Automatic Verification Methodology Based on Structural Test Patterns

Christelle Hobeika, Claude Thibeault, Jean François Boland
Laboratoire de Communication et d'Intégration de la MicroÉlectronique (LACIME)
Electrical Engineering Department, École de Technologie Supérieure,
Montréal, Québec, Canada
christelle.hobeika.1@ens.etsmtl.ca, {claude.thibeault,jean-francois.boland}@etsmtl.ca

Abstract

Functional verification is a major bottleneck in today's design flow. Current technologies are not meeting the challenges imposed by design complexity. In this paper, we propose a new simulation-based verification methodology based on the use of automatically generated structural test patterns in the RTL simulation. The presented approach generally improves the simulation-based verification's quality, keeping the integration, the applicability and the automation aspects in close proximity.

1 Introduction

Over the last few decades, technology scaling has continuously brought new challenges to the research community, from integrated circuit (IC) design to IC testing. From a design perspective, the verification task of today's ICs has become a critical part in the process. Industry estimates that functional verification takes approximately 50-70% of the total effort on a project.

Verification methodologies are grouped into two main categories: 1) Simulation-based methods that rely on vectors to simulate the design and 2) formal methods that use properties to verify the design correctness. Even if both methodologies are now widely established for design verification, simulation-based verification is the most commonly used technique. Verification engineers typically resort to extensive simulation of each design unit, and of the complete system, in order to gain confidence of its correctness. Even with verification budgets dominating design budgets, there are increasingly more bug escapes through fabrication and consequently expensive re-spins [2].

From a test perspective, the challenges mainly come from the design complexity, the evolving design techniques, the emerging defect and fault mechanisms. Over the years, the use of structural approaches based on fault models and on design for testability (DFT) concepts (namely scan-based) [3], has led to the development of efficient automatic test pattern generation (ATPG) tools [4, 5]. The resulting test infrastructure has greatly helped the test community to address previous encountered issues and should continue to facilitate the exploration of new strategies to face the incoming ones.

Design verification and testing are generally regarded as independent activities. At times, manufacturing test sets may be augmented with design verification vectors (also known as functional testing) to catch not modeled faults, although a sound basis for combining the two kinds does not exist. However, verification and test share some common ground that has been exploited in the past [6]. In fact, ATPG based approaches were adapted for formal verification problems such as equivalence checking and property checking. It helped to overcome the well-known state explosion and computational complexity problems of formal methods [7, 8]. Test generation techniques have also been exploited to improve simulation-based verification methodologies, at the gate [9] and register transfer [10] levels.

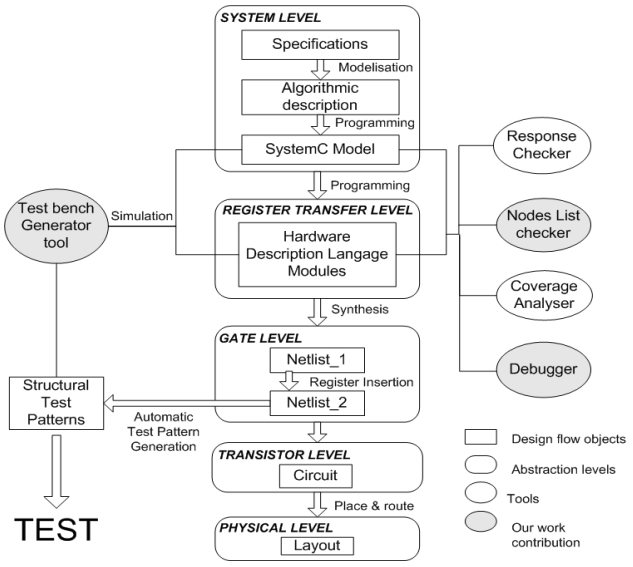

Figure 1: The additional tools of the proposed methodology in a standard implementation flow

In this paper we explore a different test/verification combination, namely the use of structural test patterns in the simulation-based verification process. These patterns are generated by an ATPG tool based on a launch-on-capture transition fault model. The result is a verification

environment that can be seamlessly integrated in the design flow, without requiring circuit modification or remodeling steps. Figure 1 describes a standard design flow and its relationship with verification and test, more specifically the scan-based test and Register Transfer Level (RTL) model verification. It also introduces our work contribution, which appears as 3 different tools: 1) a test bench generator that makes structural test patterns application possible, without any effort, 2) a nodes list checker that keeps track of the nodes coverage and 3) a debugger that tracks the errors, based on a high observability, to identify the signals responsible for the bugs. The proposed methodology reduces time and effort to obtain very high quality simulation-based verification. As well, no knowledge of the system is required to accomplish the verification.

The paper is organized as follows. Section 2 presents the proposed methodology while section 3 describes in details the verification environment. Experimental results are presented in Section 4. Finally, several remarks complete the paper.

2 Proposed Methodology

Here, we are primarily concerned with the productivity of the verification process. The efficacy of this process can be measured along two dimensions: verification time and verification coverage. The proposed methodology aims to reduce time and effort needed to verify the design, by automating the test bench generation. Moreover, it generally enhances the coverage reached by exploring the use of test patterns generated at the gate level to cover the design's faults. The cornerstone of the proposed methodology is the intuition (that became an observation) according to which a node that is difficult to test (Hard Fault) is likely difficult to verify (Dark Corner). In this section we explain the basic concepts of the proposed methodology and describe the major differences with the previous work done in this field.

Contrarily to the simulation-based approaches presented in the literature [9, 10], the proposed methodology is not based on design error modeling, it is rather exploiting the correlation between dark corners and hard faults. In [11], we showed based on experimental results that dark corners are a subset of hard faults and, that the use of structural test patterns in the RT level verification could help improve the nodes' controllability/observability and cover the dark corners.

Another major difference is that we substitute the stuck-at fault model, used mostly in all the works presented in the literature, with the launch-on-capture transition fault model which will simulate and exercise the most efficiently the model's functionality. This model is based on the stuck at model but takes transition propagation into account. In [11] we justified our choice for the transition fault model (launch on capture) as the model to generate the test patterns.

Moreover the proposed approach emulates the presence of scan register chains, which are used by manufacturing test, during RTL simulation-based verification, by associating them to state signals forcing, resulting in controllability and observability improvements. To our knowledge, such a concept has never been used before. In [11] we described the mapping between RT and gate levels despite the weak correlation between them, in order to transform structural test patterns into RTL verification patterns.

Finally based on these concepts, we present in this paper an automated verification environment that accomplish the simulation of the RT model, checks for uncovered nodes and detect the errors and the signals responsible of these errors. The whole approach is automated and the simulation is done at an RT level. The proposed methodology is shown in Fig.1. It is worth mentioning that our methodology can be applied as a late verification step after the actual synthesis, or earlier in the design process. In the latter case, a constraint-less synthesis can be used to reduce the synthesis effort, verifying at the same time if the VHDL code can be synthesized. After synthesis, scan chains registers are inserted into the netlist and finally an ATPG tool is used to generate the set of test patterns that will be applied on the RT (not synthesized) model.

3 Proposed Verification Environment

The proposed verification environment encloses the RT and gate levels. We assume that a SystemC golden model is available and used as a reference model by the verification system at the system level. Note that such a model is also required to apply any verification methodology. In this paper, the SystemC golden model used has the same level of details as the RTL model verified. It is also possible to use a transaction level SystemC model as a golden model. In fact, in [13] the authors describe how to use RTL testbenches for verification of a SystemC model at a higher abstraction level (as transaction level).

3.1 Test bench Generator Tool

One of the key components of the proposed methodology is its automated aspect. In fact, to effectively generate verification data for functional VHDL descriptions based on structural test patterns, we built up an automatic test bench generator that executes the following algorithm:

1. ***Input****: Atpg output file, HDL model of the design to be verified, simulation period.*
2. Based on the ATPG output file, build 2 arrays: scan cells, test patterns (scan chain patterns, primary inputs patterns).

3. Based on the HDL design, identify:
   - Components hierarchy inside different entities.
   - Component's interface in terms of ports, signals variables with their respective names and types.
4. Construct the test bench skeleton.
5. Identify the ports, signals to be forced: hierarchical paths, types.
6. Compute the corresponding conversions of test patterns values.
7. Construct the patterns:
   - Force Primary input (PI), signals, variables.
   - Run simulation_period.
   - Force PI.
   - Run simulation_period.
8. Complete the testbench by applying all the patterns.
9. ***Output**:Testbench.*

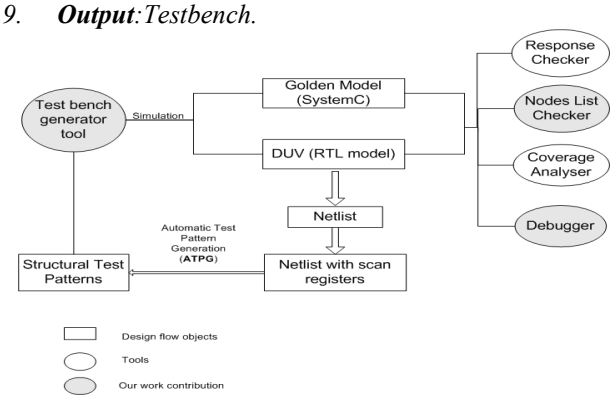

Figure 2: Verification environment

3.2 Nodes list checker

A list checker module is added to the verification environment. A list of transitions and states to be simulated based on the FSM model is created, and for each simulation the module checks which state/transition had been simulated by the patterns and update its list by removing the one simulated. After the simulation is completed, the set of nodes not covered will include the set of states and transitions remaining in the list. This list can be used for further directed simulations, to identify the target nodes for directed patterns in order to accomplish an even more complete coverage.

3.3 Debugger

In addition to the testbench generator tool, an automated component was developed to determine the cause of the errors detected during the simulation. In fact, the simulation is clock-triggered. As transition fault model is used, each pattern takes 2 clock periods to simulate the design: It forces the input and internal signal values at the first period, and at the second period the circuit reacts to its response to the first one. Based on the list of simulation patterns that will be exercising the design, and the list of design's responses to these patterns, the component is able to identify the inputs and internal signals values that caused each error. The observability is not only limited on the primary outputs. In fact to detect the errors the tool observe the state signals too, increasing the design's observability.

4 Experimental Results

We used some ITC'99 benchmarks circuits [12] as HDL models. The characteristics of these circuits are shown in table 1. They were synthesized using a commercial logic synthesizer (Synopsys's Design Vision). After synthesis, full scan registers insertion was done by a commercial DFT tool (Mentor's DFTAdvisor). The gate-level test pattern sequences were generated by a commercial ATPG tool (Mentor's FASTSCAN) for full scan designs targeting Mentor's Modelsim simulator. We believe that the proposed verification environment can be based on any commercial VHDL simulator allowing the forcing of internal signals.

Table 1: Experimental circuits' characteristics

	VHDL lines	PI/PO	FF	HL	Gates
B03	141	11/8	30	1	149
B05	332	1/6	34	1	935
B06	128	2/6	9	1	60
B07	92	1/8	49	1	420
B08	89	9/4	21	1	167
B09	103	1/1	28	1	159
B10	167	11/6	17	1	189
B13	296	10/10	53	1	339
B14	509	32/54	245	1	4,775
B18	1,424	36/23	3,320	3	68,752

In this section we aimed at evaluating the effectiveness of the proposed approach (PA) by comparing it to the 2 well-known and widely used simulation-based verification techniques: pseudo-random approach (PRA) and constrained random approach (CRA) [1]. Table 2 shows a first class of results based on the Finite State Machine (FSM) metrics (states, transitions) provided by Modelsim. It basically reflects the thoroughness of the state machine simulation but its shortcoming is that the relationship between the metric and the detection of design errors `classes is not well understood. Thus, we proceeded to fault simulation and obtained the second class of results shown in table 3 that reflects more the thoroughness of design errors detection. Based on the fault model presented in [14], we injected faults of different types in each design (ex: wrong signal source, Case/if statement, wrong gate/module type, wrong constant, FSM error…) and then simulated them to compute the fault coverage obtained for each approach. As in [14], the fault injection was performed manually, which limited the number of injection faults (20 faults per design were injected in [14]). We used our debugger to see if the error was detected.

Based on the experimental results, we can see that our approach clearly outperforms PRA. Remember that ATPG

tools generate test vectors, based on advanced algorithms, to cover hard faults. Consequently, these patterns help cover most of the hard corners that pseudo-random vectors are unable to reach, leading to a very high coverage.

Table 2: FSM Coverage comparison of PRA, CRA and PA.

	FSM state (%)			FSM transition (%)		
	PRA	CRA	PA	PRA	CRA	PA
B03	100	100	100	100	100	100
B05	85.4	100	100	64.7	88.2	90.3
B06	98.0	100	100	70.0	94.0	100
B07	100	100	100	82.4	95.4	98.2
B08	100	100	100	87.5	92.3	100
B09	75.0	96.0	100	63.6	92.3	100
B10	54.5	78.5	98	37.5	75.6	95.4
B13	63.3	86.5	93.4	45.8	75.4	91.3
B14	100	100	100	100	100	100
B18	56.0	-	98.0	47.0	-	96.8

Table 3: Fault Coverage comparison of PRA, CRA and PA.

	Nb. of injected errors	% of detected errors		
		RA	CRA	PA
B03	20	100	100	100
B05	25	48	72	88
B06	20	75	85	100
B07	20	80	85	95
B08	20	75	90	100
B09	20	55	80	95
B10	20	30	65	90
B13	25	40	68	88
B14	25	92	100	100
B18	30	43	-	87

In addition CRA coverage is equal or lower than the one of our approach. CRA can also automatically generate a large number of test cases within the parameters (constraints) specified by the verification engineer. Hence, it can hit corners and produces a high coverage too. However, with constrained random technique, human interaction is required with extensive efforts to build a verification infrastructure, to understand the code and to specify the constraints. With the proposed approach, no effort is required neither to generate the patterns nor to understand the code.

Note that the b18 circuit is a more complex design composed of thousands line of codes, many hierarchical levels as well as ports and signals of integer types. Reading and understanding the code and its functionality would take several weeks. Due to time constraints and to the fact that there is no available SystemC model for this circuit, we only used our automated approach to apply the structural test patterns, estimate the coverage, and compare it to the pseudo-random based approach This clearly shows that our methodology can be applied on such complex circuits, without any knowledge of the circuit functionality. Finally, it is worth mentioning that it took less than 3 hours to apply our complete methodology on the b18 circuit, which includes the synthesis, scan insertion, pattern generation and simulation steps (running on a Sun Blade 900MHz).

5 Conclusion

We presented a new simulation-based verification methodology based on the automated application of structural ATPG test patterns in the verification process. Our methodology emulates scan-based transition fault patterns by forcing signals during RT level simulations and compares the results to the ones of a cycle accurate golden model. In addition to the fact that it is fully automated, our methodology does not require any knowledge of the design under verification, which should significantly ease and speed up the verification process. Results showed that it could fast provide fault and code coverage that are equal to but most of the times higher than one obtained with other well known simulation-based verification approaches. If necessary, the coverage can be completed with any other verification methodology, as our methodology allows identifying the few remaining uncovered nodes.

6 References

[1] W K. Lam. "Hardware Design Verification, Simulation and Formal Method Based Approaches". Prentice Hall Modern Semiconductor Design Series, 2005 Person Education, Inc.

[2] J.M. Rabaey, S. Malik."Challenges and Solutions for Late- and Post- Silicon Design". IEEE Design & Test of Computers Vol. 25, no 4, July 2008. pp. 296-302.

[3] J. J. Savir, S. Patil. "Scan-Based Transition Test". IEEE Trans. on Computer-Aided Design of ICs and Systems, vol. 12, No. 8, August 1993, pp. 1232-1241.

[4] Mentor Graphics ."Scan and ATPG process guide".

[5] Synopsys .“TetraMAX ATPG Automatic Test Pattern Generation”. 2008 Synopsys, Inc.

[6] P. Varma. “Design Verification : Test to rescue ?” ITC 2003, p. 1292.

[7] S. Y. Huang, K. T. Cheng. , K.C. Chen. "Verifying sequential equivalence using ATPG techniques". ACM Trans. Design Automation Electronic Systems, Vol. 6, no2, April 2001, pp.244-275,

[8] K. T. Cheng, V. Agrawal. "A simulation-based directed-search method for test generation". Proc. Int. Conf. Comp. Design (ICCD), 1987, pp. 48-51

[9] M. S. Abadir, J. Ferguson, T. Krikland. "Logic design verification via test generation". IEEE Trans, on CAD, vol. 7, no 1, Jan. 1988, pp. 138-148.

[10] D. Van Campenhout, H. Al Assaad, J.P. Hayes, R. B. brown. "High-level Design verification of microprocessors via error modeling". ACM Trans. on Design Automation of Electronic Systems, vol. 3 no. 4, Oct. 1998, p.581-599.

[11] C. Hobeika,; C. Thibeault, J.F. Boland,. "Use of structural tests in RTL verification". Microsystems and Nanoelectronics Research Conference, 2008.. pp.133-136.

[12] S. Davidson. "Characteristics of the ITC’99 Benchmark Circuits". www.cerc.utexas.edu/**itc99-benchmarks**

[13] Jindal R., Jain K. "Verification of transaction-level SystemC models using RTL testbenches"**.** Formal Methods and Models for Co-Design, 2003.