JPL 2^{16} CHANNEL 20 MHz BANDWIDTH DIGITAL SPECTRUM ANALYZER

G. A. Morris, Jr., and H. C. Wilck
Communications Systems Research Section

Abstract

A 65,536 (2^{16}) channel, 20 MHz bandwidth, digital spectrum analyzer was constructed at the Jet Propulsion Laboratory. The design, fabrication, and maintenance philosophy of the modular, pipelined, Fast Fourier Transform (FFT) hardware are described. The spectrum analyzer will be used to examine the region from 1.4 GHz to 26 GHz for Radio Frequency Interference (RFI) which may be harmful to present and future tracking missions of the Deep Space Network. The design will have application to the Search for Extra-terrestrial Intelligence (SETI) signals and radio science phenomena.

I. INTRODUCTION

A 65,536 channel digital spectrum analyzer with 20 MHz of bandwidth was built at the Jet Propulsion Laboratory. The purpose of the spectrum analyzer is to detect and identify radio frequency interference which may be harmful to present and future spacecraft tracking missions of the Deep Space Network.

The block diagram of the spectrum analyzer is shown in Fig. 1. The RF system consists of an antenna and a 150K system temperature S-band receiver with 300 MHz bandwidth IF output. The IF output is fed to two complex mixers followed by analog filters and analog-to-digital (A/D) converters producing two separate complex channels of 10 MHz bandwidth each. A window function is applied to the digitized data. Then a pipelined decimation in frequency FFT is used to process these two 10 MHz channels simultaneously. The power spectrum is obtained by squaring the real and imaginary parts of the complex spectrum. The power spectrum is accumulated for a number of spectra, scaled, shuffled and input to a general purpose computer.

An extensive computer simulation was performed to determine the optimum hardware implementation to support the 60 dB dynamic range required. As a result of this simulation, the hardware is implemented using 8-bit A/D converters, 12-bit memories in the first four stages, 16-bit memories in the remaining 11 stages, and 16-bit, fixed point, hard scaled calculations in all stages.

II. COMPLEX MIXERS AND A/D CONVERTERS

Two 10 MHz wide complex channels are extracted from the 300 MHz total IF bandwidth by two complex mixers followed by low pass filters and A/D converters (Fig. 2). The input IF signal is fed to the two complex mixers together

with the output of two local oscillators. The frequencies of the local oscillators are computer controlled to allow positioning the 10 MHz channels anywhere within the 300 MHz IF bandwidth. Each complex mixer consists of two mixers whose local oscillators differ in phase by 90°. The in-phase (real) and quadrature (imaginary) outputs of the mixer each pass through a 5 MHz low pass antialiasing filter to an 8-bit 10 MHz sample rate A/D converter. These 10 MHz A/D converters are now readily available at low cost because of their use in digital conversion of television signals.

III. WINDOWING AND TEST LOGIC

The outputs from the A/D converters pass through multiplexers which allow substitution of digital test signals, and multipliers that apply a window function (Fig. 3). The output of the multipliers is fed to the FFT. The window coefficients are stored in a memory writable by the computer. This implementation allows arbitrary window functions. A sine/square wave generator produces digital test signals whose phase, frequency and amplitude are computer controlled. The saturation counters provide the computer with information for gain control.

IV. FFT BLOCK DIAGRAM

The FFT, shown in Fig. 4, consists of 15 pipelined stages (Ref. 1), each composed of a memory unit and a "butterfly" arithmetic unit. Only three types of modules are used in the entire FFT. The memory modules used for the first four stages have a maximum capacity of 2 x 8K complex words of 2 x 12 bits. The other 11 memory modules have a maximum capacity of 2 x 1K complex words of 2 x 16 bits. The same 16-bit arithmetic module type is used in all stages.

The memory modules are programmed with a dual-in-line header to provide the appropriate delay and trig coefficients for each stage in the pipeline.

The input to the FFT uses the "Biplex" method (Ref. 2) to simultaneously process two independent 10 MHz channels in the pipelined architecture FFT (Fig. 5). This method results in the full-time utilization of all memory and arithmetic elements.

V. MEMORY UNITS

A block diagram which is common to both types of memory units is shown in Fig. 6. A memory unit is composed of two delay memories and multiplexers which allow straight through or crossed input-output connection as required in the pipelined algorithm. The memory unit also contains the trig coefficient generator.

The differences between the two types of memory units concern the size and type of delay memory and type of trig generator.

The first four memory units, called 8K max on the FFT block diagram (Fig. 4), use random access memories (RAM) with a capacity of 16K complex words of 2 x 12 bits. They are implemented by multiplexing two sets of Intel 2147-3 integrated circuits to obtain 10 MHz bandwidth. The remaining 11 units, called 1K max, use RAM (Intel 2125AL) with a capacity of 2048 complex words of 2 x 16 bits.

VI. ARITHMETIC UNIT

The FFT radix 2 butterfly arithmetic unit is shown in Fig. 7. The complex adder/subtractor is placed in front of the complex multiplier in the decimation in frequency algorithm. The adder/subtractor operates on 16 bits of input data to deliver 17 bits of output. The output is scaled and rounded

to retain the 16 most significant bits. The adder is implemented with the 74S283 and the subtractor with the 74S381.

The complex multiplier is composed of four real multipliers followed by an adder (74S283) and subtractor (74S381) to combine the partial products. The real multipliers are implemented with the TRW MPY-16AJ. Two of these multipliers are connected in parallel and multiplexed to obtain a 10 MHz multiply rate. This is simple because of the input and tri-state output registers contained within the MPY-16AJ. The complete complex multiplier contains eight of the MPY-16AJ's. A fractional multiply is performed, and the 16 most significant bits are retained. The internal circuitry of the MPY-16AJ is used to round the result.

VII. FFT OUTPUT PROCESSING

Each real (R) and imaginary (I) output of the FFT is 16 bits wide. The power calculator performs the operation $R^2 + I^2$ to obtain a 31-bit power spectral line. N successive power spectra are accumulated into a 64K x 48-bit memory. This number N is chosen large enough to sufficiently reduce the power spectrum noise variance and to meet computer input-output (I/O) bandwidth limitations. The "shift and saturate" circuitry selects a 16-bit slice from the 48-bit wide accumulator output for transfer to the computer I/O buffer. If the value of a spectral line overflows the 16-bit slice, the full scale (maximum) 16-bit value is substituted. This "saturation" feature can be disabled. Bit reversed addressing during I/O buffer loading compensates for the index bit reversal inherent in the decimation in frequency FFT algorithm.

VIII. MAINTENANCE PHILOSOPHY

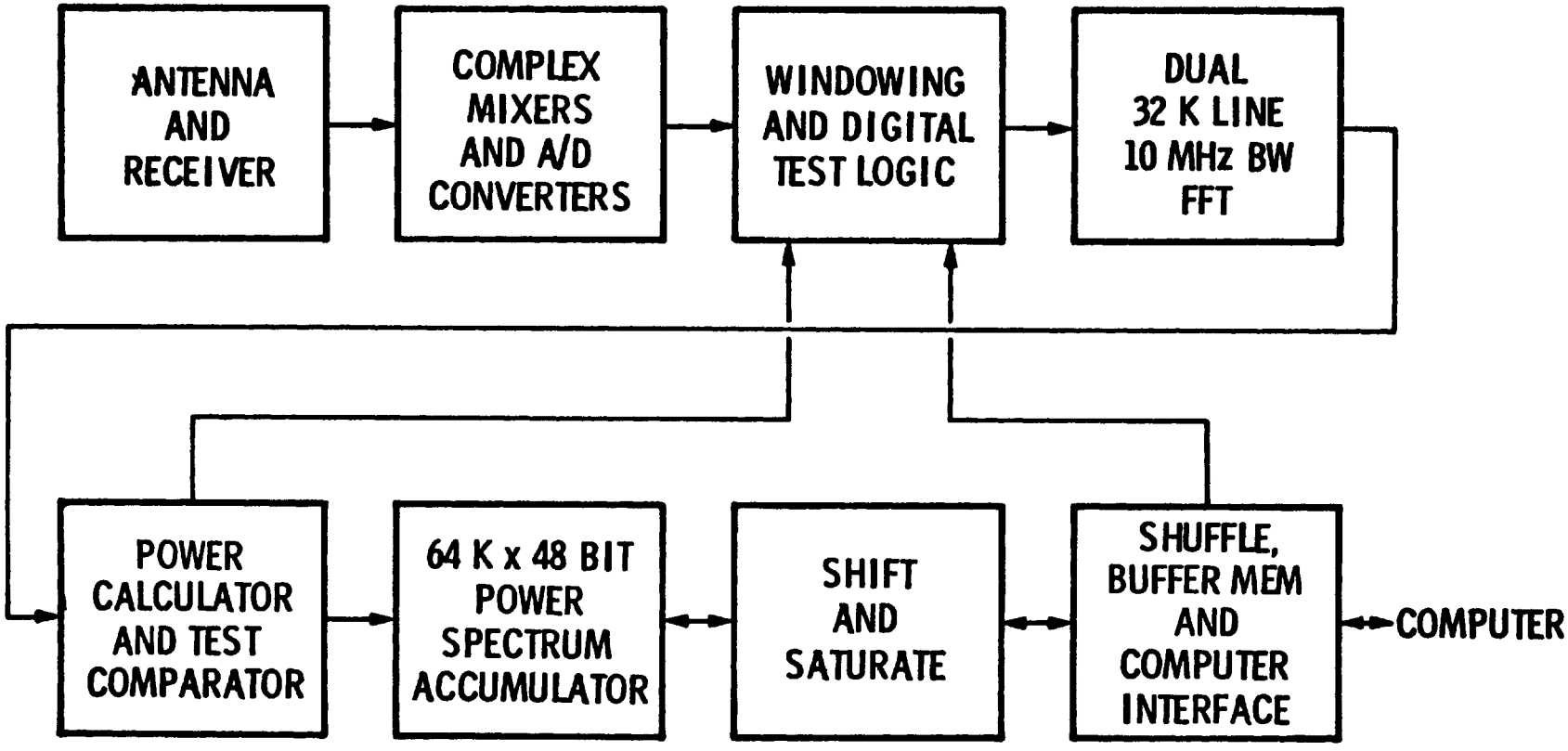
The spectrum analyzer was designed with ease of maintenance in mind, and special test hardware was incorporated.

A synthesizer is provided to inject a sine wave of controllable frequency and signal strength into the receiver for an overall test. There is a go-no-go self test for the FFT. The accumulators and buffers can be independently tested from the computer. In the case of FFT failure, digital test signals from a built in generator can be applied to the FFT input. Taps are provided at the output of each FFT stage, where intermediate results can be compared to expected results generated by computer emulation. There is also a software controlled test algorithm which, in most cases, allows FFT fault isolation to the circuit board level. Spares are provided for the three types of FFT boards. Testers were built to completely exercise the logic of these boards. The testers will be used for depot level maintenance.

REFERENCES

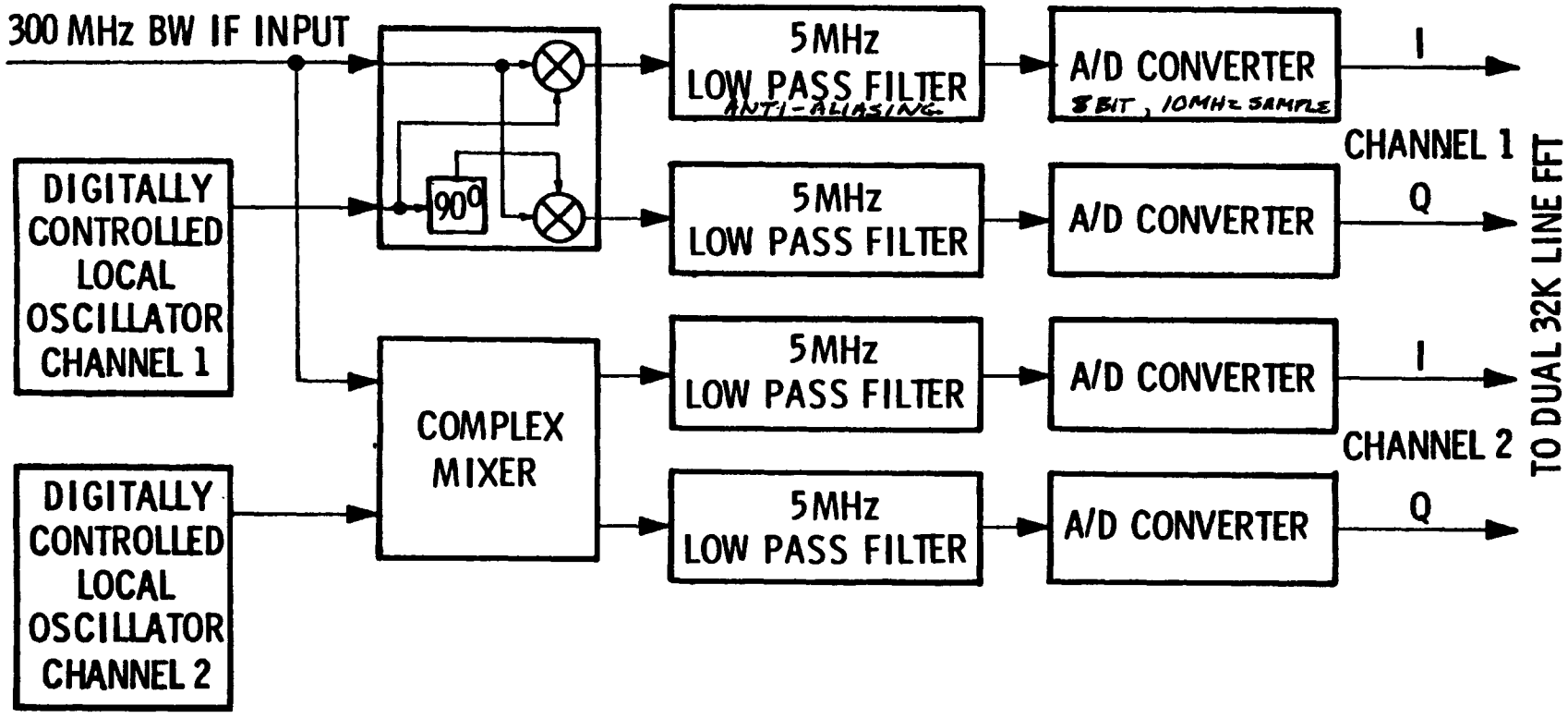
1. Rabiner, L. R., and Gold, B., Theory and Application of Digital Signal Processing, Prentice Hall, Inc., New Jersey, 1975, pp. 602-609.

2. Emerson, R. F., "Biplex Pipelined FFT," in The Deep Space Network Progress Report 42-34, Jet Propulsion Laboratory, Pasadena, California, 1976, pp. 54-59.



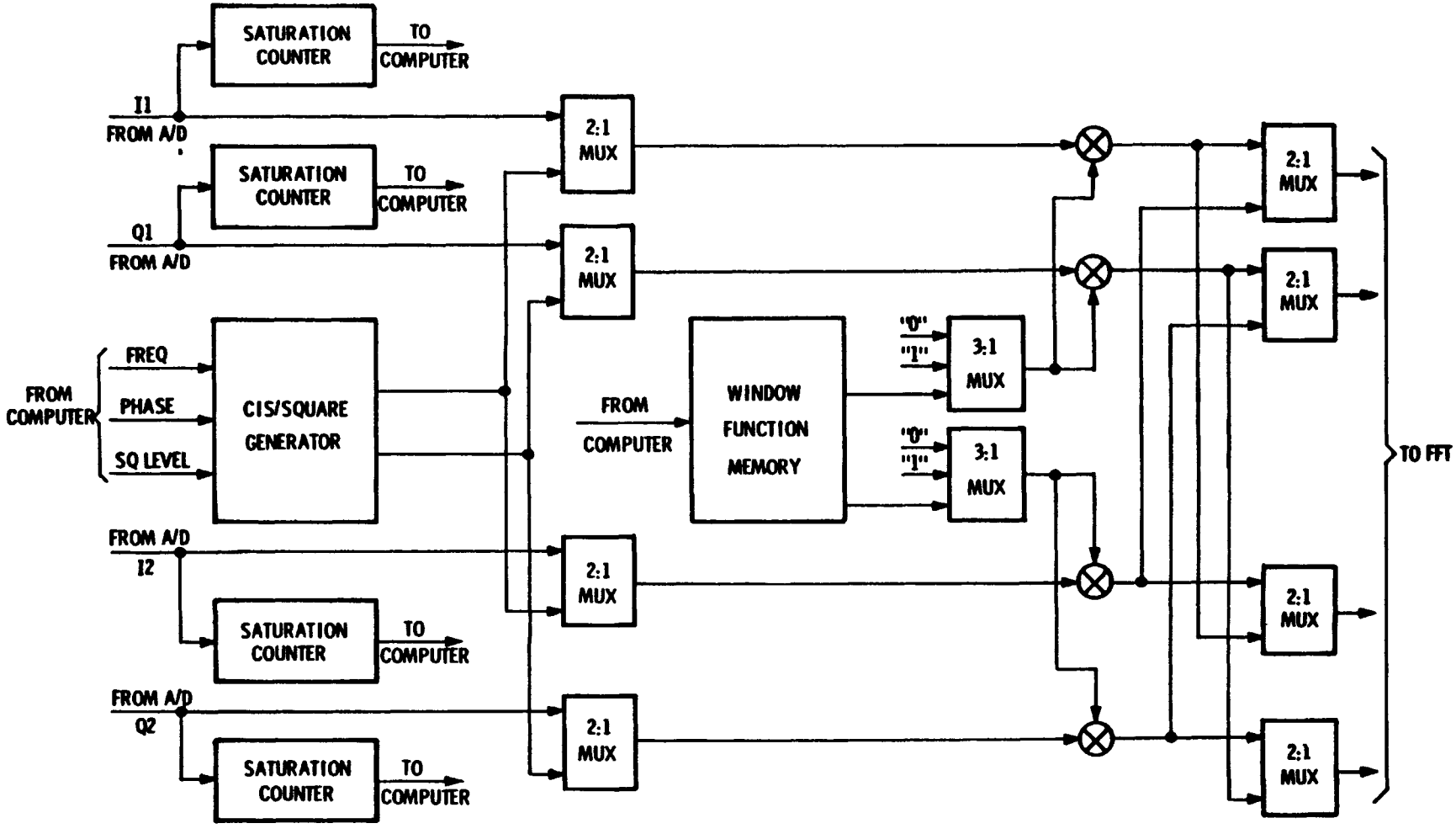
JPL 2^{16} LINE 20 MHz BANDWIDTH DIGITAL SPECTRUM ANALYZER

Fig. 1



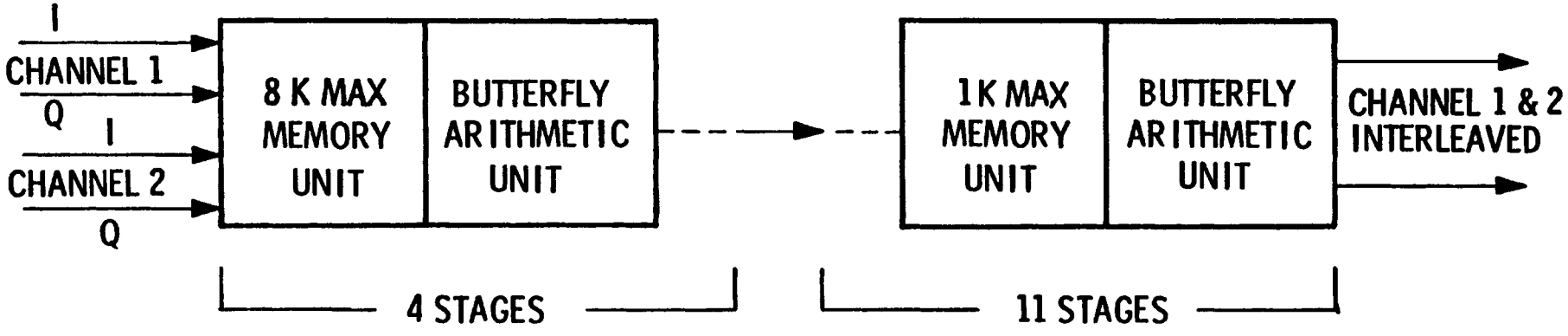
COMPLEX MIXERS AND A/D CONVERTERS

Fig. 2



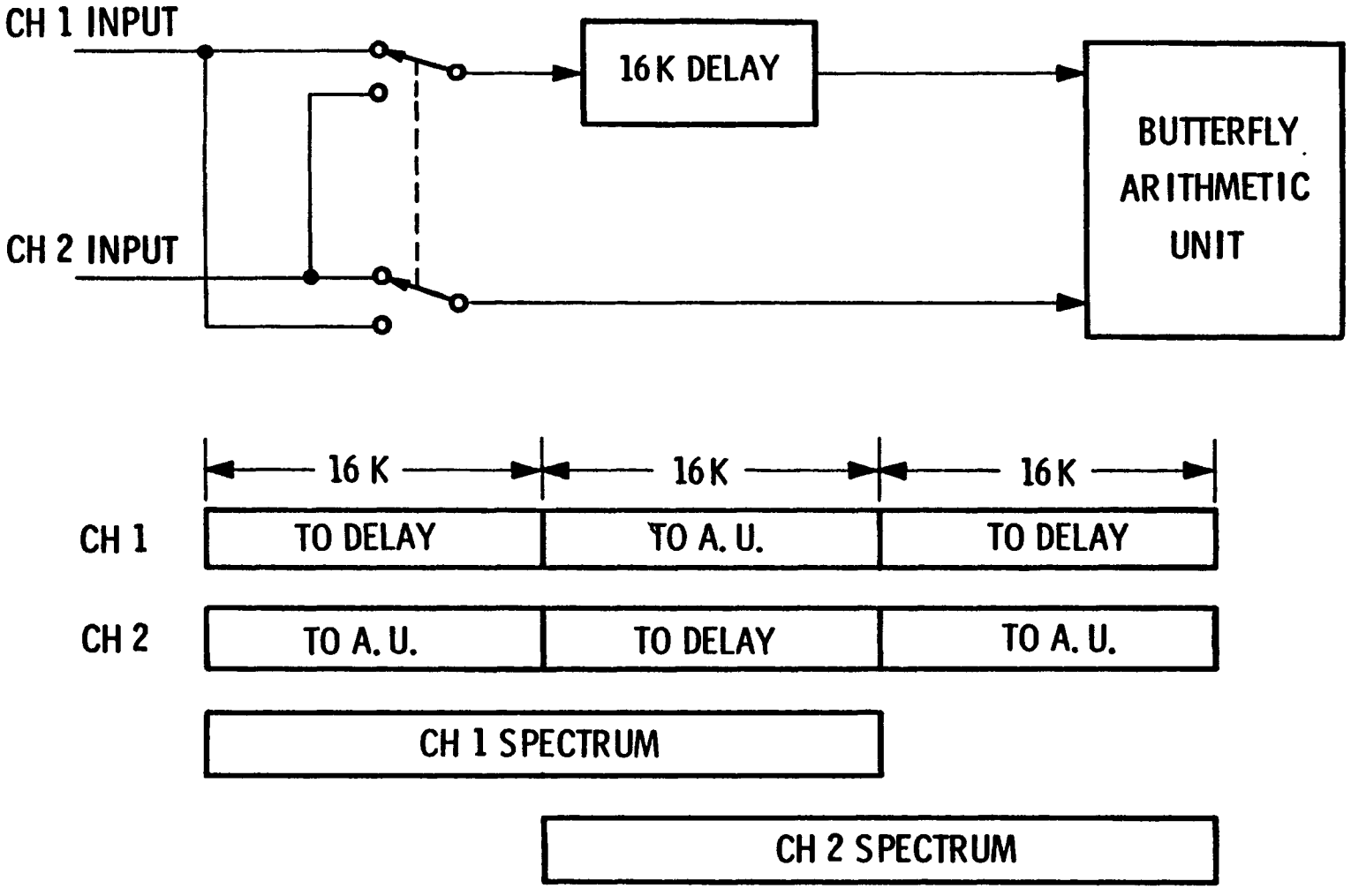
WINDOWING AND DIGITAL TEST LOGIC

Fig. 3



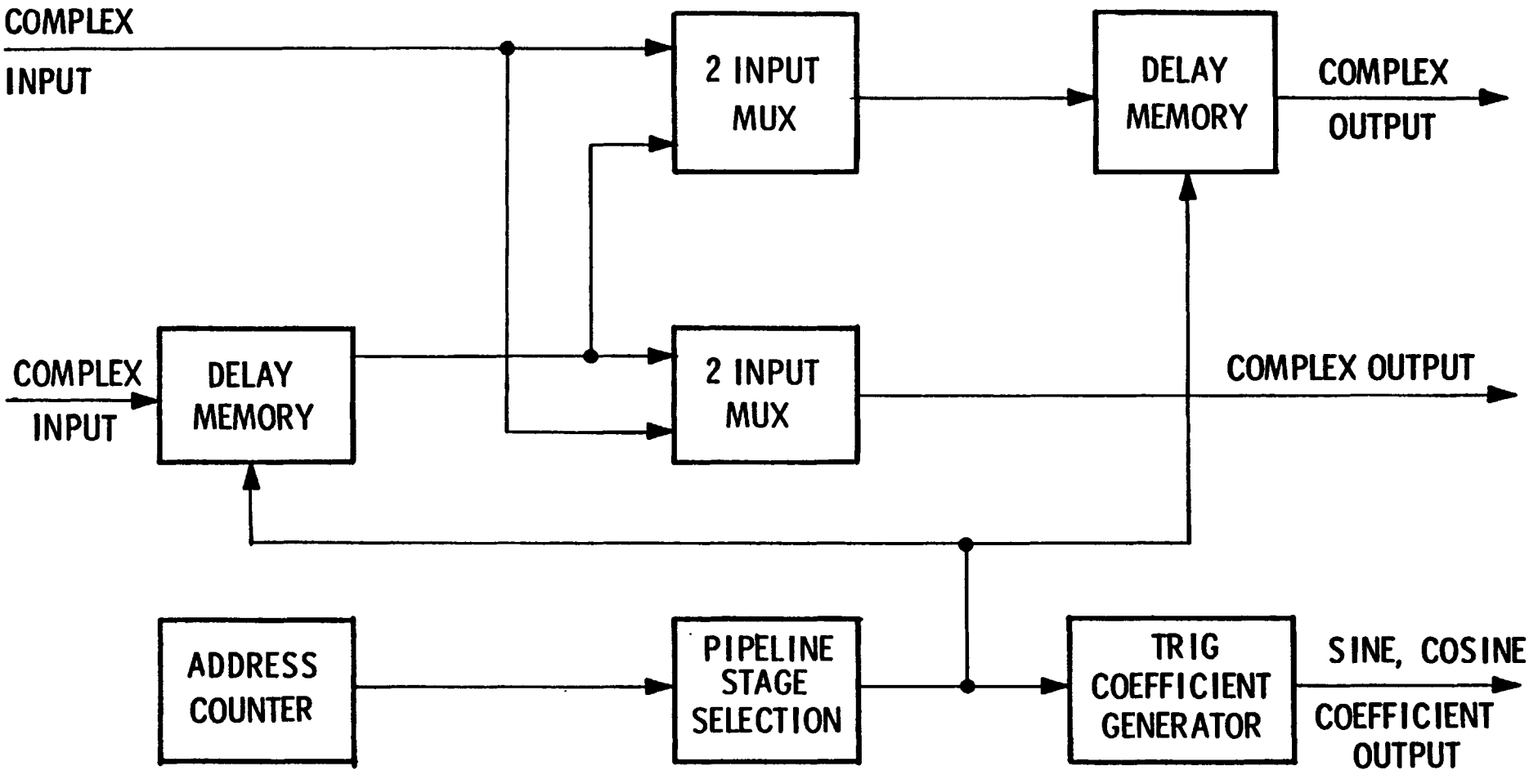
FFT BLOCK DIAGRAM

Fig. 4



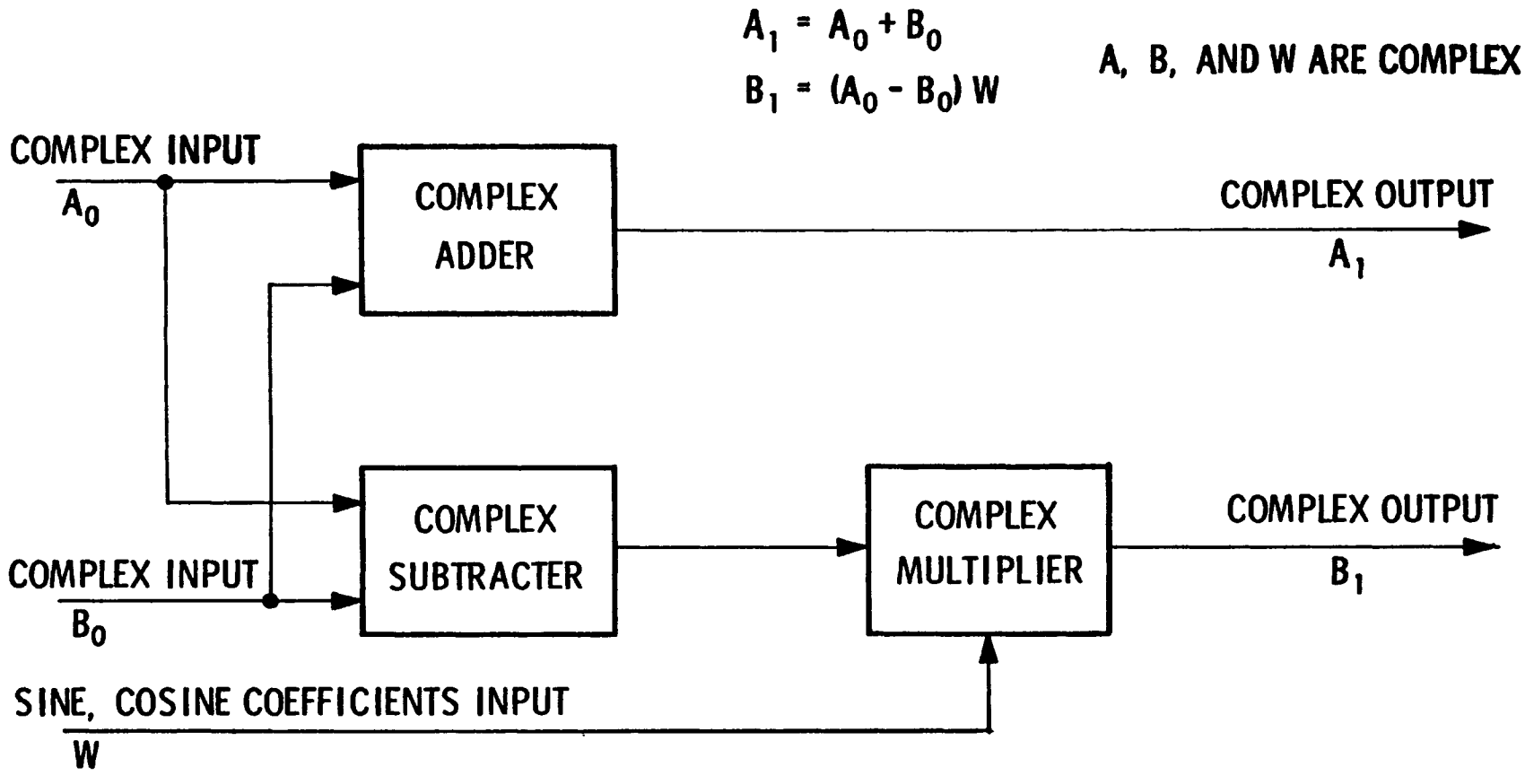
BIPLEX INPUT

Fig. 5



BUTTERFLY MEMORY UNIT

Fig. 6



BUTTERFLY ARITHMETIC UNIT

Fig. 7

Spectrometer

THEORY AND APPLICATION OF DIGITAL SIGNAL PROCESSING

Lawrence R. Rabiner
Bell Laboratories

Bernard Gold
MIT Lincoln Laboratory

TK 7868.D5R32 cpy2 -GB

1975

PRENTICE-HALL, INC. Englewood Cliffs, New Jersey

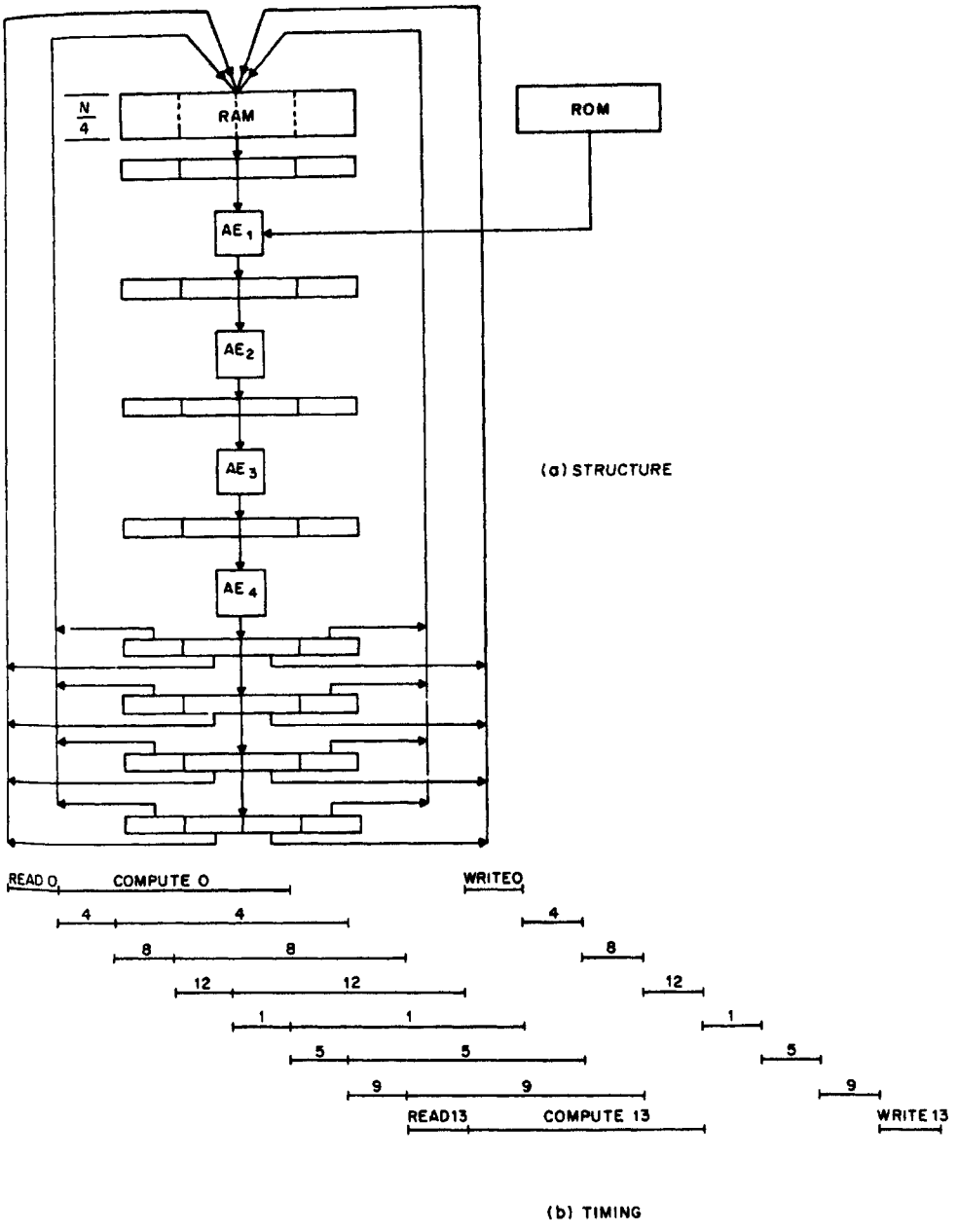


Fig. 10.23 Radix 4 parallel structure and associated timing.

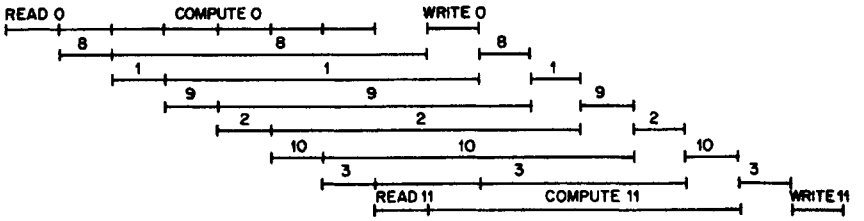


Fig. 10.22 Alternate timing diagram to Fig. 10.21.

An interesting problem for the reader is to construct a structure and determine the required timing to use the same AE to service two RAM's.

Figure 10.23 shows a radix 4 structure and its associated timing. Here the computation time is four units of memory time and eight reads are followed by eight writes. The pipeline culminates in a (4×4) permutation matrix, represented in Fig. 10.23 by four registers, each containing the result of a radix 4 butterfly.

The form of parallelism introduced in this section is based primarily on the notion of matching memory time to butterfly time. We have restricted ourselves to fixed radix systems and have assumed that we have r-fold parallelism for a radix r system. Now, in a radix r system, we have $(\log_r N)$ FFT stages. For each level, each of the (N/r) registers must be accessed twice, once to read the inputs to the butterfly and once to write back the answer. Thus, the number of computational units (or memory cycle times) needed to perform a complete FFT would be

$$C_r = \frac{2N}{r} \log_r N \tag{10.26}$$

and the number of computational units per unit of sampling interval is

$$\frac{C_r}{N} = \frac{2}{r} \log_r N \tag{10.27}$$

Equation (10.27) tells us the highest sampling rate that can be processed in real time given that we know the time per single computation (butterfly or memory). For example, for $N = 1024$ and $r = 2$, $C_r/N = 10$; thus, for a butterfly time of 100 nsec we can process a one-megasample signal.

10.11 General Discussion of the Pipeline FFT

If we go back to the flow diagrams of Figs. 10.1 through 10.8, we note that although the diagrams describe many properties of the algorithm, the precise sequence of butterflies in time is not specified. As a matter of fact, many such

sequences leading to the same result are permissible. For example, in the first stage of Fig. 10.1, we could process the pairs of inputs 0 and 8, 1 and 9, etc., in any conceivable order; the same is true for the other stages. Simplicity of programming or hardware may favor certain time sequences of computation but there are no constraints intrinsic to the structure of the algorithm. In fact, it is not even necessary to complete the first stage before beginning the second stage; for example, if we began the first stage by processing samples 0 and 8 followed by 4 and 12, we could already start the second stage.

We note also that the flow diagrams tell us nothing about the actual hardware structure in terms of the amount of parallelism. The key point we wish to make is this: *Given hardware parallelism, definite constraints begin to appear on the allowable time sequences of the individual butterflies.* In the next few sections we shall describe a class of parallel algorithms called *pipeline FFT* that contains an amount of parallelism equal to $\log_r N$. Thus, for a radix r pipeline FFT there will be $(\log_r N)$ separate hardware butterfly computations proceeding in parallel.

To give some perspective on the amount of parallelism entailed in a pipeline FFT, let us take as an example a 1024-point, or 10-stage, radix 2 FFT. In most general-purpose computers a single hardware multiplier is available. In the pipeline FFT there can be as many as 10 separate "butterfly boxes," which correspond to 40 real multipliers (since each butterfly contains a complex multiplier that contains 4 real multipliers). Thus, assuming that the pipeline FFT structure is as efficient as that of a general-purpose (g.p.) computer realization of the FFT, the pipeline FFT is 40 times faster than the g.p. computer. In turns out that the pipeline FFT structure is from 2 to 20 times more efficient than any general-purpose computer structures that we know of; thus the pipeline FFT structure is from *two to three orders of magnitude* faster. Because of its high efficiency and also because of a relatively simple control mechanism, the pipeline FFT appears at present to be the most important special FFT processor for very high-speed applications.

10.12 Radix 2 Pipeline FFT

Given $(\log_2 N)$ parallel arithmetic elements, we first must ask how flow diagrams such as Fig. 10.1 can be most efficiently implemented. *Efficiency* can be quantitatively described as the percentage of time that the arithmetic elements are kept busy computing butterflies.

For the moment, let us assume that the signal samples appear at the input sequentially, $x(0)$, $x(1)$, etc. Then Fig. 10.24 shows a very simple arrangement for performing the first stage of an FFT corresponding, for example, to the flow diagram of Fig. 10.1. The first eight samples $x(0)$ through $x(7)$ are switched into the eight-stage delay element z^{-8}. The next eight samples are switched to the other input line to the system. Assuming that the butterfly

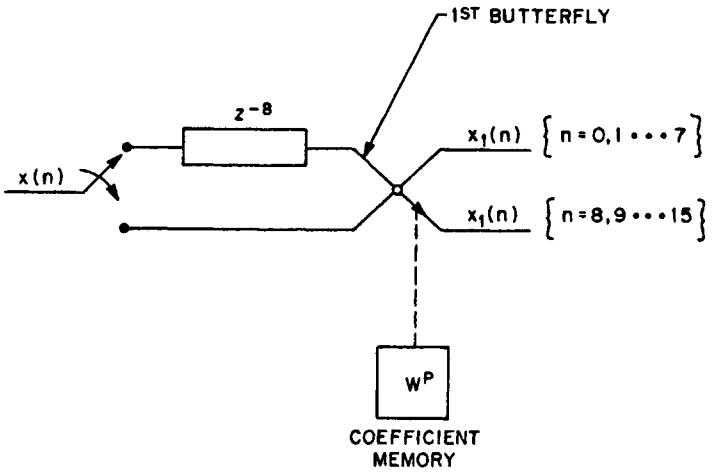


Fig. 10.24 First FFT pipeline stage.

computation time is *exactly* equal to the sampling interval, the entire first stage of the FFT is performed in the subsequent eight-sample intervals following the switching. Results of the first stage [which we have labeled $x_1(n)$] appear in parallel pairs at the butterfly output. Since the coefficient W^p changes from sample to sample, the coefficient memory must be entering its information to the butterfly at the same rate (the sampling rate) as the signal. We notice from Fig. 10.1 that the structural form of stage 1 is repeated twice in stage 2. Thus, we have to devise an arrangement that will process $x_1(n)$ $\{n = 0, 1, \ldots, 7\}$ and $x_1(n)$ $\{n = 8, 9, \ldots, 15\}$ in a manner similar to the way $x(n)$ $\{n = 0, 1, \ldots, 15\}$ was processed. This contrivance is shown in Fig. 10.25. We see that by means of appropriate delays and switching times, we line up the partly processed samples in exactly the way specified by Fig. 10.1. Thus, the "spacing" (difference between the samples in time) was eight time units for the first butterfly and four time units for the second. A complete 16-point pipeline FFT is shown in Fig. 10.26. Here we have an opportunity to observe the various symmetries and, by extrapolation, to construct pipeline FFT's with larger N. Let us make a few remarks about Fig. 10.26.

1. The delay elements in a given stage are half as long as that of the delay elements in an earlier stage.
2. The arithmetic elements are busy only half the time in the figures we have shown.
3. Each switch switches at double the rate of its predecessor.
4. The basic clocking interval of the whole system is naturally equal to the sampling rate.
5. The output is bit-reversed as a function of real time.

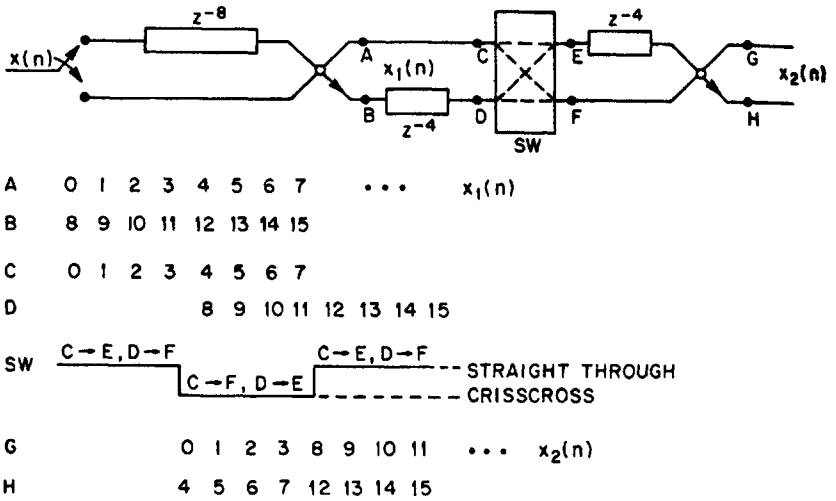


Fig. 10.25 First and second stage of 16-point pipeline FFT, radix 2, DIF.

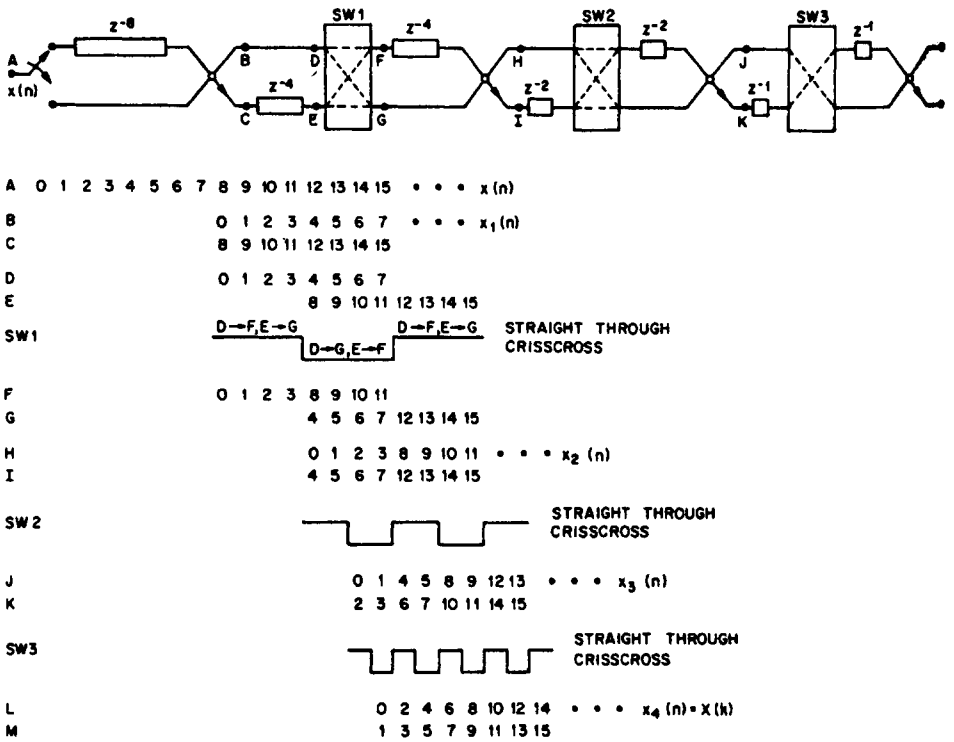


Fig. 10.26 Complete 16-point, radix 2, pipeline FFT, DIF.

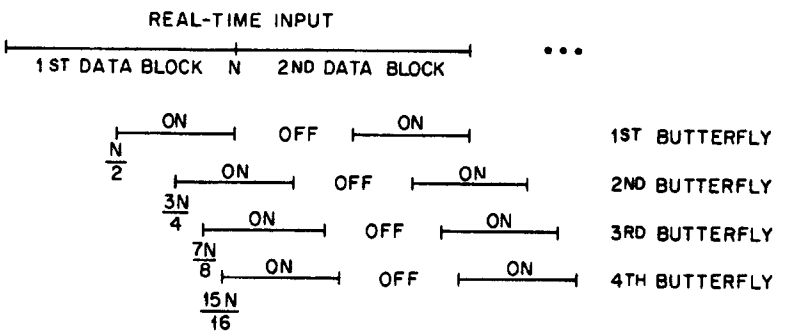


Fig. 10.27 On–off times for arithmetic elements processing contiguous blocks of data.

To prove statement 5 we notice that the indices in Fig. 10.26 are in exact correspondence with the (unlabeled) register numbers in Fig. 10.1. Since in Fig. 10.1 the resultant output is bit-reversed, so is the output of Fig. 10.26. More succinctly, Fig. 10.26 is a specific implementation of Fig. 10.1 and thus possesses all the same properties plus timing properties not specified in Fig. 10.1. We must qualify this remark somewhat by observing that the pipeline FFT structure has a two-port output so that two frequency samples at a time are available. The important point is that the indices shown on the last two lines of Fig. 10.26 are in actuality the bit-reversed indices of the output frequency samples.

With regard to statement 2, this is a rather tricky point and the on time of the AE's is really dependent on how the input is interfaced with the processor. For example, in Fig. 10.27 we chose a requirement that contiguous data blocks be processed in real time. As we see from Figs. 10.24 through 10.26, processing cannot begin until half the data block has entered the processor. Then the first stage is completed in the next $(N/2)$ cycles. At this moment, the first butterfly is turned off until the initial $(N/2)$ values of the next data block have been gathered into the z^{-8} delay element. The other AE's follow the same pattern with a delay. Therefore, the overall system efficiency is 50% since every AE is on exactly half the time. Figure 10.28 shows how system efficiency can be made 100% by using the correct input buffering scheme. After the first data block has been stored, ports (a) and (b) are simultaneously played into the processor. Because of the parallelism of the two ports, playout can be clocked at half the rate of the input sampling. Thus, the first stage of the FFT is finished just when the second data block is ready to be processed. The other stages perform the same way but with the usual pipeline delays. The advantage of this scheme is that the computational clock need be only half as fast as the input clock or, alternately, the same system as that of Fig. 10.26 can handle double the data rate; the price paid is extra input buffering and switching.

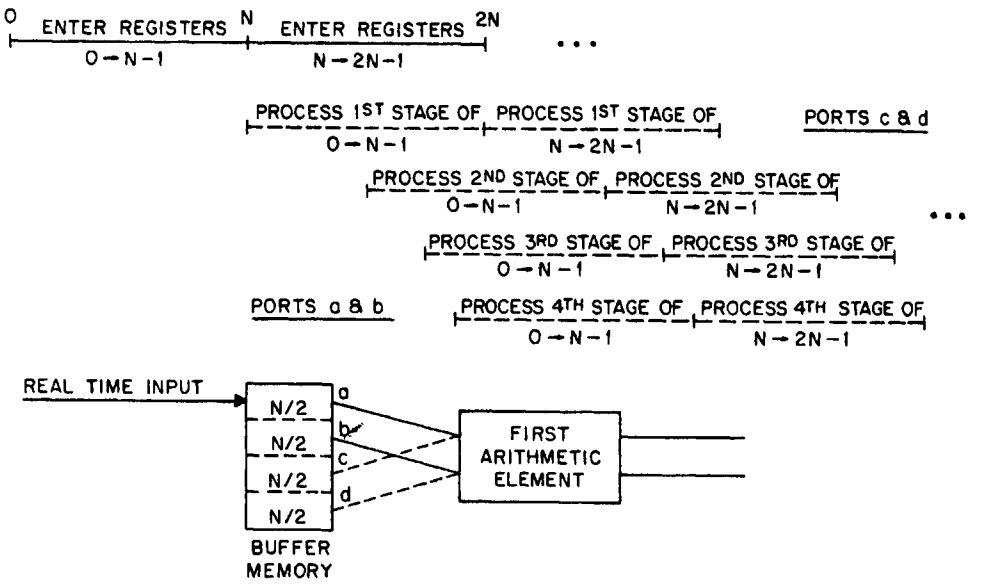


Fig. 10.28 Input buffer arrangement so that contiguous blocks of data can be processed 100% efficiently in real time.

In the special but interesting case of real-time processing with 2:1 overlap of the data blocks (as shown in Fig. 10.29), we simply connect the input to both the z^{-8} delay element and the first arithmetic element. As in Fig. 10.28, the system is 100% efficient in that all AE's are working full time. This special case fits a method of performing convolution by FFT; hence it is quite useful.

With some hindsight we can, in summary, adjust the remarks made with respect to Fig. 10.26. Remark 1 is generally true but alterations in the input buffering will influence the first stage delay; for example, in Fig. 10.28 this delay has been incorporated in the buffer system. Remark 2 need not be

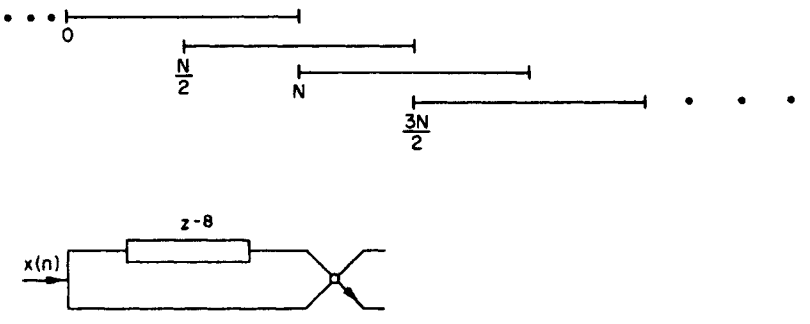


Fig. 10.29 Input configuration for real-time processing of overlapped data blocks.

true since we have shown, via Figs. 10.28 and 10.29, how the AE's can be kept constantly busy. Remark 3 is again true with the first stage being a possible exception and, as seen in Fig. 10.28, the system clock can be slowed down compared to the sampling rate. In all our configurations thus far, the result is bit-reversed and always follows the flow diagram of Fig. 10.1. It appears that other possibilities exist in radix 2 and that pipeline FFT's can be devised from other flow diagrams but at this writing no other structure seems quite as compact and elegant.

A final remark on Fig. 10.26 is that no time was allotted for computation time of the AE's. Including such time does not in any way disturb the structures but it does insert extra delays within the system equal to the number of clock times needed to perform a butterfly. If this number is greater than 1, this implies some "staging" or "pipelining" *within* each AE.

10.13 Radix 4 Pipeline FFT

Beginning with Fig. 10.14 we can work out the structure of a radix 4, 64-point pipeline FFT. As our first exercise we consider the processing of a single data block of 64 samples arranged in normal order. It turns out that a radix 4 pipeline is blatantly inefficient for such an input because the AE's will be working only one-fourth of the time. Nevertheless, this exercise will allow us to analyze the entire structure such that many of the results are applicable for 100% efficient configurations. Making the system 100% efficient is really an input buffering problem that will then be discussed for a variety of input situations.

Figure 10.30 shows a block diagram of the radix 4 pipeline FFT. It is of the same general form as radix 2 but each of the basic elements (delay, commutators, and butterflies) are now geared to radix 4 operations. Thus, the butterfly, instead of performing a complex multiply and two complex adds (as in radix 2), now performs three complex multiplications and eight complex adds. The commutator is a four-input, four-output switch and there are delay elements in three out of the four parallel lines in the system.

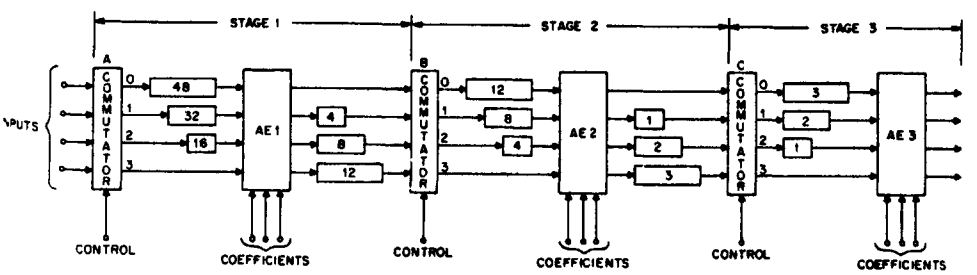


Fig. 10.30 Radix 4, 64-point, pipeline FFT.

US/PASA / JPL(

NATIONAL AERONAUTICS AND SPACE ADMINISTRATION

The Deep Space Network
Progress Report 42-34

May and June 1976

PROPERTY OF THE U. S. GOVERNMENT
RADIO ASTRONOMY OBSERVATORY
CHARLOTTESVILLE, VA.
AUG 24 1976

JET PROPULSION LABORATORY
CALIFORNIA INSTITUTE OF TECHNOLOGY
PASADENA, CALIFORNIA

August 15, 1976