CANADIAN THESES ON MICROFICHE

I.S.B.N.

THESES CANADIENNES SUR MICROFICHE

National Library of Canada
Collections Development Branch

Canadian Theses on
Microfiche Service

Ottawa, Canada
K1A 0N4

Bibliothèque nationale du Canada
Direction du développement des collections

Service des thèses canadiennes
sur microfiche

NOTICE

The quality of this microfiche is heavily dependent upon the quality of the original thesis submitted for microfilming. Every effort has been made to ensure the highest quality of reproduction possible.

If pages are missing, contact the university which granted the degree.

Some pages may have indistinct print especially if the original pages were typed with a poor typewriter ribbon or if the university sent us a poor photocopy.

Previously copyrighted materials (journal articles, published tests, etc.) are not filmed.

Reproduction in full or in part of this film is governed by the Canadian Copyright Act, R.S.C. 1970, c. C-30. Please read the authorization forms which accompany this thesis.

THIS DISSERTATION
HAS BEEN MICROFILMED
EXACTLY AS RECEIVED

AVIS

La qualité de cette microfiche dépend grandement de la qualité de la thèse soumise au microfilmage. Nous avons tout fait pour assurer une qualité supérieure de reproduction.

S'il manque des pages, veuillez communiquer avec l'université qui a conféré le grade.

La qualité d'impression de certaines pages peut laisser à désirer, surtout si les pages originales ont été dactylographiées à l'aide d'un ruban usé ou si l'université nous a fait parvenir une photocopie de mauvaise qualité.

Les documents qui font déjà l'objet d'un droit d'auteur (articles de revue, examens publiés, etc.) ne sont pas microfilmés.

La reproduction, même partielle, de ce microfilm est soumise à la Loi canadienne sur le droit d'auteur, SRC 1970, c. C-30. Veuillez prendre connaissance des formules d'autorisation qui accompagnent cette thèse.

LA THÈSE A ÉTÉ
MICROFILMÉE TELLE QUE
NOUS L'AVONS REÇUE

NL-339 (r. 82/08)

Canada

ARCHITECTURAL CONSIDERATIONS OF SPECIAL PURPOSE FFT PROCESSORS AND THEIR APPLICATIONS TO RADAR SIGNAL PROCESSING

by

Nuthalapati U. Chowdary

A thesis
presented to the University of Ottawa
in partial fulfillment of the
requirements for the degree of
Doctor of Philosophy
in
Electrical Engineering

OTTAWA, Ontario, 1984

UNIVERSITÉ D'OTTAWA
UNIVERSITY OF OTTAWA

The University of Ottawa requires the signatures of all persons using or photocopying this thesis. Please sign below, and give address and date.

ABSTRACT

Fast Fourier transform processor realization by means of stored-product ROMs is described in this thesis. Stored-product ROMs are used in place of hardware multipliers to achieve very high data throughput rates, needed for realtime radar signal processing. Typically, a per sample data throughput rate of above 100 MHz is obtained. New hardware architectures to achieve these high sampling rates are developed for radix-2 and radix-4 serial pipeline and parallel pipeline processing on small sets of samples. Though the discussion on low order FFT ($N \leqslant 32$) architectures is given in the beginning chapters, the emphasis gradually shifts to the realization methods for high order ($N \geqslant 64$) FFT processors. A new method of matrix transposer realization, Random Access Memory Array Transposer (RAMAT), is proposed for parallel processing. Modified FFT architectures for radix-2 decimation-in-time and decimation-in-frequency algorithms are proposed to reduce the finite wordlength effects encountered in the fixed-point arithmetic. Data throughput rate and hardware complexity comparisons are given for each realization method. Computer simulations are carried out to predict the error performance of each of the proposed hardware architectures.

ACKNOWLEDGEMENTS

The author wishes to express his most sincere appreciation and gratitude to his supervisor, Dr. Willem Steenaart for his guidance and encouragement in this research work.

The author is also grateful to Dr. Dipak Roy and his colleagues of Interactive Circuits and Systems Ltd., for technical consultations during the course of this work.

Thanks are due to various members of the secretarial staff of the Department of Electrical Engineering for their help in typing a part of this thesis.

Financial assistance obtained for this research work from the Department of Electrical Engineering, the Natural Sciences and Engineering Research Council of Canada, the Department of Communications, and the Department of National Defence is gratefully acknowledged.

CONTENTS

LIST OF ILLUSTRATIONS

LIST OF TABLES

Chapter I

INTRODUCTION

Spectral analysis, correlation and convolution are important tools that are required in applications such as radar signal processing, digital communications, image processing, speech processing, sonar and geophysical signal processing. Many of these problems require the use of a variety of numerical methods for their solution. A technique that has found wide applicability is the integral transform method [1]. However, numerical problems are generally solved with the aid of a digital computer which is not designed to handle the continuous waveforms that occur when integral transform methods are used. For this reason, it is necessary to convert a continuous time signal to a series of discrete data samples, and to perform numerical operations such as discrete Fourier transform on these samples.

There are several algorithms which are cost effective in terms of hardware and/or software to compute the discrete Fourier transform [2]. The choice of any algorithm mainly depends on the intended application. The cost of implementation usually increases with the accuracy and/or throughput rate required. For example, speech signal processing requires higher accuracy than radar signal processing. Howev-

er, radar signal processors operate at higher throughput rates compared to speech processors.

In radar signal processing, one of the most important signal processing operations is spectral analysis. The data throughput rate requirements of a spectral transform may vary depending upon the spectral range of the signal whose bandwidth determines the range resolution of the system [3].

Doppler processing is commonly used in pulse Doppler radar systems to improve the detection of targets distributed in a severe clutter environment. Clutter is a severe problem especially if the radar system is airborne and operating in 'lookdown' mode [4], because the transmitted energy through the mainlobe and sidelobes of the antenna strikes the ground and consequently strong echoes are reflected. These undesirable reflected echoes (clutter) obscure the echoes received from the desired targets. It is possible to significantly improve the detection capability by exploiting the Doppler frequency shift of the moving targets. Since the relative velocity between the aircraft and the ground is zero as far as the altitude return is concerned, when the antenna is looking straight down toward the earth, the clutter from the ground predominantly occupies the low frequency range of the spectrum [5]. Therefore it is possible to eliminate the ground clutter by high-pass filtering. It is also possible to eliminate rain clutter and clutter due to

slowly moving objects such as birds by this process. Furthermore, the Doppler frequency component and hence the velocity of the moving target can be extracted by passing this high-pass filtered signal through a bank of bandpass filters. Each filter output is fed to a detector with certain threshold to determine the presence of a moving target.

The Doppler frequency component of the moving target depends on the velocity of the target and may be of the order of hundreds of KHz for modern threats. To achieve unambiguous velocity measurements, the pulse repetition frequency (PRF) of the system must be at least twice as much as the maximum Doppler frequency component [6]. High PRFs are essential thus posing stringent signal processing requirements. Typically data throughput rates of several mega samples per second are required.

The formation of a directional beam in the time domain through the use of delay lines or phase shifters can quickly become cumbersome from a hardware standpoint as the number of elements increases. With the Fourier transform, however, it has been possible to take advantage of the frequency domain equivalent of time delay and performing the beamforming with a digital method [7].

Development of digital methods for antenna beamforming in the receiving mode of a radar can be advantageous in providing the system with a high degree of flexibility in "beam-

pattern management" for countering Electronic Counter Measures (ECM) and providing multiple functions efficiently [8].

Beam formation within a digital processor requires that amplitude and phase information appearing at the outputs of various receiving channels, each associated with different parts of the antenna aperture or with different beam directions, be accurately preserved by the process of analog-to-digital conversion. These complex signals are fed to a spectrum analyzer to obtain their spectral equivalents. The principle employed in this beamforming procedure is that a signal, shifted (delayed) in time is equivalent to the spectrum of the signal, multiplied by a complex exponential which is proportional to the time delay [9]. In other words, the spectral samples of the sensors can be multiplied with appropriate exponentials to form the beam in a desired direction.

Digital beamforming requires very high throughput rates of the order of several hundred MHz, since the overall bandwidth of the system is N times the bandwidth of a single channel, where N is the number of beams [10]. Digital beamforming is a cost-effective approach when as many beams are formed as the number of antenna elements.

Another example of high speed signal processing is in synthetic aperture radar where high resolution two dimen-

sional image formation demands high throughput rate signal processing [11]. Synthetic Aperture Radar (SAR) is an active microwave imaging system for all weather terrain mapping. SAR differs from 'normal aperture' radar in that the beam aperture in azimuth is synthetically reduced by processing the Doppler frequency information in the radar return signal [12]. In this way, a sharply focussed beam is synthesized from a normal length antenna, and high resolution is achieved.

Formation of synthetic aperture involves mixing the received signal on a range gate by range gate basis with a referenced function and integrating the mixer output over many interpulse periods. The mixing and integrating is the signal processing, which essentially forms Doppler cells in range gate. When samples corresponding to multiple range gates are processed, an azimuth line is formed. When successive azimuth lines are laid side by side, the map is formed [13].

The SAR processing algorithm can be thought of in terms of Doppler filtering or Fourier transforming [14]. The processing can usually be broken into two phases; range processing and azimuth processing. Most coherent radars use some form of modulation or coding of the transmitted waveform to improve resolution. Appropriate processing is performed at the receiving end to demodulate or decode the ra-

dar returns (range compression). In the second phase, Doppler processing is applied to achieve high resolution in the azimuthal direction (azimuth compression). Thus a high resolution two-dimensional image can be formed. The formation of a real time SAR image requires the processing of large amounts of data at very high throughput rates. As an example, the digital correlator currently existing at Jet Propulsion Laboratory uses three parallel AP120B array processors and achieves a computation rate of only 1/500 of that needed for real time [15].

All these coherent radar signal processors demand very high speed arithmetic computations for realtime processing. There are several other examples requiring high data throughput rates which include Electronic Warefare (EW) systems such as Ballistic Missile Defence (BMD) radar systems to detect, discriminate and track small reentry vehicles in the presence of heavy clutter and ECM [16]. All these applications may require processing on small sets of samples (≤ 32) at high throughput rates.

Historically, the most important event in the fast algorithm development has been the fast Fourier transform (FFT), introduced by Cooley and Tukey [17] in 1965, which compute DFTs with a number of operations proportional to $(N \log_2 N)$ and therefore reduces drastically the computational complexity for large transforms. Since convolutions can be comput-

ed by DFTs, the FFT algorithm can also be used to compute convolutions with a number of operations proportional to ($N \log_2 N$) and therefore played a key role in digital signal processing ever since its introduction. But even with the FFT, it is very difficult to meet the specifications of real time radar processing tasks by means of computer software.

It is obvious that for computer processing of complex samples, FFT is preferred over direct DFT, because of the drastic reduction in the computational requirements. However for special purpose hardware processing, one has to seriously consider both the FFT and DFT algorithms. Though the FFT computational algorithm requires a smaller number of computations than the DFT algorithm, the controls for the data flow required are severe [18]. The data flow problems are simplified in the case of DFT computation. However for processing low order (≤ 32) samples, where the control complexity is not significant, FFT is always preferred over DFT, because for a given data throughput rate (number of complex samples per second), FFT realization requires less hardware (number of Integrated Circuit components) compared to the DFT.

A low order FFT can be realized by an array processor to achieve high throughput rates. But this approach is expensive and inefficient because all the commercially available array processors are optimized for processing a large amount

of data (eg. 1024 point FFT). In other words, it is not cost effective to buy an array processor to calculate a 16-point FFT. A microprocessor based FFT even for processing $N \leq 32$ samples can not handle real time radar signals, because microprocessor-based systems extensively timeshare the memory and the arithmetic components. Several read/write operations reduce the overall thoughput rate and increase the processing delay. A fast Fourier transform processor based on TI9900 16-bit microcomputer [19] completed a 256-point FFT in 736 msec. A 'high performance' microprocessor-based FFT processor was designed [20] with Intel 3000 - bit slice microprocessor, a TRW multiplier, and some TTL circuits to perform a 256-point FFT in 5.4 msec. The same processor could complete a 32-point FFT in 530 μsec. A FFT processor based on TMS320 signal processing chip could perform 1024-point FFT in 78 msec at a throughput rate of 12.8 KHz [21]. One may not be interested in achieving high data throughput rates using emitter-coupled logic (ECL) technology which is expensive and consume more power [34] than transistor-transistor logic (TTL) or complementary metal-oxide semiconductor (CMOS) technology.

A 32 point radix-2 decimation-in-time FFT monolithic chip was fabricated by TRW [22]. The time required to perform a 32 point FFT is 47 μsec at a clock rate of 680 KHz. A bit serial VLSI processor implemented with a radix-4 FFT has been designed [23] to operate at 770 KHz. Two 32 point

floating point chips have been introduced for high speed signal processing [24]. Though a short clock cycle (80 nsec) signal processor has been developed [25], it takes several clock pulses to do arithmetic operations. All these above processors can hardly meet the speed requirements of modern radar signal processing. Hence to realize a high speed low order FFT processor, the best choice is a special purpose hardware design.

In the past, several fast Fourier transform processors for radar signal processing were realized with special purpose hardware. The realization of an eight-point pipeline FFT for pulse-Doppler and pulse compression radars was reported [26]. For pulse Doppler or Moving Target Indicator (MTI) processing, an eight point FFT unit was used in overlapped batch processing [27]. The processor was implemented with 12 bit fixed-point arithmetic and the A/D converter had 8 bit word. The data throughput rate of the processor was 128 KHz. RCA developed a pipeline FFT matched filter to process wideband signals in excess of 10 MHz [28].
Several special purpose hardware multipliers utilizing log-look-up table technique of multiplication were used in this FFT processor. The multiplier was operating at 16 MHz. This 6*6 bit multiplier was fabricated in a 3*4 inch printed circuit board to incorporate 25 Integrated Circuits. The 8-point FFT realization included twelve printed circuit boards for multiplications alone.

These approaches are not suitable for implementing radar signal processors to work at above 100 MHz clock rate. In a conventional method of implementing high speed FFTs, each multiplication is achieved by a separate hardware multiplier without timesharing it. This approach is prohibitive in terms of hardware complexity and cost even for a modest number of points. Investigations of new methods of realizing very high throughput rate and high accurate fast Fourier transform processors, without significantly increasing the hardware complexity and cost are reported in this thesis.

The proposed FFT processor realizations are based on stored-product ROM technique of multiplication. To reduce the cost, the power consumption, the size, the multiplication time and to increase the product accuracy, stored-product-ROMs are used instead of hardware multipliers [29]. It is not necessary to store the coefficients in a separate memory or in a register. The complex FFT multiplications are then performed using ROMs in which the products of all possible real and imaginary parts of the input samples with those of the FFT coefficients are precalculated and stored [30].

This way, we can have coefficient wordlengths as long as necessary to take advantage of the full coefficent accuracy. When an input sample addresses the ROM, the output of the ROM will be the product. The multiplication time is essen-

tially the same as ROM access time. The multiplication time tends to decrease as new developments take place in high density, high speed, and low cost ROMs. Serial pipeline and parallel pipeline FFTs can be developed to meet the radar signal processing requirements [31].

The two dimensional discrete Fourier transforms are double sums computed by summing iteratively in the two coordinate directions. It is straightforward to compute these double sums when the entire matrix can be stored in high speed memory. A problem arises when this is impossible or inconvenient. In many applications the matrix is stored on a block storage device, e.g., a disk, where the smallest record that can be accessed is an entire row or column. In this environment the direct computation of the transform is expensive because of the time involved in accessing the externally stored matrix.

One method of achieving matrix transposition, suggested by Eklundh [32] involves an efficient technique for physically transposing the matrix on disk. Once the matrix is transposed then each row may be individually retrieved and transformed, requiring only one read and write of the data to perform the FFT. Of course, transposing the matrix requires additional data handling. The disadvantage of this method is that it is not well suited for parallel processing hardware, because of many read/write operations, which reduce the overall throughput rate.

A new method of matrix transposition realization based on Random Access Memory Array Transposer (RAMAT) [33] has been proposed, suitable for parallel processing, in which RAMs are arranged in a two dimensional array. The samples coming in parallel are stored in each of the rows of RAMs. After completely storing the samples in rows, samples in the columns are read in parallel from the RAMs arranged in columns. This approach offers high throughput rates due to parallel data transfers. It is possible to realize high speed high order one dimensional FFT processors [34] and high speed low order two-dimensional FFT processor [35] by timesharing a low order FFT unit and with a matrix transposition network.

FFT processor implementation based on fixed-point arithmetic [36] offers high throughput rates at the cost of limited accuracy due to finite wordlength effects, whereas floating-point arithmetic [37] offers better accuracy but with somewhat low speed. For radar signal processing, speed is more important than accuracy. The accuracy achieved with 8-bit fixed-point arithmetic is satisfactory for most of the radar applications. However, high accuracy can be achieved without increasing the number of bits by modifying the FFT architecture. Such new methods [38,39] are proposed to reduce the effect of quantization for low order FFTs. Expressions for the mean square error and the noise/signal ratio of modified FFT architectures are given.

The original contributions of this thesis research are the development of a parallel pipeline FFT processor by means of stored-product ROMs, a high speed matrix transposition operation for parallel pipeline processing, and statistical error analyses of modified FFT configurations.

Other contributions, partly original, include the realization techniques of serial pipeline FFT processors by means of stored-product ROMs and high order one dimensional and low order two dimensional FFT processor realization methods for high speed signal processing.

The role of high speed FFT in radar signal processing is briefly reviewed with a few examples in Chapter II. The requiremets for realizing butterflies for radix-2 and radix-4 algorithms are explained in Chapter III. The radix-2 and radix-4 algorithms are chosen for the hardware realization because of their modularity. The development of radix-2 and radix-4 decimation-in-time (DIT) serial pipeline FFT processors using stored product ROMs is described in Chapter IV. A new realization method for a parallel pipeline low order(N=16) FFT processor to achieve a maximum possible throughput rate is also described. New methods of realizing high order one-dimensional and low order two-dimensional FFT processors are proposed in Chapter V. Expressions for the accumulation of roundoff errors in the proposed modified FFTs are given in Chapter VI. The detailed comparison of

hardware complexity versus speed for each of the FFT processors is given. And finally, the concluding remarks are given in Chapter VII.

Computer simulation results of some of the proposed FFT hardware architectures and practical considerations of the high speed FFT are respectively given in Appendices A and B.

Some of the contents of this thesis have already been reported or published [30,31,33,34,35,38,39].

Chapter II

ROLE OF HIGHSPEED FFT IN RADAR SIGNAL PROCESSING

2.1 INTRODUCTION

In this chapter, the application of high speed FFT processors in realtime radar signal processing is described. Three coherent radar signal processing techniques are considered, (1) Pulse Doppler Radar, (2) Beamforming Radar and (3) Synthetic Aperture Radar.

Modern high resolution radar systems require high speed signal processing (the data throughput rate per sample may vary from ten MHz to hundreds of MHz). In the case of pulse Doppler radar, it is essential to detect a high velocity target submerged in high density clutter. Coherent processing techniques improve the detection capability [40].

Beamforming radar associated with a phased array system should be able to simultaneously detect multiple targets and employ ECCM (Electronic Counter Counter Measures) techniques such as null steering and sidelobe cancellation to counter jamming. To form a beam in azimuth and elevation, two-dimensional beamforming is required [41].

Synthetic Aperture Radar (SAR) for high resolution strip mapping of the terrain involves coherent signal processing [42]. In a SAR, a large aperture antenna can be synthesized by processing a large amount of data, without actually increasing the physical dimensions of the antenna.

There are several methods of realizing a radar signal processor for each radar application. Each method of realization mainly depends on the radar system requirements. Signal processing requirements also vary over a wide range depending on the sophistication of the problem. Our description is limited to only three radar applications, to explain the role of high speed FFT.

2.2 PULSE DOPPLER RADAR

The application of high speed low order FFTs in coherent pulse Doppler radar signal processing is described in this section. Pulsed radars using elapsed time between transmitted and received signals to obtain range have (because of range ambiguity problems) a long receive time or equivalently a low pulse repetition frequency. For a specified value of peak power, this restricts the amount of energy transmitted by the radar. This power restriction primarily applies to airborne radars with weight and volume restrictions. To increase the amount of transmitted energy during the target illumination period, pulse Doppler radars were developed

[43]. These radars are characterized by a high pulse repetition frequency (PRF) to increase the transmitted energy. This high PRF, however, eliminates the capability of the radar to use time-discrimination ranging to obtain radar-target range. High PRF radar operation is strictly based on the Doppler principle. A typical design problem is chosen to emphasize the use of high speed low order FFT processor. The principle of pulse Doppler radar (PDR) is described in the following.

A simplified block diagram of a PDR is shown in Fig. 2.1. In a coherent pulse Doppler radar, there is a phase consistency between successive transmitted pulses. That is, the pulses are a part of the continuous wave. This coherence can be achieved by using a highly stable oscillator to produce a continuous wave radio frequency signal, which inturn is amplified and pulse-modulated to produce coherent transmitted pulses. When the coherent pulses are reflected, coherence is retained. On the receive side, the reflected echoes are down-converted to the intermediate-frequency (IF) and fed to the signal processor. The phase of the transmitted signal is important for coherent Doppler processing. Hence to detect the correct Doppler frequency shift, a portion of the transmitted signal is used as the local oscillator (LO) signal while heterodyning the received echo. The receiver essentially consists of microwave hardware and digital signal processor.

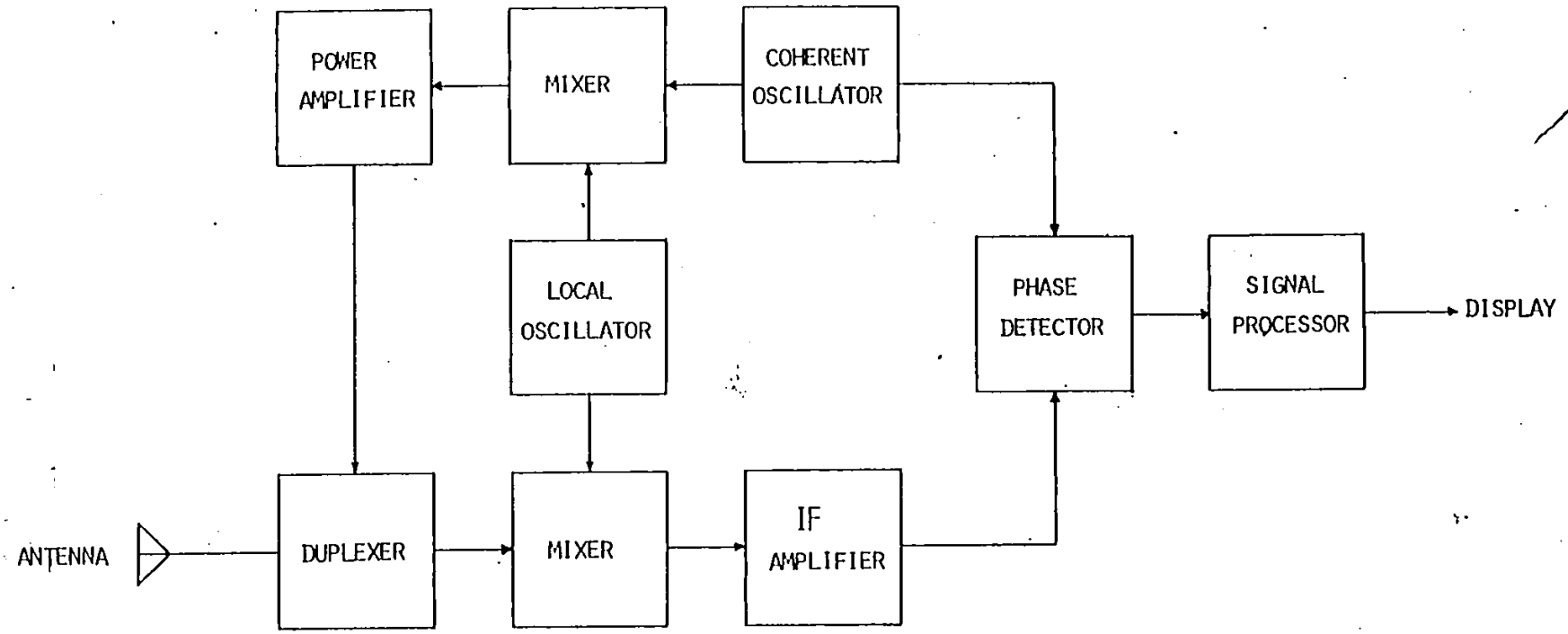


FIG. 2.1 PULSE DOPPLER RADAR

The Doppler frequency component of the signal after A/D conversion is fed to the digital signal processor which consists of a Doppler filter bank. The Doppler filter bank can be realized using a FFT processor. A typical design problem is considered in the next section.

2.2.1 Design Considerations

The requirements of a pulse Doppler radar for military applications such as land-based Mobile Radar System to accurately estimate the velocity of a target is explained through a design problem. The primary requirements of such a radar system should include the capability to detect a large velocity target (eg. fighter plane) and small antenna (easier for transportation).

The primary design parameters are

maximum anticipated target velocity	=	300 m/second
maximum permissible antenna size	=	3 meters

Design Problem:

Let the Transmitted Frequency	(f_t)	=	5 GHz (C-band)
Transmitted Wavelength	(λ)	=	0.06 m

An Antenna of a suitable dimension is chosen such that the azimuth beamwidth is less than 1.2^0 for better angular estimation.

For a Rectangular Aperture, half-power beamwidth

$$(\theta_{BW}) = 0.886\ \lambda/D \text{ radians}$$

where D = antenna width

$$(\theta_{BW}) = 50.8\ \lambda/D \text{ degrees}$$

Let Antenna Width (D) = 2.7 m, $(\theta_{BW}) = 50.8 \times 0.06/2.7 = 1.1289^0$

maximum Doppler frequency shift $(f_d) = 2V/\lambda$

where V = target velocity

$$(f_d) = 2 \times 300/0.06 = 10 \text{ KHz}$$

Pulse Repetition Frequency (PRF) $\geq 2 f_d$

Let PRF = 20 KHz

Inter Pulse Period (IPP) = 50 μsec

Let Pulse Width (τ) = 100 nsec $\leftrightarrow$ 15 m range resolution

Total Number of Range Bins = 50 μsec/100 nsec = 500

Total number of transmitted pulses within antenna beamwidth = $\theta_{BW} \times PRF/AD$

where AD = Antenna Displacement (degrees/sec)

Let Antenna Scanning rate = 60 revolutions/minute $\leftrightarrow$ AD = 360 degrees/sec

Total Number of Doppler Filters = 1.1289 x 20000/360 = 62.71 ≈ 64

2.2.2 Signal Processing Requirements

The radar transmits a set of 64 pulses per beamwidth at a PRF of 20 KHz. The received echoes are range-gated in 500 channels. The data corresponding to first channel for each of the 64 reflected echoes are used for Doppler processing (Fig.2.2). A 64 point FFT is used to do this filtering. Next the second channel data of 64 reflected pulses are used to repeat the same Doppler filtering operation. Thus five hundred 64 point FFTs have to be completed within the antenna dwell time of 3.1 msec. A 64 point FFT has to be completed in 6.2 usec. As will be explained in chapter V, it is possible to perform a 64 point FFT by timesharing a 16 point FFT unit in 2 usec using CMOS integrated circuits (basic clock time is 500 nsec).

Though it is possible to estimate a large unambiguous target velocity, the maximum unambiguous range is limited to a few miles. However the problem of second time around echoes can be avoided by using multiple PRFs. Instead of transmitting 64 pulses at a single PRF, 16 pulses are transmitted for each PRF. And a 16 point FFT can be used to process the return echoes.

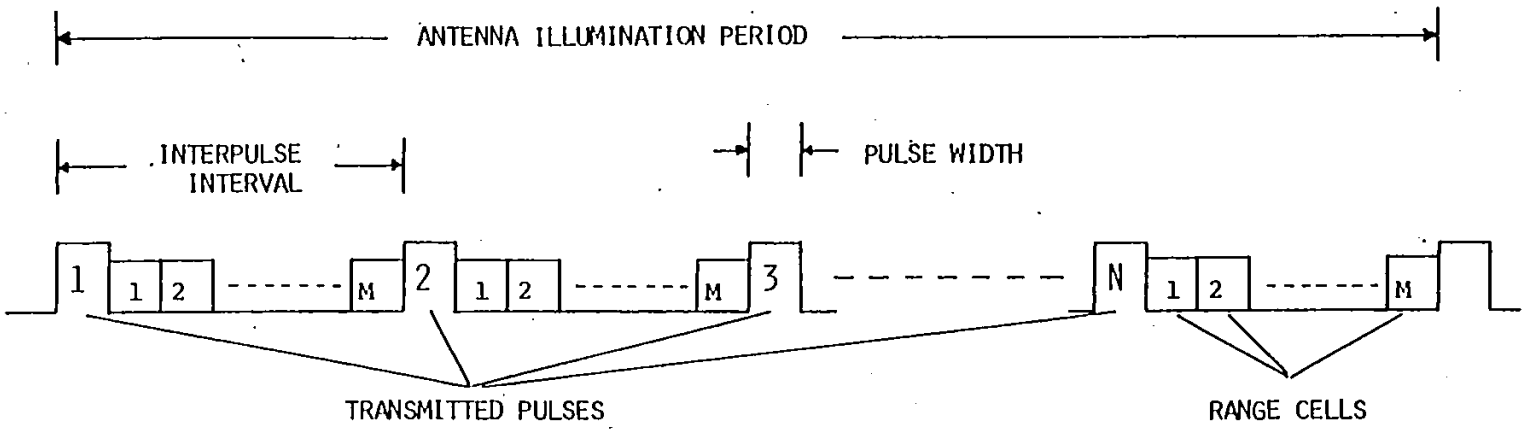


FIG. 2.2 RANGE GATING IN PULSE DOPPLER RADAR

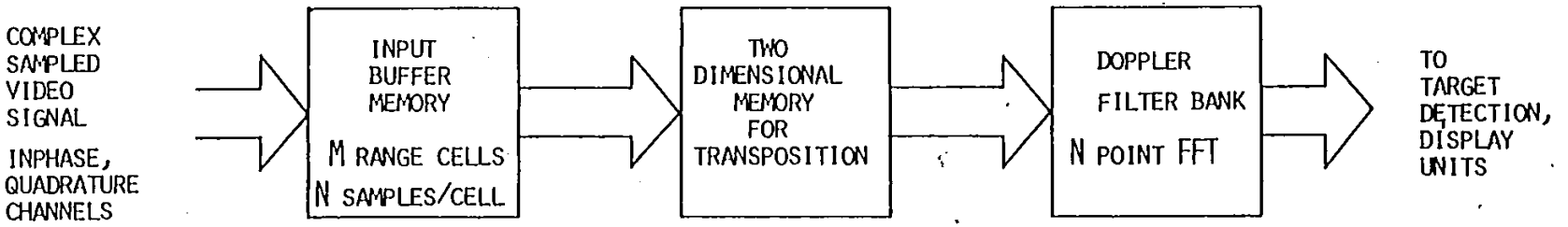


FIG. 2.3 PULSE DOPPLER RADAR SIGNAL PROCESSOR

2.2.3 Radar Signal Processor

Fig. 2.3 illustrates the signal processor for pulse Doppler radar. The inphase (I) and quadrature (Q) A/D converters sample the baseband Doppler shifted signal for each range cell and store it in the input buffer memory. The matrix transposition network holds the data corresponding to each pulse repetition interval in each row. The output of the matrix transposition network is read in columns corresponding to data for one range cell from all the 64 pulse repetition intervals (coherent processing interval). Then a 64 point FFT is used to analyze the Doppler shifted spectrum due to the moving target(s). The outputs of the FFT are used for target detection.

2.3 BEAMFORMING RADAR

A beamformer can be interpreted as a spatial filter which operates on the outputs of an array of sensors in order to enhance the amplitude of a coherent wavefront relative to that of background noise and directional interferences. In a beamforming radar, the phased array provides the capability of large angular coverage in a short period of time since the positioning of the beam is governed by electronic rather than by mechanical means. Paralleling of phase shifters for each element of the array would provide the capability of forming a number of beams equal to the number of parallel

channels. There are several methods of performing beamforming operation in realtime. The broad classification is based on time domain and frequency domain procedures. In this section, we describe a beamforming method based on the frequency domain technique using fast Fourier transform processor. Frequency domain beamforming concepts are the result of the application of the Fourier transform technique to the beamforming process.

2.3.1 Beamforming Fundamentals

The fundamentals of digital beamforming are described in this section. The objective of beamforming is to use the signals received by the sensors in a phased manner so as to preferencially detect signals coming from a particular direction (ie., signals coming in on a particular beam). In addition, by averaging over many sensors, the signal-to-noise ratio (SNR) is increased. An appropriate analogy is that beamforming is related to multidimensional spectral analysis in the same way that bandpass filtering and one dimensional spectral analysis are related.

In a conventional beamformer, the outputs of an array of sensors are combined so as to enhance the amplitude measurement of a propagating, coherent wavefront relative to the ambient background noise and spatially localized interferences. Commonly this is accomplished by correctly time

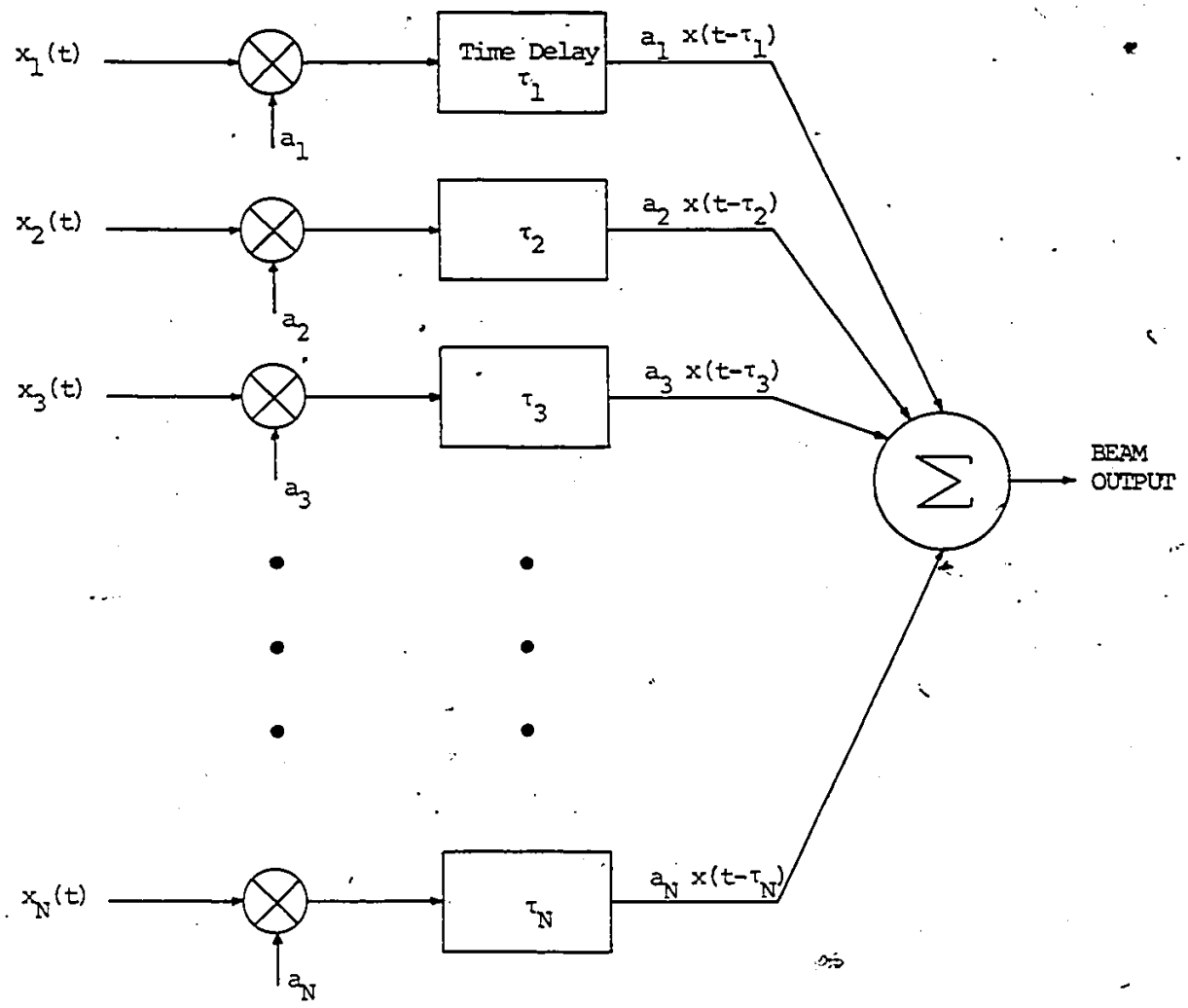


FIG. 2.4 TIME DELAY BEAMFORMER

delaying and summing weighted sensor data as shown in Fig. 2.4.

A modern beamforming technique is based on the frequency domain approach. This is due to the application of the FFT algorithm which offers arithmetic advantages. Due to the linearity of the beamforming process and certain properties of the Fourier transform, the Fourier transform of the beam output can be obtained from the Fourier transform of the sensor outputs in the following way.

$$B(\omega,\theta_\ell) = \sum_{n=1}^{N_E} a_n X_n(\omega) \cdot \exp\{-j\omega\tau_n(\theta_\ell)$$

where $B(\omega,\theta_\ell)$ and $X_n(\omega)$ denote the Fourier transforms of the beam and sensor outputs respectively, ω denotes frequency, the a_n denote the weighting coefficients and the $\tau_n(\theta_\ell)$ denote the delays required to steering the beam in the direction θ_ℓ.

Digital beamforming is performed on the element signals coherently translated to baseband, then time sampled and digitized in an A/D converter. To reduce antenna beam sidelobes, weights are generally applied to each element. The weights are real or complex numbers applied to both the real (in-phase) and imaginary (quadrature-phase) components of the baseband element signals.

A linear array is capable of forming a beam in one dimension whereas a planar (rectangular) array is capable of forming a beam in two dimensions by applying the proper-phase at each element. For a rectangular array of M*N elements in two dimensions, the two dimensional beamforming operation essentially consists of two-dimensional FFT. This can be obtained by taking an M-point FFT on each of the N columns followed by an N-point FFT on the M rows. The fan shaped beam formed in the first FFT is narrowed to a pencil beam on the second FFT. Pencil-shaped beams, having equal width on both beam axes, are desirable for many radar systems.

2.3.2 Digital Multiple Beamforming

The block diagram in Fig. 2.5 shows the basic elements of the analog receiver and digital processing chain. The RF analog signals in the elements are amplified and converted to baseband by a receiver in each channel. Low pass filters (LPFs) matched to the signal bandwidth in both the in-phase (I) and quadrature (Q) channels restrict the frequency band before the analog-to-digital (A/D) converters sample and quantize the signals to digital words. These I-Q components are treated arithmetically as the real and imaginary parts of a complex number. Element signals are processed using the fast Fourier transform algorithm to form multiple beams.

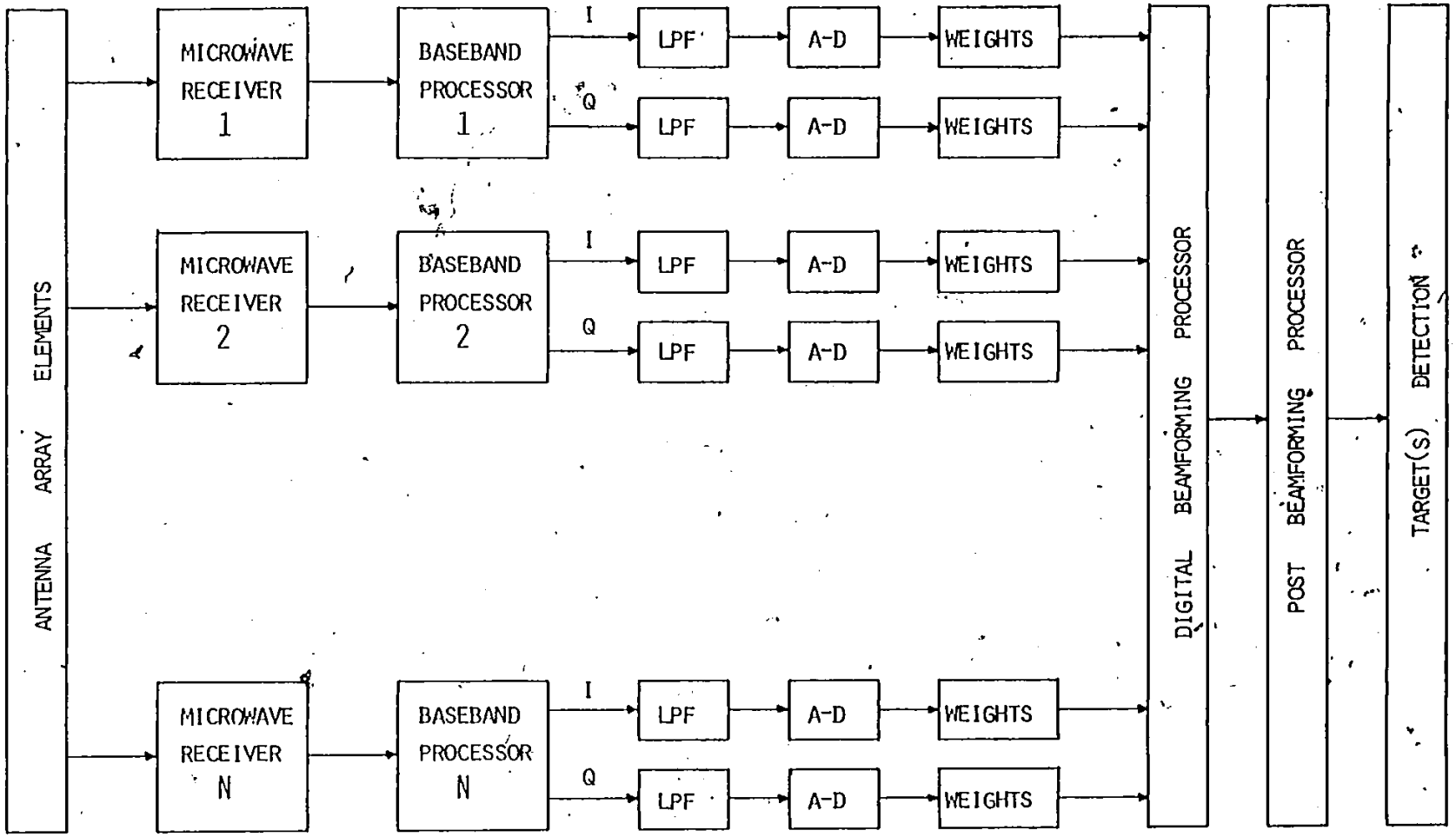


FIG. 2.5 DIGITAL MULTIPLE BEAMFORMING

Each useful beam requires its own waveform matched filter for range and Doppler processing. Thresholding determines the presence of targets. Further processing follows in the data processor.

Waveform processing for range estimating can be performed analog before beam processing as shown in Fig. 2.5. Doppler processing would be performed digitally after beamforming. One range-Doppler processor is required for each beam formed. Thresholding is done on each beam.

2.3.3 Signal Processing Requirements

Signal processing requirements for multiple beamforming vary with the number of antenna elements, geometrical configuration of the antenna array (linear, planar etc.), number of beams formed, and the system requirements for the target detection. A typical design problem is considered for the explanation.

This design example is similar to the one given for pulse Doppler radar. For the pulse Doppler radar, we considered a single beam only. Here we assume that the number of beams is equal to the number of antenna elements.

Design Problem:

Let the Transmitted Frequency (f_t) = 10 GHz (X-band)

Transmitted Wavelength (λ) = 0.03 m

Let the maximum anticipated target velocity (V) = 750 m/sec

Maximum Doppler Frequency shift (f_d) = $2 V/\lambda$

= 2 x 750/ 0.03

= 50 KHz

Pulse Repetition Frequency (PRF) $\geq$ $2 f_d$

Let the PRF = 100 KHz

Pulse Repetition Interval (PRI) = 10 μsec

Let the Range Resolution (r) = 150 meters

Bandwidth of the transmitted pulse (BW) = c /2r

where c = velocity of light (3×10^8 m/sec)

(BW) = 1 MHz

Hence transmitted pulse width (τ) = 1 μsec

Number of Range Gates = 10 μsec/1 μsec

= 10

If a two-dimensional array of 16*16 antenna elements are used in multiple beamforming radar system, 256 beams in azimuth and elevation directions have to be formed in each range resolution time of 1 usec. Assuming that two 16 point FFTs are used for row and column transforms of the two-dimensional array of samples, sixteen 16-point FFTs are to be completed in 1 usec, because of pipelining of the data from row to column FFT. Of course, matrix transposition is required between the two FFTs. Therefore the maximum time allotted for each 16-point FFT operation is 62.5 nsec.

2.4 SYNTHETIC APERTURE RADAR

In this section, the role of low order FFTs in Synthetic Aperture Radar (SAR) is discussed. Synthetic Aperture Processing is a signal processing technique to achieve high resolution in the cross-range direction beyond that which can be obtained with an antenna of fixed dimension. The principle employed in SAR is that a large antenna (with low beam width) is functionally equivalent to a small antenna (with high beam width) that occupies several points of large antenna in time. It is possible to synthesize a larger aperture (high cross-range resolution) by collecting the data at several points and coherently integrating them.

When a radar is moving with respect to the object, it is called SAR. When a radar is relatively stationary with re-

spect to the object, it is inverse SAR (ISAR). Both employ the Doppler principle in discriminating the moving objects. A typical application of SAR is the ground mapping in which an air-borne or a space-borne SAR transmits pulse of electromagnetic energy toward the ground. The time delays of the reflected echoes determine the range. The aircraft or the spacecraft moves in its direction before transmitting the next pulse. Consequently the radar position is changed for each pulse transmitted. All the pulses collected in the cross-range direction are correlated with a reference function to reconstruct a point target. Thus high resolution two dimensional (range and cross-range) ground reflectivity variations are obtained. And the next step is to do image processing of the data to form a high resolution picture of the ground. A typical geometry of SAR is shown in Fig. 2.6. The signal processing aspects of SAR are described in the following section.

2.4.1 Signal Processing Requirements

The signal processing requirements of a SAR vary over a wide range depending upon the specifications of a particular problem. In any case, the basic operation involved is a two-dimensional correlation of the samples received by the radar. The radar return echoes can be modeled as multiple convolved signals due to the point target field (reflectivity) and due to the finite beam width over which the pulse is

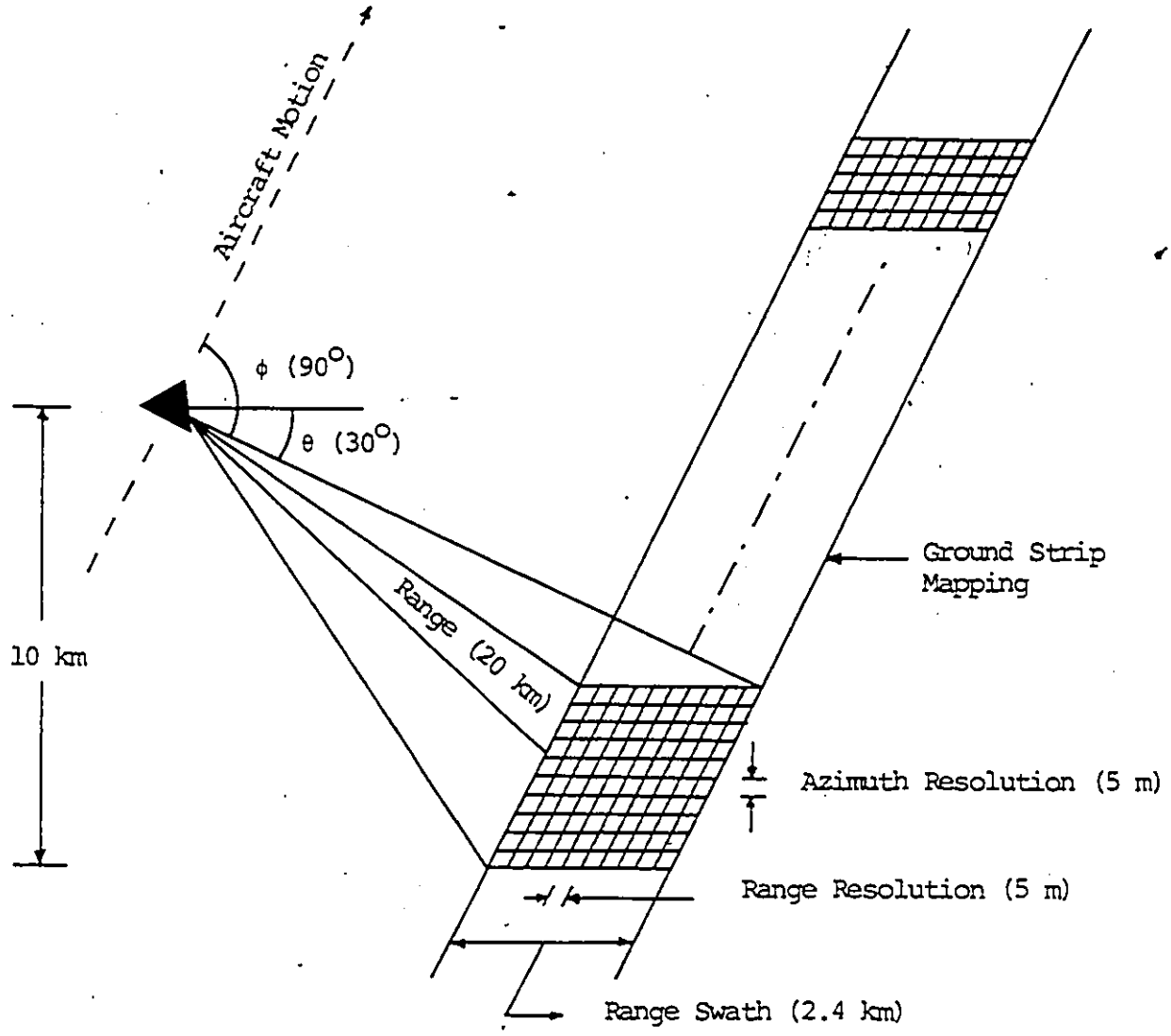


FIG. 2.6 GEOMETRY OF STRIP MAPPING SYNTHETIC APERTURE RADAR

transmitted. To recover the point target field, it is necessary to multiple correlate the return echoes with appropriate impulse responses in the range and in the cross-range. Thus one-dimensional range correlation followed by another one-dimensional azimuth (cross-range) correlation completes the SAR processing.

There are several ways of implementing the two-dimensional correlation operation which include a direct two-dimensional correlator or two one-dimensional correlators to process the data in each dimension. Furthermore, each correlation operation can be realized in the time domain (transversal-filter approach) or in the frequency domain (FFT approach). The choice of each of the methods mainly depends on the hardware complexity and the data throughput rate requirements. Power consumption and cost of realization are also important factors. For high data throughput rate applications, FFTs are preferred. The required data throughput rate depends on several factors including range resolution, azimuth resolution, range, range-swath, transmitted frequency, pulse repetition frequency, receiver bandwidth, antenna size and aircraft velocity. In the following discussion, a typical design problem of airborne SAR is considered, inorder to explain the SAR signal processing requirements.

The primary design parameters of an airborne SAR are

Aircraft Velocity	(V)	= 350 meters/second
Aircraft Altitude	(h)	= 10 Km
Transmit Frequency	(f)	= 15 GHz
Antenna Squint Angle	(ϕ)	= 90^0
Antenna Look Angle	(θ)	= 30^0

The specifications of the terrain map are

Range	(R)	= 20 Km
Range Resolution	(R_r)	= 5 m
Azimuth Resolution	(R_a)	= 5 m
Range Swath	(R_{sw})	= 2400 m

Design Procedure:

Number of range gates (N_r) = $\frac{\text{Range Swath}}{\text{Range Resolution}}$

= 2400/5 = 480

Antenna Beamwidth (θ_{BW}) = $\{R_{sw}$ Sin $\theta/R\}$

= 2400 x Sin 30^0 /20000

= 0.06 rad

$\theta_{BW} = \lambda/L$ where λ = wavelength and L = antenna size

For λ=0.02 m, L = (0.02/0.06) = 33.33 cm

The pulse Repetition Frequency (PRF) f_r should be greater than or equal to the maximum Doppler frequency shift to avoid ambiguities.

PRF (f_r) $\geq$ 2V/L

$\geq$ 2 x 350/0.33 = 2100 Hz

Let the PRF (f_r) = 2200 Hz

Interpulse Period (T) = 454.55 μsec

Let pulse compression ratio (time-bandwidth product) (Δ) = 32

Range Resolution with out pulse compression (δ_r) = 32 x 5

= 160 m

uncompressed pulse width (T_1) = $\{2 \cos \theta \, \delta_r / c\}$

where c = light velocity

$= \{2 \cos 30^0 \; 160/\{3 \times 10^8\}\}$

= 0.9237 μsec

compressed pulse width (τ) = T_1 / Δ = (0.9237/32)

= 0.0288 μsec

Doppler filter Bandwidth (f_d) = $\{2 R_a V/(\lambda R) \sin \phi\}$

= $10 \times 350/(.02 \times 20000) \sin 90^0$

= 8.75 Hz

Coherent Integration Time (t_c) = $1/f_d$ = 0.1142 sec

Number of samples/range gate = $R \lambda/(R_a)^2$

= $20000 \times 0.02/25$

= 16

The number of samples of 16 and the time duration of 0.1142 sec will result in the number of samples/second of

f_s = 16/0.1142 = 140 samples/second

Since the PRF of 2200 Hz is considerably larger than f_s, the input signal should be prefiltered and sampled at a rate of 140 samples per second.

2.4.1.1 Range Processing Requirements

The processing of echo returns from 32 range cells has to be completed in 0.0288 usec which is equal to the compressed pulse width. In otherwords, range correlation for each range cell is done on the echoes from 32 range cells, that is, 16 on either side of the range cell of interest. Hence the total time required for performing 480 range correlations is 13.85 usec. Since the inter-pulse period is 454.55 usec, a PRF buffer can be used to reduce the data throughput rate required.

2.4.1.2 Azimuth Processing Requirements

The azimuth correlator has to process the echo returns from each range cell separately. Each range line has 480 range cells. Therefore 480 azimuth correlations on 16 samples each have to be completed within the time the antenna is displaced in position by azimuth resolution, 14.28 msec. Since this time is much longer than the inter-pulse period, a prefilter can be introduced before the azimuth correlator to reduce the sampling rate and hence the throughput rate requirements. In the next section, the proposed SAR Processor configuration is described.

2.4.2 SAR Signal Processor

A possible method of realizing a high speed SAR Signal Processor to meet the data throughput rate requirements is described in this section. The simplified schematic of SAR signal processor is shown in Fig. 2.7. The architectures of Range Data Processor and Azimuth Data Processor are described in the following.

2.4.2.1 Range Data Processor

The Range Data Processor essentially consists of a PRF buffer and a correlator. As described earlier, the total time for processing one range line (480 range bins) is 13.85 usec. Since the interpulse period is 454 usec, the processor becomes inefficient because it is idle for 440 usec. It is possible to ease the data rate requirements of the processor by using a PRF buffer. The PRF buffer serves simply to stretch the occurrence of A/D output samples from each radar pulse over the time interval available between subsequent pulses. In other words, it is possible to extend the processing of 480 range-bin data over 450 usec. Hence the correlation of 32 range-bin data has to be completed in 0.9375 usec. The FFT based Range Correlator is shown in Fig. 2.8.

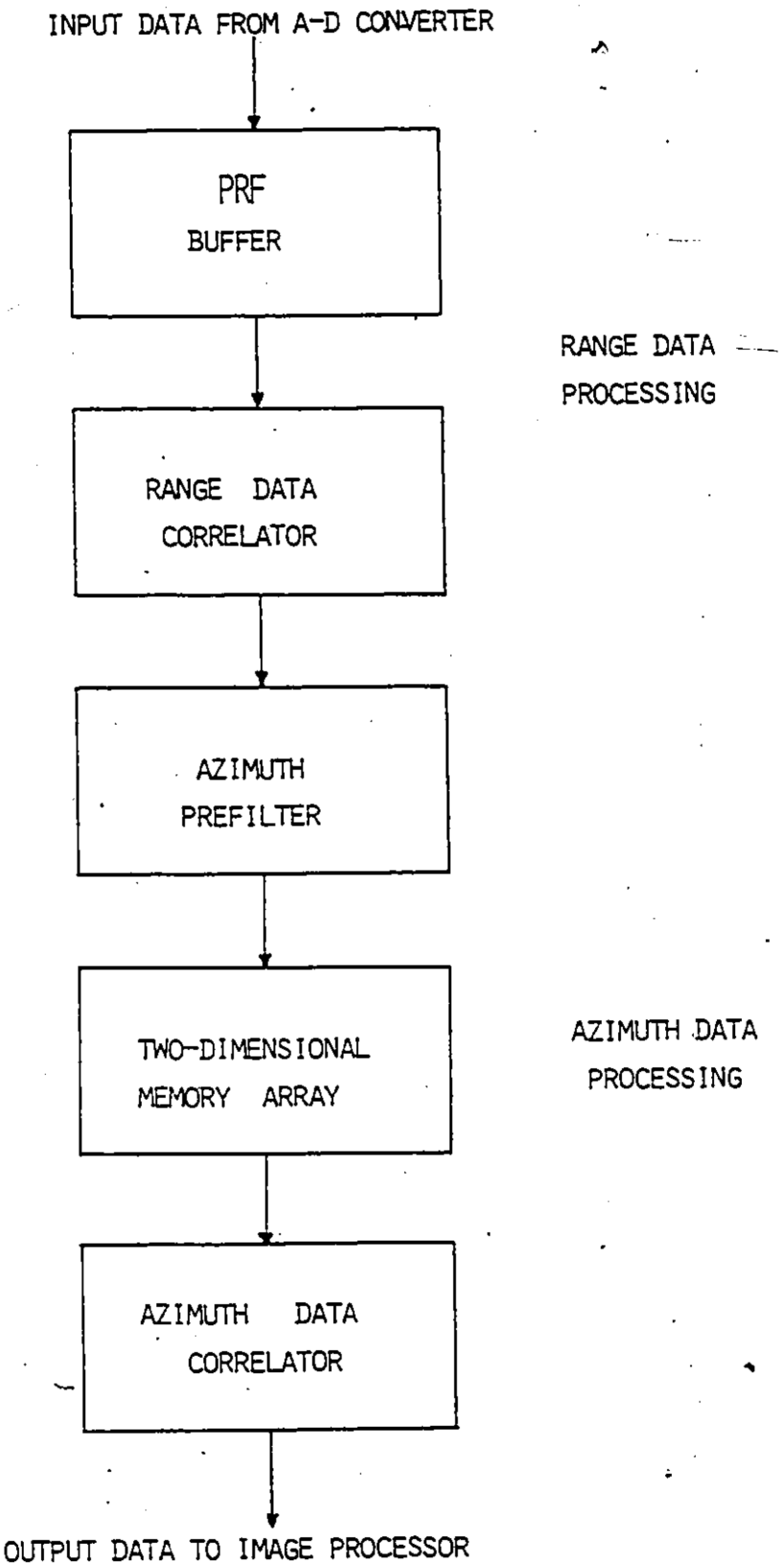


FIG. 2.7 SYNTHETIC APERTURE RADAR SIGNAL PROCESSOR

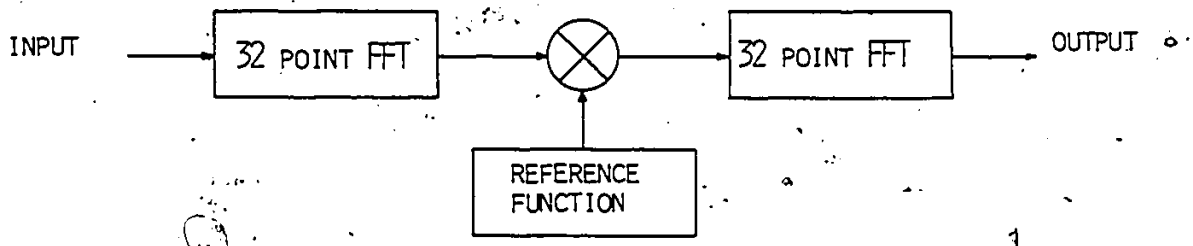


FIG. 2.8 RANGE DATA CORRELATOR

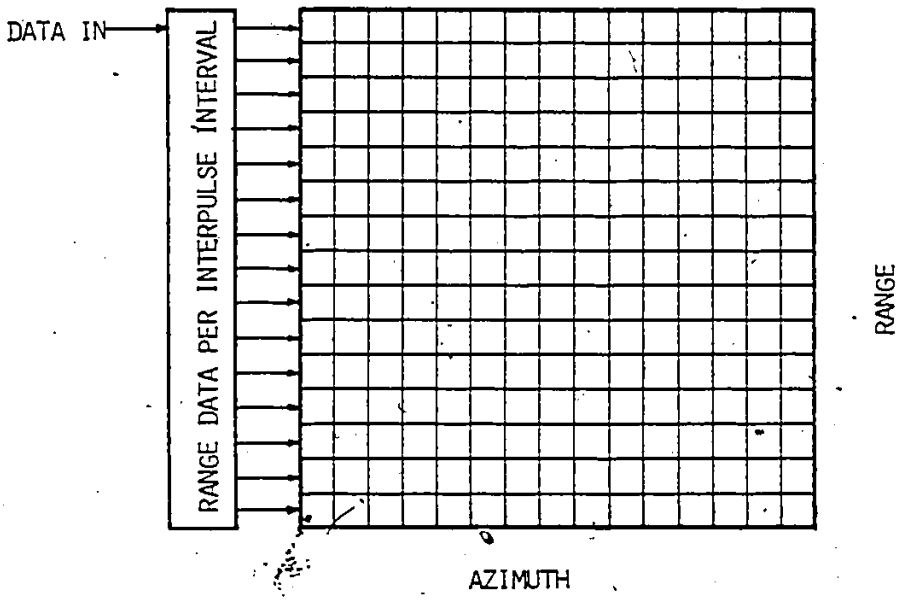


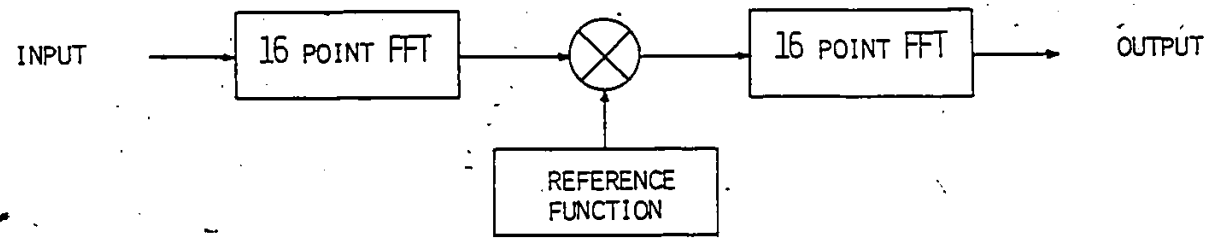
FIG. 2.9 TWO-DIMENSIONAL MEMORY ARRAY

The range correlator can be realized in the form of a FFT based matched filter. Input data is converted to frequency samples by the forward 32 point FFT. The results are multiplied by the frequency samples of the reference function. And finally, an inverse 32 point FFT is used to convert these samples to time domain samples. Overlap-save approach is used to complete all the range correlations.

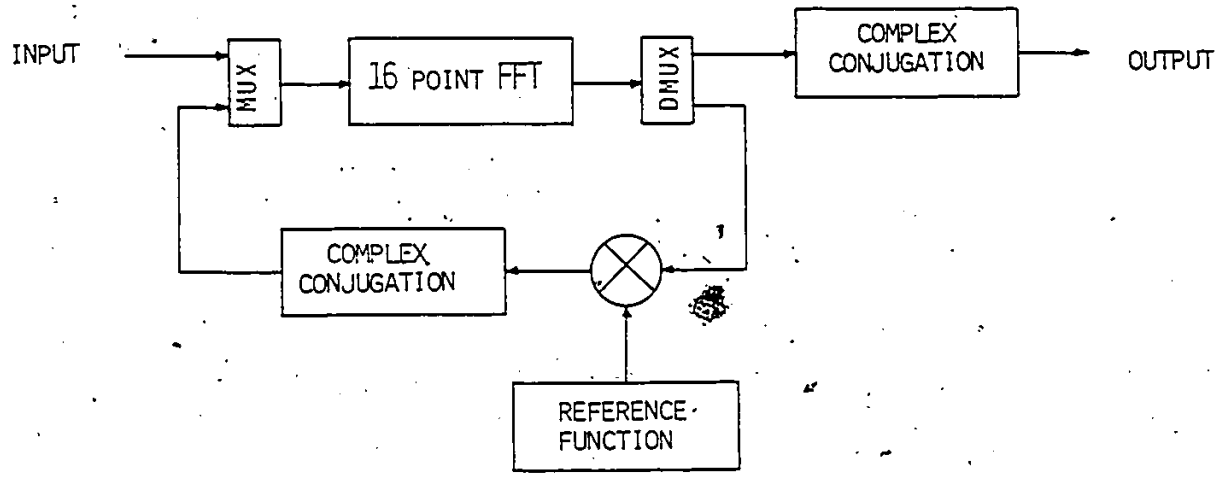
Forward and inverse FFTs and intermediate multiplications have to be completed in 0.9375 usec. It is possible to ease the data throughput rate requirements by pipelining the data through the stages of the forward FFT, intermediate multiplier and the inverse FFT.

2.4.2.2 Azimuth Data Processor

The major elements in the Azimuth Data Processor are azimuth prefilter, a two-dimensional memory, and correlator. The azimuth prefilter serves to reduce the Doppler bandwidth (and hence the azimuth sampling rate) to the minimum required for the desired azimuth resolution. The PRF of a system must initially be great enough to unambiguously sample the Doppler bandwidth associated with the illuminated beamwidth, which is often greater than that needed to obtain the desired resolution due to antenna physical size and/or antenna pattern design constraints [44]. The azimuth prefilter is thus a lowpass filter capable of outputting samples at a lower rate than the input.



(A)



(B)

Fig.2.10 AZIMUTH DATA CORRELATOR (A) TWO FFT UNITS (B) SINGLE FFT UNIT

A 0.33 m antenna can produce a wide enough Doppler bandwidth to give a 0.165 m resolution in azimuth [45]. Since our desired azimuth resolution is 5 m, only (1/32) of the bandwidth need be processed. Thus the prefilter output consists of one range line of data for every 32 range lines of input data. It is possible to improve the azimuth resolution upto a maximum of 0.165 m, by increasing the prefilter output rate.

Since the azimuth correlator must perform repetitive correlation in the cross-range dimension, it must have simultaneous access to a number of azimuth samples in each range cell. This requirement implies a need for a two-dimensional memory containing all range samples in one dimension, and a minimum of all samples in a synthetic aperture interval in the orthogonal dimension.

As shown in Fig. 2.9, the range correlated data is sequentially stored in a two-dimensional memory. After each range line (480 range correlated samples of data) is stored, the data is shifted right, and the data of the next range line is stored. After the accumulation of 16 range lines of data in the two-dimensional memory, the data has to be read 16 samples at a time from each range gate for azimuth correlation. After all the correlations (480 azimuth correlations) are completed, the data is shifted right to accomodate a new rangeline. Therefore for each azimuth

correlation, a new range line and 15 old range line information have to be used.

As mentioned earlier, 480 azimuth correlations have to be completed in 14.28 msec. Each azimuth correlation of 16 samples must be completed in 29.75 usec. The azimuth correlator is shown in Fig. 2.10(a). This is similar to the range correlator of Fig. 2.8. Since the azimuth data processing requirements are not severe, it is possible to timeshare a single 16 point FFT for both forward and inverse transforms as shown in Fig. 2.10(b). Since the azimuth correlation is range-dependent [46], the complex reference spectrum has to be updated for each range-bin processing.

The final stage in SAR processing consists of image detection. Either $(I^2+ Q^2)$ or $(I^2+ Q^2)^{\frac{1}{2}}$ must be computed using the complex (inphase and quadrature) components of the signal samples, depending upon whether an intensity or magnitude image is required.

It is clear that all the above radar systems require very high throughput rate processors for realtime signal processing. A study of finding high speed special purpose hardware architectures, suitable for these variety of signal processing applications is necessary to find cost effective solutions. Such methods are proposed in the forth coming chapters.

Chapter III

FFT PROCESSOR REALIZATION BY MEANS OF STORED-PRODUCT ROMS

3.1 INTRODUCTION

The principles of realizations of the fast Fourier transform processor using stored-product ROMs are described in this chapter. For the realization of digital filters, multipliers using Read-Only-Memories in either distributed arithmetic [47] or Stored-Product ROM form [48] have been reported. Multiplication by the stored-product ROM, an essentially parallel method of multiplication, is advantageous in terms of high speed, no coefficient quantization, and no coefficient storage. It is found that stored-product ROM concept is well suited for the FFT realization because of constant coefficients used.

Review of the basic FFT algorithms for radix-2 and radix-4 structures is given in order to find the arithmetic computations needed. The stored-product ROM technique of multiplication is outlined. The basic processing blocks, called the Arithmetic Processing Elements are described. This is the range of structures from which the 16-point FFT realization will have to be chosen.

3.2 REVIEW OF FFT ALGORITHMS

The derivation of the radix-2 and radix-4 algorithms are well described in texts [49,50]. In this section, the signal flow graphs for radix-2 and radix-4 algorithms are described. The radix-2 butterfly computation involves one complex multiplication and two complex additions [50]. Each complex multiplication involves four real multiplications and two real additions. Each complex addition is equivalent to two real additions. The number of butterflies is the same for decimation-in-time (DIT) and decimation-in-frequency (DIF) algorithms. The choice of the algorithm depends on the length of the sequence (N), because finite wordlength effects such as scaling and product round-off for both DIT and DIF algorithms are dependent on the length of the sequence [51]. In our further discussion, we consider the decimation-in-time algorithm only. A similar discussion is valid for the decimation-in-frequency algorithm. For an N-point radix-2 decimation-in-time algorithm of $(\log_2 N)$ stages, the number of butterflies is given by $[(N/2)\log_2 N]$. For a 16-point radix-2 DIT FFT (Fig. 3.1), there are no nontrivial multiplications required in the first two stages. In the third and fourth stages, there are multiplications due to the twiddle factors W^1, W^2, W^3, W^5, W^6 and W^7.

Fig. 3.2 (a) shows a signal flowgraph for a radix-4 16 point DIT FFT algorithm. The number of stages in a radix-4

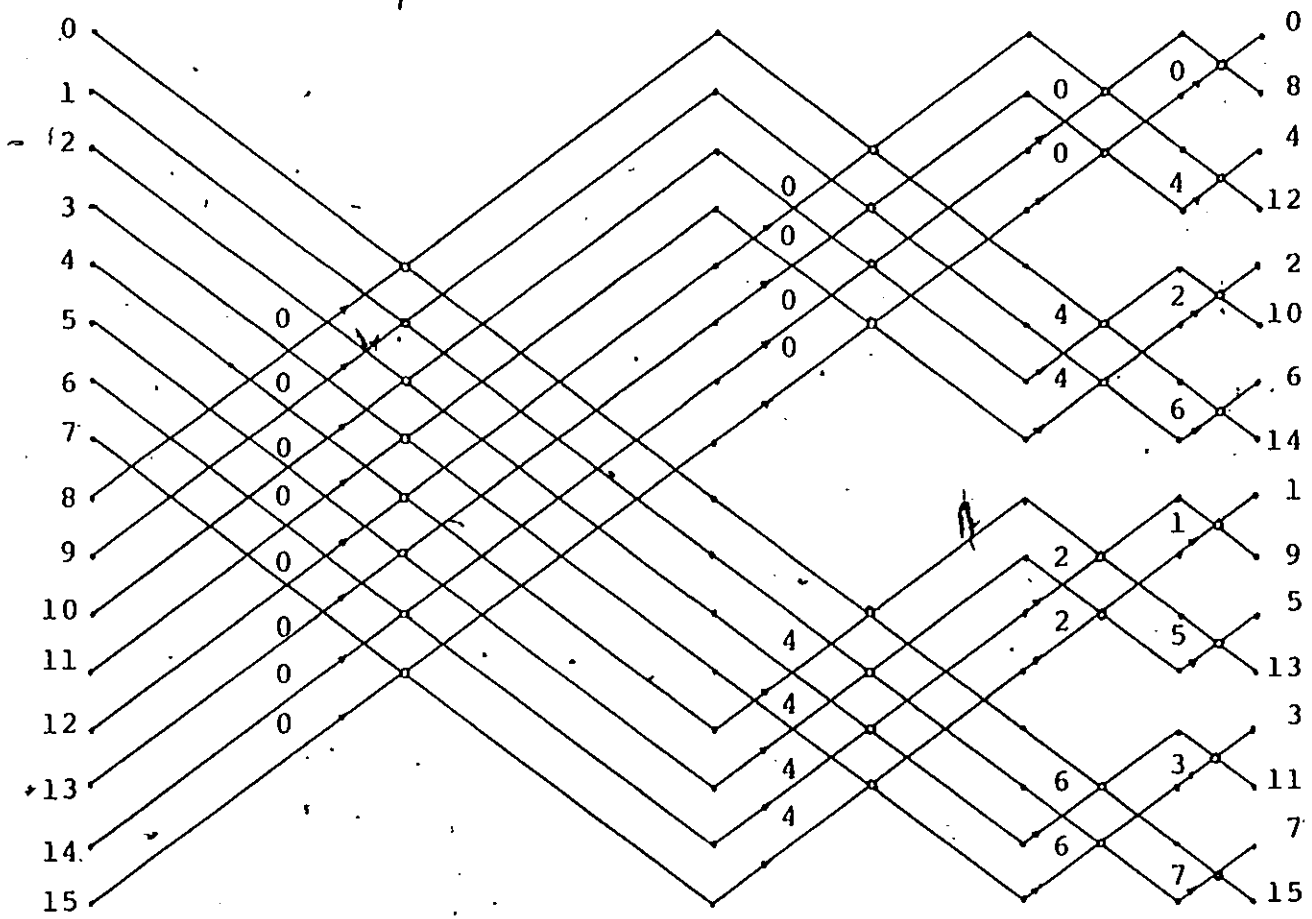
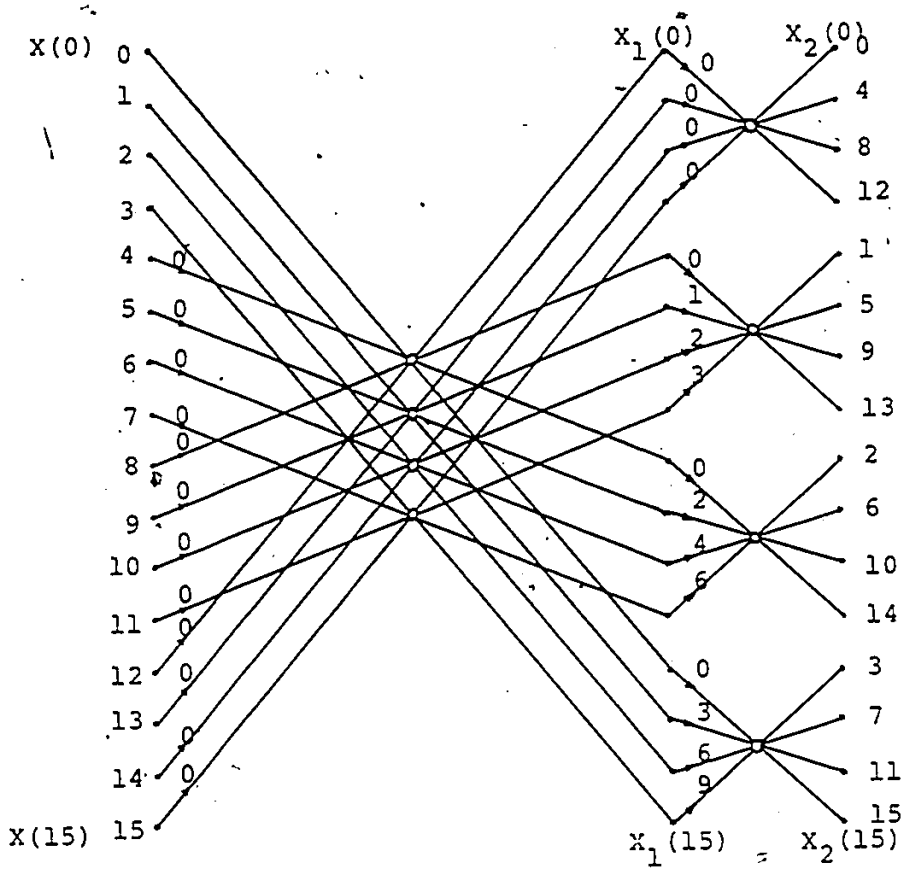
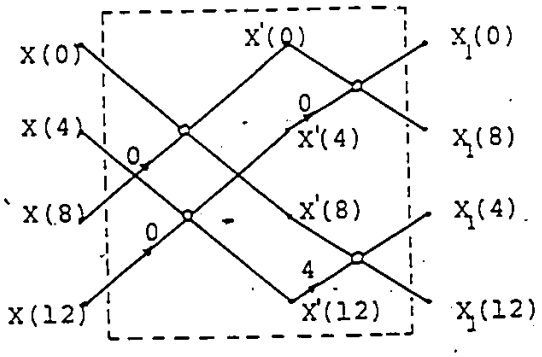


FIG.3.1 16-POINT DECIMATION-IN-TIME FFT ALGORITHM (Radix-2)



(a)



(b)

Fig.3.2 (a) 16-POINT RADIX-4 DIT FFT (b) BUTTERFLY

FFT algorithm is $\log_4 N$, and the number of butterflies is $(N/4).\log_4 N$. The butterfly diagram for the samples X(0),X(4),X(8) and X(12) is given in Fig. 3.2 (b). The samples $X_1(0), X_1(4), X_1(8)$ and $X_1(12)$ are the outputs of this butterfly. The expressions given below describe the radix-4 operation. All the input samples are assumed to be complex-valued.

$$X'(0)=X(0)+X(8)$$
$$X'(4)=X(4)+X(12)$$
$$X'(8)=X(0)-X(8)$$
$$X'(12)=X(4)-X(12)$$

$$X_1(0)=X'(0)+X'(4)$$
$$X_1(8)=X'(0)-X'(4)$$
$$X_1(4)=X'(8)+X'(12).W^4$$
$$X_1(12)=X'(8)-X'(12).W^4$$

where $W=\exp(-j2\pi/N)$

$$X_1(0)=X(0)+X(8)+X(4)+X(12)$$
$$X_1(8)=X(0)+X(8)-X(4)-X(12)$$
$$X_1(4)=X(0)-X(8)+(-j)[X(4)-X(12)]$$
$$X_1(12)=X(0)-X(8)+j[X(4)-X(12)]$$

$$X_1(0) = X(0) + X(4) + X(8) + X(12)$$
$$X_1(8) = X(0) - X(4) + X(8) - X(12)$$
$$X_1(4) = X(0) - j.X(4) - X(8) + j.X(12)$$
$$X_1(12) = X(0) + j.X(4) - X(8) - j.X(12)$$

The number of non-trivial multiplications in the radix-4 algorithm is 8 as compared to 10 in the case of radix-2 algorithm. In the radix-4 algorithm, the butterfly computation is performed on four complex samples at a time. Hence in serial processing systems, a radix-4 FFT processor works twice as fast as a radix-2 FFT processor. In parallel pipeline radix-2 and radix-4 FFT processors, the data throughput rates will be the same, because all butterflies are realized. The complexity of radix-4 algorithm is comparable to that of radix-2 algorithm. The details are given in chapter IV.

3.3 MULTIPLICATION BY STORED PRODUCT ROMS

In software as well as in hardware digital system implementation, the coefficient and signal values are both quantized by rounding or truncation. But there is an important difference in performing a multiplication with a digital multiplier or with the stored product ROM technique. For a digital multiplier (Fig. 3.3), the output is the truncated or rounded product of the input signal represented by L_s bits and of a coefficient represented by L_c bits.

An important problem then is to choose the coefficient wordlength L_c as short as possible to minimize the hardware complexity and simultaneously as long as possible to minimize the perturbations introduced in the transfer function.

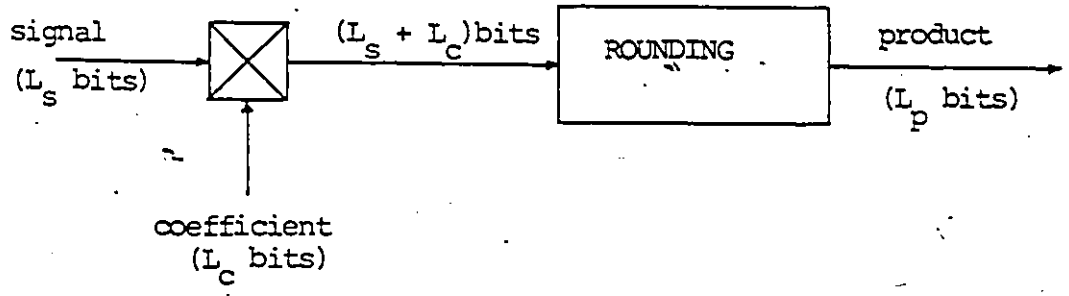


Fig.3.3 MULTIPLICATION BY DIGITAL MULTIPLIER

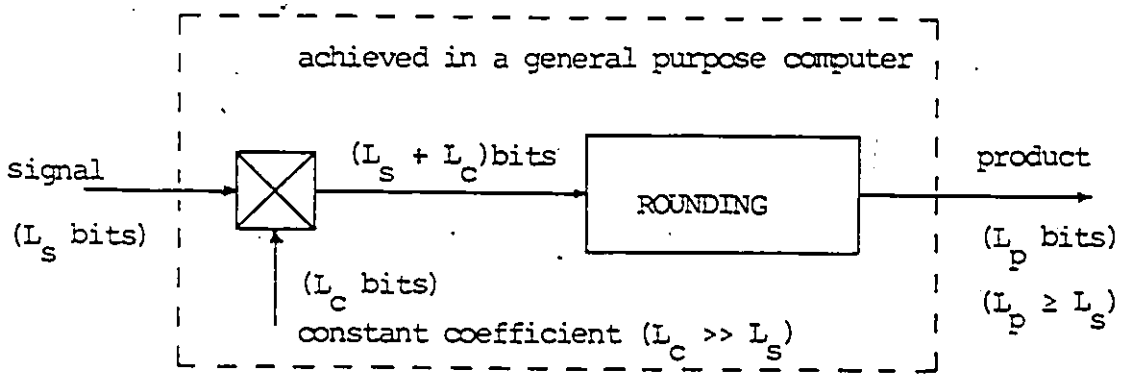


Fig. 3.4 MULTIPLICATION BY STORED PRODUCT ROM

For stored product ROM multiplication the coefficient wordlength can be very long compared to the signal and product wordlengths and this reduces the requirement for finding structures of minimal sensitivity to coefficient wordlength. For the FFT, procedure for multiplication with stored product ROMs requires that all possible products of the input signal of length L_s bits are calculated, which correspond to a coefficient (twiddle factor) of $L_c >> L_s$ bits, rounded to L_p bits and stored in the ROM for that twiddle factor. The procedure used for the computation of these products is shown in Fig. 3.4.

As the computer accuracy allows $L_c >> L_s$, the coefficient quantization effect is hereby reduced to an arbitrary low level. This method has been used, as the FFT is ideally suited to the ROM multiplication for constant coefficients and with L_s in the order of 8-16 bits.

3.4 ARITHMETIC PROCESSING ELEMENTS

3.4.1 Adder Unit (Radix-2):

The Adder Unit (AU) shown in Fig. 3.5 consists of two parallel adders and two parallel subtracters. This AU takes two complex input samples and performs complex additions/subtractions and gives two complex output samples. The AU processing time is effectively the time for one parallel addition or subtraction.

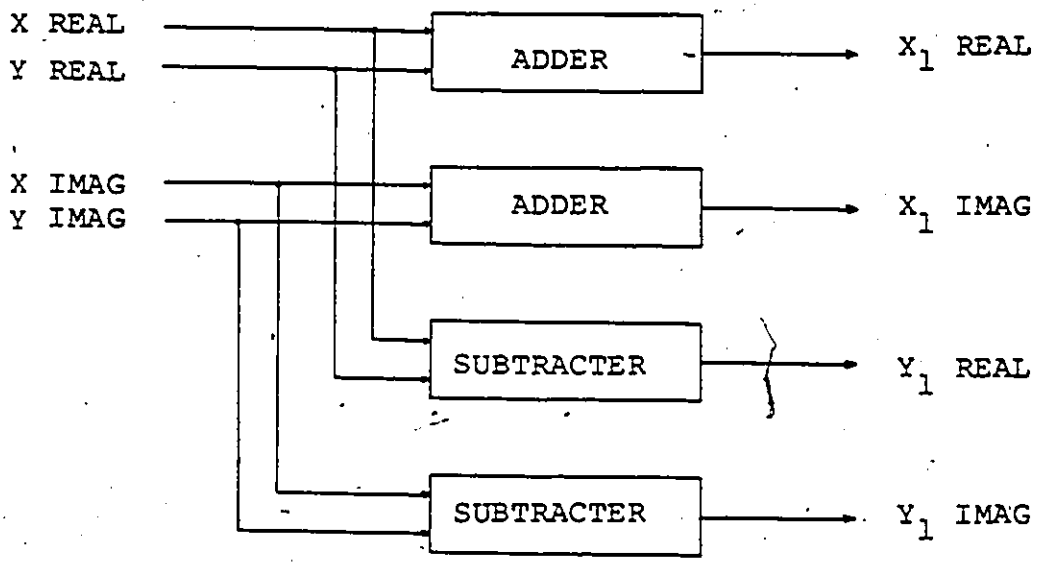


Fig. 3.5 ADDER UNIT FOR RADIX-2 FFT

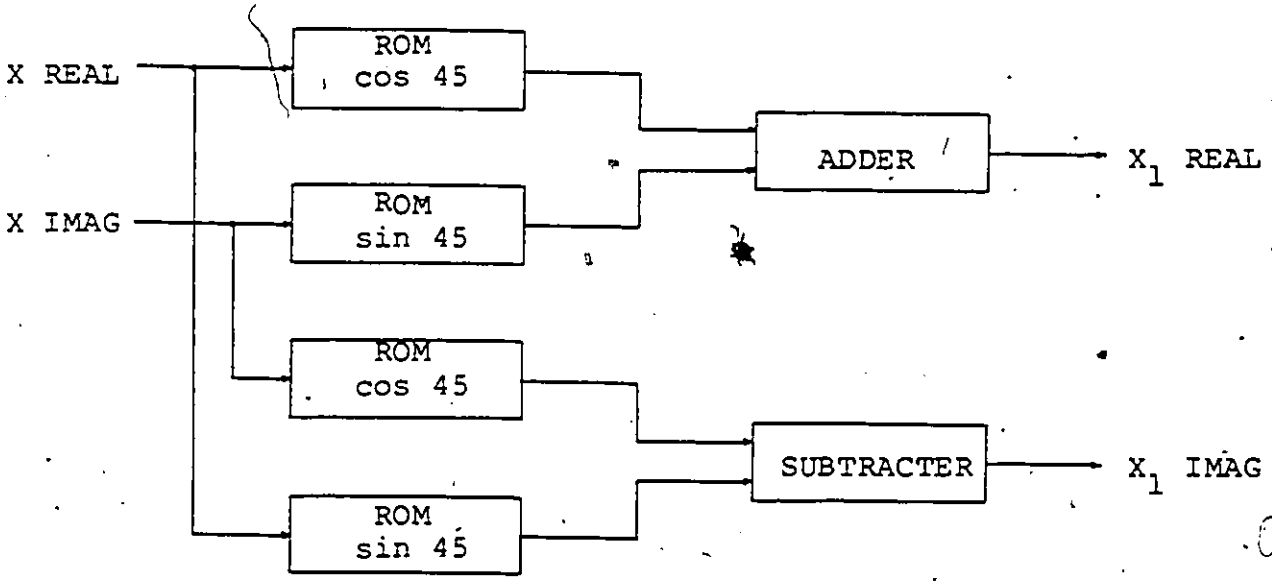


Fig.3.6 ROM MULTIPLIER CONFIGURATION FOR RADIX-2 FFT

3.4.2 ROM multiplier array (radix-2):

When there is a multiplication required, stored-product-ROMs are used. The real and imaginary parts of one of the two complex input samples address separately these ROMs to get products of the real and imaginary parts of the input sample and cosine and sine values of the twiddle factors separately and these are added and subtracted to get complex product. A typical ROM multiplier configuration for the twiddle factor W^2 is shown in Fig. 3.6. Similarly the multiplication by the twiddle factors W^1, W^3, W^5, W^6, and W^7 for the 16 point FFT can be realized with additional ROM arrays.

3.4.3 Adder Unit (Radix-4):

The AU as shown in Fig. 3.7 consists of eight parallel adders and eight parallel subtracters. This AU takes four complex input samples and performs complex additions and subtractions and gives four complex output samples. The AU processing time is effectively the time required for two parallel additions or subtractions.

3.4.4 ROM multiplier array (Radix-4):

This ROM multiplier array consists of 12 ROMs for performing three complex multiplications in each radix-4 butterfly as shown in Fig. 3.8. This Figure specifically illustrates the complex multiplications for the samples $X_1(5)$,

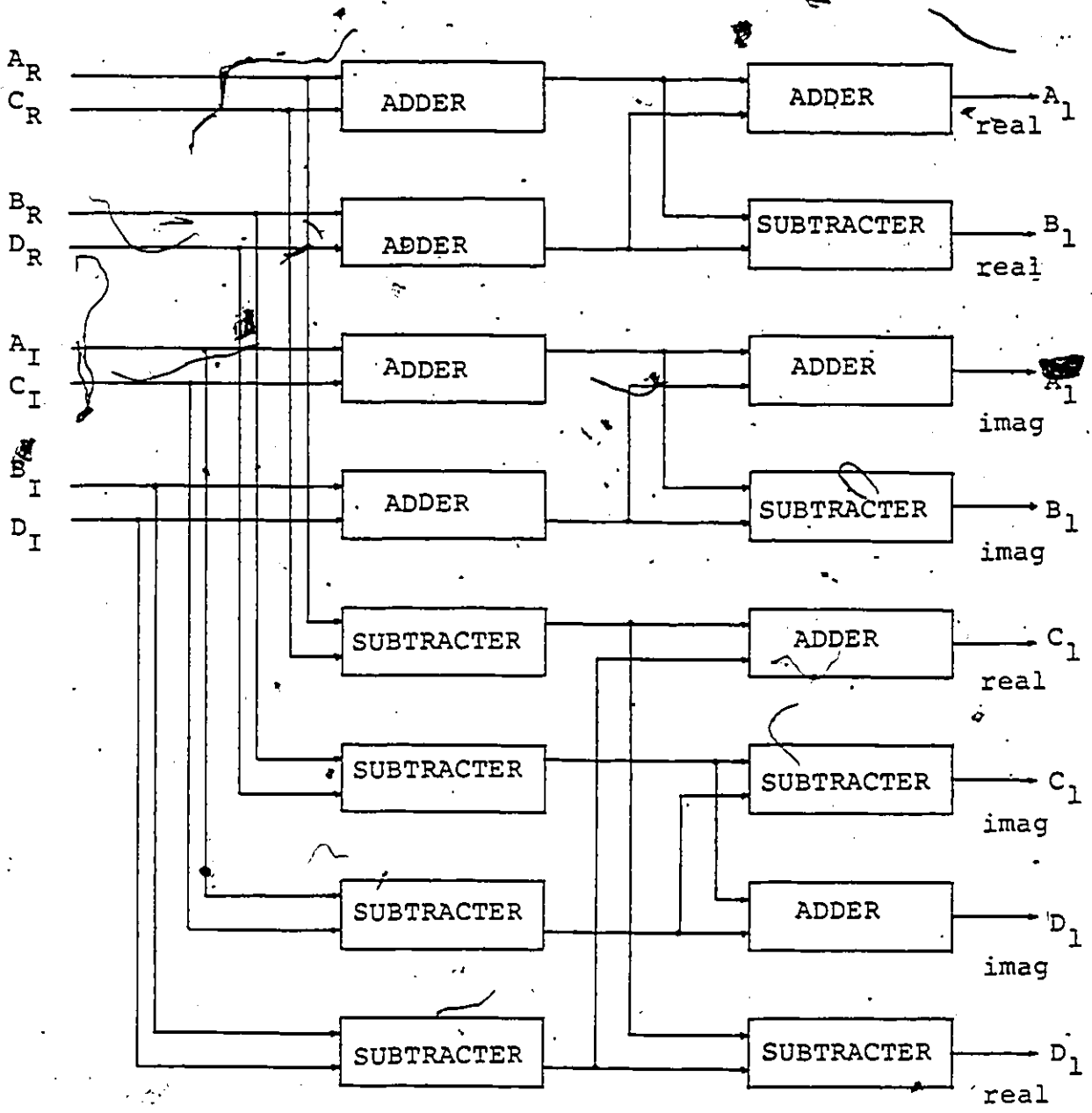


Fig. 3.7 ADDER UNIT FOR RADIX-4 FFT

(for the complex samples A,B,C,D)

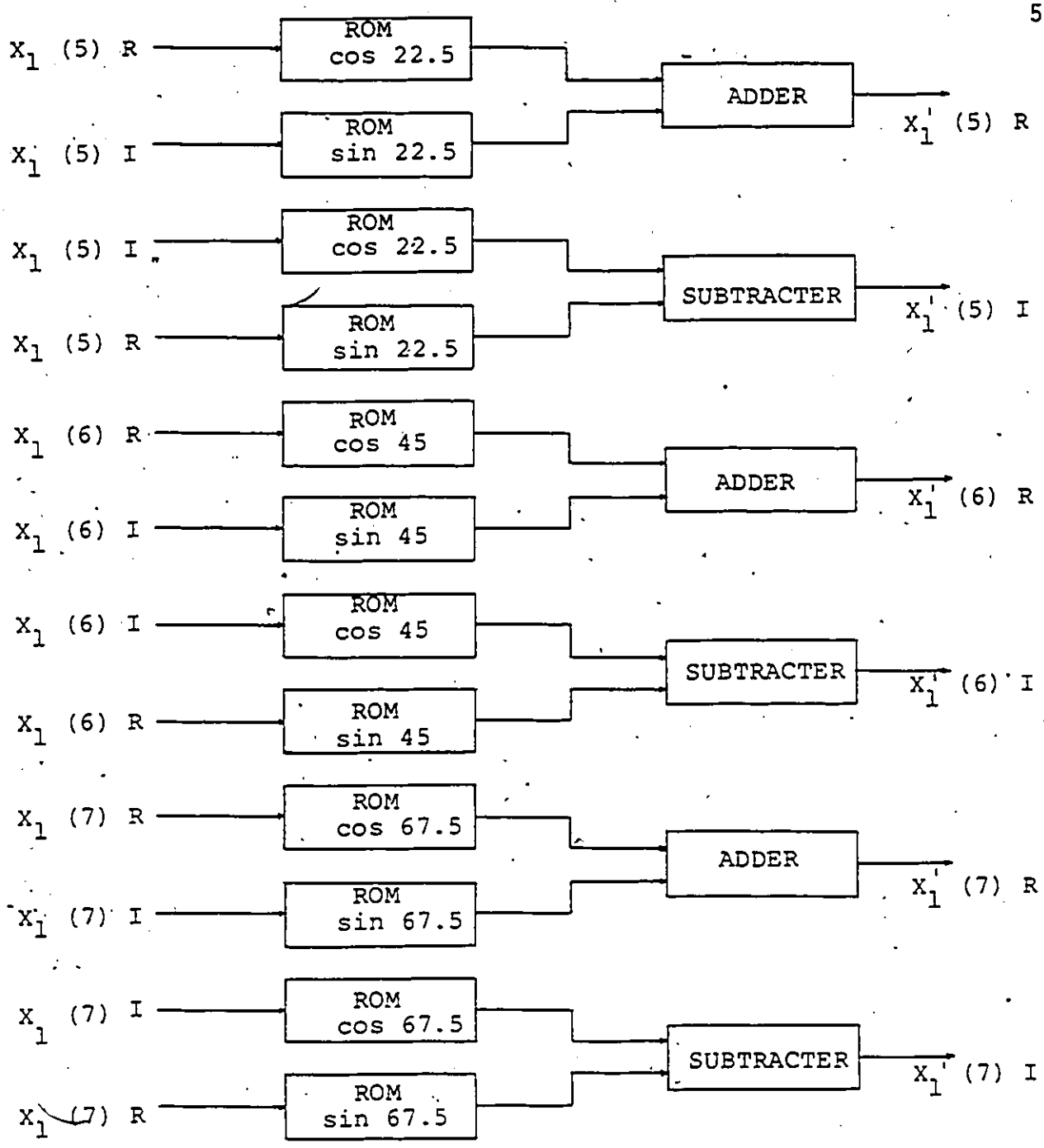


Fig. 3.8 ROM MULTIPLIER CONFIGURATION FOR RADIX-4 FFT

$X_1(6)$, and $X_1(7)$. Similar configurations exist for multiplications of the other samples $X_1(9)$, $X_1(10)$, $X_1(11)$; and $X_1(13)$, $X_1(14)$, $X_1(15)$. The multiplication time includes ROM access time and addition or subtraction time to get the complex product. If we assume that ROM access time equals addition or subtraction time, then complex multiplication time is twice the addition time.

The properties of the radix-2 and radix-4 FFT algorithms have been reviewed. The application of the stored product ROM concept to the realization of FFTs has been considered. The basic 16 point FFT processing blocks have been described. These arithmetic processing elements act as building blocks in realizing low order and higher order FFT processors as will be explained in the next two chapters.

Chapter IV

HIGH SPEED LOW ORDER FFT PROCESSORS

4.1 INTRODUCTION

Two methods of realizing real time low order FFT processors are considered in this chapter. (1) Serial Pipeline FFT and (2) Parallel Pipeline FFT. The term 'pipeline' is commonly used in modern signal processing terminology. Let us assume a signal processing requirement, where a similar arithmetic operation is repetitively performed several times in order to complete a given task. A single piece of hardware may be timeshared to perform all the similar operations, or many identical arithmetic elements may be used to perform each operation. A compromise of these two extremities is to use more than one hardware unit but less than the maximum number of arithmetic units. The data flow between successive hardware units in different stages is fixed. Consequently the data for some arithmetic elements depend on the result of previous arithmetic operations. If the data enters serially and few hardware units are processing the samples one after another, the hardware unit in the first stage does not have to complete the whole processing before sending its results to the succeeding unit in the next

stage. In other words, the data transfers through the hardware elements in a pipeline fashion just as in a pipeline where fluid is transferred through displacement. This is called Serial Pipeline FFT.

In a Parallel Pipeline FFT, the data enter and leave the processor wholly in parallel, but in a pipelined way from hardware element to the succeeding element. In this chapter, we describe hardware realization schemes for both serial pipeline and parallel pipeline methods. Each method is considered for both radix-2 16-point FFT and radix-4 16-point FFT algorithms. Hardware complexity and data-throughput rate comparisons are given.

4.2 SERIAL PIPELINE FFT PROCESSORS

A method of realizing a special purpose FFT processor that can perform digital spectrum analysis of wideband radar signals was reported [52]. Due to the nature of the data flow in the FFT processor, it was called pipeline FFT, in which the data was fed continuously (serially) to the processor. The processor consists of several identical processing blocks, one (or more) per stage of the FFT, working in parallel to achieve high data throughput rates. The principles of the operation of radix-2 16-point serial pipeline FFT and radix-4 16-point serial pipeline FFT processors, with their respective timing diagrams are described in detail in this section.

4.2.1 Radix-2 Serial Pipeline FFT Processor

In the 16-point radix-2 serial pipeline FFT configuration (Fig.4.1), the input sequence is sequentially stored in the input buffer memory. The input buffer memory is divided into four blocks, each consisting of 8 memory locations. Once the memory blocks (a) and (b) are filled, further in coming samples will be stored in the memory blocks (c) and (d). The AU in the first stage takes two samples at a time from the memory blocks (a) and (b) and performs the necessary additions/subtractions and then delivers the output to the next stage. Then the AU takes the second pair of samples and processes them. This procedure repeats until all the eight sets of two samples each are processed. Then the first stage AU starts the processing of the next set of 16 samples from the memory blocks (c) and (d). During this time the memory blocks (a) and (b) are filled with new samples.

The AU of the second stage does not have to wait till all the samples in the first stage are processed. In a pipelined architecture, the data flow is controlled such that the AU in the second stage begins its operation even if not all the samples are processed in the first stage. Similarly the processor begins its operation in the third and fourth stages before the completion in the second and third stages respectively. This procedure becomes clear with the help of a timing diagram.

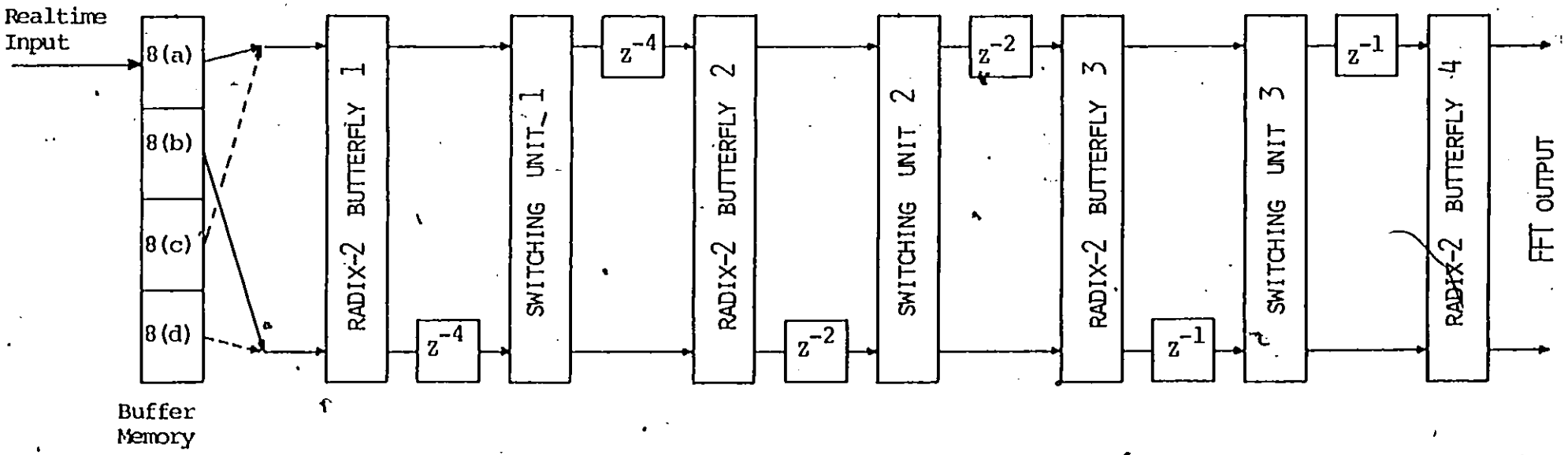


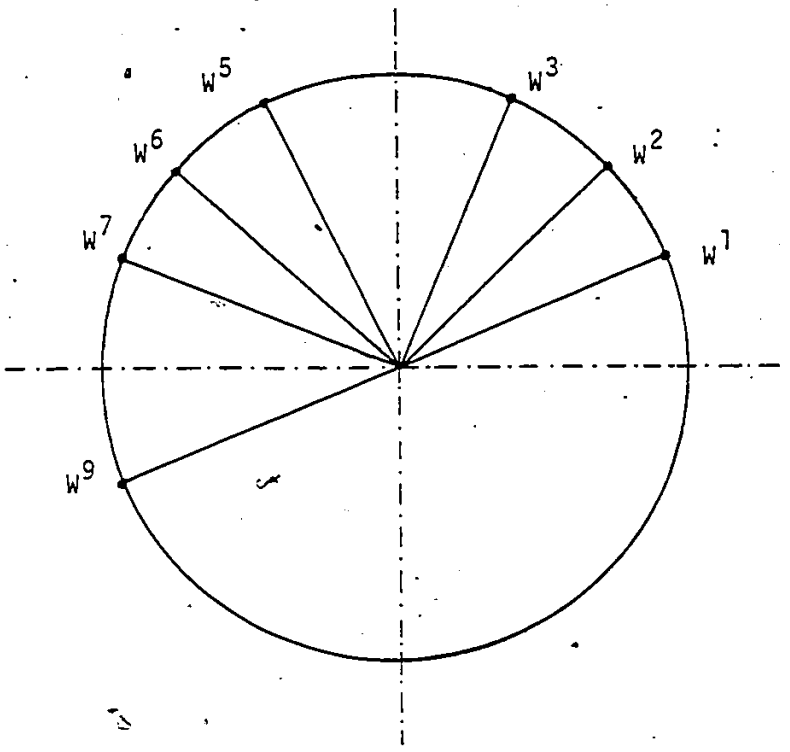
FIG.4.1 RADIX-2 16 POINT SERIAL PIPELINE FFT

However due to the extra delays involved in the multiplication and to achieve the same data throughput rate at all stages of the processor, an extra AU is included in the third and fourth stages of the processor. The required number of ROMs is reduced due to the symmetry of the twiddle factors in the Z-plane as shown in Fig. 4.2.

In the third stage, a single multiplier (ROM) configuration (for the twiddle factor W^2) is required. In the fourth stage, three separate ROM configurations (for the twiddles W^1, W^2, W^3) are required. As shown in Fig. 4.2, multiplications by W^5, W^6, W^7 are nothing but multiplications by W^1, W^2, W^3 respectively with some modification in the input connection to the AU (due to the term -j). Consequently separate ROM configurations for W^5, W^6, W^7 are not required. There are no effective multiplications involved in the butterfly calculation in the first and second stages.

4.2.2 Timing diagram for 16-point radix-2 FFT

The detailed description of data flow operations in each stage of the 16-point radix-2 decimation-in-time FFT is given with the help of a timing diagram. The timing diagram gives an estimate of total processing delay, and processing delay at each stage. It is also useful in interpreting where the arithmetic operations take place with respect to the location and hence time. The explanation is given for each stage of 16-point radix-2 FFT.



$W^5 = (-j)\ W^1$

$W^6 = (-j)\ W^2$

$W^7 = (-j)\ W^3$

$W^9 = (-1)\ W^1$

where $W = \exp(-j2\pi/16)$

$W^4 = -j$

$W^8 = -1$

Fig. 4.2 Z-plane Representation of 16 point FFT Twiddles

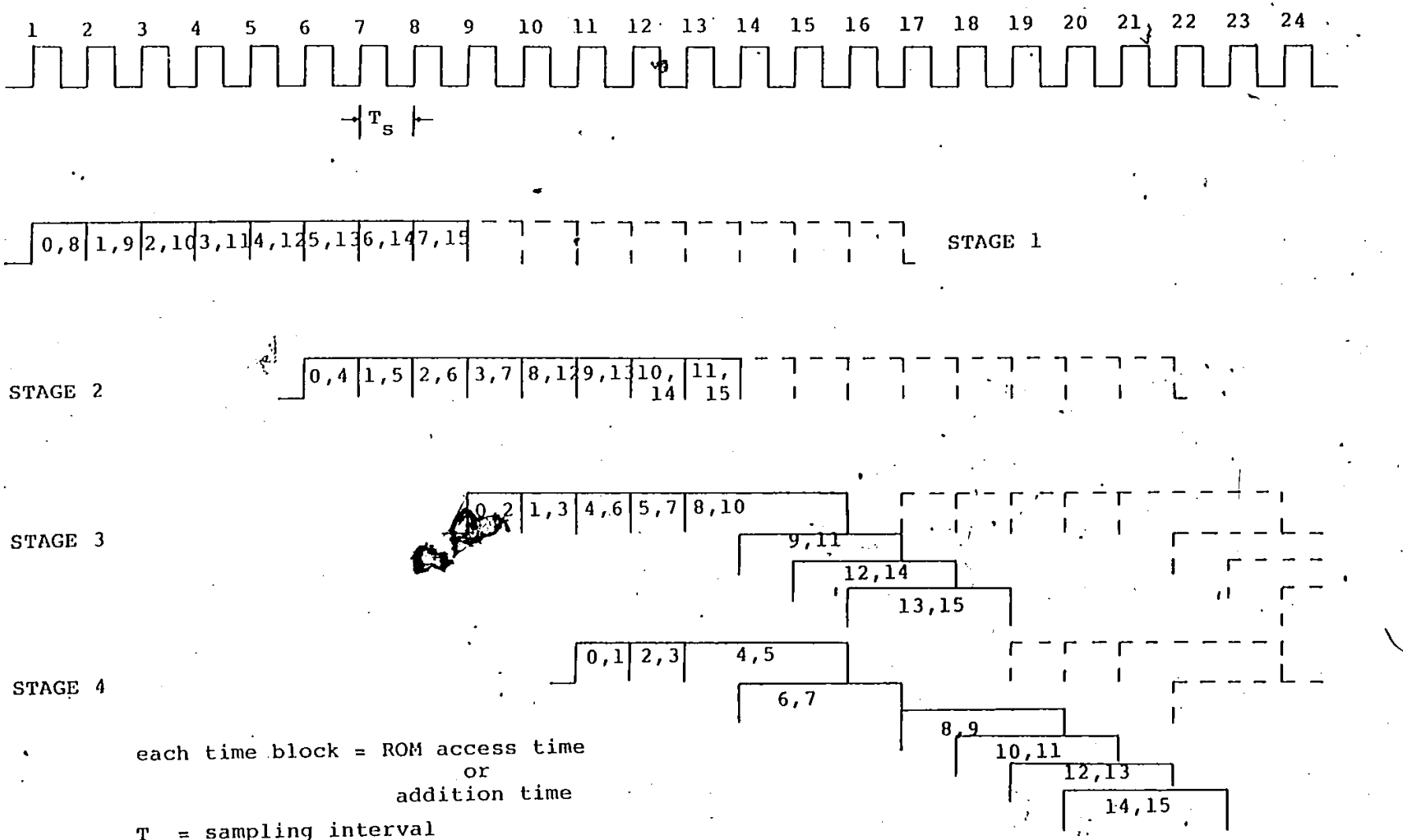


Fig. 4.3 TIMING DIAGRAM FOR 16-POINT RADIX-2 PIPELINE FFT

The computational clock timing sequence is shown in a diagram(Fig.4.3). The digital ICs in the processor will respond to the leading edge of the clock pulse. The interval between the leading edges of two successive clock pulses is equal to the sampling interval of the input signal. In the case of radix-2 serial pipeline FFT, the processor architecture is organized such that two samples are processed at a time. It is also assumed that the ROM access time is equal to the parallel addition time, so that the data flows through the hardware elements to maximize the efficiency of the processor. Hence the processor speed is entirely determined by a single addition time (or ROM access time) which leads to an optimum data throughput rate.

4.2.2.1 First Stage

At the start (leading edge) of the first clock pulse, the adder unit (AU) in the first stage takes the complex samples X(0) and X(8) and performs complex additions/subtractions and gives the output just before the start of the second clock pulse. In the timing diagram the latch time is not shown explicitly, but the addition time includes the latch time also. At the start of the second clock pulse the AU takes samples X(1) and X(9) and the process repeats till all the pairs of samples are processed. This happens at the end of the eighth clock interval. At this point, the processing in the first stage is completed.

4.2.2.2 Second Stage

Because of the pipelined architecture of the processor and its predetermined control mechanism, the processing of the samples in the second stage begins even before the completion of the processing of the same samples in the first stage. All the output samples of the first stage are subjected to some time delay as shown in the block diagram of pipeline FFT. After switching (Fig.4.4), the sequence of the samples changes. The AU in the second stage will start processing at the leading edge of sixth clock pulse, because the samples X(0) and X(4) will be available at the first stage output after only five clock intervals. The numbers that are shown in the time blocks in the timing diagram represent the corresponding samples. The processing in the second stage is completed after thirteen clock intervals.

4.2.2.3 Third Stage

The second stage outputs are subjected to an appropriate time delay and switching. The samples X(0), X(1),....., X(7) in the third stage are not to be multiplied. The samples X(8), X(9),....., X(15) require multiplications by the twiddle factors W^2 and W^6. The AU in the third stage starts processing at the beginning of ninth clock pulse, because the samples X(0) and X(2) are available at the second stage output only after the eighth clock interval. The procedure

0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	.....	Input samples
0	1	2	3	4	5	6	7										
				8	9	10	11	12	13	14	15					.....	Switching Unit 1 input
0	1	2	3	8	9	10	11										
				4	5	6	7	12	13	14	15					.....	Switching Unit 1 output
0	1	2	3	8	9	10	11										
		4	5	6	7	12	13	14	15							.....	Switching Unit 2 input
0	1	4	5	8	9	12	13										
		2	3	6	7	10	11	14	15							.....	Switching Unit 2 output
0	1	4	5	8	9	12	13										
	2	3	6	7	10	11	14	15								.....	Switching Unit 3 input
0	2	4	6	8	10	12	14										
	1	3	5	7	9	11	13	15								.....	Switching Unit 3 output

Fig.4.4 Switching Sequence for Radix-2 16 point Serial Pipeline FFT

repeats similar to the one in first and second stages, but it differs at the beginning of the thirteenth clock pulse. The real and imaginary parts of X(10) address the ROMs to get the products, and these products are added during the fourteenth clock interval. Now the AU takes the real and imaginary parts of X(8), that are latched during the thirteenth and fourteenth clock intervals, and the twiddled result of X(10). So, three clock intervals are required for the processing of samples X(8) and X(10). The sample X(11) can address the ROMs at the beginning of fourteenth clock pulse and this pipelining continues till the end of eighteenth clock interval. In the first and second stages, eight sampling intervals are required for complete processing, whereas in the third stage, ten sampling intervals are needed. The dotted lines represent the processing of the next set of 16 samples. The AU in the third stage is supposed to take the next set of samples at the seventeenth clock pulse. But the AU is involved in processing the samples X(12) and X(14) during that instant. During the eighteenth clock interval it is engaged in processing X(13) and X(15). Therefore to increase the speed of the processor an extra AU should be included which takes the next set of samples at the beginning of seventeenth clock pulse.

4.2.2.4 Fourth Stage

Here the samples X(0),....., X(3) do not require multiplications. Multiplications are required for the samples X(5), X(7), X(9), X(11), X(13), and X(15). The processing of samples X(4),....., X(7) is similar to the processing of the samples X(8), ..., X(11) of third stage. It is expected that processing of X(8) and X(9) begins at the start of the fifteenth clock pulse, but it is not possible because the samples X(8) and X(9) are available only at the end of fifteenth and sixteenth clock intervals respectively. So, the processing of X(8) and X(9) begins only at the start of seventeenth clock pulse. After that the procedure is straightforward. Again the processor requires an extra AU at the fourth stage. By the end of twenty-second clock interval the four-stage processing of the first set of 16 samples is completed.

If the clock time is 100 nsec, it takes 2.2 usec to get the first 16 point spectral frame, there after, for every 0.8 usec a spectral frame will be available at the output.

4.2.3 Radix-4 Serial Pipeline FFT Processor

The principle of operation of radix-4 serial pipeline FFT is similar to that of radix-2 serial pipeline FFT processor. Fig.4.5 shows a block diagram of the radix-4 pipeline FFT.

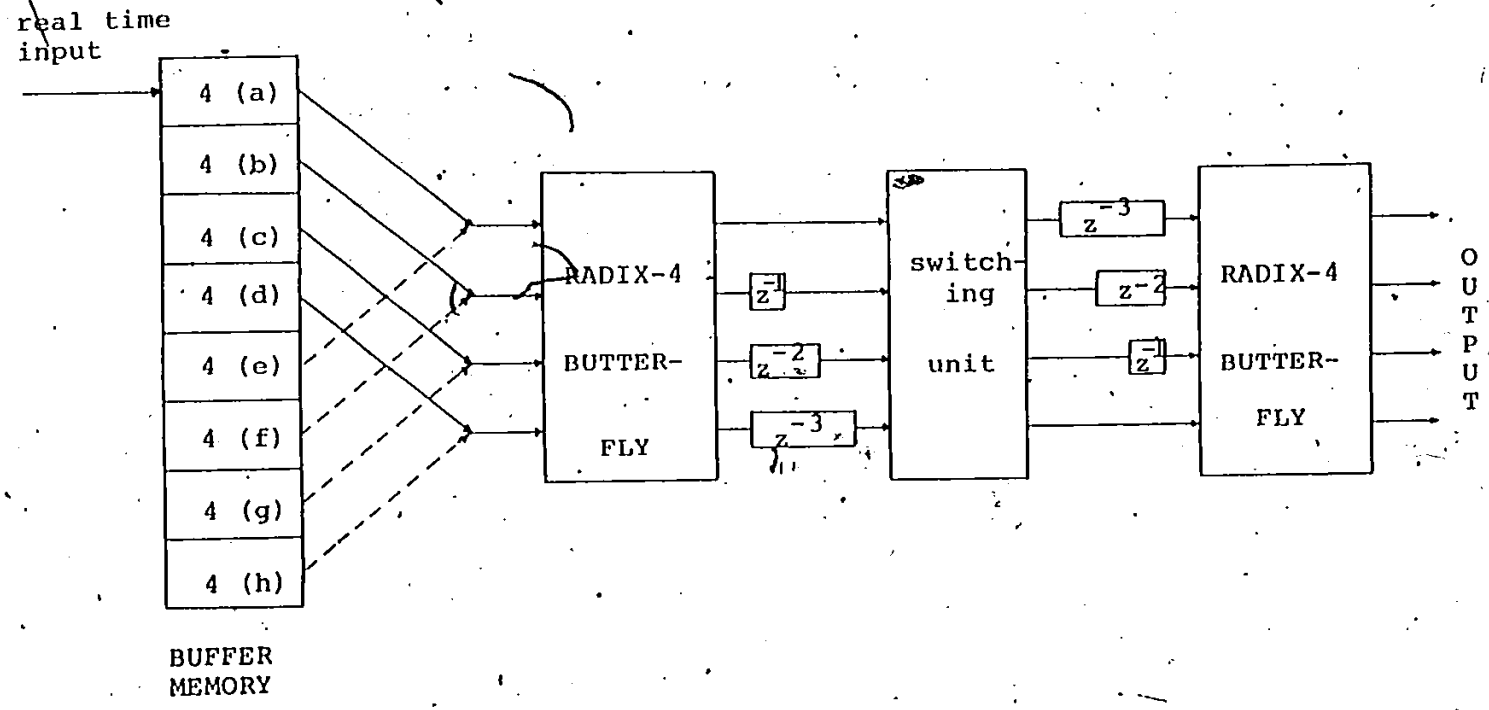


Fig. 4.5 RADIX-4 16-POINT PIPELINE FFT (serial)

It is of the same general form as radix-2 FFT but each of the basic elements (delays, commutators and butterflies) are now geared to radix-4 operations. Thus the radix-4 butterfly performs three complex multiplications and eight complex additions. 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.

Here again, due to the symmetry of the twiddle factors in the Z-plane as shown in Fig. 4.2, a single ROM multiplier array can be time shared to perform all the necessary multiplications. ROMs for the twiddle factors W^1, W^2, W^3 are required. However due to the simultaneous multiplications of the samples X(9) and X(11) by W^2 and W^6, an additional set of four ROMs for the twiddle factor W^6 is required.

4.2.4 Timing diagram for 16-point radix-4 FFT

We describe the operation of 16 point radix-4 DIT FFT with the timing diagram of Fig. 4.6. The description is given for the two stages of the radix-4 FFT. The computational clock sequence is shown in the timing diagram. In a 16 point radix-4 FFT, there are no multiplications in the first stage and two stages of additions/subtractions are required in the adder units (AUs). Thus the AU operation requires two clock intervals. The detailed description of each stage operation is given below.

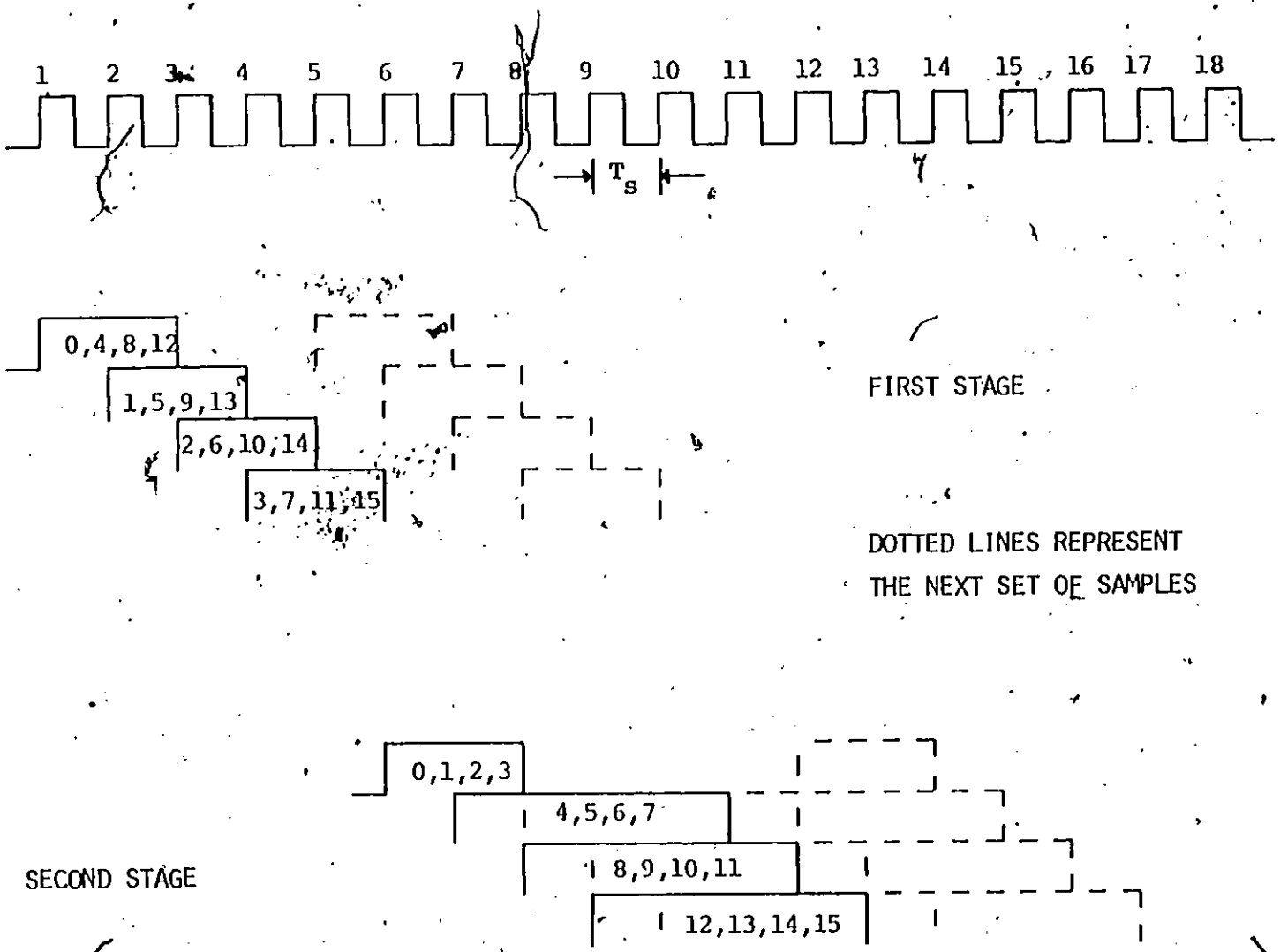


FIG. 4.6 TIMING DIAGRAM FOR 16 POINT RADIX-4 SERIAL PIPELINE FFT

4.2.4.1 First Stage

At the start (leading edge) of the first clock pulse, the AU takes the samples X(0), X(4), X(8) and X(12) and performs additions/subtractions. This process requires two clock intervals. In the figure it is shown that the samples X(0), X(4), X(8) and X(12) are processed in the first stage by the end of second clock interval. The AU can take the next set of four samples, X(1), X(5), X(9) and X(13) at the beginning of second clock pulse, because of the two stages of independent additions and subtractions in the AU and it is possible to control the data flow in a pipelined fashion. The sixteen adder/subtracter units in AU are arranged in two columns (Fig. 3.7) of eight units each. So, once the data is processed by the first column adders/subtracters then the first column adders/ subtracters can take the next set of four samples. That is the reason why it is shown in the timing diagram the processing of samples X(1), X(5), X(9) and X(13) begins at the start of the second clock pulse. This pipelining operation continues till the end of fifth clock interval. At that instant, the processing of the first set of 16 samples in the first stage is completed.

4.2.4.2 Second Stage

The first stage outputs are subjected to an appropriate time delay as shown in the block diagram of radix-4 serial

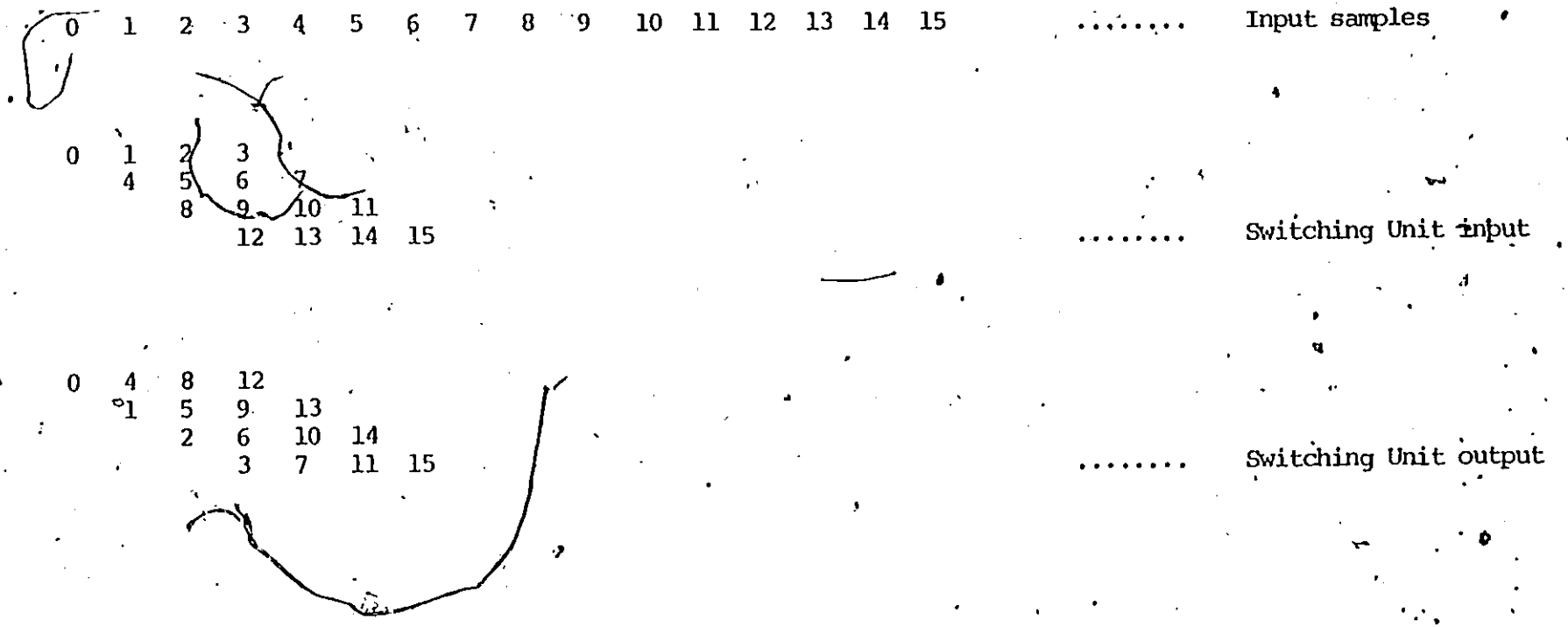


Fig. 4.7 Switching Sequence for Radix-4 16 point Serial Pipeline FFT

pipeline FFT and after switching (Fig.4.7), the samples appear in a different sequence.

At the beginning of sixth clock pulse, the AU starts processing the samples X(0), X(1), X(2) and X(3). The samples X(5), X(6), X(7); X(9), X(10), X(11); and X(13), X(14) and X(15) require multiplications by the twiddle factors. The processing of X(4), X(5), X(6) and X(7) begins during the seventh clock interval. As shown in Fig.4.6, the real and imaginary parts of X(5), X(6) and X(7) address the ROMs during the seventh clock interval. The first block of time (separated by a dotted line) in the second stage except for samples X(0), X(1), X(2) and X(3) represents a ROM access time. The products available at the outputs of ROMs are added/subtracted during eighth clock interval. By the end of eighth clock interval, the complex products of samples X(5), X(6) and X(7) and the corresponding twiddle factors are available. Now these product terms and the sample X(4) which is latched during the seventh and eighth clock intervals are applied to the AU at the beginning of ninth clock pulse. Since the AU operation requires two clock intervals, the samples X(4), X(5), X(6) and X(7) are processed completely by the end of tenth clock interval.

The processing of X(8), X(9),....., X(15) is done as shown in the diagram. Again the data flows in a pipelined way. So, by the end of twelfth clock interval, the two

stage processing of first set of 16 samples is completed. The dotted lines in the diagram represent the processing of the next set of 16 samples. In the first stage, the adders/subtracters in the first column of the AU are free by the end of fourth clock interval. So the first column of AU adders/subtracters can take X(0), X(4), X(8) and X(12) from the next set of sixteen samples at the beginning of fifth clock pulse.

It is expected that the second stage processing of the second set of 16 samples begins at the start of tenth clock pulse. However the samples X(0), X(1), X(2) and X(3) are shifted till the end of eleventh clock interval. The AU starts processing these samples at the beginning of the twelfth clock pulse. The reason for shifting X(0), X(1), X(2) and X(3) is that these samples require AU operation only but the AU is engaged during that period (tenth, eleventh, and twelfth clock intervals) in processing X(8), X(9),....., X(15) of the previous set of 16 samples. However, the adders/subtracters in the first column of the AU are free by the end of eleventh clock interval. So, the AU can start processing X(0), X(1), X(2) and X(3) of the next set of 16 samples at the beginning of twelfth clock pulse. The rest of the processing of the next set of sixteen samples is same as that of first set. So in the second stage an extra AU is not required for the radix-4 serial pipelined realization.

4.3 PARALLEL PIPELINE FFT PROCESSORS

In a serial pipeline FFT, a minimum of one butterfly unit and a maximum of two butterfly units per stage of FFT have been used. In general, a single unit is timeshared to perform all the butterfly operations in a FFT stage. If all the butterflies are realized separately and the data enter the processor in a parallel fashion and transfer to the succeeding stages in a pipelined way, maximum data throughput rates can be achieved at the cost of increasing hardware complexity. This type of realization is named parallel pipeline FFT. This approach is not preferable for higher order FFT processors because of the drastic increase in complexity, but is well suited for low order ($N \leq 32$) FFT processors. In this section, we describe two realizations, (1) radix-2 16-point FFT, and (2) radix-4 16-point FFT.

4.3.1 Radix-2 Parallel Pipeline FFT Processor

In a radix-2 DIT parallel pipeline FFT (Fig.4.8), the arithmetic operations in each stage can be performed simultaneously with dedicated AUs for each required operation. The number of butterflies in a 16 point radix-2 FFT is $N/2 \log_2 N$. For N=16, this number is 32. So this processor consists of 32 separate AUs, including 4 separate ROM configurations in the third stage and 6 separate ROM configurations in the fourth stage. Each stage is separated by an interconnection

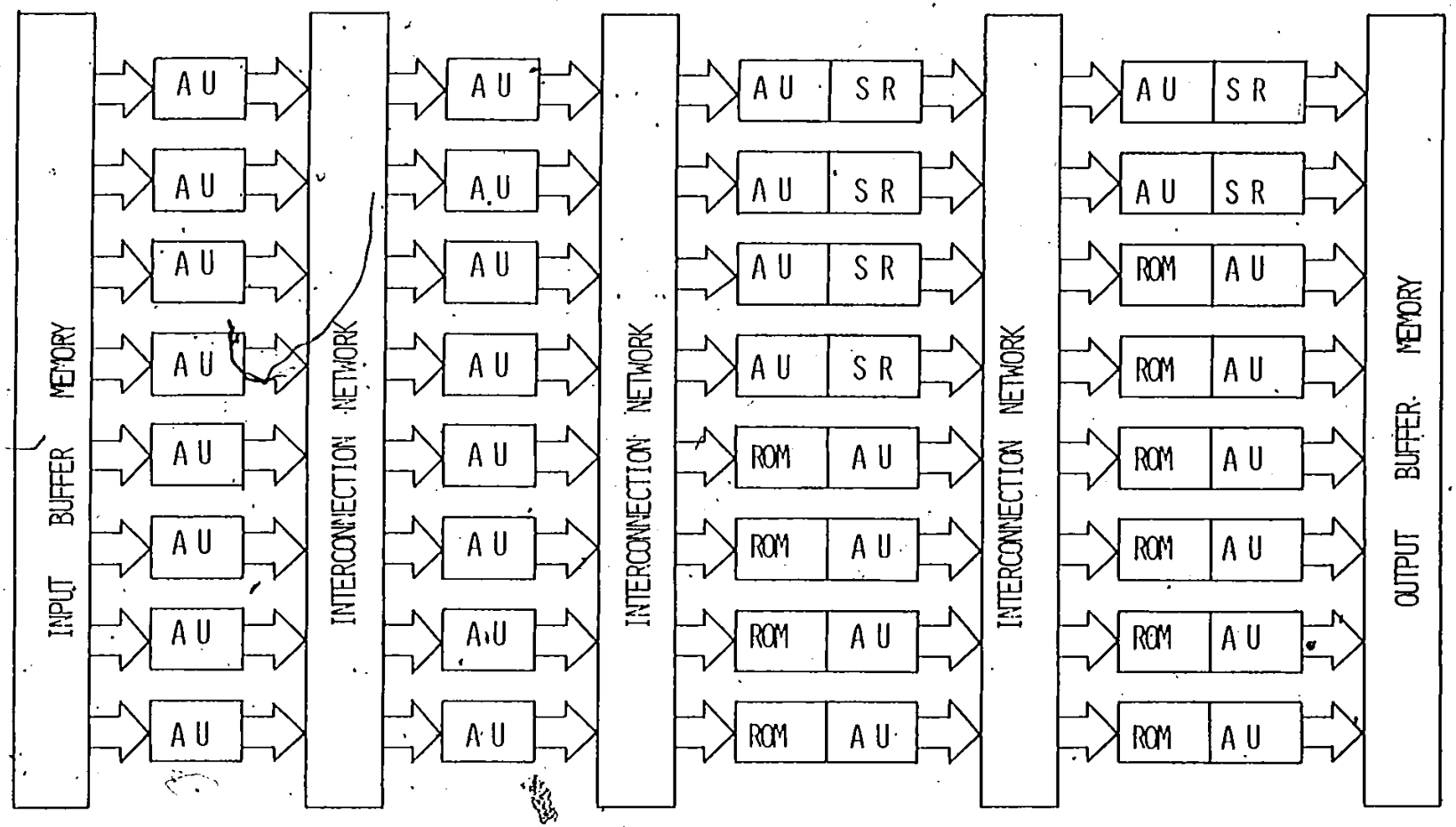


FIG.4.8 RADIX-2 16 POINT DECIMATION-IN-TIME PARALLEL PIPELINE FFT

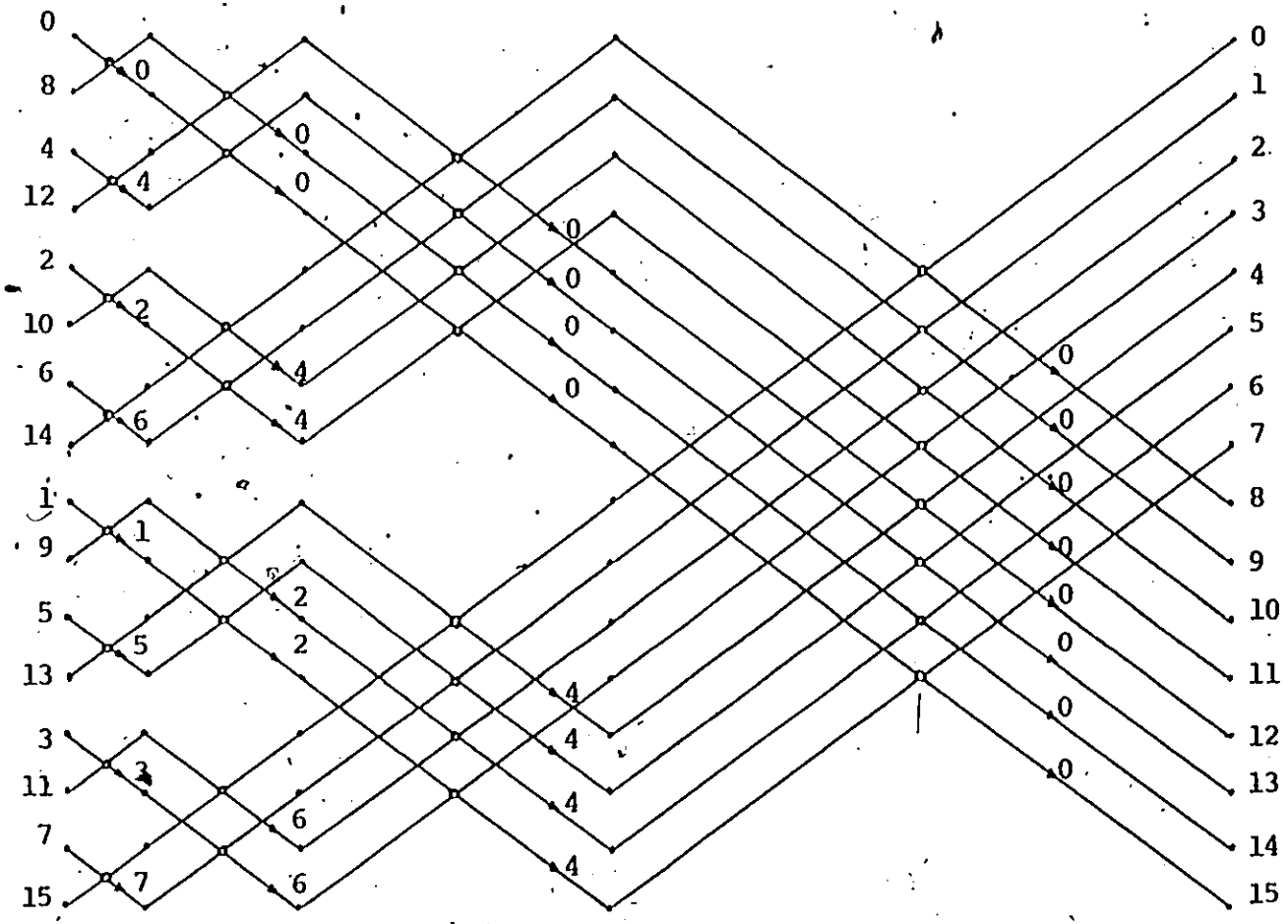


FIG. 4.9 16 POINT RADIX-2 DECIMATION-IN-FREQUENCY ALGORITHM

AU - ADDER UNIT SR - SHIFT REGISTER

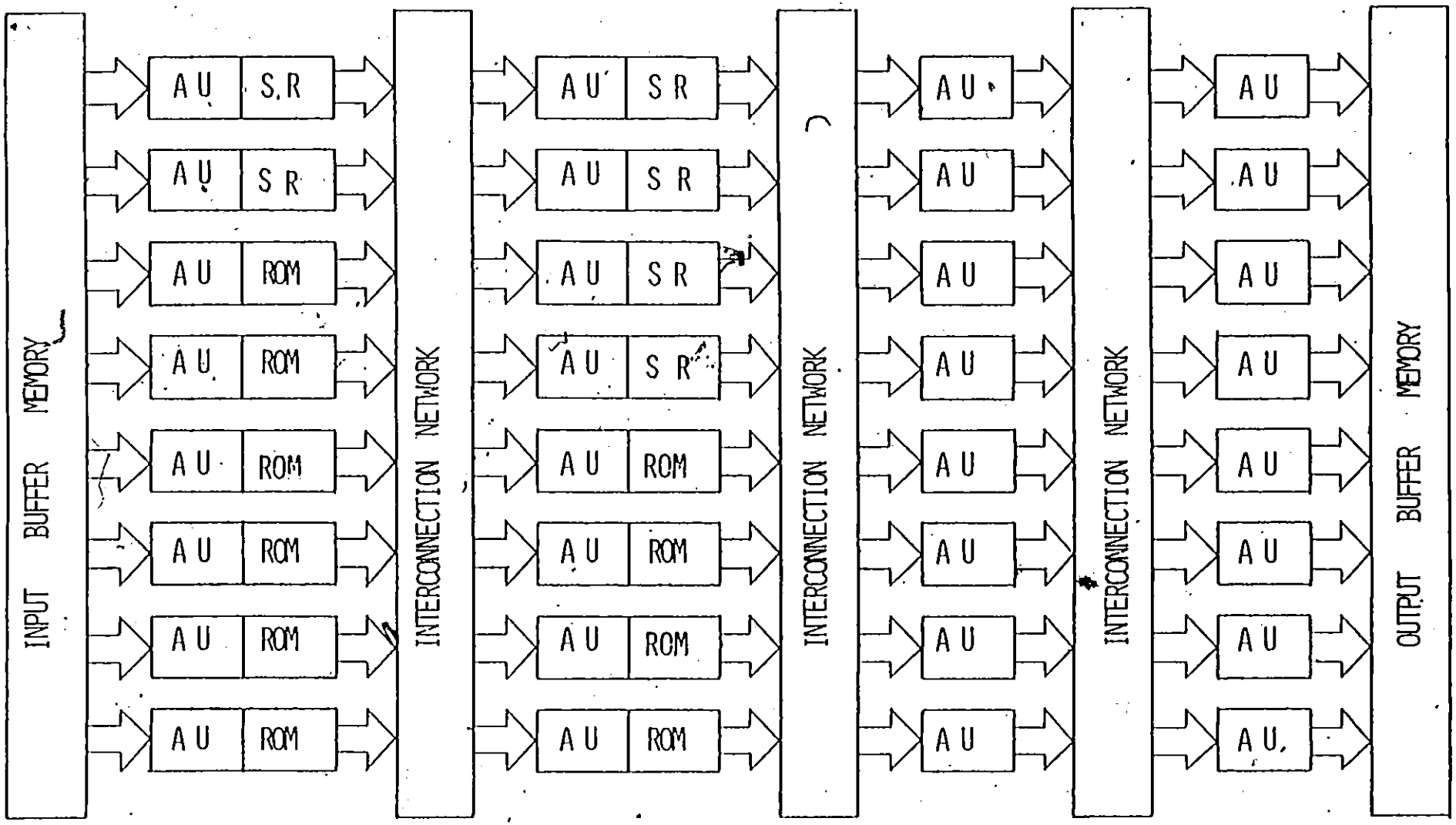


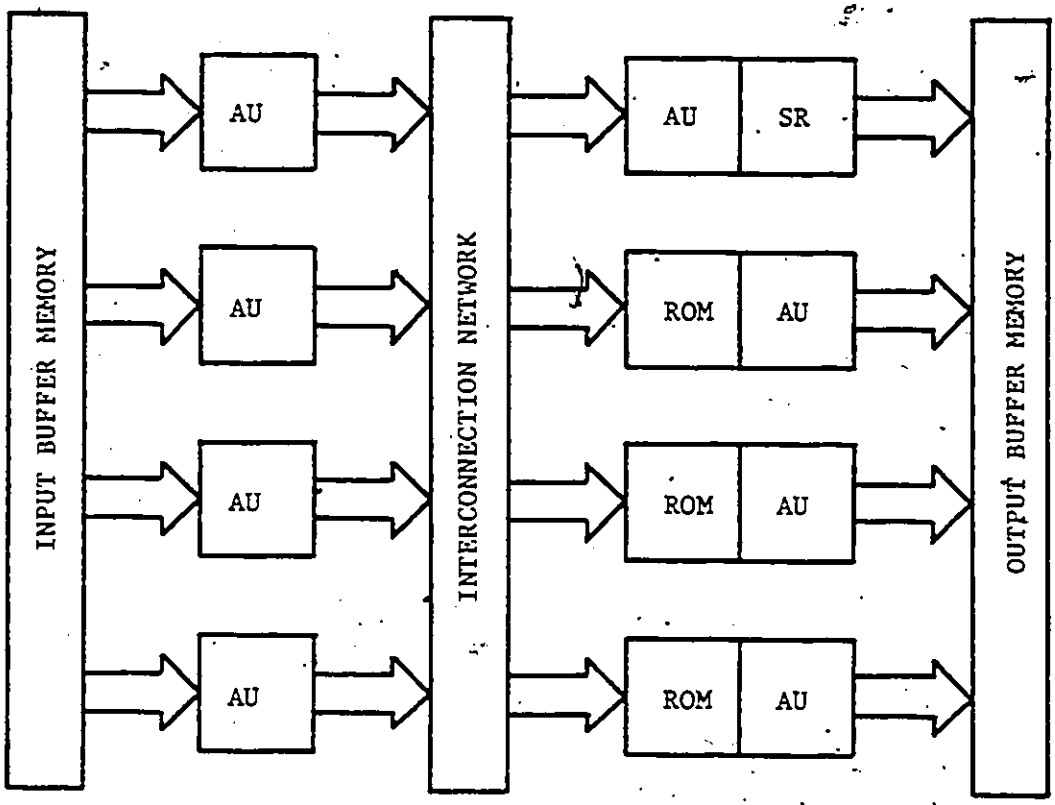
Fig. 4.10 RADIX-2 16 POINT DECIMATION-IN-FREQUENCY PARALLEL PIPELINE FFT

network to connect the outputs of each stage to the inputs of the succeeding stage in a well defined manner as shown in Fig. 3.1. A few shift registers (SRs) are deliberately introduced in the third and fourth stages to delay the samples that do not require multiplications. This is to ensure that all the samples are available at the output simultaneously. The processor reads 16 complex samples from the input buffer memory during the first clock pulse and, after processing, writes the results into the output buffer memory during the ninth clock pulse. Though this configuration demands more hardware, it can operate at extremely high input clock rates. It requires very simple control circuitry compared to the serial pipeline configuration. Furthermore, it does not require a separate bit-reversal operation at the output, because the output terminals can be routed such that they are written in the memory as normally ordered outputs.

Another method of parallel pipeline FFT implementation is based on decimation-in-frequency (DIF) algorithm (Fig. 4.9). In a DIF algorithm, multiplications are distributed in the earlier stages of the FFT algorithm. This implementation is shown in Fig. 4.10.

4.3.2 Radix-4 Parallel Pipeline FFT Processor

A realization of the radix-4 parallel pipeline FFT processor is shown in Fig. 4.11. There are eight butterflies,



AU - Adder Unit SR - Shift Register

Fig. 4.11 Radix-4 16 point Parallel Pipeline FFT

of increased complexity, capable of dealing with 4 complex inputs at a time, as compared to 32 butterflies for radix-2 case. Instead of timesharing a single AU and a ROM array multiplier as in a serial pipeline FFT processor, here a separate AU for each butterfly and three separate ROM array multipliers, similar to the one shown in Fig. 3.8 are used. The advantages of fully implemented radix-4 FFT processor are same as that of radix-2 FFT.

The 16-point radix-4 DIF algorithm is identically the same as the DIT algorithm. In the case of DIT algorithm, multiplications take place before the second stage AU operations whereas for the DIF algorithm multiplications take place after the first stage AU operations. Since there are only two stages in a 16-point radix-4 FFT, the algorithm structure is effectively the same for both DIT and DIF cases. Hence a separate figure is not shown for the parallel pipeline DIF FFT. However, this identity is not applicable for $N \geq 64$.

After the initial processing delays, the radix-2 and radix-4 parallel pipeline FFT processors can deliver 16 samples per clock interval. The detailed comparison of the radix-2 and radix-4 serial pipeline and parallel pipeline hardware realizations in terms of data throughput rate and hardware complexity is described in the following sections.

4.4 SPEED AND INITIAL PROCESSING DELAY

By referring to the Fig.4.3, it is clear that the radix-2 serial pipeline processor delivers 16 samples in eight clock intervals, whereas for the radix-4 case (Fig.4.6) only four clock intervals are required. The radix-2 parallel pipeline processor operation is four times faster than the radix-4 serial pipeline processor. As described earlier, in the radix-2 and radix-4 parallel pipeline processors, 16 samples will be available at the output for each clock interval.

The initial processing delay of the radix-2 serial pipeline processor for the first set of 16 samples is equal to 22 clock intervals. Similarly, the initial processing delay of radix-4 pipeline processor is equal to 12 clock intervals. In the radix-2 and radix-4 parallel pipeline configurations it takes eight and six clock intervals respectively to get the first output. The data throughput rate comparison is given in Table 4.1.

4.5 HARDWARE COMPLEXITY

The number of parallel adders (including parallel subtracters) for the radix-2 serial pipeline processor is 28. There are six AUs (including an extra AU in each of third and fourth stages) in the radix-2 serial pipeline processor, each consists of four adders/subtracters. And also four adders/subtracters are required in the multiplier

TABLE:4.1 DATA THROUGHPUT RATE OF LOW ORDER FFT PROCESSORS

PROCESSOR	INITIAL PROCESSING DELAY	PROCESSING RATE
16-Point Radix-2 Serial Pipeline	22 Sampling Intervals	2 Samples/Clock Interval
16-Point Radix-4 Serial Pipeline	12 Sampling Intervals	4 Samples/Clock Interval
16-Point Radix-2 Parallel Pipeline	8 Sampling Intervals	16 Samples/Clock Interval
16-Point Radix-4 Parallel Pipeline	6 Sampling Intervals	16 Samples/Clock Interval

configurations in the third and fourth stages. So the total number is 28. In the radix-4 serial pipeline processor an extra AU is not required in the second stage. The AU in each stage consists of 16 adders/subtracters and 6 adders/subtracters are required in the multiplier configuration in the second stage. So the number of adders in a 16-point radix-4 serial pipeline FFT is 38.

In the radix-2 parallel pipeline processor, 32 independent AUs are used. Each AU consists of four adders/subtracters. There are ten independent multiplier configurations each consisting of two adders/subtracters. The total number works out to be 148. Similarly for the radix-4 parallel pipeline processor, the total number of parallel adders/subtracters will be 144.

Assuming p bit input signal word length and p bit ROM product wordlength, then each ROM size will be $(2^p \cdot p)$. Each multiplier requires four ROMs. In the radix-2 serial pipeline processor, the multipliers are timeshared in the third and fourth stages. So three independent multiplier configurations in the fourth stage and one multiplier configuration in the third stage are required. Since each multiplier configuration requires 4 ROMs, sixteen ROMs are necessary for the radix-2 pipeline processor. In the radix-4 serial pipeline processor, four different multiplier configurations in the second stage are required. Again, here 16 ROMs are

TABLE: 4.2 HARDWARE COMPLEXITY OF LOW ORDER FFT PROCESSORS

CIRCUIT ELEMENTS $p_{=}$ Signal/Product Wordlength	16-POINT RADIX-2 SERIAL PIPELINE	16-POINT RADIX-4 SERIAL PIPELINE	16-POINT RADIX-2 PARALLEL PIPELINE	16-POINT RADIX-4 PARALLEL PIPELINE
p-bit Adders	28	38	148	144
$(2^{P_{xp}})$ ROMs	16	16	28	24
p-bit Shift Registers	32	28	24	16

necessary. At this point, the advantage of the equality of $\cos(45^\circ)$ and $\sin(45^\circ)$ is not taken into account in the realization of the pipeline processors in order to elucidate the principle of the operation of pipelining in the timing diagram. If we take this advantage, the reduction in ROM size is not significant in the radix-2 and the radix-4 serial pipeline processors. But it reduces significantly the ROM size when implementing radix-2 and radix-4 parallel pipeline configurations. Hence the required number of ROMs respectively for the radix-2 and the radix-4 parallel pipeline processors are 28 and 24.

The shift registers (SRs) are necessary in the serial pipeline configurations, but in the parallel pipeline realization the SRs are deliberately introduced as explained earlier. To process complex data, twice the number of SRs (for real and imaginary) are required. The hardware complexity comparison is given in Table 4.2. The question of how to choose signal and product wordlengths for a given accuracy is addressed in the Appendix-A.

Methods of realizing high speed 16-point FFT processors have been described. A prototype parallel pipeline 16-point FFT has been designed and fabricated. Some of the practical considerations of this prototype are given in the Appendix-B. The realization methods of high order FFT processors by timesharing this parallel pipeline 16-point FFT unit are described in the next chapter.

Chapter V

HIGH ORDER/TWO-DIMENSIONAL FFT PROCESSORS

5.1 INTRODUCTION

In this chapter, we describe hardware realizations for higher order one-dimensional and low order two-dimensional FFT processors. These realizations are based on timesharing of a high speed low order (N=16) one-dimensional parallel pipeline FFT processor unit. The theory of generalized FFT is reviewed. For a two-dimensional array, the results of one-dimensional transforms of each row of samples are stored in a matrix transposer and the samples are read in columns for one-dimensional column transforms. A new method of matrix transposition for high speed realizations is described. Proposed higher order one-dimensional FFT processors are shown for 64 and 256 point transforms. Also a two-dimensional 16*16 point FFT processor is described. The data throughput rates and processing delays for all the possible realizations are given. And finally hardware requirements are outlined.

5.2 THEORY OF GENERALIZED FFT

The FFT computes the discrete Fourier transform (DFT) of N input variables x(0), x(1), ..., x(N-1) where, by definition,

$$X(k) = DFT\{x(n)\} = \sum_{n=0}^{N-1} x(n) \cdot W^{nk} \qquad n = 0,1, \ldots, N-1 \tag{5.1}$$

where $W = \exp(-j2\Pi/N)$

The FFT gains speed by factoring the composite number N into the product of integers:

$$N = r_1 r_2 r_3 \cdots r_m \tag{5.2}$$

Equation (5.2) corresponds to factoring the one-dimensional signal x(n) into an m-dimensional signal. For example, if N = 64 and m = 3, we might have $r_1 = r_2 = r_3 = 4$, and the 64-point transform is translated into a three-dimensional (4 x 4 x 4) transform. If N is a prime number so that factorization of N is not possible, the original signal can be usually be augmented with zeros, and a new composite value of N is attained.

A convenient way of performing the FFT algorithm is as follows. Begin by representing N as the product of two numbers, say, $N = N_1 N_2$. If N_1 and N_2 are both composite, they can further be broken down.

$$N_1 = N_{11} N_{12}, \quad N_2 = N_{21} N_{22}$$

In this way, each iteration consists of going through the mechanics of a two-dimensional transform to achieve a one-dimensional transform.

Let the one-dimensional sequence be represented as a two-dimensional array of numbers. Now, letting the column index be m ($m = 0, 1, 2, \ldots, M-1$) and the row index be ℓ ($\ell = 0, 1, 2, \ldots, N-1$), then we see that

$$n = M\ell + m \tag{5.3}$$

The DFT of this two-dimensional array of numbers is another two-dimensional array, with m and ℓ as the signal variables, let r and s be the transformed column and row variables, which when recomposed yield a single variable

$$k = Lr + s \tag{5.4}$$

The discrete Fourier transform samples $X(k) = X(s,r)$ can be expressed as the transform of $x(n) = x(\ell,m)$ by substituting equations (5.3) and (5.4) into the definition of the DFT {equation (5.1)} giving

$$X(k) = X(s,r) = \sum_{m=0}^{M-1} \sum_{\ell=0}^{L-1} x(\ell,m) \; W^{(M\ell+m)(Lr+s)} \tag{5.5}$$

Expanding $W^{(M\ell+m)(Lr+s)}$, observing that $W^{ML\ell r} = W^{N\ell r} = 1$, and properly associating indices with summation terms, we can rearrange equation (5.5) as

$$X(s,r) = \sum_{m=0}^{M-1} W^{Lmr} \; W^{ms} \; \underbrace{\sum_{\ell=0}^{L-1} x(\ell,m) \; W^{M s \ell}}_{q(s,m)} \tag{5.6}$$

Equation (5.6), if correctly interpreted, is a valid procedure for evaluating one-dimensional transforms via two-dimensional representations.

The L-fold interval sum is really the DFT of the mth column of the array having the kernel W^M. Thus, the first step in our computation procedure would be

(1) Compute the L-point DFT of each column.

The result is a function of s and m, with m going from 0 to M-1. Let the result be q(s,m). Now equation (5.6) can be written as

$$X(s,r) = \sum_{m=0}^{M-1} W^{Lmr} W^{ms} q(s,m) \qquad (5.7)$$

From equation (5.7), step 2 in the procedure is deduced to be

(2) Obtain a new array h(s,m) by multiplying every q(s,m) by its twiddle factor W^{ms}.

$$X(s,r) = \sum_{m=0}^{M-1} h(s,m) W^{Lmr} \qquad (5.8)$$

Equation (5.8) is recognized to be the M-point DFT of each row, with the row index. Thus the final step in the procedure is

(3) Compute the DFT of each row of the h(s,m) matrix, with W^L as kernal.

The procedure above is similar to the computation of the two-dimensional DFTs. In that case, one computes row DFTs followed by column DFTs to find the answer; step (2) is missing.

The separability of the higher-dimensional kernal causes higher-dimensional DFTs to require less computation than one-dimensional DFTs with the same number of points.

5.3 RANDOM ACCESS MEMORY ARRAY TRANSPOSER

The proposed method of high speed matrix transposition can be realized with the RAM array transposer (RAMAT). The RAM array consists of several small size RAMs arranged in rows and columns. Each row or column is activated during a clock pulse and the samples respectively are stored in a row or read from a column. By a simple address sequencing, it is shown that a single RAMAT can be used for alternative WRITE-IN and READ-OUT operations. The description of a generalized RAMAT of the order (M*N) is followed by an example of (4*4) RAMAT. A timing diagram for (4*4) transposer is also given in order to elucidate the principle.

5.3.1 M*N Matrix Transposer

In a two-dimensional array (M * N), if M and N represent the number of columns and rows respectively, then a RAMAT is formed with MN random access memories (RAMs), (M+N) latches, a N bit demultiplexer and a M bit demultiplexer as shown in Fig.5.1. The real parts of the first M samples of the input data to RAMAT are stored in the first row. While the real words are stored, the imaginary words of the same samples are latched (delayed) for the time equal to the RAM access time. After the real parts are stored, the imaginary data is also stored in the first row. Now the real parts of the second set of M samples are stored in the second row, then

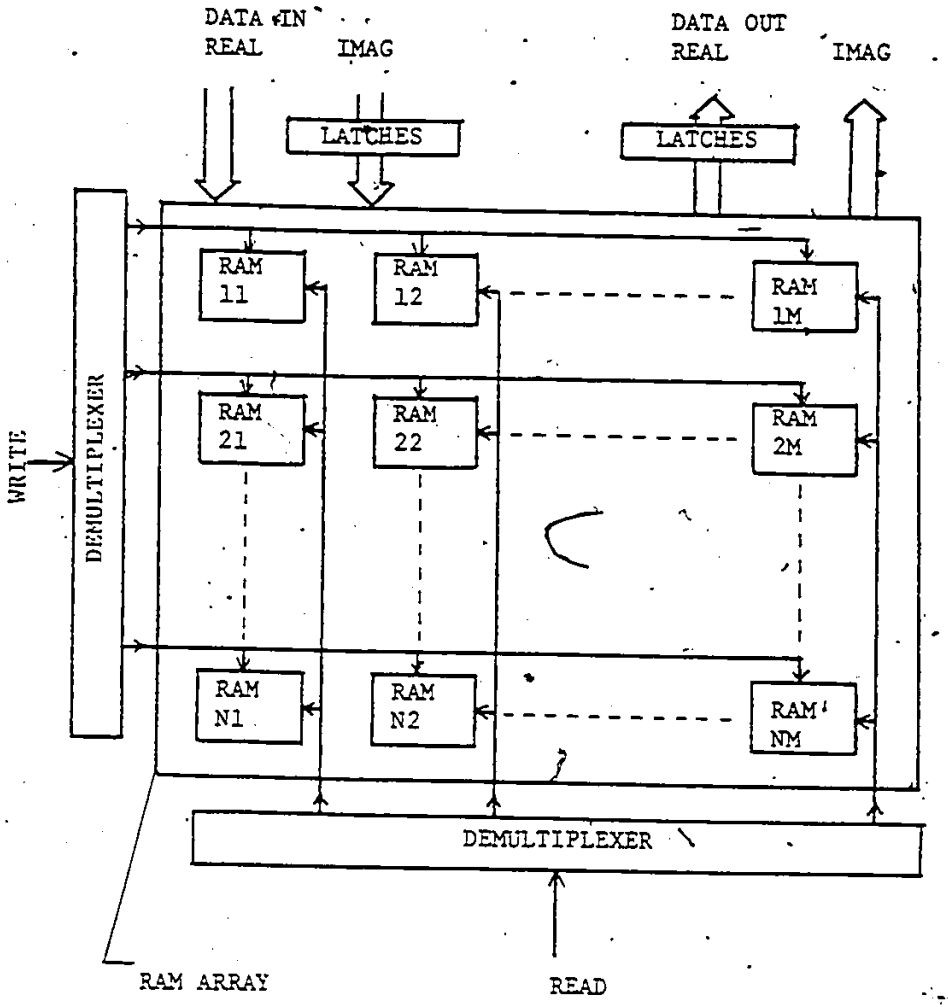


Fig.5.1 RAM Array Transposer

the imaginary data of the second set of M samples (that were latched during the previous clock interval) are stored in the second row. This procedure continues until all the RAMs are filled with MN complex samples. The 'write' control that is connected to the RAMs in each row is demultiplexed as shown in Fig.5.1. The 'read' control which connects the RAMs in each column is demultiplexed to fetch the data from the RAMAT in columns.

The real words of first N samples are read and are latched. Then the imaginary words of first N samples are read. The latched words and the presently available imaginary words constitute a set of N complex samples for possible further processing. This procedure repeats until all the column samples are read. Then the WRITE operations will be begun in the RAMAT for the next set of MN samples. An additional RAMAT is necessary for simultaneous READ and WRITE operations. However the need for an additional RAMAT can be avoided by an efficient address sequencing, as shown below.

5.3.2 4*4 Matrix Transposer

The proposed RAMAT can well be described by considering a 4*4 matrix transposition operation as shown in Fig. 5.2. The RAM array consists of 16 RAMs, arranged in 4 rows and 4 columns. The memory size of each RAM is (4*p), ie., it has

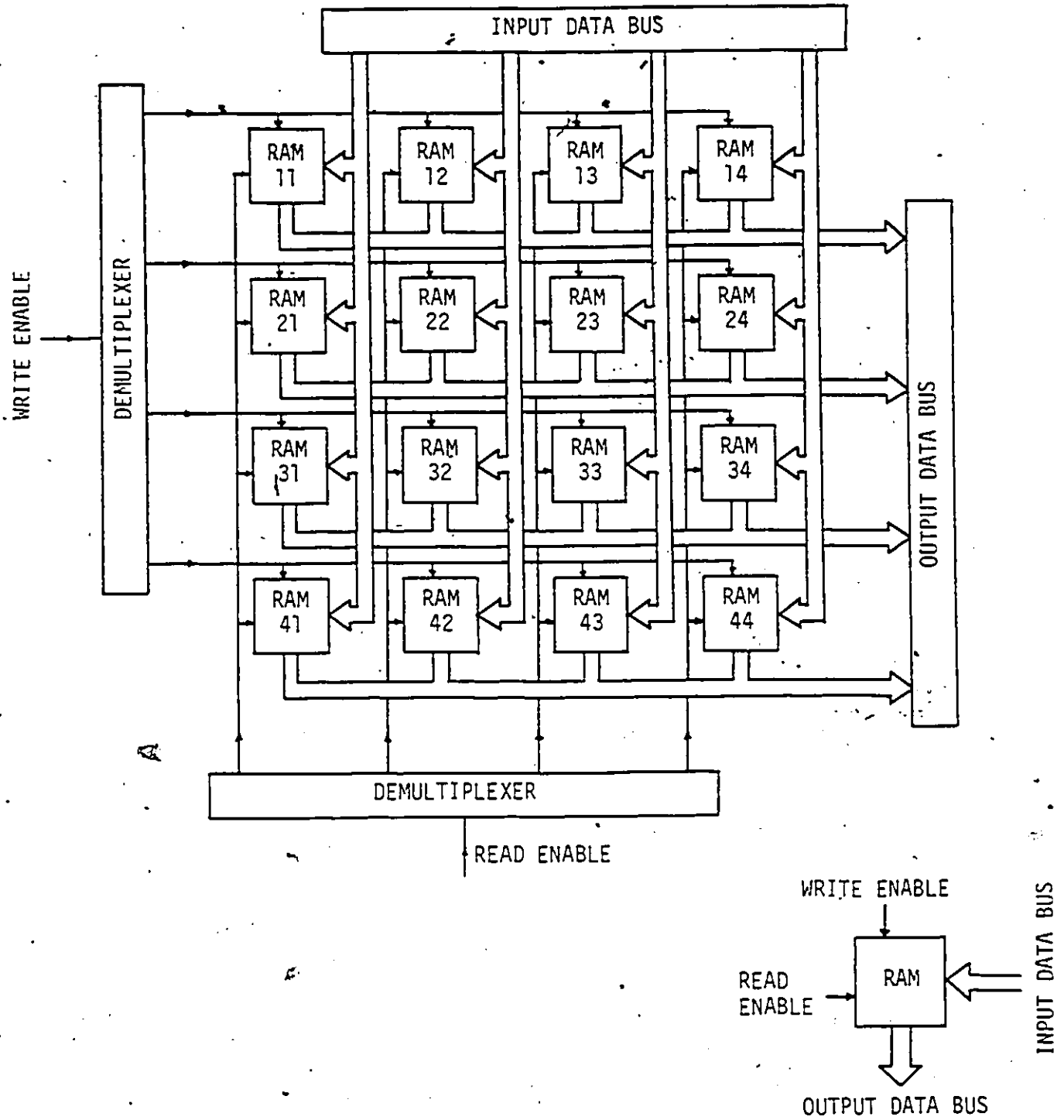


FIG. 5.2 A 4x4 MATRIX TRANSPOSER

four memory locations in which four different p-bit samples can be stored. . The address wordlength is 2 bits, because each RAM has four locations. The real and imaginary parts of the same sample will be stored in the same RAM (address locations 00 for real and 01 for imaginary). The principle of time-division multiplexing is employed for the two main operations, (1) WRITE-IN-ROWS (2) READ-OUT-COLUMNS.

5.3.3 Timing diagram for 4*4 matrix transposer

The timing diagram (Fig.5.3) explains the data flow operations in the 4*4 RAMAT. To start with, when the RAM address is 00, the first set of four real samples is stored in the first row of RAMs. During this time, the four imaginary samples are latched. Then the address is changed to 01, and the imaginary samples are stored in the same first row of RAMs. The second set of four samples are stored in the second row with the address 00 for real samples and 01 for imaginary samples. This procedure continues until all the 4 rows of RAMs are filled with real and imaginary samples.

The READ-OUT operation in the first column of RAMs begins with address 00. After the first column real samples are read out, the address is changed to 01 to read the imaginary samples from the first column. At this point in time, instead of reading the samples from the second column of RAMs,

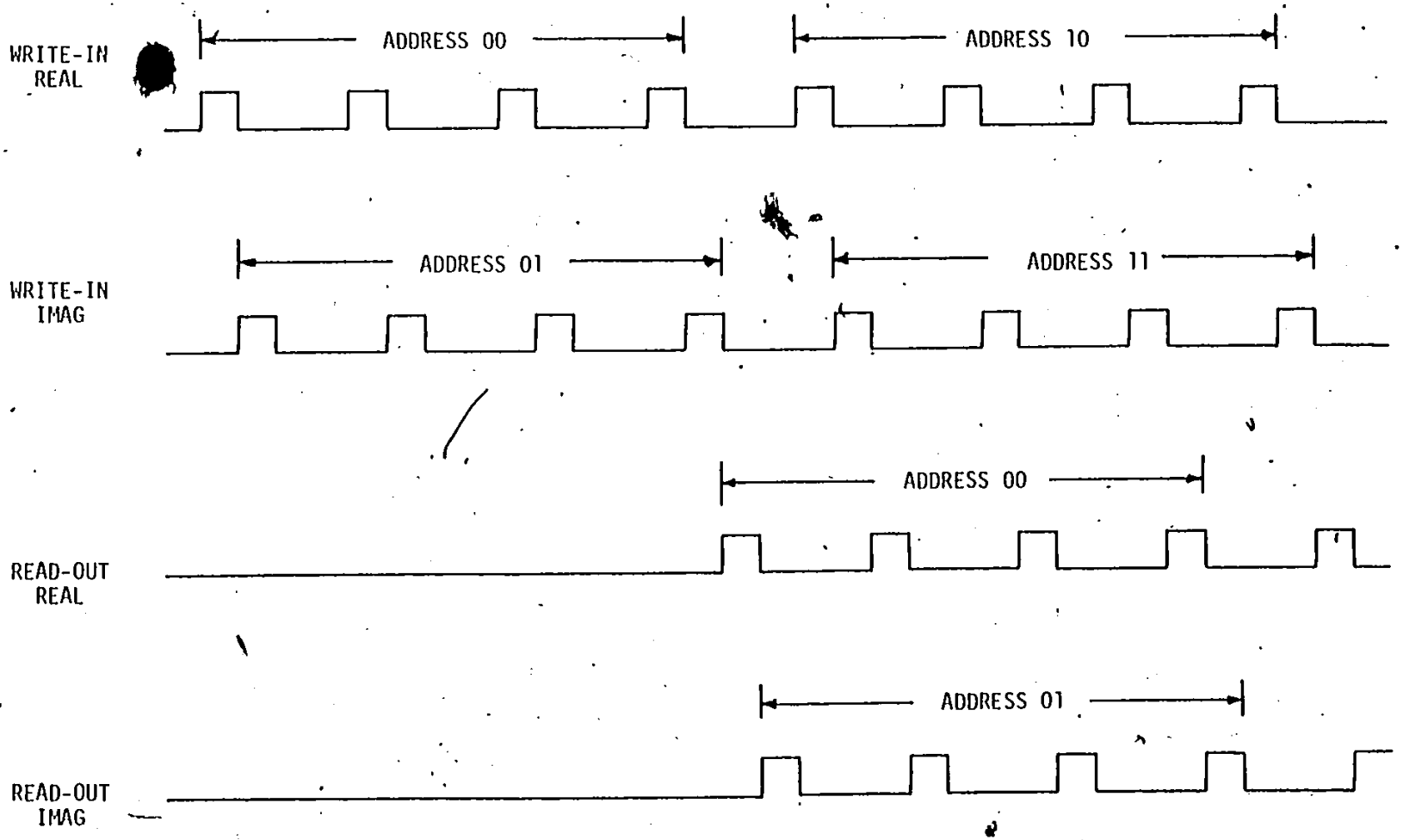


FIG. 5.3 TIMING DIAGRAM FOR A 4x4 MATRIX TRANSPOSER

a new set of four real samples is stored in the first row of RAMs with address 10. This does not conflict as the old samples still remain in 00 and 01 memory locations. The imaginary samples are stored with the RAM address 11. Now once again the address is changed to 00 and subsequently to 01 to read the real and imaginary samples from the second column of RAMs. Then the new samples are stored in the second row with the RAM addresses 10 and 11. After the WRITE-IN operation is completed in the 4th row of RAMs with addresses 10 and 11, the addresses will remain as 10 and 11 and the first column READ-OUT of samples is performed. Then the addresses are changed to 00 and 01 for storing the first set of samples in the first row of RAMs. At this point in time, one cycle is completed as far as address sequencing is concerned.

In the preceding discussion, we considered the address sequencing for the RAMs only. The address sequencing operations for demultiplexing the 'read' and 'write' control lines are relatively simple compared to that of RAMs. Here the address wordlength depends on the number of rows and the number of columns. For the 4*4 matrix transposer, two-bit address is required for each of the read and write operations.

The samples are alternatively stored-in and read-from the RAMAT. This way we can avoid an extra RAMAT. The price

paid for this is to have high speed RAMs. In general, the RAM access time should be about one-fourth of the addition time or the ROM multiplication time associated with preceding or succeeding FFT to RAMAT.

5.4 HIGHER ORDER ONE-DIMENSIONAL FFT PROCESSORS

The complexity of a parallel pipeline FFT processor increases dramatically with N, the number of samples greater than or equal to 64. Typically a 64 point parallel pipeline FFT requires thousands of Integrated Circuit components. Unless modern techniques for VLSI implementation such as Systolic Arrays [53] are considered, this approach is not worthwhile.

Another method of realizing a high order one-dimensional FFT is to employ a serial pipeline FFT processor. A 64 point FFT can be realized with a three-stage radix-4 FFT unit. Similarly 256 point FFT realization may include four-stage radix-4 or eight-stage radix-2 serial pipeline FFT. The control (switching) complexity associated with serial pipeline FFT processors prevents one from realizing these methods in most cases.

It appears that the hybrid approach with timesharing a certain amount of parallel hardware helps in improving the cost/complexity for a given data throughput rate requirement. In this section, we will describe such approaches for realizing 64 and 256 point FFT processors.

5.4.1 64 Point FFT Processor

The architecture of the proposed 64 point FFT processor is shown in Fig. 5.4. The 64 samples are arranged in a two-dimensional array (16*4) (Fig.5.5). The 16 point parallel pipeline FFT processor as described in the previous chapter is time shared to process 4 rows of 16 samples each. The output of the 16 point FFT unit is multiplied by the DFT coefficients. This is achieved by complex stored-product ROM multipliers. Due to the symmetry of the DFT coefficients, a single ROM multiplier with twiddle factors shown in Table.5.1 can be timeshared to process 4 rows of 16 samples each [30]. However, this timesharing requires complex switching across the ROMs. The output of the ROM array multipler is stored in a RAM array transposer (RAMAT). The RAMAT consists of 64 RAM elements (M=16, N=4 in Fig.5.1), each of which can store the real and imaginary values of each output sample. The data are read out from the RAMAT columns, four samples at a time. To maintain the same data throughput rate at all stages of the processor, a set of four 4-point DFTs working in parallel is timeshared to perform column transforms. The first 4 point DFT works on samples of columns 1 to 4. The second 4 point DFT processes the samples of columns 5 to 8 and so on. The outputs of all the 4-point DFT blocks are stored in the output memory for possible further processing. The data throughput rate of the 64 point DFT processor will be the same as that of 16 point FFT processor, 16 samples per clock interval.

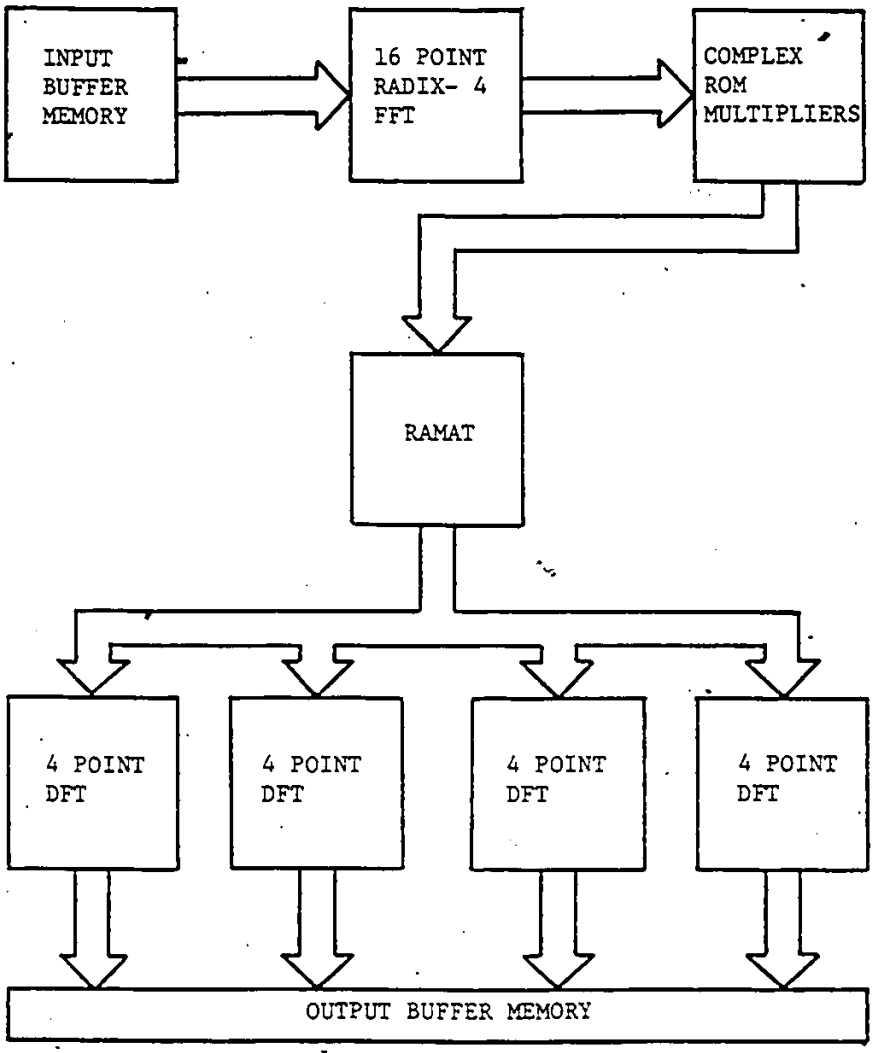


Fig. 5.4 64-point FFT Processor

x(1)	x(2)	x(3)	x(4)	x(5)	x(6)	x(7)	.....	x(16)
x(17)	x(18)	x(19)	x(20)	x(21)	x(22)	x(23)	.....	x(32)
x(33)	x(34)	x(35)	x(36)	x(37)	x(38)	x(39)	.....	x(48)
x(49)	x(50)	x(51)	x(52)	x(53)	x(54)	x(55)	.....	x(64)

Fig.5.5 Two-Dimensional Array of 16 x 4 samples

Table 5.1

Twiddle Factors for Intermediate Multiplications

(64 point FFT)

$W^r = \exp\{(-j2\pi/64)r\}$

where r is given below

1	2	3	4
5	6	7	8
9	10	11	12
13	14	15	18
20	22	24	26
28	30		

5.4.2 256 Point FFT Processor

The method of realization is similar to that of 64 point FFT processor. The block diagram of the processor is shown in Fig.5.6 in which two 16 point FFTs are used respectively for row and column transforms. Instead of having two FFTs, a single FFT can be timeshared to process row and column transforms as shown in Fig. 5.7. The 256 sample data are arranged in a (16*16) format as shown in Fig. 5.8. The output of the row FFT is multiplied by the DFT coefficients. Here once again, with a minimum number of stored-product ROM multipliers, all the multiplication requirements can be achieved with appropriate switching. Table 5.2 shows the twiddle factors for the intermediate multiplications after row FFT. The first 256 samples of data are multiplied and stored in RAMAT, which consists of 256 RAM elements (M=N=16 in Fig.5.1). Then the 16 point FFT processor starts processing the data read out in columns. Starting at this point, the 16 point FFT processor alternatively selects the data from the Input memory and from the RAMAT. In other words, the 16 point FFT processor block fetches the data from Input memory/RAMAT for every two clock intervals. Hence the data throughput rate is slowed down by a factor of two as compared to the 64 point FFT processor.

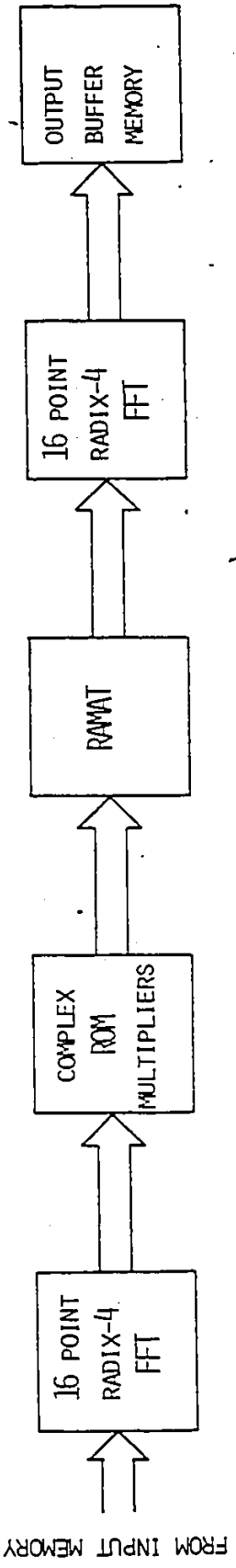


FIG.5.6 256 POINT FFT PROCESSOR

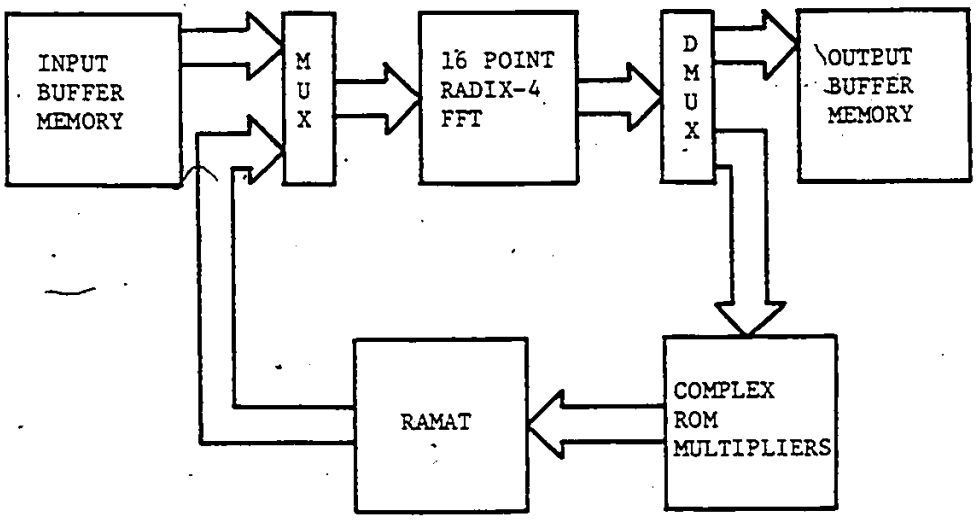


Fig. 5.7 256 point FFT Processor (Alternate form)

x(1)	x(2)	x(3)	x(4)	x(5)	x(6)	x(7)	.....	x(16)
x(17)	x(18)	x(19)	x(20)	x(21)	x(22)	x(23)	.....	x(32)
x(33)	x(34)	x(35)	x(36)	x(37)	x(38)	x(39)	.....	x(48)
x(49)	x(50)	x(51)	x(52)	x(53)	x(54)	x(55)	.....	x(64)
.....			.....		.....		.....	
.....			.....		.....		.....	
.....			.....		.....		.....	
x(241)	x(242)	x(243)	x(244)	x(245)	x(246)	x(247)	.....	x(256)

Fig.5.8 Two-Dimensional Array of 16 x 16 samples

Table 5.2

Twiddle Factors for Intermediate Multiplications

(256 point FFT)

$W^r = \exp\{(-j2\pi/256)r\}$

where r is given below

1	2	3	4	5	6
7	8	9	10	11	12
13	14	15	16	18	20
21	22	24	25	26	27
28	30	32	33	35	36
39	40	42	44	45	48
49	50	52	54	55	56
60	65	66	70	72	75
77	80	81	88	90	91
96	99	104	105	112	120
150	165	180	195	210	225

5.5 LOW ORDER TWO-DIMENSIONAL FFT PROCESSOR

Modern high speed two-dimensional FFT processors are based on extensive software to process a large array of data such as 512*512 or 1024*1024 samples for image processing, pattern recognition etc.,[54]. The processors are in most cases, a general purpose computer controlled array processor or a special purpose processor. They are generally expensive to apply to one processing purpose only.

A hybrid approach of processing requires a special purpose high speed, low order parallel pipeline FFT unit timeshared extensively with the help of a general purpose control processor. This approach seems to be preferred in applications where a small array ($\leq$ 256*256) of data is processed or where there is no sophisticated computer/array processor available.

In this section, we propose a method of realizing a high speed low order (16*16) two-dimensional FFT processor. Here again, a fully implemented parallel-pipeline 16-point FFT processor can be time-shared to do row transforms and column transforms of a two-dimensional array of 16*16 complex samples. The one-dimensional FFT unit obtains the samples in a sequence of rows from the input memory and after processing, stores the results in a RAM array transposer in rows. The complex samples are read from the transposer in columns and fed to the FFT unit again. After processing, the results

are stored in the output memory. As explained in section 5.2, the 16*16 two-dimensional FFT processor requires fewer arithmetic operations than the one- dimensional 256-point FFT processor. As a result, there are no intermediate multiplications after the row FFT. The RAM array transposer can again be used to achieve matrix transposition operation.

5.5.1 16*16 two-dimensional FFT processor

The (16*16) 2-d FFT processor architecture is shown in Fig. 5.9 in which the first 16-point FFT processor output (after row transforms) is stored in consecutive rows in the matrix transposer. The data read from the matrix transposer is fed to the second 16-point FFT for column transforms. The column transformed samples are stored in the output memory for possible further processing. Fig. 5.10 shows a method of realization in which a single 16-point FFT block is timeshared to do row and column transforms. This method saves one 16-point FFT unit at the cost of several multiplexers and demultiplexers and at half the data throughput rate. The detailed operations of the 16 point FFT unit and RAMAT have been described in the previous sections.

The 16 point FFT processor alternatively selects the data from the INPUT MEMORY and the RAMAT. In other words, the 16 point FFT processor block fetches the data from INPUT MEMORY

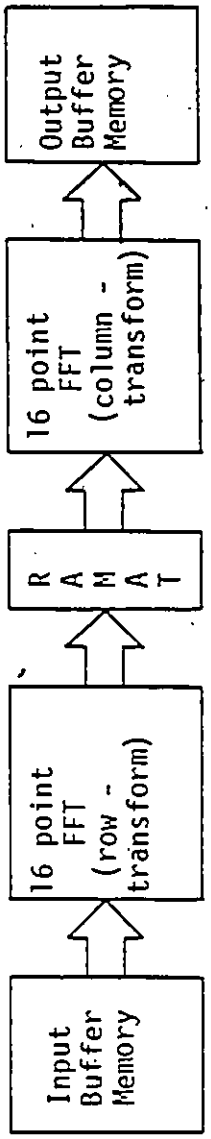


Fig.5.9 16*16 Two-dimensional FFT

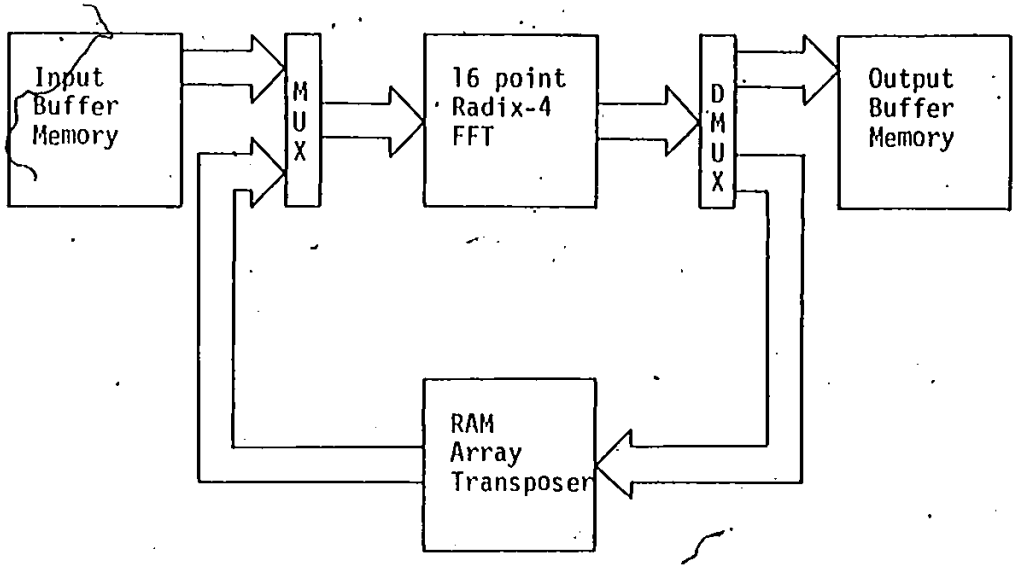


Fig.5.10 16*16 Two-dimensional FFT (Alternate form)

or RAMAT for every two clock intervals. The clock timing can be adjusted such that when the samples are read from the RAMAT and fed to the 16 point FFT unit, the output of the 16-point FFT unit is stored in the OUTPUT MEMORY.

5.6 SPEED AND INITIAL PROCESSING DELAY

Considering one-dimensional FFTs, the speed of 16 point and 64 point processors is compared. It takes four clock intervals to produce the output in the case of a 64 point FFT processor. Therefore the processing rate per sample of the 64 point FFT processor is the same as that of the 16 point FFT unit. When a single 16 point FFT unit is timeshared to realize a 256 point FFT processor, the 256 point FFT processor can operate at half the speed of the 16 or 64 point FFT processor. For example if all the one-dimensional FFT processors are realized with low power Schottky TTL Integrated Circuits, the parallel adders, the ROMs and the RAMs have access times of the order of 100, 100 and 25 nsec respectively, then the potential data throughput rates per sample of 16, 64 and 256 point processors will be 160 MHz, 160 MHz and 80 MHz respectively.

The initial processing delay of 64 point FFT processor is 20 clock intervals. To get the first spectral frame in a 256 point FFT, it takes 48 or 78 clock intervals depending on whether one or two 16 point FFT units are timeshared.

TABLE: 5.3 DATA THROUGHPUT RATE OF HIGH ORDER FFT PROCESSORS

PROCESSOR	INITIAL PROCESSING DELAY	PROCESSING RATE
64-Point FFT (One 16-Point FFT)	20 Sampling Intervals	16 Samples/Clock Interval
256-Point FFT (One 16-Point FFT)	48 Sampling Intervals	8 Samples/Clock Interval
256-Point FFT (Two 16-Point FFTs)	78 Sampling Intervals	16 Samples/Clock Interval
16x16 Points Two-Dimensional FFT (One 16-Point FFT)	46 Sampling Intervals	8 Samples/Clock Interval
16x16 Points Two-Dimensional FFT (Two 16-Point FFTs)	76 Sampling Intervals	16 Samples/Clock Interval

The (16*16) FFT processor will have the output 16 complex samples for every two clock intervals. That is equivalent to eight samples for every clock interval. Given that the maximum time for a basic operation such as parallel addition or real multiplication is 100 nano seconds for TTL ICs, the processor can operate at a per sample data throughput rate of 80 MHz. If we add an extra 16-point unit data throughput rate of 160 MHz per sample can be achieved with a proportionate increase in complexity. The same processor realized with CMOS integrated circuits with an access time of 500 nano seconds, has the data throughput rate of 16 MHz or 32 MHz depending on whether one or two FFT units used. The initial processing delay of the 16*16 two-dimensional FFT processor is 46 clock intervals. If a single FFT unit is timeshared, it takes 76 clock intervals to get the first spectral output. The data throughput rate comparison is shown in the Table 5.3.

5.7 HARDWARE COMPLEXITY

The approximate amount of hardware required for one-dimensional 64 point and 256 point FFT processors and two-dimensional 16*16 point FFT processor are shown in Table 5.4. In all these processors, 16 point radix-4 parallel pipeline FFT is a common unit. Though the RAMAT requires a large number of RAMs (eg. 256 for 2-d FFT), the memory density requirements are not severe. In fact, each RAM has at most

TABLE:5.4 HARDWARE COMPLEXITY OF HIGH ORDER FFT PROCESSORS

16-POINT FFT UNIT IS A COMMON UNIT IN ALL THESE PROCESSORS

CIRCUIT ELEMENTS (p = Signal/Product wordlength)	64-POINT FFT	256-POINT FFT	16x16 TWO-D FFT
p-bit Adders	94	30	-
$(2^p xp)$ ROMs	88	268	-
(4xp) RAMs	64	256	256
p-bit Latches	32	32	32
4-bit Demultiplexers	2	-	-
16-bit Demultiplexers	-	2	2

four memory locations. To accommodate the real and imaginary parts of a sample in the same memory location, RAM should have a capacity to hold the data of twice the wordlength. This type of RAMAT requires a simpler addressing procedure because each RAM has only two memory locations. For this case, latches to hold the data are not required.

The desired accuracy determines the wordlength, which inturn determines the hardware complexity, especially the complexity due to stored-product ROMs. Computer simulation results of the processors' accuracy performance are given in the Appendix-A.

High order (64 and 256 points) one dimensional and low order (16*16 points) two dimensional processor realization methods have been described. It has been found that the RAMAT is a highly modular method of realization suitable for parallel processing.

Chapter VI

CONSIDERATIONS OF MODIFIED FFT

6.1 INTRODUCTION

In this chapter, expressions are derived for the mean square error due to the product roundoff in modified radix-2 FFT algorithms. To reduce the mean square error at the output of a special purpose, high speed, low order ($N \leq 32$) FFT processor implemented in fixed-point arithmetic, a modified FFT architecture is considered in which an extra bit is added to the register wordlength to prevent overflow. The predicted results are compared with the results obtained by simulation.

The fixed point error analysis of conventional FFT algorithms has been reported by Welch [36] and by Thong and Liu [51]. A special purpose hardware FFT processor design based on a conventional fixed-point arithmetic retains the same signal wordlength throughout all stages. Such an implementation should incorporate a scaling procedure to avoid overflow. This leads to an error which mainly depends on the method of scaling [51], and the locations of the butterflies [55]. In a radix-2 FFT algorithm, there are $\log_2 N$ stages, where N represents the total number of complex samples.

When there is scaling at each stage irrespective of whether there is any overflow or not, the signal is attenuated by a factor of (1/N). There will be a significant reduction in the signal-to-noise ratio at the output of the FFT due to this attenuation of the signal. Instead of scaling at each stage, an extra bit can be added to all the registers to prevent overflow. In this way, the wordlength is allowed to grow 1 bit at each stage or $\log_2 N$ bits overall. Rounding to the original wordlength can be done at the output. Fig. 6.1 illustrates a modified FFT configuration for the decimation-in-time FFT algorithm.

A derivation of the expression for the mean square error (MSE) at the modified FFT output is given. In our analysis, we consider both the decimation-in-time (DIT) and the decimation-in-frequency (DIF) algorithms. The computer simulation results are compared with the predicted values. It can be shown that a significant increase in signal/noise ratio can be achieved with the modified FFT approach over the conventional uniform wordlength FFT, with out a proportional increase in the hardware complexity. The input signal quantization effect is not considered here. Coefficient wordlengths are assumed to be long due to the use of stored-product-ROM technique of multiplication. Hence errors due to coefficient quantization are neglected.

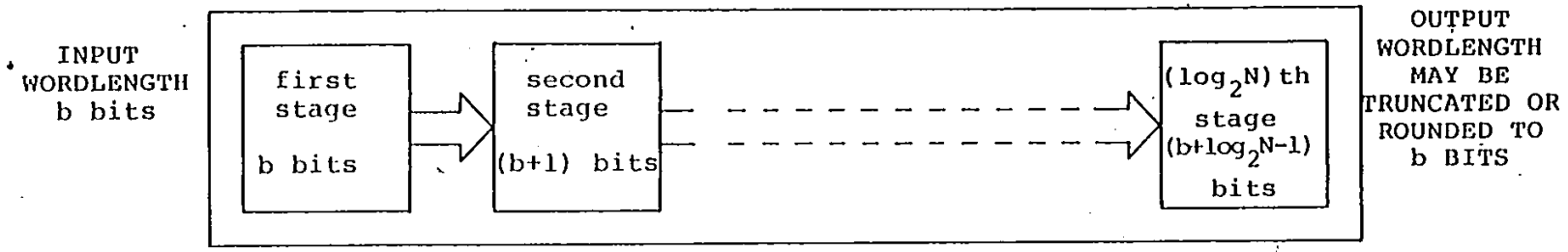


FIG. 6.1 MODIFIED RADIX-2 DECIMATION-IN-TIME FFT ARCHITECTURE

6.2 RADIX-2 DECIMATION-IN-TIME ALGORITHM

In any radix-2 DIT algorithm, there are only trivial multiplications (multiplications by ± 1, $\pm j$ where $j=\sqrt{-1}$) in the first two stages. There will be non-trivial multiplications in the third, fourth and following stages. As shown in Fig. 3.1, the number of butterflies encountering non-trivial multiplications in the third stage is (N/4). In the fourth stage, it is [(N/4)+N/8)] etc. The accumulation of roundoff error to any output node depends on the interconnection of the previous butterflies. Thus the error is not uniform in all the output nodes. For example, if $\nu(=\log_2 N)$ represents the number of stages, then the lower half(N/2) of the output nodes have an error accumulation due to $(2^{\nu-2}-1)$ error sources. The lower (N/4) of the upper half of the output nodes have an error accumulation due to $(2^{\nu-2-1}-1)$ error sources, and so on. Hence the number of butterflies contributing roundoff errors to any output node depends on the location of the output node. Figs. 6.2 and 6.3 show the accumulation of roundoff errors in the 2nd and 5th output samples of the 16 point FFT.

6.2.1 Derivation of the expression for mean square error

Let the error variance in the third stage be '$\overline{\mu_3^2}$'. If b (excluding sign bit) is the signal wordlength in the first stage, then (b+2) (excluding sign bit) represents the signal

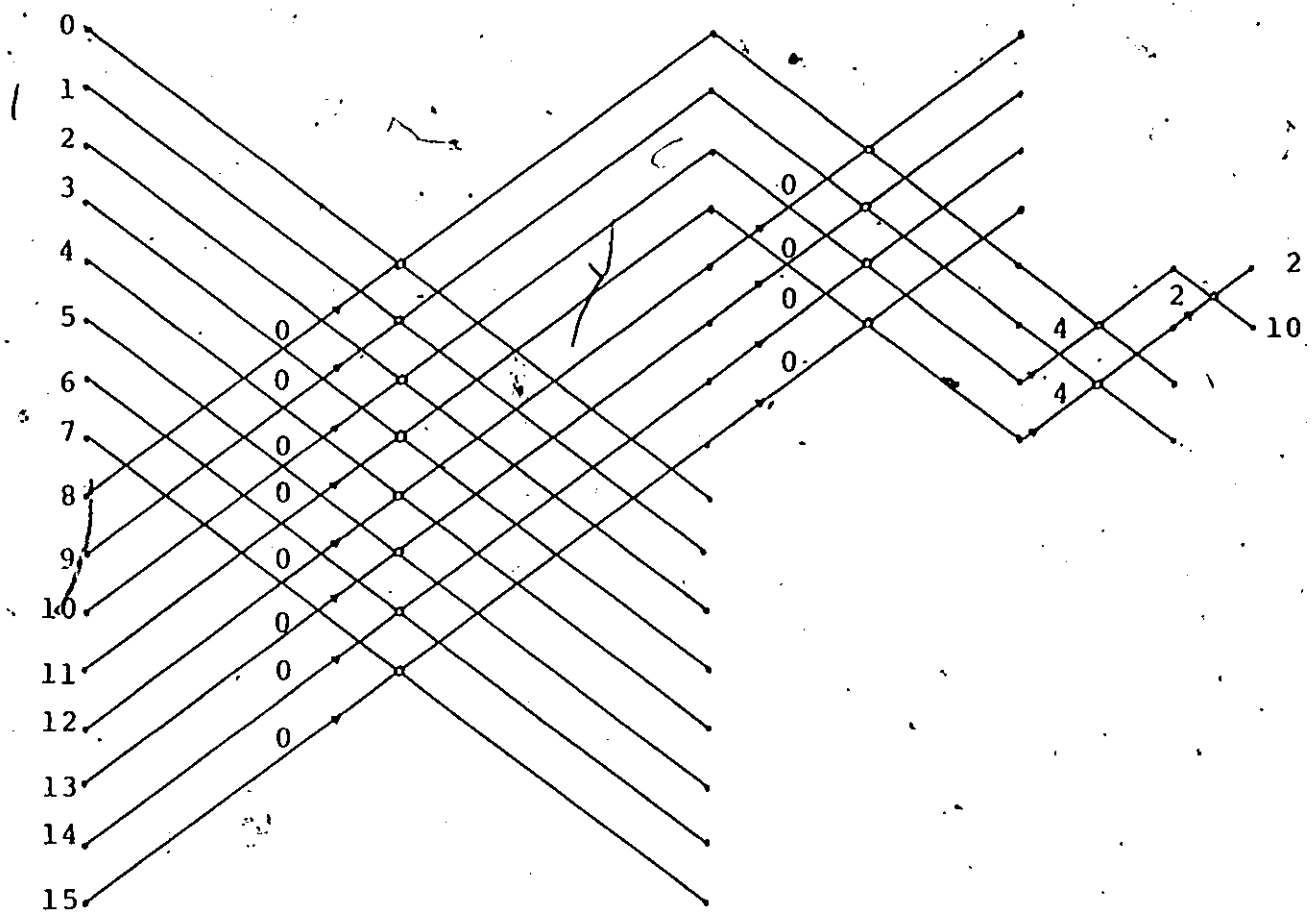


FIG.6.2 ACCUMULATION OF PRODUCT ROUND-OFF ERRORS IN 2ND (OR 10TH) SAMPLE

(16 point Radix-2 DIT FFT)

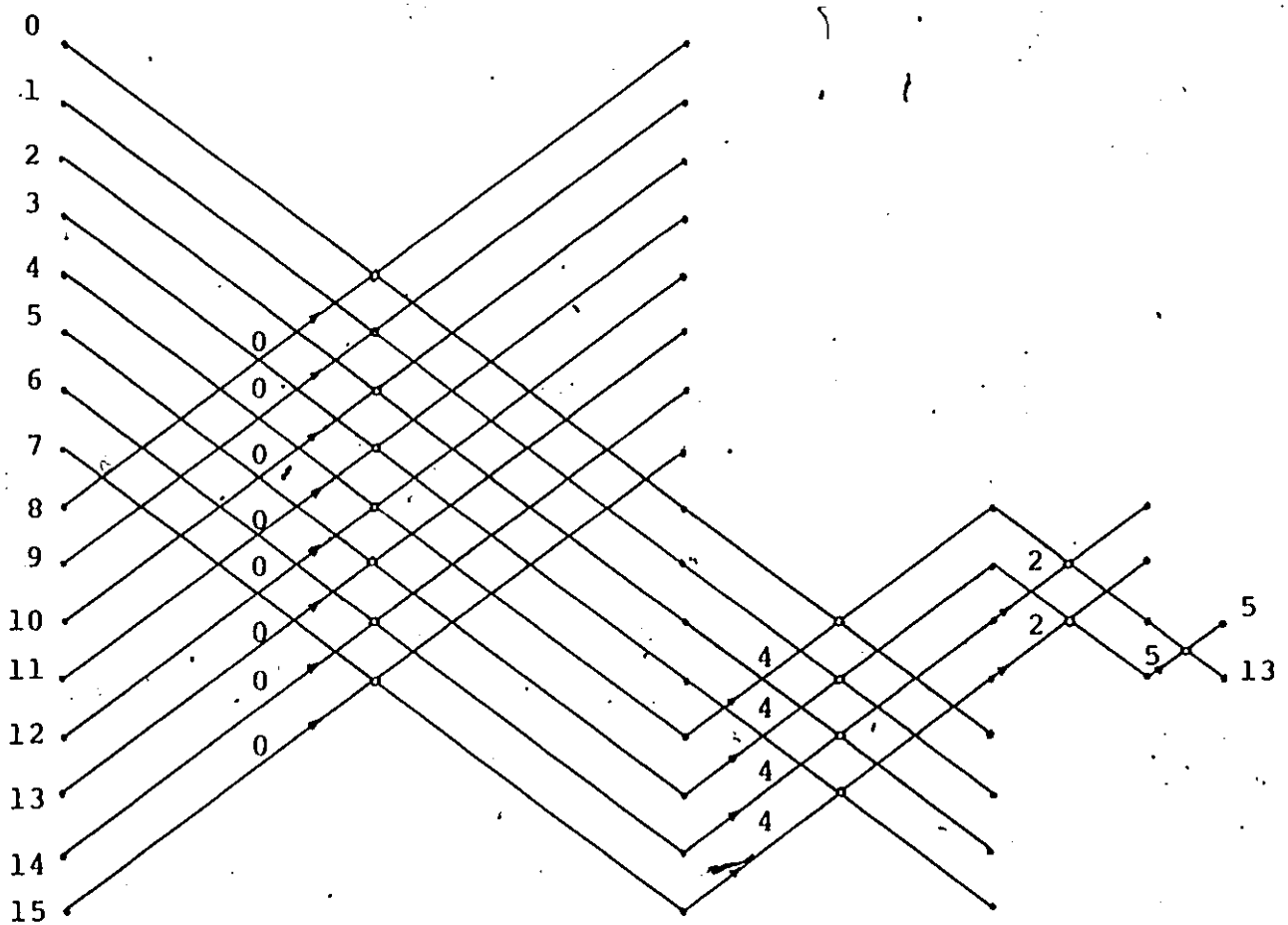


FIG. 6.3 ACCUMULATION OF PRODUCT ROUND-OFF ERRORS IN 5TH (OR 13TH) SAMPLE

(16 point Radix-2 DIT FFT)

and product wordlengths in the third stage. The rounding of the product of two b bit (excluding sign bit) numbers to b bits results in an error with variance ($\frac{2^{-2b}}{12}$). If the errors due to each of the four real multiplications are assumed to be uncorrelated, the error variance of the complex multiplication is given as $(\frac{2^{-2b}}{3})$.

The error variance in the following stages can be written as

$$\overline{\mu_3^2} = \frac{2^{-2}(b+2)}{3} \qquad \text{(third stage)}$$

$$\overline{\mu_4^2} = \overline{\mu_3^2}/4 \qquad \text{(fourth stage)}$$

$$\overline{\mu_5^2} = \overline{\mu_3^2}/4^2 \qquad \text{(fifth stage)}$$

$$\cdots \quad \cdots$$

$$\cdots \quad \cdots$$

$$\overline{\mu_\nu^2} = \{\overline{\mu_3^2}\}/\{(4)^{\nu-3}\} \qquad \text{(last stage)}$$

The sum of the total number of butterflies contributing roundoff errors to all the output nodes is given as

The total number of error sources is:

$$N_\varepsilon = N\left[\left(\frac{1}{2}\right)\left(2^{\nu-2}-1\right) + \left(\frac{1}{4}\right)\left(2^{\nu-2-1}-1\right) + \left(\frac{1}{2}\cdot\frac{1}{4}\right)\left(2^{\nu-2-2}-1\right)\right.$$
$$+ \cdots\cdots\cdots\cdots\cdots\cdots\cdots\cdots\cdots\cdots +$$
$$\left. + \frac{1}{2^{\nu-2}}\left(2^{\nu-2-(\nu-3)}-1\right)\right] \qquad (6.1)$$

$$= \sum_{m=1}^{\nu-2} \left(\frac{1}{2^m}\right)\left(2^{\nu-1-m}-1\right)N \qquad (6.2)$$

The total error variance for all the output points combined is:

$$\overline{\mu^2} = \sum_{m=1}^{\nu-2} \left(\frac{1}{2^m}\right)\left(2^{\nu-1-m}-1\right)N\overline{\delta^2_{(m+2)}} \qquad (6.3)$$

The average mean square error (assuming the mean is zero) per output point is:

$$\frac{\overline{\mu^2}}{N} = \sum_{m=1}^{\nu-2} \left(\frac{1}{2^m}\right)\left(2^{\nu-1-m}-1\right)\overline{\delta^2_{(m+2)}} \qquad (6.4)$$

where

$$\overline{\delta^2_3} = \frac{\sum_{r=3}^{\nu} \left(\frac{N}{2^r}\right)\overline{\mu^2_r}}{\left(2^{\nu-2}-1\right)}$$ is the average error variance in lower half (N/2) of output nodes

$$\overline{\delta^2_4} = \frac{\sum_{r=4}^{\nu} \left(\frac{N}{2^r}\right)\overline{\mu^2_r}}{\left(2^{\nu-2-1}-1\right)}$$ is the average error variance in lower half (N/4) of upper half of output nodes.

$$\vdots \qquad \vdots$$

$$\overline{\delta^2_\nu} = \overline{\mu^2_\nu} \qquad (6.5)$$

By substituting (6.5) in (6.4), the average mean square error (MSE) becomes:

$$\frac{\overline{\mu^2}}{N} = \left[\frac{1}{2}(2^{\nu-2}-1)\frac{\{2^{\nu-3}\overline{\mu_3^2} + 2^{\nu-4}\overline{\mu_4^2} + 2^{\nu-5}\overline{\mu_5^2} + \ldots + \overline{\mu_\nu^2}\}}{(2^{\nu-2}-1)}\right]$$
$$+\left[\left(\frac{1}{2}\right)^2(2^{\nu-3}-1)\frac{\{2^{(\nu-4)}\overline{\mu_4^2} + 2^{(\nu-5)}\overline{\mu_5^2} + \ldots + \overline{\mu_\nu^2}\}}{(2^{\nu-3}-1)}\right]$$
$$+ \ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots +$$
$$+ \left(\frac{1}{2}\right)^{\nu-2}\overline{\mu_\nu^2} \qquad (6.6)$$

This can be rewritten in the following equivalent forms

$$\frac{\overline{\mu^2}}{N} = \overline{\mu_\nu^2}\left[\left(\frac{1}{2}\right)^{(\nu-2)} + \left(\frac{1}{2}\right)^{(\nu-3)} + \left(\frac{1}{2}\right)^{(\nu-4)} + \ldots + \frac{1}{2}\right]$$
$$+ \overline{\mu_{\nu-1}^2}\left[\left(\frac{1}{2}\right)^{(\nu-3)} + \left(\frac{1}{2}\right)^{(\nu-4)} + \left(\frac{1}{2}\right)^{(\nu-5)} + \ldots + \frac{1}{2}\right]2$$
$$+ \ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots +$$
$$+ \overline{\mu_4^2}\left[\frac{1}{2^2} + \frac{1}{2}\right]2^{(\nu-4)} \qquad (6.7)$$
$$+ \overline{\mu_3^2}\left(\frac{1}{2}\right)2^{\nu-3}$$

or

$$\frac{\overline{\mu^2}}{N} = \overline{\mu_\nu^2}\left[1 - \left(\frac{1}{2}\right)^{\nu-2}\right] + \overline{\mu_{\nu-1}^2}\left[1 - \left(\frac{1}{2}\right)^{\nu-3}\right]2$$
$$+ \overline{\mu_{\nu-2}^2}\left[1 - \left(\frac{1}{2}\right)^{\nu-4}\right]2^2$$
$$+ \ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots + \qquad (6.8)$$
$$+ \overline{\mu_3^2}\left(\frac{1}{2}\right)2^{\nu-3}$$

or

$$\frac{\overline{\mu^2}}{N} = [\overline{\mu_\nu^2} + 2\overline{\mu_{\nu-1}^2} + 2^2\overline{\mu_{\nu-2}^2} + \ldots + 2^{\nu-4}\overline{\mu_4^2} + 2^{\nu-3}\overline{\mu_3^2}]$$

$$-(\tfrac{1}{2})^{\nu-2}[\overline{\mu_\nu^2} + 2^2\overline{\mu_{\nu-1}^2} + 2^4\overline{\mu_{\nu-2}^2} + \ldots + 2^{2(\nu-4)}\overline{\mu_4^2} + 2^{2(\nu-3)}\overline{\mu_3^2}] \quad (6.9)$$

Substituting the expressions for the average error variance in different stages of FFT in (6.9) yields, after simplification:

$$\frac{\overline{\mu^2}}{N} = 2^{\nu-3}\overline{\mu_3^2}[1 + (\tfrac{1}{2})(\tfrac{1}{4}) + (\tfrac{1}{2})^2(\tfrac{1}{4})^2 + \ldots + (\tfrac{1}{2})^{\nu-3}(\tfrac{1}{4})^{\nu-3}]$$

$$- (\tfrac{1}{2})^{\nu-2}4^{\nu-3}\overline{\mu_3^2}[1 + (\tfrac{1}{4})(\tfrac{1}{4}) + \ldots + (\tfrac{1}{4}\cdot\tfrac{1}{4})^{\nu-3}] \quad (6.10)$$

or

$$\frac{\overline{\mu^2}}{N} = 2^{\nu-3}\overline{\mu_3^2}\left[\frac{1 - (1/8)^{\nu-2}}{1 - (1/8)}\right] - (\tfrac{1}{2})^{\nu-2}4^{\nu-3}\overline{\mu_3^2}\left[\frac{1 - (1/16)^{\nu-2}}{1 - (1/16)}\right] \quad (6.11)$$

After simplification,

$$\frac{\overline{\mu^2}}{N} = \overline{\mu_3^2}\, 2^\nu\left[\left\{\frac{1 - (1/8)^{\nu-2}}{7}\right\} - \left\{\frac{1 - (1/16)^{\nu-2}}{15}\right\}\right] \quad (6.12)$$

since $2^\nu = N$ this becomes finally:

$$\frac{\overline{\mu^2}}{N} = N \cdot \overline{\mu_3^2}\left[\left\{\frac{1 - (1/8)^{\nu-2}}{7}\right\} - \left\{\frac{1 - (1/16)^{\nu-2}}{15}\right\}\right] \quad (6.13)$$

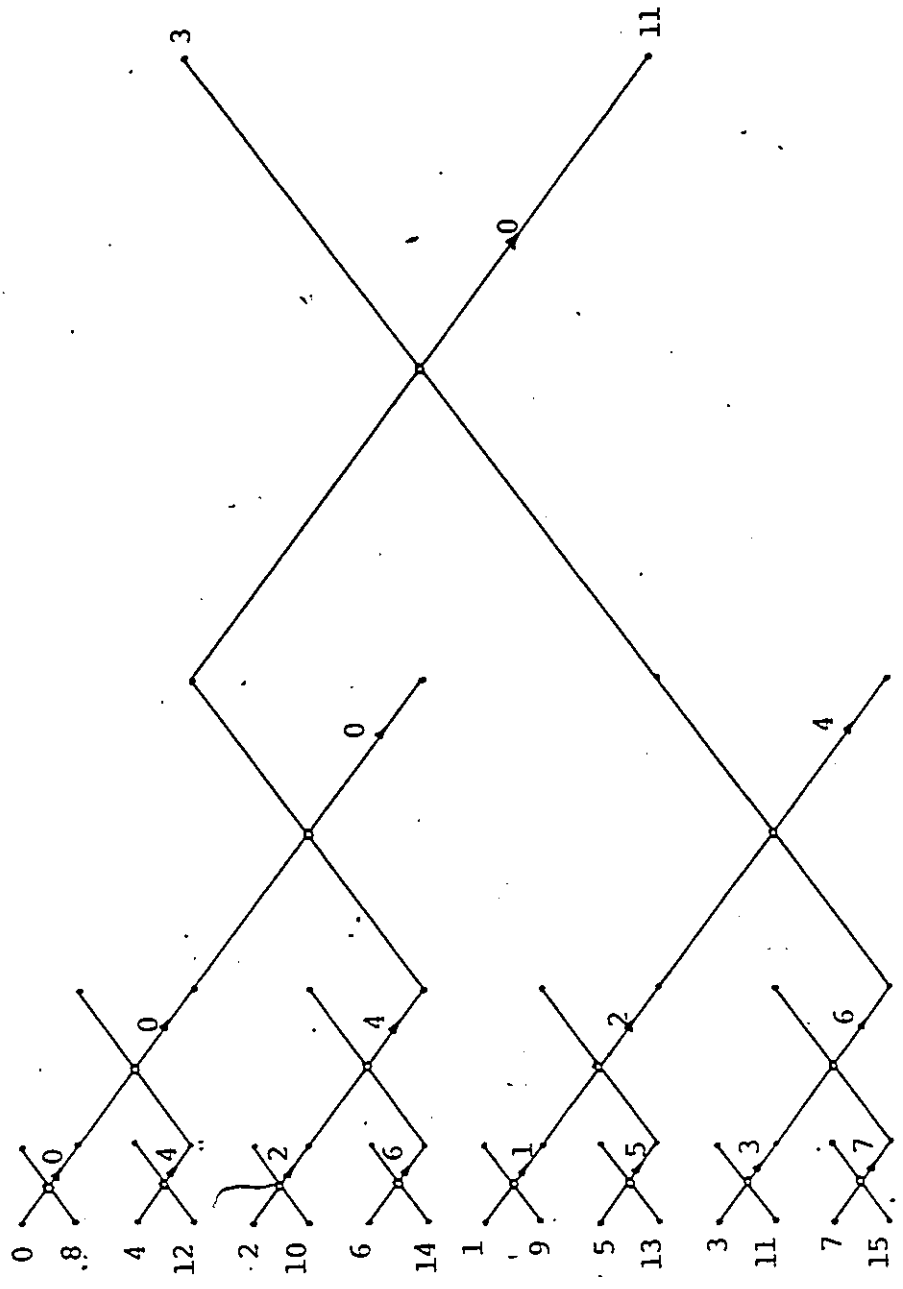


FIG.6.4 ACCUMULATION OF ROUND-OFF ERRORS IN 3RD (OR 11TH) SAMPLE OF 16 POINT DIF FFT

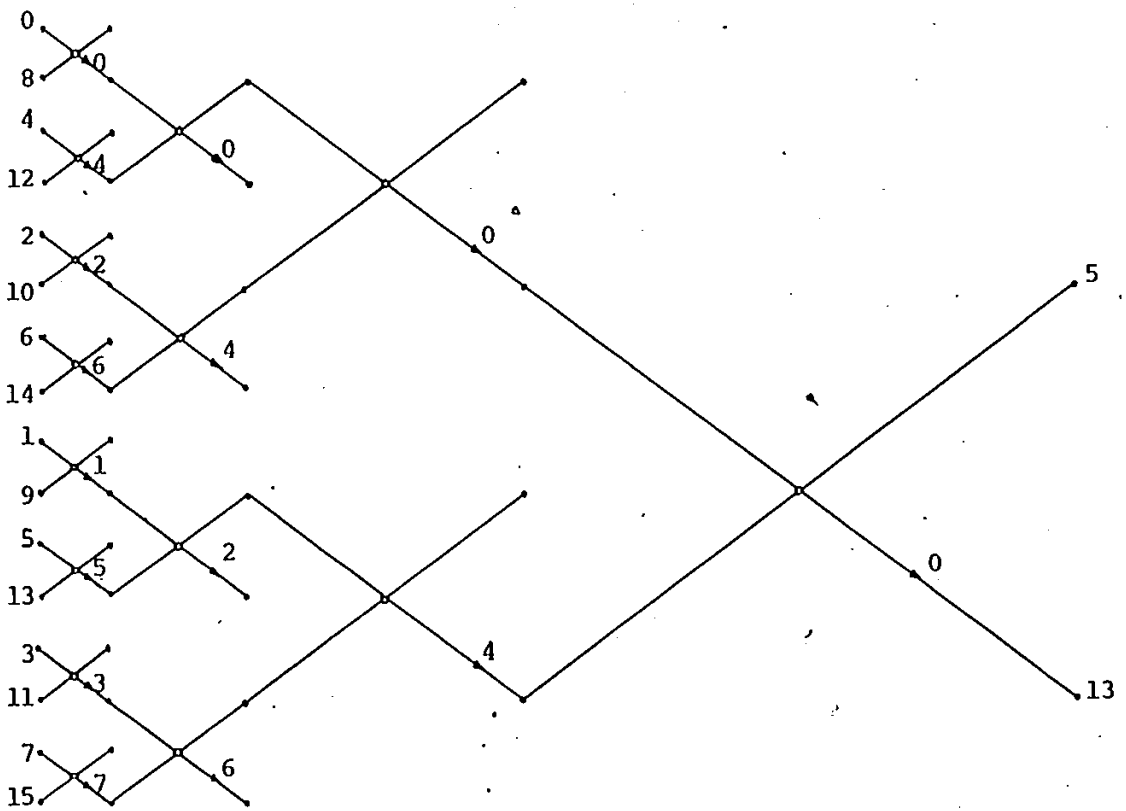


FIG.6.5 ACCUMULATION OF ROUND-OFF ERRORS IN 5TH (OR 13TH) SAMPLE OF 16 POINT DIF FFT

6.3 RADIX-2 DECIMATION-IN-FREQUENCY ALGORITHM

Unlike a radix-2 DIT algorithm, in a radix-2 decimation-in-frequency (DIF) algorithm (Fig. 4.9), there are only trivial multiplications in the last two stages. The non-trivial multiplications exist in the first $(\nu-2)$ stages. In the first stage, the number of butterflies with non-trivial multiplications is $[(N/2)-2]$. In the second stage, $2[(N/4)-2]$ butterflies encounter non-trivial multiplications, and in the third stage it is $2^2[(N/8)-2]$, etc. In general, the number of non-trivial multiplications in the $(\nu-2)$ stage is

$$2^{\nu-3}\{(\frac{N}{2^{\nu-2}})-2\} = N/4$$

As in the DIT algorithm, the error accumulations to output nodes are not uniformly distributed. Figs. 6.4 and 6.5 show error accumulations in the 4th and 14th nodes of the 16-point DIF FFT.

6.3.1 Derivation of the expression for mean square error

Let the error variance due to product roundoff in the process of complex multiplication in the first stage be μ_1^2. In DIF algorithms, multiplications take place after the addition/subtraction operations. Therefore, even in the first stage the multiplier should have an extra bit. Hence

$$\mu_1^2 = \frac{2^{-2(b+1)}}{3}$$

The error variance in the following stages can be written as

$$\overline{\mu_2^2} = \frac{\overline{\mu_1^2}}{4} \qquad \text{(second stage)}$$

$$\overline{\mu_3^2} = \frac{\overline{\mu_1^2}}{4^2} \qquad \text{(third stage)}$$

$$\cdots\cdots\cdots\cdots\cdots\cdots$$

$$\cdots\cdots\cdots\cdots\cdots\cdots$$

$$\overline{\mu_{\nu-2}^2} = \frac{\overline{\mu_1^2}}{4^{(\nu-3)}} \qquad (((\nu-2)^{th} \text{ stage})$$

The total number of error sources is:

$$N_\varepsilon = 2^{(\nu-3)}4[(2^2 - 2) + (2^3 - 2) + \ldots + (2^{(\nu-1)} - 2)] \qquad (6.14)$$

The total error variance for all the output points is:

$$\overline{\mu^2} = 4 \cdot 2^{(\nu-3)}[(2^2 - 2)\overline{\mu_{\nu-2}^2} + (2^3 - 2)\overline{\mu_{\nu-3}^2} + (2^4 - 2)\overline{\mu_{\nu-4}^2} + \ldots + \overline{\mu_1^2}(2^{\nu-1} - 2)] \qquad (6.15)$$

The average mean square error (MSE) per output point is:

$$\frac{\overline{\mu^2}}{N} = 4\,\frac{2^{\nu-3}}{N}[(2^2 - 2)\overline{\mu_{\nu-2}^2} + (2^3 - 2)\overline{\mu_{\nu-3}^2} + (2^4 - 2)\overline{\mu_{\nu-4}^2} + \ldots + \overline{\mu_1^2}(2^{\nu-1} - 2)] \qquad (6.16)$$

or

$$\frac{\overline{\mu^2}}{N} = \frac{1}{2}\left[\sum_{m=2}^{\nu-1}(2^m - 2) \cdot \overline{\mu_{\nu-m}^2}\right] \qquad (6.17)$$

This can be rewritten, with $N = 2^{\nu}$, as:

$$\overline{\frac{\mu^2}{N}} = \frac{1}{2}\left[\{2^2\overline{\mu^2_{\nu-2}} + 2^3\overline{\mu^2_{\nu-3}} + \ldots + 2^{\nu-1}\overline{\mu^2_1}\} - 2\{\overline{\mu^2_{\nu-2}} + \overline{\mu^2_{\nu-3}} + \ldots + \overline{\mu^2_1}\}\right] \tag{6.18}$$

Substituting the expressions for error variance in the different stages of FFT, in (6.18) and simplification yields:

$$\overline{\frac{\mu^2}{N}} = \frac{1}{2}\left[\{2^2\frac{\overline{\mu^2_1}}{4^{(\nu-3)}} + 2^3\frac{\overline{\mu^2_1}}{4^{(\nu-4)}} + \ldots + \frac{2^{\nu-2}\overline{\mu^2_1}}{4} + 2^{\nu-1}\cdot\overline{\mu^2_1}\} - \{2\frac{\overline{\mu^2_1}}{4^{\nu-3}} + \ldots + \overline{\mu^2_1}\}\right] \tag{6.19}$$

or

$$\overline{\frac{\mu^2}{N}} = \overline{\mu^2_1}\left[2^{\nu-2}\{1 + \frac{1}{8} + \frac{1}{8^2} + \ldots + \frac{1}{(8)^{\nu-3}}\} - \{1 + \frac{1}{4} + \frac{1}{4^2} + \ldots + \frac{1}{(4)^{\nu-3}}\}\right] \tag{6.20}$$

or

$$\overline{\frac{\mu^2}{N}} = \overline{\mu^2_1}\left[2^{\nu-2}\left\{\frac{1 - (\frac{1}{8})^{\nu-2}}{1 - (\frac{1}{8})}\right\} - \left\{\frac{1 - (\frac{1}{4})^{\nu-2}}{1 - (\frac{1}{4})}\right\}\right] \tag{6.21}$$

6.4 NOISE-TO-SIGNAL RATIO CALCULATIONS

Based on the expressions (6.13) and (6.21) for the mean square error due to product roundoff errors, it is possible to find expressions for noise-to-signal ratio at the output of the modified FFTs. The output wordlength of the modified FFT would be limited to the original (input) wordlength in two ways, (1) by truncating the output word, and (2) by rounding.

(1) Product Rounding and Scaling by Truncation:

MSE (product roundoff) = Equation (6.13) or (6.21)

MSE (scaling) = $\frac{2^{-2b}}{3}$ where 'b' is input/output wordlength

output mean square signal = (N/3) (for uniformly distributed random numbers -1 to +1)

$$\text{output N/S ratio} = \left[\frac{\text{MSE(product roundoff)} + \text{MSE(scaling)}}{\text{mean square signal}}\right]$$

(2) Product Rounding and Scaling by Rounding:

MSE (scaling) = $\frac{2^{-2b}}{12}$

output mean square signal = N/3

$$\text{output N/S ratio} = \left[\frac{\text{MSE(product roundoff)} + \text{MSE(scaling)}}{\text{mean square signal}}\right]$$

6.5 COMPUTER SIMULATION RESULTS

The architecture of the modified FFT processor is simulated on a general purpose computer to find the mean square error at the output. The input samples are uniformly distributed random numbers in the range -1 to +1. Fixed point two's complement arithmetic is considered. The detailed explanation of the computer simulation procedure is given in Appendix-A.

The computer simulation results for the 16-point DIT modified FFT are shown in Fig. 6.6. Fig. 6.7 shows the computer simulation results of a 16 point radix-2 decimation-in-time (DIT) FFT, implemented in a conventional form with uniform wordlength throughout the processor. The error performance comparison of the modified FFT and the conventional FFT is based on the average of the input and the output wordlengths in a modified FFT, because the hardware required for both schemes is approximately the same when a conventional FFT is implemented with a uniform wordlength which is equal to the average wordlengh in a modified FFT. This is a reasonable approximation for low order ($N \leq 32$) FFT processors, since the butterflies with non-trivial multiplications are not concentrated evenly throughout the FFT stages.

For example when the input wordlength to a 16-point modified FFT is 'b' bits, then the output wordlength will be (b+4) bits. Hence the mean square error (MSE) of this FFT

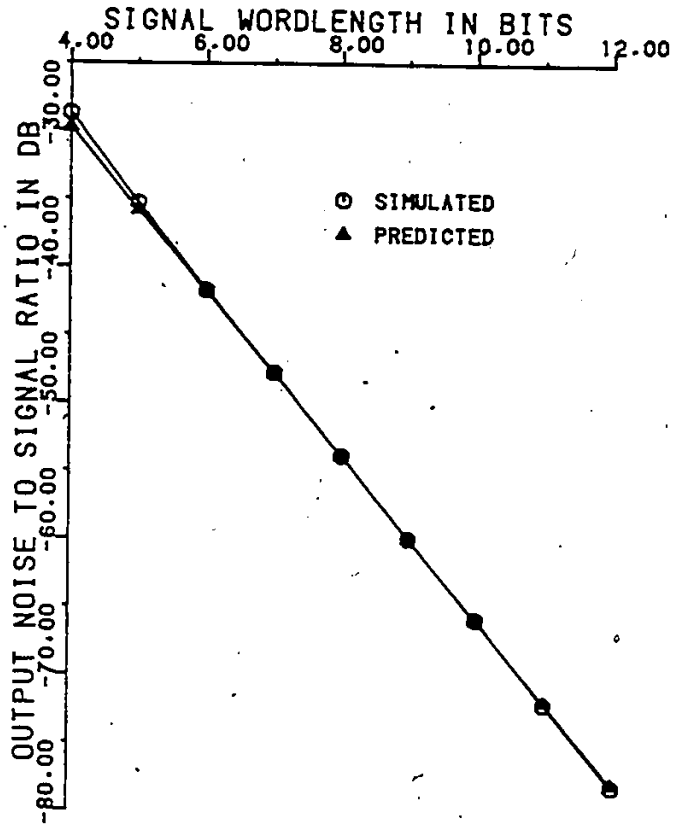


FIG.6.6 NOISE/SIGNAL RATIO OF MODIFIED DIT FFT
(PRODUCT ROUNDING AND SCALING BY TRUNCATION)
(AVERAGING OVER 100 SETS OF SAMPLES)

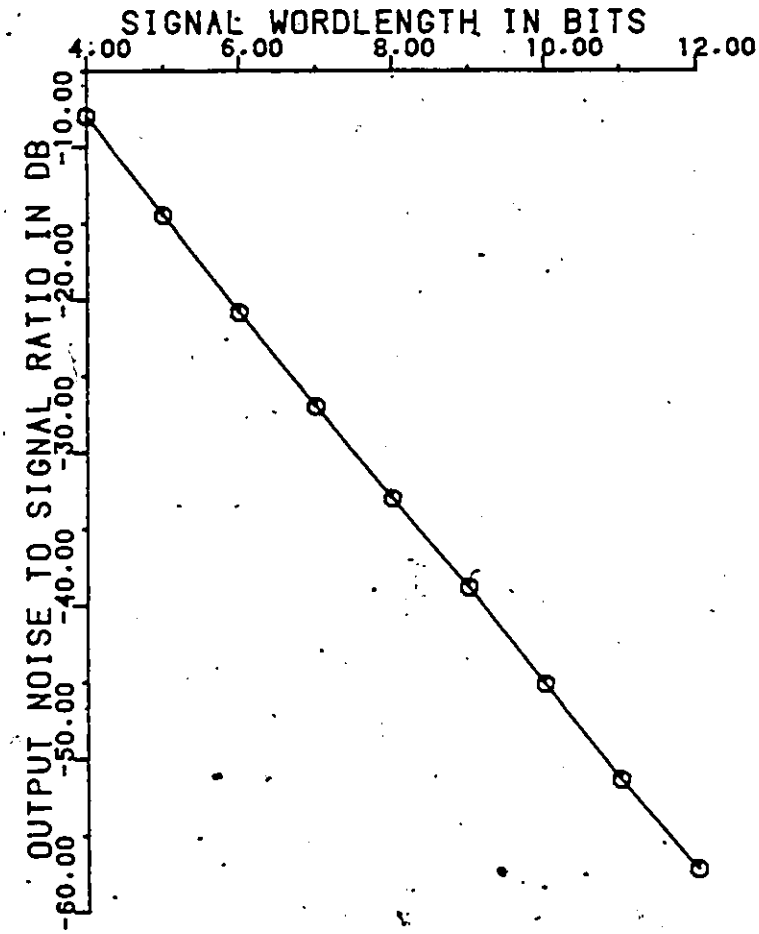


FIG.6.7 NOISE/SIGNAL RATIO OF UNIFORM WORDLENGTH DIT FFT

(PRODUCT ROUNDING, AVERAGING OVER 100 SETS OF SAMPLES)

is compared with a conventional FFT of (b+2) bits word-length. From Figs. 6.6 and 6.7, it is noted that the modified FFT offers a better signal to noise ratio than the conventional FFT.

To further improve the signal-to-noise ratio of the modified FFT, the output words can be limited by rounding, instead of truncation. For this case, the computer simulation results are shown in Fig.6.8. It is noted, from Figs. 6.6 and 6.8 that rounding offers about 6 dB improvement of signal-to-noise ratio over truncation, at the expense of little increase in the hardware complexity.

The computer simulation results of radix-2 DIF modified FFT are shown in Figs. 6.9 and 6.10; respectively for the two cases (1) scaling by truncation and (2) scaling by rounding. Here again, rounding offers better signal/noise ratio over truncation.

One interesting point is that the DIF algorithm-based FFT requires relatively less hardware as compared to the DIT FFT for the same signal/noise ratio, because in DIF FFT the multiplications are mainly concentrated in the butterflies of earlier stages where short wordlengths are required.

The fixed point error analysis is carried out for modified FFTs. Simulated results are found to be close to the

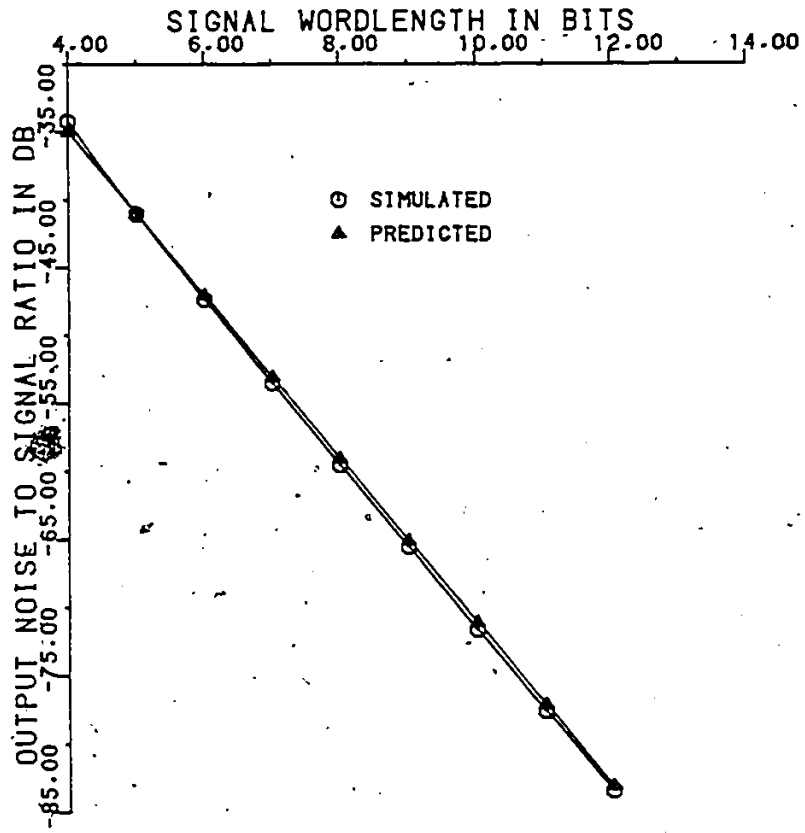


FIG. 6.8 NOISE/SIGNAL RATIO OF MODIFIED DIT FFT

(PRODUCT ROUNDING AND SCALING BY ROUNDING)

(AVERAGING OVER 100 SETS OF SAMPLES)

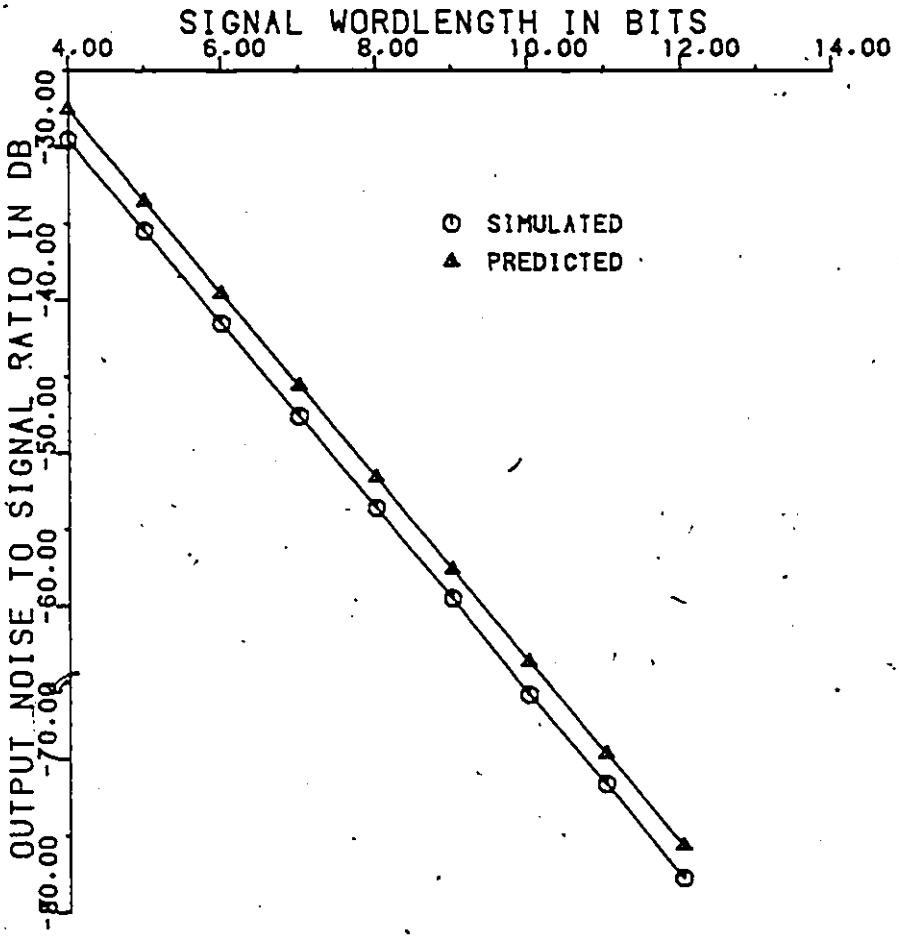


FIG.6.9 NOISE/SIGNAL RATIO OF MODIFIED DIF FFT

(PRODUCT ROUNDING AND SCALING BY TRUNCATION)

(AVERAGING OVER 100 SETS OF SAMPLES)

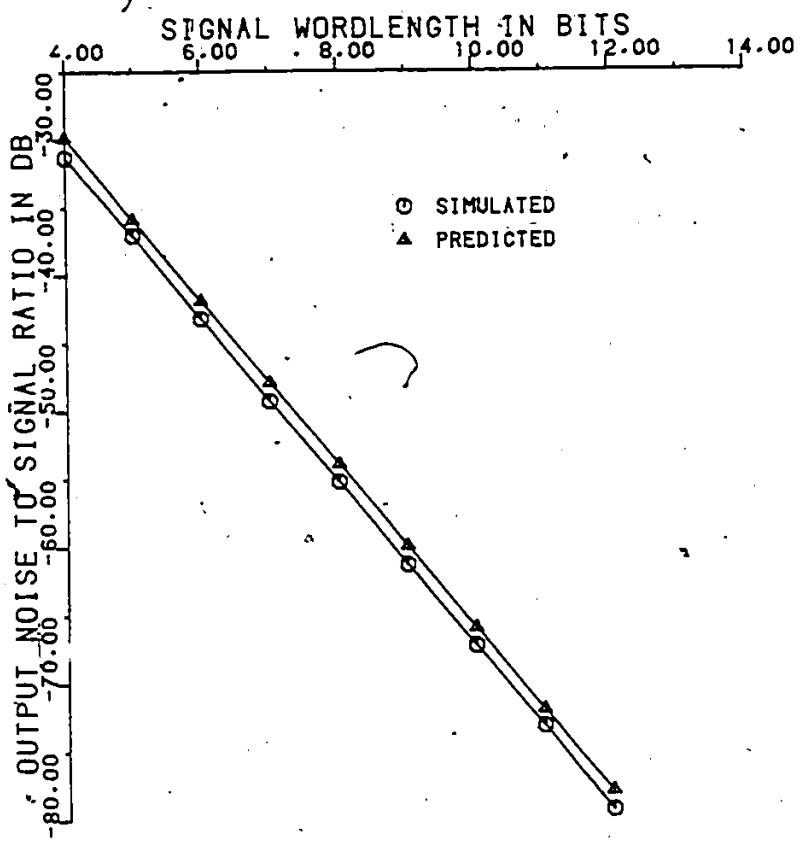


FIG. 6.10 NOISE/SIGNAL RATIO OF MODIFIED DIF FFT

(PRODUCT ROUNDING AND SCALING BY ROUNDING)

(AVERAGING OVER 100 SETS OF SAMPLES)

predicted values. The FFT output wordlength can be limited, if necessary, to a desired length for further processing. This contributes an error which is dominant than the product roundoff error. It is shown that for approximately the same amount of hardware, the modified FFT offers a better signal to noise ratio than the conventional FFT. The realization of modified low order ($N \leq 32$) FFT is worth considering in applications where the accumulation of roundoff errors is a critical problem.

Chapter VII

CONCLUDING REMARKS

A class of fast Fourier transform processors for high speed real time radar signal processing has been presented. The signal processing requirements of modern radar systems have been emphasized and the justification for the need of high speed FFT processor has been made clear. A new method of achieving very high data throughput rates in the fast Fourier transform processors has been developed.

Application of the stored-product ROM multiplication concept to the FFT processor realization is a significant contribution in this thesis research. Several realization techniques of serial pipeline and parallel pipeline FFT processors are analyzed, out of which a low order ($N \leq 32$) parallel pipeline FFT appeared to be a novel method of achieving a significant increase of per sample data throughput rate (number of complex samples/second), in excess of 100 MHz.

The utilization of a high speed low order (16 points) parallel pipeline FFT processing unit in realizing a higher order one dimensional (up to 256 points) and low order two dimensional (16*16 points) FFT processors has been consid-

ered as an important part of this thesis research. With reference to this, an original concept of a high speed matrix transposition technique for parallel pipeline processing, Random Access Memory Array Transposer (RAMAT) has been proposed. This method involves an efficient address sequencing procedure in order to minimize the number of RAMs required.

Another original contribution of this thesis research is statistical error analyses to find the accumulations of product roundoff errors in a modified low order FFT of increasing wordlength to avoid overflows in the fixed-point additions. It has been found out that for approximately the same hardware complexity, modified FFT offers a better signal/noise ratio over a conventional uniform wordlength FFT. The computer simulation results are found to be close to the predicted values.

The proposed highspeed FFT processors are not programmable in nature. They are special purpose processors dedicated to perform a particular task in high speed radar signal processing systems. The per sample data throughput rate increases with the number of samples for parallel pipeline realization whereas the data throughput rate is fixed at two samples per clock interval for radix-2 and four samples per clock interval for radix-4 serial pipeline FFTs. Parallel pipeline realization requires more hardware but simple control. On the otherhand, serial pipeline realization re-

quires less hardware but more complicated control for switching. Another method of realizing high order one dimensional FFT processors is to timeshare a 16-point FFT block in two dimensional processing. The choice of this method or a higher order one dimensional serial pipeline FFT mainly depends on the control complexity of the serial pipeline FFT, the hardware complexity and of course on the data throughput rate required. Hence one has to seriously consider the requirements of a special purpose processor before selecting one of the schemes.

Systolic array architecture is a very powerful approach for applying VLSI to signal processing [53]. This type of architecture has a number of attributes that make it compatible with VLSI. The architecture provides a technique for using multiple processors to attack a signal processing problem. Since all the processors are the same type, the result is usually a low cost system with very large processing power.

The simplicity of implementing the DFT using a systolic array has been described [56]. The number of processor modules is equal to the number of points of the DFT. Each module mainly consists of a complex multiply and accumulate operation. FIR filter realzation based on systolic array architecture has been proposed [57]. An FFT systolic processor implementation has been reported in [58].

For the 16 point parallel pipeline FFT and for the RAMAT, systolic array architectures may be efficiently applied. The basic building blocks, the adder unit in the 16 point FFT unit and the small size RAM element are potential architectures for systolic array implementation. These units realized in systolic array architectures may be cost effective solutions in future high speed high order ($N \geq 64$) signal processing systems.

Appendix A

COMPUTER SIMULATION RESULTS

A.1 INTRODUCTION

Finite wordlength effects lead to a significant reduction in signal-to-noise ratio at the output of the FFT processors. The major sources of error are due to (1) A/D quantization (2) Coefficient quantization (3) Product quantization and (4) Scaling. The coefficient quantization effects are virtually eliminated in the proposed FFT processors by using stored-product-ROMs. By assuming a longer coefficient wordlength, the products can be accurately calculated using a general purpose computer, rounded to the desired wordlength and stored in the ROMs.

The dominant source of error in fixed point FFT processing is due to scaling, to avoid overflow. When two numbers of alike sign are added, there may or may not be overflow depending on the magnitude of the individual number. For example if two 4-bit numbers A=0.1101 and B=0.1000 are added the result becomes (A+B)=1.0101. This represents an error in two's complement arithmetic unless it is scaled (shifted to right), the number becomes 0.1010. This is equivalent to dividing by 2. If the shifted last bit is 0, there will not

be any error. However if the last bit is 1 (as in the example), scaling (by truncation) leads to error.

The computer simulation procedure is shown in Fig. A.1. A sequence of uniformly distributed random numbers is generated and used as input samples. Fixed point two's complement arithmetic is used. By using 'double precision', the ideal wordlength is set at 64 bits (Amdahl 470 Processor). A fast Fourier transform subroutine is used to find the FFT of a set of random numbers with out any quantization. Another FFT subroutine is used to find the transform, assuming finite wordlengths for input signal, coefficient and product values. The difference between the FFT outputs is found for each sample and the mean square error and the mean square signal are calculated to find noise/signal ratio at the FFT output.

A.2 16 POINT FFT PROCESSOR SIMULATION

The architectures of the 16-point radix-2 and radix-4 FFTs are simulated on the computer to find the mean square error and hence the noise-to-signal ratio at the output. Both decimation-in-time (DIT) and decimation-in-frequency (DIF) algorithms are considered for radix-2 algorithm. As described in chapter IV, for the 16 point radix-4 FFT, only the decimation-in-time algorithm is shown. Since for radix-4 16 point FFT, the decimation-in-time algorithm is

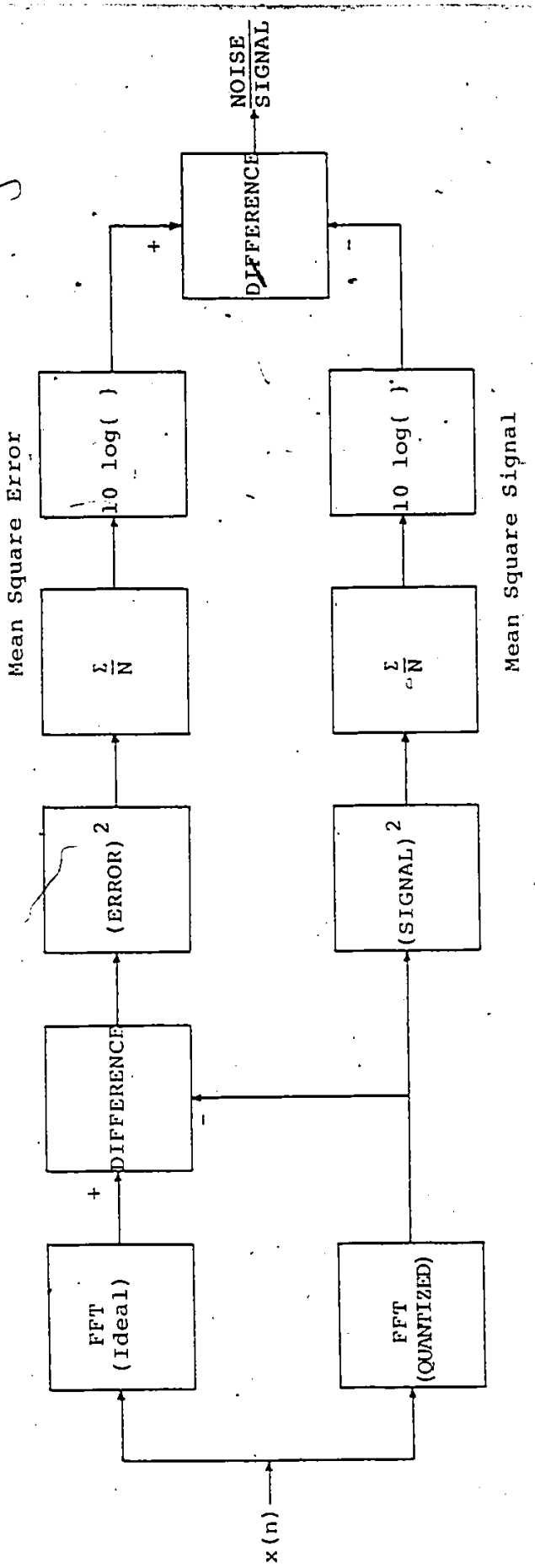


FIG. A.1 CALCULATION OF NOISE/SIGNAL RATIO AT THE FFT OUTPUT

equivalent to the decimation-in-frequency algorithm. The coefficient wordlength is set at 24 bits (ideal case) due to the stored-product ROM technique of multiplication. Because, the coefficient quantization errors become negligibly small for increasing wordlengths above 20 bits. Fig. A.2 shows noise/signal ratio at the FFT output without averaging. Fig. A.3 shows the results averaged over one hundred sets of complex samples. Scaling to avoid overflow at every stage of the FFT is considered for this simulation. The computer simulation results of the 16-point modified FFT are presented in chapter VI.

A.3 16*16 FFT PROCESSOR SIMULATION

The 16*16 point FFT processor is simulated under the same conditions as the 16 point FFT processor. Having done a rigorous computer simulation, it is found that scaling at each stage of the row FFT is sufficient, with no scaling in the column FFT. At the output of row FFT, the sample magnitude becomes so small that scaling in the column FFT is avoided. Figs. A.4 and A.5 show simulation results for cases (1) with no averaging and (2) with averaging.

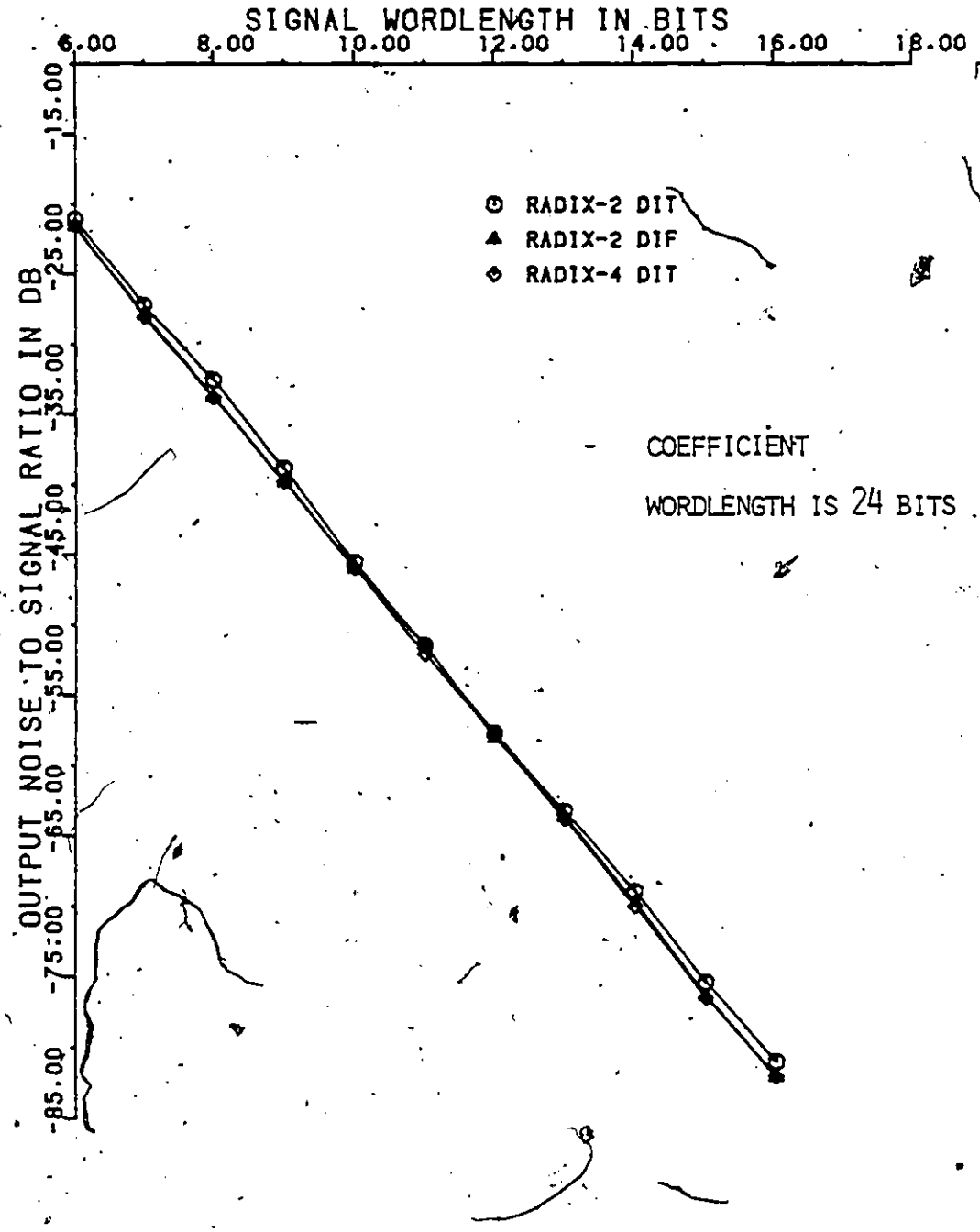


FIG. A.2 NOISE-TO-SIGNAL RATIO OF 16 POINT FFTs (NO AVERAGING)

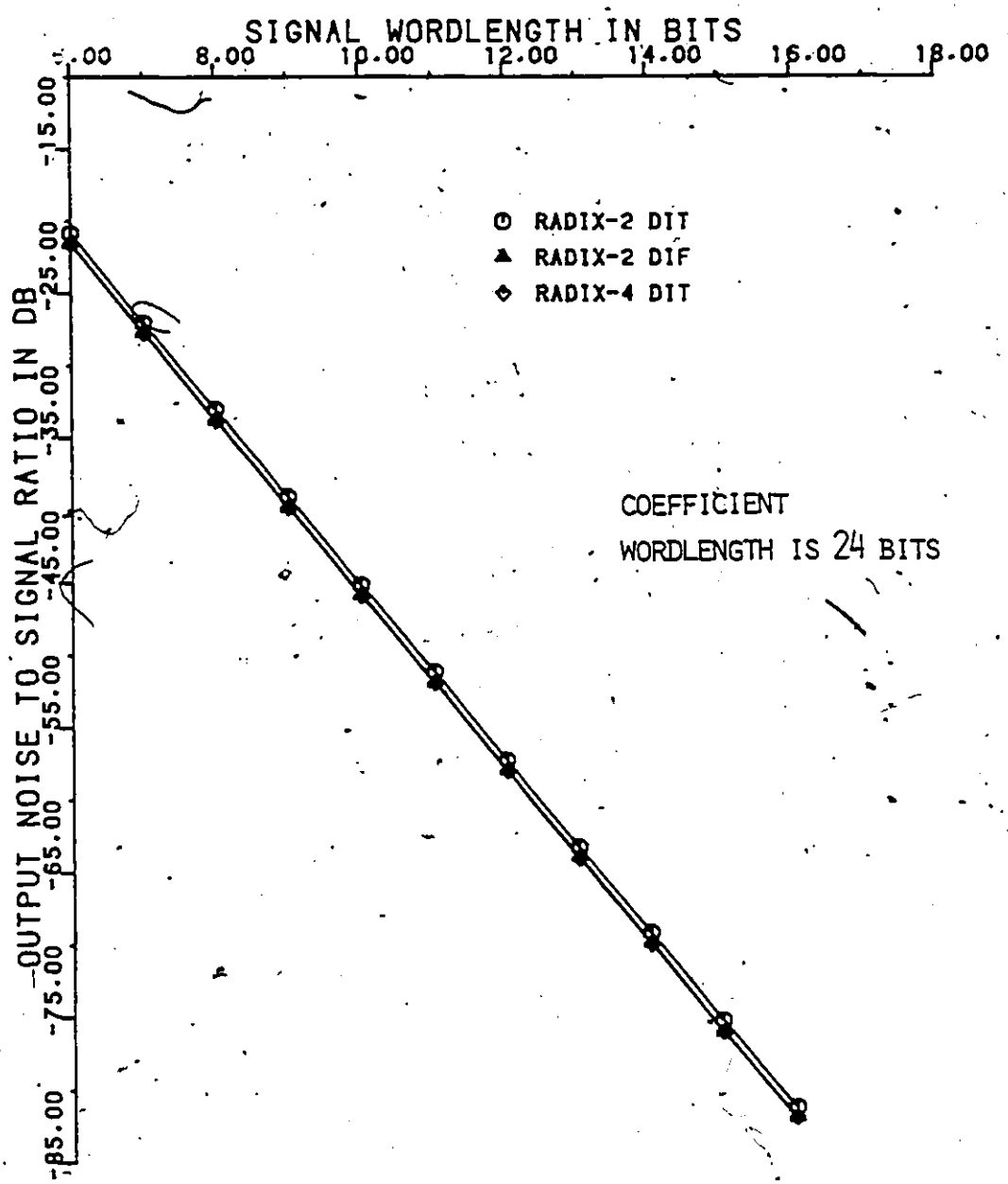


FIG.A.3 NOISE-TO-SIGNAL RATIO OF 16 POINT FFTs
(AVERAGING OVER 100 SETS OF SAMPLES)

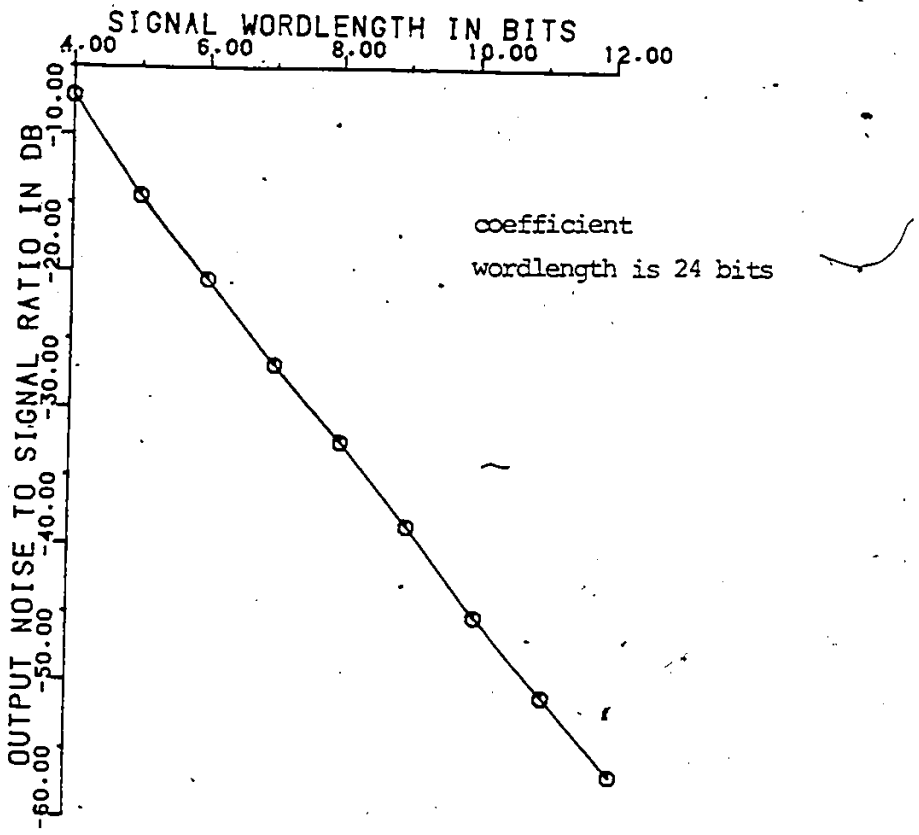


Fig. A.4 Noise/signal ratio of Two-dimensional(16 x 16) FFT

(No Averaging)

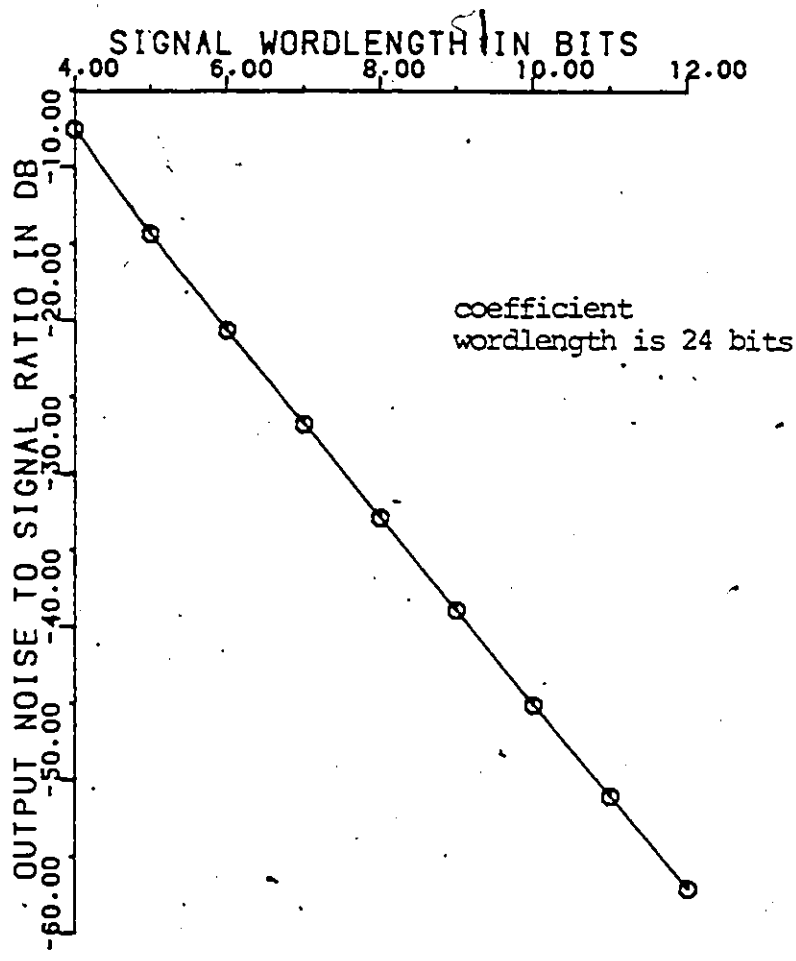


Fig. A-5 Noise/signal ratio of Two-dimensional (16 x 16) FFT

(Averaging over 100 sets of samples)

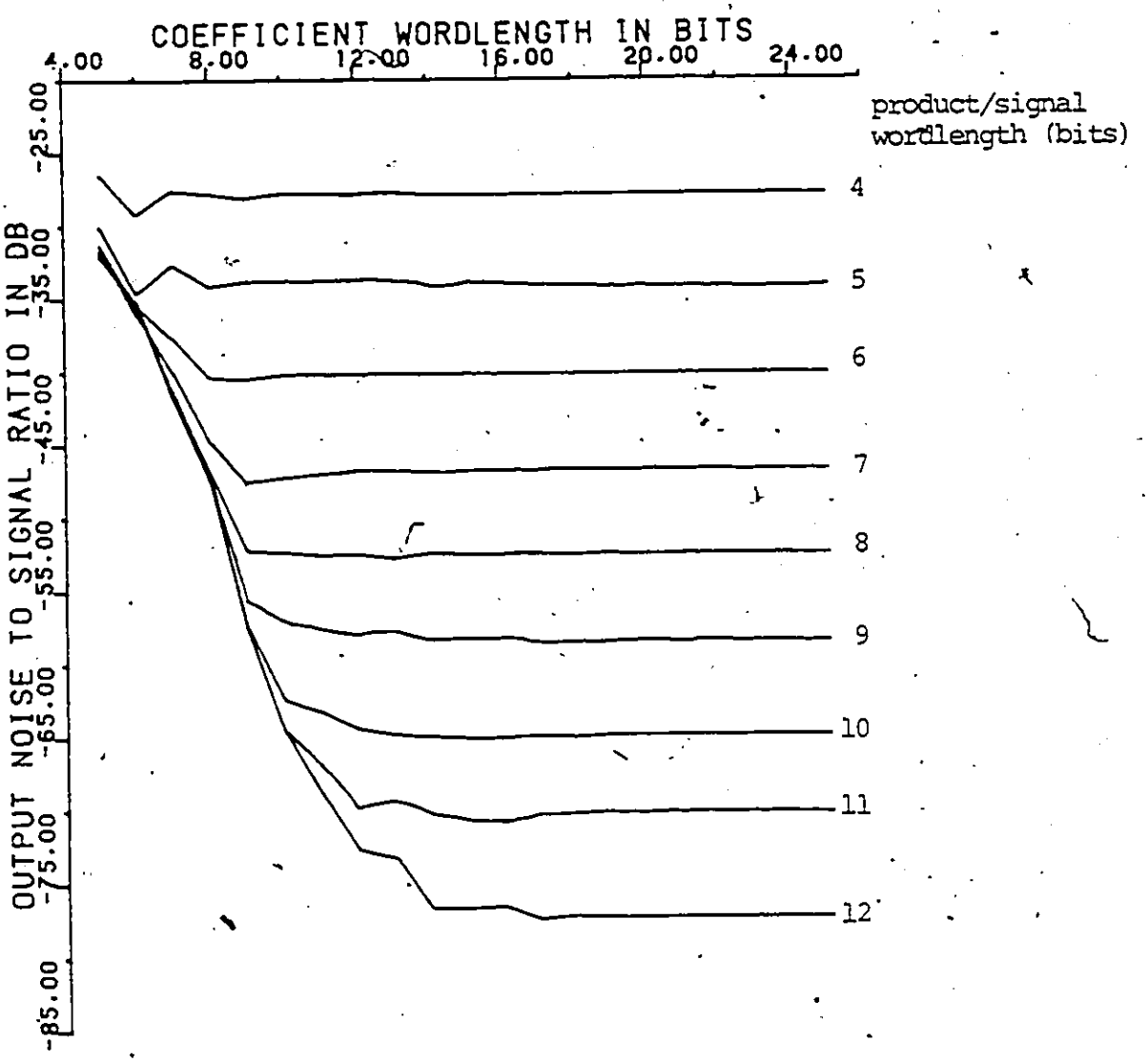


Fig. A.6 Noise/signal ratio of 64 point FFT

(modified 16 point FFT timeshared, output wordlength truncated)

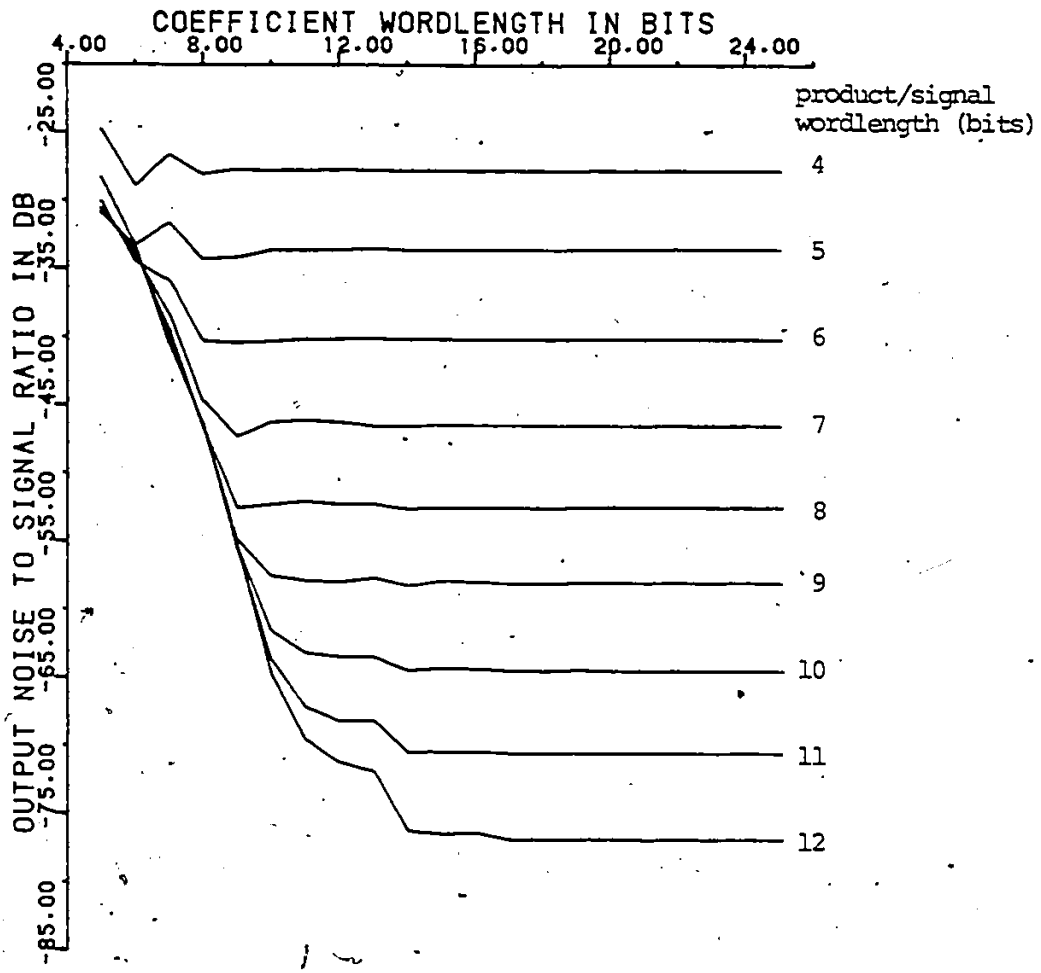


Fig.A.7 Noise/signal ratio of 256 point FFT

(modified 16 point FFT timeshared, output wordlength truncated)

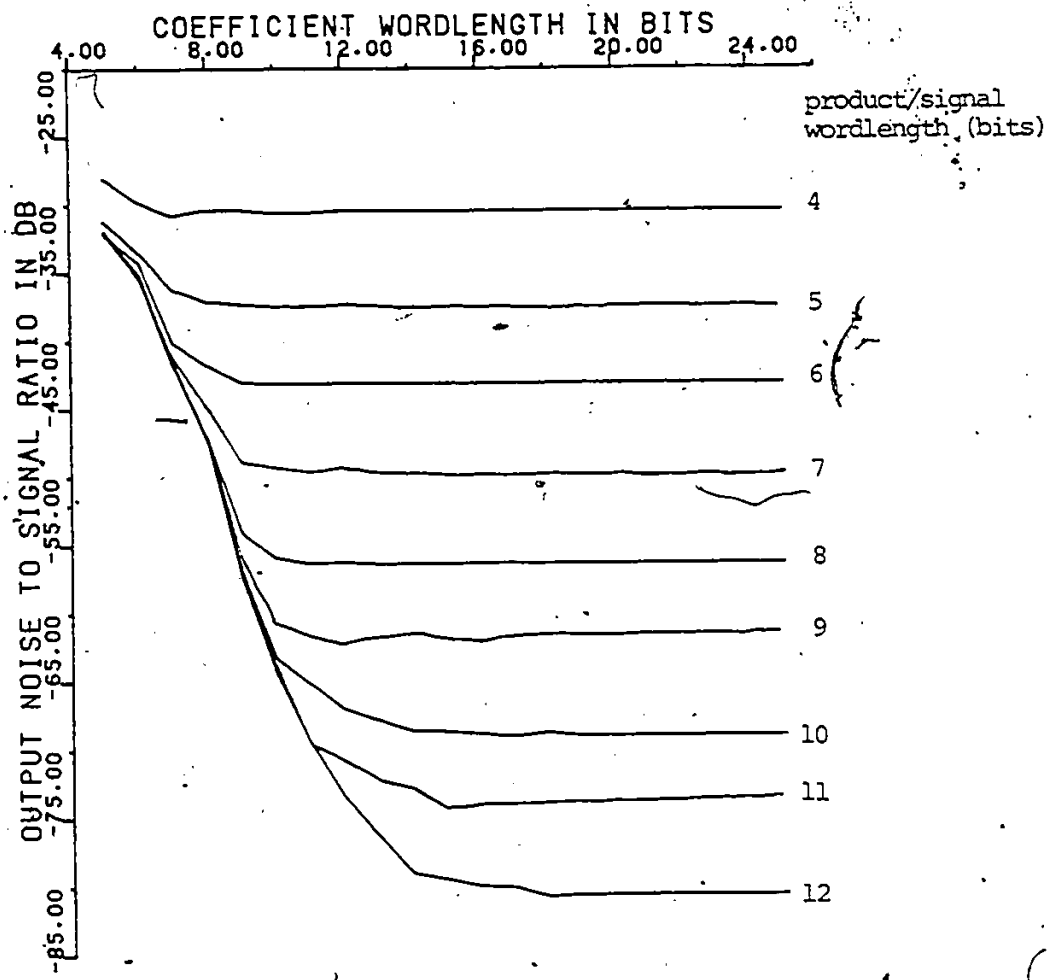


Fig. A.8 Noise/signal ratio of 64 point FFT

(modified 16 point FFT timeshared, output wordlength rounded)

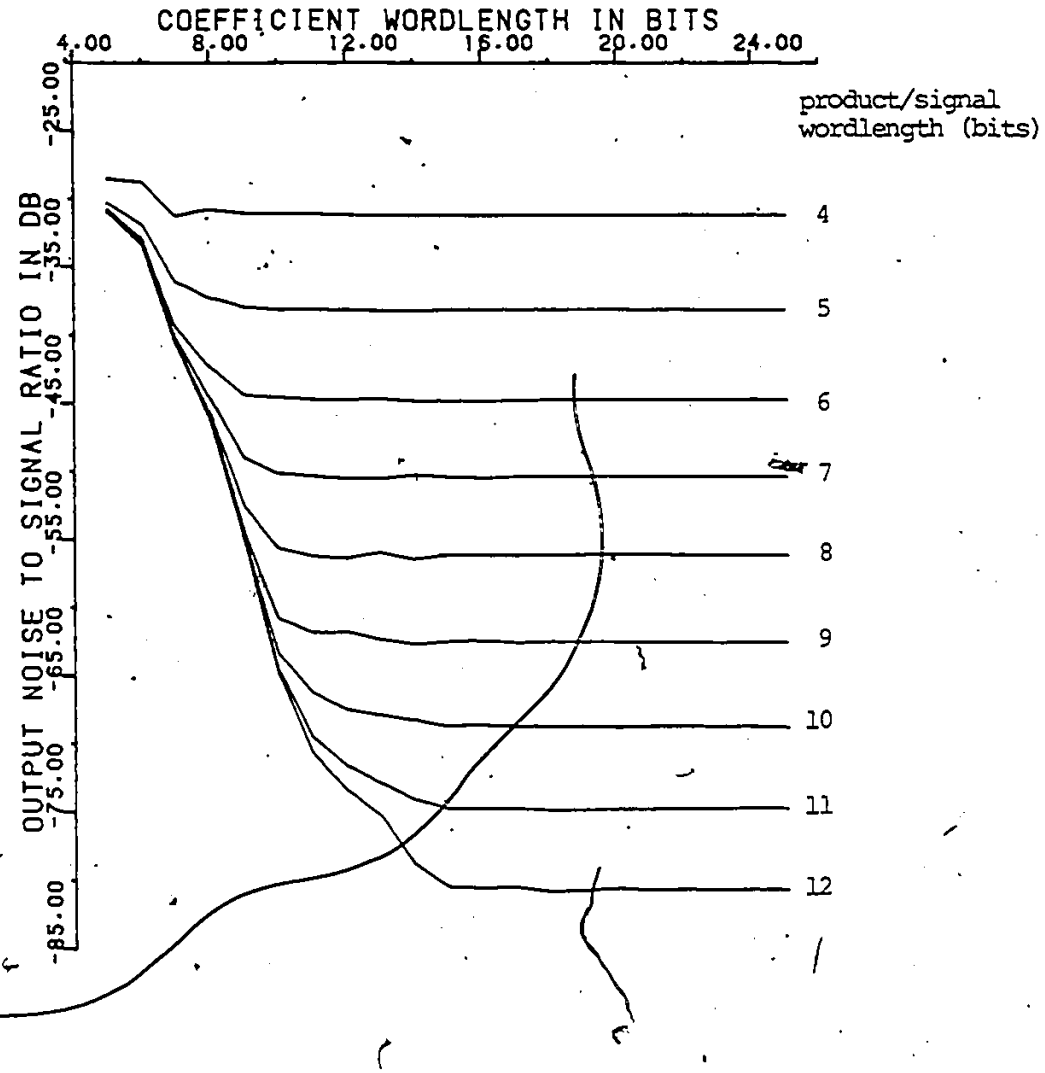


Fig.A.9 Noise/signal ratio of 256 point FFT

(modified 16 point FFT timeshared, output wordlength rounded)

A.4 64 AND 256 POINT FFT PROCESSORS SIMULATION

The radix-4 DIT FFT based 64 and 256 point FFT processors are simulated in a different way. Here, modified 16 point DIT FFT of increasing wordlength of 1 bit/stage is considered. Coefficient quantization effects are included. Figs. A.6 and A.7 show noise/signal ratio of 64 and 256 point FFT processors for the case of limiting the wordlength at the 16-point FFT output by truncation. Figs. A.8 and A.9 illustrate the simulation results for the case of rounding.

Signal/product wordlength for any hardware realization is determined by the required signal-to-noise ratio. Of course the complexity and cost of the processor implementation increases with the signal wordlength.

Appendix B

PRACTICAL CONSIDERATIONS OF HIGH SPEED FFT

The proposed high speed FFT prototype has been fabricated and tested at Interactive Circuits and Systems Ltd., for the Department of National Defence. The parallel pipeline radix-2 16-point FFT is considered for the realization. To reduce the initial prototype development cost and to examine the processor performance, an eight point FFT is used. To realize a 16-point FFT, the 8-point FFT unit is timeshared with appropriate twiddle factor multiplications at the input of a 16-point decimation-in-frequency (DIF) algorithm (Fig. 4.9). Fig. B.1 illustrates the implementation of 16-point FFT.

In a DIF butterfly signal flow graph, the output sample at the upper node is the sum of two input samples whereas the output sample in the lower node is the difference of the input samples multiplied by a twiddle factor coefficient [49]. For the hardware realization of the first stage butterflies, the sum samples appear at the upper output nodes earlier than the difference samples at the lower output nodes. Because the lower (difference) samples require complex multiplications it takes longer time to get the samples at the lower output nodes. Hence it is convenient to

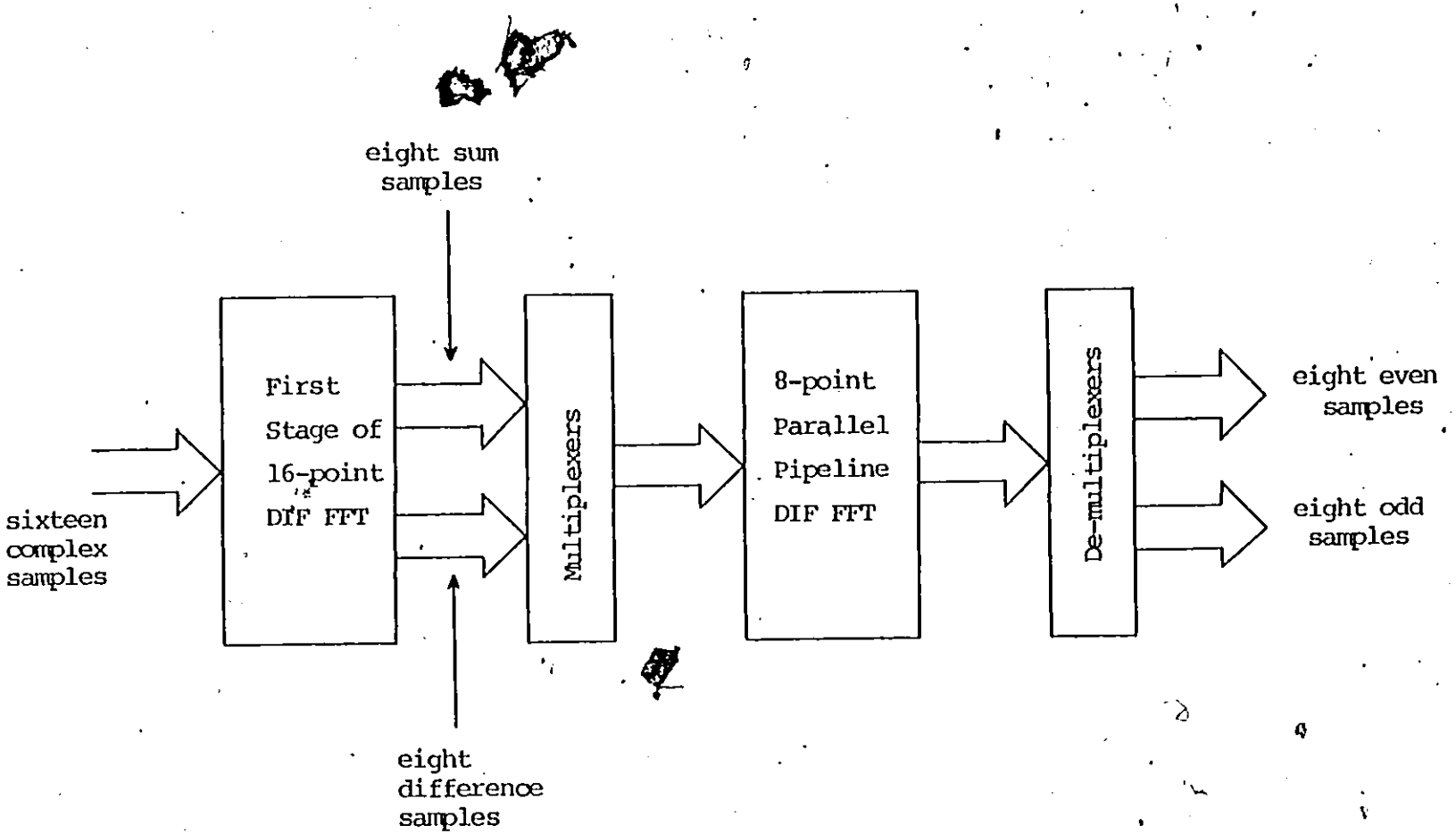


Fig.B.1 Practical Realization of 16-point FFT processor

timeshare a single 8-point FFT unit on the sum and difference samples of the first stage butterfly of the 16-point FFT. As soon as the sum samples are available at the output nodes of the first stage, the 8-point FFT unit starts processing these samples. Since the data movement in the 8-point FFT is in a parallel pipeline fashion, the 8-point FFT can accept the data as soon as the difference samples multiplications are completed. This way an eight-point FFT is multiplexed to complete a 16-point transform. This procedure leads to savings in hardware at the cost of a reduction in the throughput rate.

The radix-2 Adder Unit (AU) is shown in Fig. B.2. It consists of two adders and two subtracters. Since two's complement arithmetic is used, inverters are placed before the adders to achieve subtractions. Latches are used at the input and at the output for the parallel-pipeline operation. Three 4-bit adder chips are used to get 12-bit addition. Two 'Hex-Inverter' chips are used to get 12-bit inversion. Six 8-bit latches are used to hold the data of two sets of 12 bit complex samples each.

The radix-2 ROM multiplier array is shown in Fig. B.3. Four multipliers, each consisting of three (4Kx4) PROM chips are used. Two 12-bit adders, each consisting of three 4-bit parallel adders are included to add the product terms. Again latches are used to achieve parallel pipeline operation.

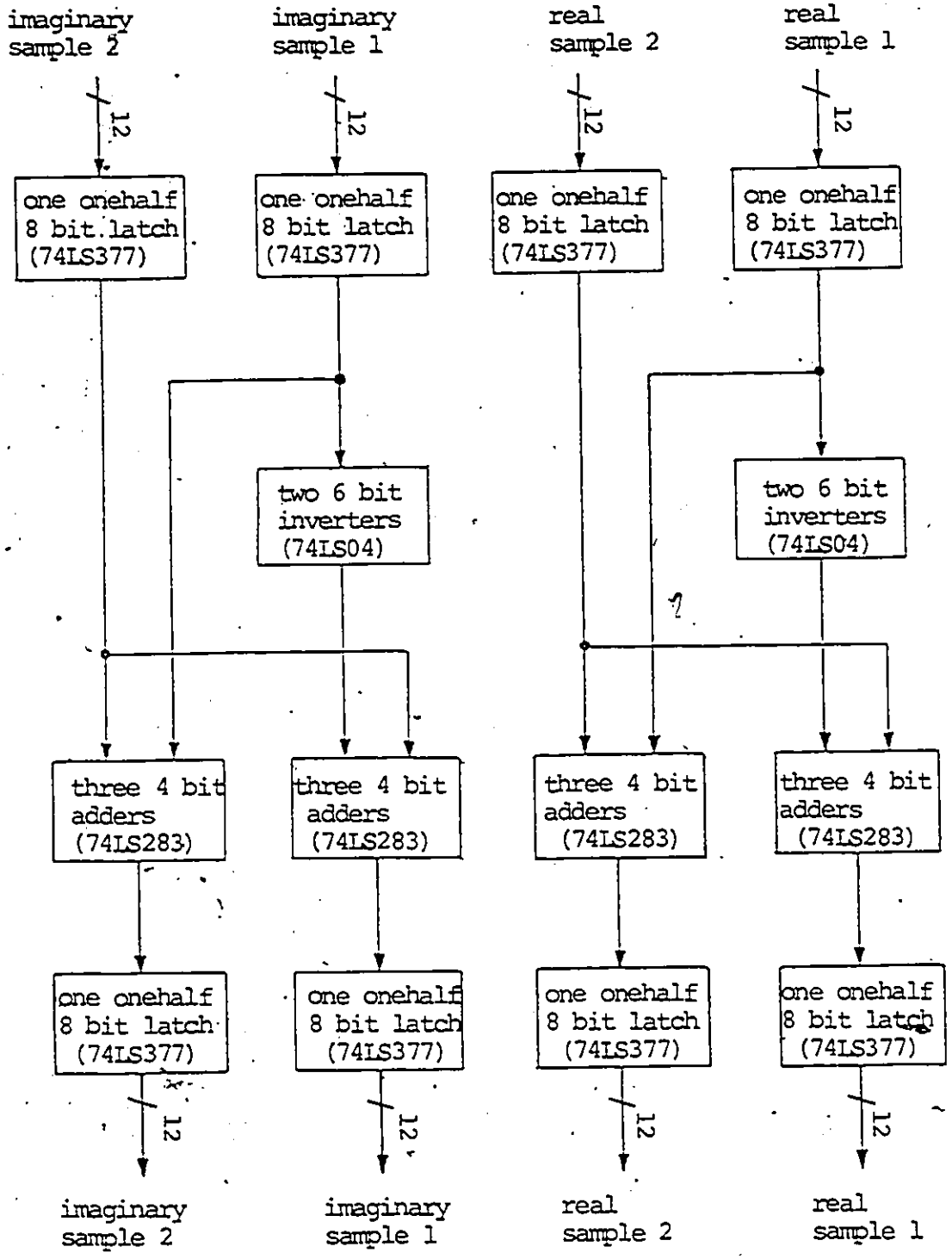


Fig.B.2 Practical Realization of Radix-2 Adder Unit

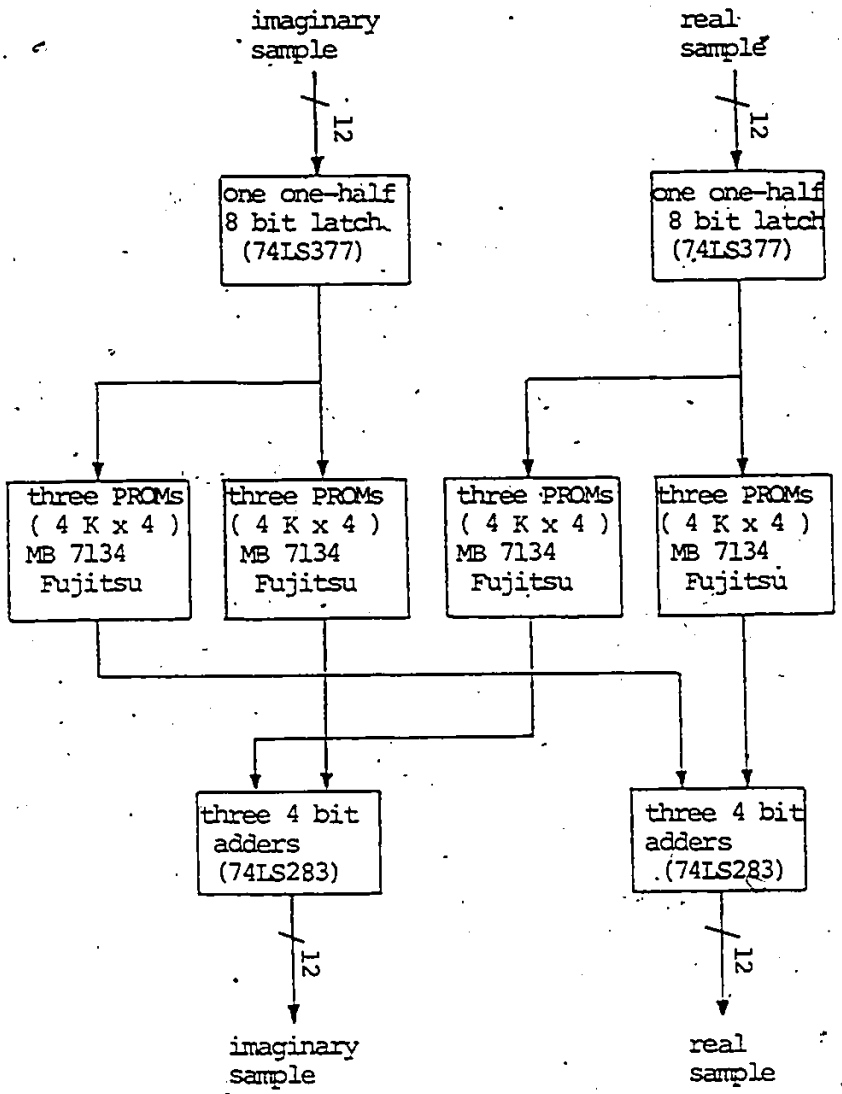


Fig.B.3 Practical Realization of ROM Array Multiplier

The fabricated 8-point FFT consists of 312 Integrated Circuit chips, consuming 42 watts of power. The per sample data throughput rate was 123 MHz. Instead of designing with the discrete logic components, VLSI techniques may be applied to reduce the number of Integrated Circuits and hence the power consumption.

REFERENCES

[1] J.W.Cooley, P.A.W.Lewis, and P.D.Welch, "Application of the Fourier Transform to Computation of Fourier Integrals, Fourier Series and Convolution Integrals", IEEE Trans. Audio and Electroacoustics, Vol.15, pp 79-84, June 1967.

[2] H.J.Nussbaumer, "Fast Fourier Transform and Convolution Algorithms", Springer-Verlag Berlin Heidelberg, New York, 1981.

[3] M.I.Skolnik, "Introduction to Radar Systems", McGraw-Hill, Inc., New York 1980.

[4] D.Mooney and G.Ralston, "Performance in clutter of Airborne Pulse MTI, CW Doppler, and Pulse Doppler Radar", IRE Convention Record, pp 55-62, 1961.

[5] L.P.Goetz and J.D.Albright, "Airborne Pulse Doppler Radar", IRE Trans. Military Electronics, Vol.5, No.2, pp 116-126, April 1961.

[6] W.W.Maguire, "Application of Pulsed Doppler to Airborne Radar Systems", IRE Conf. Aeronautical Electronics, pp 191-195, 1958.

[7] L.Armijo, K.W.Daniel, and W.M.Labuda, "Applications of the FFT to Antenna Array Beamforming", EASCON'74, pp 381-383, 1974.

[8] P.Barton, "Digital Beamforming for Radar", Proc. IEE, Vol.127, Part.F, No.4, pp 266-277, August 1980.

[9] R.G.Pridham and R.A.Mucci, "A Novel Approach to Digital Beamforming", Journal Acoustic Society of America, Vol.63, No.2, pp 425-434, February 1978.

[10] J.Litva, "Introduction to Sampled Aperture Radar Technology", Proc. Int. Electrical Electronics Conf. and Exposition, Toronto, pp 140-143, September 1983.

[11] C.Wu, B.Barkan, W.J.Karplus, and D.Caswell, "SEASAT Synthetic-Aperture Radar Data Reduction using Parallel Programmable Array Processors", IEEE Trans. Geoscience and Remote Sensing, Vol.20, No.3, pp 352-358, July 1982.

[12] K.Tomiyasu, "Tutorial Review of Synthetic-Aperture Radar (SAR) with Applications to Imaging of the Ocean Surface", Proc. IEEE, Vol.66, No.5, pp 563-583, May 1978.

[13] W.E.Arens, "Real-Time Synthetic Aperture Radar Data Processing for Space Applications", Society of Photo-Optical Instrumentation Engineers, Vol.154, pp 14-21, 1978.

[14] J.C.Kirk,Jr, "A Discussion of Digital Processing in Synthetic Aperture Radar", IEEE Trans. Aerospace and Electronic Systems, Vol.11, No.3, pp 326-337, May 1975.

[15] B.Friedlander, "A Comparison of Two SAR Processing Architectures for VLSI Implementation", Proc. Int. Conf. Acoustics, Speech, and Signal Processing, pp 919-922, 1983.

[16] W.D.Dickinson, "Processing Alternatives Tax Radar Designer's Bandwidths", Microwave System News, pp 95-108, September 1980.

[17] J.W.Cooley and J.W.Tukey, "An Algorithm for the Machine Calculation of Complex Fourier Series", Mathematics of Computation, Vol.19, No.90, pp 297-301, 1965.

[18] B.J.New and David Brian, "Bipolar VLSI facilitates Fourier transformation", Proc. Int. Conf. Circuits and Computers, pp 1084-1087, 1982.

[19] E.E.Wallingford, A.A.Sarkady, and H.M.Neustadt, "A Fixed Point FFT for a 16 Bit Microcomputer", Proc. Industrial and Control Applications of Microprocessors, pp 256-258, 1979.

[20] D.Bhagat and H.W.Mergler, "A High-Performance Microprocessor-Based FFT Processor", IEEE Trans. Industrial Electronics and Contol Instrumentation, Vol.25, No.2, pp 102-107, May 1978.

[21] S.Sweitzer, "A Low Cost FFT Chip Set", Proc. Int. Conf. Acoustics, Speech, and Signal Processing, pp 44.3.1-44.3.3, 1984.

[22] G.D.Covert, "A 32 Point Monolithic FFT Processor Chip", Proc. Int. Conf. Acoustics, Speech, and Signal Processing, pp 1081-1083, 1982.

[23] R.W.Linderman, P.P.Reusens, P.M.Chau and W.H.Ku, "Digital Signal Processing Capabilities of CUSP, a High Performance Bit-Serial VLSI Processor", Proc. IEEE Int. Conf. Acoustics, Speech, and Signal Processing, pp 16.1.1-16.1.4, 1984.

[24] B.Woo, L.Lin and F.Ware, "A Highspeed 32 Bit IEEE Floating-Point Chip Set for Digital Signal Processing", Proc. IEEE Int. Conf. Acoustics, Speech, and Signal Processing, pp 16.6.1-16.6.4, 1984.

[25] A.Rainer, W.Ulbrich and L.Gazsi, "Adder-based Digital Signal Processor Architecture for 80 nSec Cycle Time", Proc. IEEE Int. Conf. Acoustics, Speech, and Signal Processing, pp 16.9.1-16.9.4, 1984.

[26] J.K.Hartt and L.Sheats, "Application of Pipeline FFT Technology in Radar Signal and Data Processing", Eascon Record pp 216-221, 1971.

[27] G.O'Leary, "Non-Recursive Filtering using Cascade Fast Fourier Transformers", IEEE Trans. Audio and Electroacoustics, Vol.18, No.2, pp 177-183, June 1970.

[28] H.M.Halpern and R.P.Perry, "Digital Filters using Fast Fourier Transforms", Eascon Record pp 222-230, 1971.

[29] O.Monkewich and W.Steenaart, "Stored Product Digital Filtering with Non-Linear Quantization", Proc. IEEE Int. Symp. Circuits and Systems, pp 156-160, 1976.

[30] W.Steenaart and N.U.Chowdary, "Study of highspeed digital signal processing with applications to radar signals", Report on DSS contract 14SU.36001-1-2036, Department of Communications, Canada, April 1982.

[31] N.U.Chowdary and W.Steenaart, "Fast Fourier transform processor realization by means of stored-product-ROMs", Proc. 25th Midwest symposium on circuits and systems, pp 290-295", August 1982.

[32] J.D.Eklundh, "A Fast Computer Method for Matrix Transposing", IEEE Trans. Computers, Vol.21, pp 801-803, July 1972.

[33] W.Steenaart and N.U.Chowdary, "Realtime Radar Signal Processors", Report on DSS Contract 19SV.97714-2-1254, Department of National Defense, Canada, March 1984.

[34] N.U.Chowdary and W.Steenaart, "Very High Speed Fourier Transform Processor Realization Techniques", Proc. Int. Symp. Circuits and Systems, pp 26-29, 1983.

[35] N.U.Chowdary and W.Steenaart, "A High Speed Two-Dimensional FFT Processor", Proc. Int. Conf. Acoustics, Speech, and Signal Processing, pp 4.11.1-4.11.4, 1984.

[36] P.D.Welch, "A Fixed-Point Fast Fourier Transform Error Analysis", IEEE Trans. Audio and Electroacoustics, Vol.17, pp 151-157, June 1969.

[37] C.J.Weinstein, "Roundoff Noise in Floating-Point Fast Fourier Transform Computation", IEEE Trans. Audio and Electroacoustics, Vol.17, pp 209-215, September 1969.

[38] N.U.Chowdary and W.Steenaart, "Accumulation of Product Roundoff Errors in Modified FFT Algorithms", Proc. Int. Symp. Circuits and Systems, pp 272-275, 1984.

[39] N.U.Chowdary and W.Steenaart, "Decimation-In-Frequency Algorithm - An Alternative for Implementing the Modified FFT", Submitted for Publication.

[40] N.R.Gillespie, J.B.Higley and N.MacKinnon, "The Evolution and Application of Coherent Radar Systems", IRE Trans. Military Electronics, Vol.5, No.2, pp 131-139, April 1961.

[41] D.E.Dudgeon, "Fundamentals of Digital Array Processing", Proc. IEEE, Vol.65, No.6, pp 898-904, June 1977.

[42] J.C.Kirk.Jr, "A discussion of digital processing in synthetic aperture radar", IEEE trans. on Aerospace and Elecronic systems, Vol.11, No.3, May 1975.

[43] S.A.Hovanessian, "Radar Detection and Tracking Systems", ARTECH, 1973.

[44] S.A.HovanessianX "Introduction to Synthetic Array and Imaging Radars", ARTECH, 1980.

[45] D.J.Bonfield and J.R.E.Thomas, "Synthetic-aperture-radar Real-time Processing", Proc. IEE, Vol.127, part. F, No.2, pp 155-162, April 1980.

[46] D.A.Ausherman, "Digital versus Optical Techniques in Synthetic Aperture Radar (SAR) Data Processing", Optical Engineering, Vol.19, No.2, pp 157-167, March/April 1980.

[47] A.Peled and B.Liu, "A New Hardware Realization of Digital Filters", IEEE Trans. Acoustics, Speech, and Signal Processing, Vol.22, pp 456-462, December 1974.

[48] O.Monkewich and W.Steenaart, "Companding for Digital Filters", Proc. Int. Symp. Circuits and Systems, pp 68-71, 1973.

[49] L.R.Rabiner and B.Gold, "Theory and Application of Digital Signal Processing", Prentice-Hall, Inc, New Jersey, 1975.

[50] A.V.Oppenheim and R.W.Schafer, "Digital Signal Processing", Prentice-Hall, Inc, New Jersey, 1975.

[51] T.Thong and B.Liu, "Fixed-point Fast Fourier Transform Error Analysis", IEEE Trans. Acoustics, Speech, and Signal Processing, Vol.24, No.6, pp 563-573, December 1976.

[52] H.L.Groginsky and G.A.Works, "A pipeline fast Fourier transform", IEEE trans. on Computers, Vol.C-19, No.11, November 1970.

[53] H.T.Kung, "Let's Design Algorithms for VLSI Systems", Proc. Conf. Very Large Scale Integration, pp 65-90, January 1979.

[54] J.L.Potter, "Image Processing on the Massively Parallel Processor", IEEE Computer Magazine, Vol.16, No.1, pp 62-67, January 1983.

[55] A.V.Oppenheim and C.J.Weinstein, "Effects of Finite Register Length in Digital Filtering and the Fast Fourier Transform", Proc. IEEE, Vol.60, pp 957-976, August 1972.

[56] G.H.Allen, P.B.Denyer and D.Renshaw, "A Bit Serial Linear Array DFT", Proc. Int. Conf. Acoustics, Speech, and Signal Processing, pp 41A.1.1-41A.1.4, 1984.

[57] B.R.Mercy, "Systolic Array Technique Applied to Symmetric FIR Filters", Proc. Int. Conf. Acoustics, Speech, and Signal Processing, pp 34A.1.1-34A.1.4, 1984.

[58] T.Willey, T.S.Durrani and R.Chapman, "An FFT Systolic Processor and its Applications", Proc. Int. Conf. Acoustics, Speech, and Signal Processing, pp 34A.4.1-34A.4.4, 1984.