

IPF: In-Place X-Filling to Mitigate Soft Errors in SRAM-based FPGAs

Zhe Feng¹, Naifeng Jing², GengSheng Chen³, Yu Hu⁴, and Lei He¹

1. *Electrical Engineering Department, University of California, Los Angeles*

2. *School of Microelectronics, Shanghai Jiao Tong University*

3. *State Key Lab of Application Specific Circuits and Systems, Fudan University, Shanghai, China*

4. *Electrical and Computer Engineering Department, University of Alberta*

Abstract—SRAM-based Field Programmable Gate Arrays (FPGAs) are vulnerable to Single Event Upsets (SEUs). We show that a large portion (40%-60% for the circuits in our experiments) of the total used LUT configuration bits are don't care bits, and propose to decide the logic values of don't care bits such that soft errors are reduced. Our approaches are efficient and do not change LUT level placement and routing. Therefore, they are suitable for design closure. For the ten largest combinational MCNC benchmark circuits mapped for 6-LUTs, our approaches obtain 20% chip level Mean Time To Failure (MTTF) improvements, compared to the baseline mapped by Berkeley ABC mapper. They obtain 3× more chip-level MTTF improvements and are 128× faster when compared to the existing best in-place IPD algorithm.

Keywords—soft error, mitigation, SRAM-based FPGA, in-place, don't care

I. INTRODUCTION

Soft errors, also called Single Event Upsets (SEUs), which change states of SRAM cells, have posed a major barrier for the reliability of SRAM-based FPGA applications. They are generally caused by high-energy particles striking, such as neutrons coming from cosmic rays or alpha particles emitting from trace impurities in packaging materials, solder bumps, etc. Traditionally, soft errors have received attention for only space applications. However, they are more frequent nowadays in all kinds of applications due to technology scaling and supply voltage reducing. SRAM-based FPGAs are more vulnerable to soft errors compared with ASICs, because most of logic functions and interconnects in FPGAs are implemented by SRAM cells. A soft error can have a permanent impact on SRAM-based FPGAs till the configuration scrubbing is applied [1].

There are a number of studies for the soft error mitigation for SRAM-based FPGAs in the literature. Circuit solutions involve creating radiation hardened cells. Two broad solutions on the architecture level include the Triple Modular Redundancy (TMR), and the Error Checking and Correction (ECC) [2]. The techniques above come with considerable area and power overhead, and constraint themselves in applications with a high demand for the availability. Several studies have demonstrated that SEUs can be mitigated by synthesis approaches with minimal performance, area, and power overhead. Yu et al. [3] claimed the reliability can be

improved by rewriting the circuit using robust block templates. However, the approach changes LUT level placement and routing. To overcome this drawback, Feng et al. [4] proposed an In-Place Reconfiguration (IPR) approach, which maximizes identical LUT configuration bits to reduce the propagation of soft errors seen at a pair of complementary inputs. Cong and Minkovich [5] proposed to choose cuts with more Don't Cares (DCs) during technology mapping. Lee et al. [6] proposed an In-Place Decomposition (IPD) algorithm to decompose or duplicate a logic function in a programmable logic block into two subfunctions and to converge the subfunctions via a carry chain within the same block. Note that those techniques [3]–[6] consider SEUs on LUTs only, and their improvements for the reliability on the chip level would be much smaller when interconnects are taken into consideration. Jose et al. [7] proposed a rewiring algorithm for the robustness of the interconnect of the circuit. However, the interconnect fault model in Jose's paper assumed that there is only one SRAM bit in each net, resulting in that more critical routes are used to replace non-critical ones.

In essence, most previous approaches [3]–[7] on FPGA synthesis for SEUs mitigation are dedicated to incorporate DCs into designs, because an SEU occurring on a DC bit is tolerated and does not cause a failure observed at the primary outputs. For the ten largest combinational MCNC benchmark circuits, we observe that, the percentages of DC bits of the total used LUT configuration bits are about 40% and 60% when the circuits are mapped for 4-LUTs and 6-LUTs¹, respectively. The high ratio makes it difficult to further increase the number of DC bits to mitigate soft errors for those techniques (except for IPD which leverages decomposable LUTs and underutilized carry chains besides LUTs to create DCs, but IPD does not tolerate faults on interconnect at all). This motivates us in this paper to exploit existing DCs in LUTs.

Orthogonal to most previous techniques [3]–[7], we provide In-Place X-Filling (IPF)² algorithms by exploiting

¹The DC set is computed by the windowing technique proposed by Cong et al. [5].

²The term has been used for power-aware Automatic Test Pattern Generation (ATPG) [8], which minimizes power by filling DCs to reduce logic switches of circuits under testing.

existing DCs instead of creating more, to mitigate soft errors in SRAM-based FPGAs. Our algorithms decide states of DC bits to mask soft errors in fanin cones to improve the reliability of the circuit. Note that soft errors in fanin cones includes SEUs on LUT configuration bits and interconnect configuration bits. IPF algorithms not only improve the reliability of LUTs, but also mitigate SEUs on interconnects, which has more impact on the reliability on the chip level. Besides, IPF algorithms are efficient techniques. They do not demand time-consuming Binary Decision Diagram (BDD), Boolean SATisfiability (SAT) [3], [4] Integer Linear Programming (ILP) [6], or Set of Pairs of Functions to be Distinguished (SPFD) [10] to search for the functionally equivalent reconfiguration. IPF algorithms do not change LUT level placement and routing. Therefore, they are in-place synthesis algorithms and suitable for design closure.

For the ten largest combinational MCNC benchmark circuits, our approaches improve chip level Mean Time To Failure (MTTF, the counterpart of the failure rate of a chip on the system level) by 20%, compared to the baseline mapped for 6-LUTs by Berkeley ABC mapper [9]. They obtain $3\times$ more chip-level MTTF improvements and are $128\times$ faster compared to the best in-place IPD algorithm.

The rest of this paper is organized as follows. We start with the introduction to don't care in Section II followed by the IPF algorithms in Section III. The experimental results are summarized in Section IV and the paper is concluded in Section V.

II. INTRODUCTION TO DON'T CARE

The sensitivity of a configuration bit to a soft error is measured by its **criticality**, i.e., the likelihood of the failure of the chip when a soft error occurs on the configuration bit. The sensitivity of a chip to soft errors is the average of the criticalities of all the configuration bits, which is defined as the **failure rate** of the chip in the paper.

DC bits are the bits whose criticalities are zero, i.e., changing a DC bit does not change the functionality and routing of a LUT network due to the limited controllability and observability in a circuit. The care bits are the complement of DC bits in a logic network. There are two kinds of DC bits, and they are satisfiability DC bits and observability DC bits. Satisfiability DC bits are due to non-controllability because some combinations of values are never produced at the fanins of a LUT [10]. We define the corresponding LUT configuration bit as a **satisfiability DC bit**. Observability DC bits are due to non-observability because the LUT's effect on the primary outputs is blocked under some combination of the primary inputs. From the point of view of compatibility, DC bits can also be categorized into compatible DC bits, and incompatible ones. For the first category, a state change of a DC bit does not invalidate other DC bits. Satisfiability DC bits and compatible observability DC bits belong to

this category. For an incompatible DC bit, a state change makes others become care bits. Incompatible observability DC bits belong to this category. In this work, we exploit only satisfiability DC bits for the SEU mitigation for two reasons: (1) satisfiability DC bits take up about 90% of total DC bits for the ten largest combinational MCNC benchmark circuits. (2) satisfiability DC bits are compatible DC bits.

Traditionally, the DC calculation [11] uses the whole network as the context for each node, which makes the calculation unscalable. When applying the similar technique to large circuits, designs demand to be partitioned first. This approach is probably part of some industrial tools [12]. However, the partition is implementation dependent. Mishchenko et al. [13] proposed a windowing technique, which trades quality for runtime. Later on, Cong and Minkovich [5] proposed an improved windowing approach with more accurate DC calculation. In this work, we adopt Cong and Minkovich's work to calculate DCs in a logic network. Any other approach [10]–[12], [14] for DC calculation can be used here.

III. PROBLEM FORMULATION AND ALGORITHMS

A. Problem formulation

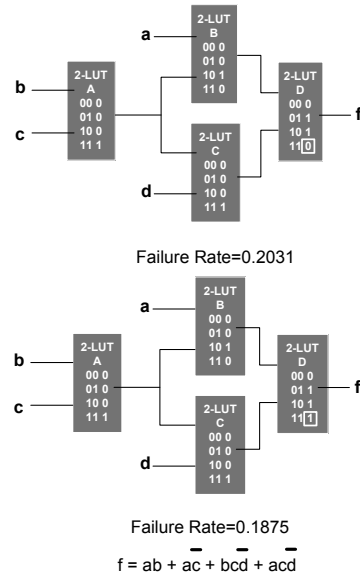


Figure 1. Example of exploiting DCs for reliability

We use Fig. 1 to illustrate how to exploit existing DCs to mitigate soft errors. Given a logic function f , there are two implementations with the same connectivities between LUTs. The configuration bit C_{11} in LUT D is a satisfiability DC bit which is inaccessible under normal situations. Therefore, the function of the circuit is the same no matter whether the state of C_{11} is 0 or 1. Nevertheless, when the

value is filled with 0 (as in the top figure in Fig. 1), the failure rate is higher than when the value is assigned as 1 (as in the bottom figure in Fig. 1). The reason is that, C_{11} in LUT D may be accessed when a soft error occurs in the fanin cone of LUT D ; hence, when C_{11} is filled with the same value as the one outputting more frequently or as that of the bit with highest criticality, even if a soft error occurs, the error may be tolerated by LUT D .

The basic idea hidden behind the example is that a LUT can still output the correct value at most cases even when a satisfiability DC bit is accessed due to a soft error, if the state of the satisfiability DC bit is assigned effectively. We formulate the in-place X-Filling problem as follows, given a circuit, decide states of satisfiability DC bits in all LUTs to increase the likelihood of masking soft errors in their fanin cones thereby to improve the reliability of the circuit.

B. Efficient heuristic algorithms

Given a circuit mapped by K -LUTs, first, we apply Cong and Minkovich’s windowing technique [5] to calculate DCs in a logic network (any other approach [10]–[12], [14] for DC calculation can be used here), and we adopt Monte Carlo simulation for the criticality calculation similar to previous work [3]–[7]. Then, LUTs are enhanced one by one by the following two algorithms in reverse topological order from primary outputs to primary inputs. The reason that we adopt reverse topological order is that, the assignment of a satisfiability DC bit in the current LUT affects assignments of satisfiability DC bits in its fanin cone.

IPF configures each LUT to tolerate soft errors in its fanin cone. Therefore, some soft errors even escaping from the current LUT, can be further masked by LUTs on the next logic level. In other word, the logic masking capability is enhanced by being accumulated among LUTs on different logic levels. IPF targets masking SEUs in fanin cones, no matter whether soft errors are on LUT configuration bits or interconnect configuration bits. Now, come the two strategies for deciding the state of a DC bit.

1) *1 hamming distance strategy*: For each satisfiability DC bit, we first analyze which LUT configuration bits are the candidates for masking. They are the n LUT configuration bits with 1 hamming distance from the satisfiability DC bit, where n is the number of inputs of the LUT. In this strategy, the state of a satisfiability DC bit is filled with the same value as that of the candidate bit with the highest criticality. Hence, whenever there is a soft error in the fanin cone, resulting in that the satisfiability DC is accessed instead of the bit with highest criticality, the output is still correct. As shown in Table I, suppose C_{111} is a satisfiability DC bit. Then, C_{011} , C_{101} , and C_{110} are the candidate bits. C_{011} has the highest criticality. Therefore, C_{111} is assigned as 1.

2) *Output frequency strategy*: In this strategy, a satisfiability DC bit is filled with more frequently outputting logic value among the candidate bits, i.e., the state of the

| Configuration bit | State | Criticality |
|-------------------|--------------------------|-------------|
| 000 | 1 | 0.5% |
| 001 | 1 | 1% |
| 010 | 0 | 1% |
| 011 | 1 | 3% |
| 100 | 0 | 1% |
| 101 | 0 | 2% |
| 110 | 0 | 1.5% |
| 111 | <i>satisfiability DC</i> | 0 |

Table I
THE LUT CONFIGURATION BITS, THEIR STATES AND CRITICALITIES
FOR A 3-LUT

satisfiability DC bit is assigned to 0 or 1 according to the sum of criticalities of all the candidate bits in the On set and Off set in the same LUT. As shown in Table I, although the criticality of C_{011} is the highest, the sum of criticalities of candidate bits in the On set is 3%, and the sum of criticalities of candidate bits in the Off set is 3.5%. As a result, C_{111} is assigned as 0.

IV. EXPERIMENTAL RESULTS

The proposed IPF algorithms are implemented in C++ on a PC with dual core CPU E4400 @ 2.00GHz and 2.0 GB of RAM. We implement IPF algorithms leveraging only satisfiability DCs for the reliability of SRAM-based FPGA designs. All the circuits enhanced by IPF have passed the functional equivalent checking by Berkeley ABC mapper [9].

Using the ten largest combinational MCNC benchmark circuits mapped for 6-LUTs by Berkeley ABC mapper as the baseline, we compare our algorithms with the existing best in-place IPD algorithm [6] for MTTF improvement. MTTF is the counterpart of the failure rate of a chip on the system level, which is the predicted elapsed time to the next failure of a system [15], inversely related to the failure rate of the chip under the same testing platform. In Table II, the column “LUT failure rate” represents the failure rate of the circuit when SEUs occur only on LUT configuration bits. The column “Chip failure rate” signifies the failure rate when SEUs occur on both LUT configuration bits and interconnect configuration bits. The evaluation of interconnect soft errors is based on another work [16], which applies logic simulation to a post-layout FPGA application, targeting the unidirectional routing in the modern FPGAs.

From the table we can see that, IPD can achieve $4\times$ MTTF improvement for “LUT failure rate”, whereas IPF can improve the MTTF by 21%. Nevertheless, when considering the interconnect errors, IPD obtains merely 7% chip level MTTF improvement, whereas the improvement by IPF can still reach to 20%. It is because IPF implicitly masks SEUs in interconnects, which has more impact on the chip level MTTF. IPF targets tolerating soft errors in fanin cones, which reduces SEUs in interconnects in fanin cones, too.

| circuits | LUT# | LUT failure rate (%) | | | | Chip failure rate (%) | | | | Runtime (s) | | |
|------------|------|----------------------|--------|---------|--------|-----------------------|--------|---------|--------|-------------|---------|---------|
| | | ABC | IPD | IPF | | ABC | IPD | IPF | | IPD | IPF | |
| | | | | Hamming | Output | | | Hamming | Output | | Hamming | Output |
| alu4 | 507 | 0.34 | 0.09 | 0.24 | 0.24 | 0.36 | 0.33 | 0.28 | 0.28 | 1466 | 19.38 | 19.54 |
| apex2 | 687 | 0.27 | 0.03 | 0.24 | 0.24 | 0.25 | 0.22 | 0.23 | 0.23 | 1137 | 6.31 | 6.29 |
| apex4 | 594 | 1.55 | 0.34 | 0.96 | 0.96 | 1.64 | 1.50 | 1.39 | 1.39 | 1430 | 15.44 | 17 |
| des | 556 | 1.79 | 1.20 | 1.71 | 1.69 | 4.16 | 4.07 | 3.66 | 3.63 | 2022 | 5.69 | 5.82 |
| ex1010 | 668 | 1.21 | 0.28 | 1.13 | 1.13 | 1.62 | 1.52 | 1.43 | 1.43 | 1635 | 22.46 | 23.55 |
| ex5p | 384 | 0.70 | 0.20 | 0.67 | 0.67 | 0.93 | 0.89 | 0.88 | 0.88 | 795 | 4.63 | 6.23 |
| misex3 | 490 | 0.54 | 0.10 | 0.38 | 0.38 | 0.58 | 0.54 | 0.38 | 0.38 | 1235 | 16.71 | 17.04 |
| pdcc | 1515 | 1.05 | 0.12 | 0.90 | 0.90 | 1.75 | 1.63 | 1.38 | 1.38 | 3429 | 854.24 | 1073.3 |
| seq | 705 | 0.66 | 0.11 | 0.52 | 0.52 | 0.73 | 0.67 | 0.59 | 0.59 | 1659 | 6.24 | 6.35 |
| spla | 1436 | 1.28 | 0.18 | 1.09 | 1.09 | 2.02 | 1.89 | 1.66 | 1.66 | 3270 | 765.42 | 924 |
| Ratio | - | 1 | 23.93% | 83.06% | 82.97% | 1 | 93.20% | 83.42% | 83.34% | 1 | 128.64× | 121.48× |
| MTTF Ratio | - | 1 | 4.18 | 1.20 | 1.21 | 1 | 1.07 | 1.20 | 1.20 | - | - | - |

Table II
FAILURE RATE COMPARISON OF SEU MITIGATION TECHNIQUES ON CHIP LEVEL

Especially, when error propagations in interconnects are similar to those between LUTs. E.g. “misex3”, the MTTF improvement is 52.6% on the chip level. On the other hand, IPD does not tolerate soft errors on interconnects at all, i.e., the 7% chip level MTTF improvement is solely from the 4× LUT MTTF improvement. Besides, IPF is 128× faster than IPD, which makes IPF more scalable for the practical cases from industry. The reason for the faster runtime is due to that our approaches do not require time-consuming BDD, SAT, ILP, or SPFD for Boolean matching.

V. CONCLUSIONS AND FUTURE WORK

In this work, we propose in-place X-Filling algorithms by exploiting DCs to improve the reliability of circuits without time-consuming BDD, SAT, ILP, or SPFD based Boolean matching for functional equivalence checking. For the ten largest combinational MCNC benchmark circuits mapped for 6-LUTs, our approaches achieve 20% chip level MTTF improvements, compared to the baseline mapped by Berkeley ABC mapper. They obtain 3× more chip-level MTTF improvements and are 128× faster when compared to the existing best in-place IPD algorithm.

The more satisfiability DC bits are changed, the better MTTF improvement is. Increasing the number of compatible DC bits during logic synthesis will be leveraged to obtain targeted trade-off between reliability and area in the future. Compatible observability DC will be also considered. We will also study how to extend the algorithms to sequential circuits and multiple soft errors. The key for sequential circuits is to build a model for criticality calculation. Besides, the impact of technology scaling makes multiple soft errors a big concern. The difficult part to extend it to multiple errors is how to consider the correlation between errors. Applying interconnect-aware criticality explicitly to IPF may lead to more chip-level MTTF improvement, which will be tried as the first option of our future work.

REFERENCES

- [1] F. Sturesson, S. Mattsson, and C. Carmichael, “Heavy ion characterization of seu mitigation methods for the virtex fpga,” in *6th European Conference on Radiation and Its Effects on Components and Systems*, Feb 2001, pp. 414–419.
- [2] S. Tam, “Single Error Correction and Double Error Detection,” in <http://www.xilinx.com>, Aug 2006.
- [3] Y. Hu and Z. Feng and L. He and R. Majumdar, “Robust FPGA Resynthesis Based on Fault-Tolerant Boolean Matching,” in *ICCAD*, Nov 2008, pp. 706–713.
- [4] Z. Feng and Y. Hu and L. He and R. Majumdar, “IPR: In-Place Reconfiguration for FPGA Fault Tolerance,” in *ICCAD*, Nov 2009, pp. 105–108.
- [5] J. Cong and K. Minkovich, “LUT-Based FPGA Technology Mapping for Reliability,” in *DAC*, 2010.
- [6] J. Lee and Z. Feng and L. He, “In-Place Decomposition for Robustness in FPGA,” in *ICCAD*, 2010.
- [7] M. Jose and Y. Hu and R. Majumdar and L. He, “Rewiring for robustness,” in *DAC*, Jun 2010.
- [8] J.-Y. Lee and Y. Hu and R. Majumdar, “Simultaneous test pattern compaction, ordering and x-filling for testing power reduction,” in *ISQED*, 2009.
- [9] “ABC: A system for sequential synthesis and verification,” in <http://www.eecs.berkeley.edu/~alanmi/abc/>.
- [10] A. Mishchenko and R. Brayton and J. Jiang and S. Jang, “Scalable Don’t-Care-Based Logic Optimization and Resynthesis,” in *FPGA*, 2009.
- [11] H. Savoj, “Don’t care in multi-level network optimization,” in *Ph.D. Dissertation, UC Berkeley*, May 1992.
- [12] A. Mishchenko and R. K. Brayton, “SAT-based complete don’t-care computation for network optimization,” in *DATE*, 2005, pp. 412–417.
- [13] A. Mishchenko, J. Zhang, S. Sinha, J. Burch, R. Brayton, and M. Chrzanowska-Jeske, “Using simulation and satisfiability to compute flexibilities in Boolean networks,” *TCAD*, vol. 25, no. 5, pp. 743–755, 2006.
- [14] K.L. McMillan, “Don’t-care computation using k-clause approximation,” in *IWLS*, 2005.
- [15] J. Jones, “Integrated logistics support handbook,” in *McGraw-Hill Professional*, Feb 1998.
- [16] Naifeng Jing and Ju-Yueh Lee and Zhe Feng and Weifeng He and Zhigang Mao and Shi-Jie Wen and Rick Wong and Lei He, “Quantitative SEU Fault Evaluation for SRAM-Based FPGA Architectures and Synthesis Algorithms,” in *FPL*, Sep 2011.