Small-Delay-Fault ATPG with Waveform Accuracy

Matthias Sauer* Alexander Czutro* Ilia Polian† Bernd Becker*

* Albert-Ludwigs-University Freiburg
Georges-Köhler-Allee 051
79110 Freiburg, Germany
{sauerm|aczutro|becker}
@informatik.uni-freiburg.de

† University of Passau
Innstraße 43
94032 Passau, Germany
ilia.polian@uni-passau.de

***Abstract*—The detection of small-delay faults is traditionally performed by sensitizing transitions on a path of sufficient length from an input to an output of the circuit going through the fault site. While this approach allows efficient test generation algorithms, it may result in false positives and false negatives as well, i.e. undetected faults are classified as detected or detectable faults are classified as undetectable. We present an automatic test pattern generation algorithm which considers waveforms and their propagation on each relevant line of the circuit. The model incorporates individual delays for each gate and filtering of small glitches. The algorithm is based on an optimized encoding of the test generation problem by a Boolean satisfiability (SAT) instance and is implemented in the tool *WaveSAT*. Experimental results for ISCAS-85, ITC-99 and industrial circuits show that no known definition of path sensitization can eliminate false positives and false negatives at the same time, thus resulting in inadequate small-delay fault detection. WaveSAT generates a test if the fault is testable and is also capable of automatically generating a formal redundancy proof for undetectable small-delay faults; to the best of our knowledge this is the first such algorithm that is both scalable and complete.**

I. Introduction

Small-delay faults (SDF) are common in nanoscale technologies. They are caused by defect mechanisms such as resistive opens and resistive shorts in the signal interconnects and in the power-distribution network of the circuit. They can also be a consequence of marginal parameter shifts of transistors in logic gates. Considerable effort has been spent to better understand modeling and simulation of SDFs [1] and automatic test pattern generation (ATPG) approaches to detect them [2], [3], [4], [5], [6], [7]. High coverage of SDFs is essential to maintain the quality of integrated circuits. The concept of *primitive faults* [8], [9] is used to reduce the number of target faults needed for high fault coverages.

An SDF is associated with an affected logic gate g within the circuit. It has a size s and an affected transition (rising or falling). If this transition occurs at the output of g, it is delayed by $\delta + s$ where δ is the nominal delay of g. In general, the application of a test pair (v_1, v_2) triggers multiple transitions at different lines of the circuit[1], including its outputs [10]. We denote the complete set of transitions on a line l, including the times of these transitions, by the term *waveform* on l under test pair (v_1, v_2). An SDF is detected by (v_1, v_2) if, for an output o, the waveforms on o under (v_1, v_2) differ at the *observation point* t_{obs} for the faulty and the fault-free circuit.

The detection of an SDF at gate g with size s necessitates the existence of a *sensitized path* from an input i to an output o of the circuit going through g and having an accumulated fault-free delay greater than or equal to $t_{obs} - s$. Intuitively, a sensitized path is composed of gates with off-path input(s) set to values which allow the propagation of transitions from the on-path input(s) to the output of the gate. Consequently, a transition launched at i can reach gate g; the SDF will result in different waveforms for the faulty and the fault-free circuit; and these differences could propagate through the remainder of the path from g to o and may be observed.

Today's SDF ATPGs select one or multiple paths through the SDF location and then attempt to sensitize these paths according to a *sensitization condition*. There exists a hierarchy of alternative sensitization conditions including hazard-free robust, robust, strong non-robust, weak non-robust, and functional sensitization [11], [12]. *Timed characteristic functions (TCF)* [13] are used to define bounds on arrival times to generate sufficient delays. Implementations such as [14] improve accuracy, but they are poorly scalable and cannot identify glitches.

None of the above-mentioned ATPG methods is accurate in enforcing conditions that are both necessary and sufficient for SDF detection. Such an inaccuracy can have detrimental effects on SDF coverage.

In this paper, we present an SDF ATPG algorithm that does not rely on the explicit notion of path sensitization or TCF. Instead, it directly considers the relationships between the possible waveforms on different lines of the circuit. For a given SDF, a set of consistent waveforms on all circuit lines required for detection is generated. The test pair is derived from the waveforms on the inputs (which are only allowed to switch once, at time 0). If no test pair is found, this constitutes the formal proof of untestability within the model assumptions. The proposed waveform-based method is related to *unrolling* [15], [16] the circuit, but uses further optimizations to create a more compact and yet accurate circuit representation.

The algorithm, implemented in the tool *WaveSAT*, relies on encoding the ATPG problem as a Boolean satisfiability instance. Waveforms are represented by a series of Boolean variables which contain logical values at discrete points of time (20 ps resolution is used for experiments). Efficient encoding of waveforms as SAT-instances is developed and combined with several optimizations in order to achieve scalability. Realistic model assumptions on small glitch filtering are added at limited cost. Experiments systematically quantify the inaccuracy obtained

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. To copy otherwise, to republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee. IEEE/ACM International Conference on Computer-Aided Design (ICCAD) 2012, November 5-8, 2012, San Jose, California, USA Copyright © 2012 ACM 978-1-4503-1573-9/12/11... $15.00

[1]The subsequent discussion assumes combinational circuits. The basic test application procedure is applicable, under restrictions, to sequential and scan circuits. The introduced algorithm can easily be adapted for these circuit types.

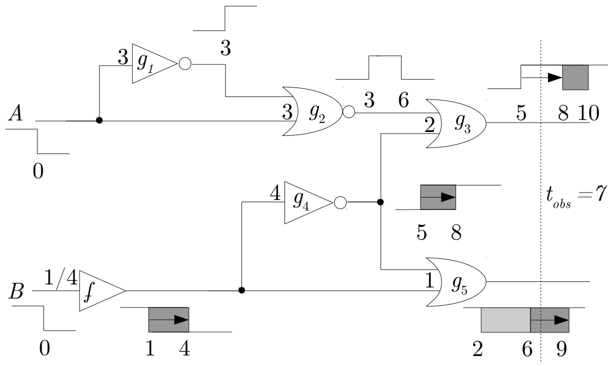

Fig. 1. Waveform accurate example of a small-delay fault at line B

when using conventional path-oriented SDF ATPG for different sensitization conditions.

Our experiments show, that path-oriented SDF ATPG methods result in double-digit percentages of incorrectly classified faults for many benchmark circuits. This includes faults that are missed even though a path of sufficient length through the fault location has been sensitized, and faults reported undetectable but for which a valid test pair exists. The breakdown of mis-classifications to these two groups varies depending on the sensitization condition used, such as robust or non-robust sensitization. Our algorithm WaveSAT achieves, for the first time, accurate resolution of all faults within the timing model assumptions, including invalidation by glitches and glitch filtering. It also yields formal proofs of SDF untestability and allows an accurate definition of SDF fault coverage metrics [17].

The remainder of the paper is organized as follows. The underlying timing model of the circuit and the relevant sensitization information are formally introduced in the next section. WaveSAT is presented in Section III. Experimental results are reported in Section IV. Section V concludes the paper.

II. Preliminaries

A. Timing model

We assume discrete time, with each time point $t_0, \ldots, t_T$ corresponding to a specific time given as integer-valued time units. For simplicity, we will just refer to the time (points) by $0, \ldots, T$. Consequently, all gates have delays expressed in integer time units. Although our framework can distinguish between rising and falling delays, we used identical delays for rising and falling transitions in the experiments for simplicity. When simulating the test pair (v_1, v_2), all inputs are assumed to switch from v_1 to v_2 simultaneously at time 0. We assume that each line stabilized its logical value under v_1 before time 0. Every line l in the circuit assumes a logical value $l(k) \in \{0, 1\}$ for each time k. The set of values $l(0), \ldots, l(T)$ is called the *fault-free waveform* on line l for test pair (v_1, v_2). For convenience, we may refer to the stable value of l under v_1 (before the application of v_2) by $l(0)$.

The SDF $f = (g, s)$ affects the output of gate g and delays all transitions on this output by s time units ($s \in \mathbb{N}$). We do not distinguish between slow-to-rise and slow-to-fall SDFs. Applying test pair (v_1, v_2) in the presence of SDF f induces on each line l the *faulty waveform* $l_f(0), \ldots, l_f(T)$. f is *detected* by (v_1, v_2) if there is at least one output o which assumes different values in absence and in presence of the fault at time t_{obs}, i.e., $o(t_{obs}) \neq o_f(t_{obs})$. Figure 1 shows a circuit with two inputs A and B, five gates (delays in time units are depicted to the left of the gates) and fault $(f, 3)$. Fault-free and faulty waveforms (indicated by grey color) are shown for all lines. The fault is detected at g_5 and not detected at g_3.

B. Path sensitization

Conventional ATPG approaches work by finding and sensitizing a path through the SDF location g. A path through gate g is a sequence of gates $g_1, \ldots, g_k$, such that g_1 is an input of the circuit, g_k is an output of the circuit, one of the gates is the target gate g, and the output of g_{j-1} drives an input of g_j for all $1 < j \leq k$. This input of g_j is called *on-path input*, all other inputs are called *off-path inputs*. Currently, we consider BUF, INV, AND, NAND, OR and NOR gates (more complex gates such as XOR gates must be mapped to these gates). The *controlling value cv* of a gate determines its output value when applied to one input; its opposite is the *non-controlling value* ncv[2]. The delay of the path is the sum of the delays of its gates (a somewhat more complex definition is necessary if slow-to-rise and slow-to-fall delays are distinguished). The *slack* of the path is the difference between t_{obs} and the delay of the path. It represents the total amount of time the path may be delayed without leading to a fault effect.

Intuitively, a path is *sensitized* by a test pair (v_1, v_2) if the delay somewhere along the path results in a delay being visible at its output. Hence, an SDF (g, s) is detected by (v_1, v_2) if a path through g with slack of s or less has been sensitized by this test pair. As we will see, the formal definition of path sensitization leads to a detection concept which does not match the more accurate definition of detection from Section II-A.

The path $g_1, \ldots, g_k$ is formally defined to be sensitized by test pair (v_1, v_2) according to a *sensitization condition*, if (v_1, v_2) launches a transition at g_1 and the sensitization condition holds for all side-inputs of all g_j with $0 < j < k$. In this paper, we consider hazard-free robust, robust, strong non-robust, weak non-robust, restricted functional and functional sensitization [11], [12].

To define the sensitization condition, it is necessary to distinguish between stable and unstable values. *Stable values* are guaranteed to be present at a logic line throughout the application of (v_1, v_2). Using the definition of waveforms, line l assumes the stable value of 0 if $l(0) = l(1) = \cdots = l(T) = 0$. (We write l = S0; S1 is defined accordingly). An *unstable value* guarantees that a signal eventually settles to this value upon application of the test pair: stable 1 at l corresponds to $l(T) = 1$. We write l = U1, which subsumes S1, a rising transition, a glitch from 1 to 0 and back to 1, and further, more complex waveforms settling to 1. U0 is the corresponding definition for unstable 0.

The path is sensitized according to the *hazard-free robust* condition if all off-path inputs of all gates have stable non-controlling values. Each extra delay along the path is guaranteed to propagate to the output. The same property can be shown for the relaxed *robust* condition: Off-path inputs of (N)AND gates have stable non-controlling value S1 if the on-path input has a falling transition and U1 if the on-path input has a rising transition. Off-path inputs of (N)OR gates have S0 (U0) if the on-path input has a rising (falling) transition.

The *weak non-robust* condition requires that each off-path input stabilizes to the non-controlling value U0 or U1. Note that

[2]The value of cv equals 0 for AND and NAND gates and 1 for OR and NOR gates; ncv is 0 for OR and NOR gates and 1 for AND and NOR gates.

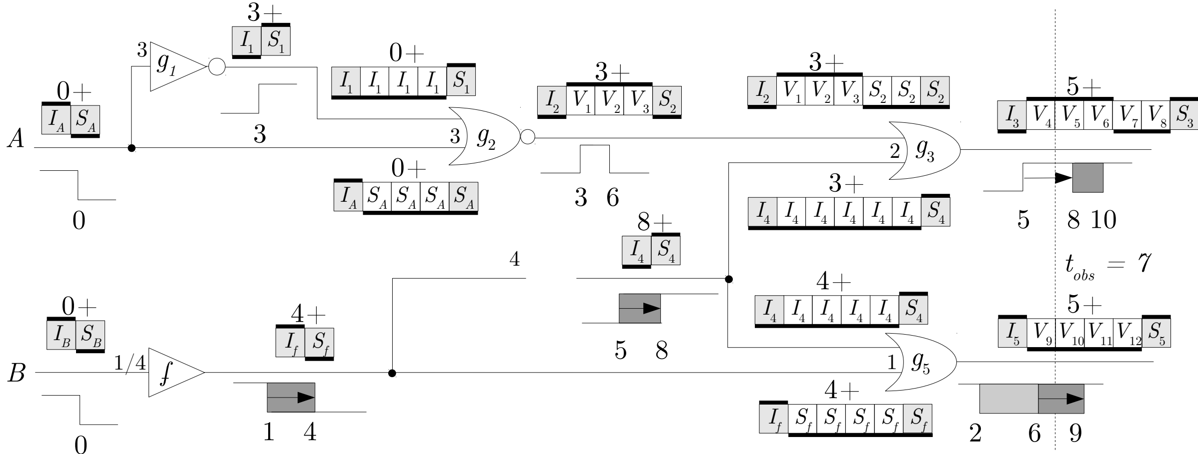

Fig. 2. Example SAT encoding

this may imply that no transition takes place on some gates of the path. For instance, path $B - f - g_4 - g_5$ in Figure 1 has one off-path input (the lower input of g_5) which has a falling transition (U0), but the output of g_5 stabilizes to 1 under both v_1 and v_2. In the example, a glitch is generated at the output which allows fault detection, but it is easy to construct paths for which weak non-robust sensitization leads to no transitions at all. Therefore, the *strong non-robust* condition requires, in addition, that the outputs of all gates on the path must settle to opposite values under v_1 and v_2. Path $B - f - g_4 - g_3$ in Figure 1 is sensitized according to this definition. The figure illustrates that strong non-robust sensitization of a path through the fault location does not guarantee fault detection even though the cumulated delay of the sensitized path (10) exceeds the observation time (7).

The *functional sensitization* condition requires that the off-path inputs of a (N)AND-gate assume the value U1 if the transition of the on-path input is rising and that the off-path inputs of a (N)OR-gate assume U0 if the transition of the on-path input is falling. Obviously, the chances to detect a fault using this definition are marginal. Similar to the strong non-robust condition, we add a requirement that a transition must be present on the output of each gate and call the resulting definition *restricted functional* sensitization.

C. Relationship

As Figure 1 shows, fault detection is possible, but not guaranteed, by sensitizing paths according to definitions other than hazard-free robust or robust. On the other hand, it is possible that there are no robustly sensitizable paths through the fault location g, but that non-robustly or functionally sensitizable paths exist. Even if robustly sensitizable paths can be obtained, they may be rather short and therefore not suited for detection of SDF sizes less than their slack; at the same time, longer paths with weaker sensitization conditions may exist.

If conventional SDF ATPG based on robust sensitization is unable to generate a test pair for a given SDF, it will classify it as untestable. This means coverage loss if the SDF was detectable. For example, in Figure 1 no hazard free robustly sensitizable path through the fault location exists but the fault is detectable. On the other hand, using a weaker definition may or may not result in a test pair that detects the fault. From the point of view of a conventional path-based ATPG, path $B - g_4 - g_3$ in Figure 1 is advantageous over path $B - g_4 - g_5$ because it has both, a stricter sensitization condition (strong non-robust vs. weak non-robust) and a smaller slack, but the fault is only detected using the second path. This is because conventional ATPG is not aware of exact timing on the off-path inputs. Even if the test pair generated by such ATPG is validated using an accurate simulator and determined not to detect the fault, there is no way to know whether the fault is undetectable or whether there is a different test pair which would detect the fault.

In the following, we present an ATPG algorithm *WaveSAT* which accurately models the timing of all on-path and off-path inputs and is able to generate test pairs that detect the fault whenever they exist. The notion of detection is based on the definition from II-A and does not incorporate any explicit path sensitization conditions.

III. SDF Test Generation Algorithm WaveSAT

For a given SDF $f = (g, s)$, WaveSAT generates a Boolean satisfiability (SAT) instance I_f. I_f is satisfiable if and only if a pair of assignments to the circuit's primary inputs exists that leads to a faulty value at an observable output at observation time t_{obs}. This instance is passed to a SAT-solver which will either return a solution which contains the test pair or proves unsatisfiability (and therefore fault redundancy within the model's assumptions).

The SAT-instance encodes all logical and temporal relationships within the circuit, in presence and in absence of the fault. This is conceptually similar to instances used in traditional SAT-based ATPG [18], [19]. However, traditional SAT-based ATPGs only consider the Boolean functionality of the circuit. In contrast, we encode the accurate (discretized) timing of the circuit, including the details of all waveforms, in the same SAT-instance. Intuitively, the value of each line in each time unit is described by at least one variable in the CNF. This necessitates using several optimizations to keep the number of extra variables low by identifying trivial portions of the waveforms.

A. SAT encoding

For each gate output g in the circuit, two Boolean variables I_g (initialization value) and S_g (stabilization value) are defined first. For a test pair (v_1, v_2), I_g is the value to which the output of g stabilizes under v_1 (i.e., before the transition is launched) and S_g is the value to which the output of g stabilizes under v_2. If A is the circuit's primary input, I_A is A's value under v_1 and S_A

is A's value under v_2. For all other gates g, I_g and S_g can be understood as the result of timing-unaware symbolic simulation of v_1 and v_2 respectively.

Let $EAT(g)$ be the *earliest arrival time* and $LST(g)$ be the *latest stabilization time* of any transition launched upon application of (v_1, v_2) at the output of gate g. Recall that we use discretized time points $0, 1, \ldots, T$. Therefore $EAT(g), LST(g) \in \{0, 1, \ldots, T\}$.

At all times before the first possible transition at g, i.e., $t \leq EAT(g)$, the output of g still assumes its value under v_1, namely I_g. At all times $t \geq LST(g)$, its logical value is S_g. In the simplest case ($LST(g) - EAT(g) = 1$), g switches only once, at time $i = EAT(g)$, meaning that the waveform of g's output consists of values I_g for times $0, \ldots, i$ and values S_g for times $i{+}1, \ldots, T$. For such gates, the waveform is completely described by I_g, S_g and the transition time i. Figure 2 shows such an encoding, e.g. for gate g_1. The time shift indicated by "3+" specifies that the transition takes place at time 3. Moreover, waveforms of this *simple shape* are always present at the primary inputs of the circuit with transition time 0.

The output waveform of a gate g's output is symbolically constructed from the waveforms of its inputs. Let us first consider the case that all waveforms at the inputs are of simple shape. To be able to combine these waveforms, they must first be aligned in time, i.e., start and end at the same time units t^* and t^{**}. This is achieved by padding as illustrated by gate g_2 in Figure 2. Its first input g_1 has waveform $[I_1; S_1]$ shifted by 3, while its second input has waveform $[I_A; S_A]$ shifted by 0. The alignment is performed by replacing g_1's waveform by $[I_1; I_1; I_1; I_1; S_1]$ and g_2's waveform by $[I_A; S_A; S_A; S_A; S_A]$, both shifted by 0 ($t^* = 0$, $t^{**} = 3$). The semantic of g_1's waveform is: it assumes value I_1 at time unit 0, keeps it for time units 1, 2 and 3, and switches to S_g at time unit 4, which is identical to the original semantics. Note that no additional Boolean variables need to be introduced for performing the shifting.

The waveforms on inputs and outputs of the gates may contain multiple transitions. To model such transitions accurately, new variables V_i are introduced when required. The output waveform of gate g contains a mixture of I, V and S variables, always starting with variable I_g and ending with variable S_g. These variables are calculated from input waveforms aligned to start at t^* and end at t^{**}. I_g describes the value at the output of g at time $t^* + D(g)$ and before, where $D(g)$ is the delay of g. Similarly, S_g describes this value at time $t^{**} + D(g)$ and thereafter. These values do not depend on the timing of the circuit and can hance be obtained by a timing-unaware encoding of g's Boolean function. The values assumed *between* $t^* + D(g)$ and $t^{**} + D(g)$ depend on the exact shape of the input waveforms. They are represented by the new V variables. Note that different gates are assigned individual sets of V variables that do not interfere with each other.

Consider gate g_2 in Figure 2. For this NOR-gate, $D(g_2) = 3$, $t^* = 0$ and $t^{**} = 3$. There are three time units between $EAT(g_2) = 3$ and $LST(g_2) = 6$, for which three new variables V_1, V_2 and V_3 are introduced. The SAT formulae describing this gate are: $(I_2 \equiv NOR(I_1, I_A))$,$(S_2 \equiv NOR(S_1, S_A))$, $(V_1 \equiv NOR(I_1, S_A))$, $(V_2 \equiv NOR(I_1, S_A))$ and $(V_3 \equiv NOR(I_1, S_A))$. The output waveform is shifted by 3 time units. Note that this number is only used during SAT formula construction to correctly perform alignments and is not part of the resulting SAT formula.

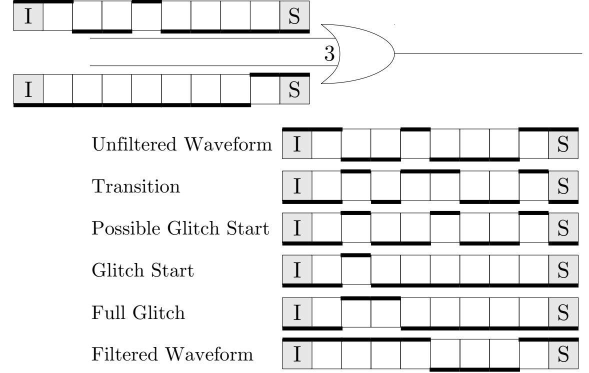

Fig. 3. Glitch filtering example

The resulting overall SAT formula (in conjunctive normal form CNF) completely describes the timing-aware logical behavior of all lines in the circuit. A detailed construction of the logic functionality of a circuit as CNF using Tseitin transformation is described in [20].

B. ATPG and simulation

To detect a fault, a test pair (v_1, v_2) must induce, at time unit t_{obs}, a value on an output o of the faulty circuit which differs from the output's fault-free stabilization value S_o. Please note, that delay faults do not change the stabilization value of any line in the circuit. Hence, in contrast to ATPG for other types of faults, a specific encoding of the fault-free logic value is not necessary.

To do so, we identify, for each output o, the variable W_o which describes its logical value at time unit t_{obs}. (W_o might be either variable I_o, variable S_o or one of the variables V_i.) We add clauses $\bigvee_{o \in \text{outputs}} (W_o \oplus S_o)$ to the CNF and search for a solution. If a solution is found, there is at least one output o where the faulty value W_o does not match the fault-free value S_o. The test pair (v_1, v_2) inducing this behavior is obtained from the I and the S variables of the primary inputs. If no solution is found, the fault is undetectable under the model's assumptions.

It is also possible to use the encoding for timing simulation. To simulate a given test pair (v_1, v_2), the I and S variables of the primary inputs are set to the required values and the SAT-solver searches for the solution. As all the other variables are implied by the primary inputs, the solution must exist and describe the complete timing information. It is also possible to check whether the fault was detected. We also used this mode to check the correctness of our implementation by comparing it with third-party simulators.

C. Glitch filtering

We extended the basic SAT model in order to filter glitches. For glitch filtering we consider waveforms at the output of each gate g and remove glitches of duration less than the (minimal) delay $D(g)$ of g. While this feature requires a considerable number of additional Boolean variables, it makes sure that no fault is detected by glitches which would be filtered by the logic gates. In addition, glitch filtering is required to match the simulation model used in e.g. Verilog.

TABLE I
ROBUST SENSITIZABLE PATH ANALYSIS

	PHAETON Validation								Runtime		
		Path found			Path too short		Unsensitizable				
Circuit	Calls	Detection	Miss	Redundant	Redundant	False Negative	Redundant	False Negative	Total	ATPG	Simulation
b12	10000 (100.00%)	9668 (96.68%)	0 (0.00%)	0 (0.00%)	48 (0.48%)	284 (2.84%)	0 (0.00%)	0 (0.00%)	15.21	3.07	0.00
b14	10000 (100.00%)	8292 (82.92%)	0 (0.00%)	0 (0.00%)	182 (1.82%)	1456 (14.56%)	2 (0.02%)	68 (0.68%)	2615.81	1451.27	0.00
b15	10000 (100.00%)	8285 (82.85%)	0 (0.00%)	0 (0.00%)	387 (3.87%)	1138 (11.38%)	12 (0.12%)	178 (1.78%)	2242.64	794.65	0.00
b20	10000 (100.00%)	9042 (90.42%)	0 (0.00%)	0 (0.00%)	170 (1.70%)	768 (7.68%)	2 (0.02%)	18 (0.18%)	4654.66	1756.08	0.00
b21	10000 (100.00%)	8810 (88.10%)	0 (0.00%)	0 (0.00%)	158 (1.58%)	1022 (10.22%)	0 (0.00%)	10 (0.10%)	4924.25	1438.18	0.00
b22	10000 (100.00%)	8554 (85.54%)	0 (0.00%)	0 (0.00%)	303 (3.03%)	1053 (10.53%)	7 (0.07%)	83 (0.83%)	5978.20	2411.12	0.00
c3540	10000 (100.00%)	8619 (86.19%)	0 (0.00%)	0 (0.00%)	331 (3.31%)	1000 (10.00%)	7 (0.07%)	43 (0.43%)	368.29	73.29	0.00
c5315	10000 (100.00%)	8193 (81.93%)	0 (0.00%)	0 (0.00%)	108 (1.08%)	549 (5.49%)	19 (0.19%)	1131 (11.31%)	96.08	46.43	0.00
c7552	10000 (100.00%)	7879 (78.79%)	0 (0.00%)	0 (0.00%)	70 (0.70%)	611 (6.11%)	27 (0.27%)	1413 (14.13%)	99.00	40.39	0.00
p35k	10000 (100.00%)	9946 (99.46%)	0 (0.00%)	0 (0.00%)	2 (0.02%)	52 (0.52%)	0 (0.00%)	0 (0.00%)	5099.94	30.01	0.00
p45k	10000 (100.00%)	8203 (82.03%)	0 (0.00%)	0 (0.00%)	0 (0.00%)	1697 (16.97%)	0 (0.00%)	100 (1.00%)	975.21	98.80	0.00
p78k	10000 (100.00%)	8946 (89.46%)	0 (0.00%)	0 (0.00%)	4 (0.04%)	960 (9.60%)	0 (0.00%)	90 (0.90%)	645.01	69.82	0.00
p81k	10000 (100.00%)	9281 (92.81%)	0 (0.00%)	0 (0.00%)	3 (0.03%)	716 (7.16%)	0 (0.00%)	0 (0.00%)	938.67	60.67	0.00

TABLE II
STRONG NON ROBUST SENSITIZABLE PATH ANALYSIS

	PHAETON Validation								Runtime		
		Path found			Path too short		Unsensitizable				
Circuit	Calls	Detection	Miss	Redundant	Redundant	False Negative	Redundant	False Negative	Total	ATPG	Simulation
b12	10000 (100.00%)	9732 (97.32%)	111 (01.11%)	0 (0.00%)	48 (0.48%)	109 (1.09%)	0 (0.00%)	0 (0.00%)	52.94	3.20	43.20
b14	10000 (100.00%)	8119 (81.19%)	1056 (10.56%)	0 (0.00%)	184 (1.84%)	641 (6.41%)	0 (0.00%)	0 (0.00%)	7130.38	3347.16	3545.08
b15	10000 (100.00%)	8468 (84.68%)	981 (09.81%)	2 (0.02%)	397 (3.97%)	142 (1.42%)	0 (0.00%)	10 (0.10%)	4819.54	1761.50	2814.32
b20	10000 (100.00%)	8166 (81.66%)	1363 (13.63%)	21 (0.21%)	151 (1.51%)	299 (2.99%)	0 (0.00%)	0 (0.00%)	25121.90	15582.80	8970.92
b21	10000 (100.00%)	8307 (83.07%)	1388 (13.88%)	2 (0.02%)	156 (1.56%)	147 (1.47%)	0 (0.00%)	0 (0.00%)	28166.30	19840.80	7818.81
b22	10000 (100.00%)	8344 (83.44%)	1197 (11.97%)	0 (0.00%)	310 (3.10%)	149 (1.49%)	0 (0.00%)	0 (0.00%)	20585.40	12877.00	7102.15
c3540	10000 (100.00%)	8372 (83.72%)	1153 (11.53%)	0 (0.00%)	335 (3.35%)	130 (1.30%)	3 (0.03%)	7 (0.07%)	378.66	124.38	209.10
c5315	10000 (100.00%)	9456 (94.56%)	328 (03.28%)	0 (0.00%)	127 (1.27%)	89 (0.89%)	0 (0.00%)	0 (0.00%)	111.36	13.65	79.81
c6288	10000 (100.00%)	7492 (74.92%)	1655 (16.55%)	2 (0.02%)	498 (4.98%)	353 (3.53%)	0 (0.00%)	0 (0.00%)	9477.55	3606.78	2962.50
c7552	10000 (100.00%)	9418 (94.18%)	312 (03.12%)	0 (0.00%)	96 (0.96%)	154 (1.54%)	1 (0.01%)	19 (0.19%)	105.79	13.38	71.38
p35k	10000 (100.00%)	7494 (74.94%)	2454 (24.54%)	0 (0.00%)	23 (0.23%)	29 (0.29%)	0 (0.00%)	0 (0.00%)	31965.00	23102.80	7585.63
p45k	10000 (100.00%)	8663 (86.63%)	792 (07.92%)	0 (0.00%)	404 (4.04%)	141 (1.41%)	0 (0.00%)	0 (0.00%)	1086.26	210.73	764.61
p78k	10000 (100.00%)	8673 (86.73%)	824 (08.24%)	2 (0.02%)	263 (2.63%)	208 (2.08%)	3 (0.03%)	27 (0.27%)	2051.45	462.19	1286.11
p81k	10000 (100.00%)	8754 (87.54%)	911 (09.11%)	0 (0.00%)	142 (1.42%)	193 (1.93%)	0 (0.00%)	0 (0.00%)	3791.07	1184.93	2225.49
p100k	10000 (100.00%)	7803 (78.03%)	1385 (13.85%)	0 (0.00%)	367 (3.67%)	445 (4.45%)	0 (0.00%)	0 (0.00%)	7970.91	3331.91	3893.40

Implementation of glitch filtering is illustrated in Figure 3. Given gate g's original output waveform ("Unfiltered Waveform"), defined by the logic function of g and the waveforms on the inputs, all glitches of duration less than $D(g) = 3$ must be removed. Therefore, five new sets of variables are introduced, and the fifth set ("Filtered Waveform" in Figure 3), which represents the glitch-free waveform, is used instead of the original set of variables ("Unfiltered Waveform"). The relationship between the new sets of variables are outlined next.

Intuitively, the first set of variables ("Transition") denotes the positions where the initial (unfiltered) waveform includes transitions from 0 to 1 or vice versa. These are potential starting times of the glitches. The set of variables "Possible Glitch Start" is obtained from the first by removing transitions less than $D(g)$ time units after another transition. This step is required to avoid incorrect identification of "spurious glitches" which will never occur because their initiating transition will actually be filtered itself. In the example, the second transition is spurious and is therefore removed. The set of variables "Glitch Start" indicates the starting time units of true (not spurious) glitches to be filtered. It is calculated by checking, for each "Possible Glitch Start" variable set to 1, whether there is another transition (variable "Transition") within $D(g)$ time units. After that, the actual duration of the glitch is calculated in variables "Full Glitch", which are then XORed with the initial waveform to obtain the final waveform with the glitch(es) filtered. Please note that all calculations needed for glitch filtering are encoded into the SAT-instance and therefore directly considered by the test generation.

D. Identification of stable timepoints

In order to improve the scalability of WaveSAT, we implemented several optimizations based on the identification of time units where no transition is possible. With this information, the introduction of many V variables can be avoided because their value is always identical to already introduced variables. For example, the output waveform of gate g_2 in Figure 2 contains three variables V_1, V_2 and V_3 with identical functionality, so V_2 and V_3 can be simply replaced by V_1. This technique is particularly effective when glitch filtering is employed, as the number of transition positions and hence the number of glitch start points is greatly reduced. In our experiments, the number of clauses was reduced by around 40% on average leading to an average runtime reduction of 50% to 75%.

E. Cone of timing influence

When WaveSAT is used in ATPG mode, only the differences at the outputs at time unit t_{obs} are of interest. We define, for each gate, the *detection-relevant interval* as the portion of the waveform which could influence an output at t_{obs}; the parts of the waveform outside this interval can be excluded from consideration. The detection-relevant interval is calculated by a simple structural algorithm, similar to the determination of EAT and LST. This technique nicely complements the usage of EAT and LST, because the detection-relevant intervals are narrow close to the circuit's outputs and wide further away from the outputs, while the intervals $[EAT, LST]$ are narrow close to the inputs and wide further away from the inputs.

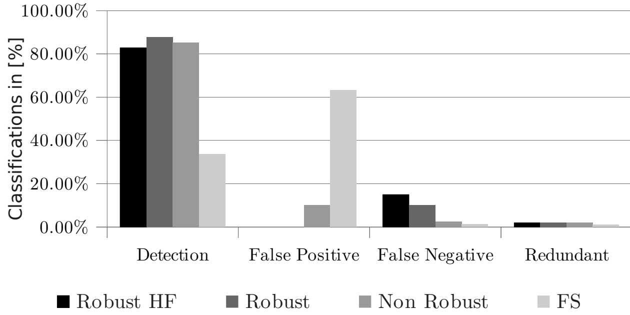

Fig. 4. Comparison of ITC-99 fault classifications depending on different path sensitization requirements

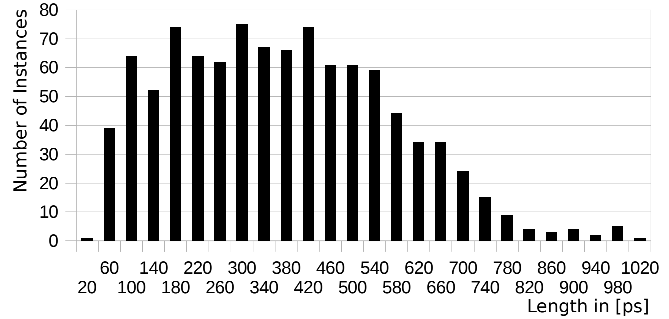

Fig. 5. Histogram of maximal fault detection window of ITC-99 b04 benchmark circuit

When used in combination with glitch filtering, it is necessary to extend the detection-relevant interval by the time needed to guarantee that glitches within this interval are detected and removed correctly.

By applying this optimization, around 75% of the clauses can be saved on average which leads to a runtime-reduction of 60% to 80% even when combined with the previous optimization.

IV. Experimental results

We applied WaveSAT to the combinational cores of reasonably sized ISCAS-85, ITC-99 and industrial benchmark circuits. All measurements were performed on an AMD Opteron computer using one 2.6 GHz-core and up to 4 GB RAM. The gate-level net lists and the corresponding real-valued delays were obtained by mapping original structural specifications (Verilog) to the Nangate 45nm Open Cell Library [21] using Synopsys Design Compiler. In our experiments, we determined a duration of 20 picoseconds to achieve high accuracy while keeping runtimes reasonable. We validated the correctness of the experimental results using an automated verification flow with a state-of-the-art commercial Verilog simulator.

As SAT-solving back-end we used a single-threaded version of the in-house SAT-solver *antom* [22] which supports efficient incremental SAT-solving with and without assumptions. In addition, we used a modified version of the preprocessor proposed in [23], [24] where the input literals were marked as "don't touch". All runtimes listed in this section are given in seconds.

In order to demonstrate the improved fault coverage achieved by WaveSAT compared to classical path-based approaches, we generated different SDFs for the 1000 most critical gates in the circuit. Criticality of a gate is defined according to length of the longest structural path through the gate. For each critical gate, we generated 10 different small delay fault sizes between the slack of the shortest and the slack of the longest structural path through g.

The observation point t_{obs} was defined to 120% of the longest structural path. For each fault instance, we computed a suitable path test using an improved version of the timing aware ATPG-tool PHAETON [25], [26]. If such a test exists, we used WaveSAT in simulation mode in order to determine if the test truly detects the fault. If PHAETON showed that no test exists under the given sensitization conditions, WaveSAT was run in ATPG mode in order to prove fault redundancy.

Table I and Table II compare WaveSAT with robust and strong non-robust path sensitization, respectively. The column "Calls" lists the number of fault instances considered. In the next columns we give statistics on the different classifications as follows:

Path found—Columns 3 to 5 list the cases, in which PHAETON was able to determine a path test according to the sensitization rules. The category is partitioned into the sub classifications "Detection", "Miss" and "Redundant". Detection is a true test that has been verified by WaveSAT. "Miss" and "Redundant" represent false positives, i.e. cases in which the simulation showed that the path chosen according to the robustness criteria does not propagate a fault effect. For a "Miss" a different test exists. In case of a redundant fault, WaveSAT proved redundancy.

Path too short—Columns 6 and 7 list the cases, where PHAETON was able to generate a path, but it was too short to detect the fault. The sub classification "Redundant" is a verified redundant fault. "False Negative" lists the cases where WaveSAT could demonstrate that a test exists although PHAETON could not find one due to too hard sensitization criteria.

Unsensitizable—A fault is classified as unsensitizable if PHAETON could not find a path at all. The results for those classifications, which are listed in columns 8 and 9, are defined as in the "Path too short" section,

The last three columns list different runtimes. "Total" gives the total runtime for the method's complete application including the time needed by PHAETON to generate the paths. "ATPG" and "Simulation" lists the times WaveSAT was run in ATPG and simulation mode respectively.

As can be seen, neither robust nor non-robust path sensitization yields accurate detection classification of the faults. Since robust paths are free of hazards before the transition occurs, no invalidation can happen. Hence, there are no false positives. However, due to too stringent sensitization criteria, a significant amount of false negatives is generated; up to 16.97% for p45k. For non-robust path sensitization, the number of false negatives is reduced to 1 or 2 percent. However, since non-robust tests may be invalidated by glitches, a serious amount of false positives (up to 24.54%) is introduced. Therefore, the total number of true detections is around the same for both methods.

We did also extensive comparisons with completely hazard-free and restricted functional sensitizable path sensitizations. Figure 4 shows the average over the results for the ITC-99 benchmark circuits. The figure lists the number of "Detections", "False Negatives", "False Positives" and "Redundant" faults for various types of sensitization criteria. The number of false positives increases with the relaxation of the pattern quality. For restricted

functionally sensitizable, around 63% of the detections are actually false positives. Likewise, the number of false negatives decreases. However, for restricted functionally sensitizable there are still false negatives. As restricted functionally sensitizable paths represent the latest possible stable transition, such instances are only testable by hazards. Hence, even restricted functionally sensitizable is too restrictive for the detection of some faults.

In a further experiment, we used WaveSAT to maximize the detection windows of a fault. Instead of just requiring a difference at the observation point, WaveSAT computed patterns that keep a stable faulty signal as long as possible before and after t_{obs}. Such tests are resilient against invalidation due to second order effects like e.g. model inaccuracies, dynamic pattern-dependent effects or differnt fault sizes. Even if the timing of the circuit is changed, the faulty circuit value is most likely visible within the whole detection window and hence the test can tolerate invalidations.

This extension is implemented by formulating a SAT-instance that requires a specific length for the detection window (e.g. 240 ps before and 240 ps after t_{obs}). By performing a binary search over the possible lengths of detection windows, the true maximum can be obtained.

The results are given in Figure 5. The x-axis gives the length of the stable fault detection window in picoseconds. The y-axis shows the number of faults. As can be seen, most fault instances fall in the area around 100 ps to 500 ps, leading to a very stable length of the detection window. In addition it is significantly beyond a typical delay of a gate which is around 100 ps in our example circuits.

Compared to the method used in the previous experiments, the runtime is increased by around a factor of 10, since for each fault, additional SAT-instances are generated.

V. Conclusions and Future Work

We presented the SAT-based small-delay ATPG WaveSAT, which explicitly models accurate timing by means of waveforms on each relevant line of the circuit. The model incorporates individual delays for each gate (with discrete resolution) and filtering of small glitches. Experimental results for ISCAS-85, ITC-99 and commercial circuits show that no known definition of path sensitization can eliminate both false positives and false negatives at the same time. WaveSAT is also capable of automatically generating a formal redundancy proof for small-delay faults; to the best of our knowledge this is the first algorithm that is both scalable and complete. In addition, fault detection windows of maximal width can be generated.

In future, we want to extend the method to the generation of functional tests for sequential circuits. This can be achieved by integrating the proposed encoding with a bounded or unbounded model checking approach. In addition, we plan to analyze the influence of process variations on the detection capabilities of different pattern sets.

VI. Acknowledgements

This work has been supported by the German Research Council (DFG) under grants GRK 1103, BE 1176-15/2 and PO 1220-2/2. We thank T. Schubert (University of Freiburg) for providing the SAT-solver *antom* and support. We further thank H.-J. Wunderlich (University of Stuttgart) for fruitfull discussions on test invalidation.

References

[1] G. L. Smith, "Model for Delay Faults Based upon Paths," in *Int'l Test Conf.*, pp. 342–349, 1985.

[2] A. K. Pramanick and S. M. Reddy, "On the Fault Coverage of Gate Delay Fault Detecting Tests," *IEEE Trans. on CAD*, vol. 16, no. 1, pp. 78–94, 1997.

[3] W. Qiu and D. M. H. Walker, "An Efficient Algorithm for Finding the K Longest Testable Paths Through Each Gate in a Combinational Circuit," in *Int'l Test Conf.*, pp. 592–601, 2003.

[4] A. Czutro, N. Houarche, P. Engelke, I. Polian, M. Comte, M. Renovell, and B. Becker, "A Simulator of Small-Delay Faults Caused by Resistive-Open Defects," in *European Test Symp.*, pp. 113–118, 2008.

[5] M. Yilmaz, K. Chakrabarty, and M. Tehranipoor, "Test-Pattern Selection for Screening Small-Delay Defects in Very-Deep Submicrometer Integrated Circuits," *IEEE Trans. on CAD*, vol. 29, pp. 760–773, May 2010.

[6] N. Devta-Prasanna, S. Goel, A. Gunda, M. Ward, and P. Krishnamurthy, "Accurate Measurement of Small Delay Defect Coverage of Test Patterns," in *Test Conference, 2009. ITC 2009. International*, pp. 1 –10, nov. 2009.

[7] S. Goel, N. Devta-Prasanna, and R. Turakhia, "Effective and efficient test pattern generation for small delay defect," in *VLSI Test Symposium, 2009. VTS '09. 27th IEEE*, pp. 111 –116, may 2009.

[8] A. Krstic, K.-T. Cheng, and S. Chakradhar, "Primitive delay faults: Identification, testing, and design for testability," *Computer-Aided Design of Integrated Circuits and Systems, IEEE Transactions on*, vol. 18, pp. 669 –684, jun 1999.

[9] K. Christou, M. Michael, and S. Neophytou, "Identification of Critical Primitive Path Delay Faults without any Path Enumeration," in *VLSI Test Symposium (VTS), 2010 28th*, pp. 9 –14, april 2010.

[10] M. S. Hsiao, E. M. Rudnick, and J. H. Patel, "Effects of Delay Models on Peak Power Estimation of VLSI Sequential Circuits," in *Int'l Conf. on CAD*, pp. 45–51, November 1997.

[11] N. K. Jha and S. K. Gupta, *Testing of Digital Systems*. Cambridge University Press, 2003.

[12] S. Reddy, *Models in Hardware Testing*, ch. 3. Springer, 2010.

[13] P. McGeer and R. Brayton, *Integrating Functional and Temporal Domains in Logic Design*. Kluwer, 1991.

[14] Y.-M. Kuo, Y.-L. Chang, and S.-C. Chang, "Efficient Boolean Characteristic Function for Timed Automatic Test Pattern Generation," *Computer-Aided Design of Integrated Circuits and Systems, IEEE Transactions on*, vol. 28, pp. 417 –425, march 2009.

[15] P. Maurer, "Two New Techniques for Unit-Delay Compiled Simulation," *Computer-Aided Design of Integrated Circuits and Systems, IEEE Transactions on*, vol. 11, pp. 1120 –1130, sep 1992.

[16] D. Tadesse, D. Sheffield, E. Lenge, R. Bahar, and J. Grodsteint, "Accurate Timing Analysis using SAT and Pattern-Dependent Delay Models," in *Design, Automation Test in Europe Conference Exhibition, 2007. DATE '07*, pp. 1 –6, april 2007.

[17] Y. Sato, S. Hamada, T. Maeda, A. Takatori, Y. Nozuyama, and S. Kajihara, "Invisible delay quality - SDQM model lights up what could not be seen," in *Test Conference, 2005. Proceedings. ITC 2005. IEEE International*, pp. 9 pp. –1210, nov. 2005.

[18] A. Czutro, I. Polian, M. Lewis, P. Engelke, S. M. Reddy, and B. Becker, "Thread-Parallel Integrated Test Pattern Generator Utilizing Satisfiability Analysis," *International Journal of Parallel Programming*, vol. 38, pp. 185–202, June 2010.

[19] S. Eggersgluess and R. Drechsler, "As-Robust-As-Possible Test Generation in the Presence of Small Delay Defects using Pseudo-Boolean Optimization," in *Design, Automation Test in Europe Conference Exhibition (DATE), 2011*, pp. 1 –6, march 2011.

[20] T. Larrabee, "Test pattern generation using Boolean satisfiability," *Computer-Aided Design of Integrated Circuits and Systems, IEEE Transactions on*, vol. 11, pp. 4 –15, jan 1992.

[21] "Nangate 45nm Open Cell Library." http://www.nangate.com.

[22] T. Schubert, M. Lewis, and B. Becker, "antom — Solver Description," in *SAT Race*, 2010.

[23] S. Kupferschmid, M. Lewis, T. Schubert, and B. Becker, "Incremental Preprocessing Methods for Use in BMC," *Formal Methods in System Design*, pp. 1–20, 2011. 10.1007/s10703-011-0122-4.

[24] N. Eén and A. Biere, "Effective Preprocessing in SAT through Variable and Clause Elimination," in *In proc. SAT'05, volume 3569 of LNCS*, pp. 61–75, Springer, 2005.

[25] M. Sauer, A. Czutro, T. Schubert, S. Hillebrecht, I. Polian, and B. Becker, "SAT-based Analysis of Sensitisable Paths," in *IEEE Design and Diagnostics of Electronic Circuits and Systems*, pp. 93–98, April 2011. Best Paper Award in the Test Category.

[26] M. Sauer, J. Jiang, A. Czutro, I. Polian, and B. Becker, "Efficient SAT-Based Search for Longest Sensitisable Paths," in *Asian Test Symp.*, November 2011.