Parallel Fault Backtracing for Calculation of Fault Coverage

Raimund Ubar   Sergei Devadze   Jaan Raik   Artur Jutman

Department of Computer Engineering
Tallinn University of Technology
Tallinn, 12618 Estonia
Tel: +372-6202252
Fax: +372-6202253
{raiub, serega, jaan, artur}@pld.ttu.ee

Abstract – A new improved method for calculation of fault coverage with parallel fault backtracing in combinational circuits is proposed. The method is based on structurally synthesized BDDs (SSBDD) which represent gate-level circuits at higher, macro level where macros represent subnetworks of gates. A topological analysis is carried out to generate an efficient optimized model for backtracing of faults to minimize the repeated calculations because of the reconvergent fanouts. The algorithm is equivalent to exact critical path tracing, however, processing the backtrace in parallel for a group of test patterns. Because of the parallelism, higher abstraction level modeling, and optimization of the topological model, the speed of fault simulation was considerably increased. Compared to the state-of-the-art commercial fault simulators the gain in speed was several times.

I. Introduction

Fault simulation is a widely used procedure in the digital circuit design flow. Test generation, fault diagnosis, test set compaction serve as examples of application of fault-free/fault simulators. Accelerating fault simulation would improve all the above-mentioned applications.

Parallel pattern single fault propagation (PPSFP) [1] has been widely used in combinational circuit fault simulation. Many proposed fault simulators incorporate PPSFP with other sophisticated techniques such as test detect [2], critical path tracing [3,4], stem region [5] and dominator concept [4,6]. These techniques have reduced further the simulation time.

Although the concepts of dominators and stem regions achieve substantial speed up in fault simulation, there are still two shortcomings [7]: some regions are repeatedly simulated [1,4], and inefficiency for circuits with high depth [2,5]. Further improvement was reached in [7] where the key idea was to simulate the circuit in the forward levelized order and to prune off unnecessary higher level gates in the early stages.

Another group of methods like deductive [8], concurrent [9] and differential fault simulation [10] are based on logic reasoning rather than simulation. Deductive fault simulation [8] can be faster than parallel fault simulation as their complexities are $O(n^2)$ and $O(n^3)$, respectively [11], where n is the number of logic gates in a circuit. There is no direct comparison between the deductive and concurrent fault simulation techniques, however, it is suspected that the latter is faster than the former because concurrent fault simulation only deals with the "active" parts of the circuit that are affected by faults [12]. Differential fault simulation [10] combines the merits of concurrent fault simulation and single fault propagation techniques, and was shown to be up to twelve times faster than concurrent fault simulation and PPSFP.

The fault analysis methods based on reasoning (deductive, concurrent and differential simulation) are very powerful since they collect all the detectable faults by a single run of the test pattern. What they cannot do is to produce reasoning for many test patterns in parallel.

The Critical Path Tracing (CPT) consists of simulating the fault-free circuit and using the computed signal values for backtracing all sensitized paths from primary outputs to primary inputs, to determine the detected faults. The trace continues until the paths become non-sensitive or end at network primary inputs. Faults on the sensitive (critical) paths are detected by the test. CPT is also based on reasoning, and can process by a single simulation run all the faults, however it works exactly only in the fanout free circuits. A modified critical path tracing technique that is linear time, exact, and complete is proposed in [13]. However, the rule based strategy does not allow simultaneous parallel analysis of many patterns. Parallel critical path tracing in fanout free regions combined with parallel simulation of stem faults was investigated in [14]. In [15] the concept of parallel critical path tracing was generalized for using it beyond the fanout free regions.

Compared to [15] where the topological model for backtracing of faults was created for all the outputs separately as a set of submodels, in this paper, a joint topological model is created for the whole circuit to avoid unnecessary repetition of the same calculation procedures. To achieve high simulation speed, the network of macros rather than gates is used. To make it possible to rise from the lower gate level to the higher macro level, the macros are modeled by structurally synthesized BDDs [16].

The organization of the paper is as follows. In Section 2, a general description of the method for parallel critical path tracing beyond reconverging fanout regions is presented. Section 3 explains the topological pre-analysis of the circuit and proposes a method for optimized building of the simulation model. Experimental results are discussed in Section 4, and Section 5 summarizes the paper.

II. Critical path tracing in a reconverging fanout region

The parallel gate-level critical path tracing inside the fanout free regions (FFR) is straightforward. Let us have an arbitrary macro with a Boolean function:

$$y = F(x_1, \ldots, x_i, x_j, \ldots x_n).$$

If a stuck-at fault is detected at the output y of the macro, then for every input the fault on it is also detected iff

$$\frac{\partial y}{\partial x_j} = 1 \cdot$$

The Boolean derivative is a Boolean formula that can be calculated in parallel for many test patterns. In order to extend the parallel critical path tracing method beyond the fan-out free regions we use the concept of partial Boolean differentials [17].

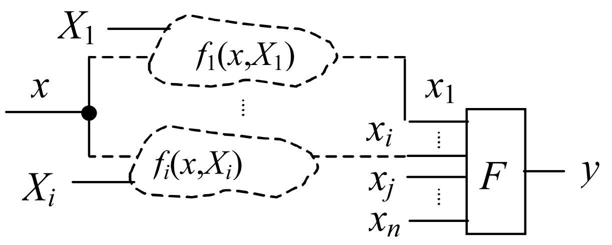

Fig 1. Reconvergent FFR in a circuit

Let us have a converging gate (or macro) F of the converging fanout region depicted in Fig.1, and represented by a Boolean function

$$y = F(x_1, \ldots, x_i, x_j, \ldots x_n).$$

Assume that the input variables $x_1,\ldots,x_i$ of the gate F are connected to the fanout stem x via macro level network represented by functions

$$x_1 = f_1(x,X_1), \ldots, x_i = f_i(x,X_i).$$

Here X_i are subsets of variables.

The function of the full convergent fanout region with a common fanout stem x and output y in Fig.1 can be represented now as a composite Boolean function

$$y = F(f_1(x,X_1), \ldots, f_i(x,X_i), x_j, \ldots x_n). \tag{1}$$

Consider the full Boolean differential [17] of the gate or subcircuit F

$$\begin{aligned} dy = dF = y \oplus F((x_1 \oplus dx_1), \ldots, (x_i \oplus dx_i), \\ (x_j \oplus dx_j), \ldots, (x_n \oplus dx_n)) \end{aligned} \tag{2}$$

Here, by dx we denote the change of the value of x because of the influence of a fault at x. According to the interpretation of dy [17], $dy = 1$ if some erroneous change of the values of arguments of the function (1) causes the change of the value of y, otherwise $dy = 0$.

In the following we show that the dependency of the output variable y of a reconverging FFR on the fanout stem x can be represented by a Boolean differential equation which in its turn can be transformed to a very efficient exact procedure for critical path tracing beyond the convergent FFRs.

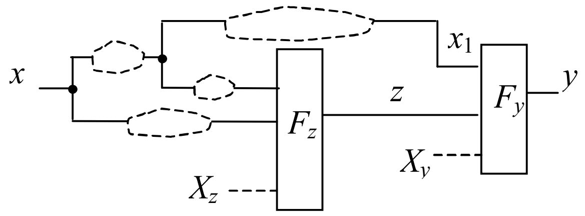

Fig. 2. Nested reconvergencies in a circuit

Theorem 1: If a stuck-at fault is detected by a test pattern at the output y of a subcircuit represented by a Boolean function $y = F(x_1, \ldots, x_i, x_j, \ldots x_n)$, then the fault at the fanout stem x which converges in y at the inputs $x_1, \ldots, x_i$, is also detected iff

$$\frac{\partial y}{\partial x} = y \oplus F((x_1 \oplus \frac{\partial x_1}{\partial x}), \ldots, (x_i \oplus \frac{\partial x_i}{\partial x}), x_j, \ldots, x_n) = 1 \tag{3}$$

The proof of Theorem 1 is given in [15].

From the formula (3), a method results for generalizing the parallel exact critical path tracing beyond the fan-out free regions. All the calculations in (4) can be carried out in parallel because they are Boolean operations.

Consider now a reconverging fanout circuit in Fig.2 represented by a composite function $y = F_y(x,z,X_y)$ where $z = F_z(x,X_z)$, whereas X_y and X_z are the subsets of variables that are not depending on x.

Theorem 2. If a stuck-at fault is detected by a test pattern on the output y of a subcircuit with two nested reconvergencies:

$$y = F_y(x_1,z,X_y) \text{ and } z = F_z(x,X_z),$$

where x_1 is depending, but X_y and X_z are not depending on x, then the fault at the common reconverging fanout stem is also detected iff

$$d_x y = y \oplus F_y(x_1 \oplus \frac{\partial x_1}{\partial x}, z \oplus d_x F_z, X_y) = 1 \tag{4}$$

The proof of Theorem 2 is given in [15].

The formula (4) can be used for calculating the influence of the fault at the common fanout stem x on the output y of the converging fanout region by consecutive calculating partial Boolean differentials, first $d_x F_z$, and then $d_x y$.

It is easy to see that the result of Theorem 2 can be iteratively generalized for an arbitrary configuration of nested reconvergencies. On the other hand, according to Theorem 1, the dependencies between fanout stems and fanout region outputs (reconverging nodes) can be represented by full Boolean differentials and thereafter transformed into fast computable critical path tracing procedures.

III. Generation of the model for fault tracing

The proposed exact parallel path tracing fault analysis is carried out in the following sessions:

- topological pre-analysis of the circuit to create a model for fault tracing along the critical paths and subcircuits;
- parallel simulation of a given set of test patterns to calculate the values of all variables of the circuit;
- parallel fault backtracing on the topological model created during the first session.

The topological pre-analysis to create a model for fault backtracing is carried out only once to serve all the next sessions of the procedure.

Compared to the previous method [15] where the topological model for backtracing of faults was created for all the outputs separately as a disjoint set of submodels, in this paper, a joint topological model is created for the whole circuit to avoid unnecessary repetition of the same calculation procedures during the third session of the fault analysis.

The topological analysis of the circuit consists of three procedures:

- creation of the Reonvergency Graph (RG) of the circuit,
- creation of the whole calculation model of the circuit.

A. Reconvergency graph

The first procedure of the topological analysis is carried out in direction from primary inputs to primary outputs of the circuit. By this procedure, all the fan-out stems and all the reconvergent fan-in nodes of the circuit will be found.

As the result of the procedure, a graph $G = (N,\Gamma)$ is created which represents a skeleton of the circuit. Let N be the set of nodes in G, and Γ the mapping in N, where

- $\Gamma(x) \subset N$ is a set of successor nodes of the node $x \in N$,
- $\Gamma^{-1}(x) \subset N$ is a set of predecessors of the node $x \in N$,
- $\Gamma^{-1}(x_i) \subset N$ is a predecessor node connected to the input i of the node $x \in N$,
- $\Gamma^*(x) \subset N$ is a transitive closure of $\Gamma(x)$, and
- $\Gamma^{*-1}(x) \subset N$ is a transitive closure of $\Gamma^{-1}(x)$.

The nodes of the graph belong to the following subsets:

$$N = FO \cup FI \cup OUT,$$

where FO is the set of all fan-out stems, FI is the set of all reconvergent internal fan-in gate outputs, and OUT is the set of all primary outputs of the circuit. Denote by $RO \subseteq FO$ the subset of all fan-out nodes which reconverge and by $RI \subseteq FI \cup OUT$ the subset of all reconvergent fan-in nodes.

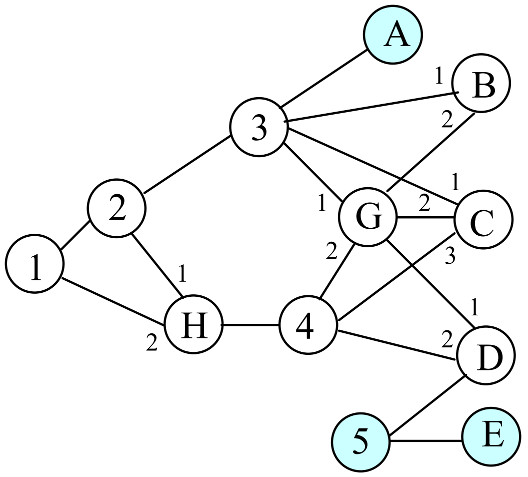

Fig 3. Reconvergency graph of a circuit

To each $x \in RO$ we refer the ordered set of nodes $RI(x) \subseteq RI$, so that for each $y \in RI(x)$ there exist at least two different converging paths from x to y. The ordering of nodes in $RI(x) = (y_1,\dots, y_i, y_j, \dots,y_k)$ is determined by transitive closure: the node y_i is ordered before y_j iff $y_j \in \Gamma^*(y_i)$.

Example 1. Consider in Fig. 3 a reconvergency graph of a circuit which represents a topological skeleton of a circuit with nodes belonging to the subsets in the following way:

$FO = \{1,2,3,4,5,G\}$,
$FI = \{H,G\}$,
$OUT = \{A,B,C,D,E\}$.

The convergency is represented by the subsets:

$RO = \{1,2,3,4\}$,
$RI = \{H,G,B,C,D\}$,
$RI(1) = (H,G,B,C,D)$,
$RI(2) = (G,B,C,D)$,
$RI(3) = (B,C)$,
$RI(4) = (C,D)$.

The procedure of path tracing in direction from inputs to outputs with creating of subsets $RI(n)$ is illustrated in Table I.

TABLE I
Topological analysis from inputs to outputs

#	n	Tracing paths and finding reconvergencies	Creation of subsets $RI(x)$, $x \in RO$
1	H	(1,2), (2,H); (1,H)	$H \Rightarrow RI(1)$
2	G	(2,3), (3,G), (H,4), (4,G)	$G \Rightarrow RI(1)$, $G \Rightarrow RI(2)$
3	B	(3,B), (G,B)	$B \Rightarrow RI(1)$, $B \Rightarrow RI(2)$, $B \Rightarrow RI(3)$
3	C	(3,C), (G,C), (4,C)	$C \Rightarrow RI(1)$, $C \Rightarrow RI(2)$, $C \Rightarrow RI(3)$, $C \Rightarrow RI(4)$
4	D	(4,G), (G,D), (4,D)	$D \Rightarrow RI(1)$, $D \Rightarrow RI(2)$, $D \Rightarrow RI(4)$

An edge (x,y) in the reconvergency graph between two neighbour nodes x and y represents a signal path of the circuit through gates without fanouts. During the topological analysis, all the paths in the circuit are traced, and the found reconvergencies are fixed in the form of subsets $RI(x)$. Before creating a joint topological model of the whole circuit for fault tracing purposes, the next step is to build an ordered set N^* of all the nodes in G. The subset N^* is built by the following procedure:

1. First, all the outputs $x \in OUT$ are included into N^*, and ordered in an arbitrary way.
2. Each node x from the remaining subset of nodes $FO \cup FI \subset N$ will be included into N^* as soon as $\Gamma(x) \subset N^*$ will be valid.
3. The procedure is finished when all the nodes of N are included into N^*.

Example 2. Consider the graph in Fig.3. According to the described procedure, one of the possible ordering can be as follows:

$$N^* = (A,B,C,D,E,G,3,4,H,2,1).$$

The ordered set N^* of nodes of the skeleton graph of the circuit will be the basis for creating the model for fault backtracing in the circuit according to the proposed method. This strategy allows to create an optimized calculation model by the breadth-first processing of nodes.

B. Creation of the optimized calculation model

In the third procedure of the topological analysis we start the creation of the calculation model $\Phi(N^*)$ for the ordered set of nodes N^* of the given topology graph $G = (N,\Gamma)$. Each node $x \in N$ in a graph G represents a node variable in a circuit to be fault simulated in parallel for a set of test patterns $t = (t_1, t_2, \dots t_\tau)$. Denote by

$$x = (x(t_1), x(t_2), \dots x(t_\tau))$$

the vector variable for representing the values of the node x at the set of test patterns t.

The model $\Phi(N^*)$ is a sequence $\Phi(N^*) = (\varphi_1, \varphi_2, \dots , \varphi_q)$ of vector formulas $YX = \varphi_k(y,x)$, $k = 1,2, \dots , q$ for calculating the vector derivatives $\partial y/\partial x$ along the signal path (x,y) in the original circuit, which corresponds to the edge in the graph G between the neighbouring nodes x and y, where $x \in \Gamma^{-1}(y)$.

All $\varphi_k(y,x)$ in $\Phi(N^*)$ are accompanied also with formulas to calculate the detectability of faults in the cones with top on y outside the topology graph G. For two consecutive paths (z,y) and (x,z) where $z \in \Gamma^1(y)$ and $x \in \Gamma^1(z)$ according to the rules of calculating Boolean derivatives, we get the topological formula

$$YX = YZ \wedge ZX \qquad (5)$$

Introduce for $x \in N$ as a *sensitivity vector*

$$X = (X(t_1), X(t_2), \dots X(t_\tau))$$

which expresses the *sensitivity* of outputs of the circuit to the erroneous values of x, so that for each $k = 1,2, \dots, \tau$, $x(t_\tau) = 1$ if

$$\exists\, y \in OUT:\ \partial y(t_\tau)\,/\partial x(t_\tau) = 1,$$

otherwise $x(t_\tau) = 0$. *Sensitivity* vectors are calculated by the formula

$$X = \bigcup_{y \in OUT_x} YX \qquad (6)$$

where OUT_x $\{y \mid y \in OUT, x \in \Gamma^{-1}(y)\}$. Let us call the vectors YX in (6) *partial sensitivity vectors* used for expressing the detectability of faults at particular outputs $y \in OUT$.

If we have calculated the *sensitivity* vector Y for $y \in N$, we can calculate also all *sensitivity* vectors for $z \in \Gamma^1(y)$ by the corresponding formulas YZ. This procedure can be continued for all $x \in \Gamma^{-1}(z)$ in their turn according to (5).

For the fan-in nodes $y \in RI$ with reconvergency from a node $x \in RO$ we specify all the inputs y_k of the node y which have paths to x and create the corresponding formulas Y_kX for calculating the Boolean derivatives $\partial y_k/\partial x$. Let us call the values of Y_kX as *activity vectors* for the paths (x,y_k). The activity vectors are needed for calculating the Boolean derivatives by formulas (3) or (4).

Activity vectors express the sensibilities of selected internal nodes y to faulty signals in respect to other internal nodes $x \in \Gamma^{*-1}(y)$ whereas the *sensitivity* vectors directly express the detectability of faults at the primary outputs of the circuit. Now the

We start to create the formulas φ_k in $\Phi(N^*)$ starting from the output nodes in the beginning of the ordered set N^*. The following procedures will be carried out.

Procedure 1. For each $y \in OUT - RI$ we put into $\Phi(N^*)$ the formulas YX where $x \in \Gamma^{-1}(y)$ for calculating the *sensitivity* vectors X. Each formula is extended by the procedure of calculating derivatives $\partial y/\partial z$ for all the nodes z in the fan-out free cone of the original circuit with the root in y.

Procedure 2. For each $y \in OUT \cap RI$ we create two formulas for each reconverging input y_k with respect to $x \in \Gamma^1(y)$: for calculating the partial sensitivity vectors YX, and for calculating the activity vectors Y_kX. The procedure of using formulas YX and Y_kX can be optimized to avoid repeated calculations along the same path.

Processing of nodes in the topology graph will go on according to the order in N^*. For each $y \notin RO$, the Procedure 2 will be used for calculating both the sensitivity and activity vectors.

Example 3. An iterative use of the Procedure 2 for calculating $\partial t(B_2)/\partial t(H)$ will create the activity formula $B_2H{=}B_2G{\wedge}G4{\wedge}4H$ and for calculating $\partial t(B)/\ \partial t(H)$ the *sensitivity* formula $BH{=}BG{\wedge}G4{\wedge}4H.$

Whenever a node $x \in FO$ where $RI(x) \neq \varnothing$ is reached then for all $y \in RI(x)$ the topological formulas φ_k are created in a form of (3) by Procedure 3 or iteratively, in a form of (4) by Procedure 4.

Procedure 3. Consider a subgraph $G(x,y)$ of the reconvergency graph G, consisting of the nodes

$$N(x,y) = \Gamma^*(x) \cap \Gamma^{*-1}(y) \cup \{x,y\},$$

where $x \in RO$, $y \in RI(x)$, $N(x,y) \cap RI(x) = y$, and $y_j \in \Gamma^*(x)$ for all of the inputs $j = 1,2,\ldots,i$ of the node y. In this case, there are no nested reconvergencies on the paths from x to y, and for calculating $\partial y/\partial x$ the formula (3) can be used. Denote the Boolean expression for calculating $\partial y/\partial x$ by fault backtracing in $G(x,y)$ from y to x based on the formula (3) as

$$YX = \varphi_k(Y) = F\,(Y_1X,\, Y_2X,\, ...,\, Y_iX). \qquad (7)$$

The arguments Y_iX of the function F in (7) represent the formulas for calculating $\partial x_i/\partial x$ in (3). If the path (x,y_i) traverses several nodes in $G(x,y)$ then the component Y_iX in (7) will be calculated according to the rule of the consecutive paths (5).

Example 4. As an example, we represent the partial sensitivity vector $B3$ in Fig.3 for calculating the Boolean derivative $\partial B/\partial 3$ according to (7), by the topological formula

$$B3 = F(B_13,\, B_2G \wedge G3)$$

which according to (3) corresponds to the expression

$$\frac{\partial B}{\partial 3} = B \oplus F((B_1 \oplus \frac{\partial B_1}{\partial 3}),(B_2 \oplus \frac{\partial B_2}{\partial G}\frac{\partial G}{\partial 3}))\,.$$

Procedure 4. Consider a subgraph $G(x,z)$ in $G(x,y)$, $z \in \Gamma^*(x)$, $z \in \Gamma^{*-1}(y)$, consisting of nodes

$$N(x,z) = \Gamma^*(x) \cap \Gamma^{*\,-1}(z) \cup \{x,\, z\}$$

where

$$x \in RO;\ z,y \in RI(x),\ N(x,z) \cap RI(x) = z, \text{ and } z_j \in \Gamma^*(x)$$

for all of the inputs $j = 1,2,\ldots,i$ of the node z.
Let $N(z,y_k) \cap RI(x) = \varnothing$, i.e. none of the nodes $G(z,y)$ except y belongs to $RI(x)$. According to (4) and (5) we calculate $\partial y_k/\partial x$, as

$$Y_kX = Y_kZ \wedge F\,(Z_1X,\, Z_2X,\, ...,\, Z_iX),$$

and $\partial y/\partial x$ according to the Procedure 3.

If the subgraph $G(x,z)$ has further nested reconvergencies we have to repeat Procedure 4 iteratively.

At each current node $y \in N^*$ the formulas $\varphi_k(y,x)$ for all $x \in \Gamma^{*\,-1}(y)$ are created as Boolean expressions along the paths (x,y) in the circuit.

Example 5. The full calculation model $\Phi(N^*)$ for the topology graph in Fig.3 is presented in Table II. The formulas marked by "*" are sensitivity vectors, whereas all other represent activity vectors.

IV. Experimental results

Table III presents the fault simulation results for the circuits of ISCAS'85 benchmark family and combinational versions of selected circuits of ISCAS'89 family (column 1) to compare different fault simulators: exact critical path tracing [13] (column 2), two state-of-the-art commercial fault simulators from major CAD vendors (columns 3, 4), our previous parallel critical path tracing method [15] (column 5), and the proposed new method (column 6).

TABLE II
Topological analysis from outputs to inputs

Simulation from outputs		32	$G_22 = G_2H \wedge H2$
1	A3*	33	$G2 = F(G_12, G_22)$
2	B3*, B_13	34	$A2^* = A3 \wedge 32$
3	BG*, B_2G	35	$B_12 = B_13 \wedge 32$
4	C3*, C_13	36	$B_22 = B_2G \wedge G2$
5	CG*, C_2G	37	$B2^* = F(B_12, B_22)$
6	C4*, C_34	38	$C_12 = C_13 \wedge 32$
7	DG*, D_1G	39	$C_22 = C_2G \wedge G2$
8	D4*, D_24	40	$C_3H = C_34 \wedge 4H$
9	D5*	41	$C_32 = C_3H \wedge H2$
10	E5*	42	$C2^* = F(C_12, C_22, C_32)$
Simulation from G		43	$D_12 = D_13 \wedge 32$
11	$G^* = \cup(BG, CG, DG)$	44	$D_2H = D_24 \wedge 4H$
12	$G3^* = G \wedge G3$, G_13	45	$D_22 = D_2H \wedge H2$
13	$G4^* = G \wedge G4$, G_24	46	$D2^* = F(D_12, D_22)$
Simulation for 3		**Simulation from 2**	
14	$B_23 = B_2G \wedge G3$	47	$2^* = \cup(A2, B2, C2, D2)$
15	$B3^* = F(B_13, B_23)$	48	$21^* = 21 \wedge 2$
16	$C_23 = C_2G \wedge G3$	**Simulation for 1**	
17	$C3^* = F(C_13, C_23)$	49	$H_11 = H_12 \wedge 21$
18	$D3^* = DG \wedge G3$	50	$H1 = F(H_11, H_21)$
Simulation from 3		51	$G_11 = G_12 \wedge 21$
19	$3^* = \cup(A3, B3, C3, D3)$	52	$G_21 = G_2H \wedge H1$
20	$32^* = 32 \wedge 3$	53	$G1 = F(G_11, G_21)$
Simulation for 4		54	$A1^* = A2 \wedge 21$
21	$B_24 = B_2G \wedge G4$, B4*	55	$B_11 = B_12 \wedge 21$
22	$C_24 = C_2G \wedge G4$	56	$B_21 = B_2G \wedge G1$
23	$C4^* = F(C_24, C_34)$	57	$B1^* = F(B_11, B_21)$
24	$D_14 = D_1G \wedge G4$	58	$C_11 = C_12 \wedge 21$
25	$D4^* = F(D_14, D_24)$	59	$C_21 = C_2G \wedge G1$
Simulation from 4		60	$C_31 = C_3H \wedge H1$
26	$4^* = \cup(B_24, C4, D4)$	61	$C1^* = F(C_11, C_21, C_31)$
27	$H^* = 4H \wedge 4$, 4H	62	$D_11 = D_1G \wedge G1$
Simulation from H		63	$D_21 = D_2H \wedge H1$
28	$H2^* = H2 \wedge H$, H_12	64	$D1^* = F(D_11, D_21)$
29	$H1^* = H1 \wedge H$, H_21	**Simulation from 1**	
Simulation for 2		65	$1^* = \cup(A1, B1, C1, D1)$
30	$G_12 = G_13 \wedge 32$		
31	$G_2H = G_24 \wedge 4H$		

The simulation times were calculated for the sets of random 10000 patterns. The time needed for topology analysis is included and is negligible compared to the gain in speed which is 2.5 times compared to the previous best method. Compared to the commercial tools C1 and C2, the average gain in speed is 41.1 and 5.8 times, respectively.

All the experiments were run on a 366 MHz SUN Ultra60 server using SunOS 5.8 operating system except the experiments for the known exact critical path fault simulator where the data are taken from [13]. The experiments in [13] were run on a 2.8 GHz Pentium 4 computer with Windows XP.

TABLE III

Comparison with previous tools

Circuit	Fault simulation time, s				
	[13]	C1	C2	[15]	New
c432	70	13.0	3.8	1.2	0.64
c499	190	3.0	2.8	2.7	0.98
c880	140	26.0	4.0	1.2	0.65
c1355	640	44.0	9.0	5.0	1.33
c1908	640	53.0	15.6	5.0	1.61
c2670	560	104.0	11.0	3.8	1.99
c3540	770	191.0	37.4	9.3	4.43
c5315	1270	246.0	28.6	6.7	3.41
c6288	4280	1159.0	139.2	134.2	46.39
c7552	1480	378.0	40.5	15.0	5.44
s4863_C	N/A	353.0	30.0	N/A	5.10
s5378_C	N/A	170.0	15.9	N/A	4.17
s6669_C	N/A	416.0	40.8	N/A	7.94
s9234_C	N/A	248.0	26.7	N/A	6.72
s13207_C	N/A	332.0	27.2	N/A	10.18
s15850_C	N/A	470.0	57.8	N/A	13.75
s35932_C	N/A	1751.0	111.6	N/A	36.22
s38417_C	N/A	1351.0	157.0	N/A	39.05
s38584_C	N/A	1399.0	115.3	N/A	34.97
Average speed gain	258.9	41.1	5.8	2.5	1

The last column clearly shows the gain in speed that was achieved by optimizing the topological model by the method proposed in the paper.

V. Conclusions

A method has been proposed for macro-level parallel exact critical path fault tracing for combinational circuits or scan-path designs. Differently from the known critical path tracing approaches, a method is proposed to create ordered topological model for parallel fault backtracing. The model is based on the full Boolean differential, which allows generalization of the parallel critical path fault tracing beyond the reconvergent fanout stems. Considerable increase in the speed of fault simulation was shown compared to the two state-of-the-art commercial fault simulators, in average 5.8 and 41.1 times, respectively.

The proposed method allows creation of the full fault table for a group of test patterns with a single pass of the algorithm.

Acknowledgements

The work has been supported by Estonian Science Foundation grants 7483, 7068, EC 6th FP IST project VERTIGO, Estonian IT Foundation (EITSA) and Enterprise Estonia.

References

[1] J.A. Waicukauski, et.al., “Fault Simulation of Structured VLSI”, *VLSI Systems Design*, Vol.6, 12, pp.20-32, 1985.

[2] B.Underwood, J.Ferguson, “The Parallel Test Detect Fault Simulation Algorithm”, *ITC*, pp.712-717, 1989.

[3] M. Abramovici, P.R. Menon, D.T. Miller, “Critical Path Tracing – An Alternative to Fault Simulation”, *DAC*, pp.2-5, 1987.

[4] K.J. Antreich, M.H. Schulz, “Accelerated Fault Simulation and fault grading in combinational circuits”, *IEEE Trans. on CAD*, Vol. 6, No. 5, pp.704-712, 1987.

[5] F. Maamari, J. Rajski, “A method of fault simulation based on stem region”, *IEEE Trans. CAD*, Vol.9, No.2, pp.212-220, 1990.

[6] D. Harel, R. Sheng, J. Udell, “Efficient Single Fault Propagation in Combinational Circuits”, *Int. Conf. on CAD*, pp.2-5, 1987.

[7] H.K. Lee, D.S. Ha, “An efficient, forward fault simulation algorithm based on the parallel pattern single fault propagation”, *Proc. ITC*, pp. 946-955, 1991.

[8] D.B. Armstrong, “A deductive method for simulating faults in logic circuits”, *IEEE Trans. Comp*., C-21(5), 464-471, 1972.

[9] E.G. Ulrich, T. Baker, “Concurrent simulator of nearly identical digital networks”, *IEEE Trans. on Comp*., 7(4), pp.39-44, 1974.

[10] W.T. Cheng, M.L. Yu, “Differential fault simulation: a fast method using minimal memory”, *DAC*, pp.424-428, 1989.

[11] P. Goel, “Test Generation Cost Analysis and Projections”, *Proc. DAC*, pp.77-84, 1980.

[12] L.-T. Wang, Ch.-W. Wu, X. Wen, “VLSI Test Principles and Architectures. Design for Testability”, *Elsevier*, 2006, 777 p.

[13] L. Wu, D.M.H. Walker “A Fast Algorithm for Critical Path Tracing in VLSI”, *Int. Symp. on Defect and Fault Tolerance in VLSI Systems*, Oct. 2005, pp.178-186.

[14] S. Devadze, J. Raik, A. Jutman, R. Ubar, “Fault Simulation with Parallel Critical Path Tracing for Combinational Circuits Using SSBDDs”, *7th IEEE LATW*, 2006, pp.97-102

[15] R.Ubar, S.Devadze, J.Raik, A.Jutman. Ultra Fast Parallel Fault Analysis on Structural BDDs. 12th IEEE European Test Symposium – ETS 2007, Freiburg, Germany, May 20-24, 2007.

[16] Jutman, A. Peder, J. Raik, M. Tombak, R. Ubar, “Structurally synthesized binary decision diagrams”, *6th Int.-l Workshop on Boolean Problems*, pp.271-278, Freiberg, Sept. 23-24, 2004.

[17] Thayse, “Boolean Calculus of Differences”, *Springer Verlag*, 1981.