RCA

January 1982
JPL Contract 955801

FINAL REPORT

ADVANCED DIGITAL SAR PROCESSING STUDY

BY

LLOYD W. MARTINSON
BRIAN P. GAFFNEY
BEDE LIU
RICHARD P. PERRY
ABRAHAM RUVIN

This work was performed for the Jet Propulsion Laboratory, California Institute of Technology, sponsored by the National Aeronautics and Space Administration under Contract NAS7-100.

RCA Government Systems Division
Missile and Surface Radar
Moorestown, NJ 08057

January 1982
JPL Contract 955801

FINAL REPORT

ADVANCED DIGITAL SAR PROCESSOR STUDY

BY

LLOYD W. MARTINSON
BRIAN P. GAFFNEY
BEDE LIU
RICHARD P. PERRY
ABRAHAM RUVIN

This work was performed for the Jet Propulsion Laboratory, California Institute of Technology, sponsored by the National Aeronautics and Space Administration under Contract NAS7-100.

RCA/Government Systems Division
Missile and Surface Radar
Moorestown, N.J. 08057

ABSTRACT

A design of a highly programmable, land based, real time synthetic aperture radar (SAR) processor requiring a processed pixel rate of 2.75 MHz or more in a four-look system was conducted. Variations in range and azimuth compression, number of looks, range swath, range migration and SAR mode were specified. A number of alternative range and azimuth processing algorithms were examined and analyzed in conjunction with projected integrated circuit, digital architecture, and software technologies.

The selected design for the Advanced Digital SAR Processor (ADSP) employs an FFT convolver algorithm for both range and azimuth processing in a parallel architecture configuration. This overall design approach met all of the system implementation and performance criteria for programmability, modularity, adaptability to VLSI, low risk, reliability and cost.

The report provides algorithm performance comparisons, a detail design of the selected system, implementation tradeoffs and the results of a supporting survey of integrated circuit and digital architecture technologies. Cost tradeoffs and projections with alternate implementation plans are presented.

TABLE OF CONTENTS

TABLE OF CONTENTS (continued)

TABLE OF CONTENTS (continued)

LIST OF ILLUSTRATIONS

LIST OF ILLUSTRATIONS (continued)

LIST OF ILLUSTRATIONS (continued)

LIST OF TABLES

LIST OF TABLES (continued)

1.0 INTRODUCTION

A general practice in synthetic aperture radar (SAR) systems has been to use a specially tailored processor to convert the raw radar signals to imagery. The advent of practical digital signal processing techniques, brought about by LSI technology, has made high performance, multi-mission programmable SAR digital signal processors feasible. This is the final report of a study to quantify various issues related to the hardware development of such a SAR processor. The generic name for the processor is the Advanced Digital SAR Processor and it shall be referred to from this point on in the report as the ADSP.

A very broad set of performance goals have been set by the Jet Propulsion Laboratory (JPL) for the ADSP. These are discussed in Section 2.1 together with the system performance requirements.

Two principle design facets are examined, processing algorithms and technology. The algorithms are further divided into time and frequency domain techniques; while the broad issue of technology includes digital integrated circuits, digital architecture and software. Section 2.2 gives a comprehensive description of the candidate processing algorithms together with hardware implementation approaches for them.

A key factor in the selection of a processing algorithm is the quality of imagery it produces. Extensive computer simulations have been developed and run to evaluate the performance of candidate algorithms. This data, given in Section 1.3, was then applied as appropriate to the hardware sizing of the various implementations.

Technology is addressed in Section 2.4 with the current state of the art and projections provided for digital integrated circuits, architecture and software.

A design recommendation for the ADSP is synthesized and described in Section 2.5. It meets all of the performance and implementation goals of the ADSP.

Key issues in the selection of an algorithm for the ADSP were its effectiveness in compensating for range migration, progammability for multiple modes, performance level and the inherent computation

requirements and memory storage. The selection of an algorithm was closely tied into the technology issues, particularly digital integrated circuits and architecture. The design selected minimizes memory storage and offers simple memory management techniques. It provides a programmable architecture in modular form which offers both incremental development and growth potential with little waste of effort. Finally, the design has inherently high reliability and maintainability features.

Cost factors, alternative development plans and tradeoffs are presented in Section 2.6 and 2.7.

Software developed under the program is provided in the appendix.

2.1 ADSP SYSTEM REQUIREMENTS

A satellite SAR system is depicted in Figure 1. As the radar moves along its flight path, multiple radar pulses are coherently processed to achieve a resolution corresponding to the length of a synthetic antenna. This length is determined by the footprint of the real antenna on the surface. The limiting resolution of the SAR is thus determined by the size of the real antenna and is equal to one half its diameter. To achieve radar returns which depict the contours of the surface, an oblique incidence angle is employed. In addition, a forward look angle off broadside is common. Resolution in the cross track, or range, dimension is achieved by using conventional radar pulse compression techniques. The total range swath covered is also constrained to the dimensions of the radar footprint.

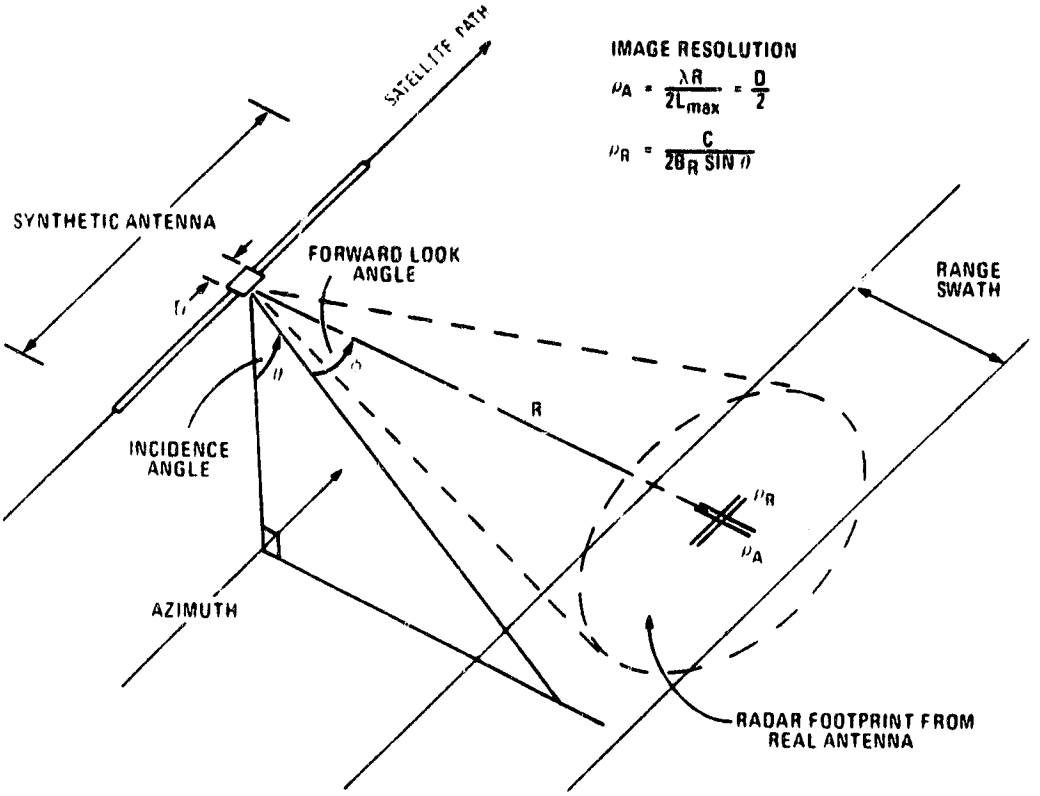


Figure 1.
SATELLITE SAR PARAMETERS

2.1.1 Functional Requirements

The ADSP requirements have been established in anticipation of processing a wide variety of SAR data. SAR missions which represent a comprehensive set for design parameter purposes are given in Table 1. Both satellite and aircraft SAR are represented.

Table 1
SAR Mission Set

MISSION SET	RESOLUTION (m)	SWATH WIDTH (km)	NUMBER OF LOOKS	INCIDENT ANGLE	FREQUENCY	POLARIZATION*	ALTITUDE (km)
SEASAT	25	100	4	22.54	1275	S	800
SIR REFLIGHT	40	50	7	20-70	1275	S	210-280
AIRCRAFT							
L	10-20	18-35	8	0-60	1275	S	3-12
X	10-20	18-35	8	0-60	9600	S	3-12
VOIR							
HI-RES 52^0	75	TBD	4	52	1275	S	250
LO-RES 52^0	300	30	30	52	1275	S	250
ICEX	150	360	6	36.42	9600	S	700
ERSAR L**	15	40	4	60^0	1275	D	300
ERSAR X**	15	40	4	60^0	9600	D	300

* S - SINGLE, D - DUAL
** PRELIMINARY

The most stressing signal processing requirement of the missions in Table 1 is the ERSAR (Earth Resources SAR) which has been selected by JPL for the baseline design mission for the ADSP. Functional requirements for the baseline mission are given in Table 2.

Both range and azimuth processing requirements are included. Specifications are also provided for range migration correction, reference function update; integrated sidelobe level and output data format.

Table 2
ADSP Baseline Functional Requirements

RANGE CORRELATOR	
INPUT SAMPLE RATE (COMPLEX WORDS)	11MHz
RANGE COMPRESSION RATIO	410
RANGE PULSE LENGTH	475 (COMPLEX SAMPLES)
RANGE INTEGRATED SIDELOBE RATIO DUE TO PROCESSING	-15dB
AZIMUTH CORRELATOR	
SWATH WIDTH (OUTPUT PIXELS)	4,000
RADAR PRF	2,500Hz
NUMBER OF AZIMUTH LOOKS (CONCURRENTLY PROCESSED)	4
AZIMUTH BANDWIDTH PER LOOK	500Hz
AZIMUTH COMPRESSION RATIO PER LOOK	350
RANGE WALK COMPENSATION ACROSS 1 LOOK	28 SLANT RANGE BINS
RANGE WALK COMPENSATION ACROSS 4 LOOKS	86 SLANT RANGE BINS
RANGE CURVATURE COMPENSATION ACROSS 4 LOOKS	7 SLANT RANGE BINS
REFERENCE FUNCTION UPDATING RATE	CROSS-TRACK EVERY 8 SAMPLES; ALONG TRACK EVERY 12,000 RADAR PULSES
AZIMUTH INTEGRATED SIDELOBE RATIO DUE TO PROCESSING	-20dB
OUTPUT PIXEL FORMAT (BOTH OPTIONS A AND B ARE REQUIRED)	A) SELECTABLE 12 BITS IN AMPLITUDE OVER 80dB DYNAMIC RANGE B) COMPLEX SINGLE LOOK PIXELS, 12 BITS I, 12 BITS Q

Changes in operating mode are anticipated to occur within a SAR mission and these changes could involve, altitude, look angle, resolution, number of looks, swath width and the range and/or azimuth weighting functions. The processor is required to operate in a continuous or burst mode. The former has the sensor emitting non-changing parameters. The latter interleaves alternate bursts with different parameters, yet permits contiguous imagery formation. Alternate bursts may involve different radar frequencies, different polarizations or different antenna pointing angles.

Design objectives of the ADSP include programmable features to permit range compression from 10 to 600, azimuth compression from 20 to 350 per look and swath width from 500 to 4000 pixels. A desired feature provides the capability to tradeoff looks, resolution and swath width within the capabilities of the processor.

Modularity in design was a key functional design objective. This feature could provide greater swath width (up to four times the baseline), more parallel looks and multiple frequencies and polarizations.

2.1.2 ADSP System Scope

The principal elements of the ADSP as shown in Figure 2 are an input data handling facility to permit either real time or recorded data to be processed, a range correlator, an azimuth correlator and a multi-look integrator. All of the high speed processing hardware is encompassed in the pipeline from data input to output. The host computer is used to set up conditions for mission operation. These will include SAR parameters and overall control commands. The hardware itself will respond to the program commands and under firmware

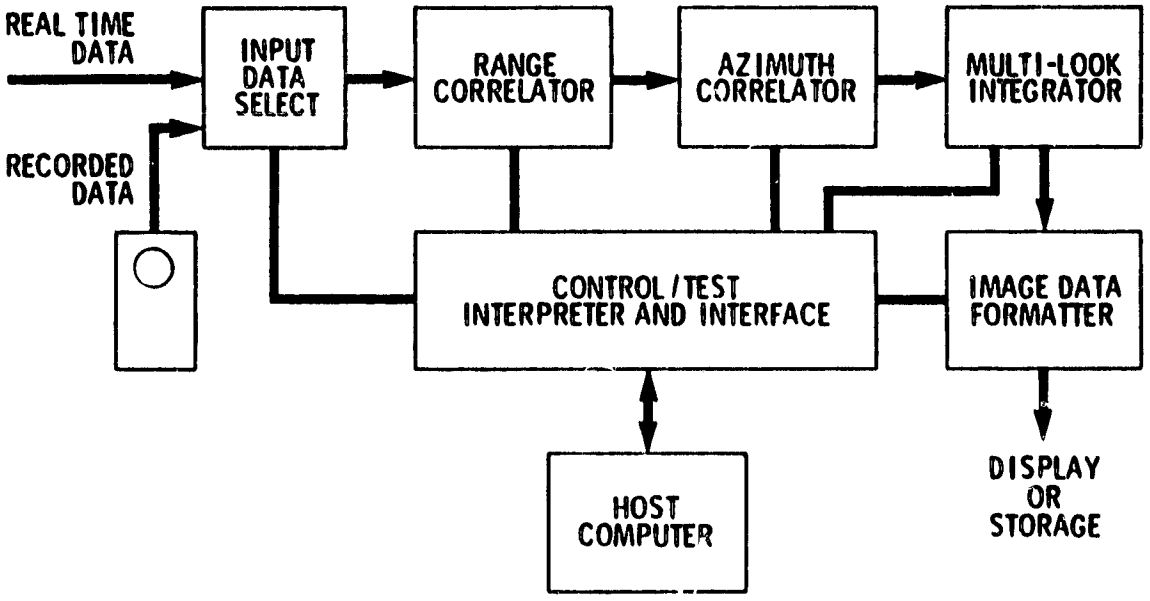


Figure 2

ADSP Elements

control will complete the required operations. A goal of the system design was to maintain software simplicity for the host computer by incorporating as much as possible of the SAR variables in the hardware.

2.1.3 Analysis of ADSP Requirements

2.1.3.1 SAR Parameter Variation

The azimuth resolution for operation near broadside is given by:

$$\delta_{AZ} = \frac{K_a \lambda R_s}{2 V_s T_L}$$

where R_s = slant range, λ = wavelength, V_s = satellite velocity and T_L = coherent look time. The azimuth beam width factor K_a, is associated with the receive weighting required to achieve a desired sidelobe level. For -36 dB sidelobe level, K_a = 1.2 for a Taylor $\bar{n}$ = 8 weighting function. This weighting produces an integrated sidelobe level, ISL, of -25 dB. When other sources of sidelobes are included such as digital granularity and the lack of interpolation in range cell migration compensation, an overall ISL of -20 dB is obtained. The chirp bandwidth associated with the coherent look time is given by: given by:

$$B_L = \frac{2 V_s^2 T_L}{\lambda R_s} = \frac{K_a V_s}{\delta_{AZ}}$$

For the ERSAR case V_s = 7.73 km/sec, δ_{AZ} = 15 m, K_a = 1.2 and B_L = 618.4 Hz for each look. At λ = 0.235 m, T_L = 0.686 sec., $B_L T_L$ = 424 and the angle compression during the look angle = 424/1.2 = 353.

For four looks the total processed bandwidth = $4B_L$ = 2473.6 Hz. The azimuth beamwidth that just matches this spectrum is given by $\theta_B = 4V_s T_L / R_S = 0.0375\ (2.15^o)$. The antenna length required for this beamwidth is $L_A = .886\ \lambda / \theta_B$ = 5.55 meters, assuming uniform illumination. The PRF of the ERSAR = 2500. The ratio of the PRF to the four-look bandwidth = 1.01. This ratio is lower than it should be for low doppler aliasing (angle grating lobes). A calculation shows that the integrated grating lobes from the four looks is only -8.2 dB. A higher ratio of PRF to four-look bandwidth would be required for reduced grating lobes; however, the range grating lobes would then be

required for reduced grating lobes; however, the range grating lobes would then be excessive unless the swath were reduced by the same ratio that the PRF was increased. The sampled range to allow for a 40 km swath at 60^{o} incidence and a 35.2 μsec transmitted pulse requires 266.1 μsec. The ratio of the 2500 PRF period to 266.1 μsec = 1.503 is a reasonable value to inhibit range grating lobes adequately. A lower ratio of periods caused by a higher PRF would cause the range grating lobes to appear.

SAR mission sets given in Table 1 can be evaluated as a function of the number of looks, swath width and required resolution for the optimum PRF and antenna parameters. For the case of dual frequency such as ERSAR-L and ERSAR-X, the real antenna lengths, PRFs, and look bandwidths would be identical. However, the antenna widths and angle compression should be scaled by the wavelength. Dual polarization reception could be simultaneous, if desired. However, dual polarization transmission would have to be orthogonal either by frequency or time. The possibility for change in look angle depends upon an adequate ratio of PRF period to A/D sampling time for the desired swath. Electronic scanning or beam switching can be employed to change the look angle if the ratio is less than 1.5 The requirement for a "burst mode" involving interleaving different frequencies, polarization, and pointing angle may also require azimuthal electronic scan. This is discussed in Section 2.1.3.2. Of course, several PRFs may be required to center the swath window between the ambiguous transmission pulses to avoid eclipsing if the look angle or orbit attitude is changed.

2.1.3.2 Mode Variations

The ADSP is required to have the flexibility to process data from a wide variety of different missions described in Table 1 and to accommodate continuous or burst modes.

Figure 3 shows the case for four looks in the continuous mode integration for azimuth cell j, where Look 4 coherently integrates the first $N_A/4$ samples and Look 1 coherently integrates the last $N_A/4$ samples over the azimuth beam. The four looks are noncoherently integrated and

registered at the correct range and azimuth cell. As each new azimuth sample is received, a beam is completed for each look and one final integrated multilook azimuth image sample is completed.

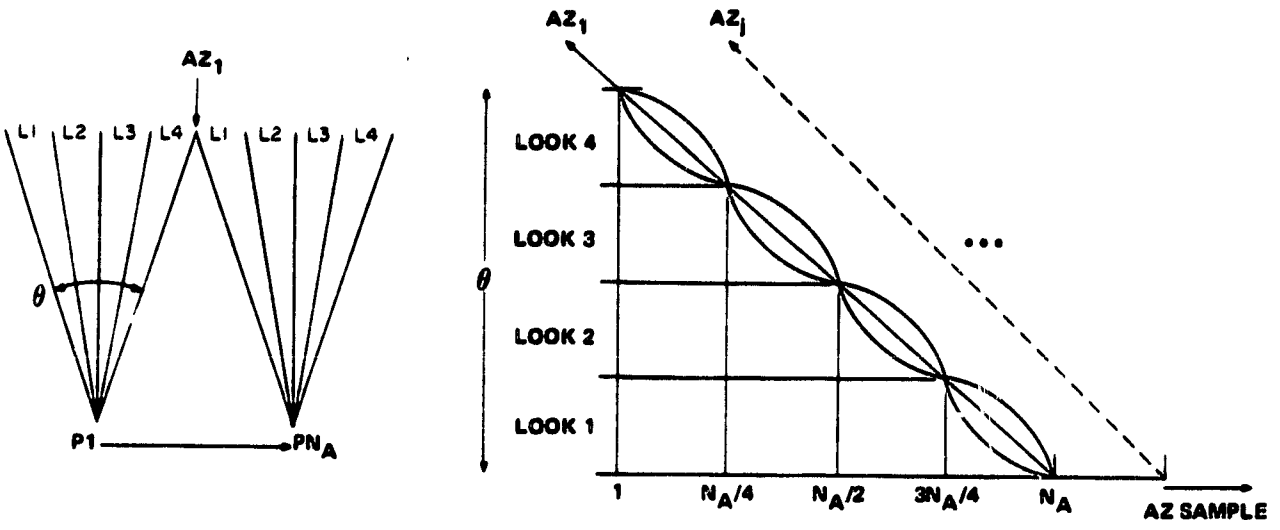


Figure 3
Continuous Mode Integration Requirements

In the burst mode, a number of variations are possible. When bursts are interleaved and do not involve frequency changes, the integration patterns shown in Figure 4 are obtained. In the case shown in Figure 4a, two polarizations are interleaved by bursts in a four-look mode. The result is that the output imagery is available on a two-look two-polarization basis. A polarization comparison of image samples can then be made. The processing can be identical in form to the continuous case except that inefficiencies occur in that not all of the antenna coverage is integrated and used in the final imagery. This is because the field must be illuminated for a full look before the data can be processed to obtain a full integrated signal-to-noise ratio with the desired resolution.

A large number of looks can be transmitted on an interrupted basis as shown in Figure 4b. In this case, 32 looks are processed. Look integration procedures may be varied as a function of time, if the burst sequences are short. In the limiting case, each look can be an unfocussed segment of the entire beam. The beam is then split in angle into the number of "looks" processed. The Multiple burst looks must be placed in proper range registration prior to integration.

The conditions change if bursts of different frequencies are transmitted. If we assume that the antenna size remains the same, the high frequency has a proportionally narrower beam. Thus, if full coverage is desired, multiple bursts at different pointing angles must be transmitted. This is illustrated in Figure 5. For simplicity, the example of high frequency is four times the lower frequency and four looks are employed. The first burst at frequency F_1 provides the four-look coverage indicated. Quadrupling the frequency narrows the antenna beam proportionally and also increases the ground resolution by the same factor. To obtain the same final image resolution, only one fourth as many pulses need be transmitted per burst. But since the beam is narrower, it must be steered to a total of four angles with a separate burst transmitted at each angle. Four looks are processed at the high frequency also to maintain the final image resolution. A strip map can be generated in this way which has one look at each frequency for each image location.

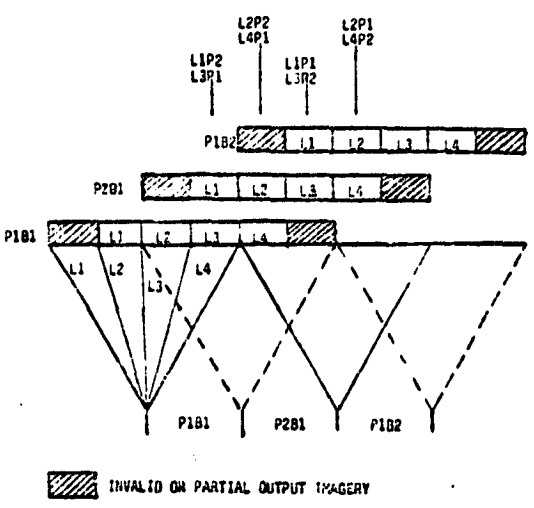


FIGURE 4a. Contiguous, Two-Polarization Bursts

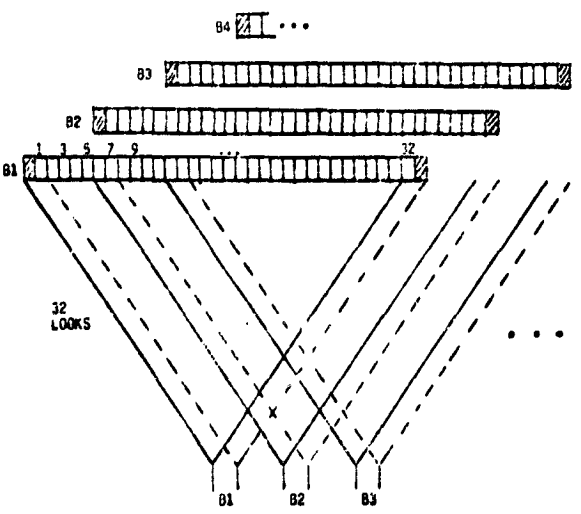


Figure 4b. Interrupted Bursts, 32 Looks

Figure 4 Interleaved Polarization Burst Mode

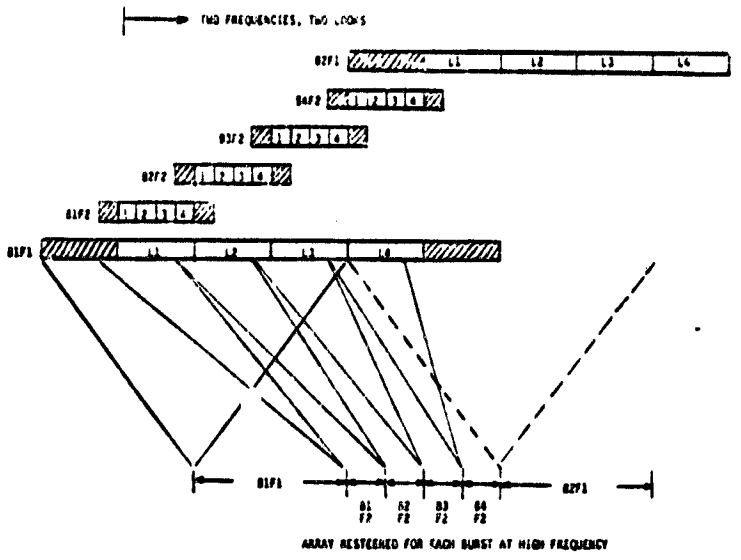


Figure 5
Interleaved Bursts At Two Frequencies

2.1.3.3. Range Cell Migration

Range cell migration occurs during the synthetic aperture integration due to the radial range change between the radar and a ground scatterer. This migration consists of a linear component with time, $\dot{r}t$, and a quadrature ratio component, $1/2\ \ddot{r}t^2$. The range cell migration function is analagous to the phase focusing function and can be derived in the same manner. The time, t, is generally referenced to the point of closest approach, r_m, for a non-rotating earth. A rotating earth component of velocity toward the radar will actually make the point of closest approach occur slightly later in time. An orbital altitude rate associated with eccentricity will also affect the time of closest approach. The radar beam pointing departure from broadside will shift the time of signal return and thus introduce an $\dot{r}$ term equal to $V_s \cos\theta_V$, where θ_V is the angle between the velocity vector and the ground scatterer.

Table 2 indicates that for ERSAR there are 86 slant range bins of range walk to be compensated for across the four looks. Assuming an incidence angle of 60^0, a ground bin of 15 meters results in a slant range bin of 15 sin 60^0 =12.99 m. Thus, the range walk is 86 x 12.99 = 1117 m or 7.45 μsec. The range curvature is given as 7 slant range bins = 90.9 m. The maximum range walk across the farthest look = 1117/4 +

3/4 x 90.9 = 347.4 m or 26.7 slant range bins (slightly less than the 28 specified). The 4-look time is 4 x 0.686 = 2.744 sec. Thus, the maximum range rate is assumed to be 407.1 m/sec. The corresponding acceleration is 96.6 m/sec^2. The coherent integration of each look must take place along the curved range cells to achieve the desired resolution. The range cell migration infers a two-dimensional matched filter requirement for which the processing algorithm alternatives in Section 2.2 have been developed.

2.1.3.4. Clutter Lock Requirements

The approximate satellite position, velocity and altitude information is provided to the SAR processor. The SAR antenna pointing (roll, pitch and yaw) will be controlled within $\pm 1^{o}$. Uncertainities in the data could easily cause the signal to depart in center frequency by more than the per look bandwidth. For example a yaw angle of 1^{o} translates to a cross-cone angle of the slant range vector of 0.83^{o} from broadside. This results in a shift in doppler of 950 Hz for the L-band 300 km ERSAR satellite. At X-band the same yaw shifts the doppler by 7153 Hz. Earth velocity for a satellite polar orbit has a radial component at the equator of V_{EQ} sin (incidence angle) = 463 x sin 60^{o} = 401 m/sec, causing a doppler shift of 3413 Hz at L-band. For accurate ERSAR mapping such as 10 m absolute location (ΔX), the doppler shift due to the earth must be known to,

$$\Delta f = \frac{2V_s \Delta X}{R_s \lambda} = \frac{2 \times 7730 \times 10}{564.4 \times .235} = 1.17 \text{ Hz or } 0.034\%$$

Clutter locking should be maintained to within 100 Hz to keep the per look tuning within 20% of the look bandwidth (4% of the PRF). Errors greater than 50% of the look bandwidth will degrade the azimuth sidelobes due to azimuth spectral aliasing. Clutter locking errors due to satellite yawing do not cause a mapping error since the doppler used to tune the processing bandwidth to the signal spectrum is employed in forming the map location.

2.1.3.5 Automatic Focusing Requirement

The doppler chirp rate f_d of a target viewed by the SAR beam must be

matched by the azimuth correlation process to achieve the azimuth beam compression and desired target resolution. It is equal to $f_d = 2\ddot{r}/\lambda$. The radial acceleration $\ddot{r}$ is a function of the satellite velocity V_s, the slant range R_s, the ratio of the target distance y from the orbit axis to the orbit radius, $R_E + h_E$, for near broadside operation for a non-rotating earth. The relation for a circular orbit is

$$\ddot{r} = \frac{V_s^2 y}{R_s (R_E + h_E)} = \frac{V_s^2 R_E \cos \alpha}{R_s (R_E + h_E)}$$

where α is the earth angle between the satellite and the earth target and h_E is the orbit height; R_E is the earth radius. The nominal value of $\ddot{r} = 100.84$ m/sec^2 at the midpoint of the ERSAR swath ($V_s = 7.73$ km/sec, $R_E = 6373.95$ km, $h_E = 300$ km, $R_s = 564.4$ km, $\alpha = 4.2^o$). Over the 40km swath, $\ddot{r}$ varies from 104.03 m/sec^2 at the near edge to 97.8 m/sec^2 at the far edge. The variation $\Delta\ddot{r} = 6.23$ m/sec^2 occurs across the 40 km swath, resulting in a change in doppler rate

$$\Delta\dot{f}_d = \frac{2\Delta\ddot{r}}{\lambda} = \frac{2 \times 6.23}{.235} = 53 \text{ Hz/sec}$$

The depth of focus, expressed in terms of allowable change in doppler rate, can be determined assuming a quadratic phase error allowance of $\pm \pi/2$ at both ends of the synthetic aperture relative to the center of the aperture. The phase error

$$\Delta\phi = \pm \frac{\pi}{2} = 2\pi \int_0^{T/2} \int_0^{T/2} \Delta\dot{f}_d \, dt^2 = \frac{\pi}{4} \Delta\dot{f}_d \, T_L^2$$

The term $\Delta\dot{f}_d = \pm 2/T_L^2$. Thus, the depth of focus, DOF = $(4/T_L^2)$, where T_L is the time to form the synthetic aperture for each look. For an azimuth compression ratio of 350 and a bandwidth of 500 Hz, $T_L = 0.7$ seconds. Thus, the depth of focus = $4/.7^2) = 8$ Hz/sec. From the point of view of a single look, the reference function does not have to change more often than 40 km (8 Hz/sec $\div$ 53 Hz/sec) = 6.04 km

swath. However, the reference function will have to change more often in a cross-track direction to permit the registration of the four looks within 0.2 azimuth cell.

The presence of a quadratic phase error corresponds to a mismatch of the reference function chirp rate with the target chirp rate. This mismatch has the effect of a doppler error between the four looks, Δf, producing a misregistration of the images, Δx.

$$\Delta f = \frac{2V_s \Delta x}{\lambda R_s}$$

For $x = 0.2$, $\delta_{AZ} = 0.2 \times 15 = 3$ meters, $\Delta f = \dfrac{2 \times 7730 \times 3}{.235 \times 564.4 \text{ km}} = 0.35$ Hz

The time interval between the four looks = $3T_L = 2.1$ sec. The allowable change in doppler rate cross-range = (0.35/2.1) =0.17 Hz/sec, Thus, for registration, a new reference function should be applied more often than 40 km x (0.17 Hz/sec ÷ 53 Hz/sec) = 126 meters of swath, corresponding to 8.4 resolution elements. The specification calls for a new reference function cross-track every 8 samples, which is reasonable. Fortunately, the chirp rate variation across the swath is deterministic from the geometrical parameter variations and may be computed from the one or two measured chirp rates refined by auto-focusing techniques.

2.2 PROCESSING ALGORITHMS

This section describes the candidate SAR processing algorithm considered for the ADSP and gives pertinent implementation data on them. Performance levels are treated separately in the following section (Section 2.3).

2.2.1 Basic SAR Processing Functions

The formation of images from synthetic radar arrays follows the same principles found with optical lenses or real radar antennas. Images can be focused in the far-field when the energy reaches the antenna as a virtual plane wave relative to the antenna diameter or in the near field. The far-field SAR is generally termed an unfocused SAR and, being relatively simple to process, is not treated in this study. Processing in a focused SAR is essentially a two dimensional matched filter process as indicated in Figure 6.

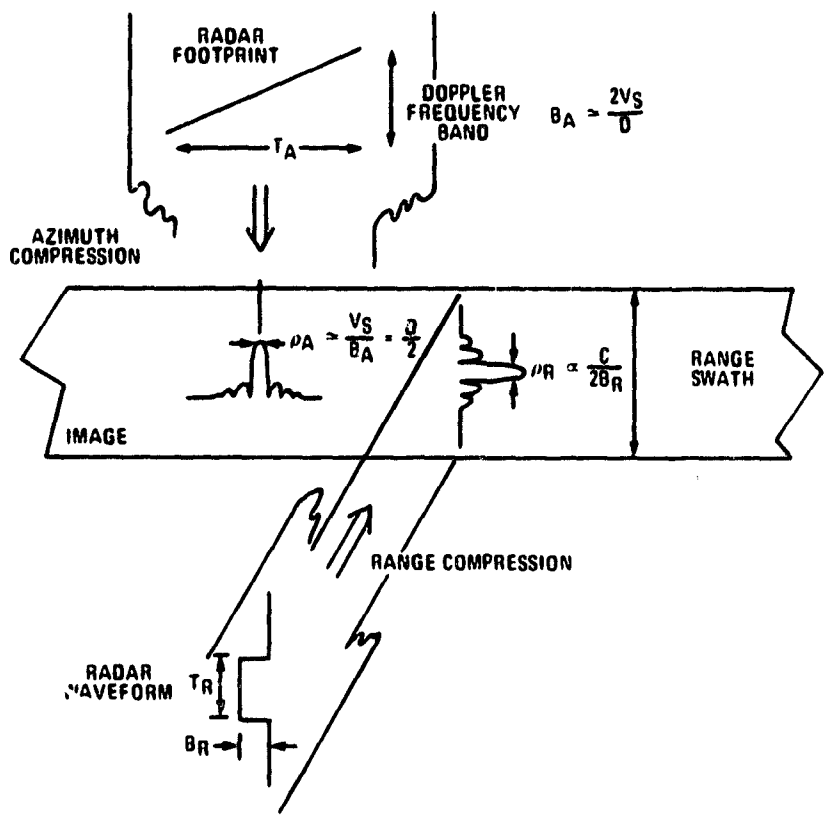


Figure 6
Focused SAR Image Processing Functions

The range, or cross-track, dimension may be processed by conventional radar pulse compression techniques, the method used depending on the type of radar waveform employed. The azimuth, or along track, dimension can be considered in the same functional sense as the range dimension. Here, the radar footprint of the real antenna with its doppler frequency response forms a quadratic phase (or linear FM) function across its width. As the real antenna moves with its scanning platform the radar footprint "waveform" moves along the image plane in azimuth in the same manner that the uncompressed radar pulse moves across the image plane in range. Thus a filter matched to the radar footprint "waveform" operating in the azimuth dimension will compress the radar footprint to the resolution defined by its time and bandwidth parameters. The limiting azimuth resolution for a focused SAR is one half the azimuth dimension of the real antenna.

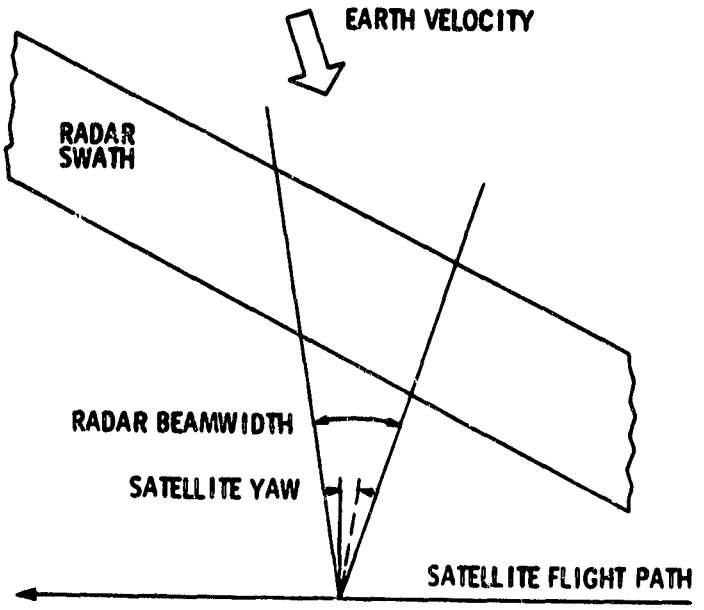


Figure 7

Satellite Radar Swath

Two other functional requirements are important in SAR processing, range migration and multiple looks. As a SAR satellite platform

moves along its track, the earth velocity causes the radar return to move linearly with the motion of the spacecraft (S/C) (See Figure 7). In addition, the range of a fixed point on the earth will vary in a quadratic manner as the S/C radar footprint moves past.

Radar returns from objects vary in amplitude as a function of the aspect angle of the radar. This tends to give radar mapping return a speckled appearance. The effect can be reduced by non-coherently summing coherently processed segments of the radar footprint "waveform". This multi-look processing reduces the final image resolution in proportion to the number of looks processed. The bandwidth of the processing required for each look is also proportionally reduced.

The combined effect of range migration, linear and quadratic range walk and multiple look processing is illustrated in Figure 8.

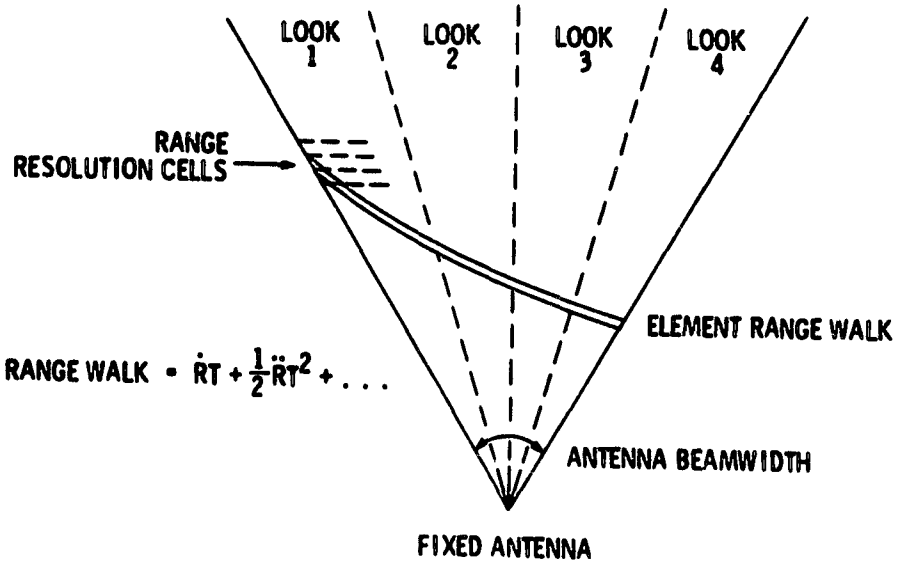


Figure 8
Azimuth Resolution Element Motion
Relative Fixed Antenna

2.2.2 Time Domain Processing

2.2.2.1 General Time Domain Processing Consideration

The general form of a time domain processor is shown in Figure 9. It essentially implements the SAR processing equations with a time domain analog of the functions. After range correlation, which consists of a tapped delay line finite impulse response filter in a time domain implementation, the signal is filtered (presummed) into bands corresponding to the looks to be processed. This serves to greatly reduce the computation requirements of the azimuth correlators. A simple example will illustrate the effect. Consider an azimuth correlator with 1000 azimuth samples over the total beamwidth with an azimuth doppler bandwidth (and sample rate) of 1000 Hz. The number of complex multiples per second to perform a time domain correlation is then 10^6 per range cell. If the signal is processed as four looks, the bandwidth per look becomes 1000/4 and the number of azimuth samples covering each look is 1000/16 giving a total computation rate for four parallel look filters of $(\frac{1000}{4} \times \frac{1000}{16} \times 4) = 10^6/16$ per range cell. Thus, the computation rate is inversely proportional to the square of the number of looks processed. The net complex computation rate for a time domain correlator is $\alpha T_A B_A / N_L^2$

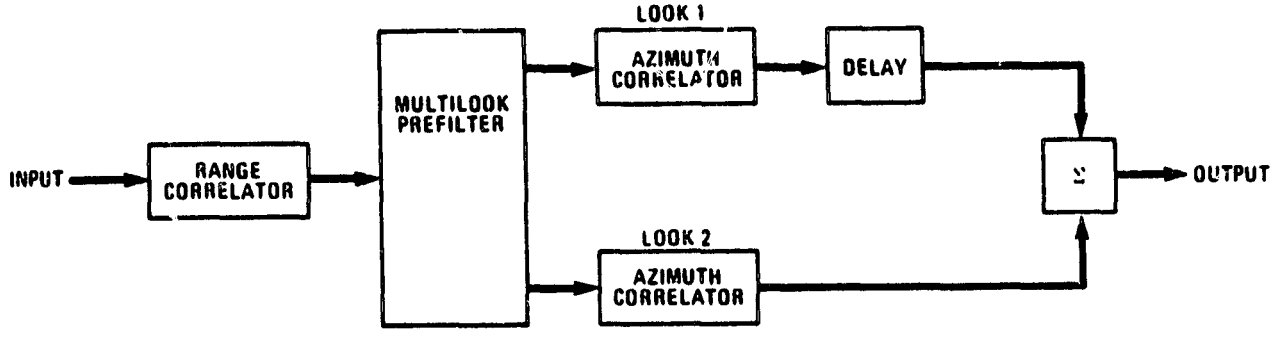


Figure 9

Multi-Look Time Domain Processor Functions

where; T_A is the synthetic aperture integration time, B_A is the doppler bandwidth, N_L is the number of looks, and α is the over-sampling factor.

After prefiltering and azimuth correlation, the multiple looks are summed. A time delay must be inserted in the look filter prior to summation to compensate for the time offset of the look energy from particular looks.

Two general types of architecture have been identified for time domain processor and have been evaluated in Reference 1. They have been designated Type A and Type B and we will continue to use those designations for clarity. Type A (Figure 10) is a classical tapped delay line matched filter in which the intertap delay lines connected serially contain the entire SAR range swath.

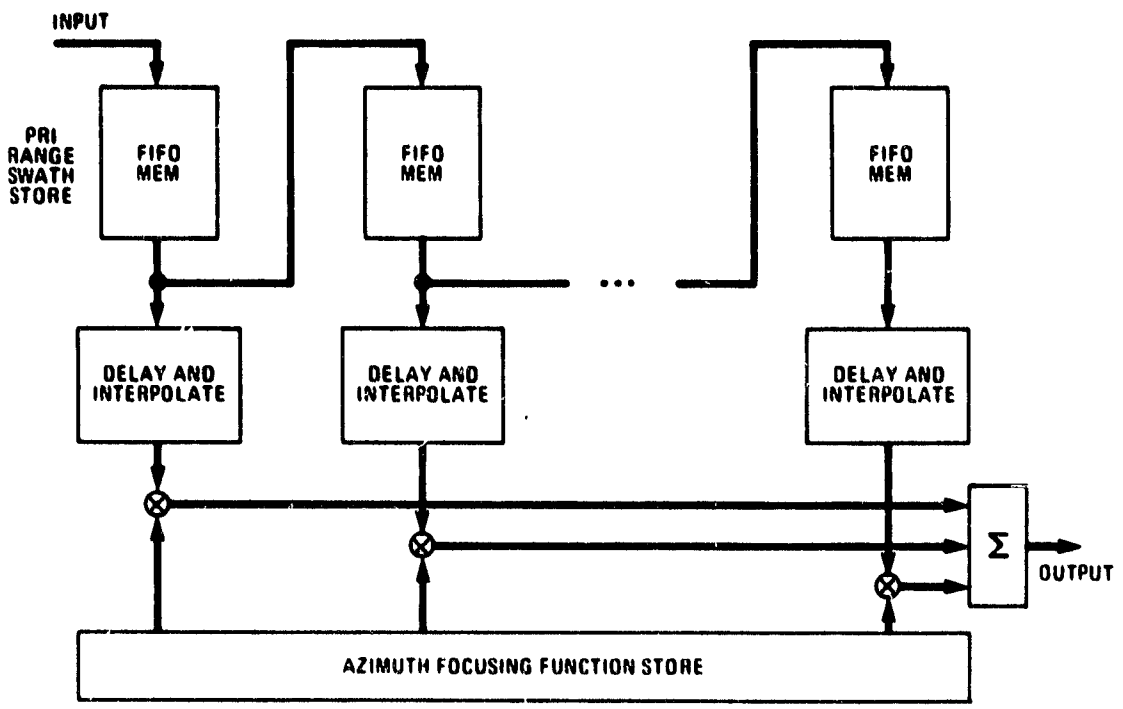


Figure 10

Serial Time Domain Azimuth Processor

Type B (Figure 11) performs azimuth filtering by the construction of parallel azimuth beam correlators, each accumulating the processed beam over the entire range swath. The general simplicity of form for these processors plus their minimization of memory storage makes them candidates for the ADSP. We will develop sufficient detail on these processors to permit evaluation on an equal hardware technology basis with the frequency domain methods.

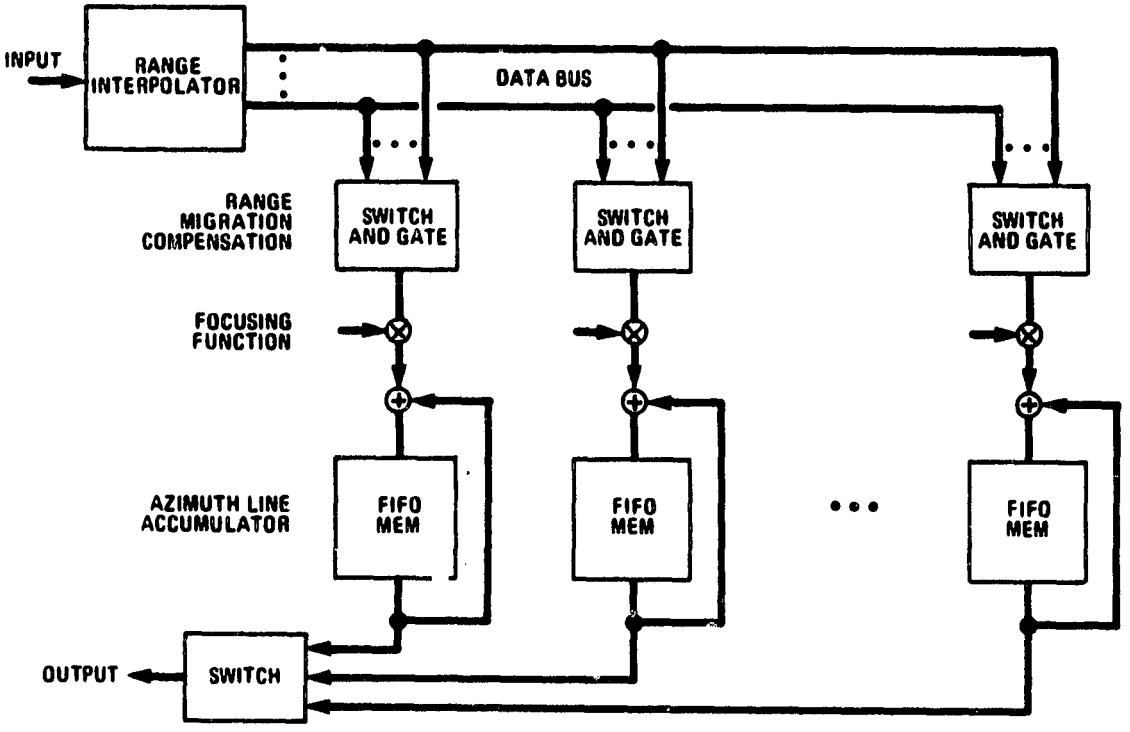


Figure 11
Parallel Time Domain Azimuth Processor (Type B)

2.2.2.2 Serial Time Domain Processor (Type A)

Figure 12 is a more detailed sketch of a Type A or serial processor. The processing is segmented into processing units labeled as correlator #1, correlator #2 ... up to correlator N. Each processing unit has a FIFO (first-in, first-out) memory large enough to hold a single pulse radar sweep. Since the range processing is completed,

the FIFO memory covers the range swath plus any uncompensated range migration. The FIFO memory can be implemented with shift registers; but random access memories which have more capacity per chip, consume less power and are less expensive are a preferred choice. In addition, the storage length of a FIFO random access memory is easily changed by address control. The address control mechanism consists of a counter, a reset control word and a comparator. On each count new data is read into the memory and the data stored is read out. When the count reaches the value of the reset control word, the comparator will output a signal which resets the counter to zero. Thus, the delay (FIFO) capacity is easily changed. The memory read-modify-write cycle time must be less than the clock period. If it is not, multiple memory units can be multiplexed to meet the speed.

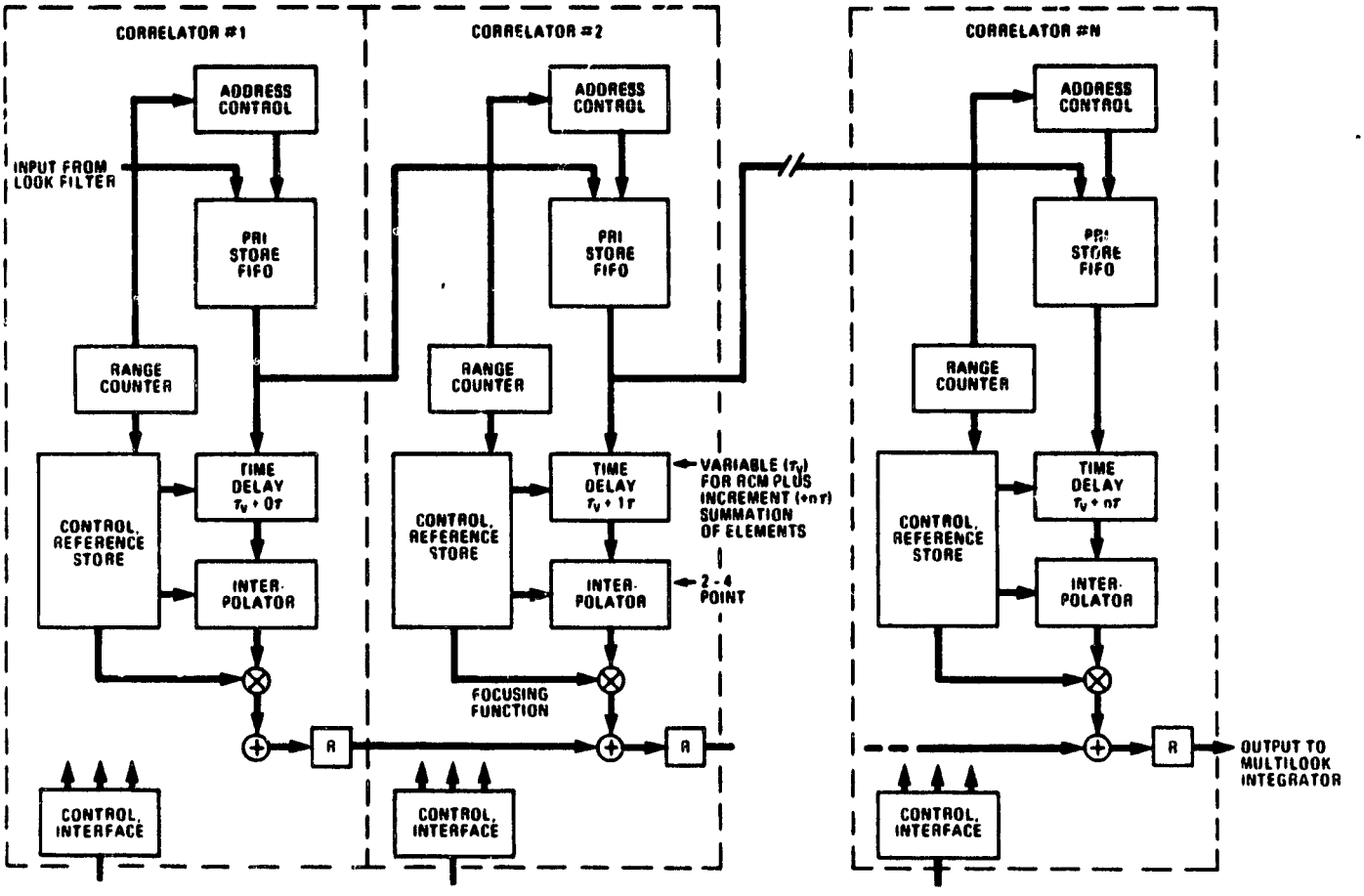


Figure 12
Detail of Type A Processor

Range migration compensation is accomplished within each correlator by a combination of variable clock interval (range cell) shifts (τ_v) plus interpolation. Simulations indicate that a two point interpolator will give adequate performance, but for the highest quality imagery it may be desirable to use a three or four point interpolator. The focusing function storage in each correlator unit is related to the required increments of range migration compensation. Each significant variation in focusing function, nominally every 8 range cells for the baseline SAR mission, has an associated range migration compensation. The total reference storage in each correlator unit is at least 4000/8 = 500 words of at least 34 bits. These bits would be assigned as 22 bits for focusing function, 7 bits for incremental range migration, and 5 bits for the interpolator.

In addition to the variable time delay in each range migration compensation memory, a fixed incremental delay is inserted to aid in conveniently obtaining an output summation. In effect, each correlator segment is operated offset in time by one range cell. Then, as the partial sums are passed from one unit to the next through register delays they are automatically added with the appropriate delays.

In the baseline SAR system requirements, the four look prefiltering will give a clock rate of 11/4 = 2.75 MHz for each azimuth correlator. Obviously, the computational hardware in each correlator unit can be operated at a higher rate and this is reflected in the three circuit tabulations given in Table 3 for the Type A system. A typical configuration for a 4X multiplexed correlator is shown in Figure 13. Note also in Table 3 that the number of interconnections is modest for the Type A correlator modules.

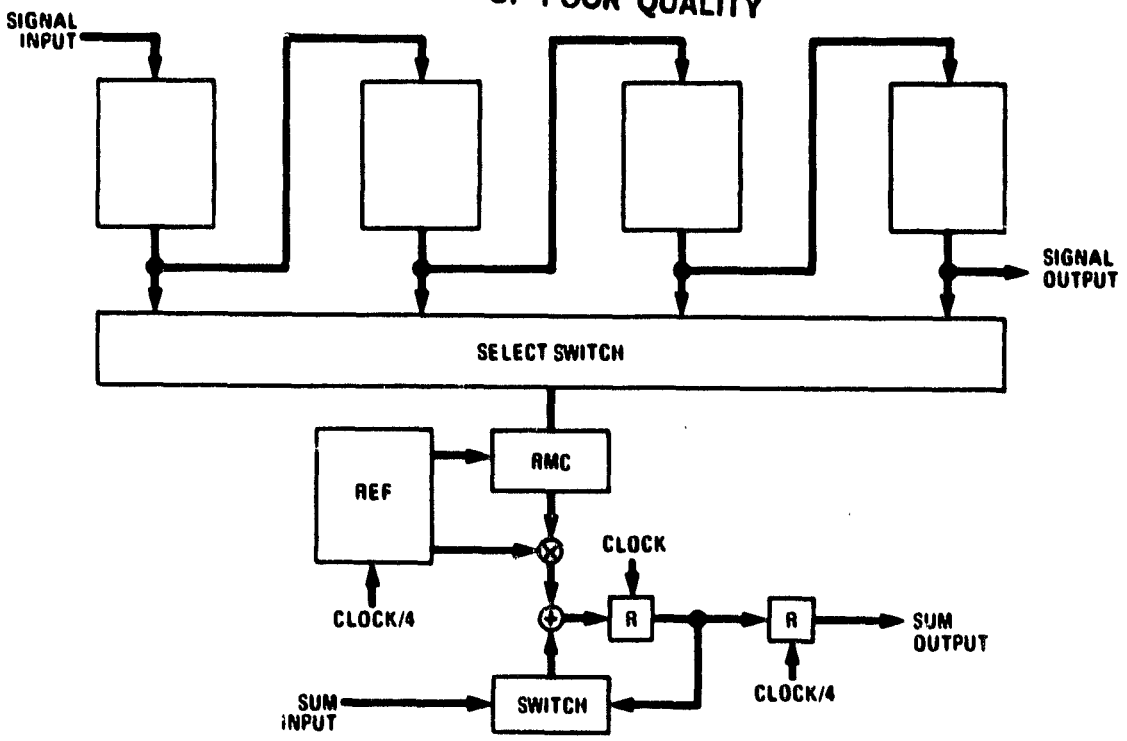


Figure 13

Multiplexed (4X) Type A Correlator Channels

Table 3

Type-A Circuits -- Interconnections

FUNCTION	CIRCUIT TYPES	CIRCUITS/MODULE 1X 2.75 MHz	2X 5.5 MHz	4X 11 MHz
PRI STORE (8K X 22)	4K X 4 RAMS	12	24	48
PRI MEMORY CONTROL	MISCELLANEOUS	10	10	10
RANGE COUNTER	COUNTERS AND REGISTERS	4	4	4
RMC AND CONTROL	MISCELLANEOUS	60	70	90
FOCUS MULT	4 MULT, 2 ADDS	15	17	19
ADDER	ADDERS AND REGISTERS	16	24	26
CONTROL INTERFACE	MISCELLANEOUS	10	12	14
	TOTALS	127	161	211

I/O		
	INPUT	22
	OUTPUT DELAYED	22
	OUTPUT SUM	30
	CONTROL IN-OUT	32
	TOTAL	106

2.2.2.3 Parallel Time Domain Processor (Type B)

A detailed diagram of the Type B time domain processor is shown in Figure 14. Because of the nature of the architecture, a single common interpolator unit can be used for all of the correlators. However, this step creates a large I/O pin requirement and the accuracy that can be used is therefore limited. The example shown has eight, two point interpolators feeding the input bus. Range migration compensation is accomplished for each correlator by selecting the appropriate interpolated point and gating the desired range cell. The entire interpolation control sequence is generated and/or stored in the Common Control Unit. As the first correlator starts its integration process, the required RCM controls as a function of range for the first radar PRI are applied. This control sequence is then passed to the second correlator and the RCM control sequence for the second PRI is applied to the first correlator. This process is repeated until the last PRI to be integrated by the first correlator is received. A similar process is used for the focusing function store. The reference words pass serially through the string of correlators. When a correlator unit completes its integration process, its output is selected and it starts the integration process again.

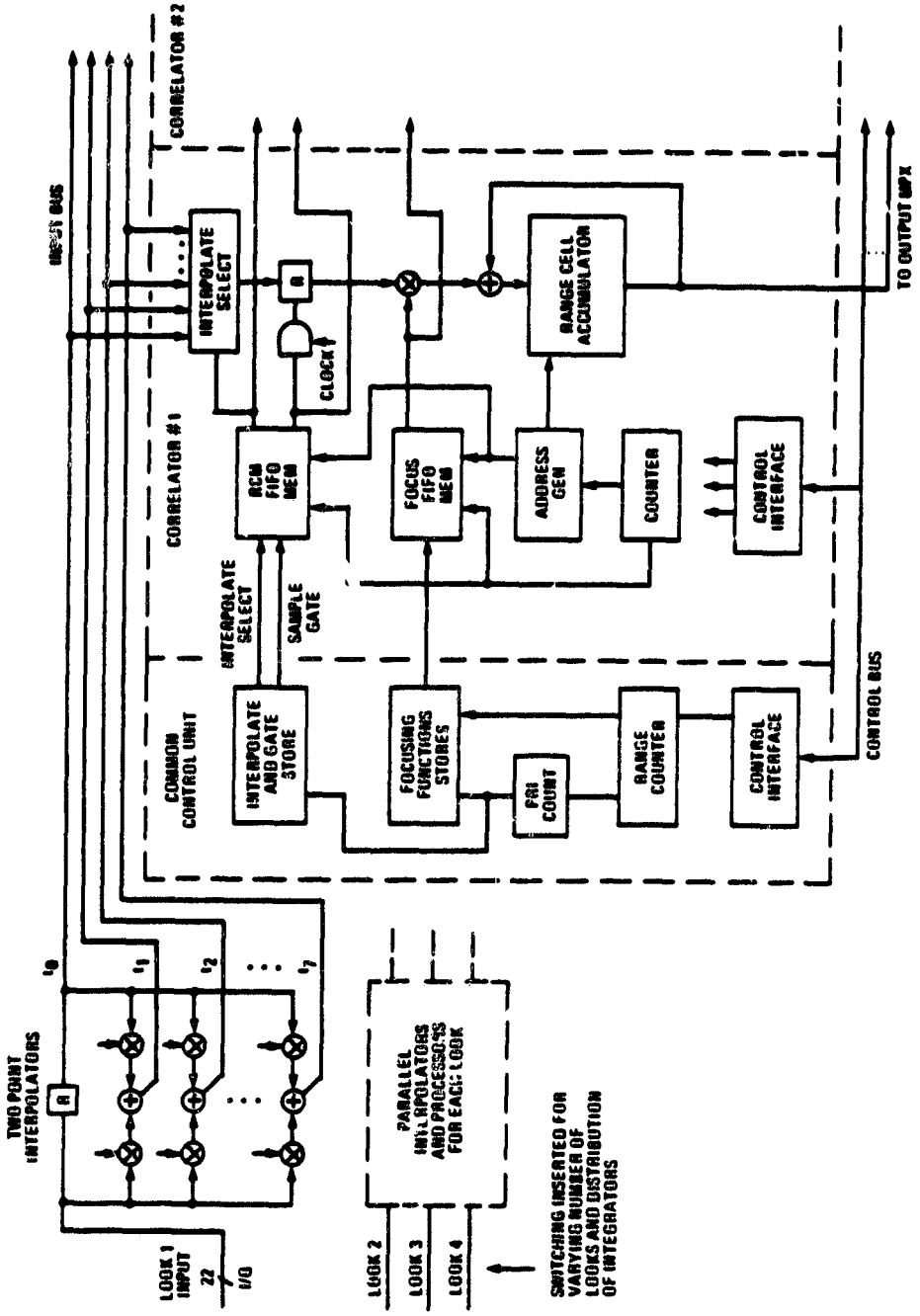


Figure 14

Detail of Type B Processor

Table 4 lists the circuit counts for various multiplexing alternatives for the Type B architecture.

Table 4

Type-B Circuits -- Interconnections

FUNCTION	CIRCUIT TYPES	CIRCUITS/MODULE 1X 2.75 MHz	2X 5.5 MHz	4X 11 MHz
SELECT	8:1 SW	22	22	22
RCM MEM AND CONT	MISCELLANEOUS	14	28	56
FUNCTION STORE	MEMORY	6	12	24
ADDRESS GEN	MISCELLANEOUS	10	18	26
RANGE CELL ACCUM	4K X 4 RAMS	16	32	64
COMPLEX MULT	4 MULT	15	17	19
ADDER	ADDERS	16	18	20
CONTROL INTERFACE	MISCELLANEOUS	10	10	10
	TOTAL CIRCUITS	109	161	237

I/O		
	INPUT	176
	FOCUS IN-OUT	48
	INTERPOLATE IN-OUT	8
	OUTPUT	30
	CONTROL	16
	TOTAL	278

2.2.2.4 Type A, B Comparison

A qualitative comparison of the Type A and Type B time domain processors is shown in Table 5. There is not a significant difference in the estimated cost or performance of the two approaches. Type B offers more flexibility for variation of SAR parameters. However, this is only true if the number of looks are held constant and the compression factor is less than the established size. For both systems, if the number of looks are reduced it takes significantly more hardware correlator modules to achieve real time processing rates and the system must be reconfigured. Our preference is with the Type B system because of its greater programmability and somewhat less fault sensitivity.

Table 5

Type-A, B Comparisons

CHARACTERISTIC	TYPE A	TYPE B
COST, SIZE	HIGH (> 2X FFT CONVOLVER)	HIGH (> 2X FFT CONVOLVER)
MODULARITY		
• METHOD	• PARALLEL CORRELATORS	PARALLEL CORRELATORS
• I/O	• PROBLEM IF COMMON INTERPOLATOR	O. K.
• FAULT SENSITIVITY	• VERY HIGH DUE TO SERIAL DELAYS AND SERIAL SUM	HIGH IN SERIAL CONTROLS
• CONTROL	• INDEPENDENT MODULE CONTROLS	SERIAL CONTROLS
PROGRAMMABILITY		
• AZIMUTH COMPRESSION	FAIR -- IF LESS THAN FULL CAPACITY, POOR -- IF GREATER	FAIR TO GOOD -- IF LESS THAN FULL CAPACITY, FAIR -- IF GREATER
• RANGE SWATH	GOOD -- MEMORY CONTROL	GOOD -- MEMORY CONTROL
• NUMBER OF LOOKS	POOR -- NEED EXTENSIVE SWITCHING, SYSTEM GROWS EXPONENTIALLY WITH FEWER LOOKS	FAIR -- INPUT MULTI-BUS SWITCHING, SYSTEM GROWS EXPONENTIALLY WITH FEWER LOOKS
• BURST MODES	IN ACCORDANCE WITH AZIMUTH COMPRESSION VARIATIONS	
• RELIABILITY	POOR DUE TO SERIAL DELAYS AND SUM	FAIR DUE TO SERIAL CONTROL PIPELINES
• IMPLEMENTATION PROBLEMS	CONTROL BUS	LARGE INPUT BUS I/O REQUIREMENTS IF COMMON INTERPOLATOR

2.2.2.5 Prefilter and Multilook Integrator

The general multilook (for four looks) prefilter situation is depicted for the input signal to the prefilter shown in Figure 15. The signal spectrum, which has been sampled at the sampling frequency f_s, is offset from the center by an amount f_{off}. The objective is to filter the signal into the four individual looks and decimate the sampling frequency by a factor of four. The clutter lock circuit will detect the offset frequency and this will be applied to the overall spectrum (point (a)) to center it about zero (point (b)). Each of the four individual look offset frequencies f_1, f_2, f_3, and f_4 are then applied and the result is low pass filtered to isolate the individual look bands. Since the bandwidth for each channel has been reduced by a factor of four, the sampling rate can also be reduced (decimated).

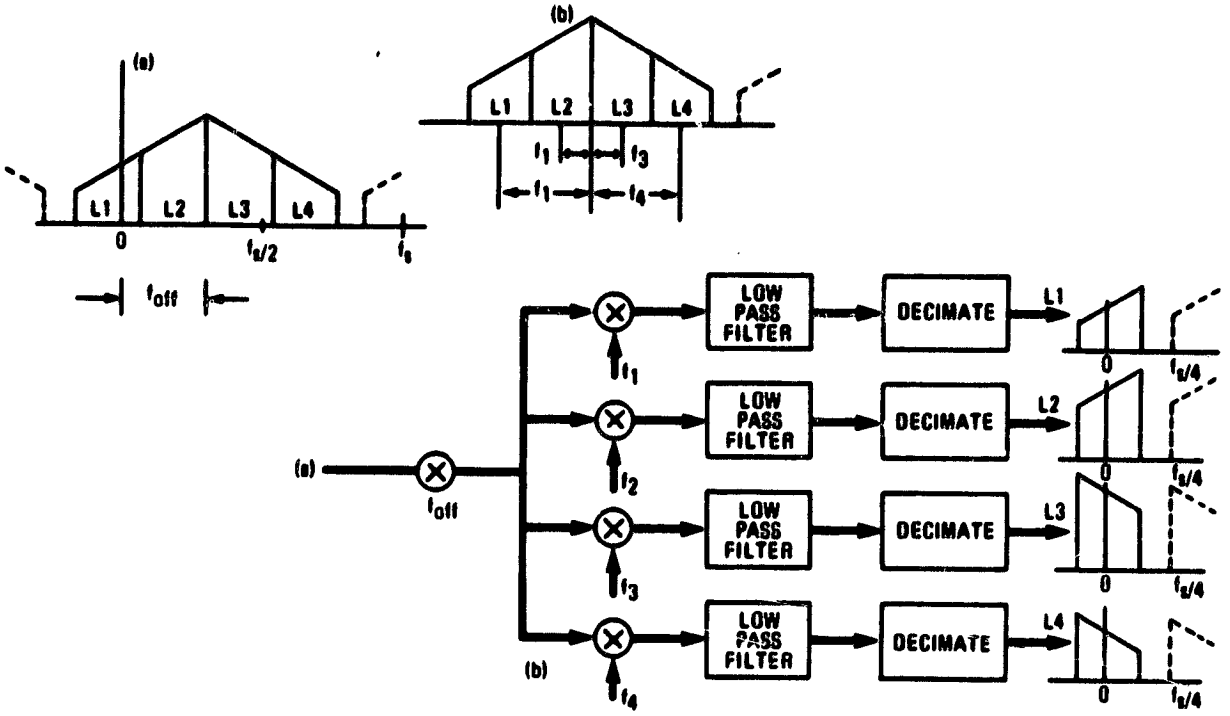


Figure 15

Multilook Filtering Functions

An implementation of the filtering and decimation function is given in Figure 16. A 32nd order low pass filter is used to achieve the desired passband and ripple characteristics. This is obtained from the approximation given in reference 2,

$$\Delta F \approx \frac{\ln 2 - \ln \delta_2}{\pi (M-1)}$$

where ΔF is the passband to stopband transition width, M is order of the filter, and δ_2 is the passband ripple,

if we set $\Delta F = .05$, and $\delta_2 = .01$ we obtain $N \approx 32$. In the implementation of Figure 16, the decimation is accomplished by the serial to parallel shift register tap registers operating at a reduced (4 to 1) clock rate. The decimation is exploited by operating the filter coefficient multipliers at the input clock rate and accumulating four products prior to the formation of the adder tree.

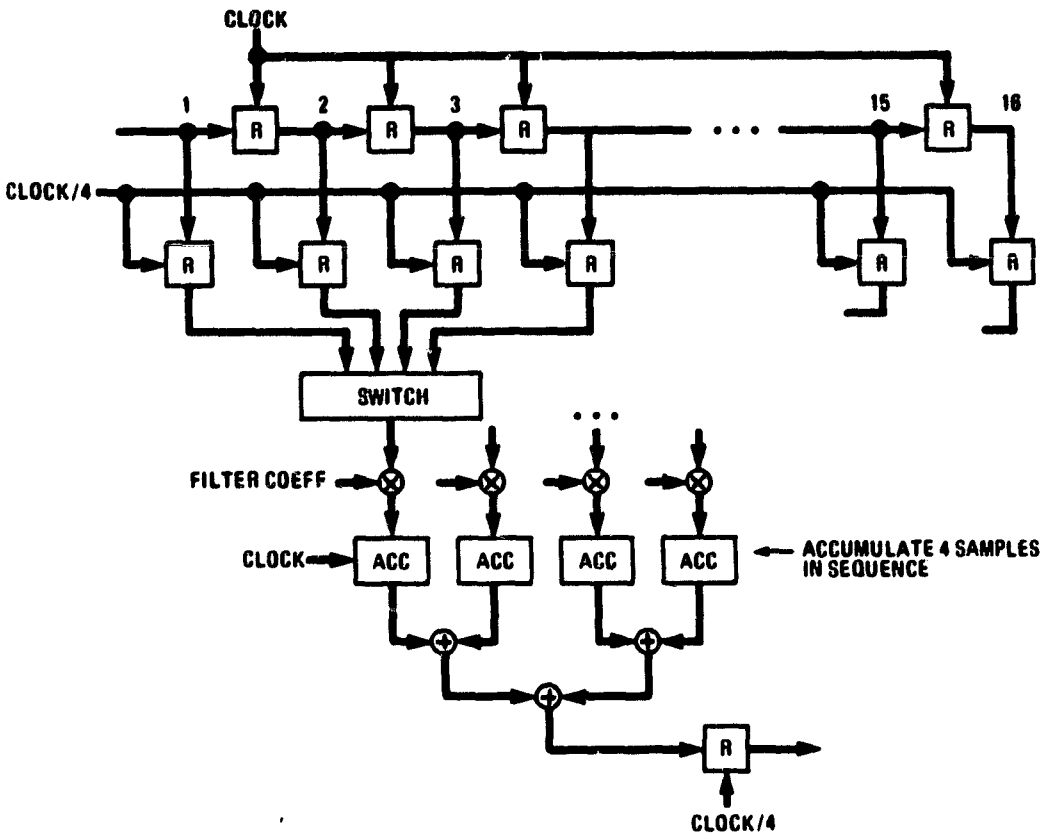


Figure 16

Low Pass Filter
(32nd Order) Decimate by 4

An implementation of the multilook integrator is shown in Figure 17a. The method shown with four individual look integrators is preferred

over the delay line approach since less memory is used. If X is the product of range cells times azimuth compression samples per look, the storage in Figure 17a is 4X. A delay line approach (indicated in inset) would require 3X + 2X + X = 6X.

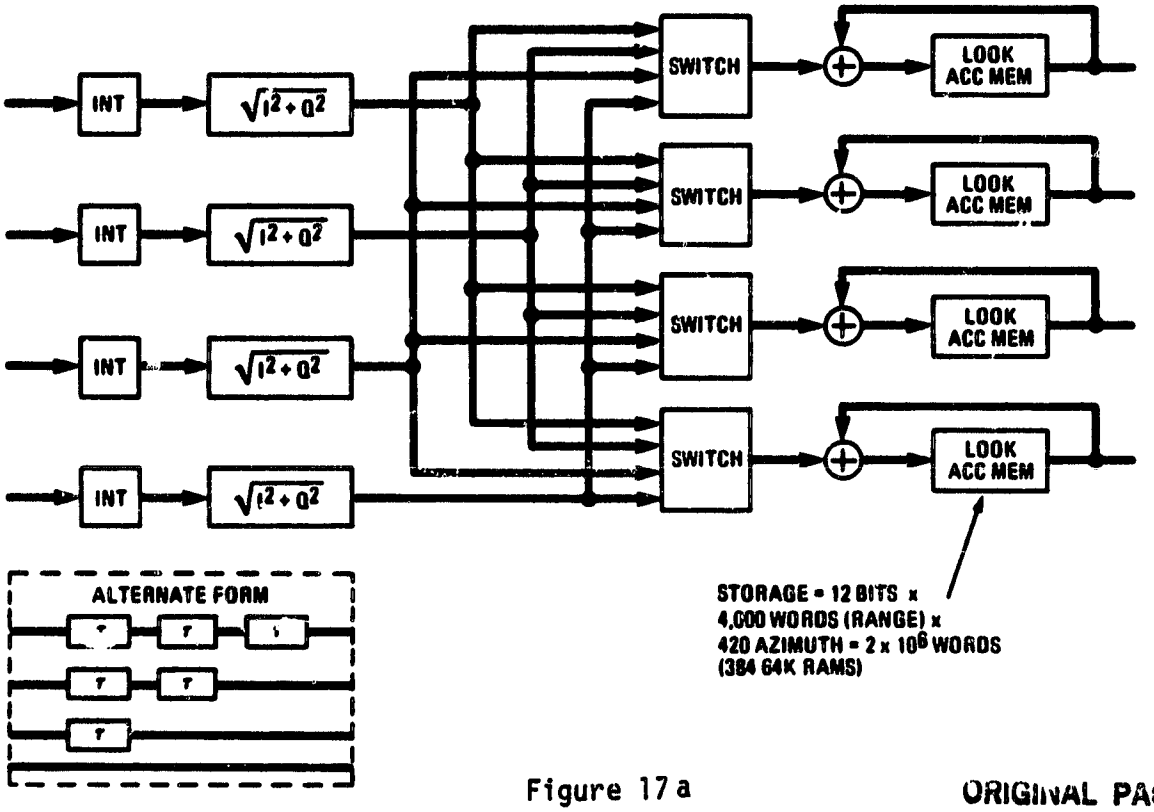


Figure 17a

Multilook Integrator

2.2.2.6 Time Domain Hardware Summary

Table 6 lists a hardware summary of the time domain processing options. The totals assume 11 MHz clock rates in the correlator modules. Although this speed is achievable, it is somewhat high for TTL application. We feel it is practical in this instance, since it is an on-module clock rate. Most of the module interface rates are at one fourth this rate or 2.75 MHz. Nevertheless, the total number of modules is 3 to 4 times as high as the number expected for an FFT convolver implementation.

Table 6

Time Domain Module Summaries

FUNCTION	NUMBER REQUIRED TYPE A	TYPE B	REMARKS
RANGE CORRELATOR	130	130	11MHz CLOCK, ~200 CIRCUITS EACH
RANGE CONTROL	1	1	
PREFILTER CHANNEL MODULATOR AND CONTROL	4	4	11MHz IN, 2.75MHz OUT
INTERPOLATOR	-	4	
AZIMUTH CORRELATOR	420	420	11.0MHz CLOCK, 4 CHANNELS/MODULE
CONTROLLER	1	1	
INTERPOLATOR AND MAGNITUDE	1	1	4 MODULES/INTEGRATOR
MULTILOOK ACCUMULATOR	16	16	500K WORDS/MODULE
CONTROLLER	1	1	
SYSTEM CONTROLLER	1	1	
INTERFACE/TEST	1	1	
	577	581	

2.2.3 FFT Convolver

Azimuth processing, using an FFT convolver, has been divided into two approaches: Case 1, discussed here, where the range and azimuth correlation are operated separately and Case 2, discussed in Section 2.2.5, where a two dimensional convolution is performed either including, or not including, the range compression function. Range compression alone using FFT convolution is discussed separately in Section 2.2.6.

FFT azimuth correlation is accomplished by forming an azimuth matched filter moving along constant range lines of the radar return. The

filter is formed by transforming the data to the frequency domain, multiplying by the complex reference of the azimuth focusing function and performing the inverse transform to convert the signal back to the time domain. The frequency transform filter thus formed performs a circular convolution of the azimuth focusing function with the signal data within the FFT window.

With multiple look processing, separate frequency filters must be provided for each look. This is done as shown in Figure 17 b with separate spectral references for each look and separate inverse FFT's. Because the frequency band coverage of a single look is a fraction of the total, the frequency samples can be decimated (reduced sample rate) prior to inverse FFT processing. In the example of Figure 17b with four looks, the number of inverse FFT points is reduced by a factor of four to coincide with a four to one reduction in bandwidth.

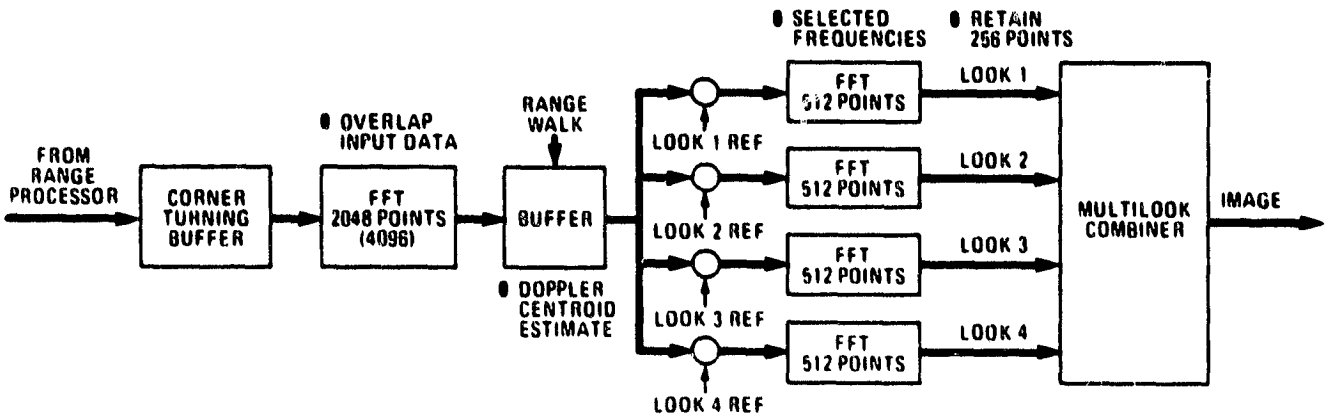


Figure 17 b

FFT Convolution-Azimuth Processing

The ability to perform range migration compensation in the FFT convolver approach is due to the unique relationship of range migration and frequency coefficients in the azimuth processor as illustrated in Figure 18. As indicated in Figure 8, the range migration correction is a function of beam angle and the frequency coefficients are representative of beam angle. Therefore, range migration correction by shifting frequency coefficients in range will correct for all of the targets with energy in that coefficient. We have shown (Section 2.3) that for adequate performance the range migration correction must be refined using interpolated sample points.

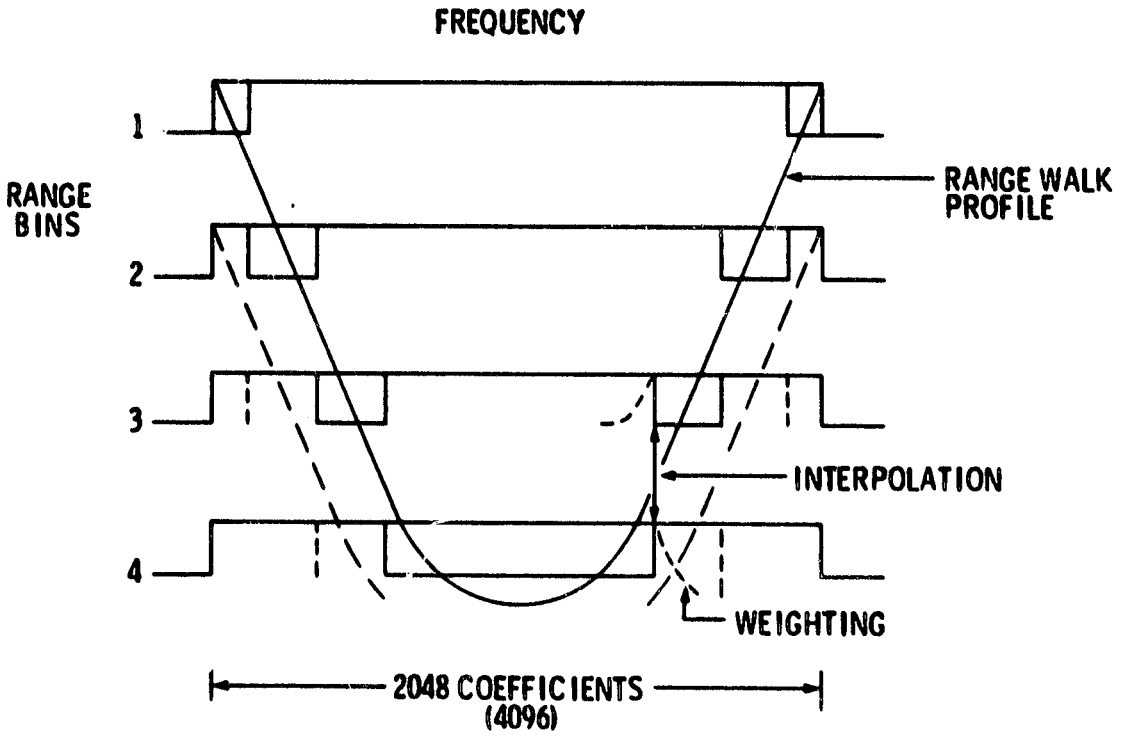


Figure 18

Unique Relationship Between Range Migration and Frequency

We have selected the FFT convolver algorithm for the recommended ADSP design and discussion of its implementation is contained in Section 2.5.

2.2.4 Subarray Processing

Subarray (or step transform) SAR processing is performed by converting segmented received time data to the frequency domain and resolving the individual doppler trajectory of the map elements. Partitioning the data into small groups or subarrays permits range walk and other correction factors to be incorporated. This partitioning also enables a continuous strip map to be processed in an efficient manner by adding only new subarray data to the already accumulated data and dropping the old subarray data.

SAR processing using subarrays consists of the following five steps.

1. Focus data over short subarray time.
2. Store subarray data over large array length.
3. Combine subarray outputs to form large array.
4. Compute new subarray data entering large aperture.
5. Drop old subarray leaving large aperture.

Figure 19 illustrates the step transform algorithm. Since the data is being continuously fed into the step transform processor, the overlapped subarray computation can be performed as a running process. Similarly, the output fine resolution can be computed as soon as a full aperture of subarrays have been computed. This is also done as a running process where new subarray data is added while the old data is dropped.

Factors to be considered in implementing the subarray approach include input spectral aliasing, short aperture sidelobes, and straddling loss. These and other factors have been analyzed and are discussed in Reference 3.

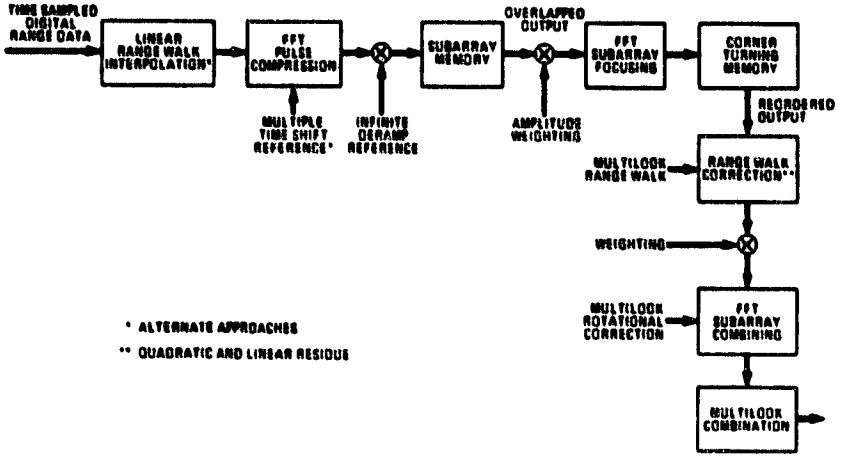


Figure 19

Block Diagram of Subarray Azimuth Processor

The functional diagram of Figure 19 illustrates the subarray approach for an infinite deramping function. The azimuth processing begins with the application of the infinite deramping function following range pulse compression. The reference function applied is illustrated in Figure 20. It is essentially a continuous frequency ramp wrapping around at the sample rate. Total frequency coverage within the sample rate will cover the entire SAR antenna beam response including multiple looks and guard band. A single target will encompass the full response.

After demodulation, the data is collected in a subarray memory which forms overlapping subarray segments which are amplitude weighted and processed in the subarray focusing FFT. Data is then placed in a corner turning memory whose output is first processed for range walk compensation prior to final subarray combining. The subarray combining data is integrated along paths as indicated in Figure 21. Coherent integration occurs across a single look at a single coarse azimuth frequency with multiple look, non-coherent integration following along the same frequency.

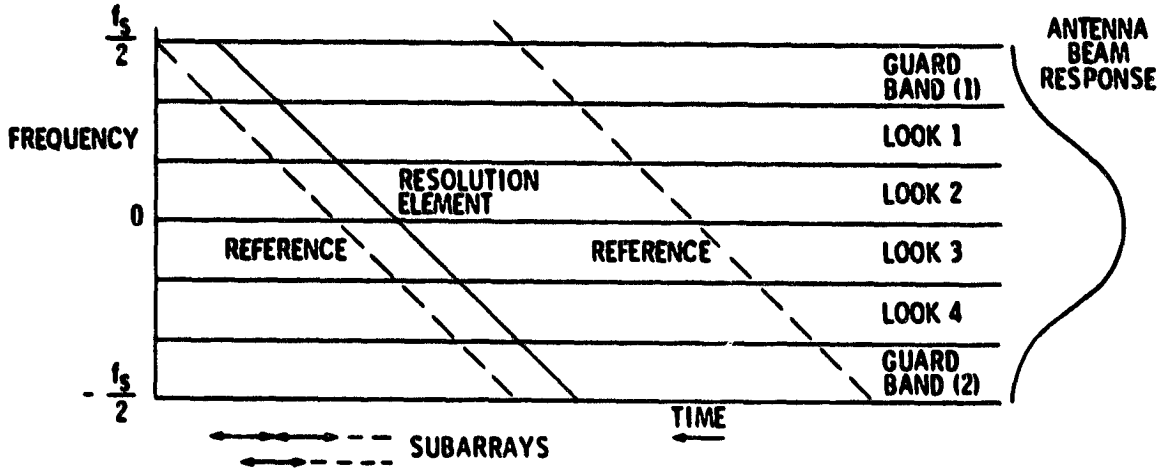


Figure 20

Continuous Deramping and Subarray Processing

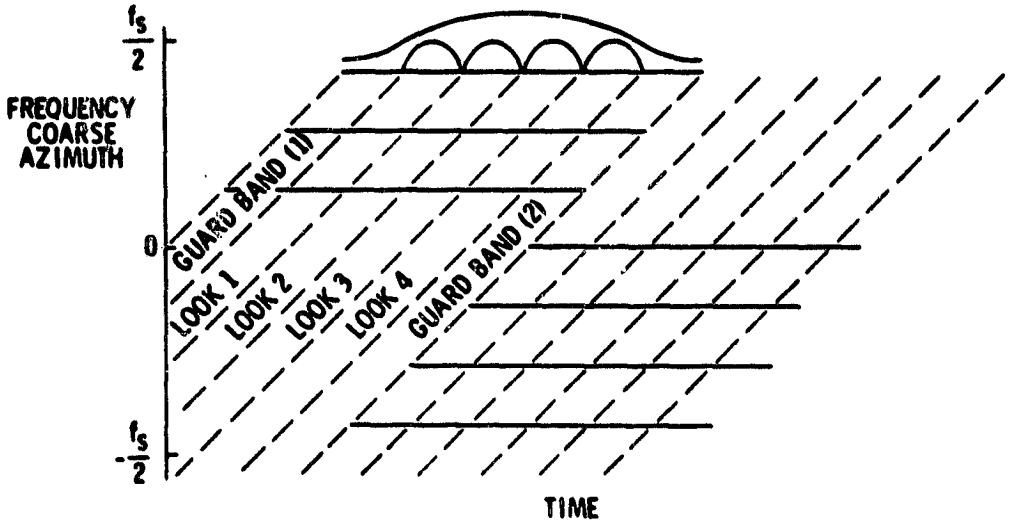


Figure 21

Deramp Coarse Resolution

Bulk storage in the subarray processor can be estimated by reference to Figure 22. The integration must occur along horizontal paths for each look. As new subarray data is received integration intervals as a function of frequency become filled. These are then read out and processed for fine resolution. Since the data can be discarded after integration the minimum storage requirements for each range cell will be contained within the four triangular patches with width covering a look integration time and height corresponding to the look bandwidth. Thus, if there are M subarray coefficients and N subarrays per look the minimum storage is MN/2 or MN if a double buffered implementation is used. For the ERSAR case, the PRF is 2500 and the look integration is 0.7 seconds. The input time-bandwidth product is then 1750. Using a subarray size of 64 points and an overlap factor of 2 to 1, the number of subarrays per look becomes 54. The number of spectral coefficients used is 4/5th of 64 = 52. The minimum memory storage is therefore 52 x 54/2 = 1404 per range cell or about 5.7 x 10^6 words to encompass 4086 range cells. A programmable design of the bulk storage unit in a subarray processor is difficult and it may simplify its control if a double-buffer arrangement can be used. In this case the total memory would be 11.4 x 10^6 words.

The complex computation rate for the baseline can be estimated per look interval per range cell as follows:

Continuous deramp function: .7 x 2500 =	1750
First FFT Wtg: 54 x 64	3456
First FFT: 54 x 32 x 6	10,368
Second FFT Wtg: 52 x 64	3328
Second FFT: 52 x 32x 6	9984
Total per range cell	28,886
Total per 4086 cells	118 x 10^6
Rate per second	168 x 10^6

The rate of 168 x 10^6 assumes equal computational load for the weighting function and the FFT butterfly computations. If the weighting is accomplished by combining it with the FFT process, half of the weighting operation are eliminated and the net computation rate

becomes 148 x 10^6 complex operations per second.

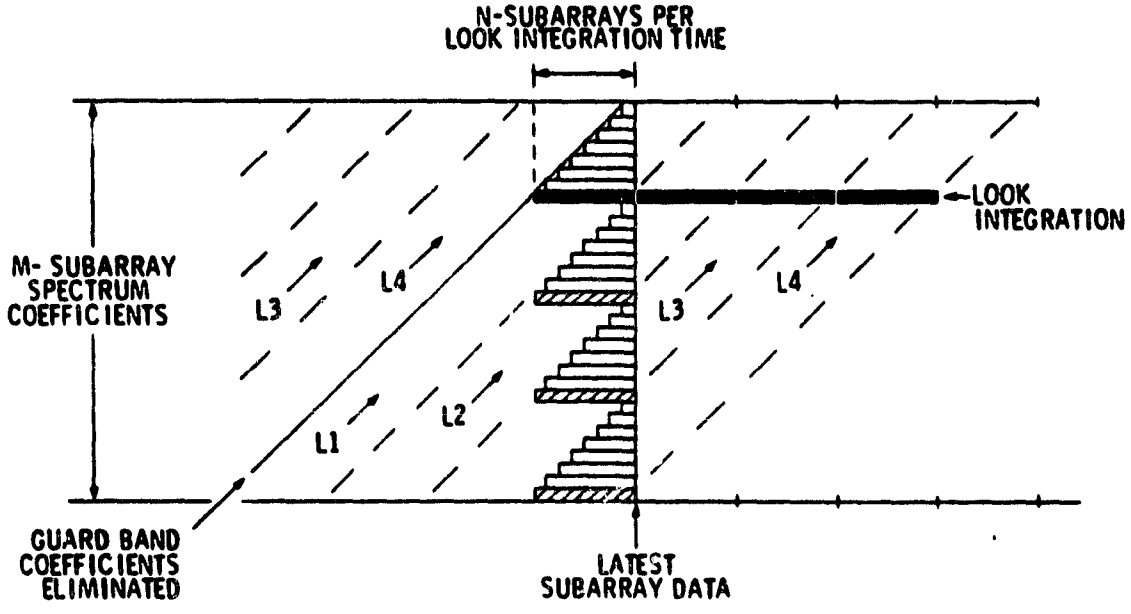


Figure 22

Bulk Storage in Subarray Process

2.2.5 Two-dimensional Convolution

The 2-dimensional convolution approach to SAR azimuth processing has been divided into two parts, one in which the range correlation is included and the second for azimuth compression only and the 2-dimensional process is applied to correct for range migration. The latter approach also includes a hybrid FFT time domain technique.

2.2.5.1 Full Range Azimuth Correlation 2-D Process

Two dimensional convolution as illustrated in Figure 23 is an ideal approach in terms of computational efficicency for a single reference, but the small depth of focus for a single reference spectrum results in reduced efficiency. For example, the azimuth reference function must be updated every 6 km of swath to maintain single look focus. Over a 40 km swath, the reference would have to be adjusted at least 9 times if a 1024 x 4096 2-D array were used. The process also rules out the incorporation of multilook registration in the process since this requires a reference update every 8 range cells.

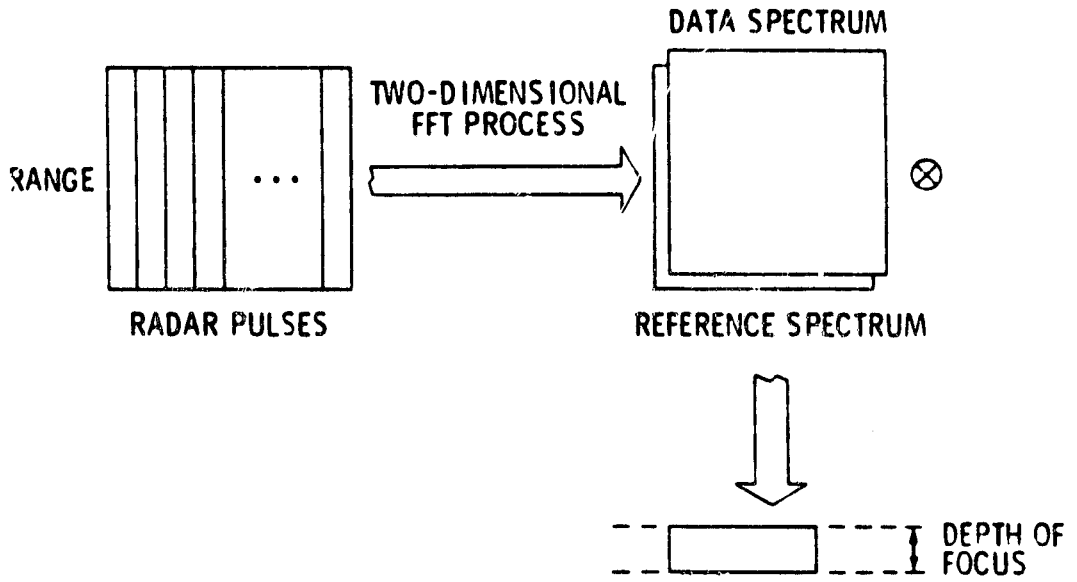


Figure 23

Two-Dimensional Convolution Using FFT Processing

2.2.5.2 2-D Convolution With Range Compressed Data

A 2-D convolution azimuth correlation procedure can also be employed on the range compressed data to accomodate range walk. This function can be done using the standard FFT algorithms (4) or other more recent algorithms such as the fast polynomial transform (5). The fast polynomial transform has been applied to SAR processing (6,7) and a short summary will be provided here.

Consider the convolution of the range compressed data $\{a(m,n)\}$ with reference $\{r(n,m)\}$, each of size MxN*. For the SAR processing in mind, M is the number of range cells for which a single focusing function could cover, typically 128 or 256, and N is the azimuth coverage, typically 4096 or 8192.

From the data array $\{a(m,n)\}$, we generate (r+1) subarrays, where $r = 1 + \log_2(N/M)$. The i-th subarray $\{a^i(m,n)\}$ is of size $Mx(N/2^i)$, $i = 1,2,\ldots,1 + \log_2(N/M)$ except array $\{a^0(m,n)\}$ which is of size Mx(M/2). To generate $\{a^1(m,n)\}$, we take each row of $\{a(m,n)\}$, partition in the middle, take the sum and difference of the corresponding elements:

$$a^1(m,n) = a(m,n) - a(m,n+M/2)$$
$$\bar{a}^1(m,n) = a(m,n) + a(m,n+M/2) \qquad 0 \leq n < N/2 - 1$$

Of the two arrays generated $\{a^1(m,n)\}$ and $\{\bar{a}^1(m,n)\}$. The first one is kept and the second is used to generate $a^2(m,n)$, row by row, as follows:

$$a^2(m,n) = \bar{a}^1(m,n) - \bar{a}^1(m,n + N/4)$$
$$\bar{a}^2(m,n) = \bar{a}^1(m,n) + \bar{a}^1(m,n + N/4) \qquad 0 \leq n \leq \frac{N}{4} - 1$$

This process is illustrated in Figure 23.

* The notation is somewhat different from references (6) and (7).

The process is repeated r times, the last two arrays are $\{a^r(m,n)\}$ and $\{a^0(m,n)\}$ as indicated in Figure 24. The operation takes a total of 2MN additions.

The operation is also precomputed on the reference array $\{r(m,n)\}$ and stored.

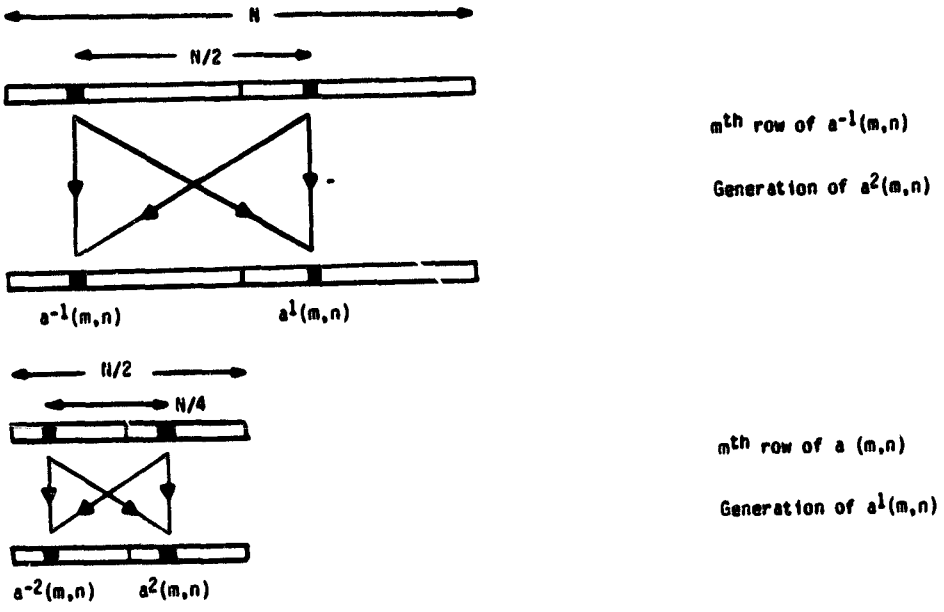


Figure 24a. Generation of $\{a^1(m,n)\}$ from $\{a(m,n)\}$ and of $\{a^2(m,n)\}$ from $\{a^{-1}(m,n)\}$.

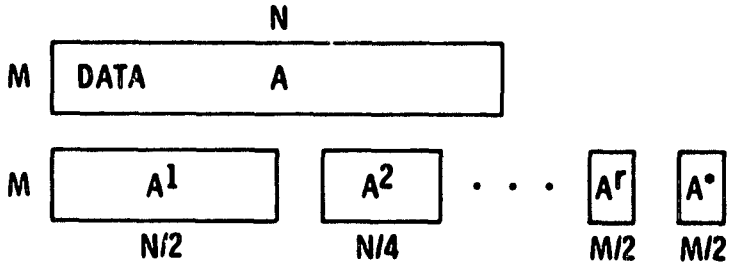


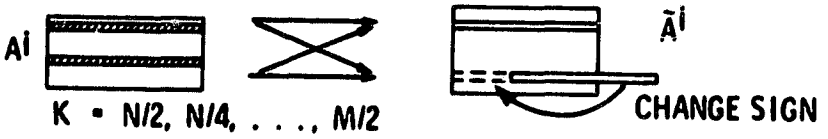
Figure 24b. Repetition of Figure 24a Process

FIGURE 24. Polynomial Transform Operations

The next step is to perform the polynomial transform of the arrays $\{a^i(m,n)\}$ and $\{b^i(m,n)\}$, $i = 0, 1, \ldots, r$. The structure of the polynomial transform is very similar to radix 2 FFT, except each row in the array is treated as a single element. To illustrate with a "DIF" structure applied to the $\{a^1(m,n)\}$ array, we first take two rows M/2 apart, and take the sum and difference of the corresponding elements, i.e.,

$$a^1(m,n) \pm a^1(m+M/2,n) \qquad 0 \leq N/2-1$$

The sum is left alone, but the difference is cyclic shifted (or wrapped around) by a prescribed amount (depending on the corresponding twiddle factor in the FFT structure) and the sign of the wrapped around part is changed as shown in Figure 25. These two rows are put back in the array and two more rows are taken out and operated on in this manner. After M/2 such operations, the first stage is completed. Another set of operation then commences, as in the 2nd stage of the FFT. That is, rows that are M/4 apart are operated on pairwise. This operation is repeated for the $\log_2 M$ stages.



ELEMENT BY ELEMENT
ADDITION AND SUBTRACTION

Figure 25. First Stage of Polynomial Transform

The "butterfly" operation in the polynomial transform can be pipelined by inserting the results back into the array in a criss-cross manner (8), as follows. For the first stage, the first pair is $\{a^1(0,n)\}$ and $\{a^1(M/2,n)\}$, the results are, for example $\{a^1_+(0,n)\}$. The 2nd pair to operated on by the butterfly should be $\{a^1(M/4,n)\}$, and $\{a^1(3M/4,n)\}$, obtaining the results $\{a^1_+(M/4,n)\}$ and $\{a^1_-(M/4,n)\}$. Now the result $\{a^1_+(0,n)\}$ should be stored in row 0, $\{a^1_-(0,n)\}$ in row M/4, $\{a^1_+(M/4,n)\}$ in row M/2, and $\{a^1_-(M/4,n)\}$ in row 3M/4.

The total number of additions for transforming the $\{a^1(m,n)\}$ array is $N\log_2 M$. So the total number of additions for transforming all the $\{a^1(m,n)\}$ arrays is $2N\log_2 M$.

Again, we note that the polynomial transforms of the reference arrays $\{r^i(m,n)\}$ can be precomputed and stored.

We denote the polynomial transforms of $\{a^i(m,n)\}$ and $\{r^i(m,n)\}$ by $\{\tilde{a}^i(m,n)\}$ and $\{\tilde{r}^i(m,n)\}$ respectively.

The next step is a cyclic like convolution of the rows of $\{\tilde{a}^i(m,n)\}$ with $\{\tilde{r}^i(m,n)\}$. The difference from the ordinary cyclic convolution is that the wrapped around part has a sign change. References (6) and (7) suggests the use of an idea proposed by Arambepola and Rayner (9) which accomplishes the convolution via a transform algorithm that is only a slight modification of the FFT algorithm by changing the twiddle factors. One could, on the other hand, use the straightforward FFT to compute the noncyclic convolution and simply do a circular shift and change the signs of part of the result. This step requires $2MN(\log_2 N-3)$ additions and $N(\frac{4N}{3} + \log_2 N-3)$ multiplications.

Finally these (r + 2) arrays are combined to give the cyclic convolution of the data $\{a(m,n)\}$ and the reference $\{r(m,n)\}$. This step involves the merging of arrays of sizes $Mx(N/2^i)$. The merge starts with the summing and differencing, element by element, two smallest arrays, each of size $Mx(N/2^r)$ (Figure 26). The result is a MxM array. This is then merged with the next size array (i=r-1), using the same element by element sum and difference. This process is continued until a final array of MxN is obtained. This is the result of the cyclic convolution.

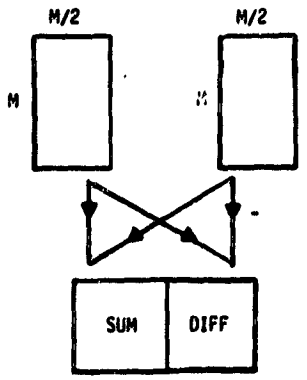


Figure 26
Merging of two MxM/2 arrays

The advantages claimed by the fast Polynomial Transform method are:

1. It requires fewer multiplications than using the FFT.
2. The process of Polynomial transformation can be performed in parallel.

The comparison with FFT in terms of arithmetic operations are summarized in Table 1 of reference 6. The number of additions are comparable and the number of multiplications have a ratio of approximately 5:3 in favor of the polynomial transform. Bergland (10) has shown in 1968 that there is a 30% saving in multiplications if one uses radix 4 rather than radix 2. Recently Nakayama (11) proposed a mixed decimation FFT algorithm which results in a 15% saving over the conventional FFT for the radix 2 case. In his algorithm, the two input butterfly computation kernel is retained. The saving in the radix 4 case is about 5$\sim$8%.

The computation of 2-dimensional transforms is conventionally carried out by first transforming each row and then transforming each column. It has been shown (12) that a simultaneous row-column decimation would result in a 25% saving. It is therefore possible to combine the simultaneous 2-dimensional decimation with the other schemes mentioned previously to obtain a substantial saving over the straightforward FFT approach. Indeed, it has been shown (13) that a 40$\sim$50% saving

possible. Needless to say, the saving is achieved at the expense of increased complexity of the algorithm. But the polynomial transform approach, requiring operations on row vectors of different length also increases the complexity considerably over that of a conventional FFT. Thus, the first purported advantage of savings in arithmetic operations is achievable with other approaches with perhaps a simpler structure of the hardware processor. This latter point definitely would require further study to establish.

The second advantage of the polynomial transform is also not convincingly demonstrated. The arrays $a^i(m,n)$ are of different sizes, the row sizes of successive arrays are two to one. Although these operations can be carried out in parallel, it is doubtful that the saving would be substantial. A considerable portion of the hardware would be idle. Essentially the speed advantage of using parallel operation would be about 2 to 1 but the hardware would be idling about half of the time.

The tentative conclusion we have reached at this time is that, in terms of simplicity of structure, a processor based on the conventional FFT or some minor modifications appears to be most attractive approach for a two-dimensional convolution.

2.2.5.3 Hybrid 2-D Process

A final 2-D process has been suggested by Wu and Liu (14). This approach, indicated in Figure 27, employs FFT convolution for azimuth correlation and a tapped delay line convolver for the range migration correction. Ordering of data appropriately for the tapped delay line convolver is accomplished by doing the azimuth processing with a multiplexed FFT structure. This method, which requires a larger memory storage, outputs data from the first FFT on a single frequency coefficient sequential range basis. Range migration correction can then be applied with a tapped delay line of length equal to the range migration. Modularity and an incremental growth and implementation are also difficult to achieve.

In principal this method is functionally equivalent to the FFT convolver algorithm using range interpolation in the range migration compensation. The

memory required is at about 2.5 times the minimum memory storage required in the FFT convolver corner turning memory. Since the minimum memory storage is about 17 x 10^6 words, the total impact on cost is large.

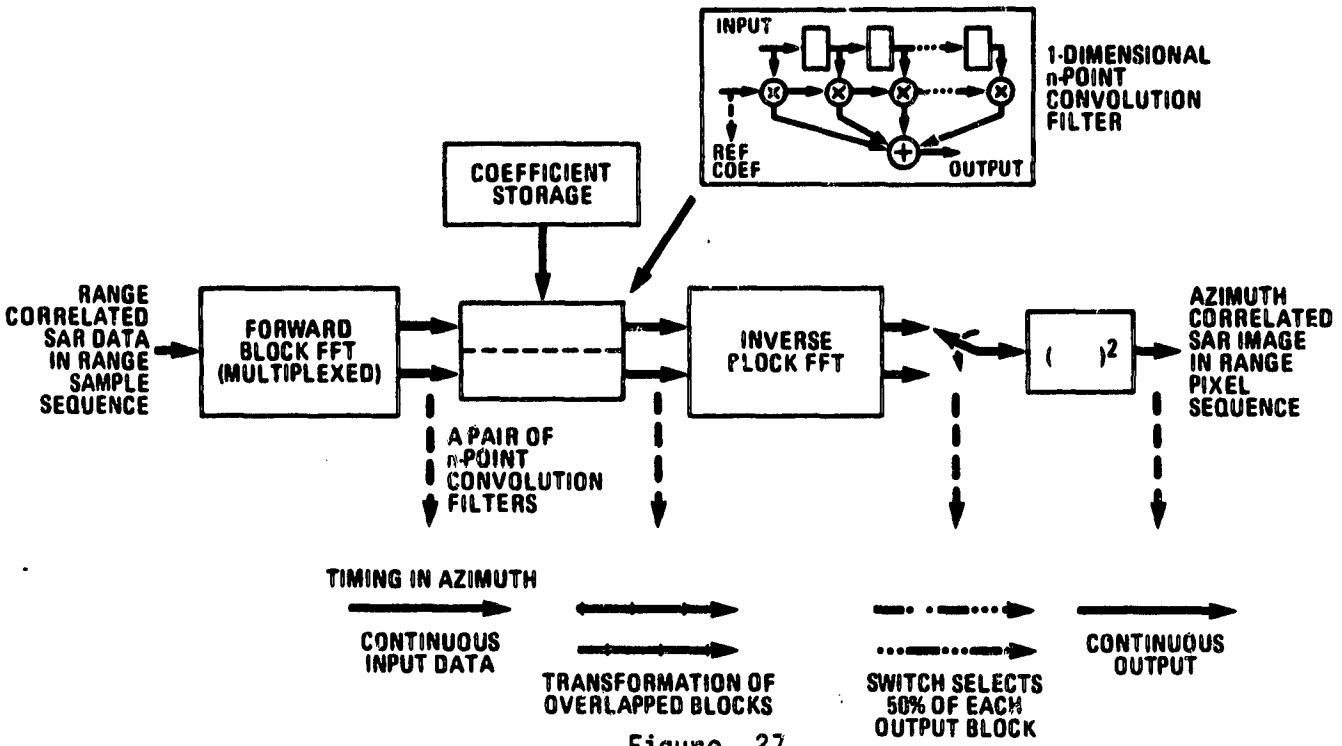


Figure 27

Hybrid SAR Processor

2.2.6. Range Correlation

2.2.6.1 Range FFT Convolver

The forward-inverse FFT frequency domain matched filter, Figure 28 is now an established technique for digital convolution for many applications (15). Matched filtering is accomplished by spectral domain multiplication with the inverse FFT providing the matched filtered time domain output.

The advantages of the forward-inverse FFT matched filter are well known and include:

- A computation-efficient algorithm.
- Complete waveform flexibility.
- Adaptable to modular construction.

- Precise, predictable performance level.
- Implementation can be adjusted to accommodate different processing speed requirements.
- Will benefit from advanced technology developments.
- Direct interpolation of output data possible.

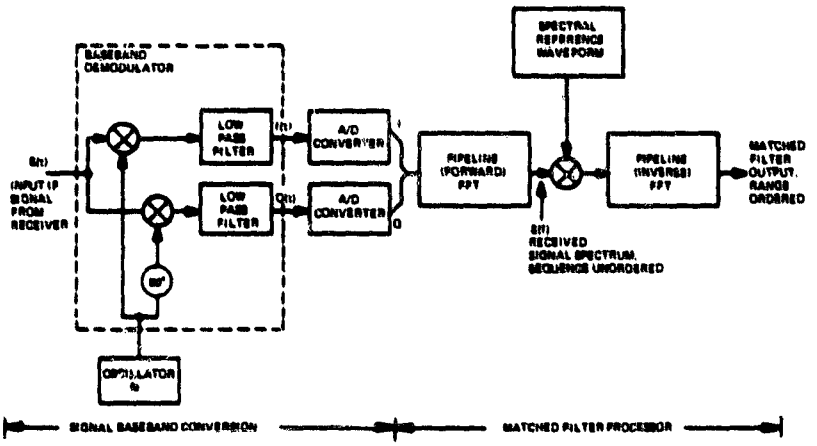


Figure 28
FFT Matched Filter System

The input data can be processed from real sampled data as might be acquired with the intention of processing in an optical processor. Using the FFT for providing the real-complex conversion is the most convenient approach. If this is done the input data window is normally twice as long, but two real channels can be processed simultaneously in the complex FFT window.

The FFT matched filter (range correlator) will functionally have the form shown in Figure 29. Because the range swath interval is a fraction of the total radar pulse repetition interval (PRI) the range data is normally buffered after A/D conversion to slow down the data rate to the range correlator.

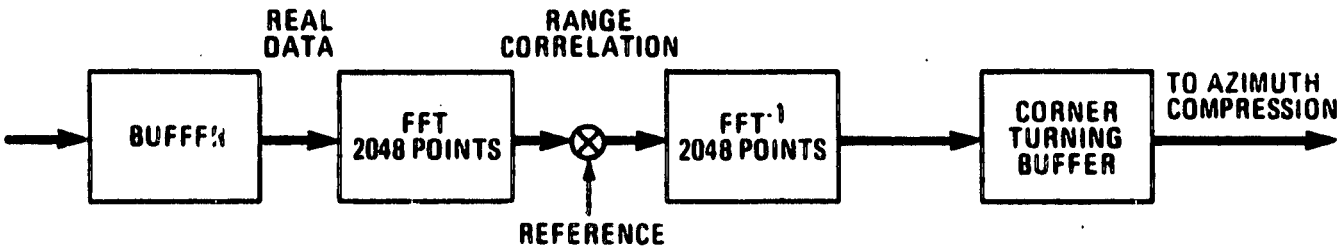


Figure 29

FFT Convolution-Range Processing

A tradeoff exists in setting the FFT processor parameters of an FFT convolver. In general, the larger the size of the FFT, the more efficient the computations. For the baseline system range processor, the total number range samples to be processed per pulse is 4000 (range image swath) plus 475 (uncompressed pulse length) plus 86 (range migration) for a total 4561 samples. This can be processed as a sliding aperture convolver with an FFT size of 1024 points or as a batch processor of 8192 or 4096 points. If the maximum FFT size is 4096, a seperate FFT of 1024 must be processed to cover the total range interval. The net computation rate for the three cases is given in Table 8 which shows that the most computationally efficient size for the baseline system is the 4096 point FFT.

Table 8

Range FFT Convolver Alternatives

CHARACTERISTIC	SLIDING APERTURE	MAX-SWATH PROCESSOR
FFT SIZE	≥ 2 WAVEFORM (1024)	$\geq 4000 + 475 + 86$ (8192 OR 4096 + 1024)
COMPUTATIONS PER PULSE	$8\lvert 512(2\log_2 1024 + 2)$ (90,112)	$\frac{8192\ \text{FFT}}{4096(2\log_2 8192 + 2)}$ (114,688) $\frac{4096 + 1024\ \text{FFT}}{2048(2\log_2 4096 + 2) + 512\ (22)}$ (64,512)
COMPUTATION RATE (COMPLEX OPERATIONS / sec)	225×10^6	8192 FFT : 287×10^6 4096 + 1024: 161×10^6

2.2.6.2 Step Transform Linear Frequency Modulated (LFM) Signal Matched Filter

The step transform subarray algorithm was originally conceived as a technique for LFM matched filter processing (16,17). The algorithm is shown symbolically in Figure 30. A received LFM ramp is demodulated by a sawtooth ramp with the same slope, producing CW frequency segments "stepped" by the frequency span of the sawtooth (Δf) and of length equal to a tooth (ΔT). A spectral analysis of the successive CW segment produces a set of frequency response functions that are stored in a time-frequency data reorder memory shown in Figure 32. Data as read from the time frequency matrix along diagonals results in a linear phase shift along the diagonal producing the "fine" range resolution at the processor output. Weighting is applied prior to the second FFT to reduce range sidelobes.

The step transform pulse compression technique provides a natural segmentation of data as a function of range. It offers a means of applying range migration compensation to small blocks of range samples. Its main disadvantage is that it is not universally programmable for waveform type or time-bandwidth product.

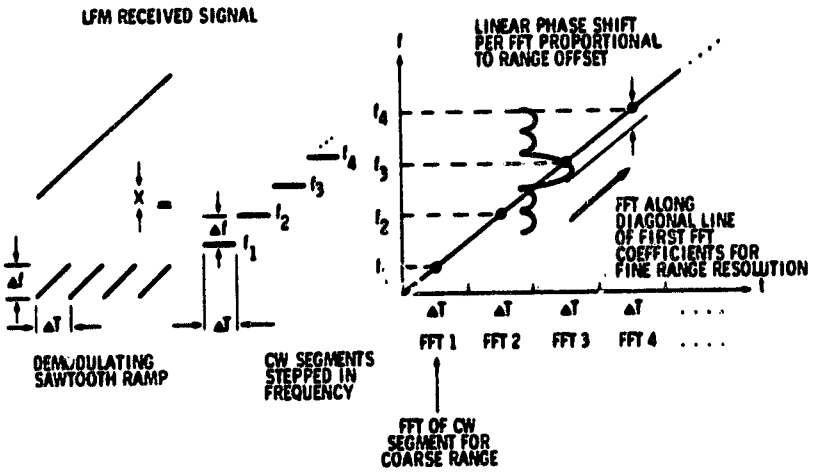


Figure 30

Step Transform LFM Pulse Compression Algorithm

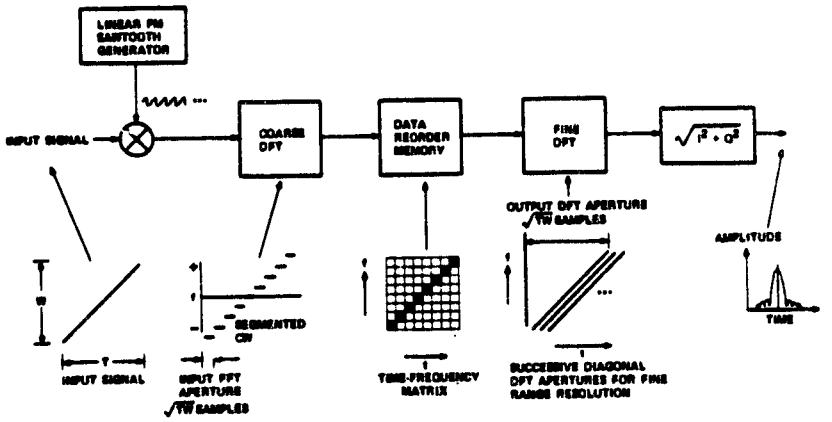


Figure 31

Step Transform LFM Range Pulse
Compression Processing

(For maximum efficiency, DFTs are implemented as FFTs)

2.2.6.3 Digital Tapped Delay Line Correlator

Tapped Delay Line Pulse Compression Filters - The output y(t) of a matched filter implemented via convolution of an input signal s(t) with the impulse response of the matched filter h(t) = x(-t), where x(t) is the transmitted waveform, is:

$$y(t) = \int_{-\infty}^{\infty} S(\tau)H^*(t-\tau)d\tau$$

The signal and filter functions are generally represented as complex inphase and quadrature samples for sample data operations. Thus, a physical realization of the filter function must accomodate the complex multiplication operation.

$$(a + jb)\ (c - jd) = ac + bd + j(bc - ad)$$

A tapped delay line matched filter then takes the form of Figure 32. The total number multipliers is over $4(TW)^2$ where TW is the time bandwidth product of the waveform. With a maximum TW product of 660 for the ADSP, the total number of taps required is 765.

A time domain range processor is very simple to use. It is only necessary to input the radar waveform of arbitrary length into the reference register together with the appropriate number of zeros. If a single unit is used as in Figure 32, however, it must operate at a clock rate of 11 MHz. The system would be partitioned into 130 modules with up to 200 circuits per module all operating synchronously at an 11 MHz clock rate. The total number of circuits for the range correlator would in this case be about equal to the total number required by both the range and azimuth correlators using FFT convolver algorithm.

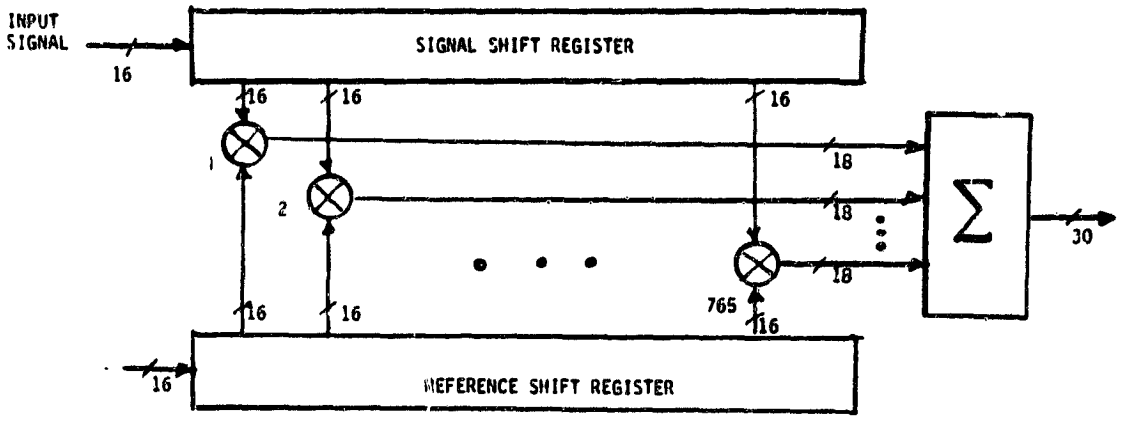


Figure 32

Time Domain Range Convolver Controls

2.2.7 Algorithm Study Summary

A graphical depiction of the SAR baseline system performance parameters for the azimuth processor are given in Figure 33. It shows the relationship of the total beam integration time of 28 seconds to the radar PRF. The time-bandwidth product per look is 0.7 seconds times 500Hz = 350. Using these parameters, a comparison was made of the azimuth processor memory size and computation rate for the three main processing algorithms as a function of the number of looks.

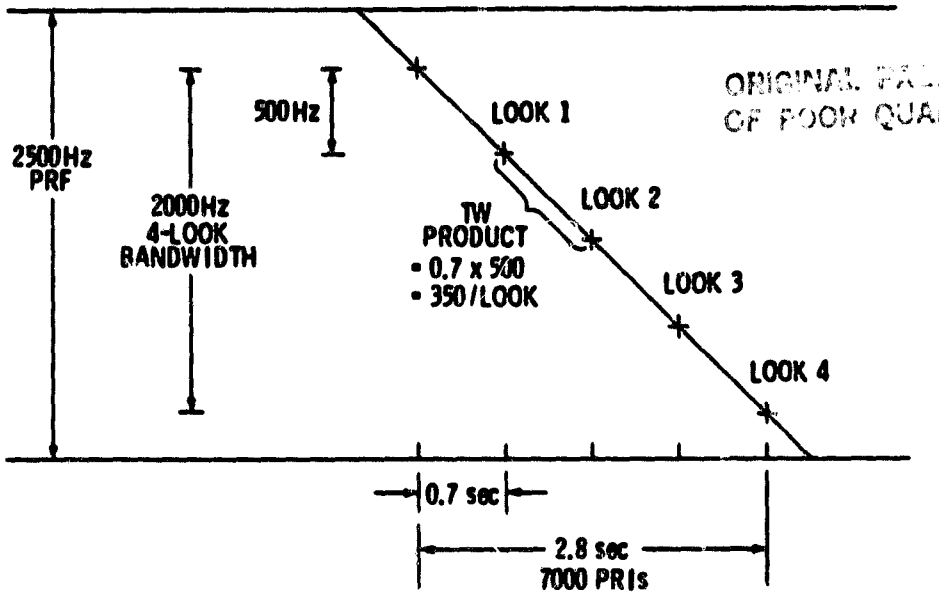


Figure 33

Baseline Azimuth Parameters

The results are given in Table 9. It shows results as generally expected with the subarray approach having the lowest computation rate and the time domain processor exhibiting the smallest memory. However, the computation rate of the time domain processor is much higher than the other approaches. This high computation rate translates into hardware as demonstrated in Section 2.2.2 and eliminates the time domain approach as a contender. The tradeoff between complexity of control and hardware is a factor between the FFT convolver and subarray approaches.

The 2-D convolution approaches have not been included since they are similar to the FFT convolver algorithm. In the approaches using 2-D convolution of range compressed data the question is of the relative advantage of 2-D convolution for range migration compensation versus range interpolation. Simulation results presented in Section 2.3 indicate that a simple two point interpolator will give an adequate integrated sidelobe performance level. A four point interpolator is close to the ideal and its implementation in the selected design is not a computational burden.

Based upon its greater ease and completeness of programmability and

its lower cost for development, the FFT convolver algorithm has been selected for the ADSP recommended design. Further details of the design and selection issues are given in Section 2.5. The design and architecture have also lent themselves to the selection of the FFT convolver for the range processor.

Table 9

Look-Parameter Azimuth Processor Variations
for Baseline SAR System

LOOKS	FFT CONVOLVER		SUBARRAY		TIME DOMAIN	
	COMP RATE	MEMORY	COMP RATE	MEMORY	COMP RATE	MEMORY
1	306	67.1	171	45.6	77,000	28.0
2	276	33.6	160	22.8	19,250	14.0
4	245	16.8	148	11.4	4,813	7.0
8	215	8.4	138	5.7	1,203	3.5

- COMPUTATION RATE IN MILLIONS OF COMPLEX OPERATIONS PER SECOND
- MEMORY STORAGE IN MILLIONS OF WORDS
- TOTALS EXCLUDE RANGE MIGRATION COMPENSATION

2.3 Performance Levels

The selection of a signal processing configuration for the ADSP required the determination and comparison of the relative performance of each candidate algorithm. The analysis/simulation of the algorithms is described in this section. The simulation results permitted cost versus performance tradeoffs. A design requirement that was particularly important in the tradeoffs was the integrated sidelobe level (ISL). A design goal of -20 db ISL in two dimensions (range and azimuth) was established as a minimum requirement.

2.3.1 Performance Procedure

The organization of the simulation was broken into two parts. The first part is the generation of a representative time signal of interest. The second was the processing of the signal using two different algorithms, the FFT-convolver and the step transform subarray.

2.3.1.1 Generation of the time Signal

The objective of the first part of the simulation was to generate the time response of a point target. The simulated target was assumed to have already been matched filtered in the range dimension by FFT convolution. The range response was Hamming weighted and was evaluated exactly at the sampling points. If the target peaked at a non-sampled point, the response at the sampling points was generated relative to the peak. The phase of the target was generated using

$$\phi(t) = 2\pi V^2 t^2 / \lambda R$$

where V is the velocity of the spacecraft (assumed to be a constant), t is the relative sampling time, λ is the wavelength of the transmitted pulse, and R is the slant range to the point source.

The effect of the range migration was obtained by displacing the Hamming weighted function. Range migration is composed of linear and quadratic components. The linear range migration component is primarily due to the earth's rotation. The quadratic component varies in a non-linear manner as a function of beam angle. Figure 34 shows the two components of the range migration as they were modeled in the

simulation.

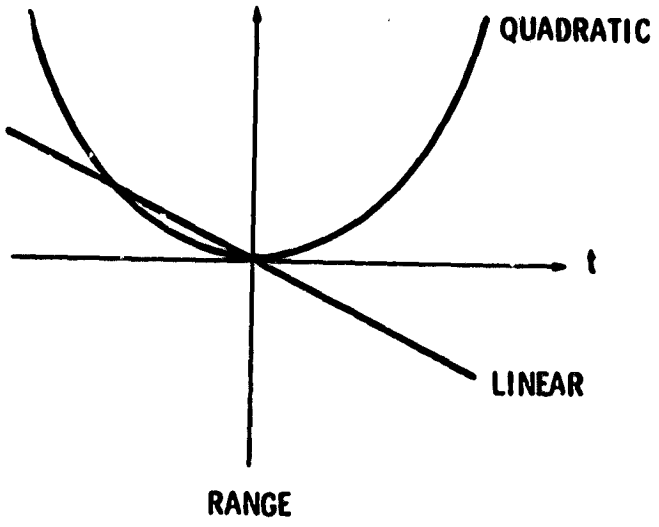


Figure 34
Quadratic and Linear Range Migration

Linear range migration is simply a slope parameter in the simulation. The quadratic displacement is given by

$$Q = (V^2 t^2/2R)/(c/2B)$$

where V is the velocity of the spacecraft, t is the relative time, R is the slant range, c is the velocity of light, and B is the bandwidth of the system. Figure 35 shows the relationship between the quadratic range migration (35a) and the corresponding linear FM function (35b). The maximum range, Rm, in Figure 35a corresponds to the maximum and minimum frequencies ($+f_m/2$ and $-f_m/2$) in Figure 35b. The minimum range migration corresponds to the zero crossing point of Figure 35b. The signal is present along limited portions of the spectrum as a function of time and generates a linear FM (LFM) response. The generated signal migrates to different range cells as a function of time and beam angle.

Figure 35a
RANGE MIGRATION

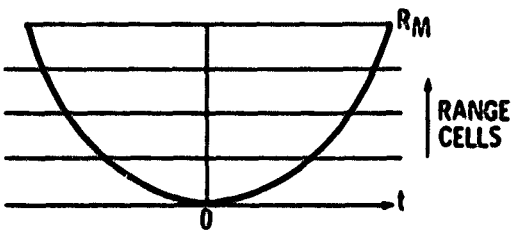
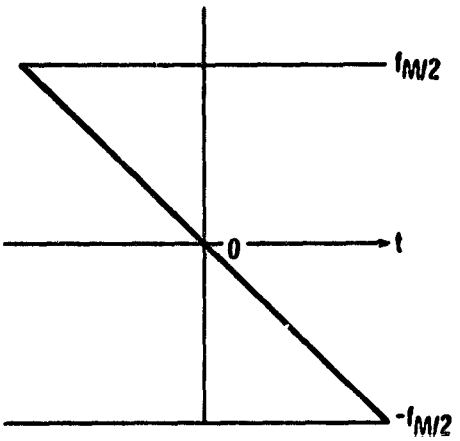


Figure 35b
LFM FOCUSSING FUNCTION



R_M -- MAXIMUM RANGE MIGRATION

f_M -- FREQUENCY CORRESPONDING TO R_M

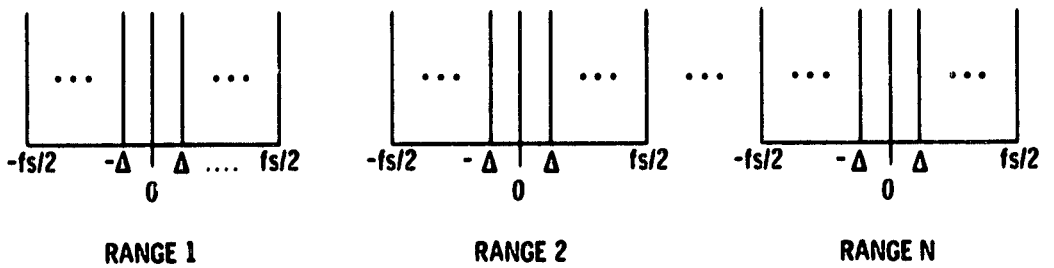
Figure 35
Generation of the Time Signal

2.3.2 FFT - Convolver Algorithm

2.3.2.1 Range Migration Compensation

The FFT-Convolver SAR processing algorithm was simulated first. In this approach, the azimuth data is transformed to the frequency domain along constant range lines. In Figure 35a, the transform is performed along the horizontal lines. Since the energy of a point source is dispersed into different range lines, compensation must be provided prior to matched filtering in the azimuth direction. Figure 36 shows the Fourier transform of N range cells. The compensation for the range migration consists of moving the Fourier coefficients into the proper range cells. The number of range migration cells is equal to $\lambda^2 R n^2 / 8 V^2 R_c$

where λ is the wavelength of the transmitter, R is the minimum slant range, V is the velocity of the spacecraft, R_c is the range cell width and n is the Fourier coefficient number.



Δ = fs/LENGTH OF THE FFT
fs -- SAMPLING FREQUENCY

● RANGE WALK PER Λ = CONST $\cdot(\Delta n)^2$
CONST = $\lambda^2 \cdot r_m / 8V^2 \cdot R_C$

$\frac{-N}{2} \leq n < \frac{N}{2}$

R_C = RANGE CELL WIDTH
λ = WAVELENGTH OF RADAR FREQUENCY
r_m = MINIMUM SLANT RANGE
V = VELOCITY OF SPACECRAFT
N = FFT TRANSFORM SIZE

Figure 36

Range Migration

The amount of range walk as a function of frequency allows for the proper alignment of the coefficients.

The simpliest adjustment to the coefficient would be to simply slide the data by integral steps (8). For example if the migration for a given coefficient were 5.4 range cells then the data would be obtained from the coefficient 5 range cells offset. For a migration of 6.7 range cells, the data would be obtained from 7 range cells array. This method, while simple, would tend to cause some output distortion.

Interpolation of the data shift eliminates the effects of the discontinuities of integral transfers. Interpolators of two, three and four points were simulated. Table 10 gives the equations of the interpolators used in the simulations.

Table 10

Interpolation Formulas

- 2pt: $(1-p)\ X_0 + p\ X_1$
- 3pt: $\frac{p(p-1)}{2}\ X_{0-1} + (1-p^2)\ X_0 + \frac{p(p-1)}{2}\ X_1$
- 4pt: $\frac{-p(p-1)(p-2)}{6}\ X_{-1} + \frac{(p^2-1)(p-2)}{2}\ X_0 + \frac{-p(p+1)(p-2)}{2}\ X_1 + \frac{p(p^2-1)}{6}\ X_2$

2.3.2.2 Azimuth Matched Filter

Following the range walk compensation, each of the range lines are matched filtered by multiplying the transform of the azimuth focusing function. The azimuth focus function has a linear FM slope of

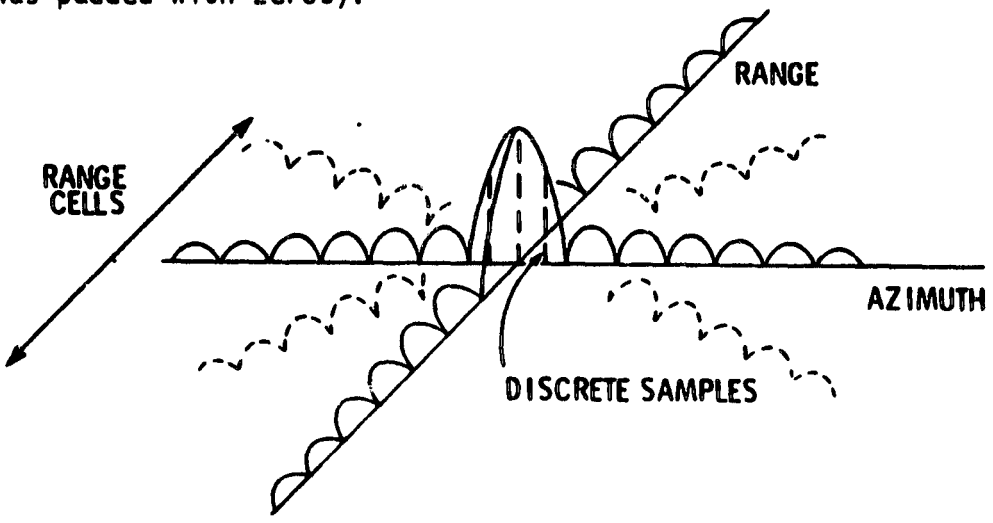
$$a = 2\pi V^2/R\lambda$$

and length determined by the maximum integration line. The simulation has the option of applying a non-uniform weighting across the time response of the azimuth matched filter. The location of the time aperture of the azimuth matched filter depends upon which of the four looks is being processed.

2.3.2.3 Weighting

Figure 37 outlines the terms that are used in the discussion of the simulation and results. The waveform in Figure 37 represents the compressed response in both range and azimuth. The mainlobe of the response can be defined in a number of ways. The most common definitions use widths about the peak to the 3 db points, 6 db points or to the first nulls. Sidelobes are then defined as everything that does not lie within the mainlobe. The peak sidelobe level versus the peak mainlobe (PSL/PML) is defined as the largest value found outside the mainlobe divided by the largest value found within the mainlobe, usually expressed in db. The integrated sidelobes versus the integrated mainlobes (ISL/IML) is defined as the integration of the region not found in the mainlobe divided by the integration of the mainlobe region,

again usually expressed in 1 dB. The PSL/PML and the ISL/IML can be given for one or two dimensions. The one dimension (1D) case uses a single range line containing the mainlobe peak. The two dimension case (2D) is the two dimensional plane of azimuth and range. Using the definitions of Figure 37, Table 11 gives a brief review of different weighting functions. Weighting selection represents a compromise between resolution or width of the peak response and sidelobe levels. This is clearly seen in comparing the increase in the mainlobe width (3 dB and null columns of Table 11) to the decrease in peak sidelobe and integrated sidelobe levels (the columns labeled PSL/PML and ISL/IML). The results of Table 11 were obtained using a signal aperture of 64 samples and a total time history of 1024 samples (the rest of the signal was padded with zeros).



PSL/ML -- PEAK SIDELOBE VERSUS THE MAINLOBE (dB)

ISL/IML -- INTEGRATED SIDELOBES VERSUS THE INTEGRATED MAINLOBE (dB)

- 1D -- ALONG AZIMUTH DIRECTION
- 2D -- ALONG AZIMUTH AND RANGE DIRECTION
- MAINLOBE -- POINTS LYING WITHIN THE FIRST NULLS FROM THE PEAK
- SIDELOBE -- ALL POINTS NOT IN THE MAINLOBE
- 3dB -- WIDTH OF THE MAINLOBE AT THE 3dB POINTS
- 6dB -- WIDTH OF THE MAINLOBE AT THE 6dB POINTS
- NULL -- WIDTH OF THE MAINLOBE TO THE FIRST NULLS

Figure 37

Sidelobe Definitions

Table 11

Ideal Characteristics of Weighting Functions

LENGTH: 64 FFT: 1,024

TYPE	PSL/ML	ISL/IML	3 dB	NULL
RECTANGULAR	13.26	9.68	1.0	2.2
HAMMING	42.48	34.42	1.5	4.5
BARTLETT	26.53	25.30	1.5	4.3
HANNING	31.50	32.91	1.7	4.5
BLACKMAN	58.14	58.89	1.9	6.6
25 dB TAYLOR	24.34	18.96	1.2	3.0
30 dB TAYLOR	30.98	23.55	1.3	3.4
35 dB TAYLOR	35.93	27.69	1.3	3.7

•PSL/ML -- PEAK SIDELOBE TO MAINLOBE RATIO (dB)
•ISL/IML -- INTEGRATED SIDELOBES TO INTEGRATED MAINLOBE (dB)
•3 dB -- RELATIVE WIDTH OF MAINLOBE TO 3 dB POINTS
•NULL -- RELATIVE WIDTH OF MAINLOBE TO FIRST NULL POINTS

2.3.2.4 Time Domain Output

The final processing of the signal in the azimuth FFT convolver is to apply the inverse Fourier transform to obtain the time domain response. In the simulation, the time domain output is not decimated as would be the case in an actual signal processor. This reduces the measurement error since the nondecimated output provides a higher output sampling rate.

2.3.2.5 Interpolation and Weighting Simulation Results

A number of simulation runs were performed to measure the peak sidelobe levels and the integrated sidelobe levels for different order interpolators. The system parameters used in the simulations were a constant spacecraft velocity of 7.45 km/sec, a slant range of 850 km, 22 MHz sampling rate in the range direction, a pulse repetition rate of 1.6 kHz and an azimuth coverage of 4096 pulses. The signal was placed in the middle of the scan and the scan was processed for four looks. The radar system pulse bandwidth in the range dimension assumed to be 19 MHz (over-sampling in the range direction by 1.16). The transmitter frequency was assumed to be 1250 MHz.

Table 12
FFT Convolver Interpolator Results

- CASE 1: NO INTERPOLATION

LOOK	1D ISL/IML	2D ISL/IML	2D PSL/ML
1	26.3	12.4	19.8
2	27.8	12.6	19.6

- CASE 2: FOUR POINT INTERPOLATION

LOOK	1D ISL/IML	2D ISL/IML	2D PSL/ML
1	31.1	28.9	35.8
2	32.8	32.6	36.3

- MEASUREMENTS IN dB
- HAMMING WEIGHTING

Table 12 summarizes the performance obtained by incorporating a four point interpolator with range migration compensation. Since the signal was placed in the middle of the scan, look 3 has the same results as look 2, and look 4 the same as look 1. Looks 3 and 4 are thus not shown in Table 12. Table 12 shows that the ISL/IML performance is improved by the four point interpolator over no interpolator in one dimension only (column labeled '1D') by only 5 db at a 30 db level. The improvement is of the order of 20 db however when a two dimensional integration is examined (column labeled '2D'). The peak sidelobe level versus the mainlobe occurs in the one dimensional line and is thus the same for either 1D or 2D (column labeled 'PSL/PML'). The large degradation in the ISL/IML with no interpolator is expected. Shifting of the Fourier coefficients without interpolation results in discontinuities that raise the level of the sidelobes in the range cells surrounding the signal.

The results in Table 12 were obtained using a Hamming weighting function across the azimuth matched filter. Table 13 illustrates the reason for using a weighting function to reduce the sidelobe levels at the expense of a reduction in the resolution in a simulation. If no weighting were employed, the results would be ambigious on the need for an interpolator in the processor. The column labeled 'Rect' shows

only a 1.5 to 2.0 improvement for the interpolated case. What has happened is that the high sidelobes of the rectangular weighting function (window) have contributed to the integrated sidelobe. Using a window with greater sidelobe suppression provides results which are sensitive to processing differences. A triangular window (column labeled 'Triangle') suppresses the PSL/PML level of the window to 27 db from 13 db for a rectangular window and also the falloff of the sidelobes is at a 12 db/octave rate versus 6 db/octave for rectangular. The mainlobe increases by approximately 40%; however, the difference due to processing can now be seen. The ISL/IML difference between no interpolator and a four point interpolator is approximately 12 db. Using Hamming window which has a PSL/ML of 43 db, a falloff rate of 6 db/octave and an ISL/IML of 34.42 db, the processing difference is seen to increase to approximately 20 db (column labeled 'Hamming'). Considering that the theoretical ISL/ML for a rectangular window is 9.68 db and a PSL/ML of 13.26 db, the effects being seen in the "rect" column for the no-interpolation case are largely due to processing (the ISL/IML is approximately 8 db), while for the four point interpolator it is the window (the ISL/IML is 9.7 db). On the other hand, both processor's performance is being measured when the Hamming window is used.

Table 13

Effect of Weighting Functions, FFT Convolver

- CASE 1: NO INTERPOLATION

	LOOK	RECT	TRIANGLE	HAMMING
ISL/IML (PSL/ML)	1	7.8 (13.5)	12.4 (19.8)	12.4 (19.8)
	2	8.1 (12.8)	-	12.7 (19.6)
AZIMUTH RESOLUTION		1		1.4
RANGE RESOLUTION		← CONSTANT →		

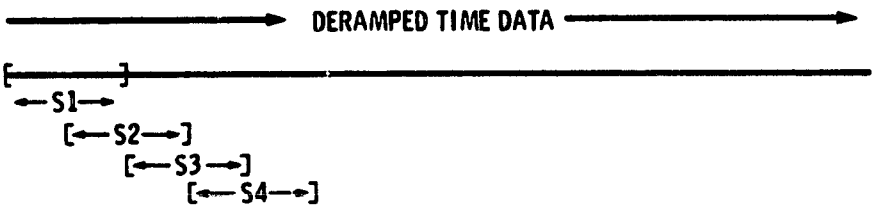
- CASE 2: FOUR POINT INTERPOLATION

	LOOK	RECT	TRIANGLE	HAMMING
ISL/IML (PSL/ML)	1	9.8 (13.5)	24.0 (26.7)	29.1 (35.8)
	2	9.7 (13.2)	25.1 (27.0)	32.8 (36.3)
AZIMUTH RESOLUTION		1	1.4	1.4
RANGE RESOLUTION		← CONSTANT →		

2.3.3 Step Transform - Subarray Approach

2.3.3.1 Description of Subarray Simulation

The second algorithm simulated was the step transform-subarray. The first step in the process after range compression is the deramping of the complex time signal by the azimuth reference LFM. The deramped signal is then sectioned into subarrays. The number of samples in each subarray and the amount of overlap between the subarrays were systems parameters examined in the simulation. An example of a subarray length of 64 with 2 to 1 overlap is depicted in Figure 38.



- EXAMPLE
  - LENGTH OF SUBARRAY -- 64
  - OVERLAP (1/2) -- 32
- SUBARRAYS
  - S1 (1-64)
  - S2 (33-96)
  - S3 (65-128)
  - S4 (97-160)
  - ⋮

Figure 38
Subarray Formation

After forming the subarrays, each subarray is transformed into the frequency domain using an FFT. The subarrays are then phase corrected using a correction factor of $\exp(-j2\pi kd/N)$ where k is the FFT coefficient number, d is the displacement between subarrays and N is the length of the FFT. The original signal has been transformed into a three dimensional array with the Fourier coefficients on one axis, the subarray number on the second axis, and the range cell number on the third. Figure 39 shows a two dimensional slice of the

three dimensional array with the range dimension fixed and the FFT length 64.

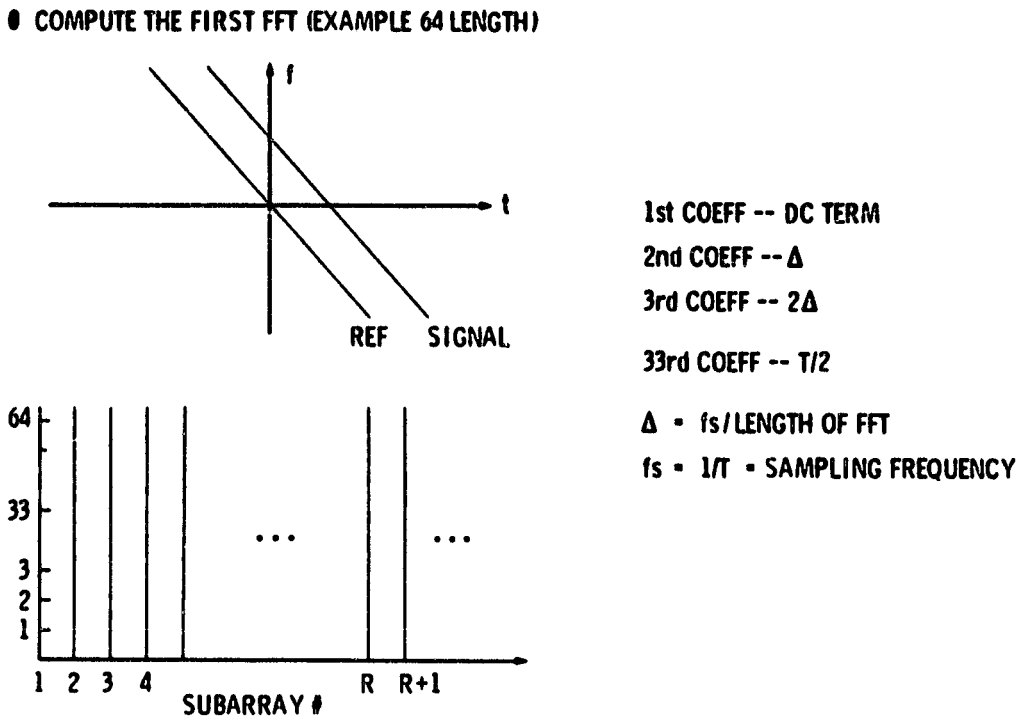


Figure 39
First FFT in Subarray Approach

Assuming that the signal depicted in Figure 39 has no range migration, then the deramped subarrayed signal would appear entirely in the plane. If for example, the difference between the deramping reference and the signal was exactly 3 Δ and the signal began at the 256 sample, then the fifth subarray would be the first to see the signal and its transform would have the fourth Fourier coefficient as the highest of the 64 coefficients (assuming no other signals are present). The fourth coefficient would be high for each subarray in which the signal was present. In the case where the signal exactly matches the reference, the first (DC coefficient) would be the highest of the 64.

However, the signal does not stay within a single range cell. The range migration traverses a quadratic function in the third dimension. Consider three signals, one that exactly matches the reference, one that is delayed in time by an amount which corresponds to the signal peaking in the 2nd Fourier coefficient and the third is delayed by an amount

equal to the third coefficient. These three signals are depicted in Figure 40a as 0, ΔT and $2\Delta T$ delay from zero time. Assuming that the first signal begins at the first sample, then the first coefficient of the first array would have to be range corrected the maximum amount. The first coefficient of the 2nd subarray would be corrected according to the range migration correction formula used in the FFT-Convolver approach. The amount of range migration and correction would diminish as the higher number subarrays are addressed until the subarray corresponding to the zero time of Figure 40a is addressed. This point is shown as the first point on the zero correction line of Figure 40b. The correction would then increase to the maximum. Since the signal was assumed to be matched to the reference, then all of the correction is applied to the first Fourier coefficient.

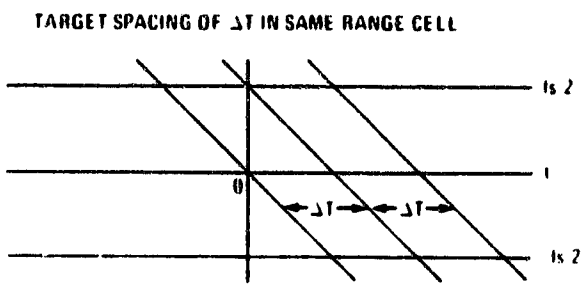


Figure 40a
Migration of Target Across Subarray

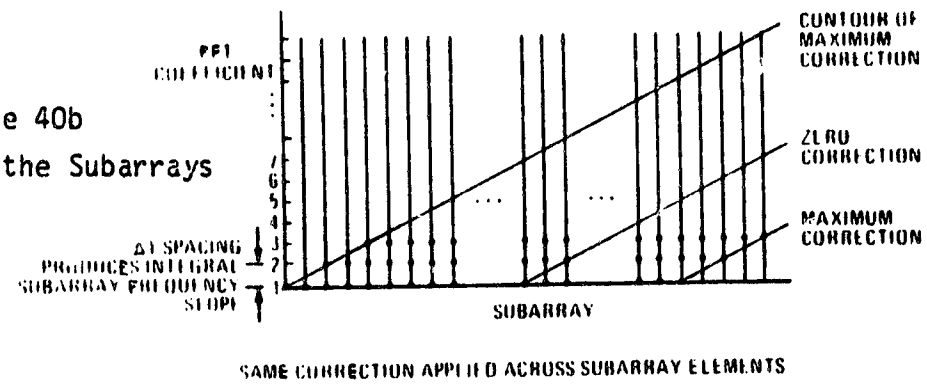


Figure 40b
Deramp and FFT the Subarrays

Figure 40
Subarray Range Migration Correction

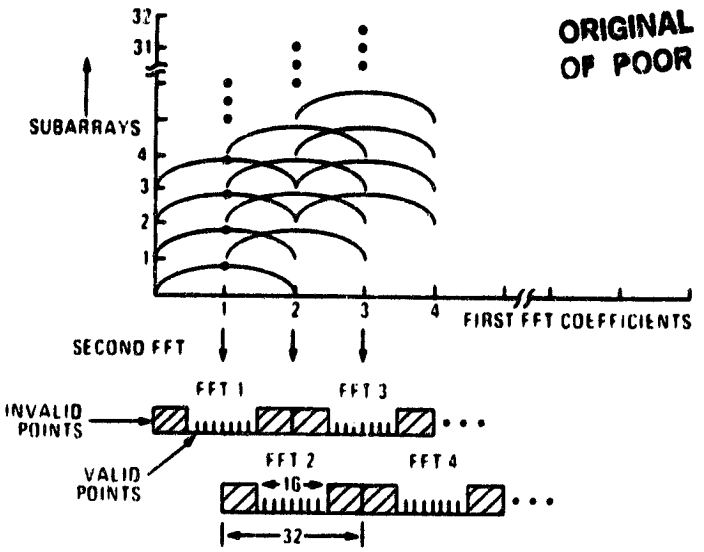
The second signal in Figure 40a would first appear at a subarray number corresponding to the time delay ΔT. Once the signal starts, the correction function would proceed from the maximum amount down to zero and back up to the maximum in the same quadratic shape as the

first signal. The only difference is that the correction is performed on the 2nd FFT coefficient. The third signal would have the same correction performed on the 3rd coefficient. Figure 40b depicts the correction across the subarrays as the straight lines labeled "contour of maximum correction", "zero correction", and "maximum correction". The slope of the parallel lines would be given by

$$\text{slope} = (f_s^2 \quad T)/(N \Delta \text{LFM} d)$$

where f_s is the azimuth sampling frequency, N is the subarray length, T is the total time that the signal is present, Δ LFM is the total change in the LFM reference and d is the number of points that the subarrays overlap. The correction scheme used for the range walk is no different than that used in the FFT-Convolver algorithm. The data is shifted in integral amounts for no interpolation and is interpolated between points otherwise.

The final part of the processing in the subarray approach is to perform the focusing FFT. The same slope that is used in the range migration correction is used to address the subarrays for the focusing FFT. In the case of a non-integral slope, the nearest subarray is chosen. The signal is then padded with zeros which minimizes the measurement errors of the analysis. The FFT is computed and the proper coefficients are selected from the FFT. Each set of coefficient are abutted to form the output. The selection of the proper coefficients is analogous to a comb filter bank where each output filter is tuned to only certain center frequencies and a given bandwidth about them. Figure 41 depicts the process of selecting the output values.



DETERMINATION OF VALID COEFFICIENTS OF FOCUSING FFT (=)

- $\Delta F = f_s/N$ (FREQUENCY SPACING OF FIRST FFT)
- $d = n\Delta T$ (OVERLAPPING DISPLACEMENT)
- $\Delta F' = (1/d)/N'$ (FREQUENCY DISPLACEMENT OF FOCUSING FFT)
- $\# = nN'/N$

Figure 41
Coefficient Selection

2.3.3.2 Subarray Simulation Results

The first issue to be examined in the step transform subarray algorithm is the amount of overlap required between successive subarrays. The results of the simulation of two overlap cases is given in Table 14. Case 1 is an overlap of 4 to 1 and case 2 is an overlap of 2 to 1. That is, case 1 represents a subarray spacing of one fourth of the subarray length and case 2, a subarray spacing of one half its length. The column labeled 'COEFF' is the FFT coefficient output by the focusing FFT (second FFT). The parameters for the second or focusing FFT were chosen so that both cases had the same number. The 33rd coefficient in column 'COEFF' corresponds to the DC component of the good coefficients and the 65th the last valid coefficient. These two coefficients represent the extremes with the other 32 coefficients lying between. As can be seen from the 'ISL/IML' columns, the greater the overlap or subarray sampling rate the better the performance. Increasing the overlap increases the spacing between the main subarray lobe and its grating lobe caused by subarray sampling. However, higher overlap ratios also increase the required amount of computational effort.

Averaging over the 64 coefficients for case 2, the average performance is found to be 25.6 db for 2D ISL/IML which is well below the 20 db required.

Table 14

Subarray Overlap

		1D		2D	
CASE	COEFF	ISL/IML	PSL/ML	ISL/IML	PSL/ML
1	65	32.0	40.7	31.9	40.8
	33	37.0	42.5	37.0	42.5
2	65	21.9	19.0	18.9	16.7
	33	32.1	37.1	31.3	37.1

		CASE 1	CASE 2
1st FFT	•	64	64
SUBARRAY SPACING	•	16	32
# SUBARRAYS	•	64	32
2nd FFT	•	256	128

•BOTH FFTs HAMMING WEIGHTING
•LOOK 2
•AVERAGE ISL/IML FOR CASE 2 OVER 64 COEFFICIENTS 25.6 dB

The effects of the grating lobes on the subarrays can also be reduced by a proper choice of the weighting function across the first FFT. The window is chosen to place a null near the grating lobe. Table 15 shows that a 30 dB Taylor weighting will meet the requirements.

The table also shows that optimum performance with the first FFT weighting function is achieved by a careful balance between mainlobe width, which affects the amplitude of the output grating lobe, and sidelobe level which sets overall ISL/IML.

Table 15

Effect of Weighting of First FFT
in Subarray Process

TYPE	COEFF	2D ISL/IML	PSL/ML
HAMMING	65	18.9	16.7
	64	18.3	17.3
	34	31.2	37.1
	33	31.3	37.1
25dB TAYLOR	65	15.5	22.9
	64	15.6	23.1
	34	34.3	41.5
	33	34.8	41.9
30dB TAYLOR	65	21.3	24.0
	64	21.4	23.6
	34	32.7	38.4
	33	32.7	38.1

*30dB TAYLOR WEIGHTING WILL MEET THE REQUIREMENTS

2.3.4 Performance Comparisons

Having chosen the parameters which will meet the requirements for the different algorithm the relative performance between algorithms is shown in Table 16. Table 16 shows the performance with and without the addition of linear range migration to the quadratic range migration. The effects of interpolator complexity is also summarized in Table 16. For the FFT/Convolver approach a 2-point interpolator would seem to be adequate since 13 db of 18 db of the processing improvement compared to no interpolator has been obtained. For the subarray approach a 3-point interpolator appears adequate. The integrated sidelobe values in Table 16 for the subarray system have been averaged over all valid coefficients out of the focusing FFT. Table 17 shows that doubling the amount of linear range migration does not affect the performance for the FFT convolver algorithms.

The reason that the subarray case in Table 16 with linear migration and no interpolator provides better performance than the quadratic only case is that the linear migration tends to offset the quadratic in look 2.

Table 16

Linear Range Migration (24 cells)

- COMPUTATION OF ISL/IML WITH LINEAR RANGE MIGRATION OF 24 CELLS ADDED TO QUADRATIC (7 CELLS)

	LINEAR MIGRATION	INTERPOLATOR POINTS 0	2	3	4
FFT CONVOLVER	NO	12.4	28.3	29.9	32.1
	YES	12.4	25.8	27.6	30.5
SUBARRAY	NO	12.5	-	24.6	25.5
	YES	13.6	19.1	24.4	25.4

DATA TAKEN FOR LOOK 2

Table 17

Linear Range Migration (40 cells)

- ISL/IML WITH A LINEAR RANGE MIGRATION OF 40 CELLS ADDED TO QUADRATIC (7 CELLS)

- FFT/CONVOLVER APPROACH

INTERPOLATION POINTS	LOOK	2D ISL/IML (PSL)
0	1	-12.4 (19.7)
	2	-12.4 (19.7)
	3	-12.4 (19.8)
	4	-12.4 (19.7)
2	1	-25.6 (30.7)
	2	-25.8 (31.3)
	3	-25.8 (31.4)
	4	-24.8 (29.5)
4	1	-29.9 (37.7)
	2	-30.5 (38.5)
	3	-30.5 (37.3)
	4	-28.2 (35.1)

2.4 Technology Survey

Technology is an important consideration in the design of the ADSP. The three principal areas designated; digital integrated circuits, digital architecture and software encompass the major design issues related to the system. This section summarizes the study,evaluation and projection of these technology issues for the ADSP and future SAR systems.

2.4.1 Digital Integrated Circuits

2.4.1.1 Survey of Integrated Circuit (IC) Manufacturers

Keys to the determination of IC technology available for the ADSP are the current and future plans of semiconductor manufacturers. An industry survey was conducted to determine plans and projections in the 1983-85 and post-1985 time frame. The companies included were:

Advanced Micro Devices	Raytheon Semiconductor
Fairchild Semiconductor	RCA Solid State
American Microsystems	Signetics
Intel	Texas Instruments
Intersil	Harris Semiconductor
Motorola	Synertek
Mostek	Monolithic Memories
National Semiconductor	

Information relative to technologies, memory size and configuration, LSI-VLSI logic functions, speed, power and cost were requested for the 1983-85 time frame. For post 1985 projections on technologies, line widths, gate density, speed, and power were requested with specific projections on memories, arithmetic functions and microprocessors.

Several of the companies surveyed declined to respond due to company policies on the release of long range plans. A summary of the results is contained in Table 18-20 and includes current technology, firm plans -1983-1985, projections 1983-85, and post 1985 projections. Random access memories, EPROMS/PROMS/ROMS, and IC logic functions are included.

2.4.1.2 IC Technology Trends

The IC technologies in current use span a large gamut of semiconductors. These include the high speed emitter coupled logic (ECL) line to the low power, relatively low speed complementary metal oxide semiconductors (CMOS) and integrated injection logic (I^2L). Bipolar transistor-transistor logic (TTL) with Schottky and low power Schottky (STTL, LSTTL) versions are the dominant semiconductor technologies now used in digital logic. N-channel MOS (NMOS) is the dominant large memory technology and silicon on sapphire (SOS) is used primarily for specialized military and space applications.

As digital logic moves to shorter channel lengths we can expect changes in the technologies used. Although we can expect the same technologies to be in general use through 1985 we will see a definite shift toward narrow channel CMOS replacing TTL and ECL because of its low power, with high speed, capability.

In the post 1985 time frame the principal technologies will be NMOS and CMOS with CMOS/SOS possibly moving into some general commercial application.

There is general agreement that CMOS will have emerged as the prime digital semiconductor technology because of its inherent low power dissipation. As line dimensions of CMOS circuits decrease, the speed also increases. CMOS circuits have been fabricated with 2 micron dimensions giving average gate delays of 1 nsec. When projecting up to 50,000 or more gates per chip, the power dissipation per gate must obviously be low to prevent thermal breakdown. At line dimensions of 1 to 2 microns and less the speed and power advantages of CMOS/SOS over CMOS are diminished, particularly when using oxide-isolated CMOS technology.

Although CMOS has been indicated to be a post 1985 technology, it could happen much faster than that. There is a growing realization of the speed-power advantage of advanced CMOS in the industry-and increased investment. When the cost of the CMOS circuits meets the NMOS technology costs, it will definitely be the choice because of its lower power which simplifies the power supply design, cooling

requirements and packaging system.

The impact of the post 1985 technology on SAR processing will be in the feasibility of on-board processing provided by low power, dense memories, and VLSI with low power per gate provided by CMOS.

Gallium arsenide (GaAs) can also be expected to emerge as the high speed technology in the post 1985 time frame. Research efforts with GaAs have produced GHz functional logic elements and A/D converters.

2.4.1.3 Random Access Memories

Random access memories have been broken down into dynamic and static categories. It can be noted that for the ADSP, static RAM's are preferable since the synchronous nature of the input and output data is more easily controlled with static RAM storage. In general, dynamic RAMs are less costly on a per-bit basis. Current static RAM's are much faster, consume more power and, in accordance with general trends, cost more per-bit for the higher speed capability.

Table 18
Random Access Memories

		CURRENT TECHNOLOGY	1983-1985 FIRM	1983-1985 PROJECTIONS	POST 1985 PROJECTIONS
DYNAMIC	SIZE	TO 64K	64K	128K-512K	64K-1M
	CONFIGURATION	x1	x8, x4, x1	x8, x16	-
	ACCESS (nsec)	100-300	200	50-200	50-100
	POWER (mW) (ACTIVE/STANDBY)	300/20	300/20	-	-
	COST/BIT (¢)	0.1	0.04-0.1	0.02-0.1	-
STATIC	SIZE	TO 16K	16K-64K	64K-256K	64K-1M
	CONFIGURATION	x1, x4, x8	x8, x4, x1	x8, x16	-
	ACCESS (nsec)	50-100	25-200	50-200	50-100
	POWER	600/100	200-1,000	-	-
	COST/BIT (¢)	0.1-0.4	0.1-0.3	0.02-0.1	-

The 1983-85 projections generally continue to follow the "Moore curve" of memory development first observed by Gordon Moore of Intel Corporation. It shows memory capacity per chip doubling every year. Perhaps more significantly; the cost per bit is projected to drop significantly, at least by some companies, in this time frame. In addition, the memory size firm committments of manufacturers is probably conservative. Figure 42 shows the state of the art for developmental memory circuits which is currently (mid 1981) 256 K bits for bulk CMOS. There is about a two year delay between development and commercial introduction. Based on the Figure 42 curve, we should see megabit dynamic memories on the market in 1985.

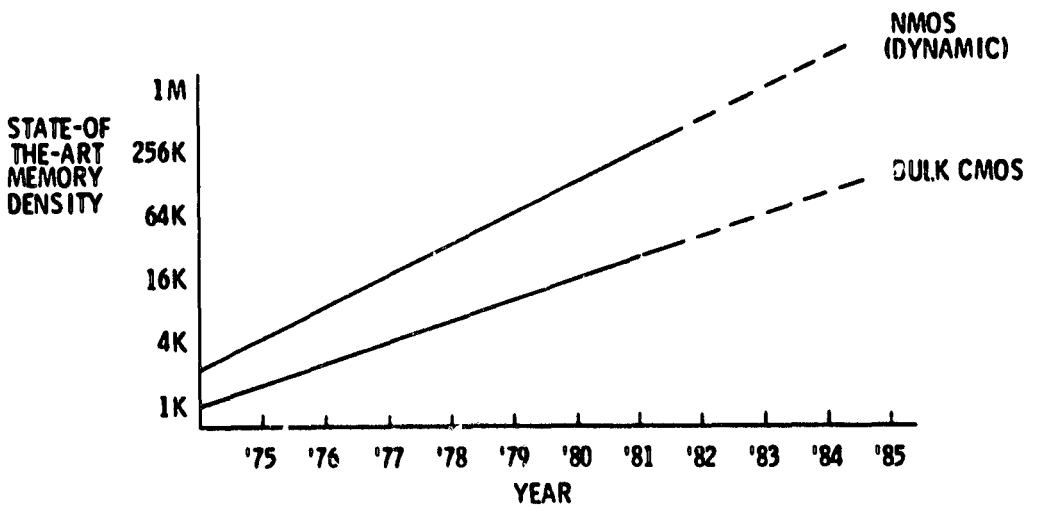


Figure 42
Memory Chip Capacity Trends

Memory cost trends are shown in Figure 43. Both chip costs and installed memory costs are included. Chip costs should be close to the 0.01 cents per bit level in 1985. This translates to about $50,000 for the chip costs of the corner turning memory of the ADSP processor in the 1985 time frame.

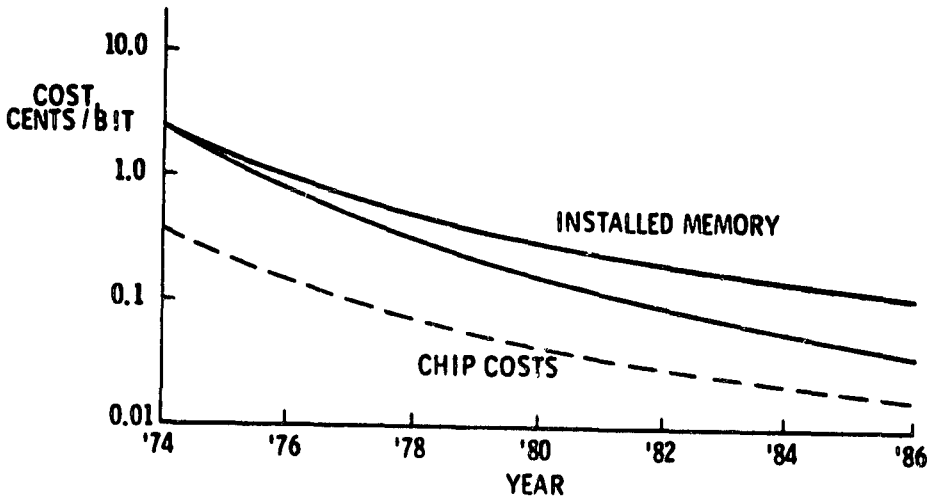


Figure 43
Memory Costs/Bit Trends

2.4.1.4 EPROMS, PROMS, ROMS

An essential part of the ADSP processor are the programmable memories used in the control firmware of the processor. Projection of circuit capabilities for erasable programmable read only memories (EPROMS), programmable read only memories (PROMS) and mask programmable read only memories (ROMS) are given in Table 19.

Table 19

EPROMS, PROMS, ROMS

		CURRENT TECHNOLOGY	1983-1985 FIRM PLANS	1983-1985 PROJECTIONS	POST 1985 PROJECTIONS
EPROMS	CONFIGURATION	2k, 4K, 8M WORDS BY 2, 4, 8 BITS/ WORD	32K, 64K		TO 1M
	ACCESS (nsec)	350-600	200		
	COST/BIT (¢)	0.2-0.4	0.1-0.3		
PROMS	CONFIGURATION	1 K, 2K WORDS BY 1, 4, 8 BITS PER WORD	TO 8K x 8		TO 1M
	ACCESS (nsec)	45-100	50		
	COST/BIT (¢)	0.3	0.1-0.3		
ROMS	SIZE	TO 64K	64K-128K	512K	TO 1M

EPROMS are fabricated with MOS technology and although there are a wide variety of configurations available, they are generally too slow for convenient application to the ADSP. PROMs on the other hand, are made with bipolar technology and are much higher in speed although not as large. However, the maximum size PROM, 16K in up to a 2K X 8 configuration is applicable to an FFT signal processor.

A mask programmable read-only memory chip (ROM) may be applicable to the ADSP. The size and speed of ROMs of up to 64K are applicable, and if the set-up charges are sufficiently low to be offset with the quantities required in the ADSP it is the preferred direction.

2.4.1.5 IC Logic Functions

IC logic function projections are summarized in Table 20. Particularly noteworthy to the ADSP (and all other signal processing application) is the expected emergence of various "FFT chips" in the 1980's. These circuits will make the principal computation requirement in the ADSP much less costly to meet.

Table 20

IC Logic Functions

CURRENT TECHNOLOGY	1983-1985 FIRM PLANS	1983-1985 PROJECTIONS	POST 1985 PROJECTIONS
ARITHMETIC	• 8 TO 16 BIT MULTIPLIERS	• 16 TO 32 BIT MICROPROCESSORS	• 32 BIT MULTIPLIERS AND MICRO-PROCESSOR, 20 MHz
MULTIPLIERS 8 x 8, 50 nsec 16 x 16, 100 nsec	• FFT CHIP	• 10,000 GATES / CHIP	• FFT PROCESSOR ON CHIP
ALUs (1981) 16 BIT FLOATING POINT AND PROGRAM-MABLE ALUs	• 1,000-10,000 GATES / CHIP		• 50,000 GATES / CHIP
OTHER	• 5-20 MHz CLOCK CLOCKS		
• 16 BIT ERROR DETECTION AND CORRECTION			
• 8 BIT COMPARATOR			
• PROGRAMMABLE LOGIC ARRAYS			

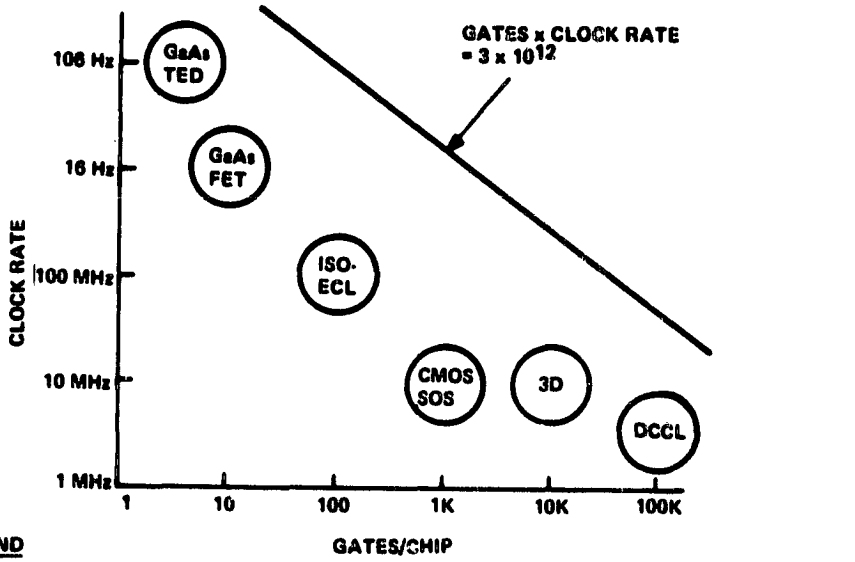
Most of the new logic circuit development in recent years has been directed toward microprocessor development. High speed LSI arithmetic circuits applicable to signal processors have not been developed to the extent possible. TRW has marketed a number of LSI arithmetic components which meet signal processing needs. Chief among these are a line of parallel multipliers of up to 24 bits. The 16 X 16 bit multiplier has a speed of about 100 nsec. TRW is also planning to release a floating point adder circuit by early 1982 which could be a key component in the ADSP.

Another circuit of special interest to the ADSP is an error detection-correction circuit. Because of the large memory in the ADSP, it may be desirable to employ error detection and correction to increase reliability.

2.4.1.6 VHSIC

Perhaps the most significant development effort now underway for the future of signal processing technology is the Department of Defense (DOD) VHSIC program - for very high speed integrated circuits.

The product of the number of gates on an IC and the clock frequency is a figure of merit for integrated digital logic that is particularly useful in assessing computing or signal processing power. The long term goals of the DoD VHSIC program is to achieve a gates-times-clock rate of 3 $x10^{12}$. This goal is compared to a number of advanced technologies in Figure 44a. All of the technologies shown are or were candidates for achieving the VHSIC goals.



LEGEND

GaAs TED – GALLIUM ARSENIDE TRANSFERRED ELECTRON DEVICE
GaAs FET – GALLIUM ARSENIDE FIELD EFFECT TRANSISTOR
ISO – ECL – DIELECTRICALLY ISOLATED EMITTER · COUPLED LOGIC
CMOS/SOS – COMPLEMENTARY METAL OXIDE SEMICONDUCTOR/SILICON ON SAPPHIRE
3D – TRIPLE DIFFUSED BIPOLAR
DCCL – DIGITAL CHARGE COUPLED LOGIC

Figure 44 a

Comparison of New Technologies on
Clock Rate - Gate Basis

The VHSIC program objectives include a shrinkage of the circuit geometric dimensions from 5 microns to 1.5 microns over 3 years and to 0.5 microns over 6 years. The scaling theory for circuit performance indicates that this size shrinkage will provide lower power-delay products. For example, in CMOS circuits, the basic geometric performance parameter is the transistor channel length,

L_c. With power dissipation proportional to L_c^2 and the delay to $1/L_c$, the power-delay product is thus proportional to L_c.

The projected VHSIC circuit capabilities will permit chip implementation of advanced signal processing functions on a chip. All contractors selected for the VHSIC Phase I programs have objectives which are oriented toward signal processing.

2.4.2 Digital Architecture

An overview of digital architecture approaches was developed in relation to the functional characteristics of the ADSP. These functional characteristics are summarized in Table 21 for the range and azimuth processing functions.

Table 21

ADSP Functional Characteristics

	RANGE PROCESSOR	AZIMUTH PROCESSOR
NATURAL PARTITIONING	• RADAR PRI • PULSE LENGTH	• RANGE CELL • LOOKS • STORAGE RANGE MIGRATION SPAN
PROGRAMMABILITY	• BANDWIDTH • TW PRODUCT • WAVEFORM • WEIGHTING • RANGE SWATH	• PRI • AZIMUTH COMPRESSION • RANGE MIGRATION • CLUTTER LOCK • FOCUS
BULK STORAGE	• NONE REQUIRED	α RANGE SWATH x AZIMUTH x COMPRESSION x LOOKS x OVERLAP FACTOR
MAJOR FUNCTIONS	• FFT • COMPLEX MULTIPLY • AMPLITUDE WEIGHTING • INTERPOLATION	• RANGE PROCESSOR FUNCTIONS PLUS: • CORNER TURNING • RANGE MIGRATION COMPENSATION • DECIMATION • MAGNITUDE, SCALING • INTEGRATION • DOPPLER (CLUTTER) ESTIMATE • AUTO-FOCUS

The first functional feature, natural partitioning, refers to partitioning approaches based on the synthetic aperture radar processing algorithms or the inherent radar operation. For example, a natural partition in the range processor is the radar PRI (pulse repetition

interval) during which a single radar pulse is transmitted and its reflected signals from the range swath coverage are received.

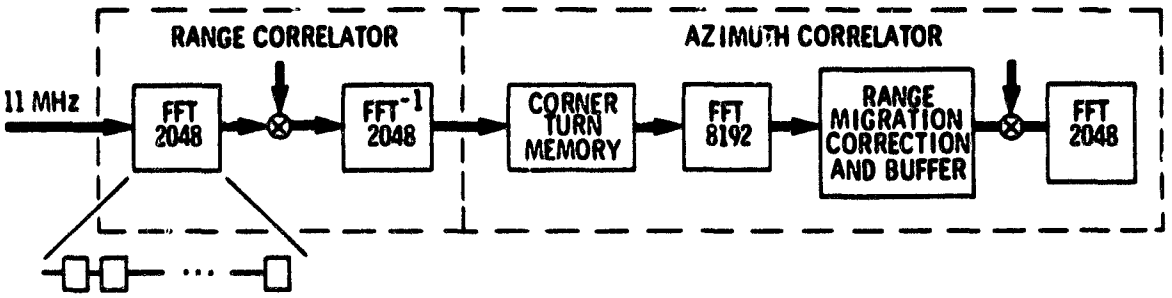
Programmability must be incorporated for the range and azimuth processes as indicated in Table 21. A major requirement is bulk storage which occurs primarily in the azimuth processor. The size of the bulk store, required for range-azimuth corner turning, is determined by the product of the range cells and pulses coherently integrated with the number of looks and signal overlap also involved. It is also dependent upon the algorithm employed.

The major functions listed in Table 21 constitute the well defined functions for range and azimuth SAR processing. A programmable processor capable of performing all of these functions will by its nature have functional capabilities not listed. Modifications in processing algorithms should therefore be possible.

Several basic architecture approaches were investigated for the ADSP: pipeline, parallel, SIMD (single instruction stream-multiple data stream), cross-bar, and various multi-bus architectures.

2.4.2.1 Pipeline Processor

A pipeline processor using the FFT convolver processing algorithm is shown in Figure 44b. It has the primary feature of minimizing hardware both memory and arithmetic processing elements. All elements of the pipeline operate simultaneously and the control is fairly straight forward. It can be programmed for changes in the waveforms or focusing function via the spectral reference functions and the FFT size. The arithmetic and memory can be scaled to match the word size as the signals progress through the pipeline. However, this approach is not usually followed since net cost reductions are generally realized if fewer design variations are used. Techniques for reliability enhancement include providing for interchangeable pipeline segments and the incorporation of extra FFT stages in the pipeline which can be switched into use when a failure occurs at any particular stage. Growth to higher capacity systems must be done by the addition of complete, parallel pipeline systems.



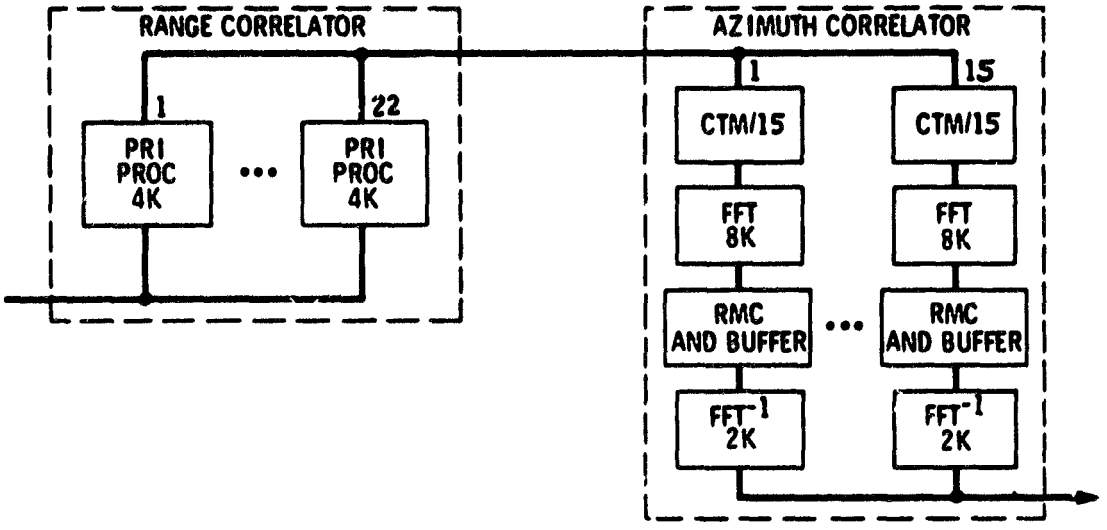
- MEMORY: CTM + RMC + 14K
- FFT STAGES: 46
- PROGRAMMABILITY: COMPRESSION RATES, PIPELINE ADJUSTMENTS
- RELIABILITY ENHANCEMENT: SWITCHED FFTs OR FFT STAGES
- MODULARITY: PARALLEL SYSTEMS

Figure 44b
Example ADSP Pipeline Processor (FFT Convolver)

2.4.2.2 Parallel Processor

An approach to implementing the ADSP with parallel processors is shown in Figure 45. In this case, a total of 52 programmable FFT processing units are used to compute the required FFT computations. In the range correlator, each unit is capable of processing a complete single pulse range scan. Successive pulses are inputted to successive processors and the total number is set to operate on the total processing load at an internal clock rate of 8 MHz. The azimuth correlator is different in that each parallel unit handles a fixed segment of range cells. This permits the bulk corner turning memory to be divided equally and allocated to the respective parallel azimuth processors. An FFT convolver function is inherent in the azimuth correlator as in the range correlator, but because of the spectral domain range migration compensation, the forward and inverse FFT's are separated. The range migration compensation unit is then a special processor in between the forward and inverse FFT. The advantage of the parallel processor is in the standardization of

hardware and high reliability. The current thrust of LSI technology permits implementation of a FFT processor at reasonable cost levels, which when constructed in quantity will further reduce the unit cost. Spare units can be employed to enhance reliability.



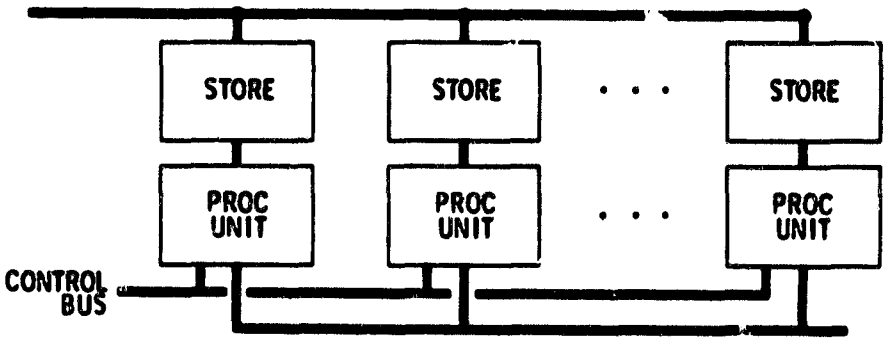
- MEMORY: CTM + 15RMC + 238K
- PROCESSOR UNITS: 52
- PROGRAMMABILITY: AT PROCESSING UNIT LEVEL
- RELIABILITY ENHANCEMENT: SPARE PARALLEL PROCESSORS
- MODULARITY: ADDITIONAL PROCESSORS

Figure 45

Example ADSP Parallel Processor (FFT Convolver)

2.4.2.3 Single Instruction - Multiple Data (SIMD)

The SIMD approach shown in Figure 46 is another parallel processor approach with an important difference. All of the processors operate in-step on the entire processing algorithm. This approach offers a common control approach, but it is not good for the ADSP. Since each unit must do the entire process, it must have a full corner turning memory or at least a significant portion of the range swath. Thus, the total memory requirements of the system are unreasonable.



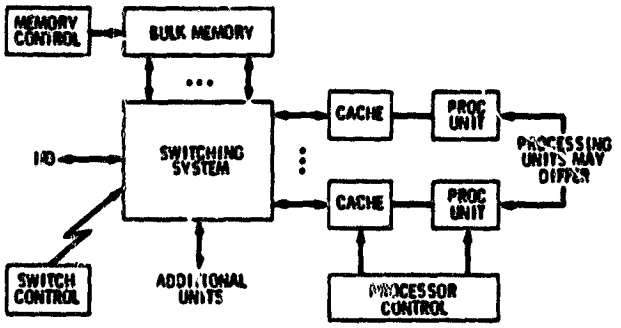
- MEMORY: ~25 CTM
- PROCESSING UNITS: ~50
- PROGRAMMABILITY: COMMON CONTROL
- RELIABILITY ENHANCEMENT: PARALLEL UNITS
- MODULARITY: ADDITIONAL PROCESSORS

Figure 46

SIMD (Single Instruction - Multiple Data Stream)

2.4.2.4 Cross-bar

Cross-bar architecture has long been used in high performance digital processors. This approach, shown in Figure 47, can be used to form a programmable pipeline signal processor. It has particular advantages in cases where several different processing units may be used in a varying sequence. The need for a variation of processors would arise in a signal processing application where computationally intensive algorithms must be performed. It may then be cost effective to construct special purpose processors for these functions. The number and types of processors might vary for each installation. A technical problem with the cross-bar approach is the design of the cross-bar switch itself. Several methods have been suggested in the literature for constructing modified variations of the cross-bar. These approaches are aimed at matching the real switching needs with an efficient switching mechanism, generally a multi-level switch matrix not unlike an FFT flow diagram. However, all of the proposed switching schemes do not provide the flexibility of a cross-bar and thus limit the overall efficiency of a general purpose processor.



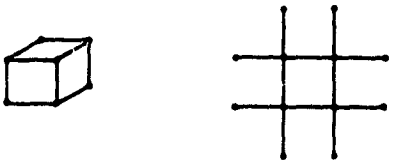
- MEMORY: K • CTM
- PROCESSING UNITS: MIX OF PARALLEL AND PIPELINE SUBSYSTEMS
- PROGRAMMABILITY: CONTROL AND SEQUENCE OF PROCESSING UNITS
- RELIABILITY ENHANCEMENT: SPARE PROCESSING UNITS
- MODULARITY: PARALLEL SYSTEMS

Figure 47
Cross-bar Architecture

2.4.2.5 Multi-bus Architecture

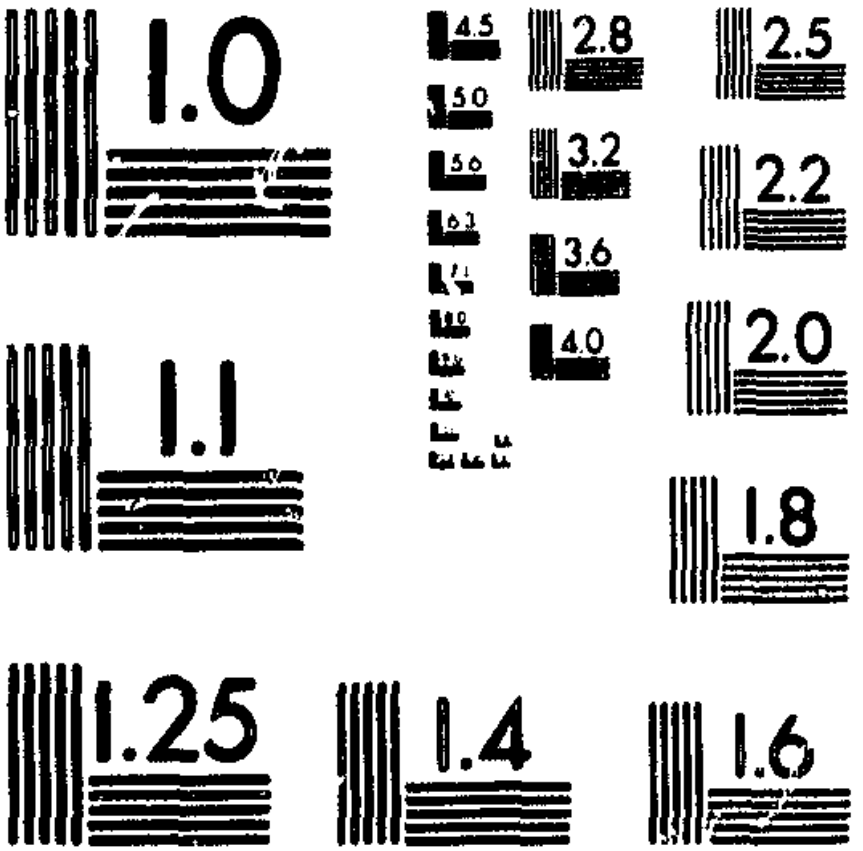
A multi-bus architecture can be used and Figure 48 does not do justice to the many variations which are possible. Recent efforts in this area have been concentrated on heirarchical schemes which partition the hardware structure into various levels of operational control. The control problems of this approach seem to be overly complicated for a signal processing system as essentially well defined as the ADSP.

STRUCTURE -- VARIABLE -- 2, 3 DIMENSIONS



- MEMORY: > MINIMUM STORAGE
- PROCESSING UNITS: > 50
- PROGRAMMABILITY: SEQUENCE, SIGNAL PATH CONTROL
- RELIABILITY ENHANCEMENT: ADAPTIVE FUNCTIONAL PATHS
- MODULARITY: ADDITIONAL PROCESSORS

Figure 48
Multi-bus Architecture



2.4.2.6 Programmable Signal Processor Developments

Several government-industry efforts to build high performance, programmable signal processors are underway. These developments are not near the mature stage where one can buy a system. A brief summary of these developments follows.

Advanced On-Board Signal Processor (AOSP)

AGENCY: DARPA
CONTRACTOR: RAYTHEON
OBJECTIVE: DEVELOP SIGNAL PROCESSING SYSTEM FOR FUTURE SPACE RADAR, ELECTRO-OPTIC AND COMMUNICATION SYSTEMS CAPABLE OF MEETING ON-BOARD SIZE, WEIGHT AND RELIABILITY REQUIREMENTS
ARCHITECTURE: MULTIPLE BUSSES INTERCONNECTING ARRAY COMPUTING ELEMENTS (ACEs)
CLOCK RATE: 10-40 MHz
MEMORY/ACE: > 10M BITS
GATES/ACE: > 100K
ACES/ADSP: 56
STATUS: STUDY/SIZING/SIMULATION/SOME BREADBOARDS UNDERWAY

S-1 Super Computer

DESIGN: LAWRENCE LIVERMORE LABORATORY
SUPPORT: OFFICE OF NAVAL RESEARCH
OBJECTIVE: TO DESIGN AN ADVANCED VECTOR PROCESSING
ARCHITECTURE: CROSS-BAR CONNECTING MEMORIES WITH 16 HIGH SPEED PROCESSING UNITS (A-BOXES)
CLOCK RATE: 57 MHz
COMPUTATION RATE: 430 μsec/4096 POINT FFT PER A-BOX
A-BOXES/ADSP: 10
STATUS: UNDER DEVELOPMENT

Enhanced Modular Signal Processor (EMSP)

AGENCY: NAVAL SHIP SYSTEMS COMMAND
OBJECTIVE: TO DEVELOP AN ADVANCED PROGRAMMABLE SIGNAL PROCESSOR FOR NAVY APPLICATIONS TO REPLACE THE AN/UYS-1
ARCHITECTURE: MODIFIED CROSS-BAR

C-2

COMPUTATION RATE: 80 M (REAL MULTIPLES)/sec
EMSPs/ADSP: ~25
STATUS: STUDY PHASE

Other processors such as the MPP - Massively Parallel Processor (NASA-Goodyear) and MRP - Multi-mode Radar Processor (Hughes) are in various stages of development. The question that must be answered is whether any of these developments are practically applicable to the ADSP.

The specialized needs of the ADSP can be best met with a processor matched to its requirements. This will also limit long term life cycle costs particularly in the area of mission software development.

2.4.3 Software

A study was completed of high order languages (HOLs) suitable for the executive and control software tasks within the ADSP host computer. The languages studied were Fortran, Pascal, SPL/I and Ada. Any one of the candidate languages is adequate for the task, but some present advantages over the others. It is desirable for documentation purposes and ease of understanding to have as much of the control software as possible written in the HOL, as opposed to assembly language subroutines. This gives languages which have real time control features an edge, since a language without them must resort to assembly language for real time control. Another key feature for readability is the degree to which a language permits and encourages structured code. The newer languages, especially Pascal and Ada, are designed to force the programmer into structured programs. Manipulation of large data arrays is important and all of the candidate languages have this capability. Finally, the availability of the language is crucial. Is it supported on the selected ADSP computer? Is it sufficiently mature to be fully documented and readily usable? Is it in widespread use? These are the key questions.

Table 22 summarizes the key characteristics of each language. Ada most closely meets the needs of the ADSP. This conclusion was reached after a careful comparison of the candidate HOLs.

The HOL which has been proven in almost every application is Fortran. In use for twenty years, Fortran is a suitable substitute for almost any language. The newest versions of Fortran include provisions for structured programming, including control constructs such as IF...THEN...ELSE. Most versions have fairly sophisticated I/O capabilities, although most are unique to one machine and not readily transportable to another. Most Fortran versions do not have any real time control structures, leaving these functions to the operating system. Fortran can handle arrays of data, although complex file structures can be difficult to handle. Fortran is less readable than the newer languages, although its widespread use may offset this, in that so many programmers readily understand it. In short, Fortran may be an acceptable vehicle for the task, but its limitations were precisely what the newer languages were designed to improve upon. It may pay to take advantage of these improvements.

SPL/I, or Signal Processing Language I, is a relatively new language developed by Intermetrics, Inc. for Navy programmable signal processing applications. As a language designed expressly for signal processing, it is an immediate candidate for ADSP. It contains many features of use in this application, especially in the number and power of its real time control constructs. These can effectively handle multiple parallel processes and the interfaces between them. SPL/I shares the structured constructs and data typing of Pascal, Ada, and others. Detailed study of the language shows strong structural similarities to other HOLs, including many of their weaknesses. SPL/I, like Pascal and Ada, makes an effort to minimize dependence on a particular machine or compiler to guarantee software transportability. As such, I/O capabilities are essentially left to the implementation instead of being defined within the language. SPL/I has only four simple library functions defined to handle I/O. This is less than the other candidate HOLs.

Another major problem with SPL-I is a lack of available implementations. Versions exist for Navy AN/UYK-7 and AN/UYK-20 computers, but outside the Navy, it is virtually unused. Much of this lack of use is due to the

expectations in the development of Ada, which has most of the same real time control features, and at the same time provides for more readable code.

Pascal, a language invented in 1970 by Niklaus Wirth of the University of Zurich, is intended primarily as a teaching language. The original intent was to provide the necessary language constructs to force a programmer to write structured programs. Many of the concepts defined by Pascal have been used in later languages, especially Ada and SPL/I. Pascal is a highly readable, easy to understand language. It is a strongly typed language, which means that all variables and constants in a program must be explicitly declared as to its type at the beginning of a program. Translation between different types is difficult unless their relationship is pre-defined. For example, this feature prevents a programmer from accidently equating feet and pounds in a calculation. Strong typing requires a programmer to think through the process he is defining before he actually codes it. Pascal incorporates many program control features which guarantee structured program flow: IF...THEN...ELSE and CASE statements provide explicit conditions for branching, making the reasons for each branch clear in each case. Pascal has array handling features which provide easy set up and manipulation of complex data structures such as linked lists.

Because Pascal was originally intended as a teaching language, it has a number of shortcomings when applied to a sophisticated real time application. The I/O defined for Pascal in its original form is severely deficient, amd numerous implementations have attempted to improve upon its I/O capabilities. This has seriously inhibited Pascal's ability to be implementation dependent, and hence transportable from one compiler to another, and from one machine to another. It also makes it difficult to compare Pascal with other languages, since some versions of Pascal may have adequate I/O for the intended application.

Another serious deficiency of Pascal is the absence of any real-time control structures. Pascal is not intended for real time applications. To be readily transportable from ore system to another, many Pascal

compilers, most notably the UCSD version, translate to a universal intermediate level language (termed a p-code), which in turn is readily translated to the machine language of host computer. This two level translation process, while useful, limits the efficiency of a Pascal program in terms of the number of machine instructions, it takes to execute a single Pascal instruction. The two level process also makes the execution of a Pascal program very difficult to predict and/or minimize.

Despite these limitations, Pascal has gained widespread acceptance in the industry. Some companies, like Texas Instruments, have standardized all of their software development around the use of Pascal as their preferred HOL. There are many versions available, and any final computer system chosen is likely to have at least one commercially available compiler to use. In addition, an intermediate p-code language could be translated to any embedded processor's instruction set, providing HOL capabilities and documentation within the system hardware, provided real-time control and I/O problems can be dealt with.

Many of the limitations of Pascal may be overcome through the use of Ada, a derivative of Pascal first developed in 1979 by a design team from Honeywell-Bull, under the direction of Jean Ichbiah. Developed as a joint Army/Air Force project, Ada is intended to be the Department of Defense standard HOL for all major systems of the future. As such, Ada was designed to be an all-purpose language, incorporating the key features of Pascal together with real-time processing constructs and improved I/O capabilities. Like Pascal, Ada is a strongly typed language, with all variables defined clearly at the beginning of the program. Ada has all the same structured program constructs as Pascal, and in addition has features to simplify separate compilation of different sections of a large program. The I/O constructs in the original Ada definition are adequate, although still somewhat less sophisticated as most Fortran implementations. This will probably be improved when Ada is actually implemented. Ada is designed to be highly readable: the design intent was to simplify documentation requirements by using a language as close as possible

to a Program Design Language (PDL) which can be used for documentation instead of flow charts. Ada's multitasking capability provides clear definitions of multiple parallel processes, and their interfaces.

The primary problem with using Ada is that currently its implementation is in its infancy. The most notable development is the Intel VLSI LAPX-432 microcomputer which is designed for the Ada HOL. The initial definition of the language has generated considerable interest within the software industry. A number of projects to develop working compilers has begun, both inside and outside the defense industry. RCA has identified 24 such projects for various machines. It is expected that by 1983 most major computer systems will have usable versions of Ada, and that by 1985 Ada will be in widespread use. As such, it appears to be most promising candidate for the ADSP. Ada combines the best features of the other languages into a single language: the versatility of Fortran, the real-time control of SPL/I, and the readability of Pascal. Table 22 summarizes the key characteristics of each language.

Table 22
Key Characteristics of Candidate HOLs

FEATURE \ LANGUAGE	SPL-I	FORTRAN	PASCAL	ADA
READABILITY	FAIR	FAIR	EXCELLENT	GOOD
REAL TIME CONTROL STRUCTURES	YES	IMPLEMENTATION DEPENDENT	NO	YES
ARRAY DATA HANDLING STRUCTURES	GOOD	FAIR	GOOD	GOOD
STRUCTURED PROGRAMMING FEATURES	YES	IMPLEMENTATION DEPENDENT (FORTRAN 77)	YES	YES
DATA TYPING	YES	NO	YES	YES
I/O HANDLING	POOR	EXCELLENT	POOR	FAIR
CURRENT USE	AN/UYK-7, AN/UYK-20 (NAVY) COMPUTERS ONLY	WIDESPREAD	WIDESPREAD	FIRST COMPILERS IN DEVELOPMENT
FUTURE USE (1983 -)	UNPREDICTABLE	WIDESPREAD	WIDESPREAD (ALTHOUGH MAY BE REPLACED BY ADA)	WIDESPREAD

2.5 ADSP SELECTED DESIGN

2.5.1 Selection of Algorithm

The FFT convolver was selected for the ADSP. The selection of a processing algorithm was made principally between the time domain, FFT convolver and subarray algorithms. The other approaches were considered in relation to these three.

2.5.1.1 Key ADSP Requirements

The key requirements for the ADSP are summarized in Table 23. Continuous variation in processing parameters, with provision for modular construction and future growth are driving factors. In addition, the processor must accommodate continuous or burst modes.

Table 23
Key ADSP Requirements

		BASELINE	VARIATION
• RANGE CORRELATOR:	COMPLEX SAMPLE RATE	11 MHz	
	PULSE SAMPLES	475	
	PULSE COMPRESSION	410	10 - 660
	RANGE SWATH	4000	500 - 4000*
	INTEGRATED RANGE SIDELOBES	-15dB	
• AZIMUTH CORRELATOR:	PRF	2500 Hz	
	LOOKS	4	*
	AZIMUTH COMPRESSION RATIO/LOOK	350	20 - 350*
	INTEGRATED AZIMUTH SIDELOBES	-20dB	
	RANGE MIGRATION	86 RANGE BINS (4 LOOKS)	

* TRADE-OFF AMONG RANGE SWATH, NUMBER OF LOOKS, AND RESOLUTION

- MODULARITY -- PROVISION FOR:
  - SWATH WIDTH -- UP TO FOUR TIMES BASELINE
  - MORE PARALLEL LOOKS
  - MULTIPLE FREQUENCIES AND POLARIZATIONS

2.5.1.2 Programmability

The variation in SAR parameters can be accommodated by the three major processing algorithms shown in Table 24. Both the time domain

and FFT convolver approaches offer full control over the SAR processing parameters such as compression factor. However, the subarray implementation cannot be readily programmed to cover a continuous selection of parameters. The subarray formation process most easily accommodates a stepped parameter control. A disadvantage accrues to the time domain approach with variations in the number of looks. A variable prefilter would be difficult to implement in this case. The Type B time domain processor is used in the comparison because it is preferred over the Type A processor.

Table 24

Algorithm Programmability

- ALL APPROACHES CAN ACCOMMODATE VARIATIONS IN RANGE SWATH THROUGH MEMORY CONTROL AND SCALING

ALGORITHM	REQUIREMENT VARIATION	METHOD	IMPACT ON HARDWARE
TIME DOMAIN (TYPE B)	COMPRESSION	VARY NUMBER OF ACTIVE CORRELATORS	MODEST INCREASE IN CONTROL FUNCTION
	LOOKS	• SWITCH CORRELATORS • VARY PREFILTERING	LARGE INCREASE IN CONTROL AND SWITCHING HARDWARE
FFT CONVOLVER	COMPRESSION	CONTROL COMMANDS AND REFERENCE FUNCTION	MODEST HARDWARE INCREASE AND CONTROL FUNCTION SOFTWARE
	LOOKS	REPROGRAM CONTROL SEQUENCE	MINIMUM HARDWARE INCREASE
SUBARRAY	COMPRESSION	VERY DIFFICULT TO OBTAIN CONTINUOUS CONTROL; CONTROL IN STEPS	LARGE INCREASE IN COMPLEXITY OF CONTROL SYSTEM
	LOOKS	REPROGRAM CONTROL SEQUENCE	

2.5.1.3 Incremental Implementation and Growth

Incremental growth or implementation can refer to variation in swath width, number of looks, or the number of processing channels. In addition, the capability of the processor to handle data at real time rates is a factor. All of these issues are in a sense related. As the swath width and number of channels increases, the processor memory and computation can be expected to increase linearly. As the number of looks increases the computation requirements generally decrease and drop dramatically for the time domain processor with a prefilter.

The algorithm and architecture selection are related and in anticipation of the selection of a parallel architecture that implementation is assumed for the FFT convolver and subarray algorithm in Table 25.

Table 25

Incremental Growth and Implementation

	PARAMETER		
INCREMENTAL GROWTH			
ALGORITHM	SWATH WIDTH	PARALLEL LOOKS	NUMBER OF CHANNELS (FREQUENCY AND POLARIZATION)
TIME DOMAIN (TYPE B)	ADD PARALLEL SYSTEM (S)	AUD ELEMENTS TO REDUCE LOOKS	ADD PARALLEL SYSTEMS
FFT CONVOLVER (PARALLEL ARCHITECTURE)	ADD RANGE AND AZIMUTH PROCESSORS	REDUCING LOOKS INCREASES FFT SIZE AND MEMORY STORAGE	ADD PARALLEL SYSTEMS
AZIMUTH SUBARRAY (PARALLEL ARCHITECTURE)	ADD RANGE AND AZIMUTH PROCESSORS	IS MOST EFFECTIVE FOR LARGE COMPRESSION RATIOS (FEW LOOKS)	ADD PARALLEL SYSTEMS
INCREMENTAL IMPLEMENTATION			
TIME DOMAIN (TYPE B)	NOT PRACTICAL BECAUSE OF POOR MODULE MEMORY TRADE	• PROGRAMMABLE PREFILTER • INCREASES REAL TIME RATE	N/A
FFT CONVOLVER (PARALLEL ARCHITECTURE)	REDUCE NUMBER OF RANGE AND AZIMUTH PROCESSORS	MORE LOOKS REDUCE SIZE OF FFTs AND MEMORY	N/A
AZIMUTH SUBARRAY (PARALLEL ARCHITECTURE)	REDUCE NUMBER OF RANGE AND AZIMUTH PROCESSORS	LOSES EFFECTIVENESS AS NUMBER OF LOOKS BECOMES LARGE	N/A

2.5.1.4 Burst Multimode Processing

The issues in burst mode processing fall into two categories; 1) the ability of the processor to accommodate mode changes with different SAR parameters with no loss of imagery and 2) the ability to process special modes such as a large members of looks in a burst mode (See Figure 5). The first case is illustrated by Figure 49. A SAR processor will have an

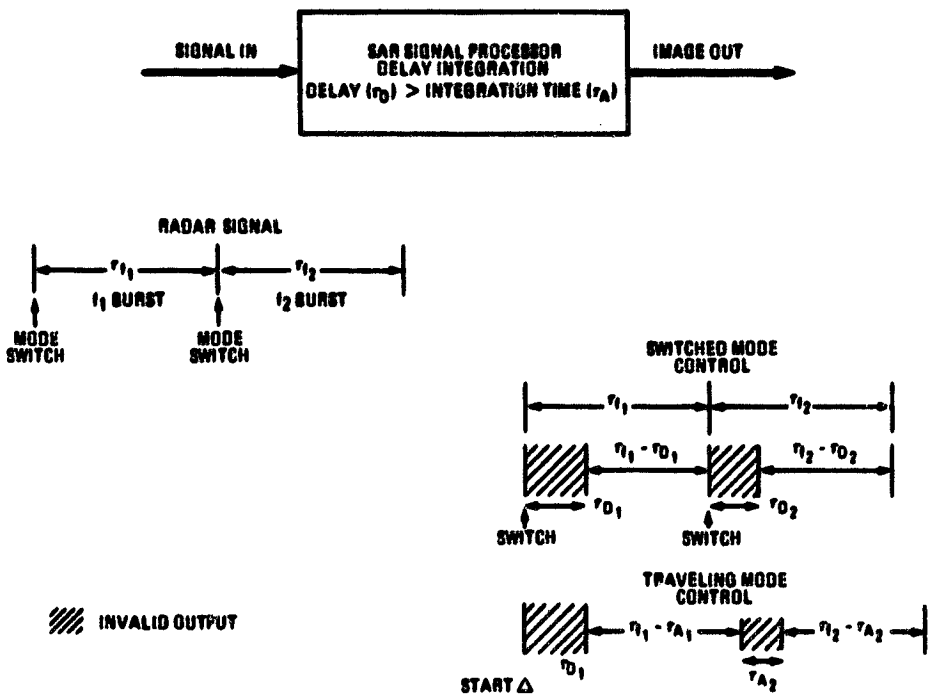


Figure 49
Burst Processing Delays

inherent delay (τ_D) from input to output which is greater than the integration time (τ_A). Radar bursts can switch instantaneously from one frequency to another as illustrated. A processor can have either a switched mode control or a traveling mode control. With a switched mode control, all of the parameters of the processor change together so the data within the processor when the mode swtich is changed is all lost. On the other hand, with a traveling control that moves with the data no additional invalid data appears beyond that equal to the integration time. The traveling mode control is obviously preferable and the time domain processor (Type B) is unique in its adaptability to a traveling mode control. The lost data can be avoided by increasing the bulk memory size by 50% for the FFT convolver and 100% for the subarray technique.

The second burst processing problem can be considered in the extreme case where the length of a burst is equal to the integration time and the bursts are of arbitrary spacing. In this situation there

can be essentially one image resolution element per look and the FFT convolver in its normal form is extremely inefficient. The time domain type B process is well matched to the case because of its mulitplier - accumulator structure. The type B processor can be used with each processing module handling a separate look for the full burst range swath. The FFT convolver and subarray systems can process this mode with some variation in their structure. The basic approach is to deramp the input signal, corner turn, FFT, correct for range all migration and integrate the appropriate multiple looks as indicated in Figure 50.

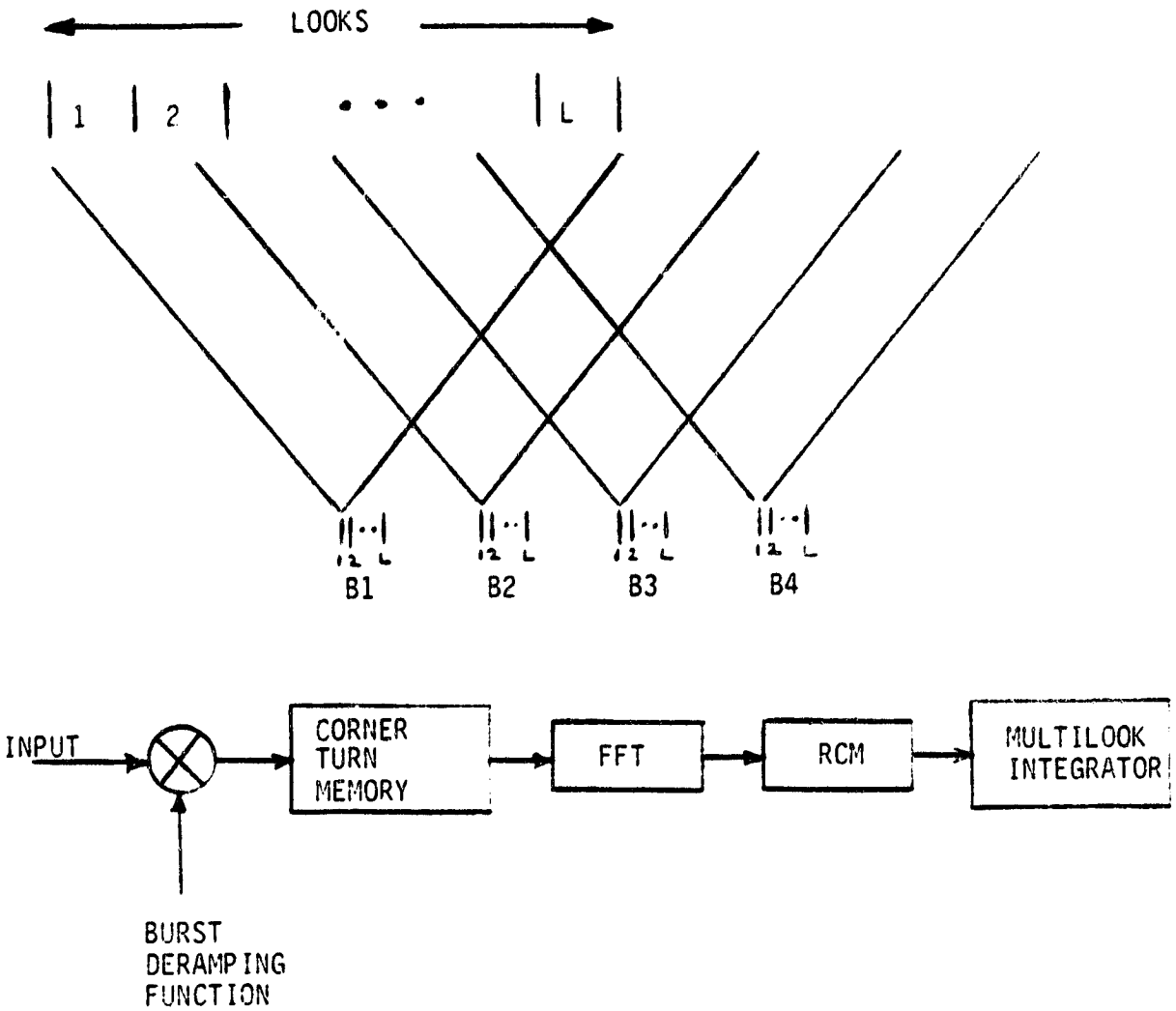


FIGURE 50. INTERRUPTED BURST PROCESSING WITH MANY LOOKS

2.5.1.5 Algorithm Cost Comparisons

Relative costs were developed for the three principal ADSP algorithms and are summarized in Table 26. The design costs of the subarray technique overshadow its lower materials costs. Materials for the time domain processor reflecting the large number of computations required dominate its cost profile.

Table 26
Relative Algorithm Costs

	TIME DOMAIN	FFT CONVOLVER	SUB-ARRAY
PROGRAM CONTROL, MONITORING AND DESIGN ASSURANCE	0.246	0.207	0.220
DESIGN	0.416	0.384	0.626
MATERIALS/SERVICES	1.214	0.263	0.174
ASSEMBLY AND TEST	0.419	0.146	0.191
TOTALS	2.295 (2.3)	1.000 (1.0)	1.211 (1.2)

2.5.1.6 Risks

Development risks associated with the candidate algorithms are indicated in Table 27. The time domains approach has problems with reliability and the control of a large hardware system at the nominal system clock rate of 11 MHz. The complexity of the subarray processor design is its major risk factor while the FFT convolver algorithm has no serious drawbacks. Its only questionable area is in achieving convenient modularity of the hardware elements - a concern that also applies to the other algorithms. General algorithm-independent risks are also listed in the table and the major one affecting the lifetime of the ADSP is meeting the correct mix between hardware and software controls.

Table 27

ADSP Algorithm Development Risks

	SPECIFIC	GENERAL APPLYING TO ALL ALGORITHMS
TIME DOMAIN	• CONTROL OF LARGE BUS SYSTEM, MULTIPLE CABINETS • 11 MHz CLOCK • RELIABILITY	TECHNICAL: • MEETING CORRECT MIX BETWEEN HARDWARE AND SOFTWARE
FFT CONVOLVER	• ACHIEVING MODULE PARTITIONING	SCHEDULE: • TIMELY AVAILABILITY OF PURCHASED ICs
SUBARRAY	• ACHIEVING REQUIRED PROGRAMMABILITY • DEVELOPMENT OF SIMPLE TEST PROCEDURE • CONTROL SYSTEM	COST: • UNPREDICTABLE INFLATIONARY TRENDS OVER LONG SCHEDULE • MEETING COST PROJECTIONS OF ICs IF PURCHASES MUST BE SPLIT IN TIME

2.5.1.7 Summary and Selection

A summary of the general characteristics of the various processing algorithms studied is given in Table 28. Particular algorithms have features which may be useful in certain applications. For example, the subarray process, which offers a minimal hardware implementation, is advantageous for a single dedicated processor for low power and size on-board applications. However, the FFT convolver algorithm was selected for the ADSP because:

- Programming for multiple modes is straightforward,
- the higher cost of memory and computations (relative to subarray) are offset by savings in control system design costs,
- a lower risk is associated with the approach,
- integrated circuit technology thrust is reducing the cost of memory and computations,
- it lends itself to modular growth,
- it provides a continuous selection of SAR parameters and,
- the ADSP has no requirements for small size, low power or special environmental conditions.

Table 28

Algorithm Summary Comparison

ALGORITHM	MAJOR FEATURE	DISADVANTAGES
FFT CONVOLUTION (1D)	EASIER TO PARTITION AND CONTROL	NOT MINIMUM HARDWARE
BLOCK FFT APPROACH	NO CORNER TURNING MEMORY	MORE MEMORY REQUIRED
2D FAST FOURIER TRANSFORM	MOST FLEXIBLE	HIGHER COMPUTATIONAL RATE DUE TO 2D OVERLAP LIMITED BY DEPTH OF FOCUS
2D FAST POLYNOMIAL TRANSFORM	LESS MULTIPLICATIONS	MORE DIFFICULT TO PARTITION
SERIAL TIME DOMAIN CORRELATION	VERY FLEXIBLE EASY REFERENCE GENERATION BEST FOR BURST MODE	REQUIRES MOST COMPUTATIONS
PARALLEL TIME DOMAIN CORRELATION	NO ADVANTAGE OVER SERIAL SYSTEM	REQUIRES MOST COMPUTATIONS REFERENCE FUNCTION DIFFICULT TO GENERATE
SUBARRAY PROCESSING	MINIMAL HARDWARE	MORE DIFFICULT TO PROGRAM

2.5.2 Selection of Architecture

A comparison of pipeline and parallel architecture is given in Table 29. The preponderance of key favorable factors make the parallel approach the optimum selection. A parallel structure is also characteristic of a number of specific processors including the Massively Parallel Processor - MPP (19), the Advanced On-Board Signal Processor - AOSP (20) and S-1 (21). Table 30 shows how these systems could be configured for the ADSP. Their use in the ADSP would depend upon the achievement of their development goals in a timely manner. In addition, they do not have the incremental implementation and growth characteristics desired. For those reasons we have recommended a parallel design tailored specifically to the ADSP requirements.

Table 29

Pipeline - Parallel Architecture Comparison

ADSP CHARACTERISTIC	PIPELINE	PARALLEL
NATURAL PARTITIONING	• EACH PIPELINE ELEMENT DOES OWN FUNCTION	• RANGE -- PRI • AZIMUTH -- RANGE SLICE
PROGRAMMABILITY	• FFTs -- LENGTH CHANGE • CONTROL -- TIMING MUST BE ADAPTABLE	• FIRMWARE PROGRAMMABLE (NO HARDWARE RESTRUCTORING)
MEMORY SYSTEM	• CAN TAILOR EACH ELEMENT TO MINIMUM SIZE • MUST OPERATE AT HIGH SPEED	• LOW SPEED OPERATION • MODULAR, CONVENIENT SIZES
RELIABILITY/ ENHANCEMENT	• MEDIUM RELIABILITY • EXTRA PIPELINE UNITS BUT CONTROL IS DIFFICULT • TESTING IS DIFFICULT	• INHERENT HIGH RELIABILITY • SPARES ARE INDEPENDENT AND EASILY INSERTED • STATUS TEST IS STRAIGHT-FORWARD
MODULARITY	• MODULARITY REDUCES EFFICIENCY	• MODULARITY INHERENT IN DESIGN
RISK	• MODERATE	• LOW

Table 30

Processor Comparisons

(FFT Convolver Algorithm -- FFTs)

	ADSP REQUIRED	MPP* PARALLEL	AOSP MULTIBUS (56 ACES -- est)	S-1 CROSSBAR (10 A-BOXES)
REAL MULTIPLICATIONS PER SECOND ÷ 10^6	2,240	910	2,240	2,240
REAL ADDS PER SECOND ÷ 10^6	3,360	4,428	3,360	3,360
APPROXIMATE MEMORY STORAGE ÷ 10^6 (BITS)	416	16**	560***	-

* MULTIPLIES AND ADDS ARE NOT SIMULTANEOUS FOR MPP

** EXTENDABLE

*** ASSUMES 5 MEMORY CHIPS PER ACE

2.5.3 Design

The design selected for the ADSP features a parallel processor implementation of the FFT convolver algorithm for both the range and azimuth correlators. It is programmable and adaptable to multiple missions. The parallel architecture allows a modular implementation suitable for expansion or reduced system capability. The parallel design achieves high system reliability and graceful performance degradation.

The system is designed with TTL compatible logic, has an internal 8MHz operating clock rate, has floating point arithmetic and employs 64K dynamic RAMs for the bulk memory. It is controlled by a host computer for mission set up and performance monitoring, but because most of the control reqirements are embedded in firmware, a minimum of software effort is anticipated for a new mission set-up.

Figure 51 shows the arrangement of the parallel processing modules. For the assumed 8 MHz clock rates, 22 parallel processors are used for the range correlator and 15 parallel channels are employed in the azimuth correlator.

The architecture and algorithm permits a maximum degree of mission programmability with a minimum of software set-up effort. The host computer is not involved in any of the real-time computations as a process proceeds but is used for configuration and test functions. All real time controls are embedded in the signal processing elements. Commands from the host computer set up the SAR processing parameters.

2.5.3.1 Range Correlator Design

The range correlator is illustrated in Figure 52. Each processing module handles a full radar PRI. The PRI/FFT unit, whose primary function is FFT processing, is the same module used for FFT processing in the azimuth correlator.

Each PRI/FFT processor consists of an input/output (I/O) memory, a working memory for computations, an arithmetic computation unit, a reference function store, and a controller. The I/O memory is large

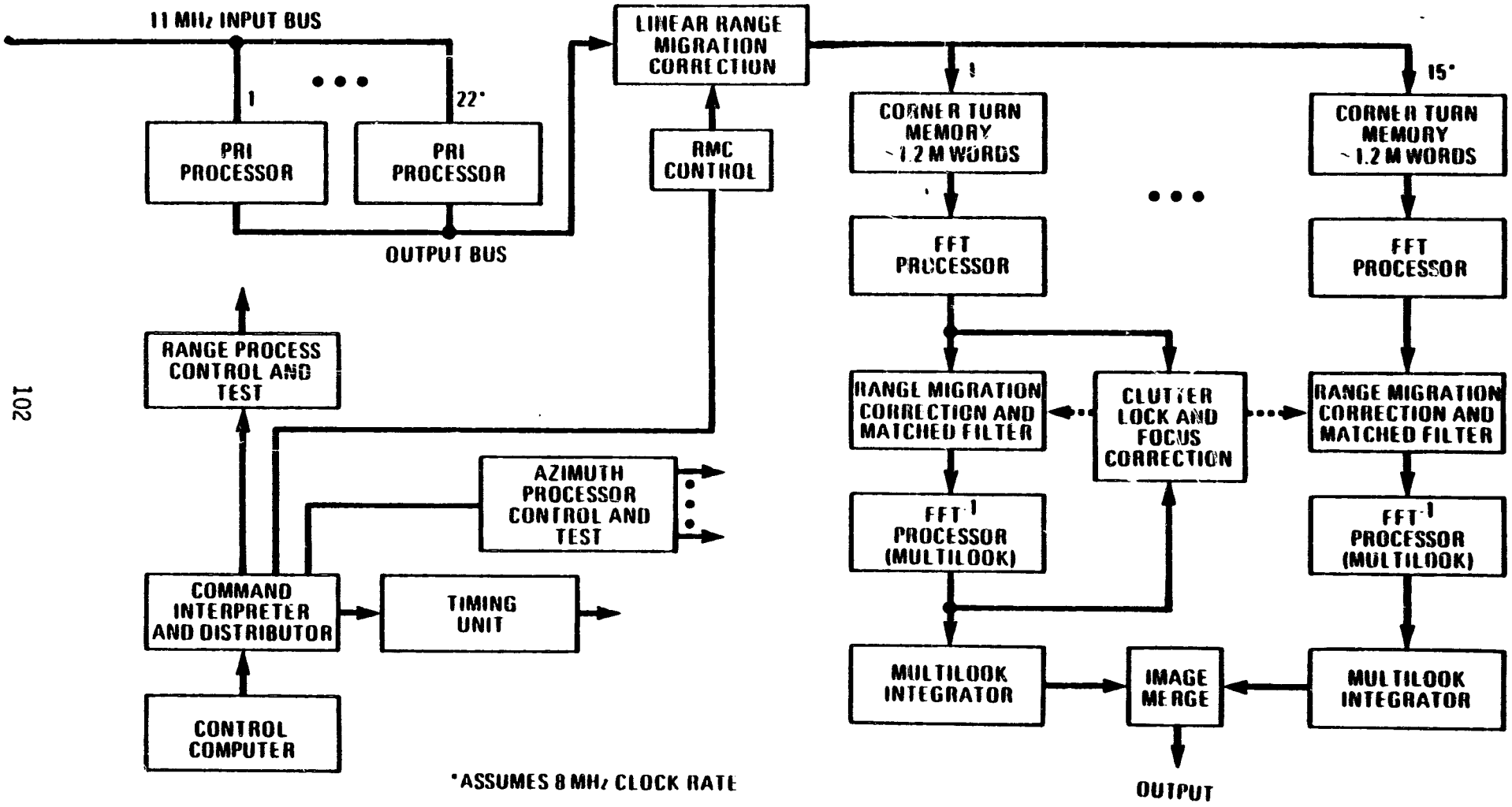


Figure 51

Advanced Digital SAR Processor Design

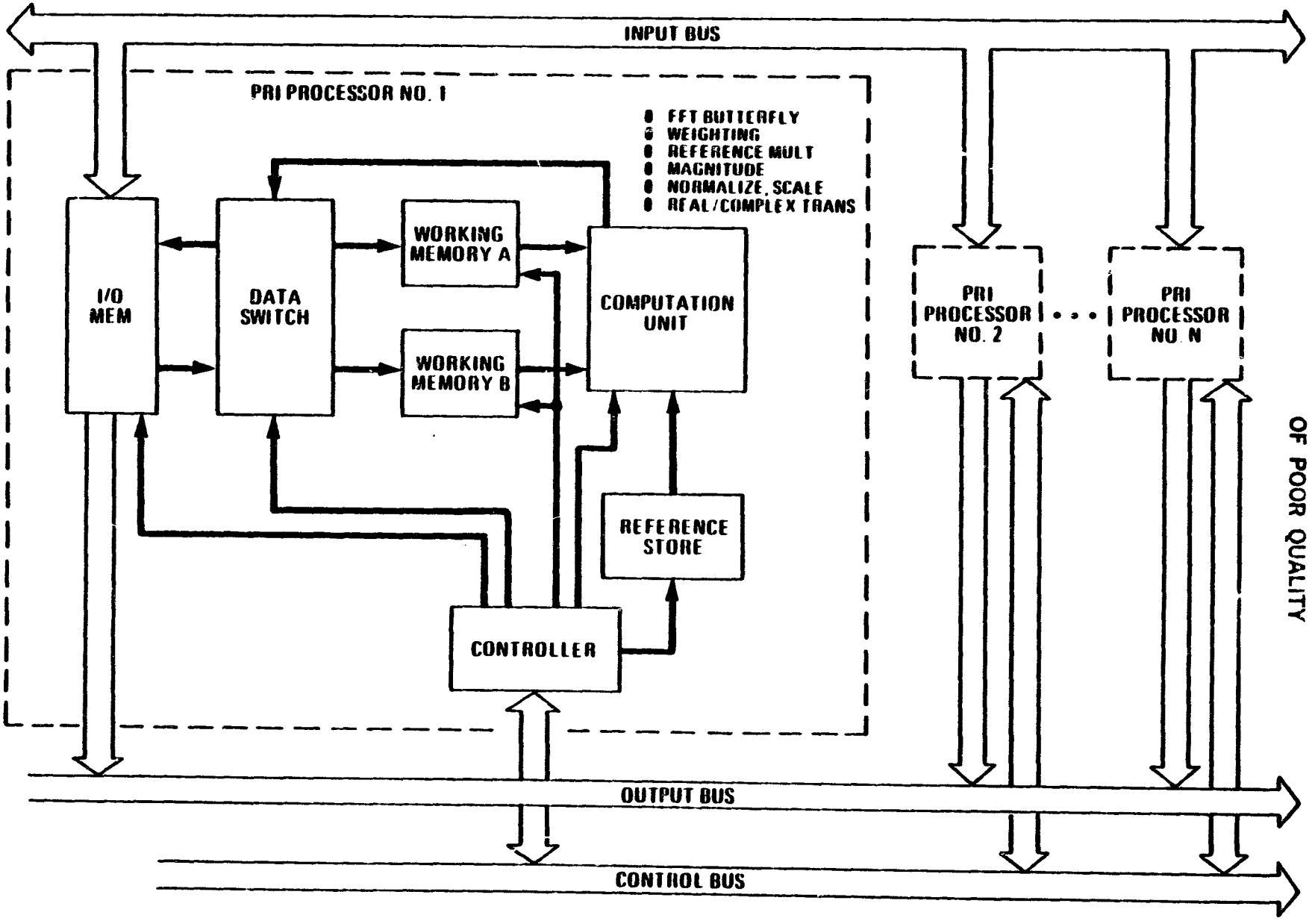


Figure 52

Range Correlator

enough to hold the full range swath window prior to range compression. Data is transferred from the I/O memory through the data switch to the working memory for processing. The computation unit can be programmed to do the FFT butterfly, weighting, reference multiplication, magnitude calculation (I^2+Q^2 or $\sqrt{I^2+Q^2}$), normalization, scaling, and either real or complex FFT computations.

2.5.3.2 Linear Range Migration Correction

From the point of view of control simplicity it appears preferable to compensate for all of the linear components of range migration separately from the quadratic terms. This is the approach employed in the recommended design where the correction is applied after range pulse compression. The implementation is simply an incremental delay plus an interpolated fractional sample delay. Our simulation showed that, with the FFT convolver processing algorithm, adequate performance could be obtained without the use of a separate linear range migration unit if interpolation were used in the frequency domain correction. However, we have included the function since it may contribute to higher quality imagery in some cases.

2.5.3.3 Corner Turning Memory

The corner turning memory module functions are given in Figure 53. A PRI buffer is used to capture the range swath to be processed. The net data rate through each parallel channel is then only about 22/15 MHz. This low data rate permits the use of slow 64K dynamic RAMs for the bulk store.

The organization of the corner turning memory modules is shown in Figure 54. With a 4096 point azimuth processing window, the windows must be repeated each 2048 points. This is accomplished by using an overlap - save sequence in the memories. After the full 4K samples are stored in two memories the next 2K of data is read in to one-half the memory while the 4K window is read out. Operation in this manner means that the total bulk corner turning memory storage is approximately the product of the range swath and the azimuth correlation extent.

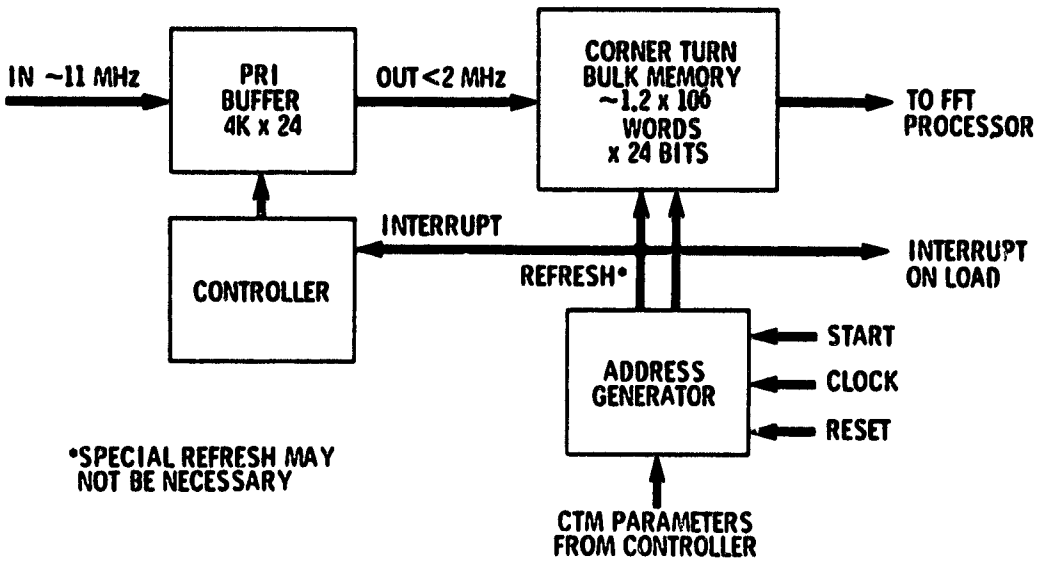


Figure 53
Corner Turning Memory Function

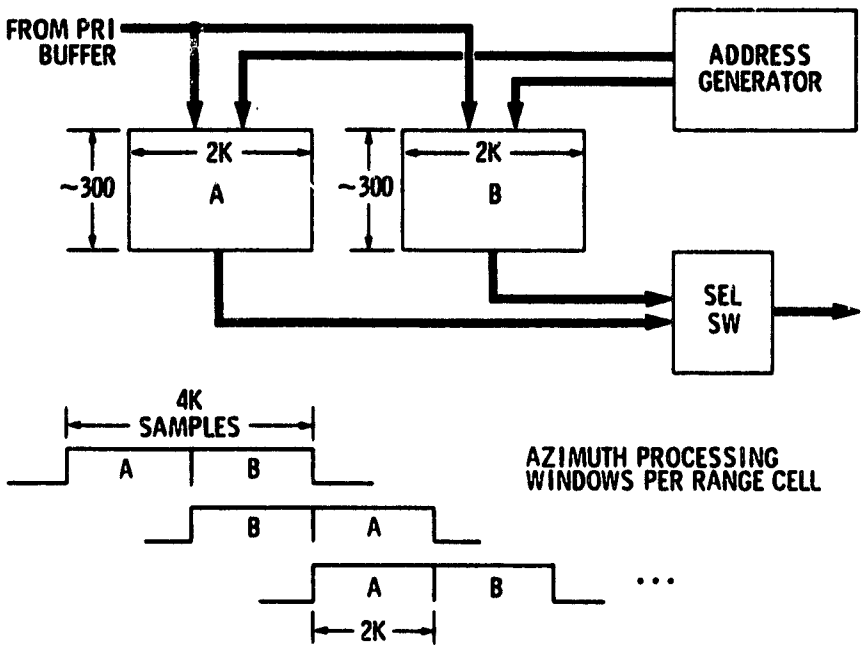


Figure 54
Corner Turning Memory Organization

2.5.3.4 Azimuth Correlation

After the first FFT is taken to convert the constant range lines to the spectral domain, the quadratic range migration correction (and any residual linear component) is applied together with the spectral reference focusing function. Bearing in mind that the data rate at the interface to this function shown in Figure 55 is less than 2 MHz, the implementation of the functions can be streamlined by time-sharing hardware elements. Most of the adaptive processes in the azimuth correlator are centered here. A separate clutter lock estimator is used. Although the focusing function generation is shown on the module, our selected approach for this function is to employ a separate FFT processor module for computing all of the references as discussed later in this section.

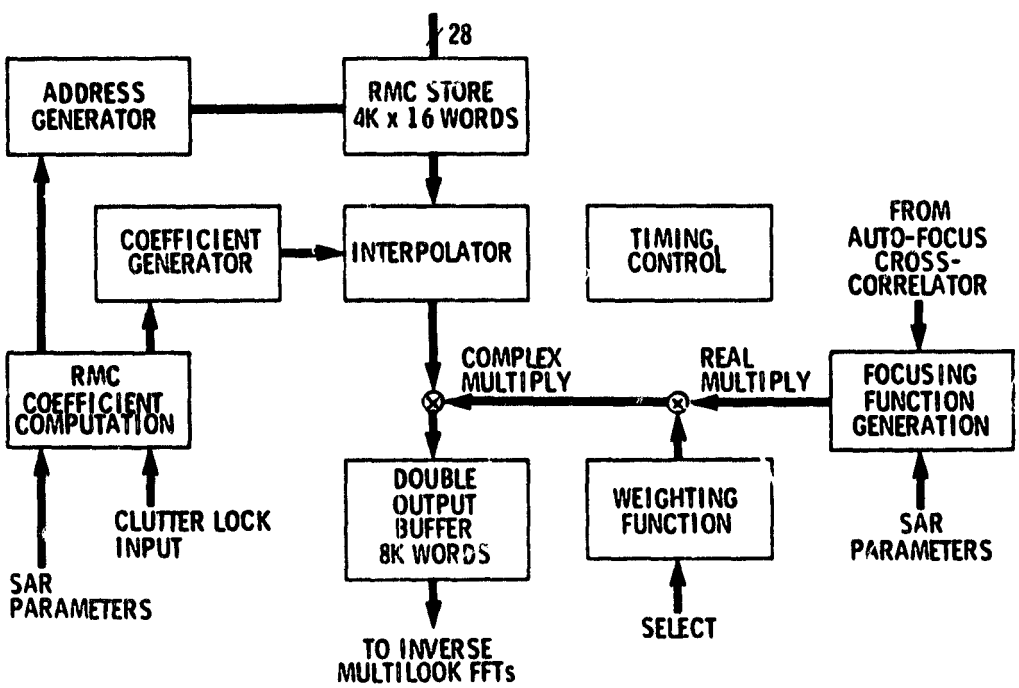


Figure 55

Range Migration Correction and Matched Filter

2.5.3.5 Multilook Integrator

Figure 56 shows the design approach for the multilook integrator. Input data is first interpolated, if necessary, for image registration or geometric mapping correction, or changes in the output image pixel density. If the autofocus is not accomplished with modification to the focusing function, multilook registration corrections can be applied here. The multiple looks are integrated after the magnitude of the complex samples are obtained. The multiple looks are integrated after the magnitude of the complex samples are obtained. The multilook memory store is approximately one-eighth as large as the corner turning memory and all of the associated functions can therefore be included on the module.

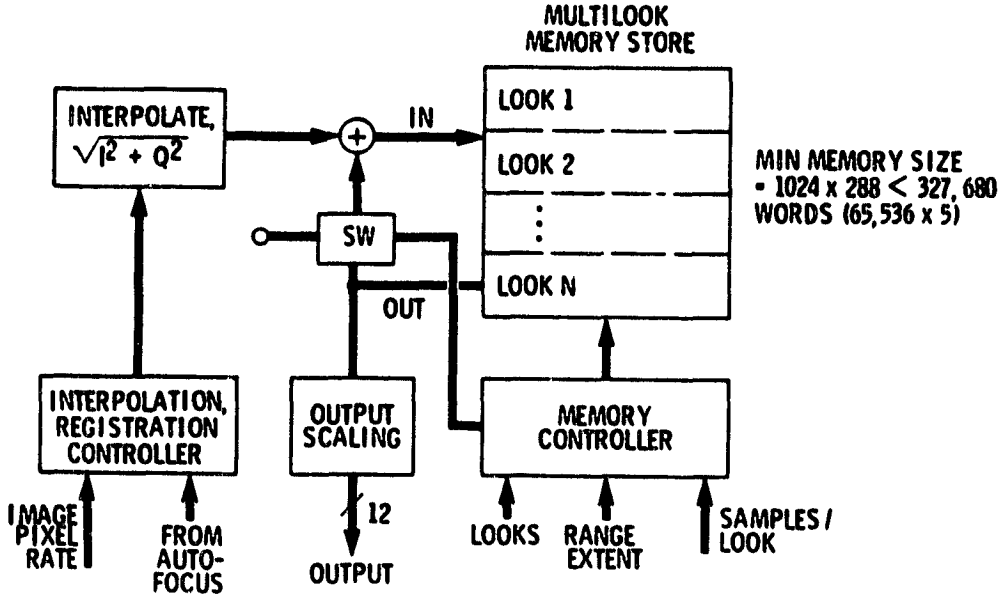


Figure 56
Multilook Integrator

2.5.3.6 Clutter Lock

A clutter lock technique was described in (22). It relies on a measure of symmetry of the mean image energy of the four looks. The normalized error is determined from

$$\frac{E_1 + E_2 - E_3 - E_4}{E_1 + E_2 + E_3 + E_4}$$

where E_{1-4} are the energy levels in the respective looks. A modification of this approach is suggested here based on the following two observations. First, the referenced approach assumes that the actual mean image energy of a particular look, say look 3, does not fluctuate too much as the terrain is changed. Although this is generally valid, there may be specific peculiarities, e.g. transition from ocean to land, that will violate this assumption. Secondly, as the frequency offset (Fc-fc) approaches zero, the difference between the mean image energy looks 2 and 3 would appear to be insensitive to changes in (Fc-fc).

The suggested modification uses two different segments of the range compressed data in such a way that the image of look 1 of the first segment coincides with the image of look 4 of the second segment. Furthermore, only the spectra of looks 1 and 4 are used to establish the symmetry, as these are much more sensitive to changes in the frequency offset (Fc-fc).

Two measures of symmetry can be used. The first is the difference of the total image energy of each look, i.e. E_1-E_4. Another measure is obtained by comparing each spectral line in the two looks. Let $\{S_1(k)\}$ and $\{S_4(k)\}$ be the k^{th} component of looks 1 and 4 respectively. Then a good measure is

$$D = \sum_k |S_1(k) - S_4(k)|.$$

This quantity appears to be much more sensitive to changes of the frequency offset (Fc-fc), as can be seen from the result of a typical simulation presented in Table 31. The disadvantage of using D to perform clutter lock is that one has to seek the minimum of D as (Fc-fc) is varied whereas the use of (E_1-E_4) needs only to search for the zero crossing.

Table 31

Clutter Lock Simulation Results

- POINT TARGET
  - TWO POINT INTERPOLATOR IN RANGE COMPENSATION ALGORITHM
  - YAW IS APPROXIMATELY 1/2 A LOOK

- RESULTS:

	$F_c - f_c$						
MEASURE	0.008	0.006	0.004	0.002	0	-0.002	-0.004
E_1	5.057	5.055	5.061	5.070	5.076	5.082	5.089
E_4	5.094	5.087	5.089	5.083	5.076	5.070	5.060
$E_1 - E_4$	-0.0364	-0.0325	-0.0286	-0.0131	-0.0004	+0.0125	+0.0287
D	3.989	3.003	1.964	1.442	0.343	1.468	1.985

2.5.3.7 Generation of Focus Function

The reference signal needed for the azimuth matched filter is of the form $\{e^{-ja(n-n_0)^2}\}_n$ where the constant a depends on the range as well as other system parameters. This dependence on range demands that new reference functions be generated fairly frequently for proper focusing, and the Fourier transform of the newly generated reference function be calculated if the matched filtering is carried out in the frequency domain. The required spectral focusing function can be generated by taking the FFT of a linear frequency ramp function which has been adjusted for the correct slope. While we have initially selected this first method, it is worthwhile to consider alternates which reduce the amount of computations. One method for reducing this computation task is described here.

Assume that $\{e^{-ja(n-n_0)^2}\}$ is the focusing function in use, and the value of a is changed to $a+\Delta$ for focusing at a different range. Rather than generate $e^{-j(a+\Delta)(n-n_0)^2}$ anew, we rewrite $e^{-j(a+\Delta)(n-n_0)^2}$ as follows:

$$e^{-j(a+\Delta)(n-n_0)^2} = e^{-ja(n-n_0)^2} e^{-j\Delta(n-n_0)^2}$$

$$= e^{-ja(n-n_0)^2} \left[1-j\Delta(n-n_0)^2 - \frac{\Delta^2(n-n_0)^4}{2} + \ldots\right]$$

If Δ is sufficiently small, the first two terms in the series expansion are sufficiently accurate for our purpose. In the transform domain:

$$\mathcal{F}\{e^{-j(a+\Delta)(n-n_0)^2}\} \approx \mathcal{F}\{e^{-ja(n-n_0)^2}\} - j\Delta\,\mathcal{F}\{(n-n_0)^2\, e^{-ja(n-n_0)^2}\}$$

$$- \frac{\Delta^2}{2}\,\mathcal{F}\{(n-n_0)^4\, e^{-ja(n-n_0)^2}\}$$

Thus, one needs to store only the three transforms on the right side of the equation and combine them using whatever value of Δ necessary to obtain the transform of the desired new reference.

To determine one range of Δ for which this approach offers sufficient accuracy, we generated a reference with Δ varying and used it for matched filtering the returns from a point target. The results were compared with those obtained when Δ=0.

If there is no yaw, then the proposed alternate will produce less than 0.5dB change in the response when Δ is kept within $\pm$50 range cells. If a yaw corresponding to half a look is present, then Δ has to be within $\pm$ 30 range cells in order to keep the change to within 0.5dB. If the yaw is increased to one look, then Δ has to be within $\pm$20 range cells.

While the alternate method offers significantly fewer computations, the availability of FFT processing modules makes the direct reference computation cost efficient. With a new reference required every eight range cells, two FFT processing modules can be applied to the task of computing the references for 15 parallel channels.

2.5.3.8 Control System

System requirements which affect the range and azimuth processor controls are listed in Table 32. These general requirements impact in various ways on the processor depending upon the algorithm and architecture employed.

Table 32

ADSP System Control Requirements

Range Processor

Range Compression Ratio: 10 to 660

Swath Width: 500 to 4000

Azimuth Processor

Azimuth Compression: 20 - 350 per look

Reference Function Update: Cross track - every 8 range cells

Along track - every 12,000 radar pulses

Required Modes

Dual Frequency

Dual Polarization

Burst

Continous

Variations within Mission in altitude, look angle, resolution, number of looks, swath width, and azimuth weighting

The FFT convolver is relatively easily controlled for the variation of system parameters. The length of the FFT can be changed to accommodate varying time bandwidth products and swath width. This is especially easily accomplished in a serial FFT where the length change is by firmware control. In addition, waveform and weighting function variations can be accommodated by either storing the additional functions in ROM or by inserting them into a temporary storage RAM via the control computer.

A continuous selection of swath widths and waveform length can be batch processed with the FFT convolver within the limits of the maximum size FFT. The FFT can also be operated in a sliding aperture mode. Then successive FFT windows are processed at intervals of the difference between the FFT size and waveform length. Table 33 lists a number of variations in FFT and waveform size and resulting complex computations per output point. The table illustrates the logarithmic increase in the required computations as the waveform and FFT size increase. It also illustrates the computational advantages of increasing the FFT size relative to the waveform length.

TABLE 33

FFT Range Correlator Convolver Variations

SIZE	WAVEFORM SIZE	RANGE COVERAGE PER FFT	COMPLEX COMPUTATIONS PER POINT
512	256	256	20
1024	512	512	22
2048	1024	1024	24
4096	2048	2048	26
8192	4096	4096	28
8192	2048	6144	18.67
8192	1024	7168	16
8192	512	7680	14.9

Controls required on the FFT convolver to meet the total programming requirements are indicated in Figure 57. Some of these controls may be most appropriately placed on a common control module for multiple range processors. The tradeoff in this regard is the consideration for multiple uses of the module, as in the azimuth correlator.

Controls for the azimuth processor are distributed to each functional element. For the FFT convolver algorithm all required controls for the FFT are embedded in the FFT module used in the range processor. The primary control centers are then in the corner turn memory, the range migration-matched filter module and the multi-look integrator. Linear range migration is a separate function in the design as is the clutter lock estimator.

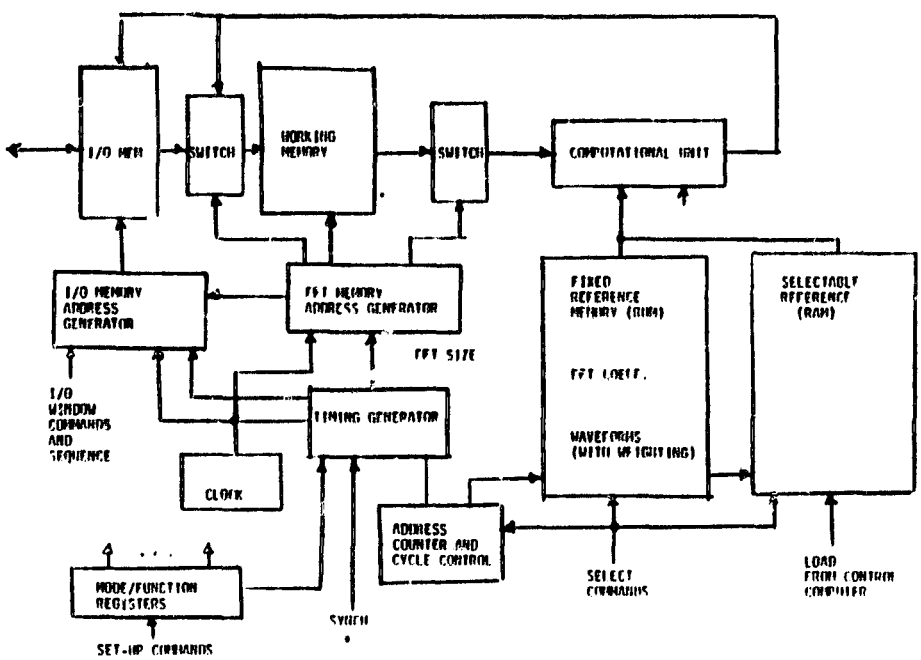


Figure 57 FFT Convolver Control

2.5.3.9 Test Subsystem

The complexity of the ADSP dictates the implementation of a diagnostic, status test subsystem in conjunction with the operating system. A conceptual test system is shown in Figure 58. The system elements would be exercised with a test signal sequence and check-sums would be obtained at selected locations in the processor. These check sums are read back to the control computer for failure localization.

In addition to the built-in-test system, a test fixture should be developed for testing and debugging functional modules. This test unit could be used both for production testing and long term maintenance of the equipment.

2.5.3.10 Physical Configuration

The ADSP system was sized for its physical configuration. The module complement for the system is shown in Table 34. Large 15" x 17" modules were assumed capable of housing 200 or more standard dual-in-line circuits. Using an average of 170 circuits per module the total number of circuits in the system is about 33,000 including 5% spares. Five standard cabinets can be used to house the ADSP with about 2000 watts dissipation per cabinet.

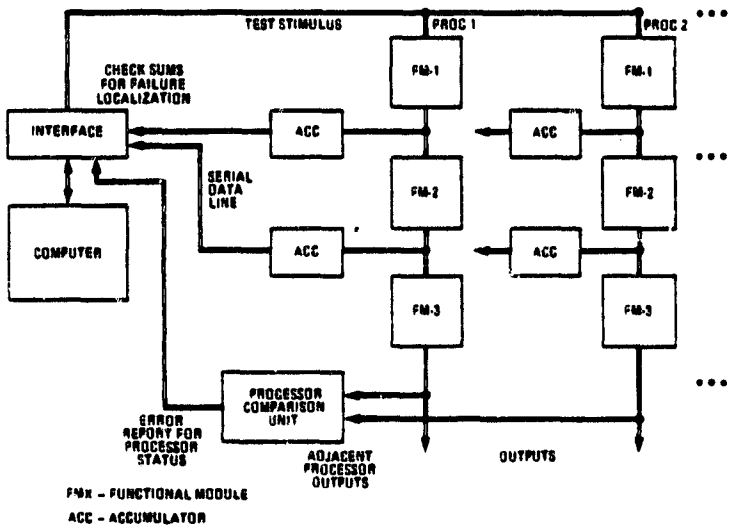


Figure 58 Test System Concept

Wirewrap interconnections were assumed. Because of the large number of circuits in the system it is recommended that they be acquired with a burn-in specification of 60 hours, particularly if plastic circuits are used. A PDP-11/23 computer with 256 K bytes of memory, dual disk, and hardware floating point was selected for the host computer although any of a large number of computers would be adequate.

TABLE 34. MODULE SUMMARY

MODULE TYPE	FUNCTIONAL UNITS	OPERATING SPARES	OTHER SPARES (OPTIONAL)
PRI/FFT Processor	54	4	4
Corner Turn Memory	60	4	4
Range Migration Correction	15	1	1
Multi-look Integrator	15	1	1
Linear Range Migration	1		1
Timing Unit	1		1
Clutter Lock and Focus	1		1
Test Probe Interface	6		1
Controllers	3		3
	156	10	17

2.5.4 Impact of Technology

The specific impact of future technology on the implementation of the ADSP can be deduced from Figures 59 and 60 which show key integrated circuit and architectural developments over the ADSP development time. All of the DC developments shown should serve to minimize the ADSP cost if they are available in a timely manner. Either 256 K bit dynamic RAM's or 64 K bit static RAMs will be available by 1983. Wide word configurations of up to 8 bits will reduce costs. Particular logic developments which can be anticipated are highlighted by a coming selection of chips specifically designed to reduce the cost of implementing the fast Fourier transform (FFT). The culmination of logic will come with the advent of VHSIC technology which is primarily aimed at signal processing applications. In the microcomputer field the recent announcement of the i APX-432 Ada 32 bit microcomputer by Intel Corporation is just the forerunner of an expected set of competitive VLSI computer systems.

Specific signal processors should also be completed within the general time frame of the ADSP as indicated in Figure 60. These developments, however, as previously indicated are not immediately applicable to the ADSP unless they have sufficient maturity, which should come at about the time the ADSP is completed.

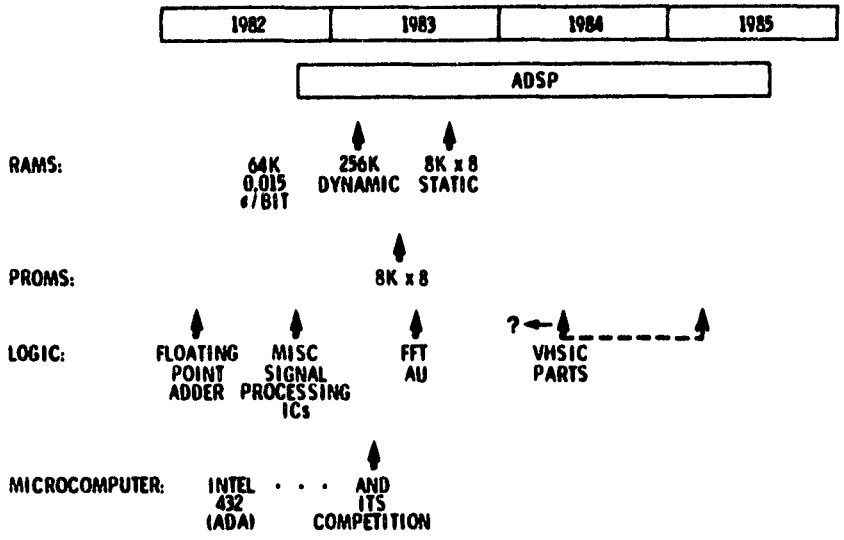


Figure 59
IC Technology Projection

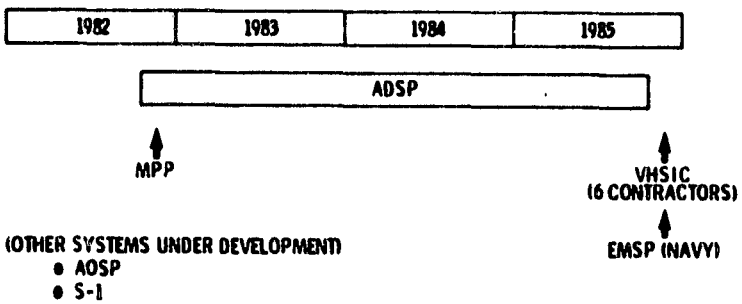


Figure 60
Signal Processor Architecture Projection

2.6 ADSP SCHEDULE AND COST FACTORS

The nominal development time for the ADSP has been established as three years. However, changing the program schedule will affect the expected cost as illustrated in Figure 61 which was derived from outputs of the RCA PRICE* program for the ADSP design. However, at shorter schedules both costs and risks tend to increase until at about 18 months the risks are too high for rational consideration. A 36 month schedule provides a higher cost, but the additional time gives a higher success expectation.

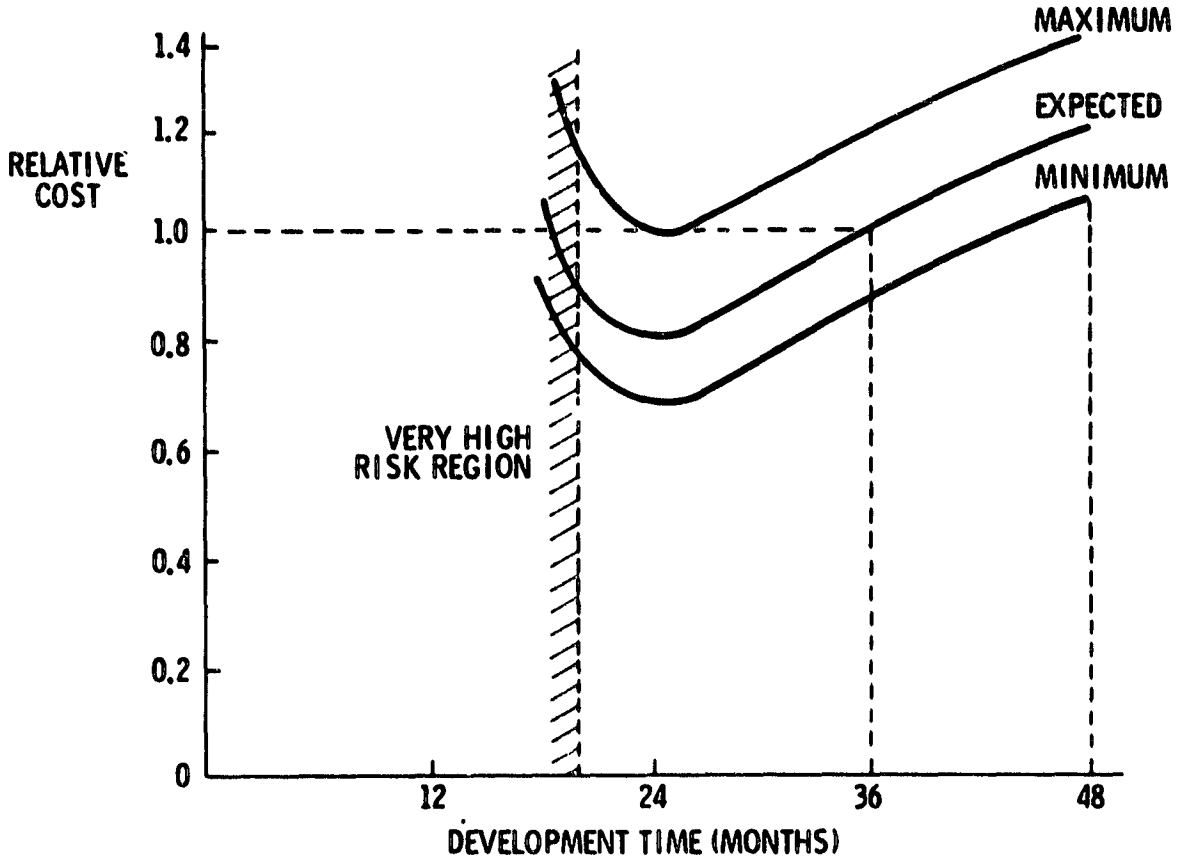


Figure 61
Effect of Schedule on Cost

*PRICE - Programmed Review of Information for Costing and Evaluation

If a two phase program is planned for the three year effort, the expenditure rates in Figure 62 will be obtained. The first 24 months of the job includes all of the design and first piece test work plus the integration of all the elements into a partial working system capable of doing all ADSP functions at a non-real time rate. The two-year design costs include all of the cabinetry and control for the full performance system. The final 12 month period then involves purchasing additional parts and modules and integrating them into a full performance system.

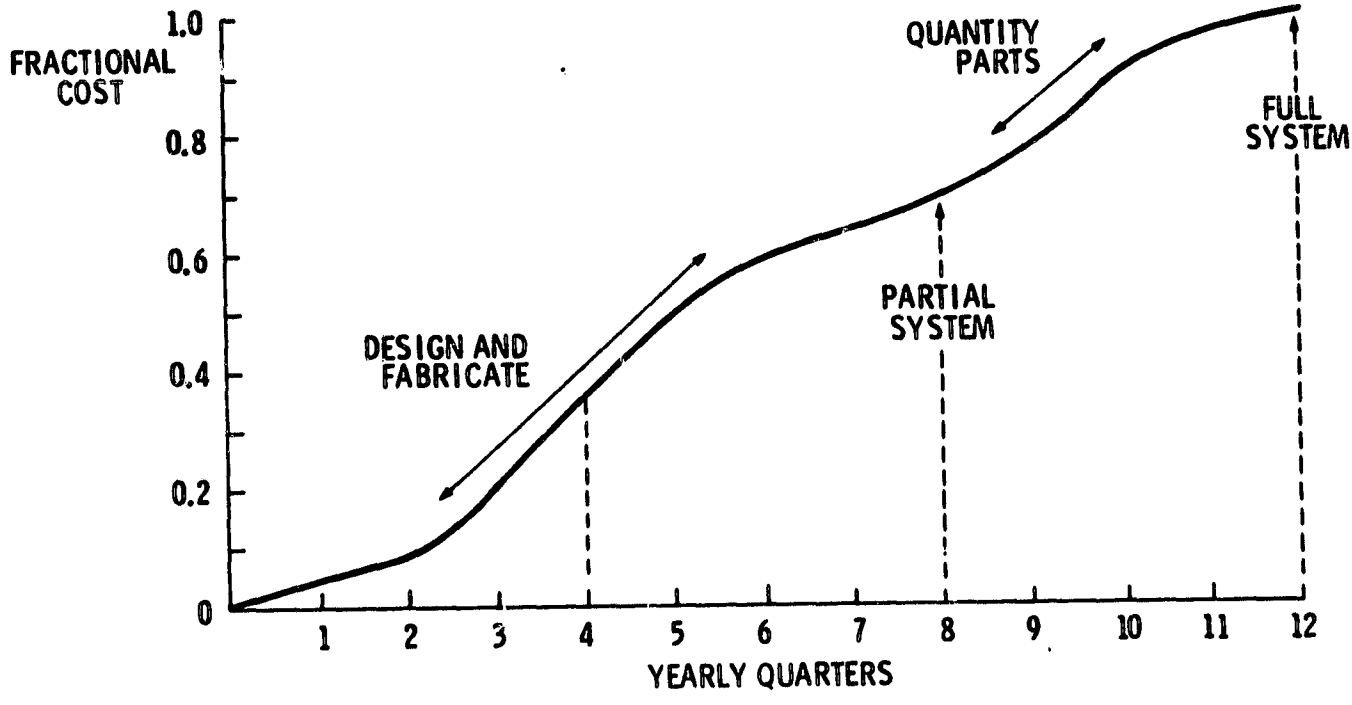


Figure 62
Typical Expenditure Rates

2.7 IMPLEMENTATION PLAN ALTERNATIVES

The recommended schedule for the ADSP is given in Figure 63. It reflects the two phase development outlined in Section 2.6.

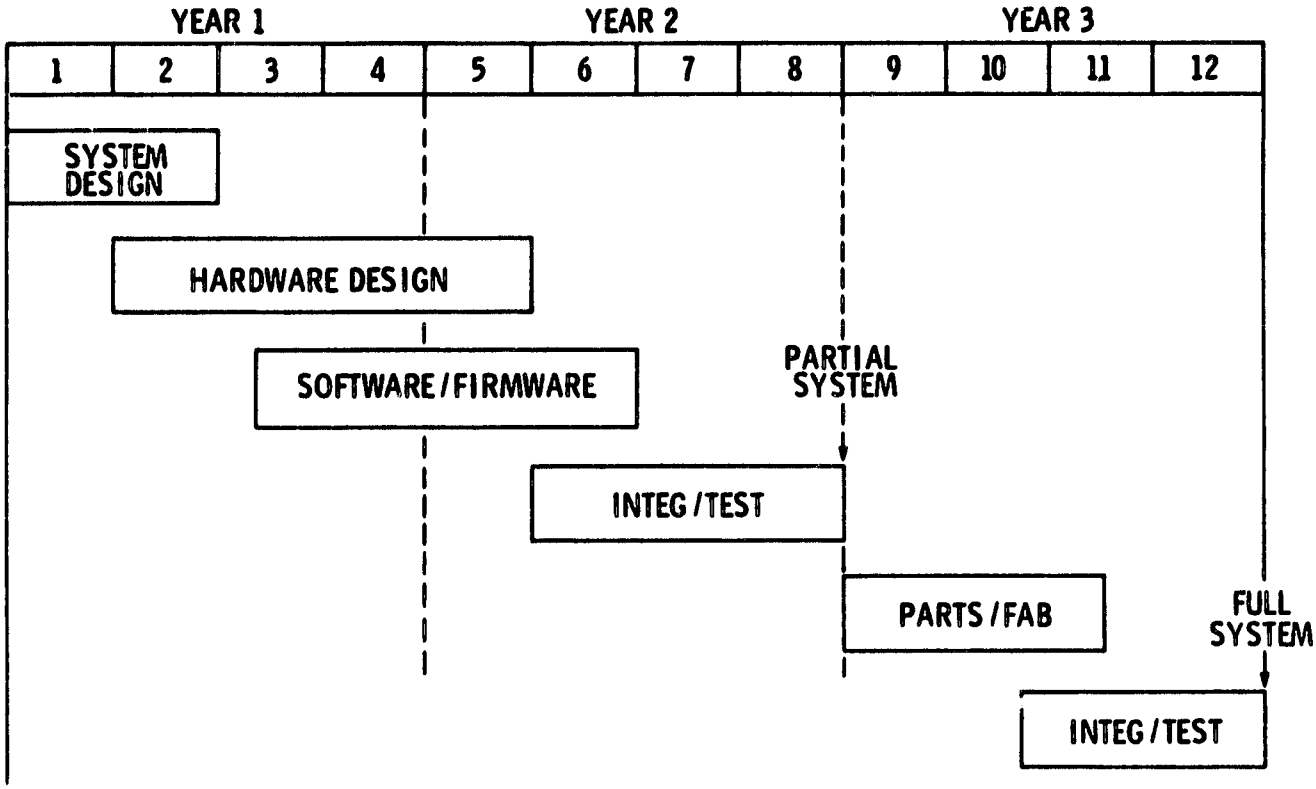


Figure 63
Recommended Schedule

The question of structuring the ADSP as a partial (non-real time) implementation with a four year development cycle was addressed. Full capability would be achieved with later add-ons. Alternatives for this

scenario are listed in Table 35. The first option is essentially to stretch the two-year development cycle of Figure 63 to four years. While this would achieve all program performance objectives, it would result in a cost increase (not including inflation) of 20 percent. In addition, the four year program could produce a technology deficient design unless the hardware design details were delayed until the last two years.

Table 35

Alternative Trade-off

OPTION	RELATIVE COST (4 YEAR)	COMMENTS
1. NONREAL TIME, FULL FUNCTIONS	0.78	• WOULD DEMONSTRATE ALL PROGRAM OBJECTIVES • FOUR YEAR PROGRAM IMPOSES 20% COST PENALTY • FOUR YEAR PROGRAM MAY BE TECHNOLOGY DEFICIENT
2. (a) NONREAL TIME, REDUCED PROGRAMMABILITY	0.73	• REDUCING PROGRAMMABILITY ELIMINATES ONE OF KEY PROGRAM OBJECTIVES • TEST FACILITIES, BOTH BUILT-IN AND MODULE, ELIMINATED
(b) CASE (a) PLUS REDUCED GROWTH	0.60	• LOSS OF PROGRAM OBJECTIVE OF EXPANDABILITY TO FULL REAL TIME SYSTEM WITHOUT ANY DESIGN • FULL SYSTEM CAPABILITY WOULD REQUIRE REPACKAGING OF HARDWARE
3. DELAY HARDWARE	?	• TOTAL COST IS UNKNOWN -- COULD BE HIGHER THAN ALTERNATIVES • APPLIES MOST ADVANCED TECHNOLOGY • HARDWARE DEVELOPMENT BETTER MATCHED TO CURRENT MISSION TIMING

The second option in Table 35 would involve reducing the level of programmability and the built in test and module test facilities.

This would eliminate one of the key program objectives, programmability, and the elimination of test facilities would make the equipment very difficult to maintain. A substantial cost reduction (.40) is achieved if the previous two factors are eliminated plus growth capability. That is a partial system, with its limited control, would be designed and built. A large disadvantage to this approach (2b) is that future expansion would require design effort and thus cause much greater ultimate costs.

The final choice would be to delay the hardware development. While the ultimate cost of this approach is not known, it does have the advantage of applying the most advanced technology, either custom built or purchased. Also since the SAR missions which might use the ADSP do not occur until 1986 and beyond, a delayed program would be more in line with mission timing.

3.0 CONCLUSIONS/RECOMMENDATIONS

General Recommendations

1) The FFT convolver algorithm is the recommended choice for a programmable SAR processor not having specific environmental requirements.
2) Parallel architecture is recommended since it provides modular structures which are matched to program needs in terms of incremental implementation and growth and the thrust of technology.

Performance Results

1) Equivalent performance can be achieved with the FFT convolver, step transform subarray, or time domain azimuth processing algorithms.
2) The performance of the step transform subarray technique is a function of the subarray overlap factor.
3) Range migration compensation without interpolation is not a recommended procedure.
4) Adequate interpolation can be achieved with a four, three, and possibly a two point interpolator.
5) Very little range spreading is observed with any processing algorithm, although some skewing of the resolution cell can occur in the subarray process.
6) The integrated sidelobe levels are comparable with either the subarray or FFT convolver approaches.
7) The mainlobe width (resolution cell) is determined by the weighting function as long as interpolation is employed.

Algorithms

1) The FFT convolver algorithm offers a high degree of programmability with moderate hardware complexity.
2) Time domain approaches, while providing some good control features, require by far the most computations which translate directly to hardware size and cost.

3) Subarray processing offers a minimum hardware approach, but its control system complexity outweighs the hardware reduction for a highly programmable, laboratory based system.
4) The subarray technique may be most appropriate for a dedicated, onboard processor where size, weight, and power requirements predominate.

Architecture

1) Parallel architecture offers the desired system characteristics of modularity, modular implementation, modular growth, high reliability, maintainability, programmability, and accommodation to new technology.
2) Pipeline architectures minimize hardware, but are difficult to make reliable and do not lend themselves to modular implementation and growth.
3) Current government and industry advanced signal processor developments are not ready for immediate use in a real SAR processor.

Integrated Circuit Technology

1) VLSI, VHSIC technology trends will have a profound impact on signal processor implementations such as the ADSP in the 1985 time frame.
2) Between 1981 and 1985 a continuing array of improved integrated circuits will become commercially available including: 256K dynamic RAMs, 8Kx8 static RAMs, 64K PROMs, FFT chips, and advanced microcomputers.
3) These developments will make on board processors realizable with modest size and power.

4.0 NEW TECHNIQUES

A modular parallel SAR signal processing design was conceived and designed which meets the severe performance requirements of the ADSP. The system, described in Section 2.5, is capable of being programmed to cover a wide, continuous range of SAR parameters by virtue of its use of the FFT convolver algorithm. The algorithm has been implemented in a parallel architecture which allows incremental implementation and growth, high reliability, programmability, maintainability, and low cost. The design is adaptable to subsequent advanced technology infusion.

REFERENCES

1. W. E. Arens, "DMSP Azimuth Correlator Implementation Architecture", JPL Interoffice Memorandum 3626-77-017, 12 July 1977

2. Rabiner, L. R. and Gold, B., Theory and Application of Digital Signal Processing, Prentice Hall 1975, p. 153.

3. RCA, "A Study of Linear Approximation Techniques for SAR Azimuth Processing", Final Report, JPL Contract 955067, 1 February 1979.

4. J. W. Cooley and J. W. Tukey, "An Algorithm for the Machine Computation of Complex Fourier Series", Math Comp Vol. 19, 1965, pp 297-301

5. H. J. Nussbaumer and P. Quandalle, "Compilation of Convolutions and Discrete Fourier Transforms by Polynomial Transforms", IBM Tr. Res. Dev. Vol. 22, May 1978, pp 134-144

6. T. K. Troung, R. Lipes, J. S. Reed, and C. Wu, "On the Application of a Fast Polynomial Transform and the Chinese Remainder Theorem to Compute a Two Dimensional Convolution", JPL-TDA Progress Report 42-57, 1980 (submitted to IEEE ASSP Trans.)

7. J. S. Reed, T. K. Troung, and K. Y. Liu, "A Parallel Pipeline Architecture of the Fast Polynomial Transform for a Real Time Synthetic Aperture Radar Processor"

8. A. Peled and B. Liu, "Digital Signal Processing - Theory, Design and Implementation", John Wiley & Sons, 1976, Appendix 5.3

9. B. Arambepola and P. J. W. Rayner, "Efficient Transforms for Multi-dimensional Convolution", Electronic Letters, Vol. 15, 15 March 1979, pp 189-190

10. G. D. Berglund, "A Fast Fourier Transform Algorithm Using Base 8 Iterations", Math. Comp. Vol. 22, 1967, pp 236-238

11. K. Nakayama, "A Mixed Decimation FFT Algorithm", submitted to the IEEE AASP Trans.

12. E. A. Hoyer and W. R. Berry, "An Algorithm for the Two Dimensional FFT", 1977 IEEE Int. Conf. ASSP, pp 552-555

13. C. Caraiscos and B. Liu, "Two Dimensional FFT Using Simultaneous Mixed Decimation", manuscript in preparation

14. C. Wu and K. Y. Liu, " A Pipelined Digital Signal Processor for Producing Real Time SAR Imaging" (Unpublished paper)

15. L. W. Martinson and R. J. Smith, "Digital Matched Filtering with Pipelined Floating Point Fast Fourier Transforms (FFTs)", IEEE Trans. Acoust., Speech, Signal Processing, Vol. AASP-23, No. 2, April 1975, p. 222

16. R. P. Perry, H. W. Kaiser, "Digital Step Transform Approach to Airborne Radar Processing", NAECON '73 Convention Record, April 1973

17. L. W. Martinson, "A Programmable Digital Processor for Airborne Radar", 1975 IEEE International Radar Conference Proceedings, April 1975

18. C. Wu, "A Digital System to Produce Imagery from SAR Data", AIAA Systems Design Driver by Sensors Conference, Pasadena, CA., October 1976

19. T. M. McWilliams, L. C. Widdoes, L. L. Wood, "Advanced Digital Technology Base Development for Navy Applications: The S-1 Project" UCID-1770S, Lawrence Livermore Laboratory, September 1977

20. Raytheon Co., "Advanced Onboard Signal Processor (AOSP), RADC-TR-80-243 Air Force Systems Command, Griffiss Air Force Base, N.Y., July 1980 (portions classified SECRET)

21. K. E. Batcher, "Design of a Massively Parallel Processor", IEEE Trans. Computers, Vol. C 29 No. 9, Sept. 1980, pp 836-840

22. C. Wu, "Electronic SAR Processors for Space Missions", Synthetic Aperture Radar Technology Conference, New Mexico State University, Las Cruces, N.M., March 1978

APPENDIX - PERFORMANCE SIMULATION PROGRAMS

PART I - SAR SIGNAL GENERATION


```
C
C
C    SARG.FOR
C
C
C   PROGRAM TO GENERATE THE RANGE WALK AND
C     THE PHASE HISTORY OF A TARGET IN A
C     SAR TYPE ENVIRONMENT
C
C
          VIRTUAL AI(4096),AQ(4096)
          VIRTUAL TEMPA(2048),TEMPB(2048),RANGER(4096)
          COMMON /INSTR/A(30)
C
C
            CALL INSTR
            CALL INSTRA
            CALL MAP(AI,AQ,TEMPA,TEMPB,RANGER,LMN)
            CALL OUTP(TEMPA,TEMPB,RANGER)
            CALL FFTR(AI,AQ,TEMPA,TEMPB,RANGER,LMN)
              END
```

```
C
C
C   SARG1.FOR
C
C
C   INSTRUCTION SET FOR THE SAR
C
C     C IS THE SPEED OF LIGHT  = 3. E + 8 ( M/SEC )
C     PI = 4.*ATAN(1.)
C
C
C        A(1)  TRANSMITTED FREQUENCY ( MHZ )   FT
C        A(2)  WAVELENGTH OF FT ( M )   C/FT    L
C        A(3)  VELOCITY OF SPACECRAFT ( KM/SEC )  V
C        A(4)  RANGE TO TARGET ( KM )    R
C        A(5)  SAMPLING RATE ( MHZ )   FS
C        A(6)  PRF  ( KHZ )
C        A(7)  NO. OF PULSES TRANSMITTED    N
C        A(8)  BANDWIDTH OF A RANGE CELL ( MHZ )  BW
C        A(9)  RATIO OF FS/BW
C        A(10) TOTAL TIME OF FLIGHT ( SEC ) N/PRF
C        A(11) CONSTANT FOR THE RANGE WALK   CRW
C              V**2/(2*R)      ( M )
C        A(12) RANGE RESOLUTION    RR
C              C/(2*BW)      ( M )
C        A(13) CONSTANT FOR THE RANGE AMPLITUDE FUNCTION
C              PI/RATIO
C        A(14) CONSTANT FOR THE PHASE CORRECTION
C              4*PI*CRW/WL     ( 1/SEC**2 )
C        A(15) HAMMING WEIGHTING ACROSS THE ANTENNA WINDOW
C              YES = 1  NO = 0
C
C        A(16) NUMBER OF RANGE CELLS LOOKED AT
C        A(17) RANGE WALK  YES = 0  NO = 1
C        A(18) BANDWIDTH OF THE SYSTEM ( MHZ )
C        A(19) RANGE CELL THAT TARGET IS IN
C        A(20) NO. OF SAMPLES PER DOPPLER FILTER
C        A(21) RANGE CORRECTION OPTION
C             0 NO CORRECTION
C             1 WU SCHEME
C             2 FRENCH
C              3 STEP TRANSFORM
C
C        A(22) RATIO OF THE SAMPLING RATE TO THE LFM RATE
C
C        A(25) LINEAR RANGE WALK
C              NO = 0  YES = MAX AMOUNT
C        A(26) MAXIMUM AMOUNT OF RANGE WALK
C        A(27) TYPE OF INTERPOLUTION FOR FRENCH SCHEME
C           2 IS 2 PT
C            3 IS 3 PT
C           4 IS 4 PT
C
C        A(30)  TEMPORY STORATE USED FOR THE DYNAMIC RANGE
C
C
C
C
C   ******************
C
C                     SUBROUTINE INSTR
C
C  ******************
C
```

```
C
      COMMON /INSTR/A(1)
C
C
C
      CALL ASSIGN(6,'TT:')
      CALL ASSIGN(5,'LP:')
      CALL ASSIGN(2,'SANG.SNC',0,'SCR',,2)
C
C
C  DISK FILE FOR THE FFT RAW DATA
C
C
      WRITE(6,5)
5     FORMAT(' NAME OF THE DISK FILE ')
      CALL ASSIGN(1,'DUMMY',-1,'NEW',,2)
C
C
C
C  ASK THE QUESTIONS
C
C
      WRITE(6,10)
10    FORMAT(' TRANSMITTED FREQUENCY ( MHZ )')
      A(1)=1250.
0     READ(6,11) A(1)
11    FORMAT(F10.0)
C
C
C  WAVELENTH  C/A(1)
C
C
      A(2)=3.E+2/A(1)
C
C
      WRITE(6,20)
20    FORMAT(' VELOCITY OF THE SPACECRAFT ( KM/SEC ) ')
      A(3)=7.95
0     READ(6,11) A(3)
C
C
C
      WRITE(6,30)
30    FORMAT(' RANGE TO THE TARGET ( KM )')
      A(4)=850.
0     READ(6,11) A(4)
C
C
      WRITE(6,40)
40    FORMAT(' SAMPLING RATE IN THE RANGE DIRECTION ( MHZ )')
      A(5)=22
0     READ(6,11) A(5)
C
C
      WRITE(6,50)
50    FORMAT(' PRF OF BEAMS ( KHZ )')
      A(6)=1.6
0     READ(6,11) A(6)
C
C
      WRITE(6,60)
60    FORMAT(' TOTAL NO. OF BEAMS LAYED DOWN ')
      A(7)=4096.
0     READ(6,11) A(7)
C
C
```

```
      WRITE(6,70)
70    FORMAT(' BANDWIDTH OF A RANGE CELL ( MHZ )'/)
      A(8)=19.
D     READ(6,11) A(8)
C
C
      WRITE(6,80)
80    FORMAT(' HAMMING WEIGHT.NG ACROSS ANTENNA WINDOW '/
     1       ' YES = 1   NO = 0 '/)
      A(15)=0.
      READ(6,11) A(15)
C
C
      WRITE(6,90)
90    FORMAT(' NO. OF RANGE CELLS LOOKED AT PER BEAM '/)
      A(16)=20.
      A(16)=6.
D     READ(6,11) A(16)
C
C
      WRITE(6,95)
95    FORMAT(' DO YOU WANT RANGE WALK '/
     1       ' YES = 0  NO = 1 '/)
      A(17)=0.
D     READ(6,11) A(17)
      WRITE(6,97)
97    FORMAT(' BANDWIDTH OF THE SYSTEM ( MHZ ) '/)
      A(18)=19.
D     READ(6,11) A(18)
C
      WRITE(6,98)
98    FORMAT(' LOCATION OF THE TARGET  WHICH RANGE CELL '/)
        A(19)=3.
D     READ(6,11) A(19)
C
C
C
      WRITE(6,99)
99    FORMAT(' RANGE CORRECTION OPTION '/
     1       ' NONE = 0   WU = 1   CAND = 2 STEP = 3 '/)
        A(21)=1.
        A(21)=2.
      READ(6,11) A(21)
C
      IF(A(21).NE.2.)  GOTO 110
        WRITE(6,102)
102   FORMAT(' TYPE OF INTERPOLATION  2 OR 4 '/)
      READ(6,11) A(27)
110   CONTINUE
C
CCCCCCCCCC
C
C   RATIO OF THE SAMPLING RATE TO THE RANGE CELL BANDWIDTH
C
C
      A(9)=A(5)/A(8)
C
CCC
C  TOTAL TIME OF THE FLIGHT  ( SECS )
C
```

```
      A(10)=A(7)*1.E-3/A(6)
C
C
C  CONSTANT FOR THE RANGE WALK
C
C    M/SEC**2
C
C
      A(11)=A(3)*A(5)/(2.*A(4))
      A(11)=A(11)*1.E+3
C
C
C  RANGE RESOLUTION ASSUMING SMALL ANGLE OF INCIDENCE
C
C
      A(12)=3.E+2/(2.*A(18))
C
C
C CONSTANT FOR THE AMPLITUDE WEIGHTED FUNCTION
C
C
      PI=4.*ATAN(1.)
      A(13)=PI/A(9)
C
C
C  CONSTANT FOR THE PHASE CORRECTION
C
C
      A(14)=4.*PI*A(11)/A(2)
C
          A(25)=0.
      WRITE(5,1050)
1050  FORMAT(' LINEAR RANGE WALK  NO = 0 YES = MAX AMOUNT ')
      READ(6,11) A(25)
C
C
C
            RETURN
            END
```

```
C
C
C     SARG1A.FOR
C
C
C
C
C
C
C     ******************
C
                        SUBROUTINE INSTRA
C
C     ******************
C
C
          COMMON /INSTR/A(1)
C
C
C
C
C
          WRITE(5,1000) (A(I),I=1,7)
1000      FORMAT(' TRANSMITTED FREQUENCY ( MHZ ) ',1PE15.8/
     1           ' WAVELENGTH ( M) ',1PE15.8/
     2           ' VELOCITY OF THE SPACECRAFT ( KM/SEC ) ',1PE15.8/
     3           ' RANGE TO THE TARGET ( KM ) ',1PE15.8/
     4           ' SAMPLING RATE IN RANGE ( MHZ ) ',1PE15.8/
     5    ' PRF IN AZIMUTH ( KHZ ) ',1PE15.8/
     6           ' NO. OF PULSES LAYED DOWN ',F7.1/)
C
C
          WRITE(5,1010) (A(I),I=8,14)
1010      FORMAT(' BANDWIDTH OF A RANGE CELL ( MHZ ) ',1PE15.8/
     1    ' RATIO OF THE SAMPLING RATE IN RANGE TO BANDWIDTH OF A ',
     2    ' CELL ',1PE15.8/' TOTAL TIME OF LOOK ( SEC ) ',1PE15.8/
     3    ' RANGE WALK CONSTANT ( M/SEC**2 ) ',1PE15.8/
     4   ' RANGE RESOLUTION ( M ) ',1PE15.8/
     5    ' CONSTANT FOR AMPLITUDE FUNCTION ',1PE15.8,' ( PI/RATIO )   '/
     6    ' CONSTANT FOR QUADRATIC PHASE ',1PE15.8, ' (SEC**-2) '//)
C
C
          WRITE(5,1020) (A(I),I=15,19)
1020      FORMAT(' HAMMING WEIGHTING ACROSS ANTENNA APERTURE ',
     1    ' YES = 1  NO = 0 ',F5.1/
     2    ' NO. OF RANGE CELLS TO PROCESS ',F5.1/
     3    ' RANGE WALK   YES = 0  NO = 1 ',F5.1/
     4    ' BANDWIDTH OF THE SYSTEM  ( MHZ ) ',1PE15.8/
     5    ' LOCATION OF TARGET  RANGE CELL NO. ',1PE15.8//)
C
C
          DELLFM=2.*A(3)**2*A(10)/(A(2)*A(4))
          DF=A(6)/A(7)
          DFB=DELLFM/DF
          A(20)=DFB/4.
          A(22)=A(6)/DELLFM
C
          WRITE(5,1030) DELLFM,A(6),DF,A(20),A(22)
1030      FORMAT(' CHANGE IN FREQUENCY ACROSS WINDOW ( KHZ ) ',
     1           1PE15.8,'  SAMPLING RATE ( KHZ ) ',
     2           1PE15.8/
     3      ' NO. OF KHZ PER FFT CELL ',1PE15.8,
     4        ' NO. CELLS PER DOPPLER FILTER ',1PE15.8/
     5       ' RATIO OF SAMPLING RATE TO LFM RATE ',1PE15.8//)
```

```
C
C
      WRITE(5,1040, A(21)
1040  FORMAT('    CORRECTION SCHEME  NONE = 0   WU = 1 ',
     1       ' CANO  = 2  STEP = 3 ',F5.1//)
C
C
      IF(A(21).EQ.2.)  WRITE(5,1045) A(27)
1045  FORMAT(' TYPE OF INTERPOLATION  TWO OR FOUR ',F5.1//)
C
C
C
C
C
      WRITE(5,1050)    A(25)
1050  FORMAT(' LINEAR RANGE WALK ADDED '/
     1  ' NO = 0  YES = MAX AMOUNT', F7.2/)
C
C
            RETURN
            END
```

```
C
C
C   SANG2.FOR
C
C
C
C********************
C
                        SUBROUTINE MAP(AI,AQ,TEMPI,TEMPQ,RANGER,LMN)
C
C********************
C
C
        VIRTUAL AI(1),RANGER(1),AQ(1)
        VIRTUAL TEMPI(1),TEMPQ(1)
        COMMON /INSTR/A(1)
C
C
        TIME=-A(10)/2.
        TSTEP=1.E-3/A(6)
        TFINAL=-TIME
C
C
        WRITE(6,6789)  TIME,TSTEP,TFINAL
6789    FORMAT(' BEG ',1PE15.8,' STEP ',1PE15.8,' END ',1PE15.8)
        CONW=A(11)
        DELTAR=A(9)
        DELT2=DELTAR*2.
        CONSR=A(12)
        CONSX=A(13)
        CONS=A(14)
C
C
        NPULSE=A(7)
        IRANGE=A(16)
        IWALKR=A(17)
        IOFFST=A(19)
        IWUOPT=A(21)
           ILINK=A(25)
        IF(ILINK.NE.0)  CONL=-2.*A(25)/(A(7)*TSTEP)
C
C
C
C
C
C
C
C
C
C
              IW=0
C
1       CONTINUE
C
C
C
C   COMPUTE THE AMOUNT OF RANGE WALK
C
        TSQ=TIME*TIME
        RANWLK=0.
        IF(IWALKR.EQ.1) GOTO 5
          WALK=CONW*TSQ
          RANWLK=WALK/CONSR
5       CONTINUE
```

```
C
          IF(ILINR.NE.0) RANWLK=RANWLK + CONL*TIME
C
C
C
C
C    FIND THE PHASE
C
          PHASE=TSQ*CONS
          IW=IW + 1
          AI(IW)=PHASE
          AQ(IW)=RANWLK
            TIME=TIME + TSTEP
          IF(IW.LT.NPULSE)  GOTO 1
C
C
C
C    FIND THE MIN AND MAX RANGE WALK
C
             XMAX=AQ(1)
             XMIN=AQ(1)
            LMAX=1
            LMIN=1
C
          DO 4 I=2,NPULSE
C
            IF(AQ(I).LT.XMAX)  GOTO 2
               XMAX=AQ(I)
               LMAX=I
2           IF(AQ(I).GT.XMIN) GOTO 4
               XMIN=AQ(I)
               LMIN=I
C
4         CONTINUE
C
C
C  NORMALIZE THE MAP TO THE MIN/MAX OF THE RANGE WALK
C
             NRANGE=IRANGE
             LMN=1
          IF(XMAX.EQ.0.)  GOTO 9
             XXMAX=IFIX(XMAX)
            IF(XXMAX.NE.XMAX) XXMAX=XXMAX + 1.
              NRANGE=NRANGE + IFIX(XXMAX)
          IF(XMIN.GE.0.) GOTO 8
              XS=-XMIN
              XXMIN=IFIX(XS)
             IF(XXMIN.NE.XS)  XXMIN=XXMIN + 1.
              NRANGE=NRANGE + XXMIN
             LMN=-XXMIN + 1.
8         IF(IWOPT.EQ.1)  GOTO 9
              NRANGE=NRANGE + 2
             LMN=LMN - 1
9         CONTINUE
C
C
          WRITE(6,6080)  LMIN,XMIN,LMAX,XMAX,LMN,IRANGE,NRANGE
6080      FORMAT(' RANGE INFORMATION '/' MIN ',I6,1X,1PE15.8/
         1 ' MAX ',I6,1X,1PE15.8/' OFFSET ',I6,/
         2   ' NO. WANTED ',I6,' NO PROCESSED ',I6//)
C
C
C
C
C
C  COMPUTE THE SIGNAL FOR EACH BEAM POSITION
```

```
C     AS A FUNCTION OF RANGE
C
C
          L=1
10        CONTINUE
C
              IW=1
             IIW=LMN
          PHASE=AI(L)
           CS=COS(PHASE)
          SS=SIN(PHASE)
          RANWLK=AQ(L)
12        CONTINUE
C
C
                   TEMPI(IW)=0.
                   TEMPQ(IW)=0.
          IF(IW.GT.NRANGE)  GOTO 20
C
               XIW=IIW - IOFFST
               XXIW=XIW - RANWLK
                CALL SINX(V1,XXIW,CONSX)
                XXIW=XXIW + DELTAR
                CALL SINX(V2,XXIW,CONSX)
                XXIW=XXIW - DELT2
                CALL SINX(V3,XXIW,CONSX)
C
C
C
                DD=V1 + 0.426*(V2 + V3)
               TEMPI(IW)=DD*CS
               TEMPQ(IW)=DD*SS
C
                 IW=IW + 1
                 IIW=IIW + 1
                GOTO 12
C
C
20        CONTINUE
C
C
C
C
C
          WRITE(2) (TEMPI(J),J=1,NRANGE),(TEMPQ(J),J=1,NRANGE)
C
                   L=L + 1
             IF(L.LE.NPULSE)  GOTO 10
C
C
100       CONTINUE
C
C
C  REARRANGE THE RANGE
C
          L=NPULSE/2
          DO 110 J=1,NPULSE
           L=L + 1
          IF(L.GT.NPULSE) L=1
          TEMPP=AQ(J)
          RANGER(L)=TEMPP
110       CONTINUE
C
C
C
                A(30)=NRANGE
```

```
            A(24)=XMAX
          WRITE(6,7699)  LXLX,NRANGE,NPULSE
7699      FORMAT(//'  NO. PROC ',I6,' NRANG ',I6,' NPULSE ',I6//)
C
              RETURN
              END
C
C
C
C
          SUBROUTINE SINX(V,X,CONST)
C
          V=1.
          DD=CONST*X
          DABSS=ABS(DD)
          IF(DABSS.GT.1.E-8)  V=SIN(DD)/DD
             RETURN
              END
```

```
C
C
C   SARGS.FOR
C
C
C
C**********************
C
                          SUBROUTINE OUTP(AI,AQ,RANGER)
C
C**********************
C
C
        VIRTUAL AI(50),AQ(50),RANGER(1)
        REAL IA(20)
        COMMON /INSTR/A(1)
C
C
        WRITE(6,10)
10      FORMAT(' PLOT OUT THE TIME RESPONSE   YES = 0 NO =1 ')
            XMAX=1.
        READ(6,11) XMAX
11      FORMAT(F10.0)
C
C
        IF(XMAX.NE.0.)  GOTO 10000
C
C
        DO 12 I=1,20
12      IA(I)=100000.
C
C
            REWIND 2
            NPULSE=A(7)
            NRANGE=A(30)
C
C
C   FIND THE MAXIMUM   KEEP THE AZIMUTH AND RANGE BIN
C
C
            XMAX=0.
        DO 100 J=1,NPULSE
C
          READ(2) (AI(I),I=1,NRANGE),(AQ(I),I=1,NRANGE)
C
C
         DO 50 I=1,NRANGE
C
            AI(I)=AI(I)**2 + AQ(I)**2
           IF(AI(I).LT.XMAX)  GOTO 50
            XMAX=AI(I)
            KJ=J
            KI=I
C
50       CONTINUE
C
C
100     CONTINUE
C
C
C
C   SCALE ALL THE POINTS TO MAX   PUT IN LOG SCALE
C
C
```

```
            NRAN=NRANGE
          IF(NRAN.GT.20)  NRAN=20
C
C
        WRITE(5,500) KI,KJ,XMAX,(I,I=1,NRAN)
500     FORMAT(//' RANGE ',I4,'    AZIMUTH ',I4,
     1  ' OF MAXIMUM    VALUE ( 0 Db REFERENCE ) ',1PE15.8//
     2  5X,20(2X,I2,2X),'   WALK '/)
C
C
C
             REWIND 2
C
C
             JJ=NPULSE/2
C
        DO 200 J=1,NPULSE
C
           READ(2) (AI(I),I=1,NRANGE),(AQ(I),I=1,NRANGE)
C
          DO 150 I=1,NRAN
C
           IA(I)=100000.
           DD=(AI(I)**2 + AQ(I)**2)/XMAX
          IF(DD.GT.0.)  IA(I)=-10.*ALOG10(DD)
C
150       CONTINUE
             JJ=JJ + 1
          IF(JJ.GT.NPULSE) JJ=1
         WRITE(5,1000) J,(IA(I),I=1,20),RANGER(JJ)
1000    FORMAT(I5,20(1X,F5.2),1X,F6.3)
C
200     CONTINUE
C
C
C
C
10000        CONTINUE
C
C
             RETURN
             END
```

```
C
C
C  SARG4.FOR
C
C
C
C******
C       WRITES OUT A FILE IN THE FOLLOWING FORMAT
C
C RECORD #1
C    ARRAY A   INITIAL CONFIGURATION   ( 20 REAL ELEMENTS )
C
C  RECORDS # 2   TO # RANGE CELLS + 1
C
C       I COMPONENT ARRAY OF FFT  AND Q COMPONENT ARRAY OF FFT
C      ( FOR THE STEP TRANSFORM IT IS NON FFT DATA )
C       EACH ARRAY IS OF LENGTH  # OF BEAM POSITIONS  REAL ARRAYS
C       IF # OF BEAMS IS NOT A POWER OF 2 THEN THE LENGTH
C       OF THE ARRAYS IS NEXT HIGHEST POWER OF 2
C
C
C  RECORD # RANGE CELLS + 2
C
C       RANGE WALK FOR EACH BEAM POSITION
C
C
C******************
C
                    SUBROUTINE FFTR(AI,AQ,XXI,XXQ,RANGER,LMN)
C
C******************
C
C
        VIRTUAL XXI(50,40),XXQ(50,40)
        VIRTUAL RANGER(1),AI(1),AQ(1)
        REAL XI(50),XQ(50)
        COMMON/INSTR/A(30)
C
C
C  BRING IN EACH RANGE ROW TO MAKE UP
C   THE AZIMUTH ARRAY
C
C
           NPULSE=A(7)
            IWALK=A(17)
           IWIND=A(15)
            NRANGE=A(16)
          ISCHM=A(21)
          INTP=A(27)
           IRANGE=A(30)
            RATIO=A(22)
          XMAX=A(26)
           XNP=FLOAT(NPULSE/2)/RATIO
           JNP=XNP + 0.5
D       WRITE(6,78999)  JNP,NPULSE,XNP,RATIO
78999   FORMAT(' JNP ',I6,' NPULSE ',I6,' XNP ',1PE15.8,' RATIO ',1PE15.8)
C
C
C
             CONST=A(2)**2*A(4)/(8.*A(3)**2*A(12))
             CONST=CONST*1.E-3
             FSTEP=A(6)*1.E+3/FLOAT(A(7))
D       WRITE(6,5566)   CONST,FSTEP
5566    FORMAT(' COST ',1PE15.8,' FSTEP ',1PE15.8//)
```

```
C
          ILINN=A(25)
        IF(ILINN.NE.0)  CONL=-2.*A(25)*A(22)/FLOAT(NPULSE)
C
C
C
C
C  WRITE OUT THE CONFIGURATION
C
          WRITE(1)  A
D       WRITE(6,8080) NPULSE,IRANGE,A(1)
8080    FORMAT(I6,1X,I6,1X,1PE15.8)
C
C  PUT ME RANGE WALK AMOUNT IN TO FILE FOR THE STEP
C
        IF(ISCHM.EQ.3) WRITE(1) (RANGER(I),I=1,NPULSE)
C
C
        DO 100 I=1,IRANGE
C
             REWIND 2
C
          DO 50 J=1,NPULSE
C
            READ(2) (XI(K),K=1,IRANGE),(XQ(K),K=1,IRANGE)
            AI(J)=XI(I)
            AQ(J)=XQ(I)
C
50          CONTINUE
C
C  WINDOW THE DATA
C
          IF(IWIND.NE.1)  GOTO 55
C
               XKK=0.0006795
                XNN=-NPULSE/2
C
             DO 52 J=1,NPULSE
C
               XX=XNN*XKK
              IF(ABS(XX).GT.0.0001) VAL=(SIN(XX)/XX)**2
                AI(J)=AI(J)*VAL
                AQ(J)=AQ(J)*VAL
               XNN=XNN + 1.
C
52            CONTINUE
C
C
C
55      CONTINUE
C
C     COMPUTE THE FFT OF THE RANGE CELL
C
          IF(ISCHM.NE.3)  CALL FFTV(NPULSE,AI,AQ)
C
C
C
C
C
C
C   WRITE OUT THE ARRAYS TO THE FILE
C
C
            WRITE(1) (AI(J),J=1,NPULSE),(AQ(J),J=1,NPULSE)
          WRITE(6,8080) NPULSE,1
C
```

```
110     CONTINUE
C
C
C
C       FIND THE RANGE WALK FOR EACH ANGLE   TAG IT
C
C
        IF(ISCHM.EQ.0)  GOTO 12000
        IF(ISCHM.EQ.3)  GOTO 13000
C
C
            REWIND 2
             NP=NPULSE/2 + 1
           NPASS=40
           NPP=NPULSE/NPASS
           NOVER=NPULSE - NPP*NPASS
            ISTART=1
           IF(NPP.LT.1)  GOTO 130
           NPIP=NPULSE + 1
           NPP2=NPIP + 1
        ISTR=-LMN + 2
         IENR=ISTR + NRANGE - 1
          IRNGR=IRANGE - NRANGE + 1
C
C
119     DO 120 J=1,NPP
C
            REWIND 1
            READ(1)
C
           DO 106 K=1,IRANGE
C
           IS=ISTART
           READ(1) (AI(L),L=1,NPULSE),(AQ(L),L=1,NPULSE)
C
C
             DO 105 LL=1,NPASS
C
               XXI(K,LL)=AI(IS)
               XXQ(K,LL)=AQ(IS)
                   IS=IS + 1
C
105            CONTINUE
C
106       CONTINUE
C
C
C  CORRECT THE ARRAY IN RANGE
C
C
          DO 110 K=1,NPASS
C
             JJ=ISTART
C
              WALKR=0.
            IF(IWALK.EQ.1)  GOTO 103
           IF(JJ.GT.NP) JJ=NPP2 - JJ
             VALUE=(FLOAT(JJ - 1)*FSTEP)**2*CONST
             WALKR=VALUE
103        IF(ILINR.EQ.0)  GOTO 104
            IF(ISTART.GT.NP) JJ= ISTART - NPULSE
                VAL2=CONL*FLOAT(JJ - 1)
              WALKR=WALKR + VAL2
104        CONTINUE
C
            IF(ISCHM.EQ.2)  GOTO 108
```

```
              IWU=ABS(WALKR) + 0.5
             IF(WALKR.LT.0.) IWU=-IWU
               IS=ISTR + IWU
               IE=IENR + IWU
              IF(IS.GT.0)  GOTO 1066
                 IS=1
                  IE=NRANGE
                GOTO 107
1066          IF(IE.LE.IRANGE)  GOTO 107
                 IE=IRANGE
                  IS=IRNGR
107           WRITE(2) (XXI(LL,K),LL=IS,IE),(XXQ(LL,K),LL=IS,IE)
0          WRITE(6,6060)    ISTART,WALKR,IWU,IS,IE
6060      FORMAT(I6,1X,1PE15.8,3(1X,I5))
             GOTO 109
108          CONTINUE
                  KLL=K
              CALL CAND(XXI,XXQ,WALKR,NRANGE,KLL,INTP,ISTR,IRANGE)
C
             WRITE(2)   (XXI(LL,K),LL=1,NRANGE),(XXQ(LL,K),LL=1,NRANGE)
C
109       CONTINUE
C            WRITE(6,9019) ISTART,JJ,VALUE,VAL2,WALKR,IS,IE
9019      FORMAT(2(1X,I4),3(1X,1PE15.8),2(1X,I4))
C
                 ISTART=ISTART + 1
C
110         CONTINUE
C
120       CONTINUE
C
C
130       IF(NOVER.LE.0)  GOTO 139
C
             NPP=1
              NPASS=NOVER
              NOVER=0
            GOTO 119
C
C
139       CONTINUE
C
C
              REWIND 1
              READ(1)
C
          DO 140 K=1,NRANGE
C
              REWIND 2
C
            DO 131 J=1,NPULSE
C
              READ(2) (XI(L),L=1,NRANGE),(XQ(L),L=1,NRANGE)
              AI(J)=XI(K)
              AQ(J)=XQ(K)
C
131           CONTINUE
C
C
             WRITE(1) (AI(L),L=1,NPULSE),(AQ(L),L=1,NPULSE)
C
C
C
140       CONTINUE
10000     CONTINUE
C
          WRITE(6,8081)    NPULSE
8081      FORMAT('    LAST ONE ',I6//)
C
            CALL CLOSE(1)
C
C
             RETURN
              END
```

```
C
C     SARG4A.FOR
C
C
C
          SUBROUTINE CANO(AI,AQ,RW,N,K,INTP,ISTR,IRANGE)
C
          VIRTUAL AI(50,40),AQ(50,40)
C
C
          IRW=RW
          DELTA=RW - FLOAT(IRW)
            L=1
            J=ISTR + IRW
          IF(INTP.EQ.4) J=J -1
          IF(INTP.EQ.3) J=J -1
          IF(DELTA.LT.0.) J=J - 1
           IF(DELTA.LT.0.) DELTA=1. + DELTA
            IF(J.LT.1)  J=1
             IRANG=IRANGE - N
D         WRITE(5,5050) K,J,ISTR,RW
D         WRITE(5,5055) (JK,AI(JK,K),JK=1,IRANGE)
5050      FORMAT(3(I6,1X),1PE15.8)
5055      FORMAT(I3,1X,1PE15.8,1X,I3,1X,1PE15.8,1X,I3,1X,1PE15.8)
          IF(INTP.EQ.2)  GOTO 100
          IF(INTP.EQ.3) GOTO 300
              IRANG=IRANG - 2
             IF(J.GT.IRANG) J=IRANG
          DELTA2=DELTA*DELTA
          DELTA3=DELTA2*DELTA
C
          AM1=-DELTA*(DELTA2 - 3.*DELTA + 2.)/6.
          A0=(DELTA3 - 2.*DELTA2 - DELTA + 2.)/2.
          A1=-DELTA*(DELTA2 - DELTA -2.)/2.
          A2=DELTA*(DELTA2 - 1.)/6.
C
C
1         CONTINUE
C
          VALUEI=AM1*AI(J,K) + A0*AI(J+1,K) + A1*AI(J+2,K) + A2*AI(J+3,K)
          VALUEQ=AM1*AQ(J,K) + A0*AQ(J+1,K) + A1*AQ(J+2,K) + A2*AQ(J+3,K)
C
           AI(L,K)=VALUEI
           AQ(L,K)=VALUEQ
C
                J=J + 1
                 L=L + 1
            IF(L.LE.N)  GOTO 1
C
C
C    ALL DONE
C
               GOTO 10000
C
100       CONTINUE
C
               DELTA2=1. - DELTA
            IF(J.GT.IRANG) J=IRANG
101       CONTINUE
C
          VALUEI=DELTA2*AI(J,K) + DELTA*AI(J+1,K)
          VALUEQ=DELTA2*AQ(J,K) + DELTA*AQ(J+1,K)
C
```

```
            AI(L,K)=VALUEI
            AQ(L,K)=VALUEQ
C
              J=J + 1
              L=L + 1
            IF(L.LE.N)  GOTO 101
C
              GOTO 10000
300       CONTINUE
C
C
              IRANG=IRANG - 1
            IF(J.GT.IRANG) J=IRANG
C
                AM1=DELTA*(DELTA - 1.)/2.
                A0=1. - DELTA**2
                A1=DELTA*(DELTA + 1.)/2.
C
301       CONTINUE
C
          VALUEI=AM1*AI(J,K) + A0*AI(J+1,K) + A1*AI(J+2,K)
          VALUEQ=AM1*AQ(J,K) + A0*AQ(J+1,K) + A1*AQ(J+2,K)
C
           AI(L,K)=VALUEI
           AQ(L,K)=VALUEQ
C
              J=J + 1
               L=L + 1
            IF(L.LE.N)  GOTO 301
C
C
C     ALL DONE
C
C
10000       CONTINUE
0         WRITE(5,5055) (JK,AI(JK,K),JK=1,N)
             RETURN
              END
```

```
C*
C*       FFTV.FOR
C*
C*
C*    FFT ALGOTRITHM FOR VIRTUAL ARRAYS
C*
C*
          SUBROUTINE FFTV(NUMBR,XT,YT)
          VIRTUAL XT(1),YT(1)
C
C   NUMBR IS THE NO. OF POINTS OF GOOD DATA
C   IF NOT A POWER OF TWO PAD TO NEXT HIGHEST
C   WITH ZEROS
C
C
             N2POW=1
             ITEMP=NUMBR
1            ITEMP=ITEMP/2
            IF(ITEMP.LE.1)  GOTO 10
             N2POW=N2POW + 1
             GOTO 1
10           CONTINUE
```

```
C
      N=2**N2POW
      IF(N.GE.NUMBR) GOTO 13
        N=N*2
        N2POW=N2POW + 1
13    CONTINUE
      IF(NUMBR.GE.N)  GOTO 20
C
         N1=NUMBR + 1
      DO 15 LM=N1,N
C
        XT(LM)=0.
        YT(LM)=0.
C
15    CONTINUE
C
C
20    CONTINUE
C
C
      M=N2POW
      DO 600 L0=1,M
      LMX=2**(M-L0)
      LIX=2*LMX
      SCL=6.283185/FLOAT(LIX)
      DO 600 LM=1,LMX
      ARG=(LM-1)*SCL
      C=COS(ARG)
      S=SIN(ARG)
      DO 600 LI=LIX,N,LIX
      J1=LI-LIX+LM
      J2=J1+LMX
      T1=XT(J1)-XT(J2)
      T2=YT(J1)-YT(J2)
      XT(J1)=XT(J1)+XT(J2)
      YT(J1)=YT(J1)+YT(J2)
      XT(J2)=C*T1+S*T2
      YT(J2)=C*T2-S*T1
600   CONTINUE
      NV2=N/2
      NM1=N-1
      J=1


      DO 635 I=1,NM1
      IF(I.GE.J) GO TO 631
      T1=XT(J)
      T2=YT(J)
      XT(J)=XT(I)
      YT(J)=YT(I)
      XT(I)=T1
      YT(I)=T2
631   K=NV2
620   CONTINUE
      IF(K.GE.J) GO TO 635
      J=J-K
      K=K/2
      GO TO 620
635   J=J+K
      NUMBR=N
      RETURN
      END
```

```
C
C
C
C      WINDOV.FOR
C
C     VIRTUAL VERSION
C
C      WINDOWING FUNCTION
C
C
C       ARGUMENTS
C
C
C    ARRAY IS THE ARRAY TO BE WINDOWED  FIRST LOCATION
C
C          ISTART IS THE FIRST ELEMENT OF THE ARRAY TO BE WINDOWED
C
C        NUMBR IS THE NUMBER OF ELEMENTS TO BE WINDOWED STARTING AT ABOVE
C
C            IOPT  IS TYPE OF WINDOW
C
C
C        0 NO WINDOWING (RECT FUNCTION )
C        1  HAMMING WINDOW
C        2 BARTLETT WINDOW    (TRIANGLE)
C        3 HANNING WINDOW
C        4 BLACKMAN WINDOW
C
C
C
C**********************************
C
                SUBROUTINE WINDOV(ARRAY,ISTART,NUMBR,IOPT)
C
C**********************************
C
        VIRTUAL ARRAY(1)
C
C
        IF(IOPT.EQ.0)  GOTO 10000
            XN=NUMBR - 1
            ISTAT=ISTART
        IF(IOPT.EQ.2)  GOTO 200
C
           PI=4.*ATAN(1.)
          PI2=2.*PI
          PHASE=PI2/XN
        IF(IOPT.EQ.4)  GOTO 100
C
          C1=0.5
          C2=0.5
        IF(IOPT.EQ.3)  GOTO 10
C
          C1=0.54
           C2=0.46
C
10      CONTINUE
C
C
        DO 20 I=1,NUMBR
C
          XN=I - 1
          WD=C1 - C2*COS(PHASE*XN)
          ARRAY(ISTAT)=WD*ARRAY(ISTAT)
```

```
            ISTAT=ISTAT + 1
C
20      CONTINUE
C
            GOTO 10000
C
100     CONTINUE
C
        PHASE2=PHASE*2.
C
        DO 120 I=1,NUMBR
C
          XN=I - 1
          WD=0.42 - 0.50*COS(PHASE*XN) + 0.08*COS(PHASE2*XN)
          ARRAY(ISTAT)=WD*ARRAY(ISTAT)
            ISTAT=ISTAT + 1
C
120     CONTINUE
C
            GOTO 10000
C
C     NUMBR IS THE NUMBER OF ELEMENTS IN THE WINDOW STARTING
C       FROM THE FIRST ADDRESS OF ARRAY
C
C
200     CONTINUE
C
            IUP=NUMBR/2
C
        DO 210 I=1,IUP
C
          WD=FLOAT(2*(I - 1))/XN
          ARRAY(ISTAT)=WD*ARRAY(ISTAT)
            ISTAT=ISTAT + 1
C
210     CONTINUE
C
            IUP=IUP + 1
C
        DO 220 I=IUP,NUMBR
C
          WD=2. - FLOAT(2*(I - 1))/XN
          ARRAY(ISTAT)=WD*ARRAY(ISTAT)
            ISTAT=ISTAT + 1
C
220     CONTINUE
C
C
            GOTO 10000
C
10000       CONTINUE
C
C
              RETURN
               END
```

PART II - FFT CONVOLVER AZIMUTH CORRELATOR

```
C
C
C   MAT.FOR
C
C
C    PROGRAM TO PROCESS THE SAR DATA FROM SARG PROGRAM
C
C
          VIRTUAL ASI(4096),ASQ(4096)
          VIRTUAL FQ(4096),FI(4096)
          COMMON /INSTR/A(30)
          COMMON /TEMP/AZ(10)
C
C
              CALL INSTR
              CALL GEN(FI,FQ)
C
C
          DO 1 I=1,4
C
            CALL DOPPLR(FI,FQ,I)
            CALL SIGNAL(ASI,ASQ,FI,FQ,I)
            CALL OUTPD(FQ,FI,ASI,ASQ)
C
1         CONTINUE
C
C
               END
```

```
C
C     MAT1.FOR
C
C
C     INSTRUCTION SET FOR THE SAR
C
C       C IS THE SPEED OF LIGHT  = 3. E + 8 ( M/SEC )
C       PI = 4.*ATAN(1.)
C
C
C         A(1)   TRANSMITTED FREQUENCY ( MHZ )   FT
C         A(2)   WAVELENGTH OF FT ( M )   C/FT    L
C         A(3)   VELOCTIY OF SPACECRAFT ( KM/SEC )  V
C         A(4)   RANGE TO TARGET ( KM )    R
C         A(5)   SAMPLING RATE ( MHZ )   FS
C         A(6)   PRF  ( KHZ )
C         A(7)   NO. OF PULSES TRANSMITTED    N
C         A(8)   BANDWIDTH OF A RANGE CELL ( MHZ )  BW
C         A(9)   RATIO OF FS/BW
C         A(10)  TOTAL TIME OF FLIGHT ( SEC ) N/PRF
C         A(11)  CONSTANT FOR THE RANGE WALK   CRW
C                V**2/(2*R)      ( M )
C         A(12)  RANGE RESOLUTION     RR
C                C/(2*BW)      ( M )
C         A(13)  CONSTANT FOR THE RANGE AMPLITUDE FUNCTION
C                PI/RATIO
C         A(14)  CONSTANT FOR THE PHASE CORRECTION
C                4*PI*CRW/WL     ( 1/SEC**2 )
C         A(15)  HAMMING WEIGHTING ACROSS THE ANTENNA WINDOW
C                YES = 1  NO = 0
C
C         A(16)  NUMBER OF RANGE CELLS LOOKED AT
C         A(17)  RANGE WALK  YES = 0  NO = 1
C         A(18)  BANDWIDTH OF THE SYSTEM ( MHZ )
C         A(19)  LOCATION OF THE TARGET
C         A(20)  NO. OF CELLS FOR EACH DOPPLER FILTER
C         A(21)  RANGE CORRECTION OPTION
C                0 NONE
C                1 WU
C                2 CANUIANS
C
C         A(22)  RATIO OF THE SAMPLING RATE TO THE LFM RATE
C         A(25)  LINEAR FM WALK ADDED NO = 0 YES = MAX AMOUNT
C
C         A(27)    TYPE WINDOW ACROSS FILTER
C         A(29)   STARTING LOCATION IN FREQ OF THE FILTER
C         A(30)   STOPPING LOCATION OF THE MATCHED FILTER
C
C
C
C
C     ******************
C                       SUBROUTINE INSTR
C     ******************
C
C
C
C         COMMON /INSTR/A(30)
C
C
C
C         CALL ASSIGN(5,'TT:')
```

```
C
C
      WRITE(6,1)
1     FORMAT(' NAME OF LINEPRINTER OUTPUT FILE '/
     1  ' LP:    DK0:NAME    MT0: NAME    '/)
C     CALL ASSIGN(5,'DUMMY',-1)
      CALL ASSIGN(5,'LP:')
C     WRITE(6,2)
2     FORMAT('  NAME OF FILE FOR OUTPUT ( DK: OR MT0:NAME ) '/)
      CALL ASSIGN(4,'OUTP.SCR',0,'SCR',,2)
      CALL ASSIGN(3,'OUTP2.SCR',0,'SCR',,2)
C
C
C     NAME OF DISK FILE TO BE PROCESSED
C
C
      WRITE(6,5)
5     FORMAT(' NAME OF DISK FILE TO BE PROCESSED '/)
      CALL ASSIGN(1,'DUMMY',-1)
C
C  READ IN THE CONFIGURATION
C
          READ(1) A
C
C
C
C
C
      WRITE(5,1000) (A(I),I=1,7)
1000  FORMAT(' TRANSMITTED FREQUENCY ( MHZ ) ',1PE15.8/
     1       ' WAVELENGTH ( M) ',1PE15.8/
     2       ' VELOCITY OF THE SPACECRAFT ( KM/SEC ) ',1PE15.8/
     3       ' RANGE TO THE TARGET ( KM ) ',1PE15.8/
     4       ' SAMPLING RATE IN RANGE ( MHZ ) ',1PE15.8/
     5    ' PRF OF THE AZIMUTH ( KHZ ) ',1PE15.8/
     6       ' NO. OF PULSES LAYED DOWN ',F7.1/)
C
C
      WRITE(5,1010) (A(I),I=8,14)
1010  FORMAT(' BANDWIDTH OF A RANGE CELL ( MHZ ) ',1PE15.8/
     1  ' RATIO OF THE SAMPLING RATE IN RANGE TO BANDWIDTH OF A ',
     2  ' CELL ',1PE15.8/' TOTAL TIME OF LOOK ( SEC ) ',1PE15.8/
     3  ' RANGE WALK CONSTANT ( M/SEC**2 ) ',1PE15.8/
     4 ' RANGE RESOLUTION ( M ) ',1PE15.8/
     5  ' CONSTANT FOR AMPLITUDE FUNCTION ',1PE15.8,' ( PI/RATIO )  '/
     6  ' CONSTANT FOR QUADRATIC PHASE ',1PE15.8, ' (SEC**-2) '//)
C
C
      WRITE(5,1020) (A(I),I=15,18)
1020  FORMAT(' HAMMING WEIGHTING ACROSS ANTENNA APERTURE ',
     1  ' YES = 1  NO = 0 ',F9.1/
     2  ' NO. OF RANGE CELLS TO PROCESS ',F5.1/
     3  ' RANGE WALK    YES = 0  NO = 1 ',F5.1/
     4  ' BANDWIDTH OF THE SYSTEM  ( MHZ ) ',1PE15.8/)
C
C
      DELLFM=2.*A(3)**2*A(10)/(A(2)*A(4))
      DL=A(6)/A(7)
      A(20)=DELLFM/DL
      A(20)=A(20)/4.
C
C  ABOVE ASSUMES WE HAVE FOUR DOPPLER FILTERS
C
      WRITE(5,1030) DELLFM,A(6),DL,A(20)
1030  FORMAT(' CHANGE IN FREQUENCY ACROSS WINDOW ( KHZ ) ',
     1       1PE15.8,'  SAMPLING RATE ( KHZ ) ',
```

```
     2          1PE15.8/
     3     ' NO. OF KHZ PER FFT SAMPLE ',1PE15.8,
     4       ' NO. OF CELLS PER DOPPLER FILTER ',1PE15.8//)
C
C
      WRITE(5,1040) A(21)
1040  FORMAT(' RANGE CORRECTION FACTOR   NONE = 0  WU = 1 ',
     1         ' CANADIANS = 2 ',F5.1//)
C
C
      WRITE(5,1050)    A(25)
1050  FORMAT(//' LINEAR FM WALK ADDED  NO = 0 YES = MAX AMOUNT ',F6.2//)
C
      WRITE(5,1155) A(22)
1155  FORMAT(//' LOCATION OF THE TARTGET ',F5.1)
C
C
C
C
C
      WRITE(6,1060)
1060  FORMAT(' TYPE OF WINDOW ACROSS MATCHED FILTER '/
     1  ' NONE = 0 HAMMING = 1 BARTLETT = 2 HANNING = 3 ',
     2   'BLACKMAN = 4 '//)
         A(27)=1.
0     READ(6,1061) A(27)
1061  FORMAT(F10.0)
C
      WRITE(5,1062) A(27)
1062  FORMAT(' TYPE OF WINDOW ACROSS MATCHED FILTER '/
     1  ' NONE = 0 HAMMING = 1 BARTLETT = 2 HANNING = 3 ',
     2   'BLACKMAN = 4 ',F5.2//)
            RETURN
            END
```

```
C
C
C   MAT2.FOR
C
C
C
C*******************
C
                      SUBROUTINE GEN(ASI,ASQ)
C
C*******************
C
        VIRTUAL ASI(1),ASQ(1)
        COMMON /INSTR/A(1)
C
C
C   GENERATE OUT THE REFERENCE SIGNAL
C    FOR MATCHED FILTERING IN THE ANGLE DOMAIN
C      LINEAR FM WAVE FORM
C
C
C
C
             NPULSE=A(7)
             TSTEP=1.E-3/A(6)
             TIME=-A(10)/2.
             TFINAL=-TIME
             CONS=A(14)
C
C
               I=1
1       CONTINUE
C
            IF(I.GT.NPULSE)  GOTO 100
C
C
             TSQ=TIME*TIME
             PHASE=TSQ*CONS
             XX=COS(PHASE)
              ASI(I)=XX
             XX=SIN(PHASE)
              ASQ(I)=XX
C
             TIME=TIME + TSTEP
             I=I + 1
         GOTO 1
C
C
100     CONTINUE
C
C
C
C
C   WRITE OUT THE TOTAL WAVE TO A TEMPORY FILE
C
        WRITE(2) (ASI(J),J=1,NPULSE),(ASQ(J),J=1,NPULSE)
C
C
             RETURN
              END
```

```
C
C
C  MAT3.FOR
C
C
C
C*********************
C
                      SUBROUTINE DOPPLR(FI,FQ,ITAG)
C
C*********************
C
C
      VIRTUAL FI(1),FQ(1)
      COMMON /INSTR/A(1)
C
C
         NPULSE=A(7)
         NUMB=NPULSE/4
        IWINDW=A(27)
C
C
          REWIND 2
        READ(2) (FI(I),I=1,NPULSE),(FQ(I),I=1,NPULSE)
C
C
C
C
C
C
C  NULL OUT THE TIME HISTORY OF NO INTEREST
C
C
          KL=1
      IF(ITAG.EQ.1)  GOTO 20
C
C
C
C
      KU=(ITAG - 1)*NUMB
      KL=KU + 1
      DO 10 K=1,KU
C
        FI(K)=0.
        FQ(K)=0.
C
10    CONTINUE
C
C
      IF(ITAG.EQ.4)  GOTO 50
C
20    CONTINUE
C
C
          KU=ITAG*NUMB + 1
        DO 25 K=KU,NPULSE
C
         FI(K)=0.
         FQ(K)=0.
C
25      CONTINUE
C
C
50    CONTINUE
```

```
C
      IF(IWINDW.EQ.0)  GOTO 4551
C
C
C   PUT HAMMING WEIGHTING ACROSS FILTER
C
          ISTART=KL
      CALL WINDOV(FI,ISTART,NUMB,IWINDW)
           ISTART=KL
      CALL WINDOV(FQ,ISTART,NUMB,IWINDW)
C
C
C
C  FIND THE SPECTRUM OF THE REFERENCE
C
C
4551    CONTINUE
          NUMBR=NPULSE
C
        CALL FFTV(NUMBR,FI,FQ)
C
10000    CONTINUE
C
          KKK=KU - KL + 1
      WRITE(6,789) KKK,KU,KL
789   FORMAT(3(1X,I6),' DEL UP LOW ')
CC
C     STORE AWAY THE STARTING AND STOPPPING LOCATIONS
C
               A(29)=1
                A(30)=NUMBR
C
C
C
C
C
           RETURN
           END
```

```
C
C
C  MAT4,FOR
C
C
C
C*******************
C
                    SUBROUTINE SIGNAL(ASI,ASQ,FI,FQ,ITAG)
C
C*******************
C
C
        VIRTUAL ASI(1),ASQ(1)
         VIRTUAL FI(1),FQ(1)
        COMMON /INSTR/A(1)
        COMMON/TEMP/TEMP(1)
C
C
C
           NPULSE=A(7)
           IRANGE=A(16)
                 ISTART=A(29)
                  IEND=A(30)
C
            REWIND 1
            READ(1)
             REWIND 3
C
C
C
C
C
             RMAX=0.
             IAR=0
             ING=0
C
C
        DO 100 I=1,IRANGE
C
          WRITE(6,8080) ITAG,I
8080    FORMAT(' SIG  DOPP ',I6,'  RANGE ',I6)
C
C
          READ(1) (ASI(J),J=1,NPULSE),(ASQ(J),J=1,NPULSE)
C
C
C  MULTIPLY BY THE REFERENCE SIGNAL SPECTRUM
C   COMPLEX CONJUGATE THE RESULT FOR A SUBSEQUENT
C   IFFTV
C
            DO 10 J=ISTART,IEND
C
             XXI=ASI(J)*FI(J) + ASQ(J)*FQ(J)
             XXQ=-ASI(J)*FQ(J) + ASQ(J)*FI(J)
             ASI(J)=XXI
             ASQ(J)=-XXQ
C
10          CONTINUE
C******
C
C
C
C  NULL OUT THE NON-OVERLAPPING REGION OF THE SPECTRUM
```

```
C
          IF(ISTART.EQ.1)  GOTO 20
            IS=ISTART - 1
           DO 11 J=1,IS
C
             ASI(J)=0.
             ASQ(J)=0.
C
11         CONTINUE
C
C
20         CONTINUE
C
          IF(IEND.EQ.NPULSE)  GOTO 30
C
            IE=IEND + 1
C
            DO 21 J=IE,NPULSE
C
              ASI(J)=0.
              ASQ(J)=0.
C
21          CONTINUE
C
C
30       CONTINUE
C
C
C      COMPUTE THE IFFT
C
C
              NUMB=NPULSE
            CALL FFTV(NUMB,ASI,ASQ)
C
C
C   STORE THE MAGNITUDE SQUARED AWAY ON DAT SLOT + 3
C
                XMAX=0.
                 IXM=0
C
C
          DO 40 J=1,NPULSE
C
            ASI(J)=ASI(J)**2 + ASQ(J)**2
            IF(ASI(J).LT.XMAX)  GOTO 40
               XMAX=ASI(J)
               IXM=J
C
40          CONTINUE
C
C
67            WRITE(3) (ASI(J),J=1,NPULSE)
          WRITE(5,1000)     I,IXM,XMAX
          WRITE(6,1000)     I,IXM,XMAX
1000      FORMAT(' RANGE ',I6,' AZ ',I6,' VALUE ',1PE15.8)
          IF(XMAX.LT.RMAX)  GOTO 100
             RMAX=XMAX
             IAR=I
             ING=IXM
C
C
C
C
100       CONTINUE
C
          TEMP(1)=NPULSE
          TEMP(2)=IAR
          TEMP(3)=ING
          TEMP(4)=RMAX
C
C
C
              RETURN
              END
```

```
C*
C*       FFTV,FOR
C*
C*
C*
C*    FFT ALGORITHM FOR VIRTUAL ARRAYS
C*
C*
          SUBROUTINE FFTV(NUMBR,XT,YT)
          VIRTUAL XT(1),YT(1)
C
C   NUMBR IS THE NO. OF POINTS OF GOOD DATA
C   IF NOT A POWER OF TWO PAD TO NEXT HIGHEST
C   WITH ZEROS
C
C
            N2POW=1
            ITEMP=NUMBR
1           ITEMP=ITEMP/2
           IF(ITEMP.LE.1)  GOTO 10
            N2POW=N2POW + 1
            GOTO 1
10          CONTINUE
C
          N=2**N2POW
          IF(N.GE.NUMBR) GOTO 13
            N=N*2
            N2POW=N2POW + 1
13        CONTINUE
          IF(NUMBR.GE.N)  GOTO 20
C
             N1=NUMBR + 1
          DO 15 LM=N1,N
C
            XT(LM)=0.
            YT(LM)=0.
C
15        CONTINUE
C
C
20        CONTINUE
C
C
          M=N2POW
          DO 600 L0=1,M
          LMX=2**(M-L0)
          LIX=2*LMX
          SCL=6.283185/FLOAT(LIX)
          DO 600 LM=1,LMX
          ARG=(LM-1)*SCL
          C=COS(ARG)
          S=SIN(ARG)
          DO 600 LI=LIX,N,LIX
          J1=LI-LIX+LM
          J2=J1+LMX
          T1=XT(J1)-XT(J2)
          T2=YT(J1)-YT(J2)
          XT(J1)=XT(J1)+XT(J2)
          YT(J1)=YT(J1)+YT(J2)
          XT(J2)=C*T1+S*T2
          YT(J2)=C*T2-S*T1
600       CONTINUE
          NV2=N/2
          NM1=N-1
          J=1
```

```
      DO 635 I=1,NM1
      IF(I.GE.J) GO TO 631
      T1=XT(J)
      T2=YT(J)
      XT(J)=XT(I)
      YT(J)=YT(I)
      XT(I)=T1
      YT(I)=T2
631   K=NV2
620   CONTINUE
      IF(K.GE.J) GO TO 635
      J=J-K
      K=K/2
      GO TO 620
635   J=J+K
      NUMBER=N
      RETURN
      END
```

```
C
C
C
C      WINDOV.FOR
C
C     VIRTUAL VERSION
C
C      WINDOWING FUNCTION
C
C
C       ARGUMENTS
C
C
C    ARRAY IS THE ARRAY TO BE WINDOWED  FIRST LOCATION
C
C          ISTART IS THE FIRST ELEMENT OF THE ARRAY TO BE WINDOWED
C
C        NUMBR IS THE NUMBER OF ELEMENTS TO BE WINDOWED STARTING AT ABOVE
C
C            IOPT  IS TYPE OF WINDOW
C
C
C        0 NO WINDOWING (RECT FUNCTION )
C        1  HAMMING WINDOW
C        2 BARTLETT WINDOW    (TRIANGLE)
C        3 HANNING WINDOW
C        4 BLACKMAN WINDOW
C
C
C
C**********************************
C
                  SUBROUTINE WINDOV(ARRAY,ISTART,NUMBR,IOPT)
C
C**********************************
C
        VIRTUAL ARRAY(1)
C
C
        IF(IOPT.EQ.0)  GOTO 10000
            XN=NUMBR - 1
            ISTAT=ISTART
        IF(IOPT.EQ.2)  GOTO 200
C
           PI=4.*ATAN(1.)
          PI2=2.*PI
          PHASE=PI2/XN
        IF(IOPT.EQ.4)  GOTO 100
C
          C1=0.5
          C2=0.5
        IF(IOPT.EQ.3)  GOTO 10
C
          C1=0.54
           C2=0.46
C
10      CONTINUE
C
C
        DO 20 I=1,NUMBR
C
          XN=I - 1
          WO=C1 - C2*COS(PHASE*XN)
          ARRAY(ISTAT)=WO*ARRAY(ISTAT)
```

```
            ISTAT=ISTAT + 1
C
20      CONTINUE
C
            GOTO 10000
C
100     CONTINUE
C
        PHASE2=PHASE*2.
C
        DO 120 I=1,NUMBR
C
          XN=I - 1
          WO=0.42 - 0.50*COS(PHASE*XN) + 0.08*COS(PHASE2*XN)
          ARRAY(ISTAT)=WO*ARRAY(ISTAT)
            ISTAT=ISTAT + 1
C
120     CONTINUE
C
            GOTO 10000
C
C     NUMBR IS THE NUMBER OF ELEMENTS IN THE WINDOW STARTING
C       FROM THE FIRST ADDRESS OF ARRAY
C
C
200     CONTINUE
C
            IUP=NUMBR/2
C
        DO 210 I=1,IUP
C
          WO=FLOAT(2*(I - 1))/XN
          ARRAY(ISTAT)=WO*ARRAY(ISTAT)
            ISTAT=ISTAT + 1
C
210     CONTINUE
C
            IUP=IUP + 1
C
        DO 220 I=IUP,NUMBR
C
          WO=2. - FLOAT(2*(I - 1))/XN
          ARRAY(ISTAT)=WO*ARRAY(ISTAT)
            ISTAT=ISTAT + 1
C
220     CONTINUE
C
C
            GOTO 10000
C
10000       CONTINUE
C
C
              RETURN
               END
```

PART III - SUBARRAY AZIMUTH PROCESSOR

```
C
C
C   DT.FOR
C
C
C    PROGRAM TO PROCESS THE SAR DATA FROM SARG PROGRAM
C      THIS PROGRAM PROCESS THE DATA USING THE STEP TRANSFURM
C
C
C
        VIRTUAL ASI(4096),ASQ(4096)
        VIRTUAL FQ(4096),FI(4096)
        COMMON /INSTR/A(30)
        COMMON /TAY/T(128)
        COMMON /TEMP/TEMP(10)
C
C
            CALL INSTR
1       CONTINUE
C
          IF(TEMP(9).GT.TEMP(8))  GOTO 10
             CALL GEN(FI,FQ)
             CALL DERAMP(FI,FQ,ASI,ASQ)
             CALL CORRCT(FI,FQ,ASI,ASQ)
             CALL FINE(FI,FQ)
             CALL OUTPD(FI,ASI,ASQ,FQ)
           TEMP(6)=TEMP(6) + 1.
           TEMP(9)=TEMP(9) + TEMP(10)
        GOTO 1
10      CONTINUE
C
           AVE=TEMP(7)/TEMP(6)
        WRITE(6,1000)  TEMP(6),TEMP(7),AVE
        WRITE(5,1000)  TEMP(6),TEMP(7),AVE
C
1000    FORMAT(//' NO. ',F5.0,' SUM ',1PE15.8/
     1    ' AVE ',1PE15.8//)
C
C
C
             END
```

```
C
C
C    OT1.FOR
C
C
C    INSTRUCTION SET FOR THE SAR
C
C      C IS THE SPEED OF LIGHT  = 3, E + 8 ( M/SEC )
C      PI = 4.*ATAN(1.)
C
C*******************
C      COMMON BLOCK INSTR
C*******************
C
C        A(1)  TRANSMITTED FREQUENCY ( MHZ )   FT
C        A(2)  WAVELENGTH OF FT ( M )   C/FT    L
C        A(3)  VELOCTIY OF SPACECRAFT ( KM/SEC )  V
C        A(4)  RANGE TO TARGET ( KM )    R
C        A(5)  SAMPLING RATE ( MHZ )   FS
C        A(6)  PRF  ( KHZ )
C        A(7)  NO. OF PULSES TRANSMITTED    N
C        A(8)  BANDWIDTH OF A RANGE CELL ( MHZ )  BW
C        A(9)  RATIO OF FS/BW
C        A(10) TOTAL TIME OF FLIGHT ( SEC ) N/PRF
C        A(11) CONSTANT FOR THE RANGE WALK   CRW
C              V**2/(2*R)      ( M )
C        A(12) RANGE RESOLUTION    RR
C              C/(2*BW)      ( M )
C        A(13) CONSTANT FOR THE RANGE AMPLITUDE FUNCTION
C              PI/RATIO
C        A(14) CONSTANT FOR THE PHASE CORRECTION
C              4*PI*CRW/WL     ( 1/SEC**2 )
C        A(15) HAMMING WEIGHTING ACROSS THE ANTENNA WINDOW
C              YES = 1  NO = 0
C             IF YES OPTION  QUESTION TYPE OF WINDOW
C
C
C        A(16) NUMBER OF RANGE CELLS LOOKED AT
C        A(17) RANGE WALK  YES = 0  NO = 1
C        A(18) BANDWIDTH OF THE SYSTEM ( MHZ )
C        A(19)
C        A(20) NO. OF CELLS FOR EACH DOPPLER FILTER
C        A(21) RANGE CORRECTION OPTION
C              0 NONE
C              1 WU
C              2 CANDIANS
C              3 STEP
C
C
C        A(22) RATIO OF THE SAMPLING FREQUENCY TO THE LFM SWEEP FREQ
C        A(23) CORRECTION OF RANGE WALK 4 PT INTERP = 1  NONE = 2
C                2 PT INTERPOLATOR = 3 3 PT INTER = 4
C        A(24) STARTING SUBARRAY      WHICH LOOK
C                    1 IS FIRST LOOK  SUB = 1
C                     2 IS 2ND LOOK SUB = TOTAL/4 + 1
C
C
C        A(25) SIZE OF THE FIRST FFT
C        A(26) NO. OF POINTS IN THE OVERLAP OF SUBARRAYS
C        A(27) NO. OF SUBARRAYS FOR THE 2ND FFT
C        A(28) SIZE OF THE 2ND OR FINE FFT
C        A(29) NO. OF SUBARRAYS CONTAINING DATA
C
C
```

```
C
C       A(30)   MAXIMUM NUMBER OF RANGE CELLS PROCESSED DUE TO WALK
C
C*****************
C     COMMON BLOCK TEMP
C*****************
C       TEMP(1) STORES THE NO. OF FINAL AZMUTH SAMPLES
C       TEMP(2) THE LOCATION IN RANGE OF THE MAX
C       TEMP(3) THE LOCATION IN AZIMUTH OF THE MAX
C       TEMP(4) THE MAXIMUM VALUE
C       TEMP(5)
C       TEMP(6) THE NUMBER OF DIFFERENT STARTING TIMES HAVE DONE
C       TEMP(7) THE RUNNING AVERAGE OF THE INTEGRATE MAINLOBE TO
C                 THE SIDELOBE FOR DIFFERENT STARTING TIMES
C       TEMP(8) THE LAST TIME FOR DIFFERENT STARTING TIMES
C       TEMP(9) THE RUNNING DELAY FOR DIFFERENT STARTING TIMES
C       TEMP(13)  THE INCREMENT FOR THE DIFFERENT STARTING TIMES
C
C
C
C
C
C
C   *****************
C     .
                      SUBROUTINE INSTR
C
C  ******************
C
C
      COMMON /INSTR/A(30)
      COMMON/TAY/TAYLOR(1)
      COMMON /TEMP/TEMP(1)
C
C
C
      CALL ASSIGN(6,'TT:')
C
C
D      WRITE(6,1)
1     FORMAT(' NAME OF LINEPRINTER OUTPUT FILE '/
     1   ' LP:    DK0:NAME    MT0: NAME    '/)
      CALL ASSIGN(5,'LP:')
      CALL ASSIGN(2,'OUTP2.SAR',0,'UNKNOWN',,2)
      CALL ASSIGN(3,'OUTP3.SAR',0,'UNKNOWN',,2)
      CALL ASSIGN(4,'OUTP4.SCR',0,'SCR',,2)
C
C
C    NAME OF DISK FILE TO BE PROCESSED
C
C
      WRITE(6,5)
5     FORMAT(' NAME OF DISK FILE TO BE PROCESSED '/)
C     CALL ASSIGN(1,'DUMMY',-1)
      CALL ASSIGN(1,'ST12.SAR',0)
C
C  READ IN THE CONFIGURATION
C
          READ(1) A
C
C
       A(25)=64
       A(26)=52.
       A(27)=52
       A(28)=128
C
```

```
      WRITE(6,6)
6     FORMAT(' SIZE OF THE FIRST FFT ')
0     READ(6,7) A(25)
7     FORMAT(F10.0)
      WRITE(6,8)
8     FORMAT(' NO. OF POINTS OF THE OVERLAAP ')
0     READ(6,7) A(26)
      WRITE(6,9)
9     FORMAT(' NO. OF SUBARRAYS USED IN 2ND FFT ')
0     READ(6,7) A(27)
      WRITE(6,10)
10    FORMAT(' NO. OF POINTS IN 2ND FFT ')
0     READ(6,7) A(28)
C
C
C
C
C
C
C
C
C
C
C
      WRITE(5,1000) (A(I),I=1,7)
1000  FORMAT(' TRANSMITTED FREQUENCY ( MHZ ) ',1PE15.8/
     1       ' WAVELENGTH ( M) ',1PE15.8/
     2       ' VELOCITY OF THE SPACECRAFT ( KM/SEC ) ',1PE15.8/
     3       ' RANGE TO THE TARGET ( KM ) ',1PE15.8/
     4       ' SAMPLING RATE IN RANGE ( MHZ ) ',1PE15.8/
     5    ' PRF OF THE AZIMUTH ( KHZ ) ',1PE15.8/
     6       ' NO. OF PULSES LAYED DOWN ',F7.1/)
C
C
      WRITE(5,1010) (A(I),I=8,14)
1010  FORMAT(' BANDWIDTH OF A RANGE CELL ( MHZ ) ',1PE15.8/
     1  ' RATIO OF THE SAMPLING RATE IN RANGE TO BANDWIDTH OF A ',
     2  ' CELL ',1PE15.8/' TOTAL TIME OF LOOK ( SEC ) ',1PE15.8/
     3  ' RANGE WALK CONSTANT ( M/SEC**2 ) ',1PE15.8/
     4 ' RANGE RESOLUTION ( M ) ',1PE15.8/
     5  ' CONSTANT FOR AMPLITUDE FUNCTION ',1PE15.8,' ( PI/RATIO )  '/
     6  ' CONSTANT FOR QUADRATIC PHASE ',1PE15.8, ' (SEC**-2) '//)
C
C
      WRITE(5,1020) (A(I),I=15,18)
1020  FORMAT(' HAMMING WEIGHTING ACROSS ANTENNA APERTURE ',
     1  ' YES = 1  NO = 0 ',F5.1/
     2  ' NO. OF RANGE CELLS TO PROCESS ',F5.1/
     3  ' RANGE WALK   YES = 0  NO = 1 ',F5.1/
     4  ' BANDWIDTH OF THE SYSTEM  ( MHZ ) ',1PE15.8/)
C
C
      DELLFM=2.*A(3)**2*A(10)/(A(2)*A(4))
      DL=A(6)/A(7)
      A(20)=DELLFM/DL
      A(20)=A(20)/4.
C
C  ABOVE ASSUMES WE HAVE FOUR DOPPLER FILTERS
C
      WRITE(5,1030) DELLFM,A(6),DL,A(20)
1030  FORMAT(' CHANGE IN FREQUENCY ACROSS WINDOW ( KHZ ) ',
     1       1PE15.8,'  SAMPLING RATE ( KHZ ) ',
     2       1PE15.8/
     3    ' NO. OF KHZ PER FFT SAMPLE ',1PE15.8,
     4      ' NO. OF CELLS PER DOPPLER FILTER ',1PE15.8//)
C
C
```

```
      WRITE(5,1040) A(21)
1040  FORMAT(' RANGE CORRECTION FACTOR   NONE = 0  WU = 1 ',
     1        ' CANADIANS = 2  STEP = 3 ',F5.1//)
C
C
C
C
C
C
C
C
C
C
      WRITE(6,1060)
1060  FORMAT(//' TYPE OF WINDOW ACROSS EACH APERTURE '/
     1  ' 1 = HAMMING  2 = BARTLETT  3 = HANNING '/
     2   ' 4 = BLACKMAN 5 25 DB TAYLOR',
     3' 6 30 DB TAYLOR 7 35 DB TAYLOR'//)
C
          A(15)=1.
D     READ(6,7) A(15)
C
      WRITE(5,1070) A(15)
1070  FORMAT(//' TYPE OF WINDOW ACROSS EACH SUBARRAY '/
     1 '  1 HAMMING  2 BARTLETT  3 HANNING ',
     2    ' 4 = BLACKMAN 5 25 DB TAYLOR',
     3   ' 6 30 DB TAYLOR 7 35 DB TAYLOR',F5.1/)
C
C
C
      WRITE(5,1080) (A(J),J=25,28)
1080  FORMAT(//' NO. OF COEFFICIENTS IN FIRST FFT ',F6.2/
     1 ' NO. OF POINTS OF OVERLAP OF SUBARRAYS ',F6.2/
     2 ' NUMBER OF SUBARRAYS USED IN FINE FFT ',F6.2/
     3 ' NO. OF POINTS IN THE SECOND FFT ',F6.2//)
C
C
10000 CONTINUE
C
          A(22)=A(6)/DELLFM
      WRITE(5,2000)    A(22)
2000  FORMAT(' RATIO OF SAMPLING FREQ TO LFM RATE ',1PE15.8/)
C
C
C
C
      WRITE(6,1090)
1090  FORMAT(' CORRECTION SCHEME FOR RANGE WALK '/
     1        ' 4 PT INTERP = 1  NONE = 2
     2         2 PT INTERP = 3  3 PT INTER = 4 '/)
          A(23)=1.
      READ(6,7) A(23)
      WRITE(6,1095)
1095  FORMAT(' SUBARRAY STARTING POINT   WHICH LOOK '/)
          A(24)=1.
      READ(6,7) A(24)
C
C
      WRITE(5,1096) A(23),A(24)
1096  FORMAT(' CORRECTION SCHEME FOR RANGE  1 = 4 PT  2 = WU '
     1     F5.2/' SUBARRAY STARTING POINT ',F5.2//)
C
C
C   CALCULATIONS FOR THE DIFFERENT STARTING TIMES
C
C   SLOPE   TIME OVER LFM EXPANDED
```

```
C
           SLOPE=A(10)/DELLFM
C
C  INCREMENT EACH DELAY TIME
C
C  SLOPE = PRF/(SIZE OF 1ST FFT * SIZE OF THE 2ND FFT/2)
C    THE /2 IS DO TO OVERSAMPLING  PADDING WITH ZEROS
C         TWICE THE LEGNTH
C
        TEMP(10)=SLOPE*A(6)/(A(25)*A(28)/2.)
C
C
C   STARTING TIME
C
C   THE /4 IS FOR THE PADDING WITH ZEROS
C
C
C  AND ENDING TIME
C
C
        TEMP(8)=TEMP(10)*A(28)/4.
        WRITE(6,1097)
1097    FORMAT(' STARTING TIME ( 0 = UP )'/)
            XXI=0.
        READ(6,7) XXI
        TEMP(9)=-TEMP(8) + XXI*TEMP(10)
C
C
C   PUT IN FOR EXPED
C
        TEMP(8)=0.
        TEMP(10)=TEMP(10)*8.

C
C
C   RUNNING SUMS AND COUNTER
C
C
          TEMP(7)=0.
           TEMP(6)=0.
C
C
        WRITE(5,1200) (TEMP(I),I=6,10)
1200    FORMAT(' COUNTERS ',2(1X,F5.0)/
     1          ' ENDING AND STARTING TIMES ',2(1X,1PE15.8)/
     2           ' INCREMENT EACH TIME ',1PE15.8//)
C
C
C
             IST=A(15)
             IFFT1=A(25)
        IF(IST.GT.0)  CALL WT(TAYLOR,IFFT1,IST)
        WRITE(6,7891) A(30)
        WRITE(5,7891) A(30)
7891    FORMAT(//' NO. OF RANGE CELLS PROCESSED ALL TOT ',F5.2)
C
C
C
C
        WRITE(6,3457) A(23)
3457    FORMAT(//' TYPE OF INTERPOLATION ',F5.1,
     1   '       1 4 PT  2 NONE  3 2 PT 4 3 PT '//)
        WRITE(5,3457) A(23)
C
C
              RETURN
```

```
C
C   DICK2.FOR
C
C
C
C*********************
C
                      SUBROUTINE GEN(FI,FQ)
C
C*********************
C
        VIRTUAL FI(1),FQ(1)
        COMMON /INSTR/A(1)
        COMMON /TEMP/TEMP(1)
C
C
C   GENERATE OUT THE REFERENCE SIGNAL
C    FOR MATCHED FILTERING IN THE ANGLE DOMAIN
C      LINEAR FM WAVE FORM
C
C
C
C
             NPULSE=A(7)
             TSTEP=1.E-3/A(6)
              DELAY=TEMP(9)
             TIME=-A(10)/2.
              TFINAL=-TIME
              TIME=TIME + DELAY
               TFINAL=TFINAL + DELAY
             CONS=A(14)
C
        WRITE(5,1000)  TEMP(6),TSTEP,DELAY,TIME,TFINAL
        WRITE(6,1000)  TEMP(6),TSTEP,DELAY,TIME,TFINAL
1000    FORMAT(//' STEP NO. ',F6.2,' TIME STEP ',1PE15.8/
     1    ' CORRESPONDING DELAY (SEC ) ',1PE15.8/
     2    ' INITIAL TIME ',1PE15.8,' FINAL TIME ',1PE15.8//)
C
C
C
C
              I=1
1       CONTINUE
C
            IF(I.GT.NPULSE)  GOTO 100
C
C
             TSQ=TIME*TIME
             PHASE=TSQ*CONS
             XX=COS(PHASE)
              FI(I)=XX
             XX=SIN(PHASE)
              FQ(I)=XX
C
             TIME=TIME + TSTEP
             I=I + 1
         GOTO 1
C
C
100     CONTINUE
C
C
C
             RETURN
              END
```

```
C
C
C
C    DICK3.FOR
C
C
C
C************************
                              SUBROUTINE DERAMP(FI,FQ,SI,SQ)
C************************
C
C
        VIRTUAL FI(1),FQ(1),SI(1),SQ(1)
        COMMON /INSTR/A(1)
C
C
C
        NPULSE=A(7)
        IWIND=A(15)
        NRANGE=A(30)
        IRANGE=A(16)
C
C
C
              REWIND 2
             REWIND 1
            READ(1)
         READ(1) (SI(J),J=1,NPULSE)
        WRITE(6,234) SI(1),SI(512),SI(1024),SI(2048),
     1     SI(3096),SI(4096)
234     FORMAT(5(1X,1PE15.8))
C
        DO 500 I=1,NRANGE
C
C  GET THE SIGNAL FROM THE ITH RANGE CELL
C
C
          READ(1) (SI(J),J=1,NPULSE),(SQ(J),J=1,NPULSE)
C
C     DERAMP THE SIGNAL
C
C
           DO 5 J=1,NPULSE
C
             XXI= + SI(J)*FI(J) + SQ(J)*FQ(J)
             XXQ= - SI(J)*FQ(J) + SQ(J)*FI(J)
             SI(J)=XXI
             SQ(J)=XXQ
C
5           CONTINUE
C
C
C   PERFORM THE FFT   STORE RESULTS AWAY ON DISK FILE # 2
C
C   USE A 50 % OVERLAP
C
C
C   PUT THE PHASE CORRECTION ON DUE TO OVERLAPPING APPERTURES
C
C
                CALL OVER(SI,SQ)
C
500     CONTINUE
C
```

```
C
C
C
C    DICK3A.FOR
C
C
C
C************************
                      SUBROUTINE OVEN(SI,SQ)
C************************
C
C
        VIRTUAL SI(1),SQ(1)
        REAL AI(128),AQ(128)
        COMMON /INSTR/A(1)
        COMMON /TAY/TAYLOR(1)
C
C    PERFORM THE FFT   STORE RESULTS AWAY ON DISK FILE + 2
C
C
C
C    PUT THE PHASE CORRECTION ON DUE TO OVERLAPPING APPERTURES
C           CORRCTION FACTON IS
C         EXP(J*2*PI*RATIO OF OVERLAY*ARRAY NO. - 1)
C
C
C
C
C
C
            NPULSE=A(7)
            ILEN=A(25)
            ILEN2=A(26)
            IWIND=A(15)
                IS=1
                IE=ILEN
                NUMB=0
C
C   OVERLAP FACTOR FOR PHASE CORRECTION
C
             D=A(26)/A(25)
             PI=4.*ATAN(1.)
              PI2=2.*PI
             PI2D=PI2*D
C
C
C
7             CONTINUE
C
                 K=1
C
              DO 10 J=IS,IE
C
                 AI(K)=SI(J)
                 AQ(K)=SQ(J)
                 K=K + 1
C
10            CONTINUE
C
C
                IS=IS + ILEN2
                 IE=IE + ILEN2
C
C
```

```
C
C
C     PUT WINDOWING ACROSS THE SUBARRAYS
C
C
          IF(IWIND.EQ.0)  GOTO 17
C
          DO 12 K=1,ILEN
C
             AI(K)=AI(K)*TAYLOR(K)
             AQ(K)=AQ(K)*TAYLOR(K)
C
12        CONTINUE
C
C
17        CONTINUE
C
C
C
               CALL FFT(ILEN,AI,AQ)
C
            IF(NUMB.EQ.0)  GOTO 20
C
C
                PHASE=PI2D*FLOAT(NUMB)
C
C
              DO 18 K=2,ILEN
C
                 XK=PHASE*FLOAT(K-1)
                  SN=SIN(XK)
                  CS=COS(XK)
                   XXI=CS*AI(K) + SN*AQ(K)
                   XXQ=CS*AQ(K) - SN*AI(K)
                        AI(K)=XXI
                        AQ(K)=XXQ
C
18                 CONTINUE
C
20        CONTINUE
C
C
C
C
C
C     STORE AWAY ON DAT SLOT + 2
C
C
            WRITE(2) (AI(K),K=1,ILEN),(AQ(K),K=1,ILEN)
C
                NUMB=NUMB + 1
C
C
             IF(IE.LE.NPULSE)  GOTO 7
C
C
              A(29)=NUMB
C
C
               RETURN
                END
```

```
C
C
C
C   OT6.FOR
C
C
C
C     CORRECTION OF RANGE WALK IN THE STEP TRANSFORM
C
C
C************************
                              SUBROUTINE CORRCT(RANGER,AI,AQ,DUMMY)
C****************************
C
C
      VIRTUAL AI(128,32),AQ(128,32),RANGER(1),DUMMY(1)
      REAL FF(256)
      COMMON /INSTR/A(1)
C
C
C
C
C   NO RANGE CORRECTION ?
C
      IF(A(17).EQ.1.)  GOTO 10000
      RATIO=A(22)
      RMOV=RATIO*A(25)/A(27)
      NPULSE=A(7)
      MAXSUB=A(29)
      MX=MAXSUB - 1
      MINSUB=1
      NMCOEF=A(25)
      NM2=NMCOEF/2
C
C
         REWIND 1
        READ(1) XXI
        READ(1) (DUMMY(I),I=1,NPULSE)
C
C
C   REORDER THE ARRAY
C
C    FIND THE MIN
C
      XMIN=DUMMY(1)
          L=NPULSE/2
      DO 3 I=1,NPULSE
          L=L + 1
        IF(L.GT.NPULSE) L=1
          RANGER(I)=DUMMY(L)
        IF(XMIN.GT.DUMMY(L)) XMIN=DUMMY(L)
3     CONTINUE
C
C
C    FIND THE LOCATION OF THE ZERO RANGE CELL
C
      LMN=1
      IF(XMIN.GE.0.)  GOTO 4
        XS=-XMIN
      XXMIN=IFIX(XS)
      IF(XXMIN.NE.XS) XXMIN=XXMIN + 1.
      LMN=-XXMIN + 1.
4     LMN=LMN - 1
C
```

```
C
C
C
      NRANGE=A(30)
      IRANGE=A(10)
      ISCEME=A(23)
C
C
C
C
           REWIND 4
      DO 100 I=1,MAXSUB
C
C
C  READ IN THE I TH SUBARRAY FOR EACH RANGE CELL AND THEN
C   DO THE CORRECTING   WRITE THE RESULTS OUT TO # 4
C
C
           REWIND 2
C
         II=I - 1
        IF(II.LT.1)  GOTO 10
           DO 5 J=1,II
5           READ(2)
10      CONTINUE
C
C
C
C
        DO 20 J=1,NRANGE
C
          READ(2) (AI(K,J),K=1,NMCOEF),(AQ(K,J),K=1,NMCOEF)
C
          IF(J.EQ.NRANGE)   GOTO 20
             DO 12 K=1,MX
12            READ(2)
20      CONTINUE
C
C
C
C
C  DO THE CORRECTION   WHOLE ARRAY AT A TIME
C
C
             START=I
           IRNG=NRANGE - IRANGE
            IRNG2=IRNG + 1
            IRNGL=IRNG
            IRNG=IRNG - 2
             IRNGLJ=IRNGL - 1
C
           ISTR=-LMN + 2
C
        DO 30 J=1,NMCOEF
C
            ISTART=START + 0.5
            IF(START.LT.0.)  ISTART=START - 0.5
          IF(ISTART.LT.1)  GOTO 35
            IELM=NM2*ISTART
            RANWLK=RANGER(IELM)
          IF((ISCEME.EQ.1).OR.(ISCEME.EQ.3))   GOTO 24
          IF(ISCEME.EQ.4)  GOTO 24
            IRW=ABS(RANWLK) + 0.5
           IF(RANWLK.LT.0.)  IRW=-IRW
            KK=ISTR + IRW
           IF(KK.LT.1)  GOTO 28
```

```
              IF(KK.GT.IRNG2) KK=IRNG2
             DO 22 K=1,IRANGE
C
               AI(J,K)=AI(J,KK)
               AQ(J,K)=AQ(J,KK)
                 KK=KK + 1
22            CONTINUE
                 GOTO 28
24            CONTINUE
C
                  IRW=RANWLK
               DELTA=RANWLK - FLOAT(IRW)
                    K=ISTR + IRW
               IF(ISCEME.EQ.1) K=K - 1
               IF(ISCEME.EQ.4) K=K - 1
               IF(DELTA.LT.0.) K=K - 1
               IF(DELTA.LT.0.) DELTA=1. + DELTA
               IF(K.LT.1)  K=1
                  L=1
             IF(ISCEME.EQ.3)  GOTO 8080
             IF(ISCEME.EQ.4)  GOTO 8090
               IF(K.GT.IRNG) K=IRNG
               DELTA2=DELTA*DELTA
               DELTA3=DELTA2*DELTA
C
               AM1=-DELTA*(DELTA2 - 3.*DELTA + 2.)/6.
               A0=(DELTA3 - 2.*DELTA2 - DELTA + 2.)/2.
               A1=-DELTA*(DELTA2 - DELTA - 2.)/2.
               A2=DELTA*(DELTA2 - 1.)/6.
C
C
26            CONTINUE
C
              VALUEI=AM1*AI(J,K) + A0*AI(J,K+1) + A1*AI(J,K+2) +
     1                 A2*AI(J,K+3)
              VALUEQ=AM1*AQ(J,K) + A0*AQ(J,K+1) + A1*AQ(J,K+2) +
     1                 A2*AQ(J,K+3)
C
               AI(J,L)=VALUEI
               AQ(J,L)=VALUEQ
                  L=L + 1
                  K=K + 1
               IF(L.LE.IRANGE) GOTO 26
C
                  GOTO 28
8080     CONTINUE
C
C      2 PT INTERP
C
            DELTA2=1. - DELTA
C
            IF(K.GT.IRNGL) K=IRNGL
C
8081     CONTINUE
C
             VALUEI=DELTA2*AI(J,K) + DELTA*AI(J,K+1)
             VALUEQ=DELTA2*AQ(J,K) + DELTA*AQ(J,K+1)
               AI(J,K)=VALUEI
                AQ(J,K)=VALUEQ
                  L=L + 1
                  K=K + 1
            IF(L.LE.IRANGE) GOTO 8081
C
                GOTO 28
8090     CONTINUE
C
```

```
            AM1=DELTA*(DELTA - 1.)/2.
            A0=1. - DELTA**2
             A1=DELTA*(DELTA + 1.)/2.
C
8091     CONTINUE
C
            VALUEI=AM1*AI(J,K) + A0*AI(J,K+1) + A1*AI(J,K+2)
            VALUEQ=AM1*AQ(J,K) + A0*AQ(J,K+1) + A1*AQ(J,K+2)
               AI(J,L)=VALUEI
                AQ(J,L)=VALUEQ
                  L=L + 1
                  K=K + 1
             IF(L.LE.IRANGE)  GOTO 8091
C
C
C
20         CONTINUE
C
C
C
               START=START - NMOV
C
30         CONTINUE
C
C
C
C
C  WRITE OUT THE REWULTS TO + 4
C
35         CONTINUE
C
C
            DO 40 K=1,IRANGE
40          WRITE(4) (AI(J,K),J=1,NMCOEF),(AQ(J,K),J=1,NMCOEF)
C
100      CONTINUE
C
C
C  WRITE THE FILE BACK TO + 2 IN THE FORM IT WANTS
C
C
             ISKIP=IRANGE - 1
              NM=2*NMCOEF
              REWIND 2
C
         DO 200 K=1,IRANGE
            REWIND 4
            IF(K.EQ.1) GOTO 110
               II=K - 1
              DO 105 I=1,II
105            READ(4)
110         CONTINUE
C
C
           DO 150 I=1,MAXSUB
C
             READ(4) (FF(J),J=1,NM)
            WRITE(2) (FF(J),J=1,NM)
           IF(I.EQ.MAXSUB)  GOTO 150
             DO 140 J=1,ISKIP
140           READ(4)
C
150        CONTINUE
C
C
200      CONTINUE
C
10000       CONTINUE
C
           RETURN
            END
```

```
C
C
C
C  DT5.FOR
C
C
C
C**********************
                        SUBROUTINE FINE (FI,FQ)
C**********************
C
C
        VIRTUAL FI(1),FQ(1)
        COMMON /COEFS/AI(1024)
        COMMON /INSTR/A(1)
        COMMON /TEMP/TEMP(1)
        COMMON /HAM/HAM(256)
C
C
C
C    COMPUTE THE FINE RESOLUTION FFT
C
C
C
C
C
            IRANGE=A(16)
            RATIO=A(22)
            NMCOEF=A(25)
             NM2=NMCOEF*2
            IFINE=A(27)
            MAXSUB=A(29)
            MINSUB=1
             IPAD=A(28)
            RMOV=RATIO*A(7)/(A(26)*A(25))
C
                STARTS=1.
          IF(A(24).EQ.2.) STARTS=A(7)/(4.*A(26)) + 1.
        WRITE(6,789) STARTS,A(24)
789     FORMAT(' STARTING SUBARRAY ',F5.2,1X,F5.2)
C
C
C
C
C
C
C    GET THE WINDOW WEIGHTS
C
            IST=1
        CALL WT(HAM,IFINE,IST)
        WRITE(5,5000) IRANGE,NMCOEF,IFINE,IPAD,
     1                      MINSUB,MAXSUB
5000    FORMAT(//' FINE    IRANGE ',I6/
     1   ' 1ST FFT ',I6,' # OF SUBS IN 2ND ',I6,
     2   ' 2ND FFT ',I6/' MIN SUB ',I6,' MAX SUB ',I6//)
C
        WRITE(6,5001) RATIO,RMOV
5001    FORMAT(' RATIO OF SF/LFMF ',1PE15.8/
     1    ' INCREMENT OF SUB ',1PE15.8//)
C
            RMAX=0.
            IAZ=0
            IRNG=0
C
```

```
C
              REWIND 2
              REWIND 3
C
        DO 200 K=1,IRANGE
C
              REWIND 4
C
C
C   TRANSFER DATA FOR THE K TH RANGE TO .DAT SLOT + 4
C
            DO 4 I=1,MAXSUB
              READ(2)  (AI(L),L=1,NM2)
              WRITE(4) (AI(L),L=1,NM2)
4           CONTINUE
C
C
C   DO THE COEFFICIENTS   1 TO HALF + 1
C
C
                        IAR=0
              CALL POSHLF(FI,RMOV,STARTS,IAR,COFMAX,ICOF)
C
C
C
C
C   DO THE NEGATIVE COEFFICIENTS
C
C
              CALL NEGHLF(FI,RMOV,STARTS,IAR,COFMAX,ICOF)
C
C
            WRITE(3) (FI(J),J=1,IAR)
C
C   WRITE IT OUT TO THE LP
C
C
        WRITE(5,1000)  ICOF,COFMAX,K
        WRITE(6,1000)  ICOF,COFMAX,K
1000    FORMAT(' MAX IS AT ',I6,' VALUE OF ' 1PE15.8,' RAN ',I5/)
C
C
        IF(COFMAX.LT.RMAX)  GOTO 200
             RMAX=COFMAX
             IAZ=ICOF
             IRNG=K
200     CONTINUE
C
C
             TEMP(1)=IAR
             TEMP(2)=IRNG
             TEMP(3)=IAZ
             TEMP(4)=RMAX
C
C
C            TEMP(1)=1920.
C            TEMP(2)=2.
C            TEMP(3)=33.
C             TEMP(4)=2.2094451665
C
C
C
C
C
C
             RETURN
```

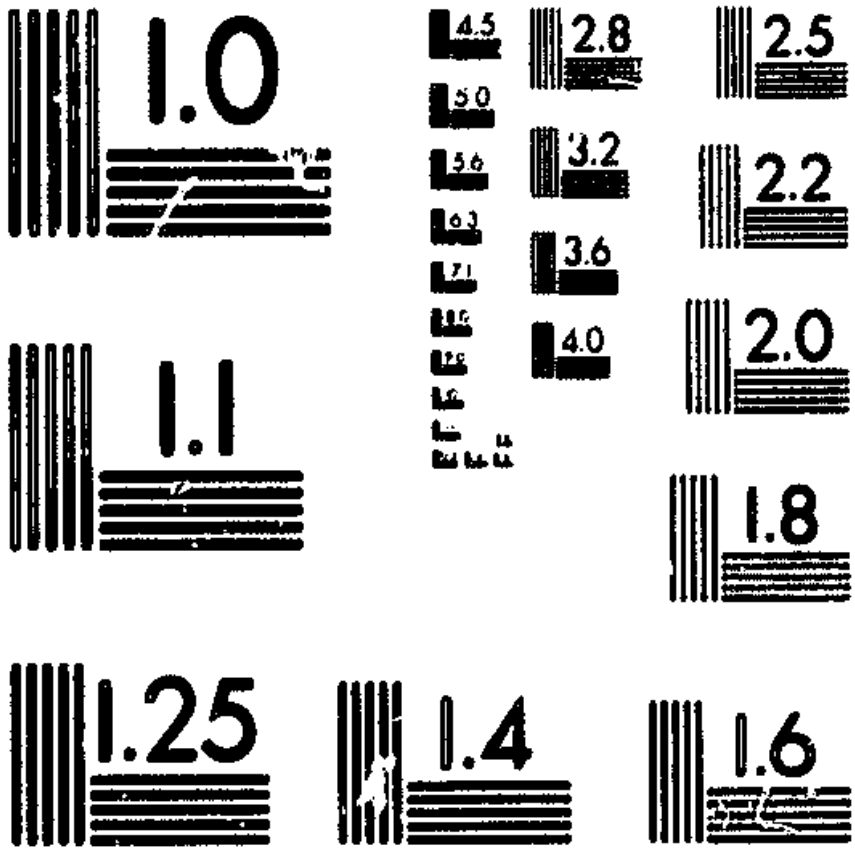
```
C
C
C
C  DTSA.FOR
C
C
C
C**********************
                        SUBROUTINE POSHLP(FI,RMOV,STARTS,IAR,COFMAX,ICOF)
C**********************
C
C
        VIRTUAL FI(1)
        COMMON /COEFS/ AI(128),AQ(128),WORKI(256),WORKQ(256)
        COMMON /INSTR/A(1)
C
C
           NMCOEF=A(25)
           IFINE=A(27)
           MAXSUB=A(29)
           MINSUM=1
            IPAD=A(28)
           NUMBR=IPAD*IFIX(A(26))/NMCOEF
        WRITE(6,456) NUMBR,IPAD,NMCOEF,A(26)
456     FORMAT(3(I6,1X),1PE15.8)
C
C
C
C
C  DO THE COEFFICIENTS   1 TO HALF + 1
C
C
            ICC=NMCOEF/2 + 1
              RUNSUM=STARTS
               ISTART=STARTS
              IEND=ISTART + IFINE - 1
C
C
            COFMAX=0.
            ICOF=0
C
        DO 50 I=1,ICC
C
          IF(ISTART.GT.MAXSUB)  GOTO 60
          REWIND 4
          IG=ISTART - 1
         IF(IG.LT.1) GOTO 10
          DO 5 J=1,IG
5         READ(4)
10       CONTINUE
C
C
C
              IE=IEND
          IF(IE.GT.MAXSUB) IE=MAXSUB
C
              JJ=0
           DO 15 J=ISTART,IE
C
           READ(4) (AI(L),L=1,NMCOEF),(AQ(L),L=1,NMCOEF)
              JJ=JJ + 1
             WORKI(JJ)=AI(I)
            WORKQ(JJ)=AQ(I)
C
```

```
15          CONTINUE
C
C
C
C    PUT IN ZEROS WHERE NO TARGET EXISTS
C
            IF(JJ.EQ.IFINE)  GOTO 20
C
                 JX=JJ + 1
               DO 17 J=JX,IFINE
                  WORKI(J)=0.
                   WORKQ(J)=0.
C
17              CONTINUE
C
C
20          CONTINUE
C
C
C
C     PUT HAMMING WEIGHTING ACROSS SUBARRAYS
C         NULL THE REST OF ARRAY
C         COMPUTE THE FINE STRUCTURE  FFT 2ND
C         FIND THE MAGNITUDE OF THE FFT ALONG LARGEST ,ECT
C
C
                  CALL PADFFT(FI,COFMAX,ICOF,IPAD,IFINE,IAR,NUMBR)
C
C
C
                  RUNSUB=RUNSUB + RMOV
                ISTART=RUNSUB + 0.5
              IEND=ISTART + IFINE - 1
C
C
50        CONTINUE
C
60              CONTINUE
C
C
                  RETURN
                  END
```

```
C
C
C
C   UTSU.FOR
C
C
C
C*********************
                    SUBROUTINE NEGMLP(FI,RMOV,STARTS,IAR,COFMAX,ICOF)
C*********************
C
C
        VIRTUAL FI(1)
        COMMON /COEFS/ AI(128),AQ(128),WORKI(256),WORKQ(256)
        COMMON /INSTR/A(1)
C
           NMCOEF=A(25)
           IFINE=A(27)
           MAXSUB=A(24)
           MINSUB=1
            IPAD=A(28)
        NUMBR=IPAD*IFIX(A(26))/NMCOEF
C
C
C   DO THE NEGATIVE COEFFICIENTS
C
C
                 XX=FLOAT(NMCOEF/2 - 1)*RMOV
                RUNSUB=STARTS - XX
                  XX=ABS(RUNSUB) + 0.5
                IF(RUNSUB.LT.0.) XX=-XX
                   ISTART=XX
                  IEND=ISTART + IFINE - 1
C
C
C
C
C
C
              IQ=NMCOEF/2 + 2
C
           DO 100 I=IQ,NMCOEF
C
             REWIND 4
              IS=ISTART
            IF(IS.LT.1) IS=1
C
C
           IF(IEND.LT.1)  GOTO 60
C
                 JJ=0
            IF(IS.EQ.ISTART)  GOTO 10
               IDIFF=-ISTART + 1
C
               DO 5 J=1,IDIFF
C
                  WORKI(J)=0.
                  WORKQ(J)=0.
C
5              CONTINUE
C
                 JJ=IDIFF
C
10        CONTINUE
```

```
C
C
C
            DO 55 J=IS,IEND
              READ(4) (AI(L),L=1,NMCOEF),(AQ(L),L=1,NMCOEF)
               JJ=JJ + 1
              WORKI(JJ)=AI(I)
              WORKQ(JJ)=AQ(I)
55          CONTINUE
C
              CALL PAUFFT(FI,COFMAX,ICOF,IPAD,IFINE,IAR,NUMBR)
C
C
60           CONTINUE
C
C
C
               RUNSUB=RUNSUB + RMOV
                 XX=ABS(RUNSUB) + 0.5
               IF(RUNSUB.LT.0.) XX=-XX
                ISTART=XX
             IEND=ISTART + IFINE - 1
C
100     CONTINUE
C
C
C
                    RETURN
                   END
```

```
C
C
C
C   DTSC.FOR
C
C
C
C*********************
                      SUBROUTINE PADFFT(FI,COFMAX,ICOF,IPAD,JJ,IAR,NUMBR)
C*********************
C
C
        VIRTUAL FI(1)
        COMMON /COEFS/ AI(128),AQ(128),WORKI(256),WORKQ(256)
        COMMON /HAM/HAM(1)
C
C     PUT HAMMING WEIGHTING ACROSS SUBARRAYS
C         NULL THE REST OF ARRAY
C         COMPUTE THE FINE STRUCTURE  FFT 2ND
C         FIND THE MAGNITUDE OF THE FFT ALONG LARGEST ,ECT
C
C
C
C
C   PUT THE  WEIGHTING ACROSS THE WINDOW
C
        DO 5 J=1,JJ
C
          WORKI(J)=HAM(J)*WORKI(J)
          WORKQ(J)=HAM(J)*WORKQ(J)
C
5       CONTINUE
C
C
C    NULL OUT THE REST OF THE ARRAY
C
C
        IF(IPAD.LE.JJ) GOTO 25
C
              JX=JJ + 1
           DO 20 J=JX,IPAD
              WORKI(J)=0.
              WORKQ(J)=0.
20         CONTINUE
C
25      CONTINUE
C
C
C
C   COMPUTE THE FINE STRUCTURE
C
           CALL FFT(IPAD,WORKI,WORKQ)
C
C
C
C    COMPUTE THE MAGNITUDE OF THE SQUARED
C
C   REORDER THE ARRAY
C
C     SINCE HALF THE FINE COEFFICIENTS ARE TO THE LEFT AND
C        THE OTHER HALF TO THE RIGHT FOR EACH GROSS COEFFICIENT
C        OVER SAMPLING BY FACTOR OF 2 TO 1 THROW OUT HALF THE COEFF
C
C
```



MICROCOPY RESOLUTION TEST CHART
NATIONAL BUREAU OF STANDARDS
STANDARD REFERENCE MATERIAL 1010a
(ANSI and ISO TEST CHART No. 2)

```
      JL=NUMBR/2
      JX=IPAD - JL +1
C
      DO 24 J=1,NUMBR
C
       XX=WORKI(JX)**2 + WORKQ(JX)**2
         JX=JX + 1
        IF(JX.GT.IPAD) JX=1
C
           IAR=IAR + 1
           FI(IAR)=XX
          IF(XX.LT.COFMAX)  GOTO 24
           COFMAX=XX
             ICOF=IAR
C
24    CONTINUE
C
C
C
C
        RETURN
        END
```

C-3

```
C
C
C
C     DICK4.FOR
C
C
C
C************************
C
            SUBROUTINE  OUTPU(A1,A2,A3,A4)
C
C************************
C
          VIRTUAL A1(1),A2(1),A3(1),A4(1)
        COMMON /INSTR/AA(1)
        COMMON /TEMP/TEMP(1)
C
C
C  PRINT OUT THE DATA  SCALE TO THE MAX PASSED IN ARGUMENTS
C
          NRANGE=AA(16)
          NPULSE=TEMP(1)
          IR=TEMP(2)
          IA=TEMP(3)
          XMAX=TEMP(4)
C
        WRITE(5,1000)   IR,IA,XMAX,NPULSE
        WRITE(5,1000)  IR,IA,XMAX,NPULSE
1000    FORMAT(//' RANGE CELL ',I6/' ANGLE ',I6/' VALUE ',1PE15.8,
     1      ' NO. PULSE ',I6//)
C
C
C
C   FIND THE LIMITS OF THE MAINLOBE ALONG THE AZIMUTH SLICE
C
          CALL SLICE(IR,IA,NPULSE,ILF,INT,XMAX,A1)
C
C
C  REORGANIZE THE FILE I/O THE AZIMUITH VERSUS RANGE
C   RATHER THEN IN THE RANGE VERSUS AZIMUTH DIR
C
C       ALSO PUT EM PEAK IN THE MIDDLE OF THE FILE
C
C
          CALL REORD(NPULSE,IA,NRANGE,A1,A2,A3)
C
C
C
C  FIND THE SUM IN THE MAINLOBE AND THE SIDLOBE FOR
C    RANGE AND AZIMUTH DIRECTIONS
C
C
C
          CALL VALUE(NPULSE,IR,IA,ILF,INT,NRANGE,A1,XMAX)
C
C
           RETURN
           END
```

```
C
C
C   DICK4A.FOR
C
C
C
C**********************
C
                      SUBROUTINE SLICE(IR,IA,NPULSE,ILF,IRT,XMAX,A1)
C
C**********************
C
C
        VIRTUAL A1(1)
C
C
C
C   FIND THE LIMITS OF THE MAINLOBE AND THE MAINLOBE
C   SUM ALONG THE ANGLE SLICE
C
C
           REWIND 3
           K=IR - 1
         IF(K.LT.1)  GOTO 2
           DO 1 J=1,K
1          READ(3)
C
2         CONTINUE
C
C
           READ(3) (A1(I),I=1,NPULSE)
C
C
C
C   DO RIGHT SIDE OFF THE PEAK
C
C
            KL=IA
            KR=IA + 1
            SUMR=0.
C
3       CONTINUE
C
        IF(KL.GT.NPULSE) KL=1
        IF(KR.GT.NPULSE) KR=1
C
        IF(A1(KR).GE.A1(KL))  GOTO 4
           SUMR=SUMR + A1(KL)
           KL=KL + 1
           KR=KR + 1
         GOTO 3
4       CONTINUE
C
             IRT=KL
            SUMR=SUMR + A1(KL)
C
C
C   FIND THE LEFT SIDE POINT
C
C
              KR=IA
              KL=IA - 1
             SUML=0.
C
```

```
5       CONTINUE
C
        IF(KR.LT.1)  KR=NPULSE
        IF(KL.LT.1)  KL=NPULSE
C
        IF(A1(KL).GE.A1(KR)) GOTO 6
C
           SUML=SUML + A1(KR)
           KR=KR - 1
           KL=KL - 1
           GOTO 5
C
C
6       CONTINUE
C
           ILF=KR
            SUML=SUML + A1(KR)
        SUMAZ=SUML + SUMR = A1(IA)
C
C
C  FIND THE SIDELOBE
C
C
          TOTL=0.
        DO 207 I=1,NPULSE
         XX=A1(I)
207     TOTL=TOTL + XX
C
C
C
C
             TOTSL=0.
              PEAKSL=0.
             IPEAK=0
C
C
C
        IF(ILF.GT.IRT) GOTO 209
        IF(ILF.LT.2)  GOTO 209
            K=ILF - 1
        DO 208 I=1,K
C
            TOTSL=TOTSL + A1(I)
          IF(A1(I).LT.PEAKSL) GOTO 208
              PEAKSL=A1(I)
               IPEAK=I
C
208     CONTINUE
C
209     CONTINUE
C
           K=IRT + 1
C
          KK=NPULSE
        IF(ILF.GT.IRT) KK=ILF -1
          DO 210 I=K,KK
             TOTSL=TOTSL + A1(I)
            IF(A1(I).LT.PEAKSL) GOTO 210
                PEAKSL=A1(I)
                 IPEAK=I
C
210       CONTINUE
C
C
C
C
```

```
      RES=TOTL - SUMAZ
C
C
C
          DBA=0.
       IF(SUMAZ.GT.0.) DBA=10.*ALOG10(RES/SUMAZ)
          DBPA=0.
       IF(XMAX.GT.0.)  DBPA=10.*ALOG10(RES/XMAX)
C
C
C
      WRITE(6,218) ILF,IA,INT,SUML,SUMR,SUMAZ,TOTL,
     1                 RES,DBA,DBPA
      WRITE(5,218) ILF,IA,INT,SUML,SUMR,SUMAZ,TOTL,
     1                 RES,DBA,DBPA
C
218   FORMAT(' MAINLOBE OF ANGLE '// LEFT POINT ',I5/
     1 ' MID POINT ',I5/' RHT POINT ',I5/' SUM LEFT ',1PE15.8/
     2 ' SUM RIHT ',1PE15.8/' MAINLOBE SUM ',1PE15.8/
     3 ' TOTAL SUM ',1PE15.8/' SIDELOBE SUM ',1PE15.8/
     4 ' SIDELOBE TO MAINLOBE ( DB ) ',1PE15.8/
     5 ' SIDELOBE TO PEAK  ( DB ) ',1PE15.8//)
C
C
C
C
          DB=0.
       IF(SUMAZ.GT.0.) DB=10.*ALOG10(TOTSL/SUMAZ)
          DBSL=0.
       IF(XMAX.GT.0.)  DBSL=10.*ALOG10(PEAKSL/XMAX)
C
      WRITE(6,220)   SUMAZ,TOTSL,PEAKSL,IPEAK,XMAX,IA,
     1                 DB,DBSL
      WRITE(5,220)   SUMAZ,TOTSL,PEAKSL,IPEAK,XMAX,IA,
     1                 DB,DBSL
220   FORMAT(//' ANGLE SUM ',1PE15.8,' MAIN SUM ',1PE15.8/
     1          ' PEAK SIDE LOBE ',1PE15.8,' LOCATION ',I6/
     2          ' MAXIMUM VALUE ',1PE15.8,' LOCATION ',I6/
     3          ' INTEGRATED  (DB) ',1PE15.8/
     4          ' PSL/PEAK (DB) ',1PE15.8//)
C
C
C
C
          RETURN
           END
```

```
C
C
C   DICK80.FOR
C
C
C
C**********************
C
              SUBROUTINE REORD(NPULSE,IA,NRANGE,A1,A2,A3)
C
C**********************
C
C
         VIRTUAL A1(1),A2(20,200),A3(20,200)
C
C
C   REORGANIZE THE FILE IN AZIMUTH VERSUS RANGE
C     RATHER THAN RANGE AZIMUTH
C    PUT THE PEAK IN THE MIDDLE OF THE FILE IN AZIMUTH DIRECTION
C
C
C
              REWIND 4
C
               IMID=NPULSE/2
               ISTART=IA - IMID
             IF(ISTART.LT.1)  ISTART=ISTART + NPULSE
C
               NPASS=400
               NPAS2=NPASS/2
              NPAS22=NPAS2
C
                    NP=NPULSE/NPASS
                 NOVER=NPULSE - NPASS*NP
                  IF(NP.LT.1)  GOTO 850
C
1           DO 820 J=1,NP
C
                 REWIND 3
                     IS=ISTART
C
               DO 810 I=1,NRANGE
C
                   READ(3) (A1(K),K=1,NPULSE)
                         ISR=IS
C
                   DO 805 K=1,NPAS2
C
                      IF(ISR.GT.NPULSE) ISR=1
                         A2(I,K)=A1(ISR)
                          ISR=ISR + 1
C
805                CONTINUE
C
C
              IF(NPAS22.LT.1)  GOTO 810
C
CC
                    DO 806 K=1,NPAS22
C
                       IF(ISR.GT.NPULSE) ISR=1
                          A3(I,K)=A1(ISR)
                            ISR=ISR + 1
C
```

```
805               CONTINUE
C
810            CONTINUE
C
C
               DO 815 K=1,NPAS2
815             WRITE(4) (A2(I,K),I=1,NRANGE)
C
C
              IF(NPAS22.LT.1)  GOTO 817
C
               DO 816 K=1,NPAS22
816             WRITE(4) (A3(I,K),I=1,NRANGE)
C
C
817              ISTART=ISR
C
C
820        CONTINUE
C
C
850      IF(NOVER.LE.0)  GOTO 140
C
            NP=1
            IF(NOVER.GT.NPAS2) GOTO 135
C
              NPAS2=NOVER
               NPAS22=0
              NOVER=0
             GOTO 1
135      CONTINUE
C
            NPAS22=NOVER - NPAS2
             NOVER=0
             GOTO 1
140      CONTINUE
C
C
             RETURN
              END
```

```
C
C
C   DICK4C.FOR
C
C
C
C*********************
C
              SUBROUTINE VALUE(NPULSE,INANGE,IA,ILF,IRT,NRANGE,A1,XMAX)
C
C
C
C*********************
C
C
        VIRTUAL A1(1)
C
C
               REWIND 4
C
C
               IMID=NPULSE/2
              IDELTA=IA - ILF
            IF(IDELTA.LT.0)  IDELTA=IDELTA + NPULSE
             LIML=IMID - IDELTA + 1
              IDELTA=IRT - IA
            IF(IDELTA.LT.0)   IDELTA=IDELTA + NPULSE
              LIMR=IMID + IDELTA + 1
C
             IPRR=LIMR + 200
             IPRL=LIML - 200
        IF(IPRL.LT.1) IPRL=1
        IF(IPRR.GT.NPULSE) IPRR=NPULSE
C
C
        WRITE(6,7899) IMID,NPULSE,IDELTA,LIML,LIMR
7899    FORMAT(//' VALUE ',6(I4,I7)//)
C
C
C
C
               TOTAL=0.
C
C
C
C
              TOTS=0.
              TOTL=0.
              VSIDE=0.
             IVSR=0
              IVSA=0
C
        DO 120 J=1,NPULSE
C
          READ(4) (A1(I),I=1,NRANGE)
C
C
C   TAG A MAINLOBE WITH IT = 1
C
                 IT=0
              IF((J.GE.LIML).AND.
     1             (J.LE.LIMR))  IT=1
C
C
```

```
              VMAX=0.
C
             DO 112 I=1,NRANGE
C
                 TOTAL=A1(I) + TOTAL
C
112          CONTINUE
C
C
C
                 SUMT=0.
                 SUMLL=0.
C
            IF(IT.EQ.0)  GOTO 118
                IV=IRANGE
                  SUML=0.
               IVL=IV
               IVR=IV + 1
113         CONTINUE
C
C
             IF(IVR.GT.NRANGE)  GOTO 114
            IF(A1(IVR).GT.A1(IVL))  GOTO 114
              SUML=SUML + A1(IVL)
              IVR=IVR + 1
              IVL=IVL + 1
              GOTO 113
C
114           CONTINUE
C
              SUML=SUML + A1(IVL)
C
             IF(IVR.GT.NRANGE) SUML=SUML + A1(NRANGE)
                IVRR=IVL
                SUMR=0.
                IVL=IV - 1
                IVR=IV
115       CONTINUE
             IF(IVL.LT.1)  GOTO 116
             IF(A1(IVL).GT.A1(IVR)) GOTO 116
C
               SUMR=SUMR + A1(IVR)
               IVR=IVR - 1
               IVL=IVL - 1
              GOTO 115
C
116          CONTINUE
C
            SUMR=SUMR + A1(IVR)
C
            IF(IVL.LT.1)  SUMR=SUMR + A1(1)
              IVLL=IVR
C
             SUMT=SUML + SUMR - A1(IV)
              IF(IVLL.LE.1)  GOTO 15
C
                 IVL=IVLL - 1
               DO 10 I=1,IVL
C
                  SUMLL=SUMLL + A1(I)
                IF(A1(I).LT.VSIDE)  GOTO 10
C
                   VSIDE=A1(I)
                   IVSR=I
                   IVSA=J
10             CONTINUE
```

```
C
15          CONTINUE
C
            IF(IVRH.GE.NRANGE)  GOTO 20
C
              IVR=IVRH + 1
              DO 17 I=IVR,NRANGE
C
                 SUMLL=SUMLL + A1(I)
                IF(A1(I).LT.VSIDE)  GOTO 17
                  VSIDE=A1(I)
                  IVSH=I
                  IVSA=J
C
17            CONTINUE
C
C
20          CONTINUE
            GOTO 119
C
118       CONTINUE
C
C
            DO 30 I=1,NRANGE
C
              SUMLL=SUMLL + A1(I)
              IF(A1(I).LT.VSIDE)  GOTO 30
C
                VSIDE=A1(I)
                IVSH=I
                IVSA=J
C
30          CONTINUE
C
C
119         CONTINUE
C
C
             TOTS=TOTS + SUMT
             TOTL=TOTL + SUMLL
C
C
C          PRINT OUT THE MAINLOBE AREA
C
C
                IF((J.LT.IPRL).OR.(J.GT.IPRR))  GOTO 92
C
             DO 89 I=1,NRANGE
C
                VAL=+999.
                XX=A1(I)
                 XX=XX/XMAX
               IF(XX.GT.0.) VAL=-10.*ALOG10(XX)
                A1(I)=VAL
C
89          CONTINUE
C
          WRITE(3,8081)  J,(A1(I),I=1,NRANGE)
8081      FORMAT(I6,10(1X,F7.2))
C
92        CONTINUE
C
C
C
C
120       CONTINUE
```

```
C
            DB=0.
           IF(TOTL.GT.0.) DB=10.*ALOG10(TOTL/TOTS)
            DBP=0.
           IF(TOTL.GT.0.) DBP=10.*ALOG10(TOTL/XMAX)
C
            TOTSL=TOTS + TOTL
C
C
C
      WRITE(5,8080)  TOTAL,TOTSL,TOTS,TOTL,DB,DBP
      WRITE(6,8080)  TOTAL,TOTSL,TOTS,TOTL,DB,DBP
8080  FORMAT(///'  TOTAL IN RANGE AND ANGLE ',1PE15.8/
     1   ' TOTAL IN RANGE AND ANGLE SUB ',1PE15.8/
     2   ' TOTAL OF SUM ',1PE15.8/
     3   ' TOTAL OF SIDE ',1PE15.8/
     4   '  SIDE/SUM (DB) ',1PE15.8/
     5    ' SIDE/PEAK (DB) ',1PE15.8//)
C
C
C
C
              VSDB=VSIDE/XMAX
            IF(VSDB.GT.0.)  VSDB=10.*ALOG10(VSDB)
C
          WRITE(5,8082)  IVSR,IVSA,VSIDE,VSDB
           WRITE(6,8082) IVSR,IVSA,VSIDE,VSDB
C
8082  FORMAT(///'  PEAK SIDE LOBE LEVEL '/
     1          'LOCATION IN RANGE ',I4,' LOCATION IN AZI ',I6/
     2           '  MAGNITUDE ',1PE15.8,' ( DB ) ',1PE15.8//)
C
C
C
C
C
            RETURN
            END
```

```
C*
C*      FFT.FOR
C*
C*
C*    FFT ALGORITHM
C*
C*
        SUBROUTINE FFT(NUMBR,XT,YT)
        DIMENSION XT(1),YT(1)
C
C   NUMBR IS THE NO. OF POINTS OF GOOD DATA
C   IF NOT A POWER OF TWO PAD TO NEXT HIGHEST
C   WITH ZEROS
C
C
          N2POW=1
          ITEMP=NUMBR
1         ITEMP=ITEMP/2
         IF(ITEMP.LE.1)  GOTO 10
          N2POW=N2POW + 1
          GOTO 1
10        CONTINUE
C
        N=2**N2POW
        IF(N.GE.NUMBR) GOTO 13
          N=N*2
          N2POW=N2POW + 1
13      CONTINUE
        IF(NUMBR.GE.N)  GOTO 20
C
           N1=NUMBR + 1
        DO 15 LM=N1,N
C
          XT(LM)=0.
          YT(LM)=0.
C
15      CONTINUE
C
C
20      CONTINUE
C
C
        M=N2POW
        DO 600 L0=1,M
        LMX=2**(M-L0)
        LIX=2*LMX
        SCL=6.283185/FLOAT(LIX)
        DO 600 LM=1,LMX
        ARG=(LM-1)*SCL
        C=COS(ARG)
        S=SIN(ARG)
        DO 600 LI=LIX,N,LIX
        J1=LI-LIX+LM
        J2=J1+LMX
        T1=XT(J1)-XT(J2)
        T2=YT(J1)-YT(J2)
        XT(J1)=XT(J1)+XT(J2)
        YT(J1)=YT(J1)+YT(J2)
        XT(J2)=C*T1+S*T2
        YT(J2)=C*T2-S*T1
600     CONTINUE
        NV2=N/2
        NM1=N-1
        J=1
```

```
      DO 635 I=1,NM1
      IF(I.GE.J) GO TO 631
      T1=XT(J)
      T2=YT(J)
      XT(J)=XT(I)
      YT(J)=YT(I)
      XT(I)=T1
      YT(I)=T2
631   K=NV2
620   CONTINUE
      IF(K.GT.J) GO TO 635
      J=J-K
      K=K/2
      GO TO 620
635   J=J+K
      NUMBR=N
      RETURN
      END
```

```
C
C
C
C     WT.FOR
C
C
C     WINDOWING FUNCTION
C
C
C      ARGUMENTS
C
C
C       ARRAY IS THE ARRAY TO STORE THE WEIGHTS IN
C        NUMB IS THE NO. OF SAMPLING ELEMENTS IN 2*PI
C
C
C           IOPT  IS TYPE OF WINDOW
C
C
C       0 NO WINDOWING ( RECT FUNCTION )
C       1  HAMMING WINDOW
C       2 BARTLETT WINDOW    (TRIANGLE)
C       3 HANNING WINDOW
C       4 BLACKMAN WINDOW
C       5 25 DB TAYLOR WINDOW
C       6 30 DB TAYLOR WINDOW
C       7 35 DB TAYLOR WINDOW
C
C
C
C*********************************
C
                SUBROUTINE WT(ARRAY,NUMBR,IOPT)
C
C*********************************
C
        REAL ARRAY(1)
C
C
        IF(IOPT.EQ.0)  GOTO 600
           XN=NUMBR - 1
        IF(IOPT.EQ.2)  GOTO 200
C
           PI=4.*ATAN(1.)
          PI2=2.*PI
          PHASE=PI2/XN
        IF(IOPT.EQ.4)  GOTO 100
        IF(IOPT.EQ.5) GOTO 300
        IF(IOPT.EQ.6) GOTO 400
        IF(IOPT.EQ.7) GOTO 500
C
          C1=0.5
          C2=0.5
        IF(IOPT.EQ.3)  GOTO 10
C
          C1=0.54
           C2=0.46
C
10      CONTINUE
C
C
        DO 20 I=1,NUMBR
C
          XN=I - 1
```

```
          WONC1 = C2*COS(PHASE*XN)
          ANRAY(I)*WO
C
20        CONTINUE
C
              GOTO 10000
C
100       CONTINUE
C
          PHASE2=PHASE*2.
C
          DO 120 I=1,NUMBR
C
            XN=I - 1
            WO=0.42 - 0.50*COS(PHASE*XN) + 0.08*COS(PHASE2*XN)
            ANRAY(I)*WO
C
120       CONTINUE
C
              GOTO 10000
C
C       NUMBR IS THE NUMBER OF ELEMENTS IN THE WINDOW STARTING
C         FROM THE FIRST ADDRESS OF ANRAY
C
C
200       CONTINUE
C
              IUP=NUMBR/2
C
          DO 210 I=1,IUP
C
            WO=FLOAT(2*(I - 1))/XN
            ANRAY(I)*WO
C
210       CONTINUE
C
              IUP=IUP + 1
C
          DO 220 I=IUP,NUMBR
C
            WO=2. - FLOAT(2*(I - 1))/XN
            ANRAY(I)*WO
C
220       CONTINUE
C
C
              GOTO 10000
300       CONTINUE
C
C
C  25 DB TAYLOR WEIGHTING
C
C
          START=-PI
C
          DO 310 I=1,NUMBR
C
          WO=0.221477*COS(START) - 0.009370*COS(2.*START) +
     1       0.046621*COS(3.*START) + 0.004927*COS(4.*START)
C
            ANRAY(I)=1. + 2.*WO
          START=START + PHASE
C
310       CONTINUE
C
            GOTO 10000
```

```
400      CONTINUE
C
C
C   30 DB TAYLOR WEIGHTING
C
         START=-PI
         DO 410 I=1,NUMBR
C
         WD=0.292696*COS(START) - 0.0157838*COS(2.*START) +
     1       0.00210104*COS(3.*START)
         ANRAY(I)=1. + 2*WD
         START=START + PHASE
C
410      CONTINUE
C
            GOTO 10000
500      CONTINUE
C
C
C    35 DB TAYLOR WEIGHTING
C
C
         START=-PI
         DO 510 I=1,NUMBR
C
         WD=0.344350*COS(START) - 0.0191940*COS(2.*START) +
     1       0.00427831*COS(3.*START) - 0.000734551*COS(4.*START)
         ANRAY(I)=1. + 2*WD
         START=START + PHASE
C
510      CONTINUE
C
C
             GOTO 10000
600      CONTINUE
C
C   UNIFORM WEIGHTING
C
         DO 610 I=1,NUMBR
C
           ANRAY(I)=1.
C
610      CONTINUE
10000       CONTINUE
C
C
               RETURN
                END
```