MODELING AND SIMULATION
VOLUME 23

Part 5
Networks, Communications, Biomedical, General

Proceedings of the Twenty-Third Annual Pittsburgh Conference on Modeling and Simulation

COMPARATIVE ANALYSIS OF ERROR SIMULATION ALGORITHMS FOR DESIGN VALIDATION

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

ABSTRACT

Error simulation is a new process that involves the simulation of a circuit to analyze its operation under various design error conditions. This results in a measure of simulation coverage for the design being analyzed. Since error simulation is similar to fault simulation, in some ways, three design error simulation algorithms have been implemented, based on the known fault simulation techniques. For these three algorithms, comparative analysis was performed, based on the number of modeled errors, simulation in a timing environment, circuit structure change, multiple errors, etc.

INTRODUCTION

Simulation[1] is an essential and commonly used method for verifying the design of digital systems. Since simulating the system becomes very costly, as systems increase in size and complexity, only a subset of the possible simulation patterns are used for verification, in most cases. Therefore, for the subset of simulation patterns, a measure of coverage is not available. Hence, it is necessary to derive a measure of simulation coverage for a given set of simulation patterns.

Error simulation[2] is the process of simulating a circuit to analyze its operation under various design error conditions and to provide a simulation coverage metric; namely,

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

This metric can have a profound impact on the entire design validation process, since it provides more realistic results than those which are presently available. Theoretical analysis[2] indicates that a metric based on the number of design errors provides a more accurate coverage measure than those which are based on the number of simulation patterns. In some way, error simulation is similar to fault simulation[3–5], which is defined as a process for measuring the effectiveness of a set of test patterns. The main difference is that instead of dealing with faults, design errors are generated and analyzed.

In the next section, three fault simulation algorithms are briefly explained, followed by an analysis of the relationship between error simulation and fault simulation; based on stuck-at fault models and design error models. Then, three error simulation algorithms are implemented based on the known fault simulation techniques. These include parallel[3], concurrent[4] and PPSFP[5] algorithms. Since error simulation is similar to fault simulation except that simulation is executed under error conditions, a general fault simulation algorithm can be used as an error simulation algorithm,

with appropriate modification. To achieve high performance error simulation, these algorithms are compared using the parameters such as the number of modeled errors, simulation in a timing environment, circuit structure change, multiple errors, etc. Finally, the results and conclusions are provided.

FAULT SIMULATION AND ERROR SIMULATION

It is impossible to consider all possible design errors, for large circuits. This gives rise to questions of how can design errors, for large circuits, be modeled to consider those that are the most likely to occur? This design error modeling is very important since it directly affects the accuracy and efficiency of error simulation. A design error model is a logical model of an actual design error which shows a different behavior from the desired behavior. Therefore, if the output is the same, although there may have been a change of internal structure, this change is not regarded as an error. Since the modeling of all possible errors, is too complex and the number of errors is too large, only a subset of all possible errors can be considered. Therefore, several design error models have been used in previous studies, including; signal errors, gate errors, local errors, and include errors[2].

Let s be a signal and g a gate. A signal error is a pin signal error which includes an s-like-source error, s-like-ground error, and s-like-inverter error. A gate error is an error in the gate itself which includes a g-like-and error, g-like-nand error, g-like-or error, g-like-nor error, and g-like-xor error, A local error is a multiple error which uses only one gate error and one signal error where the signal is connected to the gate. A g-include-s error is an error in which the gate g includes the signal s as an input, where s is not intended to be the input of g.

Even though an error simulation algorithm can be derived by modifying fault simulation algorithms, this does not imply 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 in fault models. For example, a signal-like-source model is the same as a stuck-at-1 model and a signal-like-ground model is the same as a stuck-at-0 model. Therefore, if only these two errors are in our domain set, error simulation becomes fault simulation. 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 shown in Figure 1. As such, it is suggested that

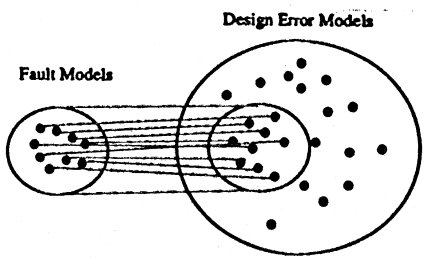

Figure 1: Mapping of Fault Models and Error Models

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 from this approach.

The main difference between error simulation and fault simulation is that error simulation handles design errors and provides simulation coverage, instead of faults and fault coverage. When considering implementation of error simulation, the important things to consider are as follows.

Firstly, error simulation should consider the timing environment. Since there are many design errors which are related to timing, and since some design errors may convert combinational circuits into sequential circuits, the consideration of timing is indispensable. The second thing to be considered is that certain design errors change the structure of the circuit. In fault simulation, with stuck-at fault models, the circuit structure never changes. Thirdly, there exist design errors which have more than one error site. These are similar to multiple faults which have multiple error sites. Finally, there are more modeled design errors than modeled faults.

ERROR SIMULATION ALGORITHMS

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 the circuit topology, an error list is generated. The structure of an error list is shown in Figure 2. All errors are levelized and

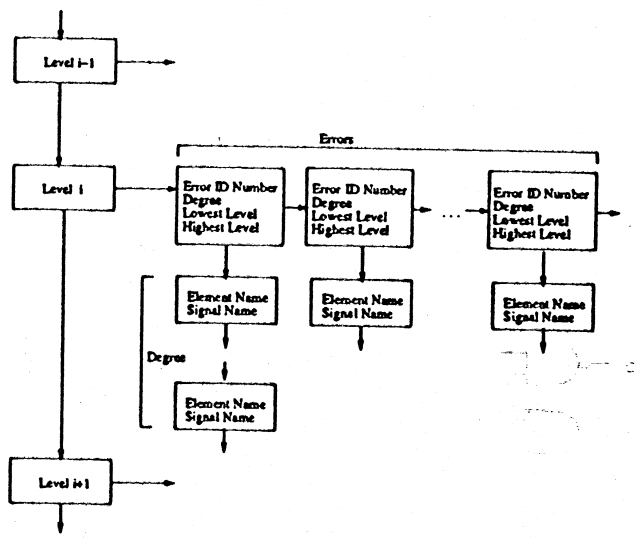

Figure 2: Error List Structure

stored in the list according to the level of error sites. For multiple errors, there is more than one error site. In this case, errors are stored according to the lowest level error site. The items in each entry are an error identification number, a pointer to the next error, a degree, the lowest level, the highest level and a pointer to the error site. An error identification number is used to identify an error. The number of the attached error sites is the same as the degree. In each attached error site, the name of an element and the name of the signal are specified.

When the error list is generated, error collapsing is considered to minimize the size of the error list. In a circuit whose behavior is represented by a function f, the set of simulation patterns which detects an error α is defined as $S_\alpha = f \oplus f_\alpha$. The set of simulation patterns which detects an error β is defined as $S_\beta = f \oplus f_\beta$. The set of simulation patterns which distinguishes α and β is defined

as $f_\alpha \oplus f_\beta$. If $f_\alpha = f_\beta$, there are no simulation patterns to distinguish α and β. Then, such errors are said to be *equivalent*. and can be represented by one error. *Error collapsing* is the process of representing equivalent errors as one single error, for simulation purposes. For example, consider a two input AND gate, whose output is Z, the name of the gate is G, and inputs are A and B, respectively. G-like-NAND and Z-like-NOT errors represent the same error. This is similar to fault collapsing in fault simulation.

Then, using the given simulation patterns, simulation is executed. For this process, three algorithms namely, Parallel, Concurrent, and PPSEP, were implemented for comparison. (In the next three sections, these algorithms are explained in detail.) Finally, error simulation provides the simulation coverage metric.

PARALLEL ERROR SIMULATION

Consider a parallel simulation environment. For 32 bit machines, 31 design errors and one good machine are assigned to each word for one pattern. Therefore, in parallel error simulation, 31 design errors are considered simultaneously. One copy represents the error-free circuit and is known as the good machine, and each of the other copies represents the circuit with one design error present.

For errors which have multiple error sites, error insertion can be executed at each error site, to model the error. After insertion, evaluation is performed. Until all 31 errors are considered, simulation is continued, although evaluation results may be the same as previous ones.

In parallel fault simulation, the circuit structures of the 31 fault machines and one good machines are the same, therefore parallel simulation can be executed. However, there is a problem with design error simulation since certain design errors can change the circuit structure. An example of this problem is shown in Figure 3. When the good circuit is an AND gate, G, and the errors are

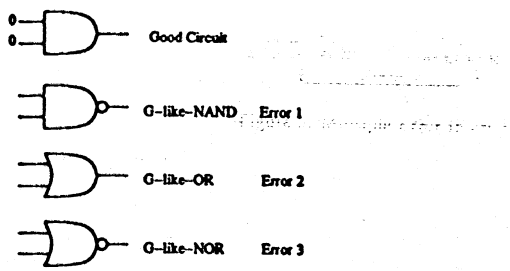

Figure 3: Problem of Parallel Error Simulation

G-like-NAND, G-like-OR, and G-like-NOR. Since the error circuits and the good circuit are different from one another, error insertion becomes difficult.

One way to solve this problem is error grouping. In the preprocessing phase, while generating the error list, the design errors are classified into several sets, which include the errors which have the same circuit structure. From the errors, 31 are simulated in parallel. However, error grouping based on the same circuit structure is very difficult, and the advantage of parallelism can disappear since there are usually a large number of sets.

Another solution is to generate error models which can accept different circuit structures. However, this is almost impossible since there are so many possibilities.

Another approach is as follows. Instead of insertion before evaluation, at each error site, evaluation is performed first, then an error effect is inserted by evaluating the primitive. In other words, the primitive evaluation at an error site is performed first. Then, instead of error insertion

according to the different structure, the error effect is computed. This effect is inserted into the evaluation result. An example of this approach is shown in Figure 4. The bit representations for

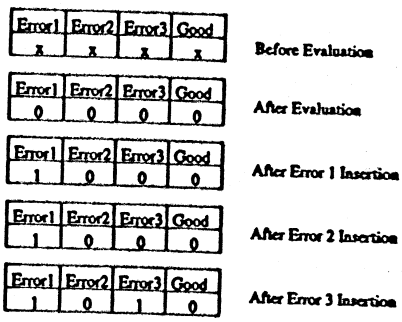

Figure 4: Solution of Parallel Error Simulation

each step are shown. First, the good circuit is evaluated, then each error effect is inserted. For Error 1 in the example, the effect of Error 1 is computed, then this effect is inserted.

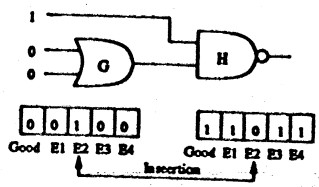

Figure 5: Multiple Error Insertion

Consider a multiple error insertion. During the simulation, before insertion, the error sites of 31 errors are compared to the current evaluation site. When a new event is evaluated, the site of this event is compared to that of 31 errors. To insert a multiple error, it requires a little more comparison, however, this does not require significant overhead. Consider the example shown in Figure 5. Suppose that Error 2 is a multiple error(G-like-NOR and H-like-OR) and it is related to error sites G and H. For the insertion of this error, after insertion at G, another insertion is executed at H.

The parallel error simulation algorithms is shown in Figure 6. First, the scheduler is initialized and all signals in the circuit are set to initialization values. For each pattern, all errors are considered. From the error list, 31 errors are selected, and the good circuit and 31 error circuits are simulated simultaneously. During the simulation, each signal is compared to the error sites of 31 errors. If a signal is the error site of an error, the error is inserted. At the primary outputs, if the error circuit value is different from the good circuit value, the error is detected and dropped from the error list. This procedure is continued until all errors are detected or all patterns are simulated.

The advantage of this approach is that the implementation is straightforward and the memory requirements are reasonable. One limitation of this approach is that it requires complex evaluation routines for functional primitives. However, this is no longer a problem due to automatic model generation[6-8]. Another limitation is that it may be a slower, compared to some other algorithms,

```
generate an error list
initialize a circuit and a scheduler
insert all simulation patterns as new events
for each pattern
   choose 31 errors from the error list
   retrieve a new event from the scheduler
   evaluate the event
   if an error site
     for each error related to this site
       execute an error model replacement
       insert an error effect
     end
   if the evaluation result is different from the old one
     or at least one unconsidered error site
     update the given delay
     insert this new event to the scheduler
   if the result of the evaluation is the same
     and there is no unconsidered error site
     delete this event
   if primary outputs
     detect errors
     delete detected errors from the error list
end
```

Figure 6: Parallel Error Simulation Algorithm

for large circuits. The main reason for this is that a new event is generated if only one bit in the word is different. Therefore it cannot take full advantage of selective trace and error dropping. In the case that there are many multiple errors, i.e. errors which have more than one error site, parallel error simulation is probably preferable. However, in the cases of numerous structural changes in a circuit, parallel error simulation requires additional evaluations.

CONCURRENT ERROR SIMULATION

The basic problem with the parallel algorithm is the need to repeat the evaluation, since the number of errors considered at any time is limited by the number of bits. Concurrent simulation[4] is a process that does not have this problem.

Considering a concurrent simulation environment, the following procedure can be used. In concurrent fault simulation, a fault list associated with each primitive is generated. In concurrent error simulation, an error site list associated with each primitive should be generated, instead of an error list. The reason for this is that there are errors which have multiple error sites. Therefore, there must be pointers to link error sites which represent a certain error. An example of error lists is shown in Figure 7. In this example, Error 3 is a multiple error and the error sites are in G and H. Then, Error 3 is shown in both error site lists of G and H and there is a link between the two sites.

After the effects of all error sites of an error are considered, the comparison between this error effect and the evaluation results of the good primitive are executed. The reason for this is that there is a possibility of not detecting the error if the error effect is not propagated; since the comparison is the same before all error sites are considered. An example is shown in Figure 8. In this example, Error 2 is different from the good value of G and Error 1 is the same as the good value of G. Hence, Error 2 is propagated and Error 1 is dropped. However, although Error 3 is the same as the good value of G, Error 3 is still considered when H is handled. Therefore, this structure is effective for

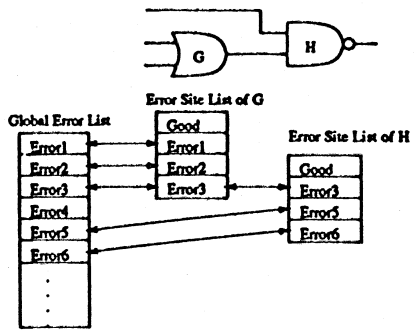

Figure 7: Error List of Concurrent Error Simulation

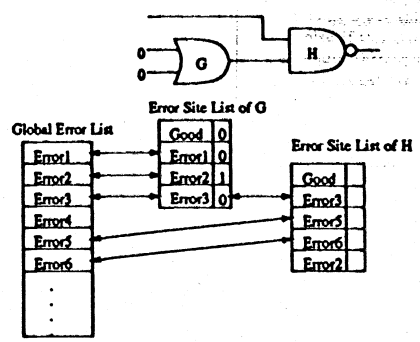

Figure 8: Error Propagation

this approach. However, the generation of error site lists is complex and it requires more memory space. Also, when dropping the detected errors, it requires more time. To drop one error many error site lists are considered, since there are many error sites for one error and these are connected using pointers.

The major advantage of concurrent error simulation, over parallel error simulation for small circuits, is its speed; due to the fact that it considered all design errors in one pass, while parallel simulation requires $\lceil E/(B-1) \rceil$ passes(where E represents the total number of errors in a circuit and B represents the number of bits in a host machine). However, considering large circuits, this is no longer true, since concurrent simulation requires more than one pass due to memory space limitations. Therefore, as circuit sizes become larger, the proportion of the number of errors to be considered in a pass is less, while in parallel simulation, the number of errors to be considered in a pass is a fixed number $B-1$. This is the major factor that degrades the performance of concurrent error simulation. For extremely large circuits, concurrent simulation might handle a small number of errors(maybe less than $B-1$) in a single pass.

The concurrent error simulation algorithm is shown in Figure 9. At the beginning of the

```
generate an error list
execute error grouping
for all error groups
  initialize a circuit and a scheduler
  insert all simulation patterns as new events
  for each pattern
    make an error site list associated with each primitive
    retrieve a new event from the scheduler
    simulate a primitive
    for each error in primitive error site list
      execute an error model replacement
      simulate an error effect
      if the evaluation result is different from
        that of the good primitive
        propagate this error
      else if this error is multiple error
        and there is unconsidered error site
        propagate this error
    end
    update the delay and insert new event
    if primary outputs,
      detect errors
      delete detected errors from the error list
        and error site lists
  end
end
```

Figure 9: Concurrent Error Simulation Algorithm

algorithm, the scheduler is initialized and all signals in the circuit are set to initialization values. According to the circuit size, the appropriate number of errors are grouped dynamically. For each group, all patterns are considered. For each pattern, the error site lists associated with each primitive are generated. During the simulation, these error site lists are updated if error effects are propagated. At the primary outputs, errors can be detected by searching the error site lists of the primary outputs. This procedure is continued until all errors are detected or all patterns are simulated.

Concurrent simulation is known to be faster than parallel algorithm, for some circuits. The

main disadvantages are that it requires more memory space in order to contain the error lists, and it may require multiple passes. In the case that there are many multiple errors, i.e. errors which have more than one error site, concurrent error simulation requires more overhead to deal with the error site lists.

PARALLEL PATTERN SINGLE ERROR PROPAGATION

The algorithm of Parallel Pattern Single Error Propagation(PPSEP) error simulation is based on the PPSFP algorithm[9] shown in Figure 10. The first step is to simulate the circuit without

```
levelize a circuit
generate an error list
for  all patterns
     choose 32 patterns which are not simulated
     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 according to the level
               if the value is the same as the value in
                         the previous error-free simulation
                         after highest error site evaluation
                    stop simulation for this error
          detect an error at primary outputs
          if this error is detected
               delete the error from the error list
               count the number of detected errors
     end
end
```

Figure 10: PPSEP Algorithm

any design errors. To do this, 32 patterns are selected and simulated in parallel. The results of the evaluation are stored for comparison. Then, an error from the error list is selected and an error model replacement is executed. While simulating a circuit according to the level, if it encounters an error site which is specified in the error list, the error model is evaluated and, then, propagation of the error effect is checked. If the value of the evaluation is the same as the value of the error-free simulation, the error effect is not propagated, indicating that this error cannot be detected. Therefore, simulation stops and another error is selected and simulation restarted. However, if the target error is a multiple error, then the simulation is continued until the last error site is considered. If the error effect is propagated to any of the primary outputs, then, the error is detected and removed from the error list. Then, the number of detected errors are recorded. For a new error, the previous procedure is continued until all errors are detected or all of the given patterns are simulated.

The structure of a circuit can be changed at the error site before simulation, using error model replacement. Error model replacement is a procedure that changes the structure of a circuit according to an error. This is done after a target error is selected. According to the error, the structure of a circuit will be different. Therefore, the strategy of considering several errors simultaneously doesn't work well. Consequently, for parallel patterns, only one error is considered at a time. After error free simulation, the circuit is changed by considering a target error. Then simulation is executed.

The reason for simulating the lowest level error first is to derive efficiency in error simulation. All errors are levelized along with a circuit, in the preprocessing phase. For multiple errors, the

lowest levels are considered. In error simulation, all first level errors are considered, then the second level errors are considered. After consideration of the first level errors, the error free simulation values are stored for further usage in the next level error simulation. Therefore, after all first level errors are considered, the first level in the circuit, is no longer simulated. Using this approach, the computation time is reduced.

This is a simple and efficient solution for error simulation. The main reason for high performance of this algorithm is parallel pattern evaluation and single design error propagation, to decide if a design error is detectable. The main advantage of this implementation is that, since it considers one error at a time, it is possible to handle the change of the circuit structure and it can easily handle multiple error sites. However, it fails to consider the timing environment, since it does not work like event driven simulation, and is limited to devices which are combinational. Therefore, to handle timing errors, some modification is required. Design errors which can produce races or hazards, which can make a circuit oscillate, and which can change a combinational circuit into a sequential one, require additional considerations.

RESULTS

In the case that there are many multiple errors, i.e. errors which have more than one error site, parallel error simulation is probably preferable since concurrent error simulation requires overhead to deal with the large error site lists. However, in the case of numerous structural changes in a circuit, concurrent error simulation is probably desirable, since parallel error simulation requires additional evaluations. However, in general, it is difficult to figure out which approach is better since the performance measures are not available, based on the number of multiple errors and structure changes. To figure out precise performance comparisons, implementations of the two approaches, with careful considerations of the above facts and many experiments with various circuits, are required.

To provide a simulation coverage metric(SCM), error simulation was implemented. In error simulation, 3 logic values(0, 1, and X) were used. The results of PPSEP error simulation are shown in Table 1. The TTL circuits[10] and benchmark circuits[11] are used for the measure of simulation

Circuit	Modeled Errors	SCM [%]	PPSEP Time [sec]
74ls145	481	100.00	1.13×10^{0}
74ls154	948	100.00	3.73×10^{0}
74ls181	2152	96.56	3.41×10^{1}
74ls251	498	100.00	2.38×10^{0}
74ls261	1118	100.00	3.26×10^{0}
74ls283	875	100.00	5.53×10^{0}
c432	6532	99.41	2.39×10^{1}
c499	9192	98.17	8.18×10^{1}
c880	22083	98.27	1.90×10^{2}
c1355	25816	94.53	6.10×10^{2}
c1908	22460	94.53	6.10×10^{2}
c2670	169750	83.34	7.33×10^{3}
c3540	64840	90.69	1.29×10^{3}
c5315	280964	97.41	4.78×10^{3}
c6288	105456	97.50	3.67×10^{3}
c7552	469408	91.94	1.51×10^{4}

Table 1: PPSEP Error Simulation Results

coverage. For each circuit, the number of modeled errors, simulation coverage(SCM), and error simulation time are represented. Modeled errors include gate errors, signal errors, local errors, and include errors related to primary inputs. SCM shows high coverages, which means that many design

errors can be detected using the patterns. Error simulation time is longer than fault simulation time since it handles structural changes of the circuit and errors which have more than one error site.

Delay error simulation time is longer than zero delay error simulation time because it requires scheduling mechanisms. The parallel and concurrent error simulation results are shown in Table 2. Concurrent error simulation is faster than parallel error simulation for these circuits. However, the number of passes of concurrent error simulation increases more rapidly than that of parallel error simulation. However, for larger circuits, concurrent simulation might not be faster than parallel error simulation since it requires many passes, due to limited memory space.

Circuit	Parallel		Concurrent	
	Time [sec/pattern]	Passes	Time [sec/pattern]	Passes
74ls145	0.06	16	0.06	1
74ls154	0.21	31	0.27	1
74ls181	0.88	70	0.96	3
74ls251	0.13	17	0.10	1
74ls261	0.16	37	0.23	1
74ls283	0.17	29	0.23	1
c432	7.66	211	1.96	2
c499	47.83	297	3.21	2
c880	149.51	713	5.90	8
c1355	152.82	833	14.17	26
c1908	204.53	725	22.95	23
c2670	706.33	5476	105.43	340
c3540	360.32	2092	81.65	130

Table 2: Parallel and Concurrent Error Simulation Results

CONCLUSIONS

Error simulation is the process of simulating a circuit to analyze its operation under various design error conditions. A zero delay error simulation algorithm, based on a parallel pattern environment, was developed. To consider errors related to timing, possible modifications of the error simulation algorithm were investigated, by considering parallel and concurrent approaches. According to the number of multiple errors and structural changes that occur, both approaches suffer from limitations. Parallel error simulation requires additional evaluations when considering circuit structure changes and concurrent error simulation requires complex error site lists to deal with multiple errors.

The best scenario is as follows. For errors with non-timing problems, the PPSEP algorithm is used. For errors with timings, parallel and concurrent algorithms are both used. For large circuits, concurrent error simulation is better than parallel error simulation. However, in extremely large circuits, parallel error simulation might be better.

REFERENCES

1. M. A. Breuer and A. D. Friedman, Diagnosis and Reliable Design of Digital Systems, 1976.

2. S. Kang and S. Szygenda, Metrics for Design Validation Based on a New Concepts of Design Error, submitted for publication 1991.

3. E. Thompson and S. Szygenda, Digital Logic Simulation in a Time-Based, Table-Driven Environment Part2. Parallel Fault Simulation, Computer, March 1975, pp. 38-49.

4. E. Ulrich and T. Baker, The Concurrent Simulation of Nearly Identical Digital Networks, DAC, June 1973, pp. 145-150.

5. W. Daehn and M. Geilert, Fast Fault Simulation for Combinational Circuits by Compiler Driven Single Fault Propagation, ITC, 1987, pp. 286-292.

6. C. Han, S. Kang, and S. Szygenda, AFMG: Automatic Functional Model Generation System for Digital Logic Simulation, ASIC Conference, September 1991.

7. H. Yang, and S. Szygenda, A Domain-Specific Automatic Programming System for the Element Routine Generation, Summer Simulation Conference, 1991.

8. C. Chuang, and S. Szygenda, The Automatic Element Routine Generator: An Automatic Programming Tool for Functional Simulator Design, 25th Simulation Symposium, 1992.

9. J. Waicukauski, E. Eichelberger, and D. Folenza, A Stastical Calculation of Fault Detection Probabilities by Fast Fault Simulation, ITC, 1985, pp. 779-784.

10. The TTL Data Book, Texas Instruments, 1981.

11. F. Brglez, and H. Fujiwara, A Neutral Netlist of 10 Combinational Benchmark Circuits and a Target Translator in FORTRAN, ISCAS, 1985, pp. 695-698.