One of the biggest challenges facing embedded device vendors is building custom hardware and software that meets expected performance, cost, and feature requirements. A major contributor to this challenge is the need to write new software to enable and control custom hardware. According to a 2002 survey by Embedded Market Forecasters (Minimizing Risk and Maximizing Opportunities in the New Economy, Krasner and Andrews), many final embedded designs fall short of expectations in key categories, such as performance, functionality, and features. According to the research, 71.5 percent of all designs were not within 30 percent of pre-design performance expectations. Equally alarming, embedded device vendors who build custom software suffer from project completion delays averaging four months. Fortunately, there is new technology available that overcomes many of the obstacles associated with embedded systems development. This article provides an overview about automated embedded systems diagnostics, and how this new technology helps meet the challenge of designing custom software and hardware for embedded devices.

Obstacles of an effective diagnostics process
While it may seem plausible that running the final software application on custom hardware is sufficient for testing designs, this assumption has flaws. In many instances, the application is incomplete or unstable when the hardware design is completed and ready for verification. This methodology can also be problematic because engineers often build applications with the assumption that the hardware is functioning properly. Applications are typically not well suited for diagnostics, i.e., they cannot easily break down tasks to identify design flaws more readily. A hardware problem may trigger an application failure but give no indication of the actual problem. Therefore, the system might simply crash.

In response to this challenge, companies have attempted to build proprietary diagnostics software that minimizes the efforts of supporting new hardware. This task requires one or more senior software engineers who also know hardware. Unfortunately, these individuals are not easy to hire or easy to spare from other development efforts.

To verify hardware designs successfully, software must be able to exercise all components. Additionally, the software must verify that each component operates in conjunction with the other components connected via the internal bus architecture. Developers must test all Integrated Circuits (ICs) FPGAs, and ASICs for adherence to the hardware bus specification, such as PCI, VME, or others. Another crucial test point involves the determination of performance characteristics. For the purposes of hardware validation during functional or manufacturing tests, it is essential to have test suites that validate the entire board automatically, uniquely identifying and logging individual errors.

Automated diagnostic software takes the driver's seat
New, off-the-shelf diagnostic software designs slash the cost of building custom embedded systems. This technology provides a ready-to-run diagnostics platform for embedded systems, automating the time-intensive steps required to support custom hardware. To efficiently verify new hardware designs, software must provide comprehensive verification and fault isolation capabilities. The solution is an automated test suite that fully exercises all board components while providing individual test execution for fault isolation. Complete test coverage is also essential, for example, exercising a board as it might function in demanding and disparate customer environments. Customizing tests in the form of scripts built from an extensive library of test primitives, and incorporating advanced test services, such as data streaming and data validation, might accomplish this exercise.

Building the bridge to custom configurations
A major impediment to commercialization of embedded software, including diagnostics, is the sheer variety of board designs. A design’s processor, memory, bus architecture, and I/O peripherals can all vary. Finding a way to support the myriad permutations of hardware components is essential to developing embedded systems software efficiently.

The process of automatically creating custom software to support custom hardware begins by creating libraries that support the processor, bus architecture, and other components. Designers build these libraries while adhering to an overall layered architecture, exposing individual interfaces for use by the higher-level libraries and services. Designers also expose the interfaces to these libraries to the diagnostics user who can execute functions at the interpreter or retrieve them from internally written code. This configuration allows for the extension of the overall system capabilities to support proprietary ASICs, FPGAs, or other ICs.
Steps for creating custom diagnostics

The first step in creating diagnostics for a custom hardware design is acquiring the hardware configuration data for the particular platform. This step requires processor and component types of data, as well as hardware layout data, such as the memory map. Many times, a hardware specification document already has this information. Figure 1 illustrates the process required to create the final diagnostics application.

The next step is to use the configuration data as input into a process that pulls from three primary libraries: executive, components, and services. The executive library provides the operating system and command interpreter. The components library contains software to initialize and control all chips and buses on the hardware platform. The services library provides higher-level test applications that mimic the final application in behavior, but take into account a large number of variations. Designers then create a customized intermediate library for the target platform that supports the basic input/output functionality required by the executive and higher-level software components. With these items in place, designers can then build a final executable image that they can download to the target and program into the appropriate memory source, such as Flash memory.

“Finding a way to support the myriad permutations of hardware components is essential …”

When the target system resets, or cycles the power, the diagnostics interpreter executes using a serial port for communications to the end user via a terminal emulation program. From this interface, the end user has the ability to interactively execute test cases, either one at a time, or in one or more groups of test cases. The designer can interactively verify the new design or run an endless looping service for compliance testing, including measuring emissions, temperature, power ranges, and more.

With this process, a user need only deliver a hardware specification, or at a minimum, the hardware configuration data, to work with off-the-shelf diagnostic software. If one needs to extend the capabilities of the diagnostics platform, one can write custom routines and automatically include them in the platform for execution at the interpreter. One can archive scripts written for a platform and use them in the future on similar platforms that take advantage of the latest technology. For companies collaborating in the same fashion, they can extract custom scripts from the target, e-mail them to the collaborating site, download them to the target, and execute them. This time-saving feature allows execution of the same test on various platforms for problem isolation. Because of the relatively simple interface, support staff or even customers can use these scripts to report problems in the field.