# A UML BASED FRAMEWORK FOR TRANSACTION LEVEL MODELING

VAIBHAV JAIN


DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

INDIAN INSTITUTE OF TECHNOLOGY DELHI

APRIL 2015

# A UML BASED FRAMEWORK FOR TRANSACTION LEVEL MODELING

by

**VAIBHAV JAIN**

Department of Computer Science and Engineering

Submitted

in fulfillment of the requirements of the degree of

**Doctor of Philosophy**

**to the**


**Indian Institute of Technology Delhi**
**April 2015**

# Certificate

This is to certify that the thesis titled **A UML Based Framework for Transaction level Modeling** being submitted by **Mr. Vaibhav Jain** for the award of **Doctor of Philosophy** in Computer Science and Engineering is a record of bona-fide work carried out by him under our guidance and supervision at the Computer Science and Engineering Department, Indian Institute of Technology Delhi. The work presented in this thesis has not been submitted elsewhere, either in part or full, for the award of any other degree or diploma.

**Anshul Kumar**
Professor
Dept. of Computer Science & Engineering
Indian Institute of Technology Delhi
New Delhi, India 110 016

**Preeti R Panda**
Professor
Dept. of Computer Science & Engineering
Indian Institute of Technology Delhi
New Delhi, India 110 016

# Acknowledgments

First of all, I wish to express my profound gratitude to my supervisors Prof. Anshul Kumar and Prof. Preeti R Panda for their continued encouragement and invaluable suggestions during this work. It has been an honor to work with them.

I would like to thank my colleague Aryabratt Sahu, Anant Vishnoi, Neeraj Goel, G Krishnaiah, BVN Silpa, Sharat Verma, Arun Parakh, Lava Bhargava and all the other colleagues for their cooperation and support. I wish to thank Prof. M Balakrishnan, Prof. G. S. Visweswaran and Dr. Kolin Paul for their valuable feedback and encouragement.

I would also like to thank John Aynsley from Doulos and Puneet Arora from Circuitsutra for their valuable feedback.

I would also like to thank the staff members of Computer Science deaprtment especially Mrs. Vandana Ahluwalia and Mr. S. D. Sharma for their help. They have opened their hearts and their doors, supplying me with resources I never could have found without them.

Finally, It could never been without my family support. I have faced real tough moments during my PhD tenure as I lost my loving father in a tragic accident and also lost my grandmother. With great love, patience and encouragement shown by my family including my wife Priyanka and mother.

**Vaibhav Jain**

# 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 SystemC based Transaction Level Modeling (TLM) [1] has become an widely used solution. TLM allows HW/SW co-simulation and early verification using virtual system models of SoCs. For SoC designs at transaction level, models from various sources may need to be integrated which may raise the interoperability problems, leading to deadlock, loss of transactions, byte-ordering issues etc. TLM standard defines rules to ensure interoperability. However, TLM library does not provide any support for verifying these rules. On the other hand, manually debugging of interoperability errors raised by violation of TLM rules at application level is quite difficult. Therefore, automatic compliance checking is desirable. Compliance checkers [2] are used to monitor any violation of TLM rules during execution and find the exact source of error. Since these compliance checkers perform online checking, these are quite memory intensive and result in a lot of overhead during verification of TLM rules. Transactional models use a generic base protocol for communication between TLM components. When these virtual models require greater amount of accuracy then it is necessary to extend these models. TLM extension mechanism allows introduction of a more detailed protocol within the TLM framework. Compliance checking of such

models requires additional checkers for verification of the new protocol introduced.

On another front, Unified Modeling Language (UML) [3] with recent developments has shown a lot of potential towards the design of real-time and embedded systems. UML offers several diagrams like sequence diagram, state machines which have already found applications in requirement specifications, test benches and architectural/behavioral modeling. 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, these profiles only help in modeling and to a certain extent code generation aspect.

This thesis is targeted towards utilizing UML model validation capabilities for verifying transactional models and showing UML potential 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 rules of other protocols built over TLM. The proposed framework includes a TLM profile which allows UML based Transaction level modeling and provides support for different coding styles. The profile also helps in verifying TLM static rules during model development. The verification of TLM rules during model development helps in detecting the modeling errors early. However, verification of TLM dynamic rules requires execution of the model. We have developed an offline strategy based on UML sequence diagrams derived from execution traces and have shown that it is more efficient than the usual approach of online verification during model execution. Next, we extended our framework by allowing compliance checking of user defined protocols. This is based on a more general dynamic rule checking strategy and support for checker generation from protocol rules.

# Contents

# List of Tables

# List of Figures