Implementing Adaptive Random Test Generation in VHDL

Armita Peymandoust, Massoud Shadfar, Zainalabedin Navabi
Electrical and Computer Engineering Department
Faculty of Engineering, Campus No. 2
University of Tehran
14399, Tehran IRAN
Tel: +98-21-800-8485; Fax: +98-21-646-1024
Email: navabi@ece.ut.ac.ir

ABSTRACT
This paper presents a VHDL test bench for random generation of test vectors for gate level structural digital circuits. Adaptive generation of test vectors is being focused here. In this approach, one of several predetermined vector generation schemes is selected based on the anticipated fault detection rate.

1. Introduction
In some of our earlier works, we have presented VHDL models for Critical Path Tracing (CPT [1]). A circuit using these models is able to report its detected structural faults on its internal lines. We have also presented VHDL fault collapsing models [2] that are able to generate a list of distinguishable faults. The test bench being presented here assumes that a fault list is available and it selects a random test vector generation scheme in order to detect faults of such a fault list with fewer number of test vectors. Each test vector is generated based on a given signal probability (probability of a line being ‘1’), determining random test generation scheme. Fault detection is done by activating critical path tracing VHDL models.

The process of test generation is an adaptive one [3]. Test vectors are generated on circuit primary inputs. Depending on controllability and observability of each circuit line, calculated for a given scheme (PI signal probability), detection probability of specific line faults (sa0 or sa1) can be calculated. Fault detection probabilities for fault not yet detected are instrumental in the calculation of the anticipated fault detection rate for a given random test generation scheme. At fixed intervals during the test generation process, fault detection rate for all possible test generation schemes are calculated, and the scheme with the best rate will be selected for the next test generation interval.

In addition to Critical Path Tracing and Fault collapsing VHDL models, adaptive test generation also requires VHDL models for calculating line controllability and observability, as well as VHDL models for application of test data with given signal probabilities to circuit primary inputs. Fault collapsing (FC) and controllability & observability (CO) models are used independently to generate circuit fault list and line CO values. LFSRs are used for generation of test vectors for various schemes. A test controller written in VHDL is responsible for application of data to circuit under test, preparing FC and CO information, selecting appropriate schemes, and reporting the generated tests.

In this paper we will describe the complete adaptive random test generation environment with emphasis on the techniques used for selection of test schemes. Section 2 presents the test environment and the role of the test controller. In section 3 we will describe environment for generation for fault lists and CO data. Section 4 shows the process of keeping track of undetected faults. In Section 6 we will discuss models for primary inputs for generation of specific adaptive random test scheme. Section 7 discusses algorithms for selecting best test vector generation scheme.

2. Test Environment
The implementation of adaptive random test generation requires the use of a VHDL circuit description and a VHDL test bench. The test bench applies data to the circuit, performs measurements and based on information received from the circuit decides the method for generation of the next set of vectors. The general block diagram is shown in Figure 1. The circuit description consists of instantiation of gate level components. The gates of the circuit are bound to critical path tracing VHDL models. Such models are capable of finding faults on their inputs and outputs for a given test vector. In addition to actual gate models, adaptive random test generation requires controllability
and observability of all gate ports as well as the reduced fault list of the circuit. Therefore the circuit description consists of CPT gate models associated with CO and FC information.

![Diagram of test generation process](image)

**Figure 1. Adaptive Random Test Generation Environment**

The required test bench consists of a test vector generation VHDL model and a test controller. The test vector generation model connects to the primary inputs of the circuit under test and is capable of producing pseudo random test vectors when commanded to do so by the test controller. The test controller specifies signal probabilities of each PI.

Initially the test controller reads CO and FC external data and associates this data with gates of the circuit under test. Then, the test controller starts the process of test vector generation by specifying a test scheme for the PI vector generation VHDL models. As tests are generated and applied to the circuit the following will take place for each test vector applied:

- **Controller:** Stores generated vectors,
- **Controller:** Keeps a count of tests,
- **Circuit:** Performs fault simulation,
- **Circuit:** Marks off detected faults.

After application of \( n \) test vectors, the test controller stops the vector generation process and re-evaluates the vector generation scheme used for generation of PI test vectors. A test generation scheme determines signal probabilities of the individual PI lines. At the start, a scheme is used which that causes all PI lines to have a probability of 0.5 for becoming '1', we will refer to this scheme as the 0.5 scheme.

The CO and remaining undetected faults of circuit gates help the test controller to decide which predefined test scheme to use for the next \( n \) test vectors. The test controller performs calculation for predicting how well each of the possible schemes will detect remaining faults. This calculation is based on CO information of each scheme and undetected fault list. Based on the result of this calculation, a test scheme for the next \( n \) test vectors will be selected.

In the sections that follow we will discuss the preparation of gate information, importing data into test controller, PI vector generation, fault simulation for marking off detected faults, and test scheme evaluation techniques.

### 3. Generation of FC and CO Records

As described in the previous section, evaluation of a test scheme requires controllability and observability of circuit lines to be known for each of the test schemes. This evaluation also requires a list of detectable faults to be available. This information is generated by running circuit gate level description with FC or CO VHDL models.

As depicted in Figure 2, a VHDL netlist corresponding to the circuit under test is bound to FC (fault collapsing) VHDL models to produce a collapsed fault list. The process of fault collapsing is referred to reducing an original fault list that contains \( sa1 \) and \( sa0 \) faults on all circuit nodes to a reduced fault list that only contains those faults that are not equivalent. Equivalent faults are those that cannot be distinguished on the primary outputs by application of test vectors on the primary inputs. VHDL models and format of the generated fault list are described in reference 2. Collapsed fault list is written to an external file.

![Diagram of FC and CO data generation](image)

**Figure 2. FC and CO Data Generation**

The same VHDL netlist used for FC, is also used for calculation of controllability and observability...
of the circuit gate inputs and output. This information is calculated for several test vector generation schemes. A test scheme defines signal probabilities for the primary inputs. This value is used for calculation of controllability of a circuit line. Figure 3 shows an example circuit with a 0.5 test scheme. The AND gate output becomes '1' only when both inputs are '1' and therefore has a probability of 0.25. Similarly the OR gate output controllability is calculated by the equation shown below.

\[ C_{out} = 1 - (1 - C_m1) \times (1 - C_m2) = 0.625 \]

Using controllability values and assuming an observability of 1 for the primary outputs, observability of each of the circuit lines can be calculated. Observability values calculated for circuit lines of Figure 3 are shown in this figure.

![Controllability and Observability](image)

VHDL models for CO calculation have been developed. A test bench applies a test scheme value (i.e., 0.5, 0.25, 0.75) and performs VHDL simulation. The gate models use C values on their input and calculate a C value for their outputs. Calculation of output C values continues until it reaches a primary output. Observability values for primary outputs are assigned to these signals I FS after all controllabilities have been calculated. The gates now operate in output to input mode, and calculate their input O values in this direction. Eventually after several delta times O values for all circuit lines are calculated. This process continues for all test scheme and the calculated CO values will be written to an external file. The CO and FC external file information will be used in predicting best test scheme.

4. Importing FC and CO Information

The process of selecting test vector generation scheme depends on CO information for each scheme. Also as faults are detected, a list of undetected faults is necessary for selection of a proper scheme. Because this information is frequently referred to by gate models and the test controller, we have used shared variables to store CO and FC information of the circuit gates.

![Figure 4. Shared Variable Record for FC & CO information](image)

At the start of simulation the test controller reads FC and CO external files discussed in the previous section. This information is stored in shared variable records (g_node) shown in Figure 4. All g_node records are arranged in a dynamic shared variable linked list (gate_table). Each g_node record has an identifier that relates it to the gate that it corresponds to. Circuit gates are identified by an id number appearing in their generic map. Records in the g_node contain undetected faults and CO information for each circuit gate. Figure 5 shows the relation between the circuit, test controller, and the shared variable gate images.

![Figure 5. Shared Variable Gate Images](image)
The VHDL code for reading FC and CO information consists of function calls using VIDL 93 TEXT10 procedures. The format of input FC and CO files are shown in Figure 6. The CO information in g_nodes remain unchanged for the entire test generation run. Gate faults that appear in the faults record of g_node, will be marked off by the gate models when detected.

```
TYPE GATE ID ( SINGLE STUCK FAULT (SSF) LIST)
AND2 ( 3 ) (1:so_1 2:sq_1 0:so_1 sq_0 )
```

NAND no. 0006 reporting:
For iteration no. 0001
5.000000e-001 : controllability of input 1
6.250000e-002 : observability of input 1
5.000000e-001 : controllability of input 2
6.250000e-002 : observability of input 2
7.500000e-001 : controllability of output
1.250000e-001 : observability of output

**Figure 6. FC and CO File Formats**

5. Detecting Faults of the FC List

As test vectors are applied to the circuit under test, circuit gate models perform critical path tracing as a means of fault simulation. This mechanism enables each gate to become aware of faults on its ports that can be detected by the applied test vector. A g_node record contains faults fields for each of gate ports. The faults type, shown in Figure 4, is an enumeration type.

When FC external files are read, the faults field of a gate port will be assigned to one of the enumeration elements of the faults field. A '0' ('1') signifies that a sa0 (sa1) is to be detected on that port. The value 'B' indicates that both sa0 and sa1 values are among the list of faults to be detected and 'N' indicates no fault to be detected.

A CPT gate model that becomes aware of a sa0 (sa1) fault being detected by a certain test vector changes its '0' ('1') fault field value to 'N'. Two shared variables keep counts of original faults and detected faults. A CPT gate model is responsible for incrementing the number of detected faults when one is detected on any of its ports. The test controller uses the count shared variable for deciding on the test scheme or for stopping the test generation when a coverage is reached.

**Figure 7. Test Application and Fault Mark Off**

Figure 7 shows a block diagram for the implementation of components involved in keeping track of the detected faults and calculation of fault coverage. The primary inputs generate a test vector and apply it to the circuit. This vector will be recorded in a linked list shared variable by the test controller. The circuit performs fault simulation and adjusts gate image fault fields. The test controller initiates the next test application by clocking the primary input components.

6. PI Test Vector

Primary input test vectors are generated by a set of parallel LFSRs. An LFSR is dedicated to generation of serial inputs for a primary input. All the PI LFSRs are coordinated by a clock signal from the test controller and a shared variable containing a fractional number representing the test generation scheme. The clock signal from the test controller signals all LFSRs to send the next data on their serial output. The LFSRs perform certain Boolean functions to make the signal probability of their outputs the same as that of the current scheme. Figure 8 shows a partial LFSR VHDL description.
PROCEDURE srpg(seed:IN qit_vector;
    state:OUT qit_vector;
    one_bit:OUT bit;
    one_prob:IN real:=0.5) IS
BEGIN
- 10 3 0 Primitive polynomial
state(9):=seed(0) XOR seed(3);
state(8 DOWNTO 0):=seed(9 DOWNTO 1);
IF one_prob = 0.25 THEN
    one_bit := seed(0) AND seed(9);
ELSIF one_prob = 0.375 THEN
    one_bit := (seed(0) OR seed(9)) AND seed(5);
ELSIF one_prob = 0.5 THEN
    one_bit := seed(0);
ELSIF one_prob = 0.625 THEN
    one_bit := (seed(0) AND seed(9)) OR seed(5);
ELSIF one_prob = 0.75 THEN
    one_bit := seed(0) OR seed(9);
END IF;
END srpg;

Figure 8. PI LFSR

All PI LFSRs have 10 register bits implementing $x^9 + x^5 + 1$ which is a primitive polynomial. The seed of each LFSR is determined by the generic parameter which is mapped to unique PI id number. On each clock, LFSRs look up the scheme shared variable is only written into by the test controller.

7. Evaluating Test Schemes

The test controller begins the process of random test generation with a 0.5 scheme. After generation on $n$ test vectors ($n$ is a user defined constant that is partially dependent on the circuit size), the test controller reads CO information of lines that still have undetected faults and makes a prediction for the rate of detection for every possible test scheme. It then selects the scheme to be used by setting the scheme shared variable to the chosen scheme (0.5, 0.25, 0.75,...). This section presents details of the selection process and evaluation techniques.

Figure 9 shows a block diagram for the selection process. As shown, here a selected scheme becomes the basis of test generation for the next test interval ($n$). The process of selecting a scheme and adapting test pattern generation continues until a predefined fault coverage is achieved. Certain provisions for guarding against continuously changing schemes while trying to achieve a hard to reach fault coverage are put into program.

Figure 9. Adaptive Process

Starting with a complete set of faults, a given random test generation scheme is able to detect easy to detect faults at a relatively high rate. As faults are detected, detecting remaining faults becomes harder and the rate lowers. Intuitively, it can be seen that for a given random test generation scheme, there is an exponential relation between the number of applied test vectors and the estimated fault coverage. At any point in the test generation process, the slope of this exponential curve provides a prediction for the rate of detecting undetected faults.

7.1 Slope Rate Calculation

To calculate this slope, we will present mathematics that lead to an expression for the exponential relation between number of random tests and fault coverage.

Detection probability of a line is determined by its controllability and observability. For a node $i$, the probability of a stuck-at-value fault $(sa_v)$ being detected is that the node is observed and it can be controlled to the complement of $v$. The detection probabilities can be calculated as shown below:

$$PD_{sa0} = C_i \cdot O_i$$
$$PD_{sa1} = (1 - C_i) \cdot O_i$$

A cumulative detection probability is defined as the probability of detecting fault $j$ given $n$
independent trails. The equation for cumulative PD for fault $j$ is shown below:

$$CPD_j(n) = 1 - (1 - PD_j)^n$$

Given the cumulative fault detection probability of a fault $j$, and averaging this quantity for a set of $M$ undetected faults yields an expression that represents the fault coverage estimate as a function of $n$.

$$FCE(n) = 1/M \times \Sigma CPD_j(n) = 1/M \times \Sigma (1 - (1 - PD_j)^n)$$

The above expression for fault coverage estimate represents an exponential form, shown in Figure 10. This indicates that fault detection becomes harder when fewer faults remain to be detected. The slope of this curve represents the rate of faults being detected. The expression shown below represents this rate ($RFD = \text{Rate of Fault Detection}$).

$$RFD = -1/M \times \Sigma (1 - (1 - PD_j)^n) \ln(1 - PD_j)$$

This expression is implemented in VHDL and is used for selecting a test scheme.

![Figure 10. Fault Coverage Estimate](image)

7.2 Selecting a Scheme

A test scheme is selected based on the rate of fault detection. The parameters used for this calculation are $M$ (total number of undetected faults), PD (probability of detecting undetected faults), and $n$ (number of tests applied).

The total number of undetected faults ($M$) is available in a shared variable. PD for each gate input or output and for all test schemes is available in the shared variable image record (Figure 5). The controller keeps a count of number of applied tests ($n$).

![Figure 11. Slope Calculation for all Test Schemes](image)

As shown in the pseudo-code of Figure 11, at fixed test increments, the controller searches gate image records and evaluates RFD values for each scheme. The scheme with the highest RFD will be used for the next set of test vectors.

![Figure 12. Test Controller VHDL Code (partial)](image)
Figure 12 shows the VHDL description for the test controller. When test generation process begins, the shared variable gate_table will be filled. The code shown includes a FOR statement that is used to clock the PEs for specific test intervals. Each test generated in this interval is recorded in an external file by the add2list_list procedure.

At the end of a test interval, the coverage impure function and the current fault coverage is compared with the desired coverage. If the desired coverage is not achieved the compute_one_prob procedure is called to predict the best test scheme in the next test interval, and sets the shared variable corresponding to the test scheme.

8. Conclusions

We have presented a test environment for adaptive test generation in VHDL. The method uses fault simulation, fault collapsing, and controllability and observability calculations, for which VHDL gate models have been developed. The test bench and other utilities used in this implementation have all been coded in VHDL. In all such implementations we have taken advantage of VHDL '93 shared variables.

The method for the selection of test schemes has been developed so that it can be implemented in VHDL. At the timing of writing of this paper, the implementation is complete, but more tests and comparisons have to be made. Running benchmarks for comparing our method with other adaptive test generation methods in part of the continued work on this project.

This implementation is part of series of projects for an all-VHDL test solution [1, 2, 3, 4, 5].

REFERENCES:


