A Heuristic Technique for Generating the Synchronizable and Optimized Conformance Test Sequences

Chul Kim*

Abstract: This paper presents a new technique for generating an optimum synchronizable test sequence that can be applied in the distributed test architecture where both external synchronization and input/output operation costs are taken into consideration. The method defines a set of phases that constructs a tester-related digraph from a given finite state machine representation of a protocol specification such that a minimum cost tour of the digraph with intrinsically synchronizable transfer sequences can be used to generate an optimum synchronizable test sequence using synchronizable state identification sequences as the state recognition sequence for each state of the given finite state machine. This hybrid approach with a heuristic and optimization technique provides a simple and elegant solution to the synchronization problem that arises during the application of a predetermined test sequence in some protocol test architectures that utilize remote testers.

Keywords: Intrinsically synchronizable transfer sequence, optimum synchronizable test sequence, conformance testing, synchronization problem, synchronizable state identification sequence

1. Introduction

When a conformance testing is performed, a protocol implementation under test (IUT) is viewed as a black box. Most of test sequence generation methods[1] are based on the finite state machine (FSM) model[2, 3]. Synchronization between the upper tester (UT) and the lower tester (LT) can be achieved by constructing a synchronizable test sequence such that the corresponding sequence of transitions causes no synchronization problem[4, 5, 6]. This paper presents basically a new method for generating an

*Corresponding Author: Department of Computer Science, Yongin University (chulkim@yongin.ac.kr)
Received November 23, 2015 Revised December 02, 2015 Accepted December 10, 2015
optimally synchronizable test sequence. Section 2 describes an FSM model testing, and introduces the synchronization problem and the related concepts of synchronization between the testers. Section 3 proposes a new technique for generating optimum conformance test sequences that do not encounter a synchronization problem. This method consists of a set of phases that constructs a tester-induced digraph related to the LT and the UT, derives intrinsically synchronizable transfer sequences (ISTSs) and synchronous state identification sequences (SSISs) to bridge between the testers, and generates an optimally synchronizable test sequence for protocol testing. Finally, conclusions are given in Section 4.

2. FSM Model Testing and Definitions Related to Synchronization Problem

A protocol entity can be specified by a deterministic FSM M with a quintuple (S, I, O, f, g), where S = the set of states of M, including a special state s1 called the initial state; I = the set of inputs, written as ip in the following, ip ∈ I ; O = the set of outputs, written as oq in the following, including the null output (nu), oq ∈ O ; f = the next-state (transition) function, S x I → S: g = the output function, S x I → O.

An FSM M is represented as a directed graph, G = (V, E), where the set of vertices V = {v1, . . . , vn} represents the set of specified states S = {s1, . . . , sn} of M and a directed edge (Tm : vj, vk : ip / oq) ∈ E represents a transition from state sj to state sk in M.

![Graph Representation](image)

Fig. 2. A graph representation of a finite state machine M

Table 1. Transition table for M in Fig. 2

<table>
<thead>
<tr>
<th>State</th>
<th>Input</th>
<th>Output</th>
<th>Next-State</th>
</tr>
</thead>
<tbody>
<tr>
<td>v1</td>
<td>a</td>
<td>x</td>
<td>y</td>
</tr>
<tr>
<td>v2</td>
<td>b</td>
<td>y</td>
<td>nu</td>
</tr>
<tr>
<td>v3</td>
<td>c</td>
<td>z</td>
<td>nu</td>
</tr>
</tbody>
</table>

(Legend) The symbol "nu" means null output.

**Definition 1:** Synchronization Problem

A pair of consecutive transitions whose labels form a sequence of input/output operations [i1 / o1, i2 / o2] encounters a LT to UT synchronization problem if both i1 and o1 are related to LT (and not to UT), where i2 is related to UT, it encounters a UT to LT synchronization problem if both i1 and o1 are related to UT (and not to LT), where i2 is related to LT.

**Definition 2:** External Synchronization
Operations

LT to UT: LT informs UT that it is now a right time to send the next message.

UT to LT: UT informs LT that it is now a right time to send the next message.

As shown in Fig. 3, the LT to UT(or UT to LT) external synchronization operation is introduced each time the LT to UT(or UT to LT) synchronization problem is encountered.

![Diagram](image)

**Fig. 3.** The flows of external synchronization operations and a test sequence

**Definition 5:** Intrinsically Synchronizable Transfer Sequence

In the intrinsically synchronizable test sequence \([i1, o1, X1, i2, o2, X2, \ldots, ik, ok, Xi, ik+1, ok+1, \ldots, Xn-1, in, on]\) of the above Definition 4, \([ik+1, ok+1]\) relating to the sequence \([ik, ok]\) is an intrinsically synchronizable transfer sequence(ISTS) if, for \(1 \leq k \leq n-1\), either

1) ok and ik+1 are related to the LT, and ok+1 is related to the UT, or

2) ok and ik+1 are related to the UT, and ok+1 is related to the LT.

3. The Proposed Method for Generating an Optimum Synchronizable Test Sequence

We introduce a new technique for generating optimum conformance test sequences that do not encounter the synchronization problem using intrinsically synchronizable transfer sequences(ISTSs) and synchronizable state identification sequences(SSISs) to bridge and synchronize the pairs of input/output interactions between the LT(or UT) and the UT(or LT). First, we construct a tester-induced digraph which is a directed graph of duplicated vertices generated according to the characteristics of input/output interactions between the testers, and then construct intrinsically synchronizable transfer sequences(ISTSs) from the tester-induced digraph. Second, we present a method for constructing synchronizable state identification sequences(SSISs) which can be used to generate an optimally synchronizable test sequence. Finally, we generate and optimize a test sequence which is synchronizable between the test subsequences.

3.1 Tester-induced Digraph Construction

**Algorithm 1:** The Construction of Tester-induced Digraph

1) For each vertex \(v_i \in V\), create a pair of vertices \(LT_i, UT_i\).

2) For each edge \((T_i, o_i, \delta_i) \in E\), create five edges as follows: \((LT, UT, \delta)/\delta_i, o_i\), if both \(\delta_i\) and \(o_i\) are related to UT, or \((LT, \delta)/\delta_i, o_i\), if \(\delta_i\) is related to UT and \(o_i\) is related to LT, or \((LT, UT)/\delta_i, o_i\), if both \(\delta_i\) and \(o_i\) are related to LT, or \((UT, LT)/\delta_i, o_i\), if \(\delta_i\) is related to LT and \(o_i\) is related to UT.

3) For each pair of the consecutive transitions \((a, b, c)/\delta_i, o_i, \delta_j, o_j\) find intrinsically synchronizable transfer sequences(ISTSs) and substitute every found edge \((T, v_i, \delta_i) \in E\) into lead edge as follows: \((LT, UT, \delta)/\delta_i, o_i\), if \(\delta_i\) is related to LT and \(o_i\) is related to UT, or \((UT, LT, \delta)/\delta_i, o_i\), if \(\delta_i\) is related to UT and \(o_i\) is related to LT.
SSIS(vk), is constructed directly from the SIT of Table 2, as shown in Table 3. In addition, each SSIS of the table can be minimally optimized in length according to the above steps of Algorithm 2.

Table 2 The state identification table (SIT) derived from the tester-induced digraph GT(V, E) in Fig. 4

<table>
<thead>
<tr>
<th>Iteration</th>
<th>Chain of Start States</th>
<th>Chain of Input/Output Pairs</th>
<th>Final State of Transition</th>
<th>Uniqueness Mark of Input/Output Pairs</th>
<th>Status</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>UT</td>
<td>UT</td>
<td>True</td>
<td>True</td>
<td>Found</td>
</tr>
<tr>
<td></td>
<td>UT</td>
<td>UT</td>
<td>False</td>
<td>False</td>
<td>Found</td>
</tr>
<tr>
<td></td>
<td>UT</td>
<td>UT</td>
<td>True</td>
<td>True</td>
<td>Found</td>
</tr>
<tr>
<td></td>
<td>UT</td>
<td>UT</td>
<td>False</td>
<td>False</td>
<td>Found</td>
</tr>
<tr>
<td>2</td>
<td>UT, UT</td>
<td>UT, UT</td>
<td>False</td>
<td>False</td>
<td>Not Found</td>
</tr>
</tbody>
</table>

Table 3. A set of LT- or UT-synchronizable state identification sequences, LT- or UT-SSIS(vk), of a FSM M constructed from the SIT in Table 2

<table>
<thead>
<tr>
<th>Start State</th>
<th>LT- or UT-Synchronizable State Identification Sequence, LT- or UT-SSIS(vk)</th>
<th>Final State</th>
</tr>
</thead>
<tbody>
<tr>
<td>w</td>
<td>LT-SSIS(w) = [a1] = [1]</td>
<td>w</td>
</tr>
<tr>
<td>w</td>
<td>UT-SSIS(w) = [a2] = [1]</td>
<td>w</td>
</tr>
<tr>
<td>w</td>
<td>LT-SSIS(w) = [a1] = [1]</td>
<td>w</td>
</tr>
<tr>
<td>w</td>
<td>UT-SSIS(w) = [a2] = [1]</td>
<td>w</td>
</tr>
</tbody>
</table>

3.3 Optimum Synchronizable Test Sequence Generation

Finally, we present that an optimum synchronizable test sequence can be generated using a tester-induced digraph and a set of minimum-length SSIS sequences.

Definition 4: Optimum Synchronizable Test Sequence

A test sequence of an FSM M is called an optimum synchronizable test sequence if there exists no synchronization problem between the transitions of test subsequences of M and if the total tour of the test sequence which tests each transition of M is possibly computed and

In the next procedure, a set of LT- and UT-synchronizable state identification sequences for each state of M, denoted as LT- (UT-)

Algorithm 2: The Derivation of State Identification Table (SIT):

1) Initialize the SIT with a chain of start states related to the LT or UT of the tester-induced digraph GT(V, E) of the FSM M and a chain of input/output pairs outgoing from the corresponding start states.

2) Compare every outgoing edges for each start state until the uniqueness of each input/output pair can be found. If found, this pair for the state is marked as True in the SIT; otherwise it is marked as False. For each state, the resulting status may be marked as Found even if only one of the corresponding outgoing edges for the state is identified. Each iteration for finding the uniqueness can be completed whenever every outgoing edges for the corresponding start states are tested.

3) If there exist an unidentified states of input/output pairs that were marked as Not Found in the previous iteration, insert each final state of the transition into the last of the chain of start states in the SIT and add separately new outgoing edges for the state marked as Not Found into the last of the chain of input/output pairs and repeat Step 2 for the last state of the chain; otherwise this algorithm ends.
minimally costed.

For example, the graph G’ shown in Fig. 5 is derived from the graph G of Fig. 2 and the graph GT of Fig. 4, and the test subsequences of Table 4.

Table 4. A set of synchronizable test subsequence TSS(Tm) for each testing edge of FSM M in Fig. 2

<table>
<thead>
<tr>
<th>Testing Edge</th>
<th>Final State</th>
<th>Synchronizable Test</th>
<th>Test Sequence</th>
</tr>
</thead>
<tbody>
<tr>
<td>T1</td>
<td>LT</td>
<td>TSS(T1)</td>
<td>LT</td>
</tr>
<tr>
<td>T2</td>
<td>LT</td>
<td>TSS(T2)</td>
<td>LT</td>
</tr>
<tr>
<td>T3</td>
<td>LT</td>
<td>TSS(T3)</td>
<td>LT</td>
</tr>
<tr>
<td>T4</td>
<td>LT</td>
<td>TSS(T4)</td>
<td>LT</td>
</tr>
<tr>
<td>T5</td>
<td>LT</td>
<td>TSS(T5)</td>
<td>LT</td>
</tr>
<tr>
<td>T6</td>
<td>LT</td>
<td>TSS(T6)</td>
<td>LT</td>
</tr>
<tr>
<td>T7</td>
<td>LT</td>
<td>TSS(T7)</td>
<td>LT</td>
</tr>
<tr>
<td>T8</td>
<td>LT</td>
<td>TSS(T8)</td>
<td>LT</td>
</tr>
<tr>
<td>T9</td>
<td>LT</td>
<td>TSS(T9)</td>
<td>LT</td>
</tr>
</tbody>
</table>

Fig. 5. The graph G’ for the graph G of M shown in Fig. 2

The problem of determining a symmetric augmentation of a graph G’ can be reduced to a minimum-cost maximum flow[7] on a graph GF = (VF, EF) constructed from G’. The graph GF for the graph G’ of Fig. 5 is shown in Fig. 6 and depicts the minimum-cost maximum flow for the graph G’ that a total of 9 edge replications in E. Thus, the tester-induced symmetric augmentation graph G* for the graph G’ is constructed from the graph GF and is shown in Fig. 7 with the minimal replications of edge (vj, vk; ip / oq) ∈ E’.

Finally, we can generate an optimally synchronizable test sequence. For an example as shown in Fig. 7, the minimum-cost tour over the dotted edges is as follows: [TSS(T1), TSS(T3), TSS(T5), TSS(T7), T2, TSS(T8), T4, T5, TSS(T6), TSS(T2), TSS(T4)]. As given in Table 5, this tour that begins at UT1 and returns to UT1 is used to generate the test sequence [b/y, c/y, d/y, c/x, a/z, a/z, a/x, b/y, c/y, c/y, d/y, c/x, a/z, a/z, b/y, c/y, d/y, c/x, a/z, a/z] with the total cost of 20 input/output operations, where no synchronization problem occurs between every pair of transitions.

Fig. 6. The graph GF of the graph G’ shown in Fig. 5 and a minimum-cost maximum flow

Algorithm 4: The mRCTP with Synchronization Property

1. Construct a directed graph G’ = (V’, E’), the so-called a tester-induced symmetric augmentation of G’, where V’ = V ∪ UT, each edge of E in E = E ∪ Eout is included in E’ zero or more times, and each edge of synchronizable TSS(Tm) in Eout is included in E’ at least once, such that the total cost of edges in E’ is minimum and the in-degree of each tester-induced vertex UT (or UT’ ∈ V) is equal to its out-degree.

2. Find an Euler tour of the resulting tester-induced symmetric graph G’, that is, a tour of G’ which traverses each edge of synchronizable TSS(Tm) in Eout exactly once.
3.4 Complexity of Our Method and Performance Evaluation

Assume that $n$ and $|E|$ are the number of vertices and edges in $G$. The overall complexity of our method (in this paper, we do not show it in detail because of the limited space) is

$$\min(\Omega(2n-1) + \frac{|E|(5|E|-2)}{2}, O(8n|E|)).$$

Table 6 shows a comparative test sequence length of major test generation methods[1] and Our Method. Length of the test sequence, in terms of number of input/output pairs, will determine the execution time for the test.

<table>
<thead>
<tr>
<th>Test Generation Methods</th>
<th>Test Sequence Length of FSM M in [1]</th>
</tr>
</thead>
<tbody>
<tr>
<td>DS method</td>
<td>67 input/output pairs</td>
</tr>
<tr>
<td>W method</td>
<td>109 input/output pairs</td>
</tr>
<tr>
<td>U/O method</td>
<td>48 input/output pairs</td>
</tr>
<tr>
<td>Our Method</td>
<td>42 input/output pairs</td>
</tr>
</tbody>
</table>

3.5 The Experimental Result of an Application to the B-ISDN Q.2931

A finite state machine of the Q.2931 connection control procedures can be simplified and modeled into a transition diagram of each message and the corresponding state as shown in Fig. 8. As shown in Table 7, the total cost of an optimum synchronizable test sequence is 62 input/output operations, and the tour begins at v1 and returns to v1.

![Fig. 7. The symmetric augmentation $G^*$ of the graph $G$ in Fig. 5](image)

![Fig. 8. The simplified FSM of B-ISDN Q.2931](image)
Table 7. An optimum synchronizable test sequence for the simplified FSM of Q.2931

| Total Cost: 62 input/output pairs of transitions |

4. Conclusions

In this paper, we proposed a new technique for generating an optimally synchronizable test sequence such that do not encounter a synchronization problem and provide an optimum length. This method which is a hybrid of a heuristic and optimization technique consists of a set of phases that constructs a tester-induced digraph related to the testers, derives intrinsically synchronizable transfer sequences (ISTSs) and synchronizable state identification sequences (SSIs) to rendezvous between the testers, and finally generates an optimally synchronizable test sequence. In summary, this test sequence can be applied in a testing system where the cost of both external synchronization operations and input/output operations are taken into consideration.

REFERENCES


Euler Tours, and the Chinese Postman”,

Author Biography

Chul Kim [Regular member]

* Feb. 1977 : Yonsei Univ.,
Electronics Engineering, B.S.
* Aug. 2000 : Yonsei Univ.,
Computer Science, PhD.
Samsung Ltd., Researcher
* Feb. 1984 – Jan. 1993 :
IBM Korea, Engineer
Univ., Dept. of Computer
Science, Professor

<Research Interests>
Protocol Engineering, Computer
Security