IPF: In-Place X-Filling Algorithm for the Reliability of Modern FPGAs

Zhe Feng, Naifeng Jing, and Lei He

***Abstract*— Modern SRAM-based field-programmable gate arrays (FPGAs) are prone to single event upsets compared to application-specific integrated circuits. We propose a synthesis-based in-place x-filling algorithm by utilizing don't cares to augment the reliability of FPGA-based designs. Compared to circuit- and architecture-based solutions, our algorithm is in place, and does not incur area, power, performance, and design time overheads. Compared to other synthesis-based algorithms, we take into account widely accepted interconnect architecture. For the 10 largest combinational MCNC benchmark circuits mapped to 6-LUT architecture, our approach achieves up to 37% greater failure rate reduction, and up to 7× runtime speedup, compared to the best known synthesis-based in-place algorithm, namely the in-place decomposition algorithm.**

***Index Terms*— Design reliability, field-programmable gate array (FPGA), in place, interconnect, single event upset (SEU), synthesis, x-filling.**

I. Introduction

Field-programmable gate arrays (FPGAs) have been widely used in different applications, such as networking, digital signal processing, and prototyping. Nevertheless, single event upsets (SEUs), also called soft errors, have posed a major barrier for the reliability of SRAM-based FPGAs. SEUs are generally caused by high-energy particle strikes, e.g., neutrons coming from cosmic rays or alpha particles emitted from trace impurities in packaging materials and solder bumps [1]. They change the values of devices such as SRAM cells and flip-flops when the charges collected from strikes are larger than a threshold. In SRAM-based FPGAs, because most logic functions and interconnects are implemented by SRAM cells, they are more vulnerable to SEUs compared to application-specific integrated circuits (ASICs). SEUs have a permanent impact on FPGAs till configuration scrubbings are applied. In the past, the SEU issue received attention only from high-reliability applications in military and aerospace areas. As modern FPGAs have advanced to 28-nm technology, the devices are prone to SEUs for most applications due to reduction in core voltage, decrease in transistor geometry, and increase in switching speed.

There have been a number of studies seeking for the solution of SEU mitigation for SRAM-based FPGAs. These solutions can broadly be divided into circuit-, architecture-, and synthesis-based techniques. The first two categories incur extensive area, power, performance, and design time overheads [2], [3]. Several studies have demonstrated that the SEU issue can be mitigated by synthesis-based approaches while minimizing the aforementioned overheads. The in-place decomposition (IPD) algorithm proposed by Lee *et al*. [4], decomposes a logic function in a logic block into two subfunctions, and converges them via a carry chain. Their work claims to reduce failure rates by 76%. However, most synthesis-based techniques either mitigate errors introduced by SEUs only on lookup tables (LUTs) [4]–[6], without considering the SEU impact on interconnects, or there is a drawback in the interconnect SEU model, e.g., the model in [7] assumes that there is only one configuration bit in each net. As a result, their improvements for the reliability are significantly smaller when SEUs on interconnects are taken into consideration (as shown in Section V). Besides, they rely on creating don't cares to tolerate errors introduced by SEUs. However, the large amount of preexisting don't cares makes them difficult to further increase don't cares without area overhead. As shown in Table I, for the 10 largest combinational MCNC benchmark circuits [8], we observe that don't cares comprise approximately 60% of utilized LUT configuration bits when the designs are mapped to 6-LUTs.[1]

TABLE I
Ratio of Don't Cares to Utilized LUT Configuration Bits

MCNC circuits	6-LUT		
	Configuration bits#	Don't cares#	SDC bits#
alu4	33408	17180	15356
apex2	45760	29063	19303
apex4	37440	20920	20319
des	54528	33185	31686
ex1010	43584	23445	23105
ex5p	23808	16193	13471
misex3	31552	15923	14751
pdc	105280	63760	58580
seq	46208	20885	17934
spla	98688	61325	56294
Average	52026	30188	27080
Ratio	1	**58.03%**	52.05%
		1	**89.70%**

This motivates us to exploit preexisting don't cares in LUTs to augment the reliability of designs. We present an LUT and interconnect analysis-based in-place x-filling[2] (IPF) algorithm, which fills don't cares to mask errors introduced by SEUs on both LUTs and interconnects. Compared to other synthesis-based algorithms, we take into account the widely accepted interconnect architecture used in Versatile Place and Route (VPR) [11] during optimization. In addition, our algorithm overcomes the slow runtime issue prevailing in most previous synthesis-based techniques because it does not search for functionally equivalent implementations, which requires time-consuming algorithms like Boolean satisfiability [5], integer linear programming [4], or set of pairs of functions to be distinguished [7]. Compared to circuit- and architecture-based solutions, our algorithm does not incur area, power, performance, and design time overheads, because there is no change of LUT level placement and routing, i.e., it is an in-place algorithm.

For the 10 largest combinational MCNC benchmark circuits mapped to 6-LUTs, our approach achieves up to 37% greater failure rate reduction, and up to 7× runtime speedup, compared to the best known synthesis-based in-place algorithm, namely the IPD algorithm.

The rest of this brief is organized as follows. We start with preliminaries introducing FPGA design representation, failure rate, and don't cares in Section II, followed by the formulation of the IPF problem in Section III. The proposed algorithm is presented in Section IV. The experimental results are summarized in Section V, followed by conclusions in Section VI.

Manuscript received August 1, 2012; revised April 23, 2013; accepted August 29, 2013. Date of publication October 11, 2013; date of current version September 23, 2014. This work was supported in part by Cisco and in part by JPL.

Z. Feng and L. He are with the Department of Electrical Engineering, University of California, Los Angeles, CA 90095 USA (e-mail: feng07@ucla.edu; lhe@ee.ucla.edu).

N. F. Jing is with the School of Microelectronics, Shanghai Jiao Tong University, Shanghai 200240, China (e-mail: jingnaifeng@ic.sjtu.edu.cn).

Color versions of one or more of the figures in this brief are available online at http://ieeexplore.ieee.org.

Digital Object Identifier 10.1109/TVLSI.2013.2282819

[1] After designs are mapped by the Berkeley mapper [9], the number of don't cares is computed by the windowing technique proposed by Cong *et al*. [6].

[2] The term has been used for power-aware automatic test pattern generation (ATPG) [10], in which power is minimized by filling don't cares to reduce logic switches of designs.

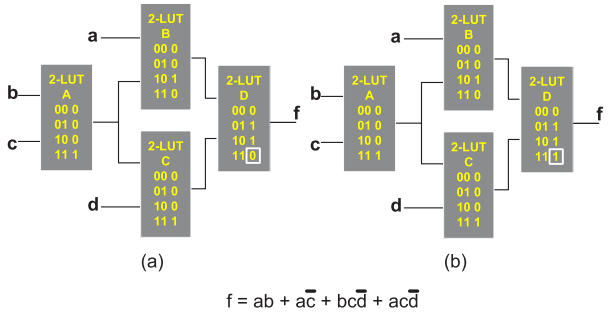

Fig. 1. Given the same functionality and topology, different implementations yield different failure rates due to the assignment of the SDC bit. (a) Failure rate = 0.2031. (b) Failure rate = 0.1875.

II. Preliminaries

A. FPGA Design Representation

An FPGA design is usually represented by a directed acyclic graph. In the graph, nodes represent LUTs, and edges represent interconnects between LUTs. If Node a drives Node b, Node a is called Node b's fan-in, and Node b is called Node a's fan-out. The node without the fan-in is called the primary input, and the node without fan-out is called the primary output. The fan-in (fan-out) cone of Node a is the nodes reachable through fan-in (fan-out) edges from Node a.

B. Failure Rate and Don't Care

In this brief, the sensitivity of a configuration bit C_i to an SEU is measured by the failure rate of the configuration bit, i.e., the frequency with which a circuit fails because of the SEU on the configuration bit, expressed in (1). The sensitivity of a circuit to SEUs can be measured by the failure rate of the circuit, which is the average of the failure rates of all the configuration bits. V denotes the full set of input vectors. $\mathrm{PO}_{\mathrm{golden}}$ is the primary output vector without the impact of SEUs. $\mathrm{PO}_{\mathrm{SEU}}$ is the primary output vector when an SEU occurs on the configuration bit C_i

$$Fr(C_i) = \frac{\sum_{\forall v \in V} (\mathrm{PO}_{\mathrm{golden}}(v) \wedge \mathrm{PO}_{\mathrm{SEU}}(v)(C_i))}{|V|}. \quad (1)$$

If the failure rate of a configuration bit is 0, the bit is a don't care bit. There are two kinds of don't care bits, i.e., satisfiability don't care (SDC) bits and observability don't care (ODC) bits due to limited accessibilities and observabilities of configuration bits in a circuit. The SDC bit is the inaccessible configuration bit in the node that does not have a full set of input permutations at their fan-ins. The ODC bit is the configuration bit that is not observable at the primary output given a set of input vectors [12]. In Fig. 1, C_{11}[3] in LUT D is an SDC bit that is not accessible. C_{00} in LUT A is an ODC bit when $a = 0$ and $d = 0$. We only focus on exploiting SDC bits for SEU mitigation for two reasons: 1) SDC bits comprise about 90% of total don't cares for the designs under test, as shown in Table I and 2) SDC bits are compatible don't cares, because flipping an SDC bit does not invalidate other don't cares.

III. Problem Formulation

In this section, we illustrate utilizing preexisting don't cares to mask the errors introduced by SEUs, and formulate the IPF problem. In Fig. 1, given a logic function f, there are two implementations with the same interconnects between LUTs. Configuration bit C_{11} in LUT D is an SDC bit that is inaccessible in a normal situation.

[3]In this brief, the configuration bit corresponding to the input ABCD is denoted as C_{ABCD}.

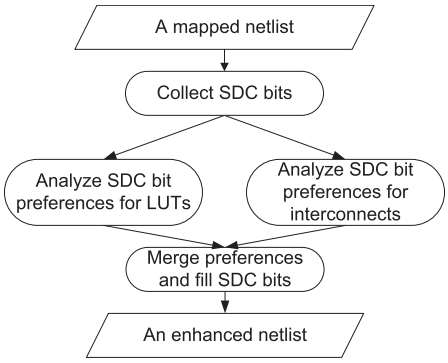

Fig. 2. Overview of the LUT and interconnect analysis-based IPF algorithm.

Under the impact of SEUs, the failure rate is greater when C_{11} is filled with 0 in Fig. 1(a) than when C_{11} is assigned 1 in Fig. 1(b). Both failure rates in Figs. 1(a) and (b) are calculated by equation (1). The reason is that SDC bit C_{11} in LUT D can be accessed when an SEU is in the fan-in cone of LUT D; therefore, when C_{11} is filled with a feasible value, even if an SEU occurs, C_{11} can be used to mask errors in LUT D.

The idea behind the example is that, in a normal circuit, SDC bits in LUTs are inaccessible. When SEUs occur in fan-in cones, SDC bits can be hit. In this situation, LUTs can still output correct values if the SDC bits are preset feasibly. More concretely, we formulate the IPF problem as follows: given a design, fill SDC bits in all LUTs to increase the logic masking for errors introduced by SEUs in their fan-in cones thereby to augment the reliability of the design.

IV. In-Place X-Filling Algorithm

In this section, we propose a synthesis-based IPF algorithm employing logic masking targeting SEUs on both LUTs and interconnects. As shown in Fig. 2, given a mapped netlist, we first collect all the SDC bits in the netlist; we perform the analyses of SDC bit preferences for LUTs and interconnects; the values of SDC bits are filled on the basis of the analyses, and an enhanced netlist is dumped.

A. SDC Bit Collection

A window-based logic simulation is performed to collect SDC bits, i.e., launching a logic simulation on a selected window, to collect the inaccessible SDC bits during the logic simulation. We adopt Cong and Minkovich's work [6] to collect SDC bits. The criteria of selected windows are as follows: 1) when choosing the window covering the node under test, priority is given to the windows covering the most nodes given a bounded input number and 2) overlapping windows are used to minimize the controllability set, i.e., making the SDC bit collection a tight lower bound compared to a full-circuit simulation.

B. SDC Bit Preferences

In order to mask errors by filling SDC bits, in a faulty circuit simulation we evaluate how many times the propagated errors can be masked if the SDC bits are preset as 0 (or 1), i.e., the SDC bit preferences to be preset as 0 (or 1). The more preferable value is assigned to the SDC bit for SEU mitigation. Here is an example illustrating the calculation of the SDC bit preference for 0. Suppose that the output sequence of an LUT is 0101 without the impact of SEUs, denoted as G, and the output sequence under SEUs is 0110, denoted as F. The difference of the two output sequences is 0011, denoted as D. For an SDC bit of the LUT, suppose that the SDC bit is hit only by the third input vector, i.e., 0010, denoted as H. For the third input vector, there is a difference between G and F,

and G outputs 0. At the same time, the SDC bit is hit according to H. The error can be tolerated if the SDC bit is preset to 0. Therefore, the SDC bit preference for 0 increases by 1. Calculations for SDC bit preferences to mask errors introduced by SEUs on LUTs are formulated in (2) and (3). count_1 counts the number of 1s in a vector. Equation (2) computes the chance that the SDC bit can be used for masking errors if preset as 1: i.e., when the SDC bit is hit, there is a difference between the outputs of the LUT with and without injecting an SEU, and the output is 1 in a normal circuit. The same logic applies for the SDC bit preference for 0 in (3)

$$\begin{aligned} 1-\text{preference} &= \text{count\_1}\{H\&D\&G\} = \text{count\_1}\{H\&(G \wedge F)\&G\} \\ &= \text{count\_1}\{H\&\bar{F}\&G\} \quad (2) \\ 0-\text{preference} &= \text{count\_1}\{H\&D\&\bar{G}\} = \text{count\_1}\{H\&(G \wedge F)\&\bar{G}\} \\ &= \text{count\_1}\{H\&F\&\bar{G}\}. \quad (3) \end{aligned}$$

When analyzing SDC bit preferences for SEUs on interconnects, we adopt the same interconnect architecture as used in VPR [11], i.e., interconnects consist of local wires, connection boxes, and switch boxes. The signal routes are directed by configuration bits in the three components. Configuration bits in interconnects can be flipped by SEUs, resulting in LUT SDC bits being hit and outputted. The calculations of the SDC bit preferences for SEUs on LUTs are applied to the analysis of interconnect too.

C. Implementation and Complexity Analysis

We gather the SDC bit preferences by performing the logic simulation on the netlist after injecting SEUs on the configuration bits in LUTs or interconnects. A logic simulation computes the values of the outputs of internal nodes and the primary outputs of a netlist, given a set of input vectors. One run of simulation propagates one set of vectors through the netlist. Its complexity increases linearly with the netlist size. According to Luckenbill's experiments [13], 1024 randomly generated input vectors yield a close estimation to the exhaustive input vectors with a mean error of 1%, and 131 072 input vectors reduce the error to 0.3%. We perform uniform 102 400 simulations for each configuration bit to evaluate the impact of SEUs. The error is negligible considering that our algorithm achieves up to 37% failure rate reduction.

To ensure the efficiency of the simulation, in our implementation the following techniques are employed:

1) performing simulations in a bit-parallel manner, i.e., simulating 32 or 64 runs at the same time;
2) performing simulations in an incremental style, i.e., except for the initial 102 400 simulations, we only resimulate and propagate the errors in the fan-in and fan-out cone of the current node, and stop the propagation immediately if there is no change of outputs of nodes;
3) using and inverter graph (AIG) [9] representation for the netlist. Simulating an AIG node benefits bitwise operations on the simulation information of the fan-ins.

The runtime of the IPF algorithm can be broken into three portions: the runtime for SDC bits collection, the runtime for evaluation of the impact of SEUs on configuration bits, and the time spent on SDC bit filling. The SDC bit filling is performed in constant time after gathering the SDC bit preferences. Considering that we adopt a window-based logic simulation for SDC bits collection and full-circuit simulation for the evaluation of SEU impact, the runtime is dominated by the evaluation [see (6)]. N_W is the number of simulations in 64-bit machine words. n and m are the number of nodes and the average number of configuration bits insides a node, respectively. T_1 denotes the simulation time spent on one node. T_{ini} denotes the simulation time for calculating the golden result without the impact of SEUs. T_{inc} denotes the simulation time for incrementally evaluating the impact of SEUs on configuration bits. L denotes the number of nodes to be resimulated in each simulation during the incremental process. In summary, the computation complexity of IPF algorithm is $O(\text{nm})$

$$T_{\text{ini}} = N_W \cdot n \cdot T_1 \quad (4)$$

$$T_{\text{inc}} = N_W \cdot L \cdot n \cdot m \cdot T_1 \quad (5)$$

$$T_{\text{total}} = T_{\text{ini}} + T_{\text{inc}} = N_W \cdot n \cdot T_1(1 + L \cdot m)(L \ll n). \quad (6)$$

V. Experimental Results

The proposed IPF algorithm was implemented in C++, and tested on a PC with dual core CPU E4400 @ 2.00 GHz and 2.0 GB of RAM. For the 10 largest combinational MCNC benchmark circuits [8], we use designs mapped by the Berkeley ABC mapper [9] as the baseline. All designs enhanced by our IPF algorithm passed the functional equivalent checking by the Berkeley ABC mapper. In terms of failure rate reduction and runtime[4], we compare our LUT and interconnect analysis-based IPF algorithm with the best known synthesis-based in-place algorithm, namely the IPD algorithm [4], and the two IPF algorithms proposed in our previous work that perform analysis on LUTs only [14].

A. Failure Rate Evaluation of SEU Mitigation Techniques at the Circuit Level

In this section, we perform the circuit level evaluation for failure rates of synthesis-based SEU mitigation techniques. "Circuit level" means that SEUs can be on LUTs and interconnects during the evaluation. We adopt Jing's work [15] to perform the interconnect evaluation in which interconnects are composed of local wires, switching boxes, and connection boxes. Each time, we inject an SEU on a configuration bit in LUT or interconnect and perform a full-circuit logic simulation. For each configuration bit, 102 400 input vectors are injected into circuits.

In Fig. 3, the x-axis lists the 10 largest combinational MCNC benchmark circuits, and the y-axis lists circuit level failure rate reductions. The "Critical conf bit" algorithm (shown in Fig. 3) and "Critical output" algorithm are the two IPF algorithms proposed in our previous work that perform analysis on LUT only [14]. The "Critical conf bit" algorithm represents the algorithm that employs SDC bits to mask errors on the most critical configuration bit. The "Critical output" algorithm utilizes SDC bits to mask the more critical output. The "LUT and interconnect based" algorithm refers to the IPF algorithm proposed in this brief. As shown in Fig. 3, when the designs are small, the IPD algorithm and our LUT analysis-based algorithms yield similar failure rate reductions. As the design size increases, our algorithms outperform the IPD algorithm. Our LUT and interconnect analysis-based algorithm always generates a design with better reliability than the IPD algorithm does. For the circuit "des," the reason why the failure rate is increased by the LUT analysis-based algorithm but the rate can be reduced by the IPD algorithm and our LUT and interconnect analysis-based algorithm is that the criticality used in LUT analysis-based algorithm does not consider whether the SDC bit is actually hit and used for logic masking; however, our LUT and interconnect analysis-based algorithm takes the SDC bit hit into account when calculating the SDC bit preference, and results in the failure rate reduction.

The IPD algorithm targets SEUs on LUT configuration bits, and yields 7% circuit level failure rate reduction, although it is known

[4]We only present comparisons for the 6-LUT mapping because the IPD algorithm has only 6-LUT mapping results in public,

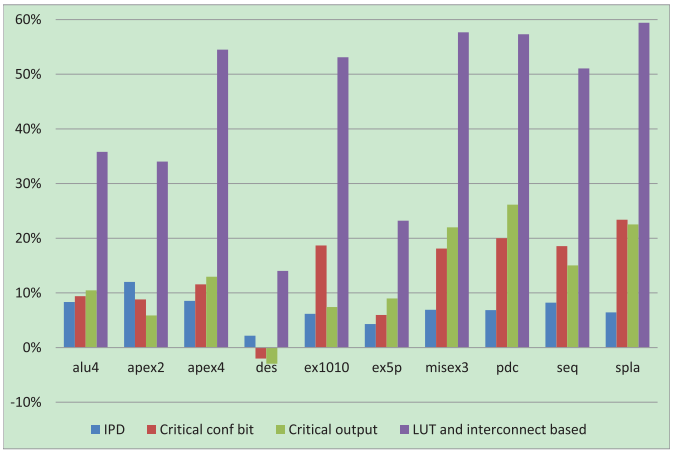

Fig. 3. Failure rate comparison of synthesis-based SEU mitigation techniques at the circuit level for the 6-LUT mapping.

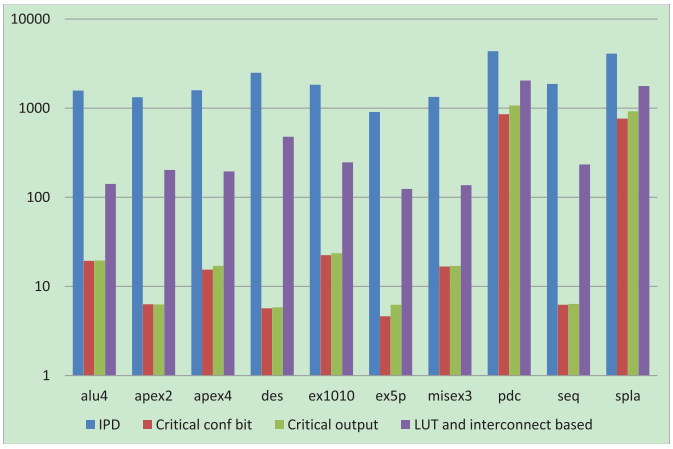

Fig. 4. Runtime comparison of synthesis-based SEU mitigation techniques at the circuit level for the 6-LUT mapping.

for high failure rate reduction when considering SEUs on LUTs only. At the circuit level, our LUT analysis-based IPF algorithms achieve 6% higher reduction compared to the IPD algorithm, and our LUT and interconnect analysis-based IPF algorithm achieves 37% greater improvement.

B. Runtime Comparison of SEU Mitigation Techniques at the Circuit Level

Fig. 4 presents runtime comparisons for the IPD algorithm and our IPF algorithms for the 6-LUT mapping. The x-axis lists the 10 largest combinational MCNC benchmark circuits, and the y-axis lists runtime in seconds. On average, our Critical conf bit and Critical output algorithms achieve 150× and 142× speedup compared to the IPD algorithm. Although our LUT and interconnect analysis-based algorithm incurs a runtime overhead for the interconnect analysis, it still achieves 7× speedup compared to the IPD algorithm. The fast synthesis time makes our IPF algorithms scalable in practice.

The reason for the fast runtime is that our approaches do not search for functionally equivalent implementations, and therefore do not need time-consuming algorithms like Boolean satisfiability [5], integer linear programming [4], or set of pairs of functions to be distinguished [7] as adopted in other synthesis-based algorithms. Furthermore, when performing the circuit analysis, we adopt a bit-parallel logic simulation on and-inverter graph and perform logic simulations incrementally.

In summary, our IPF algorithms outperform the IPD algorithm in terms of both failure rate reduction and the runtime.

VI. Conclusion

Targeting the ever-increasing SEU issue, we proposed a synthesis-based IPF algorithm by exploiting don't cares to augment the reliability of designs. Compared to circuit- and architecture-based solutions, our algorithm is in place and does not incur area, power, performance, and design time overheads. Compared to other synthesis-based algorithms, we took into account the widely accepted interconnect architecture used in VPR [11] during optimization. For the ten largest combinational MCNC benchmark circuits mapped to 6-LUTs, our approach achieved up to 37% greater failure rate reduction at the circuit level, and up to 7× runtime speedup, compared to the best known synthesis-based in-place algorithm, namely the IPD algorithm.

The more don't cares are reconfigured to mask errors introduced by SEUs, the greater failure rate reduction we can achieve. Increasing don't cares during synthesis can be leveraged to obtain the targeted tradeoff between reliability and area in the future. Furthermore, we plan to extend the IPF algorithm to handle sequential circuits. The key to this extension is to model the error propagations in sequential cycles efficiently. In addition, the impact of the technology scaling makes multiple errors introduced by SEUs a big concern. The difficulty to extend the IPF algorithm for multiple errors is in tackling the correlation between errors. The three issues will be addressed in future work.

References

[1] N. Bidokhti, "SEU concept to reality (allocation, prediction, mitigation)," in *Proc. Rel. Maintainab. Symp.*, Jan. 2010, pp. 1–5.

[2] (2012, Mar.). *Radiation-Hardened, Space-Grade Virtex-5QV Device Overview* [Online]. Available: http://www.xilinx.com

[3] P. K. Samudrala, J. Ramos, and S. Katkoori, "Selective triple modular redundancy (STMR) based single-event upset (SEU) tolerant synthesis for FPGAs," *IEEE Trans. Nucl. Sci.*, vol. 51, no. 5, pp. 2957–2969, Oct. 2004.

[4] J.-Y. Lee, Z. Feng, and L. He, "In-place decomposition for robustness in FPGA," in *Proc. IEEE/ACM Int. Conf. Comput.-Aided Design*, Nov. 2010, pp. 143–148.

[5] Y. Hu, Z. Feng, L. He, and R. Majumdar, "Robust FPGA resynthesis based on fault-tolerant Boolean matching," in *Proc. IEEE/ACM Int. Conf. Comput.-Aided Design*, Nov. 2008, pp. 706–713.

[6] J. Cong and K. Minkovich, "LUT-based FPGA technology mapping for reliability," in *Proc. IEEE/ACM Design Autom. Conf.*, Jun. 2010, pp. 517–522.

[7] M. Jose, Y. Hu, R. Majumdar, and L. He, "Rewiring for robustness," in *Proc. IEEE/ACM Design Autom. Conf.*, Jun. 2010, pp. 469–474.

[8] S. Yang, "Logic synthesis and optimization benchmarks, version 3.0," Microelectronics Center of North Carolina (MCNC), Long Island City, NY, USA, Tech. Rep., 1991.

[9] A. Mishchenko. (2011, Feb.). *ABC: A System for Sequential Synthesis and Verification* [Online]. Available: http://www.eecs.berkeley.edu/alanmi/abc/

[10] J.-Y. Lee, Y. Hu, and R. Majumdar, "Simultaneous test pattern compaction, ordering and X-filling for testing power reduction," in *Proc. Int. Symp. Qual. Electron. Design*, Mar. 2009, pp. 702–707.

[11] J. Luu, I. Kuon, P. Jamieson, T. Campbell, A. Ye, W. Fang, K. Kent, and J. Rose, "VPR 5.0: FPGA CAD and architecture exploration tools with single-driver routing, heterogeneity and process scaling," *ACM Trans. Reconfigurable Technol. Syst.*, vol. 4, no. 32, pp. 1–23, Dec. 2011.

[12] A. Mishchenko, J. Zhang, S. Sinha, J. Burch, R. Brayton, and M. Chrzanowska-Jeske, "Using simulation and satisfiability to compute flexibilities in Boolean networks," *IEEE Trans. Comput.-Aided Design Integr. Circuits Syst.*, vol. 25, no. 5, pp. 743–755, May 2006.

[13] S. Luckenbill, J.-Y. Lee, Y. Hu, R. Majumdar, and L. He, "RALF: Reliability analysis for logic faults—An exact algorithm and its applications," in *Proc. Design, Autom. Test Eur. Conf. Exhibit.*, Mar. 2010, pp. 783–788.

[14] Z. Feng, N. Jing, G. Chen, Y. Hu, and L. He, "IPF: In-place X-filling to mitigate soft errors in SRAM-based FPGAs," in *Proc. Int. Conf. FPL*, Sep. 2011, pp. 482–485.

[15] N. Jing, J.-Y. Lee, Z. Feng, W. He, Z. Mao, S.-J. Wen, R. Wong, and L. He, "Quantitative SEU fault evaluation for SRAM-based FPGA architectures and synthesis algorithms," in *Proc. Int. Conf. Field-Program. Logic Appl.*, Sep. 2011, pp. 282–285.