Concurrent error detection in FSMs using transition checking technique

Goran Lj. Djordjevic[1], Tatjana R. Stankovic[1], Mile K. Stojcev[1]

Abstract – **As a complexity of VLSI ICs increases, the inclusion of concurent error detection is becoming a major concern. In this paper, we address the problem of concurrent error detection in finite state machines (FSMs). In particular, we propose an efficient technique for partial self-checking FSM design based on on-line monitoring of FSM state transitions. We focus primarily on describing an adopted behavioral error model and analytical analysis of the proposed technique. Results concerning error coverage, obtained for benchmark FSMs, are reported.**

Keywords – **concurrent error detection, partial self-checking, finite-state machine**.

I. INTRODUCTION

Contemporary VLSI technology has evolved to a level where large systems, previously implemented as printed circuit boards with discrete components, are integrated into a single IC. One of the main side-effects of shrinking device sizes, decreasing supply voltages, and high-speed operation of current IC technologies, is increasing sensitivity to temporary faults caused by noise. In order to deal with these problems, the adoption of self-checking approaches as powerful means for concurrent error detection of both temporary and permanent faults can be a viable solution [1,2]. A self-checking circuit is essentially based on a functional unit (FU) and a checker that continuously verifies the correctness of the FU operation. This allows, detection of errors as soon as they occur, thus avoiding their propagation through the whole system.

From the reliability point of view, the control unit, typically realized in a form of FSM, is usually the most critical part of the system. Classical approaches for designing self-checking FSMs are based on either duplication, or application of specific error detecting codes. Latter techniques provide error detection for the FSM's next-state logic, by encoding the states with a special property. In [3], the FSM states are encoded using an *m*-out-of-*n* code. FSM encoding employed in [4] is achieved providing a state assignment in which each state is at a constant Hamming distance from its possible next states. In most cases, these approaches require a hardware overhead of more than 100 percent. Several approaches have been taken in the past to design self-checking sequential circuits with lower area overhead in respect to duplication. In [5], a monitoring machine is used in order to monitor the states of the original machine. This technique is effective for delay fault model. However, for the stuck-at fault model, the monitoring machine resluts in high area overhead. Work presented in [6] uses a control flow checking technique for decetion of illegal state transitions, that is based on path signatures. The proposal has low hardware overhead, buh this technique has uncertain error detection latency and low fault coverage.

In this paper, we propose an efficient method for designing partially self-checking FSM, based on on-line monitoring of FSM's state transitions. We analyze the error-detecting capability of the technique by taking into account probabilistic behavior of the FSM, and redundancy involved through increasing the number of state code bits. This analysis shows that the proposed method achieves a very high error-coverage with modest number increasing of state code bits.

The paper is organized as follows: in Section 2 we firstly introduce some basic concepts about FSMs, and then describe error model that serves as a basis for transition checking technique presented in Section 3. In Section 4 we perform analytical study of the technique. Results evaluating the proposed technique on a subset of MCNC 91 benchmark FSMs are presented in Section 5. Conclusions are given in Section 6.

II. SYSTEM MODEL

A. FSM

An FSM is defined as 6-tuple $M = (X, Y, S, f, g, s_0)$, where X, Y, and S are finite sets of inputs, outputs, and states, respectively, and s_0 is the initial (reset) state of S. The unique mapping: $f : X \times S \to S$, and $g : X \times S \to Y$ defines the so-called *state transition function* and *output function*. The *size* of M is the cardinality of set S and it is denoted as $|S|$. An FSM with $n = |S|$ states can be described by a *state transition graph* (STG) defined by a vertex (state) set $S = \{s_1, ..., s_n\}$ and related directed edge set T representing set of transitions from one state to another: $T = \{(s_i, s_j) \mid \exists x \in X, f(x, s_i) = s_j\}$. The edges of the STG are labeled with the corresponding inputs and output values. An example FSM is shown in Fig. 1(a).

A state encoding is one-to-one mapping from S to B^c, a function $E : S \to B^c$, where c is the number of state variables. By specifying the state encoding function E, we associate each symbolic state to a particular vector of c bits, i.e. to a vertex in the c-dimensional Boolean space. Notice that we have numerous degrees of freedom in the choice of E. The only constraint for E is that B^c has enough vertices to assign a different one to each symbolic state, thus $c \geq c_{min} = \lceil \log_2 |S| \rceil$.

[1]Goran Lj. Djordjevic, Mile K. Stojcev, and Tatjana R. Stankovic are with the Faculty of Electronic Engineering, Aleksandra Medvedeva 14, 18000 Nis, Serbia and Montenegro.

E-mail: {gdjordj,stojcev,tatjanas}@elfak.ni.ac.yu

Once we have specified c and E, the state of a FSM is completely expressed by c binary variables called *state code*.

Let us assume that each state of the FSM, having n states, is encoded with c number of bits – that is, the machine is implemented using c number of flip-flops. The flip-flops may generate 2^c states, out of which only n correspond to the *valid* states of the FSM. The remaining $2^c - n$ codes are kept unused. These unused state codes are referred to as the *invalid*, or *unreachable* states of the FSM, since the circuit will never be in any of these states during normal, i.e. fault-free operation. Figure 1(b) shows STG from Fig. 1(a) encoded with 2 bits. In this STG there is one invalid state, labeled as v_0 that corresponds to unused bit vector 11. Similarly, we distinguish between valid and invalid transitions. All edges in the STG defined by state transition function f of the FSM are said to be *valid*. On the other hand, any other possible transitions either between valid states or between valid and invalid state is referred to as *invalid transition*.

B. Error Model

We assume a behavioral error model, i.e. an error model that is not defined through permanent or transient physical faults in the hardware, but rather in terms of erroneous behavior of the FSM that such faults induce. In particular, given input x and present state s, state transition function f defines unique error-free transition $(s, t) = (s, f(x, s))$ and a set of all possible erroneous transition $ET = \{(s, t) \mid t \neq f(x, s)\}$. Similarly, output function g defines unique error-free output value $y = g(x, s)$, and a set of all possible erroneous output values $EO = \{y \mid y \neq g(x, s)\}$. Occurrence of either an erroneous transition or an erroneous output value represents erroneous behavior indicating presence of a fault in the next state or output logic of the FSM. Notice that in this work we restrict our attention to the faults in the next state logic, only. Concretely, we are concerned with transition errors, only, that can be described with changes in the STG executed by the FSM.

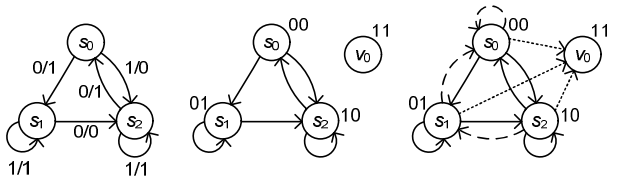


Fig. 1. STG and its corresponding extended STG: (a) original STG; (b) encoded STG; (c) extended STG.

In order to describe better the adopted error model, we introduce an *extended STG*. This graph is formed by adding invalid states and invalid transitions to the original STG. An example of extended STG, relative to the original STG from Fig. 1(a), is shown in Fig. 1(c). Additionally, in this model, we distinguish between inner and outer invalid transitions. An invalid transition between two valid states is referred to as *inner* (sketched with dashed lines in Fig. 1(c)), while one between valid and invalid state is said to be *outer* invalid transition (sketched with doted lines in Fig. 1(c)). Thus, for given STG with state set S with n states and transition set T, and c-bit state encoding E, the extended STG is defined by vertex set $S \cup V$, where V is the set of 2^c-n invalid states, and transition set $T \cup I \cup O$, where I is the set of invalid inner transitions: $I = \{(s_i, s_j) \mid s_i, s_j \in S \wedge (s_i, s_j) \notin T\}$, and O is the set of invalid outer transitions: $O = \{(s_i, v_j) \mid s_i \in S \wedge v_j \in V\}$. According to this model, the presence of fault is indicated by occurrence of either an inner or an outer invalid transition.

Notice that the adopted error model is not complete since it is unable to express all possible erroneous transitions. Namely, in this model, any erroneous but valid transition is considered to be error-free. Consequently any self-checking technique that relies upon this model is partial by definition. However, with this model, we can develop a methodology realizing FSM checking with a certain level of abstraction from the physical implementation.

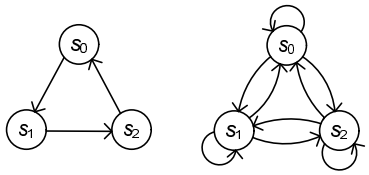


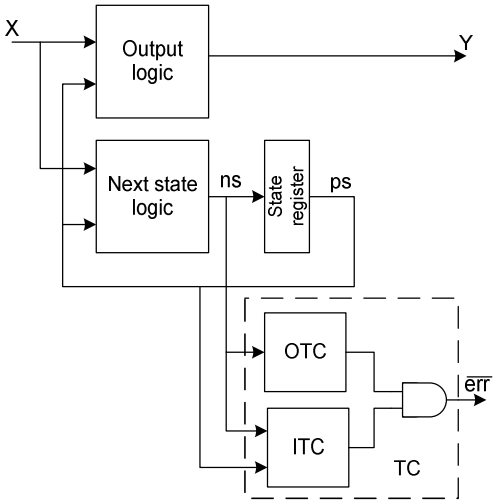
Fig. 2. Best- and worst-case examples.

Figure 2 shows two extreme cases of how the connectivity of the STG can affect the error modeling capability of the extended STG. Counter-like FSM given on the left hand side of Fig. 2, represents one extreme where all erroneous transition are captured by the corresponding extended STG. The completely connected STG shown on the right in Fig. 2 represents the other extreme where the adopted error model is useless, since all possible erroneous transitions are at the same time valid transitions. Majority of practical state machines lay between these two extremes.

II. TRANSITION CHECKING

The proposed transition checking scheme is illustrated in Fig. 3(a). Transition checker (TC) is combinational circuit that monitors transitions of the FSM (pairs of present-next state codes) and it asserts the error signal since an invalid transition is observed. In our proposal, two kinds of invalid transitions are detected by two separate modules of the TC: Inner transition checker (ITC) and Outer transition checker (OTC). The ITC is concerned with transitions between valid states, only. Its truth table is given in Fig. 3(b). The ITC outputs logic 1, that corresponds to error-free condition, if both the present state, *ps*, and the next state, *ns*, are valid states, and the pair (*ps*, *ns*) is a valid transition. The ITC indicates an erroneous condition, i.e., it outputs logic 0, if both *ps* and *ns* are valid states, but the pair (*ps, ns*) is not a valid transitions. The output left unspecified for the cases when at least one of states, *ps* or *ns*, is invalid. On the other hand, the OTC monitors the next state, only. It outputs logic 1 if the state is valid, otherwise logic 0 (Fig. 3(c)). In conjunction, ITC and OTC are capable of detecting all invalid transition of the FSM. Note that neither ITC nor OTC consider the cases when the present state is invalid, since such a situation can occure iff the previous transition was invalid, and thus detected

during the previous clock cycle. This incompleteness in the specification represents a degree of freedom that can be spent during logic optimization.



(a)

ITC	
Input condition	Output
$ps, ns \in S \wedge (ps, ns) \in T$	1
$ps, ns \in S \wedge (ps, ns) \notin T$	0
$ps \in V \vee ns \in V$	X

(b)

OTC	
Input condition	Output
$ns \in S$	1
$ns \in V$	0

(c)

Fig. 3. Transition checking scheme: (a) blok diagram; (b) truth table for ITC; (c) truth table for OTC.

III. ANALYTICAL MODEL

In this section we present an analytical model and metric for quantifying the error-detecting potential of the transition checking technique.

State error coverage in state s_i, cv_i, is defined as conditional probability that error signal is asserted by the TC, if an erroneous transition has been occurred while FSM was in state s_i. Assuming that all possible erroneous transitions are equally probable, the state error coverage is determined by the ratio of the number of invalid transitions exiting state s_i to the total number of erroneous transitions from state s_i. Let d_i denotes the outdegree of the state s_i, i.e. the number of valid transitions exiting this state. Given c-bit encoding, the number of possible erroneous transitions from any state is equal 2^c-1. Since there is 2^c-n outer, and n-d_i inner invalid transitions, the state error coverage in s_i can be expressed as:

$$cv_i = cv_i^{OUT} + cv_i^{IN} \tag{1}$$

where $cv_i^{OUT} = \frac{2^c - n}{2^c - 1}$ is the outer, and $cv_i^{IN} = \frac{n - d_i}{2^c - 1}$ is the inner error coverage in state s_i.

Notice that states error coverage is not the same for all states in the STG, since cv_i^{IN} depends on outdegree of the state. Also, during FSM operation, not all states are visited with equal probability. Thus, a state that appears more often will have greater impact on overall error coverage. Having this in mind, we define the *total error coverage* as weighted-sum of state error coverage:

$$CV = \sum_{i=1}^{n} p_i cv_i \tag{2}$$

where p_i stands for steady-state occupation probability of state s_i. By substituting right hand side of Eq. (1) for cv_i in Eq. (2) we obtain:

$$CV = \sum_{i=1}^{n} p_i cv_i^{OUT} + \sum_{i=1}^{n} p_i cv_i^{IN} \tag{3}$$

The first term in Eq. (3) represents the total outer error coverage

$$CV^{OUT} = \sum_{i=1}^{n} p_i cv_i^{OUT} = \frac{2^c - n}{2^c - 1} \tag{4}$$

while the second term is the total inner coverage

$$CV^{IN} = \sum_{i=1}^{n} p_i cv_i^{IN} = \frac{n - \sum_{i=0}^{N} p_i d_i}{2^c - 1} \tag{5}$$

Thus,

$$CV = CV^{OUT} + CV^{IN} = \frac{2^c - \sum_{i=0}^{N} p_i d_i}{2^c - 1} \tag{6}$$

The summation term in Eq. (6) accounts for combined effects of stochastical behavior of the FSM and transition density. The transition checking technique is more effective for FSM that most of the time has small number of alternative next-states. Also, from Eq. (6) it is evident that the error coverage increases by increasing the number of bits used for state encoding. For sufficeintly large c, the CV tends to 1 (100%). This result is expected, since with increase in c, the rapidly growing number of invalid states raises the probability that once an erroneous transition occurs it will be invalid transition and thus be detected by the transition checker. It is interesting to note that increase of c has opposite effect on outer and inner error coverage: while the outer error coverage increases, the inner error coverage decreases. Thus, for smaller c the error coverage is mainly determined by the ITC, while for larger c the OTC has a dominant role. Typical relationship is shown in Fig. 4.

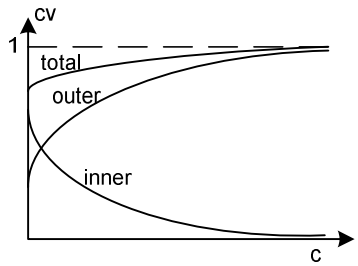


Fig. 4. Relationship between error coverage and number of state bits.

IV. RESULTS

We have analyzed several typical FSM circuits taken from the MCNC 91 sequential benchmark suite. First, the selected FSMs have been minimized by means of the program *stamina*

TABLE I

ERROR COVERAGE ANALYSIS

FSM				Coverage (outer ; inner)				
name	**n**	**e**	**cmin**	**cmin**	**cmin+1**	**cmin+2**	**cmin+3**	**cmin+4**
bbsse	16	42	4	**0.824** (0.2; 0.624)	**0.915** (0.613; 0.302)	**0.958** (0.809; 0.148)	**0.979** (0.905; 0.074)	**0.989** (0.953; 0.037)
beecount	7	23	3	**0.375** (0; 0.375)	**0.732** (0.571; 0.160)	**0.875** (0.8; 0.075)	**0.939** (0.903; 0.036)	**0.970** (0.952; 0.018)
cse	16	55	4	**0.749** (0; 0.749)	**0.878** (0.516; 0.362)	**0.940** (0.762; 0.178)	**0.970** (0.882; 0.088)	**0.985** (0.941; 0.044)
dk14	7	27	3	**0.622** (0.143; 0.479)	**0.824** (0.6; 0.224)	**0.915** (0.806;0.108)	**0.958** (0.908; 0.053)	**0.979** (0.953; 0.026)
dk16	27	102	5	**0.906** (0.161; 0.745)	**0.953** (0.587; 0.366)	**0.977** (0.795;0.182)	**0.988** (0.898; 0.09)	**0.994** (0.949; 0.04)
ex1	20	73	5	**0.933** (0.451; 0.481)	**0.967** (0.73; 0.233)	**0.984** (0.866;0.117)	**0.992** (0.933; 0.058)	**0.996** (0.967; 0.029)
keyb	19	46	5	**0.917** (0.419; 0.497)	**0.959** (0.714; 0.245)	**0.979** (0.858; 0.121)	**0.989** (0.929; 0.06)	**0.995** (0.964; 0.03)
mark1	15	36	4	**0.889** (0.267; 0.622)	**0.946** (0.645; 0.301)	**0.973** (0.825;0.148)	**0.987** (0.913; 0.073)	**0.993** (0.957; 0.036)
planet	48	71	6	**0.989** (0.245; 0.735)	**0.995** (0.630; 0.365)	**0.997** (0.815;0.181)	**0.998** (0.908; 0.09)	**0.999** (0.954; 0.045)
s1	20	80	5	**0.887** (0.387; 0.5)	**0.944** (0.698; 0.246)	**0.97** (0.85; 0.122)	**0.986** (0.925; 0.06)	**0.993** (0.963; 0.03)
s510	47	75	6	**0.968** (0.269; 0.698)	**0.974** (0.637; 0.346)	**0.992** (0.819; 0.172)	**0.996** (0.909; 0.086)	**0.998** (0.955; 0.043)
sand	32	90	5	**0.916** (0; 0.916)	**0.959** (0.508; 0.451)	**0.979** (0.759; 0.223)	**0.989** (0.878; 0.111)	**0.994** (0.939; 0.05)

(that is included in SIS [8]). Then, for each FSM, we have calculated the steady-state occupation probabilities through Markovian analysis of the STG (as done in [7]). Finally, we have applied the Eq. (6) for several values of c starting with minimal number of state bits, c_{min}. Table I reports the data for the selected examples. Columns 2, 3 and 4 give the number of states (n), the number of transitions (e), and the minimal number of state bits (c_{min}), respectively. Next five columns give error coverage values calculated for $c=c_{min} + i$, i=1, …, 4, number of state code bits. We use boldface to indicate the total error coverage (calculated according to Eq. (6)). Outer and inner error coverage (calculated according to Eq. (4) and (5)) are given in brackets separated by semicolon.

As can be seen from Table I, error coverage values follow the general relationship sketched in Fig. 4. The obtained results show that with modest increase in c, the error coverage reaches relatively high level. In particular, in 10 out of 12 cases, redundancy of two state bits is sufficient to reach the error coverage of 90%. With addition of three state bits, the error coverage excides 95%. The technique is particularly effective for FSMs with low ratio between the number of transitions, e, and the number of states n, such as, for example, circuits “planet”, “s510”, and “ex1”.

V. CONCLUSIONS

A method for the synthesis of partially self-checking FSMs with low cost concurent error detection scheme is proposed. The method is based on transition monitoring. An analitycal approach that relates to error coverage is given. We have shown that by addition of few state code bits, the proposal provides high error coverage. The results given in this paper are preliminary. Further analysis of the hardware overhead incurred by the proposed technique is required. We expect that transition checking approach could provide a basis for an efficient automatic synthesis of partial self-checking FSM.

REFERENCES

[1] P.K. Lala, *Self-Checking and Fault-Tolerant Digital Design, Morgan Kaufmann Publishers*, San Francisco, 2001.

[2] G. Lj. Djordjevic, M. K. Stojcev and T. R. Stankovic, “Approach to partially self-checking combinational circuits design”, *Microelectronics Journal*, Vol. 35 (12) , Dec. 2004, pp. 945-952.

[3] N. K. Jha and S.-J. Wang, “Design and Synthesis of Self-checking VLSI Circuits”, *IEEE Trans. On CAD of Integrated Circuits and Systems,* vol. 12, No. 6, pp. 879-887, June 1993.

[4] C. Bolchini, R. Montandon, F. Salice, D. Sciuto, "Design of VHDL-based totally self-checking finite-state machine and data-path descriptions", *IEEE trans. on VLSI Systems*, Vol. 8, no. 1, Feb. 2000, pp. 98-103.

[5] R. A. Parekhji, G. Venkatesh and S.D. Sherlekar, "Concurrent Error Detection using Monitoring Machines," *IEEE design & Test of Computers,* Vol. 12, pp.24-32, Fall 1995.

[6] R. Leveugle, and G. Saucier “Optimized Synthesis of Concurrently Checked Controller,” *IEEE Trans. Computers,* Vol. 39, pp. 419-425, Apr.1990.

[7] L. Benini and G. De Micheli, “Optimal Synthesis of Gated-Clocks for Low-Power Finite-state Machines,” *International Logic Synthesis Workshop*, 1995.

[8] E. M. Sentovich et al., "SIS: A System for Sequential Circuit Synthesis", Tech. Report UCB/ERL M92/41, Electr. Research Lab, Univ. of California, Berkeley, CA 94720, May 1992.