

RTL-TLM Equivalence Checking Based on Simulation*

Nicola Bombieri Franco Fummi Graziano Pravadelli
Department of Computer Science
University Verona, Italy
{firstname.lastname}@univr.it

Abstract

The always increasing complexity of digital systems is overcome in design flows based on Transaction Level Modeling (TLM) by designing and verifying the system at different abstraction levels above RTL. The bottom-up approach is often adopted in the design flow when already existing RTL IPs are abstracted to be reused into the TLM system. In this context, proving the equivalence between a model and its abstracted version is still an open problem. In fact, traditional equivalence definitions and formal equivalence checking methodologies presented in the literature cannot be applied due to the very different internal characteristics of the models. In this paper, we propose a methodology based on simulation which gives two important contributes. Firstly, it relies on a suite of tools to automate as much as possible the equivalence verification process. Then, a more accurate definition of the equivalence concept is proposed by giving two quality measures of stimuli automatically generated for checking the equivalence between the generated TLM and the RTL golden model.

1. Introduction

TLM is nowadays the reference modeling style for HW/SW design and verification of digital systems. TLM greatly speeds up the verification process by providing designers with different abstraction levels whereby digital systems are modeled and verified. Thus, the complexity of the modern systems can be handled by designing and verifying them through successive refinement steps [1].

In a TLM-based design flow, a system is first modeled at high-level in order to check the pure functionality, disregarding details related to the target architecture. Thus, motivated by the lack of implementation details, the simulation speed is a few orders of magnitude faster than at RTL. Then, step by step, designers refine and verify the system

description more accurately, towards the final RTL implementation.

Reuse of previously-developed Intellectual Property (IP) modules is another key strategy that guarantees considerable savings of time in transaction level modeling. In fact, modeling a complex system completely at transaction level could be inconvenient when IP cores are already available on the market, usually modeled at RTL. In this context, modeling and verification methodologies are based on transactors for converting TLM function calls to sequences of RTL signals and vice-versa [2], thus allowing the integration between TLM and RTL components.

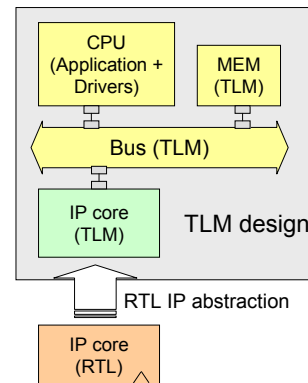


Figure 1. Abstraction of RTL IP core in TLM design flows

Nevertheless, since the integration of RTL IPs into a TLM design involves slowing down the whole system simulation, designers translate the RTL implementation of IPs into corresponding TLM models (see Figure 1). In this context, a first methodology has been recently proposed to automatically abstract RTL IPs towards TLM descriptions [3], and research effort is spent in this direction. Nevertheless, nowadays, designers mainly perform this translation task by hand. In any case, either by hand or by relying on automatic tools, abstracting RTL IPs into TLM models needs a manda-

*This work has been partially supported by the European project VER-TIGO FP6-2005-IST-5-033709.

tory step in which the equivalence between the two models is verified.

Equivalence checking between TLM and RTL implementations is still an open problem, and, to the best of our knowledge, research work in this topic is still in its early stages [4, 5]. The main problem in formally checking the equivalence between RTL and TLM models is due to the fact that, generally, there are neither temporal nor structural similarities between TLM and RTL descriptions. Thus, equivalence checking by using formal techniques cannot be considered applicable yet.

On the contrary, some techniques based on simulation have been proposed in the past to check the correctness of the TLM-RTL refinement flow [6, 7]. In [7], an incremental verification based on assertions is proposed to validate the TLM-to-RTL design refinement. However, the concept of equivalence is based on properties, i.e., two implementations are equivalent if they satisfy the same set of properties. Thus, the effectiveness of such a kind of equivalence checking depends on the quality of the defined properties and on the set of applied test vectors, since the equivalence is guaranteed only for behaviors for which a property has been defined and verified.

In this work we rely on the concept of equivalence proposed in [7], and, since the equivalence cannot be guaranteed in an exhaustive way, we extend that concept by giving a quality measure of equivalence. In particular, abstracting RTL IPs into TLM models involves two main aspects:

- *Communication.* Different TLM communication protocols and corresponding interfaces can be adopted depending on the target abstraction level.
- *Functionality.* The functionality side of the IP is implemented at a higher level of abstraction, thus leaving out all the details specific to the target architecture that slow down the simulation speed (i.e., clock temporization, bit accuracy, etc.).

Thus, in this paper, we propose a methodology that aims at two main goals:

- To automate as much as possible the equivalence verification process.
- To give two quality measures of equivalence between RTL and TLM models, for the communication and functionality sides.

The paper is organized as follows. Section 2 describes the proposed methodology and each step of the verification process. Experimental results obtained by applying the proposed methodology on real cases of study is presented in Section 3. Finally, Section 4 reports the concluding remarks.

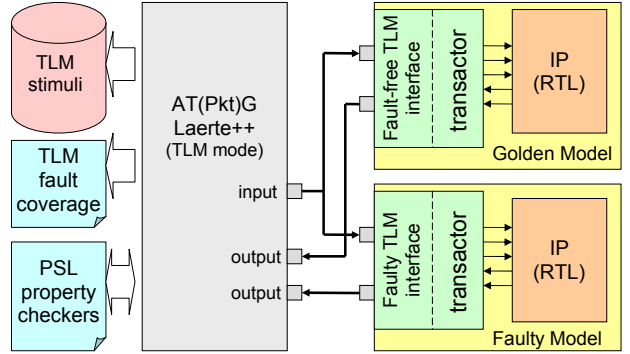


Figure 2. Generation of TLM test packets (stimuli) and PSL properties.

2. Methodology flow

We propose a methodology based on simulation to verify the equivalence between an RTL IP and the corresponding TLM model. Thus, we have the following two main targets:

1. The same set of input stimuli should be used for both implementations and the generated results should be comparable.
2. A quality measure of the input stimuli applied during the equivalence verification should be given.

For the first point, since the two models generally have different APIs and communication protocols, we propose to firstly generate a transactor for the RTL model. A transactor is a translator from a TLM function call to an RTL sequence of statements. It provides a mapping between transaction-level requests, made by TLM components, and detailed signal-level protocols on the interface of RTL IP cores. Thus, a transactor acts as interface to allow cosimulation of TLM-RTL mixed designs.

As a consequence, the same set of input stimuli can be generated at transaction-level and then applied directly to the TLM models, while through the transactor to the RTL model. Figure 2 shows an example in which a TLM automatic test packet generator (ATPktG) generates TLM stimuli. The transactor translates the TLM stimuli to the RTL DUV and, then, translates back the results to the checker. This verification technology (i.e., *transactor-based verification - TBV*) is increasingly used as it allows an easy reuse of TLM testbenches [2, 8] and TLM properties [6] at RTL. Effectiveness of TBV has been evaluated in [9].

Since the transactor generation is a tedious and error-prone task, we propose the TGEN tool [10], which implements the methodology presented in [11] to automatically generate correct-by-construction transactors. In particular,

TGEN allows designers to choose among a library of different TLM APIs compliant with the standard OSCI APIs [12].

Therefore, the same set of input stimuli can be used for both RTL and TLM implementations and the generated results can be matched. However, before matching the two implementations, we propose to generate a set of stimuli which can guarantee a quality level for the equivalence verification, as explained in the next Section.

2.1 The quality measure of equivalence

The equivalence between the RTL golden model and the corresponding abstracted TLM model relies on the following two main aspects:

1. *Correctness of the TLM interface.* The TLM model must be generated by correctly implementing the TLM API and the corresponding communication protocol. In particular, for modularity and IP-reuse reasons, the TLM interface should be compliant with a standard TLM library (e.g., OSCI TLM 2.0 library [13]).
2. *Correctness of the abstracted functionality.* The IP functionality is implemented at a higher level of abstraction, in which all the details specific to the target architecture (that are implemented at RTL) are left out. Thus, the IP high-level behavior implemented at TLM must be *equivalent* to the corresponding low-level RTL implementation.

We propose a two steps methodology for measuring the correctness of both interface and functionality of the abstracted IP with regard to the golden RTL model. In the first step, showed in Figure 2, we apply the AT(Pkt)G in order to generate a good-enough set of stimuli for checking the RTL-TLM equivalence. Since the set of stimuli must cover both the interface and functionality sides, the stimuli generation consists of the following tasks:

1. *TLM Mutation analysis.* The TLM interface of the transactor (that is correct-by-construction) is mutated by injecting TLM faults. We propose the TLM fault injection technology presented in [14]. In particular, the TLM primitives are analyzed and represented by using the extended finite state machine (EFSM) model [15]. Then, such EFSMs are mutated by identifying the relations between the proposed mutations and typical design errors. Thus, the mutated versions of the TLM primitives can be used for measuring, via mutation analysis, the quality of the test suites automatically generated by the AT(Pkt)G. In particular, the coverage of TLM faults corresponds to the quality measure of the generated stimuli for checking the correctness of the TLM interface.

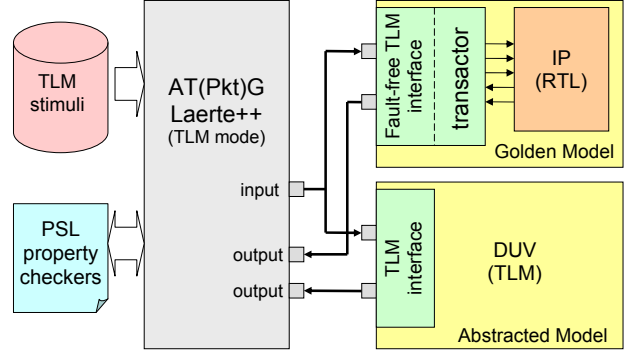


Figure 3. Golden Model RTL vs. Abstracted Model TLM equivalence checking through TLM stimuli and PSL properties.

2. *Assertion-based Verification.* A set of properties is expressed for verifying the high-level behavior of the DUV. Since we rely on the concept of equivalence presented in [6, 7], we claim that the TLM implementation of the IP functionality is equivalent to the golden model implementation if both implementations satisfy the same set of properties. The effectiveness of such a kind of equivalence checking depends on the quality of the defined properties and on the set of the applied stimuli. In particular, the AT(Pkt)G should generate a new set of stimuli which "activate" (that is, drive the DUV simulation in order to switch on the properties) and satisfy all the expressed properties. The number of activated properties and the number of satisfied properties correspond to the quality measure of the generated stimuli for checking the correctness of the abstracted functionality.

Figure 2 shows the main scheme of this step. Firstly, the AT(Pkt)G is applied to two instances of the golden model IP (RTL): through fault-free transactor and through a faulty transactor. The set of generated stimuli which allows to reach a good enough TLM fault coverage (i.e., TLM mutation analysis) is integrated with the set of stimuli generated during the assertion-based verification. The obtained set of stimuli with together to the expressed properties can be used for checking the equivalence between the golden model IP (RTL) and the abstracted TLM model, as showed in Figure 3.

As a result, the simulation process checks the equivalence between the two models giving a measure of goodness of the checked equivalence.

3. Experimental results

The effectiveness of the methodology has been evaluated by checking the equivalence between the RTL golden model

Design	TLM prim. (#)	TLM Mutations		TLM stimuli set 1 (#)	Mutation coverage (%)	PSL properties		TLM stimuli set 2 (#)	Property coverage (%)
		#app	#det			#expr	#verif		
ECC	7	70	55	48	78.6	6	6	53	100
ADPCM	2	35	27	22	77.1	6	5	16	83.3
FFT	12	225	111	86	49.3	12	10	48	83.3
FIR	5	168	152	91	90.5	12	11	56	91.7

Table 1. Experimental results

and the manually generated TLM model of the following designs:

- Error Correction Code (ECC).
- Adaptive Differential Pulse Code Modulation (ADPCM).
- Fast Fourier Transform (FFT).
- FIR Filter.

The first two designs have been provided by STMicroelectronics while FFT and FIR are RTL designs provided with the example set of SystemC 2.2 [13]. As proposed in Section 2, the transactor codes have been automatically generated by using TGEN, the tool built on the top of HIF-Suite [10], which relies on the OSCI TLM library [16] as reference library. We used the TLM feature of an automatic test pattern generator [17] for generating the TLM stimuli, checking the PSL properties, and checking the equivalence between the RTL and TLM models.

Table 1 shows the obtained results. Column *TLM prim. (#)* shows the number of instances of communication primitives used in the TLM designs for implementing the TLM interface. Each primitive has been mutated according to the mutation model proposed in Section 2.1. The number of applied (*app*) and detected (*det*) mutations is reported in Column *TLM Mutations*, while the corresponding number of stimuli generated by Laerte++ for detecting those mutations is reported in Column *TLM stimuli set 1*. Column *Mutation coverage* reports the percentage of covered mutations and corresponds to the first quality measure of the TLM stimuli applied to verify the communication side of the abstracted model.

The number of expressed (*expr*) and verified (*verif*) PSL properties is reported in Column *PSL properties*, while the corresponding number of stimuli generated by Laerte++ for verifying those properties is reported in Column *TLM stimuli set 2*. Column *Property coverage* reports the percentage of verified properties and corresponds to the second quality measure of the TLM stimuli applied to check the equivalence between the behavior implemented at TLM and RTL models.

The obtained results underline that verifying the correctness of the communication side is the main difficulty in

checking the TLM vs. RTL equivalence. Most of the verification time has been spent for checking the correctness of the TLM interfaces, although the quality of the generated stimuli, measured by the mutation coverage, has not reached a completely satisfying value.

4. Concluding remarks

The paper addressed the problem of equivalence checking between an IP RTL (the golden model) and the corresponding abstracted TLM implementation. We proposed a methodology based on simulation which gives two important contributes. Firstly, it relies on a suite of tools to automate as much as possible the equivalence verification process. Then, since any verification methodology based on simulation cannot guarantee two models equivalence in an exhaustive way, a more accurate definition of the equivalence concept is given. Two quality measures are given for measuring the goodness of stimuli automatically generated for checking the equivalence of the communication and functionality sides of the TLM model with regard to the RTL golden model. The proposed methodology has been applied to real cases of study and the experimental results have been reported.

References

- [1] L. Cai and D. Gajski, "Transaction level modeling: An overview," in *ACM/IEEE CODES+ISSS*, 2003, pp. 19–24.
- [2] D. Brahme, S. Cox, J. Gallo, M. Glasser, W. Grundmann, C. N. Ip, W. Paulsen, J. Pierce, J. Rose, D. Shea, and K. Whiting, "The transaction-based verification methodology," Cadence Berkeley Labs, Tech. Rep. CDNL-TR-2000-0825, 2000.
- [3] N. Bombieri, F. Fummi, and G. Pravadelli, "A methodology for abstracting rtl designs into tl descriptions," in *Proc. of ACM/IEEE MEMOCODE*, 2006, pp. 103–112.
- [4] A. Koelbl, Y. Lu, and A. Mathur, "Embedded tutorial: formal equivalence checking between system-

level models and RTL,” in *Proc. of ACM/IEEE IC-CAD*, 2005, pp. 965–971.

- [5] S. Vasudevan, V. Viswanath, J. Abraham, and J. Tu, “Automatic decomposition for sequential equivalence checking of system level and RTL descriptions,” in *Proc. of ACM/IEEE MEMOCODE*, 2006, pp. 71–80.
- [6] N. Bombieri, A. Fedeli, and F. Fummi, “On PSL properties re-use in SoC design flow based on transaction level modeling,” in *IEEE MTV*, 2005, pp. 127–132.
- [7] N. Bombieri, A. Fedeli, F. Fummi, and G. Pravadelli, “Hybrid incremental ABV for functional validation in TLM design flows,” *IEEE Design and Test of Computer*, vol. 24, no. 2, pp. 140–152, March-April 2007.
- [8] C. Norris Ip and S. Swan, “A tutorial introduction on the new SystemC verification standard,” 2003, white paper. www.systemc.org.
- [9] N. Bombieri, F. Fummi, and G. Pravadelli, “On the evaluation of transactor-based verification for reusing TLM assertions and testbenches at RTL,” in *Proc. of ACM/IEEE DATE*, vol. 1, 2006, pp. 1–6.
- [10] ESDGroup - University of Verona, “TGEN: Transactor Generator,” 2009, <http://esd.sci.univr.it> (To appear for downloading).
- [11] N. Bombieri, N. Deganello, and F. Fummi, “Integrating RTL IPs into TLM designs through automatic transactor generation,” in *Proc. of ACM/IEEE DATE*, 2008, pp. 15–20.
- [12] Transaction Level Modeling Working Group, “OSCI TLM 2.0,” 2007, <http://www.systemc.org>.
- [13] “<http://www.systemc.org>.”
- [14] N. Bombieri, F. Fummi, and G. Pravadelli, “A mutation model for the SystemC TLM 2.0 communication interfaces,” in *Proc. of ACM/IEEE DATE*, 2008, pp. 396–401.
- [15] D. Lee and M. Yannakakis, “Online minimization of transition systems,” in *Proc. of ACM Symposium on the Theory of Computing*, 1992, pp. 264–274.
- [16] A. Rose, S. Swan, J. Pierce, and J.-M. Fernandez, “Transaction level modeling in SystemC,” 2004, white paper. www.systemc.org.
- [17] A. Fin and F. Fummi, “LAERTE++: An object oriented high-level TPG for SystemC designs,” in *Proc. of ECSI FDL*, 2003.