DEPEND: A Simulation Environment for System Dependability Modeling and Evaluation

Center for Reliable and High-Performance Computing
Coordinated Science Laboratory
University of Illinois at Urbana-Champaign

DEPEND is an evolving simulation-based environment for the evaluation of designs from functional and dependability viewpoints.

As shown in Figure 1, DEPEND supports the VHDL hardware descriptive language as well as the C++ programming language. In addition, DEPEND provides a graphical modeling facility to allow interactive model construction. Designs under evaluation can be functional or structural descriptions of the designs. The Task & Environment Manager provides an easy-access, window-based manual page and a tool-kit to allow users to create, modify, and simulate models without leaving the DEPEND environment.

At the core of DEPEND are simulation engines supported by a fault injector, a set of fault dictionaries, and component libraries. The component libraries contain model building blocks with detailed functional descriptions and characteristics; the fault dictionaries embody possible fault effects of given fault types, devices, and circuits; and the fault injector provides mechanisms to inject faults.

DEPEND employs a hierarchical modeling and simulation approach that is intended to allow design evaluation starting from the device-level physical construct to the chip-level functional behavior, and up to the system-level dependability. With the fault dictionaries available at different abstraction levels, faults can be injected at the lower level, such as device or gate, and effects are evaluated at the higher level, such as chip or system. Figure 2 illustrates the concept of DEPEND simulation hierarchy.

In Figure 2, a fault based on a current surge fault model is injected at the device level, and the fault effect (i.e., the error) is evaluated at the system level (i.e., software behavior). In this example, the fault is injected in a transistor and results in an output 1 (instead of 0). This value in turn results in a value of 08 (instead of 00) generated by one of the functional units on the chip and, consequently, results in an execution path with the non-zero route as shown in Figure 2. This effect could result in a wrong output, or in a more severe case, a system failure.

This work was supported in part by the Advanced Research Projects Agency (ARPA) under contract DABT63-94-C-0045 and by NASA grant NAG 1-613, in cooperation with the Illinois Computer Laboratory for Aerospace Systems and Software (CLASS). The content of this paper does not necessarily reflect the position or policy of these agencies, and no endorsement should be inferred.