plex flight profile to be simulated, as well as ambient conditions and deterioration level of the engine. C-MAPSS40k has three actuators: fuel flow, variable stator vanes, and variable bleed valve. The three actuators enable off-nominal operation, which is not possible with simulations that have fuel flow as the sole actuator, since in those simulations the other actuators are implicit and assumed to operate nominally. The simulation is modular to allow users to redesign or replace components such as the engine controller or turbomachinery components without having to modify the rest of the simulation. It also enables the user to view and save any signal in the engine or controller. The package has the capability to create and validate a linear model of the engine at any operating point. Linear models can be used for control design, and C-MAPSS40k lends itself well to implementation and evaluation of advanced control designs as well as to diagnostic and prognostic system development. The simulation can be run in real time and can therefore be integrated into a flight simulator with a pilot in the loop for testing.

C-MAPSS40k fills the need for an easy-to-use, realistic, transient simulation of a medium-size commercial turbofan engine with a representative controller. It is a detailed component level model normalized to the base clock frequency. Jitter, rms or peak, absolute units, or normalized, helps the user manage relationships among phase jitter crest factor, frequency jitter crest factor, cycle-to-cycle jitter (CCJ) crest factor, and cycle-to-cycle jitter among phase jitter crest factor, frequency jitter crest factor, and cycle-to-cycle jitter (CCJ) crest factor.
Software Displays Data on Active Regions of the Sun

The Solar Active Region Display System is a computer program that generates, in near real time, a graphical display of parameters indicative of the spatial and temporal variations of activity on the Sun. These parameters include histories and distributions of solar flares, active region growth, coronal mass ejections, size, and magnetic configuration.

By presenting solar-activity data in graphical form, this program accelerates, facilitates, and partly automates what had previously been a time-consuming mental process of interpretation of solar-activity data presented in tabular and textual formats. Intended for original use in predicting space weather in order to minimize the exposure of astronauts to ionizing radiation, the program might also be useful on Earth for predicting solar-wind-induced ionospheric effects, electric currents, and potentials that could affect radio-communication systems, navigation systems, pipelines, and long electric-power lines.

Raw data for the display are obtained automatically from the Space Environment Center (SEC) of the National Oceanic and Atmospheric Administration (NOAA). Other data must be obtained from the NOAA SEC by verbal communication and entered manually. The Solar Active Region Display System automatically accounts for the latitude dependence of the rate of rotation of the Sun, by use of a mathematical model that is corrected with NOAA SEC active-region position data once every 24 hours. The display includes the date, time, and an image of the Sun in Hα light overlaid with latitude and longitude coordinate lines, dots that mark locations of active regions identified by NOAA, identifying numbers assigned by NOAA to such regions, and solar-region visual summary (SRVS) indicators associated with some of the active regions.

Each SRVS indicator is a small pie chart containing five equal sectors, each

µShell Minimalist Shell for Xilinx Microprocessors

µShell is a lightweight shell environment for engineers and software developers working with embedded microprocessors in Xilinx FPGAs. µShell has also been successfully ported to run on ARM Cortex-M1 microprocessors in Actel ProASIC3 FPGAs, but without project-integration support. µShell decreases the time spent performing initial tests of field-programmable gate array (FPGA) designs, simplifies running customizable one-time-only experiments, and provides a familiar-feeling command-line interface. The program comes with a collection of useful functions and enables the designer to add an unlimited number of custom commands, which are callable from the command-line. The commands are parameterizable (using the C-based command-line parameter idiom), so the designer can use one function to exercise hardware with different values. Also, since many hardware peripherals instantiated in FPGAs have reasonably simple register-mapped I/O interfaces, the engineer can edit and view hardware parameter settings at any time without stopping the processor.

µShell comes with a set of support scripts that interface seamlessly with Xilinx’s EDK tool. Adding an instance of µShell to a project is as simple as marking a check box in a library configuration dialog box and specifying a software project directory. The support scripts then examine the hardware design, build design-specific functions, conditionally include processor-specific functions, and complete the compilation process. For code-size constrained designs, most of the stock functionality can be excluded from the compiled library.

When all of the configurable options are removed from the binary, µShell has an unoptimized memory footprint of about 4.8 kB and a size-optimized footprint of about 2.3 kB. Since µShell allows unfettered access to all processor-accessible memory locations, it is possible to perform live patching on a running system. This can be useful, for instance, if a bug is discovered in a routine but the system cannot be rebooted: µShell allows a skilled operator to directly edit the binary executable in memory. With some forethought, µShell code can be located in a different memory location from custom code, permitting the custom functionality to be overwritten at any time without stopping the controlling shell.

This work was done by Thomas A. Werne of Caltech for NASA’s Jet Propulsion Laboratory. Further information is contained in a TSP (see page 1).

This software is available for commercial licensing. Please contact Daniel Broderick of the California Institute of Technology at danielb@caltech.edu. Refer to NPO-47495.