New Design Error Modeling and Metrics for Design Validation

Sungho Kang
Computer Engineering Research Center
2201 Donley Drive, Suite 395
Austin, Texas 78758

Stephen A. Szygenda
Engineering Science Building 240
The University of Texas at Austin
Austin, Texas 78712

Abstract

When simulation is used for design verification, a subset of simulation input patterns is used, since exhaustive simulation is usually not practical. In this case, the immediate question is how much of the design has been verified? To provide a measure of the simulation pattern coverage based on design error modeling, a new simulation coverage metric is introduced. This measure is useful for obtaining insight into the actual level of design validation, since it provides more realistic results than those which are presently available.

1 Introduction

Simulation[1] is an essential and commonly used method for verifying the design of digital systems. The main disadvantage of simulation is that as systems increase in size and complexity, simulating the system becomes very costly, both in time and space. Therefore, a measure of completeness is not available, since a subset of possible simulation patterns is used. In most cases, only a subset of all the possible simulation patterns are used for verification. Then, the immediate question arises as to how much of the design has been verified? Therefore, to obtain insight into the actual level of design verification, it is necessary to derive a simulation coverage for given simulation patterns[2].

Simulation coverage (SCP) is usually represented using simulation patterns as,

\[
SCP = \frac{\text{the number of simulated patterns}}{\text{the number of possible patterns}}
\]

This is based on the assumption that the probability of each design error occurrence and each pattern's effectiveness is uniformly distributed. This implies that simulation coverage is proportional to the number of simulated patterns. However, intuition suggests that patterns may have different weights.

In this research, the concept of design error modeling and a representation of simulation coverage based on this design error modeling are introduced. This results in a new design error validation metric as a measurement of design validation. To generate empirical results to test the proposed theory, error simulation and automatic error pattern generation (AEPG) were introduced and implemented, to measure the error coverage.

2 Theoretical Analysis

Let \( n \) be the number of primary inputs and \( u \) be the number of primary outputs in the circuit, respectively. Then, there are \( 2^n \) possible simulation patterns and \( 2^u \) possible functions, considering only 0 or 1 as inputs. Let \( M = 2^n \) and \( N = 2^u \). Let the simulation patterns be \( p_1, p_2, \ldots, p_M \) and the functions be \( f_1, f_2, \ldots, f_N \).

If a design function is given, the other possibilities of \( N \) are all design errors. An error whose output is functionally equivalent to the output of the desired function is not regarded as an error. Consider an error \( a \). Assume that under this error, function \( f_j \) is changed into \( f_k \). An error \( a \) is said to be detected by a given pattern \( p_i \) if \( f_j(p_i) \oplus f_k(p_i) = 1 \). The simulation coverage is represented as

\[
\frac{\text{the number of detected errors}}{\text{the number of errors}} \times 100 \% (1)
\]

The output can have more than one value for certain patterns. This is possible in the case of sequential devices, since the internal memory states can affect the outputs. In other words, the outputs can have different values, according to the internal memory state, for the same pattern. Let this output function be complex and the previous output function, which is uniquely decided, be simple, respectively.

2.1 Circuits with a Single Simple Output

This is the simplest case; for example, gates which have only one output and the desired behavior is represented using one function. Without loss of generality, let the desired behavior of the circuit be \( f_j \). Then, \( f_k \), where \( 1 \leq k \leq N, k \neq j \), represents all possible errors. So, there are \( N - 1 \) different possible errors. Let \( E_i \) be the set of errors detected by pattern \( p_i \), and \( \|E_i\| \) be the magnitude of \( E_i \). Then, \( \|E_i\| \) is defined as

\[
\|E_i\| = \sum_{k=1}^{N} f_k(p_i) \oplus f_j(p_i)
\]

To easily compute the number of detected errors with \( l \) patterns, an \( \Omega \) function is introduced. Let \( \Omega(M, l) \) be the number of errors detected by \( l \) patterns among
Therefore, from (I), the simulation coverage is defined
more complex. Let the number of outputs be
errors detected at the output
ber of errors detected by pattern
inputs, respectively. Also, let
ol,02,.
2.2 Circuits with Multiple Simple Outputs
If a circuit has more than one output, it is
If we assume that the total number of errors is the
The simulation coverage is given by
The simulation coverage is defined as
2.3 Circuits with a Single Complex Output
Without loss of generality, assume that the out-
xinputs related to the primary output
inputs related to the output
inputs. Then, there are
Let the number of outputs be
Let the number of errors in the circuit is given by
Therefore, the total number of errors in the circuit
Let the first pattern be
Let the second pattern be
Then, after one pattern is applied, when the second
Since the number of errors detected at the primary
The simulation coverage is given by
The simulation coverage is defined as
2.4 Circuits with Multiple Complex Outputs
Let the number of outputs be
Since the number of errors detected at the primary
The simulation coverage is given by
This is the general simulation coverage equation for
2.5 Discussions
It is obvious that, to get 100% simulation cov-
ne possible patterns are required in the simple
output cases, ignoring possible redundancy. However,
theoretical analysis has shown that the number of pat-
not proportional to the simulation coverage.

The comparison between the theoretical simulation based on the number of design errors provides more coverage and SCP of a have thought. This analysis indicates that a metric coverage, represented by simulation, is higher than we rate. The theoretical analysis leads to the conclusion for large circuits, even though this theoretical analysis is to choose a proper set of simulation patterns. This assumption of uniform distribution of patterns is inaccurate coverage than SCP, and the conventional as we consider timing, it is more complex. This gives rise in the general case, deriving that an ideal way to achieve efficient design verification Figure 1. Therefore, we can conclude that the actual analysis[3]. (It has been shown that a single fault model covers most multiple faults in a simulation environment. It is suggested that this is, also, likely for design errors.) The above assumption provides a reasonable approximation, since most design errors are related to misuse of basic primitives. If we use single errors, then only a small portion of all errors need be considered.

Consider the following error. Initially, a gate has two inputs. However, because of a design error, the gate includes the signal as an input, where s is not intended to be the input of g.

Consider the case where the number of inputs is decreased. For AND or OR gates, a change in the number of inputs can be modeled with a signal-like-source model. For NAND or OR gates, a change in the number of inputs can be modeled with a signal-like-ground model. For XOR gates, the error is modeled with a signal-like-ground model, if the number of inputs in the error-free gate is an odd number. If it is an even number, the error is modeled with a signal-like-source model.

Consider the case where the number of inputs is decreased. For AND or OR gates, a change in the number of inputs can be modeled with a signal-like-source model. For NAND or OR gates, a change in the number of inputs can be modeled with a signal-like-ground model. For XOR gates, the error is modeled with a signal-like-ground model, if the number of inputs in the error-free gate is an odd number. If it is an even number, the error is modeled with a signal-like-source model.

Assumption 1 Single Error Assumption: A single error is a design error which can be modeled by a signal error or a gate error. In the circuits to be considered, single errors are used.

This is analogous to the classical single fault assumption that has been extensively used for fault analysis[3]. (It has been shown that a single fault model covers most multiple faults in a simulation environment. It is suggested that this is, also, likely for design errors.) The above assumption provides a reasonable approximation, since most design errors are related to misuse of basic primitives. If we use single errors, then only a small portion of all errors need be considered.

Assumption 2 Local Error Assumption: In a circuit, only design errors which can be modeled using local errors are considered.

This assumption is reasonable since many design errors happen in a neighborhood.
Assumption 3 Include Error Assumption: In a circuit, design errors which can be modeled using include errors are considered.

The number of errors modeled using these three assumptions is

\[ 4k + 3(S + k) + 12(S + k) + k(S + k) - EQ, \]

where \( S = \sum_{i=1}^{k} s_i \), \( k \) is the number of gates in a circuit, \( s_i \) is the number of inputs of gate \( i \), and \( EQ \) represents the number of equivalent errors. The first four terms represent the total number of gate errors, and \( EQ \) represents the number of equivalent errors. The first four terms represent the total number of gate errors, signal errors, local errors, and include errors, respectively.

In a circuit whose behavior is represented by a function \( f \), the set of simulation patterns which detects an error \( \alpha \) is defined as \( S_\alpha = f \oplus f_\alpha \). The set of simulation patterns which detects an error \( \beta \) is defined as \( S_\beta = f \oplus f_\beta \). The set of simulation patterns which distinguishes \( \alpha \) and \( \beta \) is defined as \( f_\alpha \oplus f_\beta \). If \( f_\alpha = f_\beta \), there are no simulation patterns to distinguish \( \alpha \) and \( \beta \). Then, such errors are said to be equivalent.

Definition 6 Errors which are equivalent, can be represented by one error, since they are not distinguishable. Therefore, these errors can be collapsed. Error collapsing is the process of representing equivalent errors as one single error, for simulation purposes.

4 Error Simulation

To determine the coverage of these models, simulation is executed to analyze the effectiveness of simulation patterns, by providing a simulation coverage metric.

Definition 7 The simulation coverage metric (SCM) is defined as the number of detected errors using the given simulation patterns, divided by all possible errors that are being modeled.

\[ SCM = \frac{\text{the number of detected errors}}{\text{the number of modeled errors}} \]

As mentioned earlier, the SCP, where the simulation coverage is proportional to the number of patterns, does not provide a good measure, since it is not true that each pattern can detect the same number of design errors. Since the above definition considers the number of design errors to be modeled, it provides more accurate insight into the measure of design verification.

Definition 8 Error simulation is defined as the process of simulating a circuit to analyze its operation under various design error conditions, and to evaluate the effectiveness of simulation patterns in terms of error detection.

In error simulation, the inputs are a circuit description and a set of simulation patterns. The circuit description is translated into internal tables and, according to a circuit topology, the error list is generated. All errors are levelized and stored in the list according to the level of error sites. The error list includes a position and a type of error. When the error list is generated, error collapsing is considered to minimize the size of the error list. Then, using the given simulation patterns, simulation is executed. Finally, error simulation provides the simulation coverage. The algorithm for error simulation is shown in Figure 2.

<table>
<thead>
<tr>
<th>Algorithm of Error Simulation</th>
</tr>
</thead>
<tbody>
<tr>
<td>1. Choose 32 patterns which are not simulated.</td>
</tr>
<tr>
<td>2. Simulate a circuit without any error.</td>
</tr>
<tr>
<td>3. Record the values.</td>
</tr>
<tr>
<td>4. For all errors which are not detected,</td>
</tr>
<tr>
<td>Select an error which has lowest level in the list.</td>
</tr>
<tr>
<td>5. Execute an error model replacement.</td>
</tr>
<tr>
<td>6. Simulate a circuit.</td>
</tr>
<tr>
<td>7. If the value is the same as the value in the previous error-free simulation,</td>
</tr>
<tr>
<td>After highest error site evaluation,</td>
</tr>
<tr>
<td>Stop simulation for this error.</td>
</tr>
<tr>
<td>9. Detect an error at primary outputs.</td>
</tr>
<tr>
<td>10. If this error is detected,</td>
</tr>
<tr>
<td>Delete the error from the error list.</td>
</tr>
<tr>
<td>11. Count the number of detected errors.</td>
</tr>
</tbody>
</table>

Figure 2: The Algorithm of Error Simulation

There is no implication that fault simulation can be generally used for any reasonable measure of design validation. However, it is interesting to note that some of the design errors, that have been introduced, have a corollary to fault models. On the other hand, many of the other errors do not have a corollary. Therefore, it is obvious that fault models can be mapped into a subset of design error models. As such, it is suggested that this newly introduced metric for design validation can also be used as a metric for fault analysis, using a measure of the subset of fault models. Therefore, fault assessment may become a by-product of design validation, with all the inherent efficiencies that could result in this approach. It is also, possible to use statistical sampling techniques for the generated design errors to arrive at a quick, first pass, assessment of design coverage.

5 Results

To generate the simulation coverage, practically, error simulation was developed using 3 logic values (0, 1, and X). It can handle the combinational and sequential nature of errors since a design error can change a combinational circuit into a sequential circuit or vice versa. The results of error simulation are shown in Table 1. For the measure of simulation coverage 1000 random patterns were used for the benchmark circuits[4]. For each circuit, the number of modeled errors, simulation coverage (SCM), and error simulation time, are represented. SCM shows high coverages,
which means that many design errors can be detected using these patterns.

<table>
<thead>
<tr>
<th>Circuit</th>
<th>Modeled Errors</th>
<th>SCM [%]</th>
<th>SCM Time [sec]</th>
</tr>
</thead>
<tbody>
<tr>
<td>c432</td>
<td>6528</td>
<td>99.41</td>
<td>2.39 x 10^4</td>
</tr>
<tr>
<td>c499</td>
<td>9560</td>
<td>98.17</td>
<td>8.18 x 10^4</td>
</tr>
<tr>
<td>c880</td>
<td>22723</td>
<td>96.27</td>
<td>1.90 x 10^4</td>
</tr>
<tr>
<td>c1908</td>
<td>23519</td>
<td>94.53</td>
<td>6.10 x 10^4</td>
</tr>
<tr>
<td>c2670</td>
<td>171309</td>
<td>85.34</td>
<td>7.33 x 10^4</td>
</tr>
<tr>
<td>c3540</td>
<td>67066</td>
<td>90.69</td>
<td>1.29 x 10^4</td>
</tr>
<tr>
<td>c5515</td>
<td>284456</td>
<td>97.41</td>
<td>4.78 x 10^4</td>
</tr>
<tr>
<td>c6288</td>
<td>110224</td>
<td>97.50</td>
<td>3.67 x 10^4</td>
</tr>
<tr>
<td>c7552</td>
<td>474142</td>
<td>91.94</td>
<td>1.51 x 10^4</td>
</tr>
</tbody>
</table>

Table 1: Error Simulation Results

To compare the SCM and theoretical coverage, the 74ls154 circuit[5] was used as an example. Figure 3 shows that the results of error simulation are close to the results predicted by the theoretical analysis. As we expected, the simulation coverage curve rapidly increases in the case of applying a small number of simulation patterns, then, it slowly increases after that.

Consider the excitation of an include error. It is more difficult to excite than the other errors, since it may generate a reconvergent fanout in the circuit. If there is no new reconvergent fanout due to an include error, it is the same as local error excitation. However, if there is a new reconvergent fanout due to an include error, to excite this include error, a module and adjacent signals are introduced. Let G be the gate, S be the signal, the error be the G-include-S error. Assume that the signal S is an input signal of gate H. The point of reconvergence might be G, H or another gate(l). The module is defined as a set of gates which contain all gates from gate G or H to the point of reconvergence, according to the level of gate G and H. The adjacent signals are defined as signals which are inputs of the gates in the module, except signal S. To excite an include error, a module must be excited. Regardless of the levels of the gate G and H, the path from the signal through the gate G to the complex. To excite the error effect, the same new logic values are introduced. The value D represents a 1 in an error free circuit and a 0 in an error circuit. The value D represents a 0 in an error free circuit and a 1 in an error circuit.

Firstly, consider signal error models. To excite a signal-like-ground error, the signal must be set to 1. For a signal-like-source error, the signal must have a value 0. However, in order to excite a signal-like-inverter error, the signal can be set to 0 or 1. Notice that to excite a certain error, a signal can have more than one logic value.

Consider gate error models when the original gate is an AND, as shown in Figure 4. To excite a gate-like-and error, the two inputs can have 00, 01, 10, and 11. For patterns 00, 10, and 01, the error free value is 1 and error value is 0. The excited value is D. For a pattern 11, the good value is 0 and error value is 1. The excited value is D.

For the excitation of local errors, both signal and gate errors must be excited at the same time. For example, excitation of A-like-source error and C-like-or error is shown in Figure 5.

6 AEPG

Considering the fact that each pattern can detect a different number of design errors, the derivation of efficient simulation patterns is important. Therefore, the selection of error simulation patterns is very important. Also, for a specific design error, an error simulation pattern to detect the design error, is required. To accomplish this, automatic error pattern generation(AEPG), which is analogous to automatic test pattern generation[6] is introduced. 

![Number of Simulation Patterns](image_url)

Figure 3: Simulation Coverage of 74ls154

![Figure 4: Excitation of Gate Errors](image_url)

Figure 4: Excitation of Gate Errors

![Figure 5: Excitation of Local Error](image_url)

Figure 5: Excitation of Local Error

Consider the excitation of an include error. It is more difficult to excite than the other errors, since it may generate a reconvergent fanout in the circuit. If there is no new reconvergent fanout due to an include error, it is the same as local error excitation. However, if there is a new reconvergent fanout due to an include error, to excite this include error, a module and adjacent signals are introduced. Let G be the gate, S be the signal, the error be the G-include-S error. Assume that the signal S is an input signal of gate H. The point of reconvergence might be G, H or another gate(l). The module is defined as a set of gates which contain all gates from gate G or H to the point of reconvergence, according to the level of gate G and H. The adjacent signals are defined as signals which are inputs of the gates in the module, except signal S. To excite an include error, a module must be excited. Regardless of the levels of the gate G and H, the path from the signal through the gate G to the
point of reconvergence must be a sensitized path. In order to this, all adjacent signals must be set to have

corrected signals Module

Figure 6: Excitation of Include Errors

Figure 7: The Algorithm of Error Pattern Generation

make an error list
while a certain coverage is derived do
    generate 32 random patterns
    execute error simulation using these patterns
    simulate a circuit without any error
    record the values
    for all errors which are not detected
        select an error which has lowest level
        execute an error model replacement
        simulate a circuit
        if the value is the same as the value in
        the previous error-free simulation
            stop simulation for this error
        if this error is detected
            mark these errors "detected"
            store the patterns
    end
end
for all undetected errors in the list
    choose an undetected error from the list
    if the error is an include error
        select a module and adjacent signals
        for all excitations of the error
            excite an error according to an error type
            if there is a X path
                backtrack to a PI and set a value
            else if there is an untried PI
                change the PI value
            propagate an error effect to primary outputs
            if an error pattern is generated
                mark the error "detected"
                store the generated patterns
                break
            if an error pattern is not generated
                delete this error from the list
end

7 Conclusion

When simulation is used for design verification, complete verification is not practical since applying exhaustive simulation patterns is usually impossible. Since, in most cases, a subset of possible simulation patterns is applied, a measure of completeness is not available. Then, the immediate question is, how much of the design has been verified? As an answer to this question, a new measure of design verification coverage based on the concepts of design errors, was introduced as

$$SCM = \frac{\text{the number of detected errors}}{\text{the number of modeled errors}}$$

The theoretical basis of the new measure was discussed. To apply this in a practical situation, a number of new techniques were introduced, including: design error modeling based on reasonableness assumptions, error simulation, automatic error pattern generation. The error simulation results verified the accuracy of this approach. To derive efficient error simulation patterns, automatic error simulation pattern generation was developed.

References