SYNTHETIC APERTURE RADAR SIGNAL PROCESSING ON THE MPP

H. K. Ramapriyan
Space Data and Computing Division
NASA Goddard Space Flight Center
Greenbelt, Maryland 20771

E. J. Seiler
Science Applications Research, Inc.
Lanham, Maryland 20706

ABSTRACT

Satellite-borne Synthetic Aperture Radars (SAR) sense areas of several thousand square kilometers in few seconds and transmit phase-history signal data at the rate of several tens of megabits per second. For example, the Seasat SAR acquired data for a 100 km by 100 km area in about 18 seconds and transmitted at the rate of 85 Mbits/sec. The Shuttle Imaging Radar - B (SIR-B) has a variable swath of 20 to 50 km and acquired data over 100 kms along track in about 13 seconds. The transmission rates used were 30.4 and 45.6 Mbits/sec. The processing of the phase-history data into images with pixels representing the radar reflectances involves correlation with the two-dimensional point target response (i.e., the reference function) which is typically more than 1200 samples in each dimension. Even with the simplifying assumption of separability of the reference function, the processing requires considerable resources -- high-speed I/O, large memory and fast computation.

Processing systems with conventional hardware take several hours to process one Seasat image and about one hour for a SIR-B image. Bringing this processing time closer to acquisition times requires an "end-to-end" system solution. However, for the purposes of demonstration, we have implemented software on the present MPP configuration (with conventional I/O hardware on the VAX 11/780 host computer) for processing Seasat and SIR-B data. The software takes advantage of the high processing speed offered by the MPP, the large (32 Mbyte) Staging Buffer, and the high-speed I/O between the MPP array unit and the Staging Buffer. It is found that with unoptimized Parallel Pascal code, the processing time on the MPP for a 4096 x 4096 sample subset of signal data ranges between 18 and 30.2 seconds depending on options.

INTRODUCTION

Satellite-borne synthetic Aperture Radars (SAR's) sense areas of several thousand kilometers in a few seconds and transmit the phase-history signal data at the rate of several tens of megabits per second. The first space borne SAR was on the Seasat which was launched in June 1978. It acquired the data needed to generate a 100 km by 100 km image in about 18 seconds and transmitted at 85 Mbits/sec. The Shuttle Imaging Radar-B (SIR-B) flown aboard the Challenger during October 1984 with a selectable incidence angle capability had a variable swath of 20 to 50 km and acquired data needed for an image covering 100 km along track in about 13 seconds. The transmission rates used were 30.4 and

45.6 Mbits/sec. The space-borne radars planned for the future, with multiple frequencies, polarizations and incidence angles, will have similar or higher data rates. The processing of the phase-history data into images with pixels representing the radar reflectances involves correlation with a two-dimensional point target response (i.e., a reference function). In the case of Seasat SAR, this function extends over 1536 x 4200 samples. In the case of SIR-B, it is approximately 1200 x 1500 samples. Fortunately, it is possible to treat this two-dimensional correlation as a separable problem to a very close approximation and implement it as two one-dimensional correlations. Even so, processing the data in a reasonable time requires considerable resources: high-speed input-output, large memory and fast computation.

Processing algorithms for space-borne SAR's differ from those for air-borne SAR's due to the much larger range walk and range curvature corrections required for the former. Algorithms for space-borne SAR's have received considerable attention since 1978. Descriptions of processing algorithms can be found in papers by Cumming and Bennet (1979), Wu (1980) and Wu et al (1982). The execution of these algorithms on systems with conventional hardware (minicomputers, array processors, computer compatible tapes and disks) takes several hours for one Seasat image and about one hour for a SIR-B image. Bringing this processing time closer to acquisition times requires an "end-to-end" system solution. However, for the purposes of demonstration, we have implemented the processing algorithms on the present configuration of the Massively Parallel Processor (MPP) at the Goddard Space Flight Center (GSFC). A discussion of the MPP hardware and the host configuration system can be found in companion paper (Fischer, this volume). This implementation takes advantage of the high processing speed of the MPP, the large (32 Mbyte) Staging Buffer and the high-speed I/O between the MPP array unit and the Staging Buffer.

The purpose of this paper is to present the MPP implementation of the SAR processing algorithms and comments on the timings achieved.

2. MPP IMPLEMENTATION OF THE PROCESSING ALGORITHM

All signal processors must address, in different ways unique to the design goals and base processor hardware capabilities, the same basic processing functions, and compensate or correct for conditions such as range walk and range curvature. The basic steps in image generation can be categorized as preparation, signal processing and postprocessing as shown below:

Preparation

- Raw data extraction and reformatting
- Range Reference Function (RRF) generation
- Doppler parameter estimation for use in Range Walk Correction (RWC) and the Azimuth Reference Function (ARF)
- Frequency domain break-point calculation to facilitate range curvature correction
- Generation of trigonometric constants for use in the Fast Fourier Transform (FFT) algorithm.

Signal Processing

- Range compression
- Corner Turning (Transposition)
- Azimuth Compression
- Image Formation

Postprocessing

- Range Attenuation Compensation (RAC)
- Geometric Correction (conversion from slant range to ground range and rectification to a map projection).

The raw data extraction and reformatting consist of converting the received radar signal data from the collection medium (High Density Digital Tape - HDDT) to a medium acceptable to the processing system. Since the present configuration of the MPP and its host computer does not contain a reader for HDDT's, it is necessary to start processing on this system from a Computer Compatible Tape (CCT) or disk-file. Seasat and SIR-B signal data are available from Jet Propulsion Laboratory (JPL) on CCT's with records containing packed data. On these CCT's, Seasat data contain 4 bits per sample and the SIR-B data contain 3 to 6 bits per sample depending on the parameters selected for a given imaging interval (data take). For processing, the signal data records are unpacked into 4 or 8 bits per sample before reading into the staging buffer. The remaining preparation and signal processing steps are based on algorithms described by Wu (1982). The details of those algorithms are beyond the scope of this paper. However, the equations needed to maintain continuity of presentation and a discussion of the MPP implementation are given below. The post-processing steps of RAC and geometric correction are not considered here.

2.1 Preparation

2.1.1 Range Reference Function (RRF) Generation:

The RRF is the function used for range compression. It consists of a dechirp function (filter to match the transmitted chirp pulse) and a window function to reduce sidelobes of the matched filter response. The function is given by

$$c(t) = \cos 2\pi[(f_0 + B/2)t - \phi(t)]$$

for $0 < t < T$
$= 0$ elsewhere

where f_0 = range offset frequency (in Hz)

B = bandwidth of the transmitted pulse (in Hz)

$\phi(t) = 1/2\ Kt^2$

K = range frequency modulation (FM) rate (in Hz/sec)

$= B/T$

and T = pulsewidth (in seconds)

The values of f_0, B and T are shown below for Seasat and SIR-B.

	SIR-B	SEASAT
Range Offset Frequency, MHz	7.2	11.3825
Bandwidth, MHz	12.0	19.0
Pulsewidth, microsecs	30.4	33.8

Frequency domain weighting is used for sidelobe reduction. Several options exist for this. Hanning and Taylor window weighting have been used by JPL (personal communication, 1983) and Applied Physics Laboratory (APL) (Raff, 1983) respectively. A review of windowing and its effects on processed signals can be found in Harris (1978).

In our implementation we have used the Kaiser-Bessel window. This window provides a better combination of sidelobe suppression and compressed pulse width than the Hanning Window. If the window function is $B(\omega)$, then the RRF in the frequency domain is given by $C(\omega)^*B(\omega)$ where $C(\omega)^*$ is the complex conjugate of the Fourier transform of c(t).

To obtain the RRF in the discrete frequency domain, we embed the samples of c(t) in an N-element long complex array for some $N > T/f_s$ = signal sampling frequency), take its Discrete Fourier Transform (DFT), and multiply its complex conjugate by a sampled version of $B(\omega)$. The choice of N is a very important sizing consideration in the processor design. In our implementation, N=4096 for the reasons indicated below in subsection 2.2. In addition, for SIR-B, a notch filter is used to suppress the calibration tone. This is done by zeroing out the value of $C(\omega)^*B(\omega)$ for corresponding to the frequency of the calibration tone.

2.1.2 Doppler Parameter Computation:

The Doppler parameters are needed for computing the RWC and the ARF. Even though in our implementation the RWC occurs during range compression (and hence before azimuth compression), it is more convenient to define the ARF first and then discuss the parameters for RWC.

The ARF is given by (Wu, 1982)

$$h_1(x,r,r_0) = \sum_{i=1}^{n} g_i(x,r_0)\ \delta(r-d_i)$$

which is the range compressed response due to a point target located at $(0,r_0)$ where (x,r) are the (azimuth, range) coordinates at which h_1 is evaluated, $g_i(x,r_0)$ is the azimuth response function evaluated along a range bin at distance d_i from r_0, and is the Dirac delta function. Further,

$$g_i(x,r_0) = a_i(x)\ \exp\left[j\,\psi(x,r_0)\right]$$

where

$$a_i(x) = W_a(x)\ q\ (d_i - r_1(x))$$

$$\psi(x,r_0) = 4\pi r_1(x)/\lambda$$

$W_a(x)$ = Antenna weighting function in azimuth

$q(r)$ = range compressed point target response

$r_1(x)$ = Slant range from the satellite to the point target

λ = Wavelength of the transmitted electromagentic wave

Note that in the above equations, x=0 is the (arbitrary) origin corresponding to the time when the antenna beam center (in azimuth) passes the point target.

It is convenient to express $r_1(x)$ and $\Psi(x,r_o)$ in terms of the Doppler frequency and the Doppler rate induced by the relative motion between the spacecraft and the target. It is seen that when the relative acceleration is nearly constant,

$$\Psi(t) = \Psi(0) - 2\pi(f_d t + 1/2\ \dot{f}_d t^2)$$

where t is used as an independent variable proportional to x,

$$\Psi(0) = 4\pi r_o/\lambda$$

$$f_d = -2\ R.V/(\lambda r_o)$$

$$\dot{f}_d = -2\ (r.A + V^2)/(\lambda r_o)$$

and R, V, and A are respectively the relative displacement, velocity and acceleration vectors of the target with respect to the spacecraft at t=0. (Note that r_o is the magnitude of R).

An implementation consideration is to ensure that the number of range bins spanned by the range compressed data from a given target is kept to a minimum. This is accomplished by separating $\Psi(t)$ into two parts:

$$\Psi(t) = \Psi_1(t) + \Psi_2(t) \quad \text{where}$$

$$\Psi_1(t) = -2f_{d0}t$$

$$\text{and } \Psi_2(t) = \Psi(0) - 2[(f_d - f_{d0})\ t + 1/2\ \dot{f}_d t^2]$$

where f_{do} is a "nominal" doppler frequency (computed at the center of the image being processed). The function $\Psi_1(t)$ is merged into range compression processing to perform range walk correction which is equivalent to shifting each iso-azimuth line by a number of range bins proportional to $f_{do}t$.

It is important to determine f_d and $\dot{f}_d$ accurately, since the quality of azimuth compressed data is critically dependent on them. The values of f_d and $\dot{f}_d$ depend on the ephemeris, attitude, antenna pointing direction and position of the target (and therefore on the particular range bin of interest).

When the ephemeris and attitude data are known accurately (assuming known antenna pointing direction with respect to the spacecraft body axes), f_d and $\dot{f}_d$ can be computed from them.

When the ephemeris and attitude data are not known accurately, the signal data themselves are used for computing f_d and $\dot{f}_d$. The computation of f_d uses a clutter lock procedure. The $\dot{f}_d$ values are computed using an autofocussing technique.

Clutter lock: The clutter lock algorithm is based on the principle that in regions not dominated by strong point targets, the averaged magnitude of the azimuth frequency response matches a shifted aperture gain function. The shift in the aperture gain function required to match the above frequency response is the doppler frequency f_d. Since f_d is a function of range, it is necessary to determine this shift for

several sets of iso-range lines. In our implementation, group of 16 contiguous iso-range lines are used at intervals of 64 lines, starting from the nearest range-bin. The procedure used is as follows. The range offset frequency is first removed from the data. The data are transposed for convenient access of iso-range lines. Sets of 16 lines (4096 samples each) are then Fourier transformed and the averages of the Discrete Fourier Transform (DFT) magnitudes are found. This averaged DFT magnitude array is correlated with the aperture gain function f(a) given by

$$f(a) = (\sin a/a)^2 \text{ where}$$

$$a = 2\pi k/4096 \text{ for } k = -2047, -2046, \ldots, 2048, k \neq 0$$

and $f(0)=1$.

Now, f_d is given by

$$f_d = PRF\ (1 + K/4096)$$

where PRF = Pulse Repetition Frequency

and K = sample number at which the correlation peak occurred (ranging from -2047 to 2048).

After f_d is computed for each selected set of 16 iso-range lines it is assigned to the mid-range value of that set. Then, a least-squares fit is performed using the f_d values for all the selected sets to obtain f_d as a function of range r:

$f_d(r) = f_{d0} + mr$, where m is the slope determined by the fit

The mid-range value of f_d is used for range-walk correction during the range compression process.

Autofocus: The autofocus algorithm is based on the principle that if the f_d value is correct, the azimuth compression will result in the best focused point target response. Thus, if the f_d value is correct, each look of multiple-look processed azimuth compressed data will place image-features at the same location. However, a small error in f_d will result in shifts among the images from the multiple looks. By determining the shifts, the corrections to f_d can be computed. For this method to succeed, it is necessary to have an initial estimate of f_d which is close enough to the correct value to result in some recognizable features in the multi-look processed images. The initial f_d values are estimated using the ephemeris and attitude data. In our implementation, the two looks closest to the center of the azimuth reference function are used to produce two images. These two images are then correlated to determine the shift between them. The shift is used to correct f_d and the process is repeated. This procedure is used on several sections of the image at various range values and f_d is computed as a linear function of range through a least squares fit.

2.1.3. Frequency Domain Break-Point Calculation:

The purpose of this preparatory step is to precompute some of the parameters needed for manipulating frequency domain data during azimuth compression. Wu (1982) has shown that the Fourier transform of the desired signal after azimuth compression can be written as

$$\Omega(u,r_o) = \sum_{i=1}^{n} [S(u,d_i + r_o) A(u,r_o)] H(u,r_o)$$

where

u is the azimuth frequency variable

r_o is the range bin of interest

n is the number of range bins over which the range compressed point target response is spread

d_i is the distance from r_0 to the i^{th} range bin used in range curvature correction

S is the Fourier transform of the range compressed signal and

A_i and H are, respectively, the complex conjugates of the Fourier transforms of a_i (x) and $\exp[j\Psi(x,r_0)]$ defined in subsection 2.1.2.

Two commonly used approaches to approximating the summation in the above equation are nearest neighbor assignment and cubic interpolation. Conceptually, these approximations are treated as follows.

Let i_0 be the (possibly fractional) value of i for which A_i (u,r_0) is a maximum. Note that i is a function of (u,r_0). Let I be the integer nearest to i_0. Let I_0 be the largest integer less than or equal to i_0. Then, the nearest neighbor assignment is equivalent to replacing A_i (u,r_0) by $\delta(i-I)$. Cubic interpolation is performed using the samples of S corresponding to i in the range I_0-1 through I_0+2. (That is, the A_i have suitable "weight" values for i in this range and are zero elsewhere).

Since the range curvature depends only on f_d, the values of i_0 for the various u can be found from f_d and $\dot{f}_d$. Therefore, for each set of f_d, $\dot{f}_d$ indicated in the avoce subsection, we can precompute i_0 as a function of u. For nearest neighbor assignment, which is our present implementation, it is most convenient to store I in terms of the values of u at which it changes. These are called the frequency domain break-points. These break-points are computed during this preparatory step and are used for generating the "masks" required for range curvature correction (see subsection 2.2.3).

2.2 SIGNAL PROCESSING

The signal processing primarily involves the computation of correlations between the recorded signal and the reference functions in the range and azimuth directions. These correlations are performed most efficiently as multiplications of the functions after a transformation to the frequency domain, followed by an inverse transformation to return to the time domain. Since this is the most computationally intensive portion of the processing, it is important to use a Fast Fourier Transform (FFT) algorithm which takes full advantage of the capabilities of the MPP.

The FFT design has been chosen to maximize its efficiency subject to the MPP hardware constraints such as the 1024-bit array memory per processing element (PE) and the 32 Mbyte staging buffer capacity. One of the design constraints is that it should be possible to transpose the range compressed data within the 32 Mbyte staging buffer prior to azimuth compression so that the low-speed data transfers between the MPP and the disk (or tape) devices are minimized. This requires that the signal processing be carried out on blocks of raw data with MN samples where M and N, the numbers of samples in the range and azimuth directions, respectively are chosen such that the results of range compression for the entire block can be held in the staging buffer. To minimize the computation time, the trigonometric constants needed for the FFT are precomputed and stored in the array memory. To eliminate the need for reloading trigonometric constants between range and azimuth compression, it is necessary to choose M = N. Further, the number of correct output values produced after

correlation processing is N - RFL + 1 where the RFL is the number of samples in the reference function. Therefore, it is desirable to use as large a value of N as possible. These considerations lead to the selection of N = 4096.

In general, algorithms on the MPP are most efficient when all the PE's are kept busy performing "useful work" and the data transfers among PE's are minimized. (Note that the data transfer time can be comparable to computation time. Shifting a K-bit operand by x PE's requires (x+2)K clock cycles, while the addition of two K-bit operands requires 3K cycles). In the case of the FFT, the maximum efficiency is achieved when all the PE's are performing complex multiplications or additions and there is no data shuffling among the PE's. This requires that all of the data in an array to be transformed be resident in the memory of a PE. (In that case, 16,384 FFT's could be performed simultaneously). However, due to the limitation of 1024 bits of memory per PE, it is necessary to distribute each of the arrays to be transformed over several PE's. If each of the arrays to be transformed has N samples and is distributed over a rectangular subset of PE's of dimensions A x B, it is necessary to store C = N/(A x B) samples per PE. It is easy to see that the number of shift operations needed per FFT is of the order of (A + B). Thus the data transfers are minimized if A + B is minimized and C is as large as possible. The value of C is limited by the available array memory. For a given C, the value of A + B is minimized by selecting A = B. For the case N = 4096, A = B = 32 and C = 4 are chosen to provide the optimum data arrangement. This allows 16 FFT's to be performed simultaneously on the MPP.

The MPP is most efficient when performing integer arithmetic; this speed decreases as the length of the operands increases. Another consideration is that the transfer rate of data between the MPP and the host system is severely limited by the available hardware configuration. Thus in order to minimize the computation and transfer times the size of the operands should be chosen as small as possible while preserving the accuracy of the computations. For the SAR processing, the raw signal input is 3 to 6 bits per sample, and the processed output image is generally desired as 8 bits per pixel. Experimentation has shown that 16-bit integer calculations are sufficient to preserve full accuracy at the intermediate stages of computation.

2.2.1 Range Compression

The range compression is performed by reading the raw signal data in groups of 16 lines, performing the FFT, multiplying by the precomputed RRF performing range walk correction by multiplying by the walk reference function, performing the inverse FFT, basebanding the data, and outputting the resultant complex data.

The range walk correction is designed to minimize the number of range bins over which a point target's response extends during azimuth processing. The walk reference function is computed as described in section 2.1.2, using the value of f_d at the middle range value of the image determined by the clutterlock processing. The multiplication by the walk reference function results in a shift of each successive range line by a (possibly fractional) number of range bins.

In most signal processors the range offset frequency f_o is eliminated in a basebanding operation that shifts frequency domain data down in frequency by f_o. On the MPP, basebanding is accomplished after

restoring the data to the time domain by multiplying the data by a complex operand to shift it in frequency. This method, requiring a single complex multiplication per sample, is more efficient than the time consuming data shuffling that would be necessary to shift the data in the frequency domain.

The resulting data are subsampled to obtain 2048 complex samples per line. Then, all 16 lines of range compressed data are reduced to 16-bits per sample (8-bits real and 8-bits imaginary) and transferred to the staging buffer simultaneously.

2.2.2 Corner Turning

The corner turning is performed by writing the 4096 x 2048 complex 8-bit data to the MPP staging buffer, and reading the data into the array memory in transposed order. The transposition is performed under the direction of a Stager Control Block (SCB) file, constructed to direct the staging buffer to use row-major order for input and column-major order for output of the 2-dimensional array. The staging buffer has a capacity of 32 Mbytes and a theoretical I/O rate of 80 Mbytes/sec (overhead considerations which reduce the achievable rate are discussed in section 3). Previously the staging buffer had a 2 Mbyte capacity and an I/O rate of 20 Mbytes/sec. This smaller capacity made it necessary to perform the corner turning by writing partially transposed data to a disk file, read the data back to the staging buffer using random access I/O, and complete the transposition in the staging buffer. The disk I/O has a theoretical rate of 1.2 Mbytes/sec, while the achievable rate depends largely on the system load at the time of processing. Thus at least a 67 fold increase in speed is obtainable by using the current staging buffer configuration. Further, at present, the data transfers between the staging buffer and host files require intervention of the host. If the transfers between the array unit and the staging buffer can be performed without host intervention the speed increase becomes much greater.

2.2.3 Azimuth Compression

The azimuth compression is performed on the MPP on a group of data referred to as an Azimuth Processing Blocks (APB's), consisting of 32 lines by 4096 azimuth values per line of range compressed data. The azimuth compression produces (4097-ARFL) "correct" data values per line, where ARFL is the length of the Azimuth Reference Function. Therefore, for single-look processing the APBs are overlapped by (ARFL-1) azimuth values to process all azimuth values.

For each APB an ARF is computed in terms of f_d and $\dot{f}_d$ determined by the clutterlock and autofocus algorithms as discussed in section 2.1.2. After the ARF is computed its Fourier transform is computed and stored.

The range compressed data are read 16 lines (range bins) at a time and the Fourier transformed using the same data arrangement as was used during range compression. Prior to multiplication by the Fourier transformed ARF's it is necessary to perform range curvature correction to assure that all the data corresponding to a given slant range occur in the same range bin. To accomplish this the data are first rearranged so that the data from the 16 range bins corresponding to a given azimuth occur in the same PE. The frequency domain break-points computed as in section 2.1.3 are used to determine the number of range bins by which each sample is to be shifted. The shifts (movement of data within each PE by a known integral number of range bins) are accomplished using masked assignment

operations. The data are then rearranged in the "FFT order", multiplied by the Fourier transforms of the ARF's and inverse transformed.

2.2.4 Image Formation

The inverse transformed data above are complex. The magnitudes of these data constitute the fully correlated image result. The image values so obtained are accumulated in the staging buffer and written to disk as blocks of (2049 - RRFL/2) lines with (4097 - ARFL) pixels per line where RRFL and ARFL are the lengths of the range and azimuth reference functions, respectively.

3. RESULTS AND CONCLUSION

An example of a SIR-B image generated using the above MPP implementation is shown in figure 1. This image covers a region surrounding Mount Shasta, California. It represents the results of processing 4 blocks of signal data, each with 4096 x 4096 samples. The blocks are overlapped by ARFL = 2031 samples in the azimuth direction. The signal data are first singlelook processed to form image data as discussed above in section 2.2. Successive pixels in the azimuth direction are then averaged in groups of four to reduce speckle and approximately match the slant range pixel size.

The times required for processing one 4096 x 4096 block are shown in the table below. Processing and I/O times are reported separately. For the processing, the times also include the speed improvement resulting from eliminating the scaling from the FFT routine, and the use of a new, more efficient code generator.

The I/O times compare the previous transposition requiring the use of an intermediate disk file to the method that performs the transposition in the staging buffer. In our first implementation, the intermediate file consisted of 16-bit data. However, it was determined experimentally that there was no significant loss of image quality if only the 8 most significant bits each of the real and imaginary parts of the range compressed data were written to the intermediate file. We have use the 8-bit intermediate data for the implementation with transposition in the staging buffer.

The processing times for the range and azimuth compression reflect the speed advantage obtainable from the MPP, while it can be seen that the limitation of the system is in the I/O transfer rates. There are two major limitations to the current system that are reflected in the reported rates. The first is that data transfers from disk files are limited to approximately 1.2 Mbytes/sec. The second is that the I/O transfers require the intervention of the host computer, and the overhead involved in this intervention severely limits the rate of transfer. A transfer rate of 80 Mbytes/sec. is theoretically attainable for moving data between the array unit and the staging buffer. In the table are reported the estimated times for a conservative 20 Mbyte/sec. rate. Note that the use of the staging buffer reduces the transposition time to 1.4 seconds, which is almost negligible.

The processing times shown can be improved by optimization of the code. The FFT includes a routine that checks the range of data values at intermediate stages of the computation and "clips" values that would exceed a 16-bit representation. This routine was necessary because the existing arithmetic routines produce large negative values in the case of overflow. Overflow does not normally occur unless there is noise

Figure 1. A SIR-B Image of Mount Shasta, California Generated Using the MPP

TIMINGS FOR PROCESSING 4096 x 4096 SIR-B SUBIMAGE

Processing	FFT Incorporating Scaling	FFT with no Scaling	Improved Code Generator
Range Compression	22.4	14.2	13.7
Azimuth Compression	9.3	5.9	5.3

I/O	Transposition Via 16 Bit Intermediate File	Transposition 8 Bit Intermediate File	Transposition in Stager	Estimate with Overhead Elimination
Raw Data Input	75.4	75.4	75.4	36.4
Constants and Params.	49.5	49.5	49.5	24.1
Transposition	225.0	124.1	62.0	1.4
Image Data Output	32.1	32.1	32.1	21.6

introduced while transferring data into the array unit. To avoid the effects of noise, the data values are rescaled at the intermediate stages of computation to provide maximum dynamic range, in effect increasing the signal-to-noise ratio. Given "well-behaved" data, these extra routines (which degrade the performance) are unnecessary. Each FFT requires 3.0 msecs, while without the clipping routine the time is reduced to 1.25 msecs. Further improvements in performance can be expected by optimizing the assembly code generated from the MPP Pascal source code.

ACKNOWLEDGEMENTS

The authors would like to acknowledge the active participation, in the development of the signal processing algorithms on the MPP, by Drs. J. P. Strong and J. C. Tilton of GSFC, S. W. McCandless of the User Systems, Inc., J. A. Abeles of Science Applications Research, Inc., G. C. Floam and J. Hurst, formerly of Science Applications Research, Inc.

REFERENCES

1. Cumming, I. G. and Bennet, J. R., Digital Processing of Seasat SAR Data, MacDonald, Dettwiler and Associates, Ltd., 1979.

2. Fischer, J. R., Technical Summary of the Massively Parallel Processor, this volume.

3. Harris, F. J., On the Use of Windows for Harmonic Analysis with the Discrete Fourier Transform, Proceedings of the IEEE, 1978.

4. Raff, B. E. and Kerr, J. O., The Johns Hopkins University Applied Physics Laboratory SAR Processor, IEEE EASCON, 16th Annual Electronics and Aerospace Meeting, 1983.

5. Wu, C., A Digital Fast Correlation Approach to Produce Seasat SAR Imagery, Proceedings of the IEEE International Radar Conference, 1980.

6. Wu, C. et al, Modelling and a Correlation Algorithm for Spaceborne Signals, IEEE Transactions on Aerospace and Electronic Systems, Vol. AES-18, 1982.