

On the Minimization of Potential Transient Errors and SER in Logic Circuits using SPFD

Sobeeh Almkhaizim
CE Department
Kuwait University
Kuwait

Yiorgos Makris
CS & EE Departments
Yale University
USA

Yu-Shen Yang & Andreas Veneris
ECE Department
University of Toronto
Canada

Abstract

Sets of Pairs of Functions to be Distinguished (SPFD) is a functional flexibility representation method that was recently introduced in the logic synthesis domain, and promises superiority in exploring the flexibility offered by a design over all previous representation methods. In this work, we illustrate how the SPFD of a particular wire reveals information regarding the number of potential transient errors that may occur on that wire and may affect the output of the circuit. Using an SPFD-based rewiring method, we then demonstrate how to evolve a logic circuit in order to minimize the total number of potential transient errors in the circuit and, consequently, reduce its Soft Error Rate (SER) while controlling the effect on the rest of the design parameters, such as area, power, delay, and testability. Experimental results on ISCAS'89 and ITC'99 benchmark circuits indicate that the SER can be reduced at no additional overhead to any of the design parameters.

1 Introduction

As the CMOS technology scales toward the 32nm process and beyond, logic circuits are becoming ever more susceptible to transient errors produced by noise and radiation effects. These single random errors in the circuit may propagate to its output, thereby threatening the reliable operation of the circuit in the field. In order to cope with these errors, various mitigation and tolerance techniques have been developed by the testing community [1, 2, 3, 4, 5]. Yet, all of these solutions treat the logic design as a black box, wherein the circuit implementation is unaltered in order to preserve the design parameters and optimization efforts of logic synthesis tools. Such synthesis and optimization methods have proven successful in transforming a logic design to minimize area [6, 7, 8], improve testability [9], reduce power consumption [10] and satisfy timing requirements [11, 12]. Moreover, the recent work in [13, 14, 15] has shown that logic synthesis and optimization methods can also be used to reduce the Soft Error Rate (SER) of a circuit. Specifically, it has been ob-

served that functionally-equivalent yet structurally-different implementations of a circuit exhibit different levels of immunity to potential transient errors, paving the way for the development of logic optimization methods that explore the full design flexibility in order to minimize the SER of the produced logic implementation.

The underlying concept in all logic optimization methods is to change the functionality of nodes in the network within the functional flexibility of the design. Such flexibility has been explored through Incompletely Specified Functions (ISFs), Compatible Observability Don't Cares (CODCs), and Compatible Sets of Permissible Functions (CSPFs). At IC-CAD'96, Yamashita et. al. [16] pioneered the use of a new concept, termed *Sets of Pairs of Functions to be Distinguished* (SPFD), to represent the flexibility of nodes in Field-Programmable Logic Arrays (FPGAs). Their observations have been followed by a body of work in FPGA and logic synthesis [17, 18, 19, 20, 21, 22, 23, 24], establishing the superiority of SPFD in exploring the flexibility offered by a design [20, 24]. Very recently, SPFD have been proven to provide more optimization power than other functional flexibility representation methods [25].

In this paper, we propose a logic optimization method to minimize the number of potential transient errors and, therefore, the SER in logic circuits through the use of SPFD. In addition to the typical design objectives optimized by synthesis tools, we demonstrate how SPFD-based rewiring can also be used to minimize the number of potential transient errors that may occur in a design. We start, in Section 2, by describing an SPFD-based rewiring method which we use to generate functionally-equivalent yet structurally-different gate-level circuit implementations. We then demonstrate, in Section 3, how the number of potential transient errors appearing at the output of a circuit depends on the number of distinct minterms in the SPFD sets of the wires in the circuit. Then, in Section 4, we propose an algorithm which evolves a design through iterative selection of rewiring operations, in order to optimize a cost function reflecting both the number of potential transient errors and the rest of the design parameters of a circuit. Finally, in Section 5, we evaluate the SER reduction of the proposed method using ISCAS'89 and ITC'99 benchmark circuits.

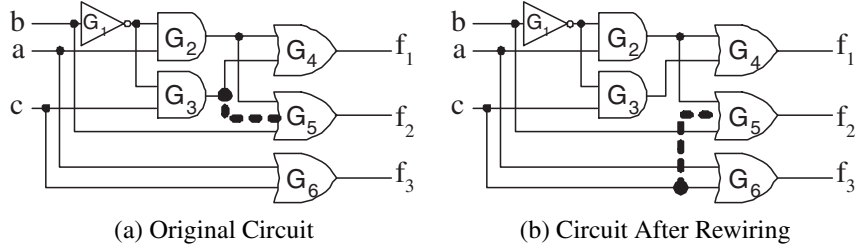


Figure 1. Example of SPFD-based Rewiring to Reduce the Number of Potential Transient Errors

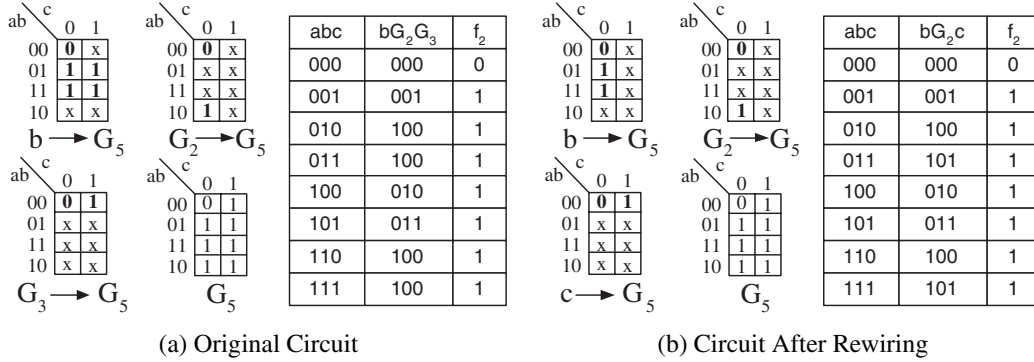


Figure 2. Karnaugh Map Representation of the SPFD sets of the Inputs/Output of G_5 and the Truth Table of f_2

2 SPFD and SPFD-based Rewiring

SPFD have been introduced in [16] to represent the functional flexibility of a node in a logic network. The *minimum SPFD* ($SPFD_{min}$) of a target wire represents the care onset minterms that must be distinguished from the care offset minterms by that wire. Moreover, a wire that *satisfies* the $SPFD_{min}$ of the target wire, i.e., one that distinguishes all the care onset minterms from the care offset minterms for the target wire, is a valid replacement. Therefore, SPFD-based rewiring proceeds by computing the SPFD of all the wires in the circuit. Subsequently, candidate wires are identified by checking whether their functionality satisfies the minimum SPFD of the target wire.

Example 1: Consider the example circuit in Fig. 1.a and the truth table of the inputs/output of G_5 in Fig. 2.a. For this example, the output of G_5 , i.e., f_2 , has to distinguish the care onset minterms (i.e., m_0) from the care offset minterms (i.e., $m_1, m_2, m_3, m_4, m_5, m_6, m_7$). Therefore, the minimum SPFD of f_2 is the set $\{(m_0, m_1), (m_0, m_2), (m_0, m_3), (m_0, m_4), (m_0, m_5), (m_0, m_6), (m_0, m_7)\}$. Now, consider the wire $G_3 \rightarrow G_5$ as the target wire for logic rewiring. The truth table in Fig. 2.a indicates that G_3 must produce 0 for the input minterm m_0 , and 1 for the input minterm m_1 ; otherwise, the output response at f_2 will be incorrect. Furthermore, the functionality of G_3 is considered “don’t care” for all of the other minterms, as a value of 0 or 1 for these minterms would not change the response value at f_2 . Thus,

G_3 must distinguish m_0 from m_1 only, which corresponds to the minimum SPFD set of $\{(m_0, m_1)\}$; any wire that distinguishes the minterms is a candidate replacement of G_3 ¹.

The computation of the SPFD of a wire necessitates logic simulation of all of the possible input patterns to the circuit, which is computationally infeasible for large circuits. In order to avoid the memory/time explosion problem, approximate SPFD (aSPFD) have been proposed and utilized to perform SPFD-based rewiring [24]. An aSPFD represents a subset of the minterms in the care set of the original SPFD. Hence, aSPFD-rewiring first computes the aSPFD of a target wire. Subsequently, all candidate replacement wires that distinguish the minterms in the aSPFD of the target wire are identified. Finally, the candidate wires are verified using a SAT-based algorithm. Verification, similar to the one in ATPG-based rewiring [13], is necessary since the aSPFD ensures the validity of the modified design using a subset of the complete input vector space and, therefore, the candidate replacements returned are only valid for this particular subset of vectors. Details regarding the implementation of the SAT-based verification step can be found in [24].

¹The minimum SPFD of a wire indicates which minterms must be distinguished from each other, but not the polarity of the logic values that must be produced for these minterms. This is attributed to the flexibility offered by the initial application that introduced the concept of SPFD, i.e., FPGAs. Specifically, FPGAs utilize Look-Up Tables (LUTs) that can be programmed to produce the required polarity of the logic value by inherently inverting the signal. In logic circuits, however, the addition of an inverter may be necessary to ensure that the correct polarity is being produced.

3 Transient Errors in Logic Circuits

In this section, we first derive the relationship between the minimum SPFD set of a particular wire and the number of potential transient errors that may appear on this wire. Then, we demonstrate how SPFD-based rewiring can be employed to minimize the total number of potential transient errors in the circuit. Finally, we discuss several special cases for the minimum SPFD set of a wire, which pinpoint our ongoing research direction on the optimal synthesis of logic circuits with the minimal number of potential transient errors.

3.1 Relating SPFD to Transient Errors

A transient error is a pair of an error location, such as a wire w_i where the error might appear, and an input pattern, such as input pattern m_j that sensitizes the error to the output of the logic circuit. Given a possible error location w_i , our objective is to determine the number of potential transient errors at w_i by examining its SPFD set. Clearly, the number of potential transient errors at w_i is the number of input patterns m_j that sensitize the error to the output, where a change in the value of w_i from the onset to the offset, or vice versa, would affect the correctness of the output. Or, in other words, the correct functionality of the circuit depends on the ability of w_i to distinguish its onset from its offset for these input patterns. This is essentially the definition used to compute the minimum SPFD set of w_i , i.e., that the output of the circuit depends on w_i distinguishing between its onset and offset for these particular input patterns.

Let w_i be a wire in the logic circuit and f be the function implemented by that wire. Assume that the $SPFD_{min}(w_i)$ contains r pairs of minterms, and that $S(w_i)$ is the care set of minterms in the $SPFD_{min}(w_i)$. Given the $SPFD_{min}(w_i)$, our objective is to find the number of potential transient errors at w_i , $TEs(w_i)$. A transient error at w_i will affect the correctness of the output function for every minterm in the care set of the $SPFD_{min}(w_i)$, i.e., for each minterm in $S(w_i)$. Thus, the number of transient errors is equal to the number of minterms in $S(w_i)$. More formally:

$$TEs(w_i) = |S(w_i)| = t_i \quad (1)$$

where t_i is the number of distinct minterms in the care set of the $SPFD_{min}(w_i)$. Thus, the number of potential transient errors at a wire is determined by examining the number of distinct minterms in the minimum SPFD set of that wire. The total number of potential transient errors in a logic circuit N is equal to the sum of the distinct minterms in the minimum SPFD sets of the wires in the circuit, i.e.,:

$$TEs(N) = \sum_{i=1}^{\# \text{ of wires}} TEs(w_i) \quad (2)$$

Example 2: Consider again the example circuit in Fig. 1.a and the associated Karnaugh map representation of the minimum SPFD sets in Fig. 2.a of the inputs/output of G_5 . The Karnaugh map representation of the SPFD sets is provided here for readability purposes. For this example, the $SPFD_{min}(G_5) = \{(m_0, m_1), (m_0, m_2), (m_0, m_3), (m_0, m_4), (m_0, m_5), (m_0, m_6), (m_0, m_7)\}$, the $SPFD_{min}(b \rightarrow G_5) = \{(m_0, m_2), (m_0, m_3), (m_0, m_6), (m_0, m_7)\}$, the $SPFD_{min}(G_2 \rightarrow G_5) = \{(m_0, m_4)\}$, and the $SPFD_{min}(G_3 \rightarrow G_5) = \{(m_0, m_1)\}$. Therefore, $S(G_5) = \{m_0, m_1, m_2, m_3, m_4, m_5, m_6, m_7\}$ with $TEs(G_5) = 8$, $S(b \rightarrow G_5) = \{m_0, m_2, m_3, m_6, m_7\}$ with $TEs(b \rightarrow G_5) = 5$, $S(G_2 \rightarrow G_5) = \{m_0, m_4\}$ with $TEs(G_2 \rightarrow G_5) = 2$, and $S(G_3 \rightarrow G_5) = \{m_0, m_1\}$ with $TEs(G_3 \rightarrow G_5) = 2$. In this circuit, 9 potential transient errors affecting the inputs of G_5 , shown in boldface in their Karnaugh map, will propagate to f_2 . Similar analysis can be performed for the other gates.

In the above example, notice that the pair (m_0, m_5) appears in the $SPFD_{min}(G_5)$ only but not in the minimum SPFD sets of its inputs. This is further illustrated in the truth table of f_2 , where m_5 is distinguished using $G_2 \rightarrow G_5$ and $G_3 \rightarrow G_5$. Hence, since (m_0, m_5) is distinguished by both wires, it does not appear in either of their minimum SPFD sets. This is the key idea explored in this work in order to minimize the number of potential transient errors in the circuit, as will be demonstrated next.

3.2 Minimization of Transient Errors

The analysis in the previous section reveals that the output of a circuit is highly susceptible to a wire, $w_{susc} \rightarrow G_i$, if the cardinality of $S(w_{susc} \rightarrow G_i)$ is large. SPFD-based rewiring provides the ability to reduce the number of pairs in the $SPFD_{min}(w_{susc} \rightarrow G_i)$ and, possibly, the number of minterms in $S(w_{susc} \rightarrow G_i)$. As demonstrated in the example in Section 3.1, a pair of minterms in the minimum SPFD set of a gate is eliminated from the minimum SPFD sets of its inputs if the pair is distinguished by at least two inputs to that gate. Therefore, the number of minterms in $S(w_{susc} \rightarrow G_i)$ can be reduced if another input to G_i , $w_t \rightarrow G_i$, is replaced with $w_{new} \rightarrow G_i$, where $w_{new} \rightarrow G_i$ distinguishes more pairs in the $SPFD_{min}(w_{susc} \rightarrow G_i)$ than $w_t \rightarrow G_i$. The candidate replacement wire w_{new} for the target wire $w_t \rightarrow G_i$ is identified by performing SPFD-based rewiring with $w_t \rightarrow G_i$ set as the target wire.

Example 3: Consider the minimum SPFD sets of the inputs to gate G_5 in Fig. 1.a. The wire $b \rightarrow G_5$ has the highest number of potential transient errors, where $TEs(b \rightarrow G_5) = 5$. Thus, $G_3 \rightarrow G_5$, is selected as the target wire for rewiring, as illustrated using the dashed line in Fig. 1.a. SPFD-based rewiring reveals that $c \rightarrow G_5$ can be used to replace $G_3 \rightarrow G_5$, as illustrated in the circuit implementation in Fig. 1.b. The Karnaugh maps representation of the SPFD of the inputs/output of G_5 and the truth table of f_2 in

the circuit after rewiring are illustrated in Fig. 2.b. The truth table of f_2 in Fig. 2.b indicates that $c \rightarrow G_5$ distinguishes two additional minterms, m_3 and m_7 , over the ones distinguished by $G_3 \rightarrow G_5$ in Fig. 2.a. Thus, the pairs in the minimum SPFD set of $b \rightarrow G_5$ that contain these minterms, (m_0, m_3) and (m_0, m_7) , are eliminated from the set. Therefore, $TEs(b \rightarrow G_5)$ is reduced from 5 to 3 and, overall, the number of potential transient errors at the inputs of G_5 is reduced from 9 to 7. Such a single rewiring operation translates into a 1.65% reduction to the SER of the circuit.

3.3 Discussion

Based on the above analysis, the relationship between the $SPFD_{min}(w_i)$ and the number of potential transient errors for certain special cases can be characterized as follows:

- $S(w_i)$ is empty: This implies that w_i is redundant and that potential transient errors affecting w_i will never propagate to the output. A stuck-at fault at w_i is untestable by off-line test methods.
- $|S(w_i)|$ is very small: This implies that the output depends on the value of w_i for few input vectors and, thus, the output has a low susceptibility to errors affecting w_i . A stuck-at fault at w_i is considered to be hard-to-test or random-pattern resilient by off-line test methods.
- $|S(w_i)|$ is very large: This implies that the output depends on the value of w_i for many input vectors and, thus, the output has a high susceptibility to errors affecting w_i . A stuck-at fault at w_i is considered to be easy-to-test by off-line test methods.

The minimum SPFD sets of the inputs to a particular gate are reduced if they share many of the pairs that appear in the minimum SPFD set of the output of the gate. Intuitively, this observation implies that the functionality of the inputs to a particular gate should be highly similar. On one extreme, all of the inputs would implement identical functions, namely, the exact functionality of the output of the gate. Clearly, such a scenario creates logic redundancy in the circuit, which is undesirable due to the created offline test challenges. On the other extreme, no two inputs share a pair in their minimum SPFD sets. This scenario produces an implementation with the highest number of potential transient errors.

The perfect balance between the above two extremes would be that each pair in the minimum SPFD set of a k -input gate is distinguished by at least two of its inputs, except k pairs that are each distinguished using only one of the k inputs. In this case, the functionality of the inputs is highly similar to the functionality of the output, yet not redundant. Thus, a test pattern is guaranteed to exist for production testing, namely the minterms in the single pair of their minimum SPFD set. The synthesis of such logic circuits that implement a desired functionality with the minimum number of potential transient errors is part of our ongoing research.

Reduce_TEs_Using_SPFD(*Design*)

```

Repeat until all wires in Design have been tried
  Compute the cost function of Design ( $CF_{old}$ );
  Compute  $SPFD_{min}()$  of the wires;
  Sort wires in descending order of  $TEs()$ ;
   $CF_{best} = CF_{old}$ ;
  While a wire in the circuit has not been tried
    Perform SPFD-based rewiring & generate  $k$  different designs;
    For each  $Design_i, i < k$ 
      Compute  $CF_i$ ;
      If  $CF_i > CF_{best}$ 
         $CF_{best} = CF_i$ ;
         $Design_{best} = Design_i$ ;
      If  $CF_{best}$  is not equal to  $CF_{old}$ 
         $Design = Design_{best}$ ;
    Exit While loop;

```

Figure 3. Greedy Heuristic for Reducing SER Using SPFD-Based Rewiring

4 Proposed Optimization Algorithm

In this section, we devise an algorithm that iteratively selects effective rewiring operations and evolves the circuit in order to optimize a cost function reflecting a given set of design objectives, including the number of potential transient errors. The proposed algorithm employs a simple greedy heuristic, wherein, at each step, rewiring is attempted for the wire with the highest number of distinct minterms in its minimum SPFD that has not been tried so far. In order to identify the most susceptible wire, we employ the aSPFD method in [24] to compute the $SPFD_{min}$ for each wire in the circuit. Then, the list of wires is sorted ($SortWires()$) in decreasing order of the number of potential transient errors that may occur in the wire, i.e., $TEs()$, and the first wire in the list is selected as a target wire ($TargetWire$). Once the target wire is selected, we perform SPFD-based rewiring in order to generate a list of k candidate designs. For each candidate design i ($Design_i$), we evaluate the cost function (CF_i), and keep the design ($Design_{best}$) that has the best improvement to the cost function (CF_{best}) over the previous design ($Design_{old}$). The process is iteratively repeated until all the wires have been tried without improvement to the cost function. The algorithm is summarized in Fig. 3.

5 Experimental Results

We experiment with a set of ISCAS'89 and ITC'99 benchmark circuits. In these experiments, we use the TSMC 0.13 μ m process, Synopsys Design Compiler is used to compute the area, delay and power of the circuit, and TetraMax is used to perform Automatic Test Pattern Generation (ATPG) and compute any loss in fault coverage during production testing. The SER is computed using SERA [26], which reports the SER for each output of the circuit. The search

Original Circuit				<i>OnlySER</i>				
Name	PI	PO	Gates	SER	Area	Delay	Power	F. C. Loss
b01	7	7	51	17.10%	-4.24%	-20.00%	-2.38%	0.00%
b02	5	5	27	10.79%	-4.55%	36.84%	-0.30%	3.58%
b03	34	34	153	0.00%	0.00%	0.00%	0.00%	0.00%
b06	11	15	55	15.85%	-19.05%	45.16%	-12.53%	2.25%
b08	30	25	171	2.40%	-0.91%	17.19%	0.13%	0.74%
b10	28	23	180	5.26%	-1.90%	21.79%	-3.38%	1.49%
s298	17	20	119	17.46%	-6.52%	37.50%	-7.88%	2.50%
s382	24	27	158	25.09%	-11.16%	11.11%	-19.01%	2.59%
s344	24	26	160	7.35%	-9.90%	10.59%	-10.48%	0.12%
s349	26	26	161	7.28%	-6.92%	10.59%	-11.15%	1.71%
s526	24	27	173	16.69%	-5.07%	22.08%	-9.12%	5.58%
s444	24	27	181	11.98%	-2.66%	20.29%	-4.95%	3.17%
s510	25	13	211	16.34%	-16.72%	3.51%	6.24%	0.00%

Table 1. Experimental Results Using SPFD-based Rewiring (*OnlySER*)

Original Circuit				<i>SERandAll</i>				
Name	PI	PO	Gates	SER	Area	Delay	Power	F. C. Loss
b01	7	7	51	0.00%	0.00%	0.00%	0.00%	0.00%
b02	5	5	27	0.00%	0.00%	0.00%	0.00%	0.00%
b03	34	34	153	0.00%	0.00%	0.00%	0.00%	0.00%
b06	11	15	55	6.53%	-3.45%	0.00%	-5.57%	0.00%
b08	30	25	171	0.00%	0.00%	0.00%	0.00%	0.00%
b10	28	23	180	1.96%	-4.19%	-1.28%	-4.72%	0.00%
s298	17	20	119	20.23%	-15.92%	-8.23%	-16.62%	0.00%
s382	24	27	158	3.34%	-1.93%	-2.72%	-3.64%	0.00%
s344	24	26	160	7.21%	-8.87%	-2.92%	-7.88%	0.00%
s349	26	26	161	3.07%	-3.46%	0.00%	-3.43%	-0.52%
s526	24	27	173	1.06%	0.00%	-2.08%	-0.35%	0.00%
s444	24	27	181	2.55%	-2.39%	-2.22%	-3.05%	-0.85%
s510	25	13	211	0.00%	0.00%	0.00%	0.00%	0.00%

Table 2. Experimental Results Using SPFD-based Rewiring (*SERandAll*)

algorithm was driven by two cost functions: *OnlySER*, which aims at minimizing the soft error rate regardless of the impact of rewiring on the other design parameters, and *SERandAll*, which reduces the SER as long as all the design parameters of the modified circuit after the rewiring operation are better than or equal to the design parameters of the initial circuit. While we only present results using the above cost functions, any other cost function can be used to drive the search algorithm [13].

The results are presented for the *OnlySER* and *SERandAll* cost functions in Tables 1 and 2, respectively, illustrating the percentile SER reduction, area overhead, delay overhead, power overhead, and fault coverage loss. The key points revealed by these results are summarized below:

- Logic rewiring finds peephole optimizations that synthesis tools may not. The results herein are consistent with other studies, wherein optimizations have proven successful in transforming a logic design to minimize area [6, 7, 8], improve testability [9], reduce power consumption [10] and reduce timing requirements [11, 12].

- The results for *OnlySER* indicate that applying SPFD-based rewiring can reduce substantially the SER of the circuit, with an average of 11.82%. For example, the SER of *s298*, *s526* and *s510* is reduced by more than 15%. However, when the search is driven by the sole objective of reducing SER, the impact of rewiring on other design parameters such as area, power, delay, and testability can be significant. For example, the delay of *b02*, *b06* and *s298* increases by more than 30%.
- By monitoring the impact on other design parameters during the search algorithm, we can eliminate its effect, as indicated by the results for the *SERandAll* cost function. As expected, however, the additional constraints placed on the search algorithm diminish the attained SER reduction. Nevertheless, this reduction is overhead-free and, as such, highly desirable. Moreover, this constrained design-space exploration often results in significant reduction in one or more of these design parameters. For example, the area of *s298* is reduced by more than 15% and its delay by 8%.

6 Conclusion

In addition to the optimization of typical design parameters, such as area, delay, and power consumption, concepts developed in the logic synthesis domain have also been often utilized to address challenges in testing, such as the identification of redundant faults. In this work, we have demonstrated how the SPFD concept is related to the number of potential transient errors in a logic circuit, and can, therefore, be utilized in order to reduce the SER. Given the large space of functionally-equivalent yet structurally-different implementations of a function, the SER can be often reduced at no additional overhead to any of the design parameters.

References

- [1] Q. Zhou and K. Mohanram, "Cost-effective radiation hardening technique for logic circuits," in *IEEE/ACM International Conference on Computer-Aided Design*, 2004, pp. 100–106.
- [2] S. Mitra, M. Zhang, T.M. Mak, N. Seifert, V. Zia, and K.S. Kim, "Logic soft errors: A major barrier to robust platform design," in *International Test Conference*, 2005, pp. 687–698.
- [3] C. Zhao, X. Bai, and S. Dey, "A scalable soft spot analysis methodology for noise effects in nano-meter circuits," in *ACM/IEEE Design Automation Conference*, 2004, pp. 894–899.
- [4] Y. S. Dhillon, U. Diril, and A. Chatterjee, "Soft-error tolerance analysis and optimization of nanometer circuits," in *Design, Automation and Test in Europe*, 2005, pp. 288–293.
- [5] N. Miskov-Zivanov and D. Marculescu, "MARS-C: Modeling and reduction of soft errors in combinational circuits," in *ACM/IEEE Design Automation Conference*, 2006, pp. 767–772.
- [6] S. C. Chang, K. T. Cheng, N. S. Woo, and M. Marek-Sadowska, "Postlayout logic restructuring using alternative wires," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, vol. 16, no. 6, pp. 587–596, 1997.
- [7] L. A. Entrena and K. T. Cheng, "Combinational and sequential logic optimization by redundancy addition and removal," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, vol. 14, no. 7, pp. 909–916, 1995.
- [8] W. Kunz, D. Stoffel, and P. R. Menon, "Logic optimization and equivalence checking by implication analysis," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, vol. 16, no. 3, pp. 266–281, 1997.
- [9] M. Chatterjee, D. Pradhan, and W. Kunz, "LOT: Logic optimization with testability - new transformations using recursive learning," in *IEEE/ACM International Conference on Computer-Aided Design*, 1995, pp. 115–118.
- [10] B. Rohlfleisch, A. Kolbl, and B. Wurth, "Reducing power dissipation after technology mapping by structural transformations," in *ACM/IEEE Design Automation Conference*, 1996, pp. 789–794.
- [11] Y. M. Jiang, A. Krstic, K. T. Cheng, and M. Marek-Sadowska, "Postlayout logic restructuring for performance optimization," in *ACM/IEEE Design Automation Conference*, 1997, pp. 662–665.
- [12] G. Stenz, B. M. Riess, B. Rohlfleisch, and F. M. Johannes, "Performance optimization by interacting netlist transformations and placement," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, vol. 19, no. 3, pp. 350–358, 2000.
- [13] S. Almukhaizim, Y. Makris, Y.-S. Yang, and A. Veneris, "Seamless integration of SER in rewiring-based design space exploration," in *IEEE International Test Conference*, 2006, pp. 29.3.1–29.3.9.
- [14] S. Krishnaswamy, S.M. Plaza, I.L. Markov, and J.P. Hayes, "Enhancing design robustness with reliability-aware resynthesis and logic simulation," in *IEEE/ACM International Conference on Computer-Aided Design*, 2007, pp. 149–154.
- [15] S. Almukhaizim and Y. Makris, "Soft error mitigation through selective addition of functionally redundant wires," *IEEE Transactions on Reliability*, vol. 57, no. 1, pp. 23–31, 2008.
- [16] S. Yamashita, H. Sawada, and A. Nagoya, "A new method to express functional permissibilities for LUT based FPGAs and its applications," in *IEEE/ACM International Conference on Computer-Aided Design*, 1996, pp. 254–261.
- [17] S. Sinha and R. K. Brayton, "Implementation and use of SPFDs in optimizing Boolean networks," in *IEEE/ACM International Conference on Computer-Aided Design*, 1998, pp. 103–110.
- [18] S. Yamashita, H. Sawada, and A. Nagoya, "SPFD: A new method to express functional flexibility," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, vol. 19, no. 8, pp. 840–849, 2000.
- [19] J. Cong, J. Y. Lin, and W. Long, "A new enhanced SPFD rewiring algorithm," in *IEEE/ACM International Conference on Computer-Aided Design*, 2002, pp. 672–678.
- [20] J. Cong, J. Y. Lin, and W. Long, "SPFD-based global rewiring," in *International Symposium on FPGAs*, 2002, pp. 77–84.
- [21] S. Sinha, A. Mishchenko, and R. K. Brayton, "Topologically constrained logic synthesis," in *IEEE/ACM International Conference on Computer-Aided Design*, 2002, pp. 679–686.
- [22] S. P. Khatri, S. Sinha, R. K. Brayton, and A. Sangiovanni-Vincentelli, "SPFD-based wire removal in standard-cell and network-of-PLA circuits," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, vol. 23, no. 7, pp. 1020–1030, 2000.
- [23] A. Mishchenko, J. S. Zhang, S. Sinha, J. R. Burch, R. K. Brayton, and M. Chrzanoska-Jeske, "Using simulation and satisfiability to compute flexibilities in boolean networks," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, vol. 25, no. 5, pp. 743–755, 2006.
- [24] Y.-S. Yang, S. Sinha, A. Veneris, and R. K. Brayton, "Automating logic rectification by approximate SPFDs," in *IEEE/ACM Asia South Pacific Design Automation Conference*, 2007, pp. 77–84.
- [25] S. Sinha, X. Wang, and R. K. Brayton, "Comparing two rewiring models," *International workshop on Logic Synthesis*, pp. 438–445, 2004.
- [26] M. Zhang and N. R. Shanbhag, "A soft error rate analysis (SERA) methodology," in *International Conference on Computer-Aided Design*, 2004, pp. 111–118.