FINAL REPORT

STUDY OF ASSOCIATIVE PROCESSING FOR INTELLIGENCE APPLICATIONS

Volume I - Basic Report

Contract XG-4124(62-7009)75R

GER-16227

AUGUST 1975

GOODYEAR AEROSPACE CORPORATION Akron, Ohio 44315

CODE IDENT NO. 25500

GOODYEAR AEROSPACE

CORPORATION

AKRON, OHIO 44315

STUDY OF ASSOCIATIVE PROCESSING FOR INTELLIGENCE APPLICATIONS

FINAL REPORT, CONTRACT XG-4124(62-7009)75R

VOLUME I - BASIC REPORT

GER-16227 AUGUST 1975

FOREWORD

This final technical report records the efforts and the results achieved on the study of associative processing for intelligence applications conducted by Goodyear Aerospace Corporation (GAC), Akron, Ohio. This report entitled, Study of Associative Processing for Intelligence Applications, is published in three volumes: Volume I - Basic Report (includes Appendix A), Volume II - Application Analysis and Validation (includes Appendices B through O), and Volume III - Error Correction Application Analysis. (Classified).

The study was conducted for the U. S. Government under Contract XG-4124(62-7009)75R. The study covers the period from 11 November 1975 to 11 August 1975. The customer project monitor is Mr. Malcom Tanigawa. The major contributors to this study were D. L. Rohrbacher (project engineer), A. E. Hale and R. G. Gall. Substantial contributions were also made by F. G. Carty, R. O. Faiss, M. Hansburg and R. W. Messner.

TABLE OF CONTENTS

SECTION	Title	Page
I	EXECUTIVE SUMMARY	1
	1. PROGRAM DESCRIPTION	1
	a. Purpose	1
	b. Approach	1
	2. SUMMARY OF RESULTS AND CONCLUSIONS	2
	3. RECOMMENDATIONS	7
II	TECHNICAL SYNOPSIS	9
	1. INTRODUCTION	9
	2. APPLICATION ANALYSIS	13
	a. Image Processing	13
	(1) General	13
	(2) Frequency Domain Filtering	16
	(3) Convolution	19
	(4) Edge Detection	21
	(5) First Differences	23
	(6) Magnification	25
	(7) Statistics	27
	(8) Image Averaging	29
	b. Text Search	33
	c. Solution of Linear Systems	37
	d. Matrix Inversion	43
	e. Matrix/Vector Product	49

TABLE OF CONTENTS (Cont'd)

SECTION	Title	Page
	f. Antenna Simulation	51
	g. Error Correction	57
	3. VALIDATION	58
	a. Requirements	58
	b. Validation Problem Description	59
	c. Approach	59
	d. Results	62
APPENDIX A - STARAN SYSTEM DESCRIPTION		

LIST OF ILLUSTRATIONS

FIGURE	Title	Page
1	Image Averaging Process for Validation.........	6
2	STARAN Comparison with Conventional Computer...	10
3	STARAN System Configurations..................	12
4	Image Averaging Problem Description...........	30
5	Simplified Flowchart for Image Averaging.......	32
6	Simplified Search Routine.....................	35
7	Array Storage Scheme for Solution of a Linear System..	40
8	Operation Counts for Solution of a Linear System	41
9	Antenna Simulation Solution Approach..........	54
10	Validation Problem Description................	60
11	Input Imagery.................................	61
12	STARAN E&T Facility Configuration.............	63
13	Top Level Flow chart - IAVG...................	64
14	Image Averaging Process for Validation - STARAN vs Sequential Machine.........................	65

LIST OF TABLES

TABLE	Title	Page
I	Selected Problems..............................	3
II	Performance Results and Comparisons Summary......	4
III	Text Search Timing..............................	36
IV	Execution Times for Solution of a Linear System..	42
V	Arithmetic Operation Counts for Matrix Inversion on a Sequential Computer.........................	45
VI	STARAN Operation Count..........................	47
VII	Execution Times for Matrix Inversion............	48
VIII	MVP Timing Summary..............................	52
IX	Antenna Simulation Timing Summary and Comparison.	56
X	Detailed STARAN Timing for Validation Program (IAVG)..	67
XI	Validation Program Statistics...................	68

SECTION I - EXECUTIVE SUMMARY

1. PROGRAM DESCRIPTION

a. Purpose

This study was initiated to investigate and validate the applicability of the STARAN* associative processor (AP) to a variety of intelligence data processing applications. The goal of this program is to provide a solid base for making judgements concerning the potential benefit that can be obtained through the use of associative processing.

b. Approach

The study was performed in three phases:

1) application survey
2) detailed analysis
3) validation

The application survey (Phase 1) consisted of the examination of a variety of intelligence problems to determine their general suitability for parallel processing. Several of the most promising problems were to be selected for detailed analysis (Phase 2). Actually, nearly all the problems surveyed were judged to be amenable to parallel processing. Consequently, detailed analysis was performed on 13 problems during the Phase 2 effort. Finally, one of the problems which was analyzed in detail was selected for validation (Phase 3). The validation phase consisted of actually programming the selected problem and executing it on the STARAN array processor in the STARAN Evaluation and Training Facility at Goodyear Aerospace Corporation (GAC) in Akron, Ohio. The actual programming and execution of the validation problem not only

*T.M., Goodyear Aerospace Corporation, Akron, Ohio.

provided a measured execution time, but also provided additional statistics which can be directly related to the cost of programming the STARAN array processor. Since most of the application problems considered in this study are currently being run on the sequential machines in the customer computer facility, measured sequential machine times were supplied by the customer for most of the problems considered. These measured sequential machine times are included in this report for comparison with the STARAN approach.

The problems that were selected for analysis, during the Phase 2 effort, are listed and briefly described in Table I. For each problem, several variations of system configuration for STARAN were assumed and performance was analyzed for each variation. Image averaging was the problem selected for the Phase 3 validation effort. This problem was also programmed for execution on the IBM 360/195 and 360/65 sequential machines by customer agency personnel. This programming effort was oriented toward maximization of performance (i.e., minimization of run time) in order to insure an equitable comparison with the performance of the corresponding GAC STARAN program.

2. SUMMARY OF RESULTS AND CONCLUSIONS

The results of the performance analysis (Phase 2 of the study is summarized in Table II. This table shows the solution time estimates for the STARAN implementation of all the problems considered. For comparison, solution times for sequential-machine implementation of each of the problems are included. With two exceptions, all the sequential machine times are measured times supplied by the customer agency. In the two cases where measured sequential-machine times were not available from the customer agency, estimates were supplied by GAC. In most cases, the measured solution times were provided for

TABLE I - SELECTED PROBLEMS

PROBLEM	DESCRIPTION
IMAGE PROCESSING	
CONVOLUTION	MODIFY EACH PIXEL OF AN IMAGE AS A FUNCTION OF THE SUM-OF-PRODUCTS OF ITS NEAR NEIGHBORS WITH AN A'PRIORI WEIGHTING MATRIX.
EDGE DETECTION	DETECT AND INTENSIFY EDGES IN AN IMAGE BY CALCULATING THE AVERAGE GRAY LEVEL DIFFERENCES OF TWO NEIGHBORHOODS ON OPPOSITE SIDES OF EACH IMAGE PIXEL (IN EACH OF FOUR DIRECTIONS), FORMING THE PRODUCT OF THESE DIFFERENCES AND SELECTING THE MAXIMUM OF THE FOUR PRODUCTS.
FIRST DIFFERENCES	CALCULATE THE DIFFERENCE BETWEEN THE GRAY VALUES OF EACH DIGITAL IMAGE PIXEL AND ITS IMMEDIATE NEIGHBOR PIXEL IN THE HORIZONTAL, VERTICAL, AND THE TWO DIAGONAL DIRECTIONS. THE MAXIMUM OF THE FOUR DIFFERENCES IS SELECTED AS THE NEW GRAY VALUE OF THE IMAGE PIXEL.
MAGNIFICATION	EXPAND AN IMAGE BY A FACTOR OF EIGHT IN BOTH HORIZONTAL AND VERTICAL DIRECTIONS USING LINEAR INTERPOLATION BETWEEN ORIGINAL IMAGE PIXELS.
FREQUENCY DOMAIN FILTERING	PERFORM IMAGE MODIFICATION BY (1) TRANSFORMING THE IMAGE INTO THE FREQUENCY DOMAIN USING A TWO-DIMENSIONAL FFT, (2) MULTIPLICATION BY A FREQUENCY DOMAIN FILTER, AND (3) TRANSFORMATION BACK TO THE SPACE DOMAIN USING TWO-DIMENSIONAL INVERSE FFT.
STATISTICS	CALCULATES THE FOLLOWING PARAMETERS OF AN IMAGE: (1) HISTOGRAM OF GRAY LEVELS, (2) MEAN GRAY LEVEL, (3) STANDARD DEVIATION, AND (4) MAXIMUM AND MINIMUM GRAY LEVELS.
IMAGE AVERAGING	COMPUTE A NEW GRAY VALUE FOR EACH SPECIFIED PIXEL OF THE INPUT IMAGE BY FINDING THE AVERAGE GRAY VALUE OF ALL THE IMAGE PIXELS WITHIN THE BOUNDARIES OF A SPECIFIED MATRIX CELL, SUCH THAT THE IMAGE PIXEL OF INTEREST IS LOCATED IN THE TOP LEFT CORNER.
TEXT SEARCH	GIVEN A LARGE TEXTUAL DATA BASE, RETRIEVE THOSE DOCUMENTS THAT SATISFY A MULTI-WORD QUERY THAT IS COMPOSED OF AN INTERSECTION OF UNIONS.
SOLUTION OF A LINEAR SYSTEM	SOLVE A SYSTEM OF 256 LINEAR EQUATIONS WITH 256 UNKNOWNS BY THE GAUSSIAN ELIMINATION TECHNIQUE.
MATRIX INVERSION	INVERT A 256 X 256 MATRIX BY THE GAUSSIAN ELIMINATION TECHNIQUE.
MATRIX/VECTOR MULTIPLY	MULTIPLY A 256 X 256 MATRIX BY A 1 X 256 COLUMN VECTOR.
ANTENNA SIMULATION	ANALYZE ANTENNA PERFORMANCE USING A MODEL BASED ON THE THIN WIRE ELECTRIC FIELD INTEGRAL EQUATION, WHICH IS SOLVED USING NUMERICAL TECHNIQUES.
ERROR CORRECTION	GIVEN A SET OF RECEIVED TELEMETRY DATA THAT CONTAINS ERRORS DUE TO POOR S/N RATIO, CREATE A SET OF ALL POSSIBLE CORRECT DATA, AND THEN USE KALMAN FILTER TO DETERMINE THE TRUE TRANSMITTED DATA.

TABLE II - PERFORMANCE RESULTS AND COMPARISONS SUMMARY

PROBLEM	STARAN ARRAY PROCESSOR				SEQUENTIAL MACHINES TIME	PERFORMANCE RATIO T_S/T_P
	NO. OF ARRAYS	ARRAY PROCESSING TIME* (SEC)	I/O TIME (SEC)	TOTAL TIME+ (SEC)		
IMAGE PROCESSING						
FREQUENCY DOMAIN FILTERING	4	1.73	0.05	1.78	IBM 360/195-39 SEC (CPU TIME)	20
CONVOLUTION (12 BITS/PIXEL)	4	1.19	0.004	1.20	IBM 360/195-62 SEC (CPU TIME)	50
EDGE DETECTION	4	1.30	0.003	1.31	IBM 360/195-133 SEC (CPU TIME)	100
FIRST DIFFERENCES	4	0.03	0.003	0.04	IBM 360/195-136 SEC (CPU TIME)	4000
MAGNIFICATION	4	0.59	0.083	0.67	IBM 360/195-769 SEC (CPU TIME)	1000
STATISTICS	1	3.90	1.2**	5.10	IBM 360/195-8 SEC (CPU TIME)	2
AVERAGING	2	0.21	0.008	0.22	IBM 360/195-8.04 SEC (CPU TIME)	40
TEXT SEARCH	32	1.23	0.49	1.71	IBM 370/145-1000 SEC (CPU TIME) (ESTIMATED)	800
	16	2.46	0.49	2.95		
	8	4.93	0.49	5.41		
MATRIX MATH OPERATIONS						
LINEAR SYSTEM SOLUTION (SP REAL, n=256)	4	2.84	0.21	3.05	IBM 360/65-135 SEC (CPU TIME)	50
INVERSION (SP REAL, n=200)	4	3.9	0.4	4.3	IBM 360/195-21 SEC (CPU TIME)	5
MULTIPLY (BY VECTOR) (SP REAL, n=128)	4	0.01	0.005	0.015	-	-
ANTENNA SIMULATION (DP COMPLEX, n=128)	4	6.98	1.65	8.63	IBM 360/195-684 SEC (CPU TIME)	100
ERROR CORRECTION (REAL TIME = 64 SEC)	1	-	-	3.71	PDP11/45 - 64 X 10^3 SEC (TOTAL TIME)	17,000

GENERAL NOTES:
1. FOR ALL IMAGE PROBLEMS, IMAGE SIZE = 512 X 512 PIXELS
2. FOR ALL IMAGE PROBLEMS, 8 BITS/PIXEL UNLESS OTHERWISE NOTED.
3. ALL SEQUENTIAL MACHINE TIMES ARE MEASURED VALUES UNLESS OTHERWISE NOTED.
4. ALL STARAN I/O TIMES BASED ON FAST INTERFACE (EXTENDED MEMORY) UNLESS OTHERWISE NOTED.
5. 25% OVERHEAD TIME INCLUDED IN ALL INDICATED STARAN TIMES
6. T_S = SEQUENTIAL SOLUTION TIME
7. T_P = PARALLEL (STARAN) ARRAY PROCESSING SOLUTION TIME

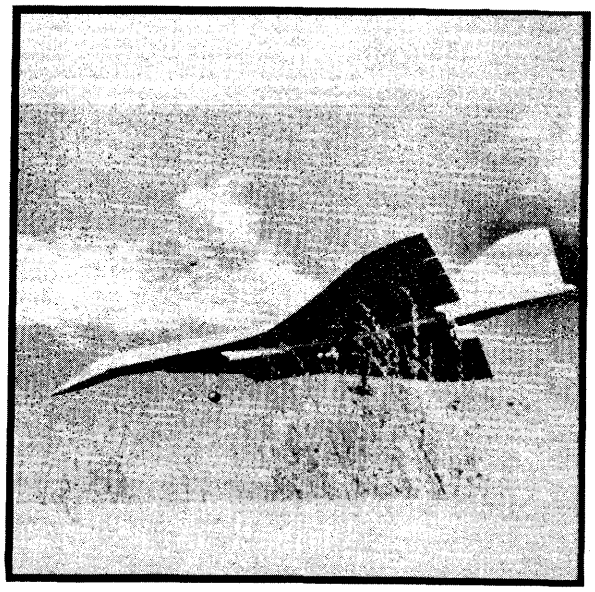
FOOTNOTES:
* COMPARE WITH SEQUENTIAL MACHINE CPU TIMES
+ COMPARE WITH SEQUENTIAL MACHINE TOTAL TIMES
** CONVENTIONAL DISC I/O
SP = SINGLE PRECISION
DP = DOUBLE PRECISION

both the IBM 360/65 and the IBM 360/195. However, only the fastest sequential machine times available from the customer are included in Table II. The times for the slower sequential machines are reported in the individual problem descriptions found in Section II, TECHNICAL SYNOPSIS. With few exceptions, STARAN provided dramatic performance increases over the fastest sequential machine, usually the IBM 360/195. Such dramatic improvements over the powerful 360/195 might naturally raise questions concerning the validity of the STARAN timing estimates. This is basic reason for the validation phase of the study. For validation, one of the problems considered (image averaging) was actually programmed and executed on the STARAN at GAC. The very same problem was programmed by the customer agency and executed on the sequential IBM 360/195 (and 360/65) computer at their facility. The image averaging process and results for validation are shown in Figure 1. Execution time was measured for both systems. The results showed the STARAN provided a performance improvement over the IBM 360/195 by a factor of 40. In addition, the measured STARAN time was about 25 percent faster than the estimated STARAN time. This directly validates the performance estimate for the image averaging problem, and by implication validates the performance estimates for all problems considered since the same basic estimating techniques were used throughout.

The image averaging process is shown also in photographic form in Figure 1. The same image with noise added was provided as an input both to the STARAN process and the customers sequential process. Photographs of the respective outputs are also shown. Note that the noise was removed, but contrast was reduced somewhat in the process. The contrast could be restored by additional processing, however, this was beyond the scope of the contract.

Ease of programming is an important consideration for any machine. There are those who believe programming for a parallel machine is

NOISEY INPUT IMAGE



STARAN ARRAY PROCESSING

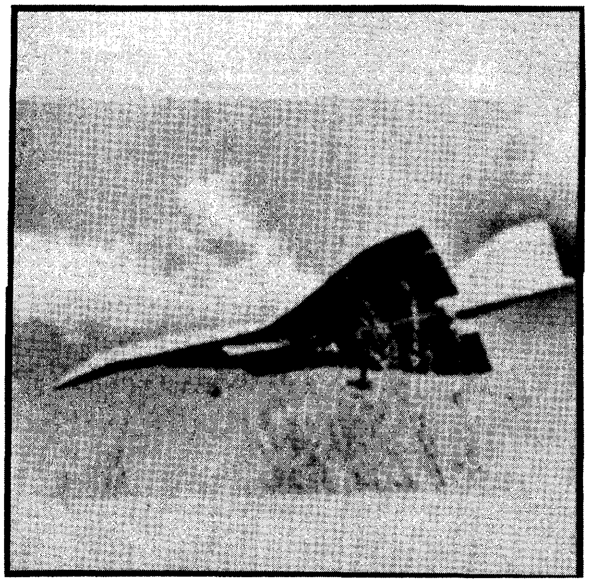
ESTIMATED TIME (SEC)	MEASURED TIME (SEC)
0.21	0.17

SEQUENTIAL MACHINE PROCESSING

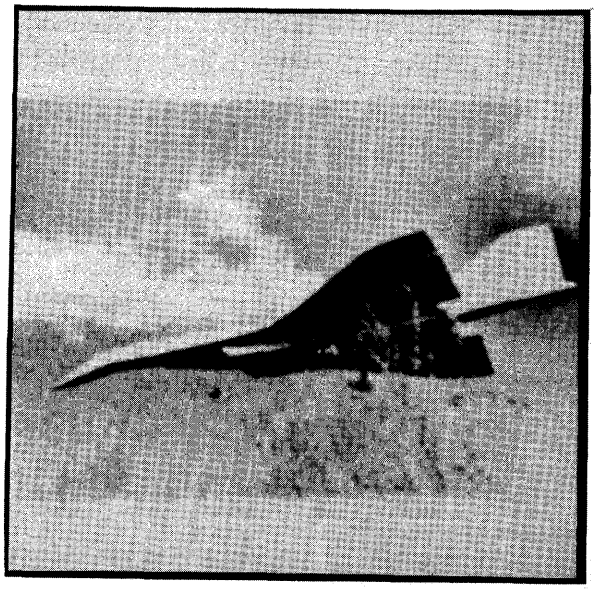
MEASURED CPU TIME (SEC)

IBM 360/195	IBM 360/65
8.04	60.0

OUTPUT IMAGES (NOISE REMOVED)



GAC - PROCESSED OUTPUT IMAGE



CUSTOMER - PROCESSED OUTPUT IMAGE

Figure 1 - Image Averaging Process for Validation - STARAN vs Sequential Machines

more complex than for a sequential machine. There are others who believe programming for a parallel machine like STARAN is not more difficult, just an exciting new way of thinking. This idea is reinforced by the results of the validation phase of this study. APPLE (Associative Processor Programming LanguagE) was used to program STARAN for the validation problem. Programming rate is one measure of programming complexity (or simplicity). The GAC programmer achieved a rate of about 9 instructions per hour for just the STARAN coding process. If the total programming effort (i.e., system analysis, coding, debugging and documentation) is used to compute the rate, the result is about 4 instructions per hour. Note that the programming rates discussed above are rates for only one isolated problem. However, they are considerably higher than the programming rates generally encountered in industry for assembly language programming for sequential machines.

3. RECOMMENDATIONS

The ultimate question must be answered is; will the integration of STARAN into the customer computer facility increase the cost-effectiveness of that facility? This study provides strong indications that the performance (a major factor of effectiveness) of the customer computer facility can be greatly improved by the addition of a STARAN array processor. The scope of this study, however, is not sufficient to provide all the necessary information to answer the total question. Therefore, this contractor recommends that a direct follow-on effort be carried out to further quantify the effectiveness as well as the cost factors involved in the addition of a STARAN array processor to the customer's computer facility. The items listed below comprise the recommended follow-on effort and, in general, should be undertaken in the order indicated:

1) Analyze additional application problems and problem types to increase spectrum of applicability and to achieve a better cross-section of the customer's computer facility workload.

2) Analyze more aspects, and in more depth, of some of the more complex problems already considered in the current study, specifically:

    (a) Error Correction
    (b) Linear Programming package

3) Validate additional problems on STARAN

    (a) Locally, by GAC programmers
        - Recommended problem: Error Correction

    (b) Remote Terminal by customer agency programmers
        - Recommended problem: Linear Programming package

4) STARAN hardware configuration/integration/cost study to determine the optimum methods for integrating STARAN into the customer's facility, the recommended STARAN hardware options and the associated costs.

5) STARAN software integration/cost study to determine the optimum manner for integrating STARAN software into the customer's facility and the associated costs.

SECTION II - TECHNICAL SYNOPSIS

1. INTRODUCTION

This study is primarily dedicated to the investigation of the applicability of the STARAN array processor to the solution of a group of data processing and computational problems typical of those currently being run at the customer's computer facility.

In order to understand the analyses in the items to follow, the reader should have a basic understanding of the STARAN system organization and how it differs with conventional digital computer organizations.

A conventional computer has one central arithmetic PE (Processing Element) (see Figure 2A). It can perform only one arithmetic operation - say add or multiply - on only one data pair at a time.

STARAN has an array of PE's, one for each word contained in memory (see Figure 2B). Each PE operates serially by bit on data in the memory word to which it is attached. The PE's simultaneously execute the instructions designated by the control unit. Therefore, in one instruction execution, the data in all or in selected words of memory are processed simultaneously by the PE's at each word.

A conventional computer has a location-addressed memory (Figure 2A). For each input item, it must search all contents of its data file one at a time until it finds the data it needs, or engage in complex software routines for storage or retrieval. Although file structures and programming techniques have lessened retrieval problems in the conventional processor, these techniques incur other disadvantages in cost and program execution time.

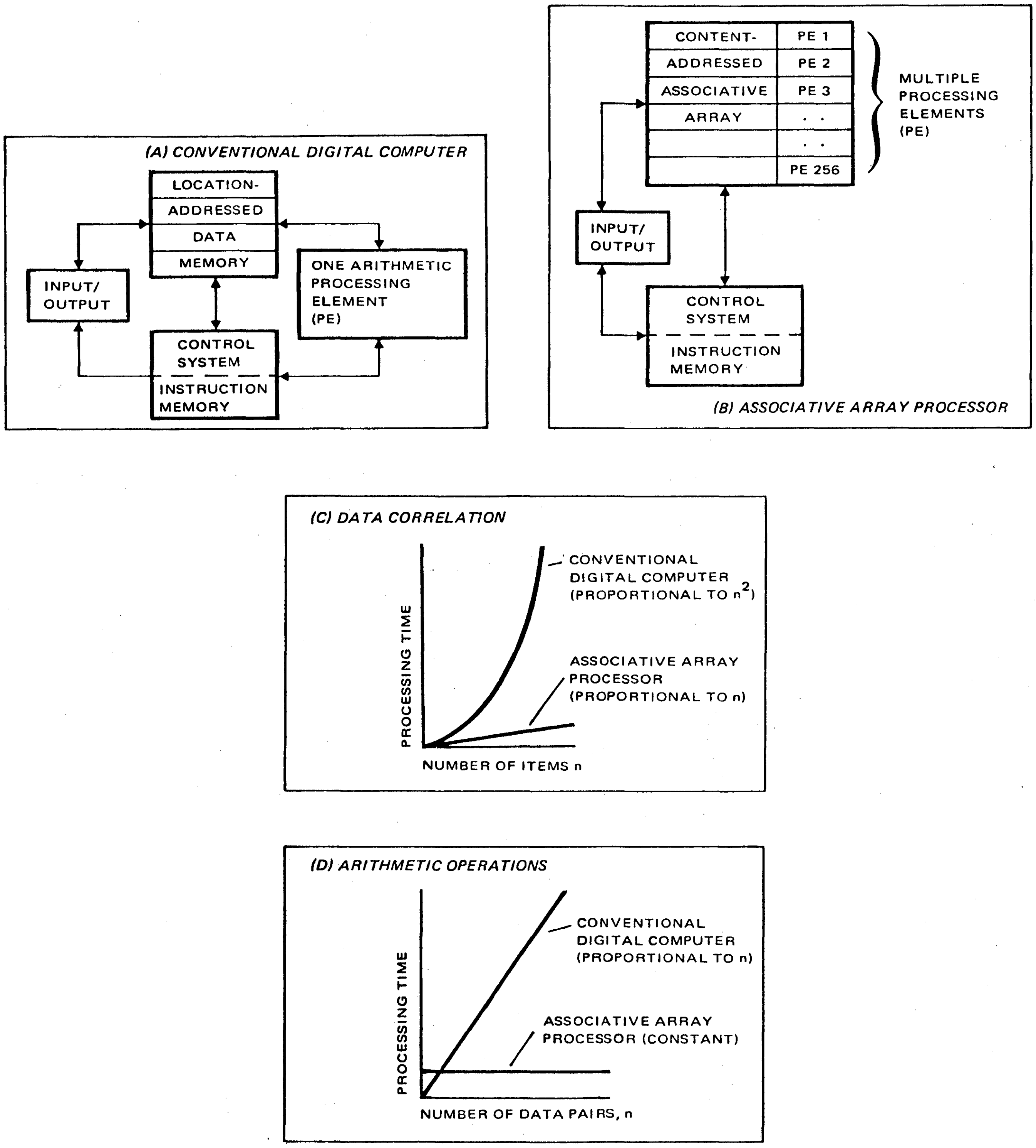

(A) CONVENTIONAL DIGITAL COMPUTER
LOCATION-ADDRESSED DATA MEMORY
INPUT/OUTPUT
ONE ARITHMETIC PROCESSING ELEMENT (PE)
CONTROL SYSTEM
INSTRUCTION MEMORY
CONTENT-ADDRESSED ASSOCIATIVE ARRAY
PE 1
PE 2
PE 3
PE 256
MULTIPLE PROCESSING ELEMENTS (PE)
INPUT/OUTPUT
CONTROL SYSTEM
INSTRUCTION MEMORY
(B) ASSOCIATIVE ARRAY PROCESSOR
(C) DATA CORRELATION
CONVENTIONAL DIGITAL COMPUTER (PROPORTIONAL TO n^2)
ASSOCIATIVE ARRAY PROCESSOR (PROPORTIONAL TO n)
PROCESSING TIME
NUMBER OF ITEMS n
(D) ARITHMETIC OPERATIONS
CONVENTIONAL DIGITAL COMPUTER (PROPORTIONAL TO n)
ASSOCIATIVE ARRAY PROCESSOR (CONSTANT)
PROCESSING TIME
NUMBER OF DATA PAIRS, n

Figure 2. STARAN Comparison with Conventional Computer

STARAN has a content-addressed (associative) memory. For each input data item, it can search all contents of its data file and identify all elements that meet the search criteria in a single memory access.

In a conventional computer, all processing and input/output (I/O) operations treat data in the word direction only.

In STARAN, I/O operations are in the word direction or, in some cases, in the bit direction; associative processing operations are in the bit direction. To accommodate accesses to either a word-slice or a bit-slice, each STARAN associative array module is provided with a multi-dimensional access (MDA) capability. Either a bit slice (bit n of any or all words) or an entire word-slice is available to the PE's or I/O channels.

For basic data correlation (see Figure 2C) a conventional computer, matching "n" input items with "n" items in memory, approaches a data-correlation (search) time proportional to n^2. STARAN compares each input item with all items in memory simultaneously; hence, the data-correlation time is proportional to the number of inputs and is independent of the number of data items in memory.

For Arithmetic Operations (see Figure 2D) with a conventional computer, the arithmetic operation time is proportional to the number of data pairs being processed. Because STARAN performs the same arithmetic operations on all data pairs simultaneously, the processing time is not affected by the number of data pairs being processed.

Four different STARAN system configurations were assumed in the study, depending upon the characteristics of each particular application problem (see Figure 3). The image processing problems

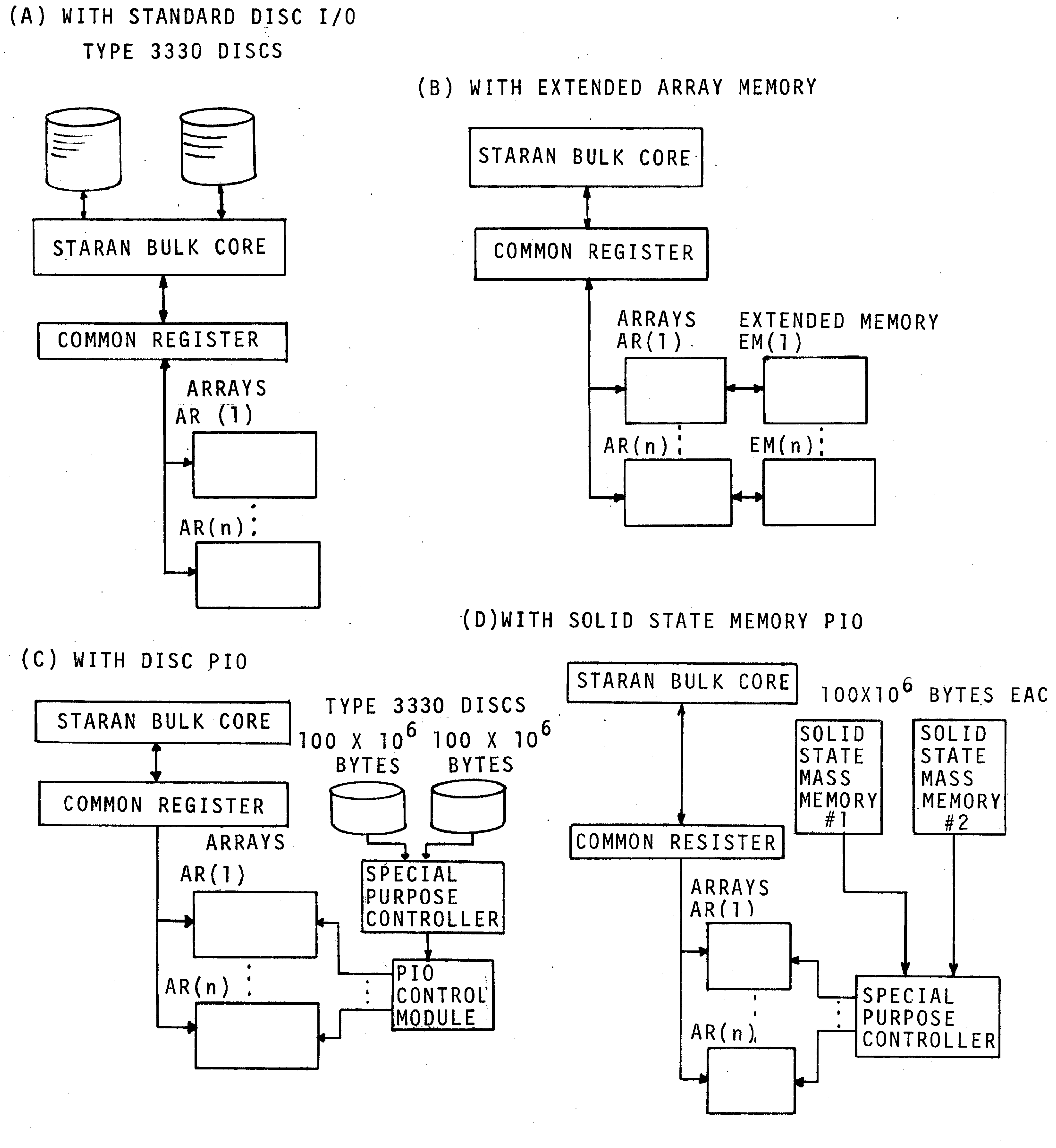


Figure 3. STARAN System Configurations

were analyzed assuming each of the two system configurations shown in Figure 3A and 3B. Only the system configuration of Figure 3B was assumed for the matrix problems, the antenna simulation problem and the error correction problem. This configuration provides the medium storage capacity and high bandwidth data transfer necessary because of the large amount of data movement associated with these problems. Each of the two system configurations shown in Figure 3C and 3D were assumed for the text search problem, because of the need for very large storage and high bandwidth data transfer.

Item 2 to follow in this section provides an abbreviated description of each of the application problems considered this study, along with the general approach for solution by STARAN and the resulting STARAN performance. Item 3 provides a summary description of the validation effort, which includes the programming and debugging of the image averaging problem along with the measured results.

For those readers who are interested in more detail, an in-depth description of each of the problem solutions as well as the validation effort, is provided in Volume II.

2. APPLICATION ANALYSIS

a. Image Processing

(1) General

This item contains a description of the application of STARAN to the general field of image processing. The problems considered include:

- Frequency Domain Filtering
- Convolution
- Edge Detection

- First Differences
- Magnification
- Statistics

Basically, the processing to be performed involves arithmetic manipulation of data contained in digital imagery in order to accomplish a particular form of image enhancement. Digital imagery is obtained by scanning a photographic negative and converting the analog intensities into digital values. In general, the image consists of a square matrix of digital image points called pixels. Each pixel consists of a binary number which represents the gray value of the particular image point. For most problems, depending on the processing and resolution required, an-8 bit number (indicating any one of 256 discrete gray levels) is sufficient.

For the problems investigated in this study, the digital image contains 512 x 512 pixels, or a total of 262,144 pixels. For this large quantity of data, it was mutually agreed to assume that the data would reside in some auxilliary storage medium. The basic processing approach would then be to read a segment of the image into the STARAN associative array(s), perform the required computations, output the results, and then repeat these operations until the entire image was processed.

Preliminary analysis of this approach and the image processing problems indicated that STARAN could provide a computational performance improvement proportional to the number of operations performed in parallel. Furthermore, analysis of the given image processing algorithms indicated that essentially the same operations are performed on each image pixel. Thus, for example, a two-array STARAN (512 words) can process an entire image row or column (512 pixels) in the same number of operations as for a single pixel.

This speed advantage, however, will not necessarily obtain in the overall problem execution time because of the necessity of image segment input-output (I/O).

It was decided, therefore, to examine two approaches to the I/O problem. In the first approach, the image data was assumed to reside on a conventional disc system, such as the IBM 3330 type. With a dual spindle unit, data would be read into STARAN using the Bulk Core memory as a buffer, and output results would be loaded into the second spindle. Overlapped I/O was not assumed, although this could be accomplished for some problems. Basically, then, this approach corresponds to the I/O technique that would be employed by a conventional sequential computer.

In the second approach, advantage was taken of the STARAN's parallel I/O (PIO) capability. The architecture of the associative arrays allows an entire word slice or bit slice (256 bits per array) to be loaded or unloaded in parallel. Connected to each array is a solid state storage area called Extended Memory (EM) which can hold an entire image. This approach allows an extremely high bandwidth I/O interface.

Finally, consideration was given to a means of loading and unloading the image storage medium. The usual origin of an image to be processed is a display system. This display system employed by the customer and most industry users is configured in such a way that the time to transfer a complete image to an external device is relatively large, and may even exceed the processing time.

Analysis of this system resulted in a conceptual design of a technique that employs a minimal amount of off-the-shelf hardware and the EM architecture of STARAN to effect a transfer speed of up to two orders of magnitude faster than the current system. This technique would allow all the image processing problems to be executed in a truly iteractive fashion (i.e., on the order of 5 seconds or less). Details of this technique are described in Volume II of this report.

(2) Frequency Domain Filtering

Frequency domain filtering is an image enhancement technique whereby a digital space domain image is: 1) transformed into the frequency domain using a two dimensional Fast Fourier Transform (FFT), 2) multiplied by a frequency domain filter, and 3) transformed back into the space domain using a two dimensional inverse FFT.

With an FFT, only $N\text{Log}_2N$ steps of operations (where each operation step, in general, involves one complex multiplication and one complex addition) are required for an N-point input. STARAN, however, by virtue of its parallel arithmetic capability, can perform simultaneously the N steps per stage of the Log_2N stages required. Consequently it can perform an N point FFT in only Log_2N steps of operations. This same speed advantage can be obtained for the inverse FFT. For the filter multiplication STARAN requires only $\frac{n}{2}$ steps as opposed to $\frac{N^2}{2}$ steps required for a conventional sequential computer. Thus, for the computational aspects of frequency domain filtering, STARAN offers a potential speed increase by a factor of N. The actual speed advantage, however, will be dependent on the STARAN architecture, particularly in terms of input-output (I/O).

The following is a simplified version of the major operations involved in frequency domain filtering:

. Input space domain image
. Perform 1st dimension FFT on each row of image and store results

- Perform 2nd dimension FFT on each column of above data and store results
- Input filter data, multiply times above data and store results
- Perform 1st dimension inverse FFT on each column of filtered data and store results.
- Perform 2nd dimension inverse FFT on each row of above data
- Output filtered space domain image.

It should be noted that, in general, each of the steps above involves an I/O or data transfer operation. This is due to the fact that: 1) the STARAN associative array can only hold a segment of the image at a time, and 2) the two-dimensional aspect of the problem requires a corner-turning operation, ie., the 1st dimension FFT operates on entire rows of the image and the 2nd dimension FFT operates on the n^{th} values of all rows (columns).

For frequency domain filtering, as well as most of the other image processing problems, two approaches to the problem solution were considered. In the first approach, the image was to reside on a 3330 type disc system. Thus, the STARAN I/O operations would be essentially the same as those for a sequential computer. Basically, the I/O involves reading a record from the disc, buffering through the STARAN Bulk Core, and transfer to the associative array (and the reverse for output). In the second approach, use was made of the STARAN's parallel I/O (PIO) capability. With this architecture, the image was to reside in what is called extended memory (EM). Extended memory consists of random access solid stage storage connected to the associative array in such a fashion so that one bit of an entire image row or column can be read into or out of the array in parallel. In the first approach, the I/O time is greater than the computational time; in the second approach the I/O time is a fraction of the computational time.

The estimated STARAN execution time for frequency domain filtering is shown below.

Major computations	3.4 sec
Input/output	
Disc to Bulk Core	4.5 sec
Bulk Core to Array	1.5 sec
Array to Bulk Core	1.1 sec
Bulk Core to disc	4.4 sec
Total	14.9 sec

These execution times were based on a two-array STARAN using the disc I/O approach. The I/O time includes such factors as rotational latency, head movement, etc. An overhead factor of 25 percent was included in order to allow for housekeeping operations.

For the extended memory/parallel I/O approach the same computational time was used and extended for operation with four arrays. The use of four arrays allows 2N steps of the FFT to be performed in parallel and so the computational time is halved. The STARAN execution time for this approach is shown below:

	Computations	I/O	Total
2 Arrays	3.4 sec	0.6 sec	3.46 sec
4 Arrays	1.7 sec	0.05 sec	1.75 sec

For solution by sequential computers, the following times were supplied by the customer:

IBM 360/195	39 sec (CPU time)
IBM 360/65	378 sec (CPU time)
HP3000	1552 sec (Total time)

Note that for the IBM computers, no I/O time was supplied. These numbers must be compared to the STARAN computation time and not the total time which includes I/O.

It can be seen that STARAN offers a substantial speed improvement, even over very large powerful sequential computers.

(3) Convolution

Convolution is an image enhancement technique that obtains similar results as frequency domain filtering but differs in that the computations are performed in the space domain. Basically, convolution involves the modification of each image pixel as a function of the sum of the products of its near neighbors and an a'priori weighting matrix.

Since the digital image consists of an matrix of pixels, to perform convolution on each pixel would require a time proportional to N^2 for a sequential computer. In the STARAN solution approach, an entire column (N pixels) of the image can be processed, in parallel, so that the execution time is proportional to N.

The solution for one pixel of the image may be visualized by overlaying the original image, pixel-by-pixel with the weighting matrix, where the central value of the weighting matrix is aligned with the image pixel of interest. Now each of the weights and its corresponding image pixel are multiplied. The sum of the resulting products then becomes the new value of the pixel of interest.

The STARAN solution may be viewed as overlaying the weighting matrix on all original image pixels in a column simultaneously. The product of the first weight and its corresponding image column is stored in a scratch field and shifted up or down a distance corresponding to the distance between the center of the matrix and the weight under consideration. The result is then accumulated into a second scratch field. This process is repeated for all values of the weighting matrix. At this point the second scratch field contains the new values for the first column of the image. Next, the weighting matrix is effectively shifted one column to the right and the previous process is repeated, except that the results are accumulated in a third scratch field. After the center column of the weighting matrix is processed, the first column of the original image emerges from the overlay and no longer participates in the computation. The original image column values can now be replaced by the new values stored in the second scratch field. From this point on, each time a column is processed, a scratch field is released, and new values can be stored in place of the original values.

The I/O requirements for this problem are relatively simple since only the original image and the weighting matrix are read in, the output is simply the new image. The execution times including overhead for a two array STARAN solution to convolution are shown below:

Major computations	2.4 sec
Input/Output	
Disc to Bulk Core	1.1 sec
Bulk Core to Array	0.4 sec
Array to Bulk Core	0.4 sec
Bulk Core to Disc	1.1 sec
Total	5.4 sec

By using the EM/PIO architecture, the I/O time could be reduced to .008 sec so that the total execution time is essentially the STARAN computation time.

In comparison, the following execution times for sequential computers are given below:

IBM 360/195	62 sec	(CPU time)
IBM 360/65	529 sec	(CPU time)

(4) <u>Edge Detection</u>

Edge detection is a technique whereby edges of objects in an image are enhanced, while other aspects of the image are subdued. Basically, this technique intensifies edges by calculating the average gray level differences of two neighborhoods on opposite sides of the pixel of interest (in each of four directions) and forms the product of these differences.

For this problem, three rectangular neighborhood sizes are employed, with a common height of three pixels and a width of 2^n $(n = 0,1,2)$ pixels. A difference in average gray level of each size neighborhood on each side of the pixel of interest is calculated. The final edge value is calculated as the product of the three intermediate edge values. Furthermore, this process is repeated in four directions, horizontal, vertical, and the two diagonal directions. Then a new image is formed by taking the maximum value of the directional set for each pixel of the image.

The following is a simplified version of the basic algorithm steps required for calculating just the vertical edge value.

- Calculate average gray level in a 3 x 1 pixel neighborhood to the left of the pixel of interest,

$$d_\ell = \frac{g_1 + g_2 + g_3}{3}$$

- Calculate average gray level in a 3x 1 pixel neighborhood to the right of pixel of interest, d_r
- Calculate $D_1 = |d_\ell - d_r|$
- Calculate d_ℓ for 3 X 2 neighborhood
- Calculate d_r for 3 X 2 neighborhood
- Calculate $D_2 = |d_\ell - d_r|$
- Calculate d_ℓ for 3 X 4 neighborhood
- Calculate d_r for 3 X 4 neighborhood
- Calculate $D_4 = |d_\ell - d_r|$
- Calculate Vertical Edge Value $D_v = D_1 \cdot D_2 \cdot D_4$

The calculations required for the other three directional edge values are similar.

In the STARAN solution to edge detection, the number of calculations can be greatly reduced by taking advantage of the parallel arithmetic capability. Consider, for example, the calculation of horizontal edge values. In this case, we are concerned with the average gray level differences between neighborhoods above and below the pixel of interest. However, the first edge value (3 pixels horizontal, 1 pixel vertical) for all pixels in an image column can be calculated simultaneously. Furthermore, since the sum of the three horizontal pixels was calculated for an entire image column, we can use this information in the calculation of the second and third edge values. Finally, we can restructure the algorithm slightly by postponing the averaging process for each neighborhood (a divide by the area of the neighborhood) until the final multiplication of the three intermediate edge values.

Using similar techniques for the other directional edge values, the following results were obtained (including overhead):

Major computations	2.6 sec
Input/output	
Disc to Bulk Core	4.2 sec
Bulk Core to Array	0.2 sec
Array to Bulk Core	0.6 sec
Bulk Core to Disc	2.6 sec
Total	10.2 sec

With the EM/PIO architecture the I/O time was considerably reduced as shown below:

	Computations	I/O	Total
2 Arrays	2.6 sec	0.0051 sec	2.6051 sec
4 Arrays	1.3 sec	0.0026 sec	1.3026 sec

The execution times for the conventional computers were:

IBM 360/195	133 sec (CPU time)
IBM 360/65	1145 sec (CPU time)

(5) First Differences

First differences is another edge detection technique whereby the difference is calculated between each digital image pixel and its immediate (first) neighbor in the horizontal, vertical, and two diagonal directions.

Basically this technique requires that the image be shifted one pixel and then subtracted from the original image. In order to make the edge enhancement relatively independent of edge orientation, the process is performed in four directions; horizontal, vertical, and the two diagonal directions. The maximum difference value of the four orientations is then used to create a new image where edges are intensified and other aspects of the image are subdued.

Execution of the first difference algorithm is extremely simple using the STARAN associative array. Basically, the array is loaded with a segment of the image on a column basis. For the vertical edge difference, we need only subtract each column from its immediate neighbor column. Since the STARAN can operate simultaneously on an entire column, only one arithmetic operation is required to generate all the vertical first differences. In the horizontal case, the column to be subtracted is first read out into the response store and then shifted one location. Then, the subtraction is performed for all values in a column. A similar operation takes place for the diagonal differences. After each difference value is calculated, it is compared to the previous value and only the maximum value is retained. The final image output will then be composed of the maximum of the four calculated difference values.

The total number of operations per image column is therefore: 4 subtracts, 1 shift, and 3 compares. The estimated execution time (including overhead) for this problem using a single array is shown below:

Major computations	0.1 sec
Input/output	
Disc to Bulk Core	1.1 sec
Bulk Core to Array	0.2 sec
Array to Bulk Core	0.4 sec
Bulk Core to Disc	0.8 sec
Total	2.6 sec

With the EM/PIO architecture, the I/O time is substantially reduced as shown below:

	Major computation	I/O	Total
1 Array	0.10	0.0102	0.1102
2 Arrays	0.05	0.0551	0.0051
3 Arrays	0.03	0.0026	0.0326

The reported execution time for sequential computers for the first difference problem is:

IBM 360/195	136 sec (CPU time)
IBM 360/65	996 sec (CPU time)

(6) Magnification

Magnification is a process whereby a digital image is expanded by a factor of eight in both the horizontal and vertical directions. The result is an enlarged image containing 64 times as many pixels.

The approach is to linearly interpolate between adjacent image pixels of the original image. For example, given a 2 X 2 matrix of image pixels, magnification will result in an 9 X 9 matrix where the corner pixels are defined by the original 4 pixels. Each of the new pixels within the matrix is calculated by adding 1/8 of the difference of the edge pixels to the previous pixel.

In the STARAN approach a segment of the original image is loaded into the associative array. This data is stored at every eighth location in both the horizontal and vertical directions in order to make space available for the interpolated data to be calculated.

In order to fill the interior matrix, the boundary (edge) pixels must first be calculated. The between-word arithmetic capability of STARAN allows the difference between all vertical edge pixels to be calculated in only one subtract operation. Then, in seven add operations, all the interpolated values between all the edge pixels in one column can be calculated. This process is repeated then for a second image-column. At this point all the vertical interpolated values have been calculated for two original image columns.

Next, the interpolated values for the horizontal direction are calculated. First, in one subtract, the differences between all vertical values previously obtained are calculated. Then in seven adds, all the interpolated values between columns are calculated.

This process is then repeated for all image columns until the complete magnified image is obtained. The execution time is thus proportional to 2 subtracts per column and 14 adds per column. The actual number of arithmetic operations is dependent on the number of arrays employed.

Shown below are the estimated execution times including overhead) for image magnification assuming a single array STARAN:

Major computations	2.4 sec
Input/output	
Disc to Bulk Core	3.5 sec
Bulk Core to Array	0.4 sec
Array to Bulk Core	12.2 sec
Bulk Core to Disc	9.1 sec
Total	27.6 sec

It can be seen from these results that because of the quantity of output data, the I/O time is an order of magnitude greater than the major computation time. This situation can be avoided by using the EM/PIO architecture with the following results:

	Major computations	I/O	Total
1 Array	2.4 sec	0.33 sec	2.73 sec
2 Arrays	1.2 sec	0.17 sec	1.37 sec
4 Arrays	0.6 sec	0.08 sec	0.68 sec

In comparison, the following sequential computer execution times were obtained:

IBM 360/195	769 sec (CPU time)
IBM 360/65	7290 sec (CPU time)

(7) <u>Statistics</u>

Statistics, as applied to digital imagery involves the accumulation and computation of statistical data on any given image. For the specific problem considered here, the following outputs are required:

- Histogram data (i.e., pixel population count for each gray level)
- Maximum gray level value
- Minimum gray level value
- Mean gray level value
- Standard deviation

Analysis of the above operations indicates that the required computations are essentially sequential in nature, so the parallel computation ability of STARAN cannot be fully exploited. For example, the calculation of histogram data involves an accumulation of the number of image pixels with a particular gray level value. Thus, each pixel must be individually examined. In the computation of maximum and minimum gray level values, however, the parallel search capability of the STARAN can be utilized to provide a significant speed improvement.

The basic approach taken to generate the histogram is to load a field of all words in the associative array with the 256 unique gray values. The image data is then read into the common register and compared to the gray level field. For each image pixel only one gray level location will match exactly. When this occurs, a count field in the same array location is incremented. When all image pixels have been processed, the count field contains the required histogram information.

The mean gray value is calculated by successively shifting and adding the count field within the array while counting the number of shifts in an index register. When the accumulated sum in location zero of the array exceeds half the total number of image pixels, the index register contains the mean gray level.

Maximum and minimum gray level values are computed by first searching the count field to find zero values, if any. The response of this search is then used as a mask for searching the gray level field. Thus, any location with a zero count is not considered in the second search operation.

The standard deviation is a straightforward calculation that uses the previously obtained population counts and mean gray value.

The estimated STARAN execution times ((with overhead) for the statistics problem using a single array are given below:

Major computations	3.9 sec
Array loading	0.1 sec
Disc I/O	1.1 sec
Total	5.1 sec

Due to the sequential nature of this problem no estimate was prepared for execution using the extended memory. However, based on the results of previous estimates, the problem should run approximately 1 second faster than the disc approach.

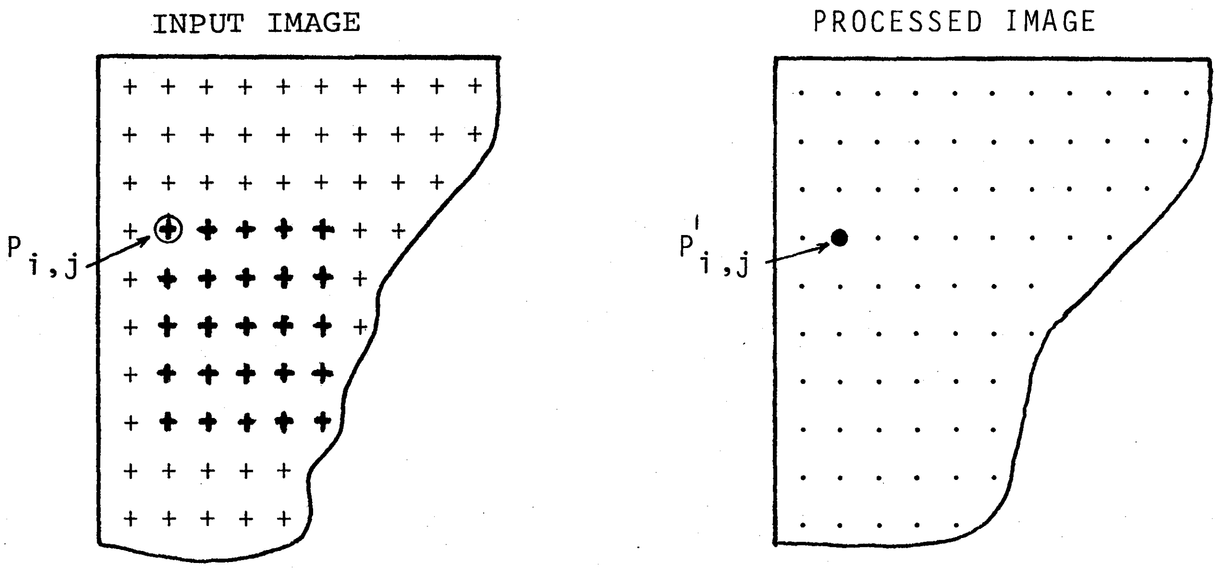
The reported sequential computer execution times are:

IBM 360/195	8 sec (CPU time)
IBM 360/65	66 sec (CPU time)
HP 3000	48 sec (total time)

(8) <u>Image Averaging</u>

Image Averaging is a technique for enhancing an image by reducing image noise. Noise in an image can be introduced from various sources. One source of noise could be the equipment which digitizes the image.

This image averaging technique requires that a new gray value be computed for each specified pixel of the input image. This is done by finding the average gray value of all the image pixels within the boundaries of a specified matrix cell such that the image pixel of interest is located in the top left corner. This process is shown by sketch and equation in Figure 4 below.



$$P'_{i,j} = \frac{\left(\sum_{m=0}^{M-1} \sum_{n=0}^{N-1} P_{i+m,j+n}\right)}{MN}$$

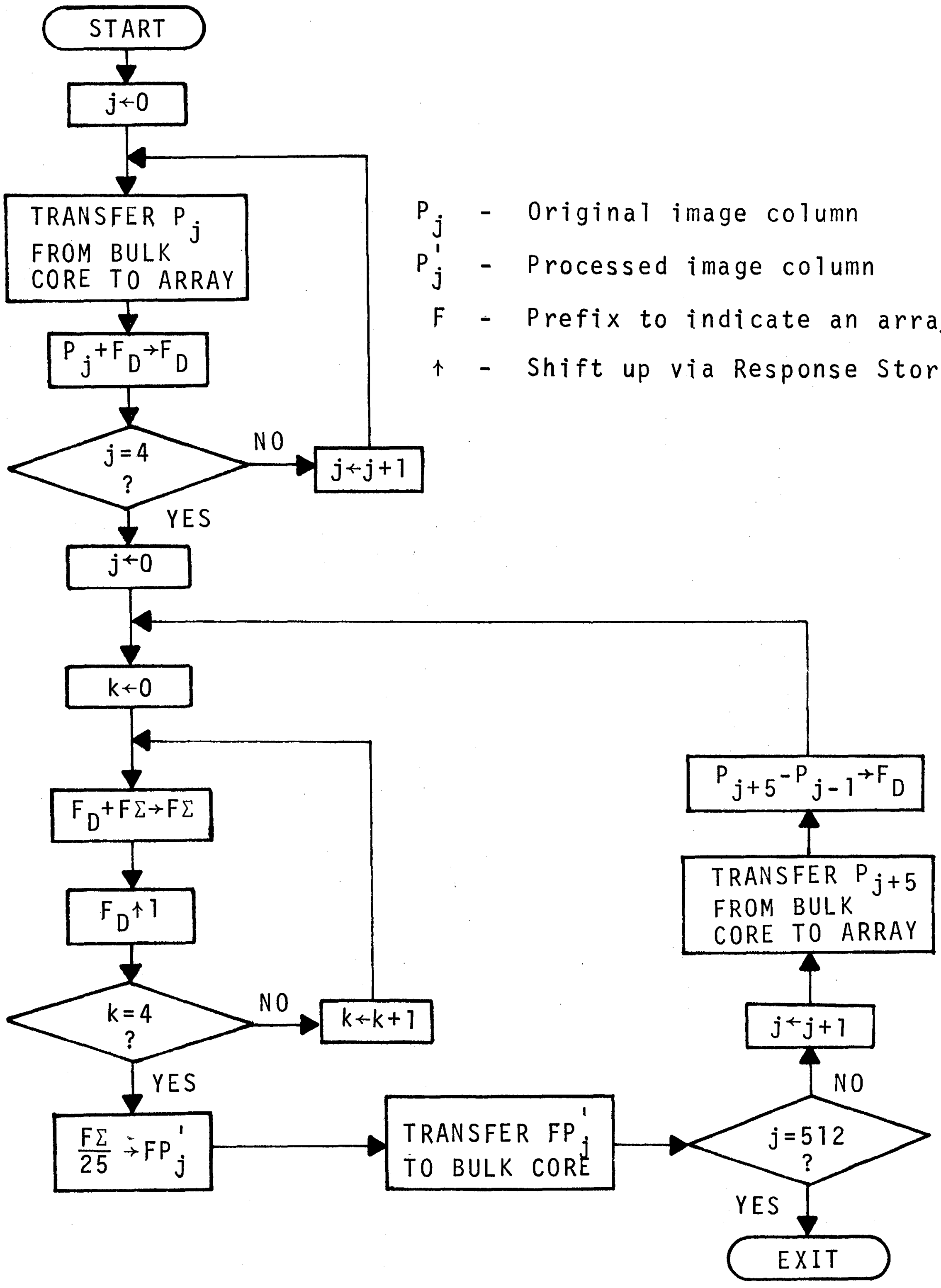
Figure 4 - Image Averaging Problem Description

The image averaging problem requires the solution of the equation indicated in Figure 4. In a sequential machine, this equation must be solved for every pixel of the image. With the application of the STARAN array processor, the equation needs to be solved only once for each complete column (or line) of the image. The STARAN system configurations shown in Figure 3A utilized in the solution of this problem. The size of the input image is assumed to be 512 pixels by 512 pixels with 8 bits to describe gray level for each pixel. Two 256 bit X 256 bit associative arrays are assumed within the STARAN system configuration. For the purpose of a timing estimate, the size of the averaging matrix cell will be fixed at 5 X 5 pixels, i.e., M = 5 and N = 5 (see Figure 4), for a total of 25 pixels.

The simplified flow chart shown in Figure 5 shows the technique used in applying STARAN to the solution of the image averaging problem. This flow chart does not show the transfer of the image between the disc and STARAN bulk core, but, it does show transfer between STARAN bulk core and the associative arrays along with the actual array processing.

The array processing time for the image averaging problem was estimated at 0.165 seconds based on a 2 - array system. This estimate is developed, in detail, in Appendix N of volume II. Applying the usual technique of adding 25 percent for overhead, brings the estimated time up to 0.21 seconds. The I/O times are essentially equal to those developed for convolution (item (3) above. The time for the total process, including overhead and based on the use of 2 arrays and conventional disc I/O, is developed below:

Array computations	0.21 sec
Input/Output	
Disc to Bulk Core	1.1 sec
Bulk Core to Array	0.4 sec
Array to Bulk Core	0.4 sec
Bulk Core to Disc	1.1 sec
Total	3.2 sec


START
$j \leftarrow 0$
TRANSFER P_j FROM BULK CORE TO ARRAY
$P_j + F_D \rightarrow F_D$
$j=4$?
NO
$j \leftarrow j+1$
YES
$j \leftarrow 0$
$k \leftarrow 0$
$F_D + F\Sigma \rightarrow F\Sigma$
$F_D \uparrow 1$
$k=4$?
NO
$k \leftarrow k+1$
YES
$\frac{F\Sigma}{25} \rightarrow FP'_j$
TRANSFER FP'_j TO BULK CORE
$j=512$?
NO
YES
$j \leftarrow j+1$
TRANSFER P_{j+5} FROM BULK CORE TO ARRAY
$P_{j+5} - P_{j-1} \rightarrow F_D$
EXIT

P_j	-	Original image column
P'_j	-	Processed image column
F	-	Prefix to indicate an array field
$\uparrow$	-	Shift up via Response Store

Figure 5. Simplified Flowchart for Image Averaging

By using extended memory parallel I/O (EM/PIO) architecture (See Figure 3B) the I/O time can be reduced to 0.008 sec for a 2 array system, or 0.004 sec for a 4-array system. This results in total processing times of 0.22 sec and 0.11 sec, respectively.

For comparison, the following execution times for sequential computers are given below:

IBM 360/195	8.04 sec (CPU time)
IBM 360/65	60.0 sec (CPU time)

The above times are actual measured times supplied by the customer agency. Since they represent only CPU times (not I/O), they are directly comparable to the STARAN times for performing only the array computations. Therefore, STARAN provides the following performation advantages over the fastest sequential machine:

2 array STARAN vs IBM 360/195	=	8.05/0.21	=	38
4 array STARAN vs IBM 360/195	=	8.05/0.105	=	77

b. Text Search

(1) Problem Description

The text search problem requires searching a very large textual data base in order to locate the documents which contain a particular combination of query words. The query words do not have to appear in any particular order in a document. The textual data base to be searched is made up of about 105,000 documents, each document averaging a little over 300 words and 6 characters per word for a total of 200×10^6 characters (or bytes). For those documents that meet the conditions imposed by the query statement, the minimum required response is readout of the document number.

The query statement is made up of a combination of logical unions and intersections of single words, and can be described by the following expression:

$$(Q_1 \vee Q_2 \vee Q_3) \wedge (Q_4 \vee Q_5 \vee Q_6) \wedge \ldots (Q_{13} \vee Q_{14} \vee Q_{15})$$

The groups of words inside each set of parenthesis are usually synonyms or at least similar words. In order to meet fully the search requirements, a given document must contain at least 1 query word from each and every intersecting (parenthetical) group of query words. Since more than one document could meet the requirements of the query statement, the entire data base must be searched. The design goal for solution of this problem is 1 second.

(2) Problem Solution

The major time-consuming aspects of this problem are: 1) I/O time, and 2) search time. Other aspects of the problem are essentially negligible from the standpoint of time. While any machine could solve this problem, most could not approach the design goal of 1 second.

In applying STARAN to this problem, two approaches were considered based on the two system configurations shown in Figure 3 views C and D. The fastest method (Figure 3D) is based on the specifications of the solid state mass memory to be developed by RADC in the near future. This approach results in a system which nearly meets the design goal.

The alternate method (Figure 3 view C) is based on the use of a conventional disc memory, specifically the type 3330 disc system. While this method results in longer response time, it provides a nearer-term alternative which is still fast in comparison with existing systems which are based on sequential processors.

Both STARAN approaches are based on the transfer of the textual data base from the mass memory media to the arrays in the same format via a special purpose controller. The controller must have the ability to sense word beginnings. The text data is stored in the arrays horizontally in three columns. The text words are left justified in these columns to a specific bit position. Each array provides storage for 2 documents. Both approaches to the problem utilize multiple arrays and all arrays used must conform to the same format since the search is performed simultaneously in all arrays. The same search strategy is used in both STARAN approaches and is shown in the simplified flow chart of Figure 6. The query statement is stored in STARAN bulk core and is made up of query words which are transferred sequentially to the common register, from which a search is performed by an exact match with all text words stored in the arrays. The responses to the query word searches are stored in designated tag columns. After all query words have been used to search the text stored in the arrays, the tag columns are evaluated in accordance with the logical operators that appear in the query statement.

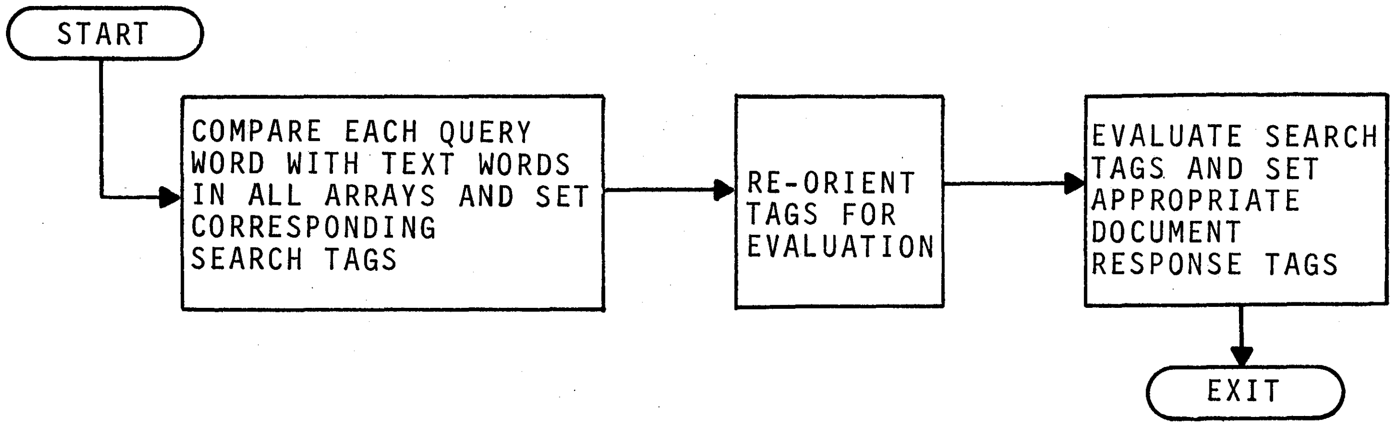

START
COMPARE EACH QUERY WORD WITH TEXT WORDS IN ALL ARRAYS AND SET CORRESPONDING SEARCH TAGS
RE-ORIENT TAGS FOR EVALUATION
EVALUATE SEARCH TAGS AND SET APPROPRIATE DOCUMENT RESPONSE TAGS
EXIT

Figure 6. Simplified Search Routine

Finally, tags are set which correspond to the documents that meet the query-search criteria. The tags are used to direct the readout of either the responding document number(s) or the complete document(s).

(3) Execution Time Estimate and Comparison

The solution time estimates for the text search problem are given in Table III. For the STARAN approach using disc storage media four arrays were utilized. Since two 3330 disc spindles are required to store the entire data base, two of the arrays are loaded simultaneously while the other two arrays are being searched. Since the search time is less than the disc transfer time, the search time is completely overlapped by the transfer time and therefore only the transfer time contributes to the total solution time.

TABLE III - TEXT SEARCH TIMING SUMMARY AND COMPARISON

	STARAN ARRAY PROCESSOR				SEQUENTIAL MACHINE	
	SS MEM + APPROACH - No. OF ARRAYS			DISC APPROACH	IBM 370/145 APPROACH (ESTIMATED)	GE SPECIAL PURPOSE SYSTEM (REPORTED)
FUNCTION	32	16	8			
LOAD TIME	0.39 SEC	0.39 SEC	0.39 SEC	128 SEC	128 SEC	N/A SEC
SEARCH TIME	0.98 SEC	1.97 SEC	3.94 SEC	OVER-LAPPED	1000 SEC	NA
DOCUMENT READOUT TIME	N*	N*	N*	N*	N*	NA
TOTAL TIME	1.37 SEC	2.36 SEC	4.33 SEC	128 SEC	1128 SEC	3200 SEC

+ SS MEM = Solid State Memory
*Negligible
NA = Not available

For the STARAN approach using solid state mass storage media, two mass memory modules are required to store the entire data base. The specifications on the proposed RADC memory indicate each memory module will have a capacity of 100×10^6 bytes (characters) and 1024 access channels. Therefore, two modules operating concurrently could load a maximum of 8 arrays simultaneously. However, the more arrays that are assumed, the fewer searches are required. The problem is timed with three different array configurations, 8, 16 and 32 arrays. STARAN can support a maximum of 32 arrays, and this configuration comes reasonably close to the design goal solution time. If STARAN could contain the entire data base, only a single 600 μsec search would be required and the total solution time would essentially be equivalent to the array load time, or about 0.4 seconds.

Concerning the sequential machine timing in Table III, the GE special purpose system time was provided by the customer agency while the IBM 370/145 time was a conservative estimate by this contractor.

A more detailed discussion of this problem solution is given in Volume II of this report.

c. Solution of Linear Systems

(1) Problem Description

The problem of solving a linear system of equations is encountered in a wide range of applications involving numerical mathematics.

Consider the linear system:

$$\left.\begin{array}{ccccccc} a_n x_1 & + & a_{12} x_2 & + \ldots & a_{1n} x_n & = & b_1 \\ \vdots & & \vdots & & \vdots & & \vdots \\ a_{n1} x_1 & + & a_{n2} x_2 & + \ldots & a_{nn} x_n & = & b_n \end{array}\right\} \quad (1)$$

The problem posed here is to develop a method for solving one or more linear systems, similar to linear system (1) above using the basic Gaussian Elimination technique.

Gaussian Elimination and some of its variations are the most common methods used to solve a dense linear system. This popularity is based on stability and timing considerations. The superior speed of the Gaussian Elimination technique is due to the fact that it requires the fewest number of arithmetic operations among equation-oriented algorithms. Equation-oriented algorithms are those which if some coefficients of an equation are operated on, all coefficients of the equation are operated on in the same manner. As a result, bookkeeping of the operations is a simple matter.

(2) <u>Problem Solution</u>

The linear system (1) above is solved by Gaussian Elimination in two stages: 1) triangularization, and 2) back-substitution.

In the triangularization stage, x_1 is eliminated from all but the first equation, x_2 is eliminated from all but the first two equations, etc. In general, x_i is eliminated from all but the first i equations. This process is carried out until a linear system of the following form is reached:

$$\left.\begin{array}{rl} x_1 + e_{12} x_2 + e_{13} x_3 + \cdots + e_{1n} x_n & = f_1 \\ x_2 + e_{23} x_3 + \cdots + e_{2n} x_n & = f_2 \\ x_3 + \cdots + e_{3n} x_n & = f_3 \\ \vdots & \quad \vdots \\ x_n & = f_n \end{array}\right\} \qquad (2)$$

The back-substitution stage of Gaussian Elimination starts with the observation that x_n is known and that the $n-1^{st}$ equation contains only x_n and x_{n-1}. Thus, by multiplying the n^{th} equation by $e_{n-1,n}$ and subtracting the result from the $n-1^{st}$ equation, the variable x_{n-1} is found. When x_n and x_{n-1} have been found, the $n-2^{nd}$ equation can be used to find x_{n-2}. This process can be continued to find x_{n-3}, x_{n-4}, ... x_1.

The STARAN system configuration selected for solution of this problem is shown in Figure 3B. The number of arrays required depends on the specific storage scheme selected.

Several storage schemes and their associated algorithms for solution of the linear system were investigated during the study. The scheme that was selected required the use of a reasonable number of arrays (four) and used them efficiently (See Figure 7).

The associated algorithm employs the basic arithmetic operations (i.e., +, -, *, and ÷) plus data movement. These operations apply irrespective of the type of machine implementation. However, the data moves specified for the STARAN implementation includes those which are used to arrange the data to enhance the amount of parallelism within the problem solution. The effect is to reduce the number of arithmetic operations. Since the data-move operations are relatively fast, the overall result is an increase in solution speed. The solution algorithm is detailed in Appendix J. The operation counts for solution of a linear system in a sequential machine as well as the STARAN array processor is shown in Figure 8.

(3) Execution Time Estimate

Table IV presents the STARAN execution times for the cases, $n = 128$ and $n = 256$. The I/O category in the table consists of the

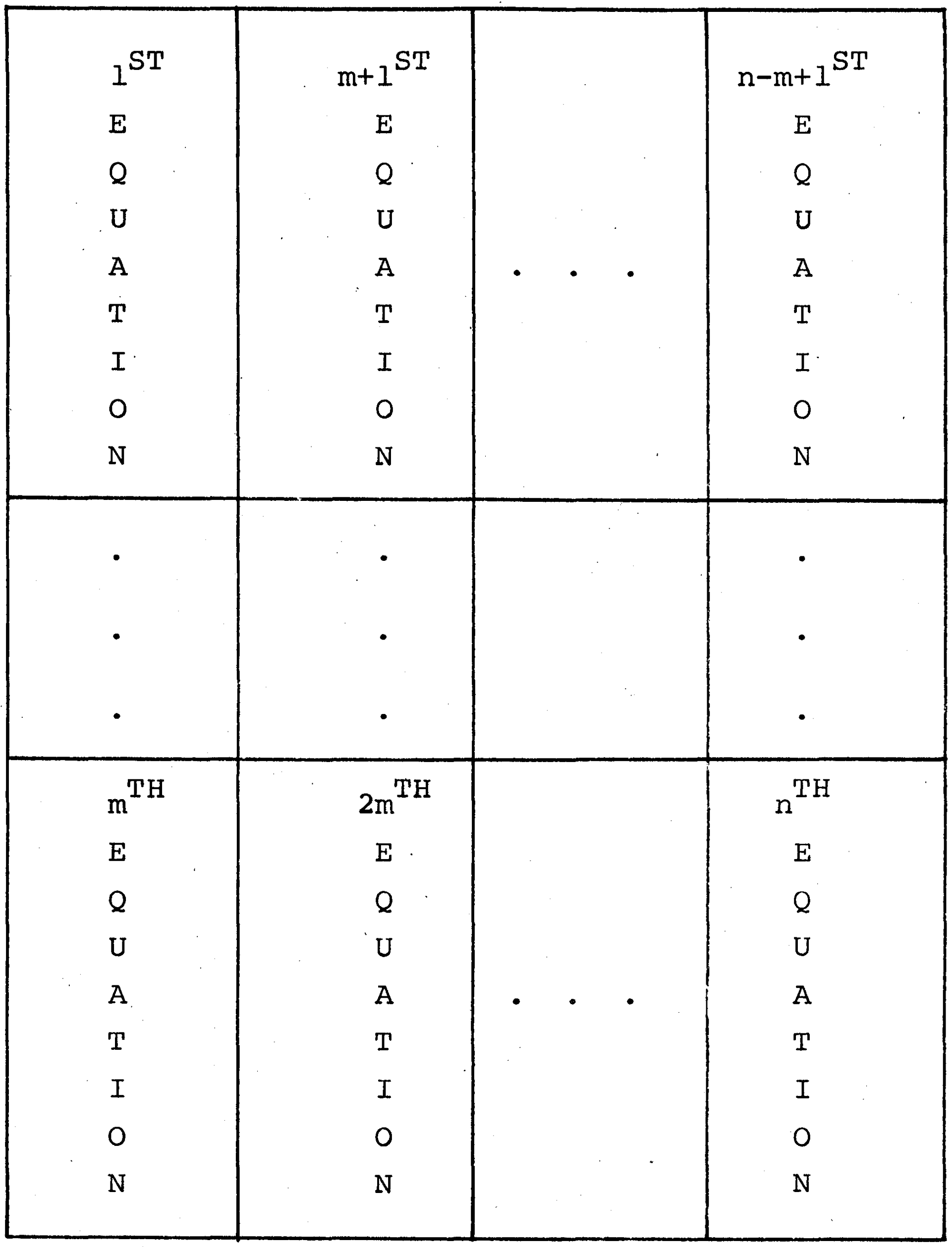


Figure 7 - Array Storage Scheme for Solution of a Linear System

Operation type	OPERATION COUNT	
	STARAN Processor	Sequential machine
+	$\left(\left\lceil \frac{n}{m} \right\rceil - 1\right) \left\lceil \log_2 n \right\rceil$	$(2n^3 + 3n^2 - 5n)/6$
-	$\frac{n}{2}\left(\left\lceil \frac{n}{m} \right\rceil + 3\right) - \left\lceil \frac{n}{m} \right\rceil - m$	
*	$\frac{n}{2}\left(\left\lceil \frac{n}{m} \right\rceil + 3\right) - \left\lceil \frac{n}{m} \right\rceil - m$	$(2n^3 + 3n^2 - 5n)/6$
÷	n	$(n^2 + n)/2$
Data Move	$\left\{\left\lceil \log_2 n \right\rceil\right\} \left\{ \frac{n}{2}\left\lceil \frac{n}{m} \right\rceil + 2n \left\lceil \log_2 n \right\rceil + \frac{n}{2} + \frac{n}{m}\right\}$	*Proportional to the total number of arithmetic operations.

*Traditionally not specified for sequential solutions.

Figure 8 - Operation Counts for Solution of a Linear System

TABLE IV - EXECUTION TIMES FOR SOLUTION OF A LINEAR SYSTEM

Variable type and precision	n	STARAN EXECUTION TIME (SECONDS)					
		Floating point software			Floating point hardware		
		Array processing	I/O	Total	Array processing	I/O	Total
Real-Single precision	128	1.510	0.044	1.554	0.382	0.044	0.426
	256	9.89	0.17	10.06	2.27	0.17	2.44
Real-Double precision	128	4.869	0.088	4.957	0.923	0.088	1.011
	256	32.06	0.34	32.40	5.38	0.34	5.71
Complex-Single precision	128	5.919	0.088	6.007	1.008	0.088	1.096
	256	37.22	0.34	37.56	6.06	0.34	6.40
Complex-Double precision	128	19.725	0.671	20.396	2.344	0.671	3.015
	256	123.31	3.87	127.18	13.65	3.87	17.52

IBM 360/65 - 135 sec (CPU time)

data transfer between the arrays and their respective extended memories. Floating point hardware is currently being developed for STARAN. Table IV provides execution times with and without the floating point hardware option. Measured sequential machine times were not available from the customer agency for this particular problem. In lieu of this, recent literature was searched for sequential machine timing data for the solution of linear systems. A document[a] was found which provided IBM 360/50 solution times for various size matrices. By extrapolation, the IBM 360/50 time was determined to be 407 seconds to solve a linear system for the case, n = 256. IBM personnel were consulted for the speed ratio between models 50 and 65 in the 360 series. They claimed that for arithmetic operations, the model 65 is faster by a factor of from 2.5 to 3. In order to be conservative the ratio of 3 was applied to arrive at 135 seconds for the IBM 360/65 to solve a linear system for the case, n = 256.

It should be noted that when stability is believed to be a problem (or when division by zero may occur), a pivoting strategy may be employed. This requires that additional operations be performed during the triangularization stage of the solution. A more detailed discussion of pivoting is included in Appendix J.

d. Matrix Inversion

(1) Problem Description

The problem of inverting a non-singular n X n matrix is frequently encountered in a variety of applications. Matrix inversion along with matrix/vector product (see item e) are both subfunctions of the antenna simulation problem which is discussed in item f.

[a]Parlett and Wang; The Influence of the Compiler on the Cost of Mathematic Software - In particular on the cost of Triangular Factorization; ACM-Toms, March 1975.

The specific requirement is to develop an algorithm for the STARAN parallel array processor, which employs direct methods for finding the inverse of a non-singular dense matrix in a finite number of steps. Provide counts of the various types of operations as well as an estimate of STARAN solution time for a large matrix. For comparison, include operation counts for the solution of the problem on a sequential machine.

(2) Problem Solution

(a) Sequential Solution - The standard direct method for inverting a non-singular n X n matrix A usually consist of first setting up an n X 2n matrix E, containing A in the leftmost n columns, and the identity matrix in the rightmost n columns. Then E is transformed via row operations, into a matrix whose leftmost n columns form the identity matrix. The rightmost n columns form the desired inverse of matrix A.

The choice of row operations can be based on the Gaussian Elimination (GE) technique or the Gauss-Jordan Reduction (GJ) technique. In the GE approach, row operations are chosen to triangularize the matrix in the left half of E. Then, row operations are applied which transforms the left half matrix of E into the identity matrix via back-substitution. In the GJ approach, the object of the row operations is to introduce zeros above and below the diagonal at each step. This technique is termed, diagonalization.

Table V shows the operation counts for both techniques (GE & GJ) are identical. The selection of technique, therefore, cannot be based on operation count. GJ requires less storage, but, GE has the better residue property.

TABLE V - ARITHMETIC OPERATION COUNTS FOR MATRIX INVERSION ON A SEQUENTIAL COMPUTER

Operation type	OPERATION COUNT	
	GE	GJ
-	$n^2(n-1)$	$n^2(n-1)$
*	$n^2(n-1)$	$n^2(n-1)$
÷	n^2	n^2

GE = Gaussian Elimination

GJ = Gauss-Jordan Reduction

(b) STARAN Solution - During the study, three STARAN array storage schemes were considered, each with its respective algorithm. For all three schemes, the concept of implicit storage was employed so that n words, rather than 2n words, was sufficient to store a row of the n X n matrix. Ususally, data is stored irrespective of its properties, i.e., data is usually explicitly stored. However, sometimes data has known values so that rather than storing this data, the algorithm is designed to assume that the data takes on these values, i.e., the data is implicitly stored. This principle of implicit storage is used in the STARAN solution of the matrix inversion problem.

Storage Scheme I stores one row of the matrix per field of the array. Storage Scheme II stores n rows per field, hence requires n^2 words and one field. Storage Scheme III stores m rows per field (where $1<m<n$), hence requires nm words and n/m fields. These three storage schemes correspond to the schemes considered in the solution of a linear system (see item c above). Storage Scheme III was chosen for use since it requires the use of a reasonable number of STARAN arrays and uses them efficiently. The STARAN system configuration shown in Figure 2B (with 4 arrays) was chosen to implement the solution of the problem. Application of the algorithm that corresponds to the Storage Scheme III results in the operation counts indicated in Table VI. The details of this algorithm are given in Appendix K.

(3) Execution Time Estimate

Table VII presents the STARAN execution times for the cases, n = 128 and n = 256. Floating point hardware is currently being developed for STARAN. The table provides execution times with and without the floating point hardware option. The I/O category in the table consists of the data transfer between the arrays and their respective extended memories. There is a significant increase in I/O time when complex double precision arithmetic is used. This increase occurs because complex double precision multiplication and division cannot be performed entirely within one word of a STARAN array. The sequential machine time shown in the table is a measured CPU time supplied by the customer agency, and should be compared with the corresponding STARAN array processing time rather than the total STARAN time.

TABLE VI - STARAN OPERATION COUNT (ASSUMED n IS A POWER OF 2)

Operation type	Operation Count
-	$\frac{n}{m}(n + m - 2)$
*	$\frac{n}{m}(n + m - 2)$
÷	n
Data moves	$\frac{n}{2}\left(\frac{n}{m}\right)(2 \log_2 n + \log_2 m + 6)$ $+ 1/2 \left(\frac{n}{m}\right)\left(\frac{n}{m} - m \log_2 n\right)$ $+ (\log_2 n + \log_2 m + 2)\left(\frac{5n}{2} - \frac{2n}{m}\right)$

TABLE VII - EXECUTION TIMES FOR MATRIX INVERSION

Variable type and precision	n	STARAN EXECUTION TIMES (SECONDS)					
		Floating point software			Floating point hardware		
		Array processing	I/O	Total	Array processing	I/O	Total
Real-Single precision	128	2.57	0.05	2.62	0.63	0.05	0.68
	256	19.6	0.4	20.0	5.1	0.4	5.5
Real-Double precision	128	8.30	0.11	8.41	1.50	0.11	1.61
	256	62.6	0.8	63.4	11.7	0.8	12.5
Complex-single precision	128	9.83	0.11	9.94	1.66	0.11	1.77
	256	72.0	0.8	72.8	13.1	0.8	13.9
*Complex-Double precision	128	32.75	1.06	33.81	3.76	1.06	4.82
	256	235.6	7.6	243.2	28.8	7.6	36.4

*Requires 2 modules of extended memory per array.

e. Matrix/Vector Product

(1) Problem Description

The capability of computing a matrix/vector product (MVP) is a basic requirement in a wide variety of matrix problems. For example, if A is the known inverse of the matrix B, then Ab is the solution to Bx = b. The matrix inversion problem was considered in the previous item d. Both matrix inversion and the matrix/vector product are subfunctions of the antenna simulation problem, which is discussed in item f below.

Another application of the MVP is that of computing the product PQ of two square matrices P and Q. Each column of PQ is the product of P with a column of Q.

In the remainder of this item we will be concerned with the calculation of Ab, where A is a square n X n matrix and b is a vector.

(2) Problem Solution

There are well-known methods for computing Ab on a sequential computer. The method we propose to utilize and modify for solution by STARAN is the standard method based on the calculation of inner products that is, $A = (a_{ij})$, $b = b_i$ and $c = Ab = (C_i)$ then,

$$C_i = a_{i'}b_1 + \dots + a_{in}b_n \qquad (1)$$

where ,

$$i = 1, \dots, n$$

Using a sequential computer, the calculation of each c_i involves n-1 additions and n multiplications. Thus to compute all

c_i, which is the desired matrix/vector product Ab, n(n-1) additions and n^2 multiplications are required on the sequential machine.

The performance of any implementation of the above method on a parallel processor naturally depends on the amount of parallelism available (relative to n) and on how A and b are stored. The STARAN system configuration shown in Figure 3B was chosen for the implementation of this problem. A total of four associative arrays was assumed as part of this configuration. With this configuration, four rows of the matrix A can be stored in one field of the arrays (one row in each array). At the start of the problem the complete matrix A would be stored in the extended memory (four rows per field). This initial storage scheme is compatible with the data configuration that results from the matrix inversion process described previously in item d. The vector b is stored in one field in each of the four STARAN arrays. This storage scheme will handle an MVP where n(max) = 256.

Basically, the solution process involves the operations of multiplication, addition and data movement. The multiplications are performed directly by field multiplies. All products for four rows are performed simultaneously. The additions are performed by the tree sum method. The geometric nature of the tree sum technique requires that a parallel data movement occur prior to each parallel addition. In general, using the tree sum technique, a sum of n terms can be found in $\log_2 n$ parallel additions. The basic tree sum approach was optimized by performing multiple tree sums simultaneously. To further optimize the process, a data exchange technique was utilized to organize the data such that the number of required parallel additions could be reduced by a factor of two. This technique not only effects a reduction in the number of additions, but, a reduction in the amount of data movement as well.

This MVP process is described in much more detail in Appendix L of Volume II.

(3) Execution Time Estimate

Table VIII presents the execution times for the cases, n = 128 and n = 256. These cases also apply to the antenna simulation problem treated in the next item. The I/O category in the table consists of the data movement between the associative arrays and their respective extended memories. Floating point hardware is currently being developed for STARAN. Table VIII indicates execution time with, as well as without floating point hardware.

The execution times indicated in the table can be reduced by utilizing STARAN array space, when available, to store intermediate results. As a result, a MVP with n = 256 and real single-precision variables can be accomplished in 65.1 ms when the floating point arithmetic is software implemented, and in 9.4 ms when hardware-implemented arithmetic is used.

The basis of the times indicated in Table VIII are developed and presented in Appendix L of Volume II.

f. Antenna Simulation

(1) Problem Description

A program is required which, given the geometry of any antenna system (within certain constraints), provides an analysis of antenna performance. Thus, the program can be used as either an antenna design tool or an antenna evaluation tool. The program should handle up to 256 antenna elements and a frequency range that covers HF (High Frequencies) and VHF (Very High Frequencies). The desired performance output should be in the form of radiation patterns.

TABLE VIII - MVP TIMING SUMMARY

Variable type and precision	n	Floating point software: Array processing	Floating point software: I/O	Floating point software: Total	Floating point hardware: Array processing	Floating point hardware: I/O	Floating point hardware: Total
		STARAN EXECUTION TIME (Milliseconds)					
Real-single precision	128	15.1	0.9	16.0	1.7	0.9	2.6
	256	63.7	3.8	67.5	8.0	3.8	11.8
Real-double precision	128	52.7	1.8	54.5	4.7	1.8	6.5
	256	216.8	7.6	224.4	21.5	7.6	29.1
Complex-single precision	128	61.8	1.6	63.4	6.3	1.6	7.9
	256	248.8	7.6	256.4	26.7	7.6	34.3
Complex-double precision	128	218.5	7.8	226.3	19.3	7.8	27.1
	256	856.0	29.2	885.2	71.5	29.2	100.7

(2) Problem Solution

The approach to the solution is shown in the simplified form in the block diagram of Figure 9.

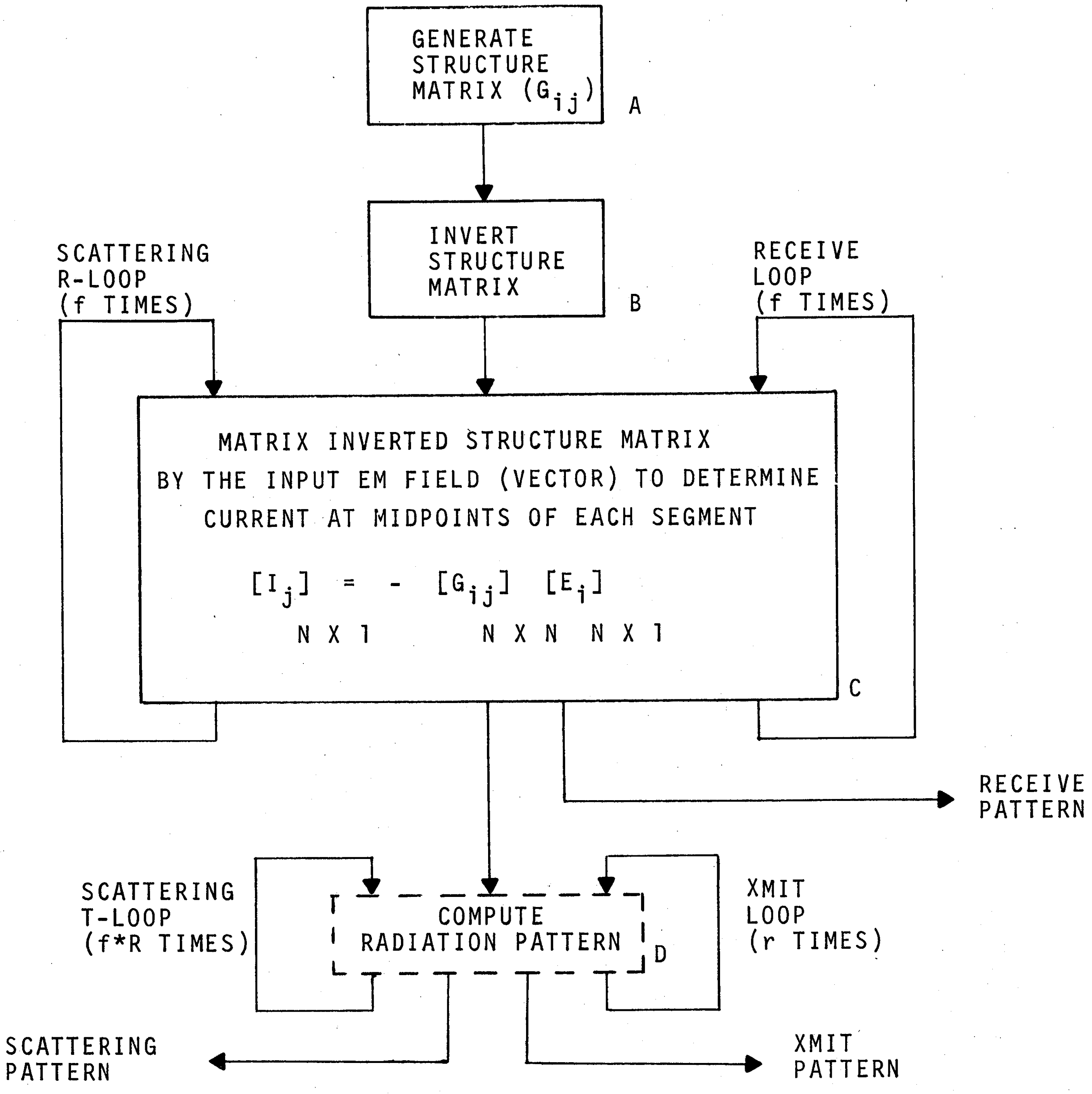
This approach has previously been programmed* for use in a sequential computer. This current study considers the application of the STARAN array processor to a major portion of the simulation model, specifically Blocks A, B and C of Figure 9. The procedure indicated in the Reference* was utilized as much as possible, however, some program modification was necessary to realize the full capability of the STARAN array processor. The following paragraphs provide some detail on the solution of only Block A (structure matrix generation) of Figure 9. A description of the approach to the solution of Blocks B and C have already been presented in items II-2d and II-2e, respectively, of this volume. However, the solution time for all three blocks (A,B and C) will be summarized herein.

The simulated antenna is defined as a geometrical configuration of conducting straight line segments. The segment-to-segment interaction between the segment currents I_j and the segment electromagnetic fields E_j provides a description of the antenna's electrical characteristics. For example, an input electromagnetic field E_i at segment i will induce currents I_j** at the N segments (subscript j) that comprise the antenna in accordance with

$$\sum_{j=1}^{j=N} G_{ij} I_j = E_i \qquad (2)$$

*Burke, G. J.; Selden, E.S.; etaal: Antenna Modeling Program; Information Systems, Menlo Park, California 94025; July 10, 1972.

**Currents I_j are scaler estimates of the current at the midpoints of segments j.


GENERATE STRUCTURE MATRIX (G_{ij})
A
INVERT STRUCTURE MATRIX
B
SCATTERING R-LOOP (f TIMES)
RECEIVE LOOP (f TIMES)
MATRIX INVERTED STRUCTURE MATRIX BY THE INPUT EM FIELD (VECTOR) TO DETERMINE CURRENT AT MIDPOINTS OF EACH SEGMENT
$[I_j] = - [G_{ij}] [E_i]$
N X 1 N X N N X 1
C
RECEIVE PATTERN
SCATTERING T-LOOP (f*R TIMES)
COMPUTE RADIATION PATTERN
D
XMIT LOOP (r TIMES)
SCATTERING PATTERN
XMIT PATTERN

f = NO. OF RECEIVE DIRECTIONS

r = NO. OF TRANSMIT (RADIATING) DIRECTIONS

Figure 9 - Antenna Simulation Solution Approach

The terms, G_{ij}, are the elements of an N by N matrix that describe the geometrical relationship between segment i and j that satisfies the electrical relationship indicated by equation 2.

Because of the relatively large storage requirements and amount of data movement associated with the required matrix operations, a high-bandwidth I/O approach was desirable. Consequently, the STARAN system configuration shown in Figure 3 view (B) was chosen for the implementation of this problem. Four associative arrays were assumed to be available as part of this configuration.

(3) Execution Time Estimate and Comparison

The STARAN solution time estimates (provided by the contractor) the sequential machine time measurements (provided by the customer agency) are recorded in Table IX. Note that sequential machine time measurements were not available for the sub-functions, but, only for total CPU time. The original total received from the customer agency included CPU time for performing the function of Block D. (Figure 9). Since this contractor did not analyze the Block D function, the sequential machine times appearing in Table IX reflect a 15 percent decrease under the original times supplied by the customer. The 15 percent is the customer's estimate for the performance of the Block D function. This adjustment makes the comparison, between STARAN and the sequential machines, more equitable.

The initial analysis effort in applying STARAN to this problem was based on the use of single precision complex arithmetic, and assumed the condition of 256 antenna segments. However, the customer-supplied sequential machine times were based on the use of double-precision arithmetic and involved 134 segments. Consequently, additional times were estimated by this contractor in order to be

TABLE IX - ANTENNA SIMULATION TIMING SUMMARY AND COMPARISON

Task	STARAN Array Processor (Estimated Time in Seconds)								Sequential Machines (Measured Time in Secs) [DP Complex]	
	Hardware Floating Point				Software Floating Point					
	SP Complex		DP Complex		SP Complex		DP Complex		IBM360/65	IBM360/195
No. of Segments →	256	128	256	128	256	128	256	128	134	134
● Generate Structure Matrix (Block A of Figure 9)									NA	NA
. Array Computations	1.78	.45	7.10	1.80	7.66	1.95	30.6	7.65		
. I/O	0.16	.09	.48	.25	0.16	.09	.48	.25		
● Matrix Inversion (Block B, Figure 9) (See also item II-2d)									NA	NA
. Array Computations	13.1	1.66	28.8	3.76	72.0	9.83	235.6	32.75		
. I/O	0.8	0.11	7.6	1.06	0.8	0.11	7.6	1.06		
● Matrix/Vector Multiply (Block C, Figure 9) (See also item II-2e)									NA	NA
. Array Computations	0.027	0.006	0.071	0.019	0.248	0.062	0.856	0.219		
. I/O	0.008	0.002	0.029	0.007	0.008	0.002	0.029	0.008		
● Radiation Pattern Computation (Block D, Figure 9)	NC	NC	NC	NC	NC	NC	NC	NC	NA	NA
Total Array (or CPU) Computations	14.9	2.12	36.0	5.58	79.9	11.84	267.1	40.62	3652	684
Total I/O	0.97	0.20	0.11	1.32	0.97	0.20	8.11	1.32	NA	NA
Overall Total	15.9	2.32	36.1	6.9	80.9	12.0	275.2	41.9		

SP = Single Precision Arithmetic
DP = Double Precision Arithmetic
NC = Not Considered
NA = Not Available

compatible with the customer-supplied conditions. These additional times appear in Table IX.

The STARAN times and the sequential machine times shown in Table IX are compatible from the standpoint of type of output pattern. In the case of the sequential machine, a single source scatter pattern was assumed. In the case of STARAN, a transmit pattern was assumed. Both patterns are equivalent from the solution time standpoint, since in both cases a single pass through Blocks A, B and C (Figure 9) is required, and multiple passes (typically 180) through Block D.

It should be noted that the sequential machines require approximately 800K of core to handle 134 antenna segments. STARAN, however, requires less than 1/10 of that storage capacity to operate on almost twice the number of antenna segments.

A more detailed discussion of this problem is given in Volume II of this report.

g. Error Correction

This problem requires the correction of received telemetry data that contain errors due to poor signal-to-noise ratio. The processing required for solution involves the generation of all high probability correct data, and then applying Kalman filter techniques to determine the true transmitted data.

The problem is currently in the developmental stage at the customer agency and is initially being implemented by the PDP-11/45 sequential machine.

In this study, the STARAN associative array processor was applied to the problem to determine performance which could then be compared with existing techniques. The STARAN system configuration in Figure 3B was utilized for the implementation and only a single associative array was required.

The application of STARAN resulted in the dramatic performance increase by a factor of 20,000 over the existing PDP-11/45 developmental implementation. Even assuming the PDP-11/45 program could be optimized by a factor of 10, the STARAN approach would still provide a performance increase by a factor of 2000. STARAN can actually process the data faster than the data inputs occur by a factor of more than 20. This time difference with respect to real time could be traded off to provide additional processing that may be required in an improved algorithm. For example, instead of generating and processing <u>all high probability</u> correct data, <u>all possible</u> correct data could be generated and processed. This approach would result in a much higher probability of a proper solution to this problem.

For the details of the problem, the solution algorithm and the implementation by STARAN, the reader is referred to the classified volume of this report - Volume III.

3. VALIDATION

a. Requirements

Solution time estimates have their place, but they are always suspect until the process is actually performed and the solution time measured. This fact is the motivation for including the validation phase in this study. This validation phase consisted of: 1), selecting one of the problems that had been analyzed (and solution time estimated) earlier in the study, 2) programming the selected problem for STARAN, and 3) executing the program in STARAN and actually measuring the solution time. In like manner, the customer agency agreed to: 1) program the same selected problem for sequential solution, 2) execute the problem on both the IBM 360/195 and the IBM 360/65 machines, and 3) measure the corresponding solution times. With the foregoing process accomplished, the data will be available to directly compare the actual STARAN solution time with the previously estimated STARAN solution time. Direct comparison can also be made between the actual STARAN solution time and solution times of current high-performance sequential machines.

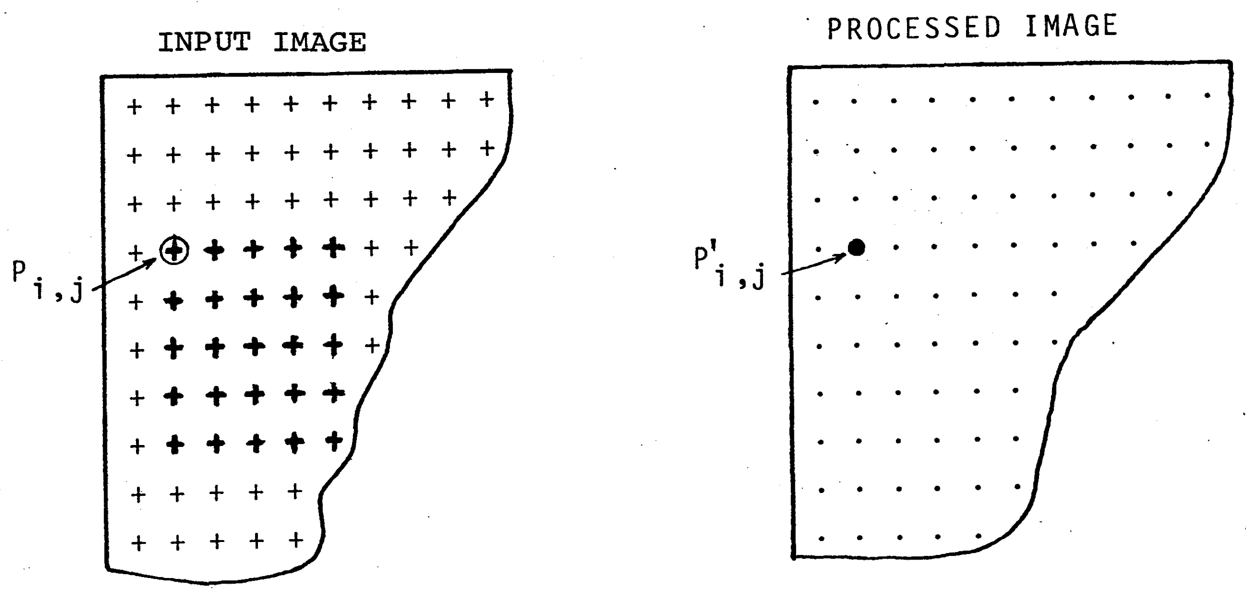
b. Validation Problem Description

The image averaging problem was selected for validation. Image averaging is a technique for enhancing an image by reducing image noise. Noise in an image can be introduced from various sources. One source of noise could be the equipment which digitizes the image.

This image averaging technique requires that a new gray value be computed for each specified point of the input image. This is done by finding the average gray value of all the image points within the boundaries of a specified matrix cell such that the image point of interest is located in the top left corner. This process is shown by sketch and equation in Figure 10 below.

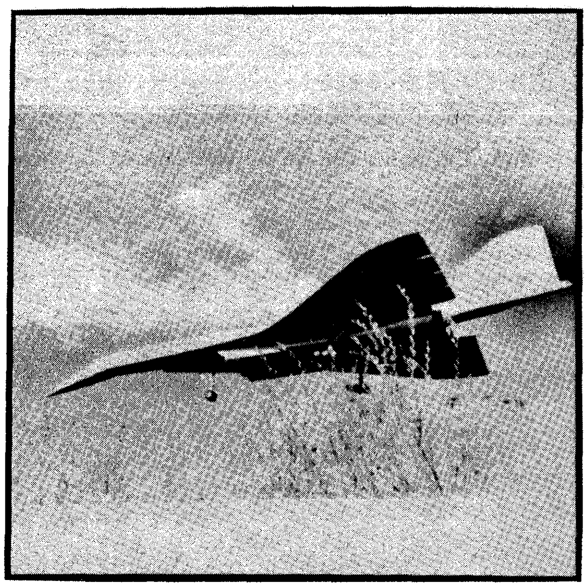
c. Approach

The customer agency provided the images which served as the input to the image averaging programs. The original image is shown in Figure 11A. The customer digitized this image and added 5 percent noise to this original image by modifying the most significant bit of the gray levels. The original image, with noise added, is shown in Figure 11B and was used as the actual data input to the image averaging programs. Both images were supplied to GAC on 9-track magnetic tape.

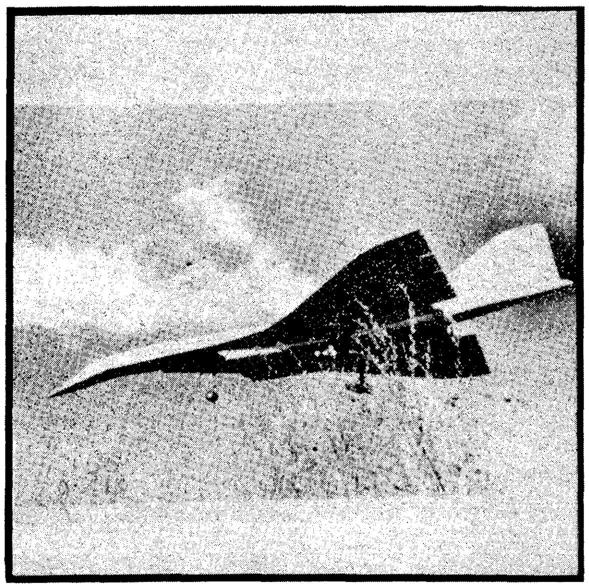


$$P'_{i,j} = \frac{\left(\sum_{m=0}^{M-1} \sum_{n=0}^{N-1} P_{i+m,j+n} \right)}{MN}$$

Figure 10 - Validation Problem Description



A. Original Image



B. Original Image with Noise Added

Figure 11 - Input Imagery (Supplied by Customer)

The image averaging problem was programmed by both the customer agency and GAC, but, for different machines. The customer agency programmed the problem in Fortran language for execution on both the IBM 360/195 and IBM 360/65 computers, and supplied the processed output image to GAC on 9-track magnetic tape along with measured CPU execution times for both the IBM 360/195 and the IBM 360/65 computers.

The image averaging problem was programmed by GAC for execution on the STARAN array processor. The program was executed in the STARAN Evaluation and Training Facility at GAC, Akron, Ohio. The equipment configuration in this facility is as shown in Figure 12. A more detailed view and description of this facility can be found in Appendix A of the this volume. The SIGMA 9 computer serves as the host computer. The I/O routines were programmed in Fortran language and executed in the Sigma 9. All other routines were programmed, using a combination of APPLE (Associative Processor Programming LanguagE) and MAPPLE (Macro Associative Processor Programming LanguagE), for execution in STARAN. The top level flow chart for the image averaging program is given in Figure 13. A detailed description of the program is provided in the program specification found in Appendix O of Volume II.

d. Results

The input image containing 5 percent noise was processed by the customer on two sequential machines, and by Goodyear. The input image and the two independently processed output images are shown in visual form in the figure. These two independently derived output images are virtually identical. Note that the noise was removed, but the contrast was reduced some what in the process. The contrast could be restored by additional processing, but this way beyond the scope of this contract.

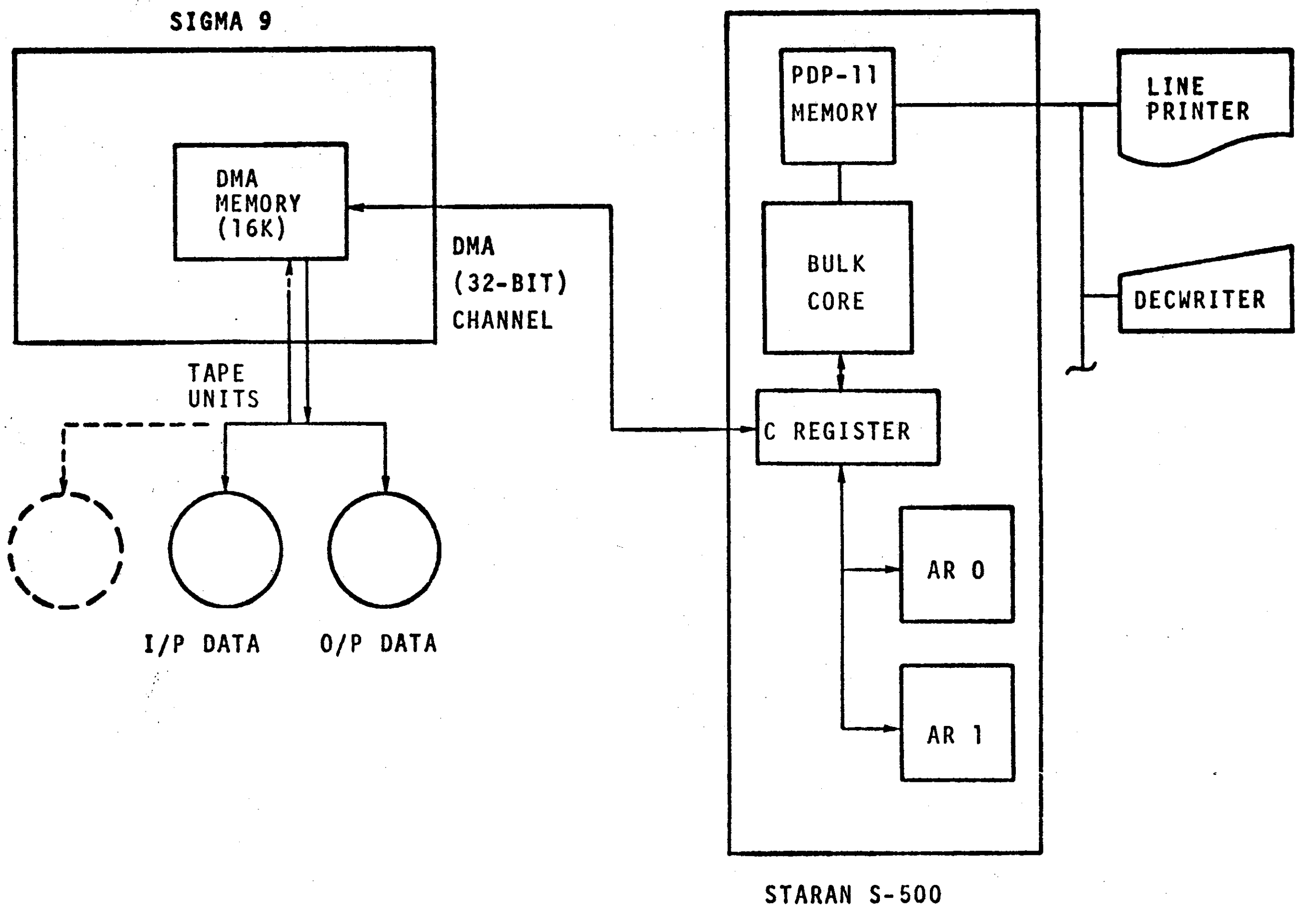


Figure 12 - STARAN Evaluation and Training Facility - Equipment Configuration

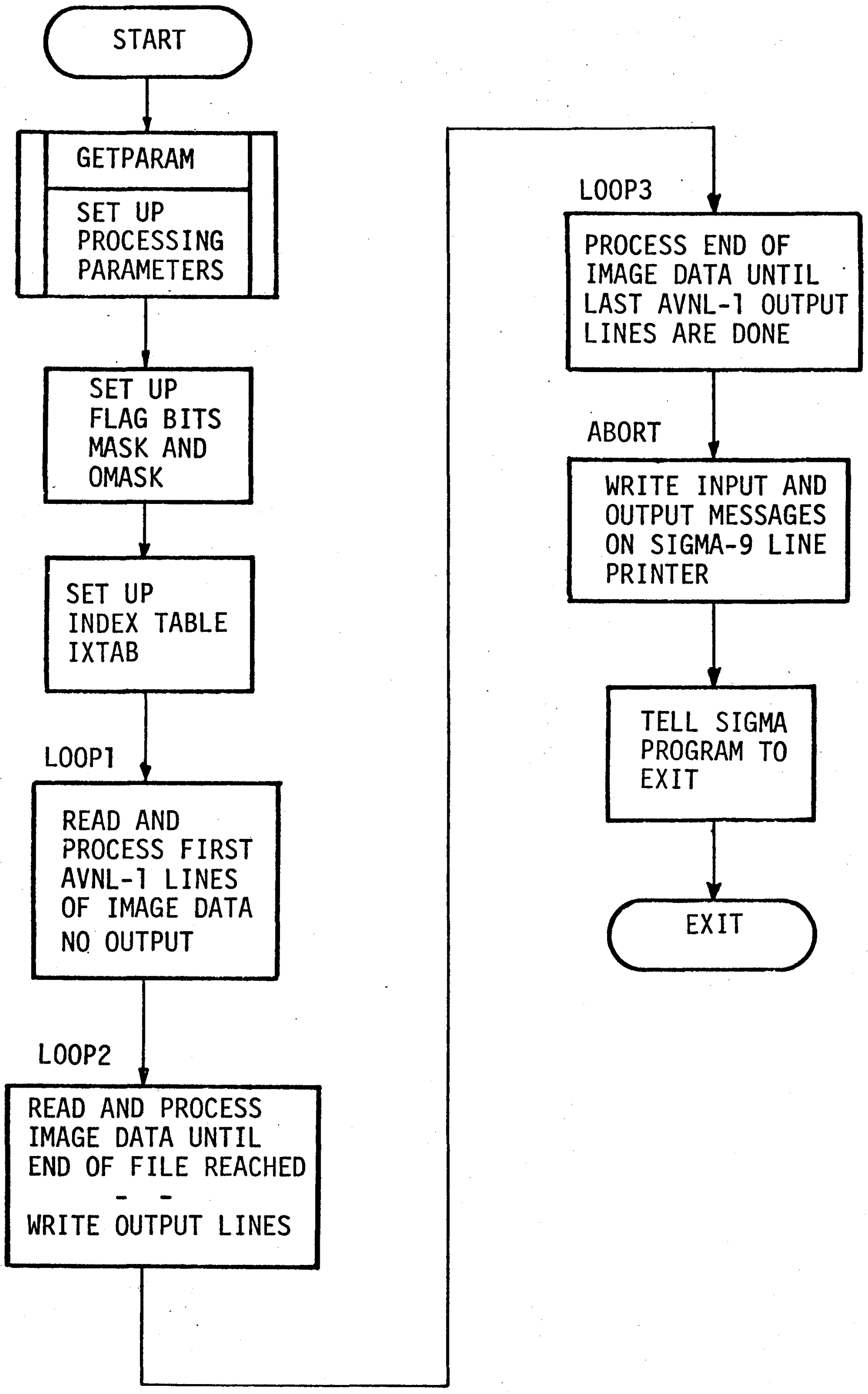
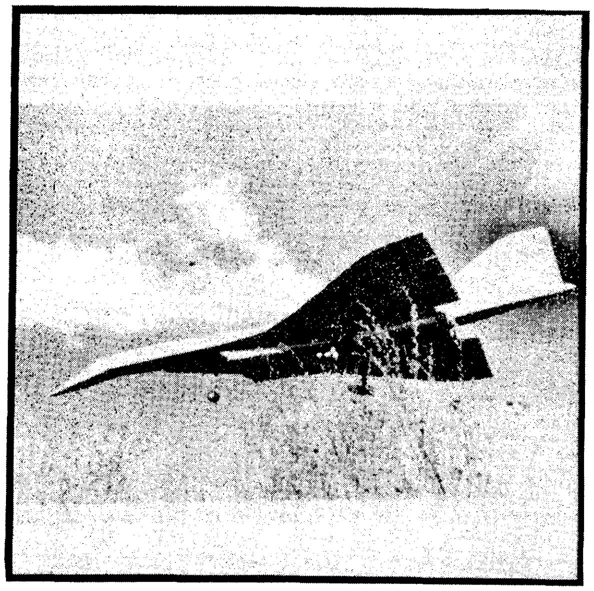

START
GETPARAM
SET UP PROCESSING PARAMETERS
SET UP FLAG BITS MASK AND OMASK
SET UP INDEX TABLE IXTAB
LOOP1
READ AND PROCESS FIRST AVNL-1 LINES OF IMAGE DATA NO OUTPUT
LOOP2
READ AND PROCESS IMAGE DATA UNTIL END OF FILE REACHED - - WRITE OUTPUT LINES
LOOP3
PROCESS END OF IMAGE DATA UNTIL LAST AVNL-1 OUTPUT LINES ARE DONE
ABORT
WRITE INPUT AND OUTPUT MESSAGES ON SIGMA-9 LINE PRINTER
TELL SIGMA PROGRAM TO EXIT
EXIT

Figure 13 - Top Level Flow Chart - IAVG

NOISEY INPUT IMAGE



STARAN ARRAY PROCESSING

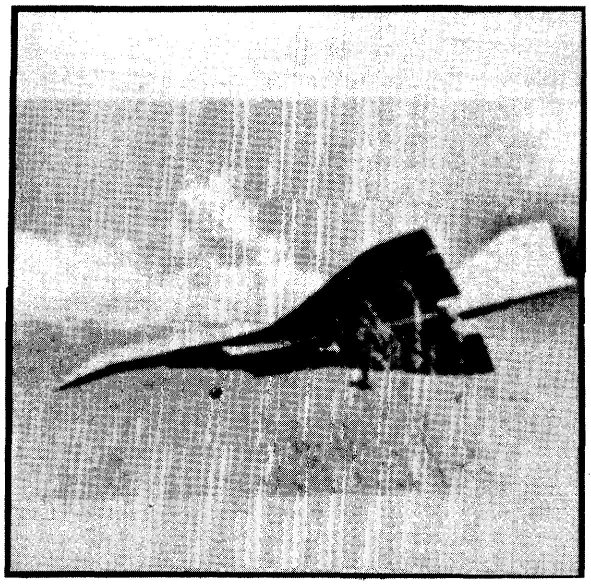
ESTIMATED TIME (SEC)	MEASURED TIME (SEC)
0.21	0.17

SEQUENTIAL MACHINE PROCESSING

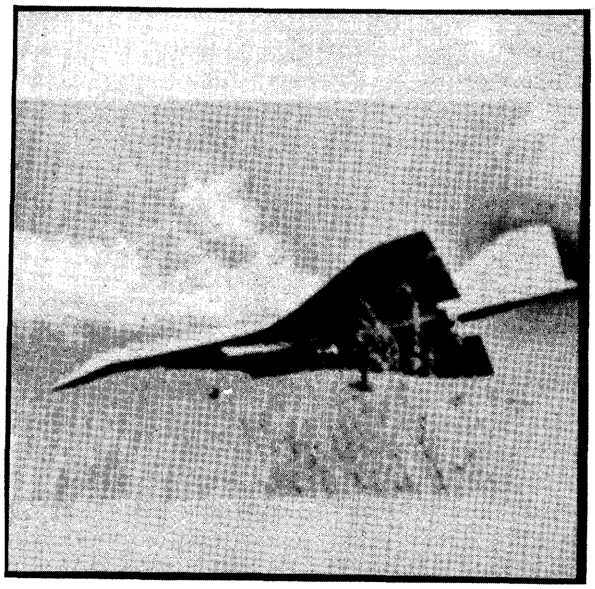
MEASURED CPU TIME (SEC)

IBM 360/195	IBM 360/65
8.04	60.0

OUTPUT IMAGES
(NOISE REMOVED)



GAC - PROCESSED
OUTPUT IMAGE



CUSTOMER - PROCESSED
OUTPUT IMAGE

Figure 14 - Image Averaging Process for Validation - STARAN vs. Sequential machine

All photographs shown in this report were converted from magnetic tape by means of a Dicomed Corporation Model E40 Image Recorder.

The measured times indicated in Figure 14 show that STARAN provides a performance improvement over the IBM 360/195 by a factor of 40. In addition, the measured STARAN time was about 25 percent faster than the estimated STARAN time. These results directly validate the performance estimate for the image averaging problem, and by implication validate the performance estimates for all problems considered in the study since the same basic estimating techniques were used throughout.

A detailed breakdown of the STARAN execution time is shown in Table X. Additional statistics that resulted from the validation effort are recorded in Table XI. Machine hours, programmer hours and instruction counts are provided.

Ease of programming is an important consideration for any machine. There are those who believe programming for a parallel machine is more complex than for a sequential machine. There are others who believe programming for a parallel machine like STARAN is not more difficult, but just a challenging new way of thinking. The contention that STARAN programming is not more difficult, but probably simpler, is reinforced by the results of this validation phase of the study. Programming rate is one measure of programming complexity (or simplicity). Applying the data found in Tables XI(B) and (C), indicates that the GAC programmer achieved a rate of about 9 instructions per hour for just the STARAN coding process. If the total programming effort (i.e., system analysis, coding, debugging and documentation) is used to compute the rate, the result is about 4 instructions per hour. This is considerably higher than the figures generally encountered in industry for assembly language

TABLE X - DETAILED STARAN TIMING FOR VALIDATION PROGRAM (IAVG)

Program portion timed	Modules* timed	Execution time(seconds)
Initialization	SET PARAM, Part of AVERAGE	0.001126
Array Processing	Balance of AVERAGE, IO2LINE, COMPTCELL, COMPTRSLT, RSLT2IO	0.169335
DMA-to-Array Transfer	MOVIN	0.301708
Array-to-DMA Transfer	MOVOUT	0.771092
Waiting for Tape Read	READ,RWAIT	9.887311
Waiting for Tape Write	WRITE,WWAIT	23.210492
Total Program	All	34.341

*Modules are described in Appendix O, Validation Program Specification.

TABLE XI - VALIDATION PROGRAM STATISTICS

(IMAGE AVERAGING)

(A) MACHINE HOURS

OPERATION DURING:	HOURS OF MACHINE TIME		
	SIGMA 9	STARAN	PDP-11/20 OFF-LINE
Assembly	-	-	2.04
Debug	-	4.47	-
Post-debug	1.39	5.16	-

(B) PROGRAMMER HOURS

TASK	HOURS	
System Analysis		21.0
Sigma-9 Coding		10.5
Appl. Prog.	9.0	
Perf. Mon. Prog.	1.5	
STARAN Coding		56.0
Appl. Prog.	40.0	
Perf. Mon. Prog.	16.0	
Sigma-9 Debug		3.0
STARAN Debug		29.0
Keypunch		8.1
Sigma-9	6.0	
STARAN	2.1	
Documentation*		26.5
TOTAL		154.1

*Appendix O, Volume II

(C) INSTRUCTION COUNT

PROGRAMS	NUMBER OF INSTRUCTIONS	
STARAN		520
Basic Program	325	
Additional For variability	163	
Additional for Perf. Mon.	32	
SIGMA-9 (I/O)		122
OVERALL TOTAL		642

programming. One reason for the comparative ease of programming afforded by STARAN can be seen by example. Consider an image made up of a matrix of pixels that must be processed. Since a sequential machine can only process one pixel at a time, the sequential programmer must provide for looping in two dimensions. However, since STARAN can process a complete column (or row) of pixels simultaneously, the STARAN programmer need only consider looping in one dimension.

The detailed program specification for the image averaging (IAVG) program is provided in Appendix O, Volume II.

APPENDIX A

STARAN SYSTEM DESCRIPTION

A copy will be made available upon a request to:

Digital Systems Marketing
Goodyear Aerospace Corporation
Akron, Ohio 44315

(216) 794-3631