Best
Available
Copy

AD/A-002 977

A STUDY OF THE ILLIAC IV COMPUTER FOR SEISMIC DATA PROCESSING

Ann Kerr, et al

Teledyne Geotech

Prepared for:

Air Force Technical Applications Center
Defense Advanced Research Projects Agency

11 October 1974

DISTRIBUTED BY:

NTIS
National Technical Information Service
U. S. DEPARTMENT OF COMMERCE

Unclassified
SECURITY CLASSIFICATION OF THIS PAGE (When Data Entered)

REPORT DOCUMENTATION PAGE		READ INSTRUCTIONS BEFORE COMPLETING FORM
1. REPORT NUMBER SDAC-TR-74-16	2. GOVT ACCESSION NO.	3. RECIPIENT'S CATALOG NUMBER AD/A-002977
4. TITLE (and Subtitle) A STUDY OF THE ILLIAC IV COMPUTER FOR SEISMIC DATA PROCESSING		5. TYPE OF REPORT & PERIOD COVERED Technical
		6. PERFORMING ORG. REPORT NUMBER
7. AUTHOR(s) Kerr, Ann; Wagenbreth, Gene; Smart, Eugene; Der, Zoltan		8. CONTRACT OR GRANT NUMBER(s) F08606-74-C-0006
9. PERFORMING ORGANIZATION NAME AND ADDRESS Teledyne Geotech 314 Montgomery Street Alexandria, Virginia 22314		10. PROGRAM ELEMENT, PROJECT, TASK AREA & WORK UNIT NUMBERS
11. CONTROLLING OFFICE NAME AND ADDRESS Defense Advanced Research Projects Agency Nuclear Monitoring Research Office 1400 Wilson Blvd., Arlington, Virginia 22209		12. REPORT DATE 16 August 1974
		13. NUMBER OF PAGES 50
14. MONITORING AGENCY NAME & ADDRESS (if different from Controlling Office) VELA Seismological Center 312 Montgomery Street Alexandria, Virginia 22314		15. SECURITY CLASS. (of this report) Unclassified
		15a. DECLASSIFICATION DOWNGRADING SCHEDULE

16. DISTRIBUTION STATEMENT (of this Report)

APPROVED FOR PUBLIC RELEASE; DISTRIBUTION UNLIMITED.

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

18. SUPPLEMENTARY NOTES

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

ILLIAC
Parallel Processing

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

Two features of the ILLIAC IV system at NASA/Ames are particularly appropriate to the processing of seismic data. One is its ability to apply a given algorithm to sixty-four different data streams simultaneously, thus providing an order of magnitude increase in processing speed over conventional machines. The second is its large data storage. The seismological algorithms for convolution filtering, beamforming, matched filtering, PHILTRE, maximum likelihood, and FKCOMB are each able to take advantage of these features in the

DD FORM 1 JAN 73 1473 EDITION OF 1 NOV 65 IS OBSOLETE

Unclassified
SECURITY CLASSIFICATION OF THIS PAGE (When Data Entered)

Reproduced by
NATIONAL TECHNICAL INFORMATION SERVICE
US Department of Commerce
Springfield, VA. 22151

processing of seismic data. The data can be arranged so that each processing element contains successive time windows on a given trace, as in bandpass filtering, or successive beam sets, as in beamforming.

Some preliminary data editing is required for each of these algorithms to arrange the data appropriately in processing element memory to utilize the ILLIAC IV computer efficiently. Data formatting schemes were designed for one algorithm (FKCOMB) which was coded and implemented on the ILLIAC; these schemes can be appropriately modified for use with other seismological algorithms.

Experience with data transfer, program entry and editing, compilation, and program execution show that while the ILLIAC system is still under development, adequate facilities do exist for development of seismological algorithms.

A STUDY OF THE ILLIAC IV COMPUTER FOR SEISMIC DATA PROCESSING

SEISMIC DATA ANALYSIS CENTER REPORT NO.: SDAC-TR-74-16

AFTAC Project No.:	VELA VT/4709
Project Title:	Seismic Data Analysis Center
ARPA Order No.:	1620
ARPA Program Code No.:	3F10
Name of Contractor:	TELEDYNE GEOTECH
Contract No.:	F08606-74-C-0006
Date of Contract:	01 July 1974
Amount of Contract:	$2,152,172
Contract Expiration Date:	30 June 1975
Project Manager:	Royal A. Hartenberger (703) 836-3882

P. O. Box 334, Alexandria, Virginia 22314

APPROVED FOR PUBLIC RELEASE; DISTRIBUTION UNLIMITED.

ABSTRACT

Two features of the ILLIAC IV system at NASA/Ames are particularly appropriate to the processing of seismic data. One is its ability to apply a given algorithm to sixty-four different data streams simultaneously, thus providing an order of magnitude increase in processing speed over conventional machines. The second is its large data storage. The seismological algorithms for convolution filtering, beamforming, matched filtering, PHILTRE, maximum likelihood, and FKCOMB are each able to take advantage of these features in the processing of seismic data. The data can be arranged so that each processing element contains successive time windows on a given trace, as in bandpass filtering, or successive beam sets, as in beamforming.

Some preliminary data editing is required for each of these algorithms to arrange the data appropriately in processing element memory to utilize the ILLIAC IV computer efficiently. Data formatting schemes were designed for one algorithm (FKCOMB) which was coded and implemented on the ILLIAC; these schemes can be appropriately modified for use with other seismological algorithms.

Experience with data transfer, program entry and editing, compilation, and program execution show that while the ILLIAC system is still under development, adequate facilities do exist for development of seismological algorithms.

TABLE OF CONTENTS

LIST OF FIGURES

[illegible]

A

Disclaimer: Neither the Advanced Research Projects Agency nor the Air Force Technical Applications Center will be responsible for information contained herein which has been supplied by other organizations or contractors, and this document is subject to later revision as may be necessary. The views and conclusions presented are those of the authors and should not be interpreted as necessarily representing the official policies, either expressed or implied, of the Advanced Research Projects Agency, the Air Force Technical Applications Center, or the US Government.

INTRODUCTION

The purpose of this study was to determine the suitability of the ILLIAC computer for processing seismic data. We have done this by looking at the computing requirements of each of several algorithms; and then, by comparing these requirements with the characteristics of the ILLIAC, we have investigated the feasibility of programming each of the algorithms on the ILLIAC. Finally, the procedure FKCOMB was actually coded for the ILLIAC and the program has been tested and run. FKCOMB is a long-period seismic signal analysis procedure, which is important in calculating discriminants between earthquakes and nuclear explosions; it may become an integral part of data processing on the Seismic Network. FKCOMB was chosen for this experiment because the large amount of processor time required prohibits its use in-house. Also, known results are available with which to compare the ILLIAC version.

The ILLIAC computer consists of a control unit, 64 arithmetic units or processing elements (PE), 128K 64-bit words of core, and 10^9 bits of disk storage. The 64 PE's execute instructions in lock step; i.e., they all execute the same instruction simultaneously. It is in this respect that the ILLIAC departs from conventional computer architecture.

The control unit decodes instructions and executes instructions for program control. It has 24-bit integer arithmetic hardware to calculate indices and addresses. There are four general purpose accumulators and a 64-word fast access memory in the CU, which also has access to all 128K of core and initiates transfers between core and disk.

The primary computational resource of the ILLIAC is the array of 64 processing elements. Each has complete arithmetic capabilities and can perform 2×10^6 multiplications per second. The capacity of all 64 PE's is about 10^8 multiplications per second. Each PE has access only to 2K of core, and has only limited capability to communicate with other PE's. Control within a PE consists of the ability programatically to disable selected PE's. When disabled, a PE's memory is protected and cannot be altered by the PE, though all other facets of instruction are performed.

The ILLIAC disk is the primary storage device. It consists of 13 head-per-track disks and two disk controllers. Together the disks hold approximately 10^9 bits. Transfers between core and disk are initiated by the ILLIAC control unit and occur in blocks or pages of 1024 words. The transfer rate is about 10^9 bits per second. Access time to any record on disk is 40 milliseconds or less. The disk can be loaded from the Tenex file system prior to program execution and unloaded after program termination. The layout of data on the ILLIAC disk is under user control and should be arranged so as to minimize access times during program execution.

The 64 processing elements provide the ability to perform vector arithmetic operations on 64 data elements simultaneously. Program logic generally requires that selected PE's be disabled to avoid redundant calculations if all 64 processing units are not required. In general, program execution time is decreased if disabling of PE's is avoided.

ILLIAC is able to perform approximately 100 million operations (i.e., a multiplication, addition, etc.) per second. Any procedure which requires fewer than 100 billion operations would have a running time of under 10 minutes. The setup time for an ILLIAC job is large enough to make such a run impractical. Thus, algorithms requiring very few computations or the use of a small data base with any algorithm are unsuitable for ILLIAC processing.

APPROACH TO DESIGNING PARALLEL SEISMOLOGICAL ALGORITHMS

Overview of ILLIAC

ILLIAC is a parallel processor. It consists of a control unit (CU), 64 processing elements (PE), 131,072 words of core memory, and 15,974,400 words of disk memory. The control unit has access to all of core memory. Its basic cycle time is 60 nanoseconds. However, greater processing power is achieved through the simultaneous execution of an instruction in each of the 64 processing elements.

The control unit fetches and decodes all instructions. After decoding, some instructions are broadcast for execution in the processing elements while others are executed in the control unit. The arithmetic capability of the control unit is limited to 24 bit two's complement addition and subtraction, masking, and comparison for use in branching. The control unit has no floating point capability. One operand at a time is processed by the control unit. The control unit also initiates data transfers between core and ILLIAC disk.

The processing power of ILLIAC resides in 64 identical processing elements. Each PE executes instructions broadcast from the CU. Though each PE has its own index registers and memory to operate upon, all 64 PE's always execute identical instructions in lockstep. Each PE has direct access to 2048 words of core memory, shown as a column in Figure 1.

There are three data paths available for communication among PE's and between the PE's and the control unit (CU). First, the CU can access all of core, so it can load a word from one processing element memory (PEM) and either use it or store it in another PEM. This method of communication is both simple and flexible, allowing for any data movement desired, but, since only one word at a time is transferred, it is relatively slow compared to the two other methods available.

Second, the CU can communicate with all PE's by broadcasting the same word to all PE's simultaneously. This method is faster than the first since sixty-four words are transmitted at once, but provides only a limited form of communication.

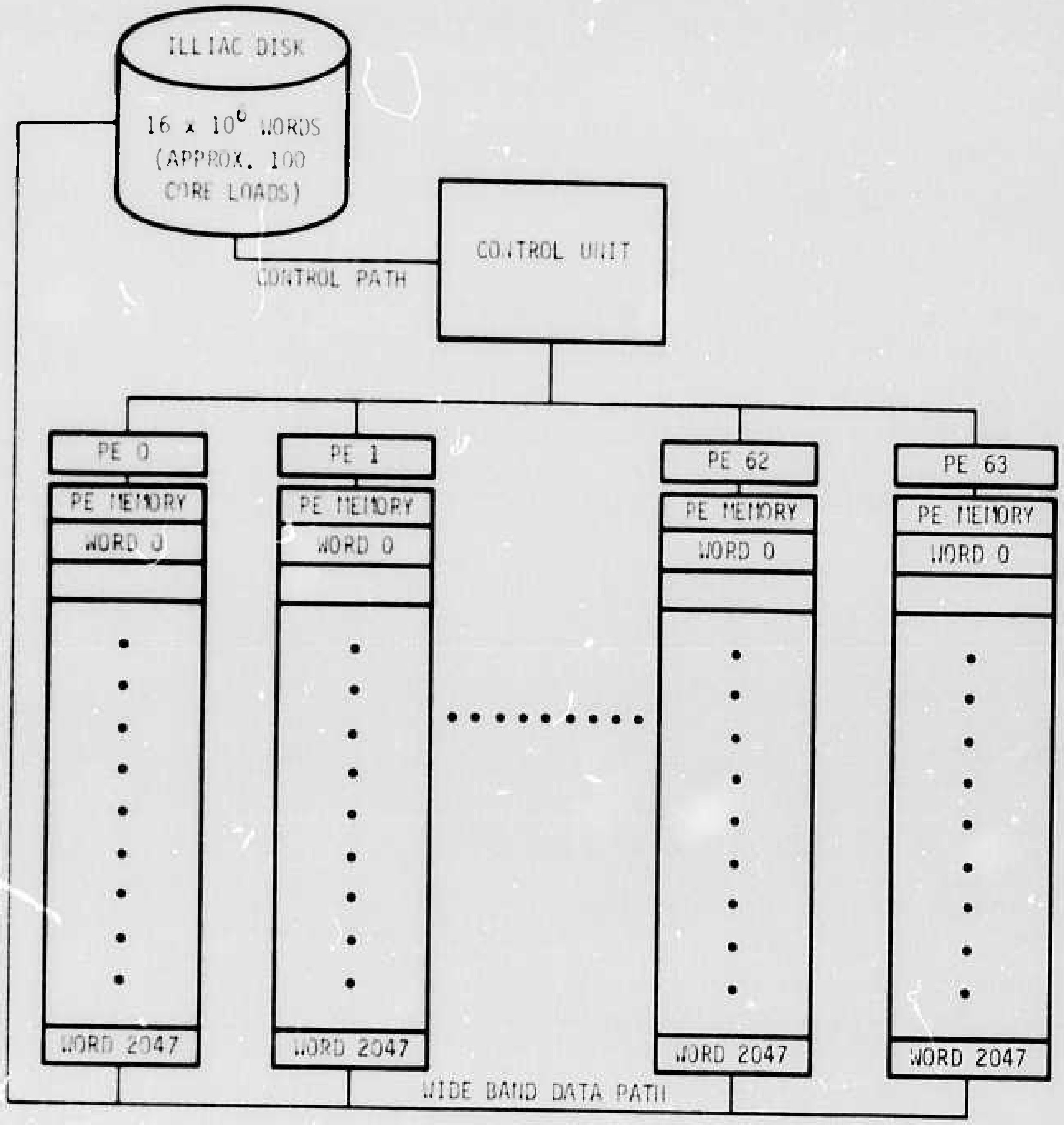

Figure 1. Representation of ILLIAC Memory Organization.

Third, the PE's can communicate with each other via the ROUTE instruction which transfers the contents of a register in each PE to the PE determined by the following scheme: If PEN (processing element number) is the number of the source PE and R is the route amount supplied with the instruction, identical in all PE's, the number of the destination PE is MOD_{64} (PEN+R). If the PE's are thought of as arranged in a circle with PE 63 adjacent to PE 0, the ROUTE instruction consists of loading the data, rotating the circle, and storing the data. This data transfer is very fast since 64 words are transferred simultaneously. It is general in that all 64 words can be different but the pattern set by the fact that the routing distance is the same for all PE's is restrictive. It does not transfer 64 words randomly distributed in core to 64 different locations simultaneously.

The primary memory used by ILLIAC is a disk memory with capacity approximately 100 times that of core memory. One page (1024 words) of memory is the minimum amount of data that can be transferred between core and disk. Although the bandwidth between core and disk is $.5\text{x}10^9$ bits per second, the average access time to a particular spot on disk is 20 milliseconds. This relatively long access time (compared to a 60 nanosecond clock time in 64 parallel processors) necessitates careful planning of disk usage. The number of disk transfers must be kept to a minimum to avoid waiting for disk accesses.

Since the most important feature of ILLIAC is its computing power (approximately 100 million multiplications per second), one of the prime objectives in the design of any ILLIAC program is to minimize execution time. The best possible result is a running time 1/64 that possible with only one PE, but due to the architecture of the machine the degree to which this is achieved is dependent upon the design of the algorithm. We discuss two general approaches to designing algorithms. First, suppose that it is necessary to code the trigonometric SIN function for ILLIAC. If the particular usage makes it possible to always compute 64 functions simultaneously, one simple has the same SIN routine running in all PE's on different data, and a speedup by a factor of 64 is very nearly achieved. (Some time is lost if there is conditional branching in the original SIN routine which is changed to enabling and disabling of PE's). A second approach is to devise a method for utilizing all 64 processors to compute one function value. No method has been devised for doing this 64 or even 10 times faster than is possible with one processor. The first approach is both faster and simpler, but certain algorithms may preclude calculation of more than one value of SIN simultaneously or may require significant overhead elsewhere in order to do so.

One misconception is that if all of the PE's are kept "busy" the machine is running at maximum efficiency. In fact this statement is not true and one must be very careful in relating the word efficiency to the use of ILLIAC. For example, consider the problem of summing groups of numbers. If it is desired to sum 64 pairs of numbers, keeping 64 different results, each PE forms one sum and the work is done 64 times faster than could be done by one processor. If however, it is desired to find the sum of one group of 64 numbers, a more complicated method must be used. In order to simplify the explanation somewhat, consider an eight PE machine and the summation of eight numbers, one in each PE. Figure 2 depicts a method whereby this can be done in three routes and three additions. Since the routes require roughly equivalent CPU time as the register loads necessary before any operation, the time taken for an 8 PE machine to sum eight numbers is equal to the time taken for three additions. If this algorithm is extended to the summation of 64 numbers within a 64 PE machine it takes 6 additions to form the sum. Given that one PE requires 63 additions to sum 64 numbers, the 64 PE machine is 63/6 or 10.5 times faster. Note that although none of the PE's are ever disabled and all are forming the sum, this algorithm does not achieve the factor of 64 speed up. However, the factor of ten speed up that is achieved makes this algorithm usable if data organization requires its use.

Design Considerations

The choice of which design approach to take for a particular problem is dependent upon data organization. There is often one approach requiring a very specific data organization which is much faster than any other. It must be decided whether the overhead and execution time involved in data transposition is compensated by decreased overhead and execution time elsewhere in the algorithm.

	STATE OF REGISTERS							
OPERATION	**PE_0**	**PE_1**	**PE_2**	**PE_3**	**PE_4**	**PE_5**	**PE_6**	**PE_7**
Initial Conditions	$\$A=I_0$ $\$R=0$	$\$A=I_1$ $\$R=0$	$\$A=I_2$ $\$R=0$	$\$A=I_3$ $\$R=0$	$\$A=I_4$ $\$R=0$	$\$A=I_5$ $\$R=0$	$\$A=I_6$ $\$R=0$	$\$A=I_7$ $\$R=0$
1. Route Contents of $\$A_N$ to $\$R_{N+1}$.	$\$A=I_0$ $\$R=I_7$	$\$A=I_1$ $\$R=I_0$	$\$A=I_2$ $\$R=I_1$	$\$A=I_3$ $\$R=I_2$	$\$A=I_4$ $\$R=I_3$	$\$A=I_5$ $\$R=I_4$	$\$A=I_6$ $\$R=I_5$	$\$A=I_7$ $\$R=I_6$
2. Add $A to $R and leave result in $A.	$\$A=I_0+I_7$ $\$R=I_7$	$\$A=I_1+I_0$ $\$R=I_0$	$\$A=I_2+I_1$ $\$R=I_1$	$\$A=I_3+I_2$ $\$R=I_2$	$\$A=I_4+I_3$ $\$R=I_3$	$\$A=I_5+I_4$ $\$R=I_4$	$\$A=I_6+I_5$ $\$R=I_5$	$\$A=I_7+I_6$ $\$R=I_6$
3. Route Contents of $\$A_N$ to $\$R_{N+2}$.	$\$A=I_0+I_7$ $\$R=I_6+I_5$	$\$A=I_1+I_0$ $\$R=I_7+I_6$	$\$A=I_2+I_1$ $\$R=I_0+I_7$	$\$A=I_3+I_2$ $\$R=I_1+I_0$	$\$A=I_4+I_3$ $\$R=I_2+I_1$	$\$A=I_5+I_4$ $\$R=I_3+I_2$	$\$A=I_6+I_5$ $\$R=I_4+I_3$	$\$A=I_7+I_6$ $\$R=I_5+I_4$
4. Add $A to $R and leave result in $A.	$\$A=I_0+I_7+$ I_6+I_5 $\$R=I_6+I_5$	$\$A=I_1+I_0+$ I_7+I_6 $\$R=I_7+I_6$	$\$A=I_2+I_1+$ I_0+I_7 $\$R=I_0+I_7$	$\$A=I_3+I_2+$ I_1+I_0 $\$R=I_1+I_0$	$\$A=I_4+I_3+$ I_2+I_1 $\$R=I_2+I_1$	$\$A=I_5+I_4+$ I_3+I_2 $\$R=I_3+I_2$	$\$A=I_6+I_5+$ I_4+I_3 $\$R=I_4+I_3$	$\$A=I_7+I_6+$ I_5+I_4. $\$R=I_5+I_4$
5. Route Contents of $\$A_N$ to $\$R_{N+4}$.	$\$A=I_0+I_7+$ I_6+I_5 $\$R=I_4+I_3+$ I_2+I_1	$\$A=I_1+I_0+$ I_7+I_6 $\$R=I_5+I_4+$ I_3+I_2	$\$A=I_2+I_1+$ I_0+I_7 $\$R=I_6+I_5+$ I_4+I_3	$\$A=I_3+I_2+$ I_1+I_0 $\$R=I_7+I_6+$ I_5+I_4	$\$A=I_4+I_3+$ I_2+I_1 $\$R=I_0+I_7+$ I_6+I_5	$\$A=I_5+I_4+$ I_3+I_2 $\$R=I_1+I_0+$ I_7+I_6	$\$A=I_6+I_5+$ I_4+I_3 $\$R=I_2+I_1+$ I_0+I_7	$\$A=I_7+I_6+$ I_5+I_4 $\$R=I_3+I_2+$ I_1+I_0
6. Add $A to $R and leave result in $A.	$\$A=I_0+I_7+$ I_6+I_5+ I_4+I_3+ I_2+I_1 $\$R=I_4+I_3+$ I_2+I_1	$\$A=I_1+I_0+$ I_7+I_6+ I_5+I_4+ I_3+I_2 $\$R=I_5+I_4+$ I_3+I_2	$\$A=I_2+I_1+$ I_0+I_7+ I_6+I_5+ I_4+I_3 $\$R=I_6+I_5+$ I_4+I_3	$\$A=I_3+I_2+$ I_1+I_0+ I_7+I_6+ I_5+I_4 $\$R=I_7+I_6+$ I_5+I_4	$\$A=I_4+I_3+$ I_2+I_1+ I_0+I_7+ I_6+I_5 $\$R=I_0+I_7+$ I_6+I_5	$\$A=I_5+I_4+$ I_3+I_2+ I_1+I_0+ I_7+I_6 $\$R=I_1+I_0+$ I_7+I_6	$\$A=I_6+I_5+$ I_4+I_3+ I_2+I_1+ I_0+I_7 $\$R=I_2+I_1+$ I_0+I_7	$\$A=I_7+I_6+$ I_5+I_4+ I_3+I_2+ I_1+I_0 $\$R=I_3+I_2+$ I_1+I_0

Figure 2. Detailed View of Rowsum Operation.

The first step in designing a seismic algorithm to run on ILLIAC is to examine similar or repeated data structures and determine how they could be organized in the processor memory and to analyze similar or repeated operations and determine how they could be divided among the processors.

Long and short period seismic data are recorded at seismic arrays consisting of a group of sensors sampled at a constant time interval. The data so recorded consists of a series of data scans. Each data scan is a time sample from each sensor. There are two structures repeated throughout the data. First, there are several channels, each identical in structure to the rest. Second, there are many identically structured time samples. In order to utilize either of these structures, time must be spent transposing the data. It would be convenient if it were possible to process the data without transposing in any way - but the input consists of data records which are formatted differently for each array.

Since the data must be restructured, it is reasonable to build a new structure which makes processing as fast and straight forward as possible. The choice between the two data structures is dependent upon the requirements of the algorithm. General discussions of the several seismic algorithms and their data requirements is contained in Section 3. A detailed discussion of the design of FKCOMB is found in Section 4.

SEISMIC APPLICATIONS ON THE ILLIAC IV COMPUTER

General

In the following paragraphs we shall discuss the suitability of the ILLIAC computer for processing seismic data using several tested algorithms.

Our investigation has revealed that the ILLIAC computer is generally suited for processing of seismic data which involves frequently repeated or simultaneous identical operations using different sets of data, and can be programmed in such a way that the processing is performed simultaneously in the 64 processing elements of this computer. Thus, if desirable, it will be feasible to use ILLIAC to process routinely all long-period data for the planned Seismic Network. In addition it can also be used for off-line processing of selected data.

In this discussion we shall concentrate on the possibilities of this computer for the detection and discrimination of seismic events using seismic array data. The computer can also be used in other seismic applications too numerous to treat here. Seismic arrays record the earth motion in two separate frequency bands, short-period and long-period, which for some purposes require different treatments because of the different nature of seismic waves recorded in the two bands. Some of the processes discussed are used for data in both bands while others are commonly used only for long or short period data.

The most common signals for investigation in the short period band are the short-period body waves, particularly the short-period P first arrival. P waves can arrive at a seismic station with a wide range of apparent velocities and from all possible azimuths. Since the bandwidth of the signal is limited, frequency filtering tends to enhance the signal/noise ratio. The detection threshold in the short-period band is low relative to that of the long-period band, and events are usually detected in this band. The arrival azimuth and apparent velocity of the short-period P waves at an array yields a preliminary epicenter location which can be used to narrow the search for waves in the long-period band. In the long-period band, long-period body waves and surface waves are the signals of interest. When used in conjunction with short-period data, they are all proven or potential discriminants between explosions and earthquakes. The most important of these is the long-period Rayleigh wave, which is used in the M_s-m_b discriminant. The Rayleigh wave has several characteristics which can be utilized by detection algorithms:

1. Waveform (path-dependent);
2. Particle motion;
3. Azimuth and apparent velocity.

Since in most cases detection already has occurred on the short-period data, it is only necessary to prove or disprove the presence of Rayleigh or other long-period waves arriving from roughly the direction of the preliminary epicenters, and to measure the wave amplitude if present.

The following seismic processing algorithms will be discussed in this report:

1. Frequency (convolution/recursive) filtering
2. Beamforming
3. Matched filtering
4. PHILTRE
5. Maximum likelihood f-k spectra
6. FKCOMB

The last four of these have only limited or no application for the short period band. One processor (FKCOMB) is discussed in more detail since it was selected to be demonstrated on the ILLIAC.

Convolution and Recursive Filtering

Simple filtering is the convolution of a seismic trace with some arbitrary function which limits the bandwidth of the output. Recursive filtering accomplishes the same result, but makes use of a feedback loop to reduce the number of arithmetic operations required.

This operation can be represented mathematically in the form:

$$y_n = \sum_{l=n}^{m} a_l x_{n-l} + \sum_{i=1}^{k} b_i y_{n-l}$$

where all indices are integers, x_i are values of the original digitized trace and y_i are values of the filtered output, and a_i, b_i, n, m, and k are constants the choice of which is dependent on the filter function to be performed.

The ILLIAC is well suited to perform convolution or recursive filtering simultaneously on all processing units. These algorithms can be used for filtering all elements of an array using the same filters to enhance a band limited signal in wideband noise, or utilized for filtering the same trace with a set of filters to perform a fast Fourier analysis or to compute spectral ratios for discrimination. The parallel algorithm can also be used to simultaneously deconvolve sixty-four seismic traces, remove instrument response, simulate seismograms produced by different instruments, or to reduce the seismogram traces simultaneously to accelerations, velocities, and displacements as functions of time.

Figure 3 is a schematic representation of possible arrangements of data in the ILLIAC memory for frequency filtering. In Figure 3a, a different channel of data is input to each PE, with the same filter applied to all PE's. In Figure 3b, a given channel of data is input to all PE's, with a different filter applied to each PE. Figure 3c represents a combination of the previous examples in which the PE's are partitioned into several sets, all of the PE's in a given set receiving the same data channel but operating with different filters.

Beamforming

Beamforming is the process of time-shifting several channels of array data and summing them to form a single channel. The time shifts chosen are the natural delays in time of arrival of a hypothetical signal crossing the array. The delays are defined with respect to some arbitrary point in space. For plane waves of constant velocity, the delays are

$$\tau_i = \vec{r}_i \cdot \vec{S}$$

where i is the index of the i^{th} sensor, $\vec{r}$ is the location of the sensor and the slowness of the signal is:

$$\vec{S} = \vec{V}/(\vec{V}\cdot\vec{V})$$

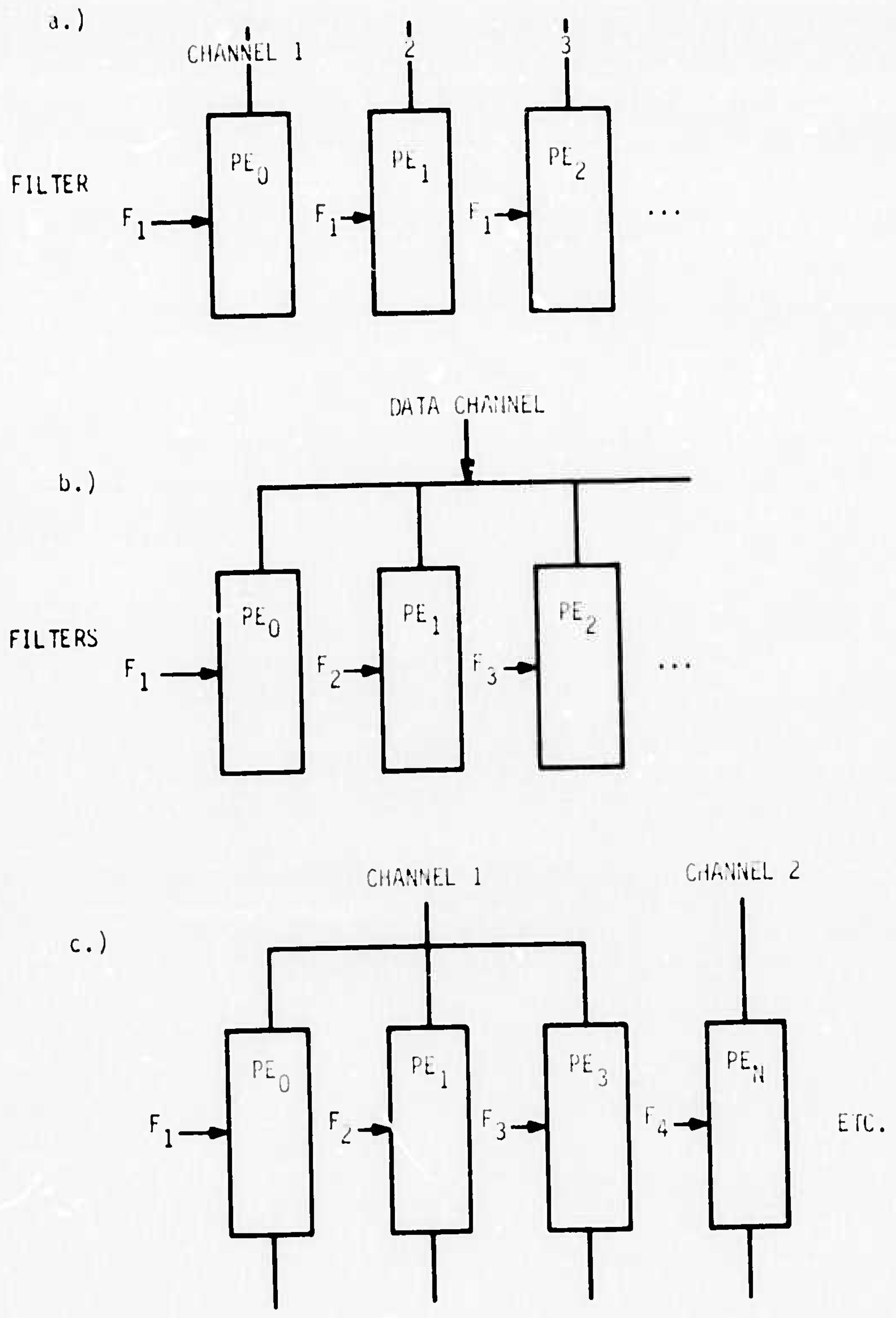

Figure 3. Suggested Data Schemes for Convolution-Recursive Filtering

where $\vec{V}$ is the velocity of the signal across the array. If one has computed delays from the true $\vec{S}$ of a given signal, that is, from its true speed and arrival azimuth, and has assumed that the signal waveform does not vary during transit, the effect of time shifting is to make all the channels appear to have been recorded at the arbitrary reference point. The effect of summing, therefore, will be to add the signal to itself N-1 times, where N is the number of channels. The signal is thus reinforced. If the noise is random and uncorrelated between array elements, it is reduced to $N^{-1/2}$ of its original amplitude by the summation. Thus beamforming has the function of increasing the effective signal-to-noise ratio.

One can estimate the speed and direction of propagation of signals by finding the maximum of the time average of the squared beam values (denoted $\overline{B^2}$) on the $\overline{S}$-plane:

$$\overline{B^2} = \sum_{j=0}^{J-1} \left[\frac{1}{N} \sum_{i=1}^{N} x_i \; (j\Delta t-\tau_i)\right]^2 = \sum_{j=0}^{J-1} B_j^2$$

where B_j is the expression in brackets, the beam of the array; x_i are the i^{th} channel data samples; Δt is the sampling interval. j is the time index; J is the number of time points over which average is taken.

The probability of the presence of the signal can be estimated by the statistic

$$F = \frac{N-1}{N} \; \frac{\overline{B^2}}{\sum_{i=1}^{N} (x_i-B)^2}$$

where the denominator is the time average of the sum of the square (or power) of the individual input traces x_i after the beam is subtracted.

This statistic is distributed approximately as a non-central F variable with degrees of freedom determined by the number of channels, bandwidth, and the time length of averaging (assuming that only uncorrelated noise is present).

The standard F tables can be used to determine the significance of detection, or the detection can be automated (Blandford, 1971).

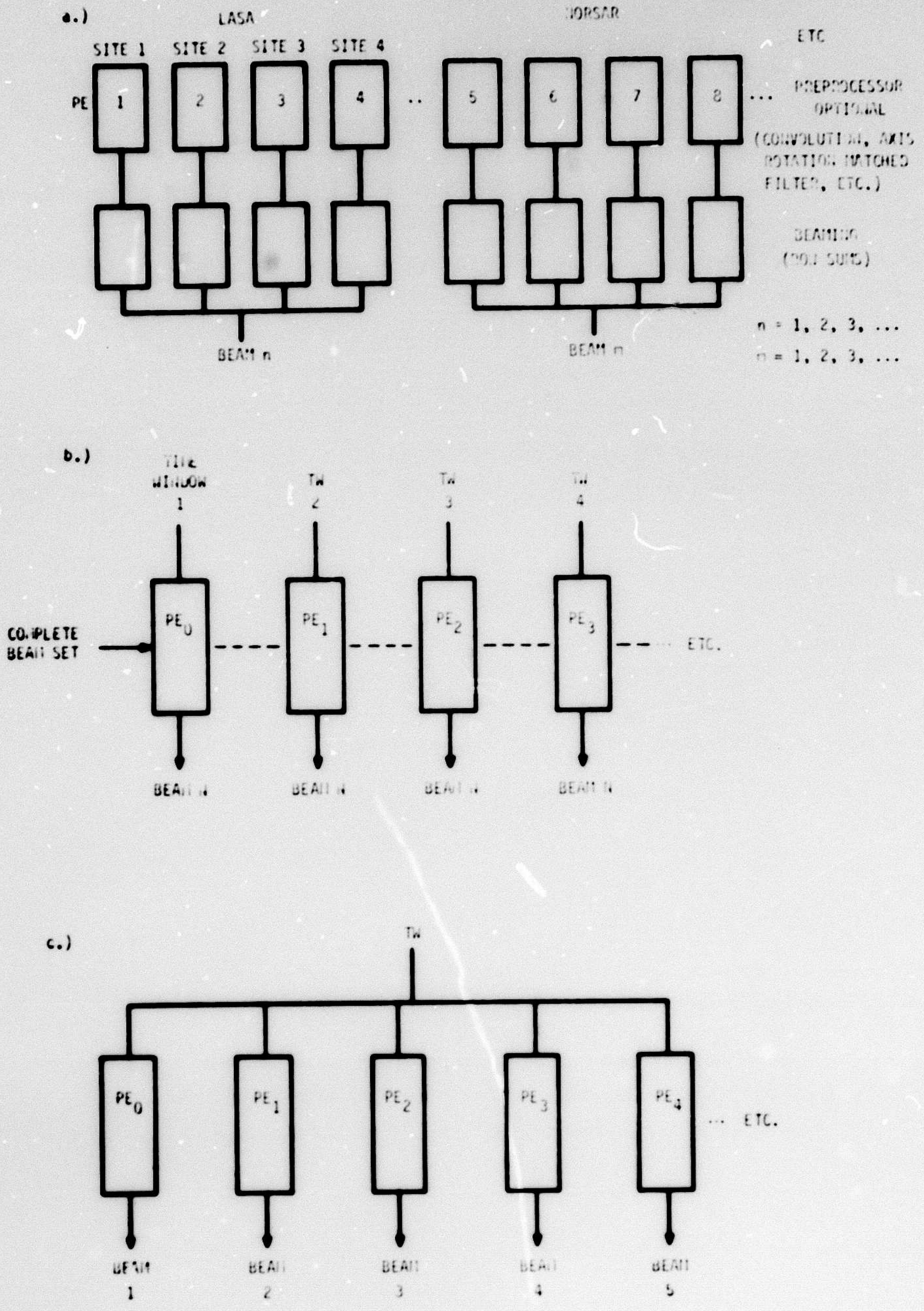

Figure 4. Suggested Data Schemes for Beamforming.

The beams can also be displayed for the visual detection of the waves of interest. For detection of surface waves and the measurement of M_s this is still the best procedure. Experienced analysts can recognize and measure seismic wave arrivals in many cases where automatic machine detection schemes fail. Routine computation of long-period beams and their storage in the mass store event files would be a valuable routine function for the NEP and would require a substantial computational power easily met by the ILLIAC. Therefore beamforming might well be the single most useful algorithm for implementation on the ILLIAC computer.

Several basic computational configurations can be used in beam processing. These are shown in Figure 4. In the first configuration (4a) each PE contains one sensor trace and the beam values are accumulated by forming row sums on the several PE's. This configuration is suitable to process several arrays simultaneously, and computing the desired beams from a single data set sequentially may use long time windows such as those needed for the recognition of dispersed surface wave trains. Another advantage of this configuration is that preprocessing of traces such as filtering processing can also be performed simultaneously prior to beaming without the need to remove the data from memory. The output of such a scheme can be directly used in network event processing. This configuration, uniquely possible on the ILLIAC IV, is the most efficient if the maximum number of PE's can be utilized. This can be achieved if the total number of sensors in the arrays are close to sixty-four, or alternatively the remaining PE's are used to compute different beams on the same arrays. To obtain continuous seismograms of long duration this seems to be the most efficient approach, since various preprocessing schemes, such as convolution filtering, coordinate rotation to obtain Rayleigh Love, SH and SV components can be performed on them without the need for excessive numbers of overlaps in the successive time windows which would be required if, as we discuss below, each PE were to contain all the channels of data required for a particular beam. Incidentally, PHILTRE can be used as a postprocessor for 64 array beams previously obtained (for 64 events) which can be run in parallel.

There are two other alternate but generally less effective computational configurations which are indicated in Figure 4. One loads all sensor traces from one array into one PE and each PE contains a different time window. The desired beams for a given time window may then be computed sequentially (Figure 4b). The other scheme (Figure 4c) loads the same time window, all traces, into as many PE's as there are desired beams and the beams are computed simultaneously. The disadvantage of the last mentioned methods is that since each PE contains all traces the corresponding time windows must be shorter due to PE memory limitations. This processing, including beaming, will require more complicated buffering schemes between core and disk. Therefore it seems that the first computational scheme has the most practical value, although the others may be used advantageously, for instance, for enhancing short body wave arrivals. The maximum utilization of the computer requires the consideration of the type of processing required, number of traces or arrays and the length of time windows to be processed.

Matched Filtering

This technique utilizes the waveform of the signal to be detected (Alexander and Rabenstine, 1967a,b). The expected waveform of the signal is used on the seismic trace as a convolution filter. Ideally the expected waveform is identical to the actual one and in the resulting output trace the signal is transformed into a pulse which is shaped like the autocorrelation of the signal waveform. In practice it is not possible to predict the actual waveform precisely, so the matched filtering results in the contraction of the actual signal, which for surface waves can be a long wave train, into a much shorter waveform. By compressing the same amount of energy into a shorter time interval, the signal/noise is increased. It also de-emphasizes signals which do not match the waveform used for filtering. The technique has been successful in detecting surface waves, and preliminary results indicate that it is a very effective preprocessor for f-k spectra analysis (FKCOMB or maximum likelihood f-k spectra) if it is applied to all elements of an array. Application of matched filtering requires that the signal waveform be known, which in turn presupposes knowledge of the approximate epicenter, which may be acquired by short-period detection. If the epicenter is known, the recordings of a nearby large event can be used as the expected waveforms. Alternatively, if the dispersion characteristics of the path are known sufficiently well, the signal waveforms can be synthesized and the resulting waveform used as a matched filter.

An alternative application of matched filtering can be relative location of events. If recordings of a reference event (preferably of an explosion) are available at a set of seismic stations, the times of maxima resulting from the matched filtering of seismic traces of nearby events with waveforms of the reference event, can be considered as relative arrival times for the purpose of event relocation in the general region surrounding the reference event. The technique also has a potential as a discriminant since azimuthal variations in the initial phases of earthquakes will cause inconsistencies in the times of occurrences of matched filter output maxima when compared to explosions.

Matched filtering is essentially convolution, and the computational advantages of convolution or recursive filters on the ILLIAC stated above apply in this case.

Possible applications of the ILLIAC (Figure 5) includes matched filtering of many sites simultaneously (each with a different matched filter), filtering several sets of array elements simultaneously with matched filters corresponding to each array, or filtering independent sites (such as LRSM sites) with their own respective matched filters. One can also use matched filters corresponding to several areas of interest routinely on the data.

PHILTRE

This process is designed for a single three-component set of long-period data. It uses a nonlinear weighting scheme of Fourier spectral components in overlapping time windows to enhance Love or Rayleigh particle motion associated with a given arrival direction (Simons 1968). First the three components of long period recordings are rotated to obtain radial transversal and vertical motion. The rotated traces are broken up into overlapping time windows and Fourier transformed, yielding the Fourier coefficients

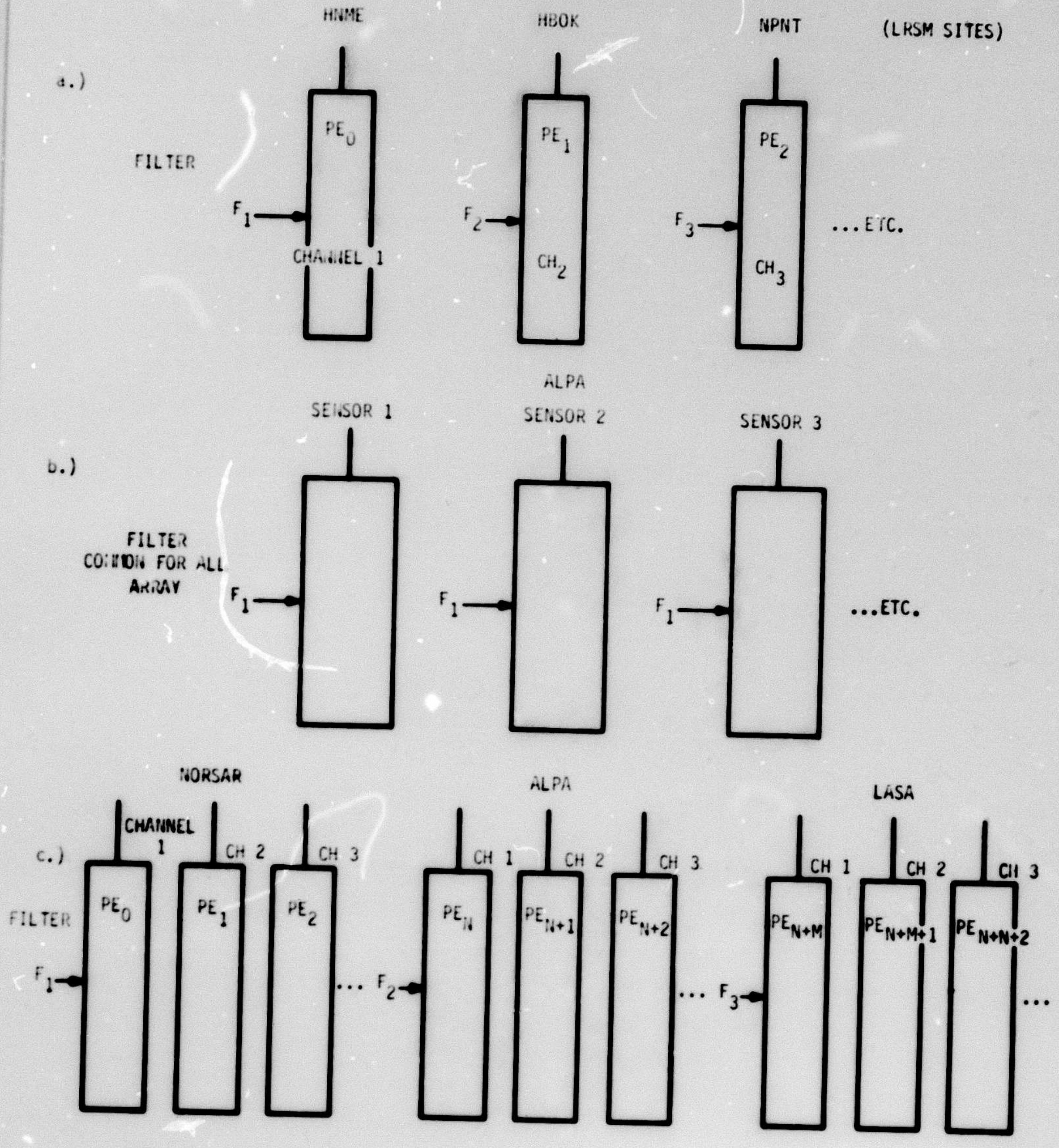

Figure 5. Suggested Data Schemes for Matched Filtering.

$$a_c(nf) = \frac{2}{T}\int_0^T c(\tau)\cdot\cos 2\pi nf\tau\cdot d\tau;$$

$$b_c(nf) = \frac{2}{T}\int_0^T c(\tau)\cdot\sin 2\pi nf\tau\cdot d\tau;$$

where $c(\tau)$ is the radial, transverse or vertical component to be analyzed, T is time, $n = 0,1,2,3,\ldots,N-1$, Nf = folding frequency, and $f = \frac{1}{T}$ = fundamental harmonic of Fourier series.

Using the absolute value of a Fourier component

$$A_c(nf) = \sqrt{\alpha_c^2(nf) + b_c^2(nf)}$$

one computes three quantities used in the weighting scheme

a.) The apparent horizontal azimuth (the angle from the radial direction)
$$\beta(nf) = \arctan \frac{A_t(nf)}{A_r(nf)}$$

b.) A measure of the eccentricity of the particle motion ellipse
$$\Psi(nf) = \arctan \frac{A_r^2(nf) + A_t^2(nf)}{A_z(nf)}$$

c.) The phase difference between the vertical and radial components
$$\alpha(nf) = \Theta_r(nf) - \Theta_z(nf).$$

The Fourier amplitude coefficients of each direction components are then weighted in the following manner

$$A'_z(nf) = A_z(nf)\cdot\cos^M[\beta(nf)]\cdot\cos^K[\Psi(nf)-.21\pi]\cdot\sin^N[\alpha(nf)]$$

$$A'_r(nf) = A_r(nf)\cdot\cos^M[\beta(nf)]\cdot\cos^K[\Psi(nf) - .21\pi]\cdot\sin^N[\alpha(nf)]$$

$$A'_t(nf) = A_t(nf)\cdot\sin^M[\beta(nf)]\cdot\sin^K[\Psi(nf)]\cdot 1$$

where $\sin^N[\alpha(nf)] \equiv 0$ if $\pi \leq \alpha(nf) \leq 2\pi$.

The $A_c'(nf)$ are the "weighted amplitude coefficients". No weights or adjustments are applied to the phase angles. The exponents M, K, and N are parameters that are read into the program. Values of M, K, and N which have worked reasonably well in practice range from 4 to 8. Note that on the vertical radial components all weighting factors vary from 1 to 0 as powers of sines and cosines depending upon the degree to which the particle motion resembles pure Love or Rayleigh waves.

The effects of the first weighting factors (functions of ε) are to attenuate transverse energy on the vertical and radial components and radial energy to the transversed component.

The second set of weighting factors depends upon the angle Ψ - a measure of the eccentricity of the Rayleigh orbit providing transversal trace does not contain too much non-Rayleigh type motion.

On the vertical and radial traces, the angle desired (0.21π) is the one corresponding to a representative value of the horizontal/vertical displacement ratio (≈ 0.8) for fundamental long-period Rayleigh waves.

The resulting Fourier coefficients are subsequently transformed back into the time domain to yield transverse traces containing only Love motion and radial and vertical traces with only Rayleigh motion and greatly reduced noise since the weighting scheme de-emphasizes noise which, even if coherent, is liable to come from a direction different from that of the epicenter.

The data dependent nature of this algorithm does not lend itself well to utilize the parallel computing feature of ILLIAC. However if large sets of data need to be processed each PE can process three components of data from a given location (Figure 6). This may make PHILTRE a practical preprocessor for arrays. Recent work by von Seggern and Sobel (1974) indicates that it is effective in revealing Rayleigh waves hidden in noise. Although further tests are needed to establish its effectiveness as a preprocessor for an f-k type detector, it utilizes a neglected aspect of surface wave detection.

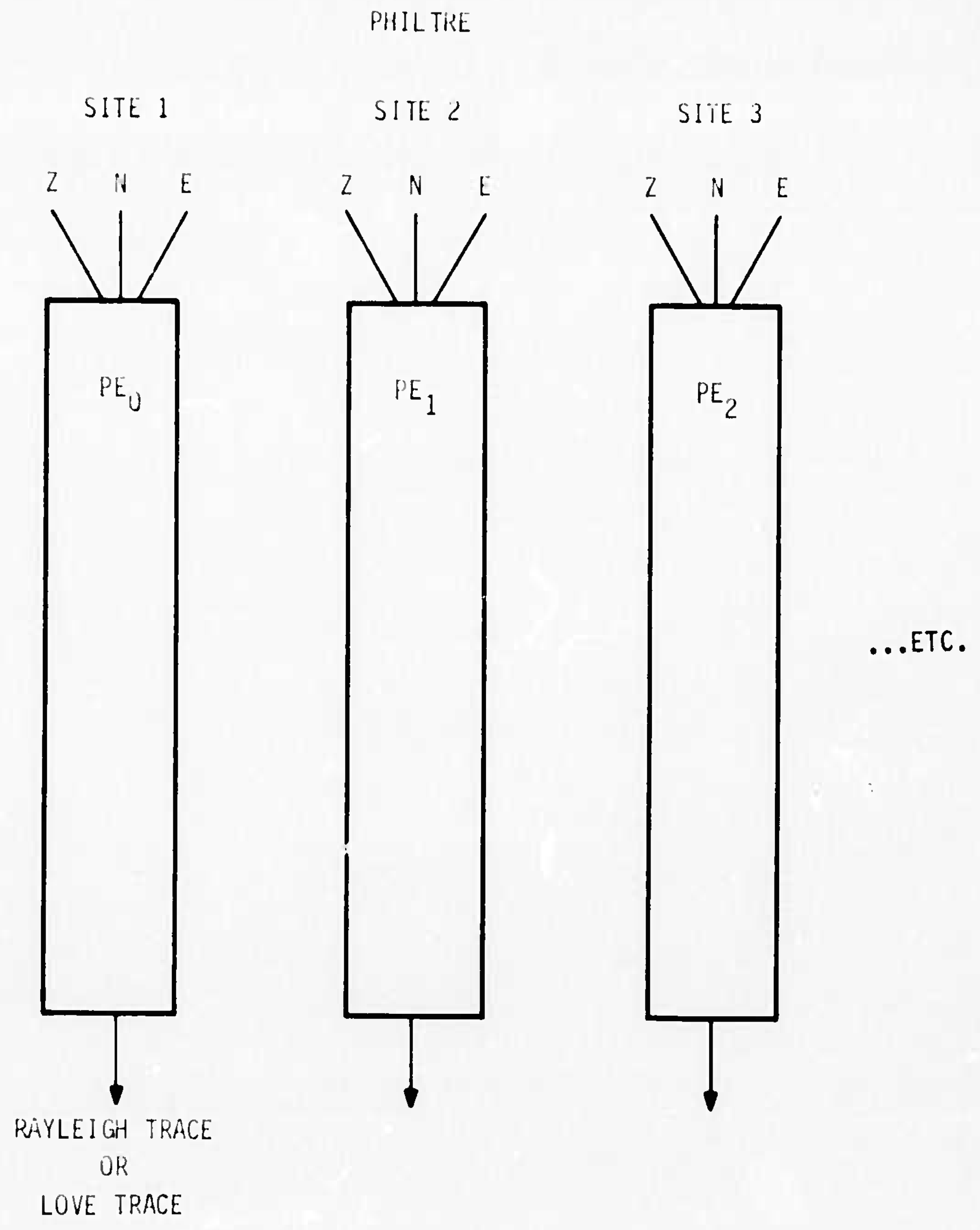

Figure 6. Suggested Data Schemes for PHILTRE.

Maximum Likelihood f-k Spectra

A maximum likelihhod f-k spectrum is the mapping of the power passed by a set of maximum likelihood filters in the plane. A maximum likelihood filter is an optimum filter which is constrained to pass a plane wave in the direction to be looked at while rejecting all the rest of the energy present, in a least mean square sense. It has the mathematical form for a given frequency

$$P(\vec{K}) = \frac{1}{\vec{u}\,\Phi^{-1}\,\vec{u}^{t}}$$

where Φ is the power spectral matrix of the sensors, K is the wavenumber and u is a vector

$$\vec{u} = (e^{i\vec{K}\vec{r}_1}, e^{i\vec{K}\vec{r}_c} \ldots e^{i\vec{k}\vec{r}_n}).$$

The position vector of the i'th sensor of the array is $\vec{r}_i$.

The maximum likelihood f-k spectrum is one of a wide family of high-resolution spectral estimators. It is characterized by reduced side lobes and higher resolution as manifested in the reduction of the width of the main lobe when compared to the simple frequency domain beam used in FKCOMB. The processor requires the estimation of the inverse of the input spectral matrix; there are fast practical ways to make this estimate, after which the multiplications with the various $\vec{u}$ vectors can be done rapidly by using all 64 parallel processors. The parallel feature can also be used to Fourier transform the individual seismic trace segments simultaneously. Algorithms are available to estimate the inverse of the array spectral matrix without actually inverting a matrix (J. W. Woods, personal communication, 1972).

If the detection of surface waves from a known epicenter is desired, the range of search in the $\vec{k}$ plane is reduced. Moreover, the absolute value of k is fixed for a given frequency, since the surface wave phase velocity for a given frequency at a given array site can be determined. Matched filtering or PHILTRE can be used as preprocessors to this processing scheme to utilize the dispersion and/or the particle motion characteristics of the signal and reduce the false alarm rate. The most practical way to use the ILLIAC computer is to apply sixty-four $\bar{u}$ vectors simultaneously using the same estimate of the computation by a factor of 64 relative to sequential processing and is the most efficient for the computation of finely spaced values in the f-k plane needed by this high-resolution process. A flow diagram in Figure 7 shows how the unique parallel computing feature of ILLIAC can be used to increase the efficiency of computing maximum likelihood f-k spectra.

FKCOMB

FKCOMB is a fast f-k analysis program that was first used in an automatic processing system for microbaragraph data (Smart and Flinn 1971). It has since been adopted for use with LP seismic data. It computes and finds the maximum of the function

$$P(\omega,k) = \left| \sum_{n=1}^{N} \{A_n(\omega) \exp[i\phi_n(\omega)]\} \cdot \exp(ik \cdot r_n) \right|^2$$

which is essentially the power in the frequency domain beam. Here ω is the angular frequency, $A(\omega) \exp[i\phi_n(\omega)]$ is the Fourier transform of the n'th seismic trace, N is the number of traces, and $\exp(ikr_n)$ are the components of the vector $\bar{u}$ in the previous section. The maximum of the function can be associated with the presence of a signal. The F test is used to determine whether a signal is present.

The methods take advantage of the fact that the signal-to-noise ratio varies with frequency, so beamforming is done frequency by frequency. Also, by staying in the frequency domain a great many beams can be examined rapidly, the number being limited only by the resolution cell of the array response. The low resolution of the process is actually an advantage when one desires to search f-k space rapidly, since the wide main lobe of the process enables one to use a wide grid spacing in the search.

The azimuth and velocity of a signal need not be assumed: one merely accepts the beam with maximum power. This fact is important for signals such as long-period seismic surface waves, which not only are dispersive (i.e., their phase velocities vary with frequency) but whose arrival azimuth may also vary with frequency because of lateral inhomogeneities in their paths.

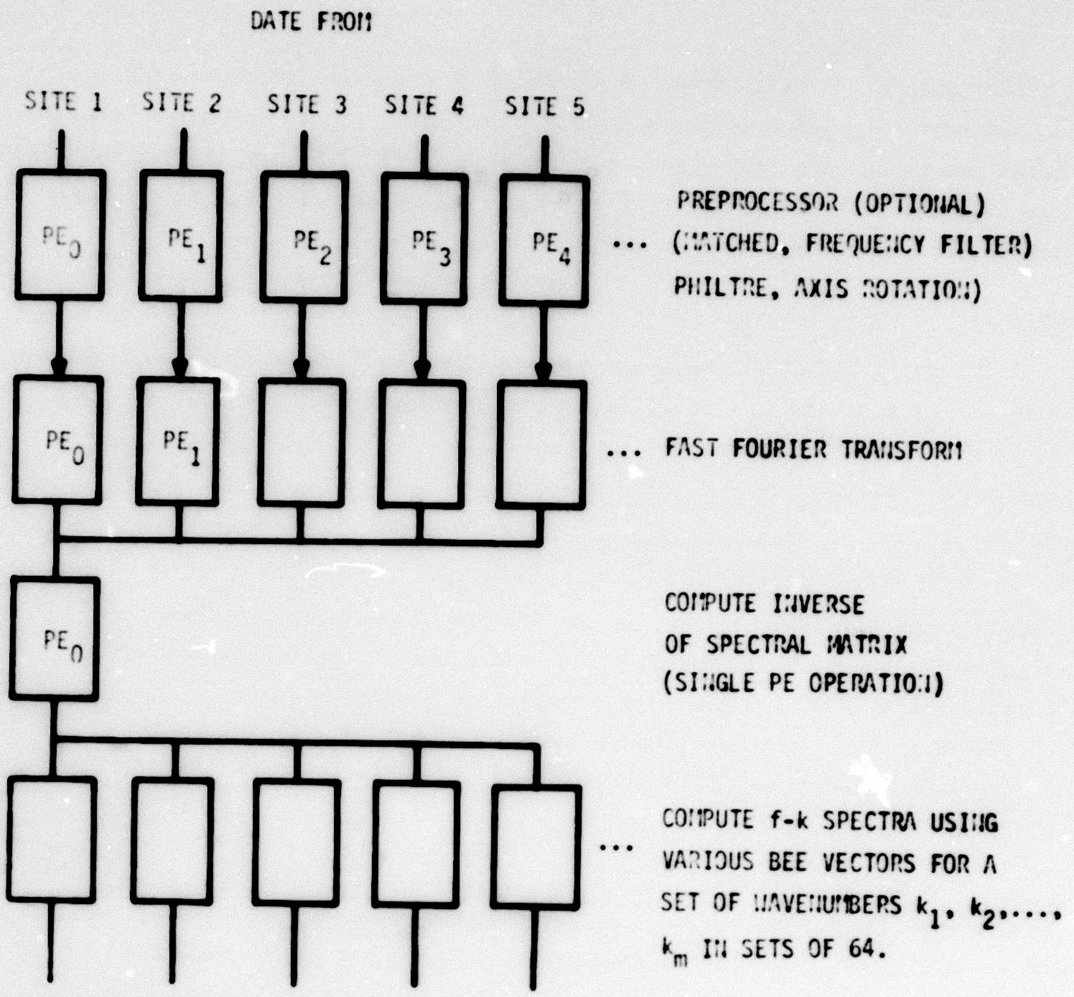
DATE FROM
SITE 1
SITE 2
SITE 3
SITE 4
SITE 5
PE0
PE1
PE2
PE3
PE4
... PREPROCESSOR (OPTIONAL) (MATCHED, FREQUENCY FILTER) PHILTRE, AXIS ROTATION)
PE0
PE1
... FAST FOURIER TRANSFORM
PE0
COMPUTE INVERSE OF SPECTRAL MATRIX (SINGLE PE OPERATION)
... COMPUTE f-k SPECTRA USING VARIOUS BEE VECTORS FOR A SET OF WAVENUMBERS k1, k2,..., km IN SETS OF 64.

Figure 7. Computation of Maximum Likelihood f-k Spectra.

Since the main advantage of the FKCOMB method is the possibility of fast search in the wavenumber vector space at a given frequency, changing frequencies as the search requires, we programmed the processor to operate on sixty-four successive time windows. This uses the ILLIAC most effectively for signal detection. The other type of application, searching sixty-four frequency levels simultaneously on the same time window, is not so efficient, since not all of the frequency bands may be needed for the search in a given iteration.

Details about the programming of FKCOMB will be given in the latter part of this report.

FKCOMB ALGORITHM DESIGN

General

The FKCOMB algorithm can be divided into the following steps:

1. Input raw long period data. Separate it by array. Extract the long period data samples and the timing words associated with those samples.

2. Divide the input into time windows. As originally input the data is ordered in the following manner:

T(1,1),T(2,1),...T(N,1), T(1,2),T(2,2),...T(N,2), T(1.S),T(2,S),...T(N,S)

where T(i,j) represents the data sample from channel i at time j, N is the number of channels and S is the number of time periods. After division into time windows the data is ordered as follows:

T(1,1),T(1,2),...T(1,TW),T(2,1),T(2,2),...T(2,TW),...T(N,1),T(N,2),...
T(N,TW),T(1,TW+1),T(1,TW+2),...T(1,2TW),...

where TW is the time window length and T(i,j) and N are as above.

3. The data is converted from the raw data format to the internal representation of the machine used. Glitches or spikes are removed and dead or noisy channels are detected and removed. (Portions of this step may be performed before step 2.)

4. A fourier transform is applied to each time window. After FFT the data is arranged as follows:

F(1,1,1),F(1,1,2),...F(1,1,TW),F(1,2,1),F(1,2,2),...F(1,2,TW),...
F(1,N,1),F(1,N,2),...F(1,N,TW),F(2,1,1),F(2,1,2),...F(2,1,TW),
F(2,2,1),F(2,2,2),...F(2,2,TW),...

where F(i,j,k), the Fourier transform output, represents frequency k, channel j, time window i.

5. Re-order the data so that it is arranged by frequency. It is then arranged as follows:

F(1,1,1),F(1,2,1),...F(1,N,1),F(1,1,2),F(1,2,2),...F(1,N,2),...
F(1,1,TW),F(1,2,TW),...F(1,N,TW),F(2,1,1),...

where F(i,j,k), TW, and N are defined as above.

6. Search frequency - wavenumber space for power maxima.

Data Editing Module One (DEM1)

Since step one is a process common to all seismological algorithms and because large input/output buffers are required it was coded as a stand alone module. The input to this module (DEM1) is the raw data as read from a low rate tape. The format of input records is shown in Figure 8. The output consists of several files, one per array, containing only the data samples applicable to long period processing. The data movement required to isolate and properly structure this data is nonparallel. There are no general structures repeated often enough to allow efficient use of the ROUTE instruction. The CU is used to move one word at a time between input buffers and output buffers. (Actually the BIN instruction is used to move blocks of eight words to the CU.) A description of each array format is given in the block data subroutine initialization of the vector CNTRL.

The reordering of data in steps 2 and 5 are not required if all data is available on a random access device, since it reflects the order in which the data will be accessed. It is necessary on ILLIAC since the size of core and long disk access time prohibit random access.

Assuming approximately 20 channels for each of three arrays, each sampling at the rate of once per second, one twenty four hour tape contains:

3 arrays/sample * 20 channels/array * 1 sample/second *
60 seconds/minute * 60 minutes/hour * 24 hours/day = 5,184,000 channels.

Moving each sample involves two memory accesses (one load and one store). Given that a memory access from the CU takes approximately .5 microseconds, the total time spent in memory accesses by DEM1 to process twenty four hours of data is on the order of 5 seconds. This is small enough that more complicated algorithms which may have permitted use of the ROUTE instruction were not considered.

Twenty four hours of data is approximately ten to the eight bits. In order to read these into core without losing a great deal of time waiting for disk access a buffer of 128 rows (512,000 bits) of core is used. 200 disk accesses are required for input. This takes up to eight seconds. Since

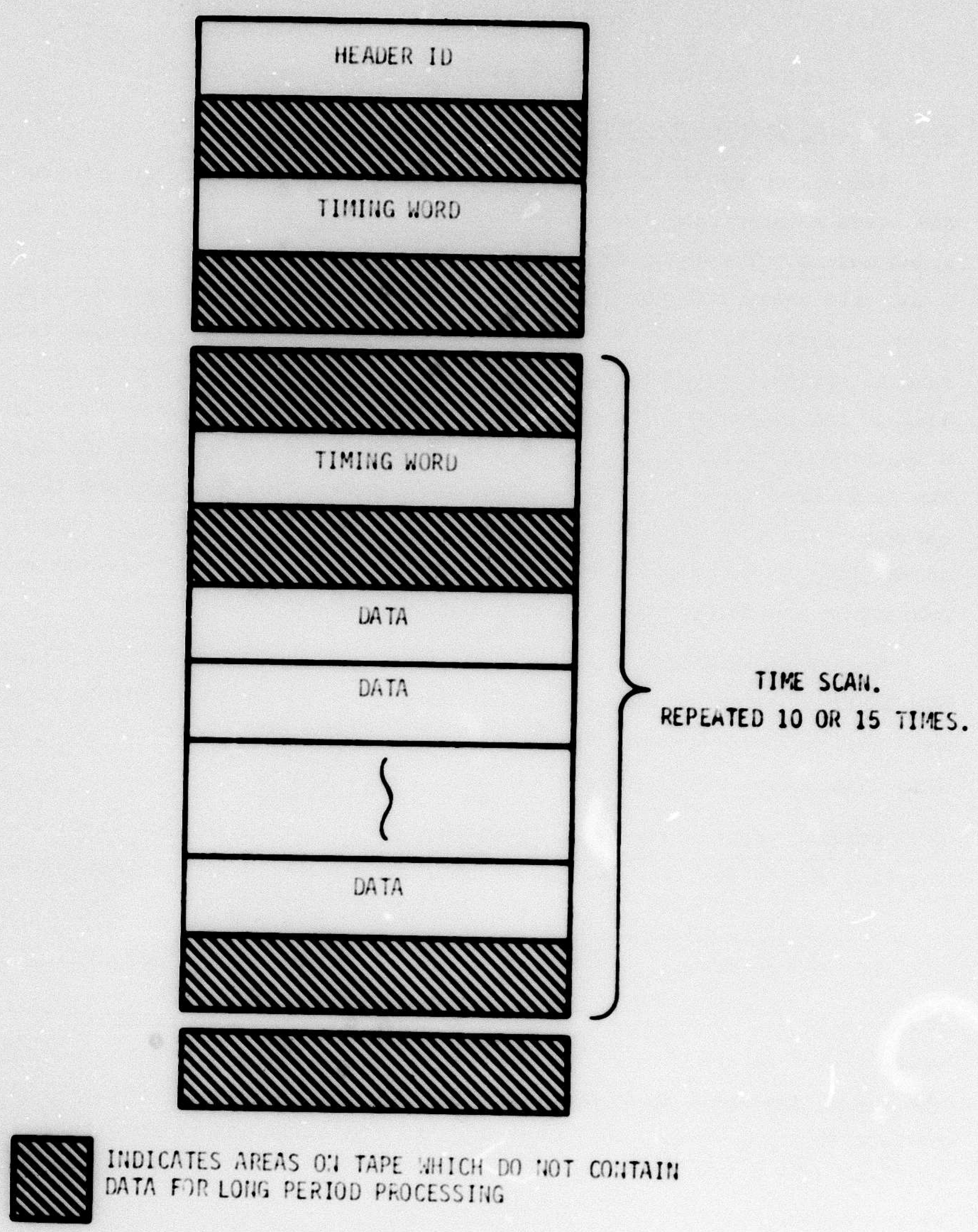

Figure 8. SDAC Lowrate Tape Format.

there are several output files, the output buffers are somewhat smaller. The total size of the output is smaller, since at least half of the input is not used in long period processing. The I/O time spent in output is therefore approximately equal to that spent in input even though the output buffers are smaller.

The actual movement of data by DEM1 is done within three nested loops. The outermost loop is gone thru once for each input record.

The next inner loop is gone through once for each time scan in each record. The innermost loop is gone thru once for each channel per time scan. All buffering is handled by an input routine and an output routine called once for each channel to transfer. In order to reduce overhead spent in subroutine calls it may be necessary to recode calls on these routines as in line code.

DEM1 transposes data in a serial fashion. It is coded so as to minimize time lost in disk and memory accesses. It puts array data in a standard format to reduce the size of the data and allow the straight forward execution of subsequent modules.

Data Editing Module Two (DEM2)

Steps 2 through 5 are performed by DEM2. The primary reason that this module was coded separately from step 6 was to shorten coding and debugging time. The relatively small amount of core memory available in each PE would necessitate the overlaying of various vectors used by step 6 and those included in DEM2 if all were included in one module. The I/O times spent writing the output from DEM2 and reading it in before step 6 would be saved, but this time is estimated to be less than 5 seconds.

Steps 2 through 5 are performed one time window at a time. A complete multi-channel time window is taken through steps 2 through 5 and the resulting output placed in an output buffer before the next time window is processed.

One multi-channel time window consists of approximately 20 channels of up to 512 samples each or approximately 10,000 data items. During step 2 it is impossible to include a complete multi-channel time window within one processing element memory. It is possible to include a single channel time window within one processing element memory, but due to the variable number of channels used for each time window, keeping track of which channels and time windows have been processed is complicated, though feasible.

An alternate approach is to use 64 PE's to process each time window. An FFT routine is available (written at the University of Illinois) which utilizes all 64 PE's to perform one FFT which runs very close to 64 times faster than one PE would do. Conversion to floating point involves no interaction between processing elements. Deglitching involves the comparison of each sample with the previous and next sample. With this data arrangement these samples are in adjacent PE's and the ROUTE instruction can be effectively utilized. The original structuring of the data into time windows (step 2) and the final transposition (step 5) are each performed serially by the CU, so are not affected by the data arrangement chosen. Spreading time windows across PE's was the approach chosen for steps 2 through 5.

Step 2 thus consists of extracting timing information from the input and, using this information, form time windows. Each time window is spread across the PE's, occupying between one and eight rows per channel, depending upon the time window size in use. Overlapping of time windows is performed by retaining whatever part of the most recent time window is still of interest and using the ROUTE instruction to back it up properly. For this reason, the buffer in which time windows are built is alternated between two halves of an array so that the last time window built is not overwritten.

Conversion to internal floating point format is the first step performed once the time windows have been formed. Each PE converts all samples within its memory and no inter PE communication is required. Deglitching is performed by routing the values from adjacent PE's and adjusting them if a glitch is encountered. (See program documentation for exact procedure.) The rowsum procedure described in section 2 is used in the variance calculation, since a summation across PE's is required. The FFT is then performed and the frequencies prepared for output. Due to the fact that not all frequencies output by FFT are of seismological interest, the output from step 4 is much smaller than the input to step 4. This data reduction is significant in that after FFT a multi-channel time window consists of approximately 20 channels and 20-30 frequencies and will fit within one processing element memory. Placing each separate time window wholly within one PEM is very convenient since the search of frequency wavenumber space for one time window is completely self contained and independent of other processing (see Figure 9). In step 5, the FFT output is written into one PEM of an output core buffer. This is done serially by the CU. When the buffer is full it is written to disk.

FKCOMB Algorithm

Since each PE completely contains one time window after step 5, the algorithm used in the search of frequency wavenumber space is essentially the same as that used in the serial version. The program reads the data file created by DEM2, which has been arranged as shown in Figure 9. Each time window contains the frequencies of interest and the algorithm is executed in parallel on the data. A search for maximum power is made on a coarse grid and then a series of finer grids is searched simultaneously in all processing elements until a maximum is found. In a given PE the mode for that PE is disabled until a maximum is found in all other PE's. The Fisher statistic, period, signal azimuth and velocity, and associated parameters are calculated and stored, and the process is continued on the next time window of data. The design of the algorithm was straightforward, and the reader is referred to the program documentation (Kerr and Wagenbreth, 1974) for a more detailed discussion of the software.

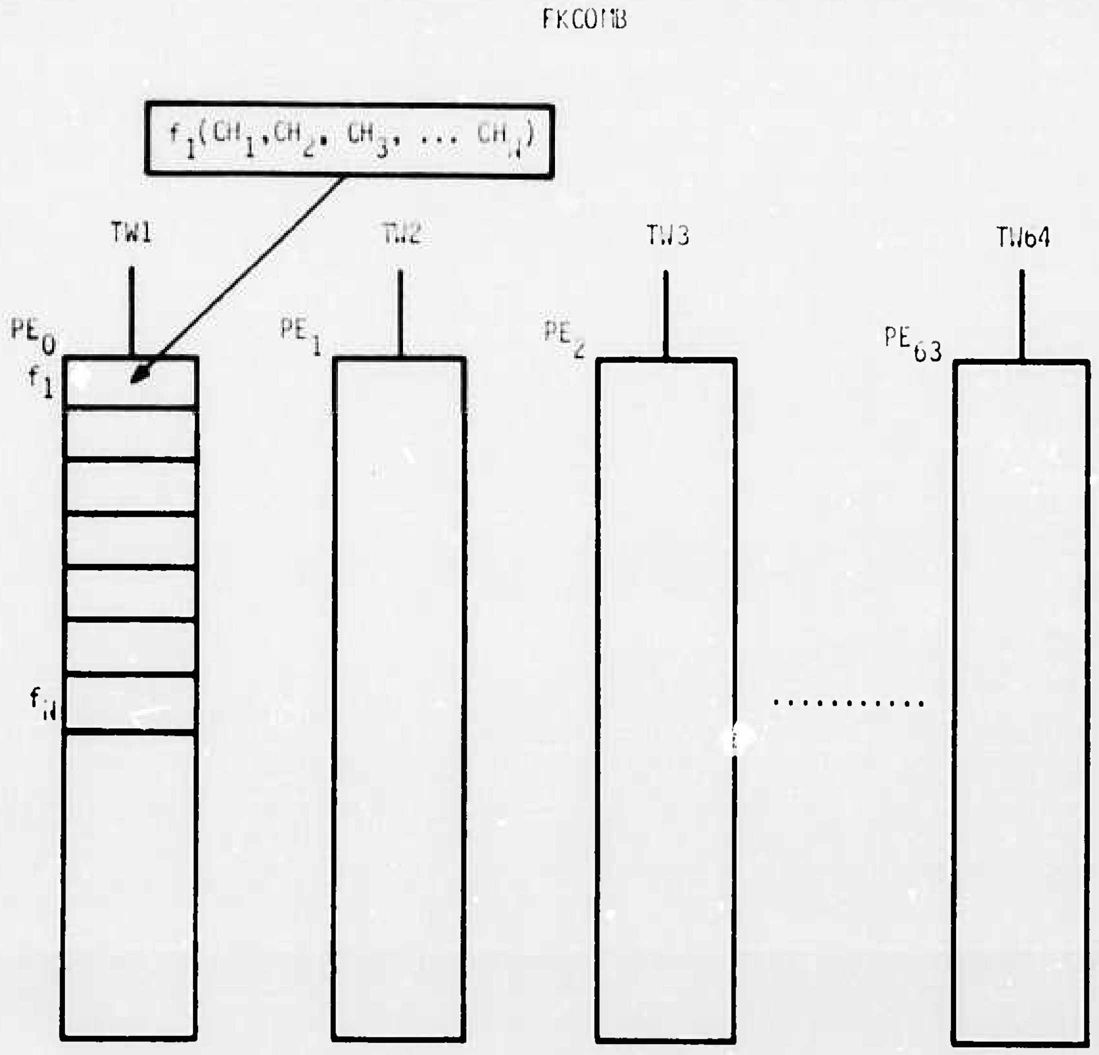

Figure 9. Input Format for FKCOMB.

The operational aspects of using ILLIAC differ significantly from those of other machines. In addition to the parallel architecture, there are two other characteristics which are important considerations for the user of the ILLIAC system. First, all of the support software such as editors and compilers run on processors other than ILLIAC. There is currently support software available on DEC, IBM, and Burroughs machines. The choice of which machine and software to use is an integral part of system development, for ILLIAC is accessible only via the ARPA Network and is routinely used remotely. The bandwidth, availability, and reliability of the network directly affect the performance of the ILLIAC system as seen by a user.

Program Entry and Storage

A basic requirement for any long-term coding effort is a reliable file system permitting easy access and modification of source codes. Two basic options were available in using the ILLIAC system. One, used by several ILLIAC coding efforts, is to maintain files on a host computer and transfer the files to the ILLIAC system via the ARPA Network whenever necessary. The second is to utilize the Tenex file system and editors included in the ILLIAC system. The first approach required frequent ARPA Network transfers and a reliable and economical host machine. Since such a host was not available to SDAC, the Tenex file system was used, and was found reliable and convenient. No work was lost due to disk or file system failures during the duration of this project. The editor DED fulfilled all requirements regarding both modification and examination of source files.

Languages

Three languages are available for preparation of ILLIAC code. There are two high-level languages, GLYPNIR and CFD, and an assembly language, ASK. The large amount of coding necessary made the use of assembly language impractical except for specific portions where bit manipulation or efficiency made it a necessity. The majority of coding was done in high-level language.

A comparison of the syntax and semantics of GLYPNIR and CFD revealed the following significant differences:

1. Ease of understanding - CFD looks much like FORTRAN and is easily interpreted or learned by a scientific programmer. GLYPNIR resembles ALGOL and is somewhat more confusing and difficult to learn.

2. Ease of coding - Once learned, GLYPNIR permits faster and clearer coding than CFD. GLYPNIR's macro facilities are a convenience not provided by CFD. GLYPNIR has some higher level constructs which require several CFD statements to implement.

3. Efficiency - CFD produces more efficient code than GLYPNIR in many instances.

The two languages are very similar in their treatment of unique ILLIAC characteristics and both provide all facilities necessary for the implementation of seismic analysis programs. Certain types of code are better suited to one language than the other, but consideration of the syntax and semantics alone indicated no clear preference.

The choice of language ultimately depended upon the support and availability of GLYPNIR and CFD. GLYPNIR is supported by the Institute for Advanced Computation as part of the ILLIAC system. It is implemented on a Burroughs 6700 located at the ILLIAC computer center and must be accessed via the ILLIAC batch queue (as discussed below). CFD is implemented on the IBM 360/67 located at NASA Ames Research Center. It is accessible routinely via the ARPA Network. The source for CFD is transportable and a version of CFD is available on the UCLA IBM 360/91. Also supported on the Ames IBM 360/67 is a translator, CFDX, which translates CFD to Fortran. With some modification due to I/O differences and inserted assembly language code, CFD programs may be translated to IBM Fortran. The translator is designed to generate code equivalent to that generated by CFD for ILLIAC. Programs may then be debugged and tested on an IBM 360 rather than on ILLIAC. CFDX is not designed to replace ILLIAC in production mode since the FORTRAN generated by writing a CFD program and translating it will not be nearly as efficient as coding in FORTRAN directly.

Due to the superior availability of CFD and the existence of the CFDX translator the decision was made to implement the FKCOMB algorithm in CFD.

As actually experienced, the availability of CFD was not as good as had been hoped, for several reasons: First, availability of the Ames IBM 360 is very poor. Between the hours of 8:00 am and 12:00 midnight (PST) use of the machine by non-priority accounts is restricted. During the eight remaining hours, the requirement that both the ILLIAC Tenex system and the Ames IBM 360 be operational for file transfers caused much lost programmer and computer time. The hours were also inconvenient. The UCLA version of CFD, due to lack of overlays, requires 400K core and runs in a slow queue (6-8 hours turnaround). Efforts to implement CFD on the SDAC IBM 360/44 were frustrated due to incompatability of the operating systems of the IBM 360/44 and the IBM 360/67. The large core requirement also posed a serious problem. It was found that the effort required to implement CFD at SDAC would not be worth the convenience of an in-house compiler. The availability of ILLIAC was sufficient (see below) to make the use of the CFDX translator uneconomical due to the alterations necessary to accommodate I/O differences and inserted assembly language statements.

Run Procedures

Coding of the FKCOMB algorithm in CFD began in April 1974. What follows is the set of procedures developed for the day to day process of running and debugging an ILLIAC program, along with experience gained and observations made during the use of these procedures.

The primary site at which compiles were done was the Ames IBM 360/67. A CFD restriction is that all subroutines must be separately compiled. Our code was divided into three programs, each consisting of a main driver and four to six subroutines. Initially all subroutines had to be compiled, but thereafter only those with code modifications required compilation. Compiling a module consists of four steps. First, after having logged in on the Ames 360/67, the source file is transferred over the ARPA network from the I4-Tenex File System, where the source files are maintained, to the Ames 360. This process is done interactively and typically takes one to ten minutes of real time, depending upon the length of the source and the load average on each machine. Approximately one out of three transfers terminated abnormally and had to be reinitiated. The failure rate increased greatly when the load on either machine was heavy. The next step is to initiate the CFD compiler. The time between the submission of a compile and its completion varied from five minutes to several hours, again dependent upon the machine load. After termination of the compile the listing generated by the CFD compiler is examined with the TSS editor, REDIT, to check for syntax errors or other abnormal termination. If errors are detected, they are corrected (being careful to make the same corrections to the original source at I4-Tenex) and the compile reinitiated. After a successful compile, the ASK assembly language source module is copied back to I4-Tenex via network transfer. This file is usually several times larger than the original source and the time taken to transfer the file is several times longer than that for the source. If several subroutines are to be recompiled, this process can consume several hours. When only small changes are necessary, this time can be saved by changing the assembly language code directly with the text editor at I4-Tenex, again being careful to make the same changes to the original source.

Once the necessary assembly language modules have been created, a batch job is submitted at I4-Tenex to perform the following tasks:

1. Assemble the ASK modules
2. Linkedit the resultant relocatable modules
3. Create a disk map file describing the actual layout of any ILLIAC disk areas to be used by this run
4. Allocate the map file created in the last step
5. Move any input files required to the appropriate ILLIAC disk area
6. Run the ILLIAC code
7. Move any output from the appropriate ILLIAC disk area to the I4-Tenex file system
8. Release the ILLIAC disk areas used.

Turnaround for ILLIAC batch jobs improved significantly from April to June 1974, but was always subject to unpredictable fluctuations and delays. In April no more than three or four turnarounds per week could be expected. By June this had increased to two per day if submitted taking into account the hours scheduled for batch jobs to be run. This required almost constant monitoring of the batch queue between the hours of 9 am and 9 pm/EDT.

Turnaround was significantly improved during the month of June by relocating to California and working at the ILLIAC site. Considerable effort was made by the ILLIAC user support group during this time to insure that SDAC jobs were given priority, and considerable progress was made during this period.

Analysis of the results of an ILLIAC run is possible by three methods. The primary and by far most convenient means is an unsophisticated form of formatted I/O called "DISPLAY". The output is readable, and predicted answers can be checked against actual results with its use. In the case of unexpected results where suitable displays were not generated to provide clues to the source of the error, we found it necessary to examine the dump files.

CONCLUSIONS AND RECOMMENDATIONS

Seismic Processing on ILLIAC

The ILLIAC computer programmed to perform seismic processing on large data bases can be a valuable tool in the development of seismic event detection and discrimination procedures. It is feasible to implement some existing algorithms on the ILLIAC which are not currently used to process large data bases, or some algorithms which are proposed but not tested due to a lack of computing power. Our experience with one algorithm (FKCOMB) which is representative of seismic analysis programs shows that a major benefit of the ILLIAC to seismic processing is its ability to operate in parallel on sixty-four different data streams, thereby reducing the time required to process large data bases. Efficiently arranging these data streams for the processing element memories is an important consideration for designing any seismic algorithms for the ILLIAC.

It is feasible to program ILLIAC to perform the algorithms reviewed in this study: convolution-recursive filtering, PHILTRE, matched filtering, beamforming, and maximum likelihood f-k estimation. Since a major factor in programming any of these algorithms is the data arrangment in core, a more detailed study of the data configurations for these algorithms would be needed to optimize the use of the computing power of ILLIAC. One algorithm (FKCOMB) was studied in detail and implemented on ILLIAC IV. Data editing schemes were devised for FKCOMB which can be used with appropriate modifications for all the seismological algorithms we reviewed.

Two independent uses for ILLIAC are suggested. First, FKCOMB and other algorithms now used selectively could be run routinely on larger data bases to better provide the services they already give on conventional machines. Second, experimental methods impractical to test via conventional machines could be tested on ILLIAC. The experience of implementing FKCOMB illustrates that the design and coding of new algorithms for ILLIAC is not significantly more difficult than for serial machines. The only phase not experimentally explored by this effort are the operational problems of manipulating the large amounts of data involved in routine processing of long and short period data on ILLIAC.

To maximize efficiency, the time consumed executing analysis algorithms should be significantly greater than the time required for data editing. A combination of algorithms such as matched filtering and FKCOMB require an order of magnitude more processing time on a given memory load than FKCOMB alone, and would thereby utilize ILLIAC more efficiently.

Programming FKCOMB

The following points represent our findings in developing FKCOMB software on the ILLIAC.

1. Faster and more reliable network file transfer between I4-Tenex and other hosts such as UCLA, Ames, TSS and SDAC would expedite the programming and use of the ILLIAC system.

2. There is no clear preference between CFD and GLYPNIR indicated by our experiences. The possibility of implementing CFD at SDAC or upgrading service at UCLA should be investigated, and an experiment made in the use of GLYPNIR before any long range decision is made.

3. Software debugging aids presently available for ILLIAC programs are minimal. Additional debugging aids would lessen the task of ILLIAC programming. User implementation of such aids on the SDAC host is feasible though at the cost of considerable effort.

4. We estimate that the time and effort required to design and code an ILLIAC program is no more than twice that required for a conventional machine. Due to the fact that ILLIAC is not fully operational at present and the necessity to handle the large amounts of data inherent in the use of an ultrafast machine, the time and effort required currently to debug an ILLIAC program may be as much as four times that required for a conventional machine.

5. Routine processing of 24 hours of long-period seismic data is not feasible at present due to the restricted availability of the ILLIAC processor and the inability at the system to handle the large amounts of data efficiently.

GLOSSARY OF TERMS

ALGOL - Higher level algorithm language on serial computers.

ALPA - Alaskan Long Period Array.

ASK - Assembly language for ILLIAC.

ASSEMBLE - Convert from mnemonic code to machine code.

BPI - Bits per inch, the unit for measuring magnetic tape recording density.

CFD - A Fortran-like higher level language developed by the Computational Fluid Dynamics group at NASA/Ames.

CFDX - A program which translates CFD code into Fortran code.

DEC - Digital Equipment Corporation.

DED - The text editing subsystem on I4-Tenex.

DEM1 - Data Editing Module 1, the first data editing program module.

DEM2 - Data Editing Module 2, the second data editing program module.

DISPLAY - Macros which invoke subroutines permitting print output of intermediate results.

DUMP - Hexadecimal representation of the contents of core and registers at termination of program execution.

EDT - Eastern Daylight Time.

FTP - File Transfer Protocol. A protocol for file transfer from one ARPA Network computer to another.

FFT - Fast Fourier Transform, a Fourier transformation program.

FKCOMB - Frequency-k wavenumber combination, the algorithm implemented on ILLIAC discussed in this report.

FORTRAN - Higher level language on serial computers.

GLYPNIR - The higher level language implemented for the ILLIAC.

IBM - International Business Machines Corporation.

ILLIAC IV - The parallel processor.

ILLIAC System - The complete computer system consisting of the parallel processor, PDP-10's, B6700, UNICON, several PDP-11's, and their software operating system.

I4-Tenex - Designates the PDP-10 of the ILLIAC system, which is the ARPA Network host for the ILLIAC system. I4 designates the system with respect to the ARPA Network.

LASA - Large Aperture Seismic Array in Montana.

LINKEDIT - Merge several machine-readable subroutines into one executable program.

NASA/Ames - (also Ames) National Aeronautical and Space Administration, Ames Research facility.

NEP - Network Event Processor.

NORSAR - Norwegian Seismic Array.

PE - Processing Element of the ILLIAC, 64 of which operate in parallel.

PHILTRE - An algorithm using a non-linear weighting scheme of Fourier spectral components to enhance Love or Rayleigh waves.

PST - Pacific Standard Time.

REDIT - The text editor for TSS.

SDAC - Seismic Data Analysis Center.

TENEX - The operating system for the PDP-10.

TSS - Time sharing system for IBM 360/67.

UCLA - University of California at Los Angeles.

UNICON - Laser mass storage system developed by Precision Instruments.

REFERENCES

Alexander, S. S., and D. B. Rabenstine, 1967a, Detection of surface waves from small events at teleseismic distance: SDL Report No. 175, Teledyne Geotech, Alexandria, Virginia.

Alexander, S. S., and D. B. Rabenstine, 1967b, Rayleigh wave signal-to-noise enhancement for a small teleseismic using LASA, LRSM and observatory stations: SDL Report No. 194, Teledyne Geotech, Alexandria, Virginia.

Blandford, R. R., 1971, An automated event detector at TFO: SDL Report No. 263, Teledyne Geotech, Alexandria, Virginia.

Capon, J., 1969, High-resolution frequency-wavenumber spectrum analysis, Proc. IEEE 57, 1408-1418.

CFD, A Fortran based language for ILLIAC IV, 1973, Computational Fluid Dynamics Branch, Ames Research Center, National Aeronautics and Space Administration.

ILLIAC IV Systems Characteristics and Programming Manual, 1971, Burroughs Corporation, Defense, Space and Special Systems Group.

Kerr, A. U., and G. Wagenbreth, A long-period processing package for ILLIAC IV, 1974 (in preparation).

Mack, H., 1972, Evaluation of the LASA, ALPA, NORSAR long period network: Seismic Array Analysis Center Report No. 6, Teledyne Geotech, Alexandria, Virginia.

Simons, R. S., 1968, PHILTRE, A surface wave particle motion discrimination process. Bull. Seism. Soc. Amer., 58, p. 629-637.

Smart, E., 1971, Erroneous phase velocities from frequency wavenumber spectral sections: Geophys. J. Roy. Astr. Soc., 26, p. 247-254.

Smart, E. and E. A. Flinn, 1971, Fast frequency-wavenumber analysis and Fisher signal detection in real time infrasonic array data processing: Geophys. J. Roy. Astr. Soc., 26, p. 279-284.

Stevens, J. E., 1971, A fast Fourier transform subroutine for ILLIAC IV: C.A.C. Document No. 17, Center for Advanced Computation, University of Illinois at Urbana-Champaign, Urbana, Illinois 61801.

System Guide for the ILLIAC IV User, 1974, Institute for Advanced Computation, Ames Research Center, Moffet Field, California 94035.

von Seggern, D. H. and P. Sobel, 1974, Performance of the PHILTRE processor at low signal to noise ratios (in preparation).

Woods, J. W. and P. R. Lintz, 1972, Plane waves at small arrays: Geophysics, 38, p. 1023-1041.