SystemVerilog 3.1a Donations

Doug Warmke
Model Technology, Inc.
18 September, 2003

Separate Compilation Proposal

- Motivations
  - SystemVerilog has a rich type system. Allow sharing of common types with modern data encapsulation methods (i.e. “package” or “namespace”).
  - Reduce reliance on $root as a way of holding design-wide definitions and data.
  - Provide a safe, reliable way for IP providers to manage IP-wide definitions and data.

Problems with existing SV 3.1

- For large projects, declarations in $root share the same scope and thus may inadvertently collide
- For systems supporting separate compilation, users are restricted in how they can manage their data in $root.
  - Should one use `include for type definitions?
  - What about instantiating a reg in $root?
  - Do definitions in $root from one compile run persist into the next run?

Proposal Summary

- Create a new kind of top level scope called a *package*.
- Make use of an *import* statement to make the package contents visible during a given compilation.
- Ensure that the semantics of importing a package allow maximizing semantic checks at compilation time (rather than deferring them until elaboration time).

DPI Exported Task Proposal

- SV 3.1 allows C code to directly call SystemVerilog functions. Users have requested the same capability for SystemVerilog tasks.

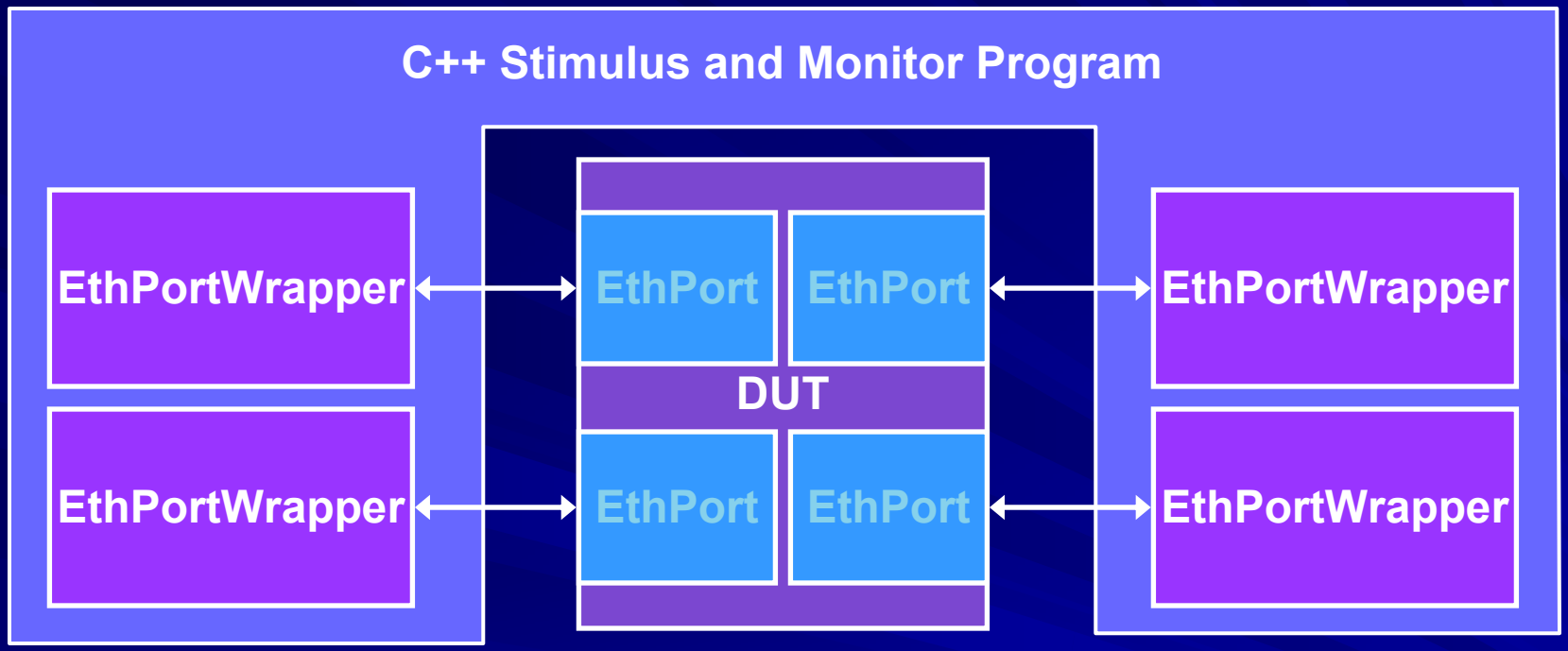
Motivation behind exported tasks

- High level C test environments
  - Users want to write test environments that run under a C imported function.
  - The test environment calls back into the SV domain in order to execute hardware transactions in the DUT.
  - Such transactions may consume time, and thus require implementation as SV tasks.
  - Think of this as a C implementation of a testbench's main stimulus-producing initial block.

More motivation for exported tasks

- Mixed C and SystemVerilog systems
  - Some simulation tools support a mix of C and SV processes running under a common scheduler.
  - Users can implement a multi-threaded transaction-level C environment that interacts with SV DUT via inter-language function calls.
  - Exported SystemVerilog tasks are a natural element of such systems.

Ethernet Packet Router Example



EthPort module implemented in SV; instantiated multiple times

EthPortWrapper implemented in C++; instantiated multiple times

All communication done at transaction level (function calls and data packets)

When DUT is implemented in RTL, the notion of time is introduced, and time consuming tasks are needed to implement C++ calls into the DUT

Proposal Summary

- Allow export declarations for tasks
- Add a new property to DPI import declarations: *timeconsuming*.
  - Imported functions which call time consuming SV tasks must be specified as *context timeconsuming* functions.
  - Imported functions which call SV tasks that do not execute any delay control or event control statements are simply *context* functions.