Exact Parallel Critical Path Fault Tracing to Speed-Up Fault Simulation in Sequential Circuits

Jaak Kõusaar, Sergei Kostin, Raimund Ubar, Sergei Devadze, and Jaan Raik

***Abstract*—We propose a new method to speed up stuck-at fault simulation for sequential circuits. The method combines exact parallel critical path tracing of faults, used so far only for combinational circuits, with traditional fault simulation in sequential circuits. For that purpose, formulas are developed for classification of faults into two classes: the faults eligible for parallel critical path tracing, and the faults which have to be simulated over the global feedbacks in the circuit by traditional methods. Combining these two approaches in fault simulation – the combinational and sequential simulation concepts – allows dramatic speed-up of fault simulation in sequential circuits, which is demonstrated by experimental results.**

***Index Terms*—sequential circuits, fault simulation, stuck-at-faults, critical path tracing**

I. Introduction

FAULT simulation in digital circuits is one of the most important tasks in the fields of test and fault tolerance. By fault simulation the fault coverage of testing can be calculated, different data structures like fault dictionaries for purposes of fault diagnosis can be created, the quality of fault tolerance can be evaluated, and many other design analysis related tasks can be solved. Whereas the fault simulators of combinational circuits are fast and efficient, the fault simulation in sequential circuits is very slow due to the need of sequential fault propagation analysis over the global feedback structures.

Therefore, accelerating of fault simulation procedures in sequential circuits would have a strong impact to all of the mentioned applications.

In the past, a lot of different concepts and methods have been proposed for fault simulation in combinational circuits based on the idea of parallel pattern single fault propagation (PPSFP) [1]. This concept has been combined with other sophisticated simulation 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.

Another trend based on the fault reasoning (deductive [7], concurrent [8] and differential simulation [9]) is proved to be as well very powerful, since these methods allow to collect all detectable faults by a single run of the given test pattern. What they cannot do, is to produce reasoning for many test patterns in parallel.

The original critical path tracing method [3, 4] eliminates explicit fault simulation for faults within Fan-out-Free Regions (FFR). However, the explicit simulation of faults at fan-outs is still needed.

A modified critical path tracing technique that excludes fault simulation for fan-out stems, and includes a system of rules to check the exactness of critical path tracing beyond the FFRs, and which is linear in time, was proposed in [10]. However, the rule based strategy does not allow parallel analysis and rule check for many patterns simultaneously. This drawback was removed in [11] by introducing a novel concept of Parallel Pattern Exact Critical Path Tracing (PPECPT) which can be applied efficiently also beyond FFRs. In [12], the same method was for a general class of X-faults, and in [13], additional gain from parallel computing to speed-up the method was achieved by implementing it in the multi-core computing environment. The main idea of the PPECPT method is in compiling a dedicated compact computing model through the circuit topology analysis, which allows exact critical path tracing in parallel for a bunch of test patterns throughout the full circuit and not only inside FFRs.

Unfortunately, for sequential circuits the critical path tracing of faults is not possible, due to the sequential (time related) dependence of signals in the circuit, which in general case leads to the need of critical path tracing over many clock cycles, and hence to the exponential grow of calculations needed.

In this paper, we develop a new method, which allows partial parallel critical path tracing of faults in sequential circuits, combined with traditional separate fault simulation and, as the result, the speed of fault simulation in sequential circuits can be drastically increased.

We consider in this paper the class of stuck-at-faults (SAF), however, the results can be extended also to other fault classes like conditional SAF, transition delays, and *X*-faults, in a similar way, as it was shown in [12] for the case of combinational circuits.

The rest of the paper is organized as follows. In Section 2, we present a short overview of the exact parallel critical path tracing in combinational circuits for analytical reasoning of faults. In Section 3, we show why the direct exact parallel critical path tracing of faults is not possible in sequential circuits. Section 4 describes the concept of combining

J. Kõusaar, S. Kostin, R. Ubar, S. Devadze and J. Raik are with the Department of Computer Engineering, TTU, Ehitajate tee 5, 19086 Tallinn, Estonia (e-mails: jaak.kousaar@gmail.com, raiub@pld.ttu.ee, sergei.kostin@gmail.com, serega@pld.ttu.ee, jaan@pld.ttu.ee)

ISSN 2080-8755 *Copyright © 2018 by Department of Microelectronics & Computer Science, Lodz University of Technology*

analytical critical path based fault reasoning with traditional methods of sequential fault simulation. Section 5 presents the algorithm of separating the faults into two parts, and describes the fault simulation method as a whole. In Section 6 we give the formulas of estimating the speed-up of the method and develop the higher bounds of speed-up. Section 7 presents experimental data, Section 8 provides a discussion on the results of experimental research, and Section 9 concludes the paper.

II. Overview of the Method of Exact Parallel Critical Path Tracing of Faults

Consider a combinational circuit as a network of fan-out free regions (FFRs) [14] where each of the FFRs can be represented as a Boolean function $y = F(x_1, x_2, \dots x_n) = F(X)$, with $X = (x_1, x_2, \dots x_n)$ as input vector.

The fault simulation for a FFR, according to the traditional critical path tracing, is equivalent to calculation of Boolean derivatives: if $\partial y/\partial x = 1$ then the fault is propagated from x to y. This check can be performed in parallel for a given subset of test patterns as illustrated in Fig. 1.

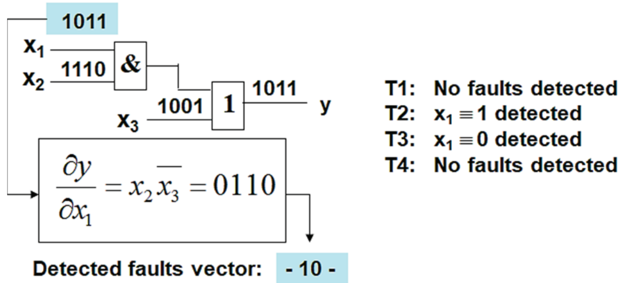
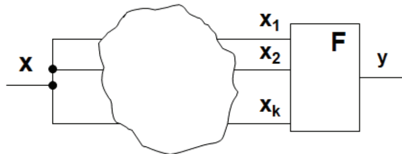


Fig. 1. Parallel fault simulation for an FFR at the gate level

In order to extend the parallel critical path tracing beyond an FFR described by a function $y = F(x_1, x_2, \dots, x_k)$, where the variable x_i, $i = 1,2,\dots,k$, represent the end nodes of paths which fan out from a joint node x, we can use the formula depicted in Fig. 2.



$$\frac{\partial y}{\partial x} = y \oplus F((x_1 \oplus \frac{\partial x_1}{\partial x}), \dots, (x_k \oplus \frac{\partial x_k}{\partial x}))$$

Fig. 2. Parallel fault simulation beyond the FFR

For generalization of this approach to any arbitrary network of FFRs, we can use the concept of Boolean differentials [15, 16], as shown in the following.

Consider the full Boolean differential for an FFR with a function $y = F(X)$ as

$$dy = y \oplus F((x_1 \oplus dx_1), \dots (x_n \oplus dx_n)) = y \oplus F(X \oplus dX) \quad (1)$$

Here, by dx we denote the change of the value of x because of a fault at x, whereas $dy = 1$ corresponds to the case when an erroneous change of the values of arguments of the function (1) due to a fault causes the change of the value of y. Otherwise, $dy = 0$.

Let x be a fan-out variable with branches which converge in a FFR $y = F(X)$ at the inputs denoted by a subset $X' \subset X$. In [12] it was shown that from the expression (1), the following relationship can be derived:

$$\frac{\partial y}{\partial x} = y \oplus F((x_1 \oplus \frac{\partial x_1}{\partial x}), \dots, (x_n \oplus \frac{\partial x_n}{\partial x})) = y \oplus F(X \oplus \frac{\partial X}{\partial x})$$

or, in the vector form as

$$\frac{\partial y}{\partial x} = y \oplus F(X' \oplus \frac{\partial X'}{\partial x}, X'') \quad (2)$$

where $X' \subset X$ is the sub-vector of variables which depends on x, and $X'' = X \setminus X'$ is the sub-vector of variables which do not depend on x.

The formula (2) can be used for calculating the impact of the fault at the fan-out stem x on the output y of the given convergent fan-out region by consecutive calculating of Boolean derivatives over the related FFR chains starting from x up to y. For that purpose, for each fan-out stem, which has convergence, the corresponding formulas like (2) should be constructed for all FFRs involved in the convergence. In the case of nested convergences, the formulas will have as well a nested structure. All these formulas will constitute a compiled partially ordered computation model for fault simulation, which can be composed by the topological analysis of the circuit [12].

Since the formulas of the compiled simulation model are Boolean, all computations on this model can be carried out in parallel for a bunch of test patterns. This computation procedure corresponds to parallel critical path tracing. The main advantage of this method is the possibility to calculate the full subset of faults, detectable by the simulated bunch of test patterns, by a single run through the compiled computation model.

III. The Problems Which Make Parallel Critical Path Tracing in Sequential Circuits Difficult

The described method for critical path tracing, both for single and parallel simulation of multiple patterns is possible only for combinational circuits, however, taking into account that the higher is the number of converging fan-out stems, the more complex is the computation model, and the slower will be the simulation run.

Consider in the following the reasons why exact critical path tracing of faults is practically impossible in sequential circuits.

The substantial problem of fault simulation in sequential circuits lies in the fact that the same fault can influence on a particular component in different time frames. This fact excludes the possibility of exploiting the powerful critical path tracing based method, explained in Chapters 2, and which has been developed for using in fault simulation in combinational circuits. The reason is in the exponential explosion of the number of nested and intersected re-converging fan-out regions over different time frames. However, this problem, as it will be shown, can be resolved if there will be a possibility to detect the fault in the first occasion when it has propagated up to any observable point without cycling in global loops.

There are two reasons why the same fault can be propagated to the same component of the circuit along different paths in different time frames:

(1) the case of a global feedback loop which includes a component to which the same fault can have influence via different time frames;

(2) the case of a sequential re-converging fan-out, where the fault may propagate to the same component from the same fan-out stem via different time frames.

The first case (1) is illustrated in Fig. 3. The circuit represented in Fig. 3 consists of three registers and four combinational blocks connected by busses. Assume, there is a fault Q in the sequential circuit on one of the outputs of the register R_2. Assume as well that the fault is propagating at the given test sequence to the primary output Y of the circuit by two successive test patterns. The first pattern propagates the fault in the clock cycle *t*-1 through combinational blocks F_3, F_4 and F_2 (black bold lines) to the register R_3, and the second test pattern propagates the erroneous signal from the register R_3 in the next clock cycle *t* via block F_4 to the primary output Y (red bold lines).

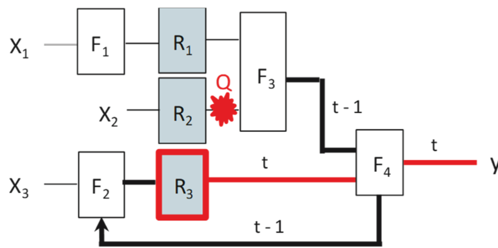


Fig. 3. Critical path tracing of a fault in a sequential circuit (the case (1))

To better understand the mechanism of critical path tracing of faults in the sequential circuit, let us unroll the sequence of two test patterns into two time frames *t*-1 and *t* of the iterative logic array of the circuit presented in Fig. 4.

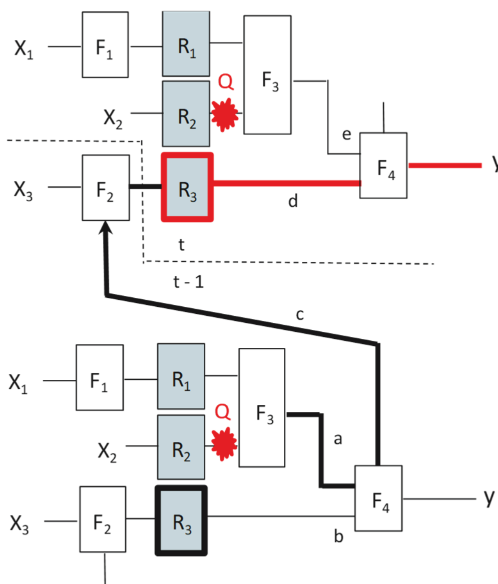


Fig. 4. Critical path tracing of a fault in a sequential circuit over two successive time frames (the case (1))

In the iterative logic array in Fig.4, the single fault Q is propagating, and in such a way its impact is re-converging on the inputs of the block F4 via two different paths: the path (Q, *a*, *c*, *d*) activated in the time frame *t*-1, and the path (Q, *e*) activated in time frame *t*.

Note that the critical path tracing of faults at the given test pattern is based on reasoning of the simulated correct signals in the circuit produced by the pattern.

Using only the correct signals makes it possible to determine in parallel all the faults which may propagate along the activated critical paths to the observable primary outputs. In Fig.4, there is activated a two cycle critical path along lines a, c (during cycle *t*-1), and d (during cycle *t*). The conditions of propagating the faults from *a* to *c* are determined by signals on the bus *b* at the time *t*-1, and the conditions of propagating the faults from *d* to the output Y are determined by signals on the bus *e* at the time *t*.

The critical path tracing of faults is carried out from the observable primary outputs towards the inputs. Starting from the output Y, the first step of the critical path tracing is to determine if the signals on the bus *e* are proper to propagate the faults from the line *d* through the block F_4 to the output Y. Since the fault Q under investigation should be considered in all time frames of the iterative array, the fault propagation conditions of the bus e are essentially depending also on the impact of the fault Q. However, this contradicts to the mechanism of critical path tracing of faults which must be carried out on the basis of using only the correct signals (also on the outputs of the register R_2).

The second case (2) of a sequential re-converging fan-out is illustrated in Fig. 5. In this case, the single fault Q is propagating to a converging point not along a global feedback, but rather along two branching paths activated at different time frames.

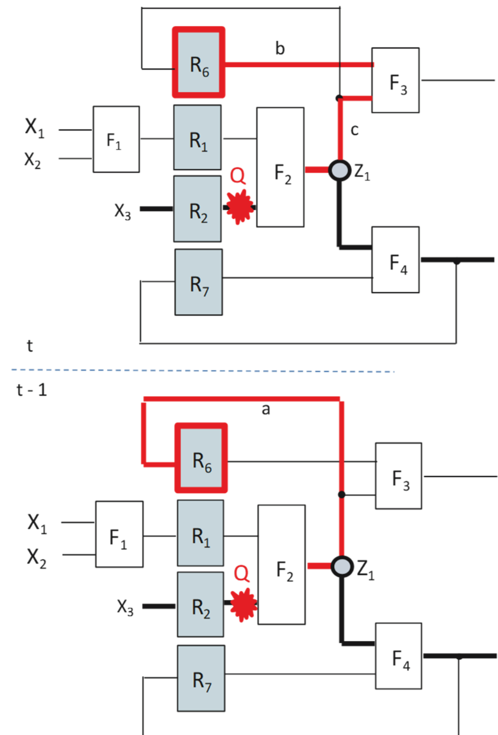


Fig. 5. Critical path tracing of a fault in a sequential circuit over two successive time frames (the case (2))

Assume, there is a fault Q in the sequential circuit on the output of the register R_1. Let us unroll the sequence of two test patterns into two time frames *t*-1 and *t* as shown in Fig. 5. The first branch in Fig. 5 consists of the path (Q, *a*, R_6) activated at time *t*-1, and of the path (R_6, *b*, F_3), activated at time *t*. The second re-converging branch is formed by the path (F_2, *c*, F_3), which is activated at time *t*. Both branches, propagating the impact of the same fault Q, are converging on the inputs of the block F_3 at the same clock cycle *t*.

For this case, we can show in the similar way as in the case (1) that the critical path tracing, using only the correct signals in the circuit, is not possible. As an example, in Fig. 5, in the time frame *t*, the faults on the line *c* cannot be back-traced, since the signal *b* at this time frame is not correct.

In such a way, the examples discussed on the basis of Fig. 4 and Fig. 5 demonstrate that in sequential circuits, each fault must be simulated separately, and the classical fault independent critical path tracing of faults used in combinational circuits is not possible.

In the next Section it will be shown how the critical path tracing method can be used still at least for one part of faults of sequential circuits which will help to considerably decrease the total amount of time needed for fault simulation in sequential circuits.

IV. Converting Sequential Fault Simulation into Combinational Critical Path Fault Tracing

Consider a sequential circuit in Fig. 6, partitioned into the combinational part, consisting of sub-circuits A, B, and C with related fault sets S_A, S_B, S_C, and a sequential part consisting of flip-flops FF. Let $S = S_A \cup S_B \cup S_C$ be the full set of faults to be simulated in the circuit.

The faults in S_A will always propagate directly to the primary output OUT_A, and never to FFs. Hence, the faults in S_A can be handled in all time frames of the given test sequence independently and they can be simulated by the parallel critical path tracing (PCPT) approach. On the contrary, the effect of the faults in S_B can never be observed directly at OUT_A without propagating one or more times through the loop of the feedback FF, and hence, these faults are tending to be self-masked (similarly to the example in Fig.3 and Fig.4). Consequently, the faults S_B cannot be analysed using PCPT, and they have to be processed by slow sequential fault simulation (SSFS) used traditionally in sequential circuits.

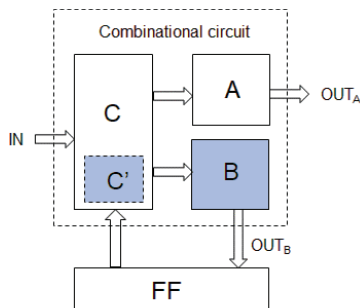


Fig. 6. A sequential circuit partitioned according to fault types

The faults in S_C represent a special case. When a test sequence is applied to the circuit, the same faults in S_C may sometimes propagate directly to the primary output OUT_A, whereas in other times they may propagate first to OUT_B, and only then, after in the feedback loop, they may be observed at the primary output OUT_A. Let the faults in S_C, which propagate first to the pseudo-output OUT_B, form a subset $S_{C'} \subseteq S_C$. The described ambivalence of the faults in S_C makes of them the key problem of the new fault simulation method.

The set of faults which propagate directly to the primary output OUT_A, without circulating in the feedback loop, can be represented as

$$S(A) = S_A \cup (S_C - S_C')$$

These faults use for propagating to the observable output OUT_A a single time frame, and hence, can be simulated in the same way as in the combinational circuits using the very fast PCPT simulation approach. Note, the PCPT is carried out using the correct signals of the circuit, which is the prerequisite of the possibility of parallel analysing concurrently all faults in the circuit.

The rest of faults

$$S(B) = S_B \cup S_C' = S - S(A)$$

need for propagating to the observable output OUT_A at least two (or more) time frames. This means that each fault, starting from the second time frame in its propagation trace up to OUT_A, will distort the states of signals in the circuit in all time frames in its own way, making the concurrent analysis of different faults not possible. Hence, all the faults in *S*(B) have to be simulated separately by conventional methods used for sequential circuits, which is a very slow procedure.

Since the conventional fault simulators for sequential circuits are targeting by this slow approach all faults *S* in the circuit, it would be promising to extract from *S* the subset of faults *S*(A) which can be simulated using fast PCPT. In this way, it would be possible to speed up dramatically the full fault simulation procedure.

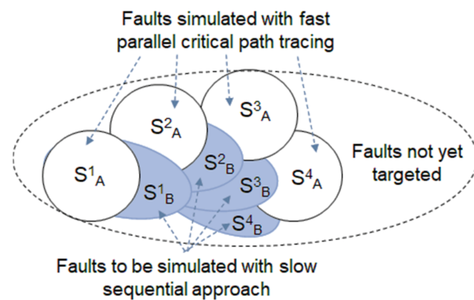


Fig. 7. Simulation based fault classification

We propose here the following simulation based method for classification of all faults into the two subsets *S*(A) and *S*(B) where $S(A) \cap S(B) = \emptyset$.

The difficulty is related to the fault set $S_{C'}$, because the content of $S_{C'}$ is strongly dependent on the test sequence. In other words, it is not possible to predefine $S_{C'}$ before fault simulation, and the content of $S_{C'}$ can be defined only "on-line" during the process of fault simulation. This additional payload may slow down the full procedure.

The method, illustrated in Fig.7, is based on applying critical path tracing for the patterns in sequence, one by one, separately for the outputs OUT_A and OUT_B. As the result of the method, we will hit two targets: first, we find immediately using critical path tracing a part of detected faults $S(A)$, and second, we extract the subset of faults $S(B)$ which will need separate slow fault simulation.

We start from the first pattern of the test sequence, and calculate the sets S^1_A and S^1_B of faults detected on the outputs OUT_A and OUT_B, respectively. The faults of S^1_A can be included immediately into $S(A)$ of detected faults by this first pattern, $S(A_1) = S^1_A$, whereas the faults S^1_B we include into $S(B_1)$.

Next, we find for the second pattern of the test sequence the sets of detected faults S^2_A and S^2_B. The sets $S(A)$ and $S(B)$ can be then updated in the following way:

$$S(A) = S(A_2) = S(A_1) \cup (S^2_A - S(B_1)),$$

$$S(B) = S(B_2) = S(B_1) \cup (S^2_B - S(A_2)).$$

Following now the example in Fig. 5, we can express the state of the fault simulation in general case after the k-th pattern of the test sequence:

$$S(A) = S(A_k) = S(A_{k-1}) \cup (S^k_A - S(B_{k-1})) \quad (3)$$

$$S(B) = S(B_k) = S(B_{k-1}) \cup (S^k_B - S(A_k)). \quad (4)$$

The procedure ends if the k-th pattern will be the last pattern of the test sequence. $S(A)$ will represent the set of faults simulated and detected very fast by the critical path tracing method, and $S(B)$ will represent the faults which need additional fault simulation by any conventional fault simulator for sequential circuits.

V. Combining Parallel Critical Path Tracing with Sequential Fault Simulation

In the previous section, we considered a test for a sequential circuit as a single test sequence where all test patterns, consisting of primary input patterns and pseudo-input patterns (the output values of flip-flops), are strongly depending on each other due to the feedback loop. This fact allowed us to exploit the power of critical path tracing only for single patterns. In Section II we described the method of parallel critical path tracing concurrently for many patterns, which was developed for combinational circuits. This parallelism is possible due to the independency of test patterns in case of combinational circuits.

In the following, we develop a method for parallel critical path tracing also for sequential circuits, which exploits the independences between the test segments in the full test sequence. We assume that each segment will consist of the initialisation of the flip-flops, and the subsequent patterns will serve for fault sensitization and fault propagation to the primary output OUT_A.

Consider a test sequence consisting of k test segments, where each i-th test segment consists of m_i test patterns $TS_i = (T_{i,1}, T_{i,1}, \dots T_{i,mi})$. Since all test segments are independent, then also the first patterns $(T_{1,1}, T_{2,1}, \dots T_{k,1})$ in all k segments, and in a similar way, the second patterns $(T_{1,2}, T_{2,2}, \dots T_{k,2})$ in all k segments, etc., can be considered together as a set of m packages $\{TPP_i\}$ of independent test patterns. The lengths of the packages, in general case, may be different, and the number of packages m is equal to the number of the patterns in the longest test sequence.

An example of converting the initial test sequence into a set of packages of independent test patterns is depicted in Fig. 8.

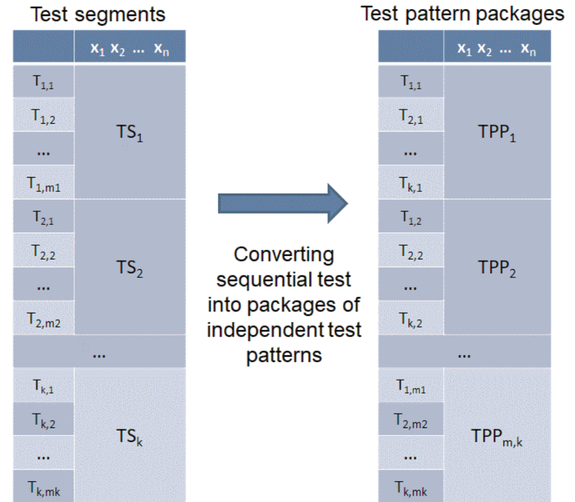


Fig. 8. Converting the set of test segments into the set of independent test pattern packages

For each package $TPP_j = (T_{1,j}, T_{2,j}, \dots T_{k,j}), j = 1,2,\dots, m$, of test patterns, parallel critical path tracing can be applied concurrently for all patterns in the package. As the result of this procedure, for each test pattern $T_{i,j} \in TPP_j$, the fault sets $S^{i,j}_A$ and $S^{i,j}_B$ will be calculated, and the detected fault sets (fault covers) will be calculated in the following way.

$$S^j_A = S^{1,j}_A \cup S^{2,j}_A \cup \dots \cup S^{k,j}_A$$

$$S^j_B = S^{1,ja}_B \cup S^{2,j}_B \cup \dots \cup S^{k,j}_B$$

An example of a fault table created by parallel critical path tracing for the given set of k packages of independent test patterns is depicted in Fig. 9.

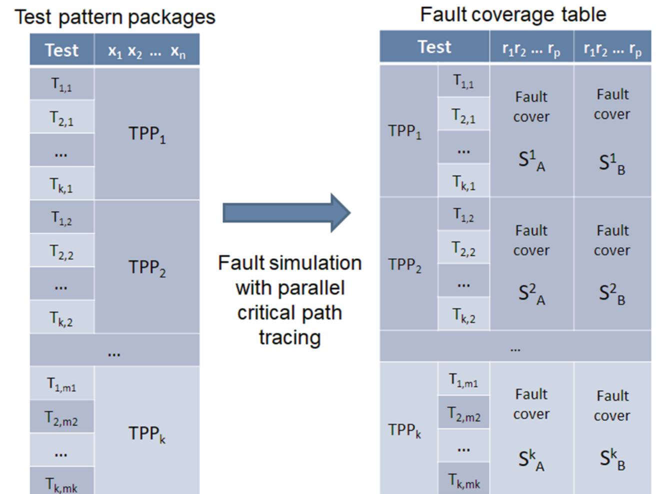


Fig. 9. Results of parallel critical path tracing of sequential test

Based on the fault sets S^j_A and S^j_B calculated by critical path tracing, the sets of faults $S(A_k)$ and $S(B_k)$ can be calculated similarly to the formulas (3) and (4), respectively.

The full procedure of the method of combining parallel critical path tracing with sequential faults simulation for sequential circuits can be presented as follows.

Algorithm 1

1: Convert the set of test segments into test packages of independent test patterns (see Fig. 8)
2: **for** each test package TPP_j
3: Apply parallel critical path tracing to calculate the fault sets S^j_A and S^j_B
4: **end for**
5: **for** $k = 1,2,..,m$ (m is the number of test packages, $S(A_0) = S(B_0) = \varnothing$)
6: $S(A_k) = S(A_{k-1}) \cup (S^k_A - S(B_{k-1}))$
7: $S(B_k) = S(B_{k-1}) \cup (S^k_B - S(A_k))$
8: **end for**
9: **return** $S(A) = S(A_m)$, $S(B) = S(B_m)$,

The set of S(A) represents the faults detectable by the given test, calculated by parallel critical path tracing (PCPT). The set of S(B) represents the faults whose detectability is not possible to determine by PCPT. The faults in S(B) have to be simulated by conventional fault simulation methods used for sequential circuits.

VI. Estimating the Speed-up of the Proposed Method

Let us calculate now the characteristics of the timing analysis and of possible speed-up of the proposed method. Let us use the following notations:

n – is the number of faults in the circuit;
n_B – is the number of faults in S(B) to be simulated by slow sequential fault simulator;
t_{anal} – the total time needed for critical path tracing and fault analysis consisting of two parts $t_{anal} = t_{CP} + t_{CL}$, where
t_{CP} – time needed for critical path tracing, and
t_{CL} – time needed for fault classification.
t_{OLD} – time needed for fault simulation of all faults in a sequential circuit by conventional simulation;
t_{seq} – the average time of sequential simulation of a fault $t_{seq} = t_{OLD}/n$.

The total time needed for the proposed method, which combines the critical path tracing with conventional sequential fault simulation can be calculated as

$$t_{NEW} = t_{anal} + (t_{seq} \times n_B) = t_{CP} + t_{CL} + (t_{OLD} \times n_B) \quad (5)$$

The speed-up of using the proposed method compared to the sequential approach can be characterized as

$$\frac{t_{OLD}}{t_{NEW}} = \frac{t_{seq} \times n}{t_{anal} + (t_{seq} \times n_B)}$$

Denote $p = t_{anal} / (t_{seq} \times n_B)$.

Since

$$t_{anal} << t_{seq} \times n_B ,$$

it would be reasonable to calculate the higher limit of speed-up when $p \to 0$:

$$\lim_{p\to 0} \frac{t_{OLD}}{t_{NEW}} = \lim_{p\to 0} \frac{n \times t_{seq}}{n_B \times t_{seq}} = \frac{n}{n_B} . \quad (6)$$

From (6) we see that the speed-up of the proposed method is substantially depending on the number of faults n_B in S(B), which cannot be simulated by critical path tracing. Note, the content of S(B), and the speed-up effect of the proposed method depends also on the given test sequence to be fault simulated.

Since the set of S(B) can be calculated by parallel critical path tracing of faults separately by each pattern, the procedure is very fast compared to the traditional methods of sequential fault simulation.

VII. Experimental Data

The goal of the experimental research was to measure and investigate the speed-up potentials of the proposed new method for fault simulation in sequential circuits compared to the conventional fault simulation. The comparison was carried out for a representative selection of 27 circuits with different complexities (numbers of possible faults) of two benchmark families ISCAS'89 [17] (16 circuits) and ITC'99 [18] (11 circuits). The numbers of SAF faults in the circuits ranged from 614 up to 129422

Experiments were run on Intel i7-6700 4 cores 3.4 GHz, 32 GB RAM 2133 MHz, Windows 10, 64-bit.

Table I presents the main characteristic data of the circuits used in the experiments. We use the following notations: the number of inputs (#in), the number of outputs (#out), the number of flip-flops (#FF), the ratio between the numbers of observable primary outputs and not directly observable flip-flops #Out/#FF, which characterizes the sequential complexity of the circuit, the number of gates (#gates) and the number of possible stuck-at-faults (#faults).

Table II presents data of the experimental research, where the benchmark circuits are ordered according to their increasing complexity (the numbers of faults). We applied to each benchmark circuit a sequential test consisting of 32 independent test segments, where each segment was created of a sequence of 50 random input patterns. We assumed that before each test sequence the circuit flip-flops can be initialized (reset). The total length of the full test sequence was 1600 test patterns.

In columns 4-6, the test sequence is characterized for all circuits by two types of fault coverages s(A) and s(B), which mean the percentages of faults in the sets S(A) and S(B), respectively, in relation to the full set of faults given in column 3. The sum s(A) + s(B) in column 6 characterizes the sets of faults which propagate to the primary outputs and flip-flops, respectively. Note, the faults in S(A) are detected directly by the critical path tracing, but the faults in S(B) are those which need additional sequential fault simulation to determine if they propagate as well via feedback loops to the primary outputs.

All simulation time costs are given in seconds. The time costs t_{CP} and t_{CL} in columns 7 and 8 characterize the time needed for critical path tracing and the time needed for fault classification, respectively. Hence, the time $t_{CP} + t_{CL}$ represents the full cost of the parallel fault simulation of faults by critical path tracing. The total time cost t_{NEW} of fault simulation by the new method proposed in the paper, where the parallel critical path tracing and the traditional sequential fault simulation are combined, is depicted in column 9.

Columns 10 and 11 present the time costs of the baseline methods – the traditional sequential fault simulation. Here, two approaches are considered – simulation with fault dropping [14] (t_{OLD_FD}), and simulation with non-fault dropping (t_{OLD_NFD}). In column 12 the advantage (the gains in time cost reduction) of the proposed method is presented, and in column 13, for

comparison, the higher bounds of the possible gains, calculated by the formula (6), are depicted.

Note, the target of this research was not to generate test patterns with highest fault coverage, rather, the goal of experiments was to investigate the speed-up of fault simulation compared with the baseline, for any test sequence generated and supposed to be the objective of fault simulation. In the current case, randomly generated test sequences were selected with equal test length for all circuits.

TABLE I
CHARACTERISTICS OF THE BENCHMARK CIRCUITS USED IN EXPERIMENTS

Circuit	#in	#out	#FF	#Out/#FF	#gates	#faults
s349	9	11	15	0.7	161	614
s386	7	7	6	1.2	159	744
s510	19	7	6	1.2	211	900
s526	3	6	21	0.3	193	984
s641	35	24	19	1.3	379	1164
s713	35	23	19	1.2	393	1266
s953	16	23	29	0.8	395	1720
s1423	17	5	74	0.1	657	2610
s1494	8	19	6	3.2	647	2864
s5378	35	49	179	0.3	2779	9122
s9234	19	22	228	0.1	5597	16756
s13207	31	121	669	0.2	7951	24882
s15850	14	87	597	0.1	9772	29682
s35932	35	320	1728	0.2	16065	65248
s38417	28	106	1636	0.1	22179	69662
s38584	12	278	1452	0.2	19253	72346
b04	11	8	66	0.1	652	2836
b05	1	36	34	1.1	927	3952
b07	1	8	49	0.2	383	1712
b08	9	4	21	0.2	149	706
b10	11	6	17	0.4	172	806
b11	7	6	31	0.2	726	2894
b12	5	6	121	0.0	944	4426
b13	10	10	53	0.2	289	1350
b14	32	54	245	0.2	9767	38982
b15	36	70	449	0.2	8367	36496
b17	37	97	1415	0.1	30777	129422

VIII. DISCUSSION OF THE EXPERIMENTAL RESULTS

The test sequences were analysed twice. First, we calculated by the proposed method of parallel critical path tracing the sets of faults $\{S^k_A\}$ which propagated to the primary outputs OUT_A, and the sets of faults $\{S^k_B\}$ which propagated to the pseudo-outputs OUT_B (see Fig. 6 and Fig. 7). The time cost of this procedure was measured as t_{CP}. Then we classified the sets of faults into two subsets $S(A) \subset \{S^k_A\}$ and $S(B) \subset \{S^k_B\}$. The subset $S(A)$ included the faults which could be declared already as detected faults, as the result of critical path tracing method, whereas the subset of faults $S(B)$ included those faults which had to be simulated by a conventional simulator of sequential circuits. The time cost of fault classification procedure was measured as t_{CL}.

The experimental results of the gain in speed-up of the new method, compared with the baseline method, in column 12, and the higher bounds for the gain, are illustrated by the charts in Fig. 10. The new method outperforms the baseline method in a broad range of up to 5 times (in average 2 times), except of only very small circuits (less than 200 gates).

On the other hand, we see that the experimental results and the theoretically calculated bounds, using the formula (6), are very close in case of small circuits. In case of larger circuits, they nearly match even exactly. This gives an excellent possibility to predict by a simple calculation of the formula (6) the expected speed-up of fault simulation by the new proposed method.

TABLE II
EXPERIMENTAL RESULTS OF COMPARING THE PROPOSED METHOD VS. CONVENTIONAL FAULT SIMULATION METHODS FOR SEQUEMTIAL CIRCUITS

No	Benchmark circuits s - Iscas'89 b - ITC'99	Complexity of circuits (# faults)	Fault coverage of the simulated test sequence (%)			Time costs for critical path tracing of faults (s)		Total time costs for fault simulation (s)			Results	
			s(A)	s(B)	S(A)+S(B)	t_{CP}	t_{CL}	t_{NEW}	t_{OLD} t_{OLD_FD}	t_{OLD_NFD}	Gain t_{OLD_NFD} / t_{NEW}	Higher bound
1	2	3	4	5	6	7	8	9	10	11	12	13
1	s349	614	8.3	88.1	96.4	0.10	0.01	0.5	0.03	0.44	0.89	1.13
2	b08	706	2.3	96.0	98.3	0.11	0.01	0.6	0.05	0.51	0.84	1.04
3	s386	744	36.0	36.0	72.0	0.12	0.01	0.3	0.10	0.49	1.61	2.78
4	b10	806	3.0	89.5	92.4	0.10	0.01	0.7	0.08	0.65	0.93	1.12
5	s510	900	28.2	65.1	93.3	0.11	0.02	0.6	0.12	0.73	1.21	1.54
6	s526	984	3.3	32.3	35.6	0.10	0.01	0.4	0.25	0.80	2.14	3.09
7	s641	1164	50.0	37.2	87.2	0.12	0.03	0.7	0.16	1.56	2.15	2.69
8	s713	1266	43.8	37.6	81.4	0.12	0.03	0.8	0.23	1.75	2.17	2.66
9	b13	1350	2.5	62.0	64.5	0.12	0.02	1.2	0.41	1.72	1.43	1.61
10	b07	1712	1.1	79.8	80.8	0.14	0.05	2.3	0.47	2.70	1.15	1.25
11	s953	1720	8.0	86.3	94.4	0.16	0.03	2.5	0.31	2.73	1.07	1.16
12	s1423	2610	2.3	68.5	70.8	0.16	0.05	4.6	1.04	6.38	1.39	1.46
13	b04	2836	1.1	89.9	91.0	0.18	0.15	7.4	0.57	7.84	1.06	1.11
14	s1494	2864	37.9	20.2	58.1	0.15	0.07	1.5	1.38	6.28	4.22	4.95
15	b11	2894	0.8	67.3	68.2	0.16	0.04	5.3	1.58	7.60	1.43	1.48
16	b05	3952	2.2	32.2	34.4	0.21	0.16	4.5	4.22	12.99	2.86	3.11
17	b12	4426	0.5	38.4	38.9	0.20	0.07	6.3	4.25	15.83	2.50	2.61
18	s5378	9122	45.7	26.2	71.9	0.36	0.24	22.9	11.00	85.21	3.72	3.82
19	s9234	16756	2.1	30.1	32.2	0.58	0.36	91.1	105.00	300.03	3.29	3.33
20	s13207	24882	6.3	45.7	52.0	0.84	0.51	306.2	139.00	667.20	2.18	2.19
21	s15850	29682	5.1	32.5	37.6	1.17	0.60	309.6	250.00	946.45	3.06	3.08
22	b15	36496	0.6	49.7	50.3	5.35	0.65	521.2	228.60	1036.21	1.99	2.01
23	b14	38982	0.5	74.6	75.1	3.48	0.73	919.3	258.80	1226.38	1.33	1.34
24	s35932	65248	8.8	75.6	84.4	1.51	0.95	2827.0	367.00	3734.21	1.32	1.32
25	s38417	69662	1.9	50.7	52.6	2.44	1.35	2516.8	1189.00	4956.59	1.97	1.97
26	s38584	72346	8.0	69.9	77.9	2.27	1.48	3255.9	704.00	4650.52	1.43	1.43
27	b17	129422	0.3	25.0	25.3	15.42	1.96	3226.2	2856.20	12857.76	3.99	4.01
Average						1.33	0.35	519.9	226.81	1130.80	2.0	2.2

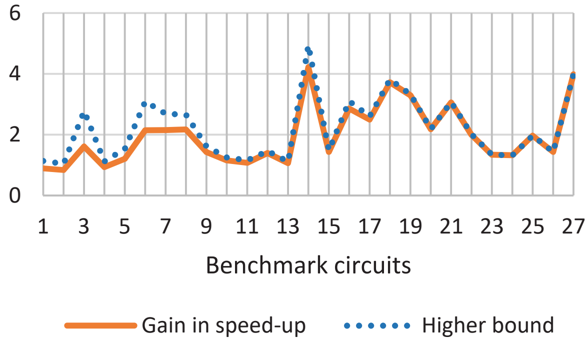


Fig. 10. Comparison of the speed-up of simulation with higher bound

Fig. 11 illustrates the dependence of the gain in speed-up of simulation on the complexity of circuits (the number of faults). To smooth the irregularities of the data for different circuits on the X-axis as shown in Fig. 10, and to take into account the trends of changing the comparable data, we compare the cumulative gain and the cumulative number of faults over the full set of benchmark circuits used in the experiments. The diagrams in Fig. 11 show that the trend of the gain is nearly linear over the full set of circuits ranked in order of complexity. We see from Fig. 11 that the linearity trend continues also in the second part of the ranked list of circuits (from 19 to 27) where the complexity starts to increase very rapidly.

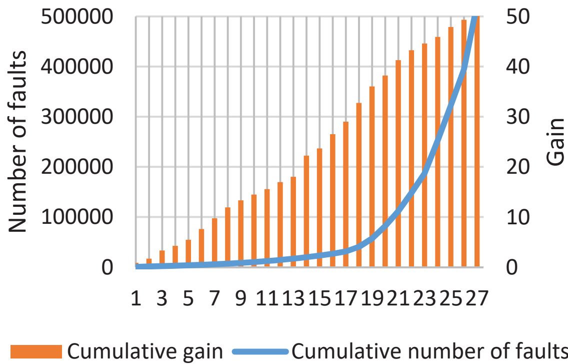


Fig. 11. Dependence of speed-up of simulation on the complexity of circuits

This finding allows to claim that the proposed method is well scalable, since the method starts to be in this comparison more efficient in case when the circuits' complexity will increase.

The cumulating functions were used in the graphics in Fig. 11 to smooth the anomalies of the gain curve in Fig. 10, and the nonlinearity of spreading the number of faults on the X-axis.

This anomaly can be explained by the two charts in Fig. 12, which illustrate the impact of the feedback intensities (the share of the faults propagating over the feedback loops) and the gain in speed-up of simulation for the full range of benchmark circuits. To characterize the feedback intensities, we use the values of s(B) in Table II, which represent the percentage of faults, which do not propagate directly to the primary output, but rather start propagating via feedback loops to the subsequent clock cycles. These faults represent the bottleneck of the proposed method, because they cannot be simulated by fast critical path tracing. In Fig. 12, we see that for all circuits where s(B) is high, the gain of the method is low, and opposite.

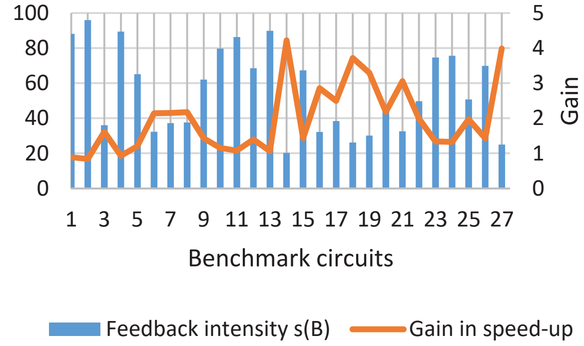


Fig. 12. Dependence of gain on the feedback level

The proposed method can be compared with two conventional fault simulation methods for sequential circuits as baselines: simulation with and without fault dropping, with time costs depicted in Table II in columns 10 and 11, respectively. The comparison results for the proposed method are shown also in Fig. 13. Here we see, that the proposed method outperforms well the traditional fault simulation without fault dropping, but loses in speed to the simulation with fault dropping.

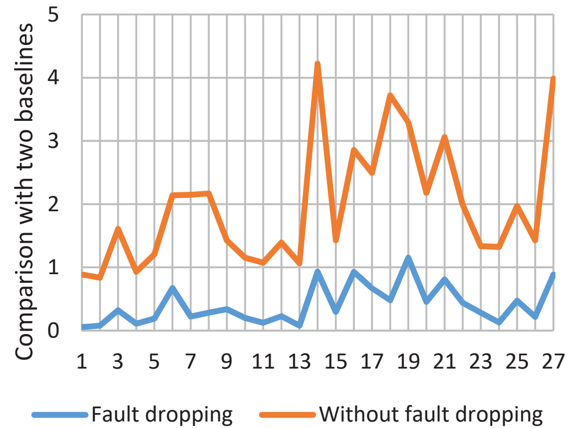


Fig. 13. Comparison of the speed of the proposed method with two baselines: sequential fault simulation with and without fault dropping

Note, however, that the fault dropping based fault simulation approach is able to evaluate only the fault coverage of the given test, whereas the simulation without fault dropping provides not only the fault coverage data, but also the data for fault diagnosis purposes, e.g. the data needed for creating fault tables and fault dictionaries.

On the other hand, the main core of the proposed method, the parallel critical path tracing of faults, is substantially tailored to get diagnostic information for each test pattern separately, in a similar way that the sequential simulation without fault dropping does. Hence, the comparison baseline for the proposed new method should be namely simulation without fault dropping, and it would be even unfear to compare the new method with simulation which uses fault dropping approach.

In Fig. 12 we saw that the high values of $s(B)$, which characterize the sequential level of the given circuit, act against the efficiency of the proposed method. The values of $s(B)$ will be high, when the ratio of the number of flip-flops and the number of primary outputs (#FF/#out) is very high. On the other

hand, we see also from Fig. 12, that if the value of $s(B)$ will reduce, then the speed of the proposed method will increase. This fact can motivate the redesign of sequential circuits for better observability to reduce the cost of testing, both, the test length and the time of fault simulation.

In Table III, an experiment is presented with 8 different designs of a circuit b17 with different numbers of flip-flops made observable by additional primary outputs. The row 1 corresponds to the original circuit b17 used in Table I and Table II, whereas the last row shows the extreme case where all flip-flops are directly observable. Here we see a dramatic speed-up of simulation and increasing of the gain, if the number of directly observable flip-flops will grow.

TABLE III
DEPENDENCE OF THE SPEED-UP GAIN ON THE OBSERVABILITY OF FLIP-FLOPS OF THE BENCHMARK CIRCUIT B17

Design No	s(A) %	s(B) %	Observable FFs, #	Time costs, s			Gain
				t_{CP}	t_{CL}	t_{NEW}	
1	0.3	25.0	0	15.80	2.01	3227	4
2	3.8	21.5	200	15.71	2.03	631	20
3	6.6	18.7	400	15.61	1.94	551	23
4	8.4	16.9	600	15.73	1.88	500	26
5	9.8	15.5	800	15.93	1.88	460	28
6	15.0	10.3	1000	15.70	2.04	312	41
7	23.6	1.7	1200	15.69	2	65	197
8	25.3	0.0	1415	15.74	2.03	18	724

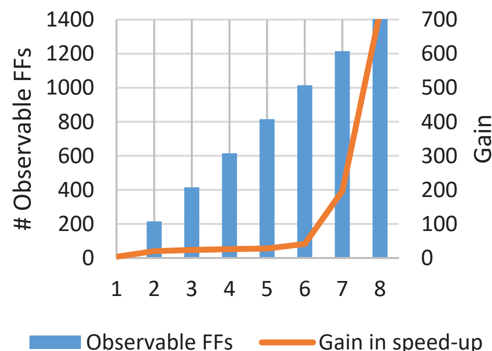


Fig. 14. Relationship between the speed-up gain and the observability of flip-flops of the benchmark circuit b17

As already mentioned, the results of the experimental research suggest a very fast procedure for prediction of the speed-up to be achieved by the new method compared to the conventional methods. Since

$$t_{anal} = t_{CP} + t_{CL} << t_{OLD}$$

the best way of such prediction would be to carry out Algorithm 1 for calculating the set S(B). The percentage of S(B) from the full set of faults will be a good criterion (higher bound according to the formula (6)) about the expected efficiency of using the proposed method.

IX. CONCLUSIONS

In this paper we have proposed a novel approach for fault simulation in sequential circuits, which allows to achieve considerable speed-up in simulation time compared to the conventional fault simulation methods. By experimental research, it was shown that for the pool of 27 benchmark circuits the average gain of speed-up was 2 times (in the best cases up to 5 times).

The high speed-up is achieved by integrating into the fault simulation process the exact parallel critical path tracing concept used so far only for fault simulation in combinational circuits.

The main novelties of the paper are as follows:

- we developed a novel method of separating the parallel critical path tracing into two parts, tracing to the primary outputs and tracing to the pseudo-outputs (to feedback loops);
- we developed a novel method for classification of the simulated faults into two classes: the faults directly detectable by combinational critical path tracing, and the faults to be simulated sequentially by traditional methods:
- the introduced fault classification approach allows to integrate combinational and sequential methods into a joint fault simulation procedure for sequential circuits.

We proposed also a very fast method for pre-evaluation of the applicability of the new fault simulation method, i.e. to predict the expected gain compared to the conventional methods. A dramatic speed-up can be expected if the size of s(B) is small. In case of the investigated benchmark circuits, the lowest value of $s(B)$ was 20% of the full set of faults, which provides speed-up gain 5 times. The size of $s(B)$ depends on the circuit characteristics, and also on the test sequence to be simulated.

ACKNOWLEDGEMENT

The work has been supported by EU's H2020 Twinning Action project TUTORIAL, Estonian institutional research grant IUT 19-1, and funded by Excellence Centre in IT in Estonia (EXCITE) project.

REFERENCES

[1] J.A. Waicukauski, et.al. Fault Simulation of Structured VLSI. VLSI Systems Design, Vol.6, No.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. Proc. 20th Design Automation Conference, 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] D.B. Armstrong. A deductive method for simulating faults in logic circuits. IEEE Trans. Comp., C-21(5), 464-471, 1972.

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

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

[10] 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.

[11] R. Ubar, S. Devadze, J. Raik, A. Jutman. Ultra Fast Parallel Fault Analysis on Structural BDDs. ETS, Freiburg, May 20-24, 2007.

[12] R. Ubar, S. Devadze, J. Raik, A. Jutman. Parallel X-Fault Simulation with Critical Path Tracing Technique. IEEE Conf. Design, Automation & Test in Europe - DATE-2010, Dresden, Germany, March 8-12, 2010, pp. 1-6.

[13] M. Gorev, R. Ubar, S. Devadze. Fault Simulation with Parallel Exact Critical Path Tracing in Multiple Core Environment. Proceedings of IEEE Conference on Design, Automation & Test in Europe - DATE-2015, Grenoble, France, 9-13 March, 2015.

[14] L.-T. Wang, C.-W. Wu, X. Wen. VLSI Test Principles and Architectures. Elsevier, 2006.

[15] A. Thayse, M. Davio. “Boolean Differential Calculus and its Applications to Switching Theory”. IEEE Trans. Comput. V.C-22, No.4, pp. 409-420, 1973.

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

[17] F. Brglez, D. Bryan, K. Kominski, "Combinational Profiles of Sequential Benchmark Circuits", Int. Symp. on Circuits and Systems, 1989, pp.1929-1934.

[18] F. Corno, M.S. Reorda, G. Squillero, "RT-level ITC'99 Bench-marks and First ATPG Results", In Proc. Of the IEEE Design & Test of Computers, Vol. 17, No. 3, 2000, pp.44-53.

Jaak Kõusaar is a software engineer at Kuehne+Nagel and a PhD student at Tallinn University of Technology. He received MSc degree in 2013 from Tallinn University of Technology (sup Prof. Raimund Ubar). His scientific interests include diagnostics of digital systems and agent-oriented programming.

Sergei Kostin received his PhD degree at the Tallinn University of Technology in 2012 on topic “Built-in self-diagnosis”. He is currently working as a researcher in the laboratory of department of Computer Systems at Tallinn University Technology. His research interests include embedded testing and diagnosis as well as reliability of digital logic circuits in nanoscale designs.

Raimund Ubar is a professor at Tallinn University of Technology. He received PhD degree in 1971 from the Bauman Technical University in Moscow, and DSc degree in 1987 from the Latvian Academy of Sciences. His scientific interests include computer science, design for testability and diagnostics of technical systems. Ubar is a member of Estonian Academy of Sciences, life senior member of IEEE, and Golden Core member of IEEE Computer Society. He was awarded from the Estonian Government by White Cross Orden of III Class.

Sergei Devadze received the Ph.D. degree in computer engineering from TU Tallinn, Estonia in 2009, and currently holds the position of researcher in this university. His research interests embrace such topics as fault tolerance and fault management architectures of digital systems, fault simulation techniques, usage of chip-embedded instrumentation and IJTAG for board and system test. He is a co-author of over 70 scientific papers in the field of digital design and test.

Jaan Raik is a full professor at Tallinn University of Technology where he received his MSc and PhD degrees in 1997 and 2001, respectively. His research interests include verification, test and dependable design of digital systems. He is a co-author of more than 300 publications in the field.