PAPER

A Method of Path Mapping from RTL to Gate Level and Its Application to False Path Identification

Hiroshi IWATA[†a)], ***Student Member***, **Satoshi OHTAKE**[†], ***Member***, ***and*** **Hideo FUJIWARA**[†], ***Fellow***

SUMMARY Information on false paths in a circuit is useful for design and testing. The use of this information may contribute not only to reducing circuit area, the time required for logic synthesis, test generation and test application of the circuit, but also to alleviating over-testing. Since identification of the false paths at gate level is hard, several methods using high-level design information have been proposed. These methods are effective only if the correspondence between paths at register transfer level (RTL) and at gate level can be established. Until now, giving restriction on logic synthesis is the only way to establish the correspondence. However, it is not practical for industrial designs. In this paper, we propose a method for mapping RTL false paths to their corresponding gate level paths without such a specific logic synthesis; it guarantees that the corresponding gate level paths are false. Experimental results show that our path mapping method can establish the correspondences of RTL false paths and many gate level false paths.

key words: *false path, high level testing, path mapping, functional equivalence*

1. Introduction

For circuit design and testing, false path information is very valuable since it can be used for reducing circuit area and the time required for logic synthesis, test generation and test application while also minimizing over-testing. From the perspective of design, since design constraints on false paths can be ignored, designers can replace gates on the false paths by smaller gates with larger delay. Furthermore, optimizing paths longer than the critical path can be skipped if they are identified as false paths since they don't have to meet design constraints. Therefore, circuit area and time required for logic synthesis can be made small by using false path information. From the testing point of view, since no test pattern can be generated for path delay faults on false paths, prior false path identification can greatly reduce ATPG time. Furthermore, since some path delay faults on false paths can become testable due to application of design for testability (DFT) and result in over-testing, this can be alleviated by false path identification.

Several false path identification methods at the gate level for combinational circuits [1]–[3] and for sequential circuits [4], [5] have been proposed. However, since it is difficult to apply false path identification methods at the gate level for large circuits containing a tremendous number of paths, some methods using register transfer level (RTL) design information, instead of gate level, have been proposed [6]–[8]. While not specifically targeting false paths, Nourani et al. [6] proposed a method using timing analysis and RTL design information to determine the actual critical path and avoid false paths longer than the true critical path. Yoshikawa et al. [7], [8] defined RTL false paths and proposed a method to identify them. However, these methods are useful only if the correspondence between paths at RTL and paths at gate level can be established. Until now, the correspondence has been established through module interface preserving-logic synthesis (MIP-LS) [7]. Currently, using MIP-LS is the only way to guarantee information on the correspondence. However, it is not practical to restrict synthesis only to MIP-LS.

In this paper, we focus on path mapping from a set of RTL false paths to gate level paths without considering MIP-LS. First, we propose a method of mapping a set of RTL paths to its corresponding gate level paths (this is called path mapping) with an arbitrary logic synthesis independent of false paths. The proposed method maps RTL signal lines composing the RTL paths to gate level nets by using the functional equivalence relation of signal lines (this is called signal line mapping). The effort required for signal line mapping is alleviated by using the uniqueness of a set of the RTL paths and the rough candidate selection method. Because the number of signal lines that uniquely identify a set of RTL paths is much lower than that of whole signal lines in the set of RTL paths, and our path mapping algorithm only needs to map the reduced signal lines, the number of RTL signal lines to be mapped is significantly reduced. Signal line mapping is achieved by checking the equivalence between signal lines and all the gate level nets; however, it is obviously not practical. Therefore, we use the method that finds candidates of the functionally equivalent nets from a gate level circuit by using a diagnosis technique [9].

Since the gate level paths mapped by our method are represented as sets of gate level nets, each gate level path does not need to be fully specified as a path, so we are able to handle bounded paths. This representation is compatible with EDA tools, like Synopsys Design Constraint (SDC). Experimental results show that many RTL paths can be mapped to gate level paths using the proposed method within a reasonable time.

Then, we consider false path mapping. The defini-

Manuscript received October 30, 2009.
Manuscript revised March 1, 2010.

[†]The authors are with the Graduate School of Information Science, Nara Institute of Science and Technology, Ikoma-shi, 630–0192 Japan.
a) E-mail: hiroshi-i@is.naist.jp

DOI: 10.1587/transinf.E93.D.1857

tion of RTL false path in [7] assumes MIP-LS and the assumption guarantees that the corresponding gate level paths are false. In this paper, we show that any corresponding gate level path mapped from the set of RTL false paths by using the proposed method with an arbitrary logic synthesis is false. Experimental results show that our path mapping method can establish the correspondences of RTL false paths and many gate level false paths.

The rest of this paper is organized as follows. Preliminaries are presented in Sect. 2. Section 3 presents the proposed RTL path mapping method. Section 4 shows that the gate level paths mapped from a set of RTL false paths with the proposed method are false. Experimental results are given in Sect. 5. Section 6 concludes the paper.

2. Preliminaries

2.1 Circuit Model

In this paper, we only consider structural RTL designs. A structural RTL design consists of a controller represented by a combinational module and a state register, and a datapath represented by RTL modules and signal lines connecting them, where an RTL module is an operational module, a register or a MUX and a signal line has an arbitrary bit width.

2.2 Gate Level and RTL Path Representation

Definition 1 (Gate level path): An ordered set of gate level nets $\left\{e_1^G, \ldots, e_n^G\right\}$ is called a *gate level path* if it satisfies the following conditions.

1. e_1^G is the net directly connected to a primary input or the output of an FF.
2. e_n^G is the net directly connected to a primary output or the input of an FF.
3. e_i^G $(i = 2, \ldots, n-1)$ is the net connecting the gates having e_{i-1}^G as an input and e_{i+1}^G as an output. □

Definition 2 (Sub gate level path): A subset of a gate level path p^G is called a *sub gate level path of* p^G. □

Definition 3 (RTL path): An ordered set of RTL signal lines $\left\{e_1^R, \ldots, e_n^R\right\}$ is called an *RTL path* if it satisfies the following conditions.

1. e_1^R is the RTL signal line directly connected to a primary input or the output of a register.
2. e_n^R is the RTL signal line directly connected to a primary output or the input of a register.
3. e_i^R $(i = 2, \ldots, n-1)$ is the RTL signal line connecting the modules having e_{i-1}^R as an input and e_{i+1}^R as an output. □

Definition 4 (Sub RTL path): A subset of an RTL path p^R is called a *sub RTL path of* p^R. □

An RTL signal line consists of one-bit signal lines as follows.

Definition 5 (Bit-sliced RTL signal line): For an RTL signal line s, each one bit signal line separated from s is referred to as a *bit-sliced RTL signal line of* s. The i-th bit of s is represented as $s[i]$. □

Definition 6 (Bit-sliced RTL path): An ordered set of bit-sliced RTL signal lines $\left\{e_1^R[k_1], \ldots, e_n^R[k_n]\right\}$ is called a *bit-sliced RTL path* if it satisfies the following conditions.

1. $e_1^R[k_1]$ is the k_1-th bit-sliced RTL signal line directly connected to a primary input or the output of a register.
2. $e_n^R[k_n]$ is the k_n-th bit-sliced RTL signal line directly connected to a primary output or the input of a register.
3. $e_i^R[k_i]$ $(i = 2, \ldots, n-1)$ is the k_i-th bit-sliced RTL signal line connecting the modules having $e_{i-1}^R[k_{i-1}]$ as an input and $e_{i+1}^R[k_{i+1}]$ as an output. □

Definition 7 (Sub bit-sliced RTL path): A subset of a bit-sliced RTL path p^R is called a *sub bit-sliced RTL path of* p^R. □

2.3 Relation between Signal Lines

Here, we first define *signal line cutting*, which is an operation needed for defining functionally equivalent signal lines.

Definition 8 (Signal line cutting): For a combinational circuit C with n inputs, m outputs and an internal signal line s, the following operation is referred to as *cutting* C *on* s.

1. Create the $(n+1)$-th new input port and the $(m+1)$-th new output port.
2. Remove the signal line s.
3. Create connections between $(n+1)$-th input port and the end point of s and between the start point of s and the $(m+1)$-th output port. □

In the following discussion, we represent the combinational circuit resulting from the above operations as $C^*(s)$.

For two functionally equivalent combinational circuits, we define a functional equivalence of signal lines as follows.

Definition 9 (Functionally equivalent signal line): For two functionally equivalent combinational circuits C_1 and C_2 with internal signal lines s_1 and s_2, respectively, s_1 *and* s_2 *are functionally equivalent* if and only if $C_1^*(s_1)$ and $C_2^*(s_2)$ are functionally equivalent. □

In the following discussion, we represent the relation of functional equivalence between signal lines s_1 and s_2 as $s_1 \equiv_l s_2$.

Figure 1 illustrates functionally equivalent signal lines. The signal lines s_1 and s_2 are functionally equivalent if the responses from $C_1^*(s_1)$ and $C_2^*(s_2)$ are identical for any input pattern.

2.4 Relation between Paths

We define the functional equivalence between the sub bit-sliced RTL path and sub gate level path as follows.

Definition 10 (Functionally equivalent path): Sub bit-

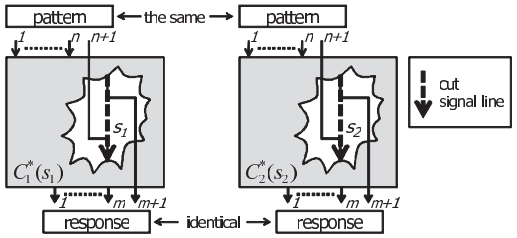

Fig. 1 Functionally equivalent signal lines s_1 and s_2.

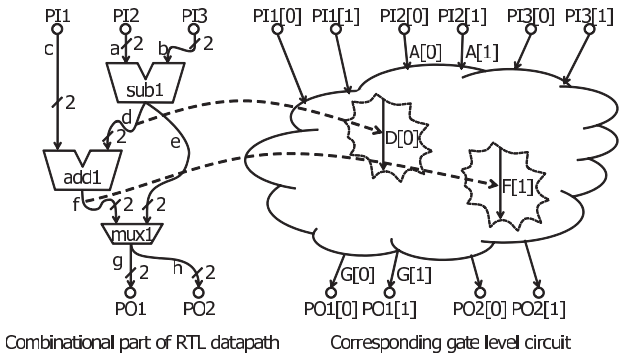

Fig. 2 An example of RTL datapath and its corresponding gate level circuit.

sliced RTL paths and sub gate level paths are simply referred to as *sub paths*. Sub paths $q_1 = \{e_{1_1}, \ldots, e_{1_n}\}$ and $q_2 = \{e_{2_1}, \ldots, e_{2_m}\}$ are *functionally equivalent* if q_1 and q_2 satisfy the following conditions.

1. $n = m$
2. $e_{1_i} \equiv_l e_{2_i}$ $(i = 1, \ldots, n)$ □

For mapping a given RTL path to gate level paths, it is sufficient to map only the RTL signal lines that uniquely identify the RTL path to gate level nets. Therefore, we provide the following definition to alleviate the signal line mapping effort.

Definition 11 (Identification of path): A sub RTL path q^R is said to *uniquely identify* an RTL path p^R if p^R is the only path that properly includes q^R. □

Definition 12 (Identification of path set): A sub RTL path q^R is said to *uniquely identify* a set of RTL paths, P^R, if P^R is the only set of RTL paths that properly includes q^R. □

For example, there exist ten RTL paths in the combinational part of RTL datapath shown in Fig. 2. Consider an RTL path $p^R = \{a, d, f, g\}$. An example of a sub RTL path of p^R is $\{a, d, g\}$. Bit-sliced RTL signal lines of RTL signal line d is $d[0]$ and $d[1]$. An example of a sub bit-sliced RTL path of a bit-sliced RTL path $\{a[1], d[0], f[1], g[1]\}$ is $\{d[0], f[1]\}$. Let $q_1 = \{d[0], f[1]\}$ and $q_2 = \{D[0], F[1]\}$ be a sub bit-sliced RTL path and a sub gate level path, respectively. q_1 and q_2 are functionally equivalent if $d[0] \equiv_l D[0]$ and $f[1] \equiv_l F[1]$, and the number of signal lines in q_1 and that in q_2 are the same. For p^R, both of sub RTL paths $\{a, d, g\}$ and $\{a, f, g\}$ uniquely identify p^R. Similarly, for a set of RTL paths, $P^R = \{\{a, d, f, g\}, \{a, e, g\}\}$, a sub RTL path $\{a, g\}$ uniquely identifies P^R.

3. Proposed Method of Path Mapping

In this section, we formulate the path mapping problem and present a solution to the problem independent of false paths. Consideration of false paths is described in Sect. 4.

3.1 Path Mapping Problem

We formulate the path mapping problem as a problem to find a set of gate level paths corresponding to a set of RTL paths.

For solving the path mapping problem, it is sufficient to consider only the RTL combinational circuit C^R, which is the combinational part of a given structural RTL design S^R, and the gate level combinational circuit C^G, which is the combinational part of a gate level design synthesized from S^R. We assume that for each input or output (I/O) signal line of C^R, there exists exactly one I/O signal line, which is functionally equivalent to the I/O signal line of C^R, of C^G. The correspondence between I/O signal lines of C^R and that of C^G is called *I/O mapping information*. We cannot apply the path mapping algorithm if the I/O mapping information is not available. However, the I/O mapping information of C^R and C^G can be obtained by preserving all the bits of the registers in S^R during logic synthesis. Since the preservation is common for logic synthesis of structural RTL designs, the assumption is reasonable.

Definition 13 (Path mapping problem):

Input
- C^R: an RTL combinational circuit
- C^G: a gate level circuit that is functionally equivalent to C^R
- The I/O mapping information between C^R and C^G
- P^R: a set of RTL paths

Output $P^G = \bigcup_{i=0}^{n} \bigcup_{j=0}^{m_i} P^G_{ij}$, where P^G_{ij} is defined as follows. Let q^R_i $(i = 1, \ldots, n)$ be a sub RTL path that uniquely identifies P^R and q^R_{ij} $(j = 1, \ldots, m_i)$ be a sub bit-sliced RTL path of q^R_i where n and m_i are the numbers of the sub RTL paths of P^R and combinations of bit-sliced RTL paths obtained by specifying the bit portion of every RTL signal line on q^R_i, respectively. Let q^G_{ij} be a sub gate level path that is functionally equivalent to q^R_{ij}. P^G_{ij} is a set of gate level paths including q^G_{ij}. □

3.2 Path Mapping Algorithm

We propose an algorithm solving the path mapping problem as follows. The algorithm establishes correspondences between a set of RTL paths, P^R, and a set of gate level paths, P^G.

1. Generate the minimum sub RTL path q^R_i $(i = 1, \ldots, n)$ that uniquely identifies P^R.

2. Try to obtain a gate level net $e^G_{ij_k}$ that is functionally equivalent to each bit-sliced RTL signal line $e^R_{ij_k}$ ($k = 1, \ldots, l$), where $e^R_{ij_k}$ is an element of a bit-sliced RTL path q^R_{ij} ($j = 1, \ldots, m_i$) of q^R_i, m_i is the number of combinations of bit-sliced RTL paths obtained by specifying the bit portion of every RTL signal line on q^R_i, and l is the number of RTL signal lines on q^R_i. Go to step 3 if $e^G_{ij_k}$ is obtained for all j and k of at least one sub RTL path q^R_i, i.e., all the RTL signal lines on some sub RTL path are mapped to gate level nets. (Otherwise, all q^R_i must be tried.)
3. For each sub gate level path $\left\{e^G_{ij_1}, \ldots, e^G_{ij_l}\right\}$ such that every $e^G_{ij_k}$ ($k = 1, \ldots, l$) is mapped from $e^R_{ij_k}$ of the corresponding sub bit-sliced RTL path $\left\{e^R_{ij_1}, \ldots, e^R_{ij_l}\right\}$ in the previous step, find all the gate level paths identified by $\left\{e^G_{ij_1}, \ldots, e^G_{ij_l}\right\}$. The set of all the obtained gate level paths is referred to as P^G_{ij}.
4. Calculate $P^G = \bigcup_{i=0}^{n} \bigcup_{j=0}^{m_i} P^G_{ij}$.

Note that we obtain the set of the minimum sub RTL paths uniquely identifying P^R exhaustively as follows. The minimum size of sub RTL paths uniquely identifying P^R is sought such that s is increased one by one by when some sub RTL path of size s uniquely identifies P^R and then all the sub RTL paths, which uniquely identify P^R, of size s are enumerated, where s is an integer and is initially 0. Since the number of modules in a circuit at RTL is very small, it is conceivable that the time required for obtaining the set of the minimum sub RTL paths is very short. This will be evaluated in the experimental results. The signal line mapping in step 2 is described in the next subsection. We assume that at most one gate level net is functionally equivalent to a bit-sliced RTL signal line for simplifying the algorithm description. In our experiments reported in Sect. 5, we did not face a case where more than one gate level net is mapped. However, we can handle multiple nets by taking into account all the paths that go through the nets. In steps 3 and 4, not all gate level paths need to be listed; it is not practical. Instead, paths are represented just by specifying nets, $\left\{e^G_{ij_1}, \ldots, e^G_{ij_l}\right\}$, that are passed through. This representation is compatible with EDA tools like SDC description.

Here, we show an example of path mapping for the RTL circuit and the gate level circuit shown in Fig. 2. We perform path mapping algorithm for a set of paths $P^R = \{\{a, d, f, g\}\}$. In step 1, minimum sub RTL paths, $q^R_1 = \{a, d, g\}$ and $q^R_2 = \{a, f, g\}$, uniquely identifying P^R are obtained. In step 2, we first try to find the functionally equivalent signal lines for $a[0]$, $a[1]$, $d[0]$, $d[1]$, $g[0]$ and $g[1]$ on q^R_1. Suppose that functionally equivalent gate level nets $A[0]$, $A[1]$, $D[0]$, $G[0]$ and $G[1]$ are found for $a[0]$, $a[1]$, $d[0]$, $g[0]$ and $g[1]$, respectively, and that for $d[1]$ is not found. Since the algorithm could not map all the RTL signal lines on q^R_1, we repeat step 2 for the other sub RTL path, q^R_2, i.e., we perform signal line mapping for $f[0]$ and $f[1]$. Suppose that gate level net $F[1]$ which is functionally equivalent net for $f[1]$ is found and the functionally equivalent gate level net for $f[0]$ is not found. We operate step 3 and 4 because all the minimum sub RTL paths uniquely identifying P^R have been tried so far. The mapped gate level paths are obtained by specifying the nets passed through as follows: $\{A[0], D[1], G[0]\}$, $\{A[0], D[1], G[1]\}$, $\{A[1], D[1], G[0]\}$, $\{A[1], D[1], G[1]\}$, $\{A[0], F[0], G[0]\}$, $\{A[0], F[0], G[1]\}$, $\{A[1], F[0], G[0]\}$ and $\{A[1], F[0], G[1]\}$.

3.3 Signal Line Mapping

In this section, we formulate the problem finding functionally equivalent nets. Then, we will show an algorithm for solving the problem. Signal line mapping algorithm is used in the proposed path mapping algorithm.

3.3.1 Signal Line Mapping Problem

We formulate the *signal line mapping problem* to find a set of nets, which is functionally equivalent to a bit-sliced RTL signal line in an RTL circuit, in a gate level circuit.

Definition 14 (Signal line mapping problem):

Input
- C^R: an RTL combinational circuit
- C^G: a gate level circuit that is functionally equivalent to C^R
- The I/O mapping information between C^R and C^G
- $e^R[k]$: the k-th bit-sliced RTL signal line of an RTL signal line e^R in C^R

Output $E^G = \left\{e^G | e^G \equiv_l e^R[k]\right\}$ where e^G is a net in C^G □

3.3.2 Signal Line Mapping Algorithm

Given an RTL combinational circuit C^R and a gate level combinational circuit C^G, checking functional equivalence between a bit-sliced RTL signal line $e^R[k]$ in C^R and a gate level net e^G in C^G can be performed by applying all the possible input patterns to both circuits $C^{R*}(e^R[k])$ and $C^{G*}(e^G)$, and comparing their output responses. This is achieved by applying equivalence checking [10], [11] for $C^{R*}(e^R[k])$ and $C^{G*}(e^G)$. However, it is not practical to explicitly check the functional equivalence for all the possible combinations between $e^R[k]$ and e^G in C^G.

Ravi et al. [9] proposed a method of finding candidates for functionally equivalent nets of a given bit-sliced RTL signal line using fault diagnosis techniques. In this paper, their method is utilized to solve the signal line mapping problem. More specifically, their method injects a stuck-at fault on the bit-sliced RTL signal line and finds the stuck-at faults, which have identical behavior of the fault under the test patterns, in the gate level circuit. The faults in the gate level circuit and the fault in the RTL circuit are said to be *equivalent*. A necessary condition of functional equivalence is that the responses of the RTL circuit and the gate

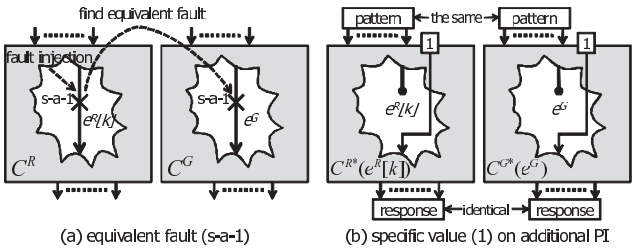

Fig. 3 Relation between equivalent faults and functionality of respective signal lines.

level circuit are identical when value v's are fixed to $e^R[k]$ and e^G, respectively (see Fig. 3 (b)). It is the same situation when s-a-v faults are assumed to be presented on $e^R[k]$ and on e^G, respectively (see Fig. 3 (a)). To make our signal line mapping algorithm complete, we perform functional equivalence checking for $e^R[k]$ and each of the mentioned candidate nets e^G. The overall algorithm to solve the signal line mapping problem is shown in the following.

1. Generate a complete test set T for all the testable stuck-at faults in C^G.
2. For each $v \in \{0, 1\}$, the following two steps are performed.
   a. Obtain a set of faulty output responses R_{fv} by applying T to the RTL circuit C^R with an injected s-a-v fault on the given bit-sliced RTL signal line $e^R[k]$.
   b. Find all the single s-a-v faults of C^G such that all the faulty circuits induced by the faults have the same output responses R_{fv} when T is applied to these circuits. A set of the nets having equivalent faults is referred to as E^{Gv}.
3. Obtain $E^G = E^{G0} \cap E^{G1}$.
4. For each $e^G \in E^G$, create $C^{R*}(e^R[k])$ and $C^{G*}(e^G)$ by cutting C^R and C^G on $e^R[k]$ and e^G, respectively.
5. Perform equivalence checking for $C^R(e^{R*}[k])$ and $C^{G*}(e^G)$ and eliminate e^G from E^G if they are not functionally equivalent.

Steps 1 to 3 are the same as the procedure for finding a functionally equivalent signal line by using the fault diagnosis technique in [9]. In [9], the complete test set T for the detectable faults in a gate level circuit is used as the input patterns for fault diagnosis. The procedure first finds s-a-0 (resp. 1) faults in C^G that are equivalent to the s-a-0 (resp. 1) fault injected on $e^R[k]$ under the test set T. Then the procedure selects gate level nets that have both s-a-0 and s-a-1 faults as the candidates of equivalent nets. These nets obtained by the steps satisfy the necessary condition of the functional equivalence. Finally, steps 4 and 5 are performed to guarantee sufficiency.

The completeness of the overall algorithm is shown in Theorem 1. In Step 1 of the signal line mapping algorithm, we employ a complete test set. The incompleteness of the test set does not affect the correctness of the signal line mapping. However, it is desirable because the number of trials of equivalence checking (steps 4 and 5) is reduced if the number of candidates is efficiently decreased by the fault diagnosis method (steps 2 and 3). Here, we assume that the fault diagnosis technique used in the algorithm can report all the suspected faults, i.e., it never misses any equivalent fault under the given input patterns. If we employ an incomplete diagnosis tool which misses suspected faults, the algorithm cannot find existing functionally equivalent nets in gate level circuit during the signal line mapping process. From the perspective of path mapping, this may lead loss of the identifiable functionally equivalent gate level paths. If the diagnosis tool reports inequivalent faults, it does not affect the correctness of the signal line mapping because equivalence checking is performed for all the functionally equivalent candidates.

Theorem 1: Given an RTL combinational circuit C^R, its synthesized gate level circuit C^G and a bit-sliced RTL signal line $e^R[k]$ in C^R. Any $e^G \in E^G$ is functionally equivalent to $e^R[k]$ if and only if E^G is the set of gate level nets obtained by the signal line mapping algorithm. □

[Proof] First, we show that Steps 1 to 3 guarantee that the primary outputs of C^G and C^R have the same response for any input pattern in T when e^G and $e^R[k]$ have the same value, which is the necessary condition of functional equivalence. Let C^R has n inputs ($x^R[i]$ ($i = 1, \ldots, n$)) and m outputs ($z^R[i]$ ($i = 1, \ldots, m$)). $C^{R*}(e^R[k])$ has $n + 1$ inputs ($x^{R*}[i]$ ($i = 1, \ldots, n + 1$)) and $m + 1$ outputs ($z^{R*}[i]$ ($i = 1, \ldots, m + 1$)). We inject an s-a-v fault on $e^R[k]$ in C^R where $v \in \{0, 1\}$. For any $t \in T$, the output response from $z^R[1], \ldots, z^R[m]$ of C^R with the s-a-v obtained by applying t to C^R with the fault and that from $z^{R*}[1], \ldots, z^{R*}[m]$ of $C^{R*}(e^R[k])$ obtained by applying $t\&v$ to C^{R*} are identical where "$a\&b$" denotes concatenation of vectors a and b.

Let C^G has n inputs ($x^G[i]$ ($i = 1, \ldots, n$)) and m outputs ($z^G[i]$ ($i = 1, \ldots, m$)). $C^{G*}(e^G)$ has $n + 1$ inputs ($x^{G*}[i]$ ($i = 1, \ldots, n + 1$)) and $m + 1$ outputs ($z^{G*}[i]$ ($i = 1, \ldots, m + 1$)). We inject an s-a-v fault on signal line e^G in C^G. For any $t \in T$, the output response from $z^G[1], \ldots, z^G[m]$ of C^G with the s-a-v obtained by applying t to C^G with the fault and that from $z^{G*}[1], \ldots, z^{G*}[m]$ of $C^{G*}(e^G[k])$ obtained by applying $t\&v$ to C^{G*} are identical.

Consequently, for any input pattern of T, the output responses from $z^{R*}[1], \ldots, z^{R*}[m]$ and $z^{G*}[1], \ldots, z^{G*}[m]$ of $C^{R*}(e^R[k])$ and $C^{G*}(e^G)$, respectively, are the same because C^R and C^G are functionally equivalent and s-a-v faults on $e^R[k]$ and e^G are equivalent under T. It is a necessary condition of functional equivalence between $e^R[k]$ and e^G.

From the assumption of fault diagnosis, all e^G are able to be gotten as E^G where e^G satisfies the above conditions. It is obvious that E^G properly include all the gate level nets that are functionally equivalent to $e^R[k]$. Therefore, we only need to show that steps 4 and 5 can exclude nets if and only if the nets are not functionally equivalent to $e^R[k]$. Clearly, e^G is eliminated from E^G if e^G is not functionally equivalent to $e^R[k]$. Otherwise, it is not eliminated. Thus the theorem holds true. □

4. RTL False Path Mapping

From the testing point of view, it is important to identify non-robust untestable paths since commercial ATPG tools cannot generate test patterns detecting path delay faults with functionally sensitizable condition. In this section, we show mapping non-robust untestable paths at RTL to non-robust untestable paths at gate level as an application of the proposed path mapping to false path identification, i.e., in this paper, we consider non-robust untestable paths to be false. In [12], Yoshikawa et al. defined non-robust untestable paths for RTL circuits as follows.

Definition 15 (RTL non-robust untestable path): An RTL path p in an RTL circuit S^R 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 S^G synthesized from S^R, where $\delta(p)$ is a set of gate level paths corresponding to p. □

In order to guarantee the correspondence between RTL-NRU and $\delta(p)$, restricted logic synthesis called *module interface preserving logic synthesis (MIP-LS)* is employed.

Under the assumption of logic synthesis, they also provide a sufficient condition of RTL-NRU based on control signals of MUXes and registers. Here, we review the condition. For a given path $p = \{e_1, \ldots, e_n\}$ in an RTL circuit, intuitively, the condition is as follows. The path p is RTL-NRU if at least one of the following is satisfied for any input sequence and any t: (1) there is no controllability to make a transition on the starting register, which drives e_1 in cycles between t and $t + 1$ (if any); (2) the value at e_{i+1} is independent of the value at e_i for some i ($i = 1, \ldots, n$) in $t + 1$ (if any); (3) the value appeared at e_n is not captured on the ending register in $t + 2$ (if any); and (4) the value appeared at e_n in $t + 2$ and stored in the register following e_n does not affect any primary output (if any). These are checked only by examining control signal values of MUXes and registers supplied from the controller. Notice that in their RTL circuit model, for an RTL circuit, state transitions of the controller are known and are completely specified for all the pairs of states and input vectors. Detailed description is available in [12].

The condition means that no transition can be propagated through an RTL-NRU path, which is identified based on the condition, in non-robust sensitization criteria or the response captured at the ending register cannot be observed. If we can remove the assumption of logic synthesis, we can utilize the identification method reported in [7] for more general circuits synthesized without the restriction. Therefore, we obtain the following theorem.

Theorem 2: For an RTL-NRU path p^R, in an RTL circuit S^R, any $p^G \in P^G$ that is mapped from p^R with our path mapping method is gate level NRU. □

[Proof] Suppose that p^R and p^G consist of $\left\{e_1^R, \ldots, e_n^R\right\}$ and $\left\{e_1^G, \ldots, e_m^G\right\}$, respectively, and each RTL signal line e_i^R of p^R has arbitrary bits. From the sufficient condition of RTL-NRU, for an input sequence, p^R satisfies at least one of the four conditions, as described above. From the assumption of the existence of I/O mapping information, combinational parts of S^R and S^G are functionally equivalent. Then, we can say the following. If it satisfies (1), all the bit-sliced signal lines of e_1^R cannot have transitions in cycles t to $t + 1$. Therefore, e_1^G cannot have any transition in t to $t + 1$. If it satisfies (3) or (4), values of all the bit-sliced signal lines of e_n^R cannot be observed. Therefore, any values on e_m^G at $t + 1$ cannot be observed. From the I/O mapping information, functional equivalence of the combinational parts of S^R and S^G, and functional equivalence of internal signal lines, we can say the following. If it satisfies (2), all the bit-sliced signal lines of e_n^R cannot have transitions that are from e_1^R and by way of e_i^R (if any) in t to $t + 1$, where e_i^R is necessary to be mapped. Therefore, e_m^G cannot have any transition that is from e_1^G and by way of e_j^G (if any) in t to $t + 1$, where e_j^G is mapped from e_i^R. Thus, the theorem holds true. □

By this theorem, we can treat gate level NRU paths in a gate level circuit synthesized with an arbitrary logic synthesis (without restricting logic synthesis to MIP-LS) through the proposed path mapping method.

5. Experimental Results

In this section, we show experimental results for evaluating our RTL path mapping method by mapping RTL paths and RTL false paths identified with the method proposed in [7]. We used three RTL benchmark circuits, LWF, Tseng and Paulin and an industrial circuit, MPEG. In these experiments, we used only the datapath part of each circuit and tried to map all the paths in the datapath. Table 1 shows the circuit characteristics of the circuits. Columns "#bit", "#PI", "#PO" and "#reg" show the bit width, the number of primary inputs, that of primary outputs and that of registers, respectively. Sub columns "MIP-LS" and "Arbitrary" under "Area (#gates)" show the circuit area synthesized by MIP-LS [7] and that without restriction, respectively. From the area comparison, we confirmed that our method eliminates the impact on logic synthesis results. In these experiments, we used Synopsys DesignCompiler to perform logic synthesis, Synopsys TetraMax to generate test patterns for gate level circuits synthesized with "Arbitrary", Cadence Encounter Test and Diagnostics as a fault diagnostic engine, Synopsys Formality to perform equivalence checking and Synopsys PrimeTime to enumerate the gate level paths on Sun Microsystems Sun Fire X4100 (Opteron 256 (3 GHz), 16 GB memories).

Table 1 Circuit characteristics.

Circuit	#bit	#PI	#PO	#reg	Area (#gate)	
					MIP-LS	Arbitrary
LWF	16	2	2	5	1,571	1,467
Tseng	8	3	2	6	1,357	1,077
Paulin	8	2	2	7	1,590	1,303
MPEG	8	5	16	241	38,183	28,454

We use the RTL path mapping ratio $Pmr = \frac{|P^{RT}|}{|P^R|} \times 100[\%]$ as an evaluation criterion, where $|P^R|$ is the total number of RTL paths in the datapath and $|P^{RT}|$ is the number of RTL paths mapped. Furthermore, to evaluate in more detail, we consider bit-sliced RTL paths in the datapath. We use the bit-sliced RTL path mapping ratio $Pmr_b = \frac{|P_b^{RT}|}{|P_b^R|} \times 100[\%]$, where $|P_b^R|$ is the total number of bit-sliced RTL paths in the datapath and $|P_b^{RT}|$ is the number of bit-sliced RTL paths mapped. Table 2 shows the path mapping ratios, bit-sliced path mapping ratios and time required for the mapping.

Table 3 shows the signal line and path mapping results in detail. Rows "#Ptotal", "#Punique", "#Stried", "#Smapped" and "#Pmapped" show the total number of RTL paths, the number of paths uniquely identified with the I/O mapping information, the number of RTL signal lines targeted by signal line mapping, the number of RTL signal lines mapped, i.e., the gate level nets that are functionally equivalent to the bit-sliced RTL signal lines found, and the number of the RTL paths mapped. Columns "RTL" and "bsRTL" under each circuit name mean bundled RTL and bit-sliced RTL, respectively. The number of bit-sliced RTL paths in an RTL path was calculated based on the number of Cartesian product of bit widths of signal lines composing the RTL path. Thanks to Definition 11 (unique identification of path), most of the RTL paths were able to be mapped only by using I/O mapping information or CPU time was able to be saved. To identify 5 and 2 paths in LWF and Tseng, respectively, we needed to perform signal line mapping 80 and 40 times, respectively. Since the average times of the signal line mapping for these circuits are 0.35 and 0.54 seconds, respectively, the total times required for the path mapping of these circuits are almost the same. The proposed method achieved 90.9% RTL path mapping ratio and 92.8% bit-sliced RTL path mapping ratio, in average. Here, we discuss the paths that are not mapped. (Bit-sliced) RTL paths that were not able to be mapped to the gate level paths existed because the algorithm was not able to find any signal line needed for path mapping, i.e., there existed no functionally equivalent net in the gate level circuits.

Table 4 shows the result of false path mapping and the time required for this mapping. Rows "#Ptotal", "#Pfalse", "Ratio", "Total", "Unique", "Ravi", "FEchk", "Pwhole" and "Pfalse" show the total number of paths, the number of false paths, the ratio of #Pfalse to #Ptotal, the total time required for false path mapping, the time required for finding candidates of functionally equivalent signal lines, the time required for equivalence checking, the time required for enumerating the whole paths in the gate level circuit and the time required for enumerating the false paths mapped, respectively. Columns "RTL" and "Gate level" under each circuit name mean the number of paths in RTL and the ones in gate level, respectively. Many gate level false paths were available with our proposed path mapping method in practical time without considering MIP-LS.

On the other hand, a sequential ATPG algorithm can identify false paths at gate level. However, sequential ATPG tools cannot identify them in a practical amount of time. For example, as reported in [8], TetraMax took about 50 hours to identify 10,000 false paths of Paulin. Since the RTL false path identification method proposed in [7] and our path mapping method took less than 1 second for several circuits, our high level identification approach is very effective.

Table 5 shows false path mapping results in detail. Rows "#Pfalse", "#Punique", "#Stried" and "#Smapped"

Table 2 Path mapping results.

	LWF	Tseng	Paulin	MPEG
Pmr[%]	73.7	90.0	100.0	100.0
Pmr_b[%]	74.2	96.8	100.0	100.0
CPU[sec]	28.14	21.74	0.30	0.10

Table 3 Signal line and path mapping results in detail.

	LWF		Tseng		Paulin		MPEG	
	RTL	bsRTL	RTL	bsRTL	RTL	bsRTL	RTL	bsRTL
#Ptotal	19	4,600,384	20	36,448	29	123,600	606	326,176
#Punique	14	3,412,544	18	31,840	29	123,600	606	326,176
#Stried	5	80	5	40	0	0	0	0
#Smapped	0	13	0	12	-	-	-	-
#Pmapped	14	3,415,360	18	35,296	29	123,600	606	326,176

Table 4 False path mapping results.

	LWF		Tseng		Paulin		MPEG	
	RTL	Gate level	RTL	Gate level	RTL	Gate level	RTL	Gate level
#Ptotal	19	1,845,916	20	856,116	29	2,307,064	606	1,784,824
#Pfalse	5	470,300	6	418,752	13	1,610,968	32	16
Ratio [%]	26.32	25.48	30.00	48.91	44.83	69.83	5.28	0.00
Total [s]		15.36		21.73		0.27		1.72
Unique [s]		0.21		0.24		0.27		1.72
Ravi [s]		15.15		17.07		0.00		0.00
FEchk [s]		0.00		4.42		0.00		0.00
Pwhole [s]		93.21		37.21		103.39		303.65
Pfalse [s]		24.26		19.07		73.53		0.22

Table 5 Details of the false path mapping.

	LWF	Tseng	Paulin	MPEG
#Pfalse	5	6	13	32
#Punique	4	5	13	32
#Stried	32	16	0	0
#Smapped	0	7	-	-

show the number of RTL false paths, the number of paths uniquely identified with the I/O mapping information, the number of bit-sliced RTL signal lines targeted by signal line mapping, and the number of bit-sliced RTL signal lines mapped, respectively. Therefore, we can say that the proposed method finds almost all gate level false paths corresponding to the given RTL false paths.

6. Conclusions

Establishing the correspondence between an RTL circuit and its synthesized gate level circuit is important for high level testing approaches. In this work, we focused on correspondence between paths in the RTL circuit and paths in the gate level circuit. Existence of the correspondence enables the technologies that handle a path at RTL as a bundle of a tremendous number of paths in the gate level circuit. There are methods to identify false paths using RTL design information [7], [8], which is feasible only if RTL paths can be mapped into gate level paths. The method can quickly identify false paths using RTL information under the assumption that correspondence between RTL paths and gate level paths is available. Until now, it has been guaranteed only by a restricted logic synthesis.

In this paper, we proposed a method to establish correspondence between a set of RTL paths and gate level paths without restricting logic synthesis. To the best of our knowledge, this is the first work that tackles RTL to gate level path mapping. Furthermore, we showed that RTL false paths identified by [7] can be mapped to gate level false paths with our proposed method. In our experiments, the proposed path mapping method was utilized as a false path mapping procedure, and many false paths were able to be found in a circuit synthesized with an arbitrary logic synthesis by using our proposed path mapping method.

Acknowledgments

The authors would like to thank Profs. Michiko Inoue and Tomokazu Yoneda of Nara Institute of Science and Technology and Yuki Yoshikawa of Hiroshima City University for valuable discussion and 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) under Grants-in-Aid for Scientific Research (B) (No. 20300018).

References

[1] K.T. Cheng and H.C. Chen, "Classification and identification of non-robust untestable path delay faults," IEEE Trans. Comput.-Aided Des. Integr. Circuits Syst., vol.15, no.8, pp.845–853, Aug. 1996.

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

[3] Y. Shao, S.M. Reddy, S. Kajihara, and I. Pomeranz, "An efficient method to identify untestable path delay faults," Proc. 10th Asian Test Symposium, pp.233–238, 2001.

[4] A. Krstić, S.T. Chakradhar, and K.T.T. Cheng, "Testable path delay fault cover for sequential circuits," Design Automation Conference, with EURO-VHDL '96 and Exhibition, Proc. EURO-DAC '96, European, pp.220–226, Sept. 1996.

[5] R. Tekumalla and P. Menon, "Identifying redundant path delay faults in sequential circuits," Proc. Ninth International Conference on VLSI Design, pp.406–411, Jan. 1996.

[6] M. Nourani and C.A. Papachristou, "False path exclusion in delay analysis of RTL structures," IEEE Trans. Very Large Scale Integr. (VLSI) Syst., vol.10, no.1, pp.30–43, Feb. 2002.

[7] Y. Yoshikawa, S. Ohtake, and H. Fujiwara, "False path identification using RTL information and its application to over-testing reduction for delay faults," 16th Asian Test Symposium, ATS '07, pp.65–68, Oct. 2007.

[8] Y. Yoshikawa, S. Ohtake, T. Inoue, and H. Fujiwara, "Fast false path identification based on functional unsensitizability using RTL information," Asia and South Pacific Design Automation Conference, ASP-DAC 2009, pp.660–665, Jan. 2009.

[9] S. Ravi, I. Ghosh, V. Boppana, and N.K. Jha, "Fault-diagnosis-based technique for establishing RTL and gate-level correspondences," IEEE Trans. Comput.-Aided Des. Integr. Circuits Syst., vol.20, no.12, pp.1414–1425, Dec. 2001.

[10] A. Kuehlmann and F. Krohm, "Equivalence checking using cuts and heaps," Proc. 34th Design Automation Conference, pp.263–268, June 1997.

[11] Synopsys, inc, Formality User Guide, c-2009.06 ed., June 2009.

[12] Y. Yoshikawa, S. Ohtake, T. Inoue, and H. Fujiwara, "A synthesis method to alleviate over-testing of delay faults based on RTL don't care path identification," 27th IEEE VLSI Test Symposium, VTS '09, pp.71–76, May 2009.

Hiroshi Iwata received his B.S. degree in information systems engineering from Nara National College of Technology, Japan, in 2007. In 2008, he received his M.E. degree in information science from Nara Institute of Science and Technology, Japan and is currently a Ph.D. student there. His research interests include VLSI CAD, design for testability, and asynchronous circuit testing. He is also a student member of IEEE.

Satoshi Ohtake received the B.E. degree in computer science from the University of Electro-Communication, Tokyo, Japan, in 1995 and the M.E. and Ph.D. degrees in information science from Nara Institute of Science and Technology, Nara, Japan, in 1997 and 1999, respectively. He was a Research Fellow of the Japan Society for the Promotion of Science from 1998 to 1999. Presently he is an Assistant Professor at the Graduate School of Information Science, Nara Institute of Science and Technology, Nara, Japan. His research interests are VLSI CAD, design for testability, and test pattern generation. He is a member of IEEE and IPSJ.

Hideo Fujiwara received the B.E., M.E., and Ph.D. degrees in electronic engineering from Osaka University, Osaka, Japan, in 1969, 1971, and 1974, respectively. He was with Osaka University from 1974 to 1985 and Meiji University from 1985 to 1993, and joined Nara Institute of Science and Technology, Nara, Japan in 1993. Presently he is a Professor with the Graduate School of Information Science, Nara Institute of Science and Technology. His research interests are logic design, digital systems design and test, VLSI CAD and fault tolerant computing, including high-level/logic synthesis for testability, test synthesis, design for testability, built-in self-test, test pattern generation, parallel processing, and computational complexity. He is the author of Logic Testing and Design for Testability (MIT Press, 1985). He has received many awards including the Okawa Prize for Publication, IEEE CS Continuing Service Award, and IEEE CS Outstanding Contribution Award. He has served as an editor and associate editors of several journals, including the IEEE Trans. on Computers, and Journal of Electronic Testing: Theory and Application, and as guest editor of several special issues of IEICE Transactions of Information and Systems. Dr. Fujiwara is a fellow of the IEEE, a Golden Core member of the IEEE Computer Society, and a fellow of the IPSJ.