Hierarchical Formal Verification Method based on Transactions

Zhiqiu Kong, Jinian Bian, Yanni Zhao, Shujun Deng
Department of Computer Science and Technology, Tsinghua University
Beijing, China
kzq07@mails.tsinghua.edu.cn, bianjn@tsinghua.edu.cn, {zhaoyn05, dengsj04}@mails.tsinghua.edu.cn

ABSTRACT

Complexity and large scale of SoC designs call for developments in today's verification techniques. On the one hand, it costs too much efforts and time in traditional RTL modeling methodologies when verifying large scale designs; on the other hand, properties requiring cycle accuracy are not efficiently verified in Transaction Level Models due to the abstraction. In this paper, we present a hierarchical verification methodology which is dedicated to make tradeoffs between verification efficiency and completeness. Design under Verification (DUV) is firstly abstracted based on transactions; refinements on them will be done later in verification flow only when verifying properties which require finer time accuracy. Furthermore, improvements on model checking techniques are also proposed.

Keywords

Formal Verification, Transactions, TLM, RTL

1. INTRODUCTION

Due to SoCs' (System-on-Chip) large scale and complexity, bus protocols which are essential in systems are becoming more and more complex in order to implement great amount of new functionalities. As a result, not only protocol designs but also verifications on them -- including both simulation and formal ones -- are facing great challenge nowadays.

Traditional verification methodology dedicated in Register Transfer Level (RTL) is obviously not efficient enough: it takes too much time and efforts; most of the time not available until late design flow which not only severely affects time-to-market but high cost will also be spent on correcting any fault founded later.

The concept of transaction is proposed to ease the efforts made for design and verification. In Transaction Level Models (TLM), function calls which package certain signals are used to model communications between different modules in system. This dramatically speeds up design/verification flow and also makes it possible for early hardware/software integration [1].

However, properties under verification may involve signals that will be abstracted away when modeling into TLM. This will almost surely affect completeness of verification. Moll et al. proposed methodology in constructing cycle-accurate TLM as a possible solution to this problem [7], although its simulation speed is much faster than RTL models, it actually sacrificed the efficiency in verification for time accuracy. Cadence [3] also introduced Transaction Verification Model (TVM) as bridge between Transaction-Based Model and RTL Model, but it is used for simulation and leave formal verification techniques un-considered. Thus, we would like to introduce a hierarchical formal verification method which dedicates to make tradeoffs between verification efficiency and completeness. In our method, DUV will be modeled into Transaction Level Model (TLM) in the beginning; elaborations on it will be performed only when signals abstracted away in TLM are required to verify a certain property.

The remainder of this paper is organized as following: Section 2 describes preliminaries for the hierarchical formal verification method; the main verification flow and details will be presented in Section 3. Section 4 concludes the paper and discusses the shortcomings of this method and future works.

2. PRELIMINARY

2.1 Formal Verification Approaches

Model Checking is the most popular techniques for formal verification nowadays. It was first introduced by E. M. Clarke and E. A. Emerson [5] and by J. P. Queille and J. Sifakis[6]. Generally, DUV are modeled into Finite State Machine (FSM), and the property being verified is expressed as temporal logic. State space exploration on FSM will be done in order to find out the property holds or not.

Symbolic Model Verification (SMV) and Bound Model Checking (BMC) are two different approaches for model checking. The former one uses Binary Decision Diagram (BDD) to explore the state space, while the latter one transfer the original problem into a satisifiability problem and check the result with specified bound.

On the one hand, SMV is more efficient than BMC and more widely used, but state explosion will occur because mostly we need to exhaust all reachable states -- which are of great quantity, especially for designs with large scales -- before we reach a conclusion. This severely restricts the application of SMV. On the other hand, although BMC is less efficient, state explosion problem could be prevented since no exploration of states is needed. What's more, development of modern SAT solvers in recent years dramatically speeds up verification process of BMC. Even some SMV solvers now adopt BMC as complementation [4]. Hence, we choose BMC as the verification approach in this paper.

In BMC procedure, a property P is expressed as Linear Temporal Logic (LTL) *L* and DUV is expressed in Kripke (FSM-like) structure *M*[4]. BMC verifies property *P* by unrolling the FSM with time bound *B*, which is very similar with SMV where state exploration is adopted. But most of the time it is unnecessary to unroll Kripke infinitely. Let's consider a property *P* which says no dead-lock will occur. Thus, if we make sure that dead-lock will occur when *M* unrolls for *B* time bounds, we can stop and conclude *M* violates *P*, time bounds more than b is unnecessary to be tried out. Otherwise, we may increase bound *B* and verify P again, or just stop and conclude that *M* satisfies *P*, where completeness of verification can not be guaranteed.

Actually the procedure is more complicated and more details can be gotten by referencing [4]. Basically, in order to verify original property P via BMC, both LTL expression *L* and Kripke structure

M will be bound *B* and then verification process is done by finding a counterexample of *P*, otherwise, bound is increased or *P* is thought to be satisfied.

2.2 Design under Verification

In order to clearly describe the hierarchical method of verifying in both TLM and RTL models, we choose DTL protocol as the example to illustrate our ideas.

Device Transaction Level (DTL) Protocol is a communication protocol developed by Philips which serves for point-to-point and synchronous data transfers. Both blocks of data and single byte are supported [7].

The basic DTL protocol consists of 24 signals, 6 of them are used for extensions such as buffer management and two-dimensional block operations [8]. In order to ease the efforts to illustrate the essential points of our ideas, we make simplification to original base DTL protocol. Only byte transfers are supported in this case and no extensions are adopted. After the simplification, only 12 signals are left. There're mainly read/write operations in bus protocol, that's why almost only signals related to read/write operations are kept, and emphasized in following explanation.

Besides the simplification, we made an extension to the original DTL protocol in respect for illustrating our hierarchical verification method. Simultaneous read/write operations are supported as mutex and synchronization mechanism is introduced. Here are the details below:

Extension 1: Multiple read transactions could be done at the same time iff. there is no write transactions accessing the same address.

Extension 2: Multiple write transactions could be done at the same time iff. there is no read/write transactions accessing the same address.

Two signals are added to support this extension: *is_trans* indicates if there's ongoing transfer; *is_rd* indicates whether the ongoing transfer is read or write; thus, 14 signals are involved after simplification and extension, and then divided into a set of 4 groups according to their functionality, please refer table 1 for details. This new protocol will be referred as DTL-S in further discussion.

Table 1. Signals and Groups of DTL-S

Group	Signals
command	cmd_valid, cmd_accept, cmd_finish
write	wr_valid, wr_addr, wr_data, wr_accept
read	rd_valid, rd_addr, rd_data, rd_accept
mutex	is_transfer, is_rd

The signals' names are self-explanatory. To describe the work flow in brief: command should be sent and set as valid before read/write operations are allowed to be performed. Mutex signals are set when the first read/write operation begins or the last operation ends in order to guarantee mutual exclusion.

2.3 Properties under Verification

Properties which are necessary to be verified can be classified into 2 groups: safety and liveness.

Safety: Mutual exclusions should be guaranteed. Here we need to make sure that no simultaneous read/write operations with the same address are performed.

For example, let's assume there're 2 processors p1 and p2 accessing the same memory via DTL-S Protocol Bus, just as what is shown in Figure 1.

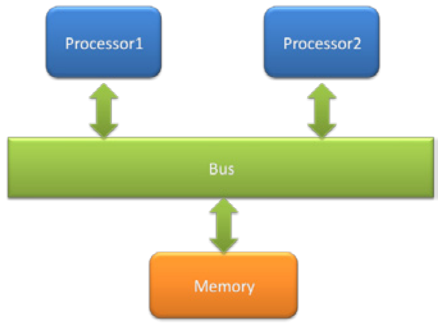

Figure 1. Shared memory access model

The following property should be guaranteed in order to make sure that no simultaneous write/read which accessing the same address will be accepted at the same time:

$$\begin{aligned} &\mathrm{G}\,!\,(\mathrm{p1.wr\_addr} == \mathrm{p2.wr\_addr} \,\&\, \mathrm{p1.wr\_accept} \,\&\, p2.wr\_accept) \\ &\quad |\,(\mathrm{p1.wr\_addr} == \mathrm{p2.rd\_addr} \,\&\, \mathrm{p1.wr\_accept} \,\&\, \mathrm{p2.wr\_accept}) \\ &\quad |\,(\mathrm{p2.wr\_addr} == \mathrm{p1.rd\_addr} \,\&\, \mathrm{p2.wr\_accept} \,\&\, \mathrm{p1.wr\_accept})) \end{aligned} \qquad 1)$$

Liveness: Every request initiated will be eventually acknowledged, i.e. in the above example:

$$\mathrm{G}(\wedge_{i=1,2}(\mathrm{pi.cmd\_valid} \rightarrow \mathrm{pi.wr\_valid} \,|\, \mathrm{pi.rd\_valid})) \qquad 2)$$

The above property should be satisfied. Note that there're some specified properties which involves signals which will be abstracted away when verification work is performed in higher levels, such as 1). If so, we need to refine the abstracted model to check the property. We'll discuss the details later in Section 3.

3. Hierarchical Verification Method

Designs in RTL, especially complex ones, are dealing with signals which are of great quantity. On the one hand, verification in RTL models costs lots of time and efforts. That's why Transaction-Based design/verification is proposed and adopted: abstractions are possible to be made to ease verification efforts and bring verification process earlier. On the other hand, there're some kinds of properties involving specified signals which are abstracted away in models based on transaction. Hence TLM are short of verifying these properties and completeness of verification cannot be achieved. Thus, tradeoffs between verification speed and completeness level need to be done, that's exactly the motivation of us to propose hierarchical verification method based on both TLM and RTL models.

In brief, we'll classify properties under verification into 2 main groups: one suited for being verified under transaction-based model and another suited for RTL model. At the beginning, we'll make abstraction to the original specification and verify properties as many as possible. For safety properties, since details which violates the property might be abstracted away, a violation of P on abstracted model surely indicates that the original model violates P, too. But on the other hand, when failing to get witness of violating P, we need to refine part of abstracted model back into detailed one to validate it again. Figure 2 shows the verification flow and the hierarchical verification structure.

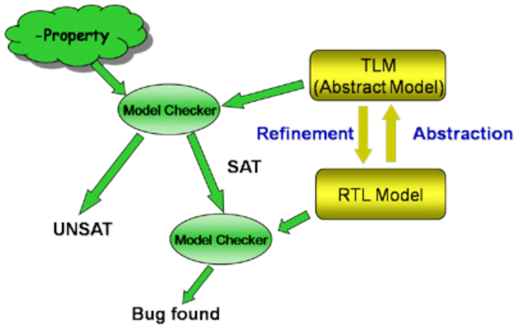

Figure 2. Hierarchical Verification Flow

The whole verification flow will be discussed later in 3 parts: Abstraction & Modeling, Model Checking & Refinements, and Improvements of model checking.

3.1 Abstraction and Modeling

The task of constructing transaction level model according to original designs attracts lots interests these years, and how to choose grain fineness while abstracting remains a problem. Cai & Gajski [10] introduced 6 different transaction level models varying from Un-timed, Approximate-timed to Cycle-Accurate in computation and communication perspective. Designers could pick one or several of these 6 models templates for different usages. Moll et al. proposed a comprehensive methodology dedicated in modeling Bus Cycle Accurate (BCA) model for DTL protocol. This methodology proves to speed up simulation speed compared to signal-level RTL models while also suites scenarios where time accuracy is required. We'll use this modeling flow in further discussion.

Nevertheless, cycle-accurate model is not always efficient enough for general verification purpose since details related to property being verified might be abstracted away. Thus, we choose Un-timed model at the very beginning, and make refinements when necessary. For application and programming purpose, we choose different coding styles defined in TLM 2.0 Standard [2] for different models: LT (loosely-timed) for un-timed model, AT(approximate-timed) for approximately-timed model, for models where finer time accuracy is required, extensions for generic payload and phases are needed [7].

There're several formal models we can adopt to express DTL-S protocol for further verification, such as Petri-nets, Communication Sequential Process (CSP), Finite State Machine (FSM) and etc. FSM is chosen as the formal model in this paper since it is adopted by traditional model checking flow, and efficient model checkers dealing with FSM can be easily found [9].

Here's the abstraction flow for un-timed model basing on transaction:

Step 1: **Signals classification**. Classify signals into several different groups according to their functionality and attributes. Some signals may need to be duplicated in several groups for ease of classification. This is already done in Table 1.

Step 2: **State Machine Creation.** To achieve this goal, we need to firstly identify state transitions in or between different groups. This can be done by identifying timing points [7]. However, it's unnecessary for us to focus on timing point since we are now using un-time model as an example. Figure 3 shows un-timed state machine of DTL-S protocol.

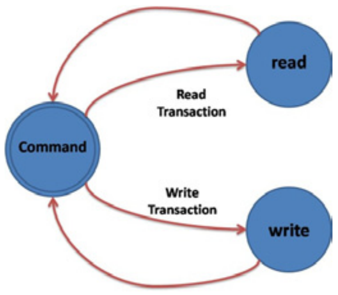

Figure 3. FSM of Untimed model for DTL-S Protocol

Step 3: **Transaction Phase Mapping.** Transaction phases could be created according to timing points identified at step2. Every state in FSM should be mapped into one transaction phase. For un-timed DTL-S protocol model, we have exactly the same transaction phases as states in FSM: COMMAND, WRITE and READ.

Step 4: **Property Conversion.** Properties under verification should be converted to new forms if signals involved in them are abstracted away in modeling process. In our case, there're no signals but transactions, so all properties need to be converted.

Safety:

$$\begin{aligned} G\,!\,(&p1.\text{phase} == \text{WRITE}\ \&\ p2.\text{phase} == \text{WRITE}) \\ |\,(&p1.\text{phase} == \text{WRITE}\ \&\ p2.\text{phase} == \text{READ}) \\ |\,(&p1.\text{phase} == \text{READ}\ \&\ p2.\text{phase} == \text{WRITE})) \end{aligned} \qquad 3)$$

In order to prevent simultaneous read/write with the same address, we just prevent simultaneous read/write transaction, no matter they'll access the same address or not. Note that some information is lost while the original property is converted to this one; this is because the signals under checking are abstracted away while modeling basing on transactions. We have no choice but convert the constraints on signals into constraints of transaction phases which the signals belong. The details of equivalently checking original property will be discussed later.

Liveness:

$$G(\wedge_{i=1,2}(pi.\text{phase} == \text{COMMAND}\ \&\ (pi.\text{phase} == \text{WRITE}\ |pi.\text{phase} == \text{READ}))) \quad 4)$$

This property is almost the same as the original one since all signals involved here are mapped into different transaction phases. Thus constraints of signals can be mapped into constraints of phases smoothly.

3.2 Model checking and Refinement

After transaction based model and properties are gotten, we could move on into model checking procedure. Many efficient model checking tools are available nowadays, such as NuSMV [9]. FSM and properties expressed in temporal logic could be easily converted into smv file which will be processed by NuSMV.

Take property 3) and 4) as examples. For property 4), since no information is lost while abstraction and modeling, the result of 4) is exactly the same as the original property 2). But for property 3), since signals *wr_addr* and *rd_addr* are abstracted away while modeling un-timed FSM, result of 3) is absolutely not always equivalent to the original 1). For example, processor1 writes into address 0x100 while processor 2 reads from address 0x200 is exactly counterexample of 3), but obviously not the one of 1).

Thus, we need to do refinement according to signals involved in 1) in order to get the exact result in further verification. Figure 2 shows FSM of DTL-S protocol after refinements.

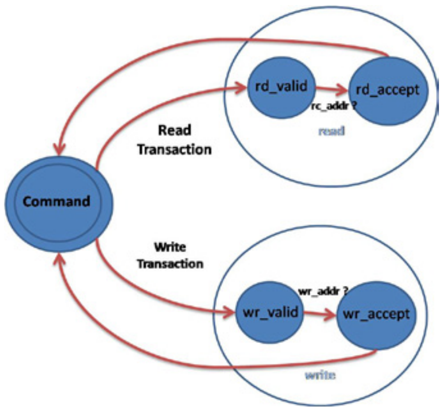

Figure 2. FSM after refinement

Read/write transaction phase are replaced by 2 refined phases respectively and signals *wr_addr*/*rd_addr* can be seen now. They will be used to check if the requested operation is valid or not. Thus property 3) could be converted back to 1) and verified.

Note that not all transaction phases are refined into RTL (command phase in above example remains unchanged), transaction phases will be refined only when needed. As a result of this, state space and problem scale will definitely be smaller than cycle-accurate TLM models, not even to say RTL models. Further experiments data are needed to prove this point.

3.3 Improvements of Model Checking

In traditional Bound Model Checking, Boolean SAT solvers are used to resolve satisfiability problem converted from the original model checking problem. While this might be appropriate in verifying RTL models with a majority of bit operations, verification in higher levels needs more powerful solvers which support bit-vector or integer operations. What's more, when observing solving mechanism of NuSMV, we found that Boolean encoding for FSM is made before BMC process. The efforts and time cost by the encoding process could be saved if we adopt a solver which supports complex operations. Thus, we plan to replace SAT solvers in BMC process with SMT Solver, by which more complex operations are supported and FSM in higher level can be directly converted to resolvable problems without Boolean Encoding process.

4. CONCLUSION & FURTURE WORK

In this paper, we present a hierarchical formal verification method to make tradeoffs between verification efficiency and completeness. DUV is abstracted basing on transactions at first, and then refined when necessary for verifying specified properties. Since refinements of original Transaction Level Models are only necessary for part of the properties, this hierarchical method could reduce the cost of verification in high probability while guaranteeing verification completeness.

However, we need do further study on this to prove efficiency of this method:

1) Efficiency of this hierarchical method depends on quantity of special properties. For example, if all properties under verification require cycle accuracy, the approach introduced here will be absolutely much slower than cycle-accurate TLM modeling methodologies due to the cost of abstraction and refinement. Hence, we need to do more investigation and analysis on protocols in application in order to find out if there're enough scenarios where hierarchical method can be adopted.

2) BMC process with SMT solver is less efficient than NuSMV in preliminary experiments of us. More study on the solving process needs to be done in order to find the bottleneck of solving. More improvements might be introduced, such as heuristic bound searching method.

5. REFERENCES

[1] Stuart Swan, SystemC transaction level models and RTL verification, Proceedings of the 43rd annual conference on Design automation, July 24-28, 2006, San Francisco, CA, USA [doi>10.1145/1146909.1146937]

[2] TLM 2 Whitepaper. May 2007. www.systemc.com

[3] D. Brahme, S. Cox, J. Gallo, M. Glasser, W. Grundmann, C. Ip, W. Paulsen, J. Pierce, J. Rose, D. Shea, and K. Whiting. The transaction-based verification methodology. Technical report, Cadence Design Systems, Inc., August 2000.

[4] A. Biere , A. Cimatti , E. M. Clarke , M. Fujita , Y. Zhu, Symbolic model checking using SAT procedures instead of BDDs, Proceedings of the 36th ACM/IEEE conference on Design automation, p.317-320, June 21-25, 1999, New Orleans, Louisiana, United States [doi>10.1145/309847.309942]

[5] Clarke, E. M.; Emerson, E. A.; Sistla, A. P. (1986), "Automatic verification of finite-state concurrent systems using temporal logic specifications", ACM Transactions on Programming Languages and Systems 8: 244, doi:10.1145/5397.5399

[6] Queille, J. P.; Sifakis, J. (1982), "Specification and verification of concurrent systems in CESAR", International Symposium on Programming, doi:10.1007/3-540-11494-7_22

[7] H.W.M. van Moll, H. Corporaal, V.Reyes, M.Boone. "Fast and Accurate Protocol Specific Bus Modeling using TLM2.0"， DATE 2009

[8] Sudeep Pasricha, Nikil Duttt . "On-chip communication architectures: system on chip interconnect",Book.

[9] Alessandro Cimatti , Edmund M. Clarke , Enrico Giunchiglia , Fausto Giunchiglia , Marco Pistore , Marco Roveri , Roberto Sebastiani , Armando Tacchella, NuSMV 2: An OpenSource Tool for Symbolic Model Checking, Proceedings of the 14th International Conference on Computer Aided Verification, p.359-364, July 27-31, 2002

[10] Lukai Cai, D. Gajski, "Transaction Level Modeling: an overview", CODES+ISSS, Oct 2003