<table>
<thead>
<tr>
<th>Title</th>
<th>On the derivation of a minimum test set in high quality transition testing</th>
</tr>
</thead>
<tbody>
<tr>
<td>Author(s)</td>
<td>Iwagaki, Tsuyoshi; Kaneko, Mineo</td>
</tr>
<tr>
<td>Citation</td>
<td>10th Latin American Test Workshop, 2009. LATW '09.: 1-6</td>
</tr>
<tr>
<td>Issue Date</td>
<td>2009-03</td>
</tr>
<tr>
<td>Type</td>
<td>Conference Paper</td>
</tr>
<tr>
<td>Text version</td>
<td>publisher</td>
</tr>
<tr>
<td>URL</td>
<td><a href="http://hdl.handle.net/10119/8480">http://hdl.handle.net/10119/8480</a></td>
</tr>
</tbody>
</table>

**Description**

Copyright (C) 2009 IEEE. Reprinted from 10th Latin American Test Workshop, 2009. LATW '09., 1-6. This material is posted here with permission of the IEEE. Such permission of the IEEE does not in any way imply IEEE endorsement of any of JAIST's products or services. Internal or personal use of this material is permitted. However, permission to reprint/republish this material for advertising or promotional purposes or for creating new collective works for resale or redistribution must be obtained from the IEEE by writing to pubs-permissions@ieee.org. By choosing to view this document, you agree to all provisions of the copyright laws protecting it.
On the Derivation of a Minimum Test Set in High Quality Transition Testing

Tsuyoshi Iwagaki and Mineo Kaneko
Japan Advanced Institute of Science and Technology (JAIST), Ishikawa 923-1292, Japan
E-mail: \{iwagaki, mkaneko\}@jaist.ac.jp

Abstract
This paper discusses a test generation method to derive high quality transition tests for combinational circuits. It is known that, for a transition fault, a test set which propagates the errors (late transitions) to all the primary outputs reachable from the fault site can enhance the detectability of unmodeled defects. In this paper, to generate a minimum test set that meets the above property, the test generation problem is formulated as a problem of integer linear programming. The proposed formulation guarantees that minimum two-pattern tests for a transition fault are generated so that the errors will be observed at all the primary outputs reachable from the fault site. A case study using a benchmark circuit is presented to show the feasibility of the proposed method.

1 Introduction

The purpose of manufacturing test is to separate defective circuits from good ones. The behavior of a defect can be expressed by a fault model. To cope with various types of defects, several fault models such as the stuck-at fault model and the transition fault model are usually considered during test generation phases. When a target fault model is specified, test engineers try to generate tests with 100% fault coverage under the fault model. Obtained tests are then applied to actual circuits for defect screening. However, some defective circuits can pass the screening due to the presence of unmodeled defects even though the fault coverage of the applied tests is 100%. One way to avoid this undesirable situation is to develop a dedicated fault model for such defects. However, since it is costly to do so in general, several alternatives which assume conventional fault models have been discussed to enhance the detectability of unmodeled defects [1, 2, 3, 4].

Multiple-detection tests [1] have been shown to have an ability of detecting unmodeled defects. In order to clarify how effective multiple-detection tests are, some metrics were discussed in [2, 3, 4]. This paper focuses on the metric in [2]. In [2], the authors considered a test set for transition faults that propagates the errors (late transitions) of each transition fault to all the primary outputs reachable from the fault site, and showed it is effective in screening defective circuits compared to a conventional test set. To derive such a test set, some test generation procedures have been proposed in [5, 6]. The procedures in [5, 6] used a Boolean satisfiability technique with some heuristics and an existing test generation tool, respectively. Given a combinational circuit and a transition fault in the circuit, the following simple question can arise:

- What is the minimum number of two-pattern tests that detect the fault at all the primary outputs reachable from the fault site?

To the best of our knowledge, there has been no answer to this question yet. One goal of this paper is to give an answer to it. In this paper, we try to tackle this problem by using a technique of integer linear programming (ILP).

The rest of this paper is organized as follows. Section 2 gives the concept of test generation using ILP, then, in Section 3, an ILP formulation is presented to derive a minimum test set for a transition fault that meets the above property. Section 4 presents a case study to show the feasibility of our proposed method, and finally, Section 5 concludes the paper and describes our future work.

2 Preliminaries

Our test generation method is based on integer linear programming (ILP). In this section, we describe how to translate the test generation problem for a transition fault in a combinational circuit into an ILP problem.

2.1 Concept of ILP-based test generation

ILP-based test generation has first been presented for the stuck-at fault model [8]. Figure 1 represents the concept of ILP-based test generation. In this framework, given a combinational circuit and a fault, the circuit and the detection condition of the fault are first translated into the corresponding constraints that consist of inequalities and equalities with integer variables (especially 0-1 variables). Then, a feasible assignment to the variables that meets the constraints is obtained by an ILP solver. The assigned values of the variables that correspond to the circuit inputs form a test for the fault. If one wants to optimize some property during
test for a transition fault, we first explain how to express the circuit behavior by using ILP constraints.

Before describing how to generate a two-pattern test for a transition fault, we first consider the circuit shown in Figure 2. For example, we can obtain the following constraints given a combinational circuit:

- Constraints for fault detection
- Constraints for the faulty circuit
- Constraints for the fault-free circuit

ILP constraints: (+ objective function)

Table 1 shows inequalities in ILP constraints to express the behaviors of primitive gates with one or two inputs. In the first column of the table, $y$ represents a gate output and each of $x_1$ and $x_2$ represents a gate input, where they can take ‘0’ or ‘1’. A feasible assignment to the variables of the inequalities for a gate corresponds to the behavior of the gate. For example, a 2-input AND gate produces ‘0’ if at least one input has ‘0’. This behavior corresponds to the first and second inequalities in Table 1. Indeed, if $x_1$ or $x_2$ takes ‘0’, $y$ has to be ‘0’ in those inequalities. Furthermore, if both inputs take ‘1’, the AND gate produces ‘1’. This behavior is expressed by the last inequality in the table. In this way, each gate in a combinational circuit can be interpreted as inequalities in ILP constraints.

Given a combinational circuit, we can obtain ILP constraints for the whole circuit by replacing each gate with its corresponding inequalities repeatedly. Now, let us consider the circuit shown in Figure 2. For example, we can obtain the following constraints for $c_{17}$:

$$G_1: x_1 + x_{10} \geq 1, x_3 + x_{10} \geq 1, -x_1 - x_3 - x_{10} \geq -2,$$

$$G_2: x_3 + x_{11} \geq 1, x_6 + x_{11} \geq 1, -x_3 - x_6 - x_{11} \geq -2, $$

$$G_3: x_2 + x_{16} \geq 1, x_{11} + x_{16} \geq 1, -x_2 - x_{11} - x_{16} \geq -2, $$

$$G_4: x_{11} + x_{19} \geq 1, x_7 + x_{19} \geq 1, -x_{11} - x_7 - x_{19} \geq -2, $$

$$G_5: x_{10} + x_{22} \geq 1, x_{16} + x_{22} \geq 1, -x_{10} - x_{16} - x_{22} \geq -2, $$

$$G_6: x_{16} + x_{23} \geq 1, x_{19} + x_{23} \geq 1, -x_{16} - x_{19} - x_{23} \geq -2. $$

Any feasible assignment for these constraints simulates the behavior of $c_{17}$. In Figure 2, when we have $x_1 = 1$, $x_2 = 1$, $x_3 = 0$, $x_6 = 1$ and $x_7 = 1$, the circuit behaves as follows: $x_{10} = 1$, $x_{11} = 1$, $x_{16} = 0$, $x_{19} = 0$, $x_{22} = 1$ and $x_{23} = 1$. These values satisfy the above constraints, and vice versa.

Given a combinational circuit $C$ and a transition fault $f$ in $C$, the following procedure is performed to generate a two-pattern test in this paper.

1. Extract the fanin cone $C^f$ reachable to $f$ and the fanout cone $C^\ell$ reachable from $f$, from $C$.
2. Copy $C$ as $C^2$.
3. Translate $C^1$, $C^2$ and $C^\ell$ into the corresponding ILP constraints, and create additional constraints to express the connection between $C^1$ and $C^\ell$.
4. Create the constraints for detecting $f$.
5. Apply an ILP solver to the above constraints.

Here, we consider Figure 2 and the slow-to-rise transition fault on $x_{11}$. To generate a two-pattern test for the fault, we first perform steps 1 and 2 of the above procedure. Figure 3 shows the obtained three circuits. Figure 3(a) represents the fault-free version of the original circuit associated with $x_{11}$. This fault-free circuit is used to generate the first vector of a two-pattern test, and the behavior of it is expressed by the following constraint:

$$G_{11}^f: x_1^f + x_{11}^f \geq 1, x_6^f + x_{11}^f \geq 1, -x_3^f - x_6^f - x_{11}^f \geq -2.$$
The behavior of it is expressed by the following constraints.

\[ x_{22} - x_{22} + e_{22} = -1 \]

To differentiate the fault-free circuit from the faulty one, we need to translate this condition into ILP constraints, we introduce the following constraints:

\[ \begin{align*}
G_1: x_1 + x_10 &\geq 1, x_11 + x_6 \geq 1, -x_1 - x_11 + x_6 \geq -2, \\
G_2: x_10 + x_22 &\geq 1, x_16 + x_23 \geq 1, -x_10 - x_22 + x_6 \geq 2, \\
G_3: x_16 + x_23 &\geq 1, x_17 + x_20 \geq 1, -x_16 - x_17 + x_20 \geq -2. 
\end{align*} \]

In this way, a two-pattern test can be generated by applying any ILP solver to all the above constraints.

In Figure 3(c), we can assume that \( x_{11} \) has a stuck-at 0 fault, and that \( x_{10} \), \( x_f \) and \( x_{10} \) have the same values of the corresponding signals of Figure 3(b), we must have the following constraints:

\[ \begin{align*}
x_{11} &= 0, \\
x_{10} - x_f &= 0. 
\end{align*} \]

Now, we consider the detection conditions for the slow-to-rise transition fault on \( x_{11} \). According to the first detection condition mentioned before, \( x_{11} \) must be set to 0 under the first vector of a two-pattern test. Hence, the following constraint is required:

\[ x_{11} = 0. \]

Moreover, according to the seconds detection condition, in order to detect the corresponding stuck-at fault, we need to differentiate the fault-free circuit from the faulty one. To translate this condition into ILP constraints, we introduce variables \( e_{22}, e_{23} \) with the following constraints:

\[ \begin{align*}
x_{22} - x_{22} + e_{22} &= 0, \\
x_{22} + x_{23} + e_{23} &= 0. 
\end{align*} \]

Each of \( e_{22} \) and \( e_{23} \) takes '1' if and only if the corresponding primary outputs of the fault-free circuit and faulty circuit take different values.

Finally, since the error must be propagated to at least one primary output, we have the following constraint:

\[ e_{22} + e_{23} \geq 1. \]

In this way, a two-pattern test can be generated by applying any ILP solver to all the above constraints.

---

**Table 1: Inequalities in ILP constraints expressing the behaviors of primitive gates**

<table>
<thead>
<tr>
<th>Gate types</th>
<th>Inequalities</th>
</tr>
</thead>
<tbody>
<tr>
<td>( y = \text{AND}(x_1, x_2) )</td>
<td>( x_1 - y \geq 0, x_2 - y \geq 0, -x_1 - x_2 + y \geq -1 )</td>
</tr>
<tr>
<td>( y = \text{NAND}(x_1, x_2) )</td>
<td>( x_1 + y \geq 1, x_2 + y \geq 1, -x_1 - x_2 - y \geq -2 )</td>
</tr>
<tr>
<td>( y = \text{OR}(x_1, x_2) )</td>
<td>( -x_1 + y \geq 0, -x_2 + y \geq 0, x_1 + x_2 - y \geq 0 )</td>
</tr>
<tr>
<td>( y = \text{NOR}(x_1, x_2) )</td>
<td>( x_1 - x_2 - y \geq 0, -x_1 + x_2 + y \geq 0, x_1 - x_2 - y \geq 2 )</td>
</tr>
<tr>
<td>( y = \text{XOR}(x_1, x_2) )</td>
<td>( x_1 - x_2 - y \geq 1, -x_1 + x_2 - y \geq 1, y \geq -x_1 - x_2 - y \geq -2 )</td>
</tr>
<tr>
<td>( y = \text{NOT}(x) )</td>
<td>( x + y \geq 1, -x - y \geq 1 )</td>
</tr>
<tr>
<td>( y = \text{BUFFER}(x) )</td>
<td>( x - y \geq 0, -x + y \geq 0 )</td>
</tr>
</tbody>
</table>

---

Figure 3(b) represents the fault-free version of the original circuit. This fault-free circuit is used together with the circuit of Figure 3(c) in order to generate the second vector of a two-pattern test, and the behavior of it is expressed by the following constraints.

\[ \begin{align*}
G_1: x_1 + x_10 &\geq 1, x_11 + x_6 \geq 1, -x_1 - x_11 + x_6 \geq -2, \\
G_2: x_10 + x_22 &\geq 1, x_16 + x_23 \geq 1, -x_10 - x_22 + x_6 \geq 2, \\
G_3: x_16 + x_23 &\geq 1, x_17 + x_20 \geq 1, -x_16 - x_17 + x_20 \geq -2. 
\end{align*} \]

Figure 3(c) represents the faulty version of the original circuit associated with \( x_{11} \). This faulty circuit is used to generate the second vector of a two-pattern test, and the behavior of it is expressed by the following constraints.

\[ \begin{align*}
G_1: x_1 + x_10 &\geq 1, x_11 + x_6 \geq 1, -x_1 - x_11 + x_6 \geq -2, \\
G_2: x_11 + x_19 \geq 1, x_1 + x_19 \geq 1, -x_11 - x_1 + x_19 \geq -2, \\
G_3: x_10 + x_22 \geq 1, x_16 + x_23 \geq 1, -x_10 - x_22 + x_6 \geq 2, \\
G_4: x_16 + x_23 \geq 1, x_17 + x_19 \geq 1, -x_16 - x_17 + x_19 \geq -2. 
\end{align*} \]

Figure 3 illustrates three circuits for fault detection: (a) Fault-free circuit for generating the first vector of a two-pattern test; (b) Faulty circuit for generating the second vector of a two-pattern test; and (c) Faulty circuit for generating the second vector of a two-pattern test.
at which the error of \( I \) never reaches for any vector pair.

For such a primary output, we prepare a 0-1 variables \( r_j \) for each \( j \). Equation \( r_j = 1 \) indicates the error of \( f \) does not reach at the \( j \)-th primary output of any copy of the circuit, and \( r_j = 0 \) indicates the error of \( f \) reaches at the \( j \)-th primary output of at least one copy. By using this variable, we have the following constraints for each \( j \).

\[
\sum_{i=1}^{\mid O \mid} e_{i,j} + r_j \geq 1 \quad (1)
\]

This means that the error of \( f \) must be propagated to the \( j \)-th primary output of at least one copy, or the \( j \)-th primary output of every copy must be redundant. Since \( e_{i,j} = r_j = 1 \) never happen for all \( i,j \), we also have the following constraints.

\[
e_{i,j} + r_j \leq 1 \quad (2)
\]

Now, we introduce a variable \( u_i \) for each \( i \) to identify copies of the circuit that are mandatory. Variable \( u_i \) takes ‘1’ if the error of \( I \) is propagated to the \( j \)-th primary output in \( C_i \), otherwise it takes ‘0.’ This state can be expressed by the following constraint.

\[
-e_{i,j} + u_i \geq 0 \quad (3)
\]

Since at least one \( u_i \) has to take ‘1’ if \( f \) is testable, i.e., a test is generated in at least one copy, we also have the following constraint.

\[
\sum_{i=1}^{\mid O \mid} u_i \geq 1 \quad (4)
\]

Finally, we have to minimize the following equation for test minimization.

\[
\sum_{i=1}^{\mid O \mid} u_i + |O_f| \cdot \sum_{j=1}^{\mid O \mid} r_j \quad (5)
\]

The first term counts the number of copies that are used for propagating the errors to all the reachable primary outputs. From inequality (1), it can be seen that \( r_j \) can be set to ‘1’ freely. To prevent \( r_j \) from being ‘1’ freely, the term is multiplied by \( |O_f| \) in the second term of the above equation. Therefore, after running an ILP solver, \( r_j \) will take ‘1’ if and only if the error of \( f \) never reaches at the \( j \)-th primary output of any copy, i.e., the \( j \)-th primary output of the circuit is redundant.

By using the values assigned to the primary inputs of copies whose \( u_i \) take ‘1,’ we can form a minimum test set for \( f \).
Table 2: Values of \(e_{i,j}\)

\[
\begin{array}{cccccc}
 j = 1 & j = 2 & j = 3 & j = 4 & j = 5 \\
 i = 1 & 0 & 1 & 0 & 0 & 0 \\
i = 2 & 0 & 1 & 0 & 0 & 1 \\
i = 3 & 0 & 0 & 0 & 0 & 0 \\
i = 4 & 1 & 0 & 0 & 0 & 1 \\
i = 5 & 0 & 1 & 1 & 0 & 0 \\
\end{array}
\]

Table 3: Values of \(u_i\)

\[
\begin{array}{c}
i = 1 \\
i = 2 \\
i = 3 \\
i = 4 \\
i = 5 \\
\end{array}
\begin{array}{c}
1 \\
1 \\
0 \\
1 \\
1 \\
\end{array}
\]

3.3 Example

To clarify our ILP formulation, we give an example here. We use a combinational circuit \(C\) with five primary outputs as an example circuit. To generate a minimum test set for a fault \(f\) in \(C\), five copies \(C_1, C_2, \ldots, C_5\) of \(C\) need to be prepared. Now, let us consider a situation where ILP constraints for the test generation were provided for an ILP solver, and, during solving the ILP problem, the temporary feasible assignment shown in Tables 2–4 was obtained.

Table 2 represents the errors of \(f\) reach at the 2nd primary output of \(C_1\), at the 2nd and 5th primary outputs of \(C_2\), at the 1st and 5th primary outputs of \(C_3\), at the 2nd and 3rd primary outputs of \(C_4\), respectively. Note that, in \(C_3\), no test is generated. Since, in any of \(C_1, C_2, C_4\) and \(C_5\), the error appears at least one primary output, each \(u_i\) except \(u_2\) takes '1' as shown in Table 3. Notice that it is possible for \(u_2\) to take '1' because it also satisfies inequality (3). However, in the final solution after solving the ILP problem, such an assignment will be rejected.

Table 4 shows that the 4th primary output of any copies has no error.

3.4 Sizes of variables and constraints

Here, we estimate the sizes of variables and constraints in our test generation problem. Let \(n\) be the number of signal lines in a combinational circuit. It is enough to prepare \(2n\) variables for fault detection (Figure 3). As mentioned in Section 3.2, since \(|O_f|\) copies of the original circuit are produced, totally \(2n \cdot |O_f|\) variables are required for fault detection. Since the additional variables of \(r_j\) and \(u_i\), where \(1 \leq i \leq |O_f|\) and \(1 \leq j \leq |O_f|\), are used to derive a minimum test set, totally \(2|O_f|\) variables are also needed. Thus, we need to prepare at most \(2n \cdot |O_f| + 2|O_f|\) variables. The number of constraints for fault detection and for test set minimization can roughly be estimated as \(O(n \cdot |O_f|)\) and \(O(|O_f|^2)\), respectively.

Table 4: Values of \(r_j\)

\[
\begin{array}{cccccc}
 j = 1 & j = 2 & j = 3 & j = 4 & j = 5 \\
 0 & 0 & 0 & 1 & 0 \\
\end{array}
\]

4 Case study

To show the feasibility of our proposed method, we performed a case study using one ISCAS '85 benchmark circuit (c2670). Our case study was done on a Linux workstation (CPU: AMD Opteron 250 2.4 GHz \(\times 2\), Memory: 8 GB), and CPLEX (version 11.01) from ILOG and Galena from [10] were used as ILP solvers. In the case study, several slow-to-rise faults in the circuit were chosen as target faults, and, for each fault, its ILP model was obtained by using a Perl program.

Table 5 shows the test generation results for the faults. Columns “Signal name” and “reachable” represent the signal name of each fault site and the number of primary outputs reachable from the fault site, respectively. Columns “variables” and “constraints” list the number of variables and constraints in the ILP model for each fault, respectively. Columns “tests,” “unobservable” and “CPU time” give the number of two-pattern tests generated by CPLEX or Galena, the number of redundant primary outputs reachable from the fault site and computation time including model construction time, respectively.

From the results, the following remarks can be made:

- If the error of a fault can be propagated to all the reachable primary outputs with one test, its computation time can be short, otherwise its computation time can increase.

- The presence of redundant primary outputs for a fault can make the computation time large.

The results also show that CPLEX did not work well for almost all instances. From this point of view, our ILP problems seem to be hard. However, Galena solved them successfully. This is because Galena is tuned specifically for 0-1 ILP problems where all variables take ‘0’ or ‘1.’ It is conceivable that our method is applicable for larger instances if we use a tuned 0-1 ILP solver.

In the future, we should verify the above remarks for various benchmark circuits. If the remarks are true, we can use those facts to improve our ILP model. For example, if we identify redundant primary outputs by using a preprocessing technique, we can remove the variables and constraints for them in our ILP model. Furthermore, this can also reduce the number of duplicated circuit copies used in our ILP model.

5 Conclusions and Future Work

In this paper, we presented an integer programming formulation to generate high quality transition tests for com-
Table 5: Test generation results for slow-to-rise faults in c2670

<table>
<thead>
<tr>
<th>Signal name</th>
<th>#reachable</th>
<th>#variables</th>
<th>#constraints</th>
<th>#tests</th>
<th>#unobservable</th>
<th>CPU time [s]</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>CPLEX</td>
<td>Galena</td>
<td>CPLEX</td>
</tr>
<tr>
<td>&quot;139&quot;</td>
<td>5</td>
<td>7,171</td>
<td>17,292</td>
<td>1</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>&quot;100&quot;</td>
<td>6</td>
<td>8,551</td>
<td>20,624</td>
<td>1</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>&quot;104&quot;</td>
<td>7</td>
<td>10,606</td>
<td>25,545</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>&quot;82&quot;</td>
<td>8</td>
<td>11,185</td>
<td>27,098</td>
<td>1</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>&quot;246&quot;</td>
<td>10</td>
<td>14,111</td>
<td>34,232</td>
<td>1</td>
<td>1</td>
<td>2</td>
</tr>
<tr>
<td>&quot;1068&quot;</td>
<td>11</td>
<td>16,748</td>
<td>40,413</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>&quot;78&quot;</td>
<td>12</td>
<td>18,613</td>
<td>44,822</td>
<td>2</td>
<td>2</td>
<td>1</td>
</tr>
<tr>
<td>&quot;1065&quot;</td>
<td>13</td>
<td>20,273</td>
<td>48,760</td>
<td>2</td>
<td>2</td>
<td>1</td>
</tr>
<tr>
<td>&quot;92&quot;</td>
<td>15</td>
<td>23,851</td>
<td>57,482</td>
<td>3</td>
<td>3</td>
<td>1</td>
</tr>
<tr>
<td>&quot;1075&quot;</td>
<td>18</td>
<td>32,457</td>
<td>78,576</td>
<td>2</td>
<td>2</td>
<td>1</td>
</tr>
<tr>
<td>&quot;1042&quot;</td>
<td>21</td>
<td>36,690</td>
<td>89,088</td>
<td>2</td>
<td>1</td>
<td>2</td>
</tr>
<tr>
<td>&quot;227&quot;</td>
<td>28</td>
<td>58,017</td>
<td>142,074</td>
<td>—</td>
<td>2</td>
<td>1</td>
</tr>
</tbody>
</table>

† Temporary solution within 3,600 seconds
‡ No feasible solution within 3,600 seconds

When a combinational circuit and a transition fault in the circuit are given, our method always generates a minimum test set that propagates the errors of the fault to all the primary outputs reachable from the fault site. In addition to theoretical interests, we believe that our discussion can be useful if one investigates a new heuristic technique for test minimization or evaluates existing heuristic techniques such as [5, 6].

In the future, we should evaluate the proposed method for various benchmark circuits, and should consider improving our ILP model and adopting heuristic techniques. Moreover, from a practical point of view, it should be important to discuss minimizing tests for not one but all faults in a circuit in our future work.

Acknowledgments

The authors would like to thank the reviewers of this paper for their helpful comments. This work was supported in part by the research promoting expenses for assistant professors of JAIST.

References