Using SystemC to Model Asynchronous Communication at Different Levels of Abstraction

Tobias Bjerregaard (tob@imm.dtu.dk), Shankar Mahadevan (sm@imm.dtu.dk), Jens Sparsø (jsp@imm.dtu.dk)
Inst. for Informatics and Mathematical Modeling (IMM), Techn. Univ. of Denmark (DTU), Denmark

The aim of the presentation is to assess the possibilities in using SystemC to model asynchronous inter-module communication at various levels of abstraction.

SystemC excels in its usefulness to model a system level design, while still supporting synthesizable RT level hardware descriptions (like traditional HDLs such as VHDL and Verilog). Thus a seamless refinement of a design can occur, during which each part of the design is implemented independently, without changing the other parts and without changing the test environment. The more traditional methodology, during which the functional specification might be done in one language (perhaps Perl, C++ or JAVA), while the implementation might be done in a traditional HDL, is cumbersome.

In a top-down design methodology, it is useful to make use of abstract communication methods, when starting the design process at the top level. In order to make the seamless transition from system-level specification to RTL implementation independently for different parts of the system, these communication methods should support mixed-mode communication. One module might still be communicating using abstract methods while another might have crystallized into RTL implementing a specific communication protocol. This is all the more true in an asynchronous design, since the communication between modules is an important key to correct operation.

We have looked at ways of using SystemC to model asynchronous communication channels at varying levels of abstraction. Our aim was to design and test a communication channel supporting transactions through an abstract CSP-like interface (implementing send and receive commands) as well as through one of many specific interfaces (e.g. 4-phase-bundled-data implementing actual request, acknowledge and data ports). It should be possible to make the choice of abstraction level independently for each end of the channel.

We made use of SystemC’s object oriented structure by first designing a base channel, which implements the abstract send and receive interfaces. We then designed a number of channels, which inherit the abstract interfaces from the base channel as well as implement a specific protocol. Below is a description of a typical design flow using our channels:

1. The designer creates a system-level design using the base channel. All modules communicate using abstract CSP-like communication commands.
2. As each module is implemented, a communication protocol is chosen. The base channel is switched with a particular sub-channel, which implements that particular protocol.
3. When all modules are implemented, the abstract methods are not used anywhere. The channels are merely sets of wires used by the communication protocols. This should be recognized by the synthesize tool, and hence be dissolved.

For demonstrating the use of our channels, we have implemented an asynchronous pipeline. First the pipeline was implemented at an abstract level, then each element in the pipeline was independently implemented as RTL code, and a specific protocol was chosen. The demonstrator showed that it was possible to continue mixed-mode simulation, and that the design was seamlessly testable throughout the transition from abstract to RTL.

The major draw back of the presented method is the fact that SystemC does not allow unconnected ports. This resulted in the need to add a terminator to the ports of the channels implementing a specific protocol, for which the abstract interface is still being used.

Future work

The methodology is not quite smooth, but it works! The need to insert terminators is a little awkward. If it was allowed to let ports float, this would not be necessary.

One could design a library consisting of mixed-protocol communication channels, allowing communication between modules implementing different protocols (like 4-phase-bundled-data to 4-phase-dual-rail).

The lack of join and fork constructs in SystemC limits its use in abstract level asynchronous design. It might be an idea to look into the possibility of implementing these features in the language.