Kuwait J. Sci. Eng. **36** (2B) pp. 141-162, 2009

Parity-based concurrent error detection with bounded latency in finite state machines

SOBEEH ALMUKHAIZIM[*], PETROS DRINEAS[**], AND YIORGOS MAKRIS[***]

[*] *Computer Engineering Department, Kuwait University, Kuwait*
E-mail: sobeeh.almukhaizim@ku.edu.kw
[**] *Department of Computer Science, Rensselaer Polytechnic Institute, USA,*
E-mail: drinep@cs.rpi.edu
[***] *Departments of Electrical Engineering and Computer Science, Yale University, USA,*
E-mail: yiorgos.makris@yale.edu

ABSTRACT

We extend a previously-developed non-intrusive concurrent error detection (CED) method for finite state machines (FSMs) to perform CED with bounded latency. The proposed method is based on compaction of the state/output bits of the FSM via parity trees and comparison to the error-free compacted responses, which are predicted through additional hardware. The corresponding optimization problem is formulated as an integer linear program and an algorithm to approximate its optimal solution through the use of linear program relaxation and randomized rounding is outlined. In order to reduce the incurred area overhead, we approximate the entropy of parity trees required for performing lossless compaction, and we select low-entropy ones. We then extend this method to support CED with *bounded latency*, wherein the overhead is further reduced at the cost of introducing a small latency in the detection of errors. Experimental results demonstrate that allowing a small bounded latency in the detection of errors yields further reduction in the parity predictor hardware overhead.

Keywords: Concurrent Error Detection, Finite State Machines, Integer Programming, On-Line Test, Parity

INTRODUCTION

Electronic circuits are employed in a wide variety of modern life activities, ranging from simple commodity devices (e.g., portable audio players, mobile phones, PDAs, etc.) to mission-critical applications (e.g., nuclear reactors, avionics, ABS brakes, etc.). While potential malfunctions in the former may cause no damage other than inconvenience, the slightest malfunction in the latter may have catastrophic consequences. Inevitably, frequent occurrence of malfunctions is a fact of life, instigated either by permanent failures (e.g., environmental impact, wear and tear), or by transient error sources (e.g., electromagnetic interference, cosmic radiation). Therefore, shielding electronic circuits against malfunctions, while meeting design constraints such as cost and performance, is of paramount importance.

Towards this end, CED methods are typically employed to monitor a circuit during its course of operation. The simplest approach is to duplicate the circuit and use a comparator to continuously monitor the outputs of the original circuit and the replica and report any discrepancy. Duplication (possibly with design diversity to avoid common-mode failures (Avizienis & Kelly, 1984)) is very effective in immediately detecting all errors. However, it incurs significant hardware overhead that exceeds 100% of the original circuit. Therefore, alternative CED methods that reduce the area overhead while preserving error detectability are necessary.

In this work, we extend a previously-developed non-intrusive parity-based CED method for FSMs (Almukhaizim et al., 2004). Parity-based CED is based on compaction of the state/output bits of an FSM via parity trees and comparison to the error-free compacted responses, which are predicted through additional hardware. Similar to duplication, this method is *latency-free*, since it detects all errors during the FSM transition in which they occur. The *key challenge* is to identify low-cost parity functions required for lossless compaction, and not necessarily the minimum number of parity trees as in (Almukhaizim et al., 2004). We formulate the problem as an integer program and we employ an algorithm based on linear programming and randomized rounding to approximate the feasible solutions. Moreover, we guide the selection of parity trees based on their entropy, which has been shown to correlate with the implementation cost of logic function. As a result, parity-based CED incurs significantly lower hardware overhead than duplication. Then, we extend this method to accommodate CED with *bounded latency*. The objective of this approach is to further 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 area overhead necessary for performing CED may be reduced, it is still possible to guarantee error detection within the specified latency bound.

The proposed method is capable of detecting all errors in a specified error model. Such a model can be prescribed by providing the error-free response and all erroneous responses for every FSM transition. Target 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, wherein an error-free response may be transformed into any arbitrary erroneous response, information theory proves that any non-intrusive concurrent error detection circuit will be as complex as the original circuit (Meyer & Sundstrom, 1975). In such cases, duplication constitutes the most appropriate solution. When a restricted fault model is specified, however, more cost-effective solutions may be devised through lossless compaction of the circuit outputs (Sogomonyan, 1970).

The use of any CED method with bounded error detection latency is dictated by the fault tolerance design methodology wherein the CED method is employed. In general, fault tolerance can be divided between an error detection phase and an error recovery phase. For real-time systems, however, the real-time constraint necessitates that error detection be immediate, rending the proposed method unsuitable for such applications. For other fault tolerance systems (e.g. checkpointing), CED with bounded detection latency can be employed safely. In general, the proposed CED method can be applied in any sequential circuit (e.g. sequential ALU, instruction scheduler, etc.) as long as the latency between the output of the controller and any induced system change is longer than the error detection latency.

The research described herein provides a cost-effective CED solution over previously-proposed methods. Specifically, the main contributions of this work, and key differences to previous methods, include:

- **Entropy-based selection of parity trees:** Unlike the previously-proposed CED method with bounded latency in (Almukhaizim et al., 2004) (where the cost of the parity predictor is minimized by reducing the *number* of parity trees required for lossless compaction), the proposed method herein selects low-entropy parity functions through a biased random walk in the set of feasible solutions, which ensures reducing the *cost* of the parity predictor. Hence, the proposed method herein provides significant cost reduction over that in (Almukhaizim et al., 2004).
- **ILP-based formulation for performing CED with bounded detection latency:** Unlike the previously-proposed latency-free CED method in (Almukhaizim et al., 2006) (where the cost of the parity predictor is minimized by selecting low-cost parity functions while ensuring that all errors are detected *immediately*), errors are detected in the proposed method herein within a given latency bound, which relaxes the constraints in selecting the parity functions by allowing more flexibility as to when errors are detected. Thus, the proposed method herein provides additional reduction over that in (Almukhaizim et al., 2006).

In short, the proposed CED method herein combines both objectives for reducing the cost of the predictor function in (Almukhaizim et al., 2004, 2006) (i.e. by allowing a bounded detection latency *and* by driving the selection of parity trees based on the entropy of the parity predictor). Thus, the proposed method is superior to the previously-proposed methods, as its implementation cost is always lower than that achieved by using a single minimization objective only (i.e. by minimizing the number of parity trees while allowing bounded detection latency, or by driving the selection of parity trees using low entropy).

RELATED WORK

The importance of concurrent test is accentuated by the plethora of methods that have been developed. Most such methods are either expensive but guarantee latency-free detection and high coverage, or are low-cost but cannot guarantee neither coverage nor a latency bound. We review next these methods and classify them according to their error detection latency.

Latency-Free Detection

Towards the high-cost end, several latency-free CED methods have been proposed for both combinational and sequential circuits (Gossel & Graf, 1993; Piestrak, 1996; Mitra & McCluskey, 2000). Reducing the area overhead below the cost of duplication typically requires redesign of the original circuit, thus leading to intrusive methodologies. Several redesign and resynthesis methods (Aksenova & Sogomonyan, 1975; Dhawan & Vries, 1988; Jha & Wang, 1993) employ parity or various unordered codes to encode the states of the circuit. Limitations of the method by Jha & Wang (1993), such as structural constraints requiring an inverter-free design, are alleviated by Touba & McCluskey (1997), where partitioning is employed to reduce the incurred hardware overhead. Utilization of multiple parity bits, first proposed by Sogomonyan (1974), is also examined by Zeng et al. (1999) within the context of FSMs. These methods render totally self-checking circuits and guarantee latency-free error detection; on the down side, they are intrusive and relatively expensive. Non-intrusive CED methods have also been proposed. The general algebraic model is introduced in (Danilov et al., 1975). Implementations based on Bose-Lin and Berger codes are presented in (Das & Touba, 1999) and (Parekhji et al., 1995), respectively. Finally, parity-based CED methods are described in (Gossel & Graf, 1993), and the general compaction-based CED methodology in (Almukhaizim et al., 2005). A parity-based partially self-checking design methodology has been described in (Mohanram et al., 2003). While most of the aforementioned methods guarantee latency-free detection of all errors in prescribed error models, their cost is often prohibitive.

Detection with Unbounded Latency

Towards the low-cost end, several non-intrusive concurrent fault detection (CFD) methods have been proposed for stuck-at faults in combinational circuits. C-BIST (Saluja et al., 1988) employs input monitoring to perform concurrent self-test. While hardware overhead is very low, the method relies on an ordered appearance of all possible input vectors before a signature indicating circuit correctness can be calculated, resulting in very long average detection latency and infinite worst-case latency. This problem is alleviated in R-CBIST (Voyiatzis et al., 1998), where the requirement for a uniquely ordered appearance of all input combinations is relaxed at the cost of a small RAM. Alternatively, average detection latency is reduced through the comparison-based method (Sharma & Saluja, 1988), which uses additional logic to predict the circuit responses for a complete test set. Similar CFD methods have been also proposed for FSMs as well (Drineas & Makris, 2003). In a different line of work, synthesis of self-monitored FSMs has been extensively examined (Leveugle & Saucier, 1990; Robinson & Shen, 1992). In this case, signatures are calculated during normal operation and compared against reference signatures at designated check points. The overhead of these schemes is very low, but the achieved coverage is also low. While the aforementioned methods guarantee that all faults are detectable, the fault detection latency is unbounded. Thus, resulting errors may elude detection and, in the worst case, the causing faults may remain undetected indefinitely.

Detection with Bounded Latency

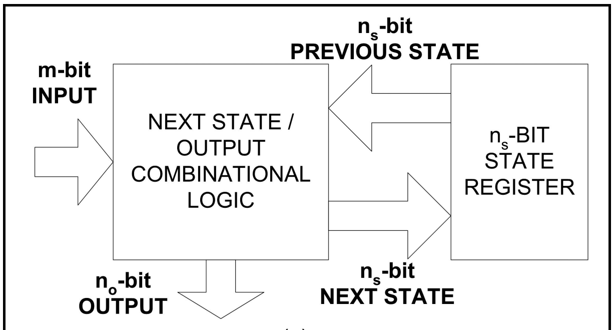
To our knowledge, the only previously proposed method that provides an upper bound to the detection latency is based on the use of convolutional codes. In this method, additional logic is utilized to generate *key* bits during every FSM transition, such that these bits are valid sequences in a convolutional code if and only if the FSM is operating correctly. Any erroneous transition in a prescribed error model will be detected with latency which will not exceed the latency of the convolutional code. The theoretical foundation for this method is described extensively by Holmquist & Kinney (1991), yet no indication of its cost is provided.

PARITY-BASED CED in FSMs

In this section, we review the parity-based CED in FSMs method (Almukhaizim et al., 2006). We first provide an overview of parity-based CED, and then model the problem of parity tree selection for lossless compaction as a set of integral inequalities. Finally, we describe how the entropy of the parity predictor can be utilized as a potential function for guiding a search algorithm that selects parity trees that incur low overhead.

Method Overview

Consider the next state/output combinational logic of the FSM shown in Fig. 1(a), which has m inputs and $n = n_s + n_o$ outputs, out of which n_s are state bits and n_o are output bits. For every combination of input and previous state, any error will manifest itself as a difference between the correct response and the erroneous response. This difference is detectable in a non-empty set of state and output bits. Each such set constitutes an *Erroneous Case (EC)*. Clearly, several combinations of a transition and an error may lead to the same erroneous case, i.e. the same set of bits on which the error may be detected during the transition. The set of all erroneous cases may be represented in the tabular format of Fig. 1(b), where columns correspond to the n state/output bits, rows correspond to the f distinct ECs, and entries of "1" in the table indicate the state/output bits at which each EC is detectable (Aksenova & Sogomonyan, 1975). This Error *Detectability Table (EDT)* is the mechanism for prescribing any target error model.



(a)

ERROR DETECTABILITY TABLE

	b_1	b_2	...	b_{ns}	b_{ns+1}	b_{ns+2}	...	b_{ns+no}
Erroneous Case_1	1			1				1
Erroneous Case_2		1						
⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮
Erroneous Case_f		1					1	1

(b)

Figure 1. Example FSM and Error Detectability Table

Detecting all errors requires that at least one state/output bit in each EC be predicted through additional hardware and compared to its actual run-time value. Instead of duplicating the circuit, however, 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 EC, also detects the EC. Therefore, it is possible that a small number of parity functions compacting the state/output bits will be adequate to cover all ECs in a prescribed error model. Using the information in the EDT, the optimization objective of parity-based CED is to *minimize the number of parity bits, k*, that need to be constructed out of the next state/output bits such that all ECs are detected. An EC is detected by a parity tree *if and only if* the parity tree comprises an *odd* number of bits that detect the EC.

Based on the above observations, parity-based CED is rather straightforward, as depicted in the form of a block diagram in Fig. 2. Given an FSM with d inputs, k parity trees are used for lossless compaction of the state/output bits. Combinational logic is employed to predict the values of the k bits that compact the n state/output bits for each FSM transition (i.e. prediction logic in the figure), and a comparator is used to detect any discrepancy. The logic functionality of the parity predictor is realized for each parity function separately, by XORing the functionality of the outputs in its parity group. Hence, the selection of which parity functions to implement for lossless compaction directly controls the cost of the associated parity predictor circuitry. Similar to Zeng et al. (1999), registers are added to hold the output and the predicted parity so that comparison is performed one clock cycle later to detect faults in the state register. Thus, all FSM errors are detected without latency but are reported one clock cycle later.

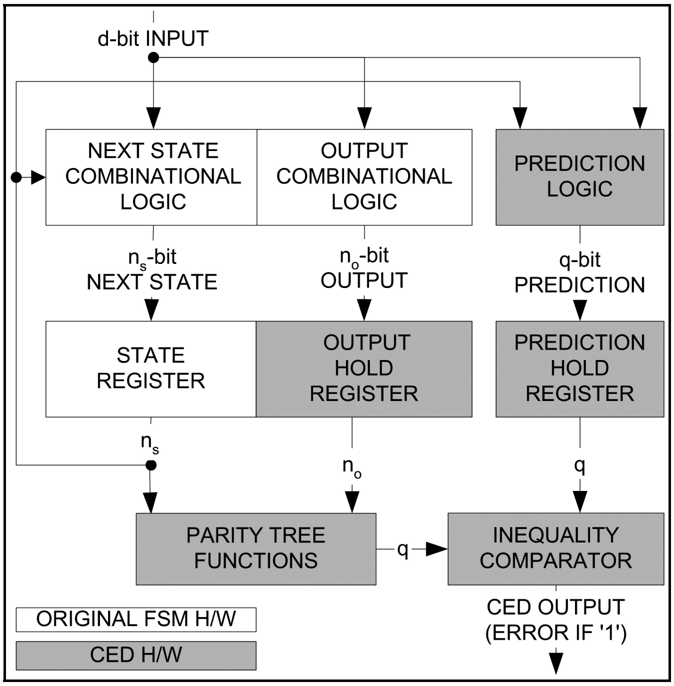


Figure 2. Parity-Based CED Method

Lossless Compaction via ILP

We start by introducing some notation and useful facts that will be repeatedly used throughout this section (Almukhaizim et al., 2006). Let $[x]$ denote the sequence $1, 2, 3, ..., x$ for any non-zero positive integer x. Given a FSM, let m be the total number of inputs of the FSM and let $n = n_s + n_o$ be the total number of outputs of the next state/output combinational logic, which we denote by $\{b^1, b^2, ..., b^n\}$. The set of ECs that need to be detected is denoted by $F = \{EC^1, EC^2, ..., EC^f\}$, where $|F| = f$. The EDT of Fig. 1(b) is stored in a matrix, which we denote by V. The dimensions of matrix V are $f \times n$, and we denote its *(i, j)*-th element by *V(i, j)* for all $i \in [f]$, $j \in [n]$. We remind the reader that for boolean variables x_1, x_2, $x_1 \otimes x_2 = (x_1 + x_2) \bmod 2$. Also, any subset of $\{b^1, b^2, ..., b^n\}$ may be represented by an n-dimensional 0-1 vector, e.g., the subset $\{b^1, b^3, b^4\}$ may be represented by [1 0 1 1... 0].

We first define the entries of the $f \times n$ matrix V, which can be either 0 or 1.

Definition 1: For all $i \in [f]$, $j \in [n]$, *V(i,j)* is set to 1 if and only if the erroneous case EC_i is detectable by the j-th output bit b_j. Otherwise, *V(i,j)* is set to 0.

Our problem may now be stated as follows:

Statement 1: Given a positive integer k, find k subsets $\beta_1, ..., \beta_k$ of $\{b^1, b^2, ..., b^n\}$ such that

$$\mathbf{cov}(\otimes\beta_1) \bigcup \mathbf{cov}(\otimes\beta_2) \bigcup ... \mathbf{cov}(\otimes\beta_k) = F,$$

or report the lack thereof.

Here, $\otimes\beta_l$ (for all $l \in [k]$) denotes the parity tree formed by the next state/output bits in β_l and $\mathbf{cov}(\otimes\beta_l)$ denotes the erroneous cases detected by this parity tree. An erroneous case EC_i is detected by the parity tree formed by the bits in β_l if and only if

$$\left(\sum_{b_y \in \beta_l} V(i, y)\right) \bmod 2 \geq 1$$

The above formula essentially checks whether the XOR of the bits in β_l detects EC_i. Thus, we can check whether the k parity trees (the parity trees corresponding to β_l, for all $l \in [k]$), detect all erroneous cases. The problem may now be stated in matrix notation:

Statement 2: Given a positive integer k, find k n-dimensional 0-1 vectors $\beta^{(1)}, ..., \beta^{(k)}$ such that

$$\sum_{l=1}^{k} [V.\beta^{(l)} \bmod 2] \geq 1_f$$

or report the lack thereof. Here, 1_f is an f-dimensional vector of 1s.

We now remove the *mod* operator by adding new variables:

Statement 3: Given a positive integer k, find vectors $\beta^{(l)},\ r^{(l)},\ w^{(l)},\ l \in [k]$, such that

$$V \cdot \beta^{(1)} = 2 \cdot w^{(1)} + r^{(1)}$$

$$V \cdot \beta^{(2)} = 2 \cdot w^{(2)} + r^{(2)}$$

$$.$$

$$V \cdot \beta^{(k)} = 2 \cdot w^{(k)} + r^{(k)}$$

$$r^{(1)} + ... + r^{(k)} \geq 1_f$$

$$\beta^{(l)} \in \{0,1\}^n$$

$$r^{(l)} \in \{0,1\}^f$$

$$w^{(l)} \in \{0,1,..., \left\lfloor \frac{n}{2} \right\rfloor\}^f,$$

or report the lack thereof.

In order to understand the above constraints, observe that the remainder $r^{(l)}$ is an f-dimensional 0-1 vector denoting whether $EC_1,..., EC_f$ are detected by the parity tree corresponding to $\beta^{(l)}$. We note that $w^{(l)}$ is also an f-dimensional vector that removes the *mod* 2 operation from the ILP formulation. Each element in the sum of the remainders $r^{(l)}$ is required to be at least one, in order to guarantee that all erroneous cases are detected. Given the above *integer program* formulation, the goal is to find a *feasible point*; namely, values for all $r^{(lq)}$, $w^{(lq)}$ and $\beta^{(l)}$ such that *all* the restrictions of statement 3 are satisfied. While the above ILP formulation is NP-Complete, it can be efficiently approximated through a well-known method combining linear program relaxation and randomized-rounding (Raghavan & Thompson, 1987).

Entropy-Driven Parity Tree Selection

The entropy of a multi-output logic function (Cheng & Agrawal, 1990), or the parity predictor in parity-based CED (Almukhaizim et al., 2006), has been shown to correlate with its area cost. Cheng & Agrawal (1990) pioneered the use of entropy for estimating the complexity of a multi-output function, and their observations have been followed by a body of work (Shanbhag, 1997; Macii et al., 1997; Nemani & Najm, 1996, 1998), establishing the correlation between the entropy of a function and the area and power consumption overhead incurred by its implementation. Essentially, smaller values of entropy[1] imply that the function is less random and, hence, its implementation overhead is expected to be low, while values of entropy close to k imply that the function is random and, hence, its implementation overhead is expected to be high.

Low-cost parity functions are identified as depicted in the flowchart in Fig. 3. Once the ILP is solved by identifying a feasible solution **u** of k parity trees that guarantee the detection of all potential errors, its hardware cost is estimated by approximating its entropy. Then, a neighboring feasible solution **v**, namely a solution that preserves detection of all ECs and is "close" to the previous solution in some well-defined proximity metric, is examined. Subsequently, the overhead of the new solution is estimated, once again by approximating its entropy, and a move is accepted or rejected to this neighbor with a probability that depends on the estimated overhead of the new solution. More specifically, "better" solutions are almost always accepted, while "worse" solutions are accepted with a probability that becomes exponentially smaller as their quality decreases. This is essentially the classical Metropolis algorithm, which is the basis of Simulated Annealing techniques (Kirkpatrick et al., 1983) that have been very successful in tackling non-linear optimization problems in various contexts. The randomized selection process is repeated a fixed number of times, and the algorithm terminates after 1000 solutions are examined with no additional reduction to the entropy of the parity predictor. Please refer to (Almukhaizim et al., 2006) for details.

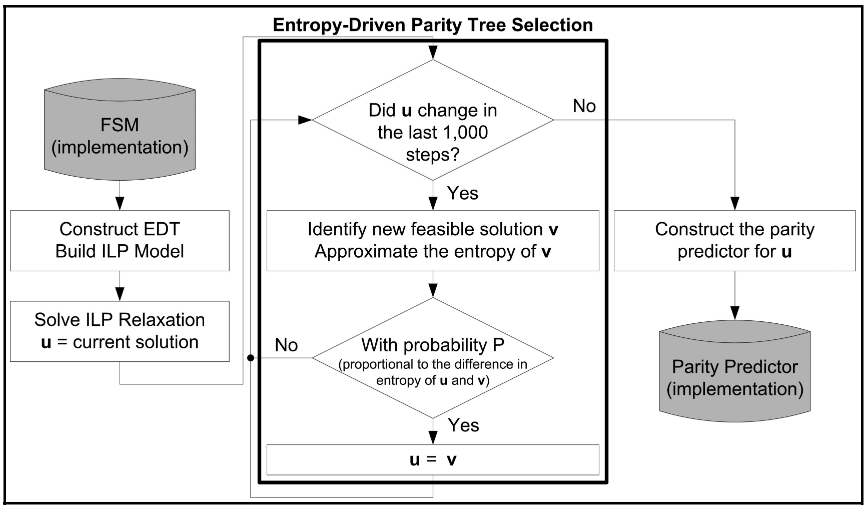

Entropy-Driven Parity Tree Selection
FSM (implementation)
Construct EDT Build ILP Model
Solve ILP Relaxation u = current solution
Did u change in the last 1,000 steps?
No
Yes
Identify new feasible solution v Approximate the entropy of v
With probability P (proportional to the difference in entropy of u and v)
No
Yes
u = v
Construct the parity predictor for u
Parity Predictor (implementation)

Figure 3. Parity-Based CED with Entropy-Driven Parity Tree Selection

(1) Recall that the entropy of a k-output function ranges between 0 and k.

PARITY-BASED CED WITH BOUNDED LATENCY in FSMs

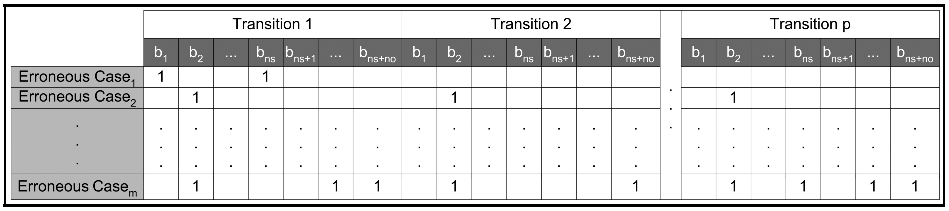
In this section, we extend the parity-based CED method to perform CED with *bounded latency* in FSMs. In essence, the objective of this approach is to reduce the overhead of CED by allowing a small delay between error occurrence and error detection. We emphasize, however, that the worst-case delay is bounded. Thus, while the hardware necessary for performing CED may be reduced, it is still possible to guarantee error detection within the specified latency. Preliminary discussions of the proposed method appear in Almukhaizim et al. (2004)

General Requirements for CED with Bounded Latency

In latency-free CED, 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 source 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 the error source remains active 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 causing 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 (SEUs). For SEU detection, the latency-free parity-based CED method should be employed. Alternatively, SEUs can also be detected with bounded latency if a CED method employs some form of memory, such as the convolutional encoding method proposed by Holmquist & Kinney (1991). Such methods, however, increase the overall cost significantly.

In order to omit immediate detection of an erroneous response, we need to enumerate all erroneous paths of length p, starting from the state where the error is initially activated. A CED mechanism should be capable of detecting the underlying error source in *all* such paths, yet not necessarily during the initial transition. Note that the error source may affect each path in more than one transition, providing more detection opportunities. 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. Similar bounded enumeration methods have been successfully employed in processor validation (Shen & Abraham, 1999).

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 errors 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 restricted error model, we can find the maximum latency of interest by finding the length of the shortest loop on each erroneous FSM and selecting the largest value.



	Transition 1							Transition 2								Transition p						
	b_1	b_2	...	b_{ns}	b_{ns+1}	...	b_{ns+no}	b_1	b_2	...	b_{ns}	b_{ns+1}	...	b_{ns+no}		b_1	b_2	...	b_{ns}	b_{ns+1}	...	b_{ns+no}
Erroneous Case$_1$	1			1																		
Erroneous Case$_2$		1							1						⋮		1					
⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮		⋮	⋮	⋮	⋮	⋮	⋮	⋮
Erroneous Case$_m$		1				1	1		1					1			1		1		1	1

Figure 4. Extended Error Detectability Table for CED with Bounded Latency

Extending the Error Detectability Table for Bounded Latency

Since the error source persists for at least p clock cycles, we do not necessarily need to detect the error during the first transition. Rather, we need to ensure that the selected parity functions will detect an error along every possible path of length p. In this case, the definition of erroneous cases needs to be slightly adjusted. Consider again the FSM shown in Fig. 1(a). For every combination of a *sequence of inputs* $A = a_1, ..., a_p$, where, $a_j \in \{0, ..., 2^m - 1\}$, $j \in 1, ..., p$, and previous state c, $c \in \{0, ..., 2^{n_s} - 1\}$, any error, e, will manifest itself as a difference between the sequence of error-free responses, *GM(A,c)*, $GM(A, c) \in \{0, ..., 2^n - 1\}^p$, and the sequence of erroneous responses, *BM$_e$(A,c)*, $BM_e(A, c) \in \{0, ..., 2^n - 1\}^p$. During each of the p FSM transitions, this difference is detectable in a set of state and output bits b_i, where $i \in 1, ..., n$. The concatenation of these p sets defines an *Erroneous Case, EC (A,c,e)*. Again, several combinations of a transition sequence, *(A,c)*, and error, e, may lead to the same erroneous case, i.e. the same p sets of bits through which the error e may be detected across the p transitions of the sequence *(A,c)*. The union of all erroneous cases, $F = \bigcup \forall_{(A,c,e)}$, is represented in tabular format in an extended error detectability table, as shown in Fig. 4. In this table, super-columns correspond to the p transitions, columns correspond to the n state/output bits, rows correspond to the f erroneous cases, and entries of “1” indicate the state/output bits at which each erroneous case may be detected.

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. 4, where erroneous case *f* 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. 4. 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 bit in every one of the *p* transitions.

Given the extended error detectability table, we are faced with the same optimization problem as in the latency-free case. Essentially, the objective is to select *k* parity trees such that all erroneous cases are detected while the CED overhead is minimized. An erroneous case is detected with bounded latency, *p*, 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*.

Overall Algorithm

In this section, we demonstrate how to extend the integer programming formulation of the latency-free CED method to incorporate bounded detection latency. First, we introduce the necessary modified notation to incorporate latency. Then, we formulate a set of integral constraints that detect all erroneous cases given a fixed number of parity trees within a given latency bound.

The bounded number of FSM transitions allowed for error detection is denoted by *p*, which is a non-zero positive integer. Essentially, $p = 1$ represents the basic latency-free CED scheme, $p = 2$ represents CED with a maximum latency of one clock cycle, $p = 3$ represents CED with a maximum latency of two clock cycles, and so on. The error detectability table of Fig. 4 is stored in a 3-dimensional array, which we denote by *V*. The dimensionality of *V* is $f \times p$; we denote the (i, j, q)-th element of *V* by $V(i,j,q)$ for all, $i \in [f]$, $j \in [n]$, $q \in [p]$. We use the notation $V(:, i, q)$ to denote all elements $V(i,j,q)$ for all $i \in [f]$. Notice that $V(:,j,q)$ is an *f*-dimensional vector; similar definitions may be given for $V(i,:,q)$ and $V(i,j,:)$. We also note that $V(:,j,q)$ corresponds to an $f \times n$ matrix whose (i,j)-th element is the (i,j,q)-th element of *V*. We first define the entries of the $f \times n \times p$ array *V*, which can be either 0 or 1.

Definition 2: For all $i \in [f]$, $j \in [n]$, $q \in [p]$, $V(i,j,q)$ is set to 1 if and only if the erroneous case EC_i is detectable by the *j*-th output bit b_j during the *q*-th FSM transition; Otherwise, $V(i,j,q)$ is set to 0.

Our problem may now be stated as follows:

Statement 4: Given a positive integer k, find k subsets $\beta_1, ..., \beta_k$ of $\{b^1, b^2, ..., b^n\}$ such that

$$\mathbf{cov}(\otimes\beta_1) \bigcup \mathbf{cov}(\otimes\beta_2) \bigcup ... \ \mathbf{cov}(\otimes\beta_k) = F,$$

or report the lack thereof.

Here, $\otimes\beta_l$ (for all $l \in [k]$) denotes the parity tree formed by the next state/ output bits in β_l and $\mathbf{cov}(\otimes\beta_l)$ denotes the erroneous cases detected by this parity tree. An erroneous case EC_i is detected by the parity tree formed by the bits in β_l if and only if

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

The above formula essentially checks whether the XOR of the bits in β_ℓ detects EC_i within q FSM transitions, $q \in [p]$. Thus, we can check whether the k parity trees (the parity trees corresponding to β_ℓ, for all $l \in [k]$), detect all erroneous cases. The problem may now be restated as follows:

Statement 5: Given a positive integer k, find k n-dimensional 0-1 vectors $\beta^{(1)}, ..., \beta^{(k)}$ such that

$$\sum_{l=1}^{k} \left[\sum_{q=1}^{p} \left(\left(\sum_{y:\beta_y^{(l)}=1} V(1, y, q) \right) \bmod 2 \right) \right] \geq 1$$

$$\sum_{l=1}^{k} \left[\sum_{q=1}^{p} \left(\left(\sum_{y:\beta_y^{(l)}=1} V(2, y, q) \right) \bmod 2 \right) \right] \geq 1$$

.

$$\sum_{l=1}^{k} \left[\sum_{q=1}^{p} \left(\left(\sum_{y:\beta_y^{(l)}=1} V(f, y, q) \right) \bmod 2 \right) \right] \geq 1$$

or report the lack thereof.

In order to understand the above constraints, consider the special case $p = 1$, which corresponds to latency-free error detection; notice that, in this special case, V is an $f \times n$ array. If

$$\sum_{l=1}^{k}\left(\sum_{y:\beta_y^{(l)}=1} V(1,y,q) \bmod 2\right) \geq 1$$

there exists some index $\bar{l} \in [k]$ such that

$$\sum_{y:\beta_y^{(\bar{l})}=1}^{V(i,y)} \bmod 2 = 1$$

Thus, the parity tree that consists of the bits that are set to 1 in $\beta_y^{(\bar{l})}$ detects the erroneous case EC_i within the first FSM transition (i.e latency-free). Hence, the constraints of Statement 5 essentially guarantee that *at least* one of the k parity trees corresponding to the vectors, $\beta_y^{(l)}$, $l \in [k]$, detects EC_i within p FSM transitions (i.e. $p-1$ clock cycles). The problem may now be stated in matrix notation:

Statement 6: Given a positive integer k, find k n-dimensional 0-1 vectors $\beta^{(1)},\ldots,\beta^{(l)}$ such that

$$\sum_{l=1}^{k}\left[\sum_{q=1}^{p}\left(V(:,:,q).\beta^{(l)} \bmod 2\right)\right] \geq 1_f$$

or report the lack thereof. Here, 1_f is an f-dimensional vector of 1s.

Notice that $V(:,:,q)$ is an $f \times n$ array denoting the erroneous cases detected by the next state/output bits with latency exactly equal to $q-1$. We now remove the *mod* operator:

Statement 7: Given a positive integer k, find vectors $\beta^{(l)}$, $r^{(lq)}, w^{(lq)}, l \in [k]$, $q \in [p]$, such that

$$\forall q \in [p], V(:,:,q)\cdot\beta^{(1)} = 2\cdot w^{(1q)} + r^{(1q)}$$

$$\forall q \in [p], V(:,:,q)\cdot\beta^{(k)} = 2\cdot w^{(kq)} + r^{(kq)}$$

$$\sum_{q=1}^{p}\left(r^{(1q)} + .. + r^{(kq)}\right) \geq 1$$

$$\beta^{(1)},..,\beta^{(k)} \in \{0,1\}^n$$

$$r^{(lq)} \in \{0,1\}^f$$

$$w^{(lq)} \in \{0,1,\ldots,\left\lfloor \frac{n}{2}\right\rfloor\}^f$$

or report the lack thereof.

In order to understand the above constraints, observe that the remainder $r^{(lq)}$ is an f-dimensional 0-1 vector denoting whether $EC_1, .., EC_f$ are detected by the parity tree corresponding to $\beta^{(l)}$ with latency $q - 1$. We note that the quotient $w^{(lq)}$ is also an f-dimensional vector that removes the *mod* 2 operation. For every $q \in [p]$, each element in the sum of the remainders $r^{(lq)}$, where $l \in [k]$, is required to be at least one, in order to guarantee that all erroneous cases are detected.

In order to identify a feasible point for the integer program, the randomized rounding technique (Raghavan & Thompson, 1987) is utilized to find the integer solution. The above formulation answers the detection problem with bounded latency. Together with the entropy-driven parity tree selection method, a set of parity trees that minimize the overhead of CED with a bounded latency can be found.

EXPERIMENTAL RESULTS

In this section, we evaluate experimentally the area overhead of the proposed parity-based CED with bounded detection latency. Since the proposed method is non-intrusive, i.e. it does not interfere with the state assignment or the implementation of the original circuit, the starting point for our method is the synthesized circuit generated by SIS (Sentovich et al., 1992) using the *rugged* script and a standard library containing only 2-input gates. For the purpose of comparison, we first implement the duplication-based CED method. Then, we construct the EDT for all single errors without latency ($p = 1$) and with bounded latency of one clock cycle ($p = 2$) and two clock cycles ($p = 3$). For the benchmark circuits, we perform exhaustive functional fault simulation using internally developed software, which is built around the fault simulator HOPE (Lee & Ha, 1996) and identifies all pairs of error-free and erroneous responses. Starting from the EDT, we use internally developed software that implements the entropy-driven parity tree selection method with/without latency, and *lpsolve* (Berkelaar, M.) to solve the ILP.

Parity-based CED without Latency

The results of duplication-based CED and parity-based CED are summarized in Table I. Under the first major heading, we provide details about the FSM circuits that were used: name, number of primary inputs, number of state bits and number of primary outputs. Under the second major heading, we provide the results of duplication-based CED: number of parity functions, number of gates, and hardware cost reported by SIS in λ^2, where λ is the smallest feature size. Under the third (fourth) major heading, the same information is reported for latency-free parity-based CED without (with) entropy-driven selection of parity trees. The use of the entropy-driven parity tree selection method increases the average area reduction, over the minimum parity tree count selection method in (Almukhaizim et al., 2004), from 25% to 40%. These results corroborate that parity-based CED significantly outperforms the duplication-based approach.

For certain highly-random parity functions, the cost of a single complex parity function may require the same or more area than a larger number of output functions. This is the case for *dk16*, where the cost of parity-based CED (with or without bounded detection latency) is higher than the cost of duplication-based CED; such example illustrates that duplication-based CED is more effective than parity-based CED in circuits with a highly-regular implementation.

Table I. Parity-Based CED without Latency ($p = 1$)

Circuit Details		Duplication-Based CED			Parity-Based CED (Almukhaizim et al., 2004)			Proposed CED Method		
Name	I/S/O	Parity	Gates	Area	Parity	Gates	Area	Parity	Gates	Area
cse	7/4/7	11	196	256128	5	131	171680	5	97	127136
donfile	2/5/1	6	97	128064	4	57	74704	4	56	75632
dk16	2/5/3	8	240	316448	6	323	428736	6	294	389760
dk512	1/4/3	7	74	96048	4	79	104400	6	61	80272
keyb	7/5/2	7	228	298352	5	82	107648	5	67	86304
pma	8/5/8	13	347	453792	6	186	243136	6	138	180496
sse	7/4/7	11	131	178640	5	80	104864	5	59	77488
styr	9/5/10	15	413	547056	8	217	287216	8	162	213904
s1	8/5/6	11	167	217616	5	121	156832	5	61	77024
s1a	8/5/6	11	153	199056	6	96	124816	5	61	77024
s27	4/3/1	4	20	25056	3	15	18096	3	7	8352
s386	7/7/6	13	123	158688	4	83	105328	4	72	92336
tav	4/2/4	6	28	34336	4	31	39440	4	26	33408
tbk	6/5/3	8	146	190240	5	160	207872	5	151	198592
tma	7/5/6	11	219	285824	5	130	169824	7	131	172144

Table II. Parity-Based CED with Bounded Latency

Circuit Name	Proposed Method ($p = 2$)			Proposed Method ($p = 3$)		
	Parity	Gates	Area	Parity	Gates	Area
cse	6	91	119712	5	93	122032
donfile	4	56	75632	4	56	75632
dk16	7	257	342896	7	257	342896
dk512	6	65	86304	5	63	83984
keyb	4	64	83984	5	70	90016
pma	6	129	170288	5	125	165648
Sse	6	50	68208	6	52	70064
styr	8	162	213904	5	151	200448
s1	5	61	77024	6	56	71456
s1a	6	63	78880	6	63	78880
s27	3	7	8352	3	7	8352
s386	4	72	92336	4	72	92336
tav	3	25	31552	3	25	31552
tbk	5	151	198592	5	151	198592
tma	4	131	150800	4	131	150800

Parity-based CED with Bounded Latency

The results of parity-based CED with bounded detection latency are summarized in Table II. The second (third) major heading indicates the cost of parity-based CED with $p = 2$ ($p = 3$). The results indicate that the addition of one clock cycle of bounded latency reduces the area overhead of the parity prediction logic by 4% over that of the parity logic required for latency-free detection, and by 10% over the area cost in (Almukhaizim et al., 2004). For certain circuits such as *cse* and *s27*, the area cost of the parity predictor is reduced by 50%. Further increase of the latency bound to two clock cycles, yields an additional 3% reduction in the area overhead. The benefits of adding latency diminish as latency increases. In smaller FSMs, errors result in a large number of self-loops. For example, this is the case for circuits *donfile, s27, s386* and *tbk*. As the FSM becomes larger, self-loops are less frequent and the benefits of increasing latency are more significant. This is for example the case for circuits *dk16*, *pma*, and *tma*. The results of the proposed method are compared to the results of previously-proposed parity-based CED methods (Almukhaizim et al., 2004, 2006) in Fig. 5. Overall, the *average* results support the efficacy of the proposed method in identifying the least-cost parity prediction circuitry.

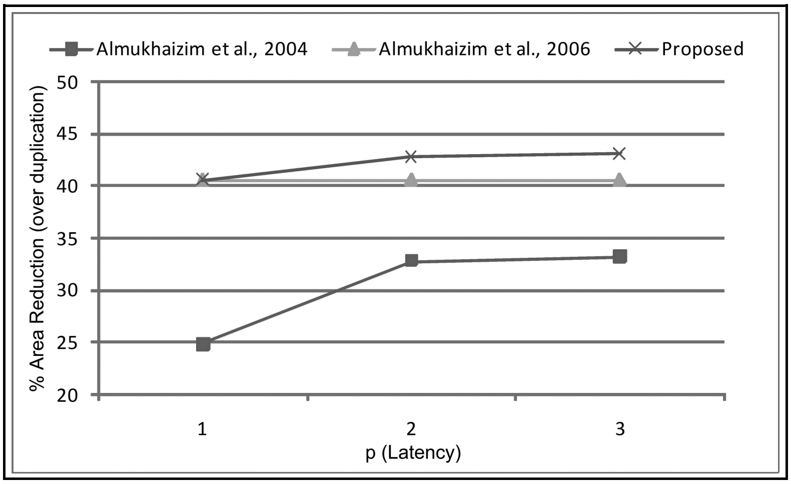


Figure 5. Comparison Between Proposed Method and Previously-Proposed Parity-Based CED Methods

CONCLUSIONS

We present a non-intrusive method for identifying appropriate parity trees for performing CED with bounded latency in FSMs. Given a restricted error model, the outputs of the circuit are compacted through parity trees and compared to the expected values that are computed through a parity predictor. Our method allows detection of errors either without latency, or with bounded latency of a given number of clock cycles. We formulate the problem of identifying the minimal number of required parity bits as an integer program and we devise an algorithm based on linear program relaxation and randomized rounding to solve it. Moreover, we guide the selection of parity trees based on their *entropy*, in order to reduce the implementation cost of the parity predictor circuitry. Experimental results confirm that a small number of parity functions are typically adequate to detect all errors in a prescribed model. Thus, the incurred overhead is significantly smaller than duplication-based CED, and is further reduced by allowing a small bounded latency in the detection of errors.

REFERENCES

Aksenova, G., & Sogomonyan, E. 1975. Design of self-checking built-in check circuits for automata with memory. Automation and Remote Control, **36(7)**, 1169-1177.

Almukhaizim, S., Drineas, P., & Makris, Y. 2004. On Concurrent Error Detection with Bounded Latency in FSMs. Proceedings of the Design Automation and Test in Europe Conference, pp. 596-601, Paris, France.

Almukhaizim, S., Drineas, P., & Makris, Y. 2005. Compaction-based concurrent error detection for digital circuits. Microelectronics Journal, **36(9)**, 856-862.

Almukhaizim, S., Drineas, P., & Makris, Y. 2006. Entropy-driven parity-tree selection for low-overhead concurrent error detection in finite state machines. IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems, **25(8)**, 1547-1554.

Avizienis, A., & Kelly, J. P. J. 1984. Fault tolerance by design diversity: Concepts and experiments. IEEE Computer, **17(8)**, 67-80.

Cheng, K.-T., & Agrawal, V. D. 1990. An entropy measure for the complexity of multi-output boolean functions. Proceedings of the ACM/IEEE Design Automation Conference, pp. 302-305, Orlando, FL, USA.

Danilov, V. V., Kolesov, N. V., & Podkopaev, B. P. 1975. An algebraic model for the hardware monitoring of automata. Automation and Remote Control, **36(6)**, 984-991.

Das, D., & Touba, N. A. 1999. Synthesis of circuits with low-cost concurrent error detection based on Bose-Lin codes. Journal of Electronic Testing: Theory and Applications, **15(2)**, 145-155.

Dhawan, S., & Vries, R. C. D. 1988. Design of self-checking sequential machines. IEEE Transactions on Computers, **37(10)**, 1280-1284.

Drineas, P., & Makris, Y. 2003. SPaRe: selective partial replication for concurrent fault detection in FSMs. IEEE Transactions on Instrumentation and Measurement, **52(6)**, 1729-1737.

Gossel, M., & Graf, S. 1993. Error Detection Circuits. McGraw-Hill, NY, USA.

Holmquist, L. P., & Kinney, L. L. 1991. Concurrent error detection for restricted fault sets in sequential circuits and microprogrammed control units using convolutional codes. Proceedings of the IEEE International Test Conference, pp. 926-935, Nashville, TN, USA.

Jha, N. K., & Wang, S. J. 1993. Design and synthesis of self-checking VLSI circuits. IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems, **12(6)**, 878-887.

Kirkpatrick, S., Gelatt, C. D., & Vecchi, M. P. 1983. Optimization by Simulated Annealing. Science, Number 4598, pp. 671-680.

Lee, H. K., & Ha, D. S. 1996. HOPE: An efficient parallel fault simulator for synchronous sequential circuits. IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems, **15(9)**, 1048-1058.

Leveugle, R., & Saucier, G. 1990. Optimized synthesis of concurrently checked controllers. IEEE Transactions on Computers, **39(4)**, 419-425.

Macii, E., Pedram, M., & Somenzi, F. 1997. High-level power modeling, estimation and optimization. Proceedings of the ACM/IEEE Design Automation Conference, pp. 504-511.

Meyer, J. F., & Sundstrom, R. J. 1975. On-line diagnosis of unrestricted faults. IEEE Transactions on Computers, **24(5)**, 468-475.

Mitra, S., & McCluskey, E. J. 2000. Which concurrent error detection scheme to choose? Proceedings of the IEEE International Test Conference, pp. 985-994, Atlantic City, NJ, USA.

Mohanram, K., Sogomonyan, E. S., Gossel, M., & Touba, N. A. 2003. Synthesis of Low-Cost Parity-Based Partially Self-Checking Circuits. Proceedings of the IEEE International On-line Testing Symposium, pp. 35-40, Kos Island, Greece.

Nemani, M., & Najm, F. 1996. Towards a high-level power estimation capability. IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems, **15(6)**, pp. 588-598.

Nemani, M., & Najm, F. N. 1998. Delay estimation of VLSI circuits from a high-level view. Proceedings of the ACM/IEEE Design Automation Conference, pp. 591-594, San Francisco, CA, USA.

Parekhji, R. A., Venkatesh, G., & Sherlekar, S. D. 1995. Concurrent error detection using monitoring machines. IEEE Design and Test of Computers, **12(3)**, 24-32.

Piestrak, S. J. 1996. Self-checking design in Eastern Europe. IEEE Design and Test of Computers, **13(1)**, 16-25.

Raghavan, P., & Thompson, C. 1987. Randomized rounding: A technique for provably good algorithms and algorithmic proofs. Combinatorica, **7(4)**, 365-374.

Robinson, S. H., & Shen, J. P. 1992. Direct methods for synthesis of self-monitoring state machines. Proceedings of the International Symposium on Fault Tolerant Computing, pp. 306-315, Boston, MA, USA.

Saluja, K. K., Sharma, R., & Kime, C. R. 1988. A concurrent testing technique for digital circuits. IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems, **7(12)**, 1250-1260.

Sentovich, E., Singh, K., Lavagno, L., Moon, C., Murgai, R., Saldanha, A., Savoj, H., Stephan, P., Brayton, R. K., & Sangiovanni-Vincentelli, A. L. 1992. SIS: A system for sequential circuit synthesis. Tech. Rep. UCB/ERL M92/41, EECS Department, University of California, Berkeley. URL http://www.eecs.berkeley.edu/Pubs/TechRpts/1992/2010.html

Shanbhag, N. 1997. A mathematical basis for power-reduction in digital VLSI systems. IEEE Transactions on Circuits and Systems, Part II, **44(11)**, 935-951.

Sharma, R., & Saluja, K. K. 1988. An implementation and analysis of a concurrent built-in self-test technique. Proceedings of the International Symposium on Fault Tolerant Computing, pp. 164-169, Tokyo, Japan.

Shen, J., & Abraham, J. A. 1999. Verification of processor microarchitectures. Proceedings of the IEEE VLSI Test Symposium, pp. 189-194, San Diego, CA, USA.

Sogomonyan, E. 1974. Design of built-in self-checking monitoring circuits for combinational devices. Automation and Remote Control, **35(2)**, 280-289.

Sogomonyan, E. S. 1970. The design of discrete devices with diagnostics in the course of operation. Automation and Remote Control, **31(11)**, 1854-1860.

Touba, N. A., & McCluskey, E. J. 1997. Logic synthesis of multilevel circuits with concurrent error detection. IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems, **16(7)**, 783-789.

Voyiatzis, I., Paschalis, A., Nikolos, D., & Halatsis, C. 1998. R-CBIST: An effective RAM-based input vector monitoring concurrent BIST technique. Proceedings of the IEEE International Test Conference, pp. 918-925, Washington, DC, USA.

Zeng, C., Saxena, N., & McCluskey, E. J. 1999. Finite state machine synthesis with concurrent error detection. Proceedings of the International Test Conference, pp. 672-679, Atalantic City, NJ, USA.

M. Berkelaar. Linear programming solver. Software package available from: URL http://www.cs.sunysb.edu/algorith/.

Submitted : *25/8/2008*
Revised : *7/6/2009*
Accepted : *11/6/2009*

استخدام ازدواجية النتيجة للتعرف على الأخطاء خلال فترة زمنية محددة من عمل الدوائر الكهربائية ذات الذاكرة

[*] صبيح المخيزيم ، [**] بيتروس درينياس و [***] يورجوس ماكريس

[*] قسم هندسة الكمبيوتر – جامعة الكويت – ص. ب. 5969 – الصفاة 13060 – الكويت.

[**] علوم الكمبيوتر – معهد رينسيلار للفنون التطبيقية – الولايات المتحدة الأمريكية.

[***] قسم الهندسة الكهربائية وعلوم الكمبيوتر – جامعة يال – الولايات المتحدة الأمريكية.

خلاصة

نطور في هذا العمل طريقة سابقة للتعرف على وجود أخطاء أثناء عمل الدوائر الكهربائية ذات الذاكرة بحيث يتم التعرف على الخطأ خلال فترة زمنية محددة مسبقاً. الطريقة المقترحة تعتمد على ضغط نتائج الذاكرة الكهربائية باستخدام ازدواجية النتيجة (فردية أو زوجية) ومقارنتها مع النتيجة المضغوطة الصحيحة التي يتم إنتاجها باستخدام دائرة أخرى إضافية. يتم اختيار عملية الضغط المستخدمة بهدف تقليل تكلفة الدائرة الإضافية، وتمثل بنموذج برنامج معادلات ذات حلول صحيحة وحله عن طريق استرخاء شروط الحل الصحيح مع تقريب النتائج بشكل عشوائي. ثم نطور النموذج المقترح لتمثيل معادلات التعرف على الأخطاء خلال فترة زمنية محددة مما يقلل من تكلفة الدائرة الإضافية على حساب تأخير بسيط لدى التعرف على الأخطاء. وتبين نتائج التجارب العملية أن إضافة فترة زمنية للتعرف على الأخطاء تؤدي إلى تقليل تكلفة الدائرة الكهربائية الإضافية بشكل كبير.