On Concurrent Error Detection with Bounded Latency in FSMs

Sobeeh Almukhaizim
Electrical Engineering Dept.
Yale University
New Haven, CT 06520, USA

Petros Drineas
Computer Science Dept.
Rensselaer Polytechnic Institute
Troy, NY 12180, USA

Yiorgos Makris
Electrical Engineering Dept.
Yale University
New Haven, CT 06520, USA

Abstract

We discuss the problem of concurrent error detection (CED) with bounded latency in finite state machines (FSMs). The objective of this approach is to reduce the overhead of CED, albeit at the cost of introducing a small latency in the detection of errors. In order to ensure no loss of error detection capabilities as compared to CED without latency, an upper bound is imposed on the introduced latency. We examine the necessary conditions for performing CED with bounded latency, based on which we extend a parity-based method to permit bounded latency. We formulate the problem of minimizing the number of required parity bits as an Integer Program and we propose an algorithm based on Linear Program relaxation and Randomized Rounding to solve it. Experimental results indicate that allowing a small bounded latency reduces the hardware cost of the CED circuitry.

1. Introduction

A plethora of research efforts have been expended in concurrent error detection for both combinational and sequential circuits [1, 2, 3]. Proposed solutions explore the trade-offs between the three key parameters of this problem: achieved *coverage*, incurred *overhead*, and potential *latency*. The vast majority of existing approaches require that errors be detected with zero latency for combinational logic or a maximum latency of one clock cycle in sequential circuits so that faults in the bi-stable elements may also be detected. Consequently, such methods [4, 5, 6, 7, 8] attempt to reduce the incurred overhead by restricting the set of detectable errors based on realistic assumptions regarding potential malfunctions. Numerous methods wherein the utmost objective is to incur very low overhead have also been proposed. In such cases [9, 10, 11], an unbounded latency between error occurrence and error detection is permitted, implying that errors may remain indefinitely undetected, thus reducing coverage. A third alternative, however, namely that of exploring the trade-off between latency and overhead while guaranteeing detection of a given set of errors has received little attention.

In this work, we present a method for performing CED with *bounded latency* in FSMs. In essence, the objective of this approach is to reduce the overhead of CED by allowing a small delay between error occurrence and error detection. We emphasize, however, that the worst-case delay is bounded. Thus, while the hardware necessary for performing CED may be reduced, it is still possible to guarantee error detection within the specified latency. The efficiency of the method depends on the structure of the original FSM and the set of targeted errors. Therefore, we first study the problem of CED with bounded latency in FSMs in its general form and we derive a set of conditions that need to be met in order to provide error detection latency guarantees. We then demonstrate that this idea can be effectively implemented by extending an existing Parity-Based CED method for FSMs to accommodate bounded latency.

The proposed method is capable of detecting *all* errors resulting from any specified fault model. Such a fault model can be prescribed by providing the error-free response and all erroneous responses resulting from faults in the model for every transition in the FSM. Target fault models are expected to be *restricted*, in the sense that the set of resulting erroneous responses should be a subset of all possible circuit responses. Indeed, for an unrestricted fault model, information theory proves that any non-intrusive concurrent error detection circuit will be as complex as the original circuit [12]. When a restricted fault model is specified, however, more cost-effective solutions may be devised [13]. To our knowledge, the only previously proposed method that provides an upper bound to the detection latency is based on the use of convolutional codes. In this method, additional logic is used to generate *key* bits during every FSM transition, such that these keys are valid sequences in a convolutional code if and only if the FSM is operating correctly. The theoretical foundation for this method is described extensively in [14], yet no indication of its cost is provided. Unfortunately, for convolutional codes of latency more than one clock cycle, the method becomes cumbersome.

The paper is organized as follows. The general requirements for performing CED with bounded latency are discussed in section 2. A parity-based implementation of CED with bounded latency for FSMs is proposed in section 3, wherein the optimization problem of minimizing the number of necessary parity bits required is also discussed. The Integer Program formulation of the parity bit minimization problem and the proposed solution based on Linear Program relaxation and Randomized Rounding are described in section 4. Experimental results on MCNC benchmark FSMs are provided in section 5, demonstrating that allowing a small, bounded latency can reduce the hardware cost while preserving the desired level of error coverage.

1530-1591/04 $20.00 (c) 2004 IEEE

2. CED with Bounded Latency

In CED without latency, erroneous FSM transitions are detected immediately. Bounded latency, on the other hand, provides more freedom as to when to detect errors. Consequently, an erroneous transition may be ignored, as long as it is *guaranteed* that the causing fault will result in another error that will be detected within p clock cycles, where p is the specified latency bound. For this to be possible, we assume that a fault remains present for at least p clock cycles after causing an error. This assumption reflects realistically permanent faults and intermittent faults due to wear-&-tear. It may also reflect transient errors, if the error source has a continuous duration of at least a few clock cycles, which is the targeted latency bound. However, it does not reflect single event upsets (SEU). For SEU, concurrent error detection allowing bounded latency would either have to restrict the bound to one or use some form of memory which increases the overall cost. One such solution is the method based on convolutional encoding, which is described in [14].

In order to omit immediate detection of an erroneous response, we need to enumerate all paths of length p, starting from the state where the error is initially activated. A CED mechanism should be capable of detecting the underlying fault in *all* such paths, yet not necessarily during the initial transition. Path enumeration, either explicit or implicit, is a costly procedure. However, since we only target a bounded latency of a few clock cycles, the exponential explosion is contained and the number of paths is manageable.

By permitting latency in error detection, we anticipate simplification of the circuit necessary for implementing a CED method. In essence, latency relaxes the constraints in designing a CED circuit by allowing more flexibility as to when faults are detected. Unfortunately, overhead reduction due to latency reaches a saturation point, after which increasing the latency bound does not provide more choices. This happens because of loops during path enumeration. As soon as a loop occurs, enumeration along this path is terminated, since any additional latency increase will result in at least one path that expands along the loop. Detecting an error along this path implies detection of errors along all paths comprising the loop. Given a fault model, we can find the maximum latency of interest by finding the length of the shortest loop on each faulty FSM and selecting the largest value.

3. Parity-Based Implementation

In this section, we develop a parity-based implementation of the general algebraic method [7] for CED in FSMs, where a set of parity trees performs lossless *compaction* of the circuit responses. Additional hardware is subsequently used to *predict* the compacted error-free responses and a comparator is employed to identify potential discrepancies between the output of the compactor and the output of the predictor.

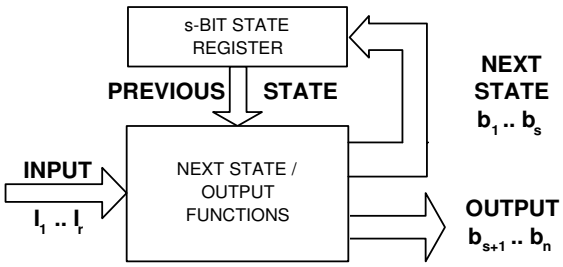

Figure 1. FSM Example

In an effort to reduce the incurred overhead, this method attempts to minimize the number of parity trees required for lossless compaction and, by extension, the size of the predictor and the comparator. We demonstrate how this method can be employed for performing parity-based CED with bounded latency in FSMs. The purpose of allowing latency is to reduce the overhead, i.e. the number of parity functions that need to be constructed in order to guarantee detection of all erroneous cases. The underlying principle for achieving this is that by allowing a small, bounded latency, we have more choices on how to detect each error and, therefore, a smaller number of parity functions might suffice. Under the assumption that the causing fault persists for at least p clock cycles after triggering an error, we do not necessarily need to detect the error during the first transition. Rather, we need to ensure that the selected parity functions are capable of detecting an effect of the fault along every possible path of length p, starting from the first erroneous state.

3.1. Problem Formulation

Consider the FSM with r inputs, s state bits, and $n - s$ outputs shown in Fig. 1. For every combination of a sequence of inputs $A = a_1, \ldots, a_p$, where $a_j \in 0, \ldots, 2^r - 1$, $j \in 1, \ldots, p$, and previous state c, $c \in 0, \ldots, 2^s - 1$, any error caused by a fault f will manifest itself as a difference between the sequence of error-free responses $GM(A, c)$, $GM(A, c) \in (0, \ldots, 2^n - 1)^p$, and the sequence of erroneous responses $BM_f(A, c)$, $BM_f(A, c) \in (0, \ldots, 2^n - 1)^p$. During each of the p FSM transitions, this difference is detectable in a set of state and output bits b_i, where $i \in 1, \ldots, n$. The concatenation of these p sets defines an *Erroneous Case*, $EC(A, c, f)$. Clearly, several combinations of transition sequences (A, c) and faults f may lead to the same erroneous case, i.e. the same p sets of bits through which the effect of fault f on transition sequence (A, c) may be detected across the p transitions. The union of all erroneous cases, $\mathcal{F} = \bigcup_{\forall(A,c,f)} EC_{(A,c,f)}$, can be represented in tabular format in an error detectability table, as shown in Fig. 2. In this table, super-columns correspond to the p transitions, columns correspond to the n next state/output bits, rows correspond to the m erroneous cases, and entries in the table indicate the state/output bits at which each erroneous case may be detected.

	Latency 1							Latency 2								Latency p						
	b_1	b_2	...	b_s	b_{s+1}	...	b_n	b_1	b_2	...	b_s	b_{s+1}	...	b_n		b_1	b_2	...	b_s	b_{s+1}	...	b_n
Erroneous Case$_1$	1			1																		
Erroneous Case$_2$		1							1								1					
. . .	. . .	. . .	. . .	. . .	. . .	. . .	. . .	. . .	. . .	. . .	. . .	. . .	. . .	. . .		. . .	. . .	. . .	. . .	. . .	. . .	. . .
Erroneous Case$_m$		1				1	1		1					1			1		1		1	1

Figure 2. Error Detectability Table for CED with Bounded Latency

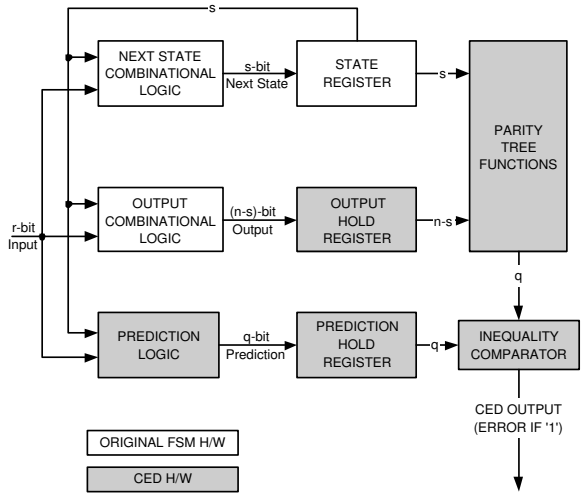

Figure 3. Proposed Methodology Overview

Detecting all circuit errors requires that at least one state/output bit in each erroneous case in $\mathcal{F}$ be predicted through additional hardware and compared to its actual runtime value. To minimize the overhead, the number of predicted bits should be small. Yet, since 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, we employ 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 compacting the state/output bits will be adequate to cover all erroneous cases.

Using the information in the error detectability table, the optimization objective of our method is to *minimize* the number of q parity bits that need to be constructed out of the next state/output bits b_1 through b_n, such that all Erroneous Cases are detected. An Erroneous Case is detected by a parity tree *if and only if* the parity tree comprises an odd number of bits b_i that detect the Erroneous Case *at any specific time-step* between 1 and p. We note that the problem may be modelled as an NP-complete minimum cover problem, for which several heuristics exist [15]. However, expanding the error detectability table to explicitly represent all possible parity functions for each latency step is infeasible, since there is an exponential number of alternative parity combinations.

The benefit of allowing bounded latency stems from the larger number of alternative ways to detect each Erroneous Case. This can be seen in the last row of the error detectability table of Fig. 2, where Erroneous Case m may be detected by more bits and combinations of bits across the p transitions than just in the first transition. Of course, it is also possible that for some Erroneous Cases latency will not provide any additional flexibility. This is the case, for example, for Erroneous Cases 1 and 2 in the table of Fig. 2. In the first case, the fault only affects the first transition and cannot be detected at a later time within p transitions. In the second case, the fault affects exactly the same bits in every one of the p transitions.

Based on the above observations, the proposed methodology is very straightforward, as depicted in the form of a block diagram in Fig. 3. Given an FSM with r inputs, s state bits, and $n - s$ outputs, XOR trees are employed to implement the q parity functions required for lossless state/output bit compaction. Combinational logic is employed to predict the values of the q bits that compact the n state/output bits for each FSM transition, and a comparator is used to identify any discrepancy. Similar to [16], 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 in the restricted error model are detected within latency of p clock cycles.

4. Proposed Algorithm

In this section, we demonstrate how to model the problem as an integer program; we subsequently use *randomized rounding* to identify *feasible points* - namely points satisfying *all* the constraints. Our integer program is an extension of the result in [17], which may be viewed as a special case of this section's formulation when latency is equal to one[1].

We start by introducing some notation that will be useful throughout this section. Let $[x]$ denote the sequence $1, 2, 3, \ldots, x$ for any non-zero positive integer x. Assume that the FSM has a total of n next state/output bits, denoted by $\{b_1, b_2, \ldots, b_n\}$ (see Fig. 1). We are also given a set of m erroneous cases, denoted by $\mathcal{F} = \{EC_1, EC_2 \ldots, EC_m\}$ and a target latency p, which is a non-zero positive integer.

The most important part of the input is a 3-dimensional array[2], which will be denoted by V. The dimensions of V are $m \times n \times p$; we denote the (i, j, k) element of V by $V(i, j, k)$

[1] We remind that in the basic parity-based method all errors are detected with a latency of one clock cycle to also detect faults in the state register.

[2] V is a tensor; since it is always 3-dimensional we avoid using tensor terminology/notation.

with $i \in [m], j \in [n], k \in [p]$. We use the notation $V(:, j, k)$ to denote all elements $V(i, j, k)$ for all $i \in [m]$; $V(i, :, k)$, $V(i, j, :)$ and all combinations are defined similarly. V is a 0-1 matrix, defined as follows:

Definition 1 *$V(i, j, k)$ is equal to 1 if and only if the erroneous case EC_i is detected by the j-th output bit b_j with latency k; otherwise, $V(i, j, k)$ is equal to 0.*

Our problem may now be stated as follows:

Statement 1 *Given a positive integer q, find q subsets $\beta_1, \ldots, \beta_q$ of $\{b_1, b_2, \ldots, b_n\}$ such that*

$$\mathbf{cov}(\oplus\beta_1) \cup \mathbf{cov}(\oplus\beta_2) \cup \ldots \cup \mathbf{cov}(\oplus\beta_q) = \mathcal{F}$$

or report the lack thereof.

Here, $\oplus\beta_\ell$ ($\ell \in [q]$) denotes the XOR of bits in β_ℓ and $\mathbf{cov}(\oplus\beta_\ell)$ denotes the erroneous cases covered by the XOR of bits in β_ℓ. An erroneous case EC_i is covered by the XOR of the bits in β_ℓ if and only if

$$\sum_{k=1}^{p} \left(\left(\sum_{b_y \in \beta_\ell} V(i, y, k) \right) \bmod 2 \right) \geq 1$$

We remind the reader that, for boolean variables x_1, x_2, $x_1 \oplus x_2 = (x_1 + x_2) \bmod 2$. Then, the above formula essentially means that the XOR of the bits in β_ℓ detects EC_i with some latency $k \in [p]$. Thus, using q parity bits (the XORs of the bits in β_ℓ, $\ell \in [q]$) we can detect all erroneous cases.

We note that if we can solve the above problem in time T, then we may easily minimize q in $T \log n$ time: since $1 \leq q \leq n$, we may perform binary search and find the optimal q, as shown in Algorithm 1.

In the following, we will denote any subset of $\{b_1, b_2, \ldots, b_n\}$ by an n-dimensional 0-1 vector (e.g. the subset $\{b_1, b_3, b_4\}$ may be represented by $[1011 \ldots 0]$). The problem may now be restated as follows:

Statement 2 *Given a positive integer q, find q n-dimensional binary vectors $\beta^{(1)}, \ldots, \beta^{(q)}$ such that*

$$\begin{aligned}
\sum_{\ell=1}^{q} \left[\sum_{k=1}^{p} \left(\left(\sum_{y:\beta_y^{(\ell)}=1} V(1, y, k) \right) \bmod 2 \right) \right] &\geq 1 \\
\sum_{\ell=1}^{q} \left[\sum_{k=1}^{p} \left(\left(\sum_{y:\beta_y^{(\ell)}=1} V(2, y, k) \right) \bmod 2 \right) \right] &\geq 1 \\
&\vdots \\
\sum_{\ell=1}^{q} \left[\sum_{k=1}^{p} \left(\left(\sum_{y:\beta_y^{(\ell)}=1} V(m, y, k) \right) \bmod 2 \right) \right] &\geq 1
\end{aligned}$$

or report the lack thereof.

In order to understand the above constraints, observe that if $\sum_{k=1}^{p} \left(\left(\sum_{y:\beta_y^{(\ell)}=1} V(i, y, k) \right) \bmod 2 \right)$ is at least 1, the XOR of the bits in $\beta^{(\ell)}$ detects the erroneous condition EC_i with latency at most p, where $i \in [m]$. Thus, in order for *at least* one of the $\beta^{(\ell)}, \ell \in [q]$ to detect EC_i, it is enough to satisfy the first constraint. We may now state our problem in matrix notation:

Statement 3 *Given a positive integer q, find q n-dimensional binary vectors $\beta^{(1)}, \ldots, \beta^{(q)}$ such that*

$$\sum_{\ell=1}^{q} \left[\sum_{k=1}^{p} \left(V(:, :, k) \cdot \beta^{(\ell)} \bmod 2 \right) \right] \geq \vec{\mathbf{1}}_\mathbf{m}$$

or report the lack thereof, where $\vec{\mathbf{1}}_m$ is an m-vector of 1s.

Notice that $V(:, :, k)$ is a 2-dimensional array (a matrix of dimensions $m \times n$) that denotes the erroneous cases captured by the output bits with latency exactly equal to k.

We now remove the *mod* operator from the statement:

Statement 4 *Given a positive integer q, find vectors $\beta^{(\ell)}, \ell \in [q]$, $r^{(\ell k)}$, $w^{(\ell k)}$, $\ell \in [q]$, $k \in [p]$ such that*

$$\begin{aligned}
\forall\, k \in [p],\ V(:, :, k) \cdot \beta^{(1)} &= 2 \cdot w^{(1k)} + r^{(1k)} \\
\forall\, k \in [p],\ V(:, :, k) \cdot \beta^{(2)} &= 2 \cdot w^{(2k)} + r^{(2k)} \\
&\vdots \\
\forall\, k \in [p],\ V(:, :, k) \cdot \beta^{(q)} &= 2 \cdot w^{(qk)} + r^{(qk)} \\
\sum_{k=1}^{p} \left(r^{(1k)} + \ldots + r^{(qk)} \right) &\geq \vec{\mathbf{1}}_\mathbf{m} \\
\beta^{(1)}, \ldots, \beta^{(q)} &\in \{0, 1\}^n \\
r^{(\ell k)} &\in \{0, 1\}^m \\
w^{(\ell k)} &\in \{0, 1, \ldots, \lfloor n/2 \rfloor\}^m
\end{aligned}$$

In order to understand the above constraints, observe that e.g. $r^{(1k)}$ is an m-dimensional 0-1 vector denoting whether $\{EC_1, \ldots, EC_m\}$ are detected by the XOR of the bits in the set represented by $\beta^{(1)}$ with latency k. We note that $w^{(1k)}$ is also an m-dimensional vector that removes the $mod\ 2$ operation. The sum of the $r^{(\ell k)}$ is, element-wise, at least one, thus guaranteeing that every erroneous case is detected.

In statement 4, we described our problem as an *integer program*. Our goal is to find a *feasible point*; namely, values for all $r^{(\ell k)}$, $w^{(\ell k)}$ and $\beta^{(\ell)}$ (a total of $2qpm + qn$ variables) such that *all* the restrictions of statement 4 are satisfied. Identifying a feasible point for an integer program is NP-complete; we, therefore, employ a technique called *randomized rounding* [18] to solve it. The idea of randomized rounding is simple: solve the linear programming relaxation of the integer program (easily done in polynomial time using

Data : matrix V
Result : vectors $\beta^{(\ell)}$ that detect all ECs
$ITER = 10^3$; $left = 1$; $right = n$;
while $left < right$ **do**
  $q = \lfloor (right - left)/2 \rfloor$;
  Find a feasible point for the LP of statement 5;
  if *none exists* **then**
    set $left = q$;
  else
    repeat
      Use randomized rounding to create an integer solution;
    until *all constraints of statement 4 are satisfied* **or** $ITER$ *repetitions are exceeded*;
    if *all constraints of statement 4 are satisfied* **then**
      set $right = q$;
    else
      set $left = q$;
    end
  end
end
Report the minimal q s.t. all constraints of statement 4 are satisfied and the corresponding $\beta^{(\ell)}, \ell \in [q]$.

Algorithm 1: The overall algorithm

e.g. the Simplex algorithm) and round the resulting *real* values *probabilistically*. We now state the linear programming relaxation of statement 4:

Statement 5 *Given a positive integer q, find vectors $\beta^{(\ell)}, \ell \in [q]$, $r^{(\ell k)}$, $w^{(\ell k)}$, $\ell \in [q]$, $k \in [p]$ such that*

$$\begin{aligned}
\forall\, k \in [p],\ V(:,:,k) \cdot \tilde{\beta}^{(1)} &= 2 \cdot \tilde{w}^{(1k)} + \tilde{r}^{(1k)} \\
\forall\, k \in [p],\ V(:,:,k) \cdot \tilde{\beta}^{(2)} &= 2 \cdot \tilde{w}^{(2k)} + \tilde{r}^{(2k)} \\
&\vdots \\
\forall\, k \in [p],\ V(:,:,k) \cdot \tilde{\beta}^{(q)} &= 2 \cdot \tilde{w}^{(qk)} + \tilde{r}^{(qk)} \\
\sum_{k=1}^{p} \left(\tilde{r}^{(1k)} + \ldots + \tilde{r}^{(qk)} \right) &\geq \vec{\mathbf{1}}_{\mathbf{m}} \\
\tilde{\beta}^{(1)}, \ldots, \tilde{\beta}^{(q)} &\in [0,1]^n \\
\tilde{r}^{(\ell k)} &\in [0,1]^m \\
\tilde{w}^{(\ell k)} &\in [0, \lfloor n/2 \rfloor]^m
\end{aligned}$$

We round each of the $\tilde{x}$ variables as follows:

$$x = \begin{cases} 1 & \text{,with probability } \tilde{x} \\ 0 & \text{,otherwise} \end{cases} \quad (1)$$

Raghavan *et al.* [18] argue that this simple algorithm identifies a feasible point with high probability, if one exists. In practice, we probabilistically round x a fixed number of times and verify that a solution is found.

5. Experimental Results

The proposed methodology has been implemented and applied on several sequential MCNC benchmarks. After performing state assignment, the circuits are synthesized and mapped onto a standard-cell library using SIS [19]. Internally developed software employing fault simulation is used to identify the error-free and erroneous responses to generate the error detectability table of Fig. 2. While the stuck-at fault model has been used as the source of errors, we emphasize that the method applies for any restricted error model, as discussed in section 2. Subsequently, Algorithm 1 is applied to compute the minimal number of parity functions for several values of latency p.

The results are summarized in Table 1. Under the first major heading, we provide details about the circuits that were used: name, number of inputs, state bits, outputs, gate count and the hardware cost reported by SIS. Under the second, third and forth major heading, we provide the minimum number of parity functions required for complete fault coverage, the gate count and the hardware cost reported by SIS for latency $p = 1$, $p = 2$ and $p = 3$, respectively. The number of parity functions (hardware cost) for the basic parity-based method with latency $p = 1$ [17] for these examples is, on average, 53.00% (22.40%) smaller than the number of functions (hardware cost) necessary for duplicating the circuit. Addition of one more clock cycle, i.e. for bounded latency $p = 2$, reduces the number of parity bits (hardware cost) by 11.70% (7.18%) over the number of parity bits (hardware cost) required for latency $p = 1$. Further increase of the bound to latency $p = 3$, yields an additional 7.23% (7.08%) reduction in the number of parity bits (hardware cost).

As discussed in section 2, the benefits of adding latency diminish as latency increases. In smaller FSMs, faults result to a large number of self-loops. For example, this is the case for circuits donfile, s27, and s386. As the FSM size becomes larger, self-loops are less frequent and the benefits of increasing the detection latency are more significant. This is for example the case for circuits pma, s298, and s1488.

The reduction in the number of parity functions and the reduction in the hardware cost of the predictor are not necessarily proportional. For example, this is the case for circuit dk16 where a latency of $p = 2$ reduces the number of parity functions by 16.67%, yet the hardware overhead is reduced by 19.94%. More surprisingly, the hardware overhead increases by 11.37% when latency, $p = 2$, is increased to $p = 3$. A single complex parity function may require the same or more area than a larger number of simple parity functions. To the best of our knowledge, the literature lacks solutions that consider the actual area cost of parity functions as a metric in choosing which parity functions to select. In the absence of such methods, the most promising direction is to reduce the number of parity functions, anticipating that, on average, functions will incur the same area cost.

Original Circuit						Latency p=1			Latency p=2			Latency p=3		
Circuit Name	Input Bits	State Bits	Output Bits	Gates	Cost	# of Trees	Gates	Cost	# of Trees	Gates	Cost	# of Trees	Gates	Cost
cse	7	4	7	196	256128	5	131	171680	5	131	171680	4	106	137344
donfile	2	5	1	97	128064	4	57	74704	4	78	103008	4	83	111360
dk16	2	5	3	240	317840	6	323	428736	5	257	342896	5	288	381872
dk512	1	4	3	74	96048	4	79	104400	4	77	101616	4	62	81200
ex1	9	5	19	263	343360	8	240	319232	7	211	280720	6	183	243600
keyb	7	5	2	228	298352	5	82	107648	4	64	83984	4	72	92800
pma	8	5	8	347	453792	6	186	243136	5	179	234784	4	149	193952
sse	7	4	7	131	178640	5	80	104864	4	64	82592	4	60	77952
styr	9	5	10	413	547056	8	217	287216	6	203	266336	5	155	206016
s1488	8	6	19	552	727552	8	256	335936	7	240	315984	6	228	300672
s27	4	3	1	20	25056	3	15	18096	3	15	18096	3	15	18096
s298	3	14	6	114	147552	8	324	428272	7	313	416672	7	295	396720
s386	7	6	7	123	158688	4	83	105328	4	83	105328	4	83	105328
tav	4	2	4	28	34336	4	31	39440	3	25	31552	3	33	42688
tbk	6	5	3	146	190240	5	160	207872	5	160	207872	5	160	207872
tma	7	5	6	218	285360	5	130	169824	4	115	150800	4	115	150800

Table 1. Experimental Results on MCNC Benchmark Circuits

6. Conclusion

We introduced a technique that allows concurrent error detection with latency in FSMs. In order to preserve the level of attainable coverage, we bound the detection latency to a few cycles and we derive the necessary conditions to detect all possible errors within the specified latency period. Since a bounded delay is permitted in the detection of errors, the trade-off between latency and hardware required for concurrent error detection can be beneficially explored. In order to assess the effectiveness of this approach, we extended a latency-free parity-based method to perform error detection with bounded latency. We formulated the problem of minimizing the number of required parity bits as an Integer Program and we devised an algorithm based on Linear Program relaxation and Randomized Rounding to solve it. Experimental results indicate a monotonic reduction in the cost of the CED hardware when latency is increased.

References

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

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

[3] S. Mitra and E. J. McCluskey, "Which concurrent error detection scheme to choose?," in *International Test Conference*, 2000, pp. 985–994.

[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] N. A. Touba and E. J. McCluskey, "Logic synthesis of multilevel circuits with concurrent error detection," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, vol. 16, no. 7, pp. 783–789, 1997.

[7] 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.

[8] C. Bolchini, F. Salice, and D. Sciuto, "A novel methodology for designing TSC networks based on the parity bit code," in *European Design and Test Conference*, 1997, pp. 440–444.

[9] R. Leveugle and G. Saucier, "Optimized synthesis of concurrently checked controllers," *IEEE Transactions on Computers*, vol. 39, no. 4, pp. 419–425, 1990.

[10] S. H. Robinson and J. P. Shen, "Direct methods for synthesis of self-monitoring state machines," in *Fault Tolerant Computing Symposium*, 1992, pp. 306–315.

[11] P. Drineas and Y. Makris, "SPaRe: selective partial replication for concurrent fault detection in FSMs," in *International Conference on VLSI Design*, 2003.

[12] J. F. Meyer and R. J. Sundstrom, "On-line diagnosis of unrestricted faults," *IEEE Transactions on Computers*, vol. 24, no. 5, pp. 468–475, 1975.

[13] E. S. Sogomonyan, "The design of discrete devices with diagnostics in the course of operation," *Automation and Remote Control*, vol. 31, no. 11, pp. 1854–1860, 1970.

[14] L. P. Holmquist and L. L. Kinney, "Concurrent error detection for restricted fault sets in sequential circuits and microprogrammed control units using convolutional codes," in *International Test Conference*, 1991, pp. 926–935.

[15] M. R. Garey and D. S. Johnson, *Computers and Intractability: A Guide to the Theory of NP-Completeness*, W. H. Freeman, 1979.

[16] C. Zeng, N. Saxena, and E. J. McCluskey, "Finite state machine synthesis with concurrent error detection," in *International Test Conference*, 1999, pp. 672–679.

[17] P. Drineas and Y. Makris, "Non-intrusive concurrent error detection in FSMs through State/Output compaction and monitoring via parity trees," in *Design Automation and Test in Europe Conference*, 2003, pp. 1164–1165.

[18] 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.

[19] E. M. Sentovich et al., "SIS: a system for sequential circuit synthesis," ERL MEMO. No. UCB/ERL M92/41, EECS UC Berkeley CA 94720, 1992.