

# Error Diagnosis in Sequential Multi-Valued Logic Networks

Rolf Drechsler

Institute of Computer Science  
Albert-Ludwigs-University  
79110 Freiburg im Breisgau, Germany  
email: drechsle@informatik.uni-freiburg.de

Alenka Žužek

Computer Systems Department  
Jožef Stefan Institute  
Jamova 39, 1111 Ljubljana, Slovenia  
email: alenka.zuzek@ijs.si

## Abstract

*In this paper we present a model for diagnosis of errors in Sequential Multi-Valued Logic Networks (SMVLN). The method allows not only to detect errors in an implementation, but also identifies the fault location. In contrast to many previously presented approaches this model does not consider a specific implementation. Instead the model assumes tests based on the transition behavior of the corresponding MVL Finite State Machine (FSM) on the functional level.*

*We present a method for constructing a minimal cost test based on AND/OR graphs using tests with MV outcomes. The model enables encoding over two-valued circuits as well as consideration of SMVLNs. The new approach provides efficient solution even for large MVL FSMs with up to 50000 states. Experimental results for randomly generated FSMs are given that demonstrate the efficiency of our approach.*

## 1 Introduction

Several circuit design methods for *Multi-Valued Logic* (MVL) have been proposed in the past few years [3, 6]. These new approaches raise hope in solving problems such as pin limitations and interconnection difficulties in VLSI design. But beside the synthesis step several tools are needed for supporting the designer. 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, since this reduces the production costs. For this, several verification approaches have been suggested (see e.g. [9]). Recently, also first methods for *MVL Networks* (MVLNs) have been discussed [4]. Furthermore, it is desirable to not only detect an error, but also to diagnose it, i.e. nowadays VLSI CAD tools should support features for error detection *and* correction.

Diagnostic algorithms presented for Boolean networks so far work mainly on the gate level and thus the result is fixed to one specific design (see e.g. [12]). 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 *Sequential MVLNs* (SMVLN), i.e. MVLNs with memory elements, on a higher level of abstraction. We model a SMVLN as a *Finite State Machine* (FSM) and describe the behavior by transition relations. Only few approaches for diagnosis for sequential circuits have been proposed so

far, but the algorithms work on the gate level description (see [5]). In contrast, we consider the problem of diagnosing faults on the functional level.

Recently, a new diagnosis model has been introduced in [5] that allows to model binary sequential circuits as FSM purely on the functional level. In this paper we show that this model can be extended to MVL: Given a FSM description of a SMVLN 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 [2].

Obviously, a critical point for the model is the construction of a “cheap” test, i.e. a test of minimal length. Due to the probabilistic 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 with MV outcomes. AND/OR graphs are mainly used in the area of artificial intelligence, but have recently been proposed for CAD problems [8]. In our approach we do not only use the underlying structure, but also make use of the efficient search algorithms known from artificial intelligence (see e.g. [7, 11]).

Thus, algorithms developed for sequential diagnosis on the system level [1] 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. In contrast to the two-valued approach, where FSMs with only 1000 states could be handled, our multi-valued approach can work with FSMs with up to 50000 states.

## 2 Sequential MVL Networks

In this section we introduce sequential multi-valued logic networks.

We start with the combinational case: In general, a *Multi-Valued Logic Network* (MVLN) can be modeled as a directed acyclic graph  $C = (V, E)$  with some additional properties: Each vertex  $v \in V$  is labeled with the name of a basic cell or with the name of a *Primary Input* (PI) or *Primary Output* (PO). The collection of

basic cells available is given by a fixed library. This library contains MIN-, MAX-<sup>1</sup>, INV- and LITERAL-gates. Of course, basic cells with arbitrary complexity, especially with an arbitrary number of inputs, are possible. There is an edge  $(u, v)$  in  $E$  from vertex  $u$  to  $v$ , iff an output pin of the cell associated to  $u$  is connected to an input pin of the cell associated to  $v$ . This means that edges contain additional information to specify the pins of the source and sink node they are connected to. Vertices have exactly one incoming edge per input pin. Nodes labeled as PI (PO) have no incoming (outcoming) edges.

To simulate the circuit, each PI may assume values from a given totally ordered finite set  $P = \{0, \dots, k-1\}$  where  $k$  denotes the number of elements of the logic. The complement (INV-gate) of a signal  $x$  is defined as  $\bar{x} = (k-1) - x$ . A LITERAL-gate  $(a, b)$  ( $a, b \in P, 0 \leq a \leq b < k$ ) has one input and one output<sup>2</sup>. For a given input  $x$  the behavior of such a gate is defined by:

$$f(x) = \begin{cases} k-1 & : a \leq x \leq b \\ 0 & : \text{otherwise} \end{cases}$$

It can then be proven that the set of gates defined above is functionally complete [10].

In a second step, we add memory elements that allow the storage of values from  $P$ . The corresponding inputs and outputs of these elements are denoted as *Secondary Inputs* (SIs) and *Secondary Outputs* (SOs) of the circuit. By adding these elements to the MVLNs as defined above we obtain *Sequential MVLNs* (SMVLNs). The resulting behavior can easily be modeled as a multi-valued FSM (analogously to the binary case).

We briefly review the main notations, that are important for the understanding in the following:

**Definition 1** A finite state machine  $M$  is defined as a quintuple  $M = (I, O, S, \delta, \lambda)$ , where  $I$  is the input set,  $O$  is the output set and  $S$  is the set of states,  $\delta : S \times I \rightarrow S$  is the next state function, and  $\lambda : S \times I \rightarrow O$  is the output function.

Since we consider a gate level realization of the FSM, we have  $I = P^n$ ,  $O = P^l$  and  $S = P^m$ .  $n$  denotes the number of primary inputs,  $l$  denotes the number of primary outputs and  $m$  denotes the number of memory elements. The functions  $\delta$  and  $\lambda$  are computed by a MVLN. We use the standard identification of SMVLNs and FSMs in the following, i.e.  $\delta$  describes the next state behavior and  $\lambda$  gives the output values.

In the following we do not distinguish between a SMVLN and its corresponding FSM.

### 3 Diagnostic Model

In this section we describe the diagnostic model to handle SMVLN. The model is generalization of the model recently introduced in [5] for two-valued circuits. (Notice that the generalization is not trivial and the use of MVL allows the handling of much larger problem instances.) To simplify the understanding we first briefly describe the main idea:

<sup>1</sup>In the binary case Min- and MAX-gates correspond to AND- and OR-gates, respectively.

<sup>2</sup>These LITERAL-gates are also called *window literals*.

In normal operation the SMVLN 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  $e$  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 evaluating test outcomes.

In the following we do not only consider one state  $s_j$ . Instead we consider a set of states  $s_l$  ( $l \in \{0, \dots, m\}$ ) 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. Furthermore, 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.

#### 3.1 Model Definition

Let  $M = (I, O, S, \delta, \lambda)$  be a FSM as defined above. (Notice that the FSM can be considered over Boolean values or multiple values.) Similarly to the basic model from [5] we have:

- The set of states  $S = \{s_0, s_1, \dots, s_m\}$ , 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;  $p = \{p_0, \dots, p_m\}$  is the set of *a priori* probabilities of the system being in state  $s_i$ .  $T = \{t_1, \dots, t_n\}$  is a finite set of available test vectors and  $c = \{c_1, \dots, c_n\}$  is the set of test costs measured in terms of length of the test e.g. determined by a test pattern generator.

The model for the general FSM is extended by:

- The set of possible outcomes of all tests  $T$  is  $R = \{r_1, \dots, r_L\}$ . (If the outcomes are used as an encoding over  $k$ -value logic, the maximal number of different outcomes is  $k^l$ , where  $l$  is the number of outputs.) The test matrix is  $D = [d_{ij}]$ , where  $d_{ij}$  is the outcome from set  $R$  of test  $t_j$  if the FSM is in the state  $s_i$ . (It is possible to extend the model to include test unreliabilities [1], that can interpret *Don't Cares*.)

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  $T$ .*

A test sequence that unambiguously identifies each state exists iff no two rows of the test matrix are identical and furthermore the number of tests (in the case of  $L$ -ary tests) is  $n > \log_L(m+1)$ .

## 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 MV AND/OR graphs. For the exact and heuristic solution we use algorithms from artificial intelligence.

The algorithms have been implemented for MV AND/OR graphs in a sequential diagnosis system for diagnosis on the system level (see also [1]). The user can choose among different algorithms that compute the 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 for large SMVLNs it is necessary to explore heuristic approaches for guiding the AND/OR graph search. The existing solution approaches can mainly be categorized into two different groups:

*Heuristic algorithms:* A class of sub-optimal algorithms provides a trade-off between optimality and computational complexity. They perform a local step-by-step optimization using problem-domain knowledge. The diagnostic tool described in this paper offers two such algorithms: *separation heuristic* and *information heuristic* [7].

*Exact 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 [1] corresponds to algorithm  $AO^*$  [11] with extension of multi-outcome tests. The algorithm  $AO^*$  is an ordered best-first search algorithm; it expands only the node of the search graph that offers the most promising way on the basis of two heuristics: (i) Huffman code-based heuristics ( $HEF_1$ ) and (ii) entropy-based heuristics ( $HEF_2$ ). Using these,  $AO^*$  is guaranteed to find an optimal solution. (For details see [11].) Experiments for randomly generated FSM in [5] showed that  $HEF_2$  is poor in comparison to  $HEF_1$ . The latter requires fewer nodes and backtracks to compute the optimal diagnostic tree. For this, we restrict ourselves to  $HEF_1$  for our experimental studies in the following.

The resulting test sequence can be described by a MV 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 correspond to the individual failure state with no ambiguity, and the average length of the tree represents the expected test cost.

Before we have a closer look at larger examples we explain the MV AND/OR graph construction for a small SMVLN in detail:

**Example 1** We consider a randomly generated FSM over 8-valued logic  $P = \{0, \dots, 7\}$ . We assume a set of eight states. A set of five tests is available for checking the FSM. For simplicity all test lengths equal one and

System states	Tests					Probabilities $p_i$
	$t_1$	$t_2$	$t_3$	$t_4$	$t_5$	
$s_0$	0	0	0	0	0	0.343
$s_1$	5	1	2	7	2	0.088
$s_2$	0	7	5	3	6	0.071
$s_3$	1	6	4	4	2	0.095
$s_4$	2	0	5	7	1	0.105
$s_5$	6	2	7	1	7	0.119
$s_6$	3	2	7	1	5	0.084
$s_7$	0	6	0	7	6	0.095

Table 1: Test matrix and fault probabilities

the output set  $O = P^1$ . Thus, we have eight different outcome for given tests. 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  $8 \times 5$  each test is represented by an outcome column vector. E.g. test  $t_5$  will have outcome 6 if the system is in the state  $s_2$  or  $s_6$ .

The optimal solution for the problem from Table 1 is obtained by applying the algorithm  $AO^*$  using  $HEF_1$ . The resulting minimal length  $J^*$  is given by:  $J^* = [p_3 + p_1] \cdot (c_5 + c_1) + [p_7 + p_2] \cdot (c_5 + c_2) + [p_0 + p_4 + p_6 + p_5] \cdot c_5 = 1.349$ .

When the diagnostic tree is used, first test  $t_5$  is applied. If the result is 1, the system is in  $s_4$ .

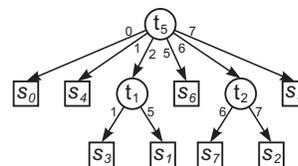


Figure 1: Optimal diagnostic tree for Example 1

## 5 Experimental Results

In this section we present experimental results for randomly generated FSMs. Similar as in [5], the test problems were generated using a uniform distribution. For simplicity all tests have cost one. (A different choice of these values gives similar results.) For all examples the number of tests was chosen around  $4 \times$  the minimal number of tests required (see Section 3.1). 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, minutes and hours.)

In [5] the experiments showed that exact algorithms using Boolean values can be reasonably applied only for small examples up to 100 states. We showed that the number of backtracks along with the number of generated nodes measures the quality of heuristic evaluation function used by the exact algorithms. The experiments reflected that ( $HEF_2$ ) is poor in comparison to ( $HEF_1$ ). Using heuristic algorithms we were able to consider FSMs with up to 1000 states.

In the following we study the behavior of the heuristic algorithm and exact algorithm, with respect to runtime and space complexity.

States	Outcomes	Algorithm	Nodes gen.	Backtracks	Cost	CPU time
100	16	HEF <sub>1</sub>	124	220	2.14	0.4
		PQ	136	0	2.38	0.1
250	16	HEF <sub>1</sub>	616	84	2.4423	2.9
		PQ	358	0	2.7735	0.5
500	16	HEF <sub>1</sub>	1025	297	2.4520	9.1
		PQ	697	0	2.7380	1.0
750	16	HEF <sub>1</sub>	2057	112	2.8966	15.5
		PQ	1115	0	3.2705	1.9
1000	32	HEF <sub>1</sub>	2400	1144	2.4921	21.4
		PQ	1351	0	2.7314	3.6
2500	32	HEF <sub>1</sub>	7090	225	2.8698	1 : 48.1
		PQ	3585	0	3.1517	13.1
5000	32	HEF <sub>1</sub>	12791	512	2.9812	4 : 47.9
		PQ	6971	0	3.3819	25.6
7500	32	HEF <sub>1</sub>	32546	2938	3.0090	16 : 39.1
		PQ	10119	0	3.4385	40.4
10000	64	HEF <sub>1</sub>	30877	512	2.8787	20 : 57.2
		PQ	13831	0	3.0752	1 : 54.9
15000	64	HEF <sub>1</sub>	41330	513	2.9574	31 : 57.4
		PQ	20384	0	3.1929	2 : 58.4
20000	64	HEF <sub>1</sub>	52784	577	2.9838	1 : 03 : 00.3
		PQ	26717	0	3.2704	4 : 51.6
25000	64	HEF <sub>1</sub>	55775	549	2.9838	1 : 16 : 23.5
		PQ	32476	0	3.2824	5 : 00.4
30000	64	HEF <sub>1</sub>	—	—	—	—
		PQ	38749	0	3.3317	6 : 43.9
40000	64	HEF <sub>1</sub>	—	—	—	—
		PQ	50478	0	3.3480	11 : 10.3
50000	64	HEF <sub>1</sub>	—	—	—	—
		PQ	63313	0	3.4004	21 : 13.7

Table 2: Performance of exact and heuristic algorithm

The results are shown in Table 2. A “—” symbolizes that the algorithm run out of space computing the complete diagnostic tree. We applied two different search algorithms presented in the previous section for each example. PQ and HEF<sub>1</sub> denote the information-heuristic algorithm and AO\* with Huffman code-based heuristics, respectively. The second column shows the maximal number of different outcomes of tests. The fourth column displays the number of nodes generated by the search algorithms. This directly reflects the space requirements of the two methods. As can be seen the heuristic algorithm uses around a factor of two less memory. Therefore it was possible to apply the heuristic method for FSMs with up to 50000 states for chosen number of outcomes, while AO\* with HEF<sub>1</sub> run out of space for FSMs larger than 25000 states.

The number of backtracks is given in the next column. The process of backtracking is performed only by exact algorithms and the consequences is also reflected in runtimes given in the last column. Note that the heuristic method took much less time to compute a solution. But the difference measured in the quality of the result of both methods is evident: The average cost of diagnostic tree computed by heuristic algorithm exceeded the outcome of the optimal average cost obtained by exact algorithms up to 15%. (The difference even increases if the examples are generated with larger number of tests.)

The diagnostic modeling presented in this paper with additional concept of multi-valued logic is considerably more profitable than two-valued, because the algorithms efficiently use maximal diagnostic resolution given by each test. Due to this, in comparison to the two-valued approach from [5], we can handle more realistic SMVLNs with a large number of states with respect to time and space complexity.

## 6 Conclusions

We presented a model for error diagnosis in sequential multi-valued logic networks modeled as FSMs. The model introduced in [5] has been extended by multi-valued outcomes tests. We used an AND/OR graph based method to determine a test with minimal cost by solving the test sequencing problem. Based on the MV encoding much larger problem instances could be handled, i.e. our experiments showed that over MV encodings solutions for FSMs with up to 50000 states can be computed, while the two-valued approach failed for more than 1000 states.

## References

- [1] A. Biasizzo, A. Žužek, and F. Novak. Sequential diagnosis tool. *Proc. Int'l Work. on Sys. Test & Diagnosis*, pp. 3–12, 1998.
- [2] X. Chen and M. Bushnell. *Efficient Branch and Bound Search with Application to Computer Aided Design*. Kluwer Academic Publisher, 1996.
- [3] K.W. Current and M.E. Hurlsten. A bi-directional current-mode cmos multiple-valued logic memory circuit. *Proc. Int'l Symp. on multi-valued Logic*, pp. 196–202, 1991.
- [4] R. Drechsler. Verification of multi-valued logic networks. *Proc. Int'l Symp. on multi-valued Logic*, pp. 10–15, 1996.
- [5] R. Drechsler and A. Žužek. Efficient functional diagnosis for synchronous sequential circuits based on AND/OR graphs. *Proc. Int'l Symposium on IC Technologies, Systems and Applications*, pp. 312–315, 1997.
- [6] D. Etiemble. On the performance of multivalued integrated circuits: Past, present and future. *Proc. Int'l Symp. on multi-valued Logic*, pp. 156–164, 1992.
- [7] E.J. Kletskey. An application of the information theory approach to failure diagnosis. *IRE Trans. on Reliability and Quality Control*, 9:29–39, 1960.
- [8] R. Drechsler W. Kunz, D. Stoffel, and A. Žužek. Decision diagrams and AND/OR graphs for design automation problems. *Proc. International Conference on Information, Communication & Signal Processing*, pp. 246–250, 1997.
- [9] K.L. McMillan. *Symbolic Model Checking*. Kluwer Academic Publisher, 1993.
- [10] J.C. Muzio and T.C. Wesselkamper. *Multiple-Valued Switching Theory*. Adam Hilger, Bristol and Boston, 1986.
- [11] 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.
- [12] A. Wahba and D. Borriore. A method for automatic design error location and correction in combinational logic circuits. *Jour. of Electronic Testing: Theory and Applications*, 8:113–127, 1996.