units (GPUs) for accelerating data analysis tasks.

The tool suite developed in this project enables scientists now to solve demanding data analysis problems in IDL that previously required specialized software, and it allows them to be solved orders of magnitude faster than on conventional PCs. The tool suite consists of three components: (1) TaskDL, a software tool that simplifies the creation and management of task farms, collections of tasks that can be processed independently and require only small amounts of data communication; (2) mpiDL, a tool that allows IDL developers to use the Message Passing Interface (MPI) inside IDL for problems that require large amounts of data to be exchanged among multiple processors; and (3) GPULib, a tool that simplifies the use of GPUs as mathematical coprocessors from within IDL.

mpiDL is unique in its support for the full MPI standard and its support of a broad range of MPI implementations. GPULib is unique in enabling users to take advantage of an inexpensive piece of hardware, possibly already installed in their computer, and achieve orders of magnitude faster execution time for numerically complex algorithms. TaskDL enables the simple setup and management of task farms on compute clusters.

The products developed in this project have the potential to interact, so one can build a cluster of PCs, each equipped with a GPU, and use mpiDL to communicate between the nodes and GPULib to accelerate the computations on each node.

This work was done by Peter Messmer of Tech-X Corporation for Goddard Space Flight Center. Further information is contained in a TSP (see page 1). ***GSC-15749-1***

Experiment in Onboard Synthetic Aperture Radar Data Processing

The algorithm runs in a parallel/multicore environment, and integrates radiation hardening by software (RHBS) self-protection strategies.

Goddard Space Flight Center, Greenbelt, Maryland

Single event upsets (SEUs) are a threat to any computing system running on hardware that has not been physically radiation hardened. In addition to mandating the use of performance-limited, hardened heritage equipment, prior techniques for dealing with the SEU problem often involved hardware-based error detection and correction (EDAC). With limited computing resources, software-based EDAC, or any more elaborate recovery methods, were often not feasible. Synthetic aperture radars (SARs), when operated in the space environment, are interesting due to their relevance to NASAs objectives, but problematic in the sense of producing prodigious amounts of "raw" data. Prior implementations of the SAR data processing algorithm have been too slow, too computationally intensive, and require too much application memory for onboard execution to be a realistic option when using the type of heritage processing technology described above.

This standard C-language implementation of SAR data processing is distributed over many cores of a Tilera Multicore Processor, and employs novel Radiation Hardening by Software (RHBS) techniques designed to protect the component processes (one per core) and their shared application memory from the sort of SEUs expected in the space environment. The source code includes calls to Tilera APIs, and a specialized Tilera compiler is required to produce a Tilera executable. The compiled application reads input data describing the position and orientation of a radar platform, as well as its radar-burst data, over time and writes out processed data in a form that is useful for analysis of the radar observations.

The application is capable of recovering from some types of SEU-induced interference with component processes and/or corruption of the shared application memory, and also writes out performance statistics designed to assist in evaluating the effectiveness of the novel RHBS techniques employed. These performance data are useful in identifying, time-stamping, and (indirectly) geo-locating SEU incidents along with the application's responses.

The tileSAR software distributes the problem of processing SAR data over an "engine" made up of a number of cooperating parallel processes (one per core). This engine is replicated three times within the Tilera processor; always one process per core, and all engines running in parallel. Each engine also includes an additional scrubbing process (core), and there is one final triple-voting process external to the engines. When distributing the SAR algorithm among the processes of each engine, the usual single-stringed implementation (each sub-task executed in sequence) is replaced with an implementation where independent operations are carried out concurrently by independent processes. Every opportunity for concurrency within this algorithm is exploited, as this dramatically reduces execution time. The result of each engine's processing is a series of output records. The processes that make up each engine share a single working set of data, collectively called the engine's "workspace." The state of each workspace at each synchronization point is expected to be identical to that of the other engines, and reflects the state of progress the engine has made through its execution of the algorithm. The combined effect of scrubbing and triple-voting enables certain types of workspace corruption to be detected and corrected such that processing may continue without interruption or error.

This work was done by Matthew Holland of Goddard Space Flight Center. Further information is contained in a TSP (see page 1). ***GSC-15757-1***