test sets are independent of the function being realized. We state this result in Theorem 1.

Theorem 1: An AND-OR array with irredundant restricted column cascades can be so arranged such that 2n + 5 tests will detect any single faulty cell in the array. Furthermore, these tests are independent of the function being realized by the array.

If an array is formed using Minnick's [12] cut point cells, then a cell which passes the vertical input to it, to the next cell in the cascade can be realized in two different ways as shown in Figs. 5 (a) and (b). In Fig. 5 (b) the horizontal input to the gate is etched if the cell is to behave like the one in Fig. 5 (a). However, in an AND-OR array the realization of Fig. 5 (b) is advantageous as the implementation of modern technology (e.g., medium scale integration, large scale integration) is easier because only two types of cells need be formed. For such a realization another type of fault, which cannot be modeled by a stuck-type fault, is the horizontal input to a cell which is not etched and, therefore, the cell behaves like an AND cell. We will show that the test set $T_1 T_2 T_3$ will also detect this fault if only a single fault is present in the array. Consider the following two test vectors:

$$\begin{bmatrix}
h_1 & v_1 & z_1 & z_2 & \cdots & z_{n} \\
a_0 & 1 & 1 & 1 & \cdots & 0 & 0 & 1 & 1 & \cdots & 0 & 0 & 1 & 1 & \end{bmatrix}$$

If cell $C_{ij}$ was not faulty, i.e., the lead $z_i$ was disconnected, then a change in $v_1$ and $v_2$ will change the output of this cell and this effect will be propagated to the output as the path from this cell to the output is sensitized. However, if $x_i$ remains connected then the output of cell $C_{ij}$ will not change with the change in $v_1$ and $v_2$ and thus the effect of the fault will appear at the output. The arguments given above and earlier in this section can be used to show that the 2n + 5 tests included in $T_1 T_2 T_3$ will test the array if only a single cell was faulty irrespective of the number of faults present in one column.

VI. CONCLUSIONS

In this correspondence we have presented some algorithms to realize any switching function by certain two-dimensional cellular arrays. The assumptions which we made were that each column cascade in the array is an irredundant cascade with fixed order of inputs to the cells. Two algorithms have been presented to design two different types of arrays. The results of Algorithm 1 can be extended for RMC expansions which use or and EQUIVALENCE functions or other possible combinations of primitives. We have shown that the AND-OR cascades are easily testable for the presence of a single faulty cell. By providing extra observable outputs from the collector row and arranging the column cascades to satisfy certain conditions (as indicated in [13, ch. 4] for RMC networks) the 2n + 5 tests given in the last section will also detect all multiple faults. It is difficult to compare the restricted AND-OR arrays with other cellular arrays [9], [10], [12], [14]-[17] from the point of view of array complexity. However, it can be seen that the restricted AND-OR arrays are easier to test for single faults than the arrays considered earlier [9], [10], [12], [14]-[17].

REFERENCES


Optimal and Near-Optimal Checking Experiments for Output Faults in Sequential Machines

R. T. BOUTE

Abstract—An algorithmic procedure for designing optimal and near-optimal checking sequences for output faults is presented. For the specific cases where minimum length cannot be guaranteed, the algorithm also determines an upper bound on the excess length of the resulting sequence. Several extensions of the method are discussed, such as the application of output checking sequences for diagnosing purposes. The possibilities of this approach in the search for algorithms that yield optimal checking sequences for more general classes of faults are illustrated by applying the method in an ad hoc fashion and obtaining a complete checking experiment.

Index Terms—Checking experiments, fault detection, fault diagnosis, imator machine, machine identification, optimal checking sequences, output faults, sequential machine, T-sequence.

I. INTRODUCTION

Moore [1], and later Hennie [2], Hsieh [3], and others have studied the problem of designing checking experiments for sequential machines. Many improvements have been proposed over the years, such as the use of I/O sequences [5] or distinguishing sets [4], leading to shorter checking experiments. However, no guarantee about minimum length is ever given, and no estimate is available on how closely the resulting checking experiments achieve the optimum.

For checking experiments, the qualification "optimum" relates to a general specification of a class of machines that have to be distinguished from the given machine. For example, an experiment that distinguishes a machine from all nonequivalent machines with at...
most the same number of internal states [2] will be called a complete checking experiment. In the context of fault detection, specifications of this kind are usually interpreted in terms of a corresponding class of faults which may affect the state. transients, assuming that do not increase the number of internal states. However, no set of state-output tables for the faulty machines is given. These features constitute the essential difference between checking experiments and testing (or diagnosing) experiments, such as described by Ponge and McCluskey [13], where a list of faults, each with the corresponding state-output table, is explicitly given on a derived from the actual realization of the machine (logic diagram).

For checking experiments, the aforementioned correspondence between a specified class of machines and its interpretation as a class of faults is usually imperfect, because the class of machines may contain elements that cannot arise from the actual insertion of a fault into such the machine. Nevertheless, an optimal checking sequence can be defined as the shortest input sequence for checking the machine against a given class of "faults," possibly including some without physical meaning. Unfortunately, it is generally not possible to determine this minimum length, not even with the usual set of assumptions [2].

The problem is essentially concerned with this optimality problem for checking experiments. The state identification and transition verification principle [1]-[5], [8] does not solve this problem, because its optimization is local instead of total, and because it is not clear a priori that successive transition verifications can lead to an optimal chain that is obtained at all. Therefore we introduce a new approach which, as far as possible, considers the checking sequence as a complete entity rather than consisting of many separate parts (successive verification of each transition). By the same token, initializing sequences are undesirable in view of optimality, because they do not directly contribute to the checking procedure itself.

The suggested approach is based on what will be called the generate-verify-augment principle. Whereas many algorithms are based on generating successively longer sequences until one with the desired properties is found, the generate-verify-augment principle starts out with designing a minimal sequence that satisfies a well-defined set of properties that is as close as possible related (in a sense described later) to the desired properties. Only in case this sequence does not satisfy the desired properties also, the latter are used as a criterion for augmenting the sequence.

Based on this principle, an algorithm is presented for generating optimal and near-optimal output checking sequences, i.e., checking sequences for faults that do not cause errors in the state transitions of the machine.

The need for output checking experiments arises, for example, in the context of checking experiments for machines with separate next-state and output logic. If these two parts of the logic are not simultaneously faulty, any sequence which is an output checking sequence which also verifies the state transitions assuming a correct output function (and no increase in the number of states) is a complete checking sequence [6], [7]. In a recent paper [8], Friedman and Menon have emphasized the importance of this observation for the design of short checking experiments. Particular applications of output checking sequences can also be found among machines whose next-state circuitry is simple as compared to the output circuitry (I/O devices, graphic displays, parallel character or code transformation).

II. DEFINITIONS AND PRINCIPLES OF THE METHOD

Notation

We denote a sequential machine [9], [10] by \( M = (I, Q, \lambda, \delta) \), where \( I, Q \) denote the set of input symbols, the set of output symbols, and the set of internal states, respectively. The next-state function is denoted by \( \delta \) and the output function by \( \lambda \). Thus \( \delta(q,x) \) is the next state when the machine is started in state \( q \) and the input \( x \) is applied. Similarly, \( \lambda(q) \) is the output of a Moore machine [9] in state \( q \), and \( \lambda(q,x) \) is the output of a Mealy machine [9] in state \( q \) when the input \( x \) is applied. Sequences of input or output symbols are represented with an overbar, e.g., \( \bar{x} \). Also, \( \lambda(q,x) \) is the output sequence of the machine started in state \( q \) and presented with the input sequence \( x \).

For machines that are not strongly connected, there are states from which not all other states can be reached. If the initial state of the machine happens to be one of these, it will not be possible to verify the behavior of the unreachable states, and the problem of designing checking experiments is not clearly defined. In such cases, it is necessary to specify in advance which initial states have to be anticipated or, equivalently, which parts of the machine are to be checked. For the sake of conciseness, we henceforth assume that the machine is strongly connected. The reader will soon realize that only minor modifications in the procedure are needed to meet the particular specifications of the problem in case the machine is not strongly connected.

Definitions

For reasons discussed earlier, designing a checking experiment with initialization generally excludes optimality (this is intuitively clear but will be illustrated later on) and a more general definition is needed.

A complete checking sequence for a (strongly connected) machine \( M = (I, Q, \lambda, \delta) \) is an input sequence \( z \) such that, for any machine \( M' = (I, Q', \lambda', \delta') \) with \( |Q'| \leq |Q| \) and for any \( q' \in Q' \), \( \lambda'(q', x) \in \delta(q, x) \) implies \( M' = M \) (denotes equivalence [9]).

The usual checking sequences described in the literature satisfy this definition as special cases.

A faulty output function for a machine \( M = (I, Q, \lambda, \delta) \) is a function \( \lambda' \), defined on the same state set \( Q \) and input set \( I \), such that the machine \( (I, Q, \lambda', \delta) \) is not equivalent [9] to \( M \).

An output checking sequence for a machine \( M \) is an input sequence \( c \) such that \( \lambda(q, c') \neq \lambda'(q', c) \) for all possible choices of a faulty output function \( \lambda' \) and all possible initial states \( q \) and \( q' \). (Note that this is equivalent to the definition of a complete checking sequence where the extra requirement \( \delta = \delta' \) is added.)

General Procedure (Generate-Verify-Augment Principle)

For a given class of faults, one starts out by establishing a set of necessary conditions that must be satisfied by any checking sequence for that class. In general, the sufficiency of these conditions will have to be satisfied to some extent (obviously as little as possible) in favor of the possibility of finding an efficient algorithm that yields minimal length sequences satisfying these conditions. The first step of the procedure is then to generate such a sequence. The second step consists in verifying whether this sequence is a checking sequence. If the result of this verification is positive, we know that an optimal checking experiment (i.e., checking sequence whose length is augmented (third step) in order to obtain the desired checking sequence. Obviously, the length of this extension is an upper bound on the excess length of the obtained checking experiment over one of minimum length, even though this minimum length is not known.

Application to Output Faults

Definitions: A T-state is an internal state of a Moore machine or a total state of a Mealy machine. A T-sequence is a sequence that cycles the machine through all T-states, whatever the initial state may be.

For machines that can be synchronized into a given initial state, the design of an output checking sequence (nonoptimal) can be considered as a purely combinational problem: after synchronizing it suffices (see, e.g., Friedman and Menon [8]) to visit all T-states of the machine (note that this is also necessary). However, if no synchronization is possible, or if optimality is to be achieved, the sequential character of the machine becomes of prime importance, and we must distinguish from what follows. In particular, the condition that an output checking sequence must cycle the machine through all T-states is no longer sufficient in all cases (although it is in most), as will be shown in the next section. Further, the necessity of this condition is not obvious any more and needs proof, because changing an entry in the output table of the machine might conceivably transform the machine into an equivalent one, in which case it is not necessary to check this entry. This matter is settled by the following lemma and theorem, which are easy to prove [6].

Lemma: In a reduced and strongly connected machine, a change
TABLE I

CONSTRUCTION OF MINIMAL-LENGTH T-SEQUENCES FOR MACHINE M₁

<table>
<thead>
<tr>
<th>input sequence</th>
<th>present</th>
<th>next acceptor state</th>
</tr>
</thead>
<tbody>
<tr>
<td>A</td>
<td>B</td>
<td>AB BA CD DB</td>
</tr>
<tr>
<td>0</td>
<td>AC BC CB DA</td>
<td>ABC CDB ADB</td>
</tr>
<tr>
<td>00</td>
<td>BA AB CDB</td>
<td>ABC CDB</td>
</tr>
<tr>
<td>01</td>
<td>ABC CDA BDC</td>
<td>ACD ABD</td>
</tr>
<tr>
<td>10</td>
<td>ACD BCD BCA ADB</td>
<td>ABC ABD</td>
</tr>
<tr>
<td>11</td>
<td>CBA CDB ACD</td>
<td>ABC ABD</td>
</tr>
<tr>
<td>000</td>
<td>ABC  BC</td>
<td>ABC ABD</td>
</tr>
<tr>
<td>001</td>
<td>ABC  BDC</td>
<td>ACB CDB</td>
</tr>
<tr>
<td>010</td>
<td>BCD</td>
<td>*</td>
</tr>
<tr>
<td>011</td>
<td>ACD  ADB</td>
<td>-</td>
</tr>
<tr>
<td>100</td>
<td>CDB ACD BDA</td>
<td>-</td>
</tr>
<tr>
<td>101</td>
<td>CDA ABD</td>
<td>*</td>
</tr>
<tr>
<td>110</td>
<td>CBA BCD ACD</td>
<td>-</td>
</tr>
<tr>
<td>111</td>
<td>BC CB</td>
<td>-</td>
</tr>
<tr>
<td>0001</td>
<td>BC</td>
<td></td>
</tr>
</tbody>
</table>

III. ALGORITHM FOR MINIMAL AND NEAR-MINIMAL OUTPUT CHECKING SEQUENCES

Following the general principles outlined previously, the algorithm consists of three parts.

1. The design of a minimal length T-sequence \( z \) (this corresponds to the necessary condition implied by the preceding theorem).
2. The verification whether \( z \) is an output checking sequence. This will be achieved by constructing the corresponding set of imitator machines (for output faults). It follows from the definitions that \( z \) is a minimal length output checking sequence if and only if this set is empty.
3) In case this set is not empty, the sequence \( z \) is augmented in order to distinguish these imitator machines from the good machine. This can be done, for example, by means of the Paige and McLachkey algorithm [13].

Design of Minimal T-Sequences

The approach used here is analogous to the tabular methods described by Hennie [5] and is completely self-explanatory.

Table I(b) illustrates the procedure for machine \( M₁ \), whose state-output table is given by Table I(a). For what concerns the terminology, we interpret it as the state table of a different machine, called acceptor [5]. The second column contains the “present” acceptor state, which consists of an unordered set of elements. Each element corresponds to some initial state \( q₀ \) of \( M₁ \) and lists all \( T \) states of \( M₁ \) that are encountered when the corresponding sequence of the first column is applied to \( M₁ \). Thus, the “initial” acceptor state is \( AB C D \), each element representing a possible initial state of \( M₁ \).

The table is constructed row by row in an obvious fashion, as is illustrated by the following example, where also a few shortcuts are suggested. Consider the present acceptor state \( AB BA CD DB \) [Table I(b)]. Let 0 be applied as the next input. The elements become then \( A B A B C D B D A \), by simply suffixing the next states of \( M₁ \) for input 0. The last symbol of each element represents the corresponding present state of \( M₁ \). Otherwise, all repetitions within the same element may be eliminated and the remaining symbols, except the last one, can be reordered, e.g., alphabetically. This yields \( BA AB CDB BDA \). Finally, it is also clear that the element \( BDA \) may be deleted, because any later completion (by applying further
Table II

Verification of the T-Sequence 1 0 1 0

<table>
<thead>
<tr>
<th>initial</th>
<th>output sequence (input)</th>
<th>T-state sequence (input)</th>
</tr>
</thead>
<tbody>
<tr>
<td>A</td>
<td>0 0 1 0 1</td>
<td>A  C  D  A B</td>
</tr>
<tr>
<td>B</td>
<td>1 0 1 0 1</td>
<td>B  C  D  A B</td>
</tr>
<tr>
<td>C</td>
<td>0 1 0 0 1</td>
<td>C  B  A  C D</td>
</tr>
<tr>
<td>D</td>
<td>1 0 1 0 1</td>
<td>D  A  B  C D</td>
</tr>
</tbody>
</table>

b) Sequence Compatibility Table for the T-Sequence 1 0 1 0

<table>
<thead>
<tr>
<th>output sequence</th>
<th>compatible initial state</th>
</tr>
</thead>
<tbody>
<tr>
<td>A</td>
<td>C</td>
</tr>
<tr>
<td>B</td>
<td>-</td>
</tr>
<tr>
<td>C</td>
<td>A</td>
</tr>
<tr>
<td>D</td>
<td>-</td>
</tr>
</tbody>
</table>

inputs) of the element BA will automatically guarantee completion of BDA (but not vice versa). Note here that the last symbol A is the same in both elements (this is not required for deleting complete elements, i.e., elements that contain all states of $M_i$). New acceptor states, such as BA AB CDB BDA, are entered in the second column as soon as they appear, while the corresponding input sequence is entered in the first column. In this fashion, the shortest T-sequences are discovered first, and the table can be terminated (this will always be possible for strongly connected machines). This corresponds to a sink state for the acceptor, denoted by an asterisk.

Remark: This design can be very much simplified by first constructing a synchronizing sequence, or at least a sequence that reduces the initial ambiguity vector [5]. However, minimality of the T-sequences is then not always guaranteed (see Section IV).

Construction of Imitator Machines (for Output Faults) Corresponding to a T-Sequence

This problem is equivalent to verifying, for each correct response $g_c \in R(\bar{x})$, the existence of some faulty output function $\lambda'$ and state $x$ such that $\lambda'(q, x) = g_c$. This can be done by comparing $g$, symbol by symbol, to each of the T-state sequences corresponding to $x$. By T-state sequence is meant the sequence of T-states that the machine cycles through when a T-sequence is applied.

For example, Table II (a) represents the T-state sequences corresponding to $\bar{x} = 1010$ for various initial states of $M_i$. The correct responses to $\bar{x}$ are also listed. Consider, for example, the correct response 10101. It cannot result from a machine (with an output fault) starting in state A, because the first and the fourth symbol in this output sequence are different while the corresponding T-states are the same. Thus state A is said to be incompatible with the output sequence 10101.

The fastest way for performing these comparisons is the other way around, in the following fashion. Let us denote T-state-sequences and correct responses by the initial states from which they result. For each T-state-sequence $q$, determine the positions of repeated T-states, and examine the corresponding positions in the various correct responses, except those equal to $\lambda(q, x)$ (this is obvious since they correspond to $\bar{x}' = \lambda$). Each time a match is found, $q$ is entered into the second column of the sequence compatibility table, in the row corresponding to the matching correct responses (first column). This is exemplified by Table II (b).

Clearly, the result of this procedure is a table which lists, for each correct response $q'$, the initial states $q$ for which there exists an output function $\lambda'$ that can yield this response, excluding most cases with $\lambda' = \lambda$. During this procedure, the corresponding $\lambda'$ output tables can be completely constructed, because $\bar{x}$ visits all T-states, and no $\lambda'$ entry will remain undefined.

It is now easy to see that, after the deletion of all $\lambda'$ tables that yield machines equivalent to $M_i$, the remaining ones constitute the set of output tables for the imitator machines [see Table II (b)]. Note that an initial state is associated with each imitator machine, e.g., an imitator machine can give the response of $M_i$ corresponding to state A only if it starts in state C, etc. This correspondence is given by the sequence compatibility table.

Remark: For $M_i$, we have chosen $\bar{x} = 1010$ in order to be able to describe the third step of the algorithm. The reader can verify that for $\bar{x} = 0101$ no imitators exist, and thus 0101 is a minimal length checking sequence for output faults in $M_i$.

Distinguishing Between the Given Machine and Its Imitators

Since initial states are associated with all machines that must be separated from each other, this can be done by a straightforward application of the algorithm described by Poage and McCluskey [13], possibly simultaneously for several sets of machines (direct product table).

In our example, the initial states A and C become B and D, respectively, after the application of $\bar{x} = 1010$. Two sets of machines have to be separated simultaneously: first, $M_i$ with initial state B and $M_i'$ with initial state D; and second, $M_i'$ with initial state D and $M_i''$ with initial state B. The states of $M_i$" and $M_i''$ are marked by means of single and double primes, respectively.

Table III (b) represents the application of Poage and McCluskey's algorithm under a form which looks slightly different from the original description in [13]. First, because two sets of machines have to be separated simultaneously, and second, because the more compact acceptor type of representation is chosen. Again, this table is self-explanatory. The initial acceptor state corresponds to the initial states of the machines that have to be separated. The next-state entries for a given input represent the situation after the application of this input. Machines that have been separated from the good machine by this input are omitted. If only the given machine $M_i$ remains, it is omitted also. The situation where all sets of machines have been separated is denoted by means of an asterisk (sink state of the acceptor).

For the example, the shortest sequence that separates all imitators from the $M_i$ is 1. Therefore, 10101 is a checking sequence for output faults which is at most one symbol longer than the optimum (which was 0101).
TABLE III
DISTINGUISHING BETWEEN MACHINE M₁ AND ITS IMITATOR MACHINES

<table>
<thead>
<tr>
<th>a) Output Tables of the Imitator Machines.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Imitator ( M₁ )</td>
</tr>
<tr>
<td>Initial States:</td>
</tr>
<tr>
<td>for ( i₁ ): A</td>
</tr>
<tr>
<td>for ( i₂ ): C</td>
</tr>
<tr>
<td>( A )</td>
</tr>
<tr>
<td>( B )</td>
</tr>
<tr>
<td>( C )</td>
</tr>
<tr>
<td>( D )</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>b) Application of the Poege and McCluskey Algorithm.</th>
</tr>
</thead>
<tbody>
<tr>
<td>(Acceptor Representation)</td>
</tr>
<tr>
<td>---------------------------------------------</td>
</tr>
<tr>
<td>Input sequence</td>
</tr>
<tr>
<td>( A )</td>
</tr>
<tr>
<td>( 0 )</td>
</tr>
<tr>
<td>( 00 )</td>
</tr>
<tr>
<td>( 01 )</td>
</tr>
<tr>
<td>( 010 )</td>
</tr>
</tbody>
</table>

Remark: Practice has shown that the condition that an output checking experiment be a T-sequence is a well-chosen necessary condition, i.e., it turns out to be sufficient for an important fraction of the examples that have been worked out. In other words, the generated output checking experiments are very often optimal. The first part of the next section points out further aspects of this observation.

IV. EXTENSIONS OF THE METHOD

A. Use of Additional Data About the Machine

In many cases, additional data make it possible to discard certain types of imitator machines that cannot physically correspond to any faulty version of the given machine. For example, we have shown elsewhere [11] that in certain realizations (e.g., all fan-out-free ones) of a combinational function the faulty function \( \phi_r \) for any single fault \( F \) satisfies either \( \phi \subseteq \phi_r \) or \( \phi_r \subseteq \phi \). If such is the case for the output function of the machine \( M₁ \), we can discard the imitator machines \( M₁' \) and \( M₂' \), since no such relation is satisfied. Then the minimal T-sequence 1010 is a checking sequence for output faults. For all examples that have been worked out, it has been observed that imitator machines satisfy \( \phi \subseteq \phi_r \) and \( \phi_r \subseteq \phi \). It may be interesting to investigate whether this phenomenon will always occur. If so, a minimal T-sequence is a checking experiment for all single faults in a fan-out-free output network.

B. Extension of the Algorithm to Semiaadaptive Testing and Diagnosis

If the output response during the application of a T-sequence is observed, this information can be used advantageously during the remainder of the experiment.

Indeed, the relevant set of imitator machines can be obtained by matching the T-state sequences (corresponding to \( z \)) with the observed response only, rather than with all possible correct responses. If the observed response does not belong to \( R(z) \), the machine is known to be faulty and the experiment can be terminated if only fault detection is intended. However, this response can be used for diagnosis also. In the context of checking experiments, we informally define diagnosis as the reconstruction of the (possibly faulty) behavior of the machine, e.g., the complete output table in the case of output faults. In any event, the desired separation between the good machine and the imitator machines only (detection), or between the imitator machines themselves also (diagnosis), can be obtained by using the algorithm of Poege and McCluskey, as described in the preceding section. Obviously, this can be done only up to equivalence classes. Note also that two strongly connected machines are equivalent if they have an equivalent state [10].

In our previous example, the response 10101 to the sequence 1010 shows that no output fault is present, as can be seen directly from the T-state sequences [Table II(a)]. On the other hand, an output sequence such as 10001 is compatible only with the T-state sequences with initial state B or D. The corresponding (generalized) imitator machines are shown in Table IV. They can be separated by the extra input sequence 00. Thus the complete2 diagnosing input sequence for output faults is 101000 and the response is either 1000101 [indicating Table IV(a)] or 1000100 [indicating Table IV(b)].

C. Simplified Algorithm for T-Sequences

As mentioned earlier, the design of a T-sequence can be simplified as follows. First, design for the machine \( M \) a shortest sequence that reduces the initial “ambiguity vector” to as few states as possible (e.g., a synchronizing sequence [5] if one exists). The sequence thus obtained can be used as the initial part of a T-sequence. The elements of the initial state of the T-sequence acceptor are then easily obtained by concatenating, for each initial state of \( M \), all T-states that are encountered when this sequence is applied to \( M \). The remainder of the procedure is as explained in the preceding section.

Although this method will not guarantee a minimal T-sequence, the resulting simplification in the design often justifies this approach. In most cases, the obtained T-sequence will not be much longer than a minimal T-sequence.

1 In the context of output faults, a (generalized) imitator machine for a given response \( z \) to the T-sequence \( x \) is a machine with the same next-state function as the good machine and for which there exists an output function \( x' \) and an initial state \( q \) such that \( x(q, z) \neq x' \). The special cases where \( x \in R(\tilde{z}) \) are included in the original definition.

2 Provided, of course, that the response observed during the application of \( z \) was 10001. We are indeed dealing with a semiaadaptive procedure.
TABLE IV
Output Tables of Imitator Machines with Response 1 0 0 1

<table>
<thead>
<tr>
<th>state</th>
<th>output</th>
<th>state</th>
<th>output</th>
</tr>
</thead>
<tbody>
<tr>
<td>A</td>
<td>0</td>
<td>A</td>
<td>0</td>
</tr>
<tr>
<td>B</td>
<td>1</td>
<td>B</td>
<td>0</td>
</tr>
<tr>
<td>C</td>
<td>0</td>
<td>C</td>
<td>0</td>
</tr>
<tr>
<td>D</td>
<td>0</td>
<td>D</td>
<td>1</td>
</tr>
</tbody>
</table>

(a)  (b)

TABLE V
Example of the Design of a Complete Checking Experiment

a) State-Output Table of Machine $M_2$

<table>
<thead>
<tr>
<th>present state</th>
<th>next state / output (input 0)</th>
<th>(input 1)</th>
</tr>
</thead>
<tbody>
<tr>
<td>A</td>
<td>A' / 0</td>
<td>B / 1</td>
</tr>
<tr>
<td>B</td>
<td>B / 1</td>
<td>A' / 0</td>
</tr>
</tbody>
</table>

c) Imitator Machine for $M_2$

<table>
<thead>
<tr>
<th>present state</th>
<th>next state / output (input 0)</th>
<th>(input 1)</th>
</tr>
</thead>
<tbody>
<tr>
<td>A'</td>
<td>B'/ 0</td>
<td>q / 0</td>
</tr>
<tr>
<td>B'/1</td>
<td>B'/1</td>
<td></td>
</tr>
</tbody>
</table>

For example, $M_1$ has a shortest synchronizing sequence 001. The corresponding initial state for the $T$-sequence acceptor consists of the elements $ABC$ and $BDC$. The shortest sequence that completes both elements simultaneously is 01. Thus, we obtain the $T$-sequence 00101, which is only one symbol longer than the minimal $T$-sequence.

D. Remarks Regarding More General Classes of Faults

At the present time, no algorithmic method is available for generating optimal or near-optimal complete checking experiments. The general generate-verify-augment principle described in Section II may be helpful for further development in this direction, as will be shown next. The main difficulty consists in finding the adequate set of necessary conditions, which turns out to be sufficient in as many particular instances as possible.

For the time being, and for lack of a better set of conditions, let us use the fact that a complete checking experiment must also be a $T$-sequence and apply the general principles in an ad hoc fashion. For the (Mealy) machine $M_2$ represented in Table V(a), 0101 is a minimal $T$-sequence. From the correct responses (0110 and 1001) and the $T$-states $A,0,A,1,B,0,B,1$ and $B,0,B,1,A,0,A,1$, it is easily verified that there are no imitator machines. Thus 0101 is an optimal output checking experiment. To verify whether it is a complete checking experiment, assume that the machine under test responds with 0110 [Table V(b)]. The states labeled 2 and c in Table V(b) must be different, because they respond differently to input 0. We give them permanent labels, namely, $a = A'$, $c = B'$. According to our definition of a complete checking experiment, we assume that there is no fault which increases the number of states. Therefore, we can conclude that either $b = A'$ and $d = B'$ or $b = B'$ and $d = A'$. Indeed, b and d are different since their responses to input 1 are different. The first possibility corresponds to the good machine, the second one to the imitator machine of Table V(c).

The state labeled $q$ is still unknown. The very existence of this imitator machine (though not completely specified) shows that 0101 is not a complete checking experiment. This was to be expected since a sequence of length 4 in a Mealy machine can verify at most 3 transitions, and a complete checking experiment must implicitly verify all transitions. If we apply 1 as next input symbol, we see that $q = B'$ (the response should be 1, otherwise, we know right away that a fault is present). This completes Table V(c). One extra input symbol suffices to distinguish Table V(a) from Table V(c) with initial states $B$ and $B'$, respectively. It is easy to verify that the input sequence 010111 also eliminates all imitator machines for the other response (100101), and thus we obtain a complete checking sequence which is at most one symbol longer than a minimal length complete checking sequence. It seems very likely that it is a minimal length complete checking sequence, as one might verify by exhaustive examination of the 32 possible input sequences of length 5.

Remark. The shortest possible complete checking experiment with initialization must have length 7 (homing, setting to initial state, 4 transitions, verification of the state after the last transition). This observation shows that no approach using initialization can yield a general procedure for optimal complete checking experiments.

CONCLUSION

A new approach for designing checking experiments has been presented. For output faults an algorithmic method, based on this approach, has been described for obtaining checking sequences that are optimal in some cases and can be expected to be close to optimal otherwise. The ease of adapting or extending the algorithm (e.g., the implementation of additional data, possible simplifications, etc.) has also been shown. In general, the desirability of avoiding initialization when optimality is required, has also been illustrated. Although the design of checking experiments is not a very recent problem [1], no general procedure for designing optimal and near-optimal complete checking experiments is available in the literature. Therefore, it is hoped that the approach presented here will stimulate further research in this direction to gradually obtain optimality for more general classes of checking experiments and, as illustrated by the example, maybe even for complete checking experiments.

REFERENCES

Evaluation of Walsh Power Spectrum of Nearly White Signals

CHUNG KWONG YUEN

Abstract—An expression for evaluating the Walsh power spectra of random signals with very short time correlations is given.

Index Terms—Fast Walsh transform, power spectrum, signal-to-noise ratio, Walsh functions, white noise.

A recent paper [1] presented a technique for estimating the Walsh power spectrum of a random signal from its autocorrelation. The method expresses the spectrum as a weighted integral of one of the derivatives of the autocorrelation and then finds an upper bound on the magnitude in terms of the maximum or the mean absolute value of that derivative. While the technique gives fairly good results for some autocorrelations, the bounds tend to be very pessimistic for correlations which are nonzero only for short time differences. This is because such functions have large and rapidly varying derivatives, giving large upper bounds.

There are some brute-force methods to deal with the problem. One is to compute the spectrum, and to simply compute it, either using an expression (see [1]) or by a recently found fast algorithm [2]. Or, we could roughly assume that the correlation is nonzero only for zero time difference, i.e., a delta function. This implies that the spectrum is, approximately, flat, but gives no information whatsoever about its “fine structure.”

In this correspondence we present a method for evaluating the Walsh power spectrum for such nearly white signals (we might call them pale signals). The method allows one to find the ith element on the spectrum in terms of the binary digits of i. In contrast, the algorithm of [2] computes the whole spectrum and is thus uneconomical for evaluating one element.

Let us start with a few definitions. We shall denote the ith Walsh function \( \text{wal}(i,t) \) with i a nonnegative integer and \( t \in [0,1) \). Given a random variable \( x(t) \), the Walsh power spectrum consists of

\[
X_i = \left\{ \left( \int_0^1 \text{wal}(i,t)x(t) \, dt \right) \right\}, \quad i = 0, 1, 2, \ldots
\]

In practice we require only the first \( 2^n \) elements. The power spectrum can be related to the autocorrelation function as

\[
X_i = \int_0^1 \text{wal}(i,t) \text{wal}(i,s) (x(t)x(s)) \, dt \, ds
\]

Denoting \((x(t)x(s))\) by \(r(t-s)\), with time invariance assumed, we have

\[
X_i = \int_0^1 \text{wal}(i,t) \text{wal}(i,s)r(t-s) \, dt \, ds
\]

If \( r \) is a delta function, then \( X_i \) is constant.

Now we assume that

\[
r(t-s) = 0, \quad \text{unless} \quad t - a \leq s \leq t + a
\]

where \( a \) is a small positive number \( a < 2^{-n} \). Then

\[
X_i = \int_0^1 \int_0^1 \theta(1-s) \text{wal}(i,t) \text{wal}(i,s) r(t-s) \, dt \, ds
\]

\[
+ \int_0^1 \int_{-a}^{1-a} \theta(s) \text{wal}(i,t) \text{wal}(i,s) r(t-s) \, ds \, dt
\]

where we have inserted step functions to ensure that \( s \) is not less than \( 0 \) and not more than \( 1 \). Changing the integration variable to \( q = s - t \) in the first part and to \( q = t - s \) in the second gives

\[
X_i = \int_0^1 \int_0^1 \theta(1-t-q) \text{wal}(i,t) \text{wal}(i,q + t) r(-q) \, dq \, dt
\]

\[
+ \theta(q-t) \text{wal}(i,t) \text{wal}(i,1-q) r(q) \, dq \, dt
\]

Let us integrate over \( t \) first. We note that in the first term the range is from \( 0 \) to \( 1 - q \) because of the step function, while in the second term the range is from \( q \) to \( 1 \). However, the two results are the same because in both we have the argument of one wal going from \( 0 \) to \( 1 - q \) and that of the other from \( q \) going to \( 1 \). Defining

\[
R(i,i;\theta) = \int_0^1 \text{wal}(i,t) \text{wal}(i,t + q) \, dt
\]

we have

\[
X_i = \int_0^1 [r(-q) + r(q)] R(i,i;\theta) \, dq
\]

We now introduce an important simplification. Let us confine \( i \) to beless than \( 2^n \). Now, \( \text{wal}(i,t) \) is constant in each of the \( 2^n \) sections \((j^{2^n} \leq i < (j+1)^{2^n})\), \( j = 0,1, \ldots, 2^n - 1 \). This means that \( R(i,i;\theta) \) varies linearly with \( q \) because \( q \) is confined to one interval. Thus, if we find \( R(i,i;\theta) = 0 \) and \( q = 2^n \), we can then find it for all \( q \) of interest by linear interpolation. By orthogonality \( R(i,i;\theta) \) is just 1. \( R(i,i;2^n) \) has been found for all \( i \), as shown in the Appendix. Thus

\[
R(i,i;\theta) = R(i,i;0) - 2q R(i,i;2^n) = 1 - (2i + 1)q
\]

It can be easily verified that this has the required values at \( q = 0 \) and \( q = 2^n \). Further, we note that \( r \) is symmetric. So

\[
X_i = \int_0^1 2q r(q) \, dq - (4i + 2) \int_0^1 r(q) \, dq
\]

If \( r \) had been replaced by a delta function, the first term, independent of \( i \), remains, but the second term would vanish. Thus, what we have is a flat spectrum with some additional fine structure.

Let us take an example, \( r(q) = C(1 - |q|/a) \) for \( |q| < a \) and 0 elsewhere. It is easy to find that

\[
X_i = Ca - (2i + 1)Ca3/3.
\]

For an approaching 0 the second term becomes negligible. However, if \( a \) is comparable to \( 2^n \), then the term can be significant as \( i \) may be \( 2^n - 1 \). We remind the reader that the formula is not applicable to larger \( i \). It might also be mentioned that the \( r(q) \) assumed in the example is the autocorrelation of certain classes of random noise [4].