# The use of Model Checking in ATPG for sequential circuits

P. Camurati M. Gilli P. Prinetto M. Sonza Reorda

Dipartimento di Automatica e Informatica
Politecnico di Torino
Turin Italy

**Abstract**

Some design environments may prevent Design for Testability techniques from reducing testing to a combinational problem: ATPG for sequential devices remains a challenging field. Random and deterministic structure-oriented techniques are the state-of-the-art, but there is a growing interest in methods where the function implemented by the circuit is known. This paper shows how a test pattern may be generated while trying to disprove the equivalence of a good and a faulty machine. The algorithms are derived from Graph Theory and Model Checking. An example is analyzed to discuss the applicability and the cost of such an approach.

## 1 Introduction

Despite Design for Testability [WiPa82], test pattern generation for synchronous sequential circuits is still needed for some design environments, e.g., those based on Partial Scan [Moto90]. A major class of devices is represented by Finite State Machines (FSMs) that are in extensive use as building blocks in a variety of applications, ranging from VLSI devices to network controllers. As reported in [Wolf90], many ASICs are "*control-dominated*", i.e., they are best modeled in terms of FSMs, rather than in terms of data path and control part. Moreover, following methodologies such as "*Macro Testing*" [BEGP86], complex devices may be partitioned for testing purposes, creating macros composed by one or more FSMs that can be tested individually.

Researchers investigated different approaches to ATPG for synchronous sequential circuits. The extension of fault simulation techniques to test generation for sequential devices has been the object of research [ACAg89], but reduced CPU time and high fault coverage for very complex circuits still cannot be guaranteed. Other works focused on extended combinational deterministic algorithms: starting from Huffmann's model that transforms sequential behavior over time frames into combinational behavior on iterated structures, the basic method extends the D-algorithm [PuRo71]. Many enhancements to it have been proposed, limiting the number of copies [Fuji85], introducing a 9-valued algebra [Muth76], searching for a path to be sensitized [Marl86], [Chen88], exploiting the benefits of techniques used in combinational ATPG [ScAu89].

A common feature of most ATPGs is that they are *structure-oriented*, i.e., the only knowledge they have of the device is its topology. Some efforts have shown a growing interest on algorithms where a certain amount of reasoning is performed or the *function* implemented by the circuit is known. [HuSe89] describes an application of Temporal Logic [ReUr71]: its operators are used both to describe future behavior and to justify the past one and a reasoning process yields a test pattern. The knowledge of the state-transition table is used in [MDNS88] to justify backward the values needed to control an observable fault until the initial state and the primary inputs are reached. The need to know the function restricts the application domain to medium-sized circuits, since the function must be extracted from the structure. Whenever the same approach is embedded within a synthesis system, this limitation is overcome easily [ChJo90], since the state transition table is available immediately.

This paper presents a method to generate test patterns for synchronous sequential circuits working in fundamental mode [McCl86]. The devices are modeled as *Finite State Machines* (FSMs) and may be Mealy or Moore machines, indifferently. This method is based on a strong interaction between Graph Theory and Model Checking [BCDM86]. The basic hypothesis is that the function realized by the device, represented by its automaton, may be extracted from its structure. Once the fault, of the traditional single stuck-at type, is inserted, a faulty automaton is created. The fault detection condition is expressed in theoretical terms within the framework of the product machine [PoMC64] that might in principle be used, although experience shows that it becomes unmanageable for other than trivial circuits. Without any loss of information, it is possible to refer to a Deterministic Finite Automaton (DFA) that is considerably smaller. Other approaches refer to the Error Latency Model (ELM) state table [ShMC76], although their objective is not test pattern generation, rather error latency estimation. The DFA is a Moore Machine and it is a suitable input for the Model Checker of the MCB system [CESi86]. The use of the DFA overcomes the Model Checker's limitation to Moore FSMs, at least for testing applications. The test detection condition is easily stated as a CTL formula and the counterexample facility of MCB is used to compute the test pattern.

Section 2 describes the theoretical framework for FSM modeling, stating the test generation problem in terms of the product machine and of the DFA and showing where Model Checking intervenes. Section 3 illustrates these concepts by means of a simple example. Conclusions are drawn in Section 4, justifying the use of this approach, defining its applicability limits, and pointing to future work.

# 2 Theoretical framework

**Definition 1**

A Finite State Machine $\mathcal{M}$ is defined by a 6-tuple [Koha70]:

$$\mathcal{M} = (I, O, S, \delta, \lambda, s^0)$$

where:

$I$ is a finite, nonempty set of input values

$O$ is a finite, nonempty set of output values

$S$ is a finite, nonempty set of states

$\delta : I \times S \rightarrow S$ is the state transition function

$\lambda : I \times S \rightarrow O$ is the output function

$s^0$ is the initial state. □

A *Mealy* machine is such that $\lambda : I \times S \rightarrow O$.
A *Moore* machine is such that $\lambda : S \rightarrow O$.

**Definition 2**

Two completely specified FSMs $\mathcal{M}_1$ and $\mathcal{M}_2$ operating on the same input and output sets are *equivalent* iff, for all input sequences, all the elements of the output sequences are equal. □

The test generation problem for a FSM may be stated as follows: given a good machine $\mathcal{M}_1$ and a single stuck-at-i (i $\epsilon$ {0, 1}) fault $\mathcal{F}$, generate a faulty machine $\mathcal{M}_2$, operating on the same input, output, and state sets, where the fault $\mathcal{F}$ changes the $\delta$ and/or $\lambda$ functions. If one can demonstrate that the two FSMs are **not** equivalent, i.e., that there is an input sequence such that the $k$-th elements ($k \geq 1$) of the two output sequences differ, then a test pattern has been found. If equivalence can be proven, then fault $\mathcal{F}$ is undetectable.

Some ancillary definitions are necessary before stating the equivalence condition of two FSMs in Theorem 1.

**Definition 3**

Given two FSMs $\mathcal{M}_1 = (I, O, S_1, \delta_1, \lambda_1, s_1^0)$ and $\mathcal{M}_2 = (I, O, S_2, \delta_2, \lambda_2, s_2^0)$, the *product machine* $\mathcal{M}_{12}$ is defined as a 6-tuple:

$$\mathcal{M}_{12} = (I, O_{12}, S_{12}, \delta_{12}, \lambda_{12}, s_{12}^0)$$

where:

$O_{12} = O \times O$

$S_{12} = S_1 \times S_2$

$\delta_{12} : S_1 \times S_2 \times I \rightarrow S_1 \times S_2 : \quad \delta_{12}(s_1, s_2, i) = (\delta_1(s_1, i), \delta_2(s_2, i))$

$\lambda_{12} : S_1 \times S_2 \times I \rightarrow O \times O : \quad \lambda_{12}(s_1, s_2, i) = (\lambda_1(s_1, i), \lambda_2(s_2, i))$

where:

$s_1 \in S_1$; $s_2 \in S_2$; $i \in I$

$s_{12}^0 = (s_1^0, s_2^0)$ □

Given two machines $\mathcal{M}_1$ and $\mathcal{M}_2$, the state set $S_{12}$ of their product machine $\mathcal{M}_{12}$ may be partitioned into two subsets $Q_{12}$ and $Z_{12}$ such that $S_{12} = Q_{12} \cup Z_{12}$ and $Q_{12} \cap Z_{12} = \emptyset$. The two subsets are defined as:

$$Q_{12} = \{\, q_{12} = (q_1, q_2) \in S_{12} : \ \forall\, i \in I \quad \lambda_1(q_1, i) = \lambda_2(q_2, i) \,\}$$
$$Z_{12} = \{\, z_{12} = (z_1, z_2) \in S_{12} : \ \exists\, i \in I \quad \lambda_1(z_1, i) \neq \lambda_2(z_2, i) \,\}$$

The following theorem, whose proof is quite obvious and is thus omitted, expresses the equivalence condition for two FSMs.

**Theorem 1**

Two FSMs $\mathcal{M}_1$ and $\mathcal{M}_2$ are equivalent iff in the product machine $\mathcal{M}_{12}$ the subset $Z_{12}$ is either unreachable from the initial state $s^0_{12}$ or empty. □

Proving that the subset $Z_{12}$ is either unreachable from $s^0_{12}$ or empty is very heavy computationally. It is possible to reduce complexity, without affecting the validity of the method, by noting that only the states of $Q_{12}$ are needed to prove equivalence and that all the states of $Z_{12}$ can be collapsed into a single failure state, indicating that the FSMs are not equivalent. This allows to transform the product machine into a DFA.

**Definition 4**

A Deterministic Finite Automaton (DFA) is defined by a 5-tuple [Koha70]:

$$DFA = (I,\ S,\ \delta,\ s^0,\ F)$$

where:

$I$ is a finite, nonempty set of input values

$S$ is a finite, nonempty set of states

$\delta : I \times S \rightarrow S$ is the state transition function,

$s^0$ is the initial state

$F$ is the set of final states $f$. □

A DFA, associated to the product machine $\mathcal{M}_{12}$, whose final state indicates a failure in proving equivalence, is defined as follows:

**Definition 5**

Given two FSMs $\mathcal{M}_1$ and $\mathcal{M}_2$ and their product machine $\mathcal{M}_{12}$, if $s^0_{12} \in Q_{12}$, the DFA associated with $\mathcal{M}_{12}$ is the 5-tuple:

$$DFA(\mathcal{M}_{12}) = (I,\ Q_{12} \cup F' \cup f,\ \delta_{q12},\ s^0_{12},\ f)$$

where:

$f$ is the final failure state
$F' = \{ z_{12} \in Z_{12} : \exists i \in I \quad \delta_{12}(q_{12}, i) = z_{12}, \quad q_{12} \in Q_{12}\}$
$\delta_{q12}$:
if $(t \in Q_{12})$ then $\delta_{q12}(t,i) = \delta_{12}(t,i)$
if $(t \in F' \wedge (\lambda_{12}(t,i) = (o_1, o_2)) \wedge (o_1 \neq o_2))$
then $\delta_{q12}(t,i) = f$
else undefined. □

By using Definition 4 and Theorem 1, Theorem 2 may be obtained.

**Theorem 2**

Given two FSMs $\mathcal{M}_1$ and $\mathcal{M}_2$, representing the good and the faulty machines, respectively, every string of the language recognized by DFA($\mathcal{M}_{12}$) is a test pattern. □

Obviously, if the final state $f$ is not reachable, the two machines $\mathcal{M}_1$ and $\mathcal{M}_2$ are equivalent and therefore fault $\mathcal{F}$ is undetectable.

The construction of the DFA($\mathcal{M}_{12}$) associated to the product machine of the two FSMs leads to the determination of all the test patterns for the fault; sometimes it is sufficient to find only one pattern, possibly the shortest one. In order to do this we use MCB [CESi86]. MCB has as an input the graph of any Moore FSM and a formula of Branching Time Temporal Logic called CTL, describing the property to be verified. We must reduce the DFA of Definition 4 5 to a structure that can be accepted by MCB. The DFA is a Moore FSM in which the output associated with all the states except the final state is 0, while the output associated with the final state is 1. It is important to note that we reduce the problem of the equivalence of two Mealy Machines to that of a simple DFA, which is a trivial Moore Machine; thus we are able to use MCB in finding a test pattern for a generic FSM. The formula to be verified by the MCB asserts that the output of the DFA has to be different from 1. Since the logic is branching in time, the **AG** operator is used to assert that the formula is true for all possible evolutions in the future, i.e., for every path and at every node on the path.

The equivalence formula, if $O$ is the output of the DFA, is:

$$\mathbf{AG}(\tilde{} \ O).$$

# 3 An example

In this section we analyze the example of Fig. 1, originally presented in [MDNS88].

The functional description of the FSM expresses the inputs to the D-FFs in terms of the state variables $x_0\, x_1$ and of the input $i$:

$$\delta_1((x_0, x_1), i) = ((\bar{i} \cdot x_0) + i \cdot x_1), \overline{(i \oplus x_1)})$$

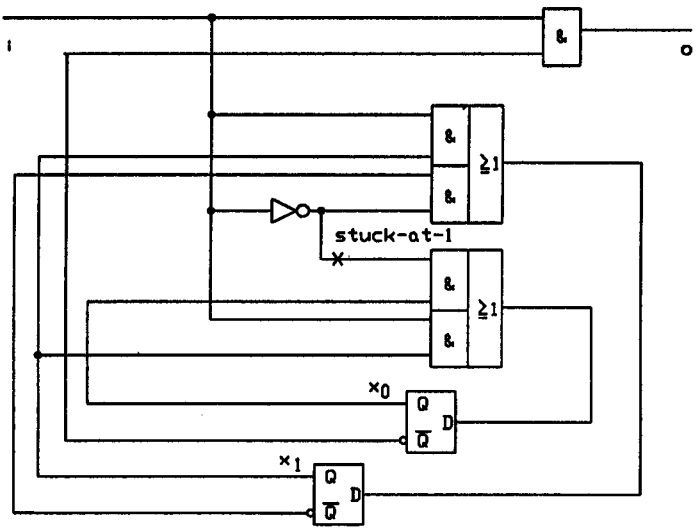


Figure 1: An example

The primary output depends on the state variables $x_0\, x_1$ and on the primary input $i$:

$$\lambda_1((x_0, x_1), i) = i \cdot \overline{x_0}$$

Fig. 2(a) shows the automaton $\mathcal{M}_1$ of the good machine. Let us consider the stuck-at-1 fault indicated in Fig. 1. The functions corresponding to the faulty FSM are:

$$\delta_2((x_0, x_1), i) = (x_0 + i \cdot x_1,\ \overline{i \oplus x_1})$$
$$\lambda_2((x_0, x_1), i) = i \cdot \overline{x_0}$$

In this particular case the fault's effect is limited to the $\delta$ function. The automaton $\mathcal{M}_2$ of the faulty FSM is shown in Fig. 2(b).

The DFA($\mathcal{M}_{12}$) of Fig. 3 is easily built according to Definition 5.

Only 6 states of this automaton are reachable from the initial state; the remaining 4 are unreachable and are not useful to determine test patterns. Two of the 6 states ($F_1'$ and $f$) are only introduced to formally define the DFA, thus only 4 states are really needed, compared with 16 states for the product machine. The shortest testing sequence is 01011.

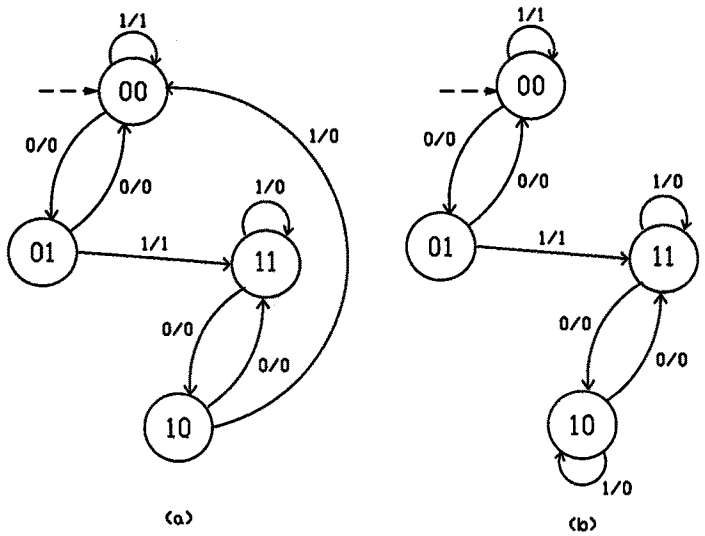


Figure 2: The good and faulty automata for the example of Fig. 1

If we use MCB with Moore FSM corresponding to the DFA of Fig. 3 as an input, the CTL formula to be disproved is:

$$\mathbf{AG}\ (\tilde{}\ O).$$

The MCB finds that the formula is false and the counterexample sequence coincides with the test pattern.

# 4 Conclusions

This paper presented an approach to ATPG for synchronous circuits modeled as Moore or Mealy FSMs. The graph representing state and output transitions is first extracted from the structural description of the circuit, then the effect of the single stuck-at fault is injected, resulting in a faulty graph. The test pattern generation algorithm is based on techniques used to prove the equivalence of FSMs by considering the product machine as a mere conceptual framework. Transforming the product machine's graph into a DFA without unreachable states reduces memory requirements. The advantage of the DFA with respect to the product machine resides in the drastic reduction of the number of states, which in the best case are almost linear in the number of states of the good machine. The Model Checking algorithm is used to disprove, when possible, a CTL formula and MCB's counterexample facility returns the test pattern. A fault

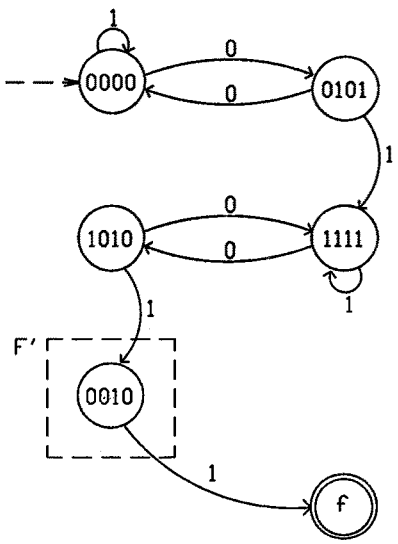


Figure 3: The DFA

simulator [CGSR89] serves the purpose of fault dropping. Preliminary experimental data, on an international benchmark set [BBKo89] (Tab. 1) indicate that the good and faulty machine's graphs are considerably smaller than their upper bound and that the CPU time to create them is reasonable. The main limit is now imposed by MCB, which is able to deal only with a reduced number of states. Results on fault coverage are encouraging: on the s298 circuit 100% coverage for detectable faults is reached.

The novelty of our research resides in modeling both the good and the faulty circuit as graphs, so that the fault detection condition can easily be stated. The use of the DFA and efficient graph simplification algorithms make this approach more efficient. Although the current version refers to the single stuck-at, the algorithm is independent of the fault model, since the latter intervenes only during the generation of the faulty graph. It is thus possible to investigate the use of other models, namely functional ones, whose goodness may be assessed giving a figure of merit in terms of single stuck-at fault coverage.

The programs to extract graphs, to build and reduce the DFA, the combinational ATPG, and the fault simulator have been implemented in separate packages, amounting to a total of 4000 C-code lines and together with MCB, they run on a SUN Sparcstation-1.

This work has been partially supported by the EEC under contract ESPRIT BRA

| Circuit | Good Machine | | | Faulty Machines (reachable states) | | | |
|---|---|---|---|---|---|---|---|
| | total states | reachable states | CPU time (s) | avg | max | min | CPU time (s) |
| s27 | 8 | 6 | 0.05 | 4 | 7 | 3 | 0.05 |
| s208 | 256 | 17 | 7.81 | 17 | 256 | 1 | 12.81 |
| s298 | 16384 | 218 | 0.82 | 189 | 347 | 2 | 0.96 |
| s344 | 32768 | 2625 | 400.17 | 3216 | 19693 | 11 | 775 |
| s349 | 32768 | 2625 | 405.81 | 3171 | 19693 | 11 | 768 |
| s382 | 2M | 8865 | 64.61 | 5999 | 13858 | 2 | 52.01 |
| s386 | 64 | 13 | 1.37 | 11 | 22 | 2 | 1.51 |
| s400 | 2M | 8865 | 66.56 | 6027 | 15790 | 2 | 53.54 |
| s444 | 2M | 8865 | 42.70 | 5929 | 15790 | 2 | 38.71 |
| s526 | 2M | 8868 | 38.07 | 6884 | 14580 | 2 | 41.45 |
| s526n | 2M | 8868 | 37.47 | 6879 | 14580 | 2 | 41.22 |

Table 1: Graphs statistics

3216 "CHARME" and by the Italian National Research Council "Progetto Strategico Collaudo".

# References

[ACAg89] V.D. Agrawal, K.T. Cheng, P. Agrawal: "A directed search method for test generation using a concurrent fault simulator," IEEE Transactions on Computer-Aided Design, Vol. 8, n. 2, February 1989, pp. 131-138

[BBKo89] F. Brglez, D. Bryan, K. Koźmiński: "Combinational profiles of sequential benchmark circuits," ISCAS'89: IEEE International Symposium on Circuits And Systems, Portland, OR (USA), May 1989, pp. 1929-1934

[BCDM86] M. Browne, E.M. Clarke, D. Dill, B. Mishra: "Automatic verification of sequential circuits using temporal logic," IEEE Transactions on Computers, Vol. C-35, n. 12, December 1986, pp. 1035-1044

[BEGP86] F.P.M. Beenker, K.J.E. van Eerdewijk, R.B.W. Geritzen, F.F. Peacock, M. van der Star: "Macro Testing: Unifying IC and Board Test," IEEE Design & Test of Computers, December 1986, pp. 26-32

[CESi86] E.M. Clarke, E.A. Emerson, A.P. Sistla: "Automatic verification of finite-state concurrent systems using temporal logic specifications," ACM Transactions on Programming Languages and Systems, Vol. 8, n. 2, April 1986, pp. 244-263

[CGSR89] G. Cabodi, S. Gai, M. Sonza Reorda: "Partitioning Techniques in Multiprocessor Simulators," ESM-89: European Simulation Multiconference, Rome (Italy), June 1989, pp. 311-317

[Chen88] W.T. Cheng: "The Back Algorithm for sequential test generation,"

ICCD'88: IEEE International Conference on Computer Design, Rye Brook, NY (USA), October 1988, pp. 66-69

[ChJo90] K-T. Cheng, J-Y. Jou: "Functional test generation for Finite State Machines," ITC'90: International Test Conference 1990, Washington, DC (USA), September 1990, pp. 162-168

[Fuji85] H. Fujiwara: "Logic testing and design for testability," The MIT Press, Cambridge, MA (USA), 1975

[HuSe89] R.V. Hudli, S.C. Seth: "Temporal Logic based test generation for sequential circuits," IFIP TC 10/WG 10.2 Working Conference on CAD systems using AI techniques, Tokyo (Japan), June 1989, pp. 91-98

[Koha70] Z. Kohavi: "Switching and finite automata theory," Computer Science Series, Mc Graw Hill, New York, NY (USA), 1970

[Marl86] R. Marlett: "An effective test generation system for sequential circuits," DAC-23: 23th IEEE/ACM Design Automation Conference, Las Vegas, NV (USA), June 1986, pp. 250-256

[McCl86] E.J. McCluskey: "Logic Design Principles with Emphasis on Testable Semicustom Circuits," Prentice-Hall, Englewood Cliffs, NJ (USA), 1986

[MDNS88] H.K.T. Ma, S. Devadas, A.R. Newton, A. Sangiovanni-Vincentelli: "Test generation for sequential circuits," IEEE Transactions on Computer-Aided Design, Vol. 7, n. 10, October 1988, pp. 1081-1093

[Moto90] A. Motohara: "Design for Testability of ASICs in Japan," IEEE 13th Annual Workshop on Design for Testability, Vail, CO (USA), April 1990, *(Oral presentation; no proceedings available)*

[Muth76] P. Muth: "A nine-valued circuits model for test generation," IEEE Transactions on Computers, Vol. C-25, n. 6, June 1976, pp. 630-636

[PoMC64] J.F. Poage, E.J. McCluskey: "Derivation of optimum test sequences for sequential machines," 5th Annual Symposium on Switching Theory and Logical Design, 1964

[PuRo71] G.R. Putzolu, J.P. Roth: "A heuristic algorithm for the testing of asynchronous circuits," IEEE Transactions on Computers, Vol. C-20, n. 6, June 1971, pp. 639-647

[ReUr71] N. Rescher, A. Urquart: "Temporal Logic," Springer-Verlag Library of Exact Philosophy N. 3, Springer-Verlag, Berlin (FRG), 1971

[ScAu89] M.H. Schulz, E. Auth: "ESSENTIAL: an efficient self-learning test pattern generation algorithm for sequential circuits," ITC'89: International Test Conference 1989, Washington, DC (USA), September 1989, pp. 28-37

[ShMC76] J.J. Shedletsky, E.J. McCluskey: "The error latency of a fault in a sequential digital circuit," IEEE Transactions on Computers, Vol. C-25, n. 6, June 1976, pp. 655-659

[WiPa82] T.W. Williams, K.P. Parker: "Design for Testability - a survey," IEEE Transactions on Computers, Vol. C-31, n. 1, January 1982, pp. 2-15

[Wolf90] W. Wolf: "The FSM network model for behavioral synthesis of control-dominated machines," DAC-27: 27th IEEE/ACM Design Automation Conference, Orlando, FL (USA), June 1990