Testability Analysis of Behavioral-Level VHDL Specifications

Erik Larsson and Zebo Peng
Department of Computer and Information Science
Linköping University, S-581 83 Linköping, Sweden
Phone: +46 13 284046, Fax: 46 13 282666, Email: {erila, zpe}@ida.liu.se

Abstract

A behavioral testability analysis technique is proposed for early prediction of testability by analyzing behavioral specifications written in VHDL. The technique extracts testability properties by an analysis of variable range, operation testability and statement reachability. It predicts the testability for the whole design with a low computational cost. Experimental results show that the behavioral testability analysis technique predicts the hard-to-test parts accurately and efficiently, and can be used to guide partial scan selection.

1. Introduction

Due to the increasing complexity of digital designs, modeling techniques at the behavioral level have been developed and many design and verification activities take place at this level. In order to reduce the test generation and application complexity, it is also important to predict testability of a design at the behavioral level and to make sure the final design is testable. At the behavioral level the functional properties of the design is explicitly captured and can be used to speed up testability analysis. These informations are difficult to extract from a gate-level design.

Early work in testability analysis is, however, usually carried out at the gate level [1,2,3,4] and the register-transfer level [5,6,7]. On the behavioral level Chen *et. al.* introduced a path analysis method to classify, based on controllabilities, the variables into two classes, completely controllable (CC) and noncompletely controllable (NCC) [8]. The classification is based on the ability to control the bits in the variables.

We propose a behavioral testability analysis technique which predicts the hard-to-test parts by extracting testability properties from a behavioral VHDL specification. Our technique analyzes the testability of the whole design with a low computational cost and it is based on analysis of variable range, operation testability and statement reachability. One feature of our approach is that the impact of loop structures in the behavioral specification is taken into account in the analysis process. The result from the testability analysis is used to guide partial scan selection as well as other DFT techniques.

We assume that the behavioral specification is given as a synthesizable subset of behavioral VHDL [9]. We also assume that Automatic Test Pattern Generation (ATPG) is used and it is random based and/or deterministic based and oriented towards the commonly used stuck-at fault model.

2. Behavioral Testability Metrics

Our behavioral testability metrics consists of *Variable Range, Operation Testability* and *Statement Reachability.*

2.1 Variable Range

If the value range of a variable is limited at a line in the behavioral specification, it reduces the test vector set and makes it harder to test the related hardware. The $VR(l, v)$ denotes the value range of variable v at line l in the behavioral specification, where $l \in L$. L is the set of lines in the behavioral specification and $defVR(v)$ is the defined value range for a variable v.

The *relative value range*, *RVR*, for a variable v at line l, is defined as:

$$RVR(l, v) = \frac{|VR(l, v)|}{|defVR(v)|}$$

2.2 Operation Testability

The *Operation Testability, OpT*, captures the change in distribution of test vectors in the output of an operation assuming all possible test vectors on its input. The optimum case is when the test vectors on an input of an operation are complete and uniformly distributed, the output vectors are also complete and uniformly distributed. This case cannot be satisfied by most operations when they are implemented on hardware. The difference of a distribution on the output of an operation and an uniform distribution, $Q(op)$, can be measured by:

$$Q(op) = \sum_{i=1}^{r} \frac{(x_i - n \times p_i)^2}{n \times p_i}$$

where x_i is the number of occurrence of value i, n is the total number of outputs, p_i is the expected probability for value i when each i is assumed to occur with the same frequency, and r is the number of possible values in the output.

Our metrics aims at showing the relative difference in disturbance of the output due to different operations performed by the modules. The *Operation Testability*, *OpT*, of an operation is defined as:

$$OpT(op) = \left(\frac{1}{Q(op)}\right)^{(1/b)}$$

where b is the word-length, i.e., number of bits of the input operands. Table 1 show the Q and the OpT values for the arithmetic operations at different word-length where a lower value for OpT corresponds to an operation which causes a design to be harder to test.

operation		add	sub	mult	div
2-bit	Q	6	6	48	13.5
	OpT	0.41	0.41	0.14	0.27
4-bit	Q	86	86	1408	1196
	OpT	0.33	0.33	0.16	0.17
8-bit	Q	21846	21846	603456	5333181
	OpT	0.29	0.29	0.19	0.14

Table 1: Q and OpT values for arithmetic operations.

We would like to use the operation testability metrics to characterize the testability of a line as a whole. We define the *Test Hardness*, *TH*(l), and *Line Operation Testability*, *LOT*(l), at a line l is given by:

$$TH(l) = \sum_{op \in Op_l} (1 - OpT(op)) \qquad LOT(l) = 1 - \frac{TH(l)}{TH_{max}}$$

where Op_l is the set of operations at line l and TH_{max} is the maximal test hardness at a line in the original specification.

2.3 Statement Reachability

Some statements in the behavioral specification are more difficult to reach then the others due to the control flow and statements that are hard to reach tend to cause test problem.

The *true* (*false*) *condition scope*, $cs_t(c_i)$ ($cs_f(c_i)$), of a condition, c_i, is the set of lines in the behavioral specification which will be executed if the condition c_i=*true* (*false*). Let C be the set of conditions in the specification and $p_t(c_i)$ is the probability of condition c_i being true and $p_f(c_i)$ is the probability of c_i being false, then the *Statement Reachability*, $SR(l)$, is given by:

$$SR(l) = \prod_{c_i \in C} \begin{cases} p_t(c_i) & l \in cs_t(c_i) \\ p_f(c_i) & l \in cs_f(c_i) \\ 1 & otherwise \end{cases}$$

3. Application of the Testability Metrics

The metrics is not targeted at any particular DFT technique. We will show how it can be used when partial scan technique is used. We assume that each variable is implemented as a dedicated register. Our objective is to identify already in the behavioral level which variables should be selected as a scan variable and eventually mapped to a scan register.

The flip-flops in the controller, the *state variable*, does not correspond to any variable in the behavioral specification. However, the flip-flops in the state variable are often the cause of problems in test generation and test application and must be considered.

The state variable is indirectly used at every line in the behavioral specification and it can only have one value at each state. If we assume that each line will correspond to one state then the value range at each line is $1/|L|$. We can then use the definition in section 2.1 to get its *RVR* [10].

The *Testability*, $T(v)$, for a variable v is given by:

$$T(v) = \sum_{l \in L} \frac{m}{n} \times (\alpha_1 RVR(l, v) + \alpha_2 LOT(l) + \alpha_3 SR(l))$$

where n is the number of times variable v occur in the behavioral specification and m is 1 if variable v occur at line l otherwise m is 0. α_1, α_2 and α_3 are user-defined constants which are used to reflect the importance of the three metrics for different test strategies.

One feature of our technique is that it allows the scan variable selection procedure to be performed in an iterative manner. After one iteration of analysis and testability insertion we can analyze the testability of the modified design and further improve the testability.

The *Relative Value Range* for a scanned variable is 1 and scanning a variable v reduces the Test Hardness of the lines that contain v to: $TH(l) = (TH(l) \cdot (i-j))/i$, where i is the number of variables and j the number of variables at line l which are selected to be included in the scan path in this iteration.

4. Experimental Results

We use our high-level synthesis tool and Mentor Graphics logic synthesis and test generation tools as the experimental platform and the benchmarks we use are Diff, Sqrt, Mag, Dct and Tseng.

First we show the correlation between our testability metrics and the fault coverage. The statement reachability, operation testability and the relative variable range for the diff benchmark are calculated using α_1=1, α_2=1 and α_3=1, which gives equal importance to the three metrics. After the analysis of the Diff benchmark we made experiments to see if the prediction for the testability of the variables was correct, the results are given in Table 2. In the table x_1 is ranked as the hardest and by scanning it we get the best fault coverage improvement, from 16.49% (no scan) to 97.61%. On the other hand, register y, is considered to be not difficult to be tested, therefore scanning it will not improve the design's testability, we get only 16.15% fault coverage as the result.

The comparison between analysis at behavioral and gate level was performed as follows. First we did a testability analysis on the behavioral VHDL to find hard-to-test variables and rank them. The variables which are hardest to test are scanned. Secondly, we let the Mentor Graphics gate level DFT-tool select the same number of flip-flops as was selected by our behavioral analysis. The results are shown in Table 3, where the fault coverage is almost the same for all benchmarks except for the Diff benchmark where the behavioral analysis outperformed the gate-level analysis.

An advantage of the behavioral testability analysis is that it takes much less time to perform the testability analysis task, and therefore speeds up the design process.

	x1	s.r	x	u1	u	y1	y	no scan
T	1.30	1.57	1.71	1.75	1.91	2.09	2.29	-
F.C(%)	97.61	97.28	97.69	91.78	92.33	15.61	16.15	16.44

Table 2: Experimental results for the Diff benchmark. Testability (T) and fault coverage (F.C) for all variables.

Design	#flipflops	#scanned flipflops	Fault coverage(%)		CPU(sec)	
			B-L	G-L	B-L	G-L
Diff	100	16	97.3	19.7	2.3	18.8
Sqrt	86	70	99.0	99.0	5.0	19.4
Mag	52	36	95.9	94.9	2.1	5.2
Dct	164	4	98.8	99.1	2.3	7.53
Tseng	213	5	96.3	93.9	2.3	15.1

Table 3: Fault coverage comparison of testability analysis at the behavioral (B-L) and the gate level(G-L).

5. Conclusions

We have proposed a behavioral testability analysis technique for early prediction of testability by analyzing the behavioral VHDL specification. The technique is based on analysis of variable range, operation testability and statement reachability. The testability for the whole design is predicted at a low computational cost, since the analysis is performed on the behavioral specification which is much simpler then its gate level implementation.

By experiments we have shown that the behavioral testability analysis technique can predict the hard-to-test parts accurately and efficient and that the testability analysis results can be used to guide the selection of partial scan registers. The testability metrics also gives an indication of the test features of the final design at a very early design stage. This information can be used by the designer to select an appropriate test strategy and to make efficient test plan for the final design.

6. References

[1] L. H. Goldstein, Controllability/Observability Analysis of Digital Circuits, *Trans. on Circ. & Sys.*,no. 9, Sep. 1979, pp 685.

[2] P. S. Parikh, M. Abramovici, Testability-Based Partial Scan Analysis, *J. of Electronic Test and appl.,* 7, 61-70, 1995.

[3] R. A. Rutman, Fault detection test generation for sequential logic by heuristic tree search, *IEEE Comp Gr Rep.,*R72-187, 1972.

[4] T. Chen, M. Breuer, Automatic Design for Testability Via Testability Measures, *Trans. CAD, vol. 4*, Jan., pp 3-11, 1985.

[5] C. H. Chen, P. R. Menon, An Approach to Functional Level Testability Analysis, *Proc. of ITC'89,* pp 373-380.

[6] X. Gu, K. Kuchcinski and Z. Peng, Testability Analysis and Improvement from VHDL Behavioral Specifications, *Proc. EURO-DAC'94*, pp. 644-649.

[7] M. L. Flottes, R. Pires, B. Rouzeyre, Analyzing testability from behavioral to RT level, *Proc. ED & TC'97,* pp 158-165.

[8] C.-H. Chen, T. Karnik and D. G. Saab, Structural and Behavioral Synthesis for Testability Techniques, *IEEE Trans. CAD*, 13(6), pp. 777-85, 1994.

[9] P. Eles, K Kuchcinski, Z. Peng, M. Minea, Compiling VHDL into a High-Level Synthesis Design Representation, *Proc. EURO-DAC'92*, pp 604-609.

[10] E. Larsson, Z. Peng, Early Prediction of Testability by Analyzing Behavioral VHDL Specifications, *Proc. NORCHIP'97,* pp 259-266.