Report KSC-TR-88-006

FINAL TECHNICAL PROGRESS REPORT

Kendall Preston, Jr.
Kensal Consulting
Building 36
5701 E. Glenn Street
Tucson, AZ 85712

15 March 1989

Progress Report for Sixth Month
N00014-88-C-0717

Distribution:

Addressee	Code	Number of Copies
Scientific Officer	N0014	1
Administrative Contracting Officer	S0302A	1
Director, Naval Research Laboratory ATTN: Code 2627 Washington, D. C. 20375	N00173	1
Defense Technical Information Center Building 5, Cameron Station Alexandria, VA 22314	S47031	12
Strategic Defense Initiative Organization ATTN: T/IS The Pentagon Washington, D. C. 20301-7100	SDIO84	1

Prepared for
OFFICE OF NAVAL RESEARCH
Department of the Navy
800 North Quincy Street
Arlington, VA 22217

Scientific Officer
Office of Naval Research
Attn: Dr. Keith Bromley
800 North Quincy Street
Arlington, VA 22217-5000

UNCLASSIFIED

SECURITY CLASSIFICATION OF THIS PAGE

REPORT DOCUMENTATION PAGE

1a. REPORT SECURITY CLASSIFICATION Unclassified	1b. RESTRICTIVE MARKINGS
2a. SECURITY CLASSIFICATION AUTHORITY	3. DISTRIBUTION / AVAILABILITY OF REPORT Unlimited
2b. DECLASSIFICATION / DOWNGRADING SCHEDULE	
4. PERFORMING ORGANIZATION REPORT NUMBER(S) KSC-TR-88-006	5. MONITORING ORGANIZATION REPORT NUMBER(S) 0001AF
6a. NAME OF PERFORMING ORGANIZATION Kensal Consulting 6b. OFFICE SYMBOL (If applicable) 0D9C9	7a. NAME OF MONITORING ORGANIZATION Office of Naval Research
6c. ADDRESS (City, State, and ZIP Code) Building 36, 5701 E. Glenn St. Tucson, AZ 85712	7b. ADDRESS (City, State, and ZIP Code) 800 North Quincy St. Arlington, VA 22217-5000
8a. NAME OF FUNDING / SPONSORING ORGANIZATION 8b. OFFICE SYMBOL (If applicable)	9. PROCUREMENT INSTRUMENT IDENTIFICATION NUMBER N00014-88-C-0717
8c. ADDRESS (City, State, and ZIP Code)	10. SOURCE OF FUNDING NUMBERS PROGRAM ELEMENT NO.: SDIO-W; PROJECT NO.: None; TASK NO.: None; WORK UNIT ACCESSION NO.: None
11. TITLE (Include Security Classification) Final Technical Progress Report	
12. PERSONAL AUTHOR(S) Kendall Preston Jr.	
13a. TYPE OF REPORT Final 13b. TIME COVERED FROM 88OCT01 TO 89FEB28	14. DATE OF REPORT (Year, Month, Day) 89MAR15 15. PAGE COUNT 69
16. SUPPLEMENTARY NOTATION	

17. COSATI CODES		
FIELD	GROUP	SUB-GROUP
17	11	None
17	05	01

18. SUBJECT TERMS (Continue on reverse if necessary and identify by block number)
Target Detection, Cellular Automaton, Infrared, Signal Processing

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

This project on subpixel target detection relates to research in the optimization of three-dimensional computing structures for use in target detection and to research in the reduction of an optimum computing to an efficiently-designed silicon chip. The work reported here has led to this final report which is divided into three sections: (1) presentation of the mathematical analysis which treats multiply-redundant massively-parallel processors for target detection, (2) a report on VLSI implementation of the target-detection processor as presented by our subcontractor Visual Information Technologies (Plano, Texas), and (3) a description of the Macintosh-based three-dimensional, target-detection software emulator designed during the course of this study for use in Phase II to predict target miss rates and target false alarm rates using three-dimensional logical processing.

20. DISTRIBUTION / AVAILABILITY OF ABSTRACT ☒ UNCLASSIFIED/UNLIMITED ☐ SAME AS RPT ☐ DTIC USERS	21. ABSTRACT SECURITY CLASSIFICATION Unclassified
22a. NAME OF RESPONSIBLE INDIVIDUAL Keith Bromley	22b. TELEPHONE (Include Area Code) 619-553-2535 22c. OFFICE SYMBOL 7601T

DD FORM 1473, 84 MAR 83 APR edition may be used until exhausted. All other editions are obsolete.

SECURITY CLASSIFICATION OF THIS PAGE
UNCLASSIFIED

FINAL TECHNICAL PROGRESS REPORT

(TECHNICAL REPORT NUMBER 6)

Topic Number: SDIO 88-10

Title: Three Dimensional Cellular Automata for Subpixel Target Detection

Contract Number: N00014-88-C-0717

From: Kensal Consulting, Tucson, Arizona (Code: 0D9C9)

To: Dr. Keith Bromley, NOSC, San Diego (Code: 7601T)

PROJECT DESCRIPTION

This project on subpixel target detection relates to research in the optimization of three-dimensional computing structures for use in target detection and to research in the reduction of an optimum computing structure to an efficiently designed silicon chip.

SUMMARY

This final report is divided into three sections. Section 1 presents the mathematical analysis which treats multiple-redundancy, massively-parallel, processors for target detection. Both two-dimensional and three-dimensional cases are studied using the architecture patented by the author (U.S. Patent 4,641,351). Section 2 contains the report on VLSI implementation of the target-detection processor as presented by our subcontractor Visual Information Technology (Plano, Texas). Section 3 furnishes a description of the MacIntosh-based, three-dimensional, target-detection emulator designed during the course of this research for use in Phase II to predict target miss rates and target false alarm rates using three-dimensional logical processing.

SECTION 1 - TECHNICAL PROGRESS

During February research continued furthering the analysis first reported in Technical Progress Report Numbers 1, 4, and 5. This work focused on the mathematical optimization of device design assuming a planar structure for executing cellular logic transforms. The optimization criterion is the maximization of pixops (picture point operations) per device. As reported in Technical Progress Report Number 4, it has been discovered that using a variable on-chip data window size improves performance. Optima are now clearly defined and a final analysis of two-dimensional (planar) structures has been completed.

Furthermore, the analysis has now been extended to the three-dimensional structure required to process simultaneously in two spatial coordinates and time (x,y,t). The results are dramatic. There are well-defined optimum designs for both the two-dimensional and the three-dimensional processor which had not heretofore been recognized. The results of this research will have a major impact assuming that prototyping of devices for cellular logic transforms is funded in response to our Phase II Proposal (to be prepared in April). Results are given below.

Two-Dimensional Case

Assume that one-bit pixels in a square data field of span S must be processed using a neighborhood processing device with a small on-chip data memory. This device is assumed to receive B pixels (bits) each clock cycle t_0 from the data field. It is also assumed to contain B LUTs (LookUp Tables) capable of flash processing all B pixels (bits) simultaneously and returning them to the data field in another, non-overlapping clock cycle t_0. Finally, it is assumed that there are multiple, redundant on-chip data windows each of which is capable of storing M pixels (bits). It is assumed that M can be configured as a matrix with an arbitrary number of rows R and columns C = M/R.

The purpose of the following analysis is to determine the minimum processing time for a specific value of M and to apply this determination to an analysis which finds the value of M which yields the maximum number of pixels processed per unit time per on-chip transistor. From this the best processing device design can be

found.

The time t_1 to fill M is given by

$$t_1 = Mt_0/B \tag{1}$$

In this time every B-bit entity must have available the eight other B-Bit entities that form the neighborhoods of all B bits. For example, if B=1, then at least eight other one-bit entities must be stored in M to form a single 3x3 neighborhood. In this case R=3, C=3, and M=9. Conversely for B=16 (halfword transfer) at least eight other 16-bit entities must be stored in M so that B bits may be processed simultaneously. This yields a minimum value of M=144, R=3, and C=48. Thus, in general the number of pixels processed per round-trip cycle through M are given by

$$N = (C-2B)(R-2) = (M/R-2B)(R-2) \tag{2}$$

The processed values of these pixels are loaded in a time t_2 given by

$$t_2 = (M/R-2B)(R-2)t_0/B \tag{3}$$

Adding equations (1) and (3) provides the round trip processing time as

$$\begin{aligned} t_3 &= t_1+t_2 \\ &= [M+(M/R-2B)(R-2)]t_0/B \end{aligned} \tag{4}$$

The number of round-trip cycles required is simply S^2/N so that the total processing time T is given by

$$\begin{aligned} T &= t_3S^2/N \\ &= S^2[M+(M/R-2B)(R-2)]t_0/B(M/R-2B)(R-2) \\ &= S^2Mt_0/B(M/R-2B)(R-2) + S^2t_0/B \end{aligned} \tag{5}$$

For a given design, M, B, and t_0 are fixed and, for a given processing task, S is fixed. Under these assumptions the second term in equation (5) is a constant as is the numerator in the first term. Thus, to minimize T, for a given task and design,

the denominator of the first term in equation (5) must be maximized. This denominator may be expressed as a function of R given by

$$f(R)= BM-2B^2R-2BM/R+4B^2 \tag{6}$$

Differentiating with respect to R and setting the result equal to 0 yields

$$2M/R^2-2B = 0 \tag{7}$$

Solving for R yields

$$R = (M/B)^{1/2} \tag{8}$$

Substituting this result in (5) yields the minimum processing time as follows

$$T_{min}(M,B,S,t_0) = S^2Mt_0/((M/B)^{1/2}-2B)^2 + S^2t_0/B \tag{9}$$

Letting S = 512 and t_0 = 100ns, equation (9) is plotted in Figures 1-4 as a function of M for B=1, 4, 8, 16, respectively, corresponding to bit, nibble, byte, and halfword entities. These figures also show the row-column configuration of M for both the minimum M and for the optimum M as determined below. Examination of these figures shows that the processing time for the optimum configuration of M (equation (8)) falls rapidly only over a relatively small range of M and reaches an asymtotic value given by $2S^2t_0/B$. This indicates that halfword (B = 16) flash processors using large values of M, such as the PHP of Carnegie-Mellon (M = 65536) are wasteful of transistors. As shown in Figure 4, little is gained in such designs by increasing M much beyond 1024. Also note that, for M = 1024, the optimum configuration of M as given in equation (8) is highly rectangular, namely, 8 rows by 128 columns.

Now consider optimization in terms of the number of pixops (pixel operations) per unit time per transistor. Using triple redundancy in M for maximum speed, the total number of transistors is given by

$$D = 18M + BL \tag{10}$$

where L is the number of transistors per LUT. It is assumed that M is a static

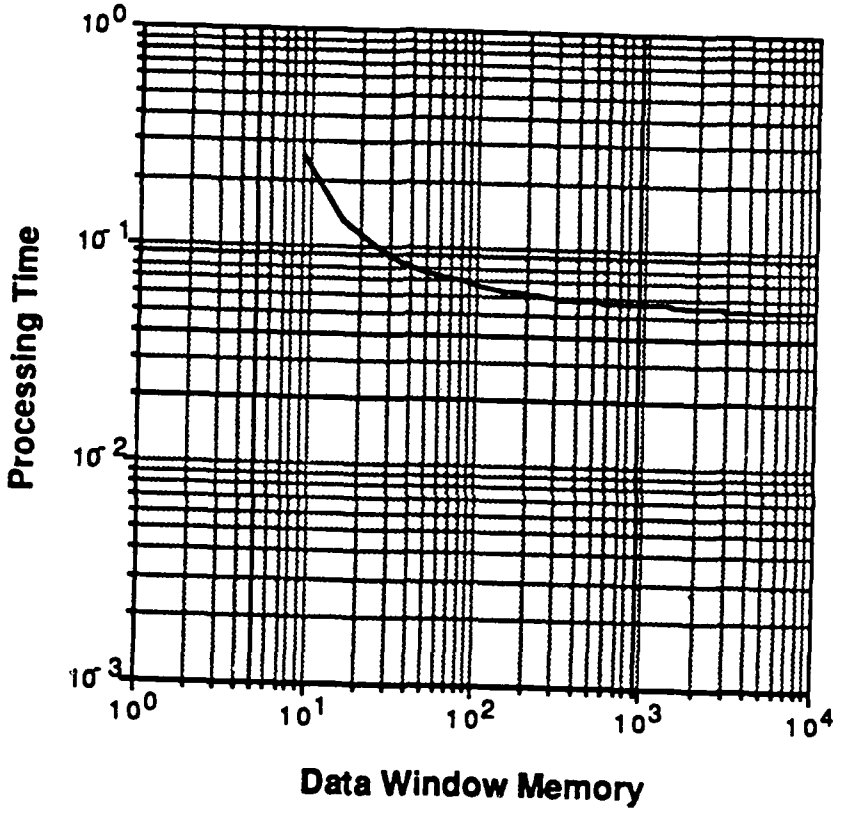
Processing Time
10^0
10^{-1}
10^{-2}
10^{-3}
10^0
10^1
10^2
10^3
10^4
Data Window Memory

Fig. 1

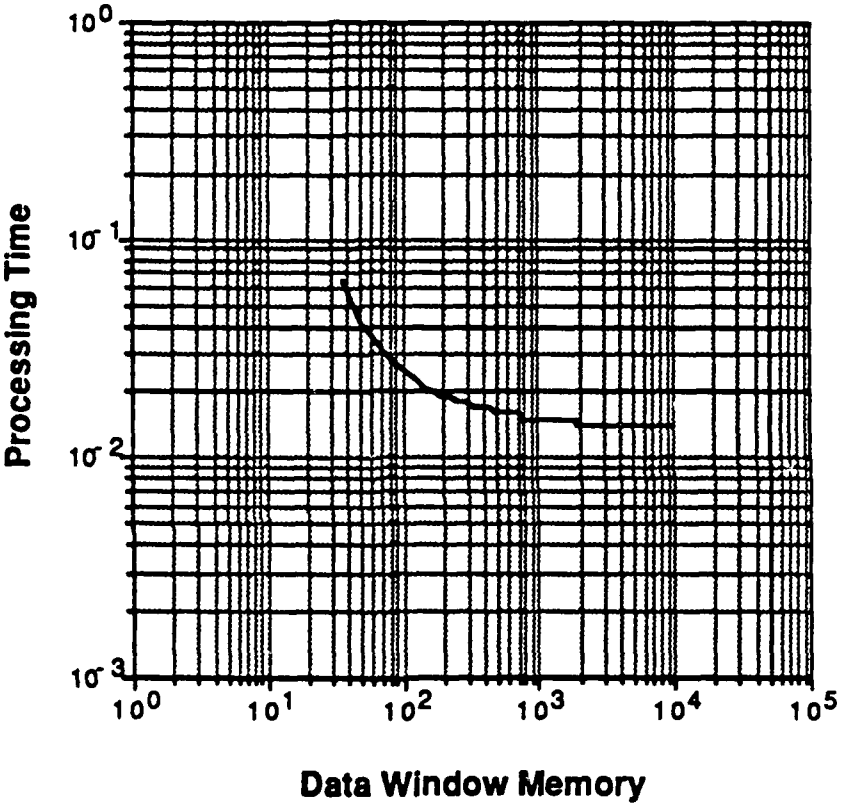
Processing Time
10^0
10^{-1}
10^{-2}
10^{-3}
10^0
10^1
10^2
10^3
10^4
10^5
Data Window Memory

Fig. 2

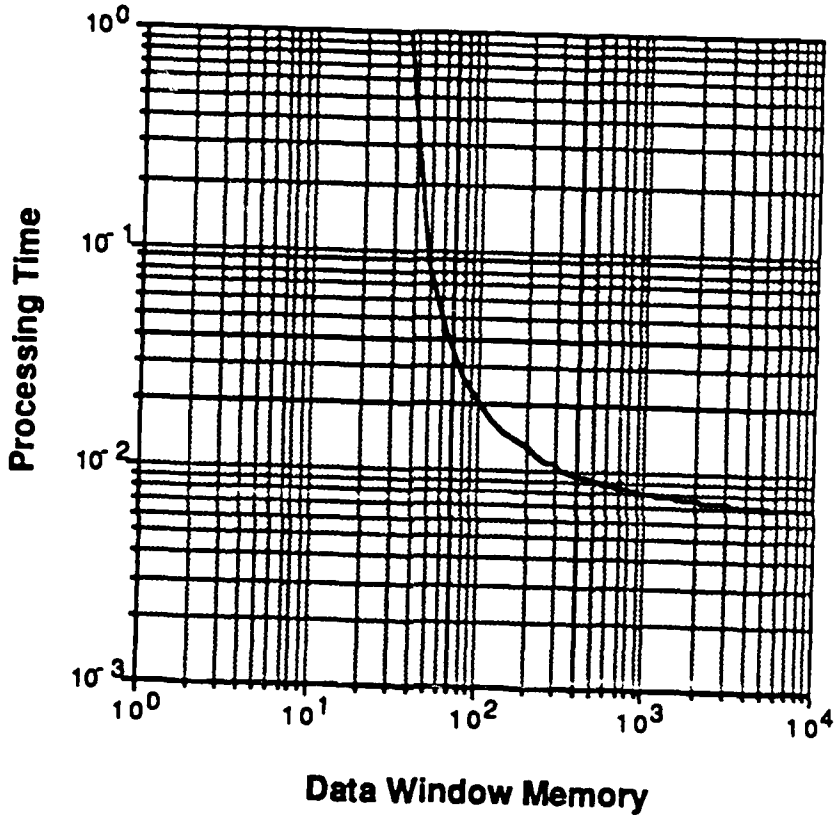
Processing Time
10^0
10^{-1}
10^{-2}
10^{-3}
10^0
10^1
10^2
10^3
10^4
Data Window Memory

Fig. 3

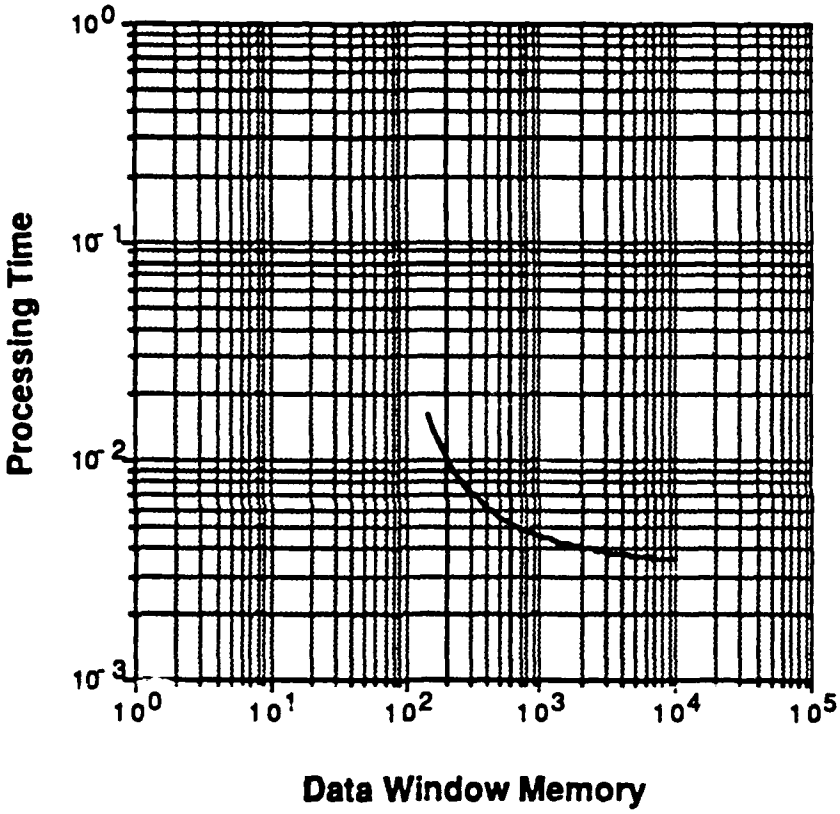
Processing Time
10^0
10^{-1}
10^{-2}
10^{-3}
10^0
10^1
10^2
10^3
10^4
10^5
Data Window Memory

Fig. 4

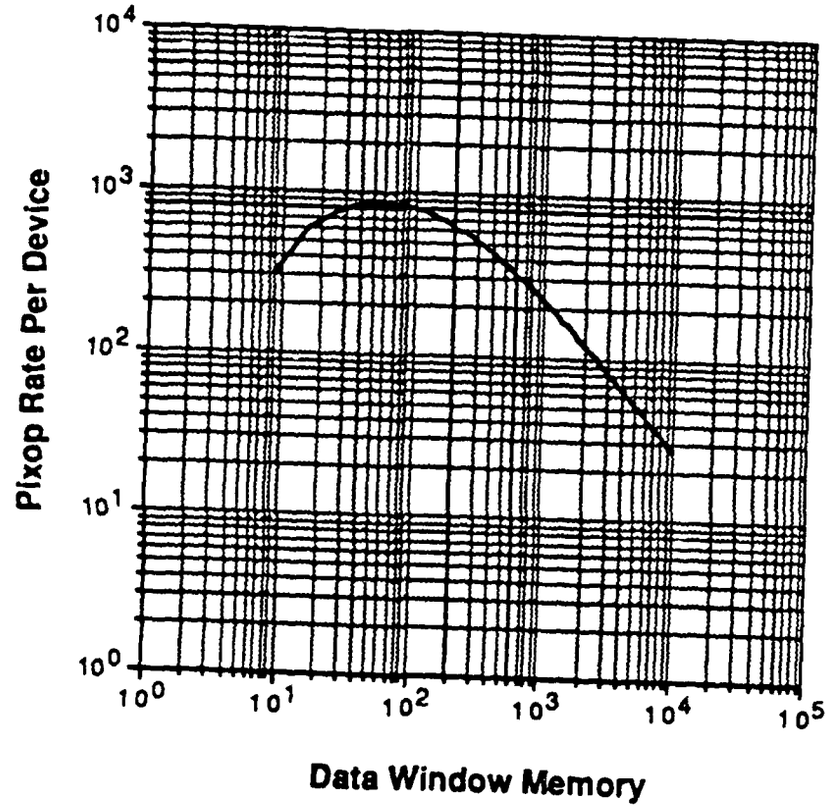
Pixop Rate Per Device
10^4
10^3
10^2
10^1
10^0
10^0
10^1
10^2
10^3
10^4
10^5
Data Window Memory

Fig. 5

RAM using six transistors per memory cell. For the 3x3 neighborhood flash processor considered here L = 6x512 = 3072. Pixops per unit time per transistor are given by

$$E = S^2/TD \tag{11}$$

Combining equations (9), (10), and (11) yields

$$E = ((M/B)^{1/2}-2B)^2/(18M+3072B)Mt_0 + B/(18M+3072B)t_0 \tag{12}$$

Equation (12) is plotted in Figures 5-8 for B=1, 4, 8, 16, respectively, using t_0 = 100ns. These plots correspond to Figures 1-4 and furnish a graphical solution to the maximization of equation (12) as a function of M. Values of all critical parameters for these four optimum cases are given in the table below.

B	R	C	M	T(ms)	D	E
1	7	7	49	77.0	3954	861
4	7	32	224	18.9	16320	850
8	7	64	448	8.6	32640	934
16	7	128	896	4.3	65280	934

Note that these design optima are independent of the span of the data field S. This again points up the fallacy that large values of M are required when processing large data fields. In fact, as M increases beyond the optima given above, efficiency falls, i.e., transistors are being wasted.

Three-Dimensional Case

Here we assume that one-bit voxels in a cubic data field of span S must be processed using a neighborhood processing device with a small on-chip data memory. This device is assumed to receive B voxels (bits) each clock cycle t_0 from the data

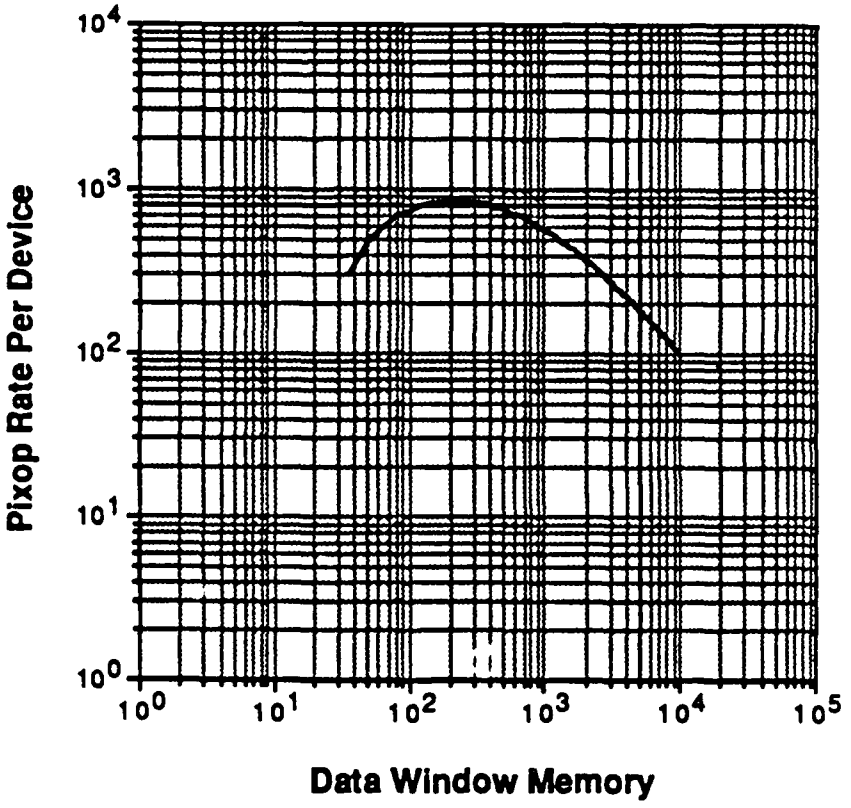
Pixop Rate Per Device
10^4
10^3
10^2
10^1
10^0
10^0
10^1
10^2
10^3
10^4
10^5
Data Window Memory

Fig. 6

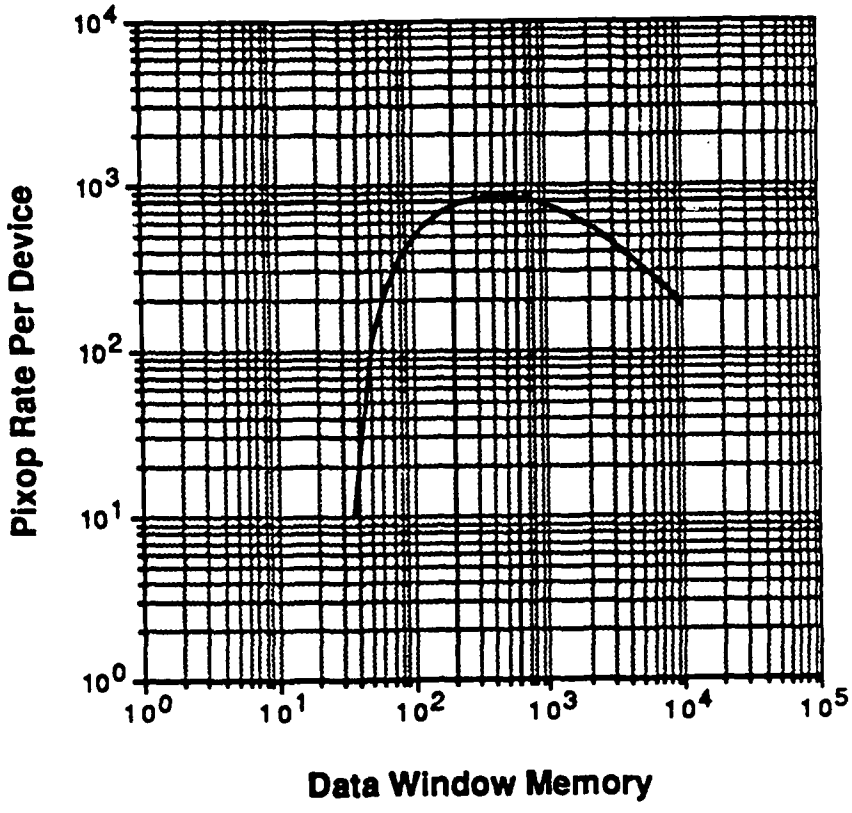
Pixop Rate Per Device
10^4
10^3
10^2
10^1
10^0
10^0
10^1
10^2
10^3
10^4
10^5
Data Window Memory

Fig. 7

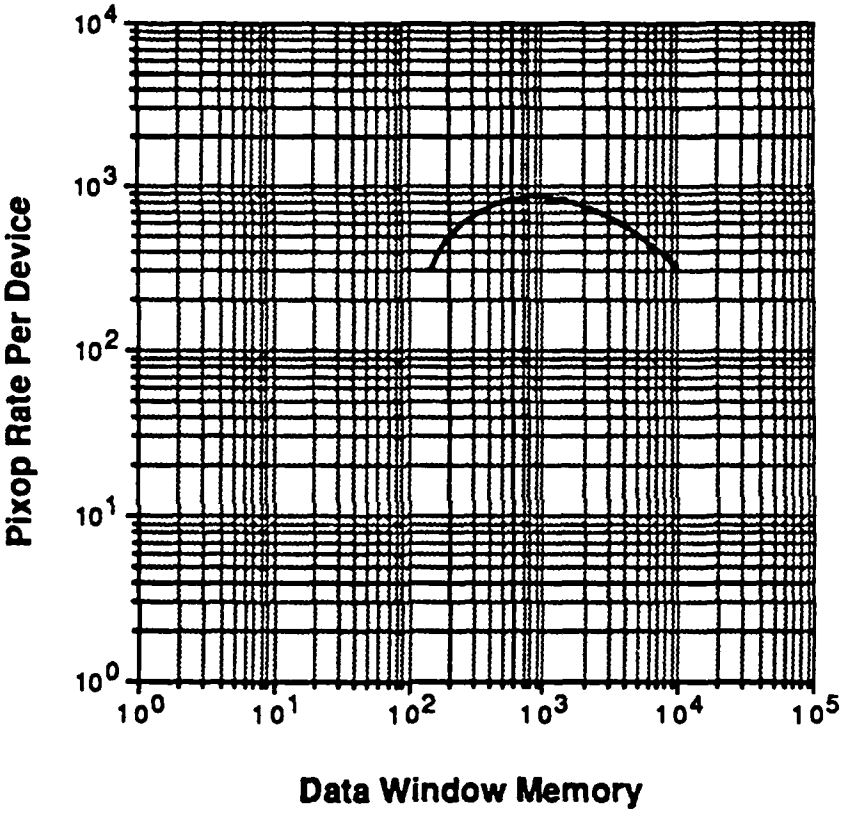
Pixop Rate Per Device
10^4
10^3
10^2
10^1
10^0
10^0
10^1
10^2
10^3
10^4
10^5
Data Window Memory

Fig. 8

field. It is also assumed to contain B LUTs (LookUp Tables) capable of flash processing all B voxels (bits) simultaneously, returning them to the data field in another, non-overlapping clock cycle t_0. It is assumed that there are multiple, redundant on-chip data windows each of which is capable of storing M voxels (bits). Finally, it is assumed that M can be configured as a matrix with an arbitrary number of rows R, columns C, and planes P where C = M/PR.

The purpose of the following analysis is to determine the minimum processing time for a specific value of M and to apply this determination to an analysis which finds the value of M which yields the maximum number of pixels processed per unit time per on-chip transistor. From this the best processing device design can be found.

The time t_1 to fill M is given by

$$t_1 = Mt_0/B \tag{13}$$

In this time every B-bit entity must have available the 13 other B-bit entities that form the neighborhoods of all 13 bits in the dodecahedral tessellation. For example, if B=1, then at least 13 other one-bit entities must be stored in M to form a single 13 voxel neighborhood. In this case R=3, C=3, P=3 and M=27. Conversely for B=16 (halfword transfer) at least 13 other 16-bit entities must be stored in M so that B bits may be processed simultaneously. This yields a minimum value of M=432 with R=3, and C=48, and P=3. Thus, in general, the number of pixels processed per round-trip cycle through M are given by

$$N = (C-2B)(R-2)(P-2) \tag{14}$$

The processed values of these pixels are loaded in a time t_2 given by

$$t_2 = (M/RP-2B)(R-2)(P-2)t_0/B \tag{15}$$

Adding equations (1) and (3) provides the round trip processing time as

$$\begin{aligned} t_3 &= t_1+t_2 \\ &= [M+(M/RP-2B)(R-2)(P-2)]t_0/B \end{aligned} \tag{16}$$

The number of round-trip cycles required is simply S^3/N so that the total processing time T is given by

$$T = t_3 S^3/N$$

$$= S^3[M+(M/RP-2B)(R-2)(P-2)]t_0/B(M/RP-2B)(R-2)(P-2)$$

$$= S^3 M t_0/B(M/RP-2B)(R-2)(P-2) + S^3 t_0/B \qquad (17)$$

For a given design, M, B, and t_0 are fixed. For a given processing task, S is fixed. Under these assumptions the second term in equation (17) is a constant as is the numerator in the first term. Since it is obvious that there is symmetry between R and P, these variables may each be set equal to a new variable Q yielding for f(Q)

$$f(Q) = B(M/Q^2-2B)(Q-2)^2$$

$$= MB-2(BQ)^2-4MB/Q+8B^2Q+4MB/Q^2-8B^2 \qquad (18)$$

Differentiating with respect to Q and setting the result equal to 0 yields

$$BQ^4-2BQ^2-MQ-M = 0 \qquad (19)$$

Solving for Q yields

$$Q = (M/B)^{1/3} \qquad (20)$$

Substituting this result in (17) yields the minimum processing time as follows

$$T_{min}(M,B,S,t_0) = S^3 M t_0/B(M^{1/3}B^{2/3}-2B)((M(B)^{1/3}-2)^2 + S^3 t_0/B \qquad (21)$$

Letting S = 64 and t_0 = 100ns, equation (21) is plotted in Figures 9-13 as a function of M for B=1, 4, 8, 16, respectively, corresponding to bit, nibble, byte, and halfword entities. Examination of these figures shows that the processing time for the optimum configuration of M (equation (20)) falls rapidly only over a relatively small range of M and reaches an asymtotic value given by $2S^3 t_0/B$.

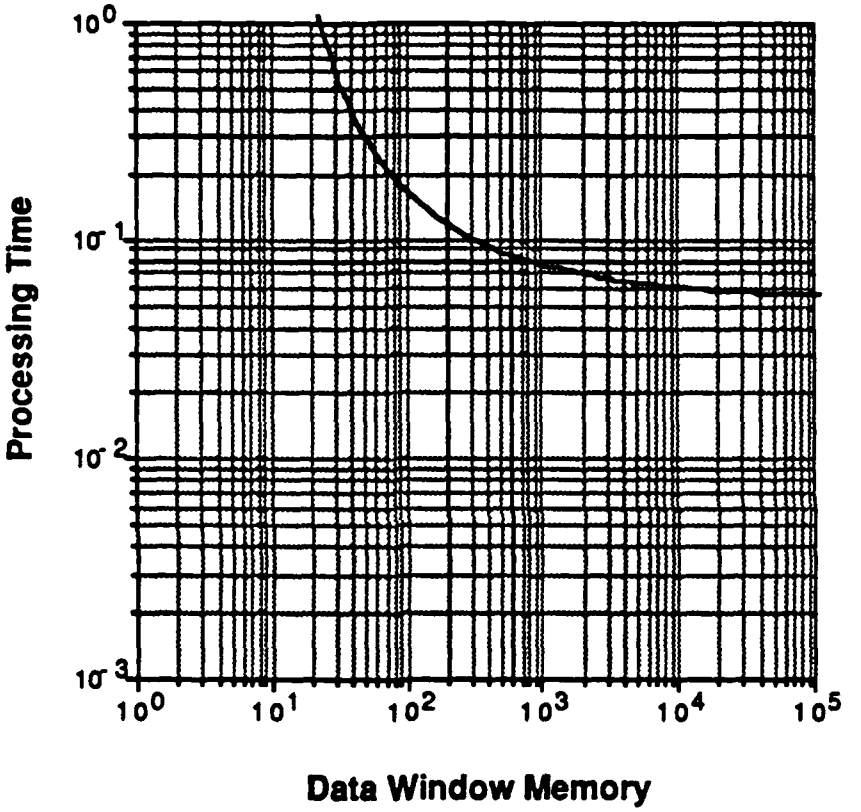
Processing Time
10^0
10^{-1}
10^{-2}
10^{-3}
10^0
10^1
10^2
10^3
10^4
10^5
Data Window Memory

Fig. 9

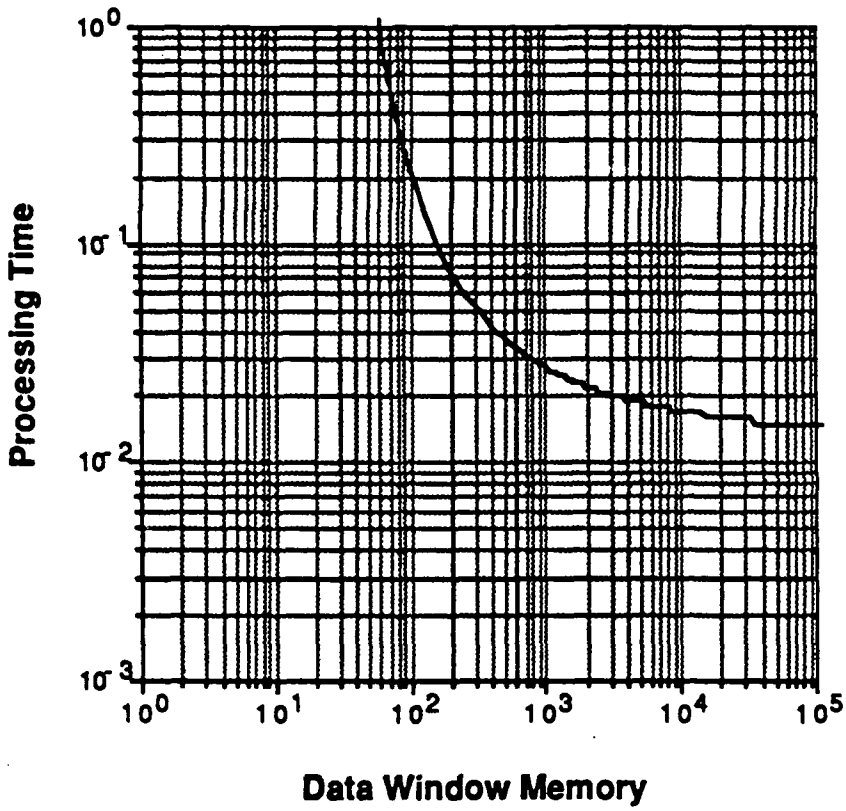
Processing Time
10^0
10^{-1}
10^{-2}
10^{-3}
10^0
10^1
10^2
10^3
10^4
10^5
Data Window Memory

Fig. 10

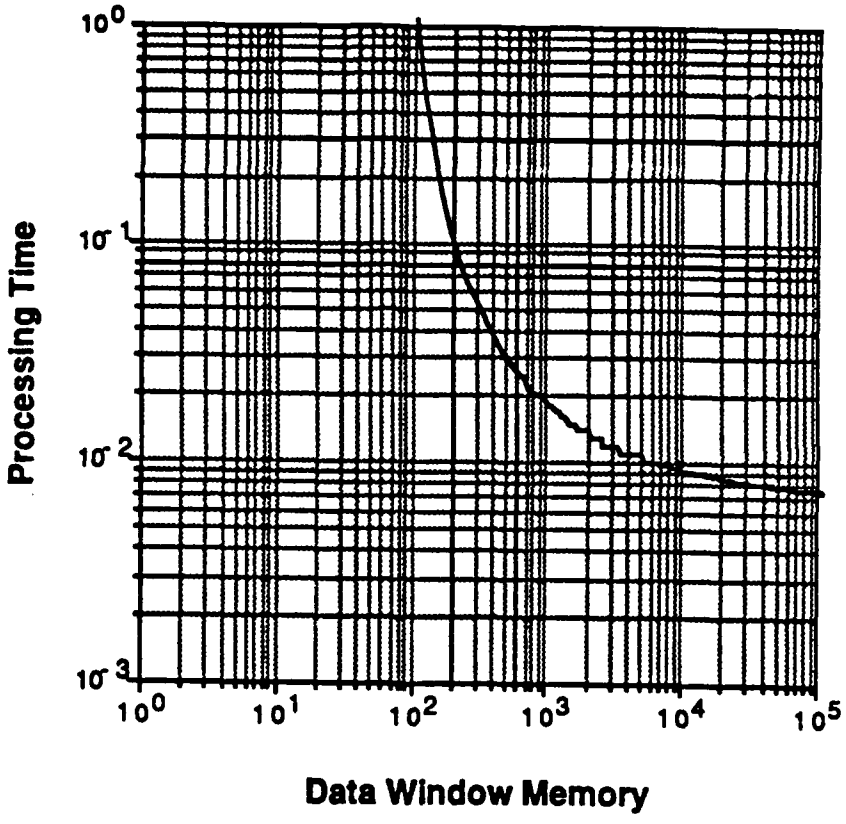
Processing Time
10^0
10^-1
10^-2
10^-3
10^0
10^1
10^2
10^3
10^4
10^5
Data Window Memory

Fig. 11

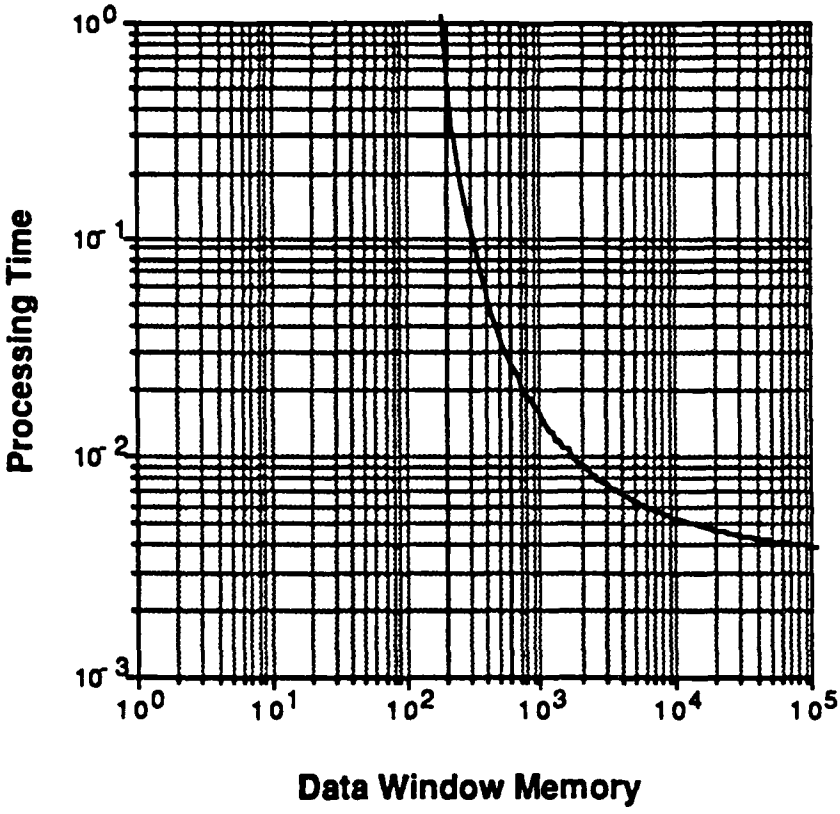
Processing Time
10^0
10^{-1}
10^{-2}
10^{-3}
10^0
10^1
10^2
10^3
10^4
10^5
Data Window Memory

Fig. 12

Now consider optimization in terms of the number of pixops (pixel operations) per unit time per transistor. Using triple redundancy in M for maximum speed, the total number of transistors is given by

$$D = 42M + BL \tag{22}$$

where L is the number of transistors per LUT and it is assumed that M is a static RAM using six transistors required per memory cell. For the 13 voxel neighborhood flash processor considered here L = 6x8192 = 49152. Pixops per unit time per transistor are given by

$$E = S^3/TD \tag{23}$$

Combining equations (21), (22), and (23) yields

$$E = B(M^{1/3}B^{2/3}-2B)((M/B)^{1/3}-2)^2/(42M+49152B)Mt_0 + B/(42M+49152B)t_0 \tag{24}$$

Equation (24) is plotted in Figures 14-17 for B=1, 4, 8, 16, respectively, using t_0 = 100ns. These plots correspond to Figures 9-13 and furnish a graphical solution to the maximization of equation (24) as a function of M. Values of all critical parameters for these four optimum cases are given in the table below.

B	R	C	M	T(ms)	D	E
1	8	8	512	88.4	70656	42
4	8	32	2048	22.1	282624	42
8	8	64	4096	11.0	565248	42
16	8	128	8192	5.5	1130496	42

Conclusions

With the completion of the analysis for both the two-dimensional and three-dimensional cases, it has now been clearly demonstrated that there is an optimum design for the multiple-redundancy, massively-parallel, track-detection

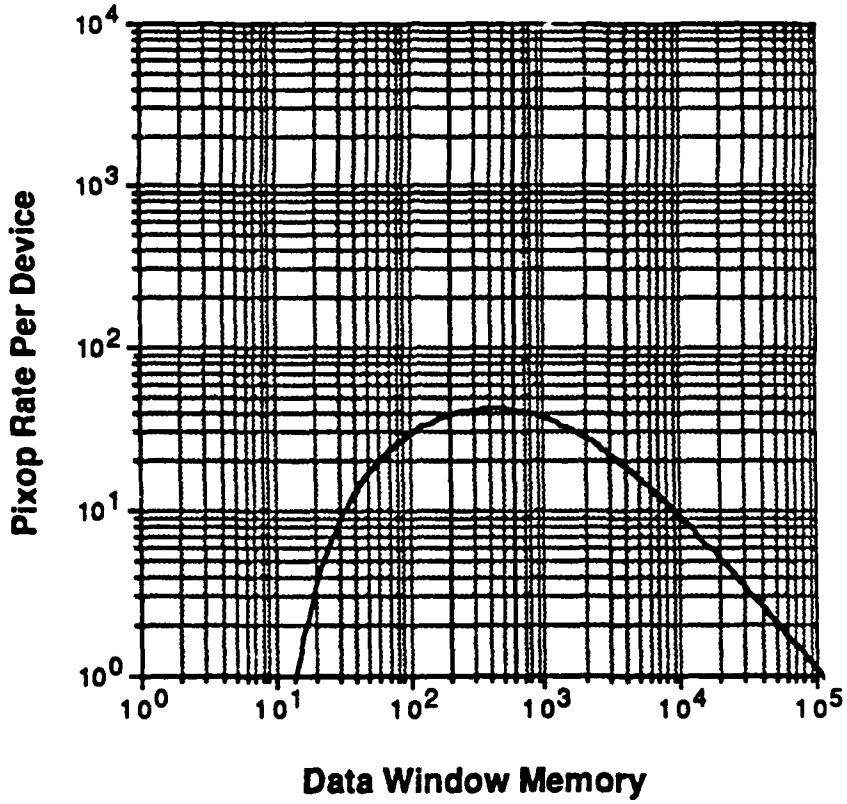
Pixop Rate Per Device
10^4
10^3
10^2
10^1
10^0
10^0
10^1
10^2
10^3
10^4
10^5
Data Window Memory

Fig. 13

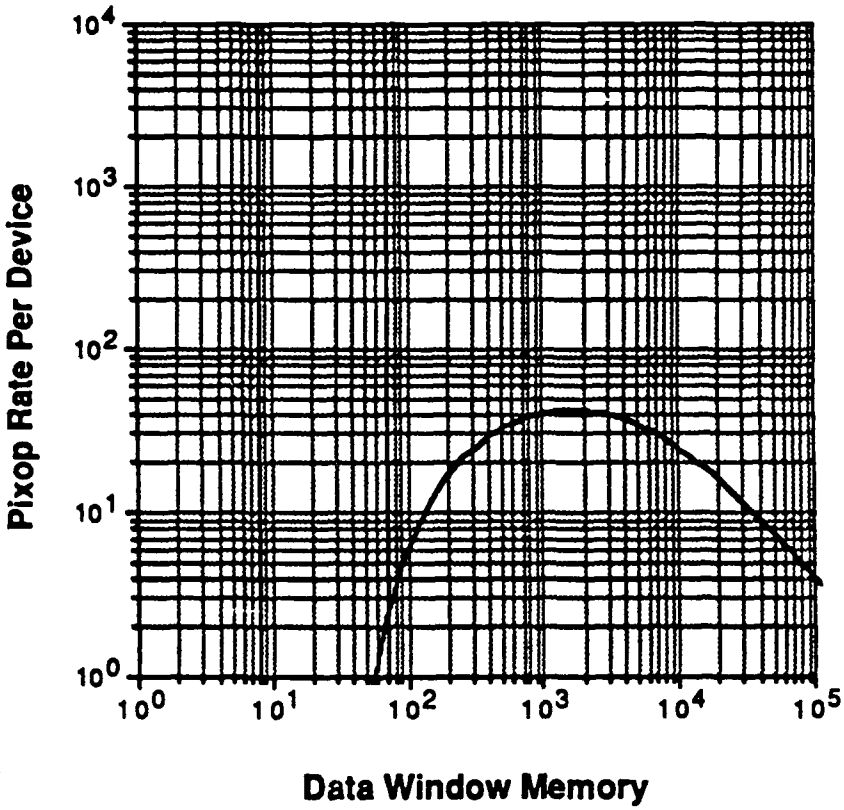
Pixop Rate Per Device
10^4
10^3
10^2
10^1
10^0
10^0
10^1
10^2
10^3
10^4
10^5
Data Window Memory

Fig. 14

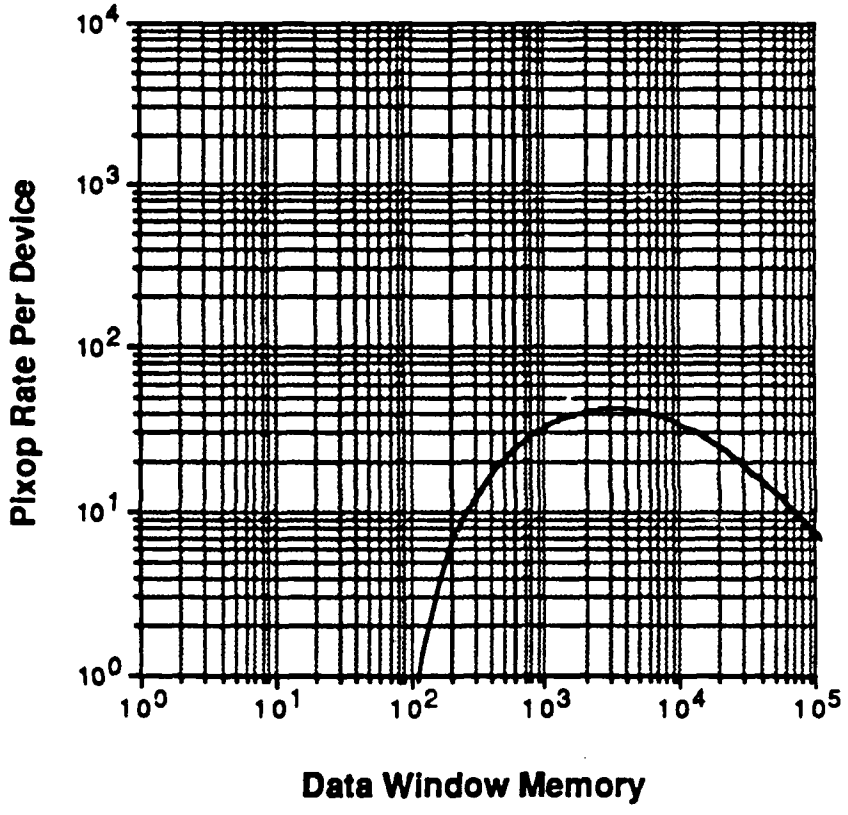
Pixop Rate Per Device
10^4
10^3
10^2
10^1
10^0
10^0
10^1
10^2
10^3
10^4
10^5
Data Window Memory

Fig. 15

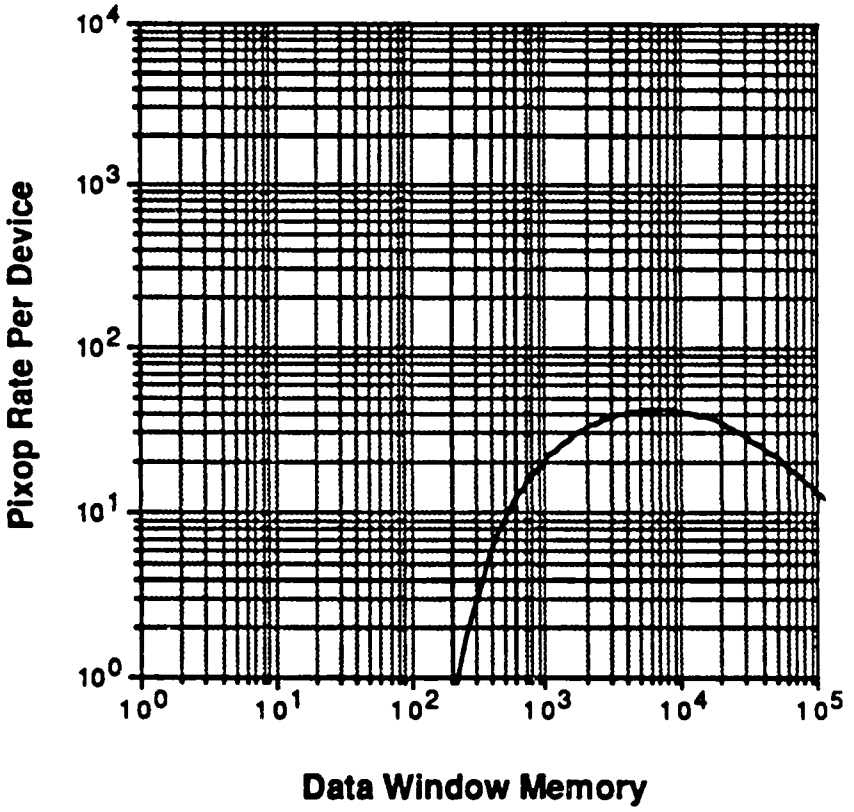
Pixop Rate Per Device
10^4
10^3
10^2
10^1
10^0
10^0
10^1
10^2
10^3
10^4
10^5
Data Window Memory

Fig. 16

processor. This will allow us to proceed in Phase II with the work with Visual Information Technologies in actually fabricating devices knowing that the resultant design will be optimized with respect to the number of operations carried out per unit time per transistor. This is of great importance in a military situation where excess transistors require excess power and contribute to a potential loss in reliability.

The following section, although not taking full advantage of the above analysis, indicates that a somewhat simplified version of the multiple-redundancy, massively-parallel, track-detection device can be reduced to silicon well within the constraints of modern technology. In Phase II, the number of devices will be expanded beyond the number described in the next section to a comfortable limit for this type of device. The possibility of incorporating a portion of the three-dimensional data field itself (SxSxS) in the device will be considered. Also to be considered is the advantage to be gained by recirculation within this data field memory.

SECTION 2

VLSI Implementation

of a

Three-Dimensional Morphology Processor

(contributed by Visual Information Technologies, Plano, Texas)

Final Report

Research Study

VLSI Implementation of a 3D Morphology Processor

Author: John P. Norsworthy
Director, VLSI Development
Visual Information Technologies, Inc.

Introduction

The purpose of this report is to outline progress that has been made in the understanding of a chip implementation of a 3-dimensional image morphology processor. This research is conducted in association with Kensal Consulting concerning a sub-pixel target detection system, and is funded by an SBIR grant from the SDIO. VITec (Visual Information Technologies, Inc.) is uniquely qualified to participate in this research because of its background in VLSI based image processing [1,2]. This report builds upon an earlier submitted progress report.

Background

In this study, pixels are binary variables, having been processed by a thresholding circuit. Furthermore, these voxels are samples from a hexahedrally tessellated space, such that each sample is equidistant from all of its neighbors. This tessellation has a physical analogy to the cubic close-packed crystal lattice [3]. In three dimensions, the hexahedral tessellation leads to a neighborhood of 13 voxels, the outer twelve of which form the vertices of a tetradecahedron, a polyhedra having 14 faces.

Since each pixel is a single-bit value, it can be regarded as a binary variable, which leads to the concept of a "logical transform" of a 3D neighborhood. In the case of the proposed chip, a structure would be built whose inputs are the 13 neighborhood voxels of the hexahedrally tessellated space, and whose single output is an arbitrary logical function of those 13 inputs. This structure directly implements a truth table with 2**13, or 8192, entries, and corresponds to a 8Kx1 RAM (Random Access Memory) which serves as a LUT (LookUp Table). It is clear that the hexahedrally tessellated space is particularly suitable for this form of processing, since a 3D neighborhood of a rectangularly tessellated space would have 27 voxels, requiring a LUT having 2**27 entries.

Architectural Studies

A proposed chip architecture is shown in figures 1, which fits well into a system proposal by Kensal Consulting. In the processor, data is loaded into shift registers, once per cycle. Although figure one shows data being loaded 8b at a time, the exact number of bits loaded at a time is a parameter which needs further study. There are seven shift registers, corresponding to the 7 voxels of a hexagonal neighborhood in a particular frame. Three frames are needed to construct a 13 voxel 3D neighborhood, however the top and bottom 3 voxels in the 3D neighborhood occupy 3 of the 7 bit positions of the hexagon (for example, see [3], page 54). Once these 7 shift registers are loaded, processing may begin. In the 8b example of figures 1, 8 neighborhoods are passed to the single LUT, one at a time for 8 cycles, where the LUT's output fills an output shift register. The shift register's contents are then outputted in parallel to an image memory. There is a 3 cycle latency in getting the pipeline started.

The LUT RAM is organized as 1Kx8 rather than 8Kx1 for two reasons. Firstly, the implementation of the RAM is easier as 1Kx8, since the column decoding of the RAM is simpler. Secondly, data can be written to the RAM in 8-bit groups (in this example), thus utilizing an existing bus to load the RAM (the input data bus).

Chip Implementation Studies

Although the processing bandwidth of the chip is presently undefined, it is likely that a standard ASIC CMOS process will be adequate to implement this chip. Presently, most CMOS integrated circuit design is being conducted using 1μ or less minimum photolithographic design rules, which allows for extremely high packing densities. For example, if one were to examine the 1989 International Solid State Circuit Conference Digest of Technical Papers, he would find that most papers involving digital circuitry use 1μ processes, while simultaneously, chip sizes continue to increase dramatically. For example, a 1 million transistor microprocessor was presented (by Intel) which had a die area of 150 square millimeters. By the time that an implementation of this chip is started, 1μ processing will be commonplace.

CMOS processing is preferred because of its low power dissipation, high speed, proliferation of use, and insensitivity to noise. CMOS performance is typically adequate for all but the most demanding performance requirements. Figure 2 shows a chip floorplan for the architecture of figure 1. The chip size is approximately 2.5x2.1 mm. This is very small by today's standards.

The LUT RAM would be implemented with a standard 6-transistor CMOS static RAM cell. The 6T cell has reduced sensitivity to soft errors because there are no internal high impedance nodes within the cell. A 6T cell has been designed, whose physical dimensions are 13.2μx21.6μ (see figure 3). Because there are 8192 bit cells, the LUT RAM dominates the chip floorplan. There is nothing unusual about this proposed RAM implementation. Existing designs in 1.5μ CMOS are presently operating near 50MHz.

The other large block of circuitry is the input shift registers, whose bit cell is shown in figure 4. The cell is very simple, and arrays nicely. As can be seen in table 1, which gives the transistor counts, the input shift registers contribute only a few transistors to the overall transistor count, and correspondingly, only a small amount of area. This is also evident from the chip floorplan.

Because the die size of this chip is so small, the chip cost would be dominated by the development cost, which would be approximately 150K$. This cost would cover prototyping, design, analysis, test program generation, and production tooling.

Conclusion

Further research should be conducted to improve on the availability of data to the LUT, so that it is always kept busy, perhaps adding complexity to the way data is brought into the proposed chip. Further study should be conducted along the lines of parallel LUTs within the chip.

References

1. Norsworthy et al, 1988 IEEE ISSCC Digest of Technical Papers, pg 158

2. Norsworthy, Proceedings of Electronic Imaging West, pp. 409-412 (1988)

3. Preston and Duff, Modern Cellular Automata, Plenum Press (1984)

Table 1
3D Morphology Chip Estimated Transistor Count

	Tx/Cell	# Cells	Control	Total
Shift Registers	11	77	100	947
RAM	6	8192	800	49,952
Pads	20	35	20	720
Misc	-	-	500	500
Grand Total				**52,119**

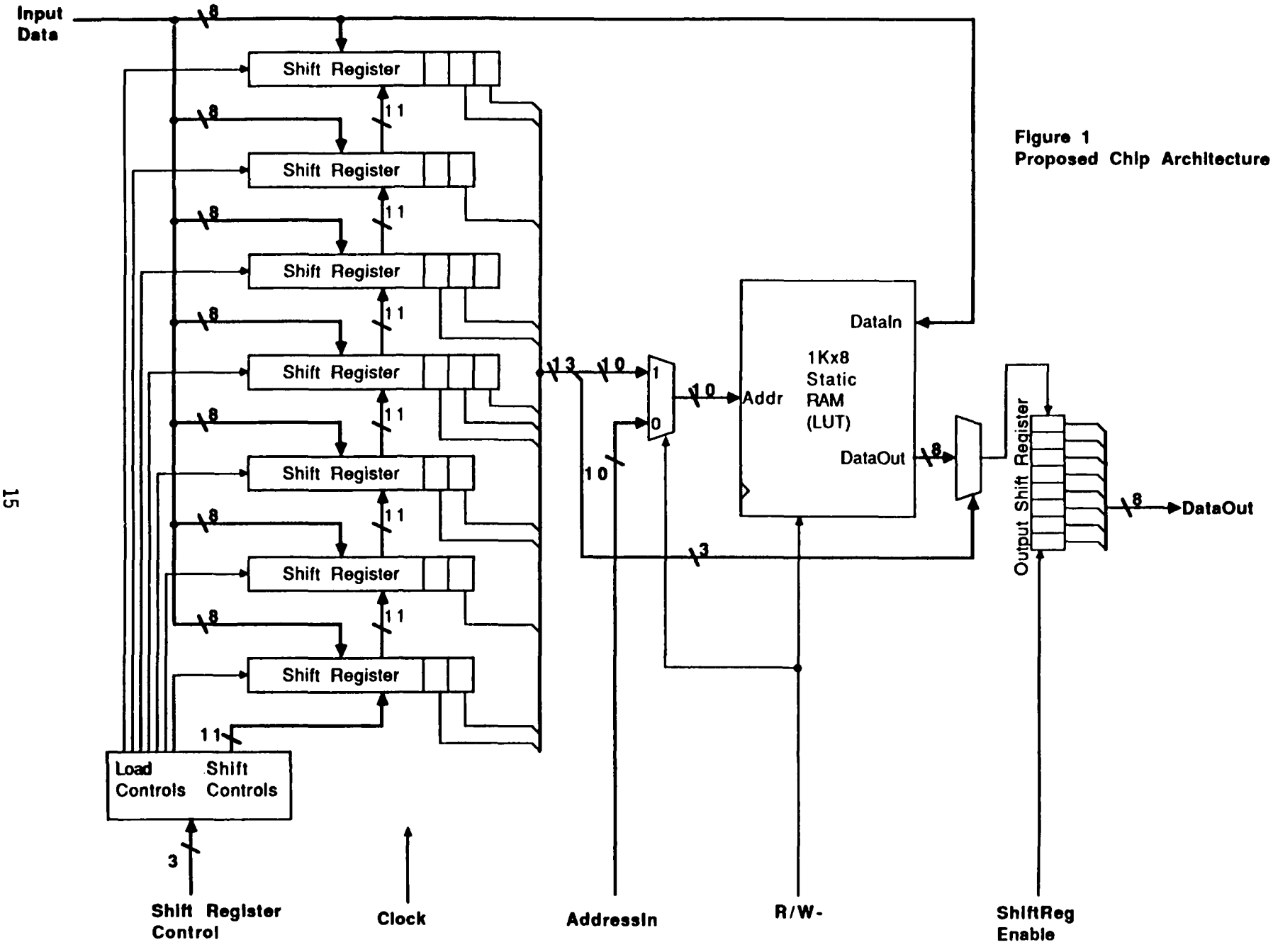

Figure 1
Proposed Chip Architecture

J Norsworthy
3/30/89

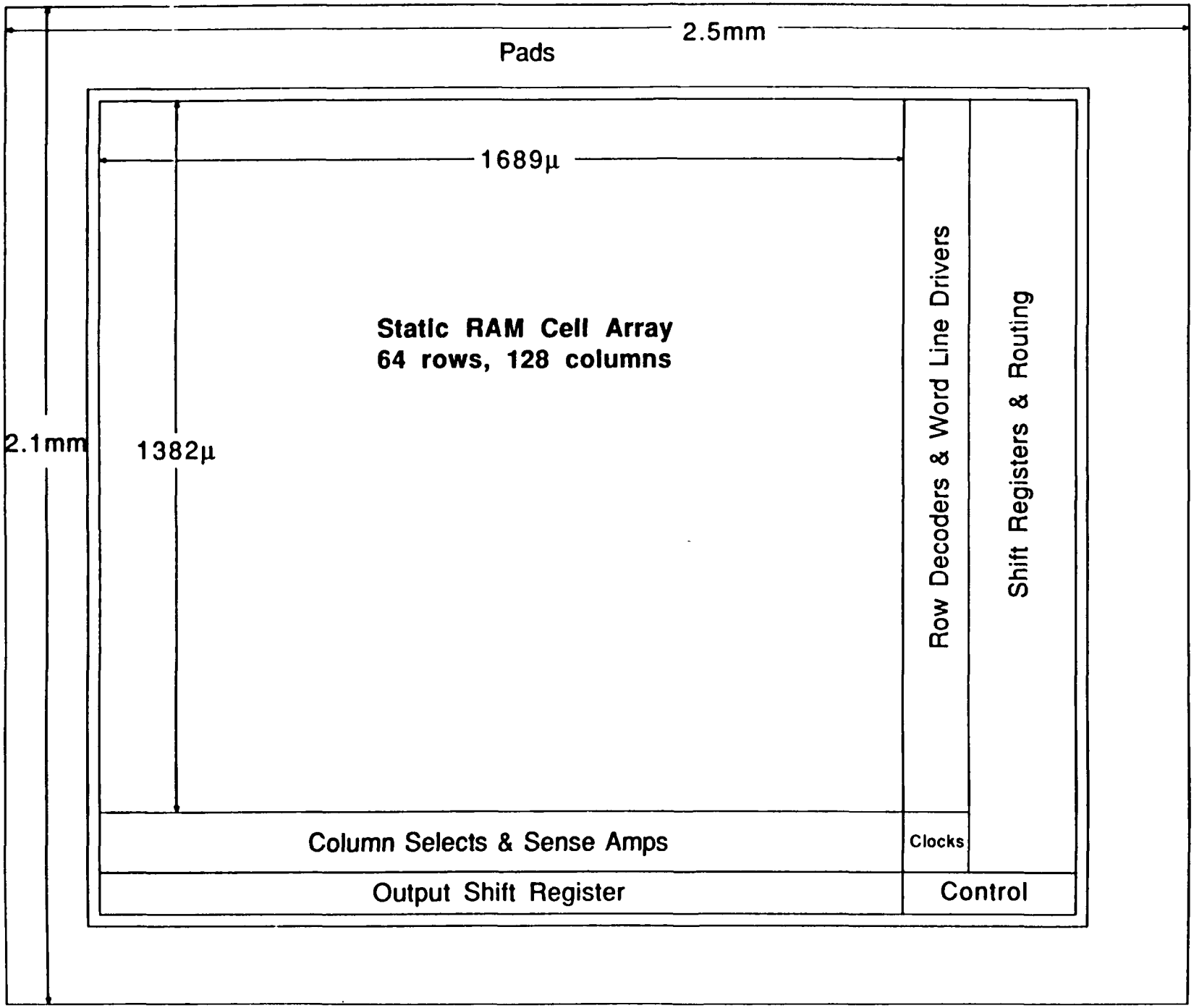

Figure 2
Chip
Floorplan

J Norsworthy
3/30/89

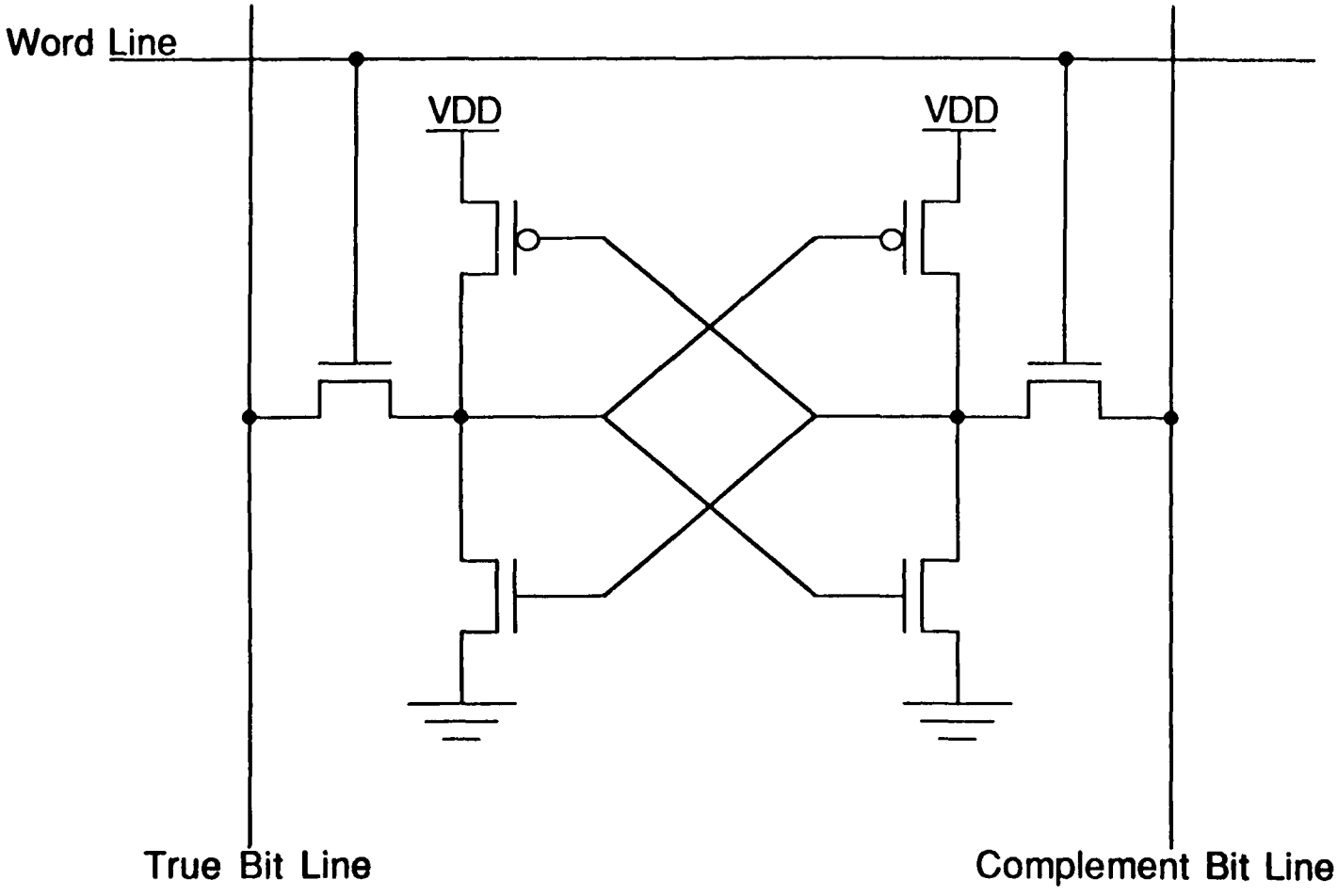

Figure 3, 6 Transistor CMOS static RAM cell
Physical Dimensions = 13.2 x 21.6 sq μ

J Norsworthy
3/30/89

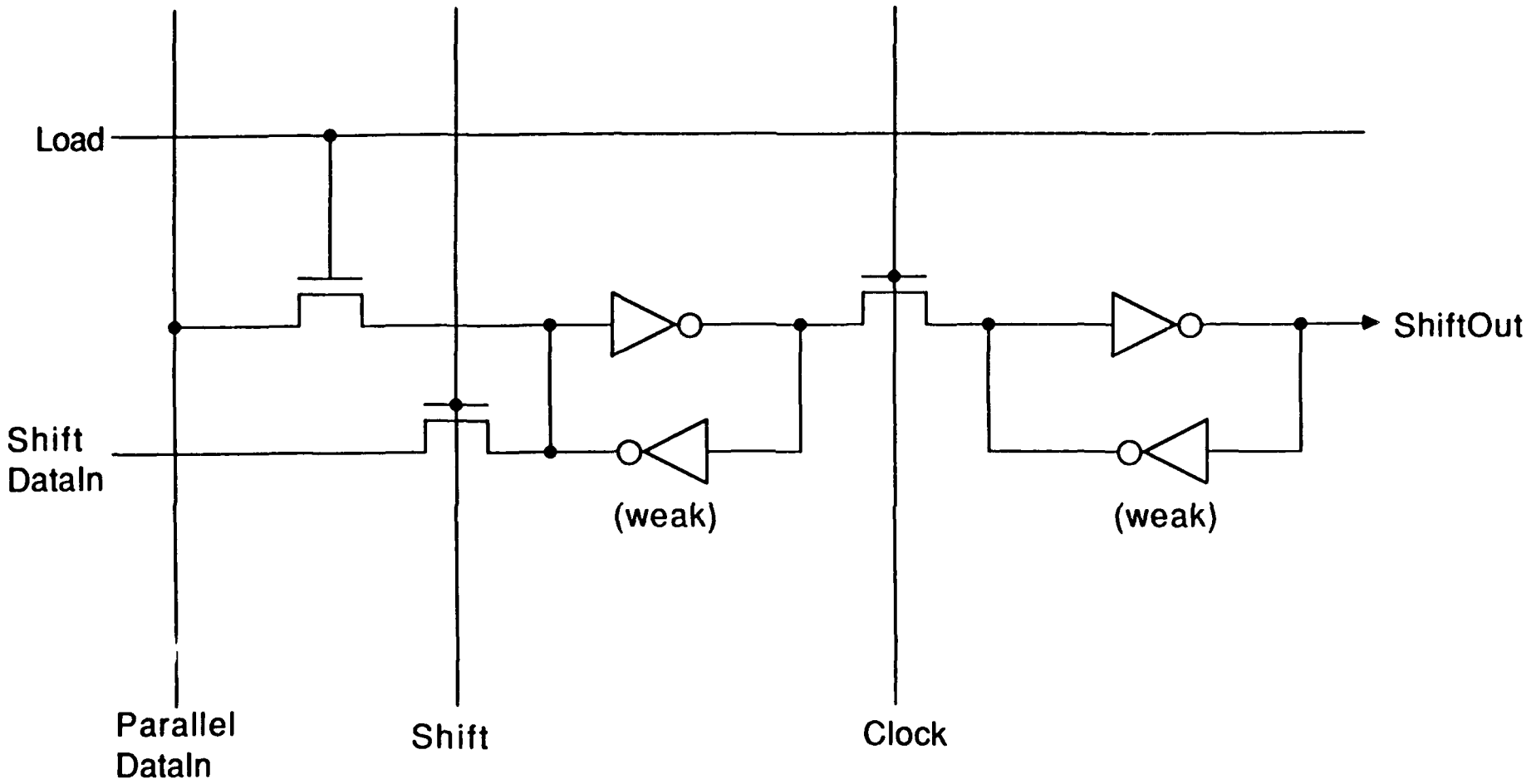

Figure 4,
11 Transistor CMOS static shift register cell
Physical Area = 3168 sq μ

J Norsworthy
3/30/89

SECTION 3

Technical Description

of the

TRIAKIS Emulator

of a

Three-Dimensional Dodecahedral Track Detection Processor

Three Dimensional Cellular Logic Data Analyzer

Kensal Consulting
PROJECT: 88507

General Description of Program:

The purpose of the Three Dimensional Cellular Logic Data Analyzer program is to demonstrate target detection in the tetradekahedral tessellation. For reference see Chapter 3 of Modern Cellular Automata. This program is designed to display and analyze three-dimensional threshold target data in any one of eight 64X64X56 workspaces. Commands are performed in the three-dimension tetradekahedral tessellation where each volume element is considered in conjunction with its twelve neighbors. The results are displayed on the screen and may be printed or stored on disk for later use. Measurements may be made on the contents of the workspace area displayed on the screen and stored for later analysis. The user interface is consistent with the standard for Macintosh™ applications as defined in Inside Macintosh, Volume I, Section 2.

Program Features:

The Three Dimensional Cellular Logic Data Analyzer program provides the following operations:

Disk File Operations - Both input and output of images in TIFF and Workspace format. Workspace format is simply a 64X64X56 bit stream unique to the Three Dimensional Cellular Logic Data Analyzer program. TIFF adds a standard Aldus/Microsoft TIFF header to the bit stream.

Display Operations - Any of the eight workspace areas provided in memory may be displayed on the screen for subsequent processing or printing. An example of a workspace area with the border set is shown in Figure 1.

Set Operations - This group of operations allows the user to set single voxels or groups of voxels such as boundaries, tracks, as well as geometric cubes and spheres. In addition, the workspace area displayed on the screen may filled with random noise or completely erased. All such set operations set voxels in the workspace area displayed on the screen to ones. A voxel which has been set to one is displayed as black, while voxels set at zero are displayed as white.

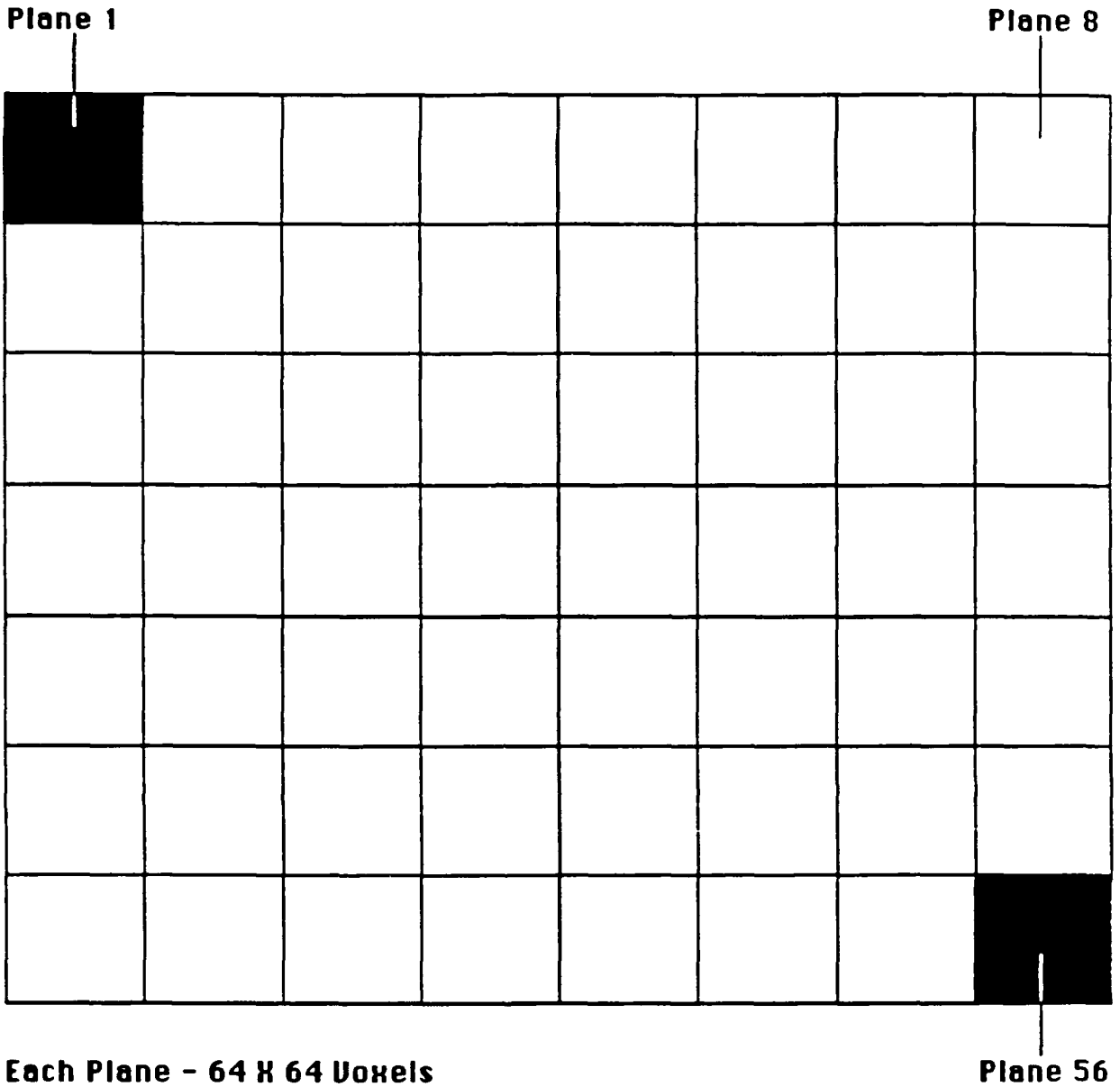

Figure 1

Boolean Operations - The program provides the AND, OR, Exclusive OR, and INVERT boolean operations from any two of the eight available workspaces into a third workspace.

Process Operations - *The AUGRED and XIFILTER operations* are available and act on the workspace which is displayed on the screen.

Copy Workspace - Any workspace area may be copied into any other workspace.

Measurement Operations - The total number of voxels in the displayed workspace may be counted. Measurements may be saved to a file on disk for later analysis.

Program Operation:

The Three Dimensional Cellular Logic Data Analyzer resides on the disk as a standard Macintosh application and is executed in the usual Macintosh fashion. The user may launch the program directly by selecting the program icon and then selecting "Open" from the desk top "File" menu, or by double clicking on program icon. The user may also launch the program indirectly by opening a workspace file, which automatically executes the Three Dimensional Cellular Logic Data Analyzer program and loads the selected workspace file into workspace area 1. The icons for the program and workspace files are shown in Figure 2. Workspace files are automatically given the icon labeled "Target 1" shown in the left of the "Triakis" window, while the program itself is represented by the icon labeled "3-D Data Analyzer" shown in the right of that window.

After the program has loaded, the main menu bar, shown in Figure 3, will be displayed. If the program was launched by selecting a workspace file, that file will be displayed on the screen underneath the main menu bar. With the exception of "Shade", any of the options on the main menu bar may be selected by pressing on them with the mouse. "Shade" is not implemented in this version of the program and has been dimmed to remind you that it is not supported.

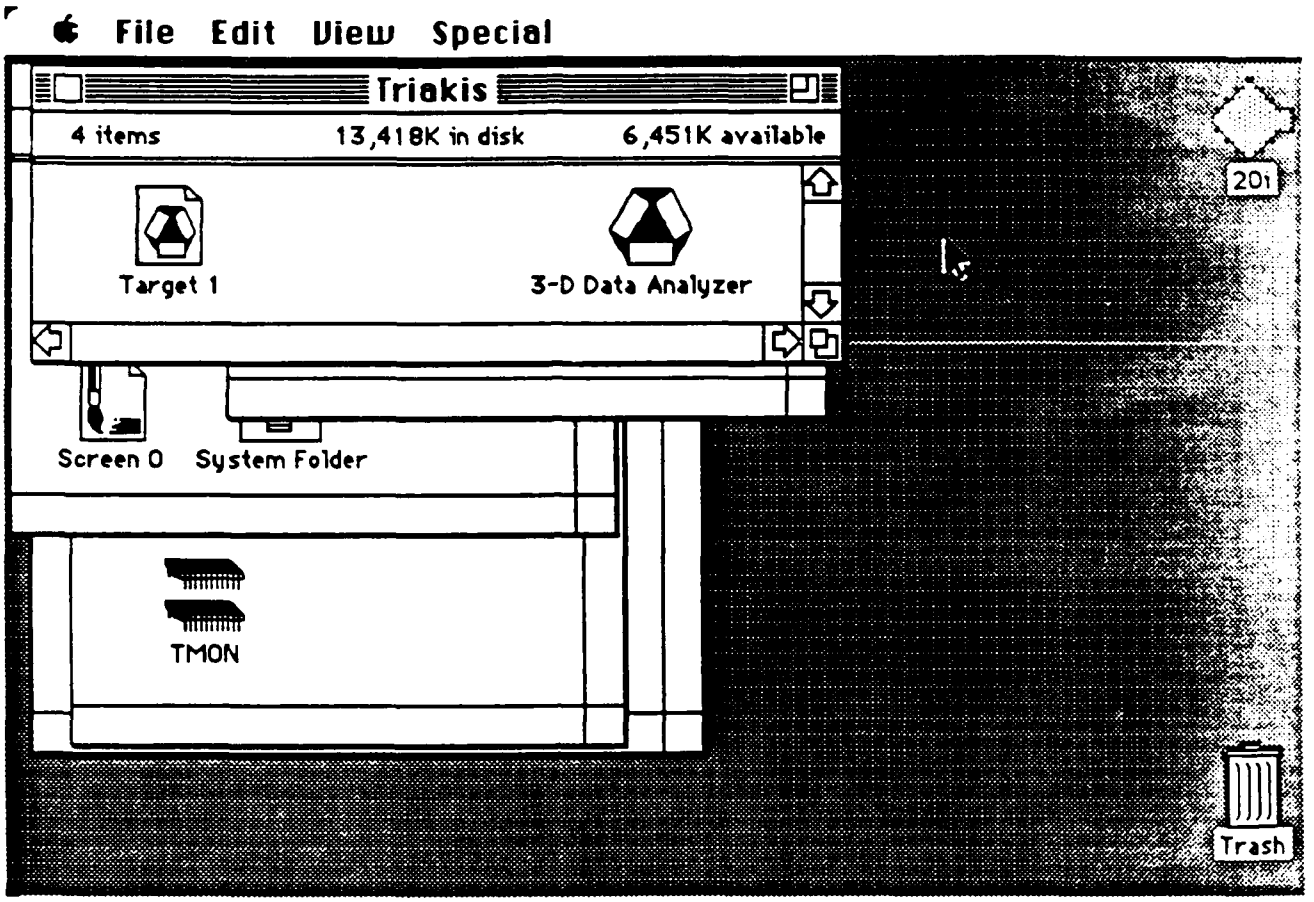

Figure 2

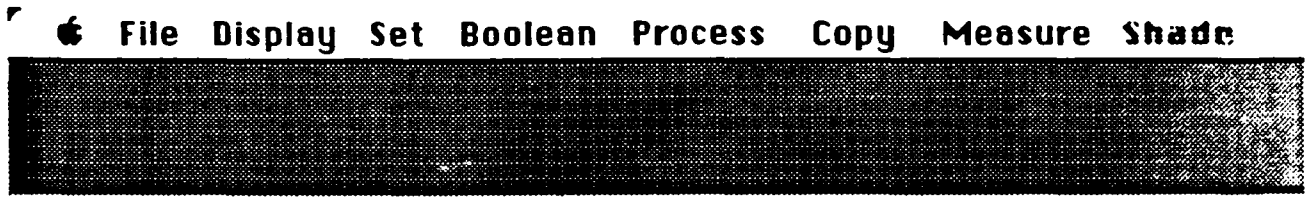

Figure 3

The File Menu: This menu contains all the commands necessary for handling workspace files on disk. Select the File menu by pressing on the word File in the main menu bar. Selecting the file menu produces the menu shown in Figure 4.

The Open Command: The Open command is used to load an existing image file into a workspace. The Open command may be executed by either selecting it from the File menu with the mouse or holding down the command key (⌘) while pressing O on the keyboard. When the Open command is selected, the selection box shown in Figure 5 appears. Select the workspace area you wish to load the image into. Next, select the folder where the image is located by using the file list and current folder boxes to traverse the disk directories. To see the files in a folder listed in the file list, double click on it with the mouse. To see the files in a folder which contains the current folder, press the current folder name (located directly above the files list) and then drag the highlight bar to the appropriate folder. To see the folders and files on another disk drive, click on the "Drive" button. To eject the floppy disk currently mounted in the drive so that another may be inserted, click on the "Eject" button. After you have selected the appropriate folder and the file you wish to load is displayed in the files list, select the file by double clicking on it. Once the selected image has been loaded, it will automatically be displayed on the screen.

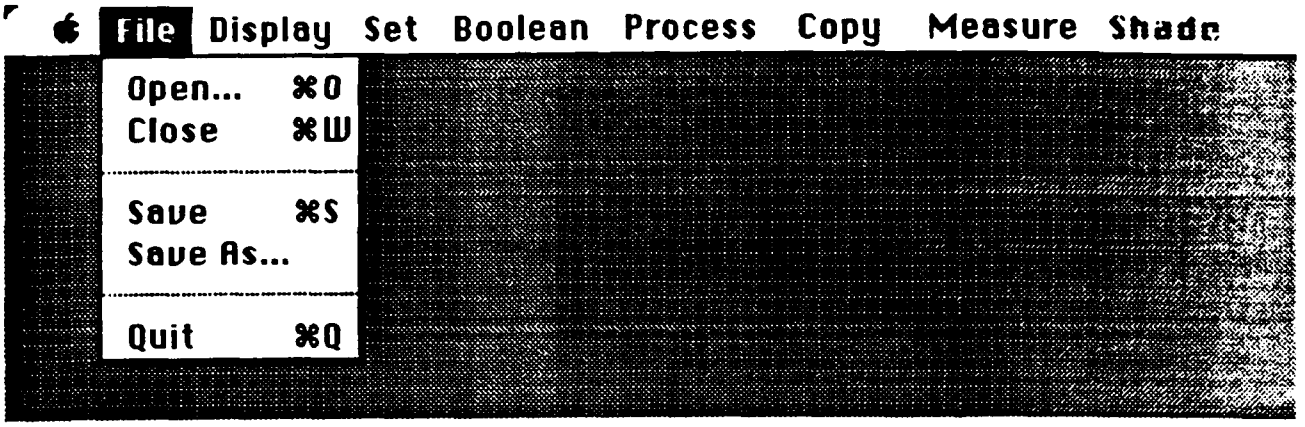

Figure 4

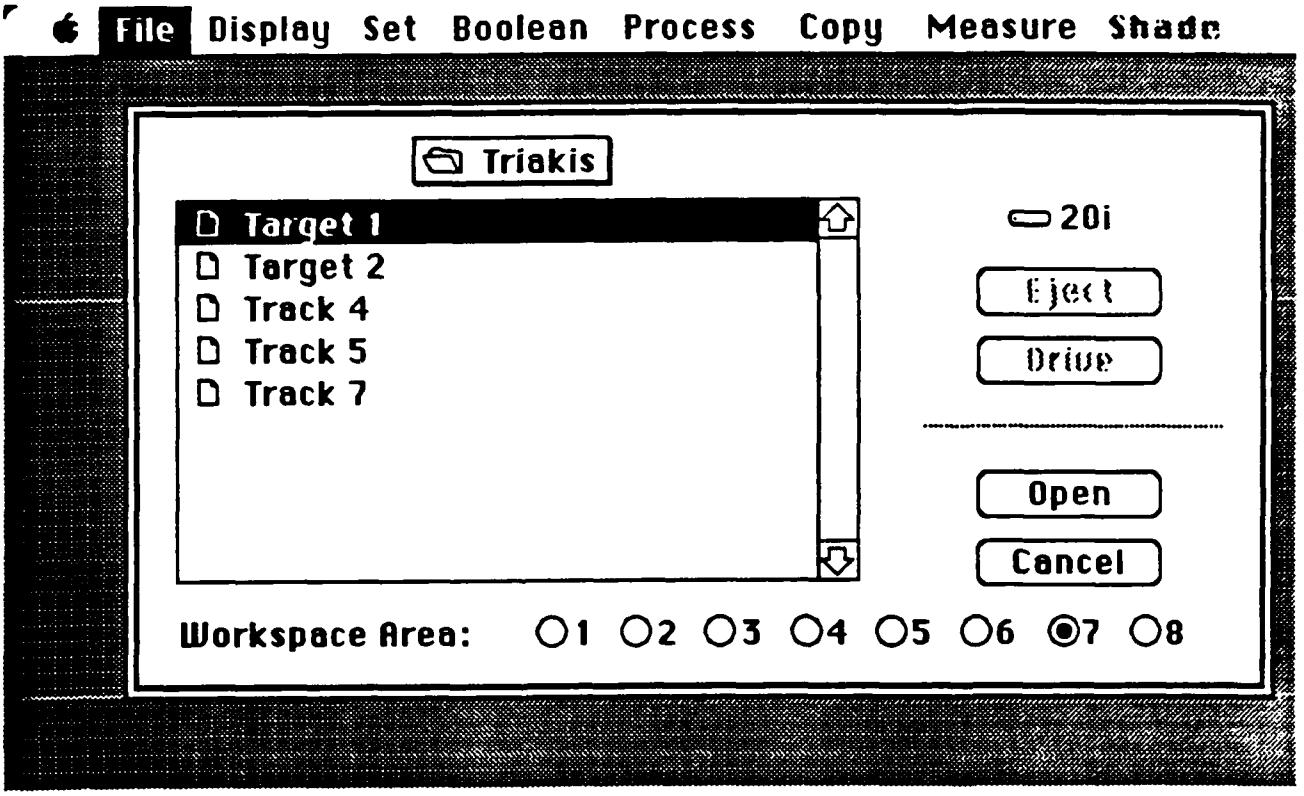

Figure 5

The Close Command: The Close command is used to close a workspace area without saving it. The Close command may be executed by either selecting it from the File menu with the mouse or holding down the command key (⌘) while pressing W on the keyboard. When the Close command is selected, the workspace area displayed on the screen along with its associated file information is cleared and control is immediately returned to the mouse unless changes have been made to the workspace since it was last saved. If you have not saved changes to the workspace you are attempting to close, you will be presented with the warning box shown in Figure 6. Pressing the return key or clicking on the "Yes" button will save the workspace before closing it in the same manner as the Save command described below. Clicking on the "No" button will close the workspace area without saving it, and clicking on the "Cancel" button will cancel the Close operation and return control to the mouse.

The Save Command: The Save command is used to save a workspace area to disk. The Save command may be executed by either selecting it from the File menu with the mouse or holding down the command key (⌘) while pressing S on the keyboard. When the Save command is selected, the workspace are displayed on the screen is saved to disk and control is immediately returned to the mouse unless the workspace area has not been assigned a file name. If the workspace does not have a file name, you will be presented with the "Save As"dialog box shown in Figure 7. Proceed according to the instructions for the "Save As" dialog box described with the "Save As" command below.

The Save As Command: The Save As command is used to save a workspace area which has not previously been given a file name or for which a new file name is desired. The Save As command may be executed by selecting it from the File menu with the mouse. When the Save As command is executed, you will be presented with the "Save As" dialog box shown in Figure 7. Select the format in which you wish to save the file by clicking on the "Workspace" or "TIFF" button. Then, select the appropriate folder for you file in the same manner as the was described with the Open command. Next, type in the appropriate file name. The file names of all existing files in the folder are shown in the dim font to remind you of which file names have already been used. When you are satisfied with the name, location, and format of the file, press return or click on the "Save" button to complete the operation. You may cancel the Save As operation by clicking on the "Cancel" button.

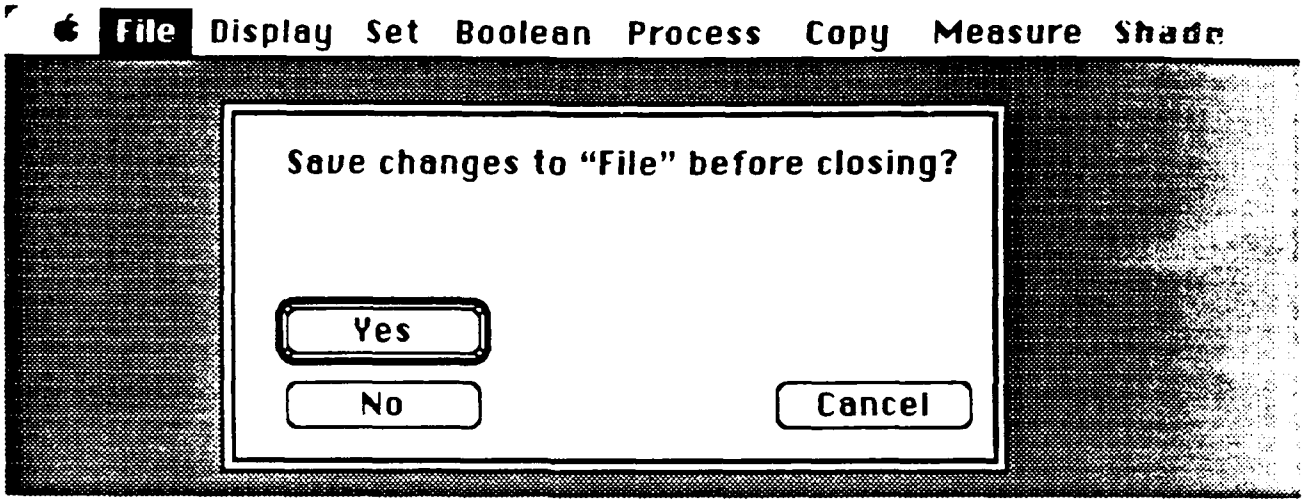

Figure 6

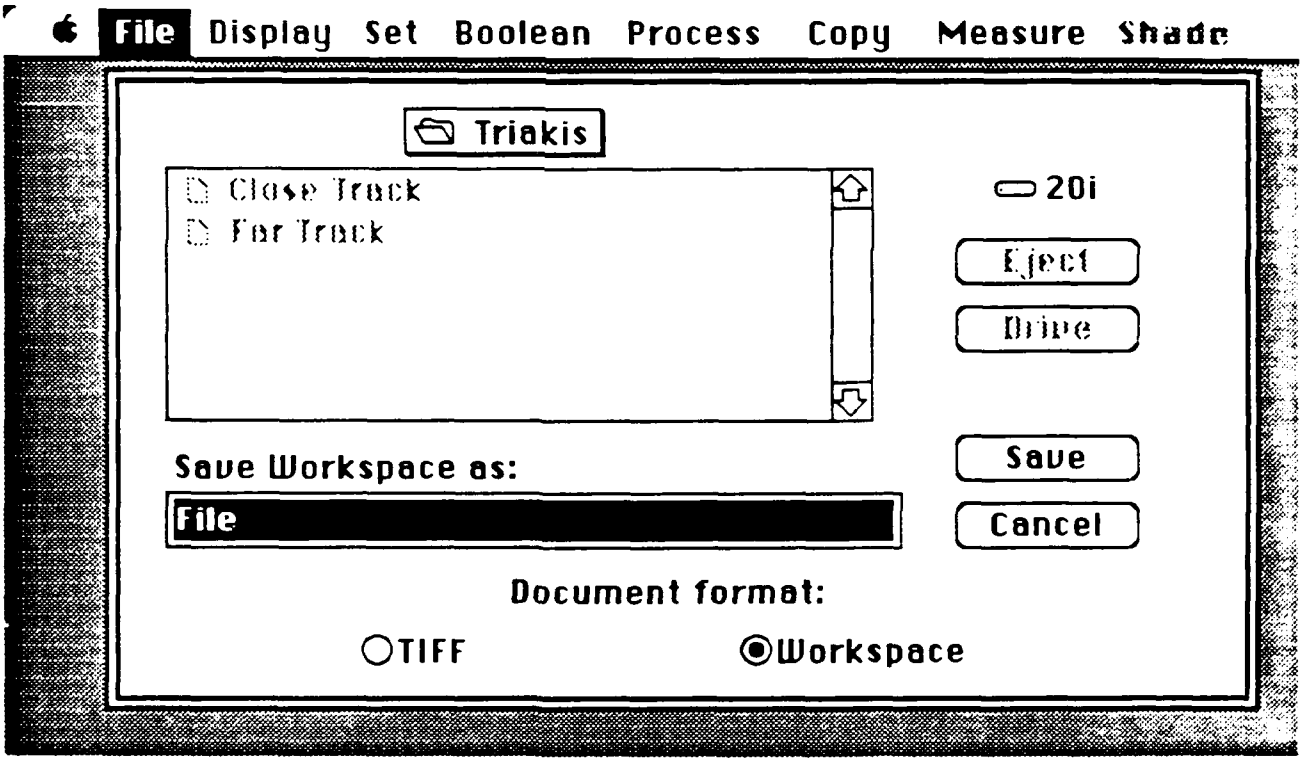

Figure 7

The Quit Command: The Quit command is used to exit the Three Dimensional Cellular Logic Data Analyzer program. The Quit command may be executed by either selecting it from the File menu with the mouse or holding down the command key (⌘) while pressing Q on the keyboard. When the Quit command is selected, the screen is cleared and control is returned to the desktop unless you have not saved one or more of the eight workspace areas. If you have not saved changes to a workspace, you will be presented with the warning box shown in Figure 8. Each of the workspaces which has not been saved will generate a separate warning box. Pressing the return key or clicking on the "Yes" button will save the workspace before quitting in the same manner as the Save command described previously. Clicking on the "No" button will quit without saving the workspace area, and clicking on the "Cancel" button will cancel the Quit operation and return control to the mouse.

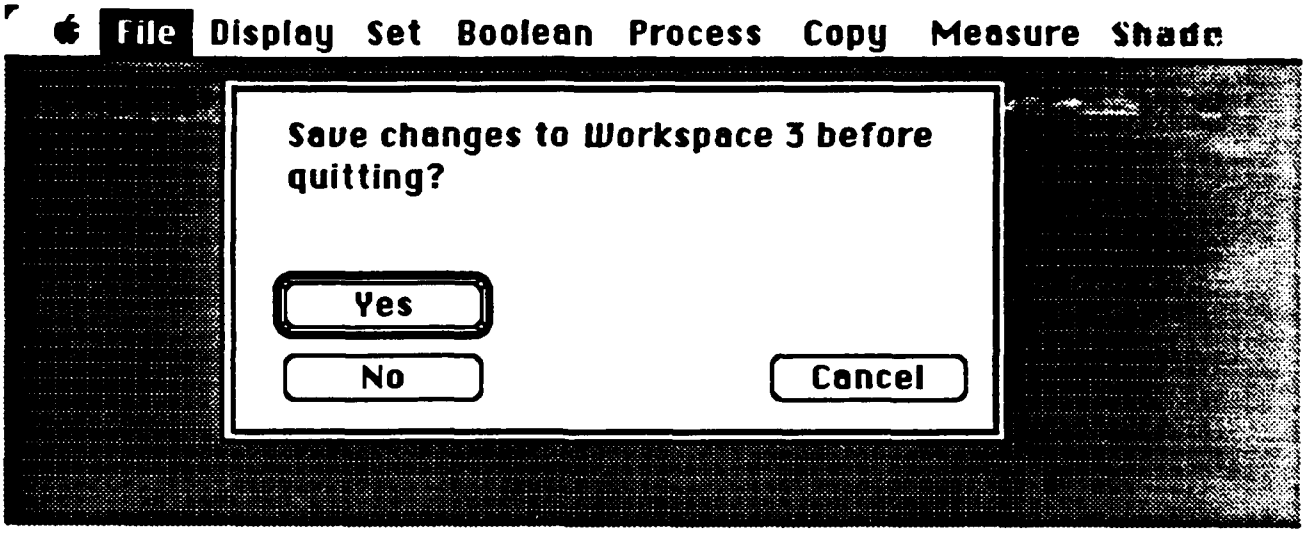
File
Display
Set
Boolean
Process
Copy
Measure
Shade
Save changes to Workspace 3 before quitting?
Yes
No
Cancel

Figure 8

The Display Menu: This menu contains the command for choosing which workspace is displayed on the screen. Select the Display menu by pressing on the word Display in the main menu bar. Selecting the Display menu produces the menu shown in Figure 9. Note that the commands shown in the dim font: Shaded, Grey Level, and Scan Display (play), are not implemented in this version of the Three Dimensional Cellular Logic Data Analyzer.

The Binary Display Command: This command allows the user to select which of the eight workspace areas is displayed on the screen. The Binary Display command may be executed by selecting it from the Display menu with the mouse. When the Binary Display command is executed, the selection box shown in Figure 10 is displayed. Select the workspace area you wish displayed by clicking on the button next to its number. The selection box will then disappear, the chosen workspace will be displayed, and control will be returned to the mouse.

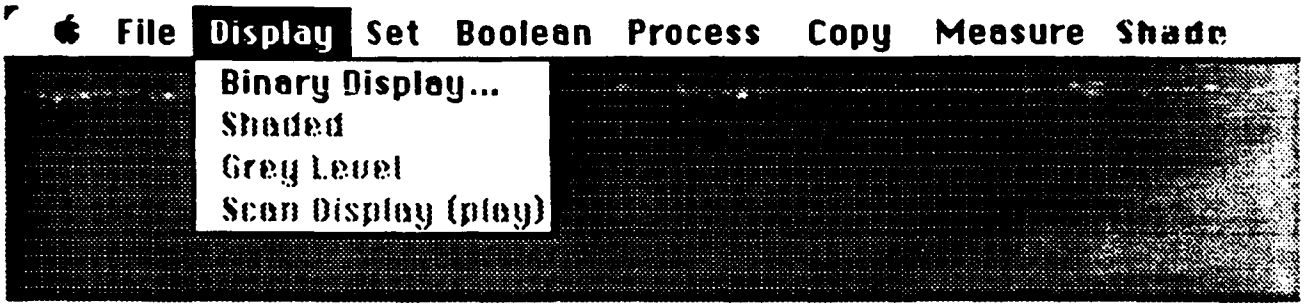

Figure 9

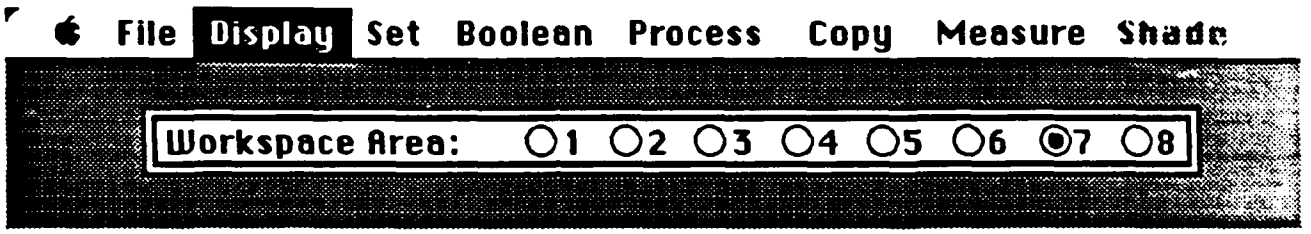

Figure 10

The Set Menu: This menu contains commands for directly changing the voxels in the workspace area displayed on the screen. Select the Set menu by pressing on the word Set in the main menu bar. Selecting the Set menu produces the menu shown in Figure 11.

The Set Voxel Command: The Set Voxel command is used to set a voxel in the workspace area displayed on the screen. The Set Voxel command may be executed by selecting it from the Set menu with the mouse. When the Set Voxel command is executed, you will be presented with the dialog box shown in Figure 12. Type in the plane, line, and point for the voxel you wish to set, typing return after each entry. If you make a mistake, you may use the keyboard editing keys or the mouse pointer to correct it. When you are satisfied with your entries, click on the "OK" button. You may cancel the Set Voxel operation by clicking on the "Cancel" button.

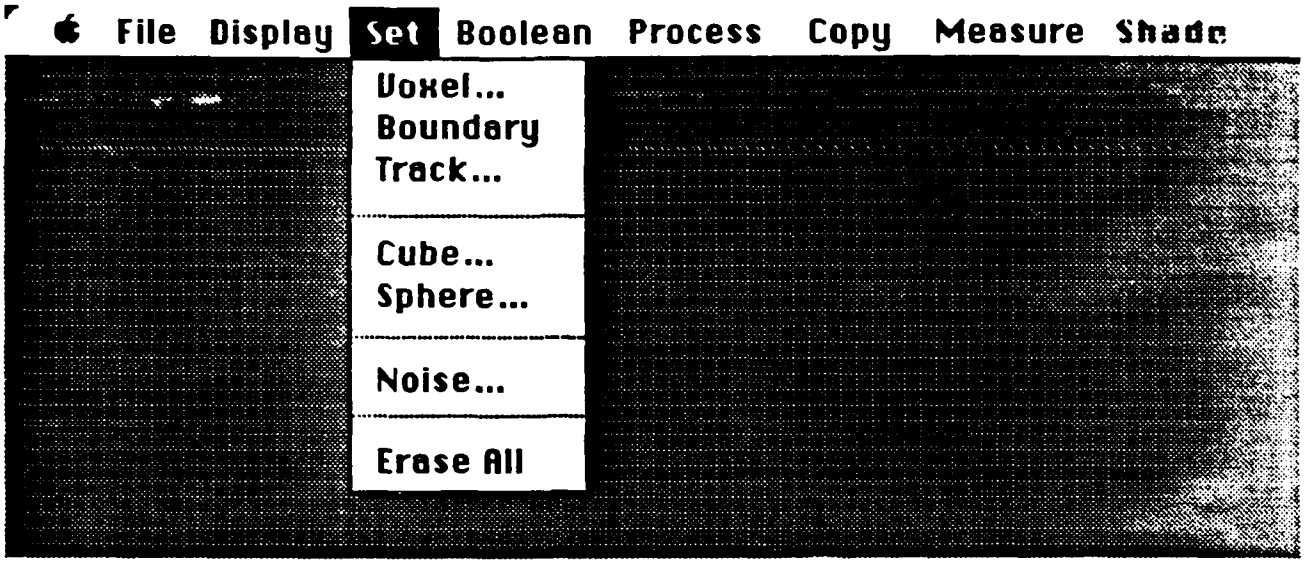

Figure 11

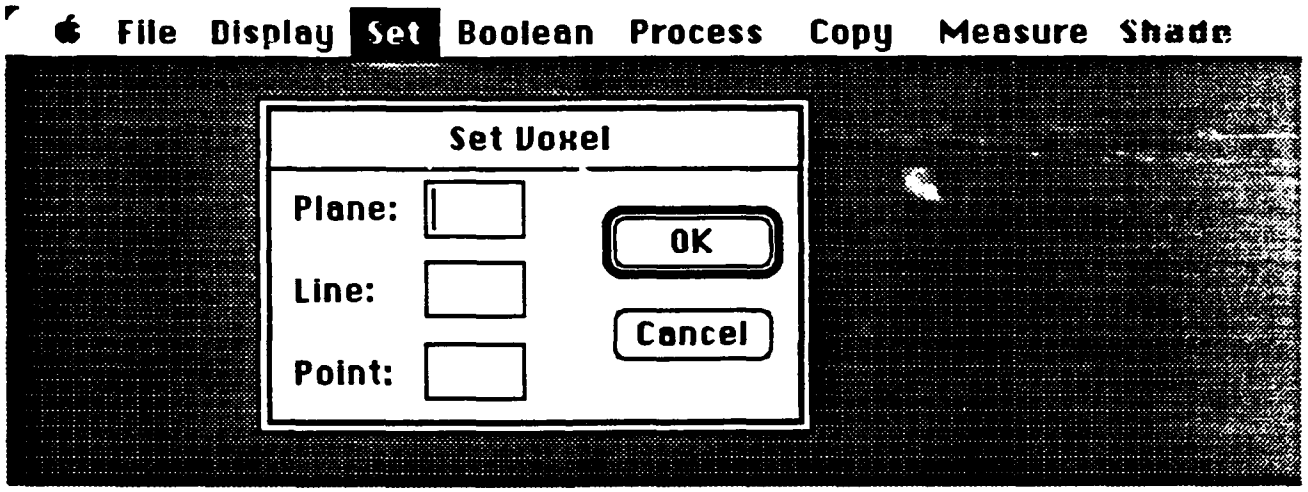

Figure 12

The Set Boundary Command: The Set Boundary command is used to set the boundary of the workspace area displayed on the screen. The Set Boundary command may be executed by selecting it from from the Set menu with the mouse. When the Set Boundary command is executed, the boundary will be set and control will immediately be returned to the mouse.

The Set Track Command: The Set Track command is used to set a track in the workspace area displayed on the screen. The Set Track command may be executed by selecting it from the Set menu with the mouse. When the Set Track command is executed, you will be presented with the dialog box shown in Figure 13. Type in the plane, line, and point for each of the endpoints of the track you wish to set, typing return after each entry. If you make a mistake, you may use the keyboard editing keys or the mouse pointer to correct it. When you are satisfied with your entries, click on the "OK" button. You may cancel the Set Track operation by clicking on the "Cancel" button.

The Set Cube Command: The Set Cube command is used to set a cube in the workspace area displayed on the screen. The Set Cube command may be executed by selecting it from the Set menu with the mouse. When the Set Cube command is executed, you will be presented with the dialog box shown in Figure 14. Type in the plane, line, and point of the lower, left, forward voxel of the cube you wish to set. Then, type in the half-width of the cube. The length of the side of the cube generated is equal to twice the half width plus one. If you make a mistake, you may use the keyboard editing keys or the mouse pointer to correct it. When you are satisfied with your entries, click on the "OK" button. You may cancel the Set Cube operation by clicking on the "Cancel" button.

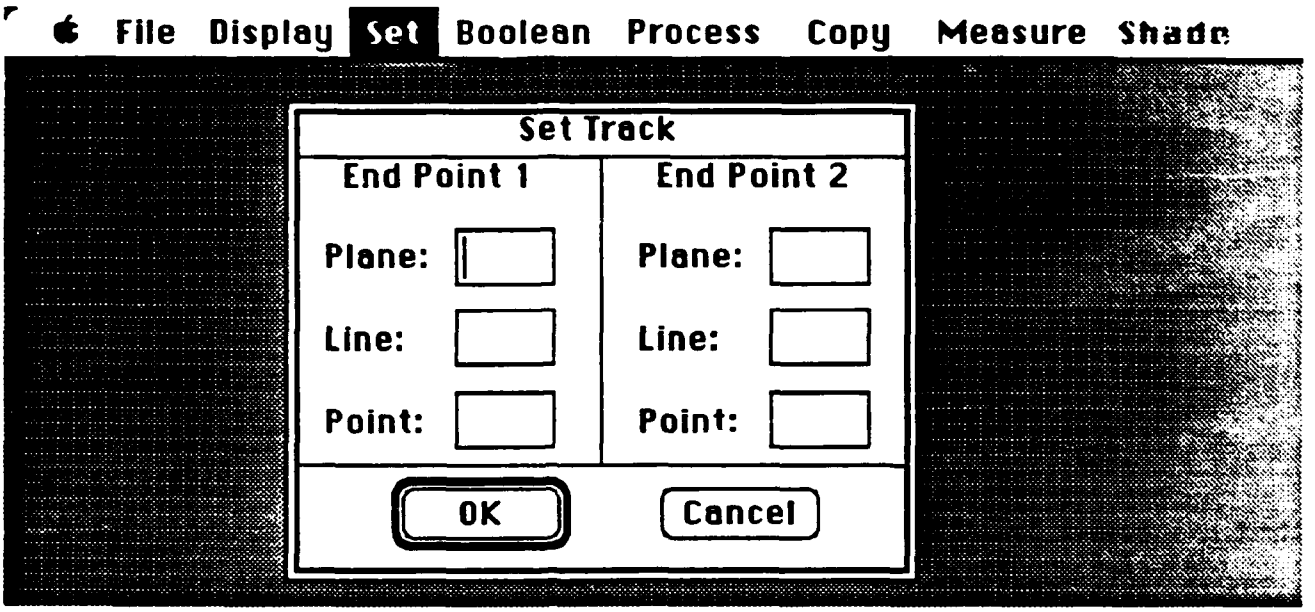

Figure 13

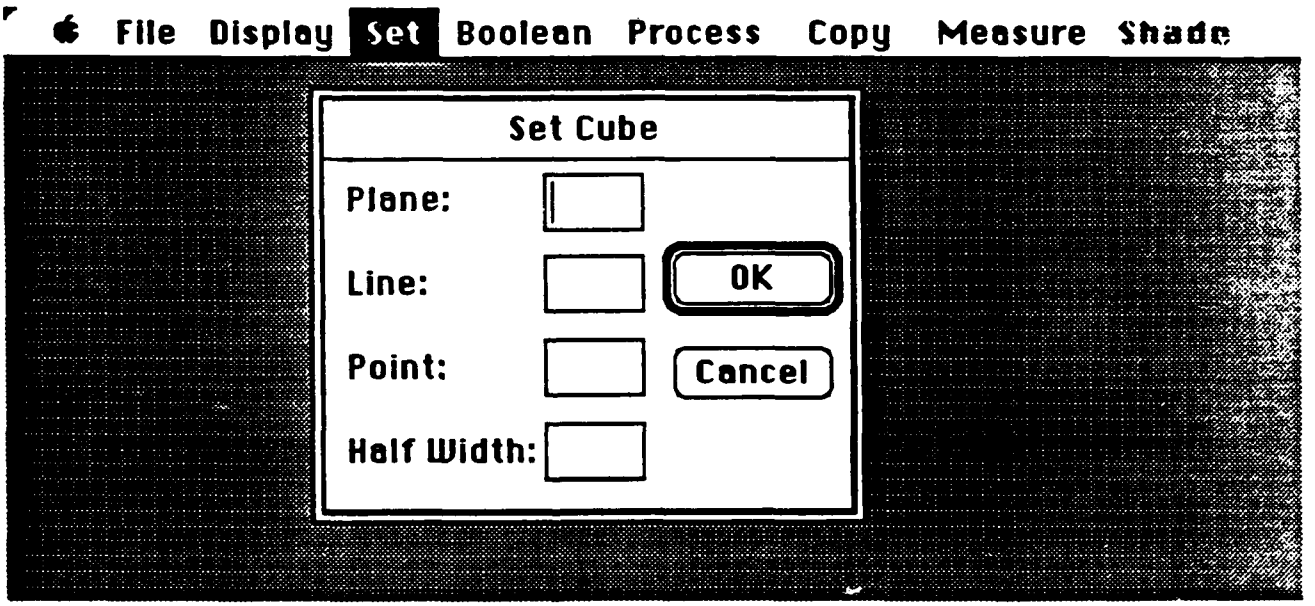

Figure 14

The Set Sphere Command: The Set Sphere command is used to set a sphere in the workspace area displayed on the screen. The Set Sphere command may be executed by selecting it from the Set menu with the mouse. When the Set Sphere command is executed, you will be presented with the dialog box shown in Figure 15. Type in the plane, line, and point for the center of the sphere, followed by its radius. The sphere generated will have a diameter equal to twice the radius plus one. If you make a mistake, you may use the keyboard editing keys or the mouse pointer to correct it. When you are satisfied with your entries, click on the "OK" button. You may cancel the Set Sphere operation by clicking on the "Cancel" button.

The Set Noise Command: The Set Noise command is used to set random noise in the workspace area displayed on the screen. The Set Noise command may be executed by selecting it from the Set menu with the mouse. When the Set Noise command is executed, you will be presented with the selection box shown in Figure 16. Select the desired sigma value by clicking on the button next to the appropriate number. The sigma value determines the amount of random noise that will be placed in the workspace area. When you are satisfied with you selection, click on the "OK" button. You may cancel the Set Noise operation by clicking on the "Cancel" button.

The Erase All Command: The Erase All command is used to erase (set to zero) all of the voxels in the workspace area displayed on the screen. The Erase All command may be executed by selecting it from the Set menu with the mouse. When the Erase All command is executed, the workspace displayed on the screen will be erased and control will be returned to the mouse. Note that, unlike the close command, any file name information associated with the workspace area being cleared will be retained.

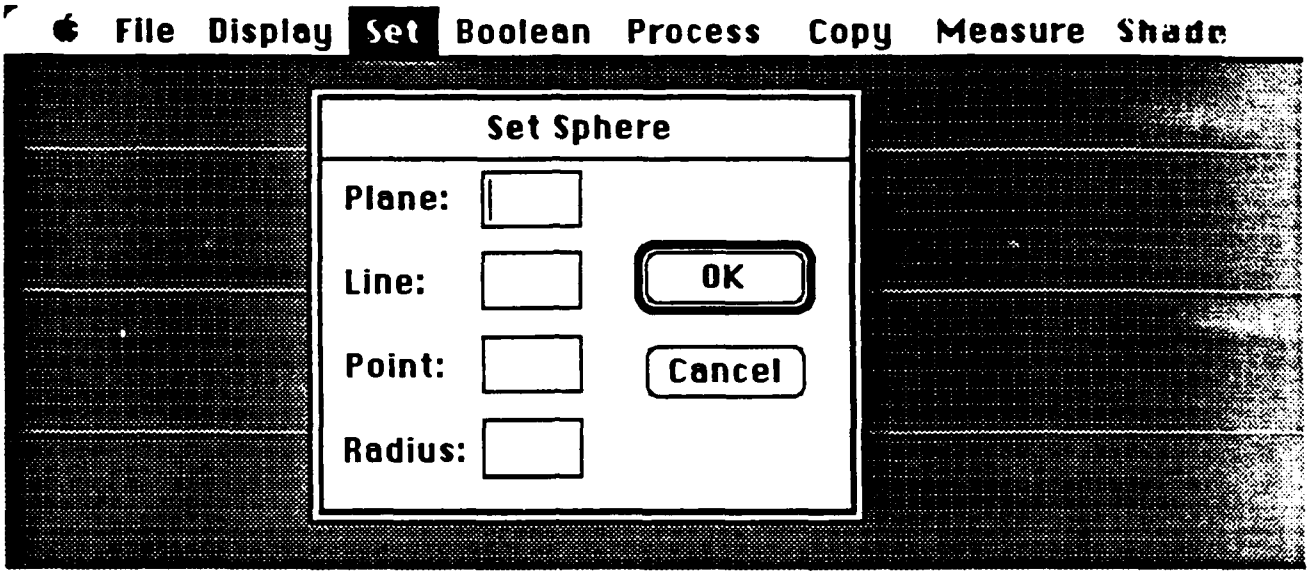

Figure 15

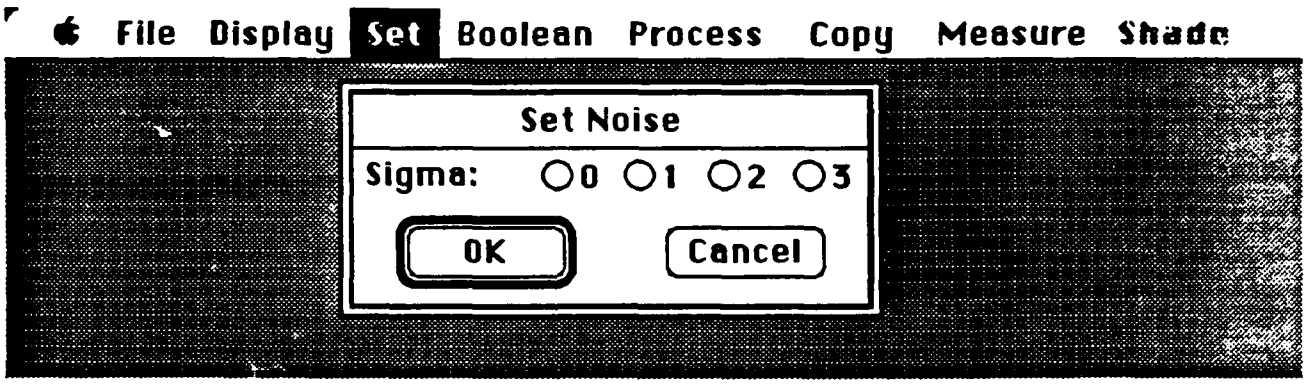

Figure 16

The Boolean Menu: The Boolean Menu contains commands for performing the boolean And, Or, Exor, and Invert operations on any of the eight workspace areas. Select the Boolean menu by pressing on the word Boolean in the main menu bar. Selecting the Boolean menu produces the menu shown in Figure 17.

The And Command: The And command is used to perform the boolean And operation from any two of the eight workspace areas into a third workspace area. The And command may be executed by selecting it from the Boolean menu with the mouse. When the And command is executed, you will be presented with the selection box shown in Figure 18. Select both of the source workspaces and the destination workspace by clicking on the button next to its number. When you are satisfied with your selections, click on the "OK" button. You may cancel the And operation by clicking on the "Cancel" button.

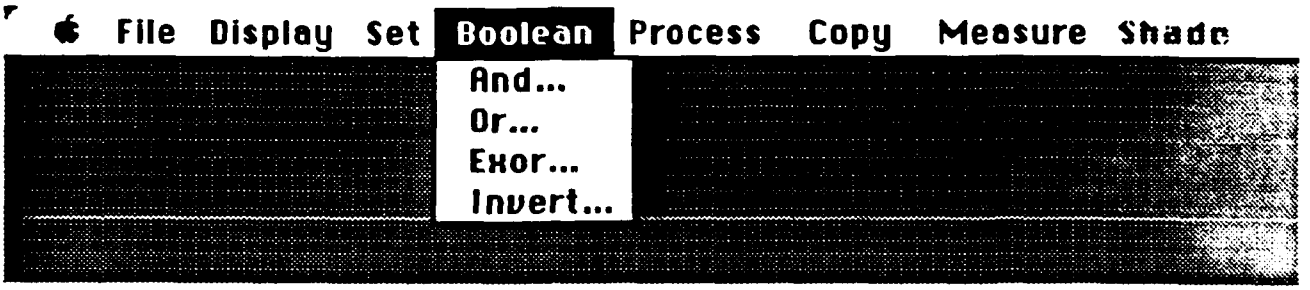

Figure 17

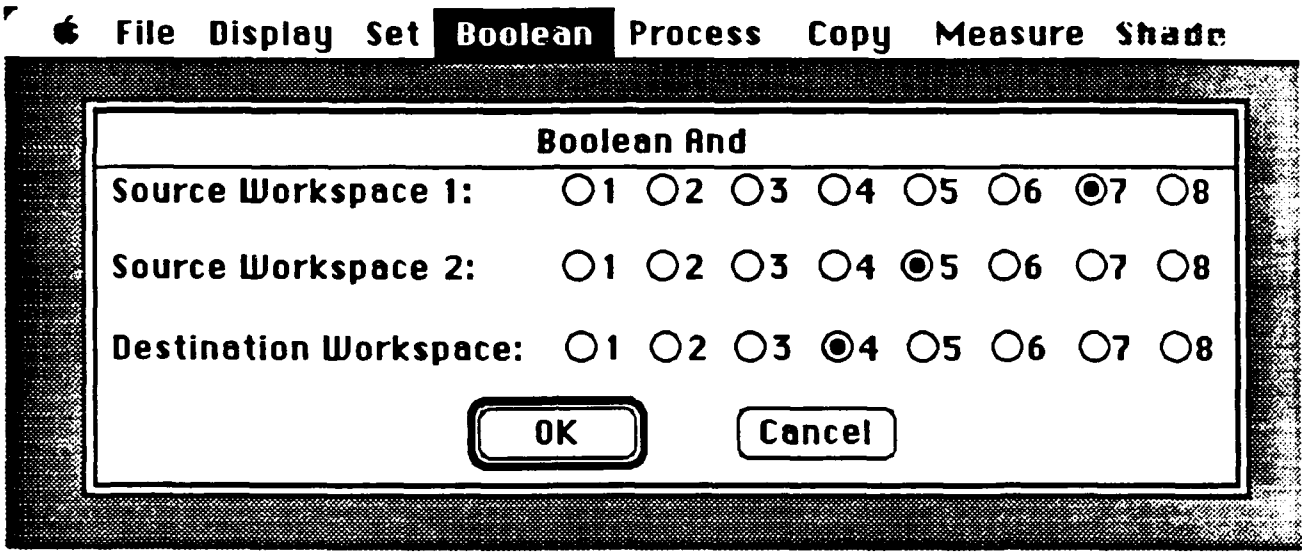

Figure 18

The Or Command: The Or command is used to perform the Boolean Or operation from any two of the eight workspace areas into a third workspace area. The Or command may be executed by selecting it from the Boolean menu with the mouse. When the Or command is executed, you will be presented with the selection box shown in Figure 19. Select both of the source workspaces and the destination workspace by clicking on the button next to its number. When you are satisfied with your selections, click on the "OK" button. You may cancel the Or operation by clicking on the "Cancel" button.

The Exor Command: The Exor command is used to perform the Boolean Exor operation from any two of the eight workspace areas into a third workspace area. The Exor command may be executed by selecting it from the Boolean menu with the mouse. When the Exor command is executed, you will be presented with the selection box shown in Figure 20. Select both of the source workspaces and the destination workspace by clicking on the button next to its number. When you are satisfied with your selections, click on the "OK" button. You may cancel the Exor operation by clicking on the "Cancel" button.

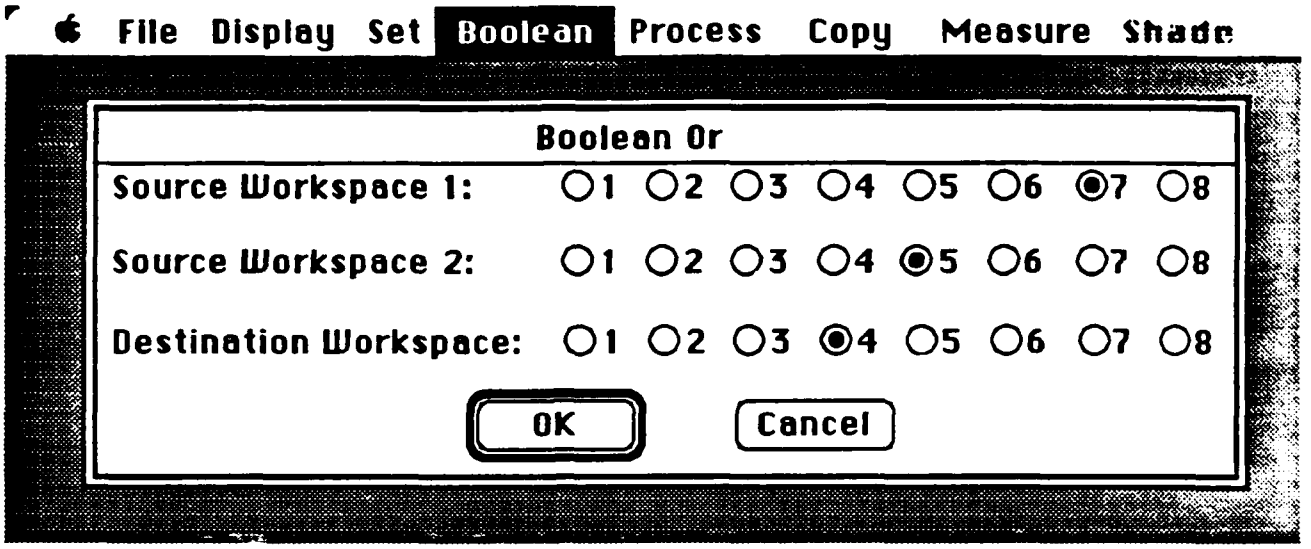

Figure 19

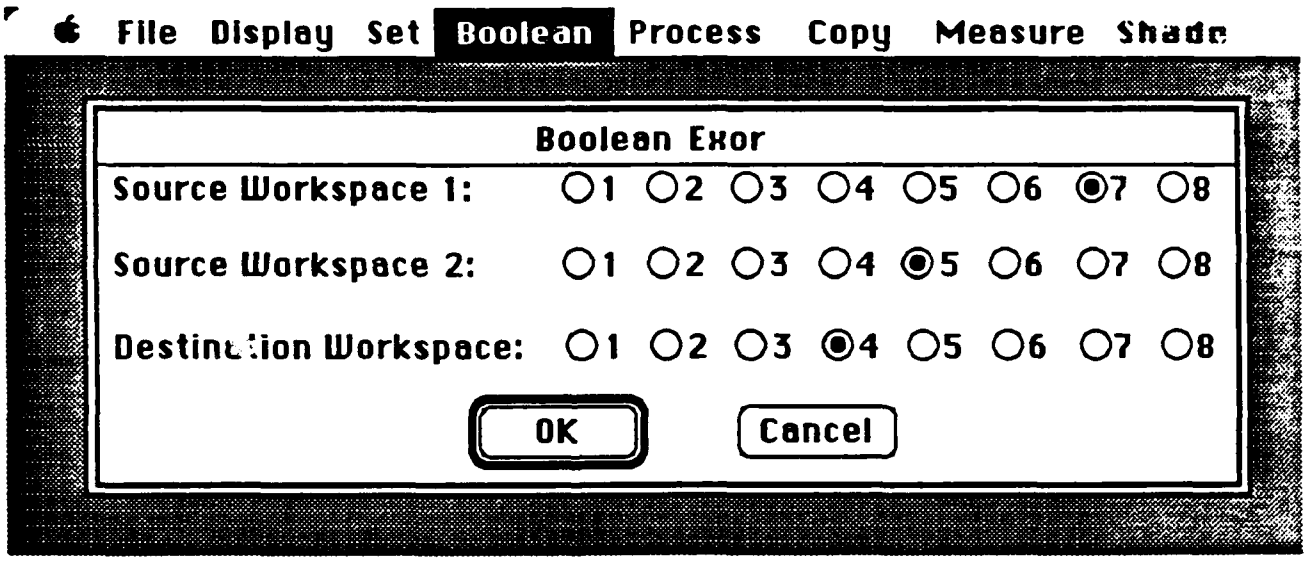

Figure 20

The Invert Command: The Invert command is used to perform the Boolean Invert operation on any of the eight workspace areas into a second workspace area. The Invert command may be executed by selecting it from the Boolean menu with the mouse. When the Invert command is executed, you will be presented with the selection box shown in Figure 21. Select the source and the destination workspaces by clicking on the button next to their numbers. When you are satisfied with your selections, click on the "OK" button. You may cancel the Invert operation by clicking on the "Cancel" button.

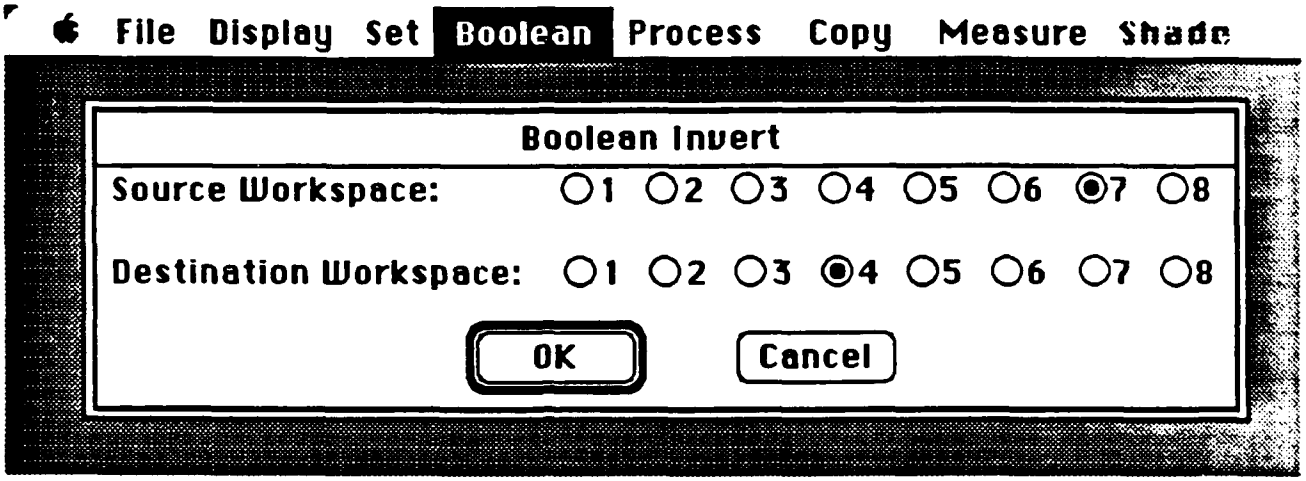
File Display Set Boolean Process Copy Measure Shade
Boolean Invert
Source Workspace: 1 2 3 4 5 6 7 8
Destination Workspace: 1 2 3 4 5 6 7 8
OK
Cancel

Figure 21

The Process Menu: This menu contains commands for performing the Augred and Xifilter cellular logic operations. Select the Process menu by pressing on the word Process in the main menu bar. Selecting the Process menu produces the menu shown in Figure 22.

The Augred Command: The Augred command is used to perform the augred cellular logic operation on the workspace area displayed on the screen. The Augred command may be executed by selecting it from the Process menu with the mouse. When the Augred command is executed, you will be presented with the dialog box shown in Figure 23. Type in the number of cycles and the factor, typing return after each entry. If you make a mistake, you may use the keyboard editing keys or the mouse pointer to correct it. Next, choose Cnum0, Cnum1, Mode, and Border by clicking on the button next to the appropriate number. When you are satisfied with you selections, click on the "OK" button. You may cancel the Augred operation by clicking on the "Cancel" button.

Figure 22

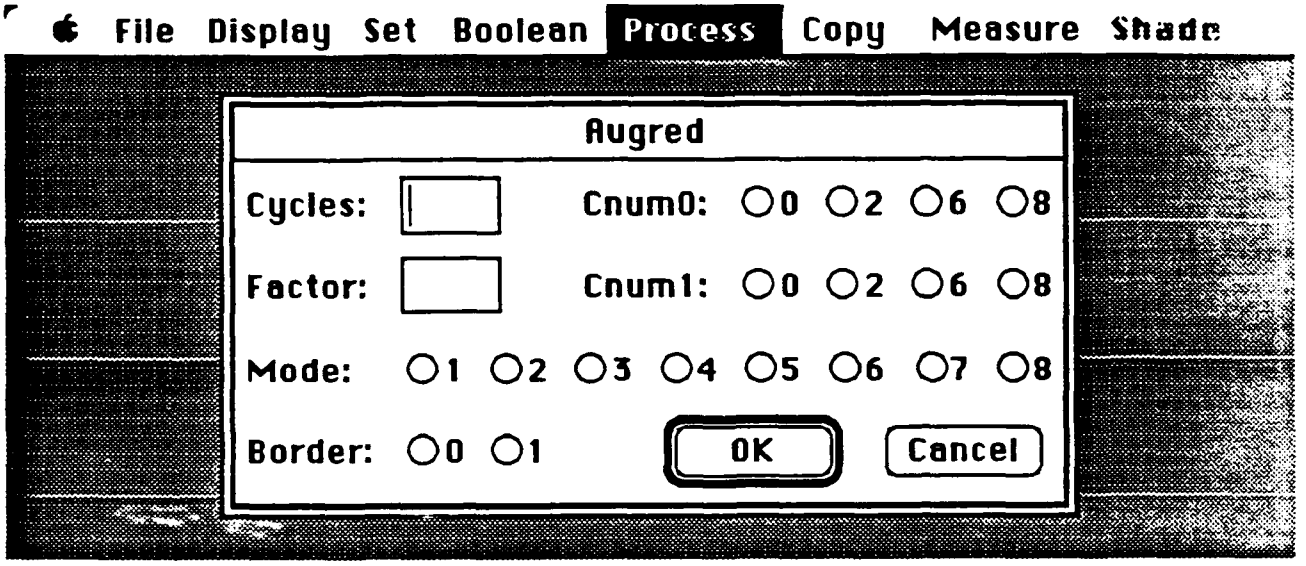

Figure 23

The Xifilter Command: The Xifilter command is used to perform the xifilter cellular logic operation on the workspace area displayed on the screen. The Xifilter command may be executed by selecting it from the Process menu with the mouse. When the Xifilter command is executed, you will be presented with the dialog box shown in Figure 24. Type in the number of cycles and the factor, typing return after each entry. If you make a mistake, you may use the keyboard editing keys or the mouse pointer to correct it. Next, choose Cnum0, Cnum1, Mode, and Border by clicking on the button next to the appropriate number. When you are satisfied with you selections, click on the "OK" button. You may cancel the Xifilter operation by clicking on the "Cancel" button.

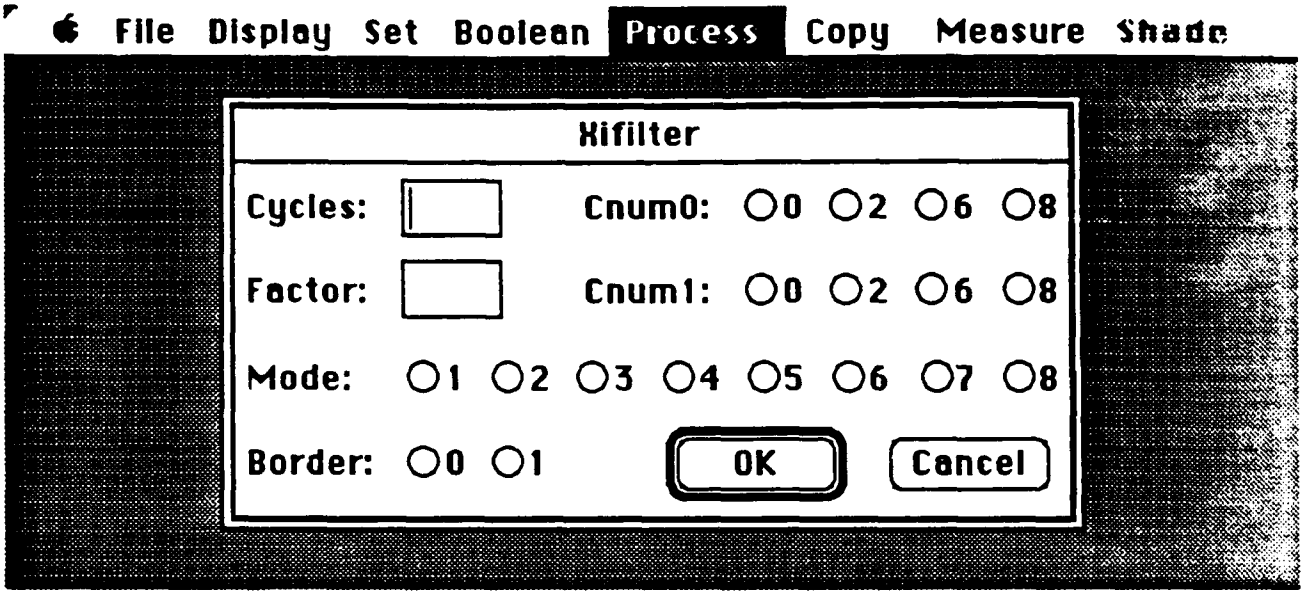
File Display Set Boolean Process Copy Measure Shade
Hifilter
Cycles:
Factor:
Mode:
Border:
Cnum0:
Cnum1:
0 2 6 8
0 2 6 8
1 2 3 4 5 6 7 8
0 1
OK
Cancel

Figure 24

The Copy Menu: This menu contains the command for copying one workspace area into another. Select the Copy menu by pressing on the word Copy in the main menu bar. Selecting the Copy menu produces the menu shown in Figure 25.

The Copy Workspace Command: The Copy Workspace command is used to copy the contents of one workspace into another. The Copy Workspace command may be executed by selecting it from the Copy menu with the mouse. When the Copy Workspace command is executed, you will be presented with the selection box shown in Figure 26. Select the source and destination workspaces by clicking on the buttons next to their numbers. When you are satisfied with your selections, click on the "OK" button. You may cancel the Copy Workspace operation by clicking on the cancel button.

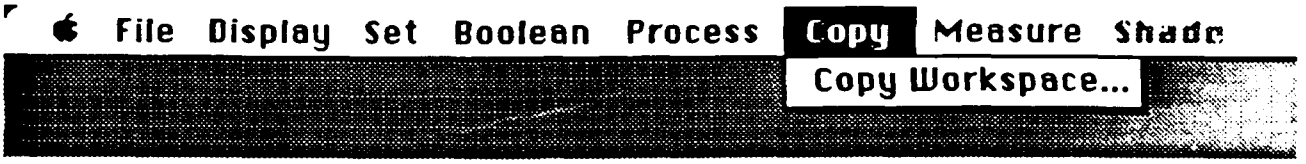

Figure 25

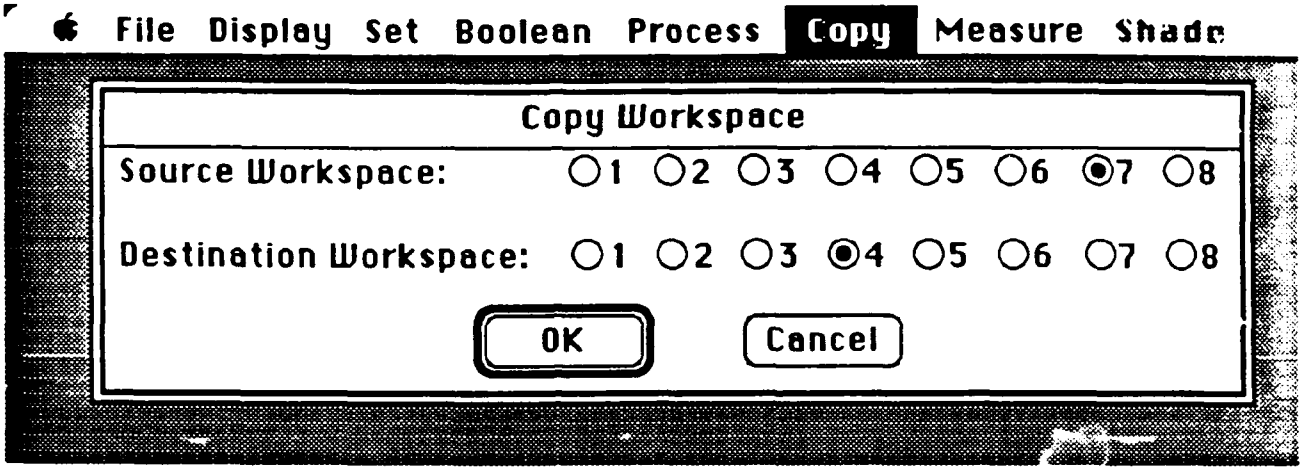

Figure 26

The Measure Menu: This menu contains commands for measuring various quantities in the workspace area displayed on the screen. Select the measure menu by pressing on the word Measure in the main menu bar. Selecting the Measure menu produces the menu shown in Figure 27. Note that only the Volume command, which measures the number of voxels that are set in the workspace displayed on the screen, is supported. The Surface, Edges, and Corners commands are shown in the dim font to remind you that they are not supported in this version of the The Three Dimensional Cellular Logic Data Analyzer.

The Measure Volume Command: The Measure Volume command is used to measure the number of true voxels in the workspace area displayed on the screen. The Measure Volume command may be executed by selecting it from the Measure menu with the mouse. When the Measure Volume command is executed, the program will calculate the volume and display the results in the box shown in Figure 28. You may then save the measurement to a file called "Measurements" by clicking on the "OK" box. If you do not wish to save the value to the measurements file, click on the cancel button.

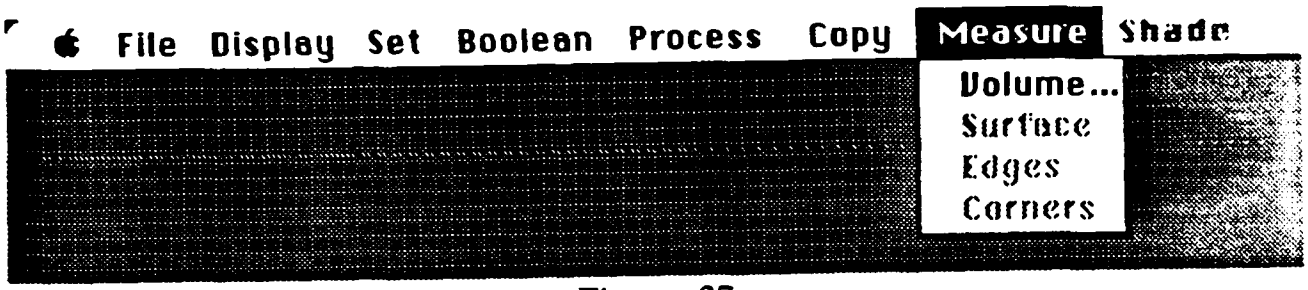

Figure 27

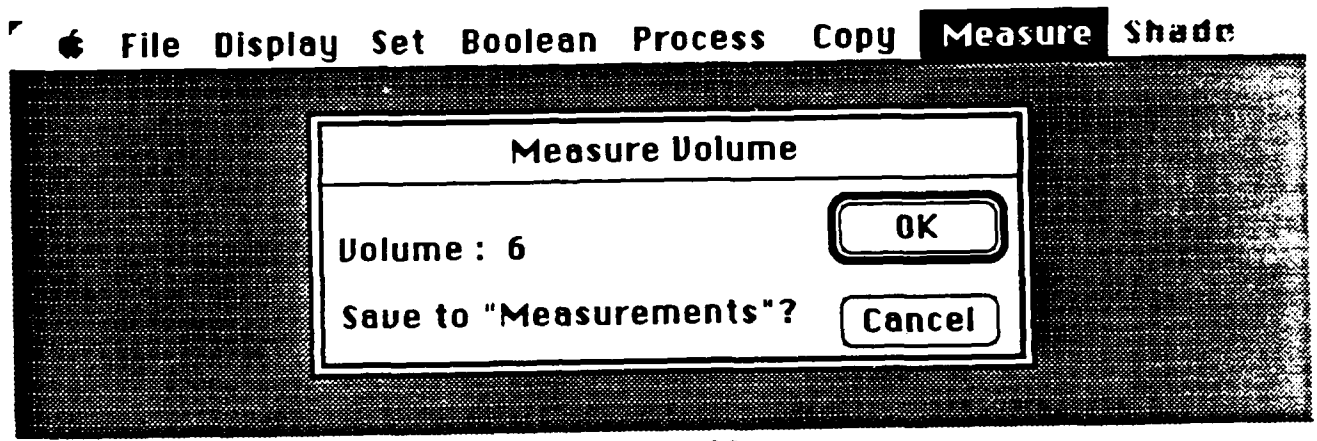

Figure 28