Non-Intrusive Concurrent Error Detection in FSMs through State/Output Compaction and Monitoring via Parity Trees

Petros Drineas and Yiorgos Makris
Departments of CS & EE, Yale University
{petros.drineas, yiorgos.makris}@yale.edu

1. Abstract We discuss a non-intrusive methodology for concurrent error detection in FSMs. The proposed method is based on compaction and monitoring of the state/output bits of an FSM via parity trees. While errors may affect more than one state/output bit, not all combinations of state/output bits constitute potential erroneous cases for a given fault model. Therefore, it is possible to compact them without loss of error information. Thus, concurrent error detection is performed through hardware that predicts the values of the compacted state/output bits and compares them to the actual values of the FSM. In order to minimize the incurred hardware overhead, a randomized algorithm is proposed for selecting the minimum number of required parity functions.

2. Problem Statement Consider the FSM with p inputs, n outputs, and k state bits shown in Fig. 1. For every combination of input IN and previous state PS, any error caused by a fault f in the stuck-at fault model will manifest itself as a difference between the error-free response $GM(IN, PS)$ and the erroneous response $BM_f(IN, PS)$. This difference is detectable in a non-empty set of the state and/or output bits $S_1 \dots S_{n+k}$; each such set is defined as an *Erroneous Case*, $EC(IN, PS, f)$. Clearly, several combinations of transition (IN, PS) and fault f may lead to the same erroneous case, i.e. the same set of bits through which the effect of fault f on transition (IN, PS) may be detected. The union of all erroneous cases may be represented in the table format of Fig. 2, where columns correspond to state/output bits, rows correspond to erroneous cases, and entries in the table indicate the state/output bits at which each erroneous case is detected.

Detecting all circuit errors requires that at least one state/output bit for each erroneous case be predicted through additional hardware and compared to the actual value. To minimize the incurred hardware overhead, we seek to minimize the number of predicted bits. Unfortunately, since stuck-at faults on a state/output bit may only be detected on this bit, it is likely that all state/output bits will be included in the solution, leading to duplication. To overcome this limitation, our method employs state/output compaction via parity trees. The *key observation* is that the parity (XOR) function of several state/output bits, an odd number of which detects an erroneous case, also detects the erroneous case. Therefore, it is possible that a small number of parity functions will be adequate to cover all erroneous cases. However, a large number of alternative parity functions exist, exponential to the number of state/output bits. Expanding the error detectability table to explicitly incorporate these functions and solving a minimum cover problem on the expanded table is infeasible. Therefore, selecting the minimum number of parity functions that will cover all erroneous cases poses an interesting problem; we present a novel algorithm based on linear programming and randomized rounding [1].

The proposed methodology is straightforward, as depicted in the form of a block diagram in Fig. 3. Given an FSM with p inputs, n outputs, and k state bits, XOR trees are employed to implement the ℓ parity functions required for lossless state/output bit compaction. Combinational logic is employed to predict the values of the ℓ bits that compact the $k + n$ state/output bits for each FSM transition, and a comparator is employed to identify any discrepancy. Registers are added to hold the output and the predicted values so that comparison is performed one clock cycle later in order to also detect faults in the State Register. Thus, all FSM errors are detected with latency of one clock cycle.

The aforementioned problem – in various forms and contexts – has been extensively studied in the literature [2, 3, 4, 5, 6, 7, 8, 9, 10, 11]. Our contribution is twofold: we formulate the problem as a set of integer constraints and we employ the randomized algorithm of [1] to efficiently – and provably – identify a solution that satisfies all the constraints.

3. Proposed Algorithm In this section, we demonstrate how to model the problem as a set of integer inequalities; we then use *randomized rounding* to identify *feasible points* - namely points satisfying *all* the constraints. Combining this idea with binary search allows us to *minimize* the number of parity bits. The FSM has a total of $n+k$ next state and output bits $\{S_1, S_2, \dots, S_{n+k}\}$. We are given a set of m erroneous cases $\mathcal{F} = \{EC_1, EC_2 \dots, EC_m\}$ and an $m \times (n+k)$ matrix V such that $V_{ij} = 1$ if erroneous case EC_i is detected by the j-th state/output bit S_j; otherwise, $V_{ij} = 0$. We remind the reader that, for boolean variables x, y, $x \oplus y = (x+y) \bmod 2$ and any subset of $\{S_1, S_2, \dots, S_{n+k}\}$ may be represented by an $n+k$-dimensional binary vector (e.g. the subset $\{S_1, S_3\}$ may be represented by $[1010 \dots 0]$). The problem may now be restated as follows:

Statement 1 *Given a positive integer ℓ, find ℓ vectors $\beta^{(1)}, \dots, \beta^{(\ell)} \in \{0,1\}^{n+k}$ such that*

$$\sum_{i=1}^{\ell} \left[\left(\sum_{j=1}^{n+k} V_{xj} \beta_j^{(i)} \right) \bmod 2 \right] \geq 1, \ \forall x = 1 \dots m$$

or report the lack thereof.

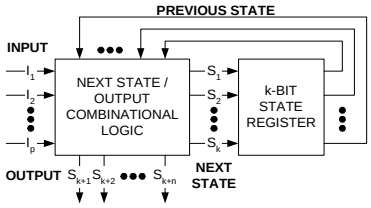

Figure 1. Example FSM

	S_1	S_2	...	S_k	S_{k+1}	S_{k+2}	...	S_{n+k}
Erroneous Case$_1$	1			1				1
Erroneous Case$_2$		1						
. . .	. . .	. . .	. . .	. . .	. . .	. . .	. . .	. . .
Erroneous Case$_m$		1					1	1

Figure 2. Error Detectability Table

In order to understand the above constraints, observe that if $\left(\sum_{j=1}^{n+k} V_{1j}\beta_j^{(i)}\right) \bmod 2 = 1$, the XOR of the bits in the set represented by the vector $\beta^{(i)}$ detects the erroneous case EC_1. Thus, in order to detect EC_1, we require that *at least one* of the ℓ subsets represented by the vectors $\beta^{(i)}$ detects the erroneous case. The same constraint is repeated for all m $EC_i \in \mathcal{F}$. We note that if we can solve the above problem in time T, then we may easily minimize ℓ in $T \cdot \log(n+k)$ time: since $1 \leq \ell \leq n+k$, we may perform binary search and find the optimal ℓ. We now remove the *mod* operator:

Statement 2 *Given a positive integer ℓ, find vectors $\beta^{(i)}$, $r^{(i)}$, $w^{(i)}$, $i = 1 \ldots \ell$ such that*

$$\begin{aligned} V \cdot \beta^{(i)} &= 2 \cdot w^{(i)} + r^{(i)}, \;\; i = 1 \ldots \ell \\ r^{(1)} + \ldots + r^{(\ell)} &\geq \vec{\mathbf{1}}_{\mathbf{m}} \\ \beta^{(1)}, \ldots, \beta^{(\ell)} &\in \{0,1\}^{n+k} \\ r^{(1)}, \ldots, r^{(\ell)} &\in \{0,1\}^{m} \\ w^{(1)}, \ldots, w^{(\ell)} &\in \{0,1,\ldots,\lfloor (n+k)/2 \rfloor\}^{m} \end{aligned}$$

In the above, $w^{(i)}$ is an m-dimensional vector which essentially removes the $mod\ 2$ operation; we also require that the sum of the $r^{(i)}$ is, element-wise, at least one, thus guaranteeing that every erroneous case is detected.

In statement 2, we described our problem as an *integer program*. Our goal is to find a *feasible point*; namely, values for all $r^{(i)}$, $w^{(i)}$ and $\beta^{(i)}$ (a total of $\ell(2m+n+k)$ variables) such that *all* the restrictions of statement 2 are satisfied. Identifying a feasible point for an integer program is NP-complete; we employ a technique called *randomized rounding* [1] to solve it. The idea of randomized rounding is simple: solve the linear programming relaxation of the integer program (which is done in polynomial time using the Simplex algorithm) and round the resulting *real* values *probabilistically*, thus forcing them to integers. We can prove that such an algorithm identifies a feasible point for the integer program of statement 2 with high probability (if one exists).

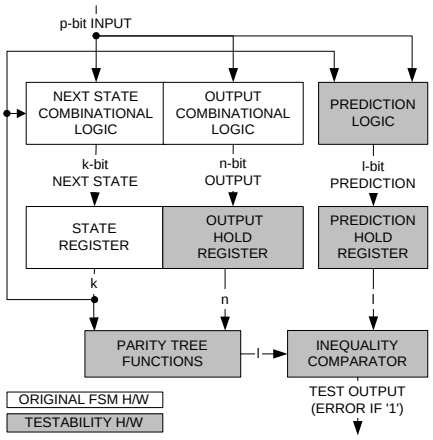

Figure 3. Proposed Methodology Overview

The proposed methodology has been implemented and applied on several MCNC benchmark FSMs. Experimental results demonstrate that a very small number of parity functions is adequate to detect all errors, thus leading to significant hardware cost reduction over duplication.

References

[1] P. Raghavan and C. Thompson, "Randomized rounding: A technique for provably good algorithms and algorithmic proofs," *Combinatorica*, vol. 7, no. 4, pp. 365–374, 1987.

[2] G. Aksenova and E. Sogomonyan, "Synthesis of built-in test circuits for automata with memory," *Automation and Remote Control*, vol. 32, no. 9, pp. 1492–1500, 1971.

[3] V. V. Danilov, N. V. Kolesov, and B. P. Podkopaev, "An algebraic model for the hardware monitoring of automata," *Automation and Remote Control*, vol. 36, no. 6, pp. 984–991, 1975.

[4] G. Aksenova and E. Sogomonyan, "Design of self-checking built-in check circuits for automata with memory," *Automation and Remote Control*, vol. 36, no. 7, pp. 1169–1177, 1975.

[5] S. Dhawan and R. C. De Vries, "Design of self-checking sequential machines," *IEEE Transactions on Computers*, vol. 37, no. 10, pp. 1280–1284, 1988.

[6] M. Gossel and S. Graf, *Error Detection Circuits*, McGraw-Hill, 1993.

[7] S. Tarnick, "Bounding error masking in linear output space compression schemes," in *Asian Test Symposium*, 1994, pp. 27–32.

[8] R. A. Parekhji, G. Venkatesh, and S. D. Sherlekar, "Concurrent error detection using monitoring machines," *IEEE Design and Test of Computers*, vol. 12, no. 3, pp. 24–32, 1995.

[9] S. J. Piestrak, "Self-checking design in Eastern Europe," *IEEE Design and Test of Computers*, vol. 13, no. 1, pp. 16–25, 1996.

[10] K. Chakrabarty and J. P. Hayes, "Test response compaction using multiplexed parity trees," *IEEE Transactions on Computer Aided Design of Integrated Circuits and Systems*, vol. 15, no. 11, pp. 1399–1408, 1996.

[11] O. Sinanoglu and A. Orailoglu, "Space and time compaction schemes for embedded cores," in *International Test Conference*, 2001, pp. 521–529.