BIBLIOGRAPHY

1 B L BUZBEE  G H GOLUB  C W NIELSON  
The method of odd/even reduction and factorization with  
application to Poisson's equation  
Stanford University Computer Science Department Tech  
Report No CS 128 1969

2 L. COLLATZ  
The numerical treatment of differential equations  
Third edition Springer 1960

3 J W COOLEY  J W TUKEY  
An algorithm for the machine calculation of complex fourier  
series  
Math Comp 19 pp 297–301 1965

4 J W COOLEY  P A W LEWIS  P D WELCH  
The fast fourier transform algorithm and its applications  
IBM Watson Research Center Yorktown Heights New York  
1967

5 R W HOCKNEY  
A fast direct solution of Poisson's equation using fourier  
analysis  
J ACM Vol 12 pp 95–113 1965

6 R W HOCKNEY  
The potential calculation  
Conference on Numerical Simulation of Plasma Los Alamos  
1968

7 R S VARGA  
Matrix iterative analysis  
Prentice-Hall 1962
Architecture of a real-time fast fourier radar signal processor

by ARTHUR S. ZUKIN and S. Y. WONG

Hughes Aircraft Company
Culver City, California

SUMMARY

This paper describes the architecture of an all-digital signal processor for a high pulse repetition frequency (high PRF) radar.\textsuperscript{1,2} The processor replaces a bank of hundreds of (approximately) 100 Hz bandwidth analog filters with an equivalent but more capable and smaller, lighter and less expensive digital system. The digital system acts in real-time by converting input signals (time domain) from analog to digital form, collecting sets of such converted signals and then performing a discrete Fourier* transform\textsuperscript{3,4} upon them. Thus, it produces a (frequency domain) result which is equivalent to the output from the bank of analog filters or from a spectrum analyzer.

Because the processor is employed on a full-time basis solely to perform the Fourier transform, it can be designed to do this task at lower cost than a general purpose computer and with lower performance logic circuits and memory than would be required in a general purpose computer. It can perform direct and inverse Fourier transforms and could be used in pattern matching and convolution.\textsuperscript{5} The processor described can be modified and/or adapted to low or medium PRF\textsuperscript{1,3,5,7} and/or synthetic array modes as well as to communications systems. For example, this basic design has been successfully used in communications systems to perform both the direct and inverse transform and serves as both demodulator and modulator.

The selected processor architecture separates the functions to be performed from each other and in most cases assigns physically distinguishable portions of the equipment to the functions because:

1. If the functions are not conceptually separated, the overall task is unduly complex. Like a conventional digital computer, the processor is a collection of simple blocks; though complex considered in the ensemble, they are easily understood separately.

2. Separating functions and associating them one-for-one with equipment demonstrates the equivalence between the generalized fast Fourier (Cooley-Tukey)\textsuperscript{8-18} procedure and the equipment.

3. By designing for the application from the outset one can pick efficient hardware for implementing the functions. The result is a straightforward design with a comparatively small number of efficient functional blocks.

INTRODUCTION

System capability

The analog system which is equivalent to the digital signal processor would have a total bandwidth of 146 kHz. (See Table I.) This would be comprised of two 73 kHz subbands.\textsuperscript{*} It would have two modes of operation.

In the first mode each subband would drive 512 filters having a matched bandwidth of 143 Hz. Thus, the system would provide a total bandwidth of 146 kHz in 1,024 143 Hz filters.

In the second mode, each subband would drive 1,024 filters having a matched (filter) bandwidth of 72 Hz. Thus, the system would provide a total bandwidth of 146 kHz in 2,048 72 Hz filters.

In the digital system, the sampling rate for both in-phase and quadrature signals* is 146,000 per second per subband. I.e., in each of the two subbands, there is a vector sampling rate of 146,000 per second and the total vector sampling rate is 292,000 per second which

\* Fast Fourier transform digital processing and digital filters and their relationship are discussed in the appendix.

\* See "Terminology Used."

---

From the collection of the Computer History Museum (www.computerhistory.org)
TABLE I — Comparison of Analog and Digital Systems

<table>
<thead>
<tr>
<th></th>
<th>Mode 1</th>
<th>Mode 2</th>
</tr>
</thead>
<tbody>
<tr>
<td>Analog System</td>
<td></td>
<td></td>
</tr>
<tr>
<td>a) Total Bandwidth, Hz (2 subbands)</td>
<td>146,000</td>
<td>146,000</td>
</tr>
<tr>
<td>b) Bandwidth per Subband, Hz</td>
<td>73,000</td>
<td>73,000</td>
</tr>
<tr>
<td>c) No. of Filters per Subband</td>
<td>512</td>
<td>1,024</td>
</tr>
<tr>
<td>d) Filter Bandwidth, Hz</td>
<td>143</td>
<td>72</td>
</tr>
<tr>
<td>Digital System</td>
<td></td>
<td></td>
</tr>
<tr>
<td>a) Total Bandwidth, Hz (2 subbands)</td>
<td>146,000</td>
<td>146,000</td>
</tr>
<tr>
<td>b) Bandwidth per Subband, Hz</td>
<td>73,000</td>
<td>73,000</td>
</tr>
<tr>
<td>c) Filters &quot;Saved&quot; per Frame</td>
<td>512</td>
<td>1,024</td>
</tr>
<tr>
<td>d) Filter Bandwidth, Hz</td>
<td>143</td>
<td>72</td>
</tr>
</tbody>
</table>

is twice the bandwidth of the analog system and satisfies the Nyquist requirement.

In the first mode, the frame time (that is, the time for one set of samples) is 7 msec, which results in a filter width of (0.007 sec)⁻¹ or 143 Hz as in the equivalent analog system. During the 7 msec frame 1,024 vector samples are made for each of the two subbands. Thus, 1,024 data enter the processor for each subband even though only half this quantity (512) filters are to be created. This is necessary because of the 2:1 oversampling mentioned in the previous paragraph. At the end of the fast Fourier procedure the desired 512 "filters" are retained and the other 512 are ignored. (If the fast Fourier procedure were done base-2, one could merely skip one-half of the last base-2 procedure.) Likewise in the second digital mode, twice as many samples are taken as the desired number of retained filters. The quantities are 2,048 vector samples in a 14 msec frame per subband and 1,024 filters are retained for each subband. The filter width is (0.014 sec)⁻¹ or 72 Hz.

It is important to note that the provision of both 512 143 Hz filters and 1,024 72 Hz filters in the same analog system is very expensive and the likely compromise for the sort of system described here would be to supply only 512 filters which in mode one would be switched from one subband to the other. Another highly likely economy move would be the use of single compromise bandwidth somewhere between 72 and 143 Hz and use of the same 512 filters for both modes one and two. Despite its lower cost the digital system does not need to make either of these sacrifices. If enough memory is provided for the 1,024 filter per subband case and enough speed for the 7 msec frame case, the "variable" filter bandwidth is "free."

Radar background information¹,²,⁶,⁷,¹⁹

Radar systems determine target range (R) by measuring time (t) between transmission of a pulse and reception of the target echo signal. Since the radar signal travels a distance of 2R, one obtains

\[ t = \frac{2R}{c} \quad \text{or} \quad R = \frac{tc}{2} \]  

(1)

where c is the velocity of propagation, roughly 160,000 nautical miles per second. If the target is moving with respect to the radar the target velocity (v or R economic) along the line of sight will cause the received echo signal to differ from the transmitted signal frequency (f) by an amount, \( \Delta f \). This is the well known doppler effect.

\[ \Delta f = \frac{(2v/c)f}{\lambda} \]  

(2)

Replacing \( c \) by \( \lambda f \) (\( \lambda \) is the wavelength of the radar) gives

\[ \Delta f = \frac{2v}{\lambda} \]  

(3)

For velocities of interest (up to 5000 feet per second) and wavelengths of interest (0.03 foot to 3 feet) the absolute value of the doppler frequency shift may be as great as 300,000 Hz and the frequency region of interest can easily span one-half megahertz.

Because the pulse radar is inherently a sampled data system, the spectrum¹,²,⁴,⁷ of the received signal is complex even in an idealized case. The spectrum of the transmitted signal consists of spectral lines separated from each other by the pulse repetition frequency (\( f_r \)); the envelope of the spectral lines has a \( (\sin X)/X \) shape, is centered about the transmitted frequency (f), and has a frequency width of \( 2/\tau \) between the first pair of zero crossings where \( \tau \) is the width of the transmitted pulse. The important attribute of the transmitted waveform is that it is comprised of \( f, f \pm f_r, f \pm 2f_r, f \pm 3f_r, \) and so on.

As a result, the received signal is not simply \( f + \Delta f \) as would be surmised from Equation (2) but contains many frequencies, \( f + \Delta f, f \pm f_r + \Delta f, f \pm 2f_r + \Delta f, f \pm 3f_r + \Delta f \) and so on. Therefore, doppler frequencies of \( F, F + f_r, F + 2f_r \) etc. are indistinguishable from each other. Because of this the total doppler frequency bandwidth that can be measured unambiguously is \( f_r \).
The consequence is that in order to measure velocity unambiguously, \( f_r \) must be at least as great as the total doppler bandwidth and the pulse repetition period (PRP) which is the inverse of the PRF may perforce be of the order of a few microseconds or 10 microseconds. This means that the maximum range which can be measured unambiguously is very, very small. For example, if the PRF is 100,000 per second (PRP is 10 microseconds) the maximum unambiguous range which can be measured is about 0.8 nautical mile and all ranges will be measured modulo-0.8 nautical mile.

On the other hand as long as the doppler frequencies span no more than 100 kHz, they can be unambiguously measured modulo-100 kHz. With the above parameters given, the system would be what is called a high PRF radar; that is, it would be a radar which measures range ambiguously but measures doppler unambiguously.

If the PRF were changed to 1000 per second, (PRP of 1000 microseconds), the unambiguous range would be 80 nautical miles and ranges would be measured modulo-80 nautical miles. But doppler frequencies could be measured unambiguously only as long as they spanned no more than 1 kHz and would be measured modulo-1 kHz. For most applications the doppler information would be essentially useless at this PRF.

With these parameters, the system would be a low PRF radar for targets of less than 80 miles range. That is, range would always be measured unambiguously but velocities of practical interest would be measured ambiguously.

When the radar parameters are such that both range and velocity are measured ambiguously it is said to be a medium PRF radar. Although this appears to be a useless system, it is not necessarily so; with modern digital equipment both range and frequency (doppler) can be measured ambiguously with each of a set of related PRF's and the unambiguous values computed from the sets of measurements. For example, the combination of ambiguous range measurements modulo-19 nautical miles and modulo-10 nautical miles yields unambiguous range out to 190 nautical miles.

Terminology used

The system described represents an effort to apply new technology to an old problem. Moreover, the technology is digital and the problem is traditionally solved with analog implementation. Therefore, it appears appropriate to define a few terms which are not familiar to all.

*Sample and hold* circuits which are the source of the input signals to the processor are used in an analog-to-digital converter when it is desirable to make a measurement of a signal and to know precisely when the input signal corresponds to the results of the measurement.

*In-phase and quadrature signals* \((I\text{ and } Q)\) refer to the component of the input signal which is in phase with a reference signal and to the component which is in quadrature (leads or lags by 90 electrical degrees) with the in-phase signal. These correspond to the real and imaginary portions of the input signal which is assumed to be the vector sum of all of the input components.

*Subbands* refer to the division of the total input frequency band into two subbands each of which contains one-half of the total bandwidth of the system. This may be done for a number of reasons. For example, the input signal may have too large range between maximum and minimum amplitude for the analog-to-digital converter. In this case, and if the distribution of energy across the band is fairly uniform, the range may be reduced by the use of subbands each covering a portion of the spectrum.

Since an all-digital processor uses no physical filters, the equivalent of analog filter responses is synthesized by the processor using the time domain to frequency domain Fourier transform.* At the end of the transform procedure the input data has been converted to a set of vector numbers representing the voltages one would have obtained from a set of analog filters with the same characteristics as the digital filters. Each vector is called a *filter* or a *complex filter* and is subsequently multiplied by its complex conjugate to obtain a quantity representative of the power which would have passed through the equivalent analog filter in the same time interval.

**SUMMARY SYSTEM DESCRIPTION**

The digital processor block diagram is shown in Figure 1. The driving circuits (sample and hold) and

*See appendix.*
and one of the three fast Fourier steps has been completed.

During the next fast Fourier step, the procedure is similar with alternate bursts of five or six memory cycles employed to read in new data and 16 memory cycle bursts employed to exchange memory data destined for the vector processor with processed data from the vector processor. There are, however, two important differences from the first step. These are:

1. In order to perform the required rotation, operands (complex) leaving the memory are multiplied by appropriate unit vectors in the vector multiplier before entering the vector processor. For a rotation of $\theta$ radians, the unit vector $\cos \theta + i \sin \theta$, is generated by the sine/cosine table which is provided with the rotation angle by the address sequence control. Trigonometric functions of angles from zero to $\pi/4$ are generated directly by an interpolation procedure; functions for angles from $\pi/4$ to $2\pi$ are derived from the directly generated functions. The procedure and its mechanization are simple and do not require extensive equipment.

2. Operands are taken in groups of eight (G-8) for the second step of a 14-millisecond 4,096 sample frame and in groups of four (G-4) for the second step of a 7-millisecond 2,048 sample frame. However, the transfers between the memory and the vector processor still occur in bursts of 16 memory cycles. The 16 words involved represent two G-8s or four G-4s.

During the last of the three fast Fourier steps processing again utilizes G-16s. However, the last step differs from the first in that the complex data is multiplied by the appropriate unit vectors in the vector multiplier. The multiplication and sine/cosine procedures are identical to those employed in the second step.

The data returned to the memory during the third (last) step represents the complex magnitudes of the filtered input signals (filters) and enters the memory during the last one-third of the frame after that in which it was obtained. During the next frame, as new data enters the memory in bursts of five or six memory cycles, an equal quantity of filters is read out of the memory. Like all information leaving the memory, the filters pass through the vector multiplier where each complex filter is multiplied by its own complex conjugate to provide the square of its magnitude according to the well-known relationship,

$$\text{magnitude}^2 = f(a + ib) = a^2 + b^2 = (a + ib)(a - ib).$$

The squared magnitudes are proportional to the power output of the equivalent analog filters and are the digital processor output signals to the threshold circuits. In these circuits they are compared against a threshold level and an output is generated whenever the threshold is crossed.

**OVERALL DESIGN DECISIONS**

Some system design decisions and equipment choices and mechanizations are described in this section.

**Memory selection**

For a particular level of capability, probably the most meaningful measure of the worth of a digital processor design is the ability to "get by" with a single economical, easily produced memory with little penalty in other portions of the processor. The memory in this design uses 4,096 words of 32 bits with a cycle time of 1.71 microseconds. Thus, it stores 131,000 bits and requires an information rate of less than 19 bits per microsecond.

Because data for an entire 7- or 14-millisecond frame is processed as a "set," the absolute minimum amount of memory that required to store the 4,096 samples obtained during the longer frame. If only this much storage is provided, processing must be performed instantaneously at the end of a data gathering period and results transferred to the using equipment instantaneously. These conditions are obviously impossible. A reasonable upper limit is the storage required for two frames of data or $2 \times 4,096 = 8,192$ samples. This results in a system in which data processing takes one frame time and occurs during the next frame after the data is gathered, and the processed data is unloaded during the second frame after its samples were loaded into the memory simultaneously with the loading of new data.

Serious effort was made to avoid using a memory large enough for two frames of data, but this was not

![Figure 3](...staggered processing memory storage requirement and frame-timing for 14 milliseconds, 2 subbands, 2,048 samples per subband per frame)
the receiving circuits (threshold circuits) are conventional and will not be considered here.

Data flow in the digital processor is designed to provide a simple sequence in which information is cycled between the memory and the vector processor until the (input) vector samples are converted to (output) vector-quantities representing the real and imaginary portions of the signal voltage which has "passed through" an equivalent analog filter. Input data (Figure 1) flows into the memory and thence from memory through the vector multiplier and vector processor and back to the memory. This loop flow occurs three times, the information returned to the memory after the third step representing the filter outputs. These are then read out of the memory through the vector multiplier in which they are converted into the squares of the (scalar) magnitudes, and the output of the digital processor.

In-phase and quadrature input signals for each of the two subbands are digitized in the analog/digital converter to provide 8-bit (seven bits plus sign) conversion of the in-phase and the quadrature signals. A total of 585,144 8-bit conversions occur each second; each set of four 8-bit conversions represents the in-phase and quadrature components \((I \text{ and } Q)\) of the signals of the two subbands and results in one 32-bit word comprised of two 16-bit halves; each half contains the \(I\) and \(Q\) for one subband. Thus, the average rate into the memory is one-quarter of 585,144 or 146,286 words per second.

Five or six input words are collected in the sample buffer and in a burst of five or six successive memory cycles (see Overall Design Decisions) are read into the memory at the same time that an equal number of filters* are read out of the memory. The procedure is an exchange of data in which data flowing into the memory replaces the data flowing out of the same memory location at a rate of one word per 1.71 microseconds memory cycle. Data leaving the memory represents the complex filter output voltage and is multiplied by its complex conjugate in the vector multiplier to form the square of the magnitude.

When the processor is first turned on, no useful processing can occur until one complete frame of data has entered the memory. Subsequently groups of 16 operands per subband (called \(G\)-16's**) are read out of the \(2M\)-word memory from locations \((0, M/16, 2M/16, 3M/16, \ldots), (1, M/16 + 1, 2M/16 + 1, 3M/16 + 1, \ldots), (2, M/16 + 2, 2M/16 + 2, 3M/16 + 2, \ldots), \) and so on. The 16 words are read in a burst of 16 memory cycles while five or six words are collected in the input buffer. Between successive 16 word bursts, a burst of five or six memory cycles is employed to load five or six new data words and unload five or six filters.

Like all data leaving the memory the \(G\)-16 passes through the vector multiplier word by word, but in this first step is passed through the vector multiplier without change and on into the vector processor. In the vector processor, the \(G\)-16 data for each of the subbands is subjected to a base-16\(^6\) fast Fourier transform procedure. The time allowed for this is five or six memory cycles. (See timing diagrams, Figure 2.) At the end of the base-16 procedure, the 16 pairs of results*** in the 16 words of the \(G\)-16 are returned word by word to the memory in a burst of 16 memory cycles to replace the 16 words of the next \(G\)-16 as they are read out of the memory through the vector multiplier to the vector processor, etc.

Ideally, the processed \(G\)-16 would be returned to locations in the memory from which the (same) input \(G\)-16 had been obtained. This would require one memory cycle to read the data plus another memory cycle to write, resulting in a total of two memory cycles per word. However, the information exchange can be performed during a single memory cycle per word. The effectively double memory speed is not obtained without penalty and forces the use of an address sequence control (see Overall Design Decisions) to keep track of the changing memory addresses. After the last \(G\)-16 passes through the vector processor, it is returned to the location from which the first \(G\)-16 was obtained.

* Which represent the processed result of data gathered two frames before the input frame.

** Each of the 16 words in the \(G\)-16 contains 16 bits of subband A data and 16 bits of subband B data.

*** One result per subband per word.
successful. A scheme called “staggered processing” was devised in which the start of the frame times for the two subbands A and B, differed by 7 milliseconds (i.e., one-half of a 14-millisecond frame) and the information for the individual subband frame was processed during a period of 7 milliseconds. This is illustrated by Figure 3; the alternate of nonstaggered processing is illustrated for the 7- and 14-millisecond frame times by Figures 4 and 5. Although potentially useful, staggered processing was not adopted because of a number of difficulties, namely:

1. It appears desirable to store (at least) two samples or pieces of data in each memory word. If staggered processing is used, the two must be from the same subband. However, it is desirable that the two represent equivalent data from the (in this case) two subbands in order that the use of twice as many trigonometric function “look-ups” and twice as fast a trigonometric table be avoided.

2. A requirement for a special sorting pass of the filtered data to obtain the desired output sequence and/or the acceptance of n parallel streams, each of which is a properly sequenced stream containing 1/nth of the results.

3. Greater complexity in the address sequence control because of the multiplicity of address patterns that accompany the staggered processing scheme.

Choice of arithmetic base of vector processor

The choice of arithmetic base8 for the vector processor is closely related to selection of the memory. The choice of a particular processor base leads to a total memory input–output data rate and the choice of a memory data rate places a lower limit on the processor (arithmetic) base. Accompanying each base is a minimal number of registers within the vector processor; this is the product of the base times the number of samples per memory word. Thus, the higher the base, the more registers are needed in the vector processor itself. However, the higher base processor can be internally mechanized as a series of lower base substeps. Thus an externally viewed base-16 processor could internally use four base-2 or two base-4 substeps. If a higher base is used in this manner, the total amount of equipment, other than registers, is only weakly dependent on the base. In the (externally viewed) base-16 design described, fewer multipliers are used in the entire system than would have been required if a base-4 processor were used.

Moreover, the use of a high base is a definite advantage in reducing the speed of the memory and the arithmetic circuits. This is so because for N samples, the number of memory cycles and of multiplications per unit of time varies as \( \log B N \) where \( B \) is the arithmetic base. Since \( \log B N = \log N / \log e B \), the required memory speed and arithmetic speed vary inversely with the natural logarithm of the base.

If the base is two or four, the vector operations within the processor are trivial; that is, they are multiplications by the sine and cosine of 0, \( \pi /2 \), \( \pi \) and \( 3\pi /4 \) and thus no explicit multiplier is required within the vector processor. Therefore, except under unusual circumstances, the lowest base to be considered should be four.

* As stated in Summary System Description and illustrated in the system block diagram (Figure 1), all vector multiplications other than those within the vector processor, and all rotation of vectors are performed by the vector multiplier through which all information flows as it leaves the memory.
In the case of the system described here, the actual selection of a base was based on the criteria indicated in the first paragraph of this section; the number of samples to be processed (4,096) was divided into the amount of time (14 milliseconds) allowed for processing. This gave a total memory cycle time of 3.42 microseconds for all accesses to each sample—assuming one sample per memory word. Because one and three sample-per-word (16 and 48 bit per word) memories led to less economical systems, the decision was made to go to two samples per word. This results in 6.84 microseconds total cycle time for all accesses to each sample. Based on this number, a curve of number of passes through the entire memory can be drawn as in Figure 6.

Assuming the previously stated minimum cycle of 1.5 microseconds is used, Figure 6 shows that the maximum number of passes through the entire memory is 4.56. In general, the use of fractional passes does not seem to offer any advantages except for two special cases:

1. Special operations at the start or end of a frame, such as time to operate on the address sequence control. Such operations probably have an effect of no greater than a few percent on the memory speed.

2. The use of a first processing step as new samples enter the memory. If a base of two is used, processing of samples could start just after one-half of the samples have been loaded; if a base of four is used, processing could start just after three-fourths of the samples are loaded; for a base of 8, processing could start just after $B - 1/B$ of the samples are loaded. Such a step was considered in this design, but was rejected because it did not result in either reducing the number of subsequent passes through the memory or in a significant simplification of the vector processor. (If a last pass through the vector processor were performed as data left the memory, the use of an input pass would have been profitable. The scheme, which was rejected, would have used 4-2 passes through the memory. Processing would be base-2 at input and base-2 at output; three base-8 processing steps would occur between input and output; $2 \times 8^3 \times 2 = 2,048$ = number of samples per subband per frame.)

Because the use of fractional passes was rejected, the selected memory was that for the next integral number of passes below the 4.6 (that is four) allowed by a 1.5-microsecond memory (see Figure 6). Since one of the four passes is required to load samples into the memory and simultaneously unload filters from the memory, this resulted in allowing three passes for processing. If the same base is used for all three steps with a total of 2,048 samples per subband, the base used must be $(2048)^{1/3} = 12.7$. A practical compromise is to use 16, 8 and 16 for the bases of the three passes. It is preferable that the passes be in the sequence listed since the symmetry (that is, having the first and last of the three passes use the same base) avoids unnecessary complexity in the address sequence control. Similarly for the 1,024 samples in the 7-millisecond case, bases of 10, 4 and 16 have been selected.

A base-8 processor requires a total of five passes through the memory which must provide a cycle time of 1.4 microseconds and eliminates registers for eight words or $8 \times 32$ bits within the vector processor; this tradeoff is about even in dollars, but for a new design favors the slower memory in schedule risk and development cost. A base-4 processor requires a memory cycle of 0.98 microsecond and eliminates registers for 12 words or $12 \times 32$ bits within the vector processor and might also save six 8-bit (scalar) multipliers, but requires much faster circuitry (arithmetic and otherwise) throughout the system. Thus no cost advantage appears to accrue to the lower base even though it requires a faster memory. Considering all such factors, the selected system appears optimum for its intended application since it is at worst no more expensive than the alternatives and is the most easily designed and developed.

Figure 6—Maximum number of passes through memory for 6.84 microseconds for all accesses to each sample

* Here "just after" means when one more sample has been digitized.

** The use of a processing pass concomitant with data input appears to complicate the timing and does make the data flow more complex.


** Actually five passes of base-4 and one of base-2 (2,048 pulses per subband).
**Burst processing**

The concept of burst processing allows uninterrupted use of a period equivalent to five memory cycles for the operations within the vector processor. Thus it permits acceptably slow arithmetic circuits within the processor without a disproportionate increase in the number of buffer registers.

The read-write cycle associated with loading a new sample into the memory occurs once for each sample and occurs three times during processing of each sample; i.e., of each four memory cycles three are required for processing alone. Thus, while 16 memory cycles are used to read a new set of 16 operands per subband into the processor and transfer a set of 16 results per subband from the processor, the average time span during which these 16 memory cycles occur is actually \( 4/3 \times 16 = 21 \frac{2}{3} \) memory cycles.

On the other hand, as noted previously, if an internal base of two is used, processing in the base-16 vector processor cannot start until the ninth operand is available and cannot start until the thirteenth operand is available if an internal base of four is used. A more stringent limitation results from the decision to always exchange memory data; each time an operand is read from the memory, there must be a result ready to put back in the memory. If this decision is retained, two undesirable alternatives result: to perform all processing in one memory cycle using very fast circuits or to provide a redundant set of registers for 16 words. The latter would allow a procedure similar to that which is used with a random access memory large enough for two frames of data and in which one-half the storage is used for input-output while the other half is used for processing. Since neither alternative is pleasant, it is highly desirable that a method be conceived to increase the allowed processing time to four or five uninterrupted memory cycles.

The scheme adopted was to perform the transfer* of the 16 operands of the two subbands from the memory (through the vector multiplier) to the vector processor in one burst of 16 memory cycles. During the 16 cycles, new samples destined for the memory from the analog/digital converter are stored in an integrated circuit buffer of six words each of which represents one sample from each of the two subbands. The six-word buffer may store either five or six words, as illustrated in the timing diagram of Figure 2.

As shown in that figure, four pairs of samples are obtained during the first 16-memory-cycle burst used to communicate with the vector processor and a fifth pair is obtained soon thereafter. The five pairs are then transferred, in a burst, to the memory as an equal number of results are read out of the memory through the vector multiplier to the threshold circuits.

The procedure is repeated in bursts as depicted in the figure with every third input burst transferring six sample pairs rather than five.

**SYSTEM DESCRIPTION**

**Address sequence control**

The functions of the address sequence control (see Figure 1) are:

1. To determine the locations (addresses) in the random access memory (RAM) from which data is read and into which data is written.
2. To provide the sine/cosine table with a measure of the rotation angles (unit vectors) to be used to rotate vectors leaving the memory for the vector processor.

*As each operand is transferred from memory to processor, a result (from a set of 16 previous operands) is transferred from the processor into the same memory position.
To avoid confusing the reader with a mathematical notion (of the fast Fourier transform procedure) involving many changeable subscripts, the explanation of the address sequence control is in terms of exemplifying logic block diagrams for the 2,048 sample per subband case.

**Description of processing sequence**

Let

\[ f(j) = \frac{1}{2048} \sum_{k=0}^{2047} m_k e^{i\theta k} \]  

where \( \theta = 2\pi/2048 \) and \( i = \sqrt{-1} \). Let

\[ j = j_2128 + j_116 + j_0 \]  

\[ k = k_2128 + k_116 + k_0 \]

where

\[ j_2, j_0, k_2, k_0 = 0, 1, 2, \ldots 15 \]  

\[ j_1, k_1 = 0, 1, 2, \ldots 15 \]

\[ f(j_2, j_1, j_0) = \frac{1}{2048} \sum_{k_2=0}^{15} \sum_{k_1=0}^{15} \sum_{k_0=0}^{15} m_{k_2128+k_116+k_0} \exp[i\theta(j_2128 + j_116 + j_0)(k_2128 + k_116 + k_0)] \]  

\[ = \sum_{k_2=0}^{15} \frac{1}{16} \exp[i\theta(j_2128 + j_116 + j_0)k_2] \sum_{k_1=0}^{15} \frac{1}{8} \exp[i\theta(j_116 + j_0)16k_1] \sum_{k_0=0}^{15} \frac{1}{16} m_{k_2128+k_116+k_0} \exp[i\theta j_0128k_0] \]

**Step 1** (Base-16) Compute

\[ f(j_0, k_1, k_0) = \frac{1}{16} \sum_{k_2=0}^{15} m_{k_2128+k_116+k_0} \exp[i\theta j_0128k_2] \]  

**Step 2** (Base-8) Compute

\[ f(j_0, j_1, k_0) = \frac{1}{8} \sum_{k_1=0}^{7} f(j_0, k_1, k_0) \exp[i\theta(j_116 + j_0)16k_1] \]

**Step 3** (Base-16) Compute

\[ f(j_0, j_1, j_2) = \frac{1}{16} \sum_{k_2=0}^{15} f(j_0, j_1, k_0) \exp[i\theta(j_2128 + j_116 + j_0)k_0] \]

*Address control for readout sequence*

Equations (9), (10) and (11) constitute the three steps in the algorithm adopted for this processor and Equations (7) and (8) define the ranges of the \( j \)'s and \( k \)'s. The first step sorts the data according to the lowest digit of the frequency representation. Since the summation is over \( k_0 \), there are 128 different \( f(j_0, k_1, k_0) \) for each \( j_0 \).

If a counter is organized in three sections corresponding to \( k_2 \), \( k_1 \) and \( k_0 \), in terms of the original input data, one counts in \( k_0 \) and carries from \( k_0 \) to \( k_1 \) and thence to \( k_2 \). This is shown in Figure 7(a).

To form the set of \( f(j_0, k_1, k_0) \) in Equation (9) (Step 1), the summation requires a count of \( k_2 \) and then progresses to cover all combinations of \( k_1 \) and \( k_0 \). It is accomplished by injecting the count pulse into the \( k_2 \) section of the counter and letting the carry flow to \( k_0 \) and thence to \( k_2 \). At the end of this step, the data has been stored according to \( j_0 \) as shown in Figure 7(a).

The next two steps of the algorithm (Equations (10) and (11)) evaluate the sets \( f(j_0, j_1, k_0) \) and \( f(j_0, j_1, j_2) \) and proceed similarly with the counter inputs as shown in Figures 7(c) and (d), respectively. Due to special system requirements, the output must be read out in order of frequency. With the counter significance, in terms of \( j \)'s shown in Figure 7(d), the carry path must be reconnected in order to read out in the proper frequency sequence. This is shown in Figure 7(e). Since this is by design* also the next input sequence, the connection for the output sequence in Figure 7(e) is the same for the next input sequence; thus Figure 7(f) is the same as Figure 7(e) with \( j \)'s replaced by \( k \)'s.

In Figure 7(b), \( k_2 \) carried to \( k_0 \) although carrying to either \( k_1 \) or \( k_0 \) was proper as long as all the combinations of \( k_1 \) and \( k_0 \) are eventually gone through. Similarly, in the step of Figure 7(g), \( k_2 \) carries to \( k_1 \) and thence to \( k_2 \) to again use the original counter configuration. As before, after the first step, the \( k_2 \) counter has the significance of \( j_0 \). The next two steps are again similarly performed as shown in Figures 7(h) and 7(j).

The output sequence shown in Figure 7(k) is identical to Figure 7(a), thus completing one cycle of counter configurations.

*Memory data exchange cycling*

The previous section explains the operation of the readout cycle and ignores the control cycle for returning processed data to the memory. The design described operates by exchanging a set of 16 (or eight) previ-
ously processed data from the vector processor for a set of 16 (or eight) unprocessed data from the memory. This way, only 16 (or eight) memory cycles are required for each batch. Thus, the particular counter configuration of Figure 7 displaces the data after processing, by 16 memory positions. For example, after processing, the 16 pieces of data from \( k_2 = 0, 1, 2, \ldots 15, k_1 = 0, k_0 = 0 \) as shown in Figure 7(b), are returned to \( k_2 = 0, 1, 2, \ldots 15, k_1 = 0, k_0 = 1 \). Finally, the 16 pieces of data from \( k_2 = 0, 1, 2, \ldots 15, k_1 = 7, k_0 = 15 \) are returned to \( k_2 = 0, 1, 2, \ldots 15, k_1 = 0, k_0 = 0 \), which was the source of the first 16 pieces.

This last step requires an extra 16 (or eight) memory cycles in addition to the 2,048 cycles required for each pass of the Cooley–Tukey algorithm.

A simple concept to keep track of the memory address precession is through the description of index registers. Consider a memory with addresses enumerated \( 0, 1, 2, \ldots k, \ldots 2^{11} - 1 \). This transfer may be hidden from outside world if an index adder is connected as shown in Figure 8.

Similarly a sequence of such transfers such as \( m_k \) to \( m_{k+p} \), followed by \( m_{k+2p} \) to \( m_{k+3p} \), etc., can be disguised with an index register. For the purpose of the address sequence generator, the index register in Figure 8 can be made in the form of an accumulating register so that it always contains \( \sum p_i \), modulo \( 2^8 \). Therefore, this configuration can keep track of any number of address shifts.

In Figure 7, if the counters shown are considered as equivalent (to the outside world) to the memory address register and if the modulo \( 2^{11} \) adder and index registers are interposed as shown in Figure 8, the corrections are made as follows:

\[
\begin{align*}
\text{Phase 1} & \\
1 \text{(b)} & \rightarrow 2^9 \\
1 \text{(c)} & \rightarrow 2^7 \\
1 \text{(d)} & \rightarrow 2^4 \\
1 \text{(g)} & \rightarrow 2^4 \\
\text{Phase 2} & \\
1 \text{(h)} & \rightarrow 2^7 \\
1 \text{(j)} & \rightarrow 2^9 
\end{align*}
\]

It is noted that once initialized the index register is never set to zero.

There are two alternate frames to be controlled. If the two occupy separate areas in the memory from addresses 0 to \( 2^{11} - 1 \) and from \( 2^{11} \) to \( 2^{22} - 1 \), the same address control may be used as identical operations delayed by one frame are performed in memory addresses that differ by \( 2^{11} \). If we add \( 2^{11} \) to the index register for the second frame, the general operational area of the memory will be shifted to the other half. The simplest way to cope with the two alternate frames is to double the equipment for the address counters and index registers while sharing the index adder. In this way, the two sequences can be generated independently.

**Phase shifter coefficients**

The three processing steps Equations (9), (10) and (11) may be written as:

**Step 1 (Base-16)**

\[
f(j_o, k_1, k_0) = \frac{1}{16} \sum_{k=0}^{15} m_{k+2^{12}+k_1+2^{16}+k_2} \exp \left[ \frac{i j_0 (\pi/8) k_2} {16} \right] \quad (12)
\]

**Step 2 (Base-8)**

\[
f(j_o, j_1, k_0) = \frac{1}{8} \sum_{k=0}^{2^7} \left[ f(j_o, k_1, k_0) \exp \left[ \frac{i j_1 (\pi/4) k_2} {16} \right] \right] \]

**Step 3 (Base-16)**

\[
f(j_o, j_1, j_2) = \frac{1}{16} \sum_{k=0}^{2^{15}} \left[ f(j_o, j_1, k_0) \exp \left[ \frac{i j_1 (\pi/1024) k_0} {2^{16}} \right] \right] \exp \left[ \frac{i j_2 (\pi/8) k_0} {2^{16}} \right] \quad (13)
\]

These equations are implemented in a base-16 vector processor that provides vector rotation in multiples of \( \pi/8 \) (and hence \( \pi/4 \)). This corresponds to the factored out exponent. The operations inside the big brackets in steps 2 and 3 are one operation to each "f" so that they may be performed as information transits from...
In Step 2, the rotations are in units of $\pi/64$. There are 128 possible entries in a table. The product of $j\theta_1$ can range from 0 to 105. If a 7-bit number is used for representing this product, the first two digits may be used to indicate the quadrant so that the table is reduced to 32 places. In Step 3, there are two exponents in the bracket. The multiples of $\pi/64$ can be treated the same as before except the multiplier is now $j\theta_2$. The other exponent involves multiples of $\pi/1024$. Since there are 226 different products of $j\theta_2$, a table of 226 places will be sufficient. The two vector multiplications may be performed in cascade.

**Vector multiplier**

The vector multiplier receives memory words at a rate of one each 1.71 microseconds and in the "phase rotation" mode (see Figure 9) receives the sine and cosine of the rotation angle at a similar rate from the trigonometric function table; in the "filter magnitude" mode (see Figure 10) used to obtain the squared magnitude no other input is required.

In both modes of operation, each 32-bit word coming from the memory is comprised of a 16-bit datum for each of the two subbands. The 16-bit quantity has an 8-bit real portion and an 8-bit imaginary portion. The real and imaginary parts of the $A$ and $B$ subband data are loaded (Figures 9 and 10) into the four 8-bit registers $A_{real}$, $B_{real}$, $A_{imag}$ and $B_{imag}$. In the phase rotation mode (Figure 9) operation occurs in four sequences (one after another) to give (in the $\alpha$ and $\beta$ registers):

1. $\alpha_{real} = \text{Real } [(A_{real} + iA_{imag}) \cos \theta + i \sin \theta]$
   
   $= A_{real} \cos \theta - A_{imag} \sin \theta$

2. $\alpha_{imag} = \text{Imag } [(A_{real} + iA_{imag}) \cos \theta + i \sin \theta]$
   
   $= A_{real} \sin \theta + A_{imag} \cos \theta$

3. $\beta_{real} = B_{real} \cos \theta - B_{imag} \sin \theta$

4. $\beta_{imag} = B_{real} \sin \theta + B_{imag} \cos \theta$

Within each sequence, the two required multiplications occur simultaneously in the two (scalar) multipliers ($M_{pier}$) and the products are immediately added in the adder/subtractor. A complete memory cycle, less only setting time for the $A$ and $B$ registers, is available for four sequences. An allowance of 400 nanoseconds for each sequence seems realistic.
**TABLE II—Operation Sequences Within Filter Arithmetic Processor**

<table>
<thead>
<tr>
<th>First Step</th>
<th>Base-4</th>
<th>Base-8</th>
<th>Base-16</th>
</tr>
</thead>
<tbody>
<tr>
<td>$R_0 = 0_0 + 0_1 + 0_2 + 0_3$</td>
<td>$R_0 = 0_0 + 0_1 + 0_2$</td>
<td>$R_0 = 0_0 + 0_1 + 0_2 + 0_3$</td>
<td></td>
</tr>
<tr>
<td>$R_1 = 0_0 - 0_1 - 0_2 + 0_3$</td>
<td>$R_1 = 0_0 - 0_1 - 0_2$</td>
<td>$R_1 = 0_0 - 0_1 - 0_2 + 0_3$</td>
<td></td>
</tr>
<tr>
<td>$R_2 = 0_0 + 0_1 - 0_2 - 0_3$</td>
<td>$R_2 = 0_0 + 0_1 - 0_2$</td>
<td>$R_2 = 0_0 + 0_1 - 0_2 - 0_3$</td>
<td></td>
</tr>
<tr>
<td>$R_3 = 0_0 - 0_1 + 0_2 + 0_3$</td>
<td>$R_3 = 0_0 - 0_1 + 0_2$</td>
<td>$R_3 = 0_0 - 0_1 + 0_2 + 0_3$</td>
<td></td>
</tr>
<tr>
<td>$R_4 = 0_0 + 0_1 + 0_2 + 0_3$</td>
<td>$R_4 = 0_0 + 0_1 + 0_2$</td>
<td>$R_4 = 0_0 + 0_1 + 0_2 + 0_3$</td>
<td></td>
</tr>
<tr>
<td>$R_5 = 0_0 - 0_1 - 0_2 - 0_3$</td>
<td>$R_5 = 0_0 - 0_1 - 0_2$</td>
<td>$R_5 = 0_0 - 0_1 - 0_2 - 0_3$</td>
<td></td>
</tr>
<tr>
<td>$R_6 = 0_0 + 0_1 - 0_2 + 0_3$</td>
<td>$R_6 = 0_0 + 0_1 - 0_2$</td>
<td>$R_6 = 0_0 + 0_1 - 0_2 + 0_3$</td>
<td></td>
</tr>
<tr>
<td>$R_7 = 0_0 - 0_1 + 0_2 + 0_3$</td>
<td>$R_7 = 0_0 - 0_1 + 0_2$</td>
<td>$R_7 = 0_0 - 0_1 + 0_2 + 0_3$</td>
<td></td>
</tr>
<tr>
<td>$R_8 = 0_0 + 0_1 + 0_2 + 0_3$</td>
<td>$R_8 = 0_0 + 0_1 + 0_2$</td>
<td>$R_8 = 0_0 + 0_1 + 0_2 + 0_3$</td>
<td></td>
</tr>
<tr>
<td>$R_9 = 0_0 - 0_1 - 0_2 - 0_3$</td>
<td>$R_9 = 0_0 - 0_1 - 0_2$</td>
<td>$R_9 = 0_0 - 0_1 - 0_2 - 0_3$</td>
<td></td>
</tr>
<tr>
<td>$R_{10} = 0_0 + 0_1 - 0_2 + 0_3$</td>
<td>$R_{10} = 0_0 + 0_1 - 0_2$</td>
<td>$R_{10} = 0_0 + 0_1 - 0_2 + 0_3$</td>
<td></td>
</tr>
<tr>
<td>$R_{11} = 0_0 - 0_1 + 0_2 + 0_3$</td>
<td>$R_{11} = 0_0 - 0_1 + 0_2$</td>
<td>$R_{11} = 0_0 - 0_1 + 0_2 + 0_3$</td>
<td></td>
</tr>
<tr>
<td>$R_{12} = 0_0 + 0_1 + 0_2 + 0_3$</td>
<td>$R_{12} = 0_0 + 0_1 + 0_2$</td>
<td>$R_{12} = 0_0 + 0_1 + 0_2 + 0_3$</td>
<td></td>
</tr>
<tr>
<td>$R_{13} = 0_0 - 0_1 - 0_2 - 0_3$</td>
<td>$R_{13} = 0_0 - 0_1 - 0_2$</td>
<td>$R_{13} = 0_0 - 0_1 - 0_2 - 0_3$</td>
<td></td>
</tr>
<tr>
<td>$R_{14} = 0_0 + 0_1 - 0_2 + 0_3$</td>
<td>$R_{14} = 0_0 + 0_1 - 0_2$</td>
<td>$R_{14} = 0_0 + 0_1 - 0_2 + 0_3$</td>
<td></td>
</tr>
<tr>
<td>$R_{15} = 0_0 - 0_1 + 0_2 + 0_3$</td>
<td>$R_{15} = 0_0 - 0_1 + 0_2$</td>
<td>$R_{15} = 0_0 - 0_1 + 0_2 + 0_3$</td>
<td></td>
</tr>
</tbody>
</table>

**NOTES:**
1. $R = \text{Result}; \ 0 = \text{Operand}$
2. $e^{i\theta} = \cos \theta + i \sin \theta$
3. $e^{i(\pi/2 - \theta)} = \sin \theta + i \cos \theta$

Operation of the vector multiplier in the filter magnitude mode is essentially similar (Figure 10) except that only two sequences are performed. These result in the following $a$ and $b$ register contents:

1. $a_{real} = A_{real}^2 + A_{imag}^2$
2. $b_{real} = B_{real}^2 + B_{imag}^2$

**Vector processor**

Within the vector processor, fast Fourier processing employs either a base-4 or base-2 procedure. For an (externally viewed) overall base of 16, two base-4 steps occur and for a base-8 procedure, a base-4 and base-2 step are combined.

Conceptually and equipment-wise, the vector processor (Figure 11) can be viewed as two 16-register sets of 16-bit registers $R_{A0}$ to $R_{A15}$ and $R_{B0}$ to $R_{B15}$. Each set stores the 16 data associated with the processing of a one-subband group of 16 operands; thus the 16 16-bit registers can contain one $G$-16 (see Summary System Description) or two $G$-8's or four $G$-4's. (Only the $G$-16 case is described.)

As a new $G$-16 enters the vector processor from the vector multiplier, it is appropriately loaded into the 32
TABLE III—Rewritten Operation Sequences Within Filter Arithmetic Processor

<table>
<thead>
<tr>
<th>First Step</th>
<th>Base-4</th>
<th>Base-8</th>
<th>Base-16</th>
</tr>
</thead>
<tbody>
<tr>
<td>$b_0 = (0 + 0 + 0)^+ + (0 + 0 + 0)^-$</td>
<td>$b_0 = (0 + 0 + 0)^+ + (0 + 0 + 0)^-$</td>
<td>$b_0 = (0 + 0 + 0)^+ + (0 + 0 + 0)^-$</td>
<td></td>
</tr>
<tr>
<td>$b_1 = (0 + 0 + 0)^+ + (0 + 0 + 0)^-$</td>
<td>$b_1 = (0 + 0 + 0)^+ + (0 + 0 + 0)^-$</td>
<td>$b_1 = (0 + 0 + 0)^+ + (0 + 0 + 0)^-$</td>
<td></td>
</tr>
<tr>
<td>$b_2 = (0 + 0 + 0)^+ + (0 + 0 + 0)^-$</td>
<td>$b_2 = (0 + 0 + 0)^+ + (0 + 0 + 0)^-$</td>
<td>$b_2 = (0 + 0 + 0)^+ + (0 + 0 + 0)^-$</td>
<td></td>
</tr>
<tr>
<td>$b_3 = (0 + 0 + 0)^+ + (0 + 0 + 0)^-$</td>
<td>$b_3 = (0 + 0 + 0)^+ + (0 + 0 + 0)^-$</td>
<td>$b_3 = (0 + 0 + 0)^+ + (0 + 0 + 0)^-$</td>
<td></td>
</tr>
<tr>
<td>$b_4 = (0 + 0 + 0)^+ + (0 + 0 + 0)^-$</td>
<td>$b_4 = (0 + 0 + 0)^+ + (0 + 0 + 0)^-$</td>
<td>$b_4 = (0 + 0 + 0)^+ + (0 + 0 + 0)^-$</td>
<td></td>
</tr>
<tr>
<td>$b_5 = (0 + 0 + 0)^+ + (0 + 0 + 0)^-$</td>
<td>$b_5 = (0 + 0 + 0)^+ + (0 + 0 + 0)^-$</td>
<td>$b_5 = (0 + 0 + 0)^+ + (0 + 0 + 0)^-$</td>
<td></td>
</tr>
<tr>
<td>$b_6 = (0 + 0 + 0)^+ + (0 + 0 + 0)^-$</td>
<td>$b_6 = (0 + 0 + 0)^+ + (0 + 0 + 0)^-$</td>
<td>$b_6 = (0 + 0 + 0)^+ + (0 + 0 + 0)^-$</td>
<td></td>
</tr>
<tr>
<td>$b_7 = (0 + 0 + 0)^+ + (0 + 0 + 0)^-$</td>
<td>$b_7 = (0 + 0 + 0)^+ + (0 + 0 + 0)^-$</td>
<td>$b_7 = (0 + 0 + 0)^+ + (0 + 0 + 0)^-$</td>
<td></td>
</tr>
</tbody>
</table>

| Second Step | No second step | |
|-------------|---------------| |

**Notes:**

1. $b = Result; O = Operand
2. $e^{i\theta} = \cos \theta + i \sin \theta
3. $e^{i(\theta/2)} = \sin \theta + i \cos \theta
4. $\cos \theta = \sin \theta = 1/\sqrt{2}
5. $0 = \text{Real} + 10^{\text{Real}}$ and $1 + i = 0$
6. $\text{Real} = 10^{\text{Real}} + 10^{\text{Real}}$  

* The switches are a part of the control and are considered separately only to facilitate the description.
** When six memory cycles are available for processing, the sixth cycle is a "dead period."

Registers by the loading and set-transfer switches. At the same time the old $G_{16}$ (the result of the previous operation) is transferred out of the registers and into the random access memory where it replaces the $G_{16}$ being read in. Transfer from the memory occurs under control of the output sequence switches. Thus, during a 16 memory cycle burst, the processed (old) $G_{16}$ in the vector processor is exchanged for a (new) $G_{16}$ from the memory; as it passes through the vector multiplier, the data of the new $G_{16}$ are multiplied by the proper unit vectors.

During the subsequent five memory cycles, the actual processing of the $G_{16}$ occurs. In this process a sequence of four-register-sets is selected by the set selection switches, transferred to the filter arithmetic processor (FAP) where they are subjected to a base-4 fast Fourier procedure and then returned to the four registers from which they came. In order to process the $G_{16}$, a total of four sets must "go through" the FAP per base-4 step per subband. This is illustrated by the base-16 column at the right of Table II. In that table, it is seen that the first base-4 step uses (for each subband) operands $O_0$, $O_1$, $O_2$ and $O_3$ to give results $R_0$, $R_1$, $R_2$ and $R_3$, and uses operators $O_4$, $O_5$, $O_6$ and $O_7$ to give results $R_4$, $R_5$, $R_6$ and $R_7$, and so on. The second step is described by the lower half of the base-16 column of Table II.

Table II presents the fast Fourier arithmetic in its most conventional form. The actual equations chosen to be mechanized are shown in Table III in which the equations have been rewritten to reduce the number of multiplier circuits required. The FAP itself is shown in Figure 12.

As demonstrated by Table III, the operations performed on the (input vector) operands and on combinations of the operands consist of multiplication by $\pm 1$ and $\pm i$ ($= \sqrt{-1}$), addition and subtraction and multiplication by the sine of $\pi/4$ and/or the sine and cosine of $\pi/8$. All operations except multiplication by...
NOTE TO LOADING AND SET-TRANSFER SWITCHES

CONTROlLED ADDERS MULTIPLY EACH OPERAND BY ± 1 OR ±i BEFORE ADDING OR SUBTRACTING. BRACKETED ITEMS ARE AVAILABLE AT INPUT TO CONTROLLED ADDERS.

Figure 12—Filter arithmetic processor

The sines and the cosine occur in the "controlled adders" of Figure 12. In the controlled adders, pairs of vectors are added to or subtracted from each other either directly or after prior multiplication by ± 1 or ±i. The results of the addition or subtraction may also be multiplied by ± 1 or ±i.

Multiplication of the appropriate information by functions of π/4 and π/8 is performed in the six 8-bit (scalar) multipliers shown in Figure 12; these multipliers are similar to those used in the vector multiplier.

Since each word entering the vector processor contains one operand per subband (and there are two subbands) and since four sets of four operands are processed in the vector processor for each of the two base-4 steps which comprise a base-16 step, a total of 2 X 4 X 2 = 16 sets of four operands must be processed by the FAP in 16 substeps during each five memory cycle period of 8.57 microseconds. This gives time of 0.53 microsecond per substep. Note that the period of 0.53 microsecond is not the time from the insertion of operands (into the input set of controlled adders) until the availability of the results in the four output registers. (As long as the delay is not great enough to endanger system timing, the allowable time is a function of the overall vector processor design and system timing is not a problem because the first four sets of results for both subbands can be made available to the memory at the end of only 10 of the 16 substeps.) The significance of the 0.53 microsecond is that no procedure in the FAP may take a time approaching 0.53 microsecond; the longest procedure in the FAP is the 8-bit scalar multiplication and this requires only 0.3 microsecond. Thus, there are no special problems in the mechanization of the vector processor.

Figure 13—Sine/cosine table block diagram

Sine/cosine table

At a rate of one set per 1.71-microsecond memory cycle, the sine/cosine table must provide sines and cosines of the multiples of the angle 2π/2048; that is, of the angles θ(2π/2048) where θ is an integer between 0 and 2047. The angles are specified to the sine/cosine table by the j’s and k’s generated by the address sequence control. An internal function of the sine/cosine table is to compute (from the j’s and k’s) the value of θ which is to be used as the independent variable in entering the table. In Figure 13, which is the block diagram of the sine/cosine table, the value of θ is computed by the multiplier, Mpier. (This operation is trivial and is not described here.) Output of the multiplier is an 11-bit value of θ.

The value of θ is converted to a 9-bit positive number θP which is ultimately used to obtain the appropriate sine and cosine of an angle between 0 and π/4; that is, all sine and cosine values are computed for angles no greater than π/4 and the computed first-octant sine and cosine are used according to the trigonometric identities for complementary and supplementary angles and for angles which differ by π radians. These identities translate into the algorithms of Table IV which demonstrates how the angle θ is translated to the first octant angle θP or its two’s complement and used in computing the sine and cosine.

Computation of the sine and cosine of θP is performed using a five-point table and linear interpolation between the five points 0, π/16, π/8, 3π/16 and π/4. The inaccuracy of the interpolation is less than one part in 200 over the range of interest. Of greater import is the fact that the magnitude of the error is less than 2−8.
TABLE IV—Determination of First-Octant Angle Used in Sine/Cosine Computation and Related Rules

<table>
<thead>
<tr>
<th>Most Significant Bit</th>
<th>Least Significant Bit</th>
</tr>
</thead>
<tbody>
<tr>
<td>$\pi$</td>
<td>$\pi/2$</td>
</tr>
<tr>
<td>$\pi/2$</td>
<td>$\pi/4$</td>
</tr>
<tr>
<td>$\pi/4$</td>
<td>$\pi/8$</td>
</tr>
<tr>
<td>$\pi/8$</td>
<td>$\pi/16$</td>
</tr>
<tr>
<td>$\pi/16$</td>
<td>$\pi/32$</td>
</tr>
<tr>
<td>$\pi/32$</td>
<td>$\pi/64$</td>
</tr>
<tr>
<td>$\pi/64$</td>
<td>$\pi/128$</td>
</tr>
<tr>
<td>$\pi/128$</td>
<td>$\pi/256$</td>
</tr>
<tr>
<td>$\pi/256$</td>
<td>$\pi/512$</td>
</tr>
<tr>
<td>$\pi/512$</td>
<td>$\pi/1024$</td>
</tr>
</tbody>
</table>

Modify signs, possibly interchange table outputs

If

1. $\theta_r \leq 2^8$
2. $2^8 < \theta_r \leq 2^9$
3. $2^9 < \theta_r \leq (2^9 + 2^8)$
4. $(2^9 + 2^8) < \theta_r \leq 2^{10}$
5. $2^{10} < \theta_r \leq \theta_{\text{max}} = 2^{11} - 1$

Use $\theta_r$.
Use 2's complement of $\theta_r$. Interchange sine and cosine outputs.
Use $\theta_r$. Put - sign on cosine and then interchange sine and cosine.
Use 2's complement of $\theta_r$. Put - sign on sine.
Using all bits except $2^8$ do as above (1 to 4) and reverse all signs at output.

while the eight arithmetic bits employed represent sign and seven magnitude bits; thus the maximum error is less than one-half of the least significant bit.

The computational procedure employs a wired table of the sines and cosines of the selected five points and also supplies the slope of the sine and cosine at the first four points. (Since $\theta_r \leq \pi/4$, no slope is required beyond $\pi/4$.) Each slope is provided to one of the two multipliers which also receive the six least significant bits of $\theta_r$; these represent the difference between $\theta_r$ and the next lower argument of the tabulated values of sine and cosine. Thus, the products of the two multipliers represent the amounts to be added to the tabular value of the sine and subtracted from the tabular value of the cosine* to obtain the function of $\theta_r$. Addition and subtraction occur in the adder and subtractor as shown in Figure 13 and selection of sine and cosine and assignment of the proper signs are performed according to Table IV.

SUMMARY

A fast Fourier digital processor designed for the real-time filtering of radar signals has been described. This processor design can perform direct and inverse transforms and is also applicable to communications systems, correlation and pattern matching, convolution and other processes using the discrete Fourier transform. The design could also be used for a stand-alone Fourier transform system or for a system to be used in conjunction with a general purpose computer.

In the real-time radar application, the processor replaces 512 (analog) filters of 143 Hz bandwidth or 1024 filters of 72 Hz bandwidth.

The processor is employed on a full time basis to perform the Fourier transform. Thus, it can be designed to do this task more efficiently and at lower cost than would a general purpose computer. Design concepts which make this possible are:

1. The use of a fast Fourier base of 16 (rather than two) in order to reduce the memory and logic circuit requirements,
2. The use of a timing scheme (burst processing) which maximizes the uninterrupted periods allowed for processing,
3. Simple flow of information through and within the processor,
4. Separation of functions to be performed from one another and performing these functions in special purpose functional equipment.

BIBLIOGRAPHY

1 M I Skonik
*Introduction to radar systems*
pp 113-126; 132-137; 151-162; 408-418
2 W W MacQuire
*Application of pulsed doppler to airborne radar systems*
Proc of the National Conference on Aeronautical Electronics pp 291-295 Dayton Ohio 1958
3 G D Bergland
*A guided tour of the fast fourier transform*
IEEE Spectrum Vol 6 No 7 pp 41–52 July 1969

*The slope of the cosine curve is negative.
APPENDIX

Fast Fourier transform digital processing and digital filters

During the last two decades, the problem of extracting or filtering out a small signal from considerably more powerful noise has been given much practical and theoretical attention for real-time processes such as radar and sonar signal processing and for delayed-time processes such as telemetry data reduction. In general, the filters and the real-time processors have been limited in complexity and conceptual sophistication by the requirement that they be realizable with acceptable quantities of hardware.

The delayed-time processes could often use very large computers, but were limited by the long, slow iterative procedures. As a result, significant effort was devoted to computational procedures. Probably the most useful of these has been the application of Fourier methods. However, as larger and larger complexes of data were attacked, these methods became more and more unwieldy because the amount of computation rose approximately as the square of the amount of data involved. Thus, if data were collected at fixed intervals over one "frame" (during which time \( N \) samples are collected), the computational effort involved in processing would be proportional to the square of the duration of the frame or for \( N \) measurements would vary directly with \( N^2 \). This is true since Fourier spectrum analysis, in digital form, with frequency resolution matched to the length of time the signal is under observation, is of the form

\[
F_i = \sum_{k=0}^{N-1} A_k e^{j2\pi ik/N}
\]  

(A-1)

where

\[
A_k = \text{the } k\text{th sampled values of the signal, } k = 0, 1, 2, \ldots, N - 1
\]

\[
i = \sqrt{-1} \text{ and both } A'\text{s and } F'\text{s are complex (in-phase and quadrature sampled)}
\]

Brute force calculations required \( N^2 \) operations since the \( NF_i'\)s are to be evaluated and each is the sum of \( N \) products. In high PRF processing, \( N \) is typically 1,000 or larger so that the amount of arithmetic by this method is prohibitively large for real-time operation.

An algorithm developed by J. W. Cooley and J. W. Tukey reduces the computational load to \( N \log_2 N \) where \( B \) is the base (typically a power of 2 such as 2, 4, 8 or 16) to which the logarithm of \( N \) was taken and also represents the number of data from the full set of \( N \) which are processed in each substep of the procedure.

From the collection of the Computer History Museum (www.computerhistory.org)
Table A-1 provides a comparison of the time for calculating a Fourier transform by a popular conventional method and by the faster Cooley-Tukey method. These are proportional to \( N^2 \) and \( N \log_2 N \), respectively. Both methods were programmed in basic FORTRAN and run on the IBM 7094.*

This radical improvement makes Fourier methods directly applicable to real-time all-digital processing of sensor information with presently available computer memories and integrated circuits. The Cooley-Tukey method will be described step-by-step, in terms of the procedures involved but without any attempt at proving their equivalence to the direct method. For proof of the equivalence of the two methods, the reader is referred to the references.

For the radar application, the important filter characteristic is the so-called amplitude frequency response, or simply frequency response, \( H(j\omega) \) which is independent of the phase of the signal at the input of the filters. This may be analyzed by either Fourier transforms or Fourier series.

The calculation of the Fourier series of Equation (A-1) can be performed digitally by calculating the complex discrete Fourier series in the form:

\[
F = \sum_{k=0}^{K-1} W_k A_k \quad \text{(A-2)}
\]

where

\( A_k = \) the \( k \)th input sample pair (the real part is the in-phase component and the imaginary part is the quadrature component)

\( W_k = \) the \( k \)th complex member of the weighting sequence

\( F = \) the complex filter response which was determined by the weighting sequence \( W_0, W_1, \ldots, W_{K-1} \)

Here, the magnitudes of the sequence of \( W \)'s determine the filter shape while the phase angles of the \( W \)'s determine the frequencies of maximum filter response. Thus, we may form a bank of \( N \) filters using equation (A-2) and the arithmetic requirement can be greatly reduced by the use of the fast transform algorithm if the following reasonable restrictions are accepted.

1. All filters in the bank are the same shape.
2. Filter response frequencies for different filters are uniformly spaced with a spacing of \( \frac{\text{sampling frequency}}{\text{any power of 2}} \)

3. The filters required should cover a total bandwidth which is a significant part of the maximum unambiguous bandwidth (the sampling frequency).

Restriction (1) means that the amplitude part of the weighting sequence is the same for each filter and allows the weights to be the roots of a unit vector with a zero phase angle. Then the amplitude sequence need be applied only once to the incoming data sequence for the entire filter bank. The weights that distinguish different filters in the bank all have the same magnitude.

If we assume that a table is available to supply the real and imaginary parts of \( e^{-j2\pi n/N} \), the major arithmetic (to evaluate Equation (A-2) in order to synthesize a filter bank) is a series of complex multiplications and additions. Using the base-two (or \( B = 2 \)) version of the fast Fourier method, one of each two input measurements (vector voltages) is rotated by an appropriate amount and added to and also subtracted from the other of a pair of vector measurements.

* From reference 12. Note that the IBM 7094 computer is slow compared to present generation airborne computers.

---

**TABLE A-1—Calculating Time**

<table>
<thead>
<tr>
<th>Number of Points</th>
<th>Conventional Method</th>
<th>Fast Method</th>
</tr>
</thead>
<tbody>
<tr>
<td>512</td>
<td>0.17</td>
<td>0.0</td>
</tr>
<tr>
<td>1,024</td>
<td>0.67</td>
<td>0.0</td>
</tr>
<tr>
<td>2,048</td>
<td>2.70</td>
<td>0.01</td>
</tr>
<tr>
<td>4,096</td>
<td>0.03</td>
<td></td>
</tr>
<tr>
<td>8,192</td>
<td>0.07</td>
<td></td>
</tr>
<tr>
<td>16,384</td>
<td>0.16</td>
<td></td>
</tr>
</tbody>
</table>

**TABLE A-2—First Step of Fast Fourier or Cooley-Tukey Process**

(16 samples per frame: \( B = \) Base \( = 2 \), \( S = \) Samples, \( T = \) Result of first step)

\[
\begin{align*}
S_0 + S_6 &= T_1 \\
S_1 + S_5 &= T_2 \\
S_2 + S_4 &= T_3 \\
S_3 + S_3 &= T_4 \\
S_0 + S_8 &= T_5 \\
S_1 + S_7 &= T_6 \\
S_2 + S_6 &= T_7 \\
S_3 + S_5 &= T_8 \\
S_4 - S_0 &= T_9 \\
S_5 - S_9 &= T_{10} \\
S_6 - S_7 &= T_{11} \\
S_7 - S_6 &= T_{12} \\
S_8 - S_{10} &= T_{13} \\
S_9 - S_{11} &= T_{14} \\
S_{10} - S_{10} &= T_{15} \\
S_{11} - S_{11} &= T_{16}
\end{align*}
\]
The procedures for the second, third, and fourth (last) steps are shown in Tables A-3, A-4, and A-5. It should be noted that for the case shown, there are four steps. This is true because using the base, B of 2, there are \( \log_2 16 \) = 4 steps as was previously described. And each step required \( N/B = N/2 \) complex multiplications and \( N/B = N/2 \) complex additions as was described.

The example given here is perhaps trivial because only 16 samples were used for each data frame and because the base used was only 2. However, if the same 16 samples were used but the base were raised to 4, the number of steps would reduce from 4 to 2 with an overall reduction ratio of 16 log 16:16 log 16 or 2:1. In the case of a base-4 approach rotations in the first step would be by multiples of 2\( \pi/4 \) and the first, fifth, ninth, and thirteenth sample would be summed after being rotated, etc. In the second and last step, rotations would be by multiples of 2\( \pi/16 \) and the first, second, third, and fourth result of the first step would be summed after being rotated, etc. Thus, the extension of the simple-16 sample base-2 example to the equally simple 16-sample base-4 example shows the general method of the fast Fourier method as well as the great reduction in processing compared to the conventional method.

### Table A-3—Second Step of Fast Fourier or Cooley-Tukey Process

(16 samples per frame: \( B = \text{Base} = 2, T = \text{Result of first step,} \ U = \text{Result of second step} \)

| \( T_1 + T_5 \) | \( T_2 + T_4 \) | \( T_3 + T_7 \) | \( e^{i(\pi/2)} = 1 + \sqrt{2} \) |
| \( T_4 + T_8 \) | \( e^{i(\pi/4)} = 0 + i\sqrt{2} \) |
| \( T_7 + T_{11} \) | \( e^{i(3\pi/4)} = -1 + \sqrt{2} \) |

\[ \sqrt{2} \text{ complex additions as was described.} \]

### Table A-4—Third Step of Fast Fourier or Cooley-Tukey Process

(16 samples per frame: \( B = \text{Base} = 2, U = \text{Result of second step,} \ V = \text{Result of third step} \)

| \( U_1 + U_5 \) | \( V_1 \) | \( e^{\theta} = 1 + i\theta \) |
| \( U_2 + U_4 \) | \( V_2 \) |

\[ \sqrt{2} \text{ complex multiplications and \( N/B = N/2 \) complex additions.} \]
TABLE A-5—Last (Fourth) Step of Fast Fourier or Cooley-Tukey Process

(16 samples per frame  \( B = \) Base = 2,  \( U = \) Result of third step,  \( F = \) Filtered outputs = Result of fourth step)

\[
\begin{align*}
V_1 + V_2 &= X_1 = F_6 & e^{i \theta} &= 1 + \text{i}0 = 1 \\
V_1 - V_2 &= X_2 = F_5 & e^{i \theta / 2} &= 0.924 + \text{i}0.383 \\
V_3 + V_4(i) &= X_4 = F_5 & e^{i \theta / 2} &= 0.707 + \text{i}0.707 \\
V_3 - V_4(i) &= X_4 = F_{12} & e^{i \theta / 2} &= 0.383 + \text{i}0.924 \\
V_5 + V_6(0.707 + \text{i}0.707) &= X_5 = F_2 & e^{i \theta / 4} &= 0 + \text{i} = \text{i} \\
V_1 - V_6(0.707 + \text{i}0.707) &= X_6 = F_{10} & e^{i \theta / 4} &= -0.383 + \text{i}0.984 \\
V_7 + V_8(-0.707 + \text{i}0.707) &= X_7 = F_4 & e^{i \theta / 4} &= -0.707 + \text{i}0.707 \\
V_7 - V_8(-0.707 + \text{i}0.707) &= X_8 = F_{14} & e^{i \theta / 4} &= -0.924 + \text{i}0.383 \\
V_9 + V_{10}(0.924 + \text{i}0.383) &= X_9 = F_1 & e^{i \theta / 4} &= -1 + \text{i}0 = -1 \\
V_9 - V_{10}(0.924 + \text{i}0.383) &= X_{10} = F_3 & e^{i \theta / 4} &= -0.924 - \text{i}0.383 \\
V_{11} + V_{12}(-0.383 + \text{i}0.924) &= X_{11} = F_9 & e^{i \theta / 8} &= -0.707 - \text{i}0.707 \\
V_{11} - V_{12}(-0.383 + \text{i}0.924) &= X_{12} = F_{12} & e^{i \theta / 8} &= -0.383 - \text{i}0.924 \\
V_{13} + V_{14}(0.383 + \text{i}0.924) &= X_{13} = F_2 & e^{i \theta / 8} &= 0 - \text{i} = -\text{i} \\
V_{13} - V_{14}(0.383 + \text{i}0.924) &= X_{14} = F_{11} & e^{i \theta / 8} &= 0.383 - \text{i}0.984 \\
V_{15} + V_{16}(-0.924 + \text{i}0.383) &= X_{15} = F_7 & e^{i \theta / 8} &= 0.707 - \text{i}0.707 \\
V_{13} - V_{16}(-0.924 + \text{i}0.383) &= X_{16} = F_{15} & e^{i \theta / 8} &= 0.924 - \text{i}0.383
\end{align*}
\]