RTL Don't Care Path Identification and Synthesis for Transforming Don't Care Paths into False Paths

Yuki Yoshikawa[1], Satoshi Ohtake[2] and Hideo Fujiwara[2]
[1]Graduate School of Information Science, 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
Kansai Science City 630-0192, Japan
E-mail:[1]*yosikawa@hiroshima-cu.ac.jp*, [2]{*ohtake, fujiwara*}*@is.naist.jp*

Abstract

Given a register-transfer level (RTL) circuit meeting a design specification, the RTL circuit may contain some functionally unused paths from one register to another within the design specification. Designers of the circuit may know some of the functionally unused paths during design processes, but not all of the functionally unused paths. If designer-unknown and functionally unused paths are identified systematically at RTL and its information about the paths are propagated to gate-level, the corresponding gate-level paths can be eliminated from the target of testing. Otherwise, they must be the target of testing. We consider testing such paths to be futile. In this work, we first present a method of identifying the functionally unused paths using RTL information, and also address synthesis for transforming the identified paths into false paths. As a result, our approaches contribute to identification of many untestable paths and reduction of the futile testing.

1. Introduction

For high speed circuits, delay testing is emphasized to guarantee the timing correctness of circuits. Two fault models to test delay defects are commonly used: the transition fault model and the path delay fault (PDF) model[1, 2]. In recent works, for the transition fault model, a method 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; nevertheless, the path delay fault model is more suitable in order to accurately test accumulative small delays along paths.

A path under the PDF model is defined as an ordered set of gates between two flip-flops. In PDF testing, the delay along the path is compared to a desired system clock period. There are two major problems associated with the PDF testing: 1) circuits may contain an exponential number of paths in the worst case, and 2) many paths are functionally untestable. Thus, several methods of identifying untestable paths have been proposed in the last decade. As approaches at gate-level, 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 may be hard to handle all the exponential number of paths in large scale circuits.

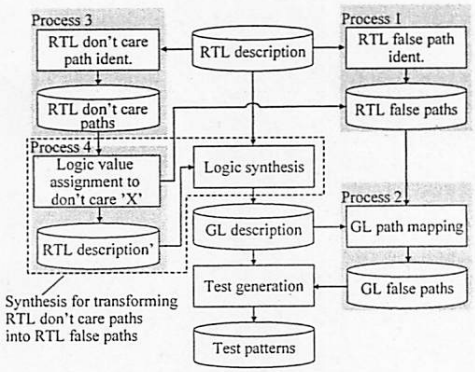

Figure 1. Whole flow of our strategy.

As an approach from upper level, a method of identifying false paths[1] using register transfer-level (RTL) information such as load-enable signals of registers and select signals of multiplexers (MUXs) is presented in our previous work[11]. In the work, paths are dealt with at RTL, called RTL paths, and a sufficient condition to identify RTL paths as RTL false was presented. Figure 1 shows the relation between our previous work and our proposed method in this paper. Our previous work corresponds to process 1 in Figure 1. An RTL path is 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, therefore the identification can be performed in a reasonable amount of time. Moreover, a mapping from RTL paths in an RTL circuit to their corresponding gate-level paths in its gate-level circuit (process 2 in Figure 1) was also addressed. Our experiments have shown that a large number of non-robust untestable paths were identified for some benchmarks.

In this paper, we consider identification of functionally unused RTL paths other than RTL false paths, which are

[1]The work is for identifying non-robust untestable paths but it can be easily extended for identifying false paths.

called RTL don't care paths. Consider an RTL circuit meeting a design specification that is designed at RTL or given as an output of high-level synthesis. If some resources, such as operational modules or system buses, are shared, functionally unused paths between two registers are embedded in the RTL circuit. With regard to control signal values (e.g., load-enable of registers or select signals of MUXs) or data signal values, the values are specified as meeting the design specification. In some control step, some control signal value may not be necessary to be specified for implementing the design specification. By focusing on such unspecified values 'Xs', our path identification method identifies the functionally unused paths as RTL don't care paths. This work corresponds to process 3 in Figure 1.

Gate-level paths corresponding to the identified RTL don't care paths can be false or true depending on an assignment value to 'X' in logic synthesis. If such RTL don't care paths are unintentionally transformed into false paths during synthesis, it is conceivable that identification of the false paths at gate level is intractable. Therefore after the process 3, we transform RTL don't care paths into RTL false paths. Our approach is to determine an assignment to unspecified values at RTL before logic synthesis and make the identified RTL don't care paths RTL false (process 4 in Figure 1). Consequently, our approach contributes to increase in the number of identifiable untestable paths at RTL.

Experimental results show that there exist a lot of RTL don't care paths in some ITC'99 benchmark circuits and most of the identified RTL don't care paths are transformed into RTL false paths.

2. Preliminaries

2.1. RTL circuit and RTL path

Our path identification method approaches to structural RTL designs as shown in Figure 2. A structural RTL design consists of a controller represented by a finite state machine, and a datapath represented by RTL modules such as MUXs, operational modules and registers, and RTL signal lines between them. They are connected to each other by control signal lines and status signal lines. The controller controls control inputs of hardware elements (e.g., load-enable signals of registers and select signals of MUXs) in the datapath. On the other hand, status signals from the datapath are fed into the controller. We assume that state transitions are completely specified for any pair of a state and an input vector.

If a target RTL circuit is described as a functional RTL, it may be difficult to directly extract its structural RTL from the description. In such a case, the information about structure can be obtained during synthesis process. For example, the commercial high-level synthesis tool *Explorations tool* (Y Explorations, Inc.)[12] has a function to transform a functional RTL to its structural RTL.

A path at gate-level is an ordered set of gates $\{g_0, g_1, .., g_n\}$, where g_0 is a primary input or a flip-flop and g_n is a primary output or a flip-flop when we consider a sequential circuit. Also $g_i (1 \leq i \leq n-1)$ is a gate. The number of gate

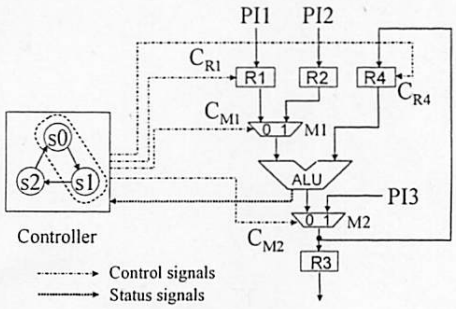

Figure 2. An RTL circuit.

level paths in a circuit is extremely large. When we consider the circuit at RTL, all gate level paths between two registers are dealt with as an RTL path which contains a bundle of paths. In this paper, we use the concept of RTL paths[13]. An RTL path is a path which starts at a primary input or a register and ends at a register or a primary output, which is passing through only combinational modules and has a bit width. The number of RTL paths in an RTL circuit is much smaller than that of gate-level paths in its gate-level circuit.

2.2. Logic synthesis

Our path identification method is applied to RTL paths in an RTL circuit and the information about the paths at RTL is propagated to gate-level paths in a gate-level circuit transformed by a logic synthesis. Then it is necessary to clarify the correspondence of RTL paths to gate-level paths. As one solution to achieve the clarification, we consider a restricted synthesis called module interface preserving-logic synthesis.

Definition 1 (Module interface preserving-logic synthesis[11]) Given an RTL circuit, if logic synthesis transforms each RTL module and each RTL signal line into an individual gate-level netlist and individual one bit signal lines, respectively, the logic synthesis is referred to as *module interface preserving-logic synthesis* (MIP-LS). □

During an MIP-LS for an RTL circuit, each RTL module is transformed to an individual gate-level netlist. Then optimization is performed within each module. Each RTL signal line connecting RTL modules is split to single-bit signal lines. Therefore, for the RTL circuit, the connectivity of all the RTL modules is guaranteed to be propagated to a synthesized gate-level circuit through any MIP-LS.

2.3. RTL path classification

We classify RTL paths into three types: *RTL false paths*, *RTL true paths* and *RTL don't care paths*, depending on whether the corresponding gate-level paths are false. Each definition is as follows.

Definition 2 (RTL false path[11]) An RTL path p in an RTL circuit is *RTL false* if any gate-level path corresponding to p in its gate-level circuit is gate-level false for any logic synthesis. □

Definition 3 (RTL true path) An RTL path p in an RTL circuit is *RTL true* if at least one gate-level path corresponding to p in its gate-level circuit is gate-level true for any logic synthesis. □

Definition 4 (RTL don't care path) An RTL path p in an RTL circuit is *RTL don't care* if there exists a logic synthesis where any gate-level path corresponding to p in its gate-level circuit is gate-level false and there also exists a logic synthesis where at least one gate-level path corresponding to p in its gate-level circuit is gate-level true. □

It may be hard under any logic synthesis to map an RTL path to all the gate-level paths corresponding to the RTL path completely. To achieve the mapping, we restrict logic synthesis to MIP-LS.

Definition 5 (RTL don't care path w.r.t. MIP-LS) An RTL path p in an RTL circuit is *RTL don't care w.r.t. MIP-LS* if there exists an MIP-LS where any gate-level path corresponding to p in its gate-level circuit is gate-level false and there also exists an MIP-LS where at least one gate-level path corresponding to p in its gate-level circuit is gate-level true. □

In this paper, we focus on identification of RTL don't care paths w.r.t. MIP-LS. RTL don't care paths w.r.t. MIP-LS are referred to as RTL don't care paths and denote as RTL-DC in the rest of this paper.

3. RTL-DC path identification

In this section, we recall sufficiency of RTL false paths and present sufficiency of RTL-DC paths.

Let C and C' be an RTL circuit and its gate-level circuit synthesized by an MIP-LS, respectively. Let p be an RTL path in C. Let $F = \{Fj | 1 \leq j \leq m\}$ be a set of flip-flops in C' corresponding to an m bit register R in C. $\tau(R)$ denotes a mapping from R to F. Let Rs and Re be registers that are the starting register and the ending register of p, respectively. Let $M_1, M_2, ..., M_l$ be RTL modules on p. Suppose that p passes through input ports $M_{1in}, M_{2in}, ..., M_{lin}$ and output ports $M_{1out}, M_{2out}, ..., M_{lout}$. Let Q be a set of all gate-level paths between $\tau(Rs)$ and $\tau(Re)$ passing through $M_{1in}, M_{1out}, M_{2in}, M_{2out}, ..., M_{lin}, M_{lout}$ in order. $\delta(p)$ denotes a mapping from p to Q.

Theorem 1 shows sufficiency for which an RTL path p in an RTL circuit C is RTL false. The conditions of Theorem 1 are based on a transition at the starting point of p, its propagation along p and its capture at the ending point. For p, $\forall g \in \delta(p)$ in a gate-level circuit C' is gate-level false if Theorem 1 is satisfied. The conditions of Theorem 1 are used in Theorem 2.

Theorem 1 [11] An RTL path p in an RTL circuit C is RTL false if one of the following three conditions is satisfied for any input sequence.

Condition 1: No transition is ever launched at the register of the starting point of p.

Condition 2: No transition at the starting point of p is ever propagated to the ending point along p.

Condition 3: No value captured into the register at the ending point of p is ever propagated to any primary output. □

Theorem 2 shows sufficiency for which p is an RTL-DC paths.

Theorem 2 An RTL path p in an RTL circuit C is RTL-DC, if one of the following three conditions or one of the three conditions of Theorem 1 is satisfied for any input sequence and there exists at least one input sequence that satisfies one of the following three conditions and no condition of Theorem 1.

Condition 1: It is unspecified whether a transition is launched at the register of the starting point of p or not.

Condition 2: It is unspecified whether the transition at the starting point of p is propagated to the ending point along p or not.

Condition 3: It is unspecified whether the value is captured into the register at the ending point of p. □

A path to transfer data from one register to another is determined by the select signal of each MUX on the path. The timing of data transfer between two registers is determined by the load-enable signal of each register. In this section, from the information on control signals, we show sufficient conditions for identifying RTL-DC paths.

The state register (SR) in a controller represents states of the controller. Control signals for each state and a next state are determined by the value of the SR and status signals from a datapath and/or the PI. We distinguish an RTL path starting at the SR in a controller from the other RTL paths. By considering state assignment, we can know the timing when transitions are launched for each bit of the SR and the directions of the transitions. Therefore we take the information about the state assignment and the directions of the transitions into account for identifying RTL-DC paths starting at the SR. A register in a datapath is referred to as a datapath register (DR).

RTL paths starting at DR/PI

Let Rs and Re be the starting register and the ending register of p, respectively. Let C_{Rs} and C_{Re} be load-enable signals of Rs and Re, respectively. If the load-enable signal of a register is equal to '1', the register loads a value; otherwise, it holds its value. Note that if the register does not have a hold function, we assume that the register has a load enable signal line and the value of that signal is always '1'. If the starting point of p is a PI or the ending point of p is a PO, the PI and the PO are treated as a register with no hold function. Let M_i and C_{M_i} $(1 \leq i \leq n)$ be a MUX on p and its select signal, respectively, where n is the number of MUXs on p. Let C_M^k be the control value of M at time k. When M_i selects the input on p, the value of the select signal is denoted as p_{M_i}. For example, suppose that p is the RTL path R_1-M_1-*ALU*-M_2-R_3 in Figure 2. When M_1 and M_2 select p, $p_{M_1} = 0$ and $p_{M_2} = 0$.

We mainly focus on non-robust untestable paths. Under single fault assumption (a single path is only affected by delay), the delay on a non-robust untestable path does not affect the circuit performance. Thus, in this paper, a gate-level path is said to be false if the path is non-robust untestable. If

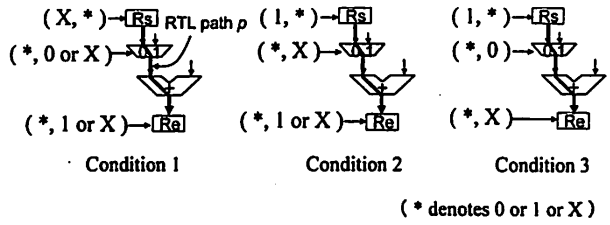

Figure 3. An example of three conditions of Theorem 4

we relax the single fault assumption, the delay on a non-robust untestable path may affect the circuit performance depending on the delays on its off-paths. Our path identification method can also deal with functionally unsensitizable paths by simple extension of the sufficient condition for non-robust untestable.

Theorem 3 shows sufficiency for which an RTL path p in an RTL circuit C is RTL false. The conditions of Theorem 3 are conditions with respect to load-enable signals of registers and select signals of MUXs. The conditions are used in Theorem 4.

Theorem 3 [11] An RTL path p is RTL false if at least one of the following conditions is satisfied for any state transition from time k to $k+1$.

Condition 1: (1) $C_{Rs}^{k} = 0$ or (2) Rs is uncontrollable at time k.

Condition 2: $\exists i | 1 \leq i \leq n, C_{M_i}^{k+1} \neq p_{M_i}$, where n is the number of MUXs on p.

Condition 3: (1) $C_{Re}^{k+1} = 0$ or (2) Re is unobservable at time $k+1$. □

The second condition of Theorem 3 means that at least one MUX on p do not select p at time $k+1$. If the condition is extended such that at least one MUX on p do not select p at time k and $k+1$, any gate-level path corresponding to p is functionally unsensitizable.

Theorem 4 An RTL path p is RTL-DC if one of the following three conditions or one of the three conditions of Theorem 3 is satisfied for any state transition from time k to $k+1$ and there exists at least one state transition that satisfies one of the following three conditions and no condition of Theorem 3.

Condition 1: $C_{Rs}^{k} = X$, where X denotes that no logic value is specified.

Condition 2: $\exists i | 1 \leq i \leq n, C_{M_i}^{k+1} = X$, where n is the number of MUXs on p.

Condition 3: $C_{Re}^{k+1} = X$. □

The conditions of Theorem 4 show the conditions of control signals for two time frames k and $k+1$. Examples of them are shown in Figure 3. If Condition 1 of Theorem 4 is satisfied for a state transition, it is unspecified whether a transition is launched at Rs at time k. If Condition 2 is satisfied for the state transition, it is unspecified whether the transition at Rs is propagated to Re along p at time $k+1$. If Condition 3 is satisfied for the state transition, it is unspecified whether the transition is captured at Re at time $k+1$.

For each state transition that satisfies at least one of the three conditions of Theorem 4 and does not satisfy any condition of Theorem 3, if a logic value is assigned to X such that the state transition satisfies at least one of the three conditions of Theorem 3, p becomes RTL-false. Otherwise, p remains RTL-DC or becomes RTL true.

RTL paths starting at SR-ff

For RTL paths starting at flip-flops in the SR (SR-ff), Theorem 4 can also be applied. The SR in a controller uploads a new value every clock cycle. It means that C_{Rs} always becomes 1 (load). Therefore, RTL paths from the SR-ff to DRs do not satisfy Condition 1 of Theorem 4. Here, we consider transitions for each SR-ff. The relation between states and values of flip-flops is determined by state assignments. We can obtain the information on state assignments during logic synthesis, or designers can also determine state assignments before logic synthesis. From the information on state assignments and state transition, we can know the timing when a transition is launched at each flip-flop. For example, let us consider state assignments to the controller in Figure 2. We assume that the SR in the controller consists of two flip-flops $(FF0, FF1)$, and $(0,0)$, $(0,1)$ and $(1,1)$ are assigned to S_0, S_1 and S_2, respectively. When S_0 transfers to S_1, $FF1$ has a rising transition. If an RTL path p satisfies Theorem 4 at any time k when a transition is launched at the starting point of p, p is RTL-DC.

4. Logic value assignment for transforming RTL-DC paths into RTL false paths

This section presents a method to transform identified RTL-DC paths into RTL false paths by assigning logic values to each unspecified part X. For an RTL-DC path, there exist more than one state transition where at least one of the three conditions of Theorem 4 is satisfied and no condition of Theorem 3 is satisfied. If a logic value is assigned to X such that for all such state transitions, one of the conditions of Theorem 3 is satisfied, then the RTL-DC path becomes RTL false. Note that our approach uses X to increase RTL false paths, however we can also use the X for reduction of area of the circuit or improvement of performance of the circuit during logic synthesis. Therefore, our proposed algorithm transforms RTL-DC paths to RTL false while keeping the number of assignments to X as small as possible. The input, output and optimization for our algorithm are as follows.

Input: A set of RTL-DC paths and a state transition table.

Output: A set of RTL false paths and a state transition table(after a logic value is assigned to each X).

Optimization:

First priority: Maximizing the number of RTL false paths transformed from RTL-DC paths.

Second priority: Minimizing the number of X to which a logical value is assigned.

4.1. Heuristic approach

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 X on control vectors. Searching all the possibility is a hard problem if $|X|$ becomes large. Therefore, we use some heuristics. First we consider logic value assignment to each X that maximizes the number of RTL false 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 already proposed[14] and we use one of the approximation algorithms. After that, we minimize the number of X to which a logical value is assigned. The minimization is that, for each logic value assigned to each X in solving the MAX-SAT problem, the logic value is replaced to X again unless the number of RTL false paths decreases by the substitution. These are our heuristic approaches to the transformation from RTL-DC paths to RTL false paths.

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 4 as an example.

The table in Figure 4 shows a state transition table of an FSM and the circuit shows the datapath part. Suppose the RTL path $P1$ from $R1$ to $R3$ in Figure 4 is RTL-DC. For the state transition $S0 \rightarrow S1$ ($i = 0, j = 1$), $P1$ satisfies the first and third conditions of Theorem 4 because the control signal value of the starting register $R1$ at $S0$ is X ($V_{S_0 1}$ =X) and that of the ending registers $R3$ at $S1$ is also X ($V_{S_1 3}$ =X). By assigning logic value '0'(hold signal) to either $V_{S_0 1}$ or $V_{S_1 3}$, $P1$ satisfies the first or third condition of Theorem 3 for $S0 \rightarrow S1$. Here, we assign variables X_1 and X_2 to $V_{S_0 1}$ and $V_{S_1 3}$, respectively. Then, an equation $\overline{X_1} \vee \overline{X_2} = 1$, which is to satisfy the conditions of Theorem 3, is obtained.

Suppose that $P2$ ($R2$-$M1$-*Add-Sub*-$R4$) is also RTL-DC. For $S1 \rightarrow S2$ ($i = 1, j = 2$), both $P1$ and $P2$ satisfy the second condition of Theorem 4 because the select signal of $M1$ at $S2$ is X ($V_{S_2 5}$ =X). We assign variable X_3 to $V_{S_2 5}$, then for $P1$ and $P2$ at $S1 \rightarrow S2$, equations $X_3 = 1$ and $\overline{X_3} = 1$ are obtained, respectively. By generating such equations for all the state transitions that satisfy only the conditions of Theorem 4, clauses C_1, C_2 for $P1$ and $P2$ are obtained. For each RTL-DC path in a circuit, one clause can be formulated like the following equations.

$$\begin{cases} C_1 = (\overline{X_1} \vee \overline{X_2}) \wedge (X_3) \\ C_2 = (\overline{X_3}) \\ \quad \vdots \\ C_n = (X_p \vee \overline{X_q} \vee X_r) \wedge (X_u \vee \overline{X_t}) \end{cases}$$

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

State transition table

PS	NS (Sj)		Output vector
(Si)	R=0	R=1	($V_{Si1}V_{Si2}V_{Si3}V_{Si4}V_{Si5}$)
S0	S1	S0	XX101
S1	S2	S0	11X10
S2	S0	S0	0111X

PS: Present state, NS: Next state,
R : Reset

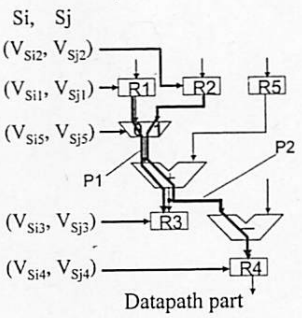

Figure 4. An example of RTL-DC paths

$C_k = 1$, the corresponding RTL path is RTL false. 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 paths transformed from RTL-DC to RTL false.

Minimization of logic value assignments

To minimize the number of logic value assignments, for the assignment T, we replace each assigned logic value with X unless the number of RTL false paths decreases by the substitution. The replacement is done targeting for clauses where T does not satisfy true because T cannot transform RTL-DC paths, which correspond to the clauses that T does not satisfy true, into RTL false. Therefore we try to replace every logic value in such clauses with X.

5. Experimental results

This section presents the effectiveness of RTL-DC paths identification and transformation from the identified RTL-DC paths to RTL false paths. We applied the identification and transformation approaches to ITC'99 benchmarks using a Sun Blade 2000 workstation. Original VHDL codes of ITC'99 benchmarks are written by 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 transition, 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 contains its original circuit function. The controller part is implemented by an FSM and the datapath part is constructed of components and interconnections between them, where a component is a register or a MUX or an operational module. We used Design Compiler and Prime Time (Synopsys) as a logic synthesis tool and a timing analysis tool, respectively.

Table 1 reports the number of identified RTL-DC and RTL false paths using the sufficient condition of Theorem 4 and that of Theorem 3, respectively. The first column shows each re-coded circuit name with "_s". The second, third and fourth columns show results for RTL paths starting at DR (datapath register). The second column shows the number of RTL paths starting at DR. The third column shows the number of RTL-DC paths that are identified by the three conditions in Theorem 4. B07_s has 21 RTL paths starting at DR

Table 1. Number of identified RTL-DC paths using the sufficient condition of Theorem 4.

	DR			SR-ff: Rise			SR-ff: Fall		
Circuit	#RTL path	#RTL-DC	#RTL-F	#RTL path	#RTL-DC	#RTL-F	#RTL path	#RTL-DC	#RTL-F
B07_s	21	3 (14%)	5 (24%)	63	11(17%)	10 (16%)	63	24 (38%)	4 (6%)
B14_s	349	148 (42%)	190 (54%)	233	0 (0%)	219 (94%)	233	211 (91%)	0 (0%)
B20_s	710	296 (42%)	380 (54%)	469	0 (0%)	438 (93%)	469	422 (90%)	0 (0%)
B21_s	710	296 (42%)	380 (54%)	469	0 (0%)	438 (93%)	469	422 (90%)	0 (0%)
B22_s	1059	444 (42%)	576 (54%)	702	0 (0%)	657 (94%)	702	633 (90%)	0 (0%)

Table 2. Number of RTL false paths transformed from RTL-DC paths.

	DR			SR-ff: Rise			SR-ff: Fall		
Circuit	#RTL path	#RTL-DC	#RTL-F	#RTL path	#RTL-DC	#RTL-F	#RTL path	#RTL-DC	#RTL-F
B07_s	21	0 (0%)	8 (38%)	63	0 (0%)	21 (33%)	63	0 (0%)	28 (44%)
B14_s	349	0 (0%)	338 (96%)	233	0 (0%)	219 (94%)	233	0 (0%)	211 (91%)
B20_s	710	0 (0%)	676 (95%)	469	0 (0%)	438 (93%)	469	0 (0%)	422 (90%)
B21_s	710	0 (0%)	676 (95%)	469	0 (0%)	438 (93%)	469	0 (0%)	422 (90%)
B22_s	1059	0 (0%)	1020 (96%)	702	0 (0%)	657 (94%)	702	0 (0%)	633 (90%)

Table 3. Number of gate-level paths corresponding to identified RTL false paths.

	DR			SR-ff: Rise			SR-ff: Fall			Total		
Circuit	#GL path	#GL-F	ratio	#GL path	#GL-F	ratio	#GL path	#GL-F	ratio	#GL path	#GL-F	ratio
B07_s	19,404	4,105	21%	2,908	955	33%	27,688	26,137	94%	50,000	31,197	62%
B14_s	48,668	47,161	96%	774	512	66%	558	216	39%	50,000	47,889	96%
B20_s	46,475	40,864	88%	2,468	1,280	52%	1057	925	88%	50,000	43,069	86%
B21_s	46,136	40,764	88%	2,576	768	30%	1288	1024	80%	50,000	42,556	85%
B22_s	47,828	38,435	80%	1,223	1,219	99%	949	815	86%	50,000	40,469	81%

and our method identified 3 of 21 (14%) RTL paths as RTL-DC . For B14_s, our method identified 148 of 349 (42%) 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 interesting and this is our future works. 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 false paths that are identified by the sufficient condition of Theorem 3. More than half of the RTL paths starting at DR in B14_s, B20_s, B21_s and B22_s were identified as RTL false. B14_s is a processor, and the controller of the processor has only two states: an instruction fetch state and an instruction execution one. Since many registers, except for the instruction register, maintain their values with hold signals at the fetch state, many RTL paths tend to satisfy the first or the third condition of Theorem 3, accordingly our method identified many RTL paths as RTL false. Note that, in this paper, more than one cycle tolerant paths are considered to be false. Dealing with such multi-cycle tolerant paths is our future work.

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 were many RTL false paths for B14_s, B20_s, B21_s and B22_s. Falling transitions tend to have opposite aspects to rising transitions.

Table 2 shows the number of RTL false paths transformed from RTL-DC. For all the benchmarks, this time it happened that all the identified RTL-DC paths were transformed to RTL false paths although it is good trend for us. However, it is important to analyze some trends for other circuits. This further examination is our future work. For B07_s and B14_s, we evaluate their area overhead during logic synthesis which are caused by a logic value assignment to 'X'. For B07_s, the area before logic value assignment was 3,101 (one NOT-gate corresponds to 1) and the area after logic value assignment became 3,105. For B14_s, the area became 24,072 from the original area 23,990. For these two circuits, the ratio of increased area was 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 3 reports the number of gate-level paths corresponding to RTL false paths that are identified from Theorem 3, or transformed from RTL-DC paths that are identified from Theorem 4. We extracted gate-level paths from the longest paths, which has larger propagation delay, by using the " report path " function of Prime Time. For every circuit B07_s, B14_s, B20_s, B21_s and B22_s, we extracted 50,000 gate-level paths from the longest, respectively. The second column shows the number of gate-level paths starting at DRs among the extracted 50,000 paths. The third column shows the number of gate-level false paths starting at DRs that correspond to RTL false paths that we identified. For B14_s, 47,161 of 49,232 (96%) paths were identified as false by our

proposed method. For B20_s, B21_s and B22_s, the ratio of gate-level false paths to the extracted paths was also high. This means our proposed method was able to identify many longest false paths. The next six columns show the evaluation for gate-level paths starting at SR-ffs with rising transitions and starting at SR-ffs with falling transitions, respectively. For B07_s, a large number of gate-level paths with falling transition were identified as false (26,137/27,688) although the ration of RTL false paths to RTL paths starting at SR-ff with falling transition is not so high (28/63). This means that RTL paths corresponding to many longest gate-level paths were identified as false. For each B14_s, B20_s, B21_s and B22_s, the number of gate-level paths starting at SR-ffs was much less than that of paths starting at DRs. The last three columns show the total number of extracted gate-level paths and identified gate-level false paths. These results report a lot of the longest false paths were identified for these benchmarks by our path identification method.

6. Conclusion

In our previous work[11], we have introduced a concept of register-transfer level (RTL) false path and have shown that gate-level paths corresponding to RTL false paths are false. In this paper, we have presented a concept of RTL don't care (RTL-DC) path. An RTL-DC path can become either an RTL false path or an RTL true path depending on logic value assignment to unspecified coordinates of output vectors of a controller. If such RTL-DC paths are unintentionally transformed into false paths during synthesis, it is conceivable that identification of the false paths at gate level is intractable. Thus, we have proposed a method for identifying RTL-DC paths and transforming the identified RTL-DC paths into RTL false paths. The experimental results have shown that our proposed method identifies many RTL-DC paths for some ITC'99 benchmark circuits and most of the identified RTL-DC paths are transformed into RTL false paths. Our approaches can contribute to identification of many gate-level untestable paths and reduction of the futile testing within a reasonable amount of time compared to gate-level path identification approaches.

Acknowledgment

The authors would like to thank Profs. Tomoo Inoue of Hiroshima City University, Michiko Inoue and Tomokazu Yoneda of Nara Institute of Science and Technology for their valuable discussion and their cooperation. This work was supported in part by Semiconductor Technology Academic Research Center (STARC) under the Research Project and in part by Japan Society for the Promotion of Science (JSPS) for Young Scientists (B) (No.17700062).

References

[1] G.L. Smith, "Model for delay faults based upon paths," Proceeding of International Test Conf., pp.342–349, Nov. 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 Conference, 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 Conference, pp.1–9, 2005.

[6] K.T. Cheng and H.C. Chen, "Classification and identification of nonrobust 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. Int. 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," IEEE the 10th Asian test symposium, pp.233–238, Nov. 2001.

[9] A. Krstic, S.T. Chakradhar, and K.T. Cheng, "Testable path delay fault cover for sequential circuits," Proc. European Design Automation Conference, pp.220–226, Sep. 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, Jan 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," IEEE the 14th Asian Test Symposium, Oct. 2007, To appear.

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

[13] M.A. Amin, S. Ohtake, and H. Fujiwara, "Design for hierarchical two-pattern testability of data paths," IEICE Trans. on Information and Systems, vol.E85-D, no.6, pp.975–984, Jun. 2002.

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