

Experimental Results of Forward-Looking Reverse Order Fault Simulation on Industrial Circuits with Scan

Irith Pomeranz⁺, School of ECE, Purdue University, W. Lafayette, IN 47907, USA
 Sudhakar M. Reddy⁺, ECE Dept., University of Iowa, Iowa City, IA 52242, USA
 Xijiang Lin, Mentor Graphics Corp., 8005 SW Boechman Rd., Wilsonville, OR 97070, USA

Fault simulation of a test set in an order different from the order of generation (e.g., reverse or random order fault simulation) is used as a fast and effective method to drop unnecessary tests from a test set in order to reduce its size. Recently in [1], we proposed an improvement to this type of fault simulation process that makes it even more effective in reducing the test set size. The procedure of [1] uses information about the first test in the test set that detects every fault, considering the test set in its original order. This information is used for dropping additional tests during the fault simulation process that considers the tests in a different order. Specifically, if a test t is not the first to detect any yet-undetected fault, t can be dropped during the fault simulation process, and tests simulated later can be used to detect the remaining faults. Tests such as t above are dropped from the test set *without simulation*. This improves the effectiveness of the fault simulation process as it allows it to drop additional tests that it would not drop otherwise, and it does so without simulating these tests. The improved procedure was named *forward-looking* fault simulation. The term forward-looking refers to the fact that certain tests are dropped because they are not necessary for detecting faults that will be detected later in the simulation process.

Next, we discuss an efficient implementation of forward-looking fault simulation in an industrial environment. We concentrate on reverse order fault simulation. Parallel pattern single fault propagation (PPSFP) simulation is used throughout our implementation of the forward-looking reverse order fault simulation process, since PPSFP is known to result in fast fault simulation for industrial circuits.

The use of parallel pattern simulation has the following implication on the identification of the first test $t_{1det}(f)$ that detects a fault f . Suppose that a fault f is simulated under M patterns, $t_i, t_{i+1}, \dots, t_{i+M-1}$, in parallel. Let the circuit have N outputs, z_1, z_2, \dots, z_N . Suppose that fault simulation shows that f is detected on output z_j of the circuit, and that at the point where the value of z_j is computed, the values of outputs z_{j+1}, \dots, z_N have not been computed yet. We have two options in this case. (1) We can stop the simulation of f , identify the first test t_{kj} that detects f on output z_j , and use t_{kj} as $t_{1det}(f)$. (2) We can continue the simulation of f until the values of all the outputs have been computed. For every output z_m on which f is detected, we can find the first test t_{km} that detects it. We can then use the lowest-indexed test out of the set $\{t_{km}\}$ as $t_{1det}(f)$.

The first option is faster since it allows the simulation of a fault to stop as soon as it is detected on the first output. However, it may not find the first test that detects the fault. The second option finds the first test that detects every fault. To make the procedure as efficient as possible, we use the first option.

During the reverse order fault simulation process, we fault simulate in parallel M patterns $t_i, t_{i+1}, \dots, t_{i+M-1}$ such that each t_i

has at least one yet-undetected fault f_k with $t_j = t_{1det}(f_k)$. We can avoid simulation of a fault f_m under a pattern t_j if $t_j = t_{1det}(f_m)$. We implement this as follows. Initially, all the M patterns are marked *unnecessary*. A pattern that is required to detect a target fault will be marked *necessary* during the simulation process. At the end, the patterns marked *unnecessary* will be dropped. When a fault f_m is simulated, we first check if $t_{1det}(f_m)$ is included in the M patterns being simulated. If it is not, the fault is simulated under M patterns in parallel. Otherwise, we check whether $t_{1det}(f_m)$ is marked *necessary* (this can happen if $t_{1det}(f_m)$ detected a fault that was simulated earlier). If $t_{1det}(f_m)$ is marked *necessary*, we mark that f_m is detected without simulating it. Otherwise, we simulate f_m under M patterns in parallel. After fault simulation, we mark the highest pattern that detects f_m as *necessary*.

It is possible to further improve the efficiency of the procedure by storing for every test t the number of faults f such that $t_{1det}(f) = t$. This number, denoted by $n_{1det}(t)$, should be decremented every time a fault f with $t_{1det}(f) = t$ is detected during the reverse order simulation process. When M patterns are selected for simulation, a pattern t_i for which $n_{1det}(t_i) = 0$ does not need to be included.

We considered six industrial circuits with scan under test sets for stuck-at faults (reported in Table 1) and test sets for transition faults (reported in Table 2). In every case, we compare the results of forward-looking reverse order fault simulation with the results of conventional reverse order fault simulation, and with the results of five passes of reverse and random order fault simulation. The test sets were generated by a commercial deterministic test generation procedure. Tables 1 and 2 are organized as follows. After the circuit name, we show the number of faults and the number of tests. Under column *reverse (reverse+rand)* we show the results of reverse order fault simulation (five passes of reverse and random order simulation), including the number of tests after compaction, and the CPU time. Under column *forw-looking* we show the results of forward-looking reverse order fault simulation, including the number of tests after compaction, the number of tests identified as redundant without simulation, and the CPU time. All the run times are given in seconds on a Sun U80 workstation.

Table 1: Results for stuck-at faults

circ	flts	orig		reverse		reverse+rand		forw-looking		
		tests	tests	CPU	tests	CPU	tests	drop	CPU	
ckt1	17K	422	358	5	338	13	339	80	5	
ckt2	65K	1737	1303	21	1244	76	1226	504	16	
ckt3	58K	3719	3040	10	2823	34	2797	877	10	
ckt4	117K	3440	2780	38	2620	139	2597	839	33	
ckt5	118K	1257	1020	24	962	121	938	315	21	
ckt6	1.8M	28384	21858	3655	20471	13768	20258	8117	3594	

Table 2: Results for transition faults

circ	flts	orig		reverse		reverse+rand		forw-looking		
		tests	tests	CPU	tests	CPU	tests	drop	CPU	
ckt1	21K	663	539	7	505	9	505	156	5	
ckt2	78K	2444	2030	43	1926	197	1900	538	36	
ckt3	80K	9622	8009	35	7647	145	7619	2001	37	
ckt4	155K	5994	5161	52	4762	287	4611	1378	49	
ckt5	134K	2074	1600	38	1462	152	1438	628	33	
ckt6	2.2M	43315	34442	9347	32210	38548	31776	11533	9287	

⁺ Research supported in part by NSF Grant No. MIP-9725053, and in part by SRC Grant No. 98-TJ-645.

[1] I. Pomeranz and S. M. Reddy, "Forward-Looking Fault Simulation for Improved Static Compaction", to appear in IEEE Trans. on Computer-Aided Design.