DIGITAL REAL-TIME SPECTRAL ANALYSIS

by

Glenn David Bergland

A Dissertation Submitted to the
Graduate Faculty in Partial Fulfillment of
The Requirements for the Degree of
DOCTOR OF PHILOSOPHY

Major Subject: Electrical Engineering

Approved:

Signature was redacted for privacy.

In Charge of Major Work

Signature was redacted for privacy.

Head of Major Department

Signature was redacted for privacy.

Dean at Graduate College

Iowa State University
Of Science and Technology
Ames, Iowa

1966

TABLE OF CONTENTS

I. INTRODUCTION

A. Motivation

Methods of obtaining spectral estimates from a time function have been useful in speech analysis (11), echo-ranging systems (1), seismic exploration (1), ocean-wave forecasting (8), meteorology (2), and in many other areas.

In some cases the process being analyzed is relatively stable so that a fixed frequency band-pass filter can be used to operate on the signal after it is mixed with a variable frequency sine wave. In effect, the input signal is frequency-swept through a fixed-frequency filter. This type of analysis produces spectra with high resolution but requires more than a minute to analyze a few seconds of the signal.

Faster types of heterodyne analyzers are available where the signal is time compressed before passing through the band-pass filter, but these require rather elaborate schemes to obtain high resolution and still operate in real time.

A stationary band-pass filter bank is another type of analyzer which operates in real time. The main problem associated with this type of analyzer is that an impractical number of filters must usually be used to obtain good resolution.

Frequently, when the analysis does not have to be done in real time, the time signal is sampled and then analyzed on a digital computer. Using the "indirect" approach discussed by

Blackman and Tukey (2), the data record is weighted by the desired lag window, the autocovariance function is found, and the Fourier cosine transform is taken. If this is performed properly, the spectral estimates will be considerably smoothed and moderately stable.

The "direct" method discussed by Blackman and Tukey involves taking the Fourier transform of the original time series and then forming the power spectrum from the Fourier transform. The effect of different data windows can be obtained either by initially weighting the data or, equivalently, by performing a convolution involving the corresponding spectral window and the spectral estimates obtained from the unmodified data.

Thus the "indirect" method corresponds to a Fourier transform of an average of products, while the "direct" method corresponds to squaring a Fourier transform. Both of these can be done on a general purpose digital computer but seldom in real time.

B. Statement and Scope of the Problem

The principal objective of this investigation was to find special purpose digital computer configurations which could be used in applying the "direct" method of finding spectral components. The machines to be discussed essentially compute the complex Fourier transform of a time sampled signal while the samples are being taken. If estimates of the "power" spectrum

of this signal are required, they are obtained by squaring the Fourier transform and smoothing the resulting spectral estimates by convolving with a Hanning spectral window, a Hamming spectral window (2), or any other spectral window.

In a special purpose machine, this convolution could be built into the analyzer quite conveniently. The main portion of this report, however, is concerned with the digital machines which form the initial complex unsmoothed spectral estimates.

II. "DIRECT" CALCULATION OF THE FOURIER TRANSFORM

The Fourier transform of a time sampled signal can be found from the original time function in several ways. The approach discussed here starts with the initial time function and results in the complex Fourier series expression considered by Cooley and Tukey (5). This expression, in most cases, can be computed while the function is being sampled through the use of a special purpose machine.

A. The Fourier Transform

The Fourier transform of a time sampled signal is found heuristically in the following manner.

Given a record of the time function T seconds long, a periodic function A(t) can be formed which is identical to the input function over (0,T) and satisfies the relation A(t + T) = A(t). The components of the Fourier transform of this periodic extension of the original time function will be used as estimates of the spectral components which were present during the 0 to T time period.

Thus A(t) can be expressed as a Fourier series of the form

$$A(t) = \sum_{j=-\infty}^{\infty} X'(j) e^{-i(j\omega_0 t)} \qquad (1)$$

where

$$X'(j) = \frac{1}{T}\int_0^T A(t)e^{i(j\omega_0 t)}dt \qquad (2)$$

each being the Fourier transform of the other (7). After being sampled, A(t) can be expressed as

$$A(t)_{\text{sampled}} = \sum_{k=0}^{N-1} A(k\Delta T)\,\delta(t-k\Delta T) \qquad 0\leq t\leq T \qquad (3)$$

where $T = N\Delta T$ = the length of the record and ΔT = the sample period (3). Therefore,

$$X'(j) = \frac{1}{T}\int_0^T \left[\sum_{k=0}^{N-1} A(k\Delta T)\,\delta(t-k\Delta T)\right] e^{i(j\omega_0 t)}dt \qquad (4)$$

where

$$\omega_0 = \frac{2\pi}{T} \qquad (5)$$

and

$$X'(j) = \frac{1}{T}\sum_{k=0}^{N-1} A(k\Delta T)\int_0^T \delta(t-k\Delta T)\,e^{i(j\omega_0 t)}dt \qquad (6)$$

Thus,

$$X'(j) = \frac{1}{T} \sum_{k=0}^{N-1} A(k\Delta T) e^{i(j\omega_0 k\Delta T)} \tag{7}$$

Note that $\omega_0 = 2\pi/T = 2\pi/N\Delta T$ and denote $A(k\Delta T)$ by $A(k)$. Then

$$X'(j) = \frac{1}{T} \sum_{k=0}^{N-1} A(k) \left[e^{2\pi i/N}\right]^{jk} \tag{8}$$

or

$$X'(j) = \frac{1}{T} \sum_{k=0}^{N-1} A(k) W^{jk} \tag{9}$$

where $W = e^{2\pi i/N}$ and $j = 0,1,\cdots,N-1$.

B. Obtaining Spectral Estimates

The $|X'(j)|^2$ values may be used as estimates of the power spectrum as they stand, but the statistical stability of these estimates may not be acceptable. Davenport and Root (6) showed that for real Gaussian random signals in the limiting case of T approaching infinity, the expected value of the calculated spectral estimates approached the true spectrum but the variance of these estimates did not approach zero.

Forming the $|X'(j)|^2$ terms is equivalent to Blackman and Tukey's "direct" method of spectral analysis using the $D_0(\tau)$ data window (2). If another type of data window is preferred for smoothing purposes, the corresponding spectral window can

be convolved with the initially obtained power spectrum. When a Hanning spectral window is used, the smoothed spectral estimates can be found from the $|X'(j)|^2$ estimates using the following expressions

$$\begin{aligned} U(0) &= 0.5|X'(0)|^2 + 0.5|X'(1)|^2 \\ U(r) &= 0.25|X'(r-1)|^2 + 0.5|X'(r)|^2 + 0.25|X'(r+1)|^2 \\ U(N-1) &= 0.5|X'(N-2)|^2 + 0.5|X'(N-1)|^2 \end{aligned} \tag{10}$$

where $r = 1,2,\cdots,N-2$.

For other spectral windows the coefficients are altered but the basic method remains the same.

III. THE RECURSIVE RELATIONS USED IN EVALUATING THE COMPLEX FOURIER SERIES

The evaluation of Equation 9 in its present form usually could not be performed on a real-time basis. Cooley and Tukey (5), however, have approached the problem in a slightly different manner and have substantially reduced the number of arithmetic operations required to evaluate a Fourier series, when N is a power of 2, through the use of a set of recursive equations.

Using the same type of approach, a set of recursive equations can also be formed whenever N can be expressed as the product of any set of integers.

The increase in efficiency and the parallel computation capability afforded through use of this approach result in making real-time digital spectral analysis practical.

A. Recursive Equations When $N = 2^m$

The recursive equations for the case of N being a power of two are presented in considerable detail by Cooley and Tukey in the April 1965 issue of Mathematics of Computation.

Cooley and Tukey considered the problem of evaluating a complex Fourier series of the form

$$X(j) = \sum_{k=0}^{N-1} A(k) W^{jk} \qquad (11)$$

where $W = e^{2\pi i/N}$, $j = 0,1,2,\cdots,N-1$.

Upon comparing these X values with the X' values of Equation 9 we see that the X values of Equation 11 are directly proportional to the X' values of the Fourier transform of the time sampled signal.

For evaluating Equation 11 when $N = 2^m$, Cooley and Tukey developed the following recursive equations:

$$A_p(j_0, j_1, \cdots, j_{p-1}, k_{m-p-1}, \cdots, k_0) \tag{12}$$

$$= \sum_{k_{m-p}} A_{p-1}(j_0, \cdots, j_{p-2}, k_{m-p}, \cdots, k_0) W^{(j_{p-1}2^{p-1} + \cdots + j_0)k_{m-p}2^{m-p}}$$

where

$$X(j_{m-1}, \cdots, j_0) = A_m(j_0, \cdots, j_{m-1}) \tag{13}$$

B. Recursive Equations When $N = r_1 r_2 \cdots r_m$

In this section Equations 12 and 13 are extended to the more general case of $N = r_1 r_2 \cdots r_m$.

First j and k must be expressed in the following form.

$$j = j_{m-1}(r_1 r_2 \cdots r_{m-1}) + j_{m-2}(r_1 r_2 \cdots r_{m-2}) + \cdots + j_1 r_1 + j_0 \tag{14}$$

$$k = k_{m-1}(r_2 r_3 \cdots r_m) + k_{m-2}(r_3 r_4 \cdots r_m) + \cdots + k_1 r_m + k_0$$

This allows Equation 11 to be written as

$$X(j_{m-1},j_{m-2},\cdots,j_1,j_0) = \sum_{k_0}\sum_{k_1}\cdots\sum_{k_{m-1}} A(k_{m-1},k_{m-2},\cdots,k_0)W^{jk} \tag{15}$$

Note that

$$W^{jk} = W^{j[k_{m-1}(r_2r_3\cdots r_m)+\cdots+k_0]} \tag{16}$$

but

$$W^{jk_{m-1}(r_2r_3\cdots r_m)}$$

$$= W^{[j_{m-1}(r_1r_2\cdots r_{m-1})+\cdots+j_0][k_{m-1}(r_2r_3\cdots r_m)]} \tag{17}$$

When the product in the exponent is formed, the term may be expressed in the following form

$$W^{jk_{m-1}(r_2r_3\cdots r_m)} \tag{18}$$

$$= \left[W^{(r_1r_2\cdots r_m)}\right][j_{m-1}(r_2r_3\cdots r_{m-1})+\cdots+j_1]k_{m-1}W^{j_0k_{m-1}(r_2\cdots r_m)}$$

Note that $r_1r_2r_3\cdots r_m = N$ and

$$W^N = (e^{2\pi i/N})^N = 1 \tag{19}$$

therefore the bracketed term of Equation 18 taken to any power is still equal to 1 and we have

$$W^{jk_{m-1}(r_2 r_3 \cdots r_m)} = W^{j_0 k_{m-1}(r_2 r_3 \cdots r_m)} \tag{20}$$

therefore

$$W^{jk} = W^{j_0 k_{m-1}(r_2 \cdots r_m)} \; W^{j[k_{m-2}(r_3 \cdots r_m)+\cdots+k_0]} \tag{21}$$

This allows Equation 15 to be written in the form

$$X(j_{m-1},\cdots,j_0) = \sum_{k_0} \sum_{k_1} \cdots$$

$$\sum_{k_{m-2}} \left[\sum_{k_{m-1}} A(k_{m-1},\cdots,k_0) W^{j_0 k_{m-1}(r_2 \cdots r_m)} \right] W^{j[k_{m-2}(r_3 \cdots r_m)+\cdots+k_0]} \tag{22}$$

If the expression in brackets is written as

$$A_1(j_0,k_{m-2},\cdots,k_0) = \sum_{k_{m-1}} A(k_{m-1},\cdots,k_0) W^{j_0 k_{m-1}(r_2 \cdots r_m)} \tag{23}$$

Equation 22 may be expressed as

$$X(j_{m-1},\cdots,j_0) \tag{24}$$

$$= \sum_{k_0} \sum_{k_1} \cdots \sum_{k_{m-2}} A_1(j_0,k_{m-2},\cdots,k_0) W^{j[k_{m-2}(r_3\cdots r_m)+\cdots+k_0]}$$

By applying Equation 19 again we see that

$$W^{jk_{m-2}(r_3r_4\cdots r_m)} = W^{(j_1r_1+j_0)k_{m-2}(r_3r_4\cdots r_m)} \tag{25}$$

This allows the innermost sum to be written as

$$A_2(j_0,j_1,k_{m-3},\cdots,k_0)$$

$$= \sum_{k_{m-2}} A_1(j_0,k_{m-2},\cdots,k_0) W^{(j_1r_1+j_0)k_{m-2}r_3r_4\cdots r_m} \tag{26}$$

leaving Equation 24 in the form

$$X(j_{m-1},\cdots,j_0) \tag{27}$$

$$= \sum_{k_0} \sum_{k_1} \cdots \sum_{k_{m-3}} A_2(j_0,j_1,k_{m-3},\cdots,k_0) W^{j[k_{m-3}(r_4r_5\cdots r_m)+\cdots+k_0]}$$

Proceeding in similar fashion, a set of recursive equations are obtained of the form

$$A_p(j_0,j_1,\cdots,j_{p-1},k_{m-p-1},\cdots,k_0)$$

$$= \sum_{k_{m-p}} A_{p-1}(j_0,j_1,\cdots,j_{p-2},k_{m-p},\cdots,k_0)$$

$$W^{[j_{p-1}(r_1r_2\cdots r_{p-1})+\cdots+j_0]k_{m-p}(r_{p+1}\cdots r_m)} \qquad (28)$$

$$p = 1,2,\cdots,m$$

Note that the last array calculated gives the Fourier sums as

$$X(j_{m-1},\cdots,j_0) = A_m(j_0,\cdots,j_{m-1}) \qquad (29)$$

Note the similarity of the form of these equations to the form of Equations 12 and 13. The fact that all of the variables in Equations 12 and 13 take on only the values of 0 and 1, however, allows further simplification as will be shown in Chapter IV.

C. Computational Savings

If Equation 11 were evaluated as written for N values of j, it can be seen that a total of N^2 operations would be required where an operation is defined as a complex multiplication followed by a complex addition. When Equation 11 is evaluated using the recursive Equations 28, only

$N(r_1+r_2+\cdots+r_m)$ operations are required. Thus use of the Cooley-Tukey algorithm results in decreasing the number of operations by a multiplicative factor of at least $(r_1 r_2 \cdots r_m)/(r_1+r_2+\cdots+r_m)$.

If a time series of 1024 values were to be analyzed, the direct evaluation of Equation 11 would require that $N^2 = (1,024)^2 = 1,048,576$ operations be performed for each set of N spectral estimates. Using the Cooley-Tukey algorithm, this would be reduced by a factor of at least 51 resulting in 20,480 operations (using $N = 2^{10}$). The actual saving, however, is even greater in this example because one-half of the 20,480 operations involve a multiplier of W^0; therefore, the computation consists of only 10,240 operations.

Cooley and Tukey (5) showed that their algorithm has the greatest advantage over direct evaluation when $r_1 = r_2 = r_3 = \cdots = r_m = 3$. Little efficiency is lost, however, when the values are chosen to be either all twos or all fours. When the values of r are made considerably higher than four, the maximum efficiency afforded through the use of this algorithm is not attained, but the computational saving is still substantial when compared with evaluating Equation 11 directly.

As an example, consider the example of N = 2,500 expressed as the product of 50 times 50. The reduction factor in this case is still 25. It should be noted, however, that if the values of r differ considerably, the efficiency of the algorithm drops accordingly. If N is expressed as the product

of 250 and 10, the reduction factor drops to 9.6. Thus, given a value of N, it is usually best to express N as the product of as many integers as possible while keeping their values near the same order of magnitude.

In the following chapters, two special purpose digital machine organizations will be discussed which make use of both the computational savings and the parallel computation capabilities afforded by the Cooley-Tukey algorithm. In these machines, the choice of the values of r depends upon the previously discussed considerations as well as the unique requirements of each analyzer.

IV. MACHINE ORGANIZATION WHERE N IS A POWER OF 2

The large savings in computation and the possibilities of parallel computation afforded by the Cooley-Tukey algorithm suggest that spectral estimates of a sampled time function can be computed while the function is being sampled. If the spectral estimates, based on data from a record T seconds long, could be computed in T seconds, the operation could go on continuously, always giving spectral estimates from the last complete record input. This operation is rather loosely called real-time spectral analysis.

In this chapter, a computer configuration is discussed which could be used to find the spectrum of a wide variety of signals in real-time. This machine, which is based on N being a power of 2, simultaneously inputs the time series from one record while outputting the spectral values from the previous record.

A. Cooley-Tukey Equations for Binary Case

When the recursive Equations 28 are specialized to the case of $r_1 = r_2 = \cdots = r_m = 2$, they result in Equation 12. Since, however, k_{m-p} is allowed to take on only the values 0 and 1, they may also be written in the form

$$A\,(j_0,\cdots,j_{p-1},k_{m-p-1},\cdots,k_0) \tag{30}$$

$$= A_{p-1}(j_0,\cdots,j_{p-2},0,k_{m-p-1},\cdots,k_0)$$

$$+ A_{p-1}(j_0,\cdots,j_{p-2},1,k_{m-p-1},\cdots,k_0)W^{(j_{p-1}2^{p-1}+\cdots+j_0)2^{m-p}}$$

$$p = 1,2,\cdots,m \quad ; \quad N = 2^m$$

Note that the N elements of the original time series are represented by the A_0 terms with arguments which range from 0 through N-1. From the list of A_0 terms a list of A_1 terms is generated as specified by Equation 30. Then the A_2 terms are formed from the A_1 terms, the A_3 terms are formed from the A_2 terms and so on until the A_m terms are calculated. The A_m terms are actually the complex spectral estimates but they must be reordered as specified by Equations 31. Then the D.C. term will appear first, the first harmonic second, the second harmonic third and so on.

$$X(j_{m-1},\cdots,j_0) = A_m(j_0,\cdots,j_{m-1}) \tag{31}$$

For the example of N = 8, Equations 30 and 31 can be written in the form:

$$A_1(j_0,k_1,k_0) = A_0(0,k_1,k_0) + A_0(1,k_1,k_0)W^{j_0 2^2}$$

$$A_2(j_0,j_1,k_0) = A_1(j_0,0,k_0) + A_1(j_0,1,k_0)W^{(j_1 2+j_0)2} \qquad (32)$$

$$A_3(j_0,j_1,j_2) = A_2(j_0,j_1,0) + A_2(j_0,j_1,1)W^{(j_2 2^2+j_1 2+j_0)}$$

$$X(j_2,j_1,j_0) = A_3(j_0,j_1,j_2) \qquad (33)$$

where

$$j = j_2 4 + j_1 2 + j_0 \qquad j_0 = j_1 = j_2 = 0,1$$

$$k = k_2 4 + k_1 2 + k_0 \qquad k_0 = k_1 = k_2 = 0,1$$

Figure 1 represents these operations diagramatically for the example of N = 8. Decimal representation of the numbers in the arguments has been used instead of the binary representation which appears in the formulas in the hope of demonstrating the operations more clearly. The lines and arrows identify the two terms from the previous set of A_i values, which are combined to form a given A_{i+1} term. The right-most term of the two will be multiplied by W raised to the appropriate power and then added to the other term thereby satisfying Equations 32. The reordering of the A_3 terms is also shown in accordance with Equations 33.

In Figures 2 and 3, all of the equations of 32 and 33 are written out for N = 8. By viewing these equations in

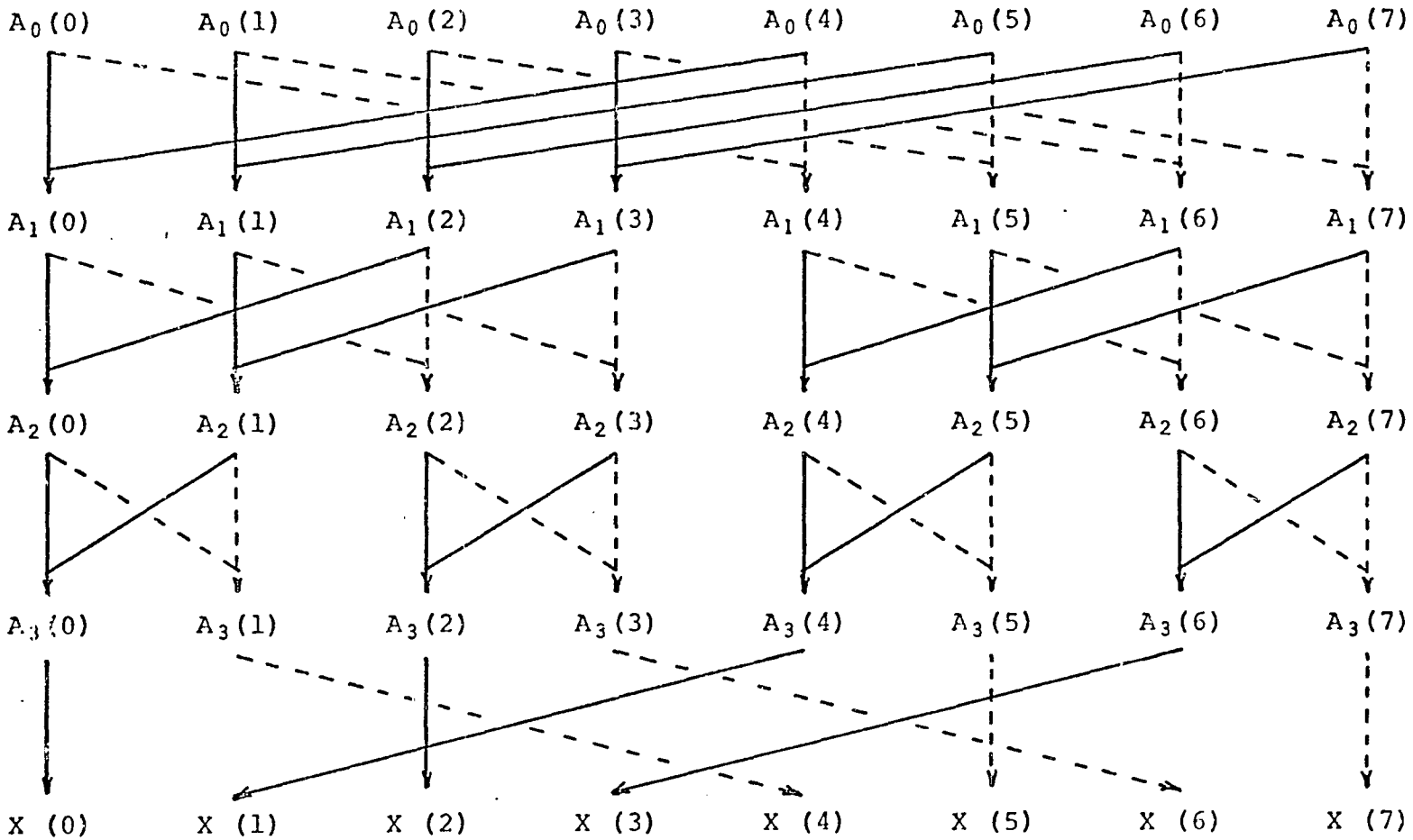


Figure 1. Equations 32 and 33 represented diagramatically

conjunction with Figure 1, the requirements on a digital machine to implement this algorithm become more apparent.

It should be noted that when the original time series is real valued, only the first N/2 values of the complex spectrum are independent and therefore need to be calculated. These N/2 values are the spectral estimates ranging between the D.C. term and one-half the sampling frequency. The last N/2 values of X represent the spectral estimates between minus one-half the sampling frequency and the D.C. term. In cases where both sides of the spectrum are desired, all N of the X terms, shown in Figure 3, should be computed.

B. The $N = 2^3$ Binary Analyzer

The two main problems associated with implementing the algorithm with a special purpose machine are concerned with the routing of the data and the generation of the W terms. Also, a configuration is desired which requires a minimal set of storage elements and a minimum number of stored W values.

A configuration which satisfies both of these conditions quite well is shown schematically in Figure 4 for the example of N = 8. This value of N is too small to be practical but it is large enough to show the principles which carry over to larger analyzers.

1. Routing the data

The function of the first stage of the analyzer, shown in

$$A_1(0) = A(0) + A(4)W^0$$

$$A_1(1) = A(1) + A(5)W^0$$

$$A_1(2) = A(2) + A(6)W^0$$

$$A_1(3) = A(3) + A(7)W^0$$

$$A_1(4) = A(0) + A(4)W^4$$

$$A_1(5) = A(1) + A(5)W^4$$

$$A_1(6) = A(2) + A(6)W^4$$

$$A_1(7) = A(3) + A(7)W^4$$

$$A_2(0) = A_1(0) + A_1(2)W^0$$

$$A_2(1) = A_1(1) + A_1(3)W^0$$

$$A_2(2) = A_1(0) + A_1(2)W^4$$

$$A_2(3) = A_1(1) + A_1(3)W^4$$

$$A_2(4) = A_1(4) + A_1(6)W^2$$

$$A_2(5) = A_1(5) + A_1(7)W^2$$

$$A_2(6) = A_1(4) + A_1(6)W^6$$

$$A_2(7) = A_1(5) + A_1(7)W^6$$

Figure 2. Equations 32 written out with the arguments expressed in decimal form

$$A_3(0) = A_2(0) + A_2(1)W^0$$

$$A_3(1) = A_2(0) + A_2(1)W^4$$

$$A_3(2) = A_2(2) + A_2(3)W^2$$

$$A_3(3) = A_2(2) + A_2(3)W^6$$

$$A_3(4) = A_2(4) + A_2(5)W^1$$

$$A_3(5) = A_2(4) + A_2(5)W^5$$

$$A_3(6) = A_2(6) + A_2(7)W^3$$

$$A_3(7) = A_2(6) + A_2(7)W^7$$

$$X(0) = A_3(0)$$

$$X(1) = A_3(4)$$

$$X(2) = A_3(2)$$

$$X(3) = A_3(6)$$

$$X(4) = A_3(1)$$

$$X(5) = A_3(5)$$

$$X(6) = A_3(3)$$

$$X(7) = A_3(7)$$

Figure 3. Equations 32 and 33 written out with the arguments expressed in decimal form

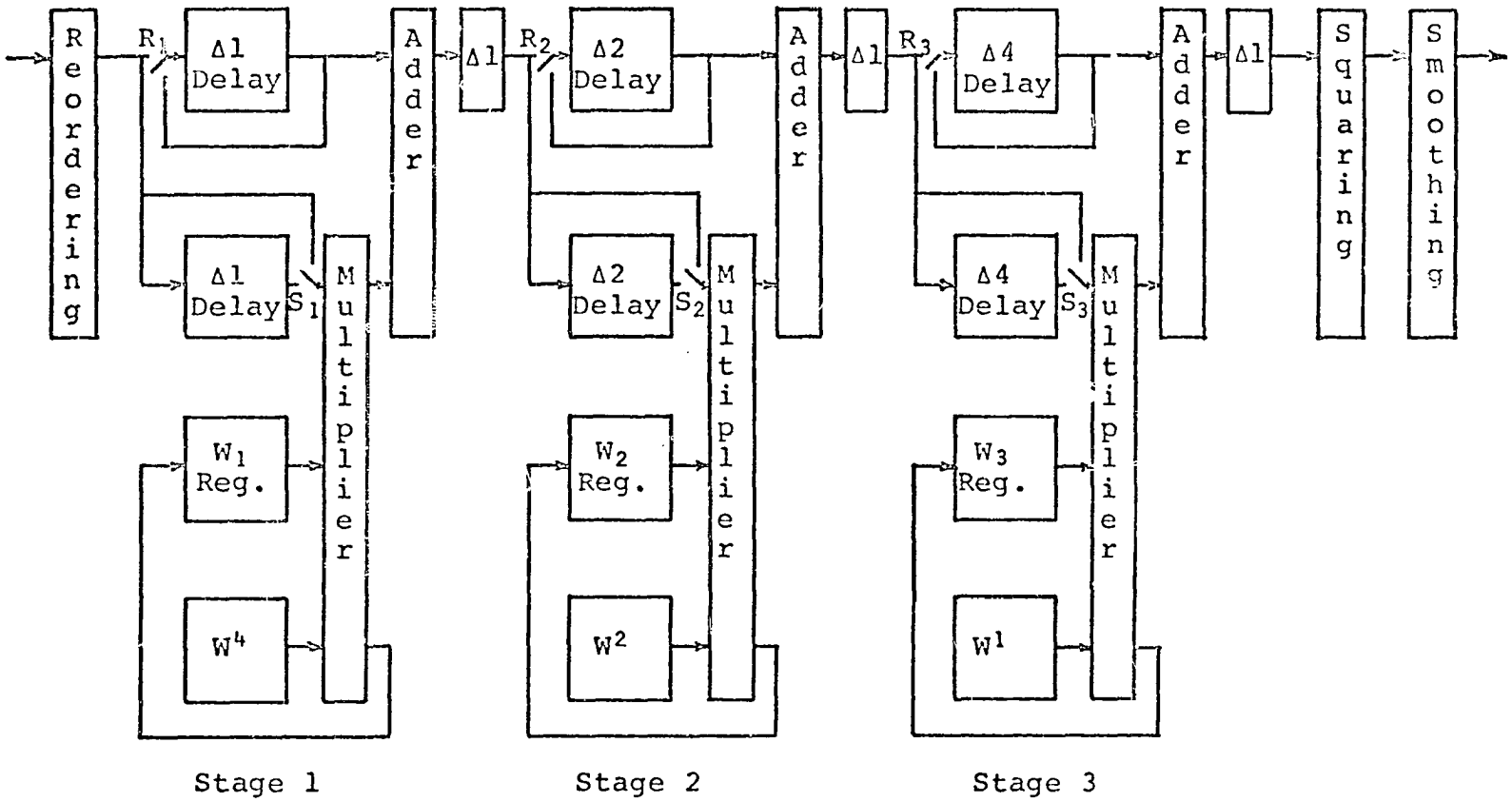


Figure 4. The $N = 2^3$ binary analyzer

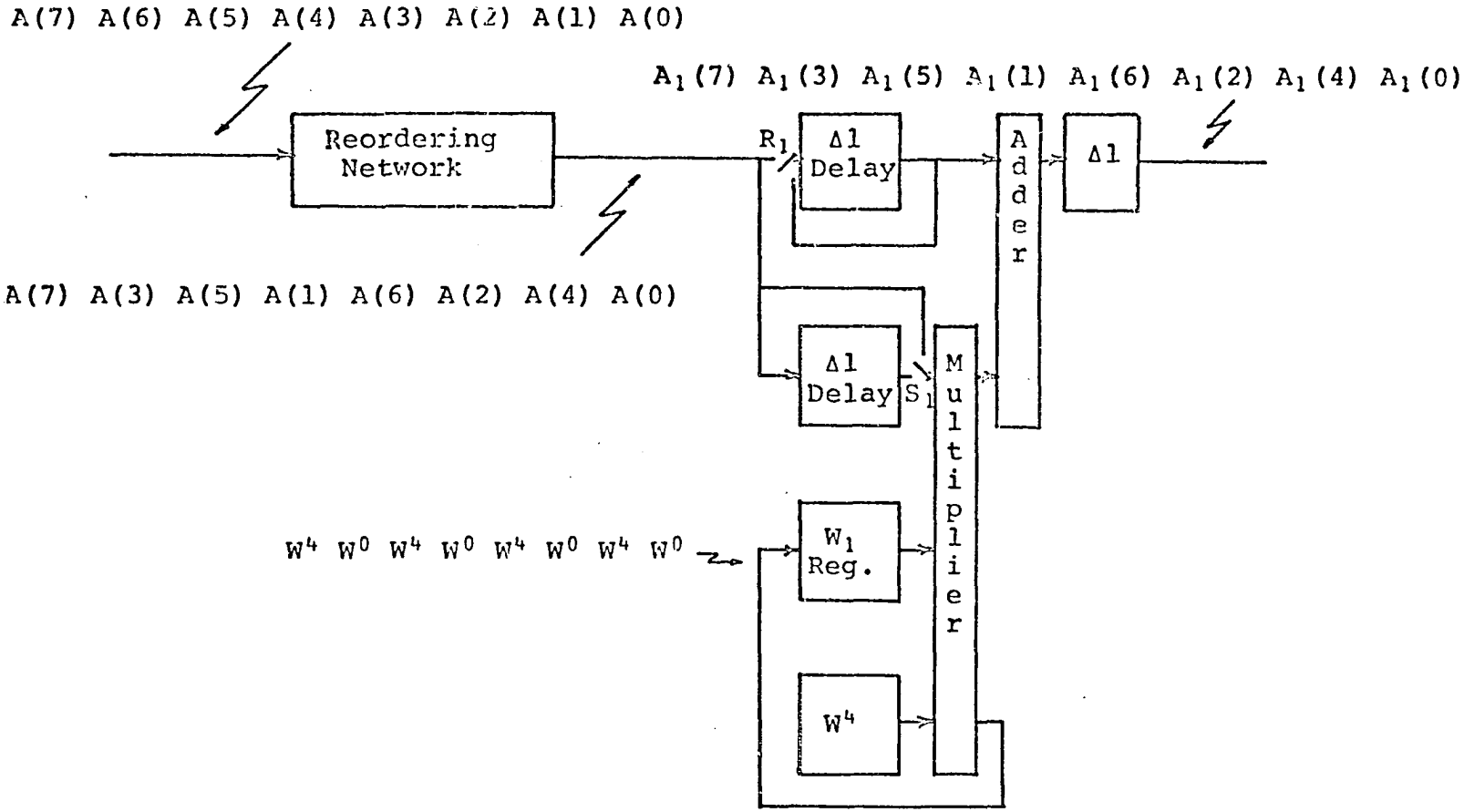


Figure 5. The first stage of the $N = 2^3$ binary analyzer

Figures 4 and 5, is to form the A_1 terms from the A_0 terms as specified by the equations in Figure 1. The second stage then forms the A_2 terms and the third stage the A_3 terms. Before entering the analyzer, the original time series is reordered by the reordering network which makes the spectral values appear in the correct order at the output and results in a considerable simplification of the generation of the W values for each stage.

The time series enters the reordering network in the order A(0), A(1), A(2), A(3), A(4), A(5), A(6), A(7) as shown in Figure 5. After the reordering, the values enter the first stage in the order A(0), A(4), A(2), A(6), A(1), A(5), A(3), A(7).

As a result of this reordering, the A(0) term is delayed by three sample periods before it enters the first stage of the analyzer. This delay is necessary so that the A(4) term, which was the fifth term of the original series, can be made the second term of the reordered series. Two possible methods of performing this reordering are discussed in Section IV.B.3.

Note that each A value is required in the evaluation of two different summations. This suggests that it must be used and then stored for later use. In the diagram of Figure 5, Switch R_1 is assumed to be in the up position when A(0) appears at the input. Just before the end of this sample period, a command pulse is sent to the shift registers, causing them to store the numbers appearing at their inputs. At

the end of this sample period, the value A(0) is therefore stored in the upper and lower shift registers.

At the beginning of the next sample period, A(4) appears at the input of the first stage. Switch S_1 is in the up position and Switch R_1 is in the down position. If we tentatively assume that W^0 can also be made available to the arithmetic unit at this time, we see that the three quantities required to form the $A_1(0)$ term are available to be operated upon. (See the equations in Figure 2). Thus, A(4) is multiplied by W^0 and the resulting product is added to A(0), forming $A_1(0)$. At the end of this sample period, a command pulse is also sent which again stores A(0) in the top shift register but stores A(4) in the bottom shift register.

At the beginning of the next sample period, Switch R_1 is returned to the up position and S_1 to the down position. This means that A(0) and A(4) are again available to be used when W^4 appears, to form the $A_1(4)$ term. At the end of this sample period, the A(2) input is read into the top shift register, the switches are changed and A(6) appears at the input.

If this procedure is carried on, it is apparent that the A_1 values will be computed in the following order: $A_1(0)$, $A_1(4)$, $A_1(2)$, $A_1(6)$, $A_1(1)$, $A_1(5)$, $A_1(3)$, $A_1(7)$.

In the second stage, the A_2 values are to be computed in the following order: $A_2(0)$, $A_2(4)$, $A_2(2)$, $A_2(6)$, $A_2(1)$, $A_2(5)$, $A_2(3)$, $A_2(7)$. As seen from the equations in Figure 2, this implies that first $A_1(0)$ and $A_1(2)$ must be presented to

the arithmetic unit, then $A_1(4)$ and $A_1(6)$, and then $A_1(0)$ and $A_1(2)$ again and so on.

In the binary analyzer shown in Figure 4 this data routing function in the second stage is also accomplished through the use of two shift registers. First the $A_1(0)$ term is stored in the upper register and then the $A_1(4)$ term. When $A_1(2)$ appears at the input, Switches R_2 and S_2 are placed in their vertical positions which simultaneously presents $A_1(0)$ and $A_1(4)$ to the arithmetic unit. Assuming that the correct value of W is supplied, the $A_2(0)$ term can then be calculated. At the end of this sample period, the $A_1(0)$ term is again stored in the first position of the upper shift register, and the $A_1(4)$ term is stored in the first position of the lower shift register.

During the next sample period, both the $A_1(4)$ and the $A_1(6)$ terms are available to the arithmetic unit and thus the $A_2(4)$ term can be calculated. If the shift registers again input and shift to the right, during the next sample period the $A_1(0)$ and $A_1(2)$ terms appear at their outputs. As these are the next terms needed, the R_2 and S_2 switches are placed in their horizontal positions and the $A_2(2)$ term is calculated.

This sequence of events continues with the switches changing states only half as often as in the previous stage, and with the A_2 values being generated in the desired order.

The following stages operate in a similar manner, except that the terms involved in each sum are separated further in

the list of terms input. This means the shift registers in a given stage must delay the numbers twice as long as in the preceding stage. In the third stage the delay is four sample periods. The period of time which the R and S switches remain in a given state is equal to the length of the time a number is delayed in going through a shift register in that stage.

If it is assumed that the multiplication and addition operations take a substantial part of the sample period, it is apparent that a buffer is required between stages which will mean that the $A_1(0)$ term appears at the input of stage two the sample period after it was calculated in stage one. This is shown schematically in Figure 4 by an additional shift register following each stage.

In practice the multiplier and adder and perhaps the buffer functions would all be performed by the arithmetic unit of each stage, but they have been shown as separate entities in the diagrams for clarity.

2. Computing the values of W

The appropriate values of W for each stage are generated through the use of De Moivre's formula which states (9)

$$(\cos \theta + i \sin \theta)^n = \cos n\theta + i \sin n\theta \qquad (34)$$

If one value of W is "wired into" each stage of the analyzer, all succeeding values required can be computed by performing one multiplication per value.

As an example, consider the W values required in computing the A_3 terms as shown in the equations of Figure 3. If the initially required value W^0 is multiplied by the stored W^1 value, W^1 will be formed in the W_3 register of Figure 4. If this W_3 register is again multiplied by W^1, W^2 will be formed and so on. Thus it is quite convenient to generate the sequence W^0, W^1, W^2, W^3, W^4, W^5, W^6, W^7, W^0, W^1, W^2, and so on (note that $W^0 = W^8$). As shown by the equations in Figure 3, this is the order in which they are needed to form the A_3 terms in the order $A_3(0)$, $A_3(4)$, $A_3(2)$, $A_3(6)$, $A_3(1)$, $A_3(5)$, $A_3(3)$, $A_3(7)$. Note that this corresponds to outputting the X values in the order X(0), X(1), X(2), X(3), X(4), X(5), X(6), X(7).

To form the values of W required by the second stage, W^2 is wired in and the sequence W^0, W^2, W^4, W^6, W^0, etc. can be conveniently generated. The sequence generated for stage one is W^0, W^4, W^0, W^4, W^0, etc.

In actual practice, the first two stages would not require the use of a conventional arithmetic unit as the multiplications by the W^0, W^2, W^4, and W^6 terms correspond to multiplications by 1, +i, -1, and -i, respectively.

3. The reordering network

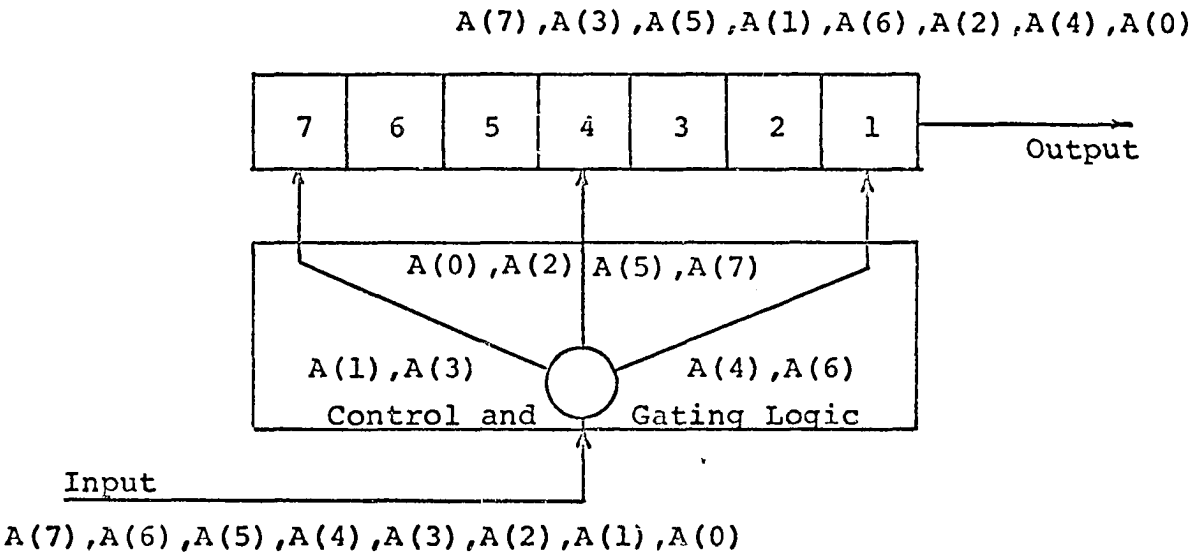
Two possible configurations for the initial reordering of the data are suggested in this section. The first configuration results in a shorter fixed delay between the time the

first sample goes in and the time the first sample comes out, but isn't as easy to generalize for different values of N.

The first reordering network is shown in Figure 6. It can be thought of as a shift register which is loaded from a number of different input lines. Each of the A values is loaded into the shift register as soon as it becomes available but, for the example of N = 8, A(1) and A(3) are loaded into location 7, A(0), A(2), A(5) and A(7) are loaded into location 4, and A(4) and A(6) are loaded into location 1. For larger values of N a considerable amount of control logic would be necessary to load each A into its correct location. The register is assumed to shift its contents to the right just before each new value is loaded into it.

The chart of Figure 6 shows the contents of each location of the shift register as a function of time. The last value loaded is circled.

The second reordering configuration is shown in Figure 7 for the example of N = 8. It consists of a two dimensional array of shift registers which will shift data either serially or in parallel. It is required that the array shifts the data in the direction of the solid arrows for one T-second period and then in the direction of the dotted arrows for the next T-second period and so on. The reordering is accomplished by inputting the N samples of each T-second record while the array is shifting along one path, and outputting the N samples while it is shifting along the other path.



Time in ΔT units	7	6	5	4	3	2	1
0	-	-	-	A(0)	-	-	-
1	A(1)	-	-	-	A(0)	-	-
2	-	A(1)	-	A(2)	-	A(0)	-
3	A(3)	-	A(1)	-	A(2)	-	A(0)
4	-	A(3)	-	A(1)	-	A(2)	A(4)
5	-	-	A(3)	A(5)	A(1)	-	A(2)
6	-	-	-	A(3)	A(5)	A(1)	A(6)
7	-	-	-	A(7)	A(3)	A(5)	A(1)
8	-	-	-	-	A(7)	A(3)	A(5)
9	-	-	-	-	-	A(7)	A(3)
10	-	-	-	-	-	-	A(7)

Figure 6. The shift register reordering network for N = 8

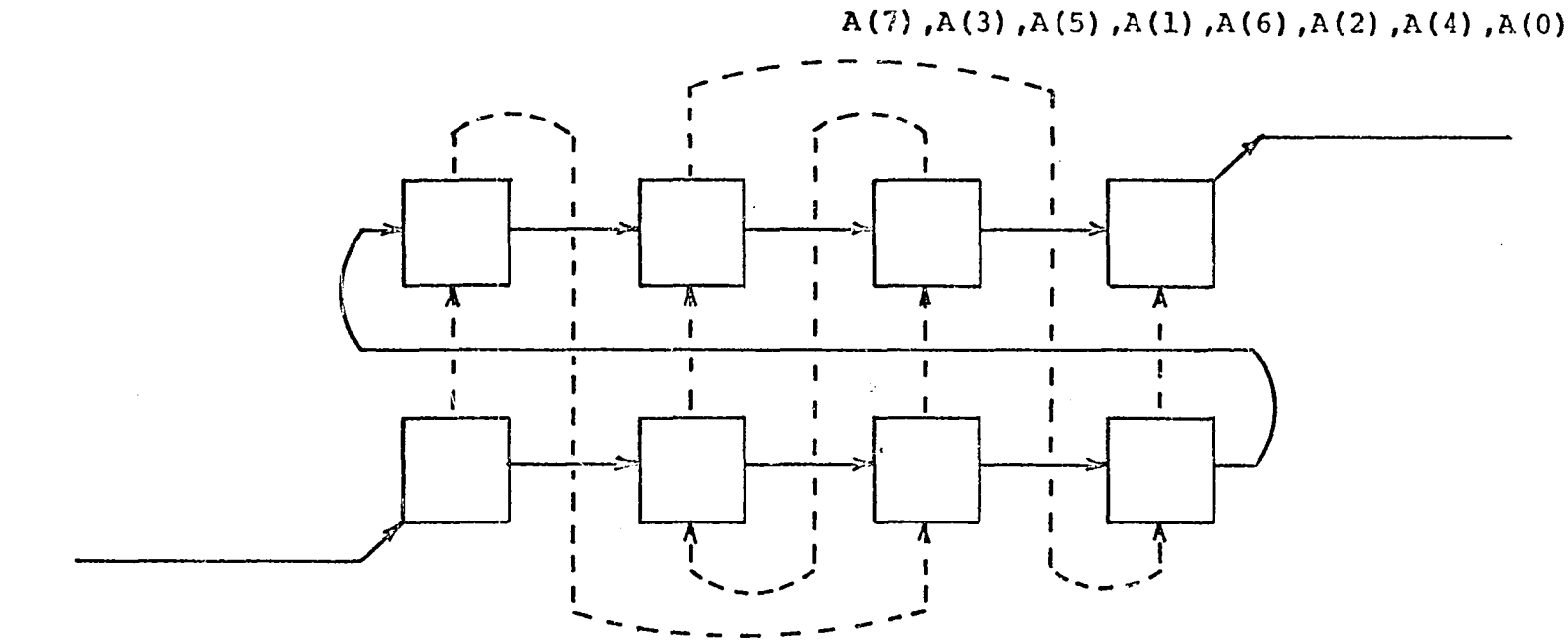


Figure 7. The array reordering network for N = 8

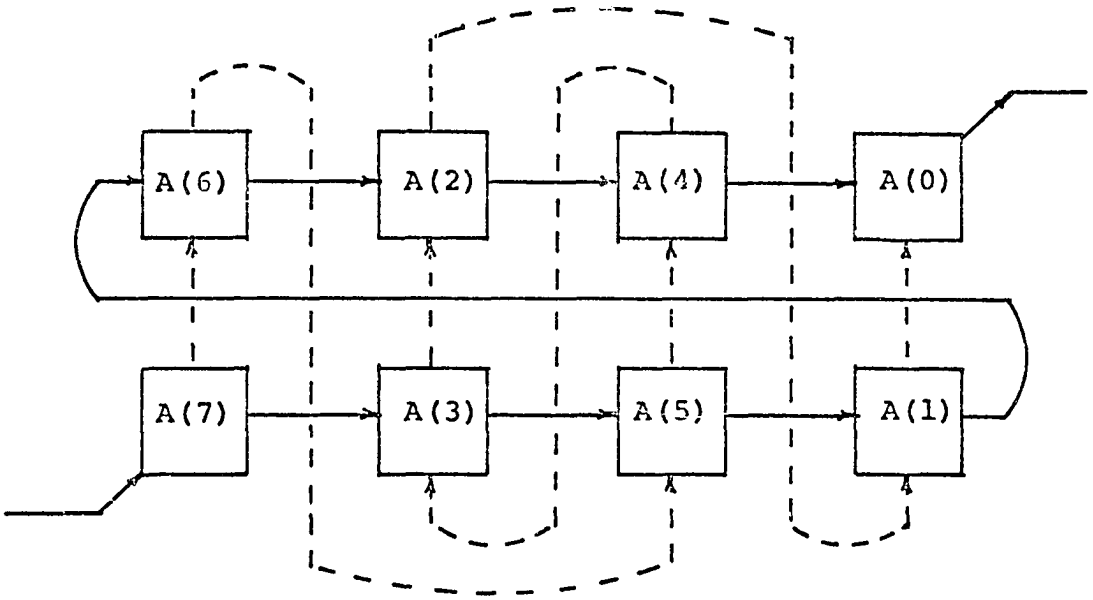
When a record has been loaded in along the dotted lines shown in Figure 7, the A values will appear as shown in Part (a) of Figure 8 just before the shifting path is changed. Note that if these values are now output along the solid line path, the reordering will have been accomplished.

When a record has been loaded in along the solid lines, the A values will appear as shown in Part (b) of Figure 8 just before the shifting path is changed. Note that if these values are now output along the dotted line path, the reordering will also have been accomplished.

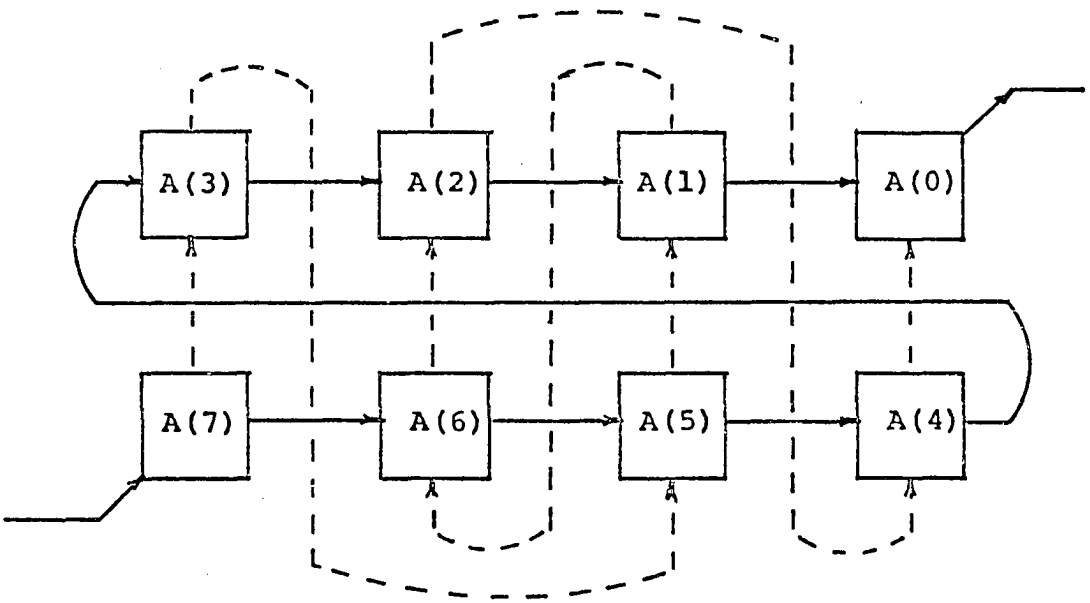
The advantage of being able to perform the reordering by either method is that the array can be outputting one set of data while inputting the next set.

For other values of $N = 2^m$, the reordering is still performed by an N element array. When m is an even number, the array will be square and when m is odd (as in the example of $N = 2^3$) the array will be rectangular.

In each case the first value will be output from the array T seconds after the first value is put in. For the N = 8 analyzer this means that the first spectral estimate from a given record would not be output until $11\Delta T$ seconds after the record was initially put into the reordering network. This could be reduced considerably, however, by performing the analysis and taking the values out of the reordering network at a rate faster than the original sampling rate.



(a)



(b)

Figure 8. Arrangement of the input series in the array reordering network (a) when input along the dotted arrows and (b) when input along the solid arrows

4. The squaring and smoothing networks

The network which forms the $|X(j)|^2$ terms from the complex valued spectral estimates, could be composed of another arithmetic unit or, if the spectrum is to be displayed in analog form, the network could consist of analog squaring circuits and an analog adder.

Data smoothing, using the "Hanning" spectral window (see Equations 10), can be done quite conveniently using the configuration of three registers shown in Figure 9. The unsmoothed spectral estimates are stored in Register A as they are computed, while at the same time the values previously in Registers A and B are moved to Registers B and C respectively. The weighting by 1/4, 1/2 and 1/4 is accomplished by having the decimal point in Registers A and C occur two places to the left of the initial position. In Register B the decimal point is one place to the left of the initial position. Each smoothed spectral estimate is obtained by adding the contents of Registers A, B and C.

Convolution with a spectral window not using the 1/4, 1/2, 1/4 weights, requires a more involved arithmetic unit.

C. The $N = 2^m$, m Stage Binary Analyzer

The machine organization shown for the $N = 2^3$ analyzer is easily extended to the case of $N = 2^m$. The general configuration is shown in Figure 10 and consists of adding more stages to the three original ones until the required value of N

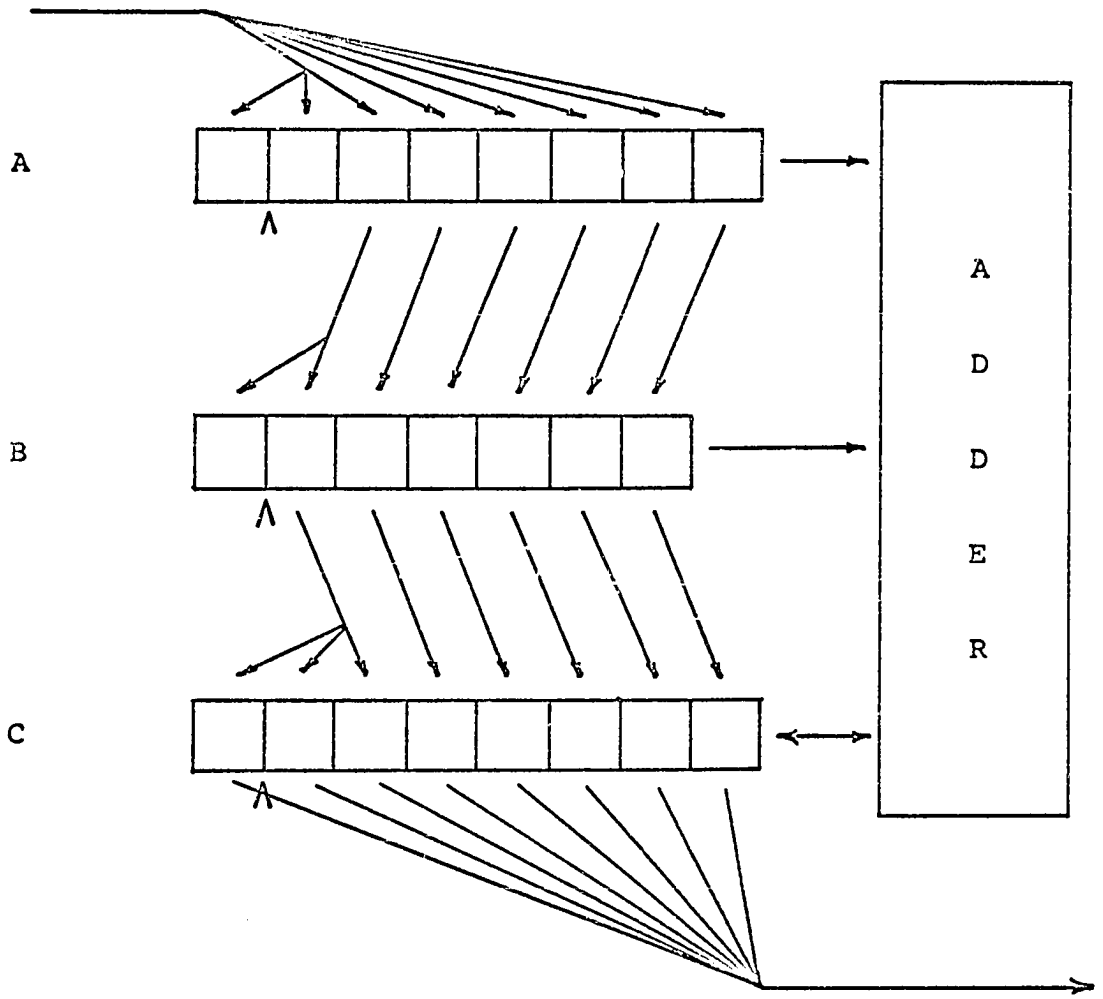


Figure 9. Data smoothing network

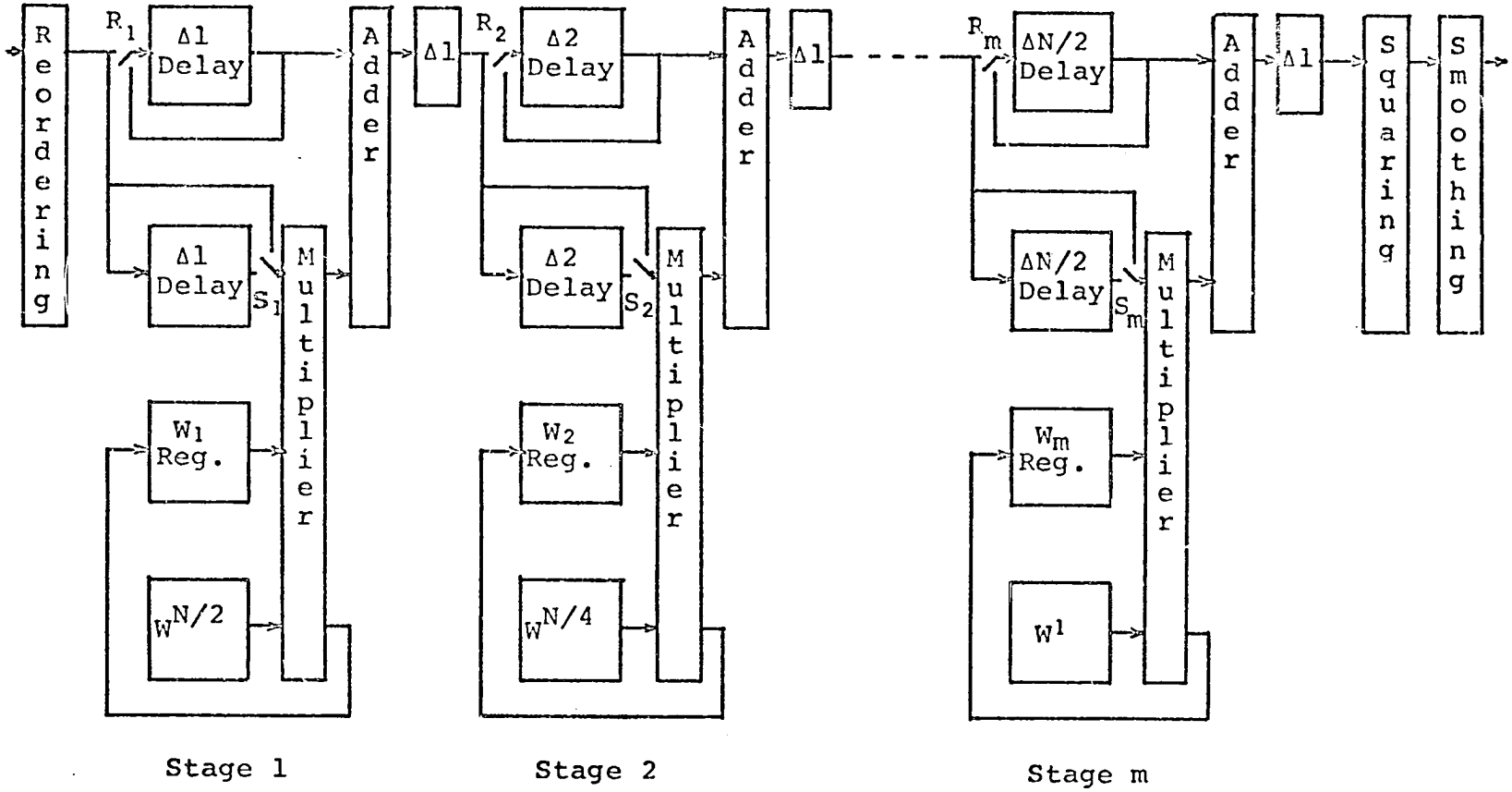


Figure 10. The $N = 2^m$ binary analyzer

is attained.

This add-on feature suggests that once a basic unit is built, extra modules could be added on or taken off to change the value of N. It should be noted that each module contains twice as many bits of storage as the one preceding it, therefore the last module is the most expensive. Thus if the complexity of the last module can be reduced, a considerable saving can be achieved in the cost of the analyzer.

In a specific design where a definite maximum value of N is known, it turns out that the lower shift register can be eliminated in the last stage. This is possible when only N/2 spectral estimates are required and results from the fact that in this case the A_{m-1} values being input are only used one time. No stages can be added on after a module built in this way, however, so this module places an upper limit on the number of modules which can be used in a given analyzer.

The only part of the analyzer which must be seriously altered, when the value of N is changed, is the reordering network. A different reordering network has to be available for each value of N.

The comments previously made about the squaring and smoothing networks are valid for any value of N.

D. Components Required

An idea of the cost involved in building a modular analyzer can be obtained by estimating the number of

components required as a function of N. These estimates are tabulated in Part (a) of Figure 11 for an analyzer using the array reordering network.

Figure 11a. Components in binary analyzer assuming 10 bit numbers

Figure 11b. Components in $r_1 r_2$ analyzer assuming 10 bit numbers

N	m	Reordering network		Analyzer	
		Flip-flops	Logic gates & invertors	Bits delay	Arithmetic units
8	3	80	320	240	1
16	4	160	640	460	2
32	5	320	1,280	950	3
64	6	640	2,560	1,920	4
128	7	1,280	5,120	3,920	5
256	8	2,560	10,240	7,780	6
512	9	5,120	20,480	15,480	7
1,024	10	10,240	40,960	30,860	8
2,048	11	20,480	81,920	61,600	9

(a)

N	r_1	r_2	Flip-flops	Logic gates & invertors	Arithmetic units
8	4	2	160	640	1
25	5	5	500	2,000	1
50	10	5	1,000	4,000	1
64	8	8	1,280	5,120	1
100	10	10	2,000	8,000	1
225	15	15	4,500	18,000	1
400	20	20	8,000	32,000	1
625	25	25	12,500	50,000	1
900	30	30	18,000	72,000	1
1,600	40	40	32,000	128,000	1
2,500	50	50	50,000	200,000	1

(b)

V. MACHINE ORGANIZATION WHERE N IS A PRODUCT OF TWO INTEGERS

The machine discussed in this chapter is applicable to the case where N may be expressed as the product of two integers. It is best suited for analyzing pulses or echo returns as it will not accept data from the next record while it is outputting the spectrum of the last record. That is, there must be a "dead time" between input records. This problem can be minimized or eliminated, however, as will be discussed in Section V.B.4.

A. The Cooley-Tukey Equations for $N = r_1 r_2$

When Equations 28 and 29 are specialized to the case of $m = 2$, they can be written in the form

$$\begin{aligned} A_1(j_0,k_0) &= \sum_{k_1} A(k_1,k_0) W^{j_0 k_1 r_2} \\ A_2(j_0,j_1) &= \sum_{k_0} A_1(j_0,k_0) W^{(j_1 r_1 + j_0) k_0} \end{aligned} \tag{35}$$

$$X(j_1,j_0) = A_2(j_0,j_1) \tag{36}$$

where

$$j = j_1 r_1 + j_0 \qquad j_0 = 0,1,\cdots,r_1-1 \qquad j_1 = 0,1,\cdots,r_2-1$$

$$k = k_1 r_2 + k_0 \qquad k_0 = 0,1,\cdots,r_2-1 \qquad k_1 = 0,1,\cdots,r_1-1$$

If r_1 is chosen to be 4 and r_2 to be 2, we can again consider the example of N = 8. For this example, we write Equations 35 and 36 as

$$A_1(j_0,k_0) = \sum_{k_1=0}^{3} A(k_1,k_0)W^{j_0k_12} \tag{37}$$

$$A_2(j_0,j_1) = \sum_{k_0=0}^{1} A_1(j_0,k_0)W^{(j_14+j_0)k_0}$$

$$X(j_1,j_0) = A_2(j_0,j_1) \tag{38}$$

where

$$j = j_14 + j_0 \qquad j_0 = 0,1,2,3 \qquad j_1 = 0,1$$
$$k = k_12 + k_0 \qquad k_0 = 0,1 \qquad k_1 = 0,1,2,3$$

Note that j and k are no longer represented in terms of a number system with a constant radix.

Conversions from the $A(k_1,k_0)$ representation to the A(k) decimal representation and from the $X(j_1,j_0)$ representation to the X(j) representation are given in the table in Figure 12. All of the terms of Equations 37 and 38 are given with decimal arguments in Figure 13.

(k_1,k_0)	(k)	(k_1,k_0)	(k)	(k_1,k_0)	(k)	(k_1,k_0)	(k)
0 0	0	1 0	2	2 0	4	3 0	6
0 1	1	1 1	3	2 1	5	3 1	7

(a)

(j_1,j_0)	(j)	(j_1,j_0)	(j)
0 0	0	1 0	4
0 1	1	1 1	5
0 2	2	1 2	6
0 3	3	1 3	7

(b)

Figure 12. Conversion from the (a) $A(k_1,k_0)$ representation to the A(k) representation and from the (b) $X(j_1,j_0)$ representation to the X(j) representation where

$$j = j_1 r_1 + j_0 = j_1 4 + j_0$$
$$k = k_1 r_2 + k_0 = k_1 2 + k_0$$

$$A_1(0) = A(0)W^0 + A(2)W^0 + A(4)W^0 + A(6)W^0$$

$$A_1(1) = A(1)W^0 + A(3)W^0 + A(5)W^0 + A(7)W^0$$

$$A_1(2) = A(0)W^0 + A(2)W^2 + A(4)W^4 + A(6)W^6$$

$$A_1(3) = A(1)W^0 + A(3)W^2 + A(5)W^4 + A(7)W^6$$

$$A_1(4) = A(0)W^0 + A(2)W^4 + A(4)W^0 + A(6)W^4$$

$$A_1(5) = A(1)W^0 + A(3)W^4 + A(5)W^0 + A(7)W^4$$

$$A_1(6) = A(0)W^0 + A(2)W^6 + A(4)W^4 + A(6)W^2$$

$$A_1(7) = A(1)W^0 + A(3)W^6 + A(5)W^4 + A(7)W^2$$

$$A_2(0) = A_1(0)W^0 + A_1(1)W^0 = X(0)$$

$$A_2(1) = A_1(0)W^0 + A_1(1)W^4 = X(4)$$

$$A_2(2) = A_1(2)W^0 + A_1(3)W^1 = X(1)$$

$$A_2(3) = A_1(2)W^0 + A_1(3)W^5 = X(5)$$

$$A_2(4) = A_1(4)W^0 + A_1(5)W^2 = X(2)$$

$$A_2(5) = A_1(4)W^0 + A_1(5)W^6 = X(6)$$

$$A_2(6) = A_1(6)W^0 + A_1(7)W^3 = X(3)$$

$$A_2(7) = A_1(6)W^0 + A_1(7)W^7 = X(7)$$

Figure 13. Equations 37 and 38 written out with the arguments expressed in decimal form

B. The $N = r_1 r_2$ Analyzer for N = 8

As before, the primary considerations in specifying a machine organization are the routing of the data and the generation of the values of W. A configuration which does quite well in both respects is shown in Figure 14 for the example of N = 8.

1. Routing the data

In this machine the initial time series enters the arithmetic unit in the order, A(0), A(1), A(2), A(3), A(4), A(5), A(6), A(7). During the first sample period, i.e. while A(0) is the most recently obtained sample, the following sequence of events takes place.

1) A(0) is multiplied by W^0 and placed in the number 7 position of the shift register.

2) The contents of the shift register are shifted one position in the direction of the solid arrows.

3) A(0) is multiplied by W^0 and stored in 7.

4) The contents are shifted again.

5) A(0) is multiplied by W^0 again and stored in 7.

6) The contents are shifted again.

7) A(0) is multiplied by W^0 again and stored in 7.

Note that at this point we have evaluated the first term in the summations for $A_1(0)$, $A_1(2)$, $A_1(4)$ and $A_1(6)$ and have stored these partial sums in the shift register. Since A(0) appears in the equations only these four times, it does not have to be saved.

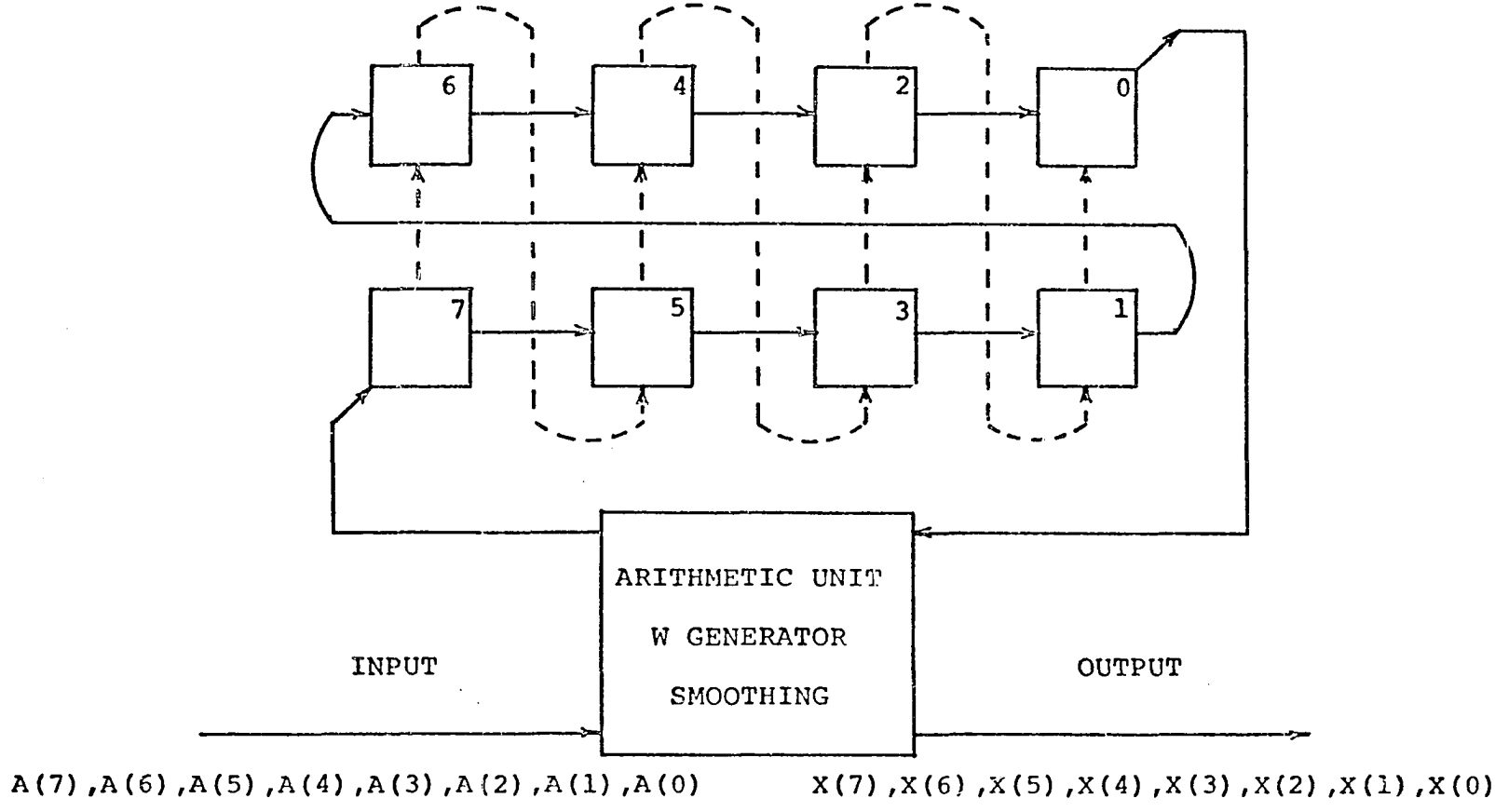


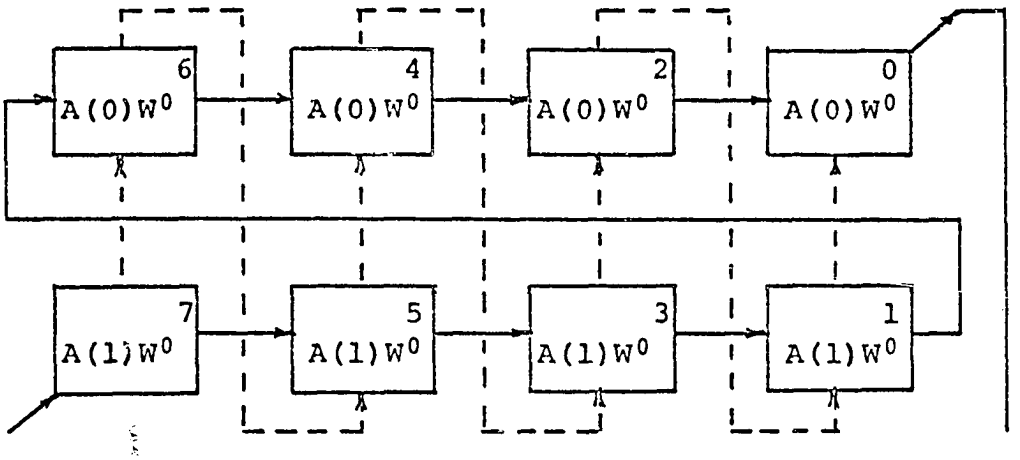
Figure 14. The N = 4·2, $r_1 r_2$ analyzer

During the second sample period, the A(1) term will be multiplied by W^0 four times and the results will be stored as the first terms of the summations which will form $A_1(1)$, $A_1(3)$, $A_1(5)$, and $A_1(7)$. The contents of each location in the analyzer at this point are shown in Part (a) of Figure 15.

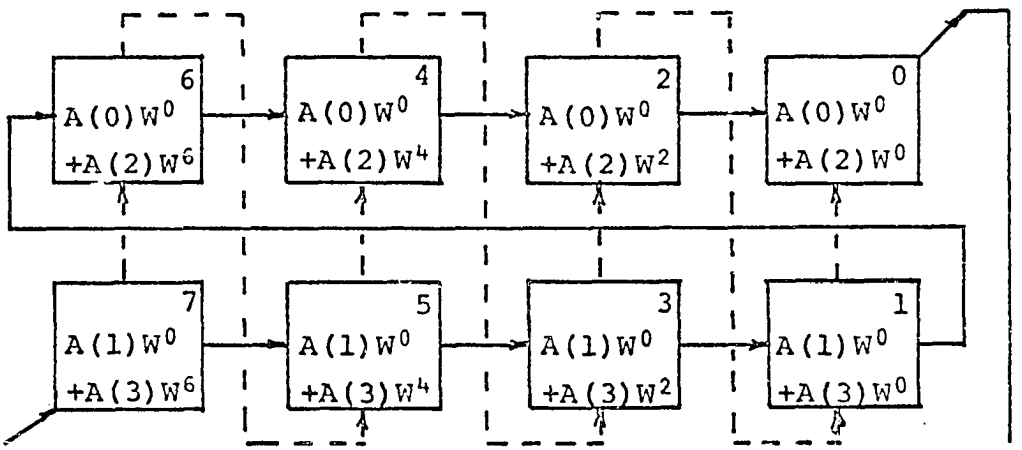
During the third sample period, A(2) appears at the input and is also multiplied by W^0. This product is added to the $A(0)W^0$ term which appears at the output of the array. Then the contents of the array are shifted one position and the sum is stored in location 7 of the array. Next the A(2) term is multiplied by W^2 and added to the next $A(0)W^0$ term output from the array. The array is again shifted and this new sum is stored in location 7. This sequence is repeated two more times while A(2) is present at the input and four more times when A(3) appears at the input.

The contents of each location in the array at this time are shown in Part (b) of Figure 15. Note that now the first two terms of each of the summations required to find the A_1 values have been evaluated and saved.

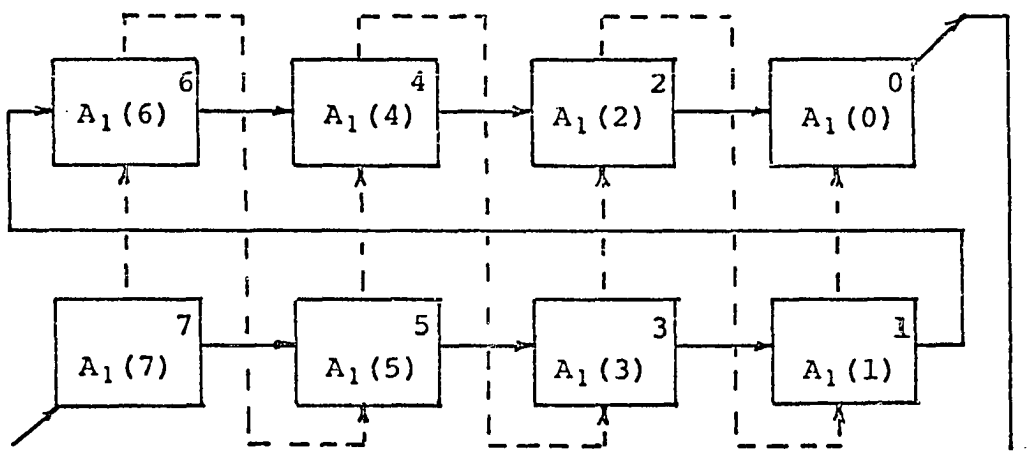
If the same pattern of shifts, multiplications and stores is continued until the A(7) term has been used, the $A_1(0)$ summation will end up in location 0, the $A_1(1)$ summation will end up in location 1, and so on as shown in Part (c) of Figure 15. Thus the A_1 values have been found by simply multiplying the generated values of W by the input value, adding the result to the output of the array, and storing the



(a)



(b)



(c)

Figure 15. Contents of the $r_1 r_2$ analyzer while operating

results back in the input of the array.

From the equations shown in Figure 13, we see that the first A_2 term required is the $A_2(0)$ term (which corresponds to X(0)). To form this term, the stored A_1 values are shifted along the dotted arrows instead of the solid arrows. This way the $A_1(0)$ term can be shifted into the arithmetic unit and multiplied by W^0, forming the first term of the $A_2(0)$ summation. Then $A_1(1)$ can also be shifted in and multiplied by W^0 and added into the $A_2(0)$ partial sum. In this example, this completes the $A_2(0)$ summation and $A_2(0)$ can be output. As both the $A_1(0)$ and $A_1(1)$ terms may be needed later, they are stored back into the input of the shift register.

The $A_2(2)$ term is computed next as it corresponds to the X(1) spectral value. As the shifting is now being done along the dotted lines, the $A_1(2)$ and $A_1(3)$ terms required to compute this $A_2(2)$ term are ready to be output from location 0 of the shift register in the correct order. Assuming that the W values are again supplied as required, the X(1) summation can be performed and the value output. As the $A_1(2)$ and $A_1(3)$ terms are required again, they are stored back into the shift register.

This same sequence of operations can be continued until all 8 of the X terms have been found, or can be stopped after the first $N/2 = 4$ terms have been found.

2. Computing the values of W

The appropriate values of W are still generated through use of De Moivre's formula (Equation 34), but more control logic is required.

Note that in forming the A_1 values, the values of W are required in the following order.

First terms of A_1 sums	W^0	W^0	W^0	W^0	W^0	W^0	W^0	W^0
Second terms of A_1 sums	W^0	W^2	W^4	W^6	W^0	W^2	W^4	W^6
Third terms of A_1 sums	W^0	W^4	W^0	W^4	W^0	W^4	W^0	W^4
Fourth terms of A_1 sums	W^0	W^6	W^4	W^2	W^0	W^6	W^4	W^2

This sequence is obtained by recalling the order in which the terms in the equations of Figure 13 are computed.

In forming the A_2 values, the values of W are required in a different order.

First A_2 sum	W^0	W^0
Second A_2 sum	W^0	W^1
Third A_2 sum	W^0	W^2
Fourth A_2 sum	W^0	W^3
Fifth A_2 sum	W^0	W^4
Sixth A_2 sum	W^0	W^5
Seventh A_2 sum	W^0	W^6
Eighth A_2 sum	W^0	W^7

A configuration which will generate values of W in this order with a moderate amount of computation is shown in

Figure 16. While the A_1 values are being computed, the following algorithm is carried out using this configuration.

1) Reset Q to W^6
2) Reset P to W^0
3) Multiply Q by W^2
4) Use the W in P
5) Multiply P by Q and store the result in P
6) Go back to 4) 7 times, then on to 7)
7) Go back to 2) 3 times, then on to 8)

While the A_2 values are being computed, the algorithm given below is followed.

8) Reset Q to W^7
9) Reset P to W^0
10) Multiply Q by W^1
11) Use the W in P
12) Multiply P by Q, store the result in P
13) Go back to 11) once, then on to 14)
14) Go back to 9) 7 times, then back to 1)

The configuration in Figure 16 shows the P and Q registers as indicated and also indicates how the W^1 and W^2 constants are wired into the arithmetic unit. Note that a maximum of two multiplications is performed in finding a single value of W.

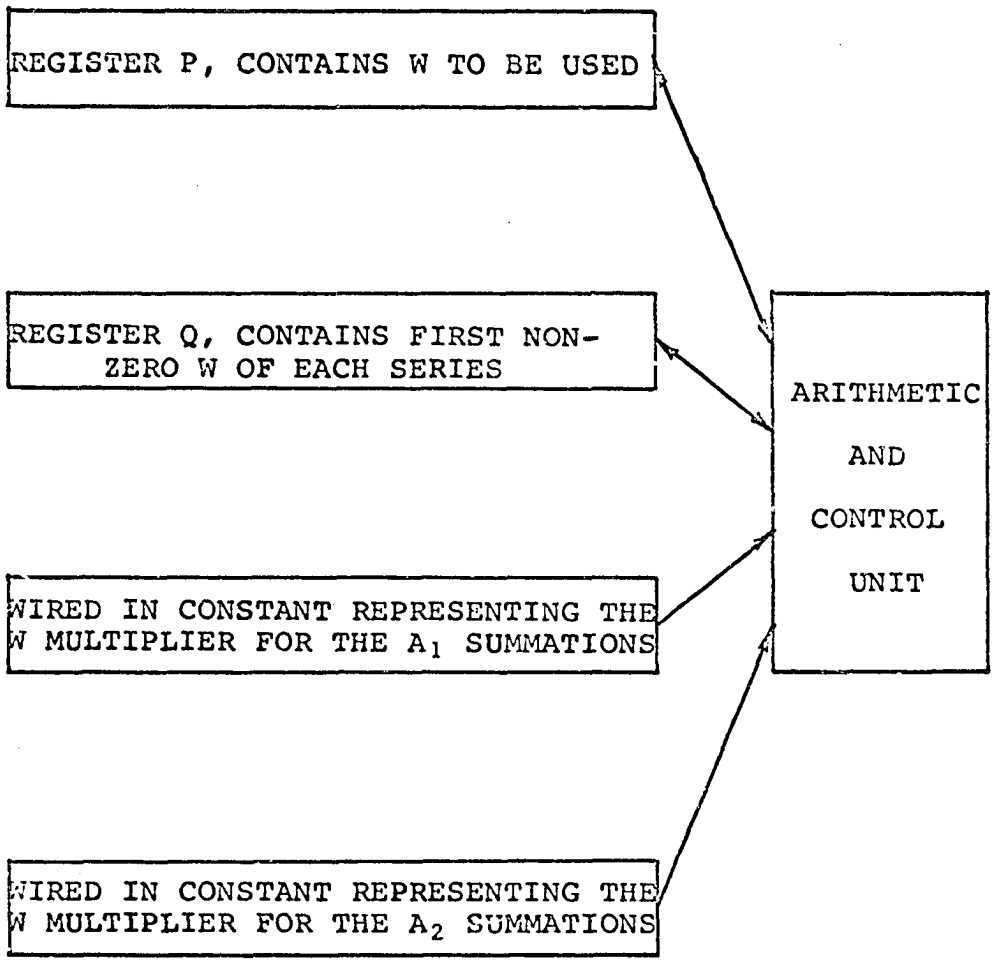


Figure 16. Configuration used in computing values of W for the r_1r_2 analyzer

3. The smoothing network

The requirements on the smoothing network for this analyzer are exactly the same as discussed in Section IV.B.4 for the binary analyzer as both analyzers evaluate essentially the same equation (i.e. Equation 11).

4. Characteristics of the N = 4·2 analyzer

This machine is somewhat more flexible than the binary analyzer because N is not required to be a power of 2. The price paid for this characteristic is that more multiplications are required during each sample period. For the example of N = 8, the binary analyzer required that each arithmetic unit perform one multiplication to find a value of W and one to form the next value of A during each sample period. The $N = r_1 r_2$ analyzer requires a maximum of 2 multiplications to form each W and 4 multiplications to form 4 terms of the summations during each sample period. In addition to the computation performed during the T second interval, more multiplications are also required upon outputting each spectral value. If we assume a multiplier which is just "keeping up", this would mean that after the $8\Delta T$ interval, more multiplications would have to be performed taking approximately $2\Delta T$ seconds. Thus there would be a $2\Delta T$ second time interval following each $8\Delta T$ second record during which the next record could not be read in.

If r_1 had been 2 and r_2 had been 4, the required

multiplication speed would be less but the "dead time" between records would be greater. Of course, if a time compression unit is used with a faster multiplier, the dead time would not have to be seen by the input signal.

C. The $N = r_1 r_2$ Analyzer in General

The data routing configuration required for any value of r_1 and r_2 is shown schematically in Figure 17. The operation is directly analogous to that of the $N = 4 \cdot 2$ analyzer.

The W generation algorithm is written for the general case as follows:

1) Reset Q to W^{N-r_2}
2) Reset P to W^0
3) Multiply Q by W^{r_2}
4) Use the W in P
5) Multiply P by Q and store the result in P
6) Go back to 4), N-1 times then on to 7)
7) Go back to 2), r_1-1 times, then on to 8)
8) Reset Q to W^{N-1}
9) Reset P to W^0
10) Multiply Q by W^1
11) Use the W in P
12) Multiply P by Q and store the result in P
13) Go back to 11), r_2-1 times, then on to 14)

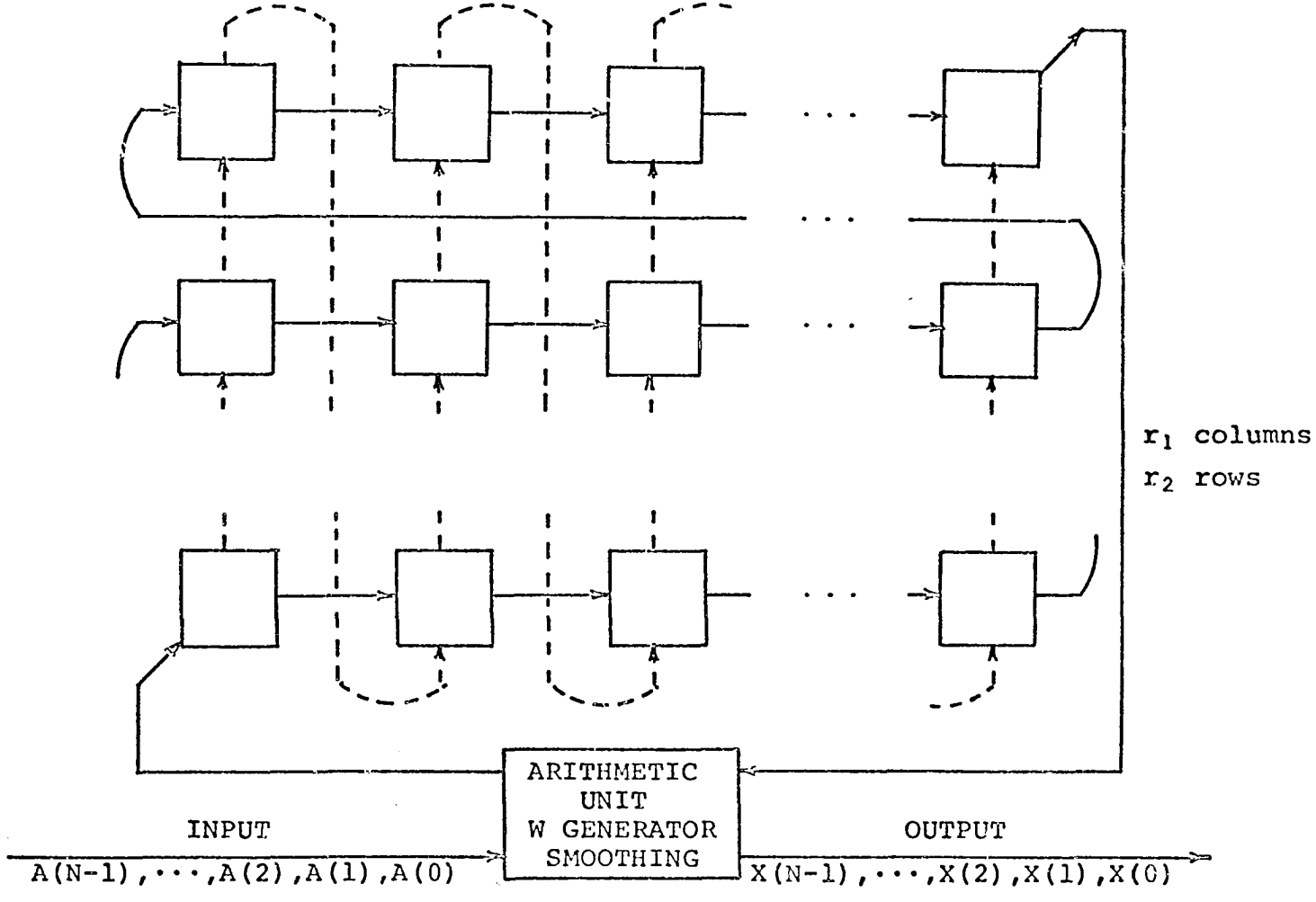


Figure 17. The $r_1 r_2$ analyzer composed of r_1 columns and r_2 rows

14) Go back to 9), N-1 times, then back to 1)
(NOTE: If only N/2 values of X are being computed, the transfer from 14) to 9) will only have to be made N/2-1 times for each record.)

During one sample period the arithmetic unit must be capable of performing a maximum of $(2r_1 + 1)$ complex multiplications. For each spectral component computed during the "dead time", $(2r_2 + 1)$ complex multiplications must be performed. A tradeoff generally occurs in the choice of the r_1 and r_2 values since r_1 determines the highest allowable sampling rate for a given arithmetic unit, and r_2 determines the length of the "dead time".

D. Components Required

An idea of the cost involved in building the r_1r_2 analyzer can be obtained by estimating the number of components required as a function of N. These estimates are tabulated in Part (b) of Figure 11 on page 41.

VI. DISCUSSION OF RESULTS

A. Application of the Analyzers

When determining which of the analyzers can be applied to a particular problem, the following points must be considered:

1) The highest required sampling rate.
2) Whether N can be made a power of 2.
3) The amount of allowable dead space between records.
4) The required multiplication speed of the arithmetic units.
5) Cost comparisons for different choices of r_1 and r_2.

If a sampling rate of 500,000 samples per second is specified, it is quite clear that the binary analyzer would be easier to implement than the r_1r_2 analyzer unless r_1 could be made 2 or 3. In the r_1r_2 analyzer this would imply a very long "dead time" between records to complete the computations. If this "dead time" is not allowable, the problem dictates the binary analyzer. At this sampling rate even the binary analyzer may need to be modified, depending on the multiplication speed of the available arithmetic units. Inputting 500,000 samples per second implies that the complex products for finding the W and A terms must be formed during every 2 microsecond sample period.

At lower sampling rates, and with a "dead time" allowed, the r_1r_2 analyzer would generally be preferred. The

constraints that $(2r_1 + 1)$ complex multiplications must be performed during each sample period, and that $(2r_2 + 1)$ complex multiplications per spectral component must be performed during the "dead time", usually lead to the desired values of r_1 and r_2 and fix a maximum allowable multiplication time.

The choice of one analyzer over the other may change considerably, however, as new devices are marketed. If the thin-film shift registers designed by Spain, Jauvtis and Fuller (12) can be sold for 10 cents per bit or less, this would mean that the modular analyzer would suddenly be relatively inexpensive. Of course, the 10 cent per bit figure may also be approached in the future for an integrated circuit version of the $r_1 r_2$ analyzer.

In any case, even though the analyzers have been discussed in one particular implementation, advances in technology, and the availability of different types of components might mean that a different implementation could be considered while keeping the basic organization of the analyzer the same.

VII. AREAS FOR FURTHER STUDY

One area which could be expanded upon is using the Cooley-Tukey algorithm to find both the Fourier transform and inverse transform, thereby allowing rather arbitrary digital filtering to be performed in real-time.

In this thesis, it has been assumed that the function being sampled is real valued (as opposed to complex valued). The implications of complex sampling and analysis is another area which could be studied considerably.

As cross correlation functions and autocorrelation functions can also be estimated from complex spectral estimates, a digital real-time cross correlator could be considered.

VIII. CONCLUSIONS

Two basic digital machine organizations have been presented which can in most cases find spectral estimates of a time function in real-time. The basic operation of the machines draws heavily upon the algorithm for the machine calculation of complex Fourier series presented by Cooley and Tukey in the April 1965 issue of Mathematics of Computation (5).

The binary analyzer is restricted to cases where N may be chosen to be a power of 2, but is very versatile in terms of allowable sampling rates. A sampling rate in excess of 500,000 samples per second is not unreasonable while the same machine could also be used with a sampling rate of less than one sample per hour. The fact that each section of the analyzer works somewhat independently of the others indicates that the possibility of changing the value of N, as well as the sampling rate, is feasible. The cost of the analyzer can be broken down into three basic sections: the cost of the arithmetic units, the cost of the shift registers, and the cost of the reordering networks.

The $r_1 r_2$ analyzer is restricted to the case where N may be chosen to be the product of two integers and where a "dead time" is permitted between records. It is therefore well suited for analysis of echo returns and siesmic returns or any phenomena involving finite length records with a "dead time"

between records.

The r_1r_2 analyzer must perform a substantial number of calculations during one sample period so it will not generally operate at as high sampling rates as the binary analyzer, but it requires fewer components so it would probably be less expensive. When no "dead time" is allowable, the r_1r_2 analyzer could be used in conjunction with a time compressor whereby the data would be read and stored during the "dead time" and then put into the analyzer at a rate faster than it was originally sampled. This would be quite convenient when one analyzer is to be time shared by several different signals as in this case the extra data storage is needed anyway. The cost of the r_1r_2 analyzer lies mainly in the array of shift registers, which must hold N complex numbers, and in the price of the one arithmetic unit.

IX. LITERATURE CITED

1. Anstey, N. A. Correlation techniques: a review. Tulsa, Oklahoma, Seismograph Service Corporation. 1964.

2. Blackman, R. B. and Tukey, J. W. The measurement of power spectra. New York, Dover Publications, Inc. 1958.

3. Brown, G. B. and Nilsson, J. W. Introduction to linear systems analysis. New York, John Wiley and Sons, Inc. 1962.

4. Chu, Y. Digital computer design fundamentals. New York, McGraw-Hill Book Company, Inc. 327-328. 1962.

5. Cooley, J. W. and Tukey, J. W. An algorithm for the machine calculation of complex fourier series. Mathematics of Computation 19: 297-301. 1965.

6. Davenport, W. B., Jr. and Root, W. L. Random signals and noise. New York, McGraw-Hill Book Company, Inc. 1958.

7. Lee, Y. W. Statistical theory of communication. New York, John Wiley and Sons, Inc. 1960.

8. Marks, W. and Pierson, W. J. The power spectrum analysis of ocean-wave records. American Geophysical Union Transactions 33: 834-844. 1952.

9. Nehari, Z. Introduction to complex analysis. Boston, Allyn and Bacon, Inc. 1961.

10. Papoulis, A. The fourier integral and its applications. New York, McGraw-Hill Book Company, Inc. 1962.

11. Potter, R. K. Introduction to technical discussions of sound portrayal. Acoustical Society of America Journal 18: 1-3. 1946.

12. Spain, R. J., Jauvtis, H. I. and Fuller, H. W. Some new thin-film shift register designs. Journal of Applied Physics 36: 1103-1104. 1965.

X. ACKNOWLEDGMENTS

The author wishes to thank Dr. H. W. Hale for his encouragement and his many helpful suggestions during the preparation of this manuscript.

This work was done under the support of a National Science Foundation Traineeship.