# Lowering Power Consumption in Concurrent Checkers via Input Ordering

Kartik Mohanram, *Member, IEEE,* and Nur A. Touba, *Member, IEEE*

***Abstract*—This paper presents an efficient and scalable technique for lowering power consumption in checkers used for concurrent error detection. The basic idea is to exploit the functional symmetry of concurrent checkers with respect to their inputs, and to order the inputs such that switching activity (and hence power consumption) in the checker is minimized. The inputs of the checker are usually driven by the outputs of the function logic and check symbol generator logic—spatial correlations between these outputs are analyzed to compute an input order that minimizes power consumption. The reduction in power consumption comes at no additional impact to area or performance and does not require any alteration to the design flow. It is shown that the number of possible input orders increases exponentially in the number of inputs to the checker. As a result, the computational cost of determining the optimum input order can be very expensive as the number of inputs to the checker increases. This paper presents a very effective technique to build a reduced cost function to solve the optimization problem to find a near optimal input order. It scales well with increasing number of inputs to the checker, and the computational costs are independent of the complexity of the checker. Experimental results demonstrate that a reduction in power consumption of 16% on the average for several types of checkers can be obtained using the proposed technique.**

***Index Terms*—Concurrent checker, concurrent error detection, input ordering, low power, online testing.**

## I. Introduction

As HIGH-DENSITY, low-cost, high-performance computing devices become more ubiquitous, there is an increased necessity to address the reliable operation of such systems. This is because as process technology scales and feature sizes shrink, integrated circuits will become increasingly susceptible to temporary faults. Several factors, including high operating frequencies, low voltage levels, small noise margins, and reduced logic depth contribute to this increased susceptibility to such faults. Although these faults cause no permanent damage, they can severely limit the reliability (and availability) of electronic systems. Temporary faults include those caused by crosstalk, substrate and power supply noise, charge sharing, etc., and pose a significant challenge to ensuring signal integrity even in present day deep submicron process technologies. In addition, current studies indicate that circuits will become increasingly sensitive to temporary faults caused by terrestrial cosmic rays and alpha particles (that originate from impurities in the packaging materials), and that this will result in unacceptable soft error rates even in mainstream commercial electronics [2], [31].

One way to detect errors caused by temporary faults is to use concurrent error detection (CED) circuitry to monitor the outputs of a circuit for the occurrence of an error. If an error is detected, then the system can recover, thereby preventing a failure. The theory and practice of the design of digital systems with CED is well documented [7], [18], [21], [26]. The use of CED techniques in mission-critical application environments, where dependability and data integrity are of importance, has several advantages:

1) circuits with CED have the capability to detect both temporary and permanent faults;
2) circuit-level techniques for CED permit early detection and containment of errors before they can propagate to other parts of the system and corrupt data;
3) use of CED not only reduces the complexity but also increases the effectiveness of system-level and application-level fault tolerance features.

Thus, CED techniques, in conjunction with high-level software-implemented error detection and correction schemes are ideal for use in situations where early detection of errors is critical for preserving the state of the system and maintaining data integrity.

High dependability comes at a cost, however, since CED schemes impose extra overhead (area, timing, and power) on the design. The primary focus of most CED methods developed in literature has been to guarantee very high levels of dependability, with overhead cost as a secondary concern. Such overhead costs are, however, not acceptable in the context of high-volume mainstream applications. Whereas the necessity to incorporate CED schemes in at least the critical sections of such designs cannot be stressed enough, there is a strong need for techniques that will make the overhead costs of incorporating CED acceptable. Thus, meeting dependability requirements with minimal overhead costs, especially for the high-volume, low-cost mainstream application market, is a significant challenge facing the research community.

Considerable effort has been directed toward reducing the area and delay penalties associated with CED. There are very efficient schemes for CED in circuits with regular structures such as adders [6], [17], programmable logic arrays (PLAs) [1], [11], multipliers [21], etc. Although the problem is much more difficult for control logic due to its typical irregular multilevel structure, several automated logic-synthesis techniques to design multilevel circuits with CED have been proposed. These techniques mostly focus on reducing area overhead

Manuscript received July 1, 2003; revised December 12, 2003. This work was supported in part by the National Science Foundation (NSF) under Grant MIP-9702236 and in part by a grant from the Hewlett-Packard Company.

K. Mohanram is with the Department of Electrical and Computer Engineering, Rice University, Houston, TX 77005 USA (e-mail: kmram@rice.edu).

N. A. Touba is with the Computer Engineering Research Center, Department of Electrical and Computer Engineering, University of Texas, Austin, TX 78712 USA (e-mail: touba@ece.utexas.edu).

Digital Object Identifier 10.1109/TVLSI.2004.836318

without compromising the capability to detect all errors under the assumed fault model. They either modify the circuit to have only unidirectional faults [8], [23], or constrain the circuit during synthesis to control the maximum number of erroneous bits by restricting fanout [3], [4], [29]. Reductions in power consumption are obtained because 1) a highly area efficient implementation for the CED circuitry is synthesized or 2) a less expensive CED scheme is chosen to meet area constraints that results in lower power consumption. Power, for long a concern confined to the realm of portable systems' design, is today a first-order factor influencing integrated circuit design at all levels of the design flow [14]. Power can also cause reliability concerns through electromigration, joule heating, and hot-electron degradation effects.

Some first work toward reducing power consumption in checkers used for CED was presented by Favalli and Metra in [5]. Their technique exploits the observation that almost all checkers used for CED are functionally symmetric with respect to their inputs, i.e., the inputs to the checker can be connected in any arbitrary order. They present an input ordering methodology that uses binary decision diagrams (BDDs) for function representation and manipulation which may not scale well with the problem size. This paper presents a new technique for input ordering that is more efficient and scalable. The proposed input ordering algorithm can be used to reduce power consumption in checkers used for CED. It exploits the functional symmetry and the spatially correlated nature of the inputs to the checker, to order the inputs such that switching activity, and hence, power consumption in the checker is reduced. The main advantage of this approach is that there are no hardware costs as no modifications are required either to the checker or the design. The only cost is the time for computing the input order that minimizes power consumption in the checker. It is shown that the number of possible input orders is exponential in the number of inputs to the checker. As a result, the optimization problem of determining the optimal input order is computationally expensive even for a small number of inputs. A fast heuristic technique to determine an input order that is near optimal with respect to reduction in power consumption is presented. It scales well with the number of inputs to the checker, and has a near constant runtime that is independent of the complexity of the checker (for the same number of inputs). Some preliminary results have been published in [13].

The paper is organized as follows. In Section II, the problem addressed in this paper is introduced in greater detail with a review of some previous research in this area. It is also proven that the number of possible input orders is exponential in the number of inputs to the checker. In Section III, possible solutions are discussed in greater detail with the aid of an example. In Section IV, the proposed solution to the problem is discussed with examples. In Section V, implementation details and experimental results for several benchmark circuits and three types of checkers (parity, two-rail code, and Berger code) are presented. In Section VI, the conclusion is provided.

## II. Concurrent Checkers and Input Ordering

Conventional schemes to design circuits with CED based on error-detecting codes such as parity, duplication, and compare, etc., employ checkers to monitor the outputs for the occurrence of an error. Fig. 1 shows the structure of a circuit that has CED capability. Based upon the scheme chosen for CED, the check symbol generator can be a copy of the original circuit (duplication and compare), parity prediction logic, codeword generator (e.g., for Berger or Bose-Lin codes), etc. The check symbol generator generates check bits and the concurrent checker determines if they (outputs and check bits together) form a codeword. The checker usually satisfies the totally self-checking property [27]. Checkers can have an adverse affect on timing (i.e., increase the clock cycle time). Hence, they are usually pipelined by adding latches right before the checker. By deferring the checker operation to the next clock cycle, the performance of the design remains unaltered at the cost of some latency on when the error is detected. An error occurring in one clock cycle is detected in the next clock cycle. Generally, this is early enough to prevent data corruption in most applications. Examples of commonly used checkers include parity checkers, two-rail code checkers, Berger code checkers, m-out-of-n checkers, etc.

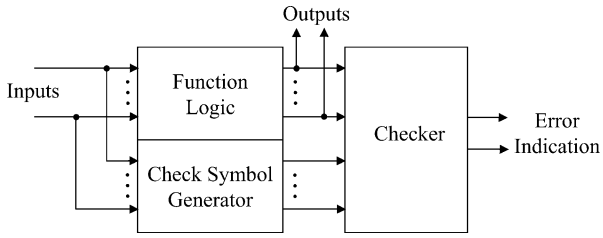



Fig. 1. Block diagram for conventional CED.

As indicated in Fig. 1, concurrent checkers for error detecting codes usually derive their inputs from the primary outputs of the function logic and the check symbol generator. These outputs can be connected to the inputs of the checker in any order, since most checkers have a regular structure and are functionally symmetric with respect to their inputs. In other words, the functionality of the checker is insensitive to various permutations (orderings) of the inputs.

In the presence of spatial and temporal correlations in the primary outputs of the circuit driving the checker (and the check symbol generator, where applicable), the input ordering presented to the checker can have a significant effect on power consumption in the checker. Correlation between vectors is of two types—spatial and temporal. Spatial correlation refers to the correlation between pairs of bits in the same vector, whereas temporal correlation refers to the correlation between pairs of vectors, spaced one or more cycles apart. Power estimation techniques usually make the spatial independence assumption about the primary inputs to a circuit and neglect spatial correlations between them [16]. A recent study of industrial circuits [24] evaluated the accuracy of the correlation assumptions made by several power-estimation methods. Large inaccuracies in total switching activity are reported when spatial correlation between signals is neglected. Assuming spatial independence provides a very conservative estimate of power consumption for checkers. It also precludes the possibility that the inputs to the checker can be ordered to reduce power consumption. Given their symmetry, the grouping of the inputs to the checker can affect power

consumption to a considerable extent due to spatial correlations in the inputs. In this paper, we address the problem of input ordering to checkers to minimize power consumption and present an algorithm that achieves this by taking spatial input correlations into consideration. Experimental results for parity, two-rail code, and Berger code checkers over a wide range of benchmark circuits are presented to demonstrate the effectiveness of this technique.

### A. Previous Work

In [5], a technique to reduce the power consumption in tree-structured two-rail code checkers was presented. Checkers were processed on a level by level basis from the inputs, identifying an input order that reduced transition probabilities at the outputs for each level. Since the outputs constitute the inputs to the next level for a tree-structured checker, the observation used is that a reduction in the transitions at the outputs of a level also reduce transitions at the outputs of cascaded blocks in successive levels.

Whereas the problem addressed in this paper is identical to that addressed in [5], the method proposed in [5] does not scale well as the number of inputs to the checker increases (since it involves an exact computation of signal and transition probabilities using BDD-based symbolic techniques). By working with spatial correlations and cost functions that capture this information, the technique presented in this paper provides a significant improvement in computational complexity. In addition, the heuristic used to evaluate input orders in [5] ranked pairs of inputs for selection in a greedy, linear fashion. This is disadvantageous from both a computational and quality of solution perspective as discussed in Section IV-C. The techniques presented in this paper allow for a tradeoff between complexity of computation and the quality of the final solution by using a novel cost function for the optimization problem.

### B. Compatibility With Other Techniques for Low-Power Checker Design

This section discusses how the proposed technique is not only different from, but also compatible with other techniques (based on reordering, resizing, and synthesis) for the design of low-power checkers.

Reordering techniques to reduce power consumption in CMOS gates work at the input and transistor levels. Input reordering methods permute only the inputs to the gates, leaving the actual realization of the gate untouched. Transistor reordering methods modify the order in which series transistors are connected in a complex CMOS gate, in addition to input reordering. These methods list all possible configurations of a complex CMOS gate and evaluate them for power consumption. The number of configurations that are evaluated is usually small. There is usually a delay tradeoff involved in such techniques, since reordering can move late arriving inputs farther away from the output of the gate contributing to an increase in delay. In [22], a multipass transistor reordering algorithm, with linear time complexity per pass, that converges to a solution in a small number of passes was presented. In [15], an algorithm that includes transitions at the internal nodes of a complex CMOS gate to derive the optimal configuration was presented. Transistor resizing techniques, resize transistors subject to delay constraints to reduce power consumption. These techniques compute the slack at each gate in the circuit and process those with a positive slack. The sizes of the transistors in such gates are reduced until the slack reaches zero or the transistors reach minimum size. In [28], an algorithm that combines both the input reordering and transistor resizing approaches is presented. These ideas were used in [12] to reduce the power-delay product of two-rail code checkers by optimal sizing of the checker's transistors.

The proposed approach differs from such reordering methods in several ways. First, previous methods target general circuits and focus on input and transistor reordering at the individual gate level, whereas the proposed method targets checker circuits and focuses on input reordering at the module level. The magnitude of the problem addressed here is larger, since $n$ inputs to a module imply $n!$ possible permutations, however, the potential for improvement is much greater. Even after structural symmetry is accounted for, the number of distinct permutations renders prohibitive the costs of exhaustive enumeration and evaluation—this is discussed in greater detail in Section II-C. In addition, the proposed method differs from general techniques such as [9] and [12] for the synthesis of low-power checkers since it does *not* consider the design of the checker in isolation from the circuit that drives it.

The proposed approach can be used to obtain an optimal permutation that reduces power consumption in the checker. The reordering, resizing, and checker synthesis techniques described above can be used independently—completely or partially—to obtain further reductions in power consumption, especially in the presence of structural asymmetries in the checker. The proposed approach thus complements other low-power synthesis techniques, whether they target general circuits or checkers.

If the checkers are pipelined, all the inputs to the checker are ready at the same instant. If pipelining is absent, an extra dimension is added to the complexity of the problem. Even in the most balanced design, different paths have different delays, and the inputs to the checker will be ready at different times. This may produce glitches in the checker since its inputs are not ready at the same instant. The possible increase in power consumption due to this extra switching activity is not addressed here. Note also, that whereas the proposed technique reduces dynamic switching power in the checker, it does not reduce the power due to leakage in the checker.

### C. Exponential Complexity

In this section, it is shown that for a parity tree with $n$ inputs, the number of unique permutations after structural symmetries are accounted for is exponential in $n$. Consider the elementary example of a parity tree with four inputs in Fig. 2.

The number of unique permutations in the absence of any structural symmetries in the balanced parity tree for the parity checker is 4! However, in the presence of structural symmetries at each level of logic in the checker, the number of unique permutations is actually three. This is shown in greater detail in Fig. 2, where the different permutations are grouped on the basis of symmetry observed at logic levels of the checker. It is clear that just three unique permutations need to be evaluated to arrive at the optimal input order that minimizes power consumption.

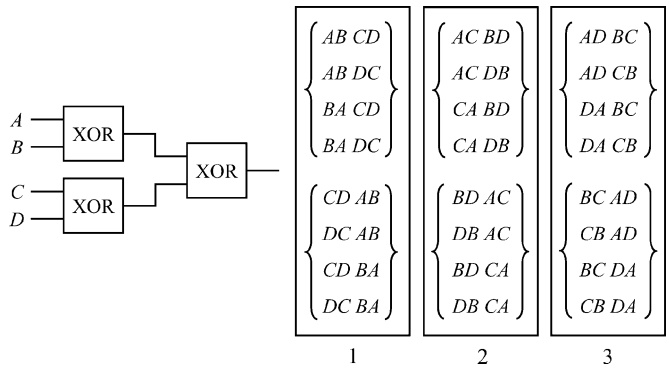



Fig. 2. Structural symmetry in parity checkers.

We now prove that for a parity checker with $n$ inputs realized using 2-input XOR gates, the number of unique input orders is exponential in $n$.

*Lemma 1:* For a parity tree with $n$ inputs, the number of unique permutations is exponential in $n$ for all $n \geq 8$.

*Proof:* Consider the XOR parity tree for $n$ inputs. Let $i$ be the largest power of $2 \leq n$, i.e., let $i$ equal $2^k$ such that $n/2 < i \leq n$. Without loss of generality, we will show that the number of unique permutations for $i$ inputs is exponential in $i$. Since the number of unique permutations is a monotonically increasing function in the number of inputs to the checker, and since $i > (n/2)$, it follows that the number of permutations for arbitrary $n$ is exponential in $n$. The number of 2-input XOR gates in the balanced parity tree for $i$ inputs is $i-1$. Using the notion of structural symmetry as explained above, the total number of unique permutations (input orders) is given by

$$\frac{i!}{\binom{2}{1}^{i/2}\binom{2}{1}^{i/4}\binom{2}{1}^{i/8}\cdots\binom{2}{1}^{i/i}} = \frac{i!}{(2)^{i/2}(2)^{i/4}(2)^{i/8}\cdots(2)^{i/i}}.$$

The expression in the denominator can be further simplified by the substitution $i = 2^k$ as follows:

$$\frac{i!}{(2)^{2^k/2}(2)^{2^k/4}(2)^{2^k/8}\cdots(2)^{2^k/2^k}} = \frac{i!}{\prod_{j=1}^{k} 2^{2^k/2^j}} = \frac{i!}{2^{\sum_{j=0}^{k-1} 2^j}} = \frac{i!}{2^{2^k-1}} = \frac{i!}{2^{i-1}}.$$

For $i \geq 8$, $i! \geq 2 \cdot 4^{i-1}$ and, hence, the above expression is greater than $2^i$. This implies that the number of unique input orders is exponential in the number of inputs. A similar analysis can be performed for two-rail code checkers, since the standard design for a two-rail code checker works with pairs of outputs of the driver circuit at a time [21].

Berger codes are a class of systematic unidirectional error detecting codes [21]. Berger code checkers are based on parallel ones counters or sorting networks in practice. We focus on ordering the inputs to the parallel ones counter to reduce power consumption, since they have been shown to be smaller, faster, and lower on power consumption as the number of inputs increases [20]. Full adders (used to implement parallel ones counters) are usually not structurally symmetric with respect to all their inputs. Hence, if permutations at higher levels are ignored, the number of equivalent permutations is bounded by the following expression for logic level 1:

$$\frac{i!}{(3!)^{i/3}}.$$

It can be shown by an analysis similar to that above for parity checkers that the number of input orders is exponential in the number of inputs to a Berger code checker for all $n \geq 9$, since $i! \geq (3!)^{(i/3)} \cdot 2^i$ for $i \geq 9$.

## III. Proposed Methodology

We use parity codes as an example to illustrate the key idea of our contribution. We present a small example on how spatial correlations in the inputs can affect power consumption in the checker. Consider the Boolean functions $F_1, F_2, \ldots$, and $F_7$ in Fig. 3. These (in some permutation) drive the inputs $x_1, x_2, \ldots$, and $x_7$ of the parity tree shown in Fig. 4.

A naïve approach to solving the problem of determining the optimum ordering of the inputs to the checker would be to exhaustively enumerate all unique input orders and to compute the exact power consumption for each of the possible solutions. The best input order can then be chosen. For the example in Fig. 4, the optimum input order obtained by exhaustive enumeration has a power consumption of 106 units. The computational costs of this method are exorbitant even for small values of $n$, since the number of possible input orders is exponential in $n$.

We propose a simulated annealing approach to solve the optimization problem of determining the best input order. Simulated annealing that uses the Metropolis Monte Carlo algorithm and a logarithmic cooling schedule is used [10]. We present two methods, both of which use the same simulated annealing framework, but differ in the complexity of the chosen cost function.

### A. Exact Simulation Method

The first method, which is computationally expensive, uses the exact routine to calculate the power consumption in the checker as the cost function. This involves the use of a power estimator that simulates the checker using an output trace of the circuit driving the checker and computes the transitions at each of the internal nodes of the checker to estimate power. We term this method the "exact simulation method." For the example in Fig. 4, the optimum input order returned by the exact simulation method has a power consumption of 106 units. However, making a call to the power estimator for each permutation encountered during the simulated annealing routine is very expensive. The total number of calls is equal to the number of input orders encountered per iteration multiplied by the number of times the temperature is reduced during the simulated annealing routine. This can result in very high computational costs as the number of inputs to the checker increases.

Whereas one approach that trades accuracy for speed would be to run the power estimation algorithm with fewer trace vec-

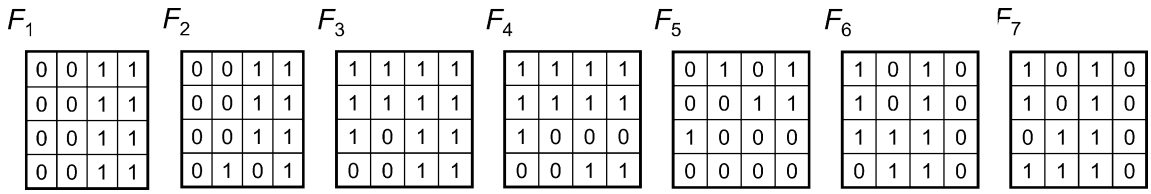



Fig. 3. Boolean functions $F_1, F_2, \ldots$ and $F_7$.

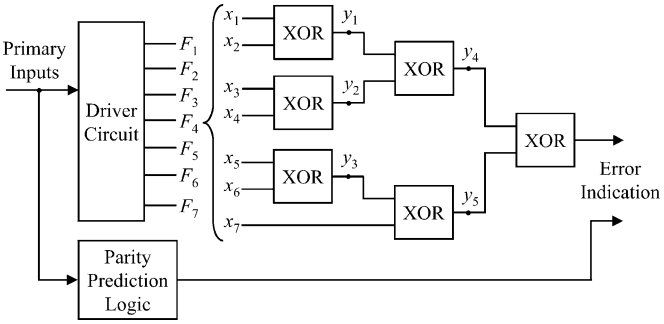



Fig. 4. Example circuit with parity encoded outputs.

tors, the computational costs are still very high as the number of inputs to the checker increases. An alternate approach that uses a simple cost function within the same simulated annealing framework to yield near optimal results is described in the next section.

## IV. Spatial Correlation Estimation Method

The second algorithm proposed in this paper uses the same simulated annealing framework as the exact simulation method, but with a reduced cost function that results in a substantial decrease in the runtime complexity. We term this method the "spatial correlation estimation method." For the example in Fig. 4, the optimum permutation returned by the spatial correlation estimation method also has a power consumption of 106 units. This method is so called because the reduced cost function is built using the values of spatial correlation that are computed for the outputs of the circuit driving the checker. The use of the reduced cost function does not involve any circuit simulation for each input order encountered during the simulated annealing routine. The reduced cost function is built *once* by a structural analysis of the checker, and all input orders are evaluated by simple substitution of spatial correlation values into this function. This avoids the use of the power estimator that is the main computational bottleneck of the exact simulation method. The pseudocode for the spatial correlation estimation method is presented in Fig. 5.

### A. Spatial Correlations and Transition Probability

The reduced cost function uses the notion of the transition probability at a node, from probabilistic techniques for power estimation, as a measure of the average switching activity at that node. We first introduce some terminology and definitions, and provide an overview of probabilistic techniques for power estimation [16]. We use the terms *signal* and *node* interchangeably throughout this document.

*Definition 1: (Signal Probability):* The signal probability $P_s(x)$ of a node $x$ in the circuit corresponds to the average fraction of clock cycles in which the node has a steady state logic value of ONE. An exact algorithm to estimate signal probabilities of all internal signals of a circuit is given in [19].

*Definition 2: (Spatial Correlation):* Two (or more) signals are spatially correlated if the values that one assumes are dependent on the values of the other. Two signals are spatially independent if they are not spatially correlated.

*Definition 3: (Temporal Correlation):* A signal is temporally correlated if the value that it assumes on the next clock cycle is dependent on its present and (or) previous values.

In other words, spatial correlation refers to the correlation between pairs of bits in the same input vector, whereas temporal correlation refers to the correlation between pairs of input vectors, spaced one or more cycles apart. We use random pattern simulation to compute the spatial correlation between the output signals of the circuit that drives the checker. If application vectors are used instead, the values of correlation obtained will reflect both temporal and spatial correlation, since the distribution of input vectors is not likely to be uniform.

*Definition 4: (Transition Probability):* The transition probability $P_t(x)$ of a node $x$ in the circuit corresponds to the average fraction of clock cycles in which the steady state value of the node is different from its initial value. Assuming temporal independence, the transition probability of a node $x_i$ is directly obtained from its signal probability $P_s(x_i)$ as

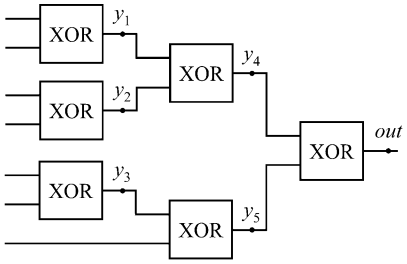
$$P_t(x_i) = 2 \cdot P_s(x_i) \cdot P_s(\bar{x}_i) = 2 \cdot P_s(x_i) \cdot (1 - P_s(x_i)).$$

Probabilistic techniques for power estimation propagate signal probabilities at the inputs into the circuit assuming spatial independence and use the temporal independence assumption to derive the transition probabilities. As discussed in Section IV-B, this can lead to large inaccuracies in the spatial correlation estimation method.

Reducing the transition probability at a node has the direct benefit of reducing the power consumption at that node. In addition, it is likely that this reduction in switching activity results in a reduction in the switching activity at the nodes that depend on this node. This can be extended to the checker as a whole, since an optimal input order will certainly reduce switching activity at nodes close to the inputs of the checker. This has a cascading effect, in that fewer transitions occur at the outputs of the gates that use these nodes as inputs, and so on to the primary outputs of the checker.

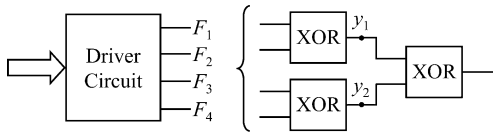
```
Algorithm Compute_Ordering ( checker_netlist, patterns ) {
        /* patterns – derived from the output trace of the circuit that drives the checker
           checker_netlist – technology mapped netlist of the checker */
        correlation = compute_correlation ( patterns ) ;
        /* compute spatial correlation between all pairs of outputs of the circuit that drives the checker */
        reduced_cost_function = build_cost_function ( checker_netlist ) ;
        /* build reduced cost function using structural analysis of the checker netlist */
        permutation = random ( ) ; /* random initial permutation of the inputs */
        permutation = simulated_annealing ( reduced_cost_function , permutation ) ;
        /* simulated annealing routine to solve the optimization problem */
        return permutation ;
}
```

Fig. 5. Pseudocode for the spatial correlation estimation method.




Reduced Cost Function = $P_t(y_1) + P_t(y_2) + P_t(y_3)$

Fig. 6. Reduced cost function for the example from Fig. 4.




Reduced Cost Function = $P_t(y_1) + P_t(y_2)$

| Input Order | $P_t(y_1)$ | $P_t(y_2)$ | Reduced Cost Function | Power |
|---|---|---|---|---|
| $F_1F_2F_3F_4$ | 0.22 | 0.22 | 0.44 | 21 |
| $F_1F_3F_2F_4$ | 0.49 | 0.49 | 0.98 | 25 |
| $F_1F_4F_2F_3$ | 0.49 | 0.49 | 0.98 | 25 |

Fig. 7. Spatial correlation estimation method for $F_1$, $F_2$, $F_3$, and $F_4$.

The reduced cost function is built by a structural analysis of the checker which is decomposed using 2-input gates. To build the reduced cost function, we compute the exact transition probability (taking spatial correlation into consideration) for all those signals that depend on one or two primary inputs to the checker. This is usually possible for nodes that reach a topological depth of two or three in the checker. The transition probability at the node is weighted by the load capacitance driven by that node. The reduced cost function for the parity checker in Fig. 4 (assuming unit load capacitance) is shown in Fig. 6. Note that the reduced cost function does not include the transition probabilities at nodes $y_4$ and $y_5$, since they depend on more than two primary inputs.

Correlations between all pairs of outputs of the circuit driving the checker are estimated by doing random vector simulation of the circuit driving the checker. Random vector simulation can be replaced by actual application vector simulation when they are available, since this best captures the correlations (both spatial and temporal) between signals. For every pair of outputs $\{F_i, F_j\}$ of the driver circuit, there are four combinations of values that can occur depending on the inputs to the circuit—00, 01, 10, and 11—and hence four values of spatial correlation (that sum to 1) to be computed. Once the correlation values are known, the exact transition probability at a node (that depends on two primary inputs) can be directly computed. This is illustrated in greater detail for a parity tree with four inputs in Fig. 7. The four inputs are driven by the functions $F_1$, $F_2$, $F_3$, and $F_4$ from Fig. 3.

The main benefit of using the reduced cost function is that many more permutations can be explored with minimal tradeoff in the accuracy and quality of the final input order that is obtained. Experimental results presented in Section V indicate that the power consumption of the final input order obtained is 16% lower than the average power consumption over 100 random input orders (when averaged over three types of checkers and all the test cases).

### B. Spatial Correlations and Accuracy

It is important to use spatial correlation values to compute the transition probability at nodes when building the reduced cost function. It is possible to use, under the spatial independence assumption, the signal probability at a node to estimate the transition probability at that node [16]. This is not as efficient, however, since correlations between pairs of inputs are not captured with sufficient accuracy. This is illustrated with an example in Fig. 8. Consider an XOR gate, driven by the functions $F_1$ and $F_2$ from Fig. 3. In Fig. 8, we compare the transition probability at the output of the XOR gate computed when spatial independence is assumed with the exact transition probability. Note that there is a significant difference (0.50 versus 0.22).

If the transition probabilities at the nodes are not accurate, the reduced cost function is not accurate. Such discrepancies can seriously affect the direction taken by the spatial correlation estimation method. For the example in Fig. 4, a solution that is not close to the optimum is obtained when spatial correlations are neglected, i.e., when spatial independence is assumed. The final input order returned by the spatial correlation estimation method using the reduced cost function built under the spatial independence assumption has a power consumption of 119 units. This is not only considerably off the global optimum of 106 units, but also very close to the maximum power consumption that was observed (126 units).

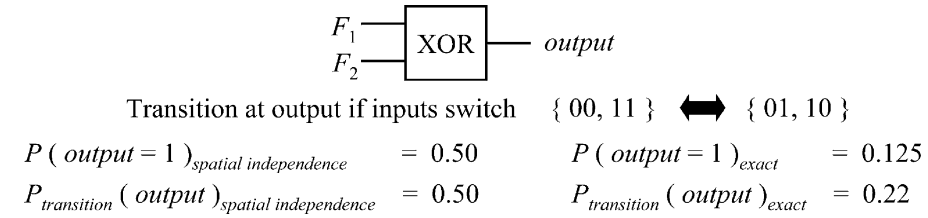



Fig. 8. Inaccuracies when spatial independence is assumed.

**Transition probabilities**

$P_{transition}(A, B) = 0.40$
$P_{transition}(A, C) = 0.41$
$P_{transition}(A, D) = 0.42$
$P_{transition}(B, C) = 0.45$
$P_{transition}(B, D) = 0.49$
$P_{transition}(C, D) = 0.50$

| Method | Optimal permutation | $P_{transition}$ (total) |
|---|---|---|
| Favalli and Metra [5] | $\{(A, B), (C, D)\}$ | 0.90 |
| Spatial Correlation Estimation Method | $\{(A, D), (B, C)\}$ | 0.87 |
| Exact Simulation Method | $\{(A, D), (B, C)\}$ | 0.87 |

Fig. 9. Advantage of proposed methods over Favalli and Metra [5].

### C. Comparison With [5]

The technique presented in [5] ranks all possible parings of inputs to the checker at each level of the checker and ranks them in increasing order of transition probability. The pair with the lowest transition probability is selected without consideration of how this affects subsequent pairings, i.e., the selection strategy is greedy and hence not globally optimal even for that particular logic level. This is presented by way of an example in Fig. 9. The cost function for the spatial correlation estimation method (exact simulation method) described in this paper evaluates an input order in terms of the transition probability (power consumption) across all pairings in the input order simultaneously and will hence select the better input order as shown in the figure.

The other disadvantage of the method presented in [5] is in the use of BDD-based symbolic techniques for evaluation and ranking of possible pairings. This may prove to be very expensive as the complexity of the logic driving the checker increases. Moreover, the need to evaluate all possible pairings (quadratic in the number of inputs) to rank input pairs at each logic level can result in very large runtimes as the number of inputs to the checker increases. By using simulated annealing, the two techniques presented in this paper can be tuned to evaluate a computationally tractable number of input orders on each pass. This coupled with the hill-climbing techniques inherent to simulated annealing present a cost-effective scalable alternative to the method proposed in [5].

### D. Testability

Note that input reordering may give rise to a testability degradation of the checker, since not all necessary codewords occur at the input of the checker. This is because a reduction in signal activity at a node in the checker may result in some signals never changing their value, thereby producing undetectable (by using codewords) stuck-at faults in the checker. It is possible to evaluate each input order encountered during simulated annealing for testability along the lines of the method suggested in [5]. By rejecting input orders that degrade the testability of the checker, it is possible to arrive at a solution that satisfies testability requirements on the checker. In most cases, however, testability is not a concern as almost all input codewords occur and are applied to the checker.

TABLE I
BENCHMARK COMBINATIONAL CIRCUITS [30]

| Circuit | Number of PIs | Number of POs | Power Consumption | |
|---|---|---|---|---|
| | | | *minimal.genlib* | *mcnc.genlib* |
| x2 | 10 | 7 | 365 | 390 |
| cu | 14 | 11 | 411 | 439 |
| sct | 19 | 15 | 728 | 802 |
| b9 | 41 | 21 | 746 | 772 |
| seq | 41 | 35 | 20189 | 21581 |
| x1 | 51 | 35 | 2511 | 2697 |
| vda | 17 | 39 | 4469 | 4713 |
| k2 | 45 | 45 | 9253 | 10602 |
| i5 | 133 | 66 | 2277 | 2237 |
| i6 | 49 | 67 | 4151 | 4390 |

## V. IMPLEMENTATION AND EXPERIMENTAL RESULTS

The synthesis tool used for all technology mapping and power estimation in this paper is SIS [25]. Some combinational benchmark circuits were chosen from the LGSynth91 suite [30]. 100 000 random vectors were used to obtain an output trace from each of the circuits. This trace was used to compute the spatial correlation between the outputs, as well as the power consumption for each of the input orders that were evaluated (for the exact simulation method). The power consumption of the input order returned by the spatial correlation estimation method, as well as the average power consumption over 100 random input orders, was also evaluated using this trace. Separate runs using the *minimal.genlib* and *mcnc.genlib* technology libraries were performed, since the optimal permutation obtained, as well as the reduction in power consumption achieved, varies according to the library used for technology mapping.

Table I provides details about the circuits that were chosen. Under the first major heading, we report the name, number of

TABLE II
POWER CONSUMPTION REDUCTION RESULTS FOR PARITY CHECKERS FOR SOME COMBINATIONAL CIRCUITS

| Circuit | *minimal.genlib* | | | | | *mcnc.genlib* | | | | |
|---|---|---|---|---|---|---|---|---|---|---|
| | Average Power | Exact Simulation Method | | Spatial Correlation Estimation Method | | Average Power | Exact Simulation Method | | Spatial Correlation Estimation Method | |
| | | Power | Time | Power | Time | | Power | Time | Power | Time |
| x2 | 80 | 72 | 1066 | 73 | 16 | 44 | 35 | 188 | 36 | 16 |
| cu | 76 | 60 | 2368 | 63 | 31 | 43 | 31 | 374 | 34 | 31 |
| sct | 289 | 262 | 3286 | 274 | 53 | 131 | 111 | 468 | 118 | 53 |
| b9 | 432 | 394 | 6119 | 425 | 97 | 207 | 140 | 769 | 152 | 97 |
| seq | 455 | 347 | 13400 | 347 | 246 | 212 | 113 | 1300 | 122 | 246 |
| x1 | 830 | 725 | 13410 | 746 | 249 | 374 | 315 | 1298 | 349 | 249 |
| vda | 867 | 710 | 14672 | 728 | 303 | 381 | 275 | 1394 | 298 | 303 |
| k2 | 572 | * | * | 444 | 396 | 281 | 188 | 1579 | 198 | 396 |
| i5 | 2230 | * | * | 2118 | 872 | 786 | 762 | 4150 | 764 | 872 |
| i6 | 2334 | * | * | 2109 | 886 | 857 | * | 4600 | 776 | 886 |

TABLE III
POWER CONSUMPTION REDUCTION FOR TWO-RAIL CODE CHECKERS FOR SOME COMBINATIONAL CIRCUITS

| Circuit | *minimal.genlib* | | | | | *mcnc.genlib* | | | | |
|---|---|---|---|---|---|---|---|---|---|---|
| | Average Power | Exact Simulation Method | | Spatial Correlation Estimation Method | | Average Power | Exact Simulation Method | | Spatial Correlation Estimation Method | |
| | | Power | Time | Power | Time | | Power | Time | Power | Time |
| x2 | 156 | 118 | 1096 | 131 | 16 | 209 | 150 | 636 | 175 | 16 |
| cu | 139 | 82 | 1802 | 106 | 31 | 182 | 111 | 981 | 143 | 31 |
| sct | 396 | 331 | 2351 | 354 | 53 | 517 | 439 | 1156 | 468 | 53 |
| b9 | 568 | 401 | 3458 | 403 | 97 | 815 | 560 | 2293 | 608 | 97 |
| seq | 560 | 309 | 6133 | 317 | 246 | 772 | 395 | 2306 | 501 | 246 |
| x1 | 972 | 765 | 5900 | 771 | 249 | 1344 | 1110 | 3370 | 1124 | 249 |
| vda | 993 | 734 | 7030 | 735 | 303 | 1335 | 998 | 3377 | 1010 | 303 |
| k2 | 762 | 483 | 8349 | 493 | 396 | 1035 | 667 | 4323 | 703 | 396 |
| i5 | 2173 | 2101 | 14712 | 2111 | 872 | 2878 | * | 5273 | 2800 | 872 |
| i6 | 2201 | 1910 | 15109 | 1953 | 886 | 3063 | 2700 | 6232 | 2712 | 886 |

primary inputs, and number of primary outputs of the circuits. Under the second major heading, we report the power consumption of the circuit (with random input patterns) mapped using the *minimal.genlib* and *mcnc.genlib* technology libraries, respectively.

Table II presents results for parity checkers for some combinational benchmark circuits chosen from the LGSynth91 suite, mapped using the *minimal.genlib* and the *mcnc.genlib* technology library. Under the first major heading, the name of the circuit is provided. Under the second major heading, we report results when the checker is mapped using the *minimal.genlib* technology library. The average power consumption of over 100 random input orderings that were used to drive the checker, the power consumption for the optimal permutation obtained using the exact simulation method, as well as the runtime, and the power consumption for the optimal permutation obtained using the spatial correlation estimation method, as well as the runtime are provided in that order. It is evident from the results that reordering the inputs results in significant power consumption reduction (12% on the average) when parity checkers are used. In addition, the spatial correlation estimation method provides a near optimal solution at a fraction of the computational cost of the exact simulation method. Note that the runtimes reported do not reflect the time taken to obtain an output trace from the driver circuit, since the same trace is used to report the power consumption in all the cases.

Under the third major heading, we report results when the checker is mapped using the *mcnc.genlib* technology library. The results are similar to those obtained with the *minimal.genlib* technology library—there is a reduction in power consumption, and the spatial correlation estimation method returns a solution close to that returned by the exact simulation method at a fraction of the computational cost.

Tables III and IV present results for two-rail code and Berger code checkers for some combinational benchmark circuits chosen from the LGSynth91 suite, mapped using the *minimal.genlib* and *mcnc.genlib* technology library respectively. The results are similar to those obtained for parity checkers—there is a reduction in power consumption, and the spatial correlation estimation method returns a solution close to that returned by the exact simulation method at a fraction of the computational cost.

Table V presents the reduction in power consumption (%) achieved over the average case for all the circuits and types of checkers, when the near optimal input order is computed using the spatial correlation estimation method. Under the first major

TABLE IV
POWER CONSUMPTION REDUCTION FOR BERGER CODE CHECKERS FOR SOME COMBINATIONAL CIRCUITS

| Circuit | *minimal.genlib* | | | | | *mcnc.genlib* | | | | |
|---|---|---|---|---|---|---|---|---|---|---|
| | Average Power | Exact Simulation Method | | Spatial Correlation Estimation Method | | Average Power | Exact Simulation Method | | Spatial Correlation Estimation Method | |
| | | Power | Time | Power | Time | | Power | Time | Power | Time |
| x2 | 268 | 230 | 3510 | 250 | 16 | 290 | 254 | 2900 | 269 | 16 |
| cu | 267 | 210 | 19140 | 210 | 31 | 283 | 212 | 24486 | 224 | 31 |
| sct | 1234 | 1101 | 22510 | 1134 | 53 | 1355 | 1099 | 29772 | 1282 | 53 |
| b9 | 2405 | 1987 | 77400 | 2161 | 97 | 2690 | 2046 | * | 2429 | 97 |
| seq | 2501 | * | * | 1765 | 246 | 2829 | * | * | 1909 | 246 |
| x1 | 7602 | * | * | 7363 | 249 | 7602 | 6800 | * | 7363 | 249 |
| vda | 6789 | * | * | 5729 | 303 | 6769 | * | * | 5729 | 303 |
| k2 | 3234 | * | * | 2668 | 396 | 3667 | * | * | 3166 | 396 |
| i5 | 28178 | * | * | 27813 | 872 | 28173 | * | * | 27826 | 872 |
| i6 | 26110 | * | * | 24462 | 886 | 26787 | * | * | 24919 | 886 |

TABLE V
REDUCTION IN POWER CONSUMPTION (%) ACHIEVED OVER AVERAGE CASE WITH THE SPATIAL CORRELATION ESTIMATION METHOD

| Circuit | *minimal.genlib* | | | *mcnc.genlib* | | |
|---|---|---|---|---|---|---|
| | Parity | Two-Rail Code | Berger Code | Parity | Two-Rail Code | Berger Code |
| x2 | 8.8 | 16.0 | 6.7 | 18.2 | 16.3 | 7.2 |
| cu | 17.1 | 23.7 | 21.4 | 20.9 | 21.4 | 20.9 |
| sct | 5.2 | 10.6 | 8.1 | 9.9 | 9.5 | 5.4 |
| b9 | 1.6 | 29.1 | 10.2 | 26.6 | 25.4 | 9.7 |
| seq | 23.7 | 43.4 | 29.4 | 42.5 | 35.1 | 32.5 |
| x1 | 10.1 | 20.7 | 3.1 | 6.7 | 16.4 | 3.1 |
| vda | 16.0 | 26.0 | 15.6 | 21.8 | 24.3 | 15.4 |
| k2 | 22.4 | 35.3 | 17.5 | 29.5 | 32.1 | 13.7 |
| i5 | 5.0 | 2.9 | 1.3 | 2.8 | 2.7 | 1.2 |
| i6 | 9.6 | 11.3 | 6.3 | 9.5 | 11.5 | 7.0 |
| Average Reduction | 12.0 | 21.9 | 12.0 | 18.8 | 19.5 | 11.6 |
| | 16.0 | | | | | |

heading, the name of the circuit is provided. Under the second major heading, we report the reduction in power consumption for parity, two-rail code, and Berger code checkers mapped using the *minimal.genlib* technology library. An average reduction of 12.0%, 21.9%, and 12.0% in power consumption is achieved. Similarly, under the third major heading, an average reduction of 18.8%, 19.5%, and 11.6% in power consumption is achieved for checkers mapped using the *mcnc.genlib* technology library. Finally, an average reduction of 16% in power consumption is achieved across the three types of checkers mapped using the two technology libraries.

The runtime of the simulated annealing routine for the exact simulation method was chosen and balanced to be between one to two orders of magnitude greater than that of the spatial correlation estimation method, depending on the complexity of the checker. If the execution time exceeded this predetermined limit, the exact simulation method was aborted and the best solution seen up to that point is reported *only* if it is better than that returned by the spatial correlation estimation method—this appears as (power consumption, $*$) under the columns for the exact simulation method in Tables II–IV. Cases where the solution of the spatial correlation method is better are reported as ($*$, $*$). Cases where the solution of the exact simulation method is poorer than that obtained using the spatial correlation estimation method, and where the exact simulation method was not aborted are given by ($*$, execution time).

### A. Runtime Versus Accuracy Tradeoffs

Our implementation decomposes the checker using 2-input gates when building the reduced cost function. If correlation values for signals considered three at a time were available, it would be possible to write the transition probabilities for some more gates when building the reduced cost function. This can improve the accuracy of the proposed procedure, and better results could be obtained by decomposing the checker to up to 3-input gates. Our experimental results indicate that it is sufficient to consider correlations between pairs of signals when building the reduced cost function.

It is also interesting to note that the computational complexity of the spatial correlation estimation method lies mainly in the estimation of the spatial correlation between pairs of outputs of the driver circuit. It is not affected by the choice of the library that is used for technology mapping or the complexity of the checker. In other words, for the same number of inputs, parity checkers are less complex than two-rail code checkers that are in turn less complex than Berger code checkers. The exact simulation method takes least time for parity checkers and the most time for Berger code checkers for the same circuit. However, the

spatial correlation estimation method takes the same time for all the three checkers for a given circuit. Thus, the spatial correlation estimation method is a very fast alternative to computing a near optimal input order as the complexity of the checker (and hence, the exact simulation method) increases.

## VI. Conclusion

As CED increasingly becomes a necessity in mainstream commercial electronics, there is an urgent need for techniques to reduce the associated overhead costs. This paper presented an efficient and scalable approach for reducing the power consumption in checkers used for CED. The method is applicable to any functionally symmetric checker. It analyzes spatial correlations between the outputs of the circuit that drives the checker to order them such that switching activity (and hence, power consumption) in the checker is minimized. The main advantages of this method are that the reduction in power consumption comes at no additional impact to area or performance and that it does not require any alteration to the design flow. The only cost is the time for computing the input ordering for the checker that minimizes power consumption. The proposed technique can be easily integrated into existing CAD tools. A possibility for future research is to look at the applicability of this technique in other fields where such functionally symmetric checkers are used—examples include comparators, voters, and error detection and correction circuitry for memories.

## References


[1] M. Boudjit, M. Nicolaidis, and K. Torki, "Automatic generation algorithms, experiments and comparisons of self-checking PLA schemes using parity codes," in *Proc. Eur. Conf. Design Automation*, 1993, pp. 144–150.

[2] N. Cohen, T. S. Sriram, N. Leland, D. Moyer, S. Butler, and R. Flatley, "Soft error considerations for deep-submicron CMOS circuit applications," in *Int. Electron Devices Meeting Tech. Dig.*, 1999, pp. 315–318.

[3] D. Das and N. A. Touba, "Synthesis of circuits with low-cost concurrent error detection based on Bose-Lin codes," *J. Electron. Testing: Theory Applicat.*, vol. 15, no. 1/2, pp. 145–155, Aug. 1999.

[4] K. De, C. Natarajan, D. Nair, and P. Banerjee, "RSYN: A system for automated synthesis of reliable multilevel circuits," *IEEE Trans. VLSI Syst.*, vol. 2, pp. 186–195, June 1994.

[5] M. Favalli and C. Metra, "Design of low-power two-rail checkers," *J. Microelectron. Syst. Integration*, vol. 5, no. 2, pp. 101–110, 1997.

[6] S. Gorshe and B. Bose, "A self-checking ALU design with efficient codes," in *Proc. VLSI Test Symp.*, 1996, pp. 157–161.

[7] M. Gössel and S. Graf, *Error Detection Circuits*. New York: McGraw-Hill, 1993.

[8] N. K. Jha and S. Wang, "Design and synthesis of self-checking VLSI circuits," *IEEE Trans. Computer-Aided Design*, vol. 12, pp. 878–887, June 1993.

[9] X. Kavousianos and D. Nikolos, "Novel single and double output TSC Berger code checkers," in *Proc. VLSI Test Symp.*, 1998, pp. 348–353.

[10] S. Kirkpatrick, C. D. Gelatt, and M. P. Vecchi Jr., "Optimization by simulated annealing," *Science*, pp. 671–680, May 1983.

[11] G. P. Mak, J. A. Abraham, and E. S. Davidson, "The design of PLAs with concurrent error detection," in *Proc. Int. Symp. Fault Tolerant Computing*, 1982, pp. 303–310.

[12] C. Metra, M. Favalli, and B. Ricco, "Tree checkers for applications with low power-delay requirements," in *Proc. Int. Symp. Defect and Fault Tolerance VLSI Systems*, 1996, pp. 213–220.

[13] K. Mohanram and N. A. Touba, "Input ordering in concurrent checkers to reduce power consumption," in *Proc. Int. Symp. Defect and Fault Tolerance VLSI Systems*, 2002, pp. 87–95.

[14] T. Mudge, "Power: A first-class architectural design constraint," *IEEE Computer*, vol. 34, pp. 52–58, Apr. 2001.

[15] E. Musoll and J. Cortadella, "Optimizing CMOS circuits for low power using transistor reordering," in *Proc. European Design and Test Conf.*, 1996, pp. 219–223.

[16] F. N. Najm, "A survey of power estimation techniques in VLSI circuits," *IEEE Trans. VLSI Syst.*, vol. 2, pp. 446–455, Dec. 1994.

[17] M. Nicolaidis, "Efficient implementations of self-checking adders and ALUs," in *Proc. Int. Symp. Fault Tolerant Computing*, 1993, pp. 586–595.

[18] M. Nicolaidis and Y. Zorian, "Online testing for VLSI—A compendium of approaches," *J. Electron. Testing: Theory Applicat.*, vol. 12, no. 1/2, pp. 7–20, Feb.-Apr. 1998.

[19] K. P. Parker and E. J. McCluskey, "Probabilistic treatment of general combinatorial networks," *IEEE Trans. Computers*, vol. C-24, pp. 668–670, June 1975.

[20] S. J. Piestrak, D. Bakalis, and X. Kavousianos, "On the design of self-testing checkers for modified Berger codes," in *Proc. IEEE Online Testing Workshop*, 2001, pp. 153–157.

[21] D. K. Pradhan, *Fault Tolerant Computing—Theory and Techniques*. Englewood Cliffs, NJ: Prentice-Hall, 1986, vol. 1, ch. 5.

[22] S. C. Prasad and K. Roy, "Transistor reordering for power minimization under delay constraint," *ACM Trans. Design Automation Elect. Syst.*, vol. 1, no. 2, pp. 280–300, Apr. 1996.

[23] V. V. Saposhnikov, A. Morosov, V. V. Saposhnikov, and M. Gössel, "A new design methodology for self-checking unidirectional combinational circuits," *J. Electron. Testing: Theory Applicat.*, vol. 12, no. 1/2, pp. 41–53, Feb.–Apr. 1998.

[24] P. H. Schneider and S. Krishnamoorthy, "Effects of correlations on accuracy of power analysis—An experimental study," in *Proc. Int. Symp. Low-Power Electron. Design*, 1996, pp. 113–116.

[25] E. M. Sentovich *et al.*, "SIS: A System for Sequential Circuit Synthesis," Univ. Calif., Berkeley, CA, Tech. Rep. UCB/ERL M92/41, 1992.

[26] D. P. Siewiorek and R. S. Swarz, *Reliable Computer Systems: Design and Evaluation*, 3rd ed. Wellesly, MA: A. K. Peters, 1998.

[27] J. E. Smith and G. Metze, "Strongly fault secure logic networks," *IEEE Trans. Computers*, vol. C-27, pp. 491–499, June 1978.

[28] C. Tan and J. Allen, "Minimization of power in VLSI circuits using transistor sizing, input ordering, and statistical power estimation," in *Proc. Int. Workshop Low-Power Design*, 1994, pp. 75–80.

[29] N. A. Touba and E. J. McCluskey, "Logic synthesis of multilevel circuits with concurrent error detection," *IEEE Trans. Computer-Aided Design*, vol. 16, pp. 783–789, July 1997.

[30] S. Yang, "Logic Synthesis and Optimization Benchmarks User Guide," MCNC, Research Triangle Park, NC, 1991-IWLS-UG-Saeyang, 1991.

[31] J. F. Ziegler *et al.*, "IBM experiments in soft fails in computer electronics (1978–1994)," *IBM J. Res. Develop.*, vol. 40, no. 1, pp. 3–18, Jan. 1996.


**Kartik Mohanram** (S'00–M'04) received the B.Tech. degree in electrical engineering from the Indian Institute of Technology, Bombay in 1998, and the M.S. and Ph.D. degrees in computer engineering from the University of Texas at Austin, in 2000 and 2003, respectively.

He is currently an Assistant Professor in the Department of Electrical and Computer Engineering, Rice University, Houston, TX.

**Nur A. Touba** (S'88–M'96) received the B.S. degree from the University of Minnesota, Minneapolis in 1990, and the M.S. and Ph.D. degrees from Stanford University, Stanford, CA, all in electrical engineering, in 1991 and 1996, respectively.

He is currently an Associate Professor in the Department of Electrical and Computer Engineering at the University of Texas at Austin.

Prof. Touba received the National Science Foundation (NSF) Early Faculty CAREER Award in 1997, and the Best Paper Award at the 2001 VLSI Test Symposium. He is on the program committees for the International Test Conference, International Conference on Computer Design, Design Automation and Test in Europe Conference, International On-Line Test Symposium, International Test Synthesis Workshop, and the European Test Workshop.