Accelerated Static Compaction for Sequential Circuits by Exploiting "Essential" Subsequences

MICHAEL DIMOPOULOS, PANAGIOTIS LINARDIS*
Department of Informatics
Aristotle University of Thessaloniki
GR-54006 Thessaloniki
GREECE

Abstract: - In this paper a GA-based method that compacts Test Sequences for sequential circuits is presented. In this algorithm from an initial set of test sequences a subset of sequences is selected and from these sequences parts of them (subsequences) are selected which are reordered and combined into one sequence covering all faults detected by the initial set. The method exploits the presence of essential sequences in the set to reduce the search space, thus easing the work of the GA. Experimental results support the usefulness of the proposed method.

Key-Words: - Sequential Digital Circuits, Sequence Compaction, Test Generation, Genetic Algorithms.

1 Introduction

As circuit size grows, the need of correct circuit functioning imposes a testing methodology to be applied to ensure proper behavior of that circuit after manufacturing. Towards that direction, Test Pattern Generation procedures are devised to validate the circuit function [1,3,4,5,6]. These procedures produce sequences of input vectors (test sequences) that when applied to circuit inputs produce output responses that enable one to check the “correct” circuit functioning. Since the length of a test sequence affects the test application time, test sequence compaction is used to reduce the test length thus reducing test application time [1, 2].

In this paper we concentrate on sequential circuits and our main focus is on compaction of test sequences for sequential circuits. Two types of compaction techniques exist: static and dynamic [1]. The dynamic compaction method works concurrently with the test generation algorithm.

Static compaction is applied at a post-processing step of the test generation process and is independent of the test generation itself. Since the Automatic Test Pattern Generation (ATPG) problem is NP-complete many ATPG algorithms end up either with one very long test sequence [2] or with a population of sequences each one partially covering the circuit faults, as for example do some algorithms based on Fault Targeting [1] or Genetic Algorithm (GA) techniques [2]. For such ATPG algorithms a static compaction step becomes necessary.

Several static compaction techniques have been proposed in the literature [9-15]. For example in [11] static compaction is applied to sets of test sequences produced during test generation by targeting a single fault or sets of faults. In [13] a Genetic Algorithm is used to statically compact test sequences by reordering them so that some of them are shortened and some others eliminated.

In [10] a procedure named Vector Selection is proposed. In this procedure, from a single test sequence T are extracted test subsequences for every fault. After collecting all subsequences a covering method is applied to select a minimal subset of sequences to detect all faults. The placement of the selected subsequences in the new (compressed) sequence follows the order by which they appear in the original sequence.

In [12], the so called Reverse Order fault simulation method, the candidate test sequences are concatenated in the reverse order to that produced by the test generation algorithm and then the new sequence is fault simulated.

In the present paper we propose a procedure which, given a set of test sequences, forms a new sequence by selecting and concatenating in a certain order parts of these sequences (subsequences) so that the new sequence has shorter length. The ordering of the sequences and the selection of the proper sequence lengths is made possible by: (a) extracting subsequences which start from the...
unknown “X” state, (b) applying a Genetic Algorithm (GA) to determine the order and length of the subsequences to be selected, and (c) exploiting the so called essential subsequences to reduce the complexity of the problem.

The paper is organized as follows: In section 2 our method is analyzed. In section 3 the proposed GA method is presented. In section 4 experimental results are given, supporting the potential of the proposed method.

2 Compaction Methodology

The test sequence compaction problem may be formulated, initially, as a set covering problem [13, 16] i.e. given a set of \( n \) test sequences, where each one partially covers the circuit faults, find the minimum subset of sequences collectively covering all faults. The complexity of this problem alone is \( n! \).

The test length (collective) may be further reduced if not the full length but only part of each sequence is selected. This process, however, increases the complexity even more. To combat the complexity of the problem we employ the following techniques.

First, in order to be free to combine segments of sequences into one sequence without loosing their testing properties (faults detected) we exploit the fact that segments of sequences which start from the unknown “X” state may be combined with others (concatenated), in any order, without compromising the obtained fault coverage. That is, the new sequence is guaranteed to detect at least, if not more, all faults that were detected by the individual segments.

Second, the complexity of the proposed algorithm is reduced by observing that, usually, within a given set of test sequences there exists a subset of so called essential subsequences [17] out of which essential subsequences may be distinguished (section 2.2).

The final selection of the subsequence lengths and the order in which they will be combined is done with the help of a GA (section 3).

2.1 Problem Formulation

The problem of extracting from a set and compressing test sequences with the purpose of achieving the shortest sequence may be defined as follows: Let \( T \) a test set consisting of the \( n \) sequences \( S_1, S_2, \ldots, S_n \) with lengths \( L_1, L_2, \ldots, L_n \) and let \( F=[f_1, f_2, \ldots, f_m] \) be the set of circuit faults \( f_i \), \( i=1,\ldots,m \), currently detected by the set \( T \). The problem posed here is to select a subset \( T_F=[S_{i_1}, S_{i_2}, \ldots, S_{i_p}] \) of \( T \) that will cover the set \( F \). Further, from each sequence of \( T_F \) is selected not the full length \( L_k \) but a subsequence \( S_{Ck} \) of a smaller length \( L_{Ck} \) \( (L_{Ck} < L_k) \), but sufficiently long so that the fault coverage is not compromised. Then all \( S_{Ck} \) are combined into one sequence \( S_0 \) having length \( L_0 \):

\[
L_0 = \text{Sum}(L_{Ck}) < \text{Sum}(L_k) < \text{Sum}(L_0)
\]

The new sequence \( S_0 \) will then cover the set \( F \) and will have a much smaller length \( L_0 \). It is noted that this is a combinatorial selection process and no fault simulations are implied here. When \( S_0 \) is applied to a circuit then, probably, extra faults, not contained in \( F \), may be detected.

While the compaction efficiency depends on the set \( T_C \) and the lengths \( L_{Ck} \) of its members, the observation that within the initial set \( T \) exist essential sequences and that for each of them there is a minimum length that can be chosen helps to simplify the selection problem.

2.2 Essential Subsequences

Within the population \( T \) of sequences it is observed that there are some that detect at least a fault not detected by any other sequence. Because of their unique fault detection characteristics these sequences must take part in every candidate combination. They are called essential sequences [17] and the corresponding faults essential faults. Also, if the other faults (non-essential) covered by the essential sequences are crossed out from the remaining sequences then many of the remaining sequences may become redundant and can be excluded from further consideration.

Following the above observation, the set \( T \) is partitioned into the sets: \( T_{ess} \), the set of essential, \( T_{ness} \), the set of non-essential, and \( T_{red} \) the set of redundant sequences. Since \( T_{ess} \) belongs by default to \( T_C \) the search space for \( T_F \) becomes now the (smaller) set \( T_{ess} \).

In an essential sequence the location of the vector that detects the last essential fault within it defines the length of the minimum subsequence, called here essential subsequence that can be extracted from that sequence without loosing its essential detection properties.

The above principles and their exploitation in the selection process will be explained with the following example.
Example

Let’s assume that we must compact the sequences $S_1$, $S_2$, $S_3$ of fig. 1.

![Fig.1. A set of test sequences](image)

Since there are three test sequences the possible orderings are $3! = 6$. In Table 1 are presented the $3!$ orderings and their impact on the final sequence.

From the table we see that the optimal ordering that gives the most compact sequence is $T_3 = (S_2, S_1, S_3)$, which gives a sequence of length $L_3 = 10$ vectors, because $S_3$ and the tail of $S_1$ (from $v_4$ to $v_7$) become redundant.

![Table 1. Sequence orderings](image)

If we observe, however, that sequence $S_2$ (fig. 1) is an essential sequence, since uniquely covers fault $f_3$, then $S_2$ may be removed straight from the beginning and included in the final test set along with the covered faults: $f_1, f_2, f_3, f_4, f_5$.

Now the task is simplified because only the faults $f_6$, $f_7$ must be covered with the remaining 2 subsequences (2 sequences instead of 3 to cover 2 out of 7 faults). Since the whole part of $S_2$ is selected the best result one can obtain is a final sequence with $L=10$ vectors.

Selecting parts of sequences further, refines our method. If only the minimum subsequence of $S_2$ (fig. 1) is selected to be included in the final test sequence, namely the essential part $S_2[v_1,...,v_5]$ i.e. that which detects fault $f_3$, then the faults ($f_1$, $f_4$, $f_7$) that $S[v_1,...,v_5]$ covers may be crossed out of the initial set of faults. Therefore from the 7 initial faults 4 remain to be covered.

If the minimum length is relaxed then from fig. 1 it is seen that the sequence (shaded parts in fig. 1) $T_{C4} = (S_2[v_1,...,v_5], S_3[v_1,...,v_4])$ covers all faults with length $L_{C4} = 9$ vectors instead of $L=10$. In the same way the sequence $T_{C3} = (S_2[v_1,...,v_5], S_3[v_1,...,v_5], S_3[v_1])$ gives also $L_{C3} = 9$ vectors. From the example it is seen that subsequence selection achieves more compact sequences than sequence reordering.

Finally, the decision of whether to increase part $S_2$ or not is left to a Genetic Algorithm to handle (section 3).

3 The GA Compaction algorithm

The strength of genetic algorithms in search optimization problems [7] is long known and they have been used to solve set-covering problems [16]. A schematic of the genetic algorithm we adopted here is seen in fig. 2.

![Figure 2. The GA-Compact schematic](image)

For the problem we are dealing here we choose to encode each chromosome (individual) as an integer string of ordered numbers. Two characteristics must be considered: a) sequence number and b) the part of the sequence (subsequence length) to include in the final solution. The above quantities are encoded as follows:

To every sequence a number is assigned e.g. $n$ for sequence $S_n$. If $L$ is the length of the largest subsequence then sequence $S_n$ is represented by a number $k \in [n*L,...,(n+1)*L)$. For example if $n=2$, $L=10$ and sequence length is $L_2=8$ the number 27
denotes sequence $S_2[v_1, \ldots, v_8]$ and the number $25=(2*10+5)$ denotes subsequence $S_2[v_1, \ldots, v_6]$. Referring to fig. 1, the selected sequence $(S_1[v_1, \ldots, v_3], S_2[v_1, \ldots, v_5], S_1[v_4, \ldots, v_4])$ is encoded as: $[3 \ 12 \ 15]$ and selection $(S_2[v_1, \ldots, v_6], S_3[v_1, \ldots, v_4])$ is encoded as: $[0 \ 12 \ 18]$.

The Fitness function of the GA is selected so that it gives emphasis to individuals with smaller lengths.

The GA procedure for this method is presented in fig. 3. Procedure Improve_population, in fig. 3, is a local search algorithm that marks the essential sequences and performs an initial ordering.

4 Experimental Results

Our GA-based Compaction algorithm (GA-Compact), has been implemented in C. The efficiency of the algorithm was measured by running the ISCAS’89 benchmark circuits [8] on a Pentium PC with 256 Mb.

In Table 2 our method GA-Compact (GA-C column) is compared against results produced by Reverse Order fault simulation [12] (column B) and by simple fault simulation (column A). The test sequences (#seq. is the number of sequences and #vectors is the total amount of vectors) were obtained by a GA-ATPG method [17]. Column #EssS represents the number of essential sequences found in the population #seq. From Table 2 we see that our method attains more compact final test sequence.

In Table 3 are presented results about the proposed compaction method (GA-Compact) with respect to the method of [13] which is a GA-based compaction method, when applied to sets of test sequences produced by GATTO and HITEC algorithms.

5 Conclusion

The test sequences compaction problem is formulated as a set-covering problem. From an initial set of test sequences a properly selected subset is reordered and combined into a more

From Table 3 we see that significant reduction to the already compacted test sequences can be obtained for the most of the circuits.
compact test sequence that detects all the faults detected by the individual sequences. An algorithm based on a GA framework to solve this set-covering problem is presented. The notions of essential faults and essential sequences are used to prune the search space. Experimental results are given to support the usefulness of the proposed method.

References: