Testing and Correcting Embedded Processors

STEPHEN A. EDWARDS

Associate Professor of Computer Science

A car takes a curve too fast. Before it spins out, its stability control system kicks in. Its microprocessors calculate and recalculate the right amount of force to apply to each wheel, adjusting the brakes many times per second until the car comes under control. Such critical systems often juggle several events at once. A car's stability system must calculate speed, momentum, spin, and dozens of other variables before each application of the brakes.

Unfortunately, its embedded processors can only perform one task at a time. To get around the problem, Associate Professor Stephen Edwards said, programmers slice tasks into many little pieces and have the processor hop between slices at such blindingly fast speeds, it presents the illusion of simultaneity.

This illusion—called concurrency—comes at a cost. Most programmers use the C language to code embedded processors. This involves lots of repetitive programming, and errors can creep in. "It's like writing a phone book by hand. People could do it, but there would be lots of mistakes," Edwards said.

Second, concurrent C programs are hard to test. C programs must be translated, or compiled, into the ones-and-zeros language of processors. Their sliced-up nature makes them hard to translate, model, and test. "The only way to tell if they will run fast enough to handle critical calculations is to test and retest programs until they appear to work," he said.

Edwards has solutions for both issues. First, he has developed a language, Software-Hardware Integration Medium (SHIM), which simplifies programming concurrent events. SHIM reduces the errors that creep into repetitive programs. "We developed algorithms that automate all the bookkeeping necessary to manage simultaneous events," he explained.

He also created a customized compiler that generates testable code. It takes C-like programs and translates the concurrent parts of the programs into a (very long) series of sequential commands. It then re-compiles them back into C.

"This lets you test your program in a model to check its speed and reliability and make improvements," Edwards said. "This will lead to more reliable behavior and maybe fewer huge recalls when embedded processors fail."

"Embedded processors hide in the environment. As hardware plummeted in price, it became possible to put them everywhere. I ask students how many processors they own. They may count their computers or smart phones, but miss their coffee makers, air conditioners, and cars. We made 10 billion embedded processors in 2008."

B.S., California Institute of Technology, 1992; M.S., California (Berkeley), 1994; Ph.D., 1997