NASA Advances Robotic Space Exploration

Built with commercial components, economical system-on-chip designs, and open source software, NASA's sophisticated probes and rovers will explore space efficiently and autonomously.

Daniel S. Katz
Raphael R. Some
Jet Propulsion Laboratory

NASA's 40-year history has generally been characterized by successful space exploration leading to tremendous amounts of new data and knowledge about the Earth, the solar system and its other planets, and the stellar spaces beyond. However, gaining more knowledge will require new capabilities in onboard processing hardware, system software, and applications such as autonomy.

Initial robotic space exploration missions were, for the most part, large flying cameras. The goal then was—and, in many cases, still is—to send back images of planets, moons, and other extraterrestrial bodies. During this first phase of robotic space exploration, the hard problems NASA faced consisted mostly of getting the instruments to their destinations in working order and getting the data back to Earth for analysis by mission scientists. These instruments, over time, evolved from cameras to more sophisticated imaging radars, multispectral imagers, spectrometers, gravity wave detectors, a host of prepositioned sensors and, most recently, rovers.

The forces shaping future NASA missions are shifting the operating paradigm from data- to knowledge-gathering missions. Building on 40 years of space-exploration expertise and previous mission data, NASA has initiated several programs dedicated to developing technologies to support this new need.

PAST SPACECRAFT

Researchers solved NASA's early data-gathering and transmission challenges with an increasingly sophisticated set of navigation, positional-sensing, and radio communication systems. Often, NASA accomplished navigation by tracking the spacecraft from the ground using large radar systems, then uploading rocket-firing commands to the onboard computer, which functioned as a sequencer. The computer executed a relatively small set of real-time programs that controlled the spacecraft and its payload, cycling through a predefined sequence of events to position the spacecraft, enable the instruments, and transmit data back to Earth.

Although maintaining a spacecraft's health required the onboard computer to perform several housekeeping and monitoring functions, the Earth-based mission team made all the real decisions. Ground stations also uploaded new programs when the mission operations engineers determined a need for the next set of functions or mission profile changes. Thus, these early spacecraft computing systems only needed to be extremely reliable and capable of real-time control functions.

To cope with the severe environments of space and the often unknown dangers that a spacecraft might encounter, engineers used ultrareliable, radiation-hardened components and devised straightforward, highly reliable error detection and mitigation mechanisms to ensure immunity from single, and sometimes multiple, faults. In *safing*, for example, when the computer detects an unexplained fault, it simply shuts down spacecraft operation and calls home for further instructions.

Engineers painstakingly handcrafted the programs run on these machines, then checked and rechecked them before upload to ensure that few, if

Published by the IEEE Computer Society
0018-9162/03/$17.00 © 2003 IEEE

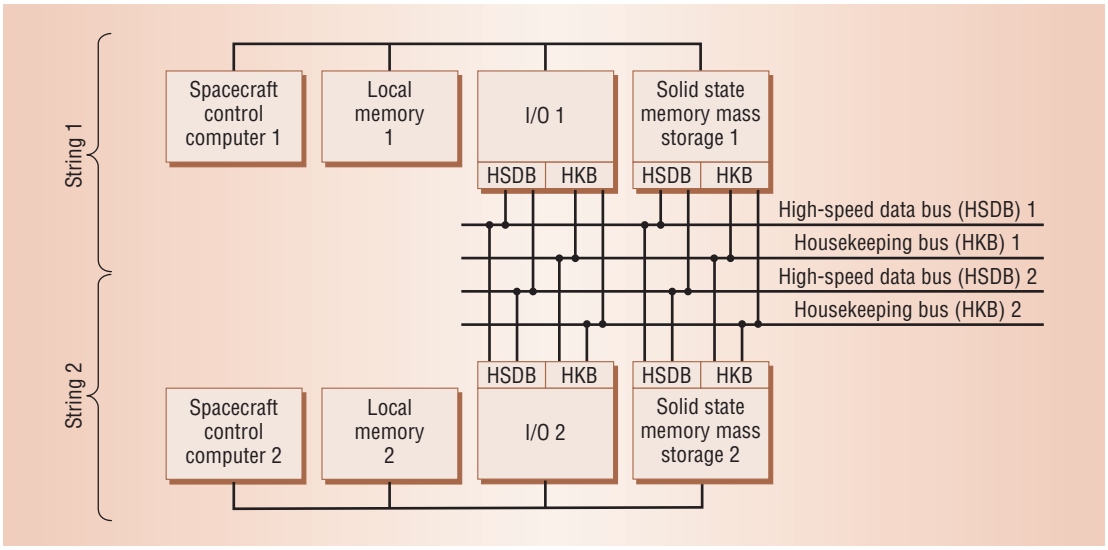

Figure 1. Command and data handling subsystem. Consisting of two sets of computing and interconnect resources, this dual-string configuration lets a failure in any component of one string be replaced by its twin in the other string.

any, bugs remained in the code. For the most part, this strategy worked well and continues to work in many NASA science missions today.

COMPUTING DESIGN CONSTRAINTS

The basic architecture of most NASA spacecraft consists of three major sections: launch vehicle, spacecraft bus, and payload. The launch vehicle gets the bus and payload into space. The bus provides the spacecraft's basic infrastructure, including the physical structure, power, and the attitude-control, navigation, and spacecraft-computing subsystems. The payload consists of the instrument package and any other support subsystems required for the spacecraft to perform its specific mission.

Figure 1 shows part of the spacecraft bus: the command and data handling subsystem. C&DH usually consists of two sets of computing and interconnect resources. This *dual-string configuration* lets a failure in any component on one string be replaced by its twin in the other string, thus providing for the whole system's long-term reliability.

Usually, the first string is powered while the second string remains dormant. A first-string element failure triggers the powering up of its second-string replacement. Within the C&DH subsystem, the spacecraft control computer provides the main computing element. The SCC typically performs spacecraft command interpretation, sequencing, and any attendant data processing. When designing spacecraft computing subsystems such as these, engineers consider the following factors:

- *long-term reliability* in the space environment;
- natural *space radiation* and its effects on the computing subsystem;
- the need for *real-time processing* and sequencing; and
- *power, mass, and cost* constraints.

Long-term reliability

Spaceborne computing subsystems must survive for up to 10 years or more in a hostile environment, with no possibility of human intervention, maintenance, or repair. Engineers who work on these highly reliable systems have developed standard methods to evaluate parts quality, qualify components, reinforce packaging, and ensure component-level reliability.

To provide a low-stress environment for the onboard electronics, the spacecraft incorporates an elaborate thermal management subsystem, electrostatic and plasma discharge control, and mechanical shock and vibration isolation. Thus, in most spacecraft, the typical thermal, electrical, and mechanical environment usually falls well within the acceptable operational range of these high-reliability parts. The space environment does, however, offer other unique hazards and design challenges, such as space radiation.

Space radiation

Engineers have traditionally handled the space-radiation environment by designing computer hardware largely immune to space radiation's effects. Three primary radiation effects concern designers:

- a TID, or *total ionizing dose*, causes semiconductor components to accumulate radiation damage that induces threshold voltage shifts, increased current leakage and, eventually, circuit failure;
- an SEU, or *single event upset*, occurs when an energetic proton or heavy ion causes an electrical transient in a circuit, resulting in a bit flip known as a *soft error*; and
- an SEL, or *single event latchup*, resembles an SEU except it results in a CMOS latchup condition—effectively a power-ground short that usually requires a power cycle to clear and, if not shut down, can cause burnout in some parts.

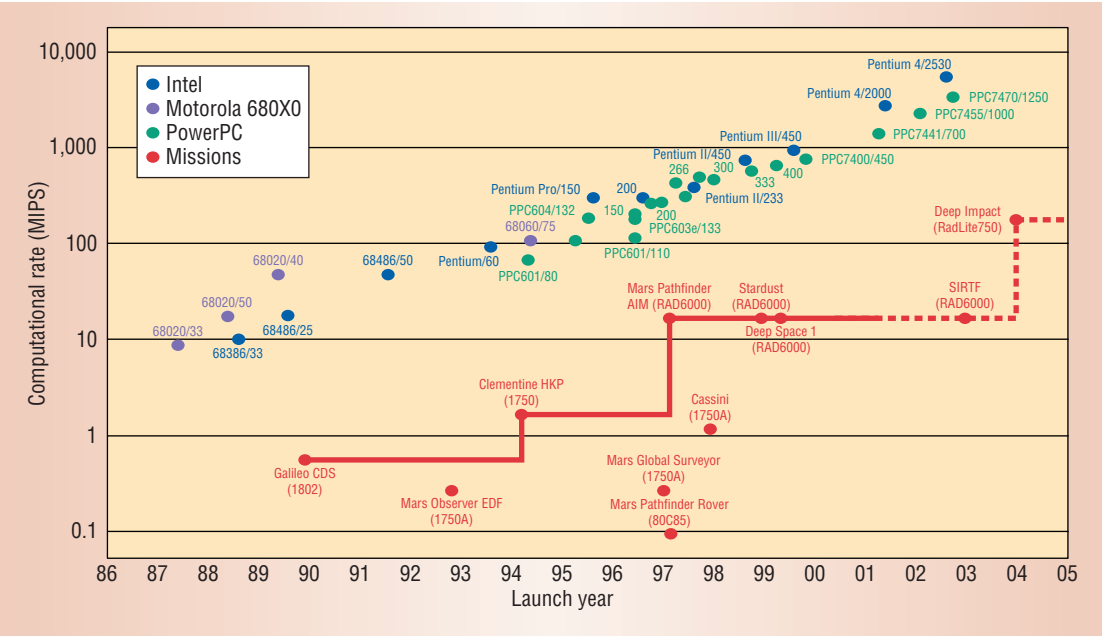

Figure 2. Performance of commodity and radiation-hardened processors over time. Radiation-hardened processors are between one and two orders of magnitude less capable at any given time.

So called *radiation-hardened* components are built using semiconductor processes modified from their commercial counterparts to be largely resistant to radiation—at the cost of power, density, and speed. NASA uses these parts for its missions, leveraging the investments made by the defense community to establish a US-based, radiation-hard components infrastructure for strategic military systems.

Developing the hardware and software for these systems requires a large and expensive development-tools infrastructure. In recent years, the defense and civilian aerospace communities have tended to move away from custom designs and toward commonality with commercial architectures. So, while designers built earlier systems on radiation-hardened custom processors such as the Mil-Std 1750 processor architecture developed by the US Air Force, they have based more recent radiation-hardened systems on the IBM R6000, MIPS R3000, and IBM/Motorola PowerPC 603 and 750 architectures.

The most capable radiation-tolerant processor today, the PowerPC 750 RadLite (www.iews.na.baesystems.com/space/rad750/), can run at up to 133 MHz. Part of JPL's X2000 next-generation avionics system (x2000.jpl.nasa.gov/), this processor will first fly on the Deep Impact mission in 2004. In comparison, commercial versions today execute at more than 1 GHz and have architectural refinements available in the PowerPC 7400 family that provide as much as an additional 400-percent performance improvement over the PPC750 architecture. Figure 2 shows a comparison of the capabilities of past, present, and near-future commodity and radiation-hardened processors.

Real-time processing

NASA's operating system of choice for many years has been VxWorks (www.wrs.com/), a simple, easily understood and configured and highly deterministic operating system. VxWorks makes it relatively easy to deterministically schedule tasks with real-time deadlines and to discretely design all aspects of task execution and data handling. Relying on this paradigm requires detailed design and analysis of software-execution-flow and system-level support code and a willingness to forego modern OS services such as memory protection, process and thread handling, and globally optimized task scheduling.

Much of the research in algorithms and applications that will run on a spacecraft is done on standard PCs or workstations running a traditional desktop OS such as Unix. Putting these codes on a spacecraft requires porting them to VxWorks or a similar real-time OS. RTOSs tend to treat all software as one large application, which means that a failure of any part can cause the entire processor to fail. Further, traditional RTOSs have minimal execution fault detection and handling capability. This vulnerability adds to the pressure on software implementers to write simple code and to perform intensive testing and validation prior to flight.

Other constraints

Power and mass pose a concern on any spacecraft, especially with regard to available power-generation equipment. Solar cells, batteries, and radio-isotope thermal generators tend to be prohibitively volume-, mass-, and cost-intensive. Given the expense of getting a large mass into space, power is always in limited supply, and the computing system receives only a small portion of the available total. Building large clusters of radiation-hardened computing nodes, for example, would require significantly more power than that available on most spacecraft.

Finally, budgeting always drives NASA mission planning. The cost of radiation-hardened custom hardware and the labor-intensive nature of reliable real-time software development make spacecraft onboard computing a costly and cost-sensitive area.

DRIVERS FOR CHANGE

Several forces drive demand for more and better standardized onboard processing. Meeting these demands will likely require radically changing NASA's spacecraft computing systems.

Shifting mission parameters

The size and frequency of missions provides one such driver. In the 1970s and 1980s, NASA launched a handful of large missions each decade. Spacecraft design and implementation could take up to 10 years and cost as much as $1 billion. Today, with mission budgets reduced by 60 to 85 percent, NASA centers such as JPL and Goddard Space Flight Center launch as many as six small- to medium-size spacecraft and one larger, major science platform annually.

Smaller spacecraft must execute these missions, with proportionately reduced power and mass. In addition, NASA can no longer afford the large, round-the-clock teams of mission-operation specialists traditionally used to monitor spacecraft health and status, track anomalies, and regularly design and upload new command sequences.

Bandwidth versus data volume

Available communication bandwidth versus sensor data volume provides another driver. Although the Deep Space Network communication system, which allows transfer of data between far-flung NASA spacecraft and Earth, has grown in capability, the data that can be taken from spaceborne instruments has grown exponentially. Current hyperspectral imagers and synthetic aperature radar can generate hundreds of gigabytes per day. Similarly, the communication systems supporting Earth-orbiting spacecraft are also severely limited in comparison to desired bandwidths for current and planned near-Earth missions. With no plans to exponentially expand NASA's communication capability, large fractions of potential data are, and will continue to be, tossed aside, dramatically reducing the potential science return from these missions.

NASA's new reality

Ultimately, future missions will require low-power, low-mass, high-performance computing. Common hardware, design reuse, and multimission production runs can address these needs.

It is no longer feasible to custom craft the software for each mission. NASA needs standardization so that a common base can exist for at least classes of similar missions. Computing systems must be modular and incrementally upgradable to allow cost-effective evolution. Needed upgrades must be judiciously developed, then provided as standard options. Furthermore, the baseline suite of components must be easily and cost-effectively configurable for each mission, resulting in customized modules that can be added to a common pool of reusable components.

Ideally, accessing an onboard science data processing system should be no more difficult than using a remote computer.

New methodologies

To facilitate cost-effective application-code development, tomorrow's onboard computing systems must offer capabilities comparable to and compatible with commercial ground systems used for software development and scientific research. This compatibility must extend to the hardware, operating system, and development-support tools such as compilers and debuggers.

Similarly, NASA must take direct advantage of research in applications and algorithms by simplifying the process through which ground-based *research software* becomes *flight software*. Ideally, accessing an onboard science data processing system should be no more difficult than using a remote computer. Mission investigators should be able to write code in the laboratory, verify that it works correctly, then transfer it to the spacecraft. Further, they should be able to retrieve the results of their code for final software test, verification, and validation.

Whole new mission classes—such as small landers, rovers, atmospheric flyers, and constellations of microspacecraft—drive the development of future spacecraft systems in several ways. These small systems do not have the requisite mass or volume budgets to support either the thermal-management systems or the power-generation subsystems of current spacecraft. Nor can they support large communication subsystems, a constraint that further strains link bandwidths. Finally, humans working at a mission control center millions of miles away simply cannot handle the complexity of controlling these robotic explorers in real time.

Many developing technologies could facilitate more advanced computing and missions, including commercially based hardware and system software components and architectures, novel alternative architectures enabled by recent advances in commercial semiconductor processes, and advanced autonomy.

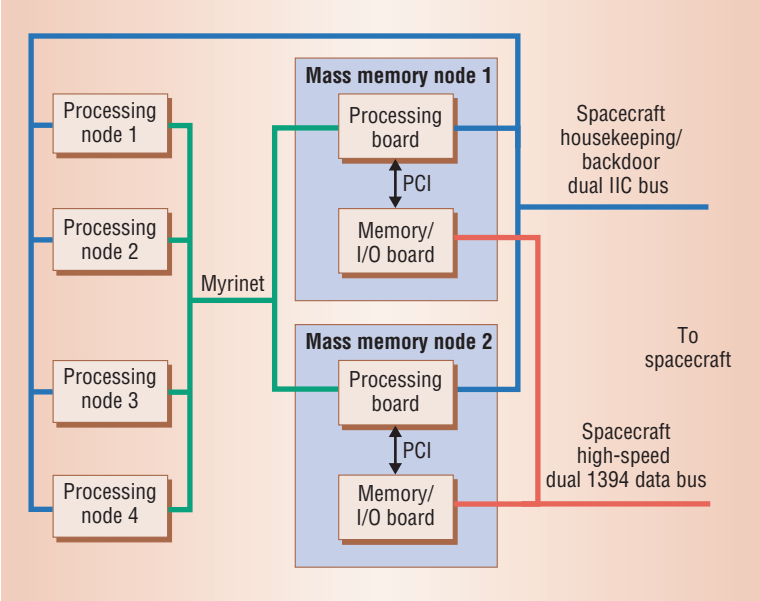

Figure 3. Remote Exploration and Experimentation architecture. Originally envisioned to give spaceborne computing systems COTS compatibility, ease of implementation, and fault tolerance, the REE project made significant advances before its 2001 cancellation.

MOVING TO COTS

Commercial off-the-shelf (COTS) components offer mission developers and data users several advantages:

- improvements in power, performance, cost, implementation schedules, and interoperability with commercial systems; and
- familiarity with COTS standards and thus ease of integration and interaction with ground-based development and analysis systems.

Although they do not pose significant challenges in most commercial applications, reliability, shock, vibration, thermal management, radiation, and dielectric charging all must be overcome in spacecraft electronics. Issues of component and system lifetimes, repair and maintenance approaches, software and system reliability, and real-time constraints compound this situation.

Former US Secretary of Defense William Perry formally started a movement toward using COTS products in 1994 when he required that military programs use them as their default choice. Military projects soon began using PPC750, G4, and Pentium processors; CPCI buses and 3U form factors; C and C++; and modern file-oriented operating systems. These developments resulted in the growth of highly reliable COTS-based components, boards, systems, system software, and development tools.

As computing becomes mission critical for a growing business sector, demand has increased for enhanced reliability, availability, real-time capability, and fault tolerance in general commercial products. In addition to various Linux entrants, older real-time operating systems, such as VxWorks, now include support for memory protection, fault detection and handling, and modern programming concepts such as processes. Next-generation I/O standards also provide enhanced fault tolerance and real-time support, such as isochronous protocols that reserve bus bandwidth for guaranteed real-time message transmission and delivery.

COTS HARDWARE

The natural evolution of complementary metal-oxide semiconductor processes, enabling faster and more complex structures, has also had the effect of improved TID and SEL tolerance. Today's digital CMOS parts can easily withstand severe natural TID levels and are mostly immune to SEL in natural space-radiation environments.

Thus, SEU tolerance is the primary radiation concern in most space-based computing systems. Although surmounting COTS susceptibility to SEU-induced errors presents a formidable challenge, engineers are developing techniques to handle low to medium SEU rates. Serendipitously, growing use of silicon-on-insulator CMOS technologies provides an order of magnitude improvement in SEU tolerance. These developments do not provide sufficient reliability for the most severe radiation environments, but they do permit COTS use in many missions.

Parallel processing

Several years ago, NASA sponsored the Remote Exploration and Experimentation (ree.jpl.nasa.gov/) project.[1] REE sought to develop a COTS-based, scalable, parallel-processing supercomputer suitable for use in many NASA missions. The project posed several questions:

- Does onboard science data processing offer a significant advantage?
- Can we build a reliable COTS-based system?
- If so, what would be the system's cost, power, mass, reliability, and throughput characteristics?
- Will the scientific community accept moving science processing to the spacecraft?

The project partnered with the Air Force Research Lab's Improved Space Architecture Concept program (www.fas.org/spp/military/program/test/isac.htm) to seek the answers to these questions on three fronts. First, they asked five science missions to define how they would use an onboard cluster computer and to write parallel science codes for use on such a system.

Second, the team built a COTS-based cluster computing testbed that provided low-level fault-tolerance support. They used the testbed to characterize these science codes and representative COTS hardware and software for susceptibility to

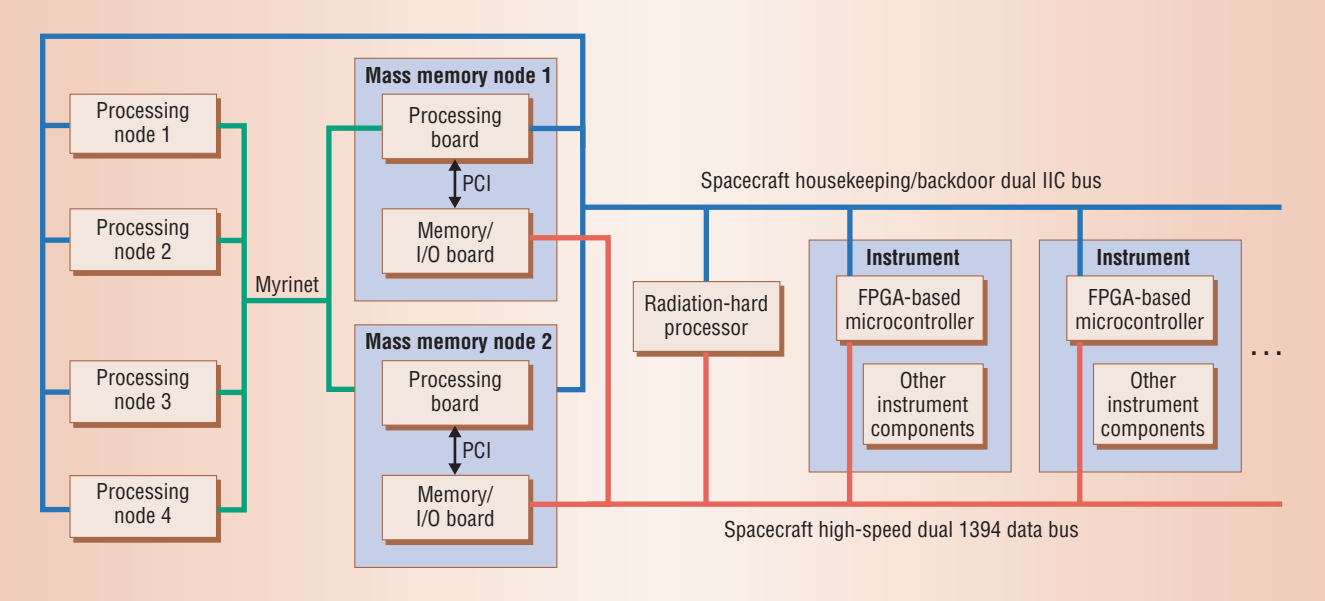

Figure 4. Spacecraft architecture with radiation-hardened core and FPGA-based instrument and subsystem controllers. This ultrareliable microcontroller augments the COTS-based computing system by detecting system failures and taking control of the spaceborne platform to recover from a computing crash.

soft errors. The team also developed a suite of software-based fault-tolerance techniques at the middleware and application levels.

Third, the team initiated an intensive radiation effects study to develop a test system and a methodology for rapidly and cost-effectively obtaining radiation-effects data on complex COTS parts.

Figure 3 shows the baseline REE architecture as originally envisioned and its place in the spacecraft computing system architecture.

These early successes showed that this technology offers a viable candidate for enabling breakthrough capabilities, new science, and vastly increased science return in next-generation spacecraft. Once considered unrealistic, COTS-based high-performance onboard computing has gained support across NASA and is spawning new and innovative research proposals to develop and use COTS-based computing in space.

Hardened cores

Architectures are being developed that use small, low-power, hardened cores to augment a COTS-based high-performance computing system. An ultrareliable, radiation-hardened computer or microcontroller is used to detect a system failure, take control of the platform, reset the COTS-based systems, re-establish system state, and bring the system back from a frozen or crashed condition.

In these architectures, a typical core, perhaps built around a Mongoose or PowerPC radiation-hardened processor running VxWorks, would control spacecraft power, communications to and from Earth, and overall spacecraft operation. It would also cause a reset or reboot of the COTS system or take over spacecraft operation long enough to weather a severe environmental condition that the COTS system cannot handle.

Ongoing studies will determine how the COTS system should interface with the hardened core, what degraded-mode operations such an architecture would afford, and what alternative radiation-hardened components could be used for the core. The challenge for designers is to design a system in which a low-performance hardened core does not hinder the higher-performing COTS-based computing system. Figure 4 shows one such system architecture.

FPGAs

Recent advances in field-programmable gate array technology offer unprecedented levels of performance at extremely low power, mass, and volume. FPGA designers can now combine on a single chip capabilities that once took a board's worth of parts. For example, Xilinx's inclusion of multiple PPC405 cores and a host of soft IP for everything from memory and I/O controllers to fast Fourier transform engines provides a low-cost approach to plug-and-play system-on-chip design. In some applications, FPGAs provide a performance-to-power ratio up to two orders of magnitude greater than current COTS processors.

In-system reconfigurability also makes FPGAs well suited for remote systems that require long system life but cannot be repaired externally. Developers can use FPGAs to build computers and microcontrollers, to implement high-performance reconfigurable coprocessors for augmenting low-performance but highly reliable radiation-hardened processors, and to embed digital signal processors in a variety of subsystems.[2] In the event of failure, the FPGAs could theoretically be reconfigured around the faulty element.

Unfortunately, FPGAs share the same susceptibility to SEU effects that plague other COTS CMOS components. In FPGAs, however, overcoming the effects can be more difficult. In addi-

Early studies clearly show that FPGA-based processing systems for spacecraft and robotics offer significant advantages.

tion to the FPGA's logic components, the large configuration memories in SRAM- or flash-based FPGAs can fall prey to SEU-induced errors. Many fault-tolerance techniques cannot provide reliable operation under these circumstances.

Initial results of NASA studies into potential FPGA uses in space-based systems have produced fault tolerance approaches capable of handling both logic and configuration faults. Engineers at JPL have proposed interfacing libraries of specialized fault-tolerant algorithms with standard COTS FPGA configuration and design tools to build fault-tolerant versions of signal- and data-processing modules. Other studies, conducted by JPL and UCLA engineers, have proposed hardware-based fault detection and handling methods that can provide real-time fault handling in FPGA-implemented processors. These early studies show that FPGA-based processing systems for spacecraft and robotics offer significant advantages over systems built from radiation-hardened or even state-of-the-art COTS processors.

COTS SYSTEM SOFTWARE

Offering significant advantages for space-based systems, COTS software can provide cost-effective development of computing systems that can gather or derive knowledge and return it to Earth.

Advanced real-time operating systems

Several RTOS vendors have developed products with features useful to space-based fault-tolerant systems. Most vendors now offer RTOS products with memory management and kernel or application isolation, while some also provide process-handling features.

Most of these products are proprietary, however, and difficult to modify for SEU protection. For this reason, some NASA researchers have turned to open source software such as Linux. Given that many scientific and engineering codes are developed on Linux workstations, a Linux-based onboard processing system would allow straightforward transition of development codes to flight systems. This would lower costs, simplify development, and reduce porting errors. A real-time, fault-tolerant Linux OS hosted on an REE system, for example, would be equivalent to a Linux cluster for the programmer, but would still provide reliable, real-time, deterministic behavior for the spacecraft engineer.

A Goddard Space Flight Center team investigated using Linux in the FlightLinux project (flightlinux.gsfc.nasa.gov/), which demonstrated the software on a breadboard system. They also built a driver to let a FlightLinux system use bulk memory in place of disk storage. The project has plans to test the FlightLinux build, load it onto an operating satellite, and test the onboard operation. Either this Linux version or another similar build will likely fly in the near future as a demonstration.

Recently, researchers have made progress in developing fault-injection-based OS test systems, with fairly promising results.[3,4] For the most part, it seems that error propagation across program boundaries is fairly limited and that most faults have little effect. Although still in progress, this work has paved the way for future researchers to develop test and evaluation tools for real-time, fault-tolerant system software. It should also give OS developers feedback on how their software reacts to SEU faults. Perhaps more importantly, an open system such as Linux can be modified relatively easily to provide fault detection and handling. Thus, a real-time, fault-tolerant Linux version may soon be a reality.

Fault-tolerant systems development tools

The REE project developed an initial version of a toolset that can systematically and cost-effectively evaluate detailed fault-susceptibility data and localize specific modules, data structures, coding techniques, and hardware-usage patterns that cause high error rates and unacceptable fault behavior. These tools enable a new approach to system test, verification, and validation through massive automated fault injection, system behavior capture, and analysis.[5,6]

REE also developed techniques for fault tolerance of scientific codes and parallel processing clusters that operate reliably in the presence of moderate SEU rates. Examples include a COTS-compatible software library of fault-tolerant scientific subroutines and parallel processing codes, as well as an early version of a fault-tolerant middleware layer for cluster computers that provides soft real-time scheduling, reliable communications, and system reconfiguration management, all with extremely low overhead.[7-9]

REE ported these analysis tools and products to a range of testbed hardware and software platforms, including PPC603, PPC750, G4, Pentium, and Sun workstations, and the Lynx, Linux, and Solaris operating systems. They proved to be applicable to, and easily ported across, this diverse set of testbeds. The significance of these findings should not be overlooked. The continual, rapid evolution of COTS systems complicates their use,

and maintaining compatibility and interoperability by porting tools and techniques to emerging COTS technologies presents a significant challenge to the use of COTS-based computing.

In addition to these software-implemented fault analysis and mitigation products, REE also developed radiation-effects test tools and methods to characterize the radiation susceptibility of COTS computers more rapidly and with more precision than previously possible.[10,11] These tools have been used on three processor generations from four major vendors. Data now exists not only for microprocessors, bridge chips, and custom I/O chips' gross susceptibility, but also for processor subsystems such as clock distribution networks, onboard caches, cache controllers, and MMUs.

This information provides insight into the vulnerabilities of new COTS components and systems. The radiation test tools can be used to test new parts in a matter of weeks rather than months or years, allowing them to keep up with COTS parts evolution. The tools also include a model for estimating computer system and subsystem SEU-induced soft-error rates and types in most space environments.[12]

The testing results have been surprisingly good: SEU thresholds have not, as expected, been reduced—they have either stayed the same or become slightly better with succeeding CMOS process generations. TID tolerance levels, meanwhile, have risen dramatically, to the point where TID is no longer a problem for most components in most environments.

ALTERNATIVE ARCHITECTURES

Rapid advances in semiconductor technology have opened another path to future spaceborne computing.[13] A new method, which allows integration of DRAM cells and CMOS logic devices on the same silicon die, could deliver improved efficiency, performance, scalability, power consumption, and reliability. One example of incorporating both logic and memory on a single chip is processor-in-memory (PIM).[14]

PIM tightly couples logic with the DRAM memory at the row buffer and sense amplifiers. JPL's Gilgamesh project seeks to develop a family of PIM-based parallel architectures for memory-intelligence-network devices. MIND will be used for both spaceborne and ground-based high-performance computing systems.[15] Developers will leverage this new fabrication technique to incorporate innovative concepts in processor-architecture designs based on dynamic adaptive resource-management methods. The Gilgamesh team expects the new component to accelerate application of high-performance computing in space and anticipates expanding its use to a broader range of computational challenges on the ground.

AUTONOMY

Autonomous spacecraft promise reduced cost, improved efficiency, and the ability to handle larger, more complex missions.

Advanced spaceborne computing enables increasingly autonomous spacecraft. Autonomy brings with it the promise of reduced cost, improved efficiency, and the ability to handle larger, more complex missions by enhancing three main areas: spacecraft control, mission operations, and science data processing and understanding. A success story in spacecraft control was the 1999-2001 Deep Space 1 spacecraft,[16] which demonstrated the ability of a spacecraft to plan its own mission from high-level goals, to navigate and successfully rendezvous with a comet, and to recognize an unexpected situation it could not deal with and call back for instructions rather than make an arbitrary, potentially mission-threatening decision.

Juggling tradeoffs

Currently, downloading data from a spacecraft poses a tradeoff between computation and communication. This tradeoff must balance the computational resources of a spacecraft against the load on the space communications channels. For example, data compression can reduce the transmitted data volume at the expense of increased computational overhead.

Scene prioritization provides one example of a near-term onboard autonomy computation that would use the constrained downlink more effectively.[17] This technique selects images observed by a science craft and prioritizes them for downlink in an order similar to that which a scientist would assign. It selects high-priority images for high-quality transmission, then chooses lower-priority images for low-fidelity transmission or deletion.

Onboard 3D scene reconstruction provides another autonomy example. It encodes the information contained in different views of an interesting scene into a unified 3D model, reducing transmitted data volume by removing the redundancy present in different views of the same scene.

Potentially synergistic, both prioritization and reconstruction are being examined for suitability on a 2009 mission to Mars.

Precision and agility

Many new missions, such as surface missions to Mars and planned missions to other planets,

These technologies will enable exploration of hostile, unpredictable environments such as Europa's subsurface oceans or Titan's atmosphere.

moons, asteroids, and comets will be built on advanced computing and new autonomy techniques. Such missions require precision landing, with real-time obstacle avoidance and landing-site optimization. Meeting these demands requires developing autonomous, real-time, multisensor fusion and image analysis at a level that cannot be performed currently.

Autonomous, real-time, high-speed navigation across the Martian surface provides an example of advanced robotic navigational processing for future missions. An autonomous rover, if built today using state-of-the-art radiation-hardened computing components, would be limited to one-sixth its maximum mechanically sustainable speed due to the in-transit image processing required to perform path planning and obstacle avoidance. Adding COTS-based hardware and software with an autonomous navigation component could increase the rover's speed to its maximum mechanical limit and let it perform on-the-fly, in-transit, science data collection and processing.

Onboard decision making

To respond autonomously to short-lived science events, onboard decision making systems use feature, change, and unusualness-detection software to analyze science data. The software detects features such as volcanic eruptions, sand dune migration, the growth and retreat of ice caps, and crustal deformations, then triggers the downlink of only selected data.[18] Subsequently, these onboard science algorithms can trigger onboard mission replanning to change the spacecraft's observation plan based on mission priorities, spacecraft health and resources, and the situational scenario. This new plan can then be executed by goal- or task-oriented execution systems that adjust it to succeed despite runtime anomalies and uncertainties. Together, these technologies will enable exploration of hostile, unpredictable environments such as Europa's subsurface oceans or Titan's atmosphere.

The 2003 Three Corner Sat mission[19] will use onboard data validation, replanning, and robust execution to maximize science return. The 2006 Autonomous Sciencecraft[20] mission will use onboard feature and change-recognition software with onboard planning and execution software to enable autonomous retargeting and downlink of only the most valuable science data. These pathfinding missions will enable even more autonomous future missions.

The Advanced Information Systems Technology, Advanced Component Technology, New Millennium, and Mars Technology programs all actively fund the development of technologies for space-based scientific research. NASA also participates in the Small Business Innovative Research and Small Business Technology Transfer programs, which extend annual solicitations for more general technologies. These and other programs are funding the development of advanced computing technologies. A new generation of onboard processing based on these ideas will usher in a new era of autonomous space exploration, enabling missions with dramatically increased science return, and helping to ensure that space exploration continues to expand human understanding. ■

Acknowledgments

We thank Michele Judd for her help in organizing some of the material in this article and John Klein for his reviews and constructive critiques. Portions of this research were carried out at the Jet Propulsion Laboratory, California Institute of Technology, under a contract with the National Aeronautics and Space Administration.

References

1. R. Some and D. Ngo, "REE: A COTS-Based Fault Tolerant Parallel Processing Supercomputer for Spacecraft Onboard Scientific Data Analysis," *Proc. Digital Avionics Systems Conf.*, IEEE Press, 1999, pp. 7.B.3-1 to 7.B.3-12.
2. N. Aranki, A. Moopenn, and R. Tawel, "Parallel FPGA Implementation of the Split and Merge Discrete Wavelet Transform," *Proc. 12th Int'l Conf. Field Programmable Logic and Applications* (FPL 2002), Springer-Verlag, 2002, pp. 740-749.
3. J.C. Fabre et al., "Assessment of COTS Microkernels by Fault Injection," *Proc. 7th IFIP Working Conf. Dependable Computing for Critical Applications* (DCCA-7), IEEE Press, 1999, pp. 25-44.
4. H. Madeira et al., "Experimental Evaluation of a COTS System for Space Applications," *Proc. Int'l Conf. Dependable Systems and Networks*, IEEE CS Press, 2002, pp. 325-330.
5. R. Some et al., "A Software-Implemented Fault Injection Methodology for Design and Validation of System Fault Tolerance," *Proc. Int'l Conf. Dependable Systems and Networks*, IEEE CS Press, 2001, pp. 501-506.
6. R. Some et al., "Fault Injection Experiment Results in Spaceborne Parallel Application Programs," *Proc.*

IEEE Aerospace Conf., IEEE Press, 2002, pp. 5-2133 to 5-2148.
7. M. Turmon et al., "Tests and Tolerances for High-Performance Software-Implemented Fault Detection," to be published in *IEEE Trans. Computers*, 2003.
8. J.A. Gunnels et al., "Fault-Tolerant High-Performance Matrix Multiplication: Theory and Practice," *Proc. Int'l Conf. Dependable Systems and Networks*, IEEE Press, July 2001, pp. 47-56.
9. M. Li et al., "Design and Validation of Portable Communication Infrastructure for Fault-Tolerant Cluster Middleware," *Proc. IEEE Int'l Conf. Cluster Computing*, IEEE CS Press, 2002, pp. 266-274.
10. G. Swift et al., "Single-Event Upset in the Power PC750 Microprocessor," *IEEE Trans. Nuclear Science*, vol. 48, no. 6, pp. 1822-1827.
11. J. Howard et al., "Update on Total Dose and Single Event Effects Testing of the Intel Pentium III (P3) and AMD K7 Microprocessors," *Proc. 3rd Military and Aerospace Applications of Programmable Logic Devices Conf.*, Cooperative Research Centre for Satellite Systems, 2001.
12. R. Some and A. Karapetian, "Radiation Fault Modeling and Fault Rate Estimation for a COTS-Based Space-borne Supercomputer," *Proc. IEEE Aerospace Conf.*, IEEE Press, 2002, pp. 5-2121 to 5-2132.
13. T. Sterling, D.S. Katz, and L. Bergman, "High-Performance Computing Systems for Autonomous Spaceborne Missions," *Int'l J. High-Performance Computing Applications*, vol. 15, no. 3, 2001, pp. 282-296.
14. P.M. Kogge et al., "Combined DRAM and Logic Chip for Massively Parallel Systems," *Proc. 16th IEEE Conf. Advanced Research in VLSI,* IEEE CS Press, 1995, pp. 4-16.
15. L. Bergman and T. Sterling, "A Design Analysis of a Hybrid Technology Multithreaded Architecture for Petaflops Scale Computation," *Proc. 13th Int'l Conf. Supercomputing*, ACM Press, June 1999, pp. 286-293.
16. K. Rajan et al., "Remote Agent: An Autonomous Control System for the New Millennium," *Proc. Prestigious Applications of Intelligent Sys. Subconference, 14th European Conf. Artificial Intelligence*, IOS Press, 2000, pp. 726-730.
17. R. Castaño et al., "Sample Selection and Virtual Return: Onboard Scene Prioritization and 3D Reconstruction," *IND Technology & Science News*, Sept. 2002; http://tmot.jpl.nasa.gov/Program_Overview_Information/IND_Program_News/INDnews16.pdf.
18. A. Davies et al., *Autonomous Sciencecraft Constellation Science Study Report, New Millenium Program Space Technology 6 (NMP-ST6),* Techsat-21 Mission; http://ase.jpl. nasa.gov/asc_science_report.pdf.
19. S. Chien et al., "Onboard Autonomy on the Three Corner Sat Mission," *Proc. 2001 Int'l Symp. Artificial Intelligence, Robotics, and Automation for Space,* 2001, IEEE Press, http://www-aig.jpl.nasa.gov/public/planning/papers/3CS-ISAIRAS-final.pdf.
20. S. Chien et al., "The TechSat 21 Autonomous Sciencecraft Constellation," *Proc. 2001 Int'l Symp. Artificial Intelligence, Robotics, and Automation for Space,* IEEE Press, 2001; www.cnes.fr/colloque/prospective_spatiale/ Satellites-techsat21.pdf.

Daniel S. Katz *is the supervisor of the Parallel Applications Technologies Group within the Exploration Systems Autonomy section (367) at the Jet Propulsion Laboratory, California Institute of Technology. His research interests include numerical methods, algorithms, fault-tolerant computing, and programming applied to supercomputing, parallel computing, cluster computing, and embedded computing. Katz received a PhD in electrical engineering from Northwestern University. He is a senior member of the IEEE. Contact him at Daniel.S.Katz@jpl.nasa.gov.*

Raphael R. Some *is a microelectronics technologist for the New Millenium Program at the Jet Propulsion Laboratory, California Institute of Technology. His research interests include fault-tolerant computing, computer architectures, and microelectronics technologies. Some received a BS in electrical engineering from Rutgers University. Contact him at rsome@jpl.nasa.gov.*