FACTA UNIVERSITATIS (NIŠ)
SER.: ELEC. ENERG. vol. 15, No. 1, April 2002, 123-136

Defect-Oriented Mixed-Level Fault Simulation in Digital Systems

Raimund Ubar, Jaan Raik, Eero Ivask and Marina Brik

Abstract: A new method for mixed level defect-oriented fault simulation of Digital Systems represented with Decision Diagrams (DD) is proposed. We suppose that a register transfer level (RTL) information along with gate-level descriptions for RTL blocks are available. Decision diagrams (DDs) are exploited as a uniform model for describing circuits on both levels. The physical defects in the system are mapped to the logic level and are simulated on the mixed gate- and RT levels. The approach proposed allows to increase the accuracy of test quality estimation, and to reduce simulation cost in comparison to traditional gate-level fault simulation methods.

Keywords: Digital system, decision diagram, fault simulator, mixed level simulation.

1 Introduction

Fault simulators of digital circuits and systems are used widely in many areas of design and test like test generation, fault diagnosis, test set compaction etc. The quality of test generation significantly relies on the efficiency of fault simulation, especially in the case of simulation-based test generators [2,3]. Traditionally, fault simulation is performed at the gate-level with using the stuck-at fault (SAF) model. On one hand, the gate-level SAF-based fault simulation is time-consuming, on the other hand the SAF model doesn't

Manuscript received March 26, 2002. An earlier version of this paper was presented at the 23rd International Conference on Microelectronics, MIEL 2002, May 12-15, 2002, Niš, Serbia.

The authors are with the Department of Computer Engineering, Technical University of Tallinn, Raja 15, 12618 Tallinn, Estonia (e-mail: `[raiub, jaan, ieero, mbrik]@pld.ttu.ee`).

represent adequately the physical defects in transistor circuits. To overcome these disadvantages, mixed-level simulation is needed which allows to use the advantage of high level information to speed up the simulation process while analysing the quality of tests still in relation to realistic physical defects.

In this paper, a new method for parametric defect modeling is presented for calculating the conditions for activating physical defects in the modules (e.g. library components) of digital circuits. A new method for multi-level defect-oriented fault simulation based on Decision Diagrams (DD) is proposed. We suppose that a register transfer level (RTL) information along with gate-level descriptions for blocks of the RTL structure are available. For defect simulation a new functional fault model is used which can be handled on the gate- and RT levels. Decision diagrams (DDs) are exploited as a uniform model for describing systems on both, RT and gate levels. In Section 2 of the paper we describe the general case of the DD model for representing digital systems hierarchically on the gate- and RT levels. In Section 3 we discuss the problem of mapping physical defects onto the logical level. Section 4 is devoted to describing the general idea of the hierarchical defect-oriented fault simulation. In Section 5 we develop the ideas of hierarchical fault simulation for using on the model of DDs. In Section 6 we discuss some experimental results, and finally, in Section 7 we present the concluding remarks.

2 The Model for Simulation

Consider a digital system as a network $N = (Z, F)$ of components where Z is the set of all variables (Boolean, Boolean vectors, integers) which represent the connections between components, inputs and outputs of the network. Denote by $X \subset Z$ and $Y \subset Z$, correspondingly, the subsets of input and output variables. $V(z)$ denotes the possible values for $z \in Z$, which are finite. Let F be the set of digital functions on Z: $z_k = f_k(z_{k,1}, z_{k,2}, \ldots, z_{k,p}) = f_k(Z_k)$ where $z_k \in Z$, $f_k \in F$, and $Z_k \subset Z$. Some of the functions $f_k \in F$, for the state variables $z \in Z_{STATE} \subset Z$, are next state functions.

Definition 1. A decision diagram (denoted as DD) is a directed acyclic graph $G = (M, \Gamma, z)$ where M is a set of nodes, Γ is a relation in M, and $\Gamma(m) \subset M$ denotes the set of successor nodes of $m \in M$. The nodes $m \in M$ are marked by labels $z(m)$. The labels can be ether variables $z \in Z$, or algebraic expressions of $z \in Z$, or constants.

For non-terminal nodes m, where $\Gamma(m) \neq \emptyset$, an onto function exists between the values of $z(m)$ and the successors $m^e \in \Gamma(m)$ of m. By m^e we denote the successor of m for the value $z(m) = e$. The edge (m, m^e) which connects nodes m and m^e is called *activated* iff there exists an assignment $z(m) = e$. Activated edges, which connect m_i and m_j make up an *activated path* $l(m_i, m_j)$. An activated path $l(m^0, m^T)$ from the initial node m^0 to a terminal node m^T is called *full activated path.*

Definition 2. A decision diagram $G_k = (M, \Gamma, z)$ represents a function $z_k = f_k(z_{k,1}, z_{k,2}, \ldots, z_{k,p}) = f_k(Z_k)$ if for each value $v(Z_k) = v(z_{k,1}) \times v(z_{k,2}) \times \cdots \times v(z_{k,p})$, a full path in G_k to a terminal node mT is activated, where $z(m^T) = z_k$ is valid.

Each function $f_k \in F$ in the system network $N = (Z, F)$ is represented by a decision diagram $z_k = G_k(Z_k)$ [5-7]. Depending on the class of digital system (or level of its representation), we may have various DDs, in which nodes have different interpretations and relationships to the system structure. In RT level descriptions, we usually decompose digital systems into control and data parts. State and output variables of the control part serve as addresses or control words, and the variables in the data part serve as data words. The functions of RTL components in the data part are described by the high-level data word variables.

The BDDs [1,4] represent a special class of DDs where all the variables in Z are binary, i.e. for all $z \in Z$, $V(z) = 2$.

Consider a digital system in Fig. 1 which consists of control and data parts. The control part is given by the output and next-state functions $y = \lambda(q', x)$, $q = \delta(q', x)$, where y is an integer output vector variable, which represents a microinstruction with 4 control fields $y = (y_M, y_z, y_{z,1}, y_{z,2})$, $x = (x_A, x_C)$ is a Boolean input vector variable, and q is the integer state variable. The value j of the state variable corresponds to the state s_j of the FSM. The apostrophe refers to the previous clock cycle.

The data path consists of the memory block M with three registers A, B, C together with the addressing block ADR, which can be represented by three DDs: $A = G_A(y_M, z)$, $B = G_B(y_M, z)$, $C = G_C(y_M, z)$, of the data manipulation block CC where $z = G_z(y_z, z_1, z_2)$, and of two multiplexers $z1 = G_{z,1}(y_{z,1}, M)$ and $z_2 = G_{z,2}(y_{z,2}, M)$. The block COND performs the calculation of the condition function $x = G_x(A, C)$.

The component level model of the system can be represented by the following set of DDs: $N_1 = \{G_q, G_y, G_A, G_B, G_C, G_z, G_{z,1}, G_{z,2}, G_x\}$. By superpositioning of DDs we can obtain the following compressed model for

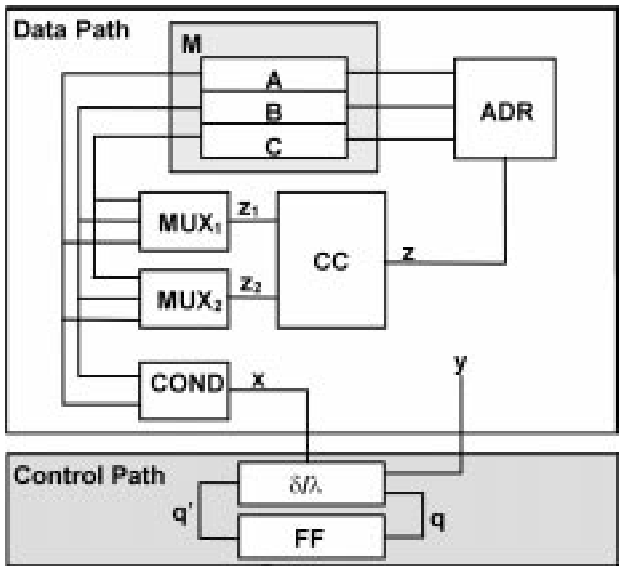

Fig. 1. A digital system.

describing the behaviour of the register A:

$$\begin{aligned} A &= G_A(y_M, z) = G_A(y_M, G_z(y_z, z_1, z_2)) \\ &= G_A(y_M, G_z(y_z, G_{z,1}(y_{z,1}, M), f_4(y_{z,2}, M))) \\ &= G_A(y_M, y_z, y_{z,1}, y_{z,2}, M) = G_A(y, M) \\ &= G_A(G_y(q', x), M) = G'_A(q', A, B, C). \end{aligned}$$

In similar way we can compress other DDs in N_1 and represent the whole system by a new *compact* DD model:

$$N_2 = \{G_q, G'_A, G'_B, G'_C\}.$$

An example of a DD $G'_A(q', A, B, C)$ for the register A with the following behaviour

$$A = \begin{cases} B' + C', & \text{if} \quad q' = 0, \\ \neg A' + 1 & \text{if} \quad q' = 1 \ \& \ x_A = 0, \\ \neg C' + B', & \text{if} \quad q' = 3 \ \& \ x_C = 1, \\ A' + B' + C', & \text{if} \quad q' = 4 \ \& \ x_A = 0 \& x_C = 0, \\ A', & \text{in other cases.} \end{cases}$$

is represented in Fig. 2.

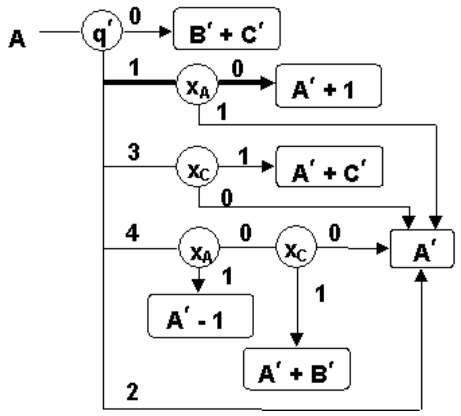

Fig. 2. DD for the subcircuit of A in the system in Fig.1.

Fault analysis on DDs is based on *path traversing* procedures. In path traversing, the values of node-variables are given, and we have to move along the path determined by these values. Suppose, a test pattern P assigns to a node variable $z(m)$ a value $e \in V(z(m))$. Suppose also, there is a defect d which influences on the variable z(m) and changes the value e to D. The defect is detected by the pattern P if the following conditions are fulfilled:

- P activates the paths: $l(m^0, m)$, $l(m^e, m^{T_e})$, $l(m^D, m^{T_D})$, and
- $z(m^{T_e}) \neq z(m^{T_D})$ is valid.

3 Mapping Defects onto the Logical Level

In this Section we present a new general fault model for describing and modeling arbitrary physical defects in the components of digital circuits and for mapping them onto the logical level.

Consider a Boolean function $y = f(x_1, x_2, \ldots, x_n)$ implemented by an embedded component in a digital circuit. Introduce a Boolean variable d for representing a given defect in the component or in the neighbourhood layout of the component, which may affect the value y by converting the Boolean function f into another function $y = f^d(x_1, x_2, \ldots, x_n, x_{n+1}, \ldots, x_p)$. Here, the new variables $x_{n+1}, \ldots, x_p$ may be introduced to describe the influence of the neighbourhood layout of the component in the presence of the physical defect d.

For example, assume there is a short between x_1 and x_5 in the circuit in Fig. 3. The faulty function $y = f(x_1, x_2) = \neg(x_1 \wedge x_2)$ in the case of the defect d can be represented as

$$y = f^d(x_1, x_2, x_3, x_4) = \neg(x_1 x_5) \vee x_2 = \neg(x_1 \neg(x_3 \vee x_4)) \wedge x_2).$$

Introduce now a generalized parametric function

$$\begin{aligned} y^* =& f^*(x_1, x_2, \dots, x_n, x_{n+1}, \dots, x_p, d) \\ =& (\neg d \wedge f) \vee (d \wedge f^d) \end{aligned}$$

as a function of a defect variable d, which describes the behavior of the component simultaneously for both possible cases. For the erroneous case the value of the *defect variable* d as a parameter is equal to 1, and for the nonerroneous case $d = 0$. In other words, $y^* = f^d$ if $d = 1$, and $y^* = f$ if $d = 0$.

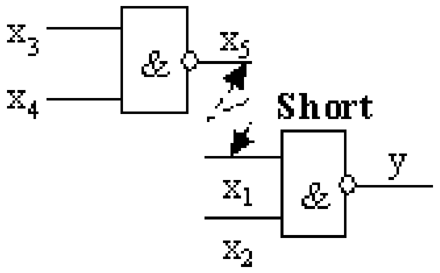

Fig. 3. A short between two signal leads.

The solution of the Boolean differential equation

$$W^d = \frac{\partial y^*}{\partial d} = 1 \tag{1}$$

describes the conditions which activate the fault d on a line y. For example, for the short in Fig. 3 we have

$$y^* = \neg df \vee df^d = \neg d \neg(x_1 \wedge x_2) \vee d(\neg x_1 \neg(x_3 \wedge x_4) \vee \neg x_2).$$

To find the conditions for activating the short to the line y we have to solve the logical equation

$$W^d = \frac{\partial y^*}{\partial d} = x_1 x_2 x_3 x_4 = 1.$$

The method of parametric defect modeling by logical conditions W^d can be generalized for the purpose of hierarchical fault simulation. A component of a circuit can be preprocessed by lower level defect simulation with the goal to generate a set of conditions W for all possible lower level defects d of the component. Each condition as a solution of $W_d = 1$ can be regarded as a higher level functional fault model for a given defect d, since in the presence of this defect the functional behavior of the component at the input where $W_d = 1$ will be erroneous. The functional fault model concept is illustrated in Fig. 4.

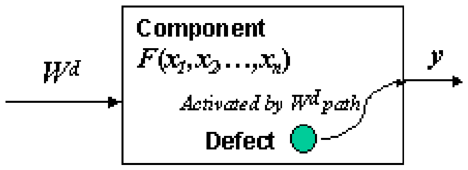

Fig. 4. Functional fault model for a physical defect.

The relationships between the functional faults (patterns) W^d and the defects d for all the logic level simple or complex gates g in the library L are given by defect tables $DT_g = \|g_{id}\|$, $g \in L$, where an entry $g_{id} = 1$ means that the input pattern i (solution of $W_d = 1$) of the gate detects the defect d, otherwise $g_{id} = 0$.

4 Hierarchical Fault Simulation

Consider a task of defect oriented fault simulation in a system represented on three levels: register transfer, gate and defect levels.

Formally, if Y is the system RTL variable representing an observable point of the system, y_M is an output variable of a gate-level module and y_C is the output of a component (complex gate) in the module with a physical defect d, then the condition of detecting the defect d on the observable test point Y can be represented as

$$W = \frac{\partial Y}{\partial y_M} \wedge \frac{\partial y_M}{\partial y_C} \wedge Wd = 1, \tag{2}$$

where $\partial Y/\partial y_M$ is the Boolean derivative calculated by the high-level simulation, $\partial y_M/\partial y_C$ is the Boolean derivative calculated by the gate-level simulation, and W_d is the functional fault condition found by the gate defect-level preanalysis.

In the fault simulation approach proposed in the paper the defect analysis is made module by module in the higher RT level network. An example of a RTL system to describe the main principles of the approach is illustrated

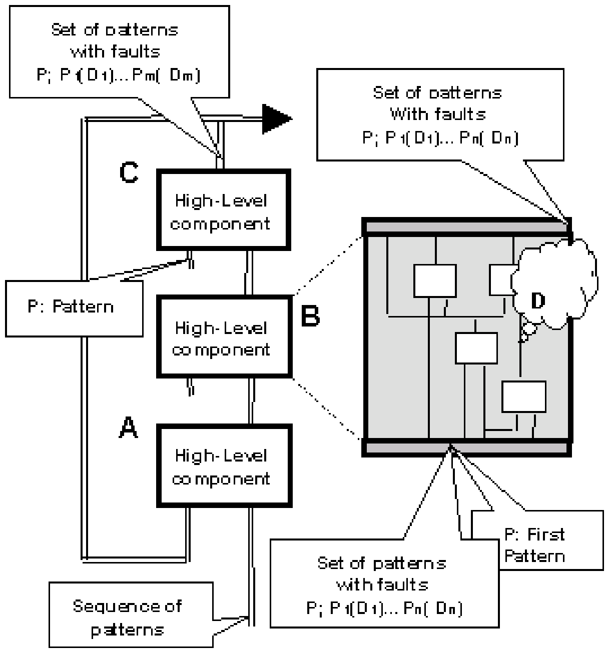
Set of patterns with faults P; P1(D1)...Pm(Dm)
Set of patterns With faults P; P1(D1)... Pn(Dn)
C
High-Level component
P: Pattern
High-Level component
B
D
A
High-Level component
P: First Pattern
Set of patterns with faults P; P1(D1)... Pn(Dn)
Sequence of patterns

Fig. 5. Hierarchical fault modeling in a digital system.

in Fig. 5. Let us have a network of a system consisting of 3 parts: A, B, and C. The block B is taken currently as the target for defect oriented fault analysis, and therefore is represented at the lower gate-network level. The test sequence before reaching the target block is simulated in the part A on the RT level, pattern by pattern, starting from the inputs of the system. Let D be the set of all defects to be simulated in the target block B. The low-level fault simulation for a given input pattern P* in the target block B is carried out by the following procedure.

Procedure 1. When the target block B is reached by a pattern P^* first, the faultfree output pattern P is calculated. Then, the low level defect analysis is carried out, and the set of all defects $d \in D_B \subset D$ activated in B by the pattern P^* is calculated by checking if at least for one output y_B of the block B the condition

$$W = \frac{\partial y_B}{\partial y_G} \wedge W^d = 1 \tag{3}$$

is fulfilled. For each activated defect $d \in D_B$ in B, the corresponding faulty output pattern of the block B is calculated. Then, the all activated defects $d \in D_B$ for which the same (faulty) output pattern of B is produced are grouped into the same subset $D_{B_i} \subseteq D_B$. As the result, a complex test pattern $T = \{P, (P1, D1), \ldots, (Pk, Dk)\}$ at the output of B will be generated where $D_1 \cup D_2 \cup \cdots \cup D_k = D_B \subset D$.

Suppose now, a block C (which is not the target block) is to be simulated at the higher level. All the input patterns of the block C can be regarded in general case as a set of complex test patterns $TS = \{T_1, T_2, \ldots, T_m\}$ where $T_i = \{P_{i,0}, (P_{i,1}, D_{i,2}), \ldots, (P_{i,k}, D_{i,k})\}$. This set can be easily reformed as a single joint complex pattern $T^* = \{P^*, (P_1^*, D_1^*), \ldots, (P_n^*, D_n^*)\}$. The high-level (RT-level) fault simulation for a given complex input pattern T* in the non-target block C is carried out by the following procedure.

Procedure 2. For each joint input pattern from the set $\{P^*, P_1^*, \ldots, P_n^*\}$ at the high-level, the corresponding output complex pattern $T' = \{P, (P_1, D_1^*), \ldots, (P_n, D_n^*)\}$ is calculated. If two input patterns P_i^* and P_j^* produce the same output pattern P_h then the two pairs (P_i, D_i^*) and (P_j, D_j^*) in T' should be merged, and the pattern P_h should be linked to a joint set of defects $D_h = D_i \cup D_j$. If the fault-free input pattern P^* and a faulty input pattern P_j^* produce the same output pattern P, then all the defects in D_j^* are self-masked, and the component (P_j, D_j^*) should be removed from the complex pattern T'. As the result, a new reduced complex test pattern $T = \{P, (P_1, D_1), \ldots, (P_k, D_k)\}$ at the output of B where $k \leq n$ may be generated from T', so that $D_1 \cup D_2 \cup \cdots \cup D_k \subseteq D_B \subset D$.

When during the fault simulation the target block B is again reached via the feedback loops by a complex input pattern $T^* = \{P^*, (P_1^*, D_1^*), \ldots, (P_n^*, D_n^*)\}$, all the patterns $\{P^*, P_1^*, \ldots, P_n^*\}$ should be fault simulated on the low-level at the presence of corresponding defects. For P, new defects activated by P are calculated by using Procedure 1. After that, a new complex pattern T^* will be created at the output of B using the operations described in Procedures 1 and 2.

5 Defect-Oriented Fault Simulation on Decision Diagrams

Fault simulation on DDs is carried out by tracing the activated paths on DDs in accordance to the given values of variables as specified in Definition 1. For example, at the given input (state) pattern $P = \{q' = 1, x_A = 0\}$ of the block A we reach the terminal node m^T of the graph G_A with label $A' + 1$ (see the highlighted path in Fig. 2). The new value of A will be $A = A' + 1$.

In high-level fault propagation in the digital system $S = (Z, F)$ through a block with function $z = f(z_1, z_2, \ldots, z_n) = f(Z')$, $Z' \subseteq Z$, which is represented by a decision diagram G_z, we proceed from the fact that the defects may have been propagated to all of the variables $z_i \in Z'$ used in labels of nodes in the graph. To each node m of the DD with the label $z(m)$, a complex pattern

$$T_{z(m)} = \{P_{z(m),0}, (P_{z(m),1}, D_{z(m),1}), \dots, (P_{z(m),kz}, D_{z(m),kz})\}$$

corresponds. From this pattern, it results that a set of defects $D_{z(m)} = D_{z(m),1} \cup \dots \cup D_{z(m),k}$ has been propagated to the node m. Let D be the set of all faults currently activated and listed in $T_{z(m)}$.

Consider the fault simulation on the decision diagram Gz as the following set of procedures.

<u>*Procedure 3*</u>. The fault-free path is simulated in accordance to the fault-free input pattern $P_{z(m),0}$, and the fault-free value of $z = z(m^{T,0})$ is calculated, where $m^{T,0}$ is the terminal node of the fault-free activated path.

Denote the set of all nodes traced in the fault-free path up to the node m (m itself not included) by $M_{FF}(m)$. Let $D_{FF}(m)$ be the set of all faults propagated to the nodes $m \in M_{FF}(m)$. The condition of reaching the node m in the fault-free path during fault simulation is the absence of all the faults in $D_{FF}(m)$. Denote by $D_{CF}(m)$ the set of faults consistent to the current faulty path from the initial node m^0 up to the node m. For the nodes m on the fault-free path we have $D_{CF}(m) = D - D_{FF}(m)$.

Denote by L the list of all nodes of the DD to be fault simulated. All the nodes met on the fault-free path are included into dynamic list L. For carrying out fault simulation of the nodes in L, either Procedure 4 or Procedure 5 will be used. As the result of the procedure the list L will be updated. Fault simulation is terminated when the list L gets empty.

<u>*Procedure 4*</u>. Fault simulation of a terminal node $m^{T,0} \in L$ with the function $z = z(m^{T,0}) = f(z_1, \dots, z_p)$ for the set of complex input patterns $T = (T_1, \dots, T_p)$, $T_i = \{P_{i,0}, (P_{i,1}, D'_{i,1}), \dots, (P_{i,ki}, D'_{i,ki})\}$, $i = 1, 2, \dots, p$, where $\forall_{i,j}$: $D'_{i,j} = (D_{i,j} - D_{FF}(m^{T,0})) \cap D_{CF}(m)$ is equivalent to Procedure 2 of high-level fault simulation discussed in Section 4.

<u>*Procedure 5*</u>. Fault simulation of a nonterminal node $m \in L$ with the variable $z(m)$ for the complex pattern $T_{z(m)} = \{P_{z(m),0}, (P_{z(m),1}, D'_{z(m),1}), \dots, (P_{z(m),km}, D'_{z(m),km})\}$ where $\forall_{i,j}$: $D'_{z(m),j} = (D_{z(m),j} - D_{FF}(m)) \cap D_{CF}(m)$, consists in the following:

- if m belongs to the fault-free path, and if $D'_{z(m)} = D'_{z(m),1} \cup \dots \cup D'_{z(m),km} = \emptyset$ no nodes will be included into L;

- if m does not belong to the fault-free path, and if $D'_{z(m)} = \emptyset$, the node m^e where $e = P_{z(m),0}$, will be included into L; for the new node m^e in L we calculate: $D_{FF}(m^e) = D_{FF}(m) \cup D_{z(m^e)}$, and $D_{CF(m^e)} = D_{CF(m)}$,
- if $D_{z(m)} \neq \emptyset$, all the nodes m^e, where $e = P_{z(m),i}$, i: $D'_{z(m),i} \neq \emptyset$, will be included into L; for all these nodes we calculate $D_{CF(m^e)} = D_{CF(m)} \cap D'_{z(m),i}$, $D_{FF(m^e)} = D_{FF(m)}$.

As the result of the fault simulation by Procedures 4 and 5 we create a complex pattern for the graph variable z: $T_z = \{P_{z,0}, (P_{z,1}, D_{z,1}), \ldots, (P_{z,kz}, D_{z,kz})\}$. All the pairs $(P_{z,i}, D_{z,i})$ where $P_{z,i} = P_{z,0}$ are eliminated since the defects $D_{z,i}$ are self-masked at this point. All the groups of pairs $\{(P_{z,i}, D_{z,i}), (P_{z,j}, D_{z,j})\}$ where $P_{z,i} = P_{z,j}$ are merged into a single pair $(P_{z,i}, D_{z,i})$, so that $D_{z,i} = D_{z,i} \cup D_{z,j}$.

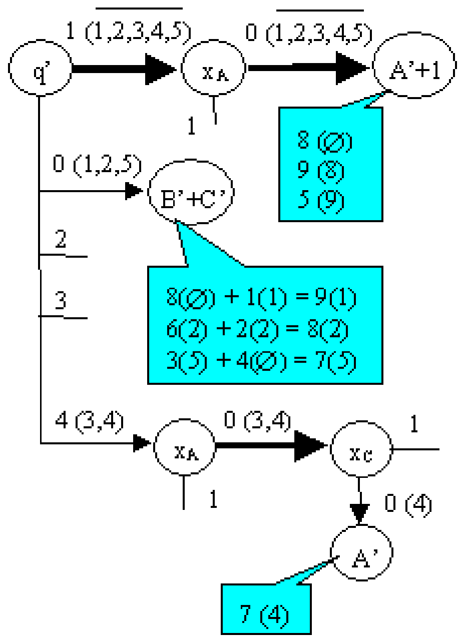

Fig. 6. Fault simulation on the graph GA on Fig. 4.

Example 4. Consider the DD G_A in Fig. 2 with a set of complex patterns: $T_q = \{1, 0(1,2,5), 4(3,4)\}$, $T_{xA} = \{0, 1(3,5)\}$, $T_{xC} = \{1, 0(4,6)\}$, $T_A = \{7, 3(4,5), 4(1,3,9), 8(2,8)\}$, $T_B = \{8, 3(4,5), 4(3,7), 6(2,8)\}$, $T_C = \{4, 1(1,3,4), 2(2,6), 5(6,7)\}$. All the paths traced during the fault simulation are highlighted and marked by details of simulation in Fig. 6. The fault free paths are shown by bold lines both, in Fig. 2 and Fig. 6. The edges on paths in Fig. 6 are labelled by pairs $e, (D)$, where e is the value of the node variable when leaving the node at this direction, and D is a subset of defects: $D_{FF}(m)$ for the next node m on the fault-free path, and $D_{CF}(m)$ for the next node m on the faulty paths. Since $D_{FF}(x_A) = \{1, 2, 3, 4, 5\}$ includes both of the defects 3 and 5 propagated to xA, no faulty paths are simulated from the node x_A: for the value $x_A = 1$: $D'_{x_A} = (D_{x_A} - D_{FF}(x_A)) = \emptyset$. From all the defects propagated to A', only the defects 8 and 9 are simulated at the node $A' + 1$. At the terminal node $B' + C'$ only the defects 1,2,5 are simulated, since only they are consistent to the condition of leaving the node q' at this direction.

After fault simulation of all 3 terminal nodes reached at the given complex pattern we compose the final result as follows: the defect 2 propagated to the node $B' + C'$ is selfmasked because the value $B' + C' = 8$ calculated for the defect 2 is equal to the fault-free value calculated at the node $A' + 1$. The defects 4 and 5 propagated to different terminal nodes are merged into the same group because they produces the same new value 7 for A. Also the defects 1 and 8 are merged into the same group. The final value of the new complex pattern for A is: $T_A = \{8, 5(9), 7(4, 5), 9(1, 8)\}$.

6 Experimental Results

For investigation the correlation between fault coverages for stuck-at faults (SAF) and the defects, we created two benchmark circuits C1 and C2 - both, tree-like combinational networks, the first with 2-levels (5 complex gates, 16 inputs, 100 defects) and the second with 3-levels (21 complex gates, 64 inputs, 420 defects). Both circuits were simulated for two tests Tmin (optimized test: 8 patterns for C1, and 16 patterns for C2) and T_{max} (not optimized test: 19 patterns for C1, and 70 patterns for C2) which both had 100% coverage for stuck-at faults. The results of the defect oriented simulation for the given tests are depicted in Table 1.

Table 1. Comparison of defect and SAF simulation.

Circuit	Number of defects	Stuck-at fault coverage	Defect coverage' %	
			T_{min}	T_{max}
C1	100	100,00	81,00	83,00
C2	420	100,00	84,29	84,76

From these experiments we see that the SAF-based fault coverage is overestimated compared to the realistic defect coverage, and that the difference between stuck-at fault and physical defect coverages reduces when the complexity of the circuit increases. In the worst case we have noticed that the 100% SAF test may cover only 50% of realistic physical defects.

In Table 2 the results of multi-level simulation for FSM benchmark circuits are shown for evaluating the proposed mixed-level fault simulation approach. A hierarchical multi-level fault simulator (HSIM) is compared to the plain gate-level simulator (GSIM). Here we see that the mixed-level fault simulation can be carried out with significally higher speed than in the case of plain gate-level simulation, the difference is between 2,7 and 121 times, or in average 35 times.

Table 2. Mixed-level fault simulation results.

FSM	Number	Test	Fault cover	Time,s	
circuit	of faults	length	%	HSIM	GISM
bbsse	562	300	74.1	0.01	0.38
dk16	1038	150	95.1	0.01	0.55
ex2	480	600	25.9	0.01	1.21
ex3	274	1000	46.2	0.01	0.77
Log	486	200	99.6	0.01	0.11
s832	1090	300	59.6	1.00	2.69
s1488	2234	400	63.48	2.00	9.17
Sand	1622	400	84.2	1.00	3.18
Styr	1734	500	74.1	2.00	6.81

7 Conclusions

We introduced a hierarchical defect oriented fault simulation method for digital systems. As a mathematical model for systematic multi-level solution for fault simulation at three levels of abstraction - RT, gate- and defect levels, decision diagrams are used. The method proposed helps to reduce dramatically the computation cost of test quality analysis in digital systems.

Acknowledgements

The work has been supported by the Estonian Science Foundation grants G3658 and G4300.

References

[1] R. Drechsler, B. Becker: *BDDs. Theory and Implementation*, Kluwer Academic Publishers, 1998, 200 p.

[2] R. Guo, I. Pomeranz and al.: *A Fault Simulation Based Test pattern Generator for Sequential Circuits.* In: Proc. 17th IEEE VLSI Symposium, April 25-29, 1999, California.

[3] D. Krishnaswamy, M. S. Hsiao and al.: *Parallel Genetic Algorithms for Simulation- Based Sequential Test Gene-ration.* In: Proc. IEEE VLSI Design Conference, 1997. pp. 475- 481.

[4] S. Minato: *Binary Decision Diagrams and Applications for VLSI CAD* Kluwer Acad. Publishers, 1996, 141 p.

[5] R. Ubar: *Multi-Valued Simulation of Digital Circuits with Structurally Synthesized Binary Decision Diagrams.* OPA (Overseas Publishers Assotiation) N. V. Gordon and Breach Publishers, Multiple Valued Logic, Vol.4 pp. 141-157, 1998.

[6] R.Ubar: *Vektorielle Alternative Graphen für digitale Systeme*, Nachrichtentechnik/ Elektronik, (31) 1981, H.1, pp. 25-29.

[7] R.Ubar: *Test Synthesis with Alternative Graphs*, IEEE Design and Test of Computers. Spring, 1996, pp.48-59.