PARALLEL COMPUTING IMPLEMENTATION FOR ScanSAR MODE DATA

K. Leung, Q, Nguyen, T. Cheng
Jet Propulsion Laboratory
California Institute of Technology
4800 Oak Grove Drive, Pasadena, CA 91109, USA

W. Tung
Pacer-Infoteeh
2700 East Foothill Blvd, Suite 200, Pasadena, CA 91107, USA

Phone: (818) 393-9045
Fax: (818) 393-0202
E-mail: kon.leung@jpl.nasa.gov

Abstract[1] - Synthetic **aperture radar (SAR) data processing has matured over the past decade with development in processing approaches that Include traditional time-domain methods, popular and efficient frequency-domain methods, and relatively new and more precise chirp-scaling methods. These approaches have been used in various processing applications to achieve various degrees of efficiency and accuracy. One common trait amongst all SAR data processing algorithms, however, is their iterative and repetitive nature that make them amenable to parallel computing implementation. With SAR's contribution to remote sensing now well-established, the processing throughput demand has steadily Increased with each new mission, Parallel computing implementation of SAR processing algorithms is therefore an important means of attaining high SAR** data **processing throughput to keep** up with the ever-increasing science demand.

This paper **concerns parallel computing implementation of a mode of data called ScanSAR. ScanSAR has the unique advantage of yielding wide swath coverage in a single data collection pass. This mode of data collection has been demonstrated on SIR-C and is being used operationally for the first time on Radarsat. The burst nature of ScanSAR data is a natural candidate for parallel computing Implementation. This paper gives a description of such an implementation experience at Alaska SAR Facility for Radarsat ScanSAR mode data, A practical concurrent processing technique is also described that allows further improvement in throughput at a slight increase in system cost.**

I. INTRODUCTION

Digital processing of synthetic aperture radar (SAR) data[1] is known to be one of the most computationally demanding engineering applications. Although under continual development for the past fifteen years, SAR data processing systems to-date rarely approach real-time processing throughput except for a few special dedicated custom-built machines such as the Alaska SAR Processor (ASP) and in low resolution applications. Largely due to limitations in computation and processing algorithm technologies, early systems such as the Interim Digital SAR Processor (IDPS) [SEASAT, SIR-B] [2-3] which were hosted on general purpose computing platforms were only capable of processing throughput rate on the order of 1/1000th real-time. This level of performance severely limited their ability to supply science with time-critical data,

With the expanding role of the Alaska SAR Facility (ASF) in acquiring, processing and archiving data from a fleet of international polar orbiting satellites [4], the SAR Processing System (SPS) at ASF is being furnished with a Radarsat ScanSAR data processing system [5] that is capable of processing a minimum of 42 minutes of ScanSAR data in an 11-hour day which translates into a processing throughput requirement of - 1/16th real-time. **This paper describes the implementation requirements for the Radarsat ScanSAR data processing system at ASF, the** hardware platform evaluation and selection process, the ScanSAR algorithm and the implementation details associated with parallelizing the processing algorithm on the selected platform.

II. ScanSAR PROCESSOR REQUIREMENTS OVERVIEW

2.1 ScanSAR Mode Processing Requirements

The ASF SPS block diagram is given in Figure 1. To promote ease of operations and maintenance, specific project-wide guidelines regarding subsystem interfaces, systems standards, coding standards, user interfaces, and error reporting are applied to each subsystem within the SPS. The emphasis is on applying to the greatest extent possible Commercial Off-the-Shelf (COTS) hardware, software, standards, and technology. UNIX operating systems is a requirement on all hardware platforms as is compliance with POS IX (Portable Operating System Interface), High-level programming languages such as ANSI C and FORTRAN are selected for ease of implementation. A client-server communication model is

---

[1] The research described in this paper was carried out by the Jet Propulsion Laboratory, California Institute of Technology, under a contract with the National Aeronautics and Space Administration.

also adopted with SAR processors acting as production servers in response to a control processor client,

The ScanSAR Processor (SSP) System at ASF is required to process daily 34 minutes of Radarsat ScanSAR data. In addition, it is required to produce another 8 minutes of ScanSAR data in a quick 2-hour turnaround mode. With the current approach of sharing processing hardware with the Precision Processor, the net throughput requirement for the SSP becomes daily processing of 42 minutes of ScanSAR data in an 11-hour period or roughly 1/16th real-time rate,

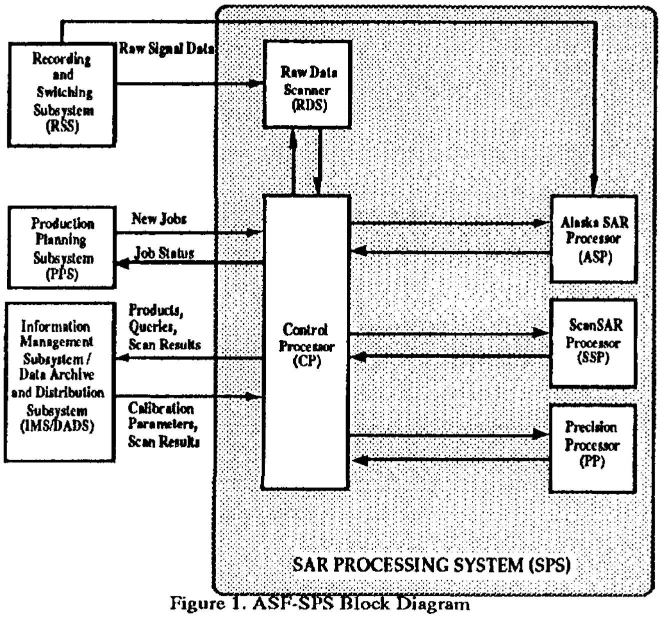

Figure 1. ASF-SPS Block Diagram

III. ScanSAR DATA CHARACTERISTICS & PROCESSING ALGORITHM

In the ScanSAR mode [6], wide swath coverage on the order of 200 km to 500 km is achieved by sweeping the antenna beam electronically in the cross-track dimension to generate multiple overlapping sub-swaths, each extending approximately 100 km in range. The resulting echo data for each subswath appears in the form of discrete 'bursts' rather than a continuous sequence. For Radarsat, swath width of -500 km can be achieved using its 4-beam modes, while -300 km swath can be obtained with the 3-beam and 2-beam modes.

ScanSAR data, when viewed on a per beam basis, resembles those collected from a burst mode SAR. The processing algorithm selected for ScanSAR data (see Figure 2) [7-8] therefore patterns closely after the one used for Magellan, a burst mode SAR that imaged Venus from 1990 to 1992. Data processing is basically handled on a per burst basis until the very last step when image data from each burst is merged to form the final multi-look image frame. The familiar frequency-domain fast correlation approach is used to compress range lines in each burst. A data corner-turn is then applied followed by azimuth processing which is accomplished using the efficient deramp-FFT method. Geometric and radiometric correction as well as pixel averaging are then applied to the resulting image pixels from each burst before they are merged together in a multi-look overlay process,

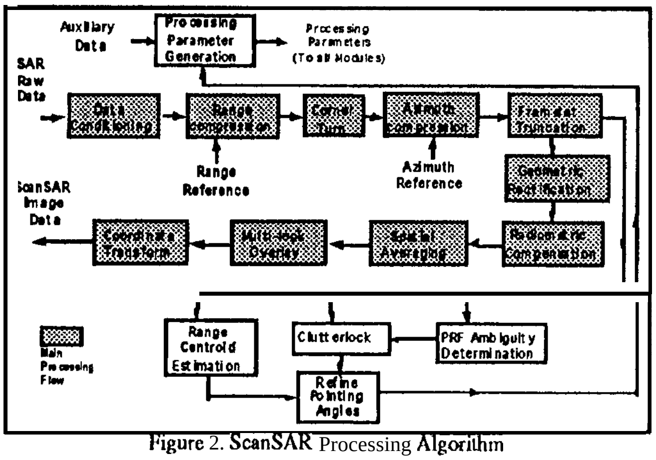

Figure 2. ScanSAR Processing Algorithm

IV. ScanSAR PLATFORM EVALUATION

The hardware selection process for the ScanSAR Processor took 6 months and 2 peer reviews to complete. This process involved the following steps:

1 initial scoping of class of machine,
2 performing computer market survey and identifying candidate platforms,
3 developing representative benchmark software and selection criteria,
4 performing benchmarking and platform evaluation,
5 selecting the target platform.

4.1 Scoping of Target Platform

Based on the requirements set forth in Section II and some prototype code, rough counts of the number of operations required to produce typical image products were compiled. This included FFT's in the range and azimuth compression processes and resampling in the projection and co-ordinate transformation process to name a few. Based on the throughput requirement, an estimate on the size of a target machine was determined to be on the order of 500 million floating point operations per second (500 MFLOPs) sustained.

4.2 Computer Market Survey

A computer market survey was then conducted to seek out suitable candidate hardware platforms. The criteria for inclusion in the evaluation process were mainly the machine's expected computational capability and its availability to support our benchmarking effort. Estimated system cost was not of initial concern noting that some of the more expensive supercomputers maybe available under time-used lease arrangements. Machines identified in this effort represented both Symmetric Multi-Processor (SMP) and Massively Parallel Processor (MPP) class of system architecture. SMP candidates included the Power Challenge series from Silicon Graphics, Inc. (SGI) and the DEC-7000 series models from Digital Equipment Corp. (DEC). MPP representatives were the CRAY T3D, the Intel Paragon, the Thinking Machine Corp.'s CM-5, and the IBM SP-2.

The hardware characteristics and configurations of the candidate platforms are listed in Table I.

Table I. Candidate Platform Characteristics

Machine	Max # of Proc's Benchmarked	Peak Rated MFLOPS per proc	Clock Rate MHz	Memory Size[1] (Mb)	Secondary cache memory (Mb)	Operating System	Programming Language
MPP MACHINE::							
IBM SP(2) (RS 6000)	32	268	67	128	N/A	AIX	F77,C
CRAY T-3D (DEC ALPHA)	128	150	150	32	N/A	UNICOS	F77,C
CM-5 (TI DSP)	32	128	128	32	N/A	CMOST	F90,F77,C
PARAGON (Intel i860 XP)	32	100	32	32	N/A	OSF/1	F77,C
SMP MACHINE:							
SGI P-Challenge (MIPS R8000)	18	300	75	2048	4	IRIX	F77,C
DEC-7000 (DEC ALPHA)	6	275	275	2048	4	OSF/1	F77,C

[1] Memory size used for benchmarking, size represents per processor for MPP and machine total for SMP.

4.3 Benchmark Software

The benchmark software covered all major computation and 1/0 steps in the ScanSAR algorithm. The benchmark code was written in FORTRAN and C, and ran on a model 670 SUN workstation with a single processor, It ingested raw data samples in 8I/8Q format from disk, performed the necessary data unpacking, performed the ScanSAR data processing steps outlined in the previous section, repacked the output pixels into byte format and output to disk. The input and output files as well as timing results obtained on the SUN were used as a reference for comparison.

The benchmark software consisted of the following categories of software modules:

1 Computation modules performing -
  a FFT's used in range compression and azimuth compression,
  b vectorized computation with indexed memory access for interpolation and resampling,
  c vectorized computation with direct memory access for multi-look overlay.

2 Data movement modules performing -
  a data packing& unpacking,
  b corner turn,
  c framelet truncation.

3 Data 1/0 modules performing -
  a disk read/write,
  b message passing between processors in MPP machines.

All software modules were written in ANSI FORTRAN or C language, each in less than 100 lines of code. Some pertinent parameters of the benchmark software are listed in Table H.

Table II. Benchmark Parameters

Number ot Range Samples	8192
Number of Azimuth Samples	
Range FFT Length	[illegible]
Azimuth FFT Length	64
No. Image Framelet Samples Along-track	~60
No. Image Framelet Samples Cross-track	1250
No. Final Image Samples Along-track	5000
No. Final Image Samples Cross-track	5000

4.4 Benchmark & Platform Evaluation

The benchmark software was first ported to each candidate platform and in all cases made initially to run on only a single processing element. The resulting timing and output files were collected and checked against the reference obtained on the SUN 670, The ported code on each candidate machine is then parallelized using standard vendor supplied routines and procedures. Although consultation from vendor on specific issues was allowed, the actual code porting and optimization on each machine were performed by a designated member of the hardware selection team so that a subjective measure of code development effort and code portability in general could be gauged.

A prioritized list of machine attributes was also developed to assure thoroughness in the evaluation and to maximize the objectivity in the platform selection process. A total of 10 specific attributes were used, listed below in descending order of importance to the ScanSAR data processing application:

1 throughput capability
2 software porting and development effort
3 operating system and compiler maturity
4 expected system reliability and maintainability
5 purchase cost
6 adaptability to other t ypes of processing algorithm
7 system expandability
8 compliance with Portable Operating System & Interface guide (pOSIX)

9. availability and support of the Open Software Foundation (OSF) Distributed Computing Environment (DCE)
10. availability of appropriate digital signal processing (DSP) library routines

4.5 Platform Selection & Description

Based on our evaluation of the candidate platforms against the list of prioritized attributes, the target platform selected is the IBM SP-2. The IBM SP-2, also known as the Scalable POWERparrallel System 9076, is a collection of RISC System/6000 processors connected together via a proprietary high performance switch (HPS) called the SP-2 communication subsystem. This scalable architecture, with the support of the HPS, affords the user scalable performance when dealing with compute- as well as I/O-intensive jobs. The user can actually execute both serial and parallel applications simultaneously while managing the whole system from a single workstation. The IBM SP-2 software is based on an open architecture AIX/6000 UNIX operating system that allows the user to easily integrate the SP-2 machine into user's existing environment. The IBM software supports a comprehensive set of AIX Parallel System Support Programs (PSSP) in addition to C and FORTRAN. To assist in parallel programming development, the IBM Parallel Environment for AIX program product provides development and execution support for parallel applications written in FORTRAN, C and C++ using the Message Passing Interface (MPI) standard. In addition, parallel libraries such as IBM Parallel Engineering and Scientific Subroutine Library (PESSL) are available to create or convert applications to take advantage of the parallel processor architecture of the SP-2.

To satisfy the requirement of processing 42 minutes of ScanSAR data in 11 hours, it is determined that 20 processing nodes are required. To enhance reliability and to retain flexibility for expansion, the 20 nodes are grouped into two 8-node and one 4-node machines. Each processing node is chosen to be the 'wide' 66MHz variety equipped with 256 MB of RAM, 4 GB of disks, an Ethernet connection and a High Performance Switch (HPS) Adapter. A FDDI controller connects one of the processing nodes of each machine to the external Control Processor (CP) to effect high speed data access. Operations on each machine is orchestrated by a model 390 control workstation equipped with 128 MB RAM, 2 GB of disks, 2 Ethernet controllers, a CD-ROM drive, an 8-mm tape drive for back-up, and an 19-inch color monitor. It is expected that each 8-node and 4-node machine can handle processing of 17 and 8 minutes of RADARSAT ScanSAR data respectively in an 11-hour day.

The hardware configuration of a single 8-node SP-2 unit is illustrated in Figure 3.

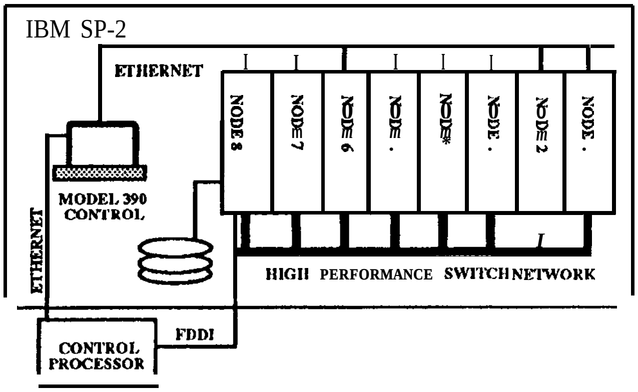

Fig. 3. Target Platform Configuration

V. IMPLEMENTATION

5.1 Sample Dataset Description

The sample dataset consists of approximate 1500 bursts (see Figure 4) representing a typical 4-beam 500 X 500 km^2 image frame, Each burst consists of approximate 65 range lines, each contains approximate 8500 samples. Each sample is represented in 2 bytes (8I/8Q) resulting in a data file size of 1.5 Giga bytes (GB). The data is generated synthetically such that point targets will result when the data is processed.

During the benchmarking process, the concept of 'fine grain' versus 'coarse grain' parallelization was explored. The difference between the two is mainly illustrated by the fact that 'fine grain' parallelization typically applies parallel processing to the lowest level of data element. In the case of ScanSAR data, 'fine grain' parallelization will effect the distribution of individual range lines (and azimuth lines in azimuth processing) to all available processing elements for processing, This approach, although effective in distributing computation workload, can often times cause throughput penalties in the form of excessive data movements amongst processing elements. With the burst nature of the ScanSAR data, it is demonstrated that the use of 'coarse grain' parallelization, where blocks of integral bursts are distributed to the available processing elements for processing, is the more efficient approach. 'Fine grain' parallelization is usually the approach taken by vendor supplied parallelization routines in the absence of user intervention.

Using the 'coarse grain' approach, the sample dataset is divided into a number of roughly equal segments (see Figure 4) equaling to number of processing elements. Each equal segment is assigned to a processing element for processing. By keeping each data burst within a processing element throughout most of the processing steps, the need for data movement and communications between processing elements is greatly reduced.

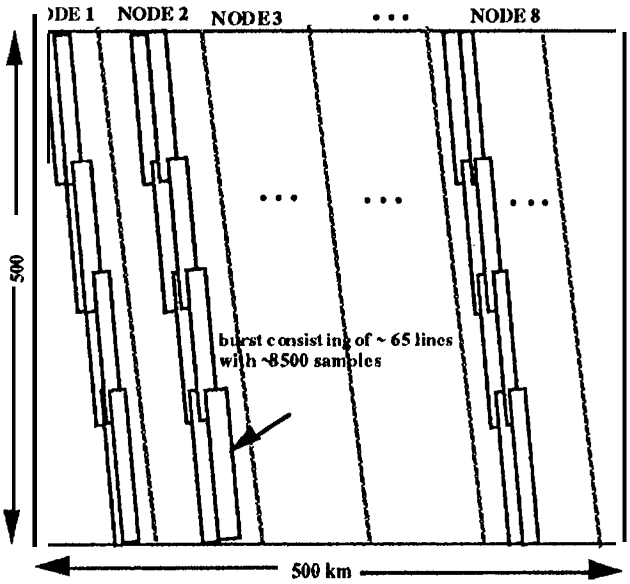

Figure 4. Segmentation of Data Frame

5.2 ScanSAR Algorithm Implementation

The ScanSAR data processing algorithm [4] is executed in four stages:

1. a raw data transfer stage where conditioned (decoded and reformatted) echo data file together with all necessary ancillary data files are transferred from the CP disk to the SP-2,
2. a pre-processing stage that effectively processes a small subset of the data to derive refined pointing information,
3. a main-processing stage that executes the CPU-intensive steps of the processing algorithm such as range and azimuth compression, geometric rectification, and radiometric compensation,
4. a post-processing stage where partially overlaid image pixels from each data segment are merged to form multi-look pixels before being projected onto a specific geometric co-ordinate...

The following sections describe the implementation details on the SP-2 for each processing stage, using the sample dataset shown in Figure 4 for illustration.

Raw Data Transfer

A typical 500 km X 500 km image frame consists of -1500 data bursts and occupies a 2 GB data file that resides on disks attached to the CP. Two options were considered in bringing the data file into the SP-2 via the FDDI network. The first option involves buffering the input data from CP first onto external disks mounted on one of the processing nodes (e.g. node 8 in Figure 3). Processing at each node will then begin by accessing data from the external disks. The second option effects the transfer from CP through one of the nodes to the local disks on each processing node via the HPS, Processing at each node will then begin by accessing data from its own local disks. Timing results for these two options based on the sample dataset are -10 minutes and ~13 minutes respectively.

Pre-Processing

Pre-processing refers to the process of refining processing parameters initially derived from ephemeris, It involves most of the processing stages in main processing except that only a small subset of the data is processed. The discussion on pre-processing implementation is therefore defered to the main processing section.

Main Processing

Main processing refers to the processing steps of range compression, corner-turn, azimuth processing, geometric rectification, radiometric compensation, pixel averaging, and partial overlay. Using the 'coarse grain' parallelization approach, contiguous data bursts from a data segment are processed by one processing node (see Figure 4). In particular, range lines from each burst arc first range compressed using the fast Fourier correlation method [9] whereby range lines are FFT'ed, multiplied with a range reference, and inverse FFT'ed. The resulting range compressed data is corner-turned using the on-board RAM memory of the processing node. Azimuth processing is then applied using the deramp-FFT method [9] whereby each azimuth line is multiplied with a frequency ramp followed by a forward FFT. Framelet truncation, geometric rectification, and radiometric compensation are then applied. The pixel data is then detected and merged with the corresponding pixels obtained from the previous bursts. A pixel averaging process is applied to achieve constant resolution and number of looks. At the end of main processing, each processing node will hold an overlaid image segment in its local disks. Timing results on a 8-node machine based on the sample dataset are 31 minutes when data is accessed from the external disks versus ~21 minutes when accessed from the local disks.

Post Processing

The post processing stage merges the overlaid image segments from each node into a single large image before performing the final projection onto a specific co-ordinate grid. This process is done at onc of the processing nodes. The resulting image data occupies a file of 25 MB at one byte per pixel, Timing results indicates -5 minutes to accomplish this stage of processing.

5.3 Processing Throughput

The handling of a particular job request by the ScanSAR Processor involves the sequential execution of the four processing stages described in the previous section. As evident in Table III, the best overall processing time for the sample dataset is -39 minutes, Assuming that the preprocessing time is equivalent to 30% of the main processing time, the total processing time for each 500 X 500 km^2 frame using an 8-node SP-2 is therefore projected to be -46 minutes, or ~1/37th real-time. The effective throughput rate for the overall system (consisting of two 8-node and one 4-node machine) is then better than 1/15* real-time which surpasses the throughput requirement of l/16th.

'l'able **III.** Timing **Results** on an 8-node SP-2

Processing time for each step **(min:sec)**	**Sequential** Processing :**Read** from external di **sks**	**Sequential** Processing: **Read** from **local** disks	X.-on current **Processing:** Main concurrent with **transfer and post-processing**	**Concurrent** Processing Main and post processing **concurr-ent with transfer**
T1:Transfer	10:30	13:09	14:58	14:58
T2:Main Processing	31:00	21:22	37:15	26:07
T3:Post Processing	5:02	5:02	20:07	5:15
Total processing time	46:32	39:33	37:15	31:22

5.4 **Parallelization** Options for Multiple Processing Jobs

Option for gaining throughput improvement exists when multiple jobs are to be processed. It is noted that during pre- and main processing, there is little *or* no 1/0 traffic on the FDDI and HPS. Similarly, the transfer stage requires little or no CPU involvement from the processing nodes. The post processing stage however does require both 1/0 and CPU. Based on the timing results obtained for each processing **stage (see Table III), two concurrent processing** schemes were studied, each handling multiple jobs simultaneously. Figure 5 depicts the two concurrent schemes. Their timing results relative to the sample dataset are displayed in Table 111. It is evident from the timing results that having data transfer performed in parallel with both main processing and post processing provides the best throughput results. Based on these results, processing throughput approaching 31 minutes per frame can be achieved when jobs of up to the size of the sample dataset are fed successive] y through an 8-node SP-2. Assuming again in the worst case that preprocessing time is equivalent to 30% of the main processing time, the throughput of the 8-node SP-2 is therefore projected to be -39 minutes per frame or roughly 1/3 1th real-time. This translates into an effective throughput rate for the ScanSAR processor system (consisting of two 8-node and one 4-node machines) of 1/1 3* real-time, surpassing the required throughput of 1/16th real-time by about 20%. However, there is a cost associated with the concurrent implementation. Job handling at the Control Processor will have to be modified to simultaneously handle and track multiple jobs, a definite complication relative to handling one job at a time. Similarly on the SP-2 side, job control and sequencing become more complicated. Also additional memory and disk capacity are required to accommodate data from mutliple image frames.

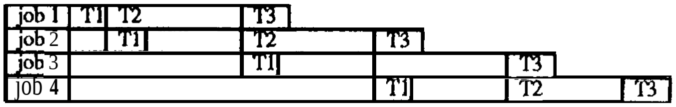

Figure 5ss. Main-processing concurrent with Transfer and Post-processing (3 concurrent jobs)

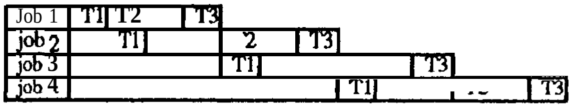

Figure 5b. Transfer Concurrent with Main-Processing and Post-Processing (2 concurrent jobs)

V. CONCLUSION& STATUS

A parallel ScanSAR data processing implementation has been presented involving a particular MPP platform, the IBM SP-2. Realistic timing results collected demonstrate the selected algorithm and architecture can satisfy the required RadarSAT **ScanSAR** data processing throughput demand at ASF. Additional means of improving the overall system throughput at a slight increase in system cost has also been identified. The current SSP is in its final stage of development and is on schedule to be operational at ASF by May 19%.

ACKNOWLEDGEMENTS

The authors wish to thank the Alaska SAR Facility Project for supporting the work described in this paper.

REFERENCES

[1] Kiyo **Tomiyasu,** "Tutorial Review of Synthetic-Aperture Radar (SAR) with Applications to Imaging of the Ocean Surf ace," Proc. I EEE, vol. 66, **pp.563-583, May** 1978,

[2] C. **Wu, B. Barkan, W. J. Karplus and D. Coswell,**"SEASAT Synthetic **Aperture** Radar data reduction using parallel programmable array processors," **IEEE** Transaction on Geoscience **and** Remote Sensing, **July** 1982, GE-20,352-358,

[3] **C. Elachi *et al,*"SIR-B** The second **shuttle** imaging radar experiment," **IEEE** Transaction on Geoscience and Remote Sensing, **Vol. GE-24,** I1o,4, pp. 445-452, 1986.

[4] K. **Leung *et*** al, "RADARSAT Processing System **at** ASF," in **preparation.**

[5] **T. Cheng, K. Leung, M. Jin and E. Chu,"ScanSAR and Precision** Processor Implementation rtt the **Alaska** SAR **facility," IGARSS'95** Technical Program, Vol **II, pp.2302-2306,** Italy, July 10-14, 1995.

[6] **R.** Keith **Raney *et al,*** **"RADARSAT,"** ***Proceedings of the IEEE, Vol. 79, No. 6,*** June 1991,

[7] **M. Chen,** M. Jin and K. Letmg, **"Implemtation** and Performance of **Magellen** Digital **Correlator** Subsystem," **IGARSS** '92 **Technical** Program **Volume** 11. **pp.** 1301-1304, Houston, **texas,** May **1992.**

[8] **K.** Leung, M. Jin, C. Wong and J. Gilbert, "SAR Data Processing for **Magellan** Prime Mission," **IGARSS** '92 **Technical** Program **Volume** 1. pp. 606-609, Houston, **Texas,** May 1992,

[9] K. **Leung,** M. **Jin,** "Processing of **ScanSAR** Mode **Data** For RADARSAT," **IGARSS '93 Technical** Program, **Volume** 1[1, pp 1185-1 **188,** Tokyo, April 14, 1993.