An RT-level Fault Model with High Gate Level Correlation

F. Corno, G. Cumani, M. Sonza Reorda, G. Squillero

Politecnico di Torino
Dipartimento di Automatica e Informatica
Torino, Italy
http://www.cad.polito.it/

Abstract

With the advent of new the RT-level design and test flows, new tools are needed to migrate at the RT-level the activities of fault simulation, testability analysis, and test pattern generation. This paper focuses on fault simulation at the RT-level, and aims at exploiting the capabilities of VHDL simulators to compute faulty responses. The simulator was implemented as a prototypical tool, and experimental results show that simulation of a faulty circuit is no more costly than simulation of the original circuit. The reliability of the fault coverage figures computed at the RT-level is increased thanks to an analysis of inherent VHDL redundancies, and by foreseeing classical synthesis optimizations. A set of "rules" is used to compute a fault list that exhibits good correlation with stuck-at faults.

1. Introduction

One of the hardest *theoretical barrier* to the diffusion of test-related tools at the RT-level is the lack of widely accepted fault models. Several variants of high level faults (or testability metrics, as they are sometimes called) have been proposed, and their relationships with stuck-at faults has been shown, either experimentally or theoretically, but such results are generally limited to some specific class of circuits (some approaches target control-dominated circuits [CSSq00b], other are more suited to data-dominated ones [FADe99], or to circuits with few interactions with the environment [FiFu00], and so on). No single fault model is universally accepted, since no comprehensive and general results, valid for all classes of circuits, are known yet.

One of the most used fault model is the *observability enhanced statement coverage* metric proposed in [DGKe96] and [FDKe98]. This fault model requires that all statements in the VHDL description are executed at least once, and that their effects are propagated to at least one primary output. Propagation is modeled implicitly, by determining whether the faulty statement may influence the output values but without hypothesizing any specific faulty value: in some cases, heuristics are needed to resolve non-determinism, and the meaningfulness of the resulting fault coverage is affected by these approximations. While this approach can be fruitfully exploited for test pattern generation [FADe99] [CSSq00b], for fault simulation we need more accurate results.

In this paper we thus adopt a particular instantiation of the observability enhanced statement coverage metric, and in particular we model *single stuck-at bit faults* on all assignment targets of the executed statements that respects an defined set of rules. With this choice, a concrete faulty behavior is simulated, and fault propagation can therefore be performed exactly, by computing the faulty machine evolution. This fault model implies observability enhanced statement coverage, since it models one of the possible fault classes on executed statements. We also define a series of rules to identify redundant faults in the fault list, obtained using the proposed fault model, in order to increase the correlation between the RT-level fault coverage and the Gate-level one. Redundancies identification is based on the reduction of the RT-level fault list taking into account analyzing the optimizations of the synthesis process, in order to eliminate faults corresponding to part of the logic optimized away by the synthesizer.

2. RT-level Fault Model

Fault models taken from software-testing [Beiz90] have three main advantages: they are well known and quite standardized; they require little calculations, apart from the complete simulation of the fault-free system; and they are already embedded in some commercial tools. However, while such metrics may be useful to validate the correctness of a design [CSSq00], they are

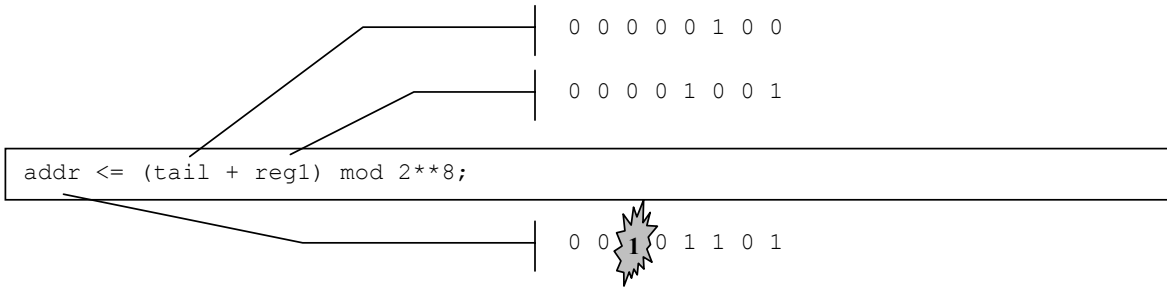

Figure 1: *RT-level Stuck-At Fault Example*

usually inadequate to foresee the gate-level fault coverage with high degree of accuracy.

In order to improve accuracy, some researchers extended software-testing metrics to cope with the peculiarities of hardware descriptions. Fallah et al. [FADe99] [FDKe98] proposed *Observability-Enhanced Statement Coverage*. They define the concept of *tag* as the possibility that an incorrect value is computed at a given location. Different tags are first *injected* in any possible location and then *propagated* during the simulation. The observability-enhanced statement coverage metric computes the number of tags that reach an observable circuit output when the test pattern is applied.

We adopt observability-enhanced statement coverage [CCSS00] and we refine it by using explicit *RT-Level single-bit stuck-at*'s instead of tags. An RT-level single-bit stuck-at fault is defined as a single-bit stuck-at in the effect of an RT-level assignment operation: when a fault is present, the affected object (signal or variable target of an assignment statement) loads the correct value, except for one bit that remains stuck to 0 or 1.

As in [DGKe96], faults are *single* and *permanent*: only one fault is inserted at a time and the fault effect is present during the whole simulation. The RT-Level single-bit stuck-at fault model does not explicitly consider control-flow faults, such as *stuck-at-true* or *stuck-at-false*, as [RiUc96] does.

Figure 1 shows the example of a stuck-at fault. The fault affects the third bit of the assignment operation, and modifies the result of the expression, after it has been computed and before it is assigned to the target signal. The faulty signal is updated as usual, according to VHDL propagation rules, but with a faulty value. Other assignments of the same signal are assumed to be fault-free, since stuck-at faults on the same signal but on different statements are considered different. More details abount the fault model can be found in [CCSS00].

During synthesis the RT level VHDL description is optimized in order to create an efficient gate-level design. The optimization process analyzes the VHDL description and simplifies all logic eliminating redundancies. In this phase some RT level stuck-at faults lose their correspondent Gate level faults. The elimination of these Gate level faults generates a discrepancy between RT and Gate fault coverage figures. In order to prevent this discrepancy is necessary to identify which parts of the logic described at the RT-level disappear during the optimization phase of the synthesis process and to eliminate the associated faults from the fault list.

The logical elements that are eliminated during optimization process are:

- assignment of constant values to a signal or variable;
- signals or variables with only few bits actually used in the system.

Another discrepancy is introduced from the different approach of the RT-level fault simulation and the Gate one. In the Gate level fault simulation the reset signal of the system is considered *fault-free*, so no faults are simulated in this part. To prevent the difference between the two fault simulation methodology, the RT-level faults concerning the VHDL part executed only when the reset signal is active must be eliminated.

Those concepts are formalized by the following set of rules.

2.1. Rule A

When a constant value is written in a variable or a signal, all the logic that was used to accomplish the constant part of the operation is reduced to a set of wires connected directly to the flip-flop. For this reason

the only faults identifiable at gate level are the ones stuck-at with a value different from the constant one.

In order to identify such useless faults it's necessary to:

- identify faults concerning variables or signals destination of an assignment instruction;
- determine if all or some of the bits of the second term of the instruction are constant, or an operation with a constant result, and calculate its value;
- eliminate the faults on the bit whose stuck-at value is equal to the constant value at the same bit.

2.2. Rule B

Sometimes in the VHDL description there are variables or constants that have only a few bit really useful for the system. During the optimization process the size of this variables or signals are reduced to the number of bit really useful. All the faults concerning the erased bits must be eliminated from the fault list.

In order to identify such useless faults is necessary to:

- determine variables or signals that are only used in conditional expressions;
- determine for any conditional expression if the second term is a constant, or an operation between constants, and calculate its value;
- calculate the subset of bit values common to the various constants;
- eliminate the faults on the bits whose stuck-at value is equal to the subset value at the same bit;

2.3. Rule C

At Gate level all the logic connected to the reset signal is considered *fault-free* for this reason all the faults concerning those instructions must be eliminate from the fault list. This hypothesis is needed due to the limitations of VHDL models, that cannot describe the correct behavior of a circuit if it fails to be correctly initialized (unknown valued don't propagate correctly across conditional statements). To provide a meaningful comparison, we do not consider faults in the reset logic, either at the RT- or gate- levels.

In order to identify such useless faults is necessary to:

- determine the part of the VHDL source executable only when reset signal is active;
- eliminate faults concerning variables or signals destination of an assignment in this part;

2.4. Examples

To illustrate some examples of application of the above rules, see Tables 1 to 3. The tables report a sample VHDL statament or fragment, the information about faults to be injected (source line, signal or variable name, bit position of the fault and stuck-at value), as well as the indication whether the application of the rule *eliminated* the fault from the fault list.

In Tab. 1, the assignment of the constant "000" to signal "State" is considered. All stuck-at-0 faults on the bits of State, in injected at this instruction, are untestable since they are not excited. According to Rule A, they are excluded from the fault list, while stuck-at-1 faults will be injected.

In Tab. 2 a different case is considered, where the value of variable "temp" is only used in a greater-than-zero comparison. In such a situation, only the bit sign of the variable is significant, and all faults on lower order bits are deleted by Rule B since they are not observable. As a matter of fact, synthesis tools are able to detect this situation, and do not generate the logic associated to lower order bits: we are effectively predicting that some RT-level faults have no physical meaning since no gate-level equivalent will be synthesized.

Finally, Tab. 3 shows an application of Rule C, where all RT-level faults dominated by the reset signal are deleted.

3. Fault Simulation at the RT-level

In order to verify the feasibility of the proposed fault model, we used a prototype implementation of a Fault Simulator [CCSS00] that, starting from a VHDL description at the RT-level, a Fault List of single-bit stuck-at faults and a Test Pattern, creates a list of detected and undetected faults.

To perform Fault Simulation we use a serial fault simulation strategy, and we simulate the good and each faulty machine, comparing their outputs. To run the simulations, the Test Pattern is first transformed to a set of commands that force the correct waveform for input signals, and the Fault List is transformed to a set of script commands for injecting faults during simulation.

Fault injection is made possible by creating routines that *change* the target signal/variable bit value during simulation, using the simulator scripting language (TCL), when a given target assignment instruction is executed. The fault injection procedures, presented in [CCSS00], must face various issues derived from the fault model, from VHDL Semantics and from the simulator itself.

VHDL instruction	eliminated	source line	name	bit	stuck-at value
State<=000;	**Yes**	**29**	**STATE**	**0**	**0**
	No	29	STATE	0	1
	Yes	**29**	**STATE**	**1**	**0**
	No	29	STATE	1	1
	Yes	**29**	**STATE**	**2**	**0**
	No	29	STATE	2	1

Table 1: *Rule A application*

VHDL source	eliminated	source line	name	bit	stuck-at value

temp= bit_vector(2 downto 0);
……..
temp := DATA_IN+REG
……
if (**temp >= 0**) then
	Yes	**29**	**TEMP**	**0**	**0**

	Yes	**29**	**TEMP**	**0**	**1**
	Yes	**29**	**TEMP**	**1**	**0**
	Yes	**29**	**TEMP**	**1**	**1**
	No	29	TEMP	2	0
	No	29	TEMP	2	1

Table 2: *Rule B application*

VHDL source	eliminated	Source line	name	bit	stuck-at value
reset='1' then **state:=a;** elsif state:=b;	**yes**	**32**	**STATE**	**0**	**0**
	yes	**32**	**STATE**	**0**	**1**
	yes	**32**	**STATE**	**1**	**0**
	yes	**32**	**STATE**	**1**	**1**
	no	34	STATE	0	0
	no	34	STATE	0	1

Table 3: *Rule C application*

This fault simulation environment allows us to compute fault coverage figures at the RT-level with a minimal CPU time overhead, since the VHDL model of faulty circuits is simulated at the same speed as the fault-free model. The only time penalty is at fault injection time instants, where some breakpoints are activated and TCL commands to modify values are executed.

4. Experimental results

To show the feasibility of the proposed fault model we selected a subset of the ITC'99 VHDL benchmarks [CSSq00].

We applied the rules described to the fault lists extracted for the chosen subset of benchmarks, and fault simulated the optimized one with three samples of input sequences:

- a pseudo-random sequence (#1), consisting of 500 vectors with up to 5 circuit reset commands
- a test sequence (#2) developed by a simulation-based gate-level developed ATPG [CPRS96]
- a test sequence (#3) generated by a state of the art commercial topological ATPG working at the gate-level.

Table 4 reports detailed results comparing the RT-level fault coverage figures obtained with the fault model we propose, with gate-level stuck-at fault coverage. The table shows that the application of the rules improves the predictive value of RT-level fault coverage figures.

A synthetic information is given in Table 5 and represented graphically in Figure 2. The correlation coefficient between RT- and gate-level fault coverage figures is incredibly low if no rules are applied, thus exposing the difficulties of modeling at the RT-level faulty behaviors. However, as we apply the rules, we are able to reach a correlation coefficient around 77%. Rules A and C are more general, and give good results on all benchmarks. Rule B, on the other hand, contributes significantly only for benchmarks that have poorly observable assignment statements, such as b04.

The experimental results show that the application of the redundancy identification rules allow the RT level fault coverage to become more and more correlated to the Gate level one. Circuits have an increment of fault coverage and an improvement of the global correlation value.

An exception is the circuit b07, in this case the presence of a ROM decreases the global correlation.

circuit	sequence	RT-level Fault Coverage				Stuck-at Fault Coverage
		no rules	rule A	rules A & B	rules A & B & C	
B01	#1	54.23%	95.06%	95.06%	97.37%	98.06%
	#2	50.00%	87.65%	87.65%	90.79%	98.45%
	#3	52.11%	91.36%	91.36%	93.42%	96.51%
B02	#1	46.43%	90.70%	90.70%	94.87%	99.33%
	#2	42.86%	83.72%	83.72%	84.62%	99.33%
	#3	42.86%	83.72%	83.72%	84.62%	99.33%
B03	#1	52.94%	67.61%	67.61%	70.49%	73.84%
	#2	50.37%	64.32%	64.32%	66.67%	74.82%
	#3	48.53%	61.97%	61.97%	63.93%	74.45%
B04	#1	47.16%	55.85%	71.59%	84.22%	91.51%
	#2	49.55%	58.69%	75.23%	88.50%	91.51%
	#3	49.55%	58.69%	75.23%	88.50%	91.92%
B06	#1	32.87%	92.73%	92.73%	98.02%	97.35%
	#2	31.94%	90.91%	90.91%	96.04%	97.35%
	#3	31.94%	90.91%	90.91%	96.04%	97.35%
B07	#1	**50.00%**	**66.93%**	**66.93%**	**75.71%**	**58.28%**
	#2	**49.25%**	**65.76%**	**65.76%**	**74.29%**	**57.28%**
	#3	**52.00%**	**66.93%**	**66.93%**	**79.52%**	**58.28%**
B08	#1	45.75%	60.63%	60.63%	81.90%	82.03%
	#2	51.42%	68.13%	68.13%	91.38%	98.15%
	#3	54.72%	72.50%	72.50%	98.28%	98.26%
B09	#1	38.89%	53.41%	53.41%	59.28%	48.89%
	#2	47.66%	65.46%	65.46%	72.85%	90.56%
	#3	50.00%	68.67%	68.67%	76.02%	91.22%
B10	#1	45.36%	69.90%	69.90%	74.43%	77.70%
	#2	51.32%	79.08%	79.08%	84.66%	92.22%
	#3	51.66%	79.59%	79.59%	85.23%	92.13%
B11	#1	56.33%	64.79%	64.79%	73.91%	79.99%
	#2	54.69%	62.91%	62.91%	71.74%	81.00%
	#3	62.86%	72.30%	72.30%	82.61%	84.52%

Table 4: *Influence of the rules on fault coverage*

	Correlation coefficient
Without rules	-0.1323
Rule A	0.6099
Rule A & B	0.7293
Rule A & B & C	0.7753

Table 5: *Influence of the rules on correlation*

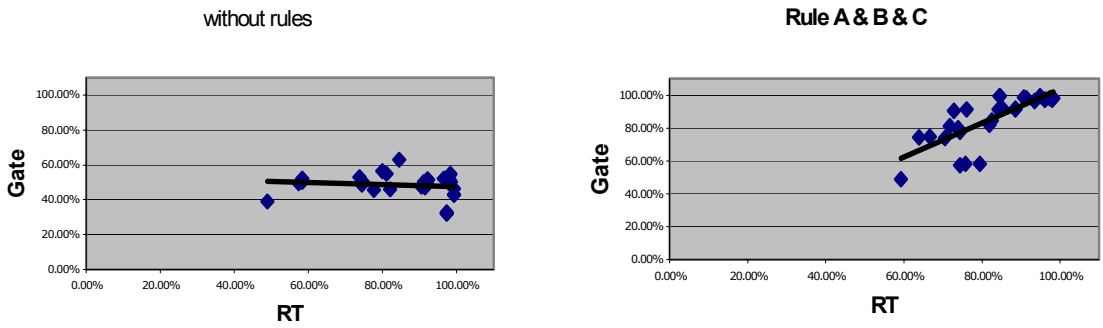

Figure 2: *Influence of the rules on correlation*

5. Conclusions

The paper shows an application of RT-level fault simulation, this proving that considering testing before synthesis is feasible.

A fault model is proposed and, thank to a careful identification of redundancies removed by synthesis, and is shown to be highly correlated with Gate-level fault coverage. This allows designers to predict circuit testability before synthesis.

6. References

[Beiz90] B. Beizer, *Software Testing Techniques (2nd ed.)*, Van Nostrand Rheinold, New York, 1990

[CCSS00] F. Corno, G. Cumani, M. Sonza Reorda, G. Squillero, *RT-level Fault Simulation Techniques based on Simulation Command Scripts*, "DCIS 2000: XV Conference on Design of Circuits and Integrated Systems", Le Corum, Montpellier, November 21-24, 2000

[CPRS96] F. Corno, P. Prinetto, M. Rebaudengo, M. Sonza Reorda, *GATTO: a Genetic Algorithm for Automatic Test Pattern Generation for Large Synchronous Sequential Circuits*, "IEEE Transactions on Computer-Aided Design", August 1996, Vol. 15, No. 8, pp. 943-951

[CSSq00] F. Corno, M. Sonza Reorda, G. Squillero, *Exploiting ITC'99 benchmarks for developing an RT-level ATPG tool*, to appear on IEEE Design & Test, Special issue on Benchmarking for Design and Test, June 2000

[CSSq00b] F. Corno, M. Sonza Reorda, G. Squillero, *High-Level Observability for Effective High-Level ATPG*, VTS-2000: 18th IEEE VLSI Test Symposium, May 2000

[DGKe96] S. Devadas, A. Ghosh, K. Keutzer, "An Observability-Based Code Coverage Metric for Functional Simulation," *Proceedings IEEE/ACM International Conference on Computer Aided Design*, 1996

[FADe99] F. Fallah, P. Ashar, S. Devadas, "Simulation Vector Generation from HDL Descriptions for Observability-Enhanced Statement Coverage," *Proceedings 35th Design Automation Conference*, 1999, pp. 666-671

[FDKe98] F. Fallah, S. Devadas, K. Keutzer, "OCCOM: Efficient Computation of Observability-Based Code Coverage Metrics for Functional Verification," *Proceedings 34th Design Automation Conference*, 1998

[FiFu00] A. Fin, F. Fummi, "A VHDL Error Simulator for Functional Test Generation," *Proceedings of the Design, Automation and Test Conference*, 2000, pp. 390-395

[RiUc96] T. Riesgo, J. Uceda, "A Fault Model for VHDL Descriptions at the Register Transfer Level," *Proceedings of EURO-DAC/EURO-VHDL*, 1996