Enabling False Path Identification from RTL for Reducing Design and Test Futileness

Hiroshi Iwata, Satoshi Ohtake and Hideo Fujiwara
Graduate School of Information Science, Nara Institute of Science and Technology
8916-5 Takayama, Ikoma, Nara 630-0192, Japan
Email: {hiroshi-i, ohtake, fujiiwara}@is.naist.jp

Abstract—Information on false paths is useful for design and test. Since identification of 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 logic level (RTL) and at gate level can be established. Until now, the correspondence has been established only by some restricted logic synthesis. In this paper, we propose a method for mapping RTL false paths to their corresponding gate level paths without such a specific logic synthesis.

Keywords—false path, high level testing, path mapping, functional equivalence

I. INTRODUCTION

For design and test of circuits, false path information is very valuable since it can be used for reducing the time required for logic synthesis, test generation and test application, and circuit area while also minimizing overtesting. From the perspective of design, 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 gate level for combinational circuits[1], [2], [3] and for sequential circuits[4], [5] have been proposed. However, since it is difficult to apply false path identification methods at 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]. Yoshikawa et al.[6] defined RTL false paths and proposed a method to identify them. However, these methods would be 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)[6]. 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. Figure 1 shows the false path identification flow. We first 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 utilize the method that finds candidates of the functionally equivalent nets from a gate level circuit by using a diagnostics technique[7].

Since the gate level paths mapped by our method are represented as signal lines, each gate level path does not need to be fully specified as a path, so we are able to handle bundled 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 definition of RTL false path in [6] 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. Experiments 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. Prelimi-

naries are presented in Section II. Section III presents the proposed RTL path mapping method. Section IV shows that the gate level paths mapped from a set of RTL false paths.

II. PRELIMINARIES

A. 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 module 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.

B. Gate level and RTL path representation

A gate level path is an ordered set of gate level nets \(\{e_1^G, \ldots, e_n^G\}\) where \(e_i^G\) is the net directly connected to a primary input or the output of an FF, \(e_i^G\) is the net directly connected to a primary output or the input of an FF, and \(e_i^G(i = 2, \ldots, n - 1)\) is the net connecting the gates having \(e_{i-1}^G\) as an input and \(e_i^G\) as an output. A subset of a gate level path \(p^G\) is called a sub gate level path of \(p^G\).

Similarly, an RTL path is an ordered set of RTL signal lines \(\{e_1^R, \ldots, e_n^R\}\) where \(e_i^R\) is the RTL signal line directly connected to a primary input or the output of a register, \(e_i^R\) is the RTL signal line directly connected to a primary output or the input of a register, and \(e_i^R(i = 2, \ldots, n - 1)\) is the RTL signal line connecting the modules having \(e_{i-1}^R\) as an input and \(e_i^R\) as an output. A subset of an RTL path \(p^R\) is called a sub RTL path of \(p^R\).

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 \(s[i]\). A bit-sliced RTL path is an ordered set of bit-sliced RTL signal lines \(\{e_1^R[k_1], \ldots, e_n^R[k_n]\}\) where \(e_i^R[k_i]\) is the \(k_i\)-th bit-sliced RTL signal line directly connected to a primary input or the output of a register, \(e_i^R[k_i]\) is the \(k_i\)-th bit-sliced RTL signal line directly connected to a primary output or the input of a register, and \(e_i^R[k_i](i = 2, \ldots, n - 1)\) 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 \(e_i^R[k_{i+1}]\) as an output. A subset of a bit-sliced RTL path \(p^R\) is called a sub bit-sliced RTL path of \(p^R\).

C. Relation between signal lines

Here, we first define signal line cutting, which is an operation needed for defining functionally equivalent signal lines. For a combinational circuit \(C\) with \(n\) inputs, \(m\) outputs and an internal signal line \(s\), the following operation is referred as cutting \(C\) on \(s\). Create the \((n + 1)\)-th new input port and the \((m + 1)\)-th new output port. Remove the signal line \(s\). 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 function equivalence of signal lines as follows.

Definition 1 (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.

D. Relation between paths

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

Definition 2 (Functionally equivalent path): Sub bit-sliced RTL paths and sub gate level paths are simply referred to as sub paths. Sub paths \(q_1 = \{e_1, \ldots, e_m\}\) and \(q_2 = \{e_1, \ldots, e_n\}\) are functionally equivalent if \(q_1\) and \(q_2\) satisfy the following conditions.

1) \(n = m\)
2) \(e_1, \ldots, e_n\) (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.
III. 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 Section IV.

A. 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 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 there exists a one-to-one correspondence between the input/output signal lines of \( S^R \) and \( C^G \). This relation is called I/O mapping information. 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. The preservation is common for logic synthesis for structural RTL designs.

**Definition 5 (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 \( P^G = \bigcup_{i=0}^{m_1} P^G_{ij} \), 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 \( n \) is the number of the sub RTL paths of \( P^R \). Let \( q^G_{ij} \) be a sub gate level path that is functionally equivalent to \( q^R_{ij} \). \( P^G \) is a set of gate level paths including \( q^G_{ij} \).

B. 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 \( c^G_{ij} \), that is functionally equivalent to each bit-sliced RTL signal line \( c^R_{ij,k} (k = 1, \ldots, l) \), where \( c^R_{ij,k} \) 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 \( c^G_{ij} \) 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) Find all the gate level paths that include \( \{e^G_{ij,k} \} \), such that every \( e^G_{ij,k} (k = 1, \ldots, l) \) is found in the previous step. The set of all the obtained gate level paths is referred to as \( P^G_{ij} \).

4) Calculate \( P^G = \bigcup_{i=0}^{m_1} \bigcup_{j=0}^{l} P^G_{ij} \).

Note that, 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 Section V, 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 step 3 and 4, not all the gate level paths need to be listed; it is not practical. Instead, paths are represented just by specifying nets \( \{e^G_{ij,k} \} \) that are passed through. This representation is compatible with EDA tools like SDC description.

C. 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.

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 6 (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 \( E^G = \{e^G[e^G] \} \) where \( e^G \) is a net in \( C^G \).
The signal line mapping problem is shown in the following. Their output responses. This is achieved by applying equivalence checking 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. [7] 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 to the fault under test, 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 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 above 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 \( R_{f/v} \) 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 \( R_{f/v} \) when \( T \) is applied to these circuits. A set of the nets having equivalent faults is referred to as \( E^{G,v} \).
3) Obtain \( E^G = E^{G,0} \cap E^{G,1} \).
4) For each \( e^G \in E^G \), create \( C^R(e^R[k]) \) and \( C^G(e^G) \) by cutting \( C^R \) and \( e^G \) 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 [7]. In [7], 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. Here we assume that the fault diagnosis technique used in the algorithm can report all the suspected faults, i.e., it never misses an equivalent fault under its input patterns.

**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.

Due to the limitation of the space, we briefly show the proof. By applying Steps 1 to 3, gate level nets satisfying the necessary condition of functionally equivalent signal line are selected. From the assumption of the diagnosis, the functionally equivalent signal lines are still available in the nets. In Steps 4 and 5, only gate level nets which are functionally equivalent to \( e^R[k] \) are selected. Detailed description is available in [8].

**IV. RTL FALSE PATH MAPPING**

In [9], Yoshikawa et al. defined non-robust untestable paths for RTL circuits as follows.

**Definition 7 (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 sufficient condition of RTL-NRU based on control signals of MUXes and registers. In this paper, we refer to an RTL path that satisfies the sufficient condition of RTL-NRU as an RTL false path. A gate level paths in which both path delay faults with rising and falling transitions at the starting point are non-robust untestable is also referred to as a gate level false path. 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 false 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 \( c_1 \) in cycles between \( t \) and \( t+1 \) (if any); (2) there is no propagatability of a value from \( c_i \) to \( c_{i+1} \) for some \( i \) (i.e. \( 1 \ldots n \)) in \( t+1 \) (if any); (3) no response on \( c_n \) is captured on the ending register in \( t+2 \) (if any); and (4) there is no observability for the captured responses (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 [9].

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 [6] for more general circuits synthesized without the restriction. Therefore, we obtain the following theorem.

**Theorem 2:** For an RTL false path \( p^{RT} \), in an RTL circuit \( S^{RT} \), any \( p^{G} \in P^{G} \) that is mapped from \( p^{RT} \) with our path mapping method in a synthesized gate level circuit is gate level false.

Due to the limitation of the space, we briefly show the proof. If any conditions of RTL-NRU (1)-(4) are satisfied, the corresponding gate level paths mapped with our proposed method are always NRU since the conditions for the module boundary on the path are preserved in the gate level circuit even if the MIP-LS assumption is removed. Detailed description is available in [8]. By this theorem, we are able to treat gate level false 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.

### V. 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 [6]. 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 I 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 “area (\#gates)” and “Arbitrary” under “Area (\#gates)” show the circuit area synthesized by MIP-LS[6] 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(3GHz), 16GB).

We use the RTL path mapping ratio \( P_{mr} = \frac{|P^{RT}|}{|P^{RT}|} \times 100[\%] \) as an evaluation criterion, where \( |P^{RT}| \) 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 \( P_{mrb} = \frac{|P^{RT}|}{|P_{b}|} \times 100\% \), where \( |P_{b}| \) is the total number of bit-sliced RTL paths in the datapath and \( |P^{RT}| \) is the number of bit-sliced RTL paths mapped. Table II shows the path mapping ratios, bit-sliced path mapping ratios and time required for the mapping.

Table III shows the signal line and path mapping results in detail. Rows “\#Total”, “\#Unique”, “\#Mapped”, “\#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. Thanks to Definition 3 (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. 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 IV shows the result of RTL false path mapping and the time required for this mapping. Rows “\#Total”, “\#False”, “Ratio”, “Total”, “Unique”, “Ravi”, “F CHECK”.

### Table I: Circuit characteristics.

<table>
<thead>
<tr>
<th>Circuit</th>
<th>#bit</th>
<th>#PI</th>
<th>#PO</th>
<th>#reg</th>
<th>Area (#gates)</th>
</tr>
</thead>
<tbody>
<tr>
<td>LWF</td>
<td>16</td>
<td>2</td>
<td>5</td>
<td>1,371</td>
<td>1,467</td>
</tr>
<tr>
<td>Tseng</td>
<td>8</td>
<td>3</td>
<td>2</td>
<td>1,357</td>
<td>1,077</td>
</tr>
<tr>
<td>Paulin</td>
<td>8</td>
<td>2</td>
<td>7</td>
<td>1,300</td>
<td>1,303</td>
</tr>
<tr>
<td>MPEG</td>
<td>8</td>
<td>16</td>
<td>241</td>
<td>38,183</td>
<td>28,428</td>
</tr>
</tbody>
</table>

### Table II: Path mapping results.

<table>
<thead>
<tr>
<th>Circuit</th>
<th>LWF</th>
<th>Tseng</th>
<th>Paulin</th>
<th>MPEG</th>
<th>PrimeTime (CPU/sec)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Pmr</td>
<td>73.7</td>
<td>90.0</td>
<td>100.0</td>
<td>100.0</td>
<td>28.14</td>
</tr>
<tr>
<td>Pmrb</td>
<td>74.2</td>
<td>96.8</td>
<td>100.0</td>
<td>100.0</td>
<td>21.74</td>
</tr>
<tr>
<td>CPU</td>
<td>0.10</td>
<td></td>
<td></td>
<td></td>
<td>0.30</td>
</tr>
</tbody>
</table>
“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 [6], TetraMax took about 50 hours to identify 10,000 false paths of Paulin. Since the RTL false path identification method proposed in [6] and our path mapping method took less than 1 second for several circuits, our high level identification approach is very effective.

Table V shows false path mapping results in detail. Rows “#Pfalse,” “#Punique,” “#Smapped” and “#Pmapped” 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.

VI. CONCLUSIONS

Establishing correspondence between an RTL circuit and its synthesized gate level circuit is important for high level testing approaches. 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 [6] 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