Transaction API Donation Abstract

Per Bojsen, Zaiq Technologies, Inc.

Revision 2.0, August 14, 2003

This document outlines Zaiq Technologies' *Transaction Level Modeling* (TLM) API donation. This donation is an extension to Accellera's SCE-MI standard [1] and provides a transactional abstraction on top of the SCE-API that enables efficient design verification (DV) using reusable components such as synthesizable *Bus Functional Models* (BFMs), synthesizable testbenches, and DV environment libraries. Furthermore, the TLM API is defined for both simulation and emulation modes and allows users to transition from simulation to co-emulation seamlessly while retaining the system level testbench—including tests and BFMs—unchanged.

Background

Today's complex systems present severe challenges to design verification (DV). Traditionally, DV has been based on HDL simulation but even with advances in simulation technology and host computer speed, simulations are becoming ever larger and slower. Simulation accelerators and co-emulation platforms promise to increase DV test execution speed by orders of magnitude. The efficient use of accelerators and emulators has been hampered in the past by the lack of a standard for interfacing a DV environment to these tools. The SCE-MI standard is a building block toward this purpose, providing a standard C/C++ API for transferring messages in the form of fixed-width bit vectors back and forth between the software and hardware sides of a co-emulation environment. This enables standardization at the transport level and allows a base level of inter-operability.

Goals

- Successful use of simulation acceleration and co-emulation technologies (collectively referred to as *emulation*) for DV of current generation systems requires the following capabilities:

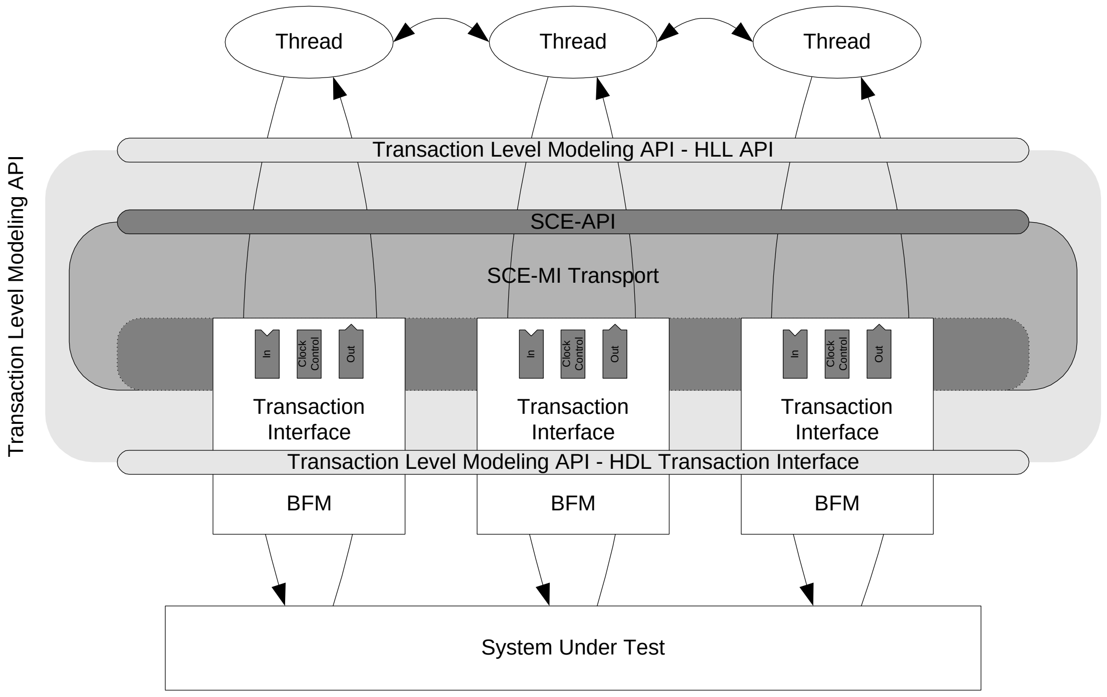
- Transaction level modeling must be used in order to reduce the communication overhead between DV environment and the *system under test* (SUT) residing in the simulator or emulator;
- Transactors and specifically BFMs must work on multiple simulator and emulation platforms without change;
- Tests and DV environment library code must work on multiple simulator and emulation platforms unchanged;
- Variable sized transactions must be handled;
- Clock control in the sense of SCE-MI's controlled and uncontrolled clocks should be abstracted such that all the user sees are SUT clocks, i.e., the equivalents of the controlled clocks, because there is no concept of an uncontrolled clock in simulation;
- Thread-level parallelism in tests that mirror the parallelism in the hardware;
- Support for multiple high level verification languages (e.g. C, C++, SystemC, SystemVerilog, e, etc.) through consistent APIs.

These needed capabilities are outside the scope of the initial SCE-MI standard. They can be and have been successfully used through the SCE-MI. They are needed by most DV projects and most test writers on a DV project, therefore efficiency and re-usability has driven encapsulation of them to form a new, higher-level API building upon SCE-MI.

Description

The TLM API provides the above by building on top of SCE-MI and bridging to multiple simulator platforms. The TLM API enables DV tests written in C/C++, SystemC, and other high-level languages—HLLs—to communicate with the SUT via HDL BFMs at the transaction level enabling efficient DV environments. The TLM API supports both simulation and emulation modes. The emulation mode is based on the SCE-API and as such can be viewed as an extension of the SCE-API or as a separate standard that builds on top of the SCE-API. Currently C, C++, and SystemC are supported HLLs. Support for other HLLs such as SystemVerilog can be added as well.

The block diagram below shows the TLM API in the context of a DV environment as implemented on an emulation platform. The HLL code, i.e., the test exists above the dark gray box, which represents the SCE-MI transport layer , while the HDL code including the BFMs and the system under test exists below the transport. The light gray shaded area represents the TLM API covered by this proposal. In simulation mode, the SCE-MI transport layer would not be present.


Thread
Thread
Thread
Transaction Level Modeling API
Transaction Level Modeling API - HLL API
SCE-API
SCE-MI Transport
In
Clock Control
Out
Transaction Interface
Transaction Level Modeling API - HDL Transaction Interface
BFM
System Under Test

The TLM API consists of the following major components

- A standardized transaction definition
- High Level Language Transaction API (HLL API)
- Transport mechanism
- Hardware-side transaction interface

In order to allow BFMs and test code from various sources to interoperate in the same transaction-based DV environment, it is necessary to standardize the concept of a *transaction*. The TLM API is built around a strict but flexible definition of what a transaction is. This definition is at the core of the TLM API and allows BFMs for similar interfaces to be interchanged without changing the test code driving the BFMs.

The HLL API encapsulates the following features

- Threading for parallelism
- Thread scheduling
- Inter-thread synchronization, communication, and signaling
- Sleep and wakeup
- Transactor-independent read and write transaction API

- Generic, user-defined transaction API
- Miscellaneous transaction API
- Interrupt handling
- Access to simulation time for logging and debugging
- Error and debug message logging functions

The transport mechanism serves as the link between the HLL API and the hardware-side transaction interface. It represents the virtual pipes between HLL threads and attached BFMs. In simulation mode it can be a PLI-based transport, or a direct kernel interface. In emulation mode, it is based on the SCE-API. Since the transport mechanism is encapsulated by the HLL API and the hardware-side transaction interface, it is transparent to the user.

The hardware-side transaction interface is based on the existing SCE-MI data and clock ports. The extended interface encapsulates logic and features often required for each BFM, including:

- Encapsulation of differences between simulation and emulation modes
- Ensures equivalency between simulation and emulation modes
- Standard interface to HLL API for BFMs
- Maintains transaction data sent to/from HLL API
- Handles zero-simulation time configuration and status transactions
- Handles transactor idling and waiting
- Handles SCE-API controlled clock for BFM in emulation mode

These features can be provided by a synthesizable HDL module, providing the extended hardware-side transaction interface identically in simulation or emulation mode, an re-usable across BFMs, projects, and vendors.

Status

The TLM API supports both simulation and emulation modes. The emulation mode is based on the SCE-API and as such can be viewed as an extension of the SCE-API or as a separate standard that builds on top of the SCE-API. The TLM API exists today and is proven on all major simulators and on the Aptix System Explorer emulator. Zaiq's system level verification platform, PREP, is based on this TLM API and has successfully been used in hundreds of projects at Zaiq's clients as well as internally at Zaiq. Currently C, C++, and SystemC are supported HLLs. Support for other HLLs such as SystemVerilog can be added as well. The hardware-side transaction interface is implemented by Zaiq's XIF core as a synthesizable Verilog module

References

[1] *Standard Co-Emulation Modeling Interface (SCE-MI), Reference Manual*, Accellera, Version 1.0, May 29, 2003.