Highly Efficient Fault Simulation Exploiting Hierarchy in Circuit Description

Bernhard H. Seiss
Institute of Electronic Design Automation, Department of Electrical Engineering
Technical University of Munich, D-8000 Munich 2, Germany

Hannes C. Wittmann

Abstract
A highly efficient fault simulation approach is presented taking advantage of the hierarchy which can be found in most of today's circuit descriptions. New, evident graphs are introduced to represent the structure of the hierarchy. Numerous experimental results for industrial circuits with up to 90,000 gates demonstrate the efficiency of the approach with respect to acceleration and reduction of the memory requirements as compared to an efficient gate level fault simulator.

Keywords
Fault Simulation, Hierarchical Circuit Description, Functional Description, Functional Block

1 Introduction
The steady increase in circuit size causes more and more designers to describe their circuits in a hierarchical manner. In order to get the design clear they describe the circuit at the highest level of hierarchy as a connection of few functional blocks. These functional blocks are described at a lower level and can consist of other functional blocks. Thus, the structure of a block has to be described only once while the block can be multiple used as a black box. On the other hand the increase in circuit size and the limited accessibility to the internal nodes of a circuit lead to an enormous increase in the cost of test preparation and test execution. Among the different tasks of test preparation the fault simulation is of great importance. Especially for combinational circuits a lot of efficient methods have been developed during the last few years [1,2,3,4]. However, most of them are restricted to circuits which are described at the gate level. Experiments with a concurrent fault simulation algorithm have shown that the hierarchy can be used to accelerate the fault simulation process [5,6,7,8,9]. Thus, our approach aims at extending an efficient fault simulation approach for combinational circuits [1,2,3] described at the gate level in order to achieve the capability of not only handling hierarchically described circuits but also use the hierarchy for accelerating the fault simulation and reducing the memory requirements. For that purpose we have extended a former approach [10] which was restricted to two levels of hierarchy in such a way that it can handle arbitrary levels of hierarchy. Moreover, we introduce several new concepts which lead to a further acceleration of the simulation process, and finally, we succeeded in illustrating the hierarchy in circuit structure by two new graphs.

2 Fault simulation at the gate level
Our fault simulation approach is based on the stuck-at-fault model. We combine the well known PPSFP-technique [11,12] with the concept of fanout free regions (FFR) that has been proposed by Hong [13]. We use parallel processing of pattern at all stages of the simulation procedure. The fanout free regions of a gate level circuit C can be determined by disconnecting all fanout branches from their stems. They divide a circuit into several disjoint subcircuits. Please note, that in the following we will distinguish stems and their fanout branches and denote fanout stems as well as branches as nodes of the circuit. Each fanout free region is uniquely described by its terminal node which is a fanout stem or primary output. Fanout free regions contain all ancestor nodes of the terminal node in the disconnected circuit which are either primary inputs or fanout branches of other stems. The expensive explicit fault simulation is restricted to the stems while the observability of all other nodes inside the fanout free regions is computed by an inexpensive procedure of linear time complexity. The number of gate evaluations is further reduced by the evaluation of a so-called check-up criterion of the fanout free regions which minimizes the number of explicit fault simulations.

3 Representation of the Hierarchy
This section introduces the formal representation of the functional blocks and illustrates the hierarchical structure of the circuit description.

Figure 1: Hierarchical Circuit

Figure 1 and 2 show a hierarchical described circuit
Figure 2: Elevation \( SS' \) of the Hierarchical Circuit and its elevation \( SS'' \). Thereby each box models a functional block whereas elementary gates have been left out for sake of simplicity. Basically, there is a difference between the type of a functional block and its instance. The type consists of the structural and functional information of this block. Every functional block of a certain type in a hierarchical circuit is called an instance of this type. The blocks in figure 1 and 2 are denoted by their type with the number of the instance as index.

In order to describe the dependencies between the blocks, two graphs are introduced:

1. The Graph of Types \( G_T = (V_T, E_T) \) that specifies for every type, in which types it is included and which types itself includes. Every node \( k \in V_T \) in the graph corresponds to a type. There is an edge \( (i, j) \in E_T \) if the type \( i \) includes the type \( j \) in the circuit description.

2. The Graph of Instances \( G_H = (V_H, E_H) \) that describes in which way instances, i.e. the functional blocks of a hierarchical circuit, are related to each other. Every node \( k \in V_H \) models an instance in the circuit. There is an edge \( (i, j) \in E_H \) is given, if the instance \( i \) contains the instance \( j \) in the circuit description.

These two graphs can be used highly efficient during fault simulation. The graph \( G_H \) is important for the simulation process. For example, with its help the FFIs can be expanded or the functional description of the instances can be used (see section 4). \( G_T \) and \( G_H \) are important to organize the necessary dates, e.g. to store the structural and processing information of the circuit. For the given example, these two graphs are shown in the figures 3 and 4. These figures show also the level of hierarchy (LoH) of each type and instance.

4 Basic Concepts of the Hierarchical Fault Simulation

The following items describe the main concepts of the hierarchical fault simulation and point out their influence on the single simulation steps with respect to savings in CPU-time and memory requirements during the fault simulation process.

Covering Status of a Functional Block: During the true value simulation the hierarchy in circuit description can be extremely useful, if there are functional blocks whose faults have been detected during the fault simulation procedure. To mark those blocks, whose internal realization needs not to be considered any more, a covering status \( A (CS_A) \) is introduced.

\[
CS_A(K) = \forall \exists \{ D(x_K/0) \cdot D(x_K/1) \} \cdot \forall \exists \{ CS_A(J) \} \quad (1)
\]

with \( D(x_K/0) \) being the detection status of the fault \( x_K \) stuck-at-0. Thereby \( des_{GH}(K) \) denotes the set of descendents of \( K \) in \( G_H \). \( CS_A(K) \) will be true, if there is no undetected fault in the considered block \( K \). Of course, this implies that \( K \) contains no other functional blocks with \( CS_A = 0 \).

By definition, a covering status \( B \; CS_B(K) \) is true, if no fanout free region, corresponding to a terminal node that lies in \( K \), contains any undetected faults. Otherwise an explicit fault simulation initialized at a node inside of \( K \) is necessary. As a consequence a true value simulation of \( K \) would have to be performed.

\[
CS_B(K) = \forall \exists \exists \{ D(j/l) \cdot D(j/l) \} \quad (2)
\]

The calculation of \( CS_A \) and \( CS_B \) can be performed very efficiently.

Functional Description of the Blocks: If there is only the relation between the inputs and the outputs of a block \( X \) of interest, the internal structure of \( X \), which means the gate level realization, needs not to be considered. In this case only the functional description \( A^X = B^X(E^X) \) is used, where \( E^X \) and
$A^X$ are the values of the block input and block output nodes, respectively, and $B^X$ represents the functional description of $X$. As a consequence, it is not necessary to proceed at the gate level of a block in order to compute the output values.

Handling the Internal Signals of a Block From Higher Levels: During the fault simulation it is advantageous to simulate on the highest possible level of hierarchy. To get no lack in fault simulation accuracy, also faults at internal nodes of a block, i.e. faults at nodes on lower levels of hierarchy, have to be considered.

Especially while considering small functional blocks, occurring frequently in hierarchical circuit descriptions, internal nodes can be treated very favorable on higher levels by using functional dependencies. On the one hand the controllability at a node has to be determined. Let $s$ be an internal node of a block $X$. The function realized by the node $s$ is represented by $s = B^X(s)$. The observability $O^X_s$ of the node $s$ at the $m$ block output nodes, on the other hand, can be expressed by $O^X_s = (O^X_{s1}, ..., O^X_{sm})$ with $O^X_{si} = b^X_i(s = 0) = b^X_i(s = 1)$, where $b^X_i(s = w)$ is the functional description of the block output $i$ while forcing the value of the node $s$ to $w$.

Multiple Exploitation of the Structural and Functional Description: A tremendous advantage of a hierarchical circuit representation lies in the fact that the structural and functional description of a block has to be stored only once per type and not for every instance of this type. This results in an enormous reduction of storage area which is especially for large circuits of great importance.

Expanding the Fanout Free Regions: Exploiting the hierarchical concept, it is possible to expand the fanout free regions of a circuit not only from a lower level to a higher level and vice versa but also to expand it over a functional block from its outputs to its inputs under certain conditions. As a consequence, the number of FFRs — and of course the number of explicit fault simulations — can be reduced.

5 Hierarchical Fault Simulation

In this section we describe the hierarchical fault simulation. The principle way of processing is the same as in case of gate level simulation, i.e. after a true value simulation for every FFR a backward traversal and for every FOS an explicit fault simulation is performed. To explain all important facts as illustrative as possible, the circuit HC of figure 5 is chosen as an example.

5.1 Parallel Processing of Patterns

A very important characteristic of our fault simulation approach is the fact that all calculations and evaluations can be performed in parallel. In order to support parallel processing of pattern during the fault simulation we use a $L$-bit wide machine word vec($a$) for representing a set of $L$ logic values of a node $a$.

5.2 Levelization

Since we use the principle of event driven fault simulation, a levelization of the circuit is necessary. This problem can be reduced to the determination of the rank of nodes in a directed acyclic graph, i.e. a standard task of graph theory. In this special case of hierarchical fault simulation a problem, which is illustrated in figure 5, may occur. The node $n_i$ which is an output of block $D_3$ feeds an input of the same block. Considering the block as a black box, a feedback appears. However, considering the gate level realization it turns out that no real feedback occurs as $n_i$ does not feed $n_1$. We call this phenomenon a "pseudo feedback".

5.3 Fault Modeling

As already mentioned, one important characteristic of the hierarchical fault simulation is the multiplicity of a hierarchical circuit simulation. Analogously to the fault simulation at the gate level, the hierarchical true value simulation is performed from the primary inputs towards the primary outputs. Whenever a functional block is reached whose covering status is not true, the logic values from the higher level are propagated to the input nodes of the lower one. Then the simulation continues inside the block.

During the true value simulation, the hierarchical description of the circuit can be used very advantageously. If there are blocks with $CS_A \cdot CS_B = 1$, the true value simulation has not to be performed for the nodes of this block. The values of the output nodes of an instance can be calculated by evaluating the functional description of the block.

For our example the true value of the output $n_1$ of $D_3$ can be found by:

$$\text{vec}(n_1) = (\text{vec}(c_1) \cdot \text{vec}(d_1)) + \text{vec}(h_1)$$

5.5 Fanout Free Regions

The observability of the nodes inside the FFRs are computed by an algorithm that is linear in time with the number of nodes [13]. Thereby the FFR is passed from its fanout stem backwards and the observabilities are determined recursively with the aid of the Boolean Difference. Hence, it is important to minimize the number of FFRs by expanding them as far as possible in order to minimize the number of explicit fault simulations which have to be performed. A fanout free region of a hierarchical described circuit can be determined, beginning at its terminal node,
with the following rules, which are illustrated by the example of figure 6.

Rule 1: The FFR(n₁) will be expanded from the output n₁ of an instance D₃ to the node n₃ inside the instance, if n₃ is the node in D₃ corresponding to n₁.

Rule 2: The FFR(n₁) will be expanded from the output n₁ of an instance D₃ to the input h₁ of this instance, if n₁ = n₁(h₁) and for all other outputs m₁ of D₃ we have m₁ ≠ m₁(h₁). Of course, the node h₁ corresponding to h₁ is also included in this FFR.

Rule 3: The FFR(i₃) is extended from the input node d₃ of an instance D₃ to the node d₃ on the higher level of hierarchy, if d₃ is the corresponding node to d₃ and if d₃ hasn't yet been attached to another FFR by Rule 2.

The following five FFRs can be determined: FFR(hₙ₁) = {b₁}, FFR(i₃) = {b₂, b₃, d₃, f₃}, FFR(n₁) = {n₁, n₃, hₙ₃, hₙ₃, hₙ₁, hₙ₅, hₙ₅, hₙ₅}, FFR(p₁) = {p₁, p₄, p₅, p₆, p₇, p₈, p₉, p₁₀}, FFR(q₁) = {q₁, q₅, q₇, q₉, q₅, q₆, m₅, m₆, m₇, m₈, q₉, q₈, q₇, q₆, q₅, q₄, q₃, q₂, q₁}. Please note that at the gate level we would have a sixth FFR(hₙ₁) = {hₙ₃, hₙ₂, b₅}. As a consequence, the hierarchical fault simulator has to perform only five instead of six explicit fault simulations which results in a reduction of the simulation time.

5.6 Backward Traversal inside the FFR

To reduce the number of explicit fault simulations during the hierarchical fault simulation a check-up criterion is introduced in order to decide, whether an explicit fault simulation is reasonable, because it can lead to the detection of at least one still undetected fault inside the corresponding FFR. However, the hierarchical description of the circuit provides additional advantages while determining the check-up criterion. One improvement is given in making advantage of the covering status CSₐ and CS₈. With their help the computation of the check-up criterion is restricted to those nodes of the FFR, which are in blocks with any undetected faults. The calculation for nodes of the FFR lying in blocks with CSₐ = 1 can be omitted.

To perform the check-up a so-called detection status Q of a node y is introduced:

\[ vec(Q(y)) = vec(y) \cdot D(y/0) + vec(y) \cdot D(y/1) \]  

Using the Boolean Difference \((n₁)ₙ\) the check-up criterion given as

\[ vec(C(n₁)) = \exists y ∈ FR(n₁) \land y \notin vec(n₁) \cdot vec(Q(y)) \]  

indicates, whether there exists at least one undetected fault, which can be stimulated by a test pattern and which can be observed at the stem. Thereby \((n₁)ₙ\) denotes the observability of the node y at node n₁. Again, the functional description of the instances can be used very efficiently.

The calculation of the check-up criterion can be fastened by using information of higher levels. For this reason the description of block internal nodes are exploited. To evaluate the equations 4 and 5 more structural information is necessary, if for an internal node y the equation for the controllability \(y = y(E)\) and additively for each output \(z\) the equation for the observability \(z_y = z_y(E)\) is given as a function of the input values \(E\). As already explained in chapter 4, these equations have to be calculated only once for every type during a preprocessing phase. During the backward traversal the observability \((n₁)ₙ\) of each node \(n₃\) of the FFR at the stem is recursively computed with the aid of the Boolean Difference, according to:

\[ (n₁)ₙ = (n₁)ₙ + (n₁)ₙ \]  

5.7 Explicit Fault Simulation of the Fanout Stems

If the check-up criterion \(C(n₁)\) of a FFR(n₁) is true, i.e. there are some undetected faults that are observable at the Fanout Stems (FOS), the fault injection is given by:

\[ vec(n₁) = vec(n₁) + vec(C(n₁)) \]

Then the fault effect is propagated towards the primary outputs. Thereby the principle of the hierarchical fault simulation is used very efficiently. If a functional block is reached, the fault effect is propagated directly from the block inputs to the block output nodes with the aid of the functional description. For example, the faulty value of \(m₁\) in our example shown in figure 5 is determined by

\[ m₁ = vec(c₁ \cdot d₁ \cdot n₁ \cdot e₁) \]

The explicit fault simulation of block internal nodes can also be performed by evaluating equations representing the functional description. As an example the simulation of \(y\) starts not with the injection at \(a₃\) but immediately at the outputs of block \(D₃\), using the following equations:

\[ vec(i₃) = vec(i₃) \oplus vec(C(i₃)) \]

\[ vec(n₁) = vec(h₁) \oplus vec(i₃) + vec(h₁) \]

\[ vec(m₁) = vec(i₃) \cdot vec(n₁) \cdot vec(c₁) \]
The advantage of this proceeding is evident. On one hand the CPU-time is reduced because the number of gate evaluations is reduced. On the other hand the true value simulation of blocks, whose internal nodes can be treated from the higher level by functional dependencies during all stages of the fault simulation procedure, can be omitted which leads to a reduction of the CPU-time. Furthermore the true and faulty values and the observabilities do not have to be stored, thus, reducing the memory requirement.

Finally, as soon as the fault effect is at a primary output, the observability of the FOS is determined by:

\[ vcc(p_1)_{out} = vcc(p_1) + vcc(p_1) \]  

and all detected faults can be marked.

### 6 Experimental Results

To show the efficiency of the proposed algorithm, the hierarchical fault simulator has been implemented in the programming language "C" on a DECstation 5000/200. We performed experiments with several industrial circuits which are hierarchical described with up to 5 levels of hierarchy and whose size vary from about 400 to 90,000 gates. For the sake of comparison a fault simulation was performed for all flattened circuits with a fast fault simulator working on gate level which represents the state of the art [1,2,3]. Of course, both fault simulators are implemented in the same programming language and run under the same operating system and the same computer.

The results of the simulation with respect to the CPU-time is shown in Table 1. For all circuits the hierarchical fault simulation was faster than its reference. Especially for large circuits an enormous speed up, which ranges up to 8.5, was achieved. A further fact that can be concluded from Table 1 is the correspondence between the acceleration and the reduction of gate evaluations during the explicit fault simulation.

This reduction of gate evaluations is caused by the exploitation of the functional description of the blocks. For the computation of an output value of a block there are basically two different ways. On one side the equation representing the function of a block can be used. Thereby the input variables of this function are imposed with the actual node values at the gate inputs and the output values are computed. On the other side a table may be used. Either of the strategies for the evaluation of a block's function mentioned above — depending on the special block — can be applied.

A further aspect of the hierarchical fault simulation is the dependency of the acceleration upon the number of levels of hierarchy. Two of the considered examples are available with various numbers of levels. The results are shown in Table 2. The main facts from Table 2 are the following. Mainly the larger the number of levels of hierarchy, the smaller is the CPU-time used. This evidently results from the exploitation of the functional description of the blocks. Please note that there is no linear dependency between the number of levels and the speed up, as the complexity of the blocks increases with the number of levels and, as a consequence, the time for evaluating the block's function. However, considering the circuit MULTPL in Table 2 a deficiency of the use of the functional description appears. The functional description of a block might be evaluated in vain because the fault cannot be propagated from its inputs to its outputs while at the gate level the fault effect propagation could have been stopped after a few gates. As a consequence no acceleration can be achieved. Additionally the number of nodes which have to be handled during the fault simulation increases with the number of hierarchical levels, because the input and output nodes of blocks are increasingly described in more and more instances. This leads to the effect that the fault simulation performs faster if four levels of hierarchy instead of five are exploited.

The exploitation of the functional description of the instances has the largest effect on the acceleration. Hence, if the time necessary for the explicit fault simulation...
Table 3: Dependency of the CPU-time and the Memory requirement of the circuit size

<table>
<thead>
<tr>
<th>Word length</th>
<th>Number of gates flat</th>
<th>Number of gate evaluations flat</th>
<th>Number of gates hierarchical</th>
<th>Number of gate evaluations hierarchical</th>
<th>CPU-time [sec] flat</th>
<th>CPU-time [sec] hierarchical</th>
<th>Memory requirement [Byte] flat</th>
<th>Memory requirement [Byte] hierarchical</th>
</tr>
</thead>
<tbody>
<tr>
<td>4</td>
<td>148</td>
<td>143</td>
<td>136</td>
<td>129</td>
<td>0.03</td>
<td>0.07</td>
<td>129200</td>
<td>129200</td>
</tr>
<tr>
<td>8</td>
<td>296</td>
<td>304</td>
<td>160</td>
<td>160</td>
<td>0.07</td>
<td>0.04</td>
<td>71104</td>
<td>71104</td>
</tr>
<tr>
<td>16</td>
<td>592</td>
<td>604</td>
<td>320</td>
<td>320</td>
<td>0.04</td>
<td>0.03</td>
<td>16490</td>
<td>16490</td>
</tr>
<tr>
<td>32</td>
<td>1184</td>
<td>1192</td>
<td>640</td>
<td>640</td>
<td>0.03</td>
<td>0.02</td>
<td>30481</td>
<td>30481</td>
</tr>
<tr>
<td>64</td>
<td>2368</td>
<td>2376</td>
<td>1280</td>
<td>1280</td>
<td>0.02</td>
<td>0.01</td>
<td>50240</td>
<td>50240</td>
</tr>
</tbody>
</table>

simulation dominates the overall simulation time, an enormous speed up can be obtained. Table 3 shows this fact by the example of a multiplier having the word length as a parameter. With increasing word length the portion of the explicit fault simulation at the overall simulation time increases, too. Therefore, an increasing acceleration of the fault simulation is obtained.

The right art of Table 1 shows the results of the hierarchical fault simulation, with regard to the storage requirements. Considered is the storage area of all essential data necessary for the simulation. Obviously, for all circuits a reduction of the storage area is obtained by taking advantage of the hierarchy in circuit description. This results from the fact that the structural and functional information of the instances has to be stored only once for each type of block. The amount of the reduction depends on the number of instances of the same type and on the size of the type. Table 3 illustrates this fact. With increasing word length and, thus, the increasing circuit size the number of instances of the same type increases as well. A further aspect is the dependency of the storage area reduction upon the numbers of levels of hierarchy as shown in Table 2. Nevertheless, two disadvantages are remarkable. Firstly, some nodes may be multiple described (on different levels). This requires additional storage area. Furthermore, supplementary storage area is necessary in order to store the information expressed by the graphs \( G_T \) and \( G_H \) which is needed to perform the hierarchical fault simulation. But, as shown in Table 1, 2 and 3 the advantages of hierarchical fault simulation prevail in every case. Finally, it is worth mentioning that the reduction of storage area has an other tremendous advantage. The phenomenon of "paging" may be avoided. As an example, a reduction of the storage area of 50% in combination with a reduction of the CPU-time of 40% can diminish the elapsed time, which is the time the user has to wait for results, up to the factor of 50.

7 Conclusion

We have presented an efficient fault simulation approach for combinational circuits which are hierarchically described. Parallel processing of patterns is applied at all stages of the fault simulation procedure. The concept of fanout free regions is extended in order to get larger fanout free regions and, thus, to reduce the number of explicit fault simulations. In addition a covering status is used for the acceleration of the true value simulation as well as the evaluation of the check-up criterion. We have further introduced a method to handle the internal nodes of a block at a higher level of hierarchy. As the experimental results for industrial circuits with up to about 90,000 gates demonstrate the new approach is significantly faster as compared to an efficient gate level fault simulator. For all circuits the hierarchical fault simulation needs less than half a minute of CPU-time on a workstation. Moreover, the hierarchical fault simulator requires up to three times less memory than the gate level fault simulator.

Acknowledgement

The authors are very grateful to Professor Dr.-Ing. K. Antreich from Technical University of Munich for his valuable suggestions and his helpful advice. In addition, the contribution of Dr.-Ing. M. Schulz should be greatly appreciated.

References