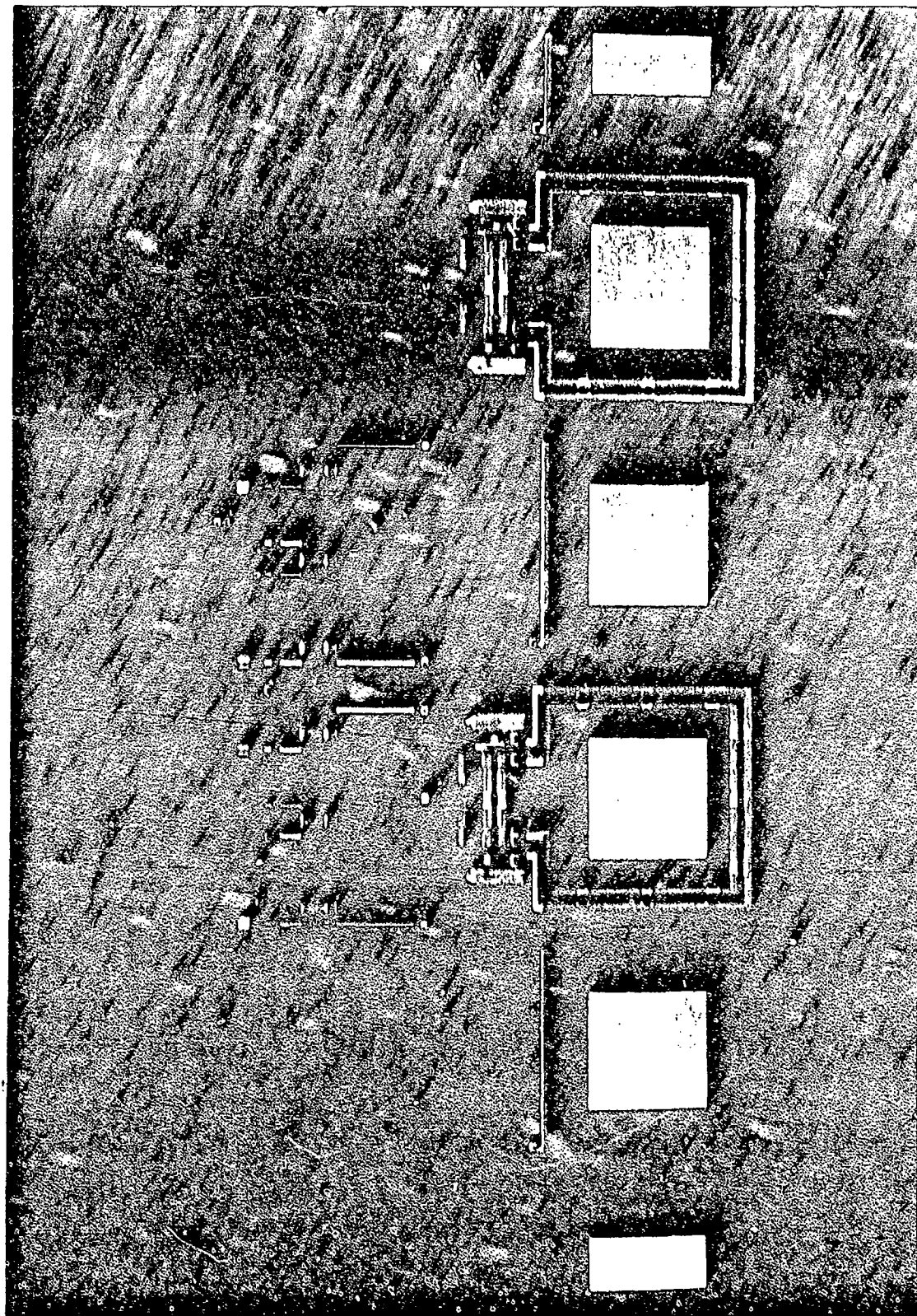
37

Figure 22 CAD display of a 4-pad section of the switching element chip. (Compare to Figure 20.)

39

REFERENCES

1. Semiannual Technical Summary, Multi-Dimensional Signal Processing Research Program, Lincoln Laboratory, M.I.T. (31 March 1983), DTIC AD-A135732.

2. M.D. Richard, C.W. Therrien, and J.S. Lim, "Image Segmentation Using Spatial Linear Prediction," *Proc. IEEE Intl. Conf. on Acoustics, Speech, and Signal Processing*, San Diego, March 1984.

3. T.F. Quatieri, "Objective Detection by Two-Dimensional Linear Prediction," *Proc. IEEE Intl. Conf. Acoustics, Speech, and Signal Processing*, Boston, 14-16 April 1983.

4. T.F. Quatieri, "Objective Detection by Two-Dimensional Linear Prediction," Technical Report 632, Lincoln Laboratory, M.I.T. (28 January 1983) DTIC AD-A126340/9.

5. J.I. Raffel, *et al.*, "A Demonstration of Very Large Area Integration Using Laser Restructuring," *Proc. IEEE Intl. Symp. on Circuits and Systems*, Newport Beach, May 1983.

6. C.W. Therrien, "On The Relation Between Triangular Matrix Decomposition and Linear Prediction," *Proc. IEEE* 71, No. 2 (December 1983).

7. C. Mead and L. Conway, *Introduction to VLSI Systems*, Section 8.3 (Addison-Wesley, Reading, Massachusetts, 1980).

8. U. Weiser and A. Davis, "A Wavefront Notational Tool for VLSI Array Design," *Proc. CMU Conference on VLSI Systems and Computations*, H.T. Kung, B. Sproull, and G. Steele, Editors, Computer Science Press, 1981.

9. H.T. Kung and M.S. Lam, "Fault-Tolerant VLSI Systolic Arrays and Level Pipelining," *Proc. SPIE*, Society of Photo-Optical Instrumentation Engineers, (to be published in 1983).

10. W.M. Gentleman and H.T. Kung, "Matrix Triangularization by Systolic Arrays," *Proc. SPIE*, 298, Real Time Signal Processing IV, Society of Photo-Optical Instrumentation Engineers, 1981.

11. S.L. Garverick and E.A. Pierce, "A Single Wafer 16-point 16 MHz FFT Processor," *IEEE 1983 Custom Integrated Circuits Conf.*, Rochester, New York, 23-25 May 1983.

REPORT DOCUMENTATION PAGE		READ INSTRUCTIONS BEFORE COMPLETING FORM
1. REPORT NUMBER ESD-TR-83-224	2. GOVT ACCESSION NO. AD-A141 273	3. RECIPIENT'S CATALOG NUMBER
4. TITLE (and Subtitle) Multi-Dimensional Signal Processing Research Program		5. TYPE OF REPORT & PERIOD COVERED Semiannual Technical Summary 1 April — 30 September 1983
		6. PERFORMING ORG. REPORT NUMBER
7. AUTHOR(s) Dan E. Dudgeon and Charles W. Therrien		8. CONTRACT OR GRANT NUMBER(s) F19628-80-C-0002
9. PERFORMING ORGANIZATION NAME AND ADDRESS Lincoln Laboratory, M.I.T. P.O. Box 73 Lexington, MA 02173-0073		10. PROGRAM ELEMENT, PROJECT, TASK AREA & WORK UNIT NUMBERS Program Element No. 62702F Project No. 4594
11. CONTROLLING OFFICE NAME AND ADDRESS Rome Air Development Center Griffiss AFB, NY 13440		12. REPORT DATE 30 September 1983
		13. NUMBER OF PAGES 52
14. MONITORING AGENCY NAME & ADDRESS (if different from Controlling Office) Electronic Systems Division Hanscom AFB, MA 01731		15. SECURITY CLASS. (of this report) Unclassified
		15a. DECLASSIFICATION DOWNGRADING SCHEDULE

16. DISTRIBUTION STATEMENT (of this Report)

Approved for public release; distribution unlimited.

17. DISTRIBUTION STATEMENT (of the abstract entered in Block 20, if different from Report)

18. SUPPLEMENTARY NOTES

None

19. KEY WORDS (Continue on reverse side if necessary and identify by block number)

target detection   image processing architectures

20. ABSTRACT (Continue on reverse side if necessary and identify by block number)

This Semiannual Technical Summary covers the period 1 April through 30 September 1983. It describes the significant results of the Lincoln Laboratory Multi-Dimensional Signal Processing Research Program, sponsored by the Rome Air Development Center, in the areas of multiprocessor architectures for image processing and algorithms for object detection and region classification in aerial reconnaissance imagery.