

Virtual Prototyping of Embedded Platforms for Wireless and Multimedia

Tim Kogel
CoWare, Inc.
Tim.Kogel@CoWare.com

Matthew Braun
Motorola, Inc.
mbraun@urbana.css.mot.com

Abstract

Most of the challenges related to the development of multi-processor platforms for complex wireless and multimedia applications fall into the Electronic System Level (ESL) domain. That is to say, design tasks like embedded SW development, architecture definition, or system verification have to be addressed before the silicon or even the RTL implementation becomes available. We believe that one of the major obstacles preventing the urgently required adoption and proliferation of an ESL based design approach is the nonexistence of an efficient and intuitive methodology for modeling complex platforms. This extended abstract gives a rough overview of a modeling methodology we have developed on the basis of SystemC based Transaction Level Modeling (TLM) in order to remedy this lack of modeling competence.

1. Introduction

It is generally agreed that SystemC is the language of choice for system level modeling and that TLM nicely abstracts from the implementation details of the communication in complex SoC platforms [1]. However these basic ingredients are by no means sufficient to address today's (and not to mention tomorrow's) design challenges.

We will first elaborate on the different ESL design tasks to derive a set of corresponding TLM use-cases. These use-cases impose several general requirements on a modeling methodology, like e.g. *modeling efficiency, flexibility, and high simulation speed*. It turns out that the reusability of platform specific TLM models for multiple design tasks represents the most important requirement, simply to justify the modeling effort. Hence after a brief introduction and methodology outline we will present and discuss a general pattern for the creation of TLM models, which can be deployed to solve different ESL design tasks. Not surpris-

ingly the reusability of the platform models is founded on the orthogonalization of concerns [2]. As the major contribution of our approach we have turned this general principle into a practical recipe for the creation of SystemC based TLM models. The basic idea is to define a set of modeling objects which act as a generic layer between the user-defined behavior and the protocol specific bus interface.

This modeling methodology is currently deployed in several real-life design projects. We will present the results from a project in the wireless communication domain, where a complete multi-processor platform is modeled using the outlined approach. The final virtual prototype is used by several design groups and for different purposes like embedded Software development, or performance analysis.

2. Methodology Overview

So far four major use cases for Transaction-Level Modeling (TLM) have been identified. Each of these TLM views effectively supports one particular Electronic System Level (ESL) design task:

- The Functional View (FV) supports the creation of an executable specification of the application.
- The Architects View (AV) [3] supports the exploration of architectural alternatives to specify the platform architecture.
- The Programmers View (PV) [4, 5] supports the development and verification of embedded software.
- The Verification View (VV) [6] supports the joint verification of the hardware and software implementation in the system context.

2.1. TLM based ESL design flow

When designing an SoC the selection of a design flow is dependent on the design tasks that will be needed.

An example design flow is depicted in figure 1. In this flow functional models are created for the different algorithms that will be used in the design. These functional models are used by architects to define the right HW/SW partitioning and to explore the system architecture. The result is a high-level, functionally correct model that can serve as a executable specification for the design. From this model a SW development model can be extracted. The goal here is to remove any detail that is not important for the embedded SW designer, this in order to achieve acceptable simulation speeds to allow for SW development and debugging. The same executable specification can be refined to a more accurate model that is used by verification engineers.

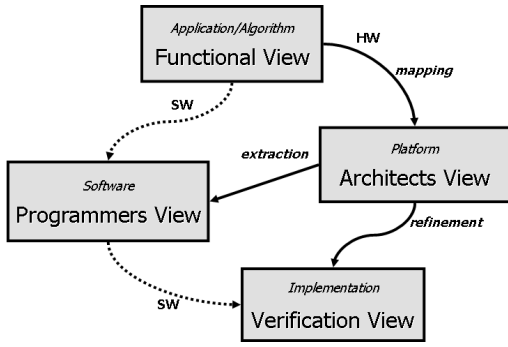


Figure 1. TLM based ESL design flow

The goals of our modeling methodology are high simulation speed, modeling efficiency, and reusability for the different ESL design tasks. Given that the different tasks have different accuracy requirements, the modeling style needs to support multiple levels of accuracy.

As shown in figure 2, the key idea to enable reusability for multiple abstraction levels is a mutual separation of communication, behavior, and timing. In other words, we decompose any model of a platform peripheral into several orthogonal aspects. This separation can be supported using the concept of a *Bus Library*, which contains the bus model and a set bus transactors.

The peripheral initiator and target models are specific for a particular platform and therefore have to be created by the user. These models can be reused for different bus models by separating the user-defined behavior from the protocol specific interface of the respective bus model. This separation happens in two layers: The Storage and Synchronization Layer exhibits a generic TLM interface on the behavior and the communication side. The bus transactor layer converts the generic communication interface into the bus-specific TLM interface.

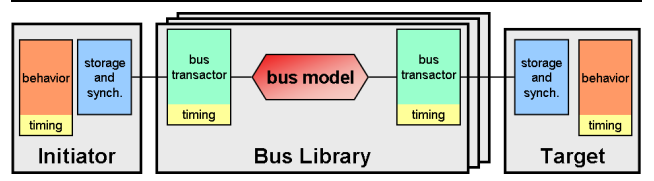


Figure 2. Orthogonalization of Concerns

The level of timing information depends on the use case. The Programmers View requires little or no timing information, whereas an Architects View model should be timing approximate. Therefore also the timing needs to be separated as much as possible from the actual behavior to achieve reusability for different use cases. In our methodology the timing can be modeled as part of the user-defined behavior or simply annotated in the bus transactor [3].

2.2. The TLM Standard

The foundation of TLM is defined by the OSCI TLM working group as communication through function calls [7]. The goal is to abstract certain details of the implementation and work these out at a later stage in the design flow. TLM provides a way of minimizing the number of events and amount of information that have to be processed during simulation. Instead of driving the individual signals of a bus protocol, the goal is to exchange only what is really necessary: the transaction attributes and the data payload. Since TLM also reduces the amount of detail the designer must handle, it makes modeling easier. Common HDL languages enable using certain behavioral abstraction styles. While this abstracts certain details of the computation below the clock cycle level, it does not abstract the communication interfaces. TLM is using function calls to encapsulate the details of the communication interface. This makes it easier to experiment and play with different communication architectures without having to recode all the models. This leads to a much more efficient coding style and faster simulation speed since unnecessary simulation events are suppressed. Using the SystemC interface methods mechanism, complex transactions can be modeled with just a few function calls.

2.3. TLM Interconnect Models

So far the cycle accurate modeling of interconnect models has been a major focus of TLM. Here each type of communication architecture and bus protocol is represented as a set of function calls and events. At the higher abstraction levels, the communication interfaces

can be unified to a single API for the Programmers View use-case as well as a single API for the Architects View use-case. This kind of generality is not possible for the VV use case, because a cycle-accurate bus model always exhibits a protocol-specific API.

In general we observe, that the accuracy and the simulation speed of the bus model has a dominate impact on the complete platform simulation and therefore determines the TLM use model:

- A PV bus model used for software development can be very abstract, but it needs to simulate very fast. One PV bus model can represent all kinds of buses.
- An AV bus model used for architecture exploration needs to have some timing information to estimate the performance of the system. A configurable AV bus can represent a family of similar buses.
- A VV bus model used for HDL co-simulation needs to be fully cycle accurate. Therefore a VV bus model is specific for a particular bus architecture.

Transaction-level models interconnect models are predominantly developed by IP and ESL tool providers, because the creation of fast and flexible TLM bus models requires advanced modeling know-how.

2.4. TLM Peripheral Models

As depicted in figure 3, we propose a generic modeling pattern for TLM peripherals. According to this pattern any TLM model should be split into three orthogonal parts: communication, behavior, and timing. As outlined before, the communication part can be further separated into a bus-specific transactor as well as a generic register interface layer. The major motivation to keep these parts independent is to improve flexibility and reuse.

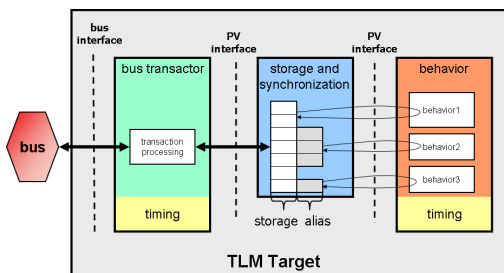


Figure 3. Generic TLM Peripheral Pattern

The reusable register interface objects naturally represent platform storage elements like memories, regis-

ter files, and bit-fields. Common functionality like automatic address or bit decoding can be factored into these objects to improve the modeling efficiency. Additionally, the register objects provide synchronization primitives to associate behavior with the storage access. This behavior gets triggered when the storage object is read or written. Enhanced implementations of the register objects can also improve the analysis and debugging support.

References

- [1] T. Grötter, S. Liao, G. Martin, S. Swan. *System Design with SystemC*. Kluwer Academic Publishers, 2002.
- [2] K. Keutzer, S. Malik, A.R. Newton, J.M. Rabaey, A. Sangiovanni-Vincentelli. System-level design: Orthogonalization of concerns and platform-based design. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, 19(12):1523–1543, December 2000.
- [3] Tim Kogel, Anssi Haverinen, James Aldis. OCP TLM for Architectural Modeling, July 2005. OCPIP whitepaper.
- [4] Adam Donlin. Transaction Level Modeling: Flows and Use Models. In *CODES+ISSS*, September 2004.
- [5] Frank Ghenassia. *Transaction-Level Modeling with SystemC: TLM Concepts and Applications for Embedded Systems*. Springer Verlag, 2005.
- [6] Bart Vanthournout, Serge Goossens, Tim Kogel. Developing Transaction-level Models in SystemC, June 2005. CoWare whitepaper.
- [7] Adam Rose, Stuart Swan, John Pierce, Jean-Michel Fernandez. Transaction Level Modeling in SystemC.