

TUFFAN: A TLM Framework for Fast Architecture Exploration of Digital Systems

Sheis Abolmaali, Parisa Razaghi and Zainalabedin Navabi

ECE Department, University of Tehran, IRAN

{sheis, parisa, navabi}@cad.ece.ut.ac.ir;

Abstract

For design of today's large digital systems, a modeling approach for early exploration of different possible architectures is useful. The modeling approach should be fast, simple and powerful to enable examining several possible architectures. Also, it should contain facilities that support user-defined and future standard communication architectures. In this paper, we introduce a Transaction Level framework that contains predefined components for fast and easy specification of a system being designed. Our approach is based on bus-based communication architectures. Some models of standard buses have been included to help user examining different architectures. Some approximate-timed components are also added to improve the behavior of system models and for better modeling of time-dependent functionalities of communications.

1. Introduction

Today electronic digital systems become large and complex and usually they are implemented in a single chip. Internal components of these SoCs are usually connected by using existing communication IPs. Since selecting a proper communication structure among several existing standards in a SoC design process is a complicated task which needs many time-consuming simulations and computations, a primary executable high-level Model of these components allows the designer to rapidly explore different communication architectures of the system in the early stages of design.

In SoC design, after selecting the system structure at a high level of abstraction, a designer must refine the complete system into a lower level of abstraction till reaching a proper RTL model. In order to be able to cope with rapid exploration of different communication structures, the high-level models of interconnection parts should be designed in a well-

defined format to facilitate the process of synthesizing of these parts into their RTL models.

Transaction Level Modeling (TLM) is a new technique to model different abstraction levels [1]. Designers can model their systems much faster due to less detail in higher abstractions. Therefore, fast design space exploration and evaluating the suitability of different configurations can be performed much easier and faster with this modeling scheme. OSCI has initiated a TLM library[2] for developing an executable high level model of digital systems. It contains several powerful classes that allow a designer to model wide range of configurable high level components. However, this generality causes problems and ambiguities in the system level design methodology and in the process of high level synthesis.

In this paper, we introduce a method for modeling the communication parts of embedded systems at different levels of abstraction to achieve fast and accurate models. We will introduce a framework which has been developed based on OSCI SystemC TLM library. An important feature of our framework is that it contains several basic classes for modeling ports, modules, and simple buses to enable fast and accurate modeling of communication parts. In addition, some models related to common standard buses used in SoC design have been implemented based on our framework. By this way, architecture exploration will be performed with minimum code changing and in shorter time. Furthermore, since the framework has been developed in an object-oriented language, its classes can be extended to new standards.

In Section 2, first we explain how user can explore several architectures with the aid of our TLM framework. Section 3 shows proposed modeling architecture of our framework. Section 4 illustrates modeling of a sample system using our framework. Network Processor (NP) implementation by our framework is presented in Section 6 as a case study. Comparison of our framework with other existing frameworks is described in Section 7. Finally, section 8 concludes the paper by mentioning our future works.

2. Architecture exploration using a TLM framework

In order to restrict the generality problem of existing libraries, we have developed a framework which includes a set of predefined classes for describing ports and different kinds of modules (master, slave or master-slave). Most existing systems use a standard bus as the base of their communication structure; so, a system designer needs high level models of several general purpose buses. Since these buses have complicated transfer protocols and arbitration mechanisms even in high-level of abstraction, their implementation is a difficult task for the designer. Therefore, we have prepared a set of bus models to assist system designers to develop their system accurately and more reliably.

We used SystemC and OSCI TLM library to model flexible communication parts. Our models ignore many low level details related to the actual physical transaction and makes system simulation and testing very fast. We have also developed an approximate-timed model that contains proper constructs for first-level performance analysis.

For utilizing our framework, a designer partitions his/her system to a number of master and slave modules then, if needed, some master-slave modules. The designer just uses our classes for describing master and slave modules, and selects one of our bus models for the system communication structure. Since these components are provided by the framework, system modeling is very fast and easy, and architecture exploration is provided to the designer with minimum changes in his/her codes. The defined classes are powerful enough to enable the designer to model each system component accurately.

3. Modeling architecture

We have proposed a modeling architecture which uses our framework classes. Figure 1 depicts the structure of our proposed model. This architecture contains three layers: *application layer*, *bus communication layer* and *common infrastructure layer*.

Common infrastructure layer: Common infrastructure layer includes several necessary virtual base classes. There are five basic virtual classes provided by this layer: initiator ports, basic address decoding and non-address decoding slave module, basic shared bus module, and basic multiplexed bus module classes. These classes provide basic structure for modeling corresponding physical components in a real system. We have selected OSCI TLM *Transport*

interface as our communication mechanism to exchange data.

The initiator port virtual class corresponds to the master-side controller of buses. It has some virtual high level methods, like *read* and *write*, which can be used for data transferring between modules and hides the data coding of bus transactions from users.

Virtual class for basic address decoding slave module constructs the required infrastructure of all slave-side controllers of shared buses. Like initiator port class, it contains the same virtual high level communication methods. Basic non-address decoding slave module virtual class is very similar to corresponding address decoding virtual class, but it is used in modeling slave-side controllers of multiplexed buses.

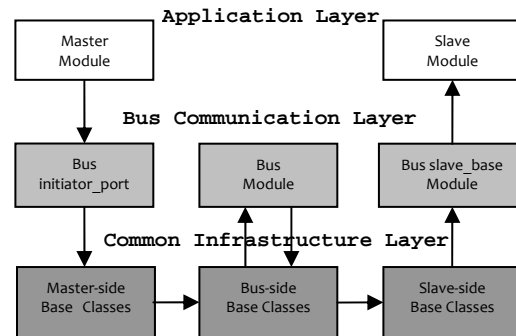


Figure 1. Modeling architecture

Basic multiplexed bus module virtual class has necessary constructs for modeling core multiplexed bus modules. The number of masters and slaves that are connected to this bus are configured. Also, some arbitration mechanisms have been considered in this class. Some methods have been included to model priority arbitration algorithms in approximately-timed models. Other arbitration functionalities like accepting new requests and granting the most prior request are modeled in this class. Basically, real shared bus cores have only arbitration functionality and the communication lines are shared among all system devices. But since we are modeling in high level of abstraction and are using *sc_export* mechanism for communication, there are no shared lines among devices. So we have developed a virtual class for shared bus cores to emulate their behavior.

Bus communication layer: In bus communication layer, mentioned classes specialized for a specific bus. Indeed, the functionalities of bus core and its related controllers in masters and slaves are implemented for a specific bus. High level virtual methods of initiator port virtual class are implemented here for a given bus. The task of initiator port is conversion of high level

function calls to the lower level requests that are compatible with bus protocol. By this, low level details of bus transactions become hidden from user. In slave modules, the reverse actions are implemented. Of course, the implementation of these functionalities that are related to a given bus protocol are provided through framework classes.

Application layer: In the application layer, user utilizes underlying bus services through high level communication methods. This way, all transaction details become hidden from user. The user only calls high-level methods, like read and write, to perform a transaction on a desired bus.

4. System modeling using TUFFAN framework

In this section, we will illustrate how a designer can develop a system by using our framework components. Figure 2 shows an example which utilizes framework classes to model a simple communication between two AMBA/AHB master and slave modules.

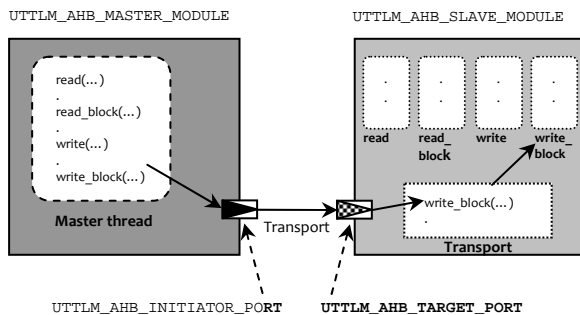


Figure 2. An AHB point-to-point communication using TUFFAN components

This example shows an un-timed model at transport level. So, it can be used to validate the functionality of system after interconnecting its module at first level. Since only master modules have thread processes, models developed with this framework are very fast. Figure 2 illustrates required framework classes for modeling a communication which are detected by capital letters, and also the procedure of converting high level transfer function calls to lower level *Transport* communication.

Figure 3 displays simple AHB system. It illustrates how desired system can be constructed only by connecting defined master and slave modules together through defining a `UTTLM_AHB_CORE_MODULE` object and binding initiator ports to their related target ports. Architecture exploration is done only by replacing one bus macros with other bus macros.

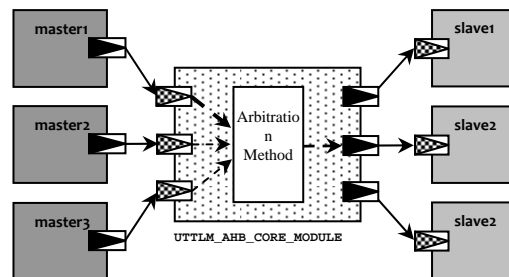


Figure 3. Simple AHB system modeled by TUFFAN classes

Another extremely important means which we have considered in our framework is computation and communication delay modeling. By adding delays, the behavior of the model is more similar to the behavior of the actual system and its concurrency is closer to the real system. It also helps the designer to evaluate the performance of the system.

5. Case study

In this section, we present the implementation of a NP in transaction level using TUFFAN framework. We have designed our network processor from the router specification in [9]. Our design consists of two main parts: communication and forwarding units. Figure 4 shows the diagram of communication and forwarding units.

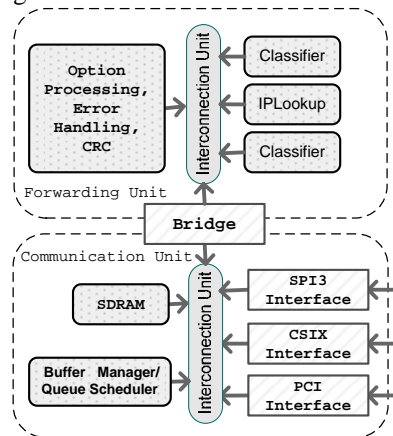


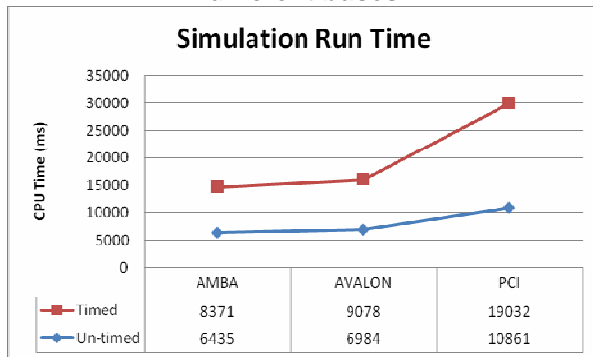
Figure 4. Block diagram of designed NP

Since the framework models the hardware concepts like concurrency, and event handling, we focused on the implementation of functionality of system modules. As shown in Figure 4, we have two interconnect units in our network processor. For implementing these parts, we have used bus models which are implemented based on TUFFAN framework. We can easily configure the selected bus with the number of master and slave ports, the width of address and data

lines and the address range of each slave port. The master and slave modules and their ports are implemented by our framework classes.

For verifying our design we have developed a test-bench and simulated our TLM design in C/C++ environment. By designing this NP in less detail than the timed RTL model, we have gained a noticeable simulation performance. We have implemented this network processor using AMBA, AVALON and PCI buses from our library of components. Table 1 compares the simulation run time of NP for different bus models at two levels of un-timed and timed.

Table 1. Simulation run-time for NP with different buses



6. Comparison with other existing frameworks

In recent years a lot of attention has been given to modeling and synthesis at higher abstraction levels.

There has been valuable work in defining different abstraction levels and modeling issues, but most of them are application dependent and are case study about usage of TLM, like multimedia applications, embedded systems and multi-processor SoCs [5], [6], and [7]. Some projects focus on the lower levels of abstraction, mostly cycle-accurate, to keep the precision of design [5], [8] and [4], which makes the simulation time longer. Also, they generally use TLM FIFOs and channels, which have more events and simulation delays.

The main contribution of our work is developing a powerful and general framework for fast architecture exploration. Since we have used the object oriented nature of SystemC, our framework is strong enough to model many common communication protocols. Different address and data widths, burst transactions, byte-enable capability, different kinds of transaction terminations, several bus supported commands (like PCI commands), different address spaces, parity checking and error reporting are provided via our

framework modeling components. In addition, several arbitration algorithms are included in bus core modules.

Our approach supports user-defined architectures. Model developers can simply add different performance and power analytical computations, according to a specific application, to the framework. High level synthesis of system communication parts is another goal that is considered in our work.

7. Conclusions and future works

In this paper we have presented TUFFAN framework for fast and accurate exploration goal. The next improvement will be to extend our framework to model power consumption for making better decisions during architecture exploration. Furthermore, we will consider more levels of abstraction at transaction level modeling below the transport layer to satisfy TLM synthesis purpose.

8. References

- [1] L. Cai, and D. Gajski, "Transaction Level Modeling: An Overview," Proc. of CODES+ISSS, 2003, pp. 19-24.
- [2] Open SystemC Initiative website: <http://www.systemc.org>
- [3] S. Pasrichat, N. Dutt, and M. Ben-Romdhanet, "Extending the Transaction Level Modeling Approach for Fast Communication Architecture Exploration," Proc. of DAC Conference, 2004, pp. 113-118.
- [4] S. Pasrichat, N. Dutt, and M. Ben-Romdhanet, "Using TLM for Exploring Bus-based SoC Communication Architectures," Proc. of ASAP Conference, 2005, pp. 79-85.
- [5] T.S. Rajesh Kumar, C.P. Ravikumar and R. Govindarajan, "MAX: A Multi Objective Memory Architecture eXploration Framework for Embedded Systems-on-Chip," Proc. of VLSI Design Conference, 2007, pp. 527-533.
- [6] H. Shen and F. Petrot, "MPSoC Communication Architecture Exploration Using an Abstraction Refinement Method," Proc. of VLSI Design Conference, 2008, pp. 403-408.
- [7] P. Martinelli, A. Wellig and J. Zory, "Transaction-level prototyping of a UMTS outer-modem for System-on-chip validation and architecture exploration," Proc. of RSP Workshop, 2004, pp. 193-200.
- [8] Wolfgang Klingauf, "Systematic Transaction Level Modeling of Embedded Systems with SystemC," Proc. of DATE Conference, 2005, vol. 1, pp. 566-567.
- [9] Requirements for IP Version 4 Routers, RFC 1812, 1995.