Abstract

In recent time we have witnessed a rapid growth in the design of complex Systems-on-Chip devices (SoC). As the complexity of such systems is increasing, new methodologies are required to close the productivity gap for SoC design. Raising the abstraction level for SoC design using Transaction level modeling (TLM) has become an obvious solution. TLM allows HW/SW co-simulation and early verification of virtual system models of SoCs. For SoC design at transaction level, models from various sources may need to be integrated which may raise the interoperability issues like like deadlock, loss of transactions, byte-ordering issues etc. TLM standard defines rules to tackle this. However, manually debugging violation of such errors at application level is quite difficult. Therefore, compliance checkers are available which monitors violation of TLM rules during execution. However, online compliance checking is memory intensive and generates lot of overhead during verification of TLM rules. There is another issue with transactional models which use a generic base protocol for communication between TLM components. However, it only solves part of the problem as when virtual models requires greater amounts of accuracy then it is necessary to extend models based on generic protocol. TLM extension mechanism allows to introduce detailed protocol in place of the base protocol. However, compliance checking of such models would bring separate checkers for base protocol and the protocol introduced over a TLM framework.

On another front, Unified Modeling Language (UML) with recent developments
has shown lot of potential towards the design of real-time and embedded systems. With UML 2 and later versions, UML can be tailored for modeling of a specific domain like SoC. Now, SoC designers are looking towards UML for improving the specification, design, implementation and verification processes. In recent time, several UML profiles have been proposed towards real-time and embedded systems like UML profile for SoC, System Modeling Language (SysML) etc. However, this profiles only helps in modeling and to a certain extent code generation aspect. Moreover, UML offers several diagrams like sequence diagram, state machines which have already found applications in requirement specifications, test benches and architectural/behavioral modeling.

This thesis is targeted towards showing the potential of UML for verifying transactional models beyond modeling and code generation abilities. We have proposed a UML based framework for TLM that allows modeling, code generation and verification of TLM rules and other protocols rules over a TLM framework. We have used the UML profile mechanism to develop a UML based TLM profile. The profile enables modeling of transactional models (especially TLM 2) according to different coding styles. We used Object Constraint Language (OCL) expressions to represent TLM rules and included them in TLM profile. With the help of model validation support, violations of TLM rules can be identified along with the exact source of error. We used UML sequence diagram for verification of dynamic rules and proposed an offline strategy for rule checking. We extended the sequence diagram based strategy to allow protocol specific checking and suggested use of a unified protocol checking over a TLM framework.

The key contributions of the thesis are as follows.

- We have proposed a SysML based TLM profile which allows modeling of transactional models at UML level. Moreover, the profile along with code generator enables SystemC code generation from a UML based transactional model.
• We have extended TLM profile to include TLM static rules expressed as OCL constraints. This enables checking of violations of these rules during model development with the help of proper tool support.

• We have proposed a TLM dynamic rule checking strategy based on UML sequence diagram. The proposed strategy verifies TLM dynamic rules against a SystemC based transactional model and reports any violations and the source of violation, if found in the model.

• We extended our strategy from TLM base protocol checking to a detailed protocol checking over a TLM framework. We suggested use of unified protocol checkers in place of using a protocol checker for each protocol used in a mixed transactional model.

• We also suggested automation of rule checker based on the protocol specification. The rules are specified using first-order logic (FOL) and converted to Java expression and can be validated using UML protocol state machine and sequence diagrams.