International Symposium on IC-Technology, Systems & Applications (ISIC'97)
pp. 312-315, Singapore, 1997

EFFICIENT FUNCTIONAL DIAGNOSIS FOR SYNCHRONOUS SEQUENTIAL CIRCUITS BASED ON AND/OR GRAPHS

Rolf Drechsler[1] and Alenka Žužek[2]

[1]Inst. of Comp. Sci., Albert-Ludwigs-University, 79110 Freiburg im Breisgau, Germany
email: drechsle@informatik.uni-freiburg.de
[2]Comp. Sys. Dept., Jožef Stefan Institute, Jamova 39, 1111 Ljubljana, Slovenia
email: alenka.zuzek@ijs.si

Abstract

In this paper we present a new model for diagnosis of errors in Synchronous Sequential Circuits (SCC) on the functional level. In contrast to many previously published approaches we do not consider a specific implementation. Instead we use tests based on the transition behavior of the corresponding Finite State Machine (FSM). Thus, the approach can be used for verification and test. We describe a method for constructing a minimal cost test based on AND/OR graphs. Exact and heuristic methods are presented. First experimental results for randomly generated FSMs are given that demonstrate the efficiency of our approach.

1 Introduction

Nowadays, circuit design is becoming more and more complex. Thus, the error probability also increases. Since time-to-market aspects are increasingly important it is desirable to detect errors as early as possible. Additionally, this also reduces the production costs. For this, nowadays CAD tools should also support features for error diagnosis, i.e. error detection *and* correction.

Diagnostic algorithms presented so far work only on the gate level and thus the result is fixed to one specific design (see e.g. [9]). These approaches have the drawback that for each implementation of the function the error diagnosis has to be performed again, e.g. after resynthesis.

In this paper we consider SSCs on a higher level of abstraction, i.e. we consider a SCC as a *Finite State Machine* (FSM) and model the behavior by transition relations. Only two approaches for diagnosis for SSC have been proposed so far, but both algorithms work on the gate level description [2, 8]. In contrast, we consider the problem of diagnosing faults on the functional level.

For this, we first introduce a new model for this level of abstraction. Given a FSM description we assume with a given probability that the system is "in error", i.e. the FSM is currently in a wrong state. (Notice that the approach can be applied to the reset state, but also to each state during normal operation.) By applying test vectors it can be checked whether this assumption was correct. If a faulty state is identified it is not only determined that a fault occurred, also the faulty transition is identified. The test vectors can be derived analogously to some classical approaches to test pattern generation for sequential circuits. (For an overview of the different methods that can be applied, if the state transition graph becomes too large see Chapter 4 of [1].)

Obviously, a critical point for the model is the construction of a "cheap" test, i.e. a test of minimal length. Due to the probablistic assumption we often have to check several different states. Thus, it is desirable to have a test sequence that checks the more likely faults first. We give an algorithm for identifying a test sequence of minimal length based on AND/OR graphs. AND/OR graphs are mainly used in the area of artificial intelligence, but have recently been proposed for CAD problems [7]. In our approach we not only use the underlying structure, but also make use of the efficient search algorithms known from artificial intelligence, i.e. exact and heuristic algorithms (see e.g. [3, 4, 5, 6]). Thus, algorithms developed for sequential diagnosis on the system level [10] can be applied in the model described here. We give experimental results to demonstrate the efficiency of our approach. We randomly generate FSMs and determine a minimal test with respect to our model.

2 Synchronous Sequential Circuits

We describe a *Synchronous Sequential Circuit* (SSC) as a *Finite State Machine* (FSM) $M = (I, S, O, \delta, \lambda)$. In the following we will exchange the notations usual for SSCs and FSMs without mentioning it further.

We have the input set $I = \mathbf{B}^n$, the output set $O = \mathbf{B}^l$ and the set of states $S = \mathbf{B}^k$ with $\mathbf{B} = \{0, 1\}$. The next-state function δ and the output function λ are computed by a *Combinational Logic Circuit* (CLC) C. In the following we will not make any assumption about the realization of the CLC, i.e. we only argue on the functional level. The inputs (outputs) of the CLC which are connected to the outputs (inputs) of the memory elements are called secondary inputs (outputs). We also call the secondary inputs the *present state variables* and the secondary outputs the *next state variables*.

3 Diagnostic Model

In this section we describe our new diagnosis model for SSCs. Before we formally introduce the model

we give an informal description to simplify the understanding.

3.1 Basic Idea

In normal operation the SCC should be initialized (by a reset signal or a reset sequence). Then starting from the initial state s_0 dependent on the inputs of the circuit the FSM goes to different states.

If everything is operating correctly so far lets assume that the system is in state s_i after k steps. We now assume with some probability that the system might be in a different state, say s_j. For one given state s_i and one state s_j it is easy to compute a sequence of input assignments such that the correctness is verified.

In the following we do not only consider one state s_j. Instead we consider a set of states s_l ($l \in \{0, \ldots, m\}$) that are the present state in the case of an error. For each state there exist tests that check whether this fault might be possible. For each state a different probability might be considered since depending on the correct state other parts in the FSM may be effected, e.g. as a result of state encoding.

Beside the definition of the model we want to consider in the following the problem of finding an optimal test sequence, i.e. for all assumed errors the cheapest (=shortest) test sequence should be obtained that uniquely identifies a faulty state.

In the rest of this section we define our model more formally.

3.2 Model Definition

Let $M = (I, S, O, \delta, \lambda)$ be a FSM as defined above. Then we have:

1. The set of states $S = \{s_0, s_1, .., s_m\}$ is given, where s_j denotes the *fault-free* state and s_i, $(0 \leq i \leq m)$ denotes one of m potential faulty states of the system. (We also allow that all states may be considered as faulty states. Some states can also be excluded by setting the fault probability to 0 (see below).)

2. The set of probabilities $p = \{p(s_0), .., p(s_m)\}$, where $p(s_i)$ is the *a priori* probability of the FSM being in the state s_i.

3. $T = \{t_1, t_2, .., t_n\}$ is a finite set of available tests and $c = \{c_1, .., c_n\}$ is the set of test costs measured in terms length of the test e.g. determined by a test pattern generator. The cost c_i is the sum of costs of operations needed to perform test t_i. (It is also possible to extend the approach by using dependent test, i.e. the cost of a test may vary depending on the direct predecessor. This description is left out for the simplicity of the model description.)

4. The test matrix $D = [d_{ij}]$, $d_{ij} \in \{0, 1\}$, such that test t_j fails, if the system is in state s_i, if d_{ij} is 1. (In [10] also real values in the range between 0 and 1 have been considered as the consequence of the inclusion of test unreliabilities.)

5. $Sym = \{sym_1, .., sym_r\}$ is a finite set of symptoms. In a symptom matrix $F = [f_{ij}]$, where $0 \leq f_{ij} \leq 1$, the element f_{ij} denotes the *a priori* probability of system state s_i being the candidate failure state in the presence of symptom sym_j. (*Symptoms* can be defined based on the information about the state encoding, e.g. only states that differ in only one bit should be considered. This obviously reduces the search space and simplifies the test sequencing problem.)

Based on these definitions the identification of an efficient test can be formulated as follows:

Find a test sequence of minimal average cost that is able to isolate each state using the test set T and the symptoms Sym.

4 Construction of Optimal Test by AND/OR Graphs

In this section we give an algorithm for the solution of the test sequencing problem described above based on AND/OR graphs. For the exact and heuristic solution we use algorithms from artificial intelligence.

4.1 AND/OR Graphs

We now briefly review the main notations and definitions for AND/OR graphs:

Definition 1 An *AND/OR graph* is a rooted DAG $G = (V, E)$ with $V = V_{AND} \dot{\cup} V_{OR} \dot{\cup} \{r\}$ with the following properties:

1. There is a single root node r.
2. Adjacent nodes belong to different sets in V, i.e. nodes in V_{AND} and V_{OR}, respectively, are never connected by an edge.
3. The leaves of the graphs are nodes $v \in V_{OR}$.

This is a purely structural definition. The interpretation with respect to our context will be as follows:

The root of the AND/OR graph represents the problem to be solved. This problem is subdivided by some tests (corresponding to nodes in V_{AND}) in several sub-problems that have to be solved (nodes in V_{OR}). Each time a test is applied the set of possible faulty states is reduced. The tests are applied until a unique state is identified (if possible). With respect to our model the optimal solution in the AND/OR graph represents the best average test cost that can be expected (see below).

4.2 Algorithms

The algorithms for exact and heuristics solving of the sequencing problem have been implemented for AND/OR graphs. The algorithms have been implemented in a sequential diagnosis system for diagnosis on the system level [10]. The user can choose among different algorithms that compute diagnostic decision trees. These algorithms require different amounts of time and space and provide different solution quality.

It is well known that the construction of the optimal decision tree is an NP-hard problem, therefore it is necessary to explore heuristic approaches for guiding the AND/OR graph search.

The existing solution approaches can be mainly categorized into two different groups:

Sub-optimal test algorithms: A class of sub-optimal algorithms provides a trade-off between optimality and computational complexity. They perform a local step-by-step optimization. The diagnostic

System states	Tests t_1	t_2	t_3	Probabilities $p(s_i)$
s_0	0	1	1	0.25
s_1	1	1	0	0.15
s_2	1	0	1	0.60

Table 1 – Test matrix and fault probabilities

tool described in this paper offers two such algorithms: *separation heuristic* and *information heuristic* [3].

Optimal test algorithms: Also the exact approaches use problem-domain knowledge in the form of a *Heuristic Evaluation Function* (HEF), to avoid enumerating the entire set of potential solution trees. The HEF is an easily computable heuristic estimate $h(x)$ of the optimal cost-to-go, from any node of ambiguity subset to the goal nodes of zero ambiguity.

The implementation of the optimal AND/OR graph search techniques in [10] corresponds to algorithm AO^* [4]. The algorithm AO^* is ordered best-first search algorithm; it expands only the node of the search graph that offers the most promising way of reaching the goal nodes on the basis of two heuristics: (i) the first is derived by appealing to the analogy between the test sequencing and the Huffman coding problem (HEF_1) and (ii) entropy-based heuristics (HEF_2).

Using heuristics HEF_1 or HEF_2, AO^* is guaranteed to find an optimal solution. The description of the heuristics and the proof of optimality of AO^* is given in [5]. Decision between using HEF_1 and HEF_2 depends on the size of the problem. For large problem instances a heuristic entropy-plus-one-based, called HEF_3, is preferred. In general HEF_3 does not guarantee optimal solutions, but it outperforms HEF_1 or HEF_2 clearly with respect to runtime as will be demonstrated in the next section. (For more details see [5, 6].)

The resulting test sequence can be described by a binary AND/OR decision tree, with OR nodes containing current candidate system states, and by AND nodes denoting tests for partitioning the set of candidate system states. The leaves of the tree correspond to the individual failure state with no ambiguity, and the average length of the tree represents the expected test cost of the system.

Example 1 We consider a FSM with three possible states $S = \{s_0, s_1, s_2\}$. A set of three tests labeled t_1, t_2, and t_3 is available for checking the FSM. For simplicity of the example we chose the test lengths $C = \{1, 1, 1\}$. The *a priori* probabilities of the FSM being in one of the states along with the test matrix is given in Table 1. In the test matrix of dimension 3×3 each test is represented by a binary column vector, e.g. test t_1 will pass if the system is in the fault-free state s_0, hence, "0" is assigned in the first position of test t_1. "1"is assigned for states s_1 and s_2, therefore test t_1 will fail if the FSM is in any of these two states.

The complete AND/OR graph for this example is provided in Figure 1. The initial OR node S represents the test sequencing problem to be solved. Any search algorithm for solving this problem chooses one of the three successor AND nodes t_1, t_2, or t_3 for expansion. If e.g. node t_1 is chosen, the problem left is to find a solution for the right successor. (The left successor already corresponds to the individual state s_0.) To partition between s_1 and s_2 either t_2 or t_3 can be used.

The optimal solution for the problem from Table 1 is obtained by applying the algorithm AO^* using HEF_1. The corresponding branches in the AND/OR graph in Figure 1 are given in bold. The sequence (t_2, t_1) is found to yield the minimal length J^*. (In fact, the sequence (t_2, t_3) is also optimal). It is given by:

$$J^* = p(s_0) * c_2 + [p(s_1) + p(s_2)] * [c_1 + c_2] = 1.4$$

For this, when the diagnostic tree is used, first test t_2 is applied. If test t_2 fails, we continue with test t_1. If this test passes, the system is classified to be fault-free.

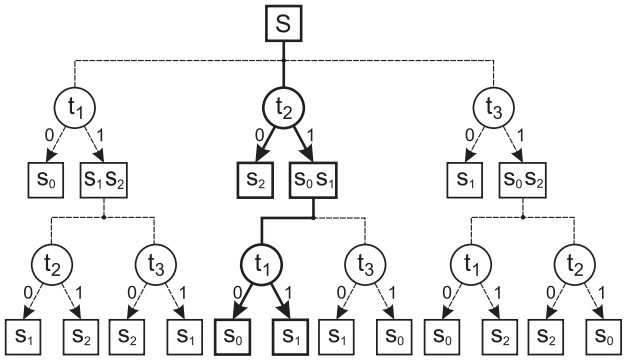

Figure 1 – AND/OR graph

5 Experimental Results

In this section we present experimental results for several randomly generated FSMs. The test problems were generated using a uniform distribution. For simplicity all tests have cost one. (A different choice of these values has no influence on the algorithm.) For all examples the number of tests was chosen $2\times$ the minimal number of tests required (see [5]). We made no assumptions on symptoms, i.e. we considered the problem in its full generality, even though our software can handle these cases. All experiments have been carried out on an *HP 9000-J210* workstation with *CPU PA-7200* and 160 MByte of main memory. All runtimes are given in CPU seconds and minutes.

In the following we study the behavior of our algorithms, i.e. the exact and the heuristic algorithms, with respect to runtime and space complexity.

We first consider small problem instances, i.e. examples for which the exact result can be determines within one hour of CPU time. The results are shown in Table 2. A "-" symbolizes that the algorithm took more than the given time limit. We applied all different search algorithms presented in the previous section for each example. PQ, HEF_1, HEF_2 and HEF_3 denote the information-heuristic algorithm and

States	Algo-rithm	Nodes gener.	Back-tracks	Cost	CPU time
10	PQ	19	0	3.0848	0.1
	HEF1	25	4	3.0848	0.4
	HEF2	46	26	3.0848	0.5
	HEF3	19	0	3.0848	0.4
20	PQ	39	0	4.0598	0.1
	HEF1	152	79	4.0043	1.0
	HEF2	207	331	4.0043	1.7
	HEF3	59	7	4.1667	0.6
30	PQ	59	0	4.8318	0.3
	HEF1	227	79	4.7641	2.2
	HEF2	581	923	4.7641	11.1
	HEF3	91	14	4.9180	1.2
50	PQ	99	0	5.5727	1.5
	HEF1	1015	887	5.4425	15.5
	HEF2	1843	5137	5.4425	1 : 51.7
	HEF3	178	32	5.6438	2.6
70	PQ	139	0	6.0877	1.9
	HEF1	2567	2777	5.9365	1 : 55.4
	HEF2	4040	11596	5.9365	12 : 14.8
	HEF3	222	36	6.1119	5.1
90	PQ	179	0	6.3416	4.4
	HEF1	2146	1412	6.3046	1 : 13
	HEF2	7658	20879	6.3046	45 : 30.9
	HEF3	247	28	6.4198	9.2
100	PQ	199	0	6.5678	5.5
	HEF1	3912	3493	6.5079	4 : 37.5
	HEF2	-	-	-	-
	HEF3	433	134	6.6994	15.6
120	PQ	239	0	6.7851	7.4
	HEF1	9669	10235	6.7229	32 : 46.3
	HEF2	-	-	-	-
	HEF3	663	287	6.8814	30.5

Table 2 – Comparison of exact vs. heuristic methods

*AO** with Huffman code-based heuristics, entropy-based heuristics and entropy entropy-plus-one-based heuristics, respectively. The third column displays the number of nodes generated by the search algorithms. The next column shows the number of backtracks. These two items are important for comparing the efficiency of the heuristics, that provide optimal solutions, i.e. HEF_1 and HEF_2. Note that HEF_1 requires fewer nodes and backtracks than HEF_2, which reflects that entropy-based heuristics are poor in comparison to Huffman code-based heuristics. The best average costs of the resulting diagnostic decision tree and the runtimes are given in the last two columns. As can be seen the heuristic algorithms determine the exact result only for the smallest example considered. For the larger problem instances the results are not bad, even though non-optimal. The exact algorithms can only be applied up to 100 states within reasonable time bounds.

In a second series of experiments we studied the behavior of the heuristic algorithms for larger problem instances, i.e. FSMs with up to 1000 states are considered. For these examples the exact algorithms fail due to their exponential behavior. The results are given in Table 3. As can been seen PQ is much faster and additionally obtains smaller average test costs. Even for FSMs with 1000 states PQ terminates in less than 10 CPU minutes. Thus, the presented algorithm is also applicable to larger problem instances.

6 Conclusions

We introduced a new model for diagnosis for synchronous sequential circuits modeled as FSMs. The process runs on a functional level, i.e. the resulting tests are independent of a gate level realization.

States	Algo-rithm	Nodes gener.	Back-tracks	Cost	CPU time
150	PQ	299	0	7.1437	12.1
	HEF3	1292	621	7.1468	1 : 28.4
200	PQ	399	0	7.5129	15.5
	HEF3	1688	671	7.5570	2.4
300	PQ	599	0	8.0944	34.6
	HEF3	5106	3331	8.1145	15 : 38.7
400	PQ	799	0	8.5232	1 : 09.1
	HEF3	5540	3318	8.5954	27 : 14.4
500	PQ	999	0	8.8451	1 : 36.8
	HEF3	-	-	-	-
600	PQ	1199	0	9.1218	2 : 29.1
	HEF3	-	-	-	-
800	PQ	1599	0	9.4977	4 : 51.1
	HEF3	-	-	-	-
1000	PQ	1999	0	9.8252	8 : 42.8
	HEF3	-	-	-	-

Table 3 – larger problem instances

We proposed an AND/OR graph based method to determine a test with a minimal cost by solving the test sequencing problem. Since this problem is in general NP-hard we use exact methods for small problem instances and heuristic methods for larger examples. Experimental results have been demonstrated that our approach can be applied to large FSMs within reasonable time bounds.

References

[1] X. Chen and M. Bushnell. *Efficient Branch and Bound Search with Application to Computer Aided Design*. Kluwer Academic Publisher, 1996.

[2] M. Fujita. Methods for automatic design error correction in sequential circuits. In *European Design & Test Conf.*, pages 76–80, 1993.

[3] E.J. Kletsky. An application of the information theory approach to failure diagnosis. *IRE Trans. on Reliability and Quality Control*, 9:29–39, 1960.

[4] N.J. Nilsson. *Principles Of Artificial Intelligence*. Springer Verlag, 1982.

[5] K.R. Pattipati and M.G. Alexandridis. Application of heuristic search and information theory to sequential fault diagnosis. *IEEE Trans. on Systems, Man, and Cybernetics*, 20:872–887, 1990.

[6] K.R. Pattipati and M. Dontamsetty. On a generalized test sequencing problem. *IEEE Trans. on Systems, Man, and Cybernetics*, 22:392–396, 1992.

[7] D. Stoffel, W. Kunz, and S. Gerber. AND/OR graphs. In *Technical Report, MPI-I-95-602*, 1995.

[8] A. Wahba and D. Borrione. *Design Error Diagnosis in Sequential Circuits*, volume 987 of *LNCS*. CHARME, 1995.

[9] A. Wahba and D. Borrione. A method for automatic design error location and correction in combinational logic circuits. *Jour. of Electronic Testing: Theory and Applications*, 8:113–127, 1996.

[10] A. Žužek, F. Novak, A. Biasizzo, I. Savnik, and B. Cestnik. Sequential diagnosis tool for system maintenance and repair. *Electronical Review, Ljubljana*, (62)3-4:224–231, 1995.