A Synthesis Method to Alleviate Over-testing of Delay Faults Based on RTL Don't Care Path Identification*

Yuki Yoshikawa[1], Satoshi Ohtake[2] , Tomoo Inoue[1] and Hideo Fujiwara[2]
[1]Graduate School of Information Sciences, Hiroshima City University
3-4-1 Ozuka-higashi, Asaminami, Hiroshima 731-3194, Japan
[2]Graduate School of Information Science, Nara Institute of Science and Technology
Ikoma, Nara 630-0192, Japan
E-mail:[1] {*yosikawa, tomoo* }*@hiroshima-cu.ac.jp,* [2]{*ohtake, fujiwara*}*@is.naist.jp*

Abstract

A register-transfer level (RTL) circuit meeting a design specification may contain some functionally unused paths. If functionally unused paths can be easily identified at RTL, the information can be utilized to eliminate the corresponding gate-level paths from the target of testing. Testing such gate-level paths is considered to be futile. In this paper, we present a method for identifying such functionally unused paths, called RTL don't care paths, using RTL information, and a method of synthesis for transforming the identified paths into untestable paths which will never do a mischief. As a result, our approaches contribute to identification of many untestable paths and reduction of over-testing.

1. Introduction

For high speed digital circuits, delay testing is emphasized to guarantee the timing correctness of circuits. Transition fault model and path delay fault (PDF) model are commonly used to test delay defects [1, 2]. For the transition fault model, the number of faults in a circuit is linear in terms of the number of gates. However, extra delay caused by the fault is assumed to be large enough. In recent works, methods of selecting longest testable paths[3, 4] and a measure of evaluating its ability to detect small delays, called statistical delay quality model(SDQM)[5], have been proposed. On the other hand, the path delay faults can model accumulative small delays along a path while the PDF model has problems: circuits have tremendous number of paths and many paths are functionally untestable.

To make transition fault test accurate, faults should be sensitized along testable paths. Similarly, for path delay test, untestable paths should be eliminated to ease handling paths. To achieve these, several methods of identifying untestable paths have been proposed in the last decade. For combinational circuits, the techniques presented in [6, 7, 8] are approaches for identifying combinationally untestable paths. The works for sequential circuits[9, 10] are equally important. However, these gate-level approaches would consume long CPU time for dealing with the large number of paths.

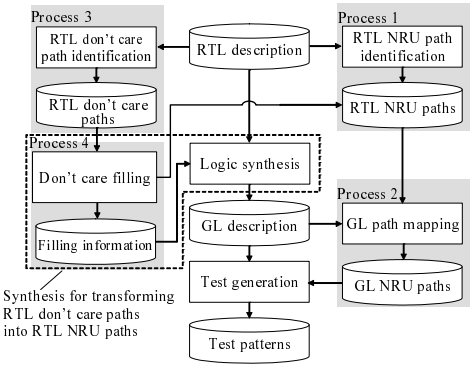


Figure 1. Whole flow of our strategy.

As an approach from higher-level of design abstraction, a method of identifying *non-robust untestable* (NRU) paths focusing on data transfers at RTL have been presented[11]. In the work, paths are dealt with at RTL, called RTL paths. An RTL path can be considered as a bundle of gate-level paths between two registers. The total number of RTL paths in a circuit is much smaller than that of gate-level paths, and therefore the path identification can be performed in a reasonable amount of time. The flow of the work is shown as processes 1[11] and 2[12] in Figure 1. In the process 1, RTL-NRU paths are identified for RTL paths in a circuit. Moreover, in the process 2, mapping from the identified RTL-NRU paths to their corresponding gate-level paths is established. The information of identified paths can be utilized to alleviate test generation effort and/or to reduce over-testing.

In this paper, we consider identification of functionally unused RTL paths which are called RTL don't care (RTL-DC) paths in addition to RTL-NRU paths. During RTL design coding, in case the outputs of an RTL module in a datapath are incompletely specified for its input domain or state transitions in a finite state machine are incompletely specified, data signals in the datapath and/or control signals

*This work was supported in part by Semiconductor Technology Academic Research Center under Research Project.

1093-0167/09 $25.00 © 2009 IEEE
DOI 10.1109/VTS.2009.27

from the controller may include don't care values Xs. If a path from a register to another is not guaranteed to have a data transfer at RTL, it is said that the path is unused within the given design specification. The identification of such paths is done in process 3. Gate-level paths corresponding to the identified RTL-DC paths can be NRU or non-robust testable (NRT) depending on an assignment value to each X in logic synthesis. If such RTL-DC paths are unintentionally transformed into NRU paths during synthesis, it is conceivable that identification of the NRU paths at gate-level is intractable. Therefore after the process 3, we transform RTL-DC paths into RTL-NRU paths (Process 4). Consequently, our approach contributes to increase in the number of identifiable NRU paths at RTL, i.e., the number of paths need to be tested decreases.

Experimental results show that there exist a lot of RTL don't care paths in some re-coded ITC'99 benchmark circuits and most of the identified RTL don't care paths are transformed into RTL-NRU paths. Additionally, we show over-testing reduction by eliminating path delay faults corresponding to the identified RTL-NRU paths from the target of test generation.

2. RTL path classification

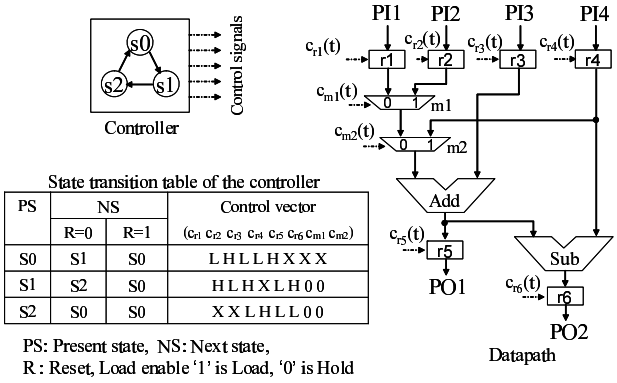
2.1. RTL circuit

Our path identification method is applied to RTL paths in structural RTL designs. If a target RTL circuit is described as a functional RTL, we can extract the structural information during synthesis process. For example, Explorations tool[13], which is a high-level synthesis tool, can generate a structural RTL from its original functional RTL. An example of a structural RTL circuit is shown in Figure 2. A structural RTL circuit consists of a controller, represented by a finite state machine, a datapath, represented by RTL modules such as MUXs, operational modules and registers, and RTL signal lines between them. For convenience, fan-out branches on RTL signal lines are treated as RTL modules to describe RTL paths uniquely as defined in the next subsection. State transitions are assumed to be completely specified for all pairs of a state and an input vector.

2.2. RTL path classification

An RTL path in an RTL circuit is an ordered set of RTL modules $\langle r_s, m_1, m_2, .., r_e \rangle$, where r_s is a register or a primary input, r_e is a register or a primary output. Also m_i is a combinational RTL module and the output of m_i connects with an input of m_{i+1}. A path $\langle r1, m1, m2, Add, r5 \rangle$ in Figure 2 is an example of RTL path.

A gate-level path p in a combinational circuit is said to be non-robust untestable (NRU) if any transition at the start of p is not propagated to the end of p unless any off-input on p has delay. Note that, in case of sequential circuits, launch and capture of transitions are controlled by the load-enable signals of FFs, which are the start and end points of



State transition table of the controller

PS	NS		Control vector
	R=0	R=1	(c_{r1} c_{r2} c_{r3} c_{r4} c_{r5} c_{r6} c_{m1} c_{m2})
S0	S1	S0	L H L L H X X X
S1	S2	S0	H L H X L H 0 0
S2	S0	S0	X X L H L L 0 0

PS: Present state, NS: Next state,
R : Reset, Load enable '1' is Load, '0' is Hold

Figure 2. An RTL circuit C_{RTL}.

paths. That is, a path p in a sequential circuit is non-robust untestable if, for any cycle t, (1) the control signal for the start FF is load-disable (or hold) at t, (2) at least one off-input on p has controlling value at $t+1$, or (3) the control signal for the end FF is load-disable (or hold) at $t+1$.

Suppose an RTL circuit C_{RTL} and a gate level circuit C that is an implementation of the RTL circuit C_{RTL}. In C there exists a bundle of paths $\delta(p)$ that correspond to an RTL path p in C_{RTL}. In general, for an RTL circuit, there exist multiple implementations of gate level circuits with different optimality. Further, if logic synthesis systems differ, the resultant logic circuits may also differ independent of the objective optimality. Therefore, letting C_{GL} denotes a set of possible gate level circuits implementing C_{RTL}, we can classify RTL paths into the followings.

Definition 1 (RTL non-robust untestable path) An RTL path p in the RTL circuit C_{RTL} is *RTL non-robust untestable (RTL-NRU)* if all the gate-level paths in $\delta(p)$ are non-robust untestable (NRU) for any gate-level circuit $C \in C_{GL}$. □

Definition 2 (RTL non-robust testable path) An RTL path p in the RTL circuit C_{RTL} is *RTL non-robust testable (RTL-NRT)* if at least one gate-level path in $\delta(p)$ is non-robust testable (NRT) for any gate-level circuit $C \in C_{GL}$. □

Definition 3 (RTL synthesis dependent path) An RTL path p in the RTL circuit C_{RTL} is *RTL synthesis dependent (RTL-SD)* if there exists a gate-level circuit $C \in C_{GL}$ where all the gate-level paths in $\delta(p)$ are NRU and also can exist $C' \in C_{GL}$ where at least one gate-level path in $\delta(p)$ is NRT. □

The RTL path classification relates to RTL designers' concern. Whether the resultant gate-level paths corresponding to RTL-SD paths are NRU or NRT depends on logic synthesis, and hence the RTL-SD paths must be don't care for designers. As well, RTL-NRU paths result in gate-level paths on which the delay of transitions will not affect the logic behavior, and accordingly the RTL-NRU paths can be also regarded as don't care. Consequently, if an RTL path is RTL-NRU or RTL-SD, the path is said to be RTL don't care

(RTL-DC).

Since we identify gate-level NRU paths from RTL paths, it is necessary to clarify the correspondence between RTL path p and its corresponding gate-level path in $\delta(p)$. It can be achieved by the method of mapping proposed in [12]. In this paper, as one solution to completely achieve the clarification, we consider module interface preserving-logic synthesis (MIP-LS)[11]. During an MIP-LS, for an RTL circuit, optimization is performed within each module, and each RTL signal line connecting RTL modules is just split to single-bit signal lines, i.e., the connectivity of all the RTL modules in the RTL circuit is guaranteed to be propagated to a synthesized gate-level circuit through any MIP-LS. For example, a logic synthesis tool DesignCompiler (Synopsys) is able to run MIP-LS with default setting unless the flatten option is set.

3. RTL-DC path identification

3.1. Control-dependent RTL-DC path

The load-enable signals of registers determine the time when a data transfers from one register to another, and the select signals of MUXs determine the path to transfer data. The table in Figure 2 shows control vectors to control registers and MUXs for each state of the controller. Figure 3 shows RTL circuit at cycle $t-1$ and t and the original RTL circuit is shown in Figure 2. Let p be an RTL path. Let $c_r(t)$ be the load-enable signal of a register r at cycle t. A value provided by r at a cycle t will be determined according to the load-enable signal from the controller during the previous cycle, $c_r(t-1)$. In case of $c_r(t-1) = \mathrm{L}$, register r *loads* a value produced by the logic connected to the input of register r, and will provide the loaded value by cycle t. Otherwise (i.e., in case of $c_r(t-1) = \mathrm{H}$), it will keep the output with the same as the previous one stored at cycle $t-1$. Let $M(p)$ be a set of MUXs on a path p and let $c_m(t)$ be the select signal of $m \in M(p)$ in cycle t. A path is said to be *sensitized* at cycle t if all the MUXs on p select p at cycle t, i.e., $\forall m \in M(p)[c_m(t) = p]$.

Definition 4 (csd-uncontrollability) Suppose a register r, and let P be a set of paths whose end register is r. Register r is said to be *control-signal-dependently uncontrollable* (csd-uncontrollable, for short) at cycle t, if, for any path $p \in P$,

- for each multiplexer $m \in M(p)$ on path p, $c_m(t) = c_m(t-1)$, and
- if path p is sensitized, i.e., for each $m \in M(p)$, $c_m(t) = p$, then the start register r_s of path p satisfies $c_{r_s}(t-1) = \mathrm{H}$, or r_s is csd-uncontrollable at cycle $t-1$. □

Definition 5 (csd-unobservability) Suppose a register r, and let P be a set of paths whose start register is r. Register r is said to be *control-signal-dependently unobservable* (csd-unobservable, for short) at cycle t, if both of the following two conditions are satisfied.

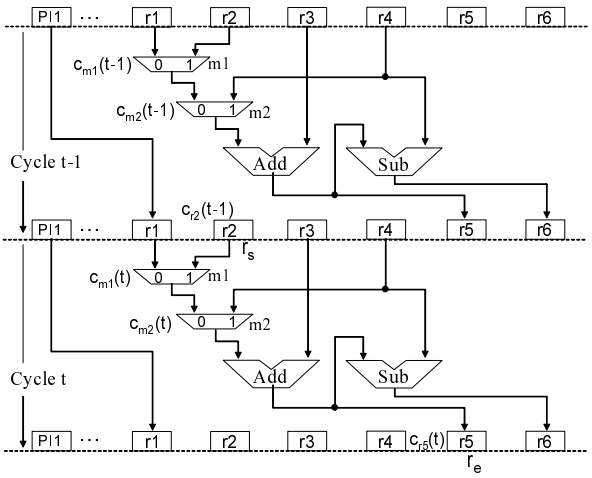


Figure 3. An RTL circuit C_{RTL} at two cycles t-1 and t

- For any path $p \in P$,
  - path p is not sensitized at the next cycle $t+1$ (i.e., $\exists m \in M(p)[c_m(t+1) \neq p]$), or
  - for the end register r_e of p, $c_{r_e}(t+1) = \mathrm{H}$ or r_e is csd-unobservable at cycle $t+1$.
- If $c_r(t) = \mathrm{H}$, r is also csd-unobservable at cycle $t+1$.

Let us consider an example for Definitions 4 and 5 using Figure 3. The state transition table with control vectors is shown in Figure 2. Suppose the register $r5$ at cycle t and the state transition from $s1$ to $s2$. The states $s1$ and $s2$ correspond to cycle $t-1$ and t, respectively. There are four paths whose end is $r5$, and the MUXs existing on the four paths are $m1$ and $m2$. The control signals of $m1$ and $m2$ at cycle t and $t-1$ are $c_{m1}(t)$=$c_{m1}(t-1)$=0 and $c_{m2}(t)$=$c_{m2}(t-1)$=0, respectively, and thus $r5$ loads a value $r1+r3$ at both cycles $t-1$ and t. Additionally, for cycle $t-1$, the load-enable signals of $r1$ and $r3$, which are the start register of the synthesized path in cycle t, are $c_{r1}(t-1)$=H and $c_{r3}(t-1)$=H. Therefore, the value of $r5$ at cycle t is the same as that at cycle $t-1$, which means that $r5$ is uncontrollable at cycle t.

Suppose the register $r4$ at cycle $t-1$ the state transition from $s0$ to $s1$. The states $s0$ and $s1$ correspond to cycle $t-1$ and t, respectively. There are two paths whose start is $r4$. For the path p=$\langle r4, m2, Add, r5\rangle$, $m2$ blocks the propagation of $r4$ because of $c_{m2}(t)$=0. For the path p=$\langle r4, Sub, r6\rangle$, $r6$ does not capture the propagated value because of $c_{r6}(t)$=H, that is, $r4$ is unobservable at cycle $t-1$.

Definition 6 (csd-unsensitizability) A path $p = \langle r_s, m_1, m_2, ..., r_e\rangle$ is said to be *control-signal-dependently unsensitizable* (csd-unsensitizable, for short) at a cycle t if at least one of the following conditions is satisfied.

1. For the start register r_s of p, $c_{r_s}(t) = \mathrm{H}$, or r_s is csd-uncontrollable at cycle t.
2. Path p is not sensitized at cycle $t+1$ (i.e., $\exists m \in M(p)[c_m(t+1) \neq p]$).

3. For the end register r_e of p, $c_{r_e}(t+1) = \mathrm{H}$, or r_s is csd-unobservable at cycle $t+1$. □

For example, in Figure 3, suppose p=$\langle r2, m1, m2, Add, r5\rangle$ and the state transition from $s0$ to $s1$, and they correspond to cycle $t-1$ and t, respectively. The load-enable signal $c_{r2}(t-1)$ is H, and thus no transition is launched on the path at cycle t. For the same path, let us consider the state transition from $s1$ to $s2$. A transition is launched at cycle t because of $c_{r2}(t-1)$=L, but it is blocked at $m1$ because $c_{m1}(t)$=0. Next, let us suppose the state transition from $s2$ to $s0$. A transition at $r2$ may be propagated to $r5$ at cycle t because of $c_{r2}(t-1)$=L, $c_{m1}(t)$=1 $c_{m2}(t)$=X, but the propagated value is not captured into $r5$ because of $c_{r5}(t)$=H. The path p=$\langle r2, m1, m2, Add, r5\rangle$ is unsensitizable at any cycle so that there is no cycle where a transition at the start point $r2$ is captured into the end register $r5$. Generalizing that, we can have the following theorem for a sufficient condition of RTL-NRU path.

Theorem 1 A path is RTL-NRU if the path is csd-unsensitizable at *any* cycle. □

As mentioned in Sect. 2, an RTL design includes some paths on which data transfers are not taken account by its designers. The control signals for the modules on such paths must not be specified in its description, and are generally denoted by X (don't care). We can define such paths as follows.

Definition 7 A control vector for a path $p = \langle r_s, m_1, m_2, ..., r_e\rangle$ is said to be *sensitization-unspecified* at a cycle t if at least one of the following conditions is satisfied.

1. For the start register r_s of p, $c_{r_s}(t) = \mathrm{X}$.
2. There exists at least one MUX $m \in M(p)$ on p such that $c_m(t+1) = \mathrm{X}$.
3. For the end register r_e of p, $c_{r_s}(t+1) = \mathrm{X}$.

Such a path p is called a sensitization-unspecified path (su-path) with cycle t. □

Lemma 1 For an su-path p with a cycle t, there exists an assignment to the unspecified control signal so as to make path p csd-unsensitizable at cycle t. □

Lemma 1 implies that an RTL-SD path can be RTL-NRU by a logic assignment. Therefore, there exists a logic synthesis to generate a gate-level circuit where all the gate-level paths corresponding to an RTL-SD path are NRU.

In Figure 3, consider path p=$\langle r2, m1, m2, Add, Sub, r6\rangle$ and the state transition from $s2$ to $s0$. The states $s2$ and $s0$ correspond to cycle $t-1$ and t, respectively. The path is su-path at cycle $t-1$ because of $c_{r_2}(t-1) = \mathrm{X}$, $c_{m_1}(t) = \mathrm{X}$, $c_{m_2}(t) = \mathrm{X}$ and $c_{r_6}(t) = \mathrm{X}$. If we assign a logic value such as $c_{r_2}(t-1) = \mathrm{H}$ or $c_{m_1}(t) = 0$ or $c_{m_2}(t) = 1$ or $c_{r_6}(t) = \mathrm{H}$, then the su-path can be csd-unsensitizable, and consequently p becomes RTL-NRU.

From Theorem 1 and Lemma 1, we can have the following theorem.

Theorem 2 A path is RTL-DC if the path is csd-unsensitizable or sensitization-unspecified at *any* cycle. □

4. RTL-DC path transformation

This section presents a heuristic algorithm to maximizing the number of RTL-NRU paths transformed from RTL-DC paths while keeping the number of logic value assignments to Xs as small as possible.

4.1. Heuristic approach

Consider a control vectors including Xs for all the states of a controller as shown in Figure 2. When a logic value is assigned to each X, the number of combinations of the assignments is $2^{|X|}$, where $|X|$ is the total number of Xs on control vectors. Searching all the possibility is a hard problem if $|X|$ is large. Therefore, we use some heuristics. First we consider logic value assignment to each X that maximizes the number of RTL-NRU paths transformed from RTL-DC paths. This task can be reduced to the problem called Maximum-Satisfiability (MAX-SAT). Some approximation algorithms for solving the MAX-SAT problem have been proposed[14] and we can use one of the algorithms. After that, we minimize the number of Xs to which a logical value is assigned because Xs can be used for area and performance optimization of a circuit during logic synthesis.

Reduction to the MAX-SAT problem

Given a set of m clauses $C_1 \ldots C_m$ in conjunctive normal form over n logical variables, the MAX-SAT problem is to find a truth assignment for the logical variables that satisfies a maximum number of clauses. Here we show how to formulate as the MAX-SAT problem using a circuit in Figure 2 as an example.

Suppose a path p=$\langle r2, m1, m2, Add, Sub, r6\rangle$ in Figure 2. The path p is RTL-DC because of $c_{r_2}(t-1) = \mathrm{X}$, $c_{m_1}(t) = \mathrm{X}$, $c_{m_2}(t) = \mathrm{X}$ and $c_{r_6}(t) = \mathrm{X}$. To transform p into RTL-NRU, we need to assign a logic value such as $c_{r_2}(t-1) = \mathrm{H}$ or $c_{m_1}(t) = 0$ or $c_{m_2}(t) = 1$ or $c_{r_6}(t) = \mathrm{H}$. If we assign variables X_1,X_2,X_3,X_4 to each control signal $c_{r_2}(t-1)$,$c_{m_1}(t)$,$c_{m_2}(t)$ and $c_{r_6}(t)$, respectively, then the equation $\overline{X}_1 \vee \overline{X}_2 \vee X_3 \vee \overline{X}_4 = 1$ is obtained. Similarly, path p=$\langle r1, m1, m2, Add, Sub, r6\rangle$ is also RTL-DC because of $c_{r_1}(t-1) = \mathrm{X}$, $c_{m_1}(t) = \mathrm{X}$, $c_{m_2}(t) = \mathrm{X}$ and $c_{r_6}(t) = \mathrm{X}$. If we assign variable X_5 to $c_{r_1}(t-1)$, then the equation $\overline{X}_5 \vee X_2 \vee X_3 \vee \overline{X}_4 = 1$ is obtained. For each RTL-DC path in a circuit, one clause can be formulated like the following equations.

$$\begin{cases} C_1 = \overline{X}_1 \vee \overline{X}_2 \vee X_3 \vee \overline{X}_4 \\ C_2 = \overline{X}_5 \vee X_2 \vee X_3 \vee \overline{X}_4 \\ \quad\quad \vdots \\ C_n = (X_p \vee \overline{X_q} \vee X_r) \wedge (X_u \vee \overline{X_t}) \end{cases}$$

If the total number of RTL-DC paths is n, a set of clauses is $C = \{C_1, C_2, \ldots, C_n\}$. Under an assignment satisfying $C_k = 1$, the corresponding RTL path becomes RTL-NRU. The objective of the MAX-SAT problem is to find an assignment T for maximizing $\sum_{k=1}^{n} C_k$. The assignment T maximizes the number of RTL-NRU paths.

Minimization of logic value assignments

To minimize the number of Xs where logic values are assigned, for the assignment T, we replace each assigned logic value with X unless the number of RTL-NRU paths decreases by the substitution. In particular, we can replace the assigned values that have no contribution to the transformation into RTL-NRU with Xs, and also replace some of assigned values in clauses where T does not satisfy true.

5. Experimental results

This section presents the effectiveness of RTL-DC paths identification and transformation from the identified RTL-DC paths into RTL-NRU. We applied the identification and transformation to ITC'99 benchmarks using a SunFire V490 workstation. Original VHDL codes of ITC'99 benchmarks are written in functional RTL, accordingly control signals for registers and MUXs from a controller at each state are not clear. To clarify control signals for registers and MUXs at each state, we have re-coded B07, B14, B20, B21 and B22 such that each circuit is composed of a controller and a datapath separated from each other. Each re-coded circuit has its original circuit function. We used Design Compiler and Prime Time (Synopsys) as a logic synthesis tool and a timing analysis tool, respectively.

RTL-DC path identification

Table 1 reports the number of identified RTL-DC and RTL-NRU paths. The first column shows each re-coded circuit name with "_s". The second column shows the number of RTL paths starting at datapath register (DR). The third column shows the number of identified RTL-DC paths. The identified RTL-DC paths include RTL-NRU paths that can be identified by the sufficient condition of Theorem 1. For B14_s, our method identified 338 of 349 (96%) RTL paths as RTL-DC. For B20_s, B21_s and B22_s, the ratios of identified RTL-DC paths are almost the same as that of B14_s. This is because B20, B21 and B22 are composed of B14s and/or a modified version of B14s as components. Analysis for another circuit having a different structure such as B15 is also interesting. The CPU time required for identifying RTL-DC paths for each B14_s, B20_s, B21_s and B22_s was a few seconds. The fourth column shows the number of RTL-NRU paths that are identified by Theorem 1. More than half of the RTL paths starting at DR in B14_s, B20_s, B21_s and B22_s were identified as RTL-NRU.

Columns under "SR-ff:Rise" and "SR-ff:Fall" show results for RTL paths starting at SR-ffs with rising transitions and RTL paths starting at SR-ffs with falling transitions, respectively. The SR (state register) in each controller of B14_s, B20_s, B21_s and B22_s consists of one flip-flop because it has two states. The fetch state and the execution state correspond to logic '0' and '1', respectively. Rising transitions at SR-ffs are launched at the execution state and the transitions tend not to be captured at the ending registers because many registers hold their values at the next fetch state. Therefore, there are many RTL-NRU paths for B14_s, B20_s, B21_s and B22_s. In contrast, falling transitions at the fetch state tend to be unspecified whether they are captured or not. For both transitions, our proposed method can identify many RTL paths as RTL-DC.

Transformation from RTL-DC path to RTL-NRU path

In this experiments, for each benchmark circuit, all the identified RTL-DC paths were transformed into RTL-NRU paths by a logic value assignment to Xs of load-enable signals. Accordingly, the number of RTL-NRU paths in each circuit after a logic value assignment is the same as column #RTL-DC. We evaluate the area overhead during logic synthesis due to the logic value assignment. For B14_s, the area is 24,072 compared to the original area 23,990 (one NOT-gate corresponds to 1), that is, the increasing ratio is less than 1%. By logic value assignment, the area of a controller part increases, however that is much smaller than that of a datapath part. Table 2 reports the number of gate-level paths that correspond to RTL-DC and RTL-NRU paths. For b07_s, we extract all the gate-level paths in the circuit. For b14_s,b20_s,b21_s,b22_s, We extracted 10,000 gate-level paths from the longest paths, which has larger propagation delay, by using Prime Time (Synopsys). In the results, all the gate-level paths corresponding to the RTL-DC paths are NRU because all the identified RTL-DC paths were transformed into RTL-NRU. For b14_s, 3,816 paths are newly identified as NRU by finding RTL-SD paths, and totally 8,550 of 10,000 paths are identified as NRU. For b20_s,b21_s,b22_s, about 6,000 of 10,000 paths are identified as NRU.

Over-testing reduction by RTL-DC path identification

While design-for-testability (DFT) techniques are generally used in order to reduce test generation complexity, they induce over-testing problems. In general, DFT techniques make a large number of untestable faults testable. However the delay faults that become testable do not affect circuit performance because the faults was originally untestable. Therefore we consider testing such path to be over-testing. If the path delay faults (PDFs) on the identified NRU paths are excluded from the target of test generation, the over-testing must be reduced. Table 3 shows results of over-tested PDFs for b14_s with the fully-enhanced scan DFT technique and its reduction by using the identified NRU path information. We target 9,580 of 10,000 PDFs due to some constraints of TetraMax ATPG. When a test generation is done for 9,580 faults, 5,938 faults are detected with 668 test patterns. On the other hand, if 5,218 faults, which subtracts the identified NRU PDFs (4,734) from the all PDFs (9,580), are targeted, 3,772 faults are detected with 396 patterns. For the last case, if 1,426 faults , which subtracts 8,550 from 9,580, are targeted, 814 faults are detected with 90 patterns. Now let us consider fault simulation using the generated test patterns. If we apply the generated 668 patterns to all the 9,580 faults, 5,938 faults are detected, that is there is no over-testing reduction. Next, if we apply the generated 396 patterns to the 9,580 faults, 4,106 faults are detected, so that over-testing

Table 1. Number of identified RTL-DC and RTL-NRU paths.

	DR			SR-ff: Rise			SR-ff: Fall		
Circuit	#RTL path	#RTL-DC	#RTL-NRU	#RTL path	#RTL-DC	#RTL-NRU	#RTL path	#RTL-DC	#RTL-NRU
B07_s	21	8	5	63	21	10	63	28	4
B14_s	349	338	190	233	219	219	233	211	0
B20_s	710	676	380	469	438	438	469	422	0
B21_s	710	676	380	469	438	438	469	422	0
B22_s	1059	1020	576	702	657	657	702	633	0

Table 2. Number of gate-level paths corresp. to RTL-DC and RTL-NRU paths.

Circuit	# GL path	#GL path(RTL-DC)	#GL path(RTL-NRU)
B07_s	1,980	376	108
B14_s	10,000	8,550	4,734
B20_s	10,000	5,773	5,374
B21_s	10,000	6,291	5,123
B22_s	10,000	5,604	4,110

Table 3. Reduction of over-tested path delay faults (b14_s).

	Test generation (TG)						Fault simulation (FS)		
Target faults	#Faults	#DT_{TG}	#ATPG_UT	Effic.(%)	#Test pat.	TG time(sec.)	#Faults	#DT_{FS}	#OT reduc.
All faults F_{all}	9,580	5,938	3,642	100	668	328	9,580	5,938	-
F_{all} - $F_{NRU\_bef.}$	5,218	3,772	1,446	100	396	128	9,580	4,106	1,832
F_{all} - $F_{NRU\_aft.}$	1,426	814	612	100	90	83	9,580	1,612	4,326

of 1,832 faults are reduced. For the last case, if the generated 90 patterns are applied to the 9,580 faults, only 1,612 faults are detected, and thus we succeed in over-testing reduction for 4,326 faults. As another advantage of excluding NRU PDFs, test generation time becomes about a quarter compared to no PDFs exclusion.

6. Conclusion

In our previous work[11], we have introduced a concept of register-transfer level non-robust untestable (RTL-NRU) path and have shown that gate-level paths corresponding to RTL-NRU paths are NRU. In this paper, we have presented a concept of RTL synthesis dependent (RTL-SD) path. An RTL-SD path can become either RTL-NRU or RTL testable depending on logic value assignment to unspecified coordinates of output vectors of a controller. If such RTL-SD paths are unintentionally transformed into NRU paths during synthesis, these paths should be identified or tested at gate-level. However, handling paths at gate level is intractable. Thus, we have proposed a method for identifying RTL don't care (RTL-DC) paths, which include RTL-SD and RTL-NRU paths, and transforming the identified RTL-DC paths into RTL-NRU paths. Our approaches can contribute to identification of many gate-level NRU paths within a reasonable amount of time compared to gate-level path identification approaches, and also reduction of over-testing induced by DFT techniques.

Acknowledgment

This work was supported in part by Semiconductor Technology Academic Research Center (STARC) under Research Project, and in part by Japan Society for Scientific Research (B) (No.20300018) and for Young Scientists (Start-up) (No.19800035).

References

[1] G.L. Smith, "Model for delay faults based upon paths," *Proc. International Test Conf.*, pp.342–349, 1985.

[2] A. Krstic and K.T. Cheng, Delay Fault Testing for VLSI Circuits, *Kluwer Academic Publishers*, 1998.

[3] Y. Shao, S.M. Reddy, I. Pomeranz, and S. Kajihara, "On selecting testable paths in scan designs," *IEEE European Test Workshop*, pp.55–58, 2002.

[4] W.Qin, J. Wang, D.M.H. Walker, D. Reddy, X. Lu, Z. Li, W. Shi, and H.Balachandran, "K longest paths per gate test generation for scan-based sequential circuits," *Proc. International Test conf.*, pp.223–231, 2004.

[5] 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," *International Test conf.*, pp.1–9, 2005.

[6] K.T. Cheng and H.C. Chen, "Classification and identification of non-robust untestable path delay faults," *IEEE Trans. Computer-Aided Design of Integrated Circuits and Systems*, vol.15, no.8, pp.854–853, Aug. 1996.

[7] S. Kajihara, K. Kinoshita, I. Pomeranz, and S.M. Reddy, "A method for identifying robust dependent and functionally unsensitizable paths," *Proc. International Conf. on VLSI Design*, pp.82–87, 1997.

[8] S.M. Reddy, S. Kajihara, and I. Pomeranz, "An efficient method to identify untestable path delay faults," *Proc. 10th IEEE Asian Test Symp.*, pp.233–238, 2001.

[9] A. Krstic, S.T. Chakradhar, and K.T. Cheng, "Testable path delay fault cover for sequential circuits," *Proc. European Design Automation conf.*, pp.220–226, 1996.

[10] R. Tekumalla and P.R. Menon, "Identifying redundant path delay faults in sequential circuits," *Proc. 9th International Conf. VLSI Design*, pp.406–411, 1996.

[11] Y. Yoshikawa, S. Ohtake, and H. Fujiwara, "False path identification using RTL information and its application to over-testing reduction for delay faults," *Proc. 14th Asian Test Symp.*, pp.65–68, 2007.

[12] H. Iwata, S. Ohtake, and H. Fujiwara, "An approach to RTL-GL path mapping based on functional equivalence," *Digest of Papers of the 9th IEEE Workshop on RTL and High Level Testing (WRTLT'08)*, To appear, 2008.

[13] Y Explorations, Inc., Explorations tool, http://www.yxi.com/index.html.

[14] B. Korte and J. Vygen, Combinatorial Optimization: Theory and Algotithms, *Springer-Verlag*, 2005.