Chapter 12

Interfacing with C / C++

In Verilog, you can communicate with C routines using the Programming Language Interface. With the three generations of the PLI: TF (Task / Function), ACC (Access), and VPI (Verification Procedural Interface), you can create delay calculators, connect and synchronize multiple simulators, and add debug tools such as waveform displays. However the PLI's greatest strength is also its greatest weakness. If you just want to connect a simple C routine using the PLI, you need to write dozens of lines of code, and understand many different concepts such as synchronizing with multiple simulation phases, call frames, and instance pointers. Additionally, the PLI adds overhead to your simulation as it copies data between the Verilog and C domains, in order to protect Verilog data structures from corruption.

SystemVerilog introduces the Direct Programming Interface (DPI), an easier way to interface with C, C++, or any other foreign language. Once you declare or "import" the C routine with the `import` statement, you can call it as if it were any SystemVerilog routine. Additionally, your C code can call SystemVerilog routines. With the DPI you can connect C code that reads stimulus, contains a reference model, or just extends SystemVerilog with new functionality. Currently SystemVerilog only supports an interface to the C language. C++ code has to be wrapped to look like C.

If you have a SystemC model that does not consume time, and that you want to connect to SystemVerilog, you can use the DPI. SystemC models with time-consuming methods are best connected with the utilities built into your favorite simulator.

The first half of this chapter is data-centric and shows how you can pass different data types between SystemVerilog and C. The second half is control centric, showing how you can pass control back and forth between SystemVerilog and C. While the actual