A Behavioral-Level Testability Enhancement Technique

Erik Larsson and Zebo Peng
Dept. of Computer and Information Science, Linköping University, Sweden
Email: {erila, zpe}@ida.liu.se

Abstract[1]

A testability enhancement technique for improving the testability of a behavioral-level VHDL specification is proposed. The technique is based on a set of transformations and the selection of transformations is guided by a behavioral-level testability analysis algorithm. The main advantage of our technique is that it can be used in the very early design stage, and it reduces the complexity of the testability analysis and enhancement tasks. Experimental results show the efficiency of our approach.

1. Introduction

The testability of a digital system can be enhanced by many Design for Testability (DFT) techniques. One effective and commonly used DFT technique is full scan. Several methods have recently been developed where a subset of the flip-flops in the design are selected to be included in the scan chain.

Several other DFT techniques have been developed. At the gate-level, Chen *et al.* use test point insertion to improve the controllability and/or the observability of a wire in a circuit [1]. The technique is used by Gosh *et al.* to multiplex variables that are hard to control [5]. At the behavioral level, Varma *et al.*, Carletta *et al.* and Chen *et al.* have proposed techniques which are also based on test point insertion [10, 4, 2]. Hsu *et al.* and Gu *et al.* use such a technique to improve the controllability of hard-to-test branches [7, 6]. Testability enhancement can also be carried out during the synthesis process, such as high-level synthesis [4].

In this paper, we propose a behavioral-level testability enhancement technique which modifies the VHDL specification in order to improve its testability. Our technique is based on a set of testability improvement transformations including *read insertion*, *write insertion*, a set of *boolean insertions* and *reach insertion*. Furthermore, we propose a strategy for selecting these testability improvement transformations.

Our input specification is given as a subset of behavioral VHDL, which is synthesizable by a high-level synthesis tool [3]. We 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. Our testability enhancement technique is guided by a testability metrics $T(v)$, for each variable v, in the behavioral specification, based on *Value Range*, *Operation Testability* and *Statement Reachability* [8, 9].

2. Testability Improvement Transformations

We define a set of transformations applicable directly to the behavioral VHDL specification, using some examples.

If the contents of variable y in Figure 1(a) is hard to control, a *read insertion*, highlighted in the box, can be used. An extra primary input, $test_i$, is added to indicate if the design is in normal mode or test mode. PI is a primary input which can be an existing one or an extra one added only for test purposes. The penalty introduced by adding an extra primary input only for test is usually too high. Here, we assume that an existing primary input can be used.

If a variable x in Figure 1(b) or (c) is hard to observe, a *write insertion* can be applied to improve its observability. The added code is highlighted in the box. Here we distinguish two types of *write insertions*. The first type writes a value directly to a primary output dedicated for test, Figure 1(c). In such case, we do not have to check whether the output is produced during the test mode or the normal mode. However, if we apply *write insertion* and it is possible to use an existing primary output for observation, we only have to add one extra pin, $test_j$, to check whether we are in test mode or normal mode, illustrated in Figure 1(b). Again, the cost of extra primary outputs which will only be used for test is usually unacceptable. Therefore we usually assume the case where an existing primary output is used.

```
(a) Read
    :
if test_i then
   y:=PI;
end if;
x:=y+5;
    :

(b) Write
    :
x:=y+5;
if test_j then
   PO:=x;
end if;
    :

(c) Write
    :
x:=y+5;
new_PO:=x;
    :
```

Figure 1: Read insertion (a) and Write insertion (b,c).

Traditional test point insertion can be applied on certain structures in the behavioral specification, such as loops and branches, which are known to cause test problems. We define three types of *boolean insertions: or insertion, and insertion* and *and/or insertion* and they are used on the following VHDL constructs: IF, WHILE, FOR and CASE statements. An example is given in Figure 2(a) where an *and insertion*, highlighted in the box, is used to make it easier to exit a loop in the test mode.

The *or insertion* and the *and insertion* require knowledge about which outcome (true or false) of a condition is hard to achieve. The *and/or insertion* as shown in Figure 2(b) gives us full control of the outcome of condition c. When in test mode, $test_l$ is true, and condition c can be set to true or false. However, two extra primary inputs are required.

Some statements in a behavioral VHDL specification are harder to reach compared to other statements due to the control flow [8,9]. The *reach insertion* transformation is used to make the hard-to-reach statements easier to reach when the design is in the test mode. In Figure 2(c) we illustrate the *reach insertion* on a small VHDL example and a corresponding control-flow graph. By adding an extra primary input pin, $test_m$, to indicate whether the design is in the normal mode or the test mode, the hard-to-reach statements can be easily reached in the test mode.

```
(a) And insertion
    :
while z<10 and test_k loop
   z:=z+1;
end loop
    :

(b) And/Or insertion
    :
if test_l then
   c := PI;
end if;
if c then
    :
else
    :
end if;

(c) Reach insertion
    :
if not test_m then
   while x<10 loop
      x:=x+1;
   end loop;
end if;
y:=x+4
    :
```

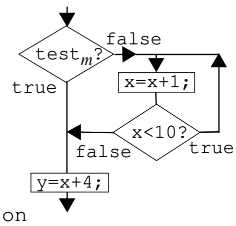

Figure 2: Boolean insertion (a, b) and reach insertion(c).

1. This work has partially been supported by the Swedish National Board for Industrial and Technical Development (NUTEK).

3. Application of the Enhancement Technique

Our behavioral-level transformation selection strategy is illustrated in Figure 3. First we perform testability analysis of the behavioral-level VHDL specification and we rank the hard-to-test parts, using our testability metrics [8, 9].

The penalty of using a boolean insertion is low and it is targeted towards a well-known test problem, we give high priority to selecting it. In our approach we select boolean insertion for a variable if it is among the *k* hardest-to-test variables according to our test metric, where the designer define *k*.

For a given hard-to-test condition we have to decide which boolean insertion to use. The boolean insertions affect the statement reachability and the variable range and to guide the selection, we define a relative improvement metric. The set of lines affected by a condition *c* is captured by the condition scope for a condition being true, $cs_t(c)$, and being false, $cs_f(c)$ [8,9]. The *Condition Testability*, $CT(c)$, for a condition *c* is:

$$CT_i(c) = \sum_{l \in cs_t(c) \cup cs_f(c)} (RVR(l, v) + SR(l, v))/n$$

where *n* is the number of times a variable occurs at the lines in the condition scope and *i* is 0 when no insertion is used, 1 for and insertion, 2 for or insertion and 3 for and/or insertion. Let the *Relative Condition Testability*, $RTC_i(c)$, for a condition *c* be defined as (*i* is defined as above):

$$RCT_i(c) = CT_i(c) - CT_0(c)$$

If boolean insertion is not applicable on any of the *k* hardest variables in the list of hard-to-test variables, we use read insertion or write insertion. Note, read insertion and write insertion are applicable on lines with assignments and arithmetic operations. If variable *v* is the left-hand variable in such a line, it affects the observability, otherwise the controllability.

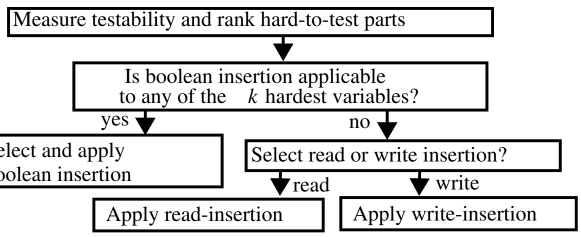

Figure 3: Transformation selection strategy.

4. Experimental Results

We use the CAMAD high-level synthesis tool [3] and Mentor Graphic's logic synthesis and test generation tools as the experimental platform. The benchmarks we use are the Differential equation (Diff) and Counter (Cnt).

We first performed experiments to show the usefullness of our strategy for selecting read insertion or write insertion for a given variable. The testability analysis of a design is performed and the variables are ranked according to their testability. For each ranked variable we use our strategy to determine if read insertion or write insertion is appropriate. For the lines in the behavioral VHDL specification where read insertion and write insertion can be considered we compare the ratio observability/controllability (o/c). A high o/c value indicates that a read insertion is preferable, while a low value indicates that a write insertion is to be preferred. The results of the experiments performed on the Diff benchmark are presented in Table 1. For instance, variable x_1 which is ranked as the hardest to test with testability of 1.30 has a o/c ratio of 0.87. It is a high value compared to the o/c ratio for all other variables, which indicates that we should use a read insertion. By using a read insertion on variable x_1 we achieve a fault coverage of 94.5%. We also performed experiments where we applied the not-selected insertion. For instance, the fault coverage for variable x_1 when we used the write insertion was only 15.7%.

In the second experiment, we assume that we could freely use boolean insertion, read insertion and write insertion accordingly. The variables were ranked according to their testability. If boolean insertion was applicable on any of the *k* hardest to test variables, the variable was selected for boolean insertion. If not, we select read insertion or write insertion for the hardest to test variable. We let *k* be 50% of the variables in this experiments. The results are presented in Table 2, which indicate that a high fault coverage is achieved at a very low area penalty.

Variables	T(v)	Read/write selection				
		o/c	Selected insertion	F.C(%)	Not selected insertion	F.C(%)
x_1	1.30	0.87	read	94.5	write	15.7
x	1.71	0.86	read	94.5	write	15.7
u_1	1.75	0.55	write	95.2	read	14.1
u	1.91	0.51	write	94.7	read	16.9
y_1	2.09	0.86	read	12.7	write	70.7
y	2.29	0.85	read	13.1	write	16.5

Table 1: Experiments on read/write selection.

Design	Transformation	Fault coverage (%)		Area (mm^2)		
		No dft	Dft	No dft	Dft	Overhead(%)
cnt	1 or insertion	39.7	96.3	0.5525	0.5539	0.25
diff	1 or insertion	16.4	96.8	7.3596	7.3674	0.10

Table 2: Experiments on testability transformation selection.

5. Conclusions

We have proposed a behavioral-level testability enhancement technique for modification of behavioral-level VHDL specifications to make them more testable. The technique includes read insertion, write insertion, a set of boolean insertions and reach insertion. We have also defined a transformation selection strategy based on a testability analysis technique directly applicable to the behavioral VHDL specifications.

The main advantage of our approach is that it can be used in the early design stage, and it reduces the complexity involved in the testability analysis and enhancement tasks. By experiments we have shown the efficiency of our approach and the transformation selection strategy.

References

[1] T. Chen, M. Breuer, Automatic Design for Testability Via Testability Measures, *IEEE Transactions on Computer-Aided Design, vol. CAD-4,* No. 1, Jan. 1985, pp 3-11, 1985.

[2] C.-H. Chen, T. Karnik, D. G. Saab, Structural and Behavioral Synthesis for Testability Techniques, *IEEE Transactions on Computer-Aided Design*, 13(6), pp. 777-85, 1994.

[3] P. Eles, K Kuchcinski, Z. Peng, M. Minea, Compiling VHDL into a High-Level Synthesis Design Representation, *Proceedings of the EURO-DAC,* 7-10 Sep. 1992, pp 604-609.

[4] J. Carletta, C. Papachristou, Behavioral Testability Insertion for Datapath/Controller Circuits, *JETTA 11,* pp. 9-28, 1*997.*

[5] I. Ghosh, A. Raghunathan, N. K. Jha, Design of Hierarchical Testability of RTL Circuits Obtained by Behavioral Synthesis, *Proceedings of the ICCD*, October 1995.

[6] X. Gu, E. Larsson, K. Kuchcinski, Z. Peng, A Controller Testability Analysis and Control Enhancement Technique, *Proceedings of European Design &Test Conference*, pp. 153-157, 1997.

[7] F. Hsu, E. Rudnick, J. Patel, Enhancing High-Level Control-Flow for Improved Testability, *Proc. of ICCAD*, Nov., 1996.

[8] E. Larsson, Z. Peng, Early Prediction of Testability by Analyzing Behavioral VHDL Specifications, *Proceedings of NORCHIP Conference,* pp. 259-266, Tallinn, Nov., 1997.

[9] E. Larsson, Z. Peng, Testability Analysis of Behavioral-Level VHDL Specifications, *Proceedings of ETW,* pp. 143-144, 1998.

[10] K. Varma, P. Vishakantaiah, J. Abraham, Generation of Testable Designs from Behavioral Descriptions using High Level Synthesis Tools, *Proc. of VLSI Test Symposium,*pp.124-130, 1993.