University of California

Los Angeles

Robust Design for FPGAs

A dissertation submitted in partial satisfaction

of the requirements for the degree

Doctor of Philosophy in Electrical Engineering

by

Ju-Yueh Lee

2013

Abstract of the Dissertation

Robust Design for FPGAs

by

Ju-Yueh Lee

Doctor of Philosophy in Electrical Engineering

University of California, Los Angeles, 2013

Professor Lei He, Chair

Field programmable gate arrays (FPGAs) use memory cells, primarily static random-access memory (SRAM) cells to implement field programmability for logic and interconnect, which is a preferable platform due to its high performance and low non-recurring engineering cost. To increase the logic density and the integration capability, modern FPGAs use ever advancing process technologies and smaller devices. However, smaller devices are more vulnerable to environmental upsets caused by high energy particle hits and internal noise, and may change their logic states as a result. Such an upset is called a "Soft Error", which is recently acknowledged as the most critical reliability issue for FPGAs.

In the contexts of system failure and circuit functional failure, this dissertation studies the effects of soft errors caused by environmental upsets of modern FPGA architectures and presents novel methods for soft error tolerance to improve FPGA robustness from system to circuit levels. This dissertation first presents a comprehensive soft error analysis framework for SRAM-based FPGAs. By using a stochastic soft error model, the soft error sensitivities toward functional failures of a design implemented on an FPGA are quantitatively identified. At the system level, a novel FPGA configuration memory (CRAM) soft error mitigation technique by Heterogeneous CRAM Scrubbing (HCS) is proposed. Next, at the circuit level, two in-place resynthesis techniques are proposed: In-Place Decomposition

(IPD) for FPGA logic elements and In-Place inVersion (IPV) for FPGA interconnect components. In contrast to existing redundancy techniques, the proposed techniques are attractive because they do not change circuit global placement and routing and hence, have negligible cost on performance, area, and design closure. Furthermore, a co-optimization algorithm leveraging the proposed IPV technique for soft error and leakage reduction is proposed. Finally, this dissertation also demonstrates validation of the IPV technique on an industrial FPGA.

The dissertation of Ju-Yueh Lee is approved.

Glenn Reinman

Puneet Gupta

M.-C. Frank Chang

Lei He, Committee Chair

University of California, Los Angeles

2013

To My Family.

Table of Contents

List of Figures

List of Tables

Acknowledgments

There are many individuals I would like to gratefully acknowledge. This dissertation could not be completed without their support and help.

Firstly, I would like to give my greatest gratitude to Prof. Lei He, my adviser, for his great patience and support. Over the years, he continuously guided me not only to improve my problem solving skills, but also to develop a good personality and become a better researcher. I would not be able to finish my dissertation without his tremendous help and encouragement.

I would like to thank my dissertation committee members, Prof. Frank Chang, Prof. Puneet Gupta, and Prof. Glenn Reinman, for their guidance and feedback, which helped improve the quality of my research and dissertation.

I would also like to thank Prof. Rupak Marjumdar. I enjoyed collaborating with him during my first year research study on optimization algorithms. His inspiring comments and ideas were always driving me towards higher quality research.

I am very grateful to have had the opportunity to collaborate with many great researchers in UCLA Design Automation lab. I enjoyed collaborating with them during all of the amazing research projects we have worked on together. Especially, I would like to thank Dr. Yu Hu, Dr. Feng Zhe, Mr. Naifeng Jing, Mr. Juexiao Su, Mr. Haik Kalantarian, and Mr. Cheng-Ru Chang. Mr. Naifeng Jing and I jointly worked on many research projects including Quantitative Soft Error Evaluation Platform and In-Place inVersion for FPGAs as presented in Chapter 2 and 5. Mr. Cheng-Ru Chang helped on the verification of the Heterogeneous CRAM Scrubbing technique discussed in Chapter 3. The work on In-Place Decomposition in Chapter 4 included significant contributions from Dr. Yu Hu, Dr. Feng Zhe, and Mr. Haik Kalantarian. The validation of the proposed technique on industrial FPGAs was supported by Mr. Juexiao Su. I gratefully acknowledge them for helping to complete this dissertation.

I would like to thank Dr. Austin Lesea, Dr. Steve Trimberger, and Dr. Ashok Moghe for their support and collaboration regarding our research during my internship at Xilinx and Cisco.

Finally, I am grateful for my family, especially for my parents and my wife, for providing their continual support and encouragement.

Vita

1982	Born, Taipei, Taiwan
2000-2004	B.S., Computer Science National Chiao Tung University Hsinchu, Taiwan
2004-2006	M.S., Electrical Engineering National Taiwan University Taipei, Taiwan
2008-present	Ph.D. program Electrical Engineering University of California, Los Angeles, CA, USA
2010	Research Intern Xilinx Inc. 2100 Logic Drive, San Jose, CA, USA
2010	Research Intern Cisco Inc. 821 Alder Dr., Milpitas, CA, USA

Publications

Ju-Yueh Lee, Yu Hu, Rupak Majumdar, and Lei He,“ Simultaneous test pattern compaction, ordering and X-filling for testing power reduction”, *ISQED*, 2009, pp. 702-707

Ju-Yueh Lee, Yu Hu, Rupak Majumdar, Lei He, and Minming Li, “Fault-Tolerant Resynthesis for Dual-Output LUTs”, *SELSE*, 2009.

Ju-Yueh Lee, Yu Hu, Rupak Majumdar, Lei He, and Minming Li, “Fault-tolerant resynthesis with dual-output LUTs”, *ASP-DAC*, 2010, pp. 325-330.

Samuel B. Luckenbill, Ju-Yueh Lee, Yu Hu, Rupak Majumdar, and Lei He, “RALF: Reliability Analysis for Logic Faults - An exact algorithm and its applications”, *DATE*, 2010, pp. 783-788.

Ju-Yueh Lee, Zhe Feng, and Lei He, “In-place decomposition for robustness in FPGA.” *ICCAD*, 2010, pp. 143-148.

Wenyao Xu, Jia Wang, Yu Hu, Ju-Yueh Lee, Fang Gong, Lei He, and Majid Sarrafzadeh, “In-Place FPGA Retiming for Mitigation of Variational Single-Event Transient Faults”, *IEEE Trans. on Circuits and Systems*, 2011.

Naifeng Jing, Ju-Yueh Lee, Chun Zhang, Jiarong Tong, Zhigang Mao, and Lei He, “Fault modeling and characteristics of SRAM-based FPGAs”, *FPGA*, 2011.

Naifeng Jing, Ju-Yueh Lee, Zhe Feng, Weifeng He, Zhigang Mao, Shi-Jie Wen, Rick Wong, and Lei He, "Quantitative SEU Fault Evaluation for SRAM-Based FPGA Architectures and Synthesis Algorithms", *FPL*, 2011, pp. 282-285.

Naifeng Jing, Ju-Yueh Lee, Weifeng He, Zhigang Mao, and Lei He, "Mitigating FPGA interconnect soft errors by in-place LUT inversion", *ICCAD*, 2011, pp. 582-586.

Naifeng Jing, Ju-Yueh Lee, Zhe Feng, Weifeng He, Zhigang Mao, and Lei He, "SEU fault evaluation and characteristics for SRAM-based FPGA architectures and synthesis algorithms", *ACM Transactions on Design Automation of Electronic Systems*, 2012.

Ju-Yueh Lee, Cheng-Ru Chang, Naifeng Jing, Juexiao Su, Shi-Jie Wen, Rick Wong, and Lei He, "Heterogeneous configuration memory scrubbing for soft error mitigation in FPGAs", *FPT*, 2012, pp. 23-28.

Ju-Yueh Lee, Juexiao Su, and Lei He, "In-Place LUT Polarity Inversion to Mitigate Soft Errors for Virtex FPGAs", submitted to *Embedded System Letters (ESL).*

Ju-Yueh Lee, Naifeng Jing, and Lei He, "Soft Error and Leakage Mitigations for FPGAs using In-Place LUT Polarity Inversion", submitted to *FPGA*, 2014.

CHAPTER 1

Introduction

1.1 Introduction to Field Programmable Gate Arrays (FPGAs)

An FPGA consists of a 2D array of Configurable Logic Blocks (CLBs) that are selectively connected by global routing as illustrated in Figure 1.1. The CLB architecture can be characterized by (k, N), i.e., it contains a cluster of N Look-Up-Tables (LUTs) with k inputs. These LUTs use configuration memory (CRAM) to implement desired functions and are connected by local MUXes, allowing CLB inputs and outputs to be routed to and from each LUT within the cluster. Global routing resources connect CLBs through connection boxes and switch boxes by wires deployed in directional channels with a width of W (the number of tracks).

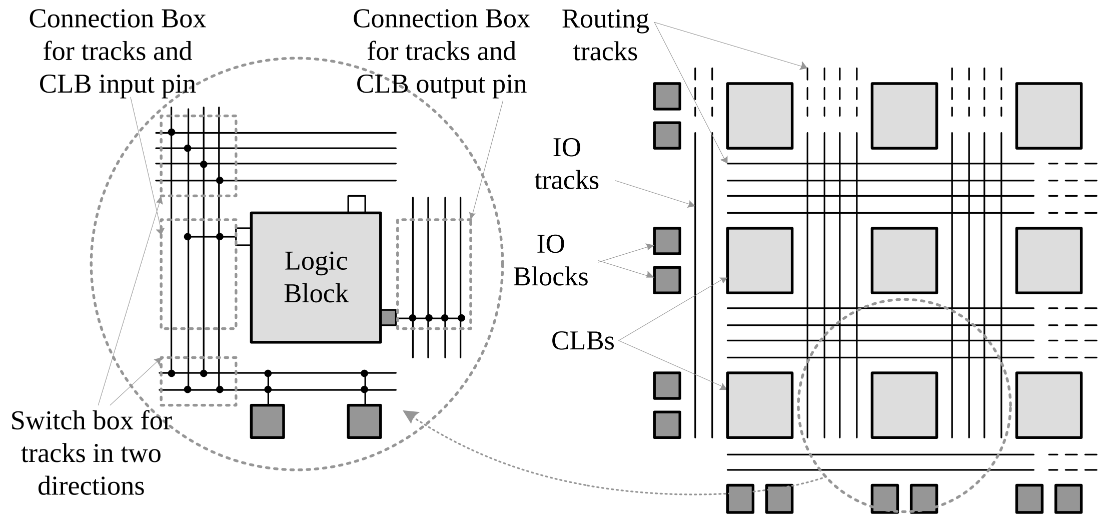

Connection Box for tracks and CLB input pin
Connection Box for tracks and CLB output pin
Routing tracks
IO tracks
IO Blocks
CLBs
Logic Block
Switch box for tracks in two directions

Figure 1.1: Island style FPGA architecture overview

1.2 Background and Motivation

SRAM-based FPGAs are preferable developing platforms due to their rapid time to market and low non-recurring engineering cost. Benefiting from the continuous technology scaling in the past decades, SRAM-based FPGAs enable higher density, lower power consumption and faster speed to implement more complex designs, which are increasingly used not only for rapid prototyping but also for deployed circuit systems, such as internet line cards and routers. At the same time, higher integration demands a large quantity of memory cells. As Fig. 1.2 shows, the amount of memory elements used in Xilinx Virtex FPGA families have rapidly increased, while the most used elements are CRAM cells to implement the programmable logic blocks for functionality and routing resources for interconnects [Xild]. However, augmented memory size leads to degraded reliability because these memory cells are vulnerable to soft errors. Consequently, soft error tolerance has gained growing significance for modern SRAM-based FPGAs and become the essential step towards dependable FPGA designs and applications.

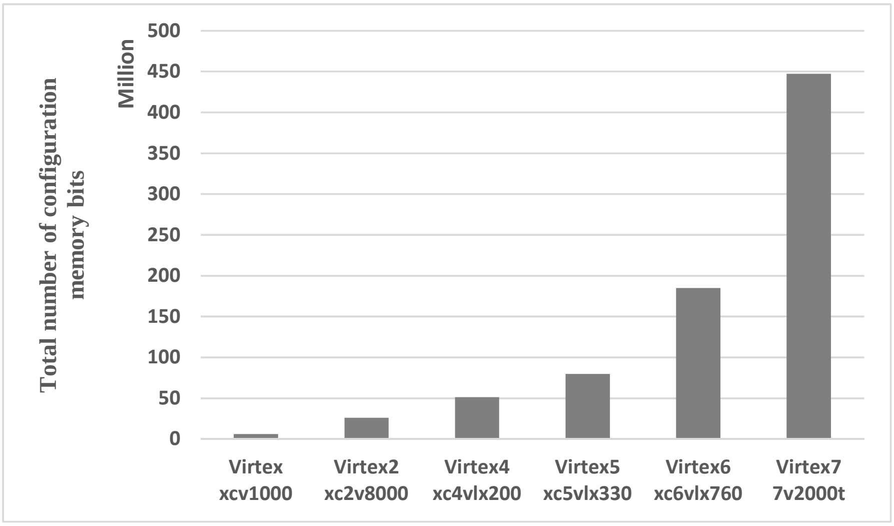


Figure 1.2: CRAM sizes of Xilinx Virtex FPGA families.

Single Event Upset (SEU) is among the major causes of soft errors in SRAM-based FPGAs. When a high energy particle, such as the secondary particles liberated when a neutron collides with a silicon atom or the alpha particle emitted from a contaminant of an electronic device, strikes a static memory element, particularly an SRAM cell or a flip-flop, if the charge deposited in the particle is large enough (called the critical charge), it can affect the voltage of a circuit node and may change the logic value stored in a static memory element, i.e. causing inadvertent $0 \rightarrow 1$ or $1 \rightarrow 0$ logic switch. In addition, power supply disturbance and electromagnetic interference are may also cause SEUs. In contrast to hard errors, such as device defects, soft errors typically imply a temporary malfunction or interruption of operation.

In general, no shielding solution is available because it requires more than 30 meters of water to completely protect electronic equipment from these high energy particles. Furthermore, a well known fact of soft error effects is that at 65nm devices began to grow unreliable and that trend continued, which has become a major cause of circuit malfunctions. As device dimensions shrink, the device supply voltage aggressively scales down in order to minimize the power consumption such that the critical charge of a circuit node decreases, rendering steadily increasing soft error rate (bit flip). Although the growth of soft error rate for SRAM cells slows down primarily because of the saturation of the supply voltage, the chip level soft error rate continuously increases due to dramatic density growth as Fig. [ITY10] shows, depicting the soft error rate under different process technologies [Bid10]. Notably, the chip level soft error rate has increased by 7× from 130nm to 22nm process technologies.

Redundancy is the most common soft error mitigation technique to achieve a higher level of reliability [NSB12]. The first level of protection is the ability to know an error has occurred. To achieve this feature, dual rail logic implementations, i.e. replication of logic functions or sub-functions, have been proposed

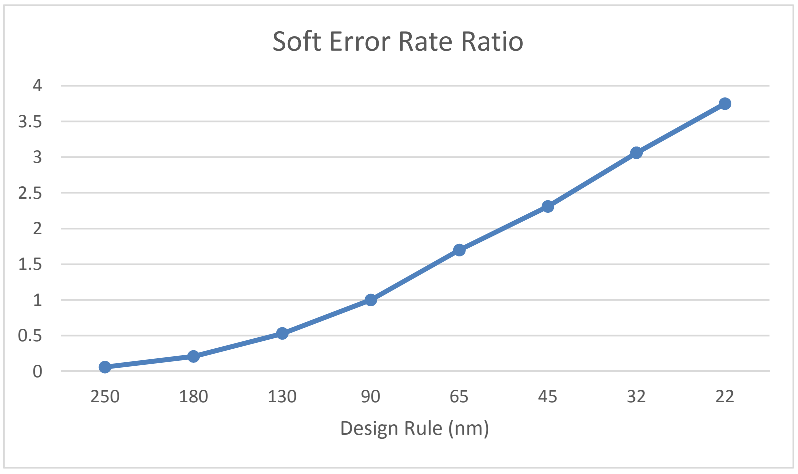


Figure 1.3: Soft error rate scaling trend.

for error detection [DR08, FAL12, LSR11]. The second level of protection is recovery from errors. For higher reliability requirements, triplication with a voting system is suggested, such as Triple Modular Redundancy (TMR) technique [LV62, KSC05a, SRK04, GKB11, PCG08, ZIO10, PCC08, KSC05b], which has been widely used in mission critical applications. However, the above redundancy techniques typically require 2× to 6× power and area overhead. Particularly, the extra circuit cost can be overtly expensive while not all applications require the highest level of reliability.

Another well known technique includes resynthesis for robustness in the Computer-Aided Design (CAD) flow. In a typical logic synthesis, the optimization objectives are usually focused on performance, power consumption, and area. To further improve reliability, logic resynthesis algorithms for reliability have been proposed and they are applied after the typical logic synthesis flow [HFH08, FHH09]. However, they may change the circuit structure, bringing overhead to the physical design, i.e. placement and routing, and thus require additional circuit cost and longer synthesis iterations for design closure.

In order to meet the industrial demands for reliability while maintaining a minimum circuit cost, in-place local rewriting is the most popular technique in logic resynthesis. An in-place resynthesis alters local components, such as logic gates or look-up-tables (LUTs), with minimum or no change to the global placement and routing, such that the optimization effort from the previous stage is preserved. Therefore, they do not require extra circuit cost and have minimum overhead on area, power, and performance. In Application-Specific Integrated Circuits (ASICs), this is usually referred to as Enginerring Change Orders (ECOs). In FPGAs, because of their flexible configurability, higher design freedom and potentials in in-place resyntheses for further optimization is allowed. Due to low overhead, in-place methods are preferred in non-mission-critical applications, such as networking or information systems.

The overall objective of this research is to exploit and develop optimization techniques for reliability against soft errors for SRAM-based FPGAs with minimum or no cost. Specific goals of this research are as follows.

- A systematic study of the soft error impact on modern FPGA architectures.
- Exploit efficient soft error mitigation methods with minimum or no cost.

1.3 Dissertation Contributions

In this dissertation, a robust FPGA design flow is proposed as Fig. 1.4 illustrates. Compared to most of the existing techniques which do not carefully consider the heterogeneity of the soft error effects and cause extensive overhead, this dissertation studies soft errors thoroughly in various FPGA components and architectures. Targeting at both system and circuit levels, several soft error tolerant techniques are proposed. In particular, the focus of the proposed techniques is to mitigate soft error impact while maintaining minimum or no cost.

Specifically, the primary contributions of this dissertation are as follows:

- This dissertation presents a comprehensive SEU induced soft error analysis

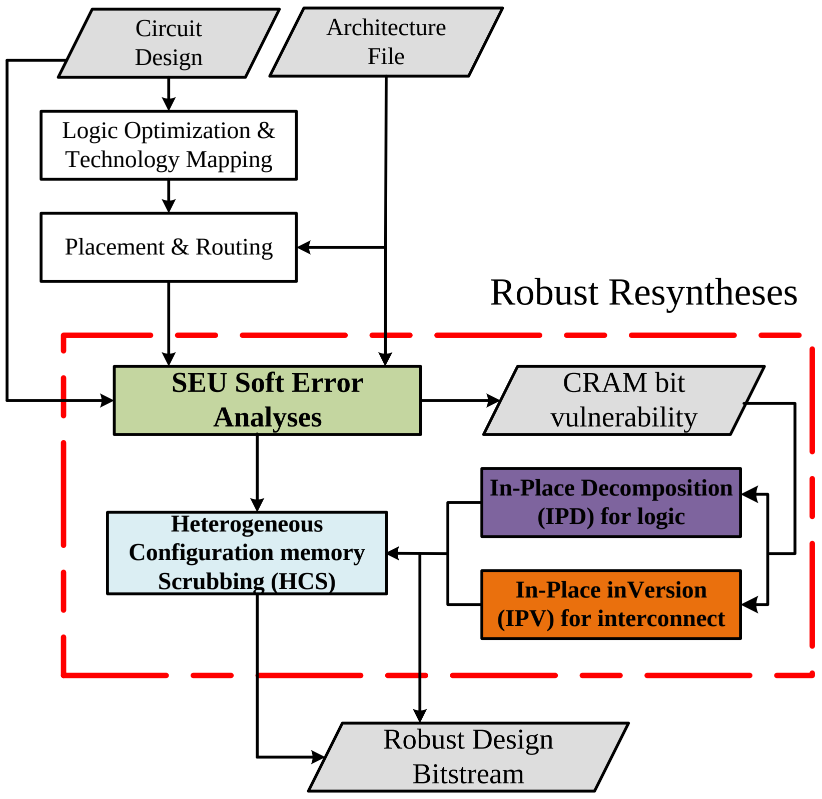

Circuit Design
Architecture File
Logic Optimization & Technology Mapping
Placement & Routing
Robust Resyntheses
SEU Soft Error Analyses
CRAM bit vulnerability
In-Place Decomposition (IPD) for logic
Heterogeneous Configuration memory Scrubbing (HCS)
In-Place inVersion (IPV) for interconnect
Robust Design Bitstream

Figure 1.4: Robust resynthesis CAD flow

framework for SRAM-based FPGAs. By using a stochastic soft error model, it quantitatively evaluates the sensitivity of a soft error to functional failures. It performs soft error analyses with respect to different FPGA components, architectures, and applications, which yields useful insights for developing robust FPGA designs and CAD algorithms. Our detailed analysis reveals that soft errors on interconnect carries more functional failures and thus explicit consideration of soft error mitigation on interconnect is necessary.

- By quantifying the sensitivity of SEUs to failures at system level, this dissertation for the first time estimates the soft error tolerance improvement by CRAM scrubbing. Based on system failure sensitivity information, this dissertation presents Heterogeneous CRAM Scrubbing (HCS) for FPGAs. Our experiment demonstrates that HCS increases 60% of improvement on system mean-time-to-failure (MTTF) compared to the conventional homogeneous CRAM scrubbing approach.

- At circuit level, this dissertation first presents an In-Place Decomposition (IPD) resynthesis technique leveraging modern FPGA architecture features in the programmable logic blocks (PLBs). This method focuses on soft errors on logic elements, i.e. look-up-tables (LUTs), which decomposes a logic function on an LUT into two or more subfunctions and combines them via converging logic to reduce the soft error rate caused by SEUs. Such decomposition can be implemented using the decomposable LUT and carry chain in the original PLB without changing the PLB-level placement and routing. Experimental results demonstrate IPD improves MTTF by 4.55$\times$ on average for FPGA architecture similar to Altera Stratix-IV.

- For interconnect robustness, this dissertation presents In-Place LUT polarity inVersion (IPV), to exclusively mitigate the SEU impact on FPGA components of modern unidirectional routing structures. This technique leverages the error masking scheme of FPGA routing multiplexers (MUXes). By developing efficient algorithms to reassign LUT polarities of IPV, experimental results demonstrate 4$\times$ MTTF improvement.

- Furthermore, a co-optimization algorithm for soft errors and leakage power consumption reduction using IPV technique is presented, which is able to achieve the same level of soft error improvement plus 30% leakage power reduction on average.

- Lastly, this dissertation also presents a validation of IPV technique on an industrial FPGA. Using a Xilinx Virtex FPGA as a case study, we propose a new soft error model and a modified IPV algorithm based on static signal probability, where the experimental results show that IPV reduces MTTF by 2$\times$ approximately.

In particular, all of the proposed techniques in this dissertation do not change circuit global placement and routing, and therefore lead to a fast design closure with minimum overhead.

1.4 Dissertation Outline

The remainder of this dissertation is organized as follows: Chapter 2 first discuses the soft error impact and modeling for SRAM-based FPGA. Then, we introduce our SEU soft error analysis framework. In Chapter 3, we present our novel CRAM scrubbing approach, Heterogeneous CRAM Scrubbing, to explicitly improve FPGA CRAM reliability against failures at the system level. Chapter 4 presents In-Place Decomposition (IPD), which specifically improves reliability of FPGA logic elements against soft errors. Chapter 5 presents for the first time an in-place resynthesis technique, In-Place LUT inVersion (IPV), specifically targeting on reliability improvement for FPGA routing components. In addition, the proposed IPV technique not only mitigates soft errors, but also leakage for FPGA routing multiplexors, and a co-optimization algorithm for soft errors and leakage is presented in Chapter 6. Moreover, we present validation of the IPV algorithm on Xilinx Virtex FPGA in Chapter 7. Finally, Chapter 8 concludes the work and discuses the future direction.

CHAPTER 2

Quantitative SEU Soft Error Evaluation for SRAM-Based FPGA Architectures

In order to design robust FPGA circuits, architectures, or synthesis algorithms with respect to SEU, it is required to investigate the SEU impact and identify the SEU-sensitive circuit elements in FPGAs [CVR03, BBB04a]. Previous works estimating the soft error sensitivity, or failure rate, are mainly simulation based [BCD11, RRV02, HAW05, BCD12, VSC04], hardware emulation-based [LBN10, KAJ12, ITA12], radiation-based, or a combination of the techniqeus [JCG03, GCZ03, BBB04b]. However, it is hard for those methods to predict the failure on a specific CRAM bit during design time. Software-based simulation and analytical approaches have also been proposed [KPM07, AT05, ATM07, AT07], but without explicit consideration for interconnect. In addition, these existing estimating approaches assume a bidirectional routing architecture. However, modern FPGA routing architecture has shifted from conventional bidirectional routing towards unidirectional routing, where the failure sensitivity has not yet been studied.

Targeting generic FPGA architectures and applying logic simulation on a post-layout FPGA application, we develop a comprehensive SEU fault evaluation framework for SRAM-based FPGAs. It can quantify the failure rate, that is, the probability of an SEU induced soft error that causes a system failure on a basis of each CRAM bit. We analyze the SEU induced soft errors in various circuit resources, and predict their impact on circuit functionality with a unified metric of soft error rate for each configuration bit. In the proposed framework, the soft error is analyzed based on detailed post-layout circuit information. By exploring different architectural combinations, we reveal different failure sensitivities for various resources and observe that architectural variations can influence the overall reliability notably. We envision that our work will cast useful insights for more robust FPGA circuit and architecture. In the later chapters, we will use this platform to develop and evaluate SEU mitigation techniques.

2.1 SEU Induced Soft Errors in FPGA

Different circuit elements in an FPGA behave differently when affected by an SEU. In this section, we study the fault behavior of the CRAM bits in LUTs, local routing MUXes, and global routing PIPs according to the generic architecture description preceding and their micro-architecture typically used in FPGAs. Both the bidirectional and unidirectional routing architectures are considered. Note that previous work treats multiple CRAM bits on one net as a single node [AT05]. In our framework, each bit is evaluated based on the detailed post-layout circuit for more realistic predication of its failure sensitivity.

2.1.1 SEU on Look-Up-Tables (LUTs)

A typical implementation of an LUT is illustrated in Figure 2.1, where an LUT is the basic configurable logic element in FPGAs. For a k input LUT, there are 2^k CRAM bits for the desired logic function. The k inputs make a cascading selection on these bits and provide one bit as the final LUT output.

The behavior of an SEU on an LUT CRAM bit is straightforward. An SEU on any CRAM bit of the 2^k bits may flip the LUT output when the affected bit happens to be accessed under certain input patterns. The affected value may further propagate throughout the logic network and finally result in a functional

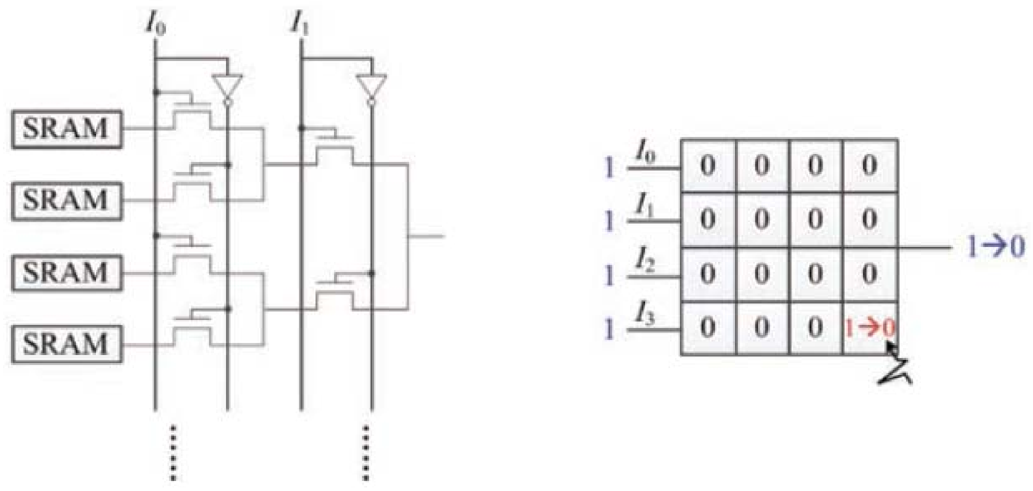


Figure 2.1: SEU on an LUT.

failure when it reaches the primary outputs of the circuit.

2.1.2 SEU on Intra-CLB Routing

The intra-CLB routing connects the CLB inputs (and outputs) to LUT inputs (and outputs) within each CLB. In the generic architecture to be evaluated in this work, we allow for a full connection for all the inputs and outputs. Local routing primarily uses MUXes for signal selection. Therefore, for the typical implementation of a local routing network, as illustrated in Figure 2.2, each k input pin of the N LUTs has its own input MUX with several CRAM bits, which are programmed to select any of the CLB inputs, as well as the N outputs of the LUTs within the same CLB. At the same time, the N LUT outputs can also be connected to any of the CLB outputs via the local routing MUXes. The MUXes enable the arbitrary interconnecting capability within each CLB.

Figure 2.2 further shows a typical structure for an encoded MUX under an SEU. The CRAM bits controlling this MUX select one signal to drive the MUX output. Once affected by the SEU, one of the encoded CRAM bits flips its state and thus select an erroneous input signal onto the output. Different from the

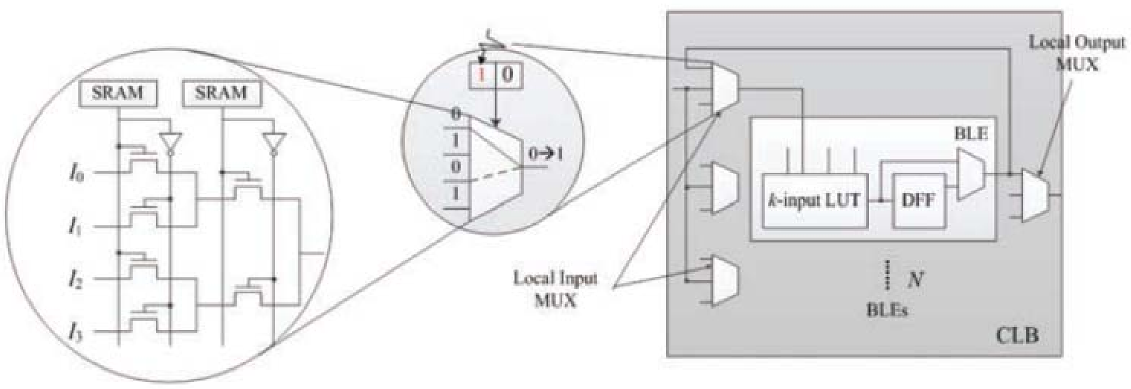


Figure 2.2: SEU on an input MUX inside a logic block.

SEU on an LUT, the affected configuration bit can always mistakenly select an irrelevant signal disregarding the logic status of the net or the entire circuit. The irrelevant signal has the chance of being propagated to primary outputs and leads to a functional failure.

2.1.3 SEU on Bidirectional Routing

Conventionally, inter-CLB routing is typically interconnected via bidirectional pass transistors [SV06, GB07], and its connectivity within a connection box or a switch box is configured by CRAM bits. Once affected by an SEU, these bits either Temporarily Switch-To-0 (TST0) or Switch-To-1 (TST1).

Figure 2.3 illustrates an SEU induced open fault, that is, TST0, which breaks the originally connected wires at the faulty point in the connection box or switch box. The outgoing wire from the open point carries an unknown signal whose value depends on the FPGA circuit. Without loss of generality, we assume that the broken net will be tied off to either Vdd or Gnd [RCS05] such that the following transistors can be prevented from being conducted as short circuits, which should be avoided in CMOS designs. However, the tied-off value may bring an input error to the immediate fan-outs of the faulty point if it is different from the desired value.

The error may propagate through the fan-out network and finally be observed at primary outputs as a circuit failure.

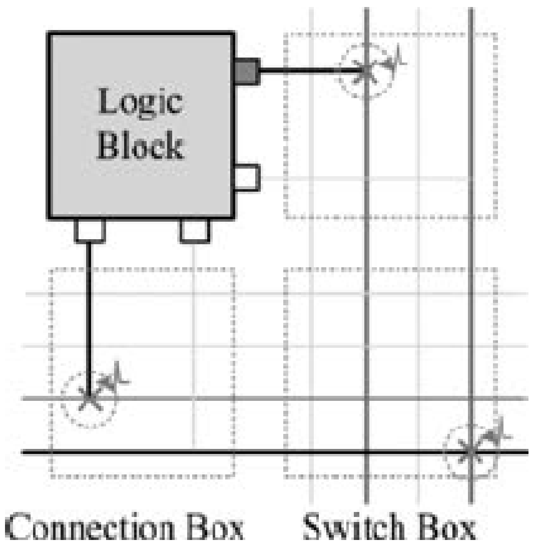


Figure 2.3: Open fault in connection box or switch box in a schematic view.

SEU induced short fault, that is, TST1, bridges two adjacent wires when they both pass through the same connection box or switch box. Figure 2.4 illustrates an example, where two nets are bridged due to an SEU in the top-right connection box. In fact, bridging may not always inject faults into the circuit. It depends on the driving logic and strengths along the two nets. If both the driving signals are the same, the signal at the faulty point is forwarded without fault. Only when the two nets are driven by opposite logic values is the net bridging likely to cause circuit failures.

For the two nets bridged caused by an SEU, the key concern is what logic values are forwarded to the following logic from the bridging point. That is, the steady logic values of d_1 d_4 are concerned, as the example in Figure 2.4 shows. To get their values, we derive the equivalent interconnect circuit as in Fig. 2.5(a), where the resistance and capacitance are respectively modeled according to the physical layout after placement and routing. Although the driving signals

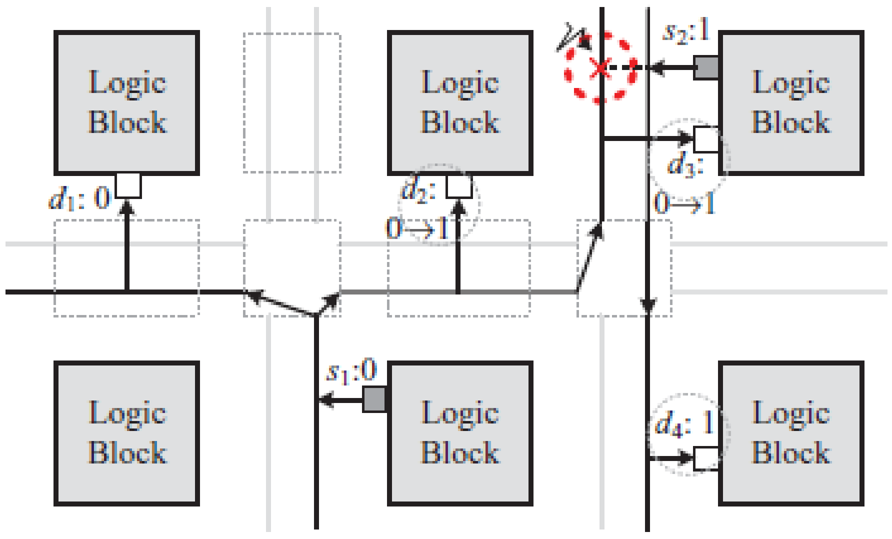


Figure 2.4: Short fault in a schematic view (bridging at the crossing point).

of the two nets s_1 and s_2 change at circuit frequency, the transition time of the faulty signal is typically small compared to that of the clock period. As a result, we can statically analyze the circuit by ignoring the interconnecting and sink capacitance from Figure 2.5(a), according to the study in [GYM05], on a bridged circuit. Then, a resistance network with R_1, R_2, and R_b is left, as in Figure 2.5(b), whose resistance values are calculated by the physical architectural parameters and routing distances of the concerned wires from their respective driving blocks. Hence, the signal values at the faulty point can be calculated by a voltage dividing between Vdd and Gnd along the wires, and the logic values of d_1 d_4 can be obtained accordingly.

In addition, the impact of bridging may vary along the affected wires. Still considering net s_1 in Figure 2.4, the logic values on points d_2 and d_3 are possible to flip due to voltage dividing, while d_1 may remain its value because it is nearer to its driving source at s_1. Similarly, d_4 may also be flipped depending on its driving distance to its source s_2. This behavior makes the bridging fault quite different from that of the LUTs. It is likely that a single wire is decomposed to

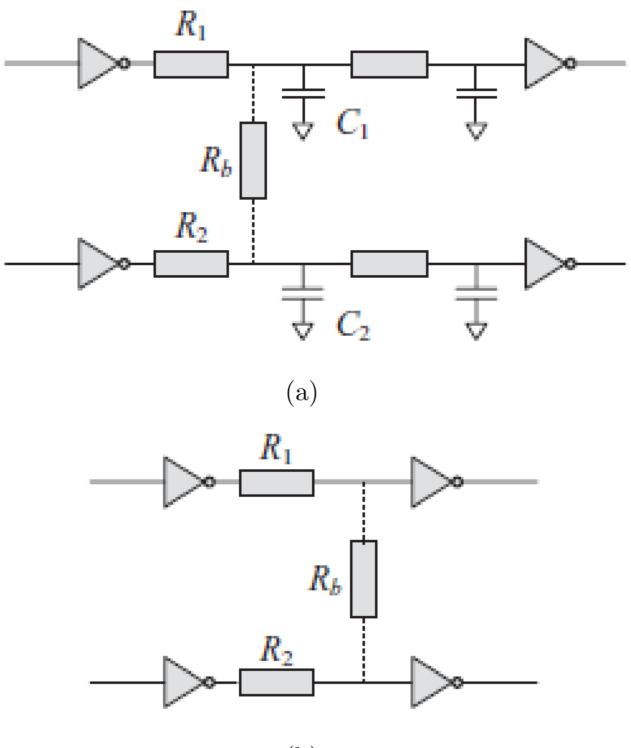


Figure 2.5: The bridged circuit model [Gao et al. 2005].

carry different logic values, and multiple faults may be injected into the circuit at one time.

2.1.4 SEU on Unidirectional Routing

For inter-CLB routing, modern FPGAs have shifted from bidirectional routing towards unidirectional routing architecture. In this new routing architecture, connection boxes and switch boxes employ directional wires to route signals and use MUXes for signal interconnection. As a result, the fault behavior in this unidirectional routing is different from that of bidirectional pass transistors. In this work, we consider the strict use of single-driver directional routing [LLT04].

Once affected by the SEU, one of the encoded CRAM bits for routing MUXes flips its value and thus select an erroneous input pin onto the MUX output, similar to the affected local routing seen in Figure 2.6. The erroneous signal may be further propagated to primary outputs to be finally observed.

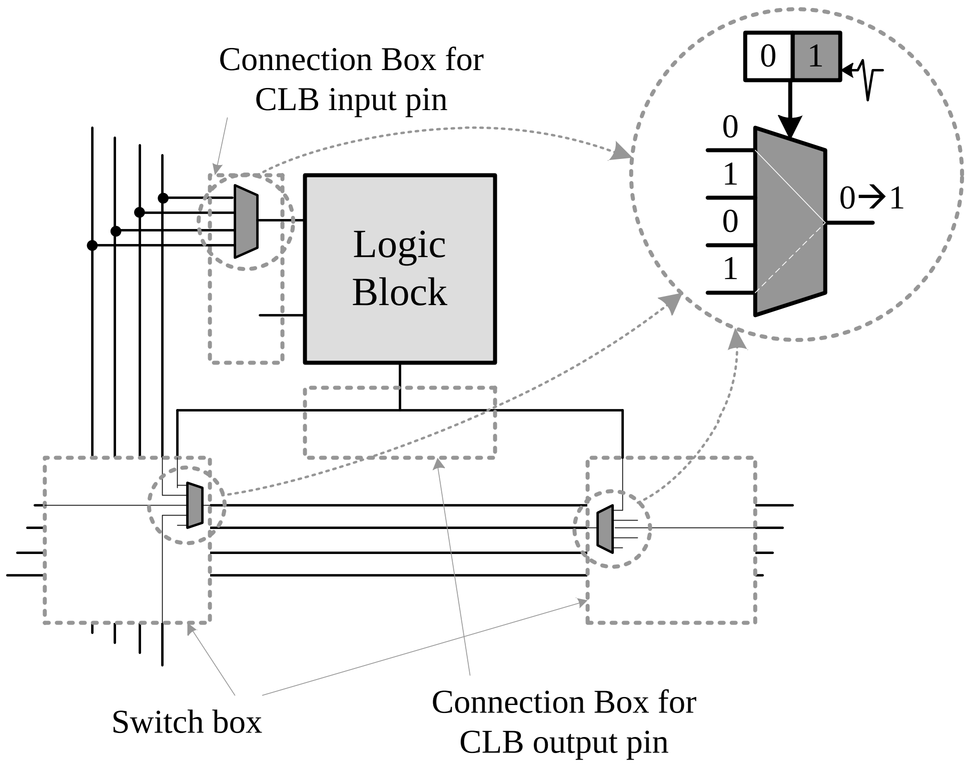

Connection Box for CLB input pin
0
1
0
1
0
1
0→1
Logic Block
Switch box
Connection Box for CLB output pin

Figure 2.6: An SEU on a connection box and switch box in a unidirectional routing architecture.

The unidirectional routing architecture is mainly made up of MUXes, which raises the signal selection fault instead of the open fault or short (bridging) fault in the conventional bidirectional routing when an SEU occurs.

2.2 SEU Evaluation Framework

We now present our SEU fault evaluation framework. Our framework performs the fault analysis on each CRAM bit under the single fault assumption, that is, at any time, at most one SEU exists in the FPGA. This is reasonable, because compared to SEUs, simultaneous multiple-bit SEUs (MBU) have less chances to happen in current FPGAs [Cha09].

2.2.1 Soft Error Sensitivity Evaluation

Previous work [HFH08, FHH09] have studied the sensitivity of functional failure to SEU fault on LUT CRAM bits by introducing the metric of criticality for LUTs, or called Soft Error Rate (SER). In this work, we leverage the idea and extend it to interconnects, and present a unified metric of SER definition as follows.

For a circuit C with n primary inputs, the soft error rate SER_b of one CRAM bit b, which configures an FPGA resource like an LUT or a routing resource, is the probability that an error can be observed at the primary outputs due to the SEU on b.

$$SER_b(x) = \frac{1}{2^n} |\{x | C_b(x) \neq C_{\bar{b}}(x)\} \tag{2.1}$$

where $x \in (0, 1)^n$ is one of the vectors in the exhaustive input set X. $C_b(x)$ is the circuit output without SEU fault under x, and $C_{\bar{b}}(x)$ is the circuit output when bit b is flipped. When $C_{\bar{b}}(x)$ and $C_b(x)$ mismatches, the system is said to encounter failures which should be attributed to bit b. Therefore, by identifying detectable errors under input set X on the circuit, the metric of SER reveals the possibility of an SEU on a CRAM bit that results in FPGA failures, which is generally acknowledged as the failure rate by an SEU.

In general, the SER of bit b can be obtained by exhausting all the 2^n permissible vectors in a complete input set X, which is extremely time consuming. In practice, it can be approximated by Monte Carlo-based simulation , which can provide good accuracy as implied in [LLH10]. In addition, it applies to any circuit element as long as it has CRAM bits in it as in Eq. 2.2. For example, we can quantify the chip failure rate from each CRAM bit in the circuit by aggregating the SER of one bit and the probability of SEU that happens on that bit.

$$\sum_{b \in elem} SER_b \cdot Pr(b \xrightarrow{SEU} \bar{b}) \tag{2.2}$$

where *elem* is the set of CRAM bits used by the design implemented on the FPGA.

2.2.2 Framework Overview

Figure 2.7 illustrates the flow of our proposed SEU fault evaluation framework for SRAM-based FPGAs. Given a circuit netlist, it first applies logic optimization and technology mapping onto the LUTs. The mapped circuit is packed into logic blocks then placed and routed by physical design tools. The SEU analysis starts right after the placement and routing, taking the post-layout circuit, FPGA architectural file, and circuit logical function as inputs. Based on the metric of SER, our framework evaluates the failure rate or sensitivity of each CRAM bit in various elements according to their fault behavior, as described in Section 2, for example, the open and short faults in bidirectional routing or the selection faults in unidirectional routing. After the SEU analysis, SEU-induced soft errors are injected into the simulator, which then performs logic-level simulation on the faulty circuit and calculates the SER for each bit automatically.

As an important evaluation step towards robust FPGA design, our framework can identify the most failure-sensitive circuit element and evaluate the applicability of various fault mitigation schemes. This evaluation can be applied as early as possible to be helpful during design time. Based on physical layout information, our framework is able to reveal the failure sensitivity for each CRAM bit in the FPGA circuit that is vulnerable to SEUs. In addition, the framework is universal and flexible to different FPGA architectures by adding new micro-architectural descriptions of the circuit element concerned. As a result, we envision that by shedding light on the hidden relation between CRAM bits and FPGA functional failures, our proposed framework will be helpful in designing more robust FPGA

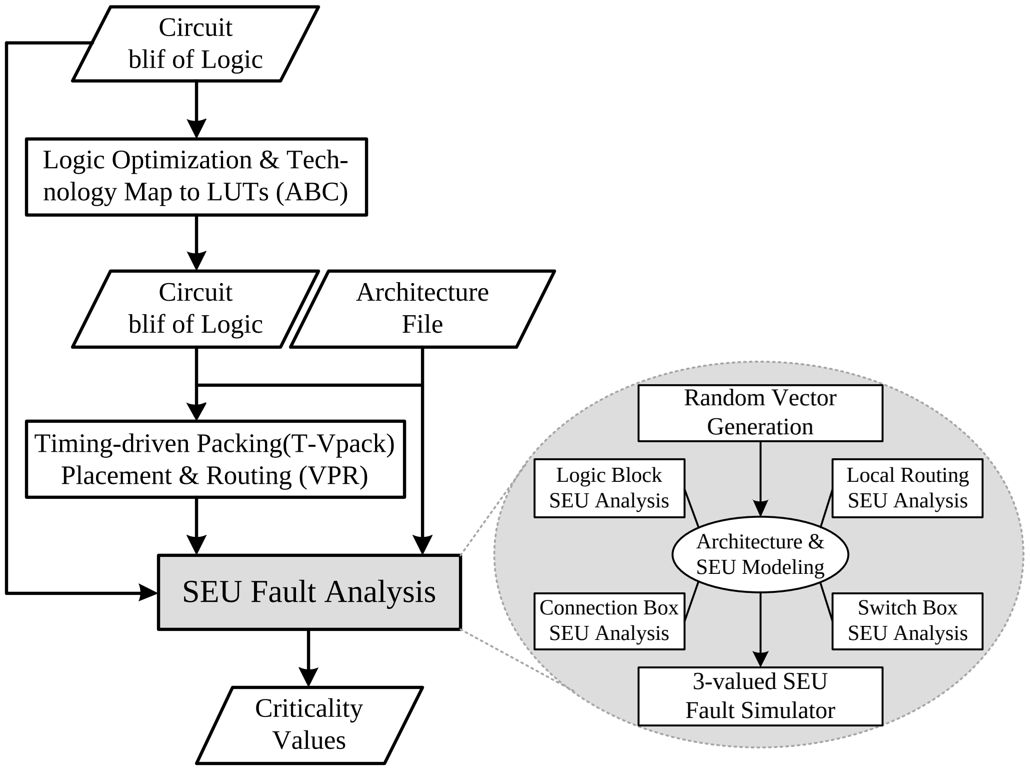


Figure 2.7: Overview of our SEU evaluation framework.

circuits, architectures, and synthesis algorithms.

2.3 SEU Characteristics on Architectures

In the experiments, the 10 largest MCNC combinational circuits are used as our test benchmark circuits with their statistics shown in Table 2.1. For the benchmark circuits, we first apply logic optimization and technology mapping to 4- and 6-input LUTs (LUT size of k) using the Berkeley ABC tool [ABC] to represent the most popular used LUT input sizes in practice. The mapped circuits are packed by different logic block sizes (cluster size N) of 4, 6, 8, and 12 by the T-VPack tool. As a result, their combinations cover eight different architectural settings representing different cases, like smaller LUT with larger cluster, or larger LUT with smaller cluster, and some other common settings. Then, the circuit under each case is placed and routed by the VPR tool [LKJ11] for minimum dimension and routing channel width, such that it generates the FPGA array as compactly as possible without involving extra unused bits that exceed the actual need of the circuit.

Circuit	# Gates	# Inputs	# Outputs
ex5p	527	8	63
apex4	722	9	19
misex3	735	14	14
alu4	730	14	8
ex1010	850	10	10
apex2	942	39	3
seq	1020	41	35
des	1498	256	245
spla	2237	16	46
pdc	2326	16	40

Table 2.1: Size Statistics for the 10 MCNC Benchmark Circuits

In this experiment, we characterize the SEU induced soft errors with respect to different circuits and CLB architectures. The hardware model and detailed SEU fault behavior on them have been discussed in Section 2. For the SER calculation, we performed Monte Carlo simulation with 10K vectors, which consumes an acceptable runtime and provided relatively accurate estimations of the SER values, according to the study in [LLH10]. The CRAM bits in all logic and routing resources are evaluated based on their physical information obtained after placement and routing, which contribute to the majority of the configuration bits in an FPGA. There are other CRAM bits with much smaller numbers, but may configure clocks, resets, or other modules for control. It will be our future work to model their diversified behavior when affected by an SEU. We suppose a uniform distribution of the probabilities for each bit to go faulty, that is, $Pr(b \xrightarrow{SEU} \bar{b})$ values are identical in Equation 2.2 for all the CRAM bits. In this way, we can simplify our SEU evaluation by focusing on their sensitivity to failure. In this section, we will first report the SEU characterization from different perspectives.

Note that the experimental results are based on the averaged data of the ten benchmark circuits if no circuit name is explicitly specified.

2.3.1 SER under Different CLB Architectures

Under bidirectional routing with different CLB architectures, Figure 2.8(a) shows the proportion of CRAM bit numbers in different circuit elements, and Figure 4.14(b) shows the proportion of their SER values. As a brief overview of the two plots, several observations can be made. (1) The routing resources hold the majority of the CRAM bits, from 61% to 87% approximately, while contributing even more in total SER, over 90% in these cases. This means that functional failure is most likely due to routing rather than LUT. (2) In terms of SER, there is no single circuit element dominating the overall SER. However, local routing MUXes have a larger proportion, while the proportion of their CRAM bits is relatively small compared to other elements. This indicates that they are the most failure-sensitive elements in an FPGA. (3) Increasing LUT size k and cluster size N increases LUT bits, but their SER proportion is nearly the same, which is less than 10% for all the cases. (4) With the increasing LUT size k and cluster size N, local routing MUXes contribute more in the total SER, because both k and N enlarge the number and the size of local routing MUXes and shrink the global routing network at the same time.

We further show Figure 2.9(a) for a detailed view of the SER values from different circuit elements. The x-axis lists all the circuits under test in different architectural settings, and the y-axis gives the summed SER value for each case. From the figure, one can see that each circuit presents significantly different failure sensitivities due to their inherent logic. Moreover, failure sensitivities of the same circuit under different settings may also vary. It is interesting to note that a larger LUT input size k provides a notable reduction of the total SER value, because a larger LUT input size k shrinks the network dimension, which helps to reduce the routing CRAM bits that are more vulnerable to failures. In contrast, cluster size N balances the impacts of switch boxes versus connection boxes and local routing MUXes, while the total SER values of cases with medium cluster size N are generally lower than other cases of the same LUT size.

We also report detailed SER values from different elements under the unidirectional routing in Figure 2.9(b). One can see that it presents similar patterns to those of bidirectional routing. A most significant difference is that in unidirectional routing, the switch boxes hold the largest number of CRAM bits among the three routing elements and dominate the overall SER as well. The reason is due to the micro-architecture in unidirectional routing, where CLB outputs go directly into switch box MUXes nearby and only CLB inputs are multiplexed in connection boxes.

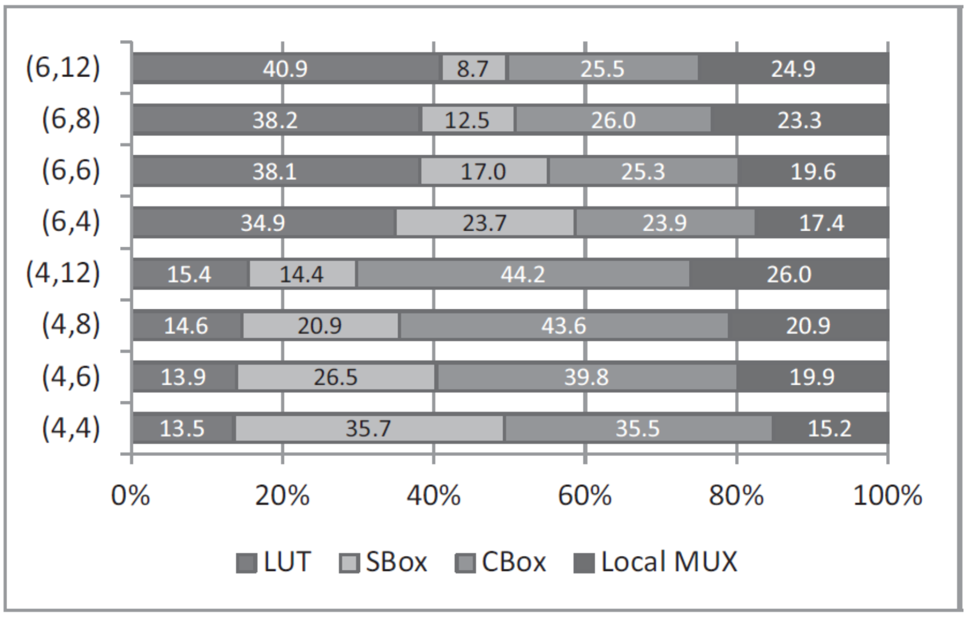
Further, Figure 2.10 reports the SEU evaluation time for the unidirectional routing architecture to provide a sketch of the efficiency of our evaluation framework. The experiments are performed on a personal desktop with an Intel i3-CPU with 3G RAM. One can see that a smaller LUT input size k generally results in a longer evaluation time, because smaller LUT sizes require more interconnect and involve more CRAM bits on routing for evaluation. Note that in our evaluation framework, all the CRAM bits will be evaluated, each with 10K input vectors. Therefore it will be time consuming when a circuit has millions of CRAM bits. To accelerate the evaluation, several techniques can be applied on our framework in the future, such as packing the bit-wise input signals for parallel simulation and packing circuit nodes into larger blocks to avoid unnecessary node traversal when a block is fault free. In addition, due to the inherent parallelism in fault simulation, the framework can be easily deployed across different machines to gain further boost-up.

2.3.2 SER Breakdown in Bidirectional Routing

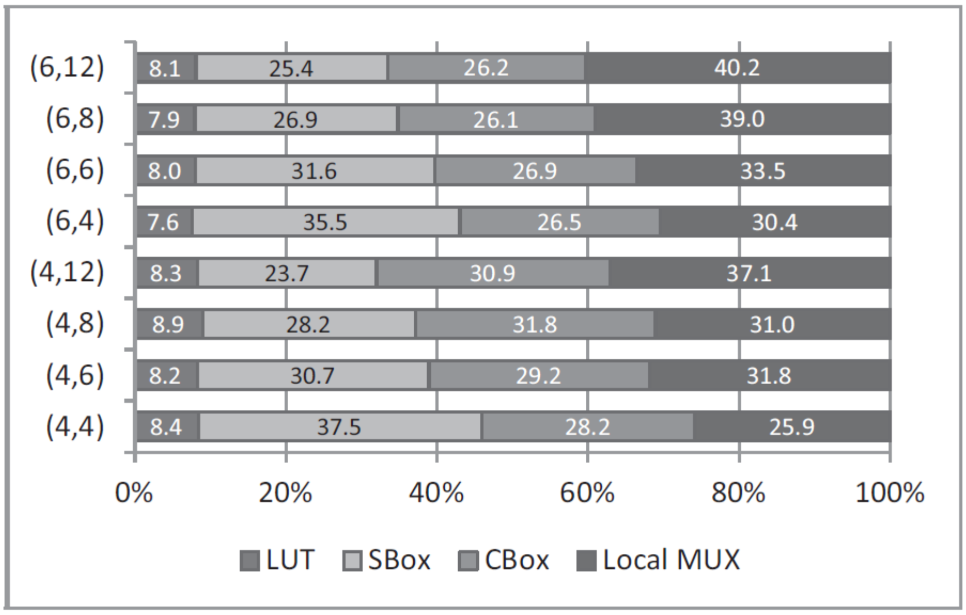
As discussed in Section 2.1, SEUs on the routing CRAM bits in bidirectional routing induce both open and short faults. Figure 2.11 shows the SER breakdown. An SEU-induced short fault is more sensitive to functional failure than that of on open fault, almost $1.3x$ in switch boxes and $4.5x$ in connection boxes on average, in terms of their accumulated SER values. This is because most of the time, switch boxes have utilization rates lower than 30%, and the rates of connection boxes are even lower. Typically, a lower utilization rate in a switch box or a connection box provides more possibility for a short fault. At the same time, the sum of short and open SER values in switch boxes is notably reduced when the LUT input size k and CLB size N increase, because the sensitive CRAM bits in switch boxes rely completely on the dimension of the global routing network, which shrinks with an increasing LUT input size k and CLB size N.

2.4 Summary

A comprehensive SEU fault evaluation framework for SRAM-based FPGAs has been proposed in this chapter. Based on the post-layout FPGA application, the proposed framework is capable of quantifying the SEU fault-induced functional failures for exact configuration bits in various circuit elements, such as LUTs, connection boxes, switch boxes, and local routing multiplexers. The SEU induced soft errors were characterized by several existing FPGA architectures differentiated by CLB sizes, LUT sizes, and routing structures. Detailed soft error characteristics from various perspectives can be found in our experiments. The SEU fault evaluation framework provides detailed information for identifying the most critical configuration bits or circuit elements to develop new fault mitigation algorithms, which will help on developing robust FPGA architectures and SEU mitigation techniques.

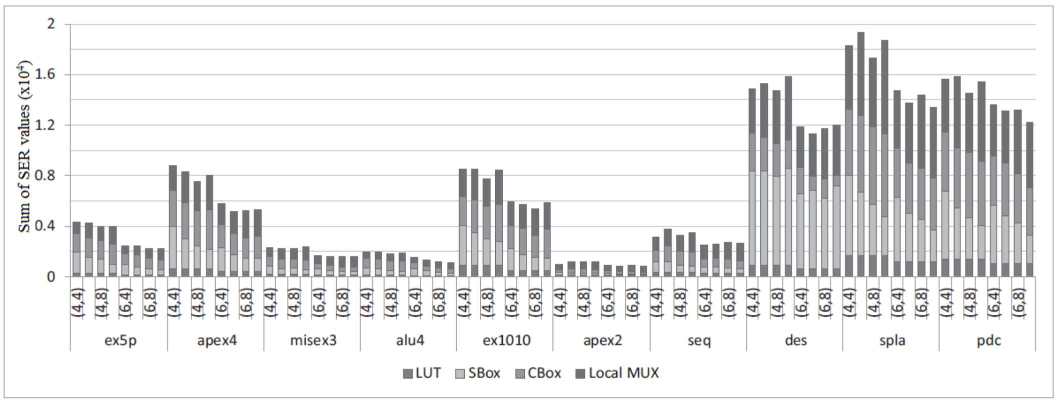


(a)

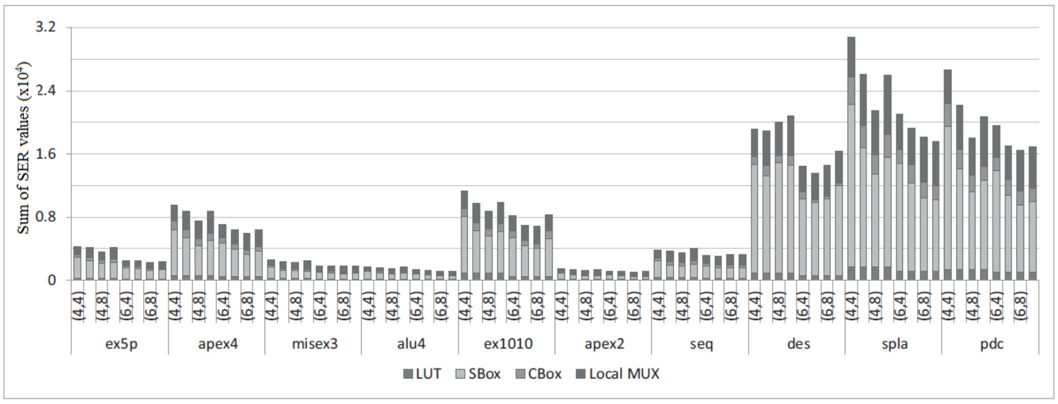


(b)

Figure 2.8: In bidirectional routing, (a) CRAM bits and (b) total SER proportions from different circuit elements under different CLB architectural settings.



(a) Bidirectional routing



(b) Unidirectional routing

Figure 2.9: Detailed SER values for all the circuits under (a) bidirectional routing (b) unidirectional routing.

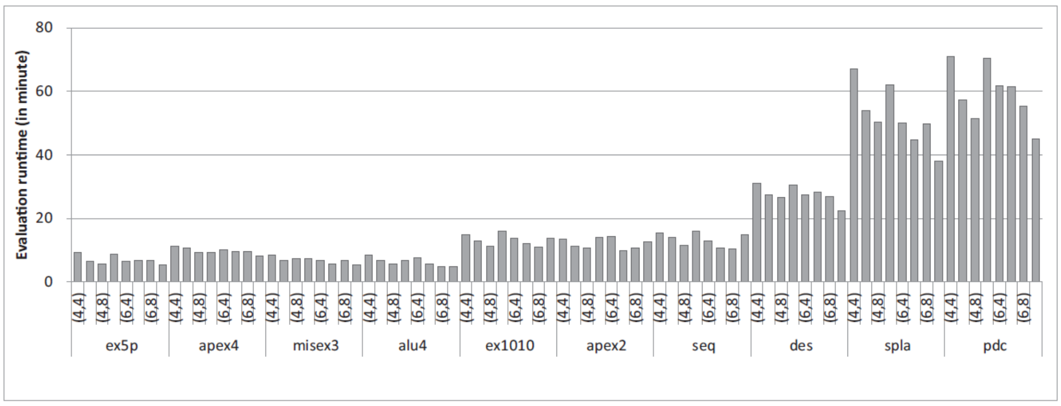


Figure 2.10: Runtime for the SEU evaluation by our framework.

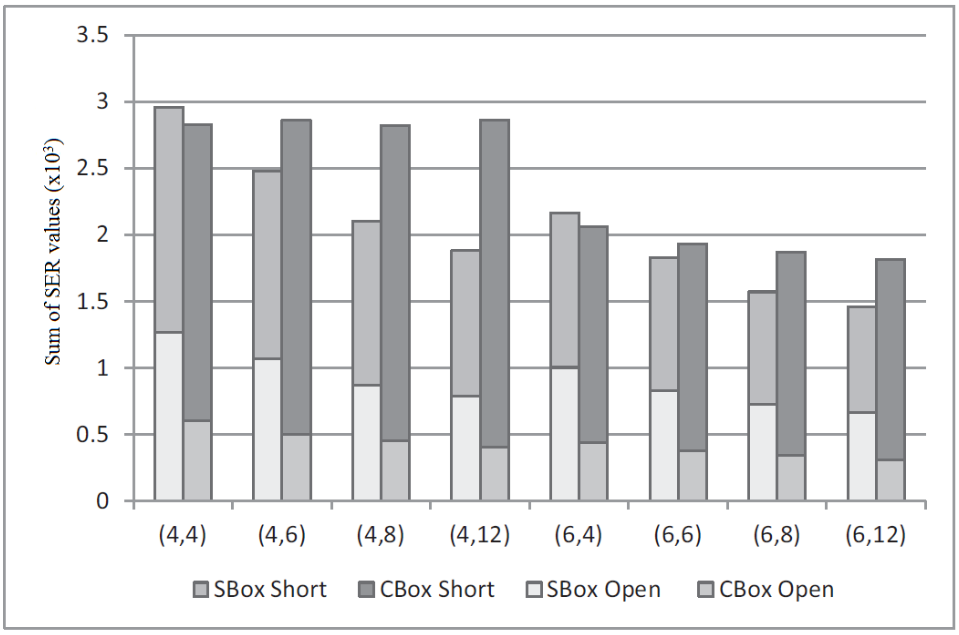


Figure 2.11: SEU-induced open and short fault breakdown in switch boxes and connection boxes in bidirectional routing.

CHAPTER 3

Heterogeneous Configuration Memory Scrubbing for Soft Error Mitigation in FPGAs

State-of-the-art SRAM-based FPGAs utilize a huge amount of configuration memory elements to implement logic and interconnect configurability for complex circuit functionality [Xilc]. Reliability of CRAM has become an issue of increasing importance for SRAM-based FPGAs over the last decade due to soft errors caused by Single Event Upsets (SEU) [Muk11]. Advances in manufacturing process technology and the extensive use of SRAM have resulted in a higher susceptibility to SEUs.

Although radiation hardened techniques for FPGAs to prevent CRAM from upset have been proposed [RPD07, McC09, acta, Xila, SAS10a], they result in lowered logic density and increased area. To improve FPGA CRAM reliability against SEUs, it is common to employ CRAM soft error detection or correction techniques such as Cyclic Redundancy Check (CRC) [SB04], Error Correction Code (ECC) [LSR12], or scrubbing [BPP08] for CRAM. For CRAM scrubbing techniques, [BPP08] evaluates two different memory scrubbing schemes including internal CRAM scrubbing solution using CRC and ECC [Cha09] and an external scrubbing solution designed by NASA/GSFC REAG. Without frame by frame readback and ECC checking, REAG scrubber has improved performance over the solution developed by Xilinx. [HSW09] provides a way to perform partial reconfiguration with memory scrubbing and improves the configuration time for different designs. However, none of the techniques have demonstrated quantitatively how effective or how much improvement they can make with respect to MTTF.

In addition, SEUs have varying impacts and sensitivities toward circuit behaviors. Rather than mitigating all the logic errors in the circuit, a difficult challenge is to prevent critical failures in the system. While some SEUs may cause system failures, others may not have a significant impact and are considered system don't cares. For example, an SEU on circuit control path is likely to cause unrecoverable system failures. In contrast, a single error on the circuit data outputs such as video or voice streaming rarely affects the system operation and only lowers the quality of the streaming content, which is usually insignificant. However, all of the existing CRAM scrubbing methods are homogeneous. That is, all of the CRAM bits are considered equally critical, which limits the improvement on system MTTF. Therefore, tuning fault tolerant techniques to consider critical system failure and system don't care becomes particularly important for FPGA applications.

In this paper, we first demonstrate for the first time how much CRAM scrubbing can quantitatively improve the system MTTF measured by the stochastic system vulnerability factor of CRAM bits. Then, we propose a Heterogeneous CRAM Scrubbing (HCS) technique, to make an intelligent use of application information to mitigate system failures to the greatest extent. To fully leverage the HCS technique, we develop a dynamic programming approach that solves the HCS problem efficiently. Our experimental results first show a module level study using the 10 largest combinational MCNC benchmark circuits that the optimized HCS technique achieves average system MTTF improvement of approximately 20%. Then, a system level study on a H.264/AVC decoder implemented on a Xilinx Virtex-5 FPGA shows an estimation of roughly 60% system MTTF improvement compared to the existing homogeneous CRAM scrubbing method. Note that the proposed HCS technique can be applied to most of the CRAM scrubbing solutions, such as the external CRAM scrubbing method in [BPP08], by replacing the golden bitstream with the HCS result. Because modern FPGAs feature dynamic and partial reconfiguration, HCS does not require a stall and can be performed without interrupting the circuit operation. In addition, HCS virtually does not lead to any overhead on performance, power, and cost.

The rest of this chapter is organized as follows. Section 3.1 introduces the stochastic modeling of SEU induced soft errors and its evaluation metrics. Section 3.2 presents the idea of how CRAM scrubbing improves circuit system robustness and the problem formulation of the proposed HCS technique. Section 3.3 presents the dynamic programming approach to solve the HCS problem. Section 3.4 shows the experimental results on MCNC benchmark for module level study and a H.264/AVC decoder for system level study. Section 3.5 summarizes this chapter.

3.1 Preliminaries

3.1.1 System Robustness Evaluation

Mean Time to Failure (MTTF) and Failures in Time (FIT) are two most commonly used design metrics to quantitatively evaluate the robustness of circuit systems. FIT is defined by the number of failures occurred in one billion hours while the circuit system is operating. On the other hand, MTTF is reversely proportional to the summation of the FIT values for every component in the system according to [MER05], which can be calculated by:

$$MTTF(in\,hours) = 10^9/FIT_{total} \tag{3.1}$$

The FIT value of an FPGA can be further decomposed into two factors. First is the probability that an SEU occurs on a CRAM bit, which is the bit upset rate measured by upsets per billion hours and denoted by $R_{intrinsic\_error}$ that depends on operating environment. The other factor is the Architecture Vulnerability

Factor (AVF) which is the probability of a fault leading to a system failure, i.e. a visible error at the circuit primary outputs. As a result, the FIT value of an FPGA circuit can be given by

$$FIT_{total} = \sum AVF \cdot R_{intrinsic\_error} \tag{3.2}$$

To quantitatively measure the sensitivity of a CRAM bit to a system failure, we define System Failure Rate (SFR). Under the single fault assumption, where at most one fault exists in the circuit at a time, the SFR of a CRAM bit evaluates the probability of the SEU on the bit that leads to a system failure,

$$SFR_b = Pr(C_b(x) \neq C_{\bar{b}}(x) | b \xrightarrow{SER} \bar{b}) \tag{3.3}$$

where $x \in (0,1)^n$ is the exhaustive set of input vectors and $C_b(x)$ is the circuit status without SEU on b under x, and $C_{\bar{b}}(x)$ is the circuit operation status when bit b is flipped due to an SEU. In general, the circuit operation status is usually defined by the register values of the main finite state machine or the circuit outputs that significantly affect the system operation. In general, SFR_b can be obtained by an exhaustive fault simulation on 2^n input vectors. For practical use, we can use Monte Carlo fault simulation to estimate SFR values efficiently.

While the metric of SFR_b is used in this chapter as our evaluating measurement to estimate the probability of system failures due to an SEU on a specific CRAM bit, we will show in Section 3.3 that it is the key factor and can be converted to AVF, FIT and MTTF on system failures when CRAM scrubbing is applied.

3.1.2 SEU Manifesting

It is a well known fact that not every SEU impacts the circuit functionality. Specifically, whether an SEU can induce system failures is determined by two factors: (a) the spatial factor, i.e. the fault location when the SEU occurs and (b) the temporal aspect, i.e. the circuit status when in operation.

Practical applications typically do not fully utilize all of the CRAM bits in an FPGA. In addition, only a portion of the CRAM bits occupied by the application affect the circuit operation, which are generally defined as critical bits of the design. On the contrary, most CRAM bits have no effect on circuit functionality, and are considered as don't-care bits of the design. In other words, a circuit function is changed only if an SEU occurs at a critical bit. Further discussions regarding critical bits and don't-care bits will be presented in Chapter 4.

In terms of the temporal aspect, an SEU requires time to manifest itself at the output and seldom raises a failure immediately. For example, this may happen if an SEU occurs at some component that is intermittently idling during circuit operation. Consequently, the SEU does not cause failures until the component is active again and the fault is propagated to circuit outputs. For many cases, a circuit requires a certain amount of time to response to an SEU, and another period of time for the SEU induced errors to propagate to the circuit primary outputs or FFs. The total amount of time required above is defined as SEU Time to Manifest (TTM):

$$TTM = t_2 - t_1 \tag{3.4}$$

where t_1 is the time when an SEU occurs in the circuit and t_2 is the time when the circuit fails.

3.2 Problem Formulation

The spatial and temporal factors indicate that an SEU requires a particular location and time to induce a failure. This suggests that if the SEU on a critical bit can be corrected before it induces an error, the system can stay healthy and failures can be avoided. In this section, we consider memory scrubbing, and propose a heterogeneous CRAM scrubbing technique and its problem formulation.

We first define time to scrub of a CRAM bit as the scrubbing interval N of a CRAM bit (i.e. the number of clock cycles to rewrite the bit). For a conventional homogeneous memory scrubbing, which sequentially rewrites every CRAM bit of the entire CRAM, the time to scrub each bit is equivalent to the time for the scrubbing unit to rewrite the entire CRAM. Therefore, the scrubbing interval is the total number of CRAM bits to be scrubbed divided by the number of CRAM bits that can be scrubbed in one clock cycle. To improve the scrubbing interval, only those CRAM bits used by the design are required to be scrubbed. When a circuit size increases by occupying more critical bits, the time required between scrubbings of a specific bit becomes longer, which degrades the AVF. A straightforward solution to improve the AVF is to increase the speed of the CRAM scrubbing. However, scrubbing speed is constrained by FPGA configuration interface.

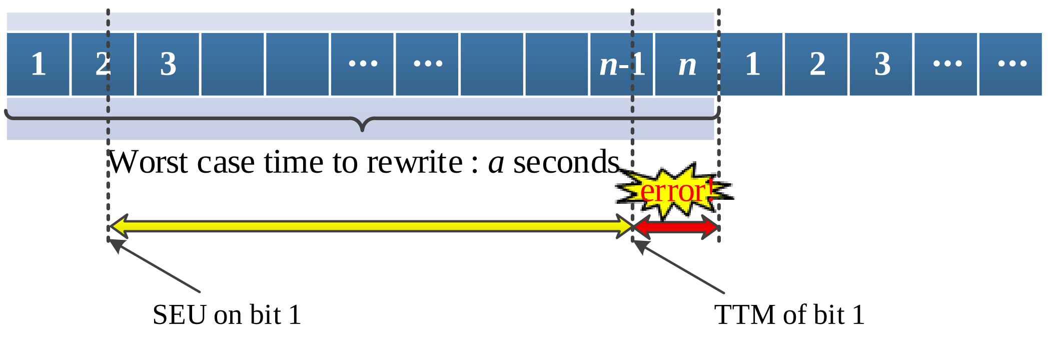
Based on the CRAM bit soft error analysis in Chpater 2 and other works such as [LLH10] and [CM10], the vulnerability varies dramatically from node to node in the circuit, which highly depends on the circuit structure and application functionality. Intuitively, it helps to scrub critical components with higher vulnerability more aggressively. In other words, more vulnerable bits should be scrubbed with a higher rate, and those robust bits should be scrubbed less frequently. Therefore, memory scrubbing intervals for different CRAM bits could be adjusted according to their vulnerability for higher AVF reduction. We define the memory scrubbing schedule as a sequence of CRAM bits to be scrubbed, and the sequence is applied periodically to refresh the entire CRAM. Note that each CRAM bit is appeared at least once in the sequence to prevent the accumulation of SEUs.

Figure 3.1 illustrates a simple example of how rescheduling the memory scrubbing sequences can prevent circuit failure. Suppose a circuit with n critical CRAM bits with a homogeneous scrubbing unit, which refreshes these n bits sequentially as shown in Figure 3.1(a). Suppose bit 1 encounters an SEU right after being scrubbed, and can raise system failures after a fixed interval of cycles, i.e. the TTM of bit 1, independent of the scrubbing sequence. Then, in scenario (a), the system fails after the TTM of bit 1 because the time to scrub bit 1 is longer than the TTM of bit 1 and cannot prevent the system failure. However, if we reschedule the scrubbing unit for bit 1 as in scenario (b), which rewrites bit 1 twice in the schedule, the SEU on it can be cleared before it leads to failures. As a result, finding a new scrubbing schedule helps to improve the AVF, and this problem can be formulated as the following.

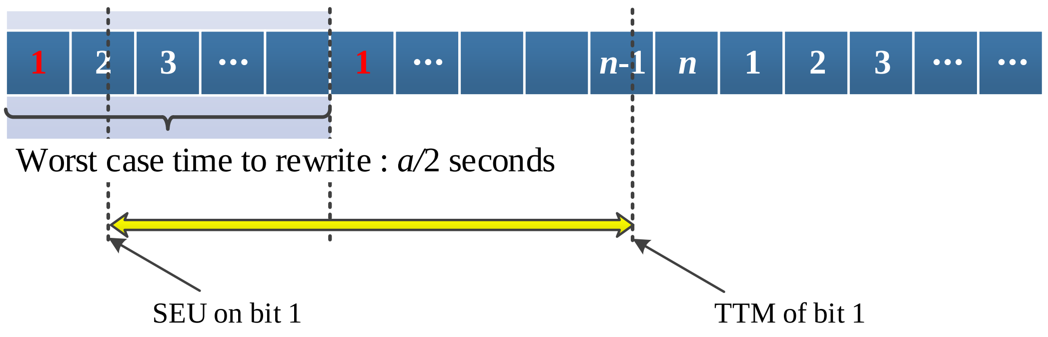
Formulation 1. *Given a circuit C placed and routed onto an FPGA, the SFR values for each CRAM bit in C, the operating frequency F_c of C, and a memory scrubbing unit operating at F_m which refreshes W CRAM bits in each clock cycle, schedule the memory scrubbing units to refresh each CRAM bit (or a group of CRAM bits in sequence) in C in terms of its ordering and rate such that the AVF of C can be minimized.*

3.3 Heterogeneous Configuration Memory Scrubbing Algorithm

In this section, we present the reliability improvement with CRAM scrubbing based on the stochastic fault modeling presented in Section 3.1. Then, we pro-



(a) Homogeneous CRAM scrubbing



(b) Heterogeneous CRAM scrubbing

Figure 3.1: Examples of homogeneous and heterogeneous CRAM scrubbing

pose a dynamic programming based algorithm that can solve the heterogeneous memory scrubbing scheduling problem optimally with a given length of the schedule.

3.3.1 AVF Update with CRAM Scrubbing

Many scrubbing techniques and error correction methods by ECC code have been proposed. However, no quantitative metric has been presented to show how much improvement it can provide. In the following, we present a stochastic method to estimate the improvement of CRAM scrubbing.

According to definition of Eq. 3.3, SFR is the probability that an error is observed at the circuit primary outputs by a random input vector. For each clock cycle, we apply different random input vectors to the primary inputs. Assuming that an SEU occurs at a CRAM bit b, and the scrubbing interval of the bit is N clock cycles, the probability that the circuit does not produce any error after N clock cycles is

$$AVF_b = 1 - (1 - SFR_b)^N \tag{3.5}$$

Note that N depends on the status of the scrubbing, i.e. at which bit the CRAM scrubbing unit is rewriting. In this work, we assume a conservative and worst case condition that an SEU always occurs right after the CRAM bit has just been rewritten. In other words, N is the worst case time to scrub. Therefore, for the existing homogeneous CRAM scrubbing technique, N is the time to scrub the entire CRAM or the entire design.

3.3.2 Dynamic Programming Algorithm for Heterogeneous Memory Scrubbing Scheduling

According to Eq. 3.5, reducing the time to scrub N of one CRAM bit can reduce its AVF. However, scrubbing a CRAM bit more frequently generally result in scrubbing other CRAM bits less frequently under the assumption that only one CRAM bit can be scrubbed at a time. Hence, a careful schedule of CRAM scrubbing process becomes essential for AVF improvement of the circuit. In this section, we propose a dynamic programming based algorithm that assigns the memory scrubbing rates (how frequently the CRAM bits are scrubbed) heterogeneously for CRAM bits based on the AVF update in the previous section to minimize the average AVF of the circuit.

In fact, the number of CRAM bits that can be written into CRAM at a time is determined by the CRAM and the FPGA architectures. Therefore, we define the CRAM granularity as the following.

Definition 1. *We define the minimal number of CRAM bits that can be accessed at a time as CRAM granularity. As a result, the atomic operation of the memory scrubbing unit is to rewrite a CRAM bit or a group of CRAM bit according to the CRAM granularity.*

In order to obtain a feasible solution and solve the problem efficiently in the proposed algorithm, we fix the length of the memory scrubbing schedule, which is defined as L as follows.

L: the number of atomic operations in the scrubbing sequence for given CRAM granularity.

Then, the HCS algorithm assigns scrubbing rates for CRAM bits, i.e. how many times a CRAM bit or a group of CRAM bits are scrubbed in the given length of the schedule. Furthermore, we define

SR: the number of CRAM bits scrubbed in a clock cycle of circuit operation, which is calculated by the memory scrubbing throughput divided by the circuit operating frequency. Note that the circuit application on the FPGA and the memory scrubbing unit can operate under different clock domains, and SR is calculated under the circuit application clock domain.

S_b: how many times the CRAM bit or a group of CRAM bits b is rewritten in the memory scrubbing schedule, i.e., the scrubbing rate of bit b.

Given L, SR, and the SFR for each CRAM bit (group), we present the mathematical formulation of the memory scrubbing scheduling problem as follows.

$$\begin{array}{ccc} Minimize & average\_AVF = \sum_{i=1}^{n} AVF_i/n \\ Subject\,to & AVF_i = 1-(1-SFR_i)^{Q_i} \\ where & Q_i = \frac{L}{S_i \cdot SR} \end{array} \tag{3.6}$$

Note that AVF is additive and can be calculated independently for each bit. Therefore, the problem above can be recursively divided into subproblems and solved optimally. We propose a dynamic programming algorithm that minimizes the total AVF of a circuit given L, SR, and SFR_c, where SFR_c contains the CRAM bits (group) and their SFR in the circuit C. The CRAM scrubbing scheduling problem with fixed length of the schedule can be solved optimally by the proposed dynamic programming approach and the pseudo code is given below.

Algorithm 1 Pseudo-code of HCS algorithm

```
 1: set $best\_avf\_dp = \emptyset$;
 2: $optimal\_avf = avf\_dp(L, SFR_c)$;
 3: procedure AVF_DP($l, sfr$)
 4:     if $avf\_dp(l, sfr)$ has been solved and exists in $best\_avf\_dp$ then
 5:         return the result of $avf\_dp(l, sfr)$ in $best\_avf\_dp$
 6:     end if
 7:     if $|sfr| == 1$ then
 8:         return $avf(sfr[0], l)$
 9:     end if
10:     if $l = 0$ then
11:         return $\infty$
12:     else
13:         $min\_avf = \infty$
14:         for $S_i = 1; S_i \leq l; S_i = S_i + n$ do
15:             $current\_avf = avf(sfr[0], S_i) + avf\_dp(l - S_i, sfr\prime)$
16:             if $current\_avf < min\_avf$ then
17:                 $min\_avf = current\_avf$
18:             end if
19:         end for
20:         add $(avf\_dp(l, sfr)$, $min\_avf)$ to $best\_avf\_dp$
21:         return $min\_avf$
22:     end if
23: end procedure
```

In the pseudo code presented above, given L and SFR_c of a circuit C, where SFR_c is the set of all CRAM groups in C with their SFR values. We define $avf\_dp(L, SFR_c)$ as the function to solve CRAM scheduling problem. $optimal_avf$ stores the optimal AVF for C and $best\_avf\_dp$ is an array storing the optimal solutions for the subproblems of $avf\_dp(l, sfr_c)$, where $l \leq L$ and sfris a subset of SFR_c. $avf(sfr[b], S_i)$ calculates the AVF of a CRAM group b by its SFR value with the scrubbing rate S_i in the schedule. In line 4 and 5, if the subproblem $avf\_dp(l, sfr)$ has been solved, it returns the optimal result for $avf\_dp(l, sfr)$ stored in $best\_avf\_dp$. In line 7 and 8, if there is only one CRAM group that has not been assigned the scrubbing rate, it allocates the group with the maximum scrubbing rate with l and returns its AVF value. In line 10 and 11, if $L = 0$ and sfr contains at least one CRAM group, it returns ∞ because it cannot provide a valid solution. This means schedule sequence has been fully assigned, but each CRAM group is required to be scrubbed at least once in the scrubbing schedule. From line 13 to line 21, the function recursively and increasingly assigns a scrubbing rate for the first CRAM group in sfr, and the scrubbing length is reduced by the scrubbing rate assigned to the group. Note that n is step size for scrubbing rate assignment, and each time the scrubbing rate is increased by n. Then, the avf_dp function is called recursively to find the optimal solution for the subproblem of the rest of the CRAM groups, $sfr\prime$, with the remaining length of the schedule, $l - S_i$. Then, based on the optimal scrubbing rate assignments can be found, where the AVF is minimized.

3.4 Experimental Results

The proposed dynamic programming algorithm for HCS optimization is implemented in C++ on a Ubuntu server with Xeon 2.4GHZ CPU and 24GB memory. In this section, we evaluate the proposed HCS technique on two study cases: (a) a module level study using the 10 largest MCNC combinational benchmark circuits and (b) a system level study using an H.264/AVC video decoder implemented on a Xilinx Virtex-5 xc5vlx110t FPGA. For the module level, an error is recorded when a logic output is different from the right value and all logic outputs are treated equally. For the system level, an error is recorded considering so called system level dont cares. Specifically, errors for control modules (such as entropy coding and the main finite-state machine in the decoder) are recorded in the same way as that for the aforementioned module level study. On the other hand, errors for datapath modules are recorded only when the logic output errors are larger than the given thresholds. Clearly, the system level case study is similar to the practice and is therefore more relevant.

3.4.1 Module Level Case Study: MCNC benchmark circuits

The 10 largest combinational MCNC benchmark circuits are first optimized and mapped to 6-input LUTs using Berkeley ABC technology mapper [ABC]. The mapped circuits are packed using a cluster size of 8 using T-VPack tool, and placed and routed by the Versatile Place and Route (VPR) tool set [BR97], with a minimum dimension setting generating an FPGA as compact as possible. On the placed and routed circuit, we apply Monte Carlo simulation for SEUs on CRAM bits to calculate the SFR for each CRAM bits with an assumption that any error at the primary outputs induces a system failure.

To evaluate the effectiveness of the proposed dynamic programming approach, we first apply the traditional homogeneous memory scrubbing and calculate the average AVF value ($AVF_{typical}$) as the baseline for comparison purpose. Then, we apply the dynamic programming algorithm to generate an HCS solution and calculate the optimized AVF value ($AVF_{optimized}$). Due to the fact that MTTF is inversely proportional to AVF, we show the MTTF improvement ratio of HCS over the baseline by calculating $AVF_{typical}/AVF_{optimized}$. We assume in the experiment that $SR = 100$, where the memory scrubbing unit rewrites 100 CRAM bits when a circuit runs a clock cycle (a rate similar to that in Xilinx Virtex-5 assuming the circuit is running at 50Mhz), and $L = 10 \times |B|$, i.e. the length of HCS is 10 times the total number of CRAM bits used by the circuit. We set the step size of

Circuit	Circuit properties			MTTF improvement			
	# of CRAM bits	Dimension		CRAM Granularity			
		x,y	w	16	64	256	1024
alu4	103425	12,12	32	24.53%	29.70%	20.81%	12.21%
apex2	165888	14,14	26	25.64%	23.70%	21.41%	20.39%
apex4	149505	14,14	26	18.75%	12.35%	8.44%	4.86%
des	688128	15,15	36	10.16%	10.13%	6.61%	6.54%
ex1010	193536	17,17	32	17.50%	14.60%	8.63%	6.54%
ex5p	88064	17,17	32	18.75%	16.06%	13.80%	8.26%
misex3	97208	17,17	34	22.83%	21.86%	18.26%	13.64%
pdc	522240	42,42	18	18.53%	15.51%	6.83%	3.24%
seq	171008	24,24	48	24.44%	19.36%	14.78%	10.13%
spla	466944	25,25	46	20.54%	15.13%	6.68%	3.21%
Average	-	-	-	20.17%	17.84%	12.62%	8.90%

Table 3.1: MTTF IMPROVEMENT ON 10 LARGEST COMBINATIONAL MCNC BENCHMARK CIRCUIT

the scrubbing rate assignment to 5 (n=5) in the proposed dynamic programming approach to achieve a good tradeoff between the quality of the result and the runtime.

Table 3.1 summarizes the MTTF improvement of the HCS solutions under various CRAM granularities. Note that some circuits have the same dimension but use different numbers of CRAM bits because only CRAM bits actually used by the circuits should be counted. The results show that on average the HCS achieves roughly 20% improvement over homogeneous memory scrubbing with CRAM granularity of 16. A general trend shows that as CRAM granularity increases, the improvement by HCS decreases. This is because larger CRAM granularity typically result in lower design freedom for seeking a better HCS solution. For granularity of 1024, the proposed HCS can still provide an average improvement of approximately 9%.

We also list the runtime of the dynamic programming approach in Table 3.2. On average the proposed algorithm returns an optimal solution in approximately two hours for the smallest CRAM granularity in our experiment. The runtime

Circuit	Runtime (s)			
	CRAM granularity			
	16	64	256	1024
alu4	2659	455	94	13
apex2	5344	1052	244	45
apex4	4466	960	203	35
des	20053	2699	581	136
ex1010	5261	1622	344	65
ex5p	1829	324	65	8
misex3	2577	425	83	11
pdc	17506	3036	655	143
seq	5012	325	67	12
spla	12350	2439	522	116
Average	7705.7	1333.7	285.8	58.4

Table 3.2: HCS runtime

decreases dramatically with larger CRAM granularity because it also reduces the solution search space in the dynamic programming algorithm. When the CRAM granularity is 1024, the proposed technique can solve the circuits in a minute while providing a good improvement over the baseline.

We also investigate the impact of different HCS length, and the results are illustrated in Figure 3.2. We set the HCS length to a multiple of the number of CRAM bits used by the circuit and is swept by $L = L_{ratio} \times |B|$ along the x-axis in the figure. As one can see, the improvement by HCS begins to converge when the L_{ratio} is equal to 9. Note that larger L generally has better quality of the result but requires larger storage space for the implementation of HCS. For example, for CRAM scrubbing method using the golden bitstream, $L_{ratio} = 2$ requires twice as larger memory as the original one. On the other hand, scrubbing methods using ECC checking and correction technique [Cha09] can store the memory address to be scanned only, which greatly reduces size of storage and also minimizes the overhead caused by larger L.

Figure 3.3 depicts the MTTF improvement ratio for different SR values. The improvement increases when SR increases, which in general shows that the HCS

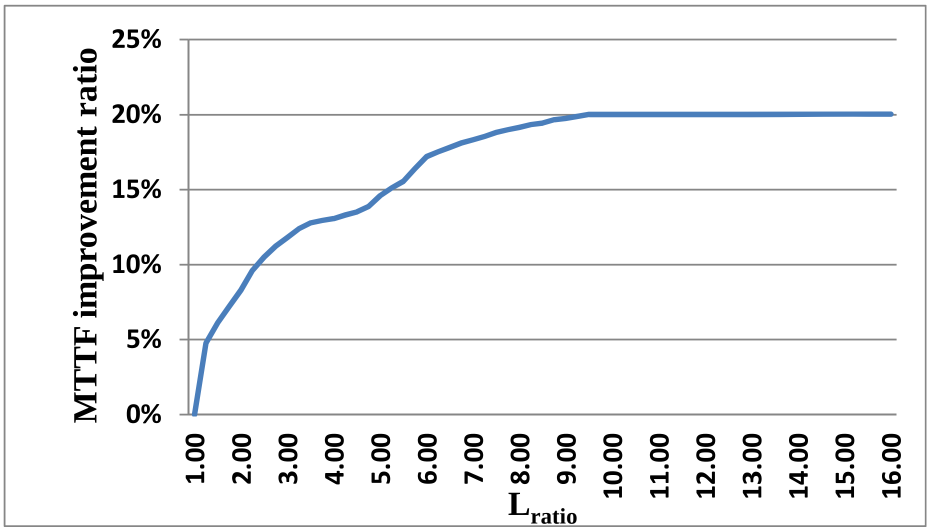


Figure 3.2: MTTF improvement for different length of HCS.

technique has better performance in terms of MTTF when the CRAM scrubbing unit runs at a higher throughput.

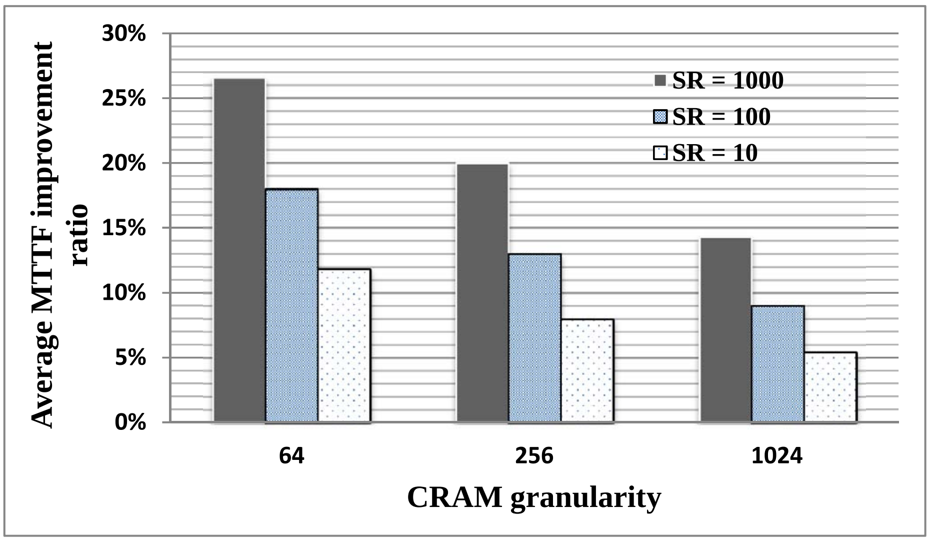


Figure 3.3: MTTF improvement ratio under different SR value.

From the experimental results we show that even a circuit module has high sensitivity to system failures on its CRAM, the proposed HCS technique and the dynamic programming algorithm can still provide a good HCS solution with reliability enhancement by 6% 26% over the existing homogeneous CRAM scrubbing.

3.4.2 System Level Case Study: H.264/AVC Decoder

For complex circuit systems and applications, not every circuit output error should be considered as a system failure. Instead, only errors occurred at the circuit critical outputs or registers cause system failures, which require a system reset to bring the system back to operation. For system level study, we use a H.264/AVC decoder, which is one of the most popular applications in FPGAs. H.264/AVC is a video coding standard which provides high compression efficiency via complex functionality and feature, such as entropy coding, transformation, filtering, and estimation. For example, an error in a single video frame can be considered a system don't care since the human vision system ignores such error during watching video [CO05]. However, a soft error in video decoder could degrade the quality in the successive video frame or even worse, lose the control of a system. For this reason, a good mitigation strategy is crucial to improve system MTTF.

Unlike the module level SFR estimation, SFR estimation at the system level is much more challenging due its complexity. To solve those issues, we provide an approach of bottom-up SFR estimation. We assume control path has higher SFR than data path. Especially in video coding, an error in video output may be an acceptable error while an error in FSM output is a critical fault. Meanwhile, some modules with lossy compression functionality have error-tolerance capability like filter and IDCT. So we assume their SFR is lower than control based module like main controller. Based on the assumptions, we define the critical registers in H.264/AVC decoder such as register in controller and entropy coding since the error in those registers will accumulate in time domain and make system enter an unknown state. Specifically, the proposed SFR estimation method can be divided into the following two steps: (1) divide the whole system into smaller modules; (2) transfer SFR in nodes into SFR in CRAM bits for the modules. The details of the two steps are as follows.

First, we divide the system into modules by functionality illustrated in Figure 3.4, and the modules are further divided into submodules. This strategy refers to actual placement and routing to define dimension for each module. We assume the CRAM bits in each module have high spatial locality. So the difference in SFR on CRAM bits for one module is relatively small, and we use the average SFR of the module as the SFR of the CRAM bits that belong to the module.

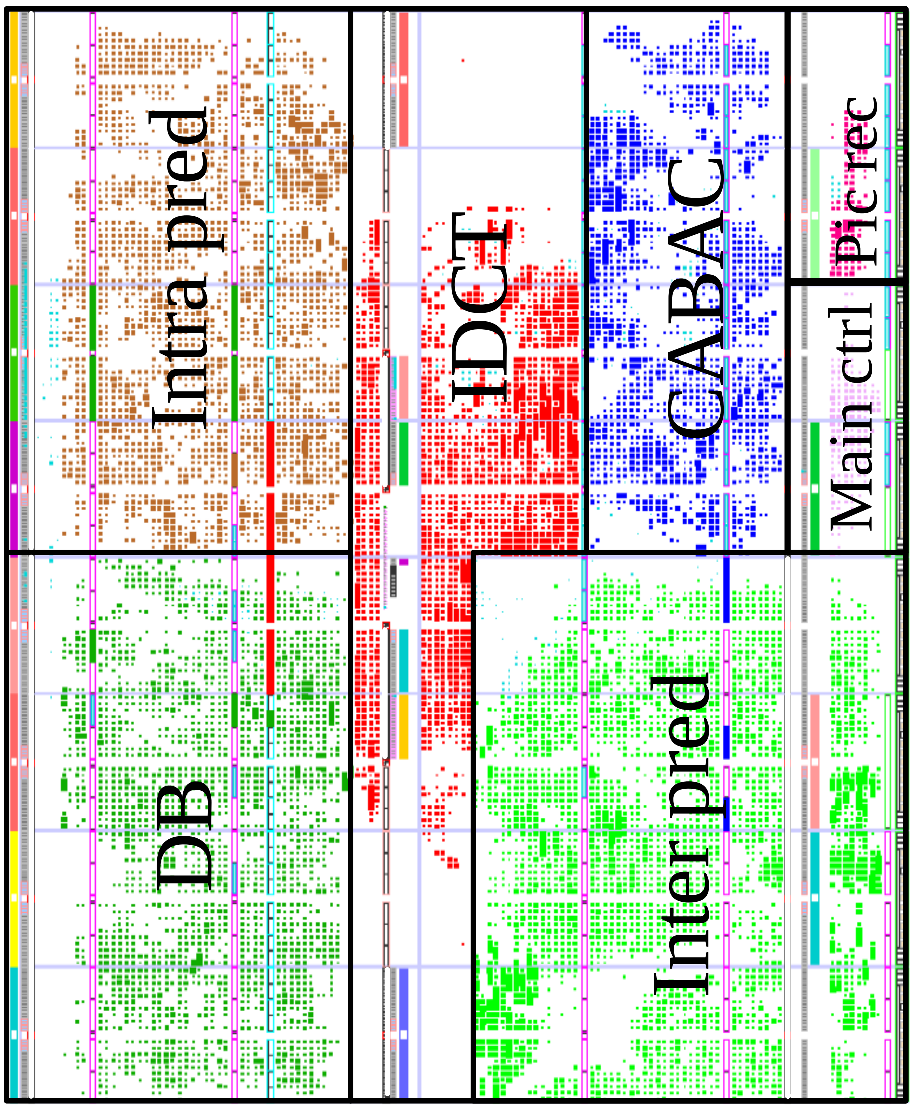


Figure 3.4: H.264/AVC decoder implemented on Xilinx FPGA Virtex-5 XC5VLX110T.

In the second step, we estimate the SFR of each module via Monte-Carlo fault simulation. For each module, the faults are injected to LUTs in the post-mapped simulation model. Then, we perform post-mapped simulation and capture the simulation result of the critical registers and the critical outputs. By comparing with the golden circuit result, we can calculate the average SFR for each module simulated. Lastly, the average SFR is assigned to the CRAM bits that belong to the module.

Table 3.3 and 3.4 show the experimental result for MTTF improvement and runtime under different CRAM granularities. For system-level analysis, the allocation in CRAM is proportional to FPGA resource although there are still some unused CRAM bits in each module. The system MTTF improvement is the weighting summation for each module improvement. The experiment in granularity of 1312 (which is as the same setup as Xilinx Virtex-5 configuration [19]) shows nearly 40.84% MTTF improvement. If we change granularity to 256 and 64, the improvements can even achieve 54.71% and 60.01%. The reason for such large improvement is that high percentages of CRAM bits have low SFR. This gives the rescheduling algorithm room to improve MTTF for other modules with high SFR. Negative MTTF improvement in the IDCT and de-blocking filter shows that the rescheduling algorithm sacrifices MTTF in those two modules. The bad module MTTF in the IDCT and de-blocking filter don't degrade system MTTF. In fact, the SFR in IDCT and deblocking filter are rarely low so there is no significant drawback even if we increase their MTTF via low scrubbing frequency. However, system MTTF improves significantly since other modules have more time slots to enhance MTTF via memory scrubbing in higher refresh frequency.

3.5 Summary

A Heterogeneous CRAM Scrubbing technique is proposed in this paper to improve the robustness of FPGAs against SEU induced system failures. To estimate the

Functional block	# of FF	# of LUT	# of CRAM bits	MTTF improvement		
				CRAM granularity		
				64	256	1312
CABAC	2278	9027	1093462	37.91%	35.21%	21.17%
Intra prediction	5094	14634	3636129	60.43%	40.54%	35.24%
IDCT	2471	10405	2971154	-1.97%	-1.87%	-2.09%
Picture Reconstruction	547	416	165051	28.27%	15.64%	8.27%
Deblocking Filter	6648	7613	3638842	-0.84%	-0.75%	-1.07%
Main Controller	418	466	205257	55.53%	36.73%	21.66%
Inter Prediction	8157	16133	5856249	71.73%	57.65%	48.57%
System	-	-	-	60.01%	54.71%	40.84%

Table 3.3: MTTF IMPROVEMENT ON THE H.264/AVC DECODER

CRAM granularity	Runtime (s)
64	75012
256	20850
1312	5398

Table 3.4: HCS RUNTIME FOR THE H.264/AVC DECODER

improvement from CRAM scrubbing, we use a stochastic fault modeling technique which allows us to calculate the MTTF improvement from CRAM scrubbing. Considering different levels of impact from SEU induced soft errors, we present a dynamic programming algorithm providing a HCS solution that effectively improves the system MTTF based on the system failure sensitivities of the CRAM bits. Our system-level estimation using a H.264/AVC decoder implemented on a Xilinx Virtex-5 FPGA shows that the proposed HCS method improves MTTF by 60% compared to the existing homogeneous CRAM scrubbing. Such improvement virtually does not have any overhead, i.e., virtually no change on area, performance and power at the system.

CHAPTER 4

In-Place Decomposition for Robustness in FPGAs

Robustness with respect to SEU in FPGA has been extensively studied in the literature [DH06, JMA12, BCD11]. To mitigate the impact of SEU, previous research has proposed radiation hardened static memory elements which reduce the susceptibility of SEUs [ZCW11, SAS10b, WCW09]. Specific FPGA architectures have been developed such as radiation hardened FPGAs from Xilinx [Xilb] and anti-fuse based FPGAs from Actel [Actb]. Although the aforementioned hardening techniques improve resilience against SEUs, radiation hardened devices usually do not meet the industrial demand due to the additional cost and low densities.

Recently, several logic resynthesis algorithms have been proposed to improve robustness for commodity FPGA with minimal area, power, or performance penalties. ROSE [HFH08] evaluates robustness during iterative logic re-mapping and remaps a logic block to a robust block template with path reconvergence. However, ROSE can change the topology of the LUT-level network, resulting in physical re-synthesis and potentially slowing down design closure between logic and physical resyntheses. IPR [FHH09] performs logic transformation while preserving the topology of the LUT-level network, and removes the aforementioned design closure problem. It maximizes the identical configuration bits corresponding to complementary inputs of an LUT such that the faults seen at a pair of complementary inputs has less possibility of propagation and the overall reliability is optimized. In essence, ROSE and IPR use logic masking/redundancy to migrate the impacts of

SEUs, and their MTTF improvements could be limited when circuits are heavily optimized for area and therefore have little implicit logic redundancy.

The state of art FPGAs such as Xilinx Vertix 7 and Altera Stratix IV [Xilc, alt] use decomposable LUT, where an LUT can be decomposed into two or more smaller LUTs and a second output pin is also provided (see Figure 4.1(a) and Figure 4.1(b)), to improve both performance and logic density [CR06]. Leveraging decomposable LUTs and under-utilization of large-sized LUTs, [LHM10] proposed to duplicate the logic inside a not fully used LUT, and increase MTTF without increasing the number of LUTs. However, the coding to combine the duplicated logic is done in the fanout LUTs. This leads to extra interconnects and potentially heavy area penalty and slows down design closure between logic and physical resyntheses.

In addition to decomposable LUT, the programmable logic block (PLB) in modern FPGA chips also have dedicated carry chain or adder (see Figures 4.1(a)) and 4.1(b)). While the carry function can be implemented by LUT in the same PLB, these carry chain circuits are built in as alternative circuits to obtain high speed for applications such as networking with extensive carry computation [ZP10]. However, the chip level utilization rate of these built-in carry chains is typically less than 20%.

This chapter proposes an in-place decomposition (IPD) for robustness in FPGA. IPD decomposes the logic function originally implemented by one decomposable LUT into two subfunctions to be implemented by smaller LUTs and to combine (also called converge in this paper) the subfunctions by carry chain. This decomposition introduces redundancy as the original logic function is now implemented with extra circuit (i.e. carry chain) and such redundancy is used explicit to make circuit more robust against soft errors.

We propose an ILP (integer linear programming) algorithm to solve IPD optimally inside each PLB and applied ILP iteratively to PLBs at the chip level. For

10 largest MCNC combinational benchmark circuits synthesized by ABC mapper, FMD from [LHM10] improves MTTF by 10%, but the proposed IPD improves MTTF by 3.33× for the PLB architecture similar to the Altera Stratix-IV when the 20% utilization rate of carry chain is assumed. The gap between 10% and 3.33× is the improvement due to performing logic converging within the same PLB.

The rest of this chapter is organized as follows. Section 4.1 presents the preliminaries and problem formulation. Section 4.2 discusses the IPD problem and its properties. Section 4.3 presents IPD algorithm. The experimental results given in Section 4.4 and this chapter is concluded by 4.5. To the best of our knowledge, this work is the first systematic study on robustness using both the built-in carry chain (or adder) and decomposable LUT features of modern FPGAs.

4.1 Preliminaries and Problem Formulation

4.1.1 Dual-output LUT and Built-in Carry Chain/Adder PLB Architecture

Emerging FPGAs employ dual-output features in their LUTs to increase performance and density. For example, Xilinx Virtex-7 has a decomposable 6-input LUT architecture with a dual-output capability which can be configured as a single 6-input function or two independent functions with a total of 5 unique inputs (see Figure 4.1(a)). Altera Stratix-IV uses an 8-input adaptive logic module (ALM), which contains two adaptive LUTs (ALUT) as shown in Figure 4.1(b). An ALM has similar features as the Virtex-5 6-input LUTs but with different input sharing constraints. Moreoever, an ALM can implement any single 6-input function but only a subset of single 7-input functions. Table 4.1 summarizes the input sharing constraints when all input pins are utilized for both architectures. Specifically, the input sharing constraints limit that the two large functions implemented on a dual-output LUT must have some inputs in common. Otherwise, only two independent small functions (without any input shared) can be implemented. For example, when the second output is used, an ALUT in an ALM can implement up to two 6-input functions with four inputs shared between the two ALUTs. In particular, two functions that have number of inputs less than six can be implemented on an Stratix-IV ALM if the total number of unique inputs are less than eight. Only when two 6-input functions are implemented on an Stratix-IV ALM, the two functions are required to have the same operations with four inputs shared.

size of dual-function	# of shared inputs
Virtex-7 6-input LUTs	
5 , 5	5
Stratix-IV ALM	
4 , 4	0
5 , 3	0
5 , 4	1
5 , 5	2
6 , 6	4

Table 4.1: Input sharing conditions (and also decomposition cases to be used by IPD algorithm) for Virtex-7 6-input LUTs and Stratix-IV ALM

In addition to decomposable LUTs, the programmable logic block (PLB) in modern FPGA chips contains a dedicated carry chain or adder (see Figures 4.1(a)) and 4.1(b)). While the carry function can be implemented by a LUT in the same PLB, these carry chain circuits are built in as alternative circuits to obtain high speed. Therefore, it was *not* the original intention of the FPGA architecture design to use both the carry chain and LUT in the same PLB. Although new synthesis tools have been developed to leverage both carry chain and LUT in a same PLB, the chip level utilization rate of carry chains is typically less than 20%.

4.1.2 Prevailing Technique Leveraging Dual-Output Feature for FPGA Robustness

Fully masked duplication (FMD) has been proposed in [LHM10], where the explicit duplication of a function is performed by utilizing an unused second output of a dual-output LUT. Then, AND or OR encodings for logic masking for robustness enhancement are added at all direct fan-out LUTs of a duplicated LUT (see Figure 4.2) when unused input pins are available for fanout LUTs. A more flexible PMD (partially masked duplication) has also been developed. Note that the encoding of AND or OR is called converging logic in this work.

4.1.3 Formulation of In-place Decomposition

To formulate the in-place decomposition (IPD) method, we first define the decomposition and converging of a function. Given an n-input function F, decomposition transforms F into two subfunctions, $F1$ and $F2$, where the total number of unique inputs is equal to n, and converging logic $\odot$ is applied such that $F = F1 \odot F2$. Figure 4.3 is an example of decomposition, where a 5-input function F is transformed into a 4-input subfunction $F1$ and a 3-input subfunction $F2$ with two inputs shared. Then, the outputs of $F1$ and $F2$ are combined by the converging logic.

By taking advantage of the dual-output feature of LUTs, two subfunctions can be implemented in a single dual-output LUT. Decomposition can be divided into two categories, fully-input-shared decomposition where all inputs are the same for the two subfunctions, and partially-input-shared decomposition, which includes the case where there is no common input for the two subfunctions.

Unlike FMD and PMD in [LHM10], which require spare input pins of LUTs for converging, IPD utilizes the unused dedicated carry chain (or adder) within the same PLB to converge the two decomposed subfunctions. Note that although there are up to $2^4 = 16$ types of converging gate, an AND type and an OR type converging gates are sufficient to cover all error masking capabilities from all converging gate (explained in Section 4.2, which can be implemented by a carry chain/adder.

IPD offers clear advantages over the duplication technique in [LHM10]. First, duplication implies two identical subfunctions. It is an over-simplified case of decomposition that enables distinguished subfunctions, which could differ in terms of functionality and function size and lead to more potential for improvement. Secondly, IPD performs logic decomposition and converging within the original PLB. It does not change PLB level placement and routing, and therefore there is no PLB level penalty on timing, area and design closure. The only timing overhead comes from the logic delay of the signal propagating through the converging logic implemented by the high performance carry chain. However, it has been reported that the majority delay in FPGA is contributed by its routing structure [Geo00, DeH96, CX11], and the signal propagation of Xilinx Virtex-5 carry chain is 17 times faster than the LUT [ZP10]. Therefore, the timing overhead of the converging logic is negligible considering the timing delay of the entire circuit. In contrast, the duplication (namely FMD) algorithm from [LHM10] carries out duplication and encoding (called converging in this paper) in multiple PLBs, requiring extra interconnects between PLBs and a heavy penalty in area, delay and design closure.

With respect to the above discussion, we formulate the IPD problem as follows.

Formulation 2. *Given a circuit C, IPD decomposes the logic function for a PLB into two subfunctions such that the two subfunctions are implemented by the decomposable dual-output LUT in the PLB and are combined (or converged) by the carry chain in the PLB, and the resulting soft error rate is minimized.*

While our formulation and algorithms presented here apply to any converging logic, for simplicity of presentation, we consider only AND and OR converging logic.

4.2 In-Place Decomposition Properties

In this section, we first introduce an efficient soft error rate update method, and then present two important properties and their proofs for optimal SER reduction which significantly improves the efficiency of the proposed algorithm. Next, upon identifying don't cares in LUTs, we show that the flexibility of decomposition can be further improved when don't cares are considered.

4.2.1 Updating SER after In-Place Decomposition

In most of the fault tolerant FPGA resynthesis algorithm, such as ROSE and IPR, the reliability evaluation (i.e. SER update) is done through fault simulation, which results in excessive overhead in runtime. In this section, we present a analytical method where SER update after decomposition can be obtained efficiently given the SER of each CRAM bit of an LUT.

According to Eq. 2.1, we can explain the SER of an LUT SRAM bit in the following way. Assume that L_i is i_{th} bit of an LUT L, the SER of an LUT SRAM bit SER_{L_i} is the probability that the erroneous content of SRAM bit L_i is accessed and used as the LUT output, multiplied by the probability that the erroneous output of L is propagated to the primary output. Thus, SER_{L_i} of the an LUT before decomposition can be formulated as the following equation,

$$SER_{L_i} = I(L, x_i) \times O(L,x_i) \times P(L, x_i), \tag{4.1}$$

and the average SER of LUT L is

$$SER_L = \sum_{0 \leq i \leq 2^N} SER_{L_i} \tag{4.2}$$

where x_i is a permissible input vector of L which retrieves the content of L_i, $I(L, x_i)$ is the probability of x_i occurring at L, and N is the number of inputs of LUT L. The probability of passing the error from LUT L to the primary outputs under LUT input vector x_i can be divided into two parts, the internal observability $O(L, x_i)$ and the external observability $P(L, x_i)$. The internal observability $O(L, x_i)$ is the probability that the soft error on L_i can be observed at the PLB output under LUT input vector x_i, and the external observability $P(L, x_i)$ is the probability that the error can propagate from the PLB output to the circuit primary outputs. Note that x_i is one-to-one mapped to the LUT SRAM bit L_i before decomposition, and the internal observability $O(L, x_i)$ is 1.0 because the output signal of LUT is directly routed to the PLB output without any logic masking capability before decomposition (see Figure 4.1. Therefore, SER_{L_i} before decomposition is:

$$SER_{L_i} = I(L, x_i) \times P(L, x_i), \tag{4.3}$$

After decomposition, assuming that the decomposed two subfunctions are implemented by the two smaller LUTs, $L1$ and $L2$ of a decomposable LUT L, the two decomposed LUTs share the same inputs for L. That is, a input vector x_i of L retrieves two CRAM bit instead of one CRAM bit, one from $L1$ and the other one from $L2$. The total SER of LUT L after decomposition becomes the the summation of SER of every CRAM bit in $L1$ and $L2$. In addition, because the decomposed subfunction can be smaller than the original function, there are multiple permissible input vectors of L that can retrieve the same CRAM bit of the decomposed subfunction. Therefore, the decomposed LUT $L\prime$ can be calculated as follows.

$$\begin{aligned} SER_{L\prime} &= \tfrac{1}{2^N} \; (\sum_{i=1}^{m(L1)} SER_{L1_i} + \sum_{j=1}^{m(L2)} SER_{L2_j}) \\ &= \tfrac{1}{2^N} \; (\sum_{i=1}^{m(L1)} \sum_{x_k \in \psi(L1_i)} I(L,x_k) \times O(L1,x_k) \times P(L,x_k) + \\ &\qquad \sum_{j=1}^{m(L2)} \sum_{x_k \in \psi(L2_j)} I(L,x_k) \cdot O(L2,x_k) \cdot P(L,x_k)) \end{aligned} \tag{4.4}$$

$m(L1)$ and $m(L2)$ are the number of SRAM bits of $L1$ and $L2$, and $\psi(L1_i)$ denotes the set of input vectors that can retrieve LUT CRAM bit $L1_i$ ($\psi(L2_j)$ denotes the set for $L2_j$, respectively). Specifically, because each input vector retries two LUT SRAM bits after decomposition, one from $L1$ and the other from $L2$, each input vector x_k contributes $I(L,x_k) \times O(L1,x_k) \times P(L,x_k)$ and $I(L,x_k) \times O(L2,x_k) \times P(L,x_k)$ in the SER calculation of (4.4). Therefore, the average SER of an LUT after decomposition can be rewritten to:

$$\begin{aligned} SER_{L\prime} &= \frac{1}{2^N} \sum_{i=1}^{2^N} I(L,x_i) \times O(L1,x_i) \times P(L,x_i) + I(L,x_i) \times O(L2,x_i) \times P(L,x_i) \\ &= \frac{1}{2^N} \sum_{i=1}^{2^N} I(L,x_i) \times P(L,x_i) \times (O(L1,x_i) + O(L2,x_i)) \end{aligned} \tag{4.5}$$

Combining Eq. 4.3 and Eq. 4.5, and based on the fact that x_i and L_i are one-to-one mapped. We can rewrite Eq. 4.5 to the following equation.

$$SER_{L\prime} = \frac{1}{2^N} \sum_{i=1}^{2^N} SER_{L_i} \cdot (O(L1,x_i) + O(L2,x_i)) \tag{4.6}$$

According to (4.6), the SER update after decomposition mainly depends on the internal observabilities $O(L1,x_i)$ and $O(L2,x_i)$. Since converging logic is applied to the outputs of $L1$ and $L2$ after decomposition, some internal observabilities can be reduce to 0 according to the type of the converging logic and the truth tables of $L1$ and $L2$. The update of $O(L1, x_i)$ depends on the output value of $L2$ under input vector x_i, and $O(L2, x_i)$ depends on that of $L1$, respectively. For example, in the case of using AND as the converging logic, $O(L1, x_i)$ can be reduced to 0 if the output of $L2$ is '0' under input vector x_i, and $O(L1, x_i)$ equals to 1.0 if the output of $L2$ is '1'. Therefore, the observability and the average SER of LUT can be updated very efficiently after decomposition.

In particular, duplication is a special case of decomposition, and the above SER update process is also valid for duplication. However, an efficient method for updating SER for duplication can be found in [LHM10].

4.2.2 Optimality of In-Place Decomposition

For decomposition to three or more subfunctions, TMR provides an optimal solution for LUT SER reduction if triplication and implementation of a voter for convergence is allowed. TMR is capable of tolerating any single node error, and therefore the SER of every node in the circuit is reduced to zero. However, for decomposition into two subfunctions, it is unclear how an optimal solution can be achieved. In this section, we discuss the optimality of decomposition into to two subfunctions. We reveal two important properties and show that the duplication provides an optimal fault rate reduction if it is allowed.

Lemma 1. *An AND and an OR are the optimal forms of logic for converging the two decomposed subfunctions.*

After IPD decomposes a logic function into two separate functions, these functions are converged using a logic gate, which can provide logic masking for faults in the two LUTs used to implement the subfunctions. In this proof, we demonstrate that AND and OR gates are the optimal forms of converging logic in terms of their logic masking capability, by comparing the logic masking capability of every possible form of converging logic.

Figure 4.4 illustrates every 2-input logic function, including two for constant logic and four which serve as single-input functions. Converging logic provides logic masking when the output of the gate remains constant when the value of one of the inputs is flipped. Therefore, XOR functions do not provide any logic masking capability because there is no scenario in which the output remains constant when an input is toggled.

Therefore, 8 out of 16 gates do not provide any logic-masking during convergence. The other 8 functions are AND or OR functions, including cases with inverted inputs which are equivalent to NOR and $NAND$ gates. However, an inverter does not provide any logic masking capability either. Therefore, each gate has exactly the same logic masking capability as AND or OR.

Lemma 2. *Decomposition is optimal when the two sub-functions and are identical to the original function.*

Suppose the Boolean function implemented by the original decomposable LUT L is F. After applying fully-input-shared decomposition, the Boolean function F is decomposed into one converging gate and two subfunctions, $F1$ and $F2$, where $F = F1 \odot F2$. The SER can be reduced by masking logic propagations from $F1$ and/or $F2$ with respect to the input patterns of F. Under fully-input-shared decomposition, since each input pattern is one-to-one mapped to one SRAM bit from $F1$ and one SRAM bit from $F2$, the configuration of an CRAM bit in $F1/F2$ is independent to one in $F1/F2$. Therefore, optimal SER reduction can be obtained by optimizing the fault rate of input patterns independently. Lemma 1 states that feasible converging logic can be based only on AND or OR type functions. For AND converging logic, both the outputs of $F1$ and $F2$ must be '1' when the output of F is '1'. In this case, the AND gate provides no logic masking since the output will change if either input logic value is flipped.

However, in the case when the output of F is ‘0’, the inputs $F1$ and $F2$ can be “00”, “01”, or “10”. The first case, when $F1$ and $F2$ are both ‘0’, provides twice the logic masking capabilities of the other two cases because the logic value of the output is retained when either input is flipped, in the case of AND converging logic. Therefore, logic-masking is maximized when the two decomposed subfunctions are identical.

In the case of OR converging logic, consider the case when F is ‘1’. This can only occur when $F1$ and $F2$ are “01”, “10”, or “11”. It is clear that the latter case, in when $F1$ and $F2$ are identical, provides twice the logic masking capabilities of the case when $F1$ and $F2$ have opposite values. In conclusion, logic masking is maximized when the two decomposed subfunctions are duplicated.

4.2.3 Decomposition with Don’t Care Flexibility

Based on the CRAM utilization and the impact of SEU of a circuit implemented on an FPGA, CRAM bits can be divided into the following three categories [Le]:

- *Essential bits*: Those CRAM bits that are used by the design.
- *Critical bits*: A subset of essential bits; those essential bits which cause functional failures at the primary outputs of the circuit if their values are flipped. A bit b is critical if there exists an input vector x such that $C_b(x) \neq C_{\bar{b}}(x)$.
- *Dont-care bits*: Those bits that do not affect the circuits primary outputs when they change state, even if they are used by the design. A bit b is a dont-care bit if $C_b(x) = C_{\bar{b}}(x)$ for any input vector x.

In general, the set of essential bits is a superset of critical bits contains a subset of don’t-care bits as Figure 4.5 illustrates. For LUTs, the essential bits are those

LUTs that have functions mapped on them. In addition, these LUT essential bits can be further divided into critical bits and don't-care bits.

Dont-cares have been commonly and widely used in circuit optimization [MBJ11, SBT91, MB05, McM05, CM10]. For example, [CM10] proposed to consider logical dont-cares when perform technology mapping in FPGA to improve circuit reliability. In this section, we propose an improved decomposition with consideration of logic dont-cares of LUTs. In order to preserve the an LUT function after decomposition, obviously a strait forward method is to perform boolean matching for each minterm of the LUT function, regardless of whether the minterms are dont-cares or not. In this manner, the decomposition result may be limited due to Boolean matching of those dont-care minterms. In fact, those don't care minterms do not necessarily required to be mapped explicitly. Therefore, we propose to consider leveraging the dont-care information to increase the flexibility of decomposition.

By removing these constraints of mapping don't-care bits, the IPD algorithm is given more design freedom to perform an effective decomposition. For example, consider the LUT depicted in Table 4.2, before decomposition. Though they are all essential bits, but CRAM bit correspond to input 000, 010, 100, and 110 are also dont-care bits, where the SER of don't-cares bits are 0. Ordinarily, the Boolean equation representing this LUT would be:

$$Y = ABC + \bar{A}\bar{B}\bar{C} \tag{4.7}$$

When the two minterms of Boolean function Y need to be matched for decomposition, there is no valid decomposition to smaller subfunctions for function Y. However, assuming that the LUT bit corresponding to inputs "000" never occur in the circuit, which is defined as a satisfiability don't-care bit. The SER of the LUT bit is 0 and it can be ignored during Boolean matching of the bit because it does

Inputs	LUT Value	SER
000	1	0
001	0	0.1
010	0	0
011	0	0.1
100	0	0
101	0	0.1
110	0	0
111	1	0.1

Table 4.2: An example of a LUT in which decomposition is improved when LUT Boolean matching is avoided for dont-care bits.

not have impact on the circuit functionally. Thus, the new equation representing this LUT is:

$$Y = ABC \tag{4.8}$$

With the dont-care information, the function Y can be decomposed to smaller functions as Figure 4.6 shows. This example clearly shows that decomposition considering every essential bit limits the quality of the result in the original IPD algorithm. In this manner, the revised IPD, which identifies the don't-care bits of LUTs and does not perform Boolean matching for don't-care bits, is able to represent logic functions with fewer critical bits, and is also able to perform decomposition in a manner which more effectively minimizes the total SER of the circuit.

There are several methods one can use to identify dont-care bits in a circuit. The most common method to identify dont-care bits is to perform exhaustive simulation with all possible input vectors, which results in excessive runtime. Therefore, several previous works have focused on improving the runtime while providing good quality of dont-care identification results [MBJ11, SBT91, MB05, McM05, CM10]. In this work, we use the method similar to [CM10], where a windowing algorithm is used to identify subcircuits that are small and to effectively identify their dont-cares.

4.3 IPD Algorithm

4.3.1 ILP Formulation for SER Optimization of an LUT

According to (4.6), the SER update of an LUT after decomposition is independent to that of other LUTs. Therefore, the overall optimal SER of IPD on a given circuit C is the summation of the optimal decomposition of each LUT in C. For a given LUT $L \in C$, we formulate the IPD problem to an Integer Linear Programming (ILP) problem as follows.

$$\text{Minimize} \sum_{i=1}^{2^N} SER_{L_i} \times (O(L1, x_i) + O(L2, x_i)),$$

subject to the following five sets of constraints.

4.3.1.1 Decomposition selection constraint

$$\sum_{j=1}^{\phi(L)} d_j \leq 1$$

$\phi(L)$ is the set of decomposition templates with different sizes of the two subfunctions (such as those in Table 4.1 for Xilinx and Altera PLBs) that are applicable to a decomposable LUT L, and $d_j \in \{0, 1\}$ is 1 if the $j_t h$ decomposition template is selected for LUT L, and the this constraint guarantees that there is at most one decomposition template is selected and applied to the LUT. This constraint allows the algorithm to solve multiple decomposition templates at a time and only selects the template leading to the lowest SER.

4.3.1.2 Boolean matching constraints of a decomposition template d_j for the LUT

$$\sum_{x_i \in mt(L)} t(j, x_i) = |mt(L)| \times d_j, \quad 1 \leq j \leq \phi(L)$$

$mt(L)$ is the set minterms of L that are required to be matched, represented by the corresponding input vector x_i. $t(j, x_i)$ is a binary variable, where $t(j, x_i) = 1$ indicates the Boolean function of L and that of decomposition d_j are equivalent under input vector x_i. This set of constraint guarantees that a decomposition template d_j cannot be selected if there exists any inconsistency in the decomposed Boolean function for those minterms.

4.3.1.3 Boolean matching constraints for a minterm

$$\begin{aligned} 0 \leq L1(j, x_i) + L2(j, x_i) - 2 \cdot t(j, x_i) \leq 1, &\quad if L(x_i) = 1 \\ 2 \leq L1(j, x_i) + L2(j, x_i) + 2 \cdot t(j, x_i) \leq 3, &\quad if L(x_i) = 0 \\ 1 \leq j \leq \phi(L), \; x_i \in mt(L) & \end{aligned}$$

The third set of constraints perform Boolean matching for decomposition template j with AND converging logic for a minterm (corresponding to the input vector x_i), where $L(x_i)$ is the output value of LUT L under LUT input vector x_i, and $L1(j, x_i)$ and $L2(j, x_i)$ are the output values of L1 and L2 of decomposition template j under input vector x_i. For simplicity, we show only constraints for AND converging logic, since constraints for OR converging logic can be easily inferred from the above constraints.

4.3.1.4 Observability update constraints

$$
\begin{gathered}
0 \le d_j + L2(j, x_i) - 2 \cdot P1(j, x_i) \le 1 \\
0 \le -1 \cdot d_j + L2(j, x_i) + 2 \cdot B1(j, x_i) \le 1 \\
-1 \cdot P1(j, x_i) + O(L1, x_i) \ge 0 \\
B1(j, x_i) + O(L1, x_i) \le 1 \\
1 \le j \le \phi(L),\ x_i \in mt(L)
\end{gathered}
$$

If decomposition template j is applied with AND converging logic, the above constraints calculate the internal observability for each input vector according to the function of $L1$ and $L2$. $P1(j, x_i)$ and $B1(j, x_i)$ are binary variables which represent the propagation and masking of the signal from $L1$. For simplicity, we show only constraints for observability update for $L1$ with AND converging logic. The constraints for $L2$ and OR converging logic can be generated similarly.

4.3.1.5 Default observability constrains

$$
\begin{gathered}
O(L1, x_i) + \sum_{j=1}^{\phi(L)} d_j \ge 1 \\
O(L2, x_i) + \sum_{j=1}^{\phi(L)} d_j \ge 1 \\
1 \le j \le \phi(L),\ x_i \in mt(L)
\end{gathered}
$$

The last set of constraints implies that when none of the decomposition templates are feasible, the observabilities are 1.0 for any input vector x_i. These constraints guarantee that a decomposition template can be selected only if it leads to lower SER compared to the original implementation.

4.3.2 IPD for Critical Bit Reduction

In the previous section, we proposed an IPD algorithm for SER reduction, which mitigates the impact of SEUs on LUT CRAM bits. In this section, we discuss how IPD can reduce the number of critical bits in a design. As defined in Section 4.2.3, critical bits are those bits which modify the circuits external functionality when impacted by an SEU.

Decomposition achieves critical bits reduction for the following two reasons:

4.3.2.1 Complete fault coverage

Figure 4.7 illustrates the complete fault coverage of duplication. Suppose that the original LUT is a 5-input AND function, and each of its 32 CRAM bits is critical. Then, we apply duplication to the function using AND converging logic, doubling the number of critical bits. Since the two subfunctions are identical, the inputs to the AND converging logic are identical. When both inputs to the AND gate are "00", any single $0 \rightarrow 1$ fault of at input is masked. Therefore, all of the 0-bits in both LUTs are completely covered and immune to SEUs, and are reduced from critical bits to dont-care bits. In this example, the number of critical bits is reduced from 32 to 2.

4.3.2.2 Reduction in the number of CRAM bits needed to implement a function

Because two decomposed logic functions can use less CRAM bits than the original function, decomposition can reduce the number of critical bits in the circuit. Figure 4.8 illustrates the decomposition of a single 6-input AND function to two 3-input AND subfunctions combined by 2-input AND logic, where such decomposition can be implemented by dual-output LUT and the carry chain. Assuming that all of the CRAM bits are critical and the 2-input AND logic is implemented by carry-chain, the number of critical bits in this example can be reduced from 64 to 16, assuming all of the CRAM bits used for the subfunctions after decomposition are critical bits.

Combining the two factors above, we show that IPD can lead to a reduction in the number of critical bits. While duplication is unique and its feasibility is determined by the number of inputs of the LUT and the availabilities of the second output of the LUT and the built-in carry chain/adder, decomposition has varying forms such as the number of input pins shared and the assignments of pins to the decomposed LUTs. By evaluating all possible forms of decomposition and duplication in the experiment, we found that the optimal critical bit reduction can be derived from either duplication or decomposition. Therefore, unlike IPD for fault rate reduction, duplication does not guarantee an optimal critical bit reduction. Note that IPD of a LUT, for both cases of duplication and decomposition, does not have any impact on the critical bits on other LUTs. This property implies that by solving the critical bit reduction for each LUT we can guarantee that the number of critical bits of the entire circuit is reduced.

4.3.3 Overall IPD Algorithm

The overall IPD algorithm flow is illustrated in Figure 4.9. By taking advantage of the two properties described in the previous section, the IPD algorithm has a pruned solution space and thus improved performance.

Given a circuit C, the IPD algorithm begins with calculating the SER of each LUT SRAM using the soft error evaluation platform proposed in Chapter 2 and performs dont-care identification. For each LUT, we first check whether the LUT has not been processed and the PLB the LUT belongs to has an un-used carry chain or adder. For an LUT that has an un-used carry chain, if fully-input-shared decomposition can be applied and it uses less number of critical bits compared to the original number before decomposition, we simply perform fully-input-shared decomposition because it results in the optimal fault rate reduction as explained in the previous section. On the contrary, if it cannot be applied, we perform decomposition to smaller functions. We first generate all valid decomposition templates for the LUT. A decomposition template is defined by the size of the decomposed LUTs, i.e. the number of their inputs, and the number of inputs shared between the decomposable LUTs, and it is valid if the decomposition can be mapped onto the PLB architecture described in Figure 4.1 and Table 4.1. Then, we perform the ILP-based algorithm for each of those templates and select the one that results in the minimal SER with lower number of critical bits. After all the LUTs are processed, we update the full-chip SER of C and the total number of critical bits.

4.4 Experimental Results

The proposed IPD algorithm is implemented in C++ with the mosek [mos] solver on an Ubuntu Linux powered server with Xeon 2.4GHz CPU and 2Gb memory. The 10 largest MCNC combinational benchmark circuits are tested on both the Xilinx Virtex-5 6LUT architecture and the Altera Stratix-IV ALM architecture. All the benchmarks are first mapped into 6-input LUT logic networks by the Berkeley ABC technology mapper [ABC] with edge flow optimization, which has a special property that the mapped circuits are more suitable to be packed into dual-output LUTs [JCC08] than the mapped circuits from other settings. Then, the LUT merge algorithm in [AKA08] is used to merge pairs of small LUTs (<4 inputs) into dual-output LUTs. In this section, we perform two sets of experiments. First, we perform IPD without considering don't-cares of LUTs on PLB architecture similar to Xilinx Virtex 7 and Altera Stratix-IV. In addition, we also perform FMD for comparison comparison purpose. Next, we perform IPD with don't-care flexibility on the same benchmark to show SER and critical bits improvement by don't-cares.

4.4.1 IPD matching all minterms

For FPGA architectures under study, we apply IPD under different utilization rates of build-in carry chain, e.g., 10% means that randomly, 10% of build-in carry chain inside the used logic blocks are not available to be used by IPD. We compute MTTF as reversely proportional to the chip level SER since there is no routing or area change.

4.4.1.1 Characteristics of circuits and architectures

Figure 4.10 illustrates LUT utilization in terms of the size of logic functions mapped for the 10 largest MCNC combinational circuits. In addition, Figure 4.11 shows the SER contribution for those functions before applying IPD. Both figures show that 6-input functions have the greatest impact on the circuits in terms of SER, due to their large number and the greater amount of CRAM bits they utilize.

For Xilinx 6LUT architecture with five shared input pins, in-place duplication can be used for functions with up to 5 inputs without losing optimality. Also note that no decomposition is available for 6-input functions for this architecture. Therefore, IPD is in fact the in-place duplication with AND or OR as the converging logic for Xilinx 6LUT.

As shown in Figure 4.12, Altera ALM architecture has two adjustable LUTs (ALUTs). The upper two 4-input LUTs belong to one ALUT, and the lower two 4-input LUTs belong to the other. Note that three inputs are shared among all of the four LUTs. According to the architecture of an ALUT, it contains two 4-input LUTs with three inputs shared. In other words, the two LUTs in an ALUT have a total of five unique inputs. Therefore, a function with up to five inputs can be decomposed and implemented by one ALUT. As shown in Figure 4.13, we can perform decomposition for 6-input functions by using three 4-input LUTs (with three inputs shared) plus two converging gates implemented by the two adders.

Note that ALM can implement a subset of 7-input functions, which can be decomposed in the similar way like 6-input functions except that we utilize all four LUTs in the ALM (See Figure 4.13).

circuit	SER									
	BASE	FMD	IPD							
			Xilinx 6LUT				ALM			
			0%	10%	20%	30%	0%	10%	20%	30%
alu4	0.34%	0.33%	0.25%	0.26%	0.25%	0.27%	0.09%	0.11%	0.13%	0.17%
apex2	0.29%	0.26%	0.18%	0.19%	0.20%	0.21%	0.03%	0.05%	0.07%	0.12%
apex4	1.16%	1.10%	0.93%	0.95%	0.97%	0.99%	0.31%	0.41%	0.49%	0.60%
des	1.42%	1.41%	1.17%	1.20%	1.21%	1.27%	0.80%	0.85%	0.92%	0.95%
ex1010	1.24%	1.05%	0.51%	0.57%	0.65%	0.72%	0.27%	0.37%	0.47%	0.54%
exp5p	0.73%	0.62%	0.46%	0.48%	0.51%	0.52%	0.24%	0.30%	0.32%	0.39%
misex3	0.55%	0.49%	0.32%	0.34%	0.37%	0.38%	0.10%	0.15%	0.16%	0.23%
pdc	0.91%	0.83%	0.52%	0.56%	0.61%	0.63%	0.16%	0.22%	0.31%	0.38%
seq	0.63%	0.56%	0.39%	0.42%	0.44%	0.45%	0.11%	0.15%	0.21%	0.28%
spla	1.14%	1.05%	0.70%	0.74%	0.78%	0.82%	0.20%	0.31%	0.40%	0.48%
GeoMean	0.75%	0.68%	0.47%	0.50%	0.52%	0.55%	0.17%	0.22%	0.28%	0.35%
Ratio	1.00	0.91	0.63	0.67	0.70	0.73	0.22	0.30	0.37	0.47
MTTF Ratio	1.00	1.10	1.59	1.50	1.43	1.36	4.51	3.32	2.70	2.12

Table 4.3: Summary of IPD experimental results.

The experimental results for the baseline algorithm, FMD and the proposed IPD are summarized in Table 4.3 and Table 4.4. In the Table, "Xilinx 6LUT" are results for Xilinx Virtex-7 6LUT architecture, and "ALM" are results for Altera Stratix-IV ALM architecture, assuming 0%, 10%, 20%, and 30$ carry chain/adder utilization rates. For all of the benchmark circuits, we show the IPD improvement on the SER reduction and the MTTF improvement ratio compared to both the original circuits (baseline) and FMD algorithm. The MTTF improvement decreases as more (from 0% to 30%) carry chains are used. We also obtain higher

Circuit	Runtime (s)
alu4	1466
apex2	1137
apex4	1430
des	2022
ex1010	1635
ex5p	795
misex3	1235
pdc	3429
seq	1659
spla	3270
Average	1808

Table 4.4: IPD runtime.

MTTF for Altera ALM than for Xilinx dual-output 6LUT.

While FMD improves MTTF only by 10% on average, IPD (with a conservative 20% utilization rate for carry chain) improves average MTTF by 1.43× and 2.70× for Xilinx and Altera architectures, respectively. When all carry chains are available (utilization rate is 0%), IPD improves MTTF by up to 2.43× (see "ex1010") for 6LUT, and up to 9.67× (see "apex2") for ALM.

Because in-place duplication is used exclusively for Xilinx architecture, the gap between 10% and 1.43× is the improvement due to performing logic converging within the same PLB. Because the Altera architecture uses both in-place decomposition and in-place duplication, the gap between 1.43× and 2.70× is a good indicator of improvement due to decomposition.

Table 4.3 also presents runtime of the IPD algorithm. Generally, each PLB takes about 10s to obtain the optimal solution for duplication or decomposition.

4.4.1.2 IPD with don't-care flexibility

In the ten benchmark circuits evaluated in this paper, anywhere from 15% to 66% of used CRAM bits were discovered to be don't-care bits, with an average of 42%

Circuit	# of LUT Essential Bits	# of LUT Don't-care Bits	Don't-care Ratio
alu4	20244	5951	0.293
apex2	26008	4000	0.153
apex4	21082	10414	0.493
des	31596	11837	0.374
ex1010	24444	11522	0.471
ex5p	13176	8056	0.611
misex3	18464	6025	0.326
pdc	56480	37365	0.661
seq	25600	5679	0.221
spla	53396	34766	0.657

Table 4.5: LUT don't-care ratio.

of those used bits being don't-care bits. By removing the matching constraints for these don't-care bits, IPD is able to further increase the MTTF, decreasing the total circuit fault rate and reducing the number of critical bits.

Table 4.5 shows the number of don't-care bits in each circuit. Circuits with higher numbers of don't-care bits will typically benefit more from the revised IPD than those with limited numbers of don't-care bits, if all other factors are held constant.

The comparison of the MTTF improvement between the IPD matching all minterms and the improved IPD with don't-care flexibility is presented in Figure 4.14, where the average MTTF improvement ratio is increased from 3.7× to 4.55× with 0% carry chain/adder utilization rate and from 2.63× to 3.33× with 20% carry chain/adder utilization rate, respectively. Leveraging LUT don't-care information, the results indicate a substantial MTTF improvement, where 66% further MTTF improvement can be achieved. Particularly, with a 20% carry chain/adder utilization, the IPD with don't-care flexibility shows a significant improvement on the "ex1010" benchmark over the original IPD, where the SER is reduced by half approximately ($0.37 \rightarrow 0.19$) and thus the MTTF improvement ratio doubles by identifying don't-cares.

In addition, IPD can lead to a reduction in the number of critical bits after

Circuit	Orig.	IPD	IPD with DC
alu4	14293	11954	12475
apex2	22008	17841	18893
apex4	10668	8724	9136
des	19759	16809	17490
ex1010	12922	9523	10321
ex5p	5120	4658	4748
misex3	12439	10181	10698
pdc	19115	15495	16181
seq	19921	15922	16644
spla	18630	15401	16126
Average	15488	12651	13271

Table 4.6: Critical bit reduction after IPD and IPD with don't care flexibility assuming 20% carry chain/adder utilization.

decomposition. Note that decomposition not only reduces utilization of CRAM bits but also converts critical bits to don't-care bits in the subfunctions due to logic masking by the converging logic. Unlike duplication, where the critical bits and dont-care bits can be easily updated, we perform circuit simulation after IPD to calculate the number of critical bits.

In the ten benchmark circuits used in our experimentation, the average number of critical bits is 15487 before IPD. As summarized in Table 4.6, it is reduced to 12651 after IPD, and 13271 after IPD with don't-care flexibility, assuming a 20% carry-chain utilization. The corresponding improvement ratios are 0.821 and 0.860, respectively. The experimental results clearly show that the IPD algorithm not only reduces the SER but also reduces the number of critical LUT CRAM bits. The results show that the number of critical bits after IPD with don't-care flexibility is slightly higher than the original IPD algorithm. This indicates the tradeoff between SER and critical bits reduction. However, the proposed IPD algorithm targets on the optimal SER reduction and then reduces critical bits as the second objective, as presented in Figure. 4.9, resulting in slightly increased number of critical bits.

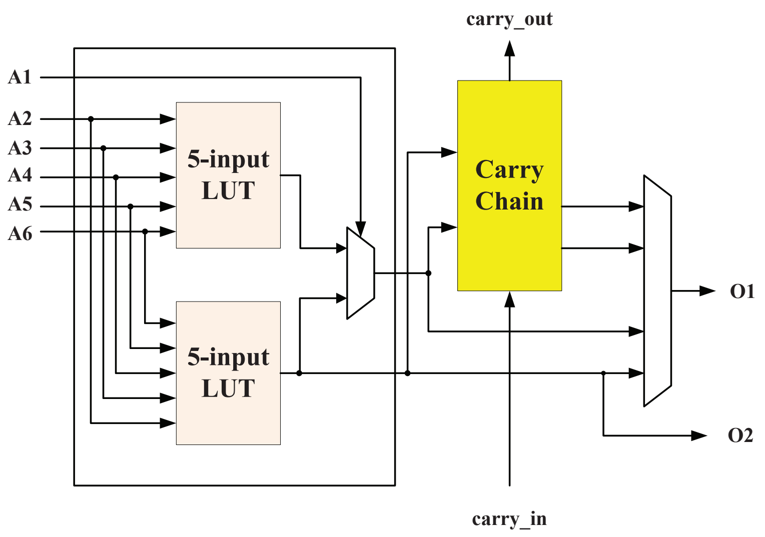
4.4.1.3 IPD improvement indicator

Figure 4.15 presents a good indicator about how much IPD could improve the reliability of the system. The (blue) bars represent the absolute difference between SER of "on" set and "off" set of circuits. The "on" (resp. "off") set is the CRAM bit set with logic '1' (resp. '0'). The (red) squares and the curve represent the MTTF improvement for corresponding circuits. Both of the two values are normalized for better demonstration. One conclusion that we can draw from the figure is that the more the gap between the SERs of "on" set and "off" set is, the more IPD increases the MTTF. This is because the large gap means either the "on" set dominates the SER of the circuits where faults can be efficiently masked by OR converging logic, or the "off" set dominates the SER where faults can be masked by AND converging logic.

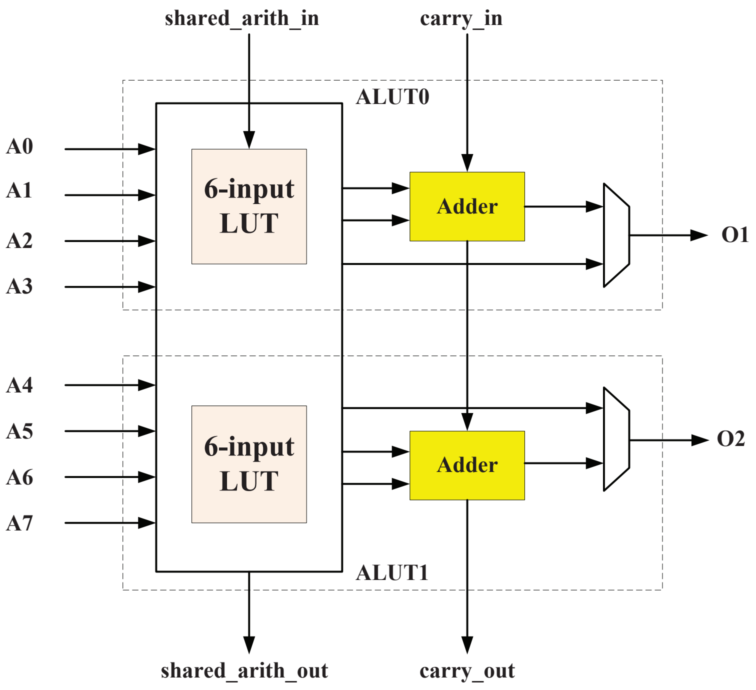
4.5 Summary

Leveraging decomposable LUT and built-in carry chain in modern FPGA, we have developed an in-place decomposition (IPD) to decompose the logic function originally implemented by one decomposable LUT into two subfunctions to be implemented by smaller LUTs and to combine (called converge in this paper) the subfunctions by carry chain. The logic decomposition and converging are carried out within the original PLB.

We have developed an ILP (integer linear programming) algorithm to solve IPD optimally within an PLB and iteratively apply IPD to PLBs at the chip level. We have also revealed the condition when in-place duplication is optimal and duplication can therefore be used instead of ILP-based IPD for better algorithm efficiency. For 10 largest MCNC combinational circuits synthesized by ABC mapper and with conservative 0% and 20% utilization rates for carry chain/adder, IPD improves MTTF by 4.55× and 3.33×, respectively, for PLBs similar to those in Altera Stratix-IV.



(a) Xilinx Virtex-5 6-input LUT architecture



(b) Altera Stratix-IV ALM architecture

Figure 4.1: Dual-output LUT FPGA architecture

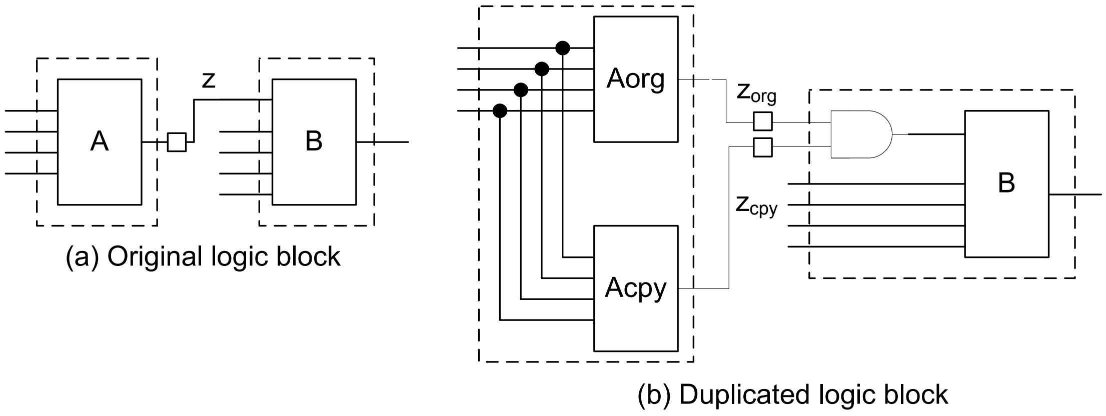
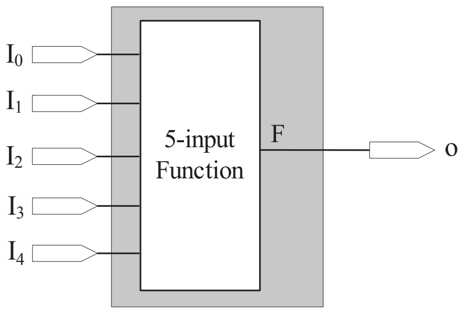
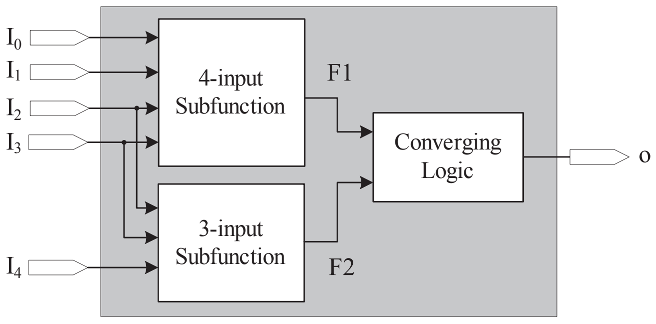


Figure 4.2: LUT duplication (LUT A) and `AND`-encoding of its fanout (LUT B)



(a) Original 5-input Function



(b) Decomposed Function

Figure 4.3: An Example of Decomposition

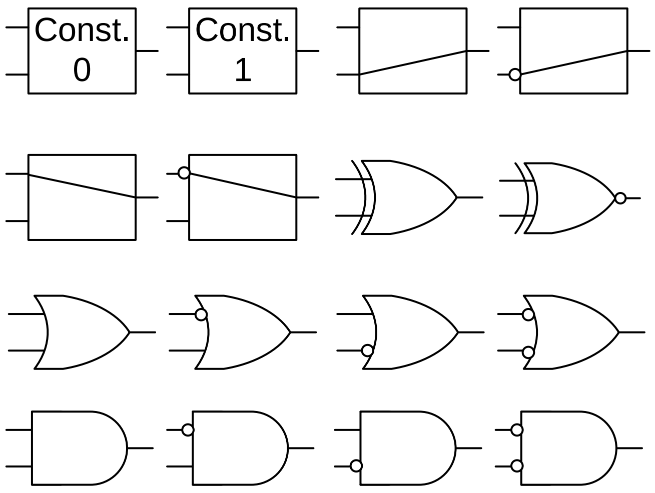


Figure 4.4: 2-input functions.

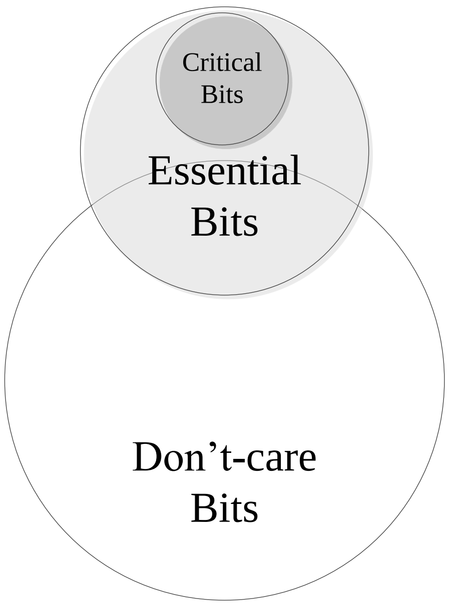


Figure 4.5: CRAM bits categories with respect to soft error impact on functionality.

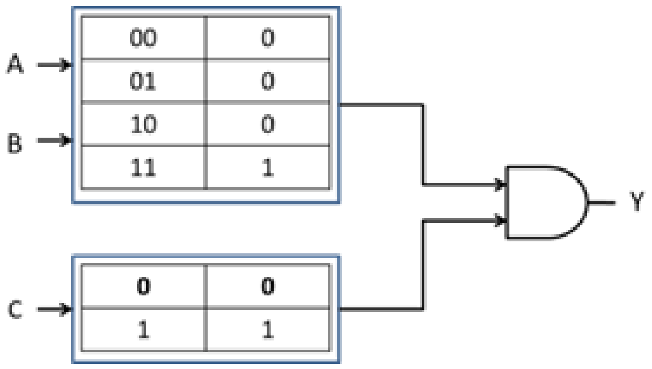
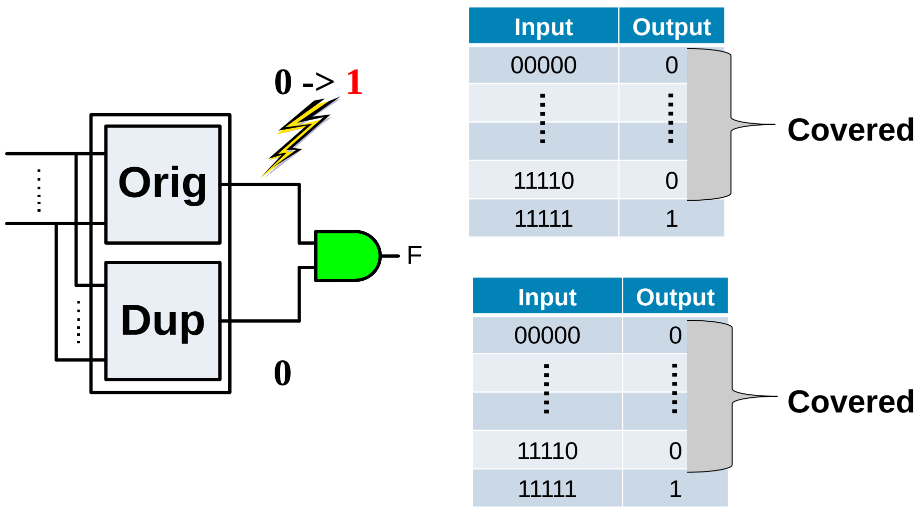


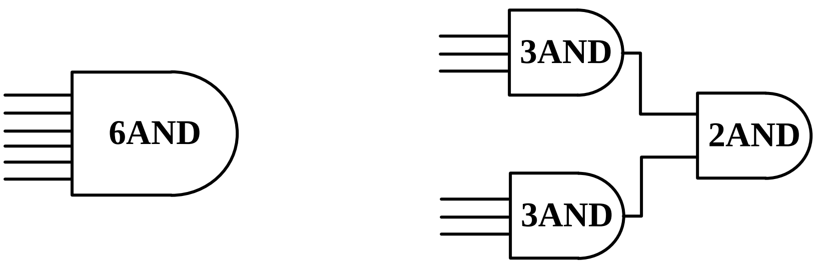
Figure 4.6: Decomposition of $Y = ABC$.



Input	Output
00000	0
⋮	⋮
11110	0
11111	1

Input	Output
00000	0
⋮	⋮
11110	0
11111	1

Figure 4.7: Critical bits reduction by duplication.



(a) before decomposition (b) after decomposition

Figure 4.8: Decomposition to smaller functions for critical bit reduction.

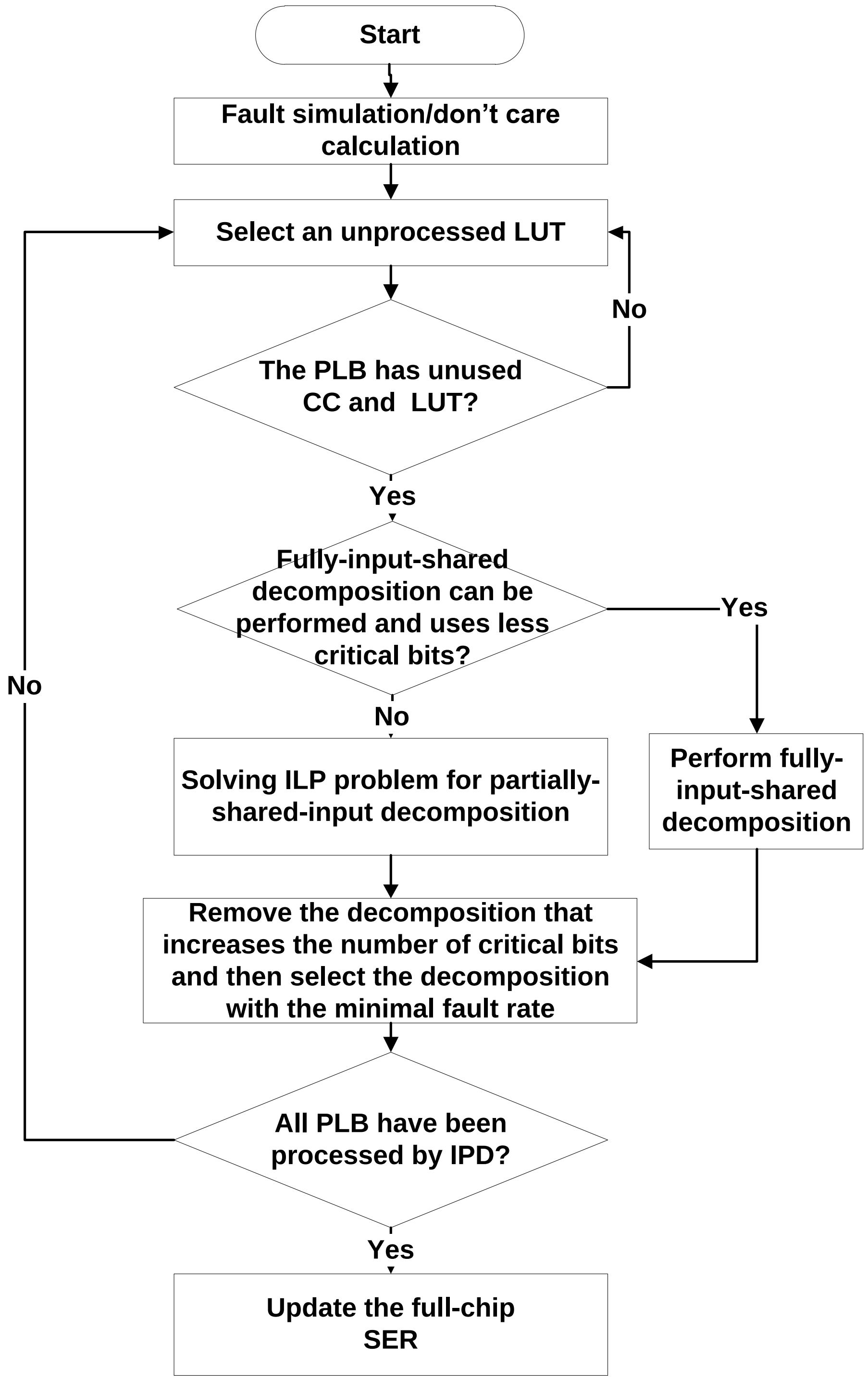

Start
Fault simulation/don't care calculation
Select an unprocessed LUT
No
The PLB has unused CC and LUT?
Yes
Fully-input-shared decomposition can be performed and uses less critical bits?
Yes
No
No
Solving ILP problem for partially-shared-input decomposition
Perform fully-input-shared decomposition
Remove the decomposition that increases the number of critical bits and then select the decomposition with the minimal fault rate
All PLB have been processed by IPD?
Yes
Update the full-chip SER

Figure 4.9: IPD overall algorithm flow.

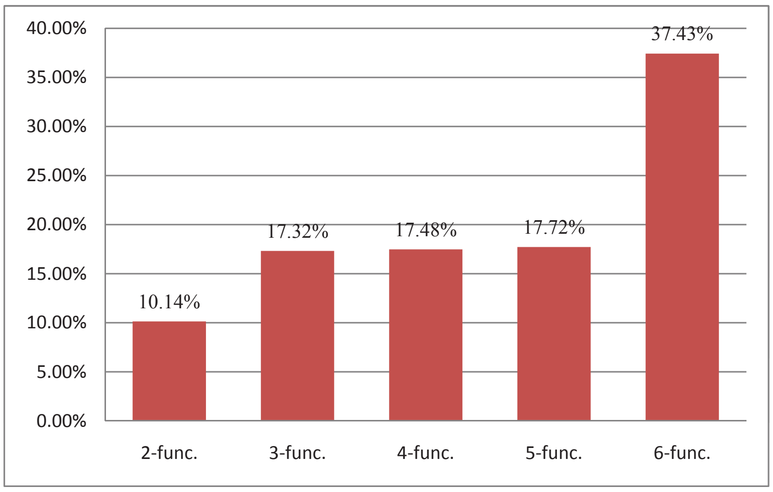


Figure 4.10: Distribution of different sizes of functions in the 10 largest MCNC combinational benchmark circuits.

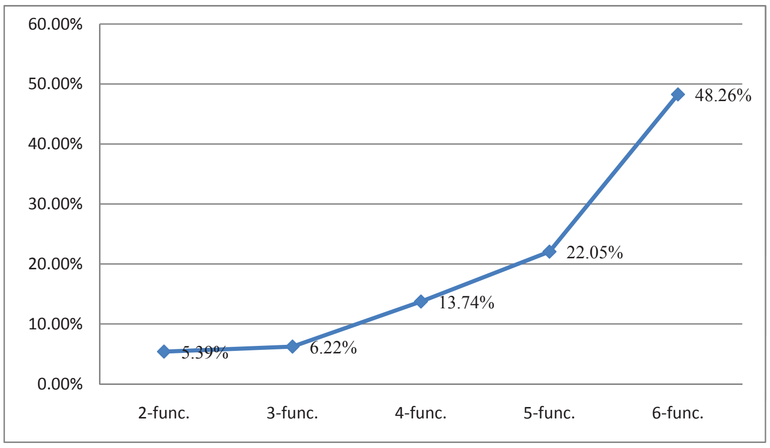


Figure 4.11: SER contribution for different size of functions.

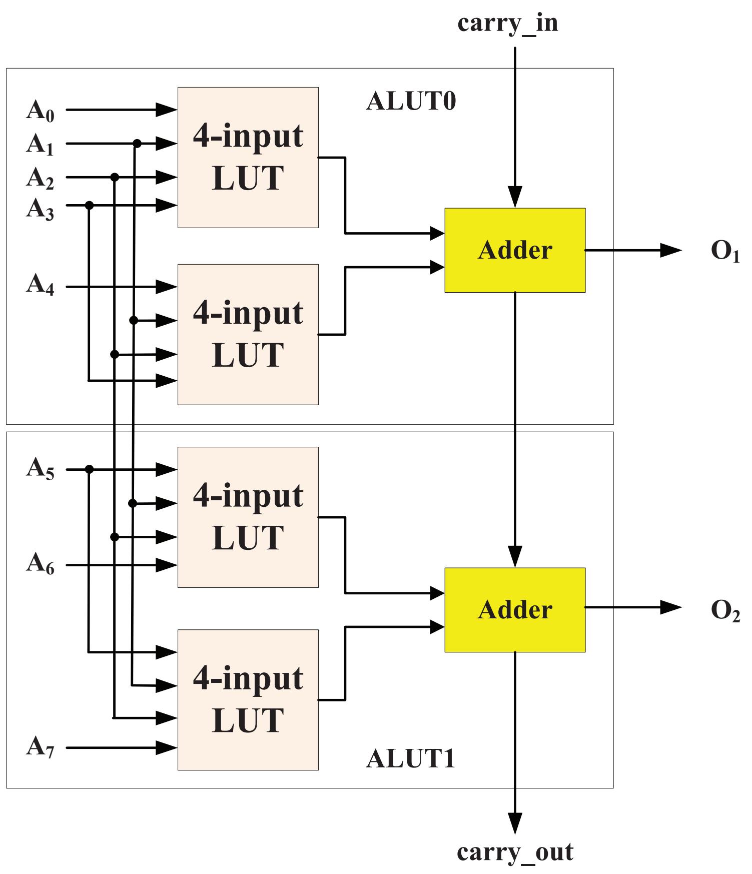


Figure 4.12: Altera Stratix-IV ALM arithmetic mode

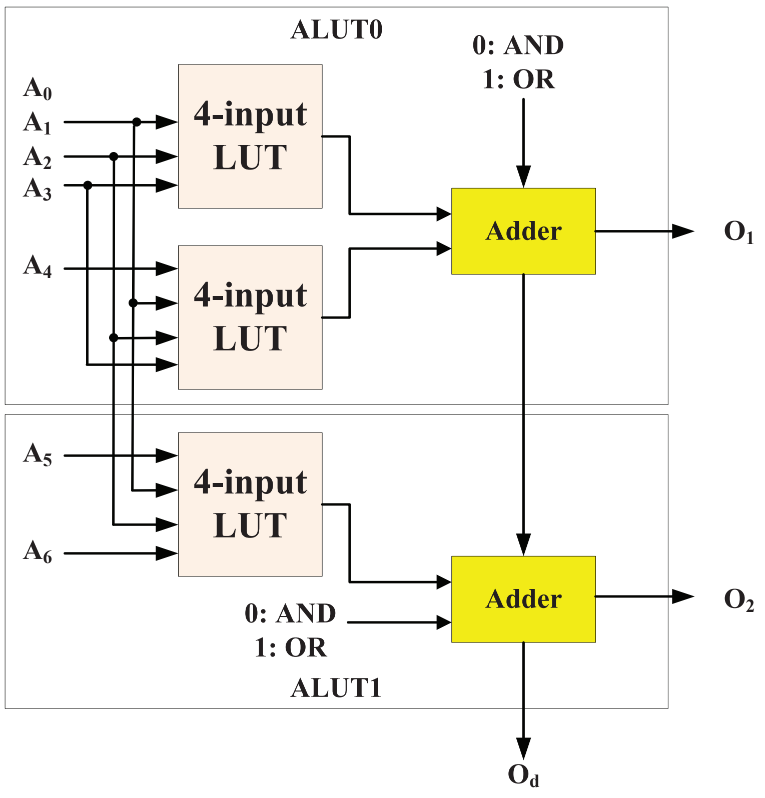
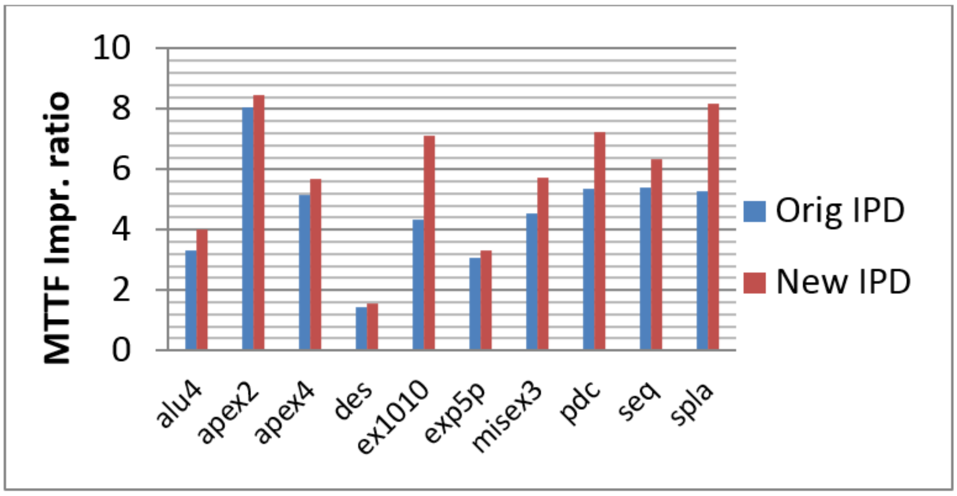
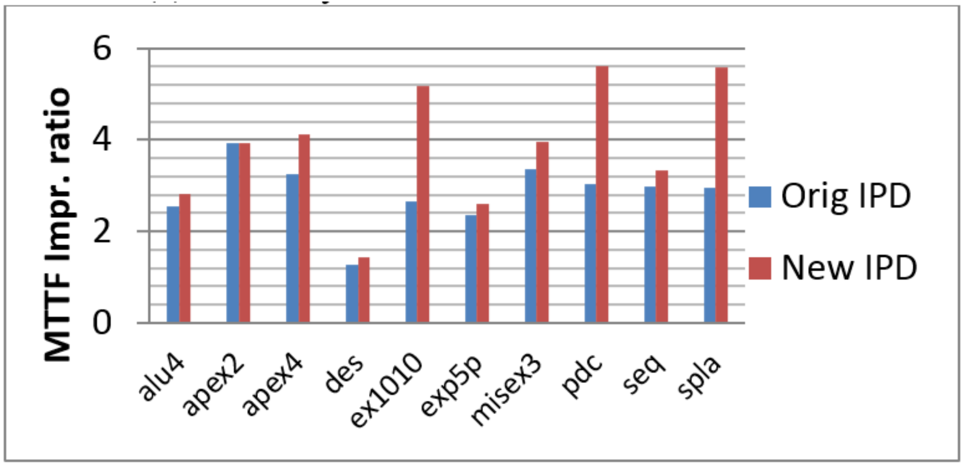


Figure 4.13: Decomposition for 6-input or 7-input function



(a) 0% carry chain/adder utilization rate.



(b) 20% carry chain/adder utilization rate.

Figure 4.14: MTTF improvement ratio comparison of the IPD matching all minterms and IPD with don't-care flexibility.

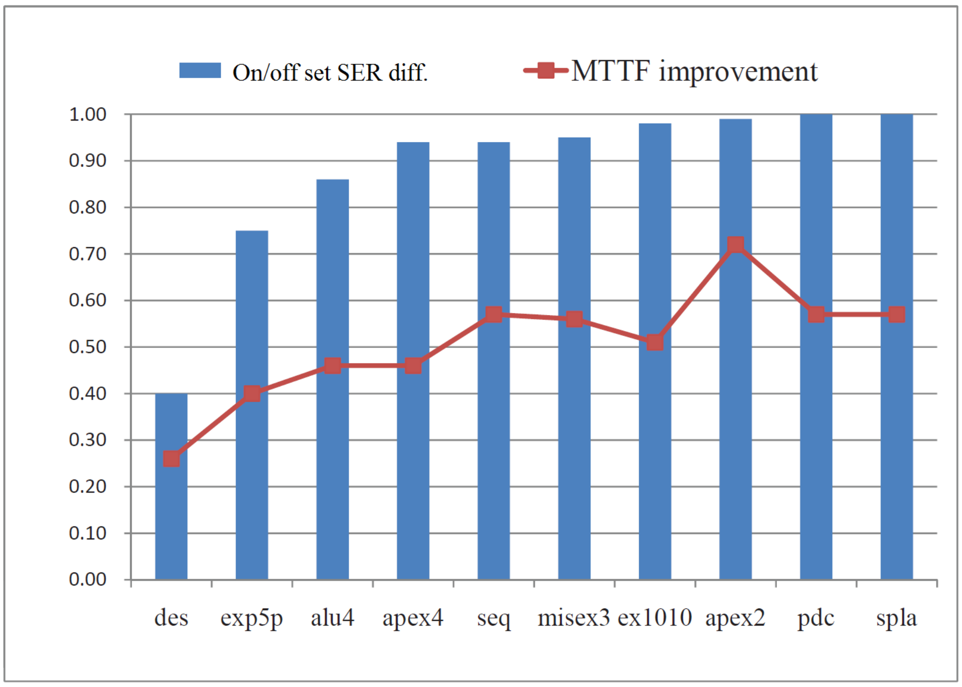


Figure 4.15: IPD improvement indicator.

CHAPTER 5

Mitigating FPGA Interconnect Soft Errors by In-Place LUT Inversion

Modern SRAM-based FPGA use SRAM cells to configure logic and interconnects, and the number of SRAM cells used for configuration can be up to 450 million in a Xilinx Virtex-7 chip. These SRAM cells suffer from soft errors caused by cosmetic radiation or circuit internal noise, and the soft error rate (SER) increases due to technology scaling. To reduce the SER, classic Triple Module Redundancy (TMR) employs circuit redundancy both in LUT and interconnect but with high overhead in area, power and performance. Recent logic re-synthesis techniques, such as ROSE [HFH08], IPR [FHH09], and [JHM10] leverage logic masking to reduce SER in LUTs. However, these techniques do not explicitly consider the interconnect SER and therefore the chip level SER reduction could be limited due to the interconnect dominance in FPGA.

An FPGA architecture is mainly defined by Configuration Logic Blocks (CLBs) and routing architectures as illustrated in Figure 5.1. Interconnects are critical since they contribute a large portion of the total FPGA area and total configuration bits. Modern FPGA routing has shifted from bidirectional routing towards the unidirectional routing architecture, where both the inter-CLB routing (including connection boxes and switch boxes) and the intra-CLB routing typically employ directional MUXes to route the input signals. To select an output out of the signals, each MUX is configured by several encoded CRAM bits, and these bits contribute to the majority of the configuration bits in FPGA. For example, we observe that interconnects contribute to nearly 80% of the CRAM bits for the 10 largest MCNC benchmarks when they are synthesized to the minimum FPGA architectures with 6-input LUTs.

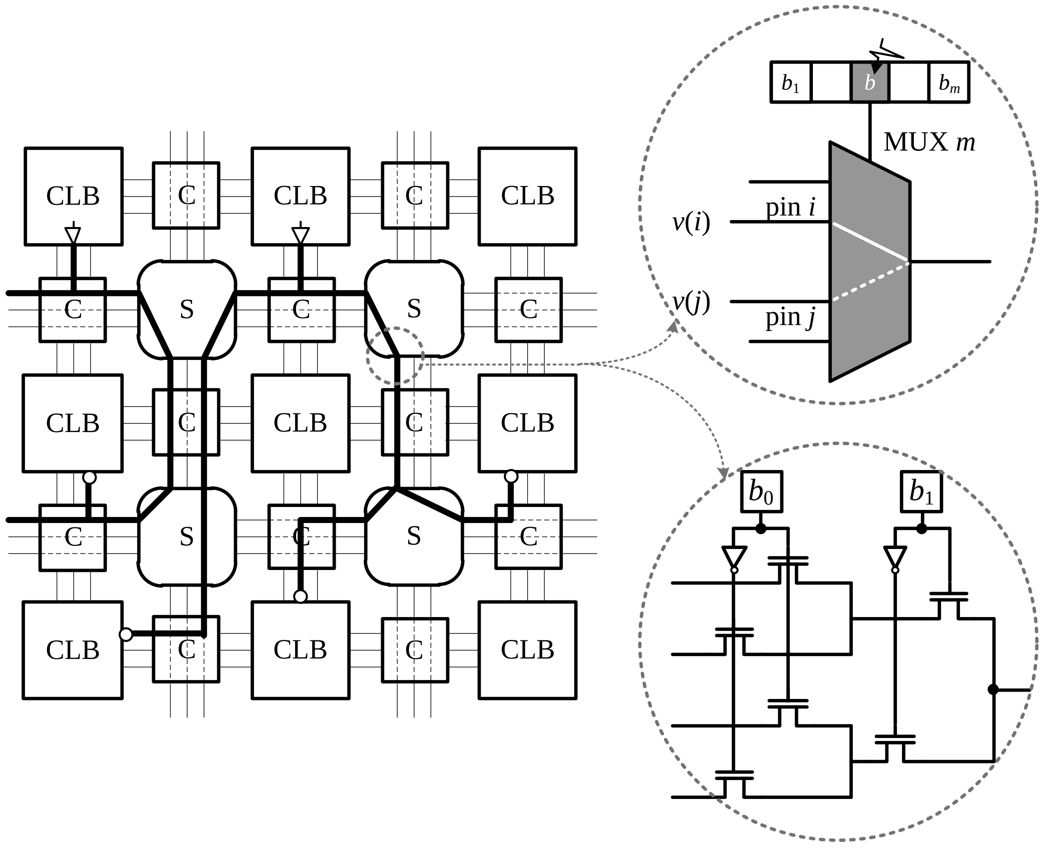


Figure 5.1: FPGA unidirectional routing and multiplxer structure.

As Figure 5.1 illustrates, when one of the interconnect CRAM bits flips its value due to soft error on a MUX, an erroneous input signal (dotted) is then selected. If this erroneous signal from the faulty MUX reaches the primary outputs of the chip, a functional failure at the chip level occurs. Note that this fault mechanism is not a bridging fault as studied for bidirectional routing.

Modern FPGA has shifted towards the multiplexer-based (MUX-based) unidirectional routing architecture [LLT04, SCW09], where the interconnect fault mechanism is different from conventional bidirectional routing as previous studies in [GB07, RCS05]. In this chapter, considering MUX-based unidirectional routing, we propose In-Place inVersion (IPV) of LUT logic polarities to reduce interconnect SER, and reveal a locality and NP-Hardness of IPV problem. We then develop an exact algorithm based on the binary Integer Linear Programming (ILP) and a heuristic based on the Simulated Annealing (SA), both enabled by the locality. We report the averaged results for 10 largest MCNC bench-marks mapped to 4- and 6-input LUTs by ABC [ABC] and then placed and routed by VPR [LKJ11]. From the experiment results, SA obtains the same SER reduction for the circuits that can be solved exactly by the ILP and therefore it is highly effective. Specifically, SA-based IPV obtains nearly 4× reduction and runs 30× faster. Furthermore, combining IPV and IPD leads to 5.3× SER reductions at the chip level. This does not change placement and routing, and thus has no impact on design closure. To the best of our knowledge, it is the first in-depth study on SER reduction for modern MUX-based FPGA routing by the in-placed logic resynthesis.

The chapter is organized as follows. We formulate the IPV problem in section 5.1 and present IPV properties in section 5.2. Then, two IPV algorithms are presented in 5.3. Section 5.4 shows the experimental results and section 5.6 concludes this chapter.

5.1 Problem Formulation

5.1.1 In-Place LUT Polarity Inversion in FPGAs

The logic inversion technique can be briefly illustrated in Figure 5.2, which is a local modification involving two atomic operations: driver logic polarity inversion and its fan-out adjustment. To illustrate the core idea, in a hard-wired circuit, it can be done by inserting several inverters along the polarity-inverted path as shown in the figure.

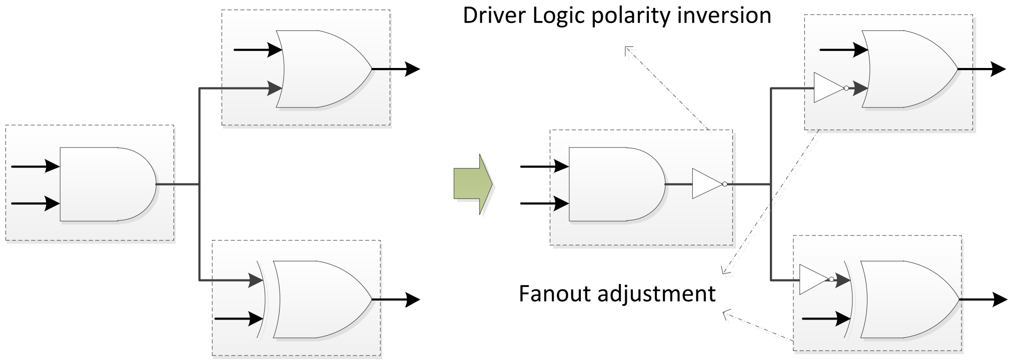


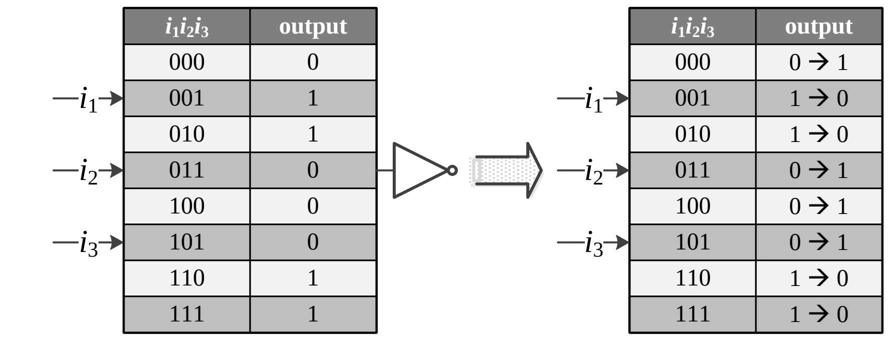
Figure 5.2: Logic polarity inversion.

In the FPGA scenario, where a net is directly driven by an LUT and directly connected to the input pins of the immediate fan-out LUTs, the two polarity inversion operations can be performed by inverting all of the CRAM bits of the driving LUT and encoding input inverters into the direct fan-out LUTs as Figure 5.3 illustrates, without any extra circuit cost.

In Figure 5.3(a), to encode the inverter at the output, we simply invert all of the configuration bits in the LUT truth table. In Figure 5.3(b), to encode the inverter at input i_3 into the LUT, we perform pair-wise LUT configuration bits swapping, where two configuration bits are swapped if their input vectors only differ in i_3, such as “010” and “011”. The LUT polarity inversion above is an in-place mitigating strategy only involving the reassignment on some LUT logic polarities, and thus introduces no overhead. The atomic operation reserves the circuit functionality without any constraints and thus can be applied as flexible as required.

5.1.2 Motivating Example of SER Reduction by LUT Polarity Inversion

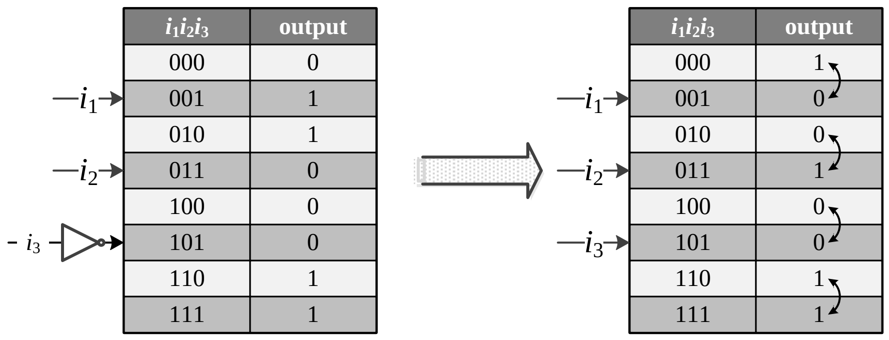
As illustrated in Figure 5.1, when MUX m has one of its CRAM bits b flipped due to SEU, the output is driven by net j instead of the desired net i. If j carries



$i_1i_2i_3$	output
000	0
001	1
010	1
011	0
100	0
101	0
110	1
111	1

$i_1i_2i_3$	output
000	$0 \rightarrow 1$
001	$1 \rightarrow 0$
010	$1 \rightarrow 0$
011	$0 \rightarrow 1$
100	$0 \rightarrow 1$
101	$0 \rightarrow 1$
110	$1 \rightarrow 0$
111	$1 \rightarrow 0$

(a) Driver LUT truth table inversion.



$i_1i_2i_3$	output
000	0
001	1
010	1
011	0
100	0
101	0
110	1
111	1

$i_1i_2i_3$	output
000	1
001	0
010	0
011	1
100	0
101	0
110	1
111	1

(b) Fan-out adjustment.

Figure 5.3: Two atomic operations of in-place LUT polarity inversion in FPGAs.

a different logic value $v(j)$ from $v(i)$, a fault is injected onto the inputs of the immediate fan-outs of m. The fault has a chance to be further propagated to circuit outputs and contribute to SER_b. On the contrary, if $v(j)$ equals to $v(i)$, no fault is injected even if SEU happens. That is, the fault can be instantly masked at m. In addition, SER_b also depends on the observability of MUX m, $obv(m)$, which indicates if the fault can be masked by logic during its propagation to circuit outputs. As a result, SER_b can be given by

$$(v(i) \oplus v(j)) \cdot obv(m) \tag{5.1}$$

Such error masking for MUXes can be used to mitigate soft errors on interconnects. When the logic value $v(j)$ is different from $v(i)$, we can invert $v(j)$ at its driving LUT to force them to be identical. Note that logic polarity can be independently determined on each input and the output of an LUT in FPGA (see one example in Figure 5.4). Similar techniques have been used to optimize timing [Zhu07] and power [AN06]. Here, we use an example in Figure 5.5 to show how logic polarity inversion helps to reduce SER on a routing MUX. In the example, given the observability for the MUX and the two logic values on pin i and pin j, the probability of $v(i) \neq v(j)$ is 0.5 and the $obv(m) = 1$ when $v(i) \neq v(j)$. According to Eq. 5.1, the SER of the CRAM bit b is 0.5. However, if j is inverted, all of the errors from $v(i) \neq v(j)$ are canceled becaused of the inversion of $v(j)$ and the new errors from $v(i) \neq v(\bar{j})$ are masked due to the observability $obv(m)$. Consequently, the SER of b_k can be reduced from 0.5 to 0 by inverting the logic polarity on net j.

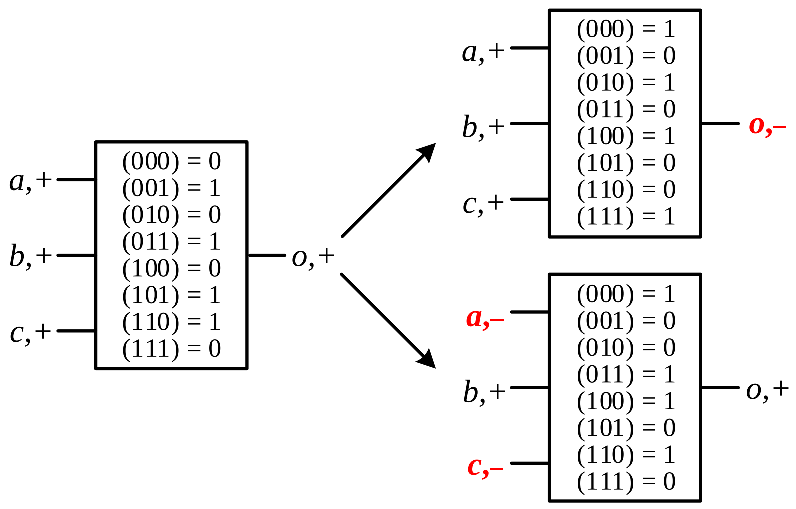
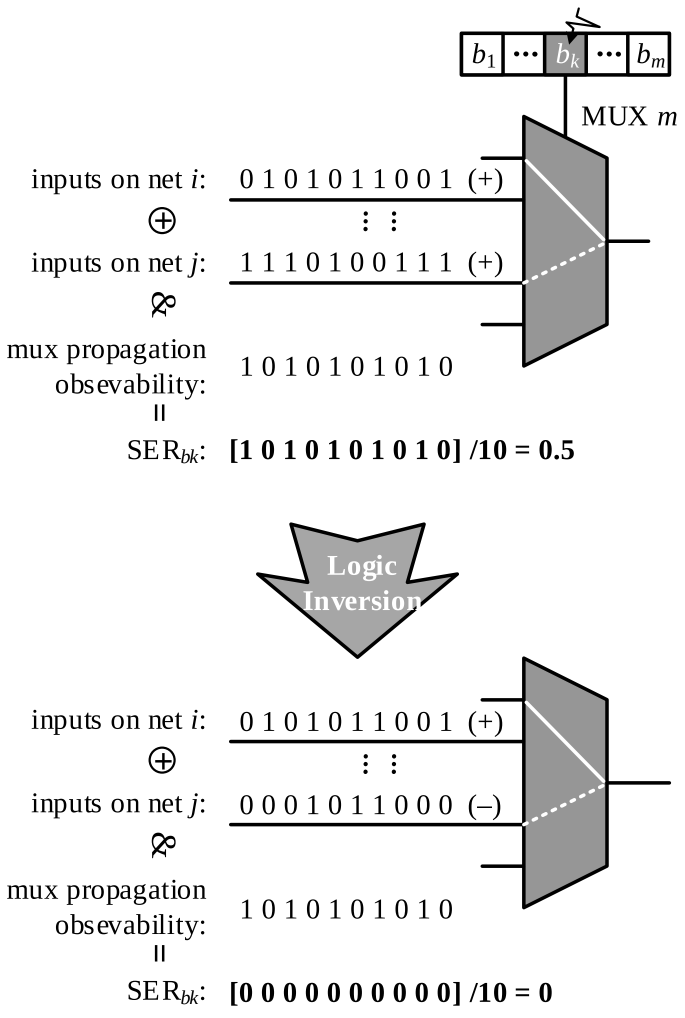


Figure 5.4: Atomic operations for LUT logic polarity inversion.

Based on the SER reduction on MUXes discussed above, we propose to increase the fault masking capabilities on routing MUXes by logic polarity inversion. This may be trivial for a single MUX. However, an inverted LUT output can change the masking capabilities of multiple MUXes that the LUT drives. It is likely that the CRAM bits in different routing multiplexer require the opposite logic polarities of



b_1 ··· b_k ··· b_m
MUX m
inputs on net i: 0 1 0 1 0 1 1 0 0 1 (+)
⊕
inputs on net j: 1 1 1 0 1 0 0 1 1 1 (+)
mux propagation obsevability: 1 0 1 0 1 0 1 0 1 0
=
SER$_{bk}$: [1 0 1 0 1 0 1 0 1 0] /10 = 0.5
Logic Inversion
inputs on net i: 0 1 0 1 0 1 1 0 0 1 (+)
⊕
inputs on net j: 0 0 0 1 0 1 1 0 0 0 (–)
mux propagation obsevability: 1 0 1 0 1 0 1 0 1 0
=
SER$_{bk}$: [0 0 0 0 0 0 0 0 0 0] /10 = 0

Figure 5.5: LUT polarity inversion improves fault tolerance on interconnects.

the same LUT to maximize their masking capability individually. This scenario is illustrated in Figure 5.6, where m_1 may require LUT 2 as negative to locally mitigate the fault, while m_2 may require LUT 2 to stay positive. It is difficult to find an optimal logic polarity assignment for all the LUTs that can minimize the interconnect SER. The optimal inversion problem will be formulated next.

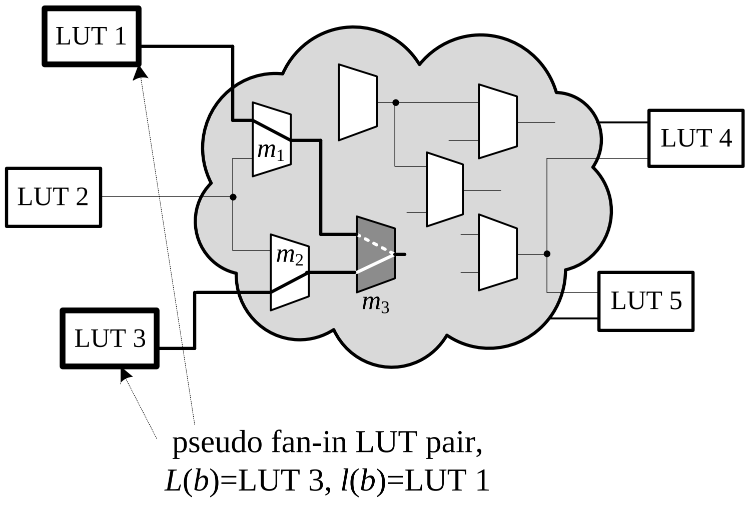


Figure 5.6: The pseudo fan-in pair of a routing CRAM bit subject to SEU.

5.1.3 IPV Problem Formulation

Formulation 3 (In-Place inVersion Problem)**.** *Given a circuit, assign the logic polarity for each LUT, such that the SER for all multiplexer-based interconnects is minimized.*

We provide a bipartite graph representation in Figure 5.7 for a better illustration of our IPV problem, where each node L_i on the bottom represents one of the n LUTs in the circuit, and each node b_k at the top represents one of the m CRAM bits used in the routing MUXes of the placed and routed circuit. An edge between nodes L_i and b_k is generated if the driven signal by L_i is either correctly or wrongly selected by b_k at a certain MUX.

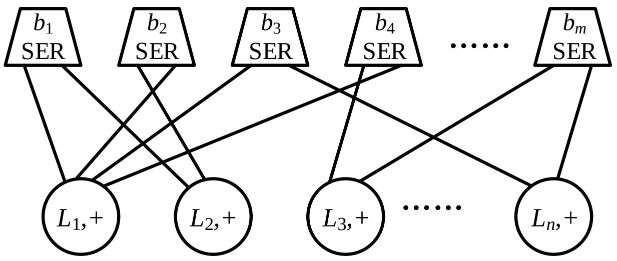


Figure 5.7: IPV problem representation by a bipartite graph.

Given the single fault assumption, each MUX CRAM bit b is connected to two LUT nodes, as illustrated in Figure 5.6. We call the exactly two LUTs as the pseudo fan-in LUT pair and denote them as $L(b)$ and $l(b)$. In this example, for the faulty bit b in m_3, $L(b) =$ LUT 3 that is the desired driving LUT and $l(b) =$ LUT 1 that is the driving LUT selected due to SEU.

Therefore, in the bipartite graph of Figure 5.7, an edge $e(L_i, b_k)$ connects b_k with its pseudo fan-in LUT pair. Each node at the top has exact two incoming edges, but the degree of each node at the bottom depends on the number of MUXes it connects to. As shown in the figure, each top node is annotated with a bit SER value that is associated with the polarities of its pseudo fan-in LUT pair. Thus, our IPV problem reassigns the polarity for each L_i such that the total routing SER can be minimized for all $b_i|(i = 1, \cdots, m)$.

5.2 IPV Properties

In this section, we present the locality property of the pseudo fan-in pairs in the IPV problem, which greatly improves the efficiency for SER update after polarity inversion. In addition, we present and NP-hardness of the IPV problem.

5.2.1 The Locality of Bit SER upon Polarity Inversion

In the IPV problem, when one or multiple LUTs are selected to be inverted for fault masking, the SER_R with inversion, which can be denoted as SER'_R should be updated accordingly after each inverting operation. Intuitively, each update needs an iterative fault simulation of the circuit and it is highly time-consuming. However, we show that SER'_R can be analytically updated by pre-calculating the impact of inversion on SER for each bit, based on the following theorem.

Theorem 1 (Locality of Bit SER upon Polarity Inversion). *Under the single fault assumption, the SER_b for each routing CRAM bit is solely decided by its pseudo* *fan-in LUT pair.*

SER_b is the error rate when neither of its pseudo fan-in pair LUT $L(b)$ nor $l(b)$ is inverted. Although the polarity of an other LUT may be inverted at the same time, by the two atomic operations of polarity inversion, only the value at the net driven by the LUT is affected. Moreover, polarity inversion of a driving LUT of a net does change the functions of other LUTs or the error masking capability on other nets. Therefore, the SER_b does not change regardless of the polarities of other LUTs. It is solely decided by its pseudo fan-in LUT pairs, under the single fault assumption.

5.2.2 Intractability of IPV Problem

As previously mentioned, the IPV problem is difficult since different routing CRAM bits may require opposite polarities of the same LUT in order to enable the fault masking capability locally. Even with the locality of bit SER, we say that

Theorem 2. *IPV problem is NP-Hard.*

Proof. We prove our IPV optimization problem by a reduction from the binary Max-Sum (labeling) problem which is known to be NP-Hard [Wer07]. The binary Max-Sum (labeling) problem is stated as to maximize the sum of unary and binary functions of discrete variables, i.e. as computing

$$maxmize\left[\sum_{t\in T} g_t(x_t) + \sum_{\{t,s\}\in E} g_{t,s}(x_t, x_s)\right] \tag{5.2}$$

where an undirected graph $G(T, E)$, a finite set X and quality values $g_t(x_t)$, $g_{t,s}(x_t, x_s) \in \mathbb{R} \cup -\infty$ are given, as illustrated in Figure 5.8(a). In the problem, each node $t \in T$ is assigned with a label $x_t \in X$ such that the quality or the objective function as defined in Eq. 5.2 is maximized. Note that by converting the quality values to negative, Eq. 5.2 can be turned to a minimization function.

The problem reduction is performed as follows. Given a binary Max-Sum problem instance of an undirected graph $G(T, E)$ with $n/2$ nodes and a finite set X containing two labels $0, 1$, the reduction begins with multiplying the objective function by 1. Then, for each node t in T, we add an extra node t' into T and connect it to t by an extra edge $e'(t, t')$. Accordingly, each binary quality value between t with labeling x_t and its connected extra node t' with either labeling, i.e. $g_{t,t'}(x_t, x_{t'})$ is set equivalent to the unary quality value of $g(x_t)$. That is, $g_{t,t'}(x_t = 0, x_{t'} = 0) = g_{t,t'}(x_t = 0, x_{t'} = 1) = g_t(x_t = 0)$ and so for $x_t = 1$. Thus, a new undirected graph $G(T', E')$ is constructed as shown in Figure 5.8(b), which can be done in polynomial time.

Following we show that the above transformation is a reduction of the problem. Considering the constructed graph G' as above that has n nodes, each pair of nodes connected by an edge is the pseudo fan-in LUTs of a certain routing multiplexer bit. Meanwhile, the four pair-wised quality values between each pair of nodes with a label 0 or 1 for each node can be seen as our SER quadruplet values of $\{SER_{b^{ij}}, ij = 00, 01, 10, 11\}$ that is illustrated in Figure 5.8(c). Then in G', each of the n nodes to be labeled by 1 or 0 is actually one of the n LUTs in circuit whose polarity is to be inverted or not. Since each binary quality value between node t and its connected extra node t' is equivalent to the unary quality value $g(x_t)$, it is clear that

$$\sum_{\{t,s\}\in E} g_{t,s}(x_t, x_s) + \sum_{t\in T} g_t(x_t) = \sum_{\{t,s\}\in E} g_{t,s}(x_t, x_s) + \sum_{\{t,t'\}\in E'-E} g_{t,t'}(x_t, x_{t'}) \quad (5.3)$$

$$= \sum_{\{t,s\}\in E'} g + t, s(x_t, x_s) \quad (5.4)$$

Then, note that the labeling problem has been transformed to a minimization problem, we say that a labeling result $x_t \in X^T$ on G that can maximize the quality of labeling also yields a minimized solution for our IPV problem and vice versa. This completes the proof.

$\square$

5.3 IPV Algorithms

5.3.1 Locality based SER Calculation

Based on the locality theorem (Theorem 1), the SER'_R with inversion can be calculated with a comparable complexity to SER_R. To do this, we extend the SER_b to quadruplicated values of $\{SER_b^{00}, SER_b^{01}, SER_b^{10}, SER_b^{11}\}$, which is called the SER quadruplet of bit b. Each quadruplet provides four error rates indicated by the two superscripted numbers, representing if one of its pseudo fan-in LUTs is inverted or not, as in Eq. 5.5. The SER quadruplet for each CRAM bit b in routing multiplexers is calculated through simulation.

$$SER'_b = \begin{cases} SER_b^{00}(= SER_b) & +L(b)\& + l(b) \\ SER_b^{01} & -L(b)\& + l(b) \\ SER_b^{10} & +L(b)\& - l(b) \\ SER_b^{11} & -L(b)\& - l(b) \end{cases} \tag{5.5}$$

For abbreviation, we denote it as $SER'_b[P_{L(b)}, P_{l(b)}]$, where P is a function representing the polarities of LUTs $L(b)$ and $l(b)$, i.e. + or -. Thus, the total routing SER'_R can be written as

$$SER'_R = \sum_{b \in R} SER'_b[P_{L(b)}, P_{l(b)}] \tag{5.6}$$

Eq. 5.6 reveals that the total routing SER for a given circuit can be updated as the algebraic sum upon each CRAM by its SER quadruplet. In this way, the iterative fault simulation after every reassignment of the LUT polarity can be avoided. The two algorithms proposed in the following for IPV problem are also based on this locality theorem.

5.3.2 Binary ILP Based Algorithm

Eq. 5.6 enables a one-off fault simulation for SER quadruplet such that SER'_R can be updated in a short time after each tentative LUT inversion. In addition, they intuitively imply a binary ILP formulation for our IPV problem. Based on the ILP technique, we can ideally find the globally optimal selection of the inverted LUTs which can provide us an insight on the capability of IPV improvement.

We design a set of binary variables inv_i to denote whether an LUT i is inverted or not, i.e. if $inv_i = 1$, the polarity of LUT i is positive. At the same time, we use an inverting quadruplet $\{f_b^{00}, f_b^{01}, f_b^{10}, f_b^{11}\}$ of binary values to denote the polarities of the pseudo fan-in LUTs for each routing bit b. As a result, the binary ILP formulation for our IPV problem is given by

$$\begin{aligned}
min \quad SER'_R = \textstyle\sum_{b\in R}\sum_{ij=00,01,10,11} SER_b^{ij}\cdot f_b^{ij} \\
s.t. \qquad f_b^{00} \le 1 - inv_s \\
f_b^{00} \le 1 - inv_t \\
f_b^{00} + 1 \ge 1 - inv_s + 1 - inv_t \\
inv_s = L(b), inv_t = l(b)
\end{aligned} \tag{5.7}$$

The set of three constraints in Eq. 5.7 models the fact that exactly one SER value in the quadruplet $\{SER_b^{ij}\}$ should be selected for each bit, by masking with the inverting quadruplet f_b^{ij}. In fact, variable f_b^{ij} is the function over $P_{L(b)}$ and $P_{l(b)}$ to select one SER value out for each bit b with respect to its pseudo fan-in

LUT pair. Other constraints on f_b^{01}, f_b^{10} and f_b^{11} can be similarly written as those in Eq. 5.7.

In our ILP formulation, by forcing corresponding $inv_i = 0$ in the constraints, it also applies to the situation that some LUT input or output polarities are not invertable.

5.3.3 Simulated Annealing based IPV Algorithm

In addition to the ILP approach for the optimal solution, we also develop a Simulated Annealing (SA) based algorithm to improve the efficiency of IPV in runtime. The SA based algorithm starts from the initial circuit with positive logic polarities for all the LUTs. Then, it switches to another LUT polarity assignment at each step by inverting the logic polarity of a LUT. The objective function of the new logic polarity assignment is evaluated by locality based SEU calculation provided in Eq. 5.6, which is efficient and well fit to the SER update in SA.

We follow the standard annealing for the IPV problem. The annealing starts from temperature of 0.008, and is updated by a decreasing factor of 1.003. The annealing always accepts the better LUT polarity assignment in terms of SER, and rejects the worse assignment selectively. The SA algorithm stops till the minimum temperature of 2.0e-6 is reached.

5.3.4 Overall Algorithm Flow

As illustrated in Figure 5.9, our approach that mitigates the SEU fault on FPGA interconnects by fault masking consists of three phases. Starting from the given netlist of a circuit, it first applies logic optimization and technology mapping onto LUTs. The mapped circuit is packed into logic blocks, then placed and routed by physical design tools. Secondly, in order to obtain the bit SER values, we develop an SEU fault analysis framework which starts right after placement and routing. This framework performs logic simulation based on post-layout circuit information to calculate the fault rate for each CRAM bit in routing MUXes in the form of SER quadruplet. This is the basis for both the binary ILP approach and the SA-based algorithm. Finally, after SER quadruplets for all the bits are obtained, we start both the ILP solver and the SA-based algorithm to seek for the reassignment of logic polarities for all the LUTs. The result with maximal reduction on interconnect fault rate is selected, and then back-annotated to the initial circuit by the atomic logic inversion operations to finish our proposed resynthesis flow.

5.4 IPV Experimental Results

5.4.1 Experimental Settings

In our experiments, the ten largest MCNC combinational circuits are used as the test benchmark circuits as in Chapter 4. We leverage the parameterized architecture in VPR [11] to characterize different FPGA architectures to study the architecture impact on the proposed IPV resynthesis algorithm. We first apply logic optimization and technology mapping onto k=4-, 6-input LUTs by Berkeley ABC [ABC]. The mapped circuits are packed by two different CLB architectural settings respectively, i.e. 4-input LUT (k) with a cluster size (N) of 4 and 6-input LUT with a cluster of 8. Then, VPR [LKJ11] is used to implement a minimum dimension for each of the benchmark circuit, which generates an FPGA array as compact as possible without involving extra unused bits that exceeds the actual need of the circuit. After that, we applied the Monte Carlo based fault simulation to generate the bit SER quadruplet values. Note that our IPV algorithm can be applied on any circuit to mitigate interconnect SEU fault as long as the SER values for each CRAM bit are available. Then, the IPV ILP problem is solved optimally by Mosek tool [mos].

Circuit	LUT size k=4, Cluster size n=4						
	# LUT	Dimension		Int. SER reduction		Runtime (s)	
		x,y	w	ILP	SA	ILP	SA
ex5p	622	12,12	32	2.51	2.51	4131.4	35.53
alu4	744	14,14	26	2.04	2.05	36000*	41.34
misex3	773	14,14	26	3.05	3.05	4830.04	44.92
apex4	821	15,15	36	4.79	4.79	2990.69	58.06
apex2	1014	17,17	32	3.91	3.91	584.79	64.75
seq	1084	17,17	32	3.32	3.32	2115.36	78.45
ex1010	1120	17,17	34	7.26	7.26	4132.28	70.36
des	1750	42,42	18	1.16	1.17	36000*	71.58
spla	2229	24,24	48	17.22	17.22	4602.59	183.75
pdc	2304	25,25	46	14.60	14.60	3159.54	206.31
Average	-	-	-	5.99	5.99	-	-

Table 5.1: Interconnect SER Reduction for k=4 and N=4

5.4.2 Comparison between the ILP and SA approaches

The experimental results on the ten benchmark circuits are listed in Table 5.1 and 5.2, which demonstrate the reduction ratios for the interconnect SER before and after applying IPV, from both the ILP and SA approaches. From the table, we can see that our IPV significantly improves the SER on FPGA interconnects with respect to the routing MUXes in the unidirectional routing architecture. For example, for 4-input LUT with a cluster size of 4, (k,N)=(4,4), the interconnect SER can be reduced by 1.2× to 17.2× for the benchmark circuits tested, with an average of around 6×. For 6-input LUT with a cluster size of 8, (k,N)=(6,8), the SER can be reduced by about 5.4× on average.

From Table 5.1 and 5.2, we also observed that in most circuits, the SER reduction in CLB setting of (k,N)=(6,8) is slightly smaller than that in (k,N)=(4,4). The reason is that larger CLB setting can generally reduce the routing network, due to the minimum dimension setting for placement and routing. At the same time, different circuits present completely different error masking capabilities as the reductions are not comparable among different circuits. In these circuits, the

Circuit	LUT size k=6, Cluster size n=8						
	# LUT	Dimension		Int. SER reduction		Runtime (s)	
		x,y	w	ILP	SA	ILP	SA
ex5p	458	7,7	38	1.81	1.90	36000*	25.58
alu4	524	8,8	26	1.96	1.99	36000*	27.20
misex3	530	8,8	28	2.98	2.98	3845.59	29.16
apex4	618	9,9	46	4.91	4.91	5876.77	42.97
apex2	729	10,10	38	3.75	3.75	295.2	51.34
seq	782	10,10	40	3.40	3.40	7284.36	57.91
ex1010	682	10,10	44	7.46	7.46	5899.2	55.80
des	1056	42,42	14	1.07	1.09	36000*	34.45
spla	1524	14,14	60	14.05	14.05	811.53	141.50
pdc	1609	14,14	62	12.51	12.51	5474.11	153.77
Average	-	-	-	5.39	5.40	-	-

Table 5.2: Interconnect SER Reduction for k=6 and N=8

“des” has the lowest improvement ratio. By analyzing the circuit structure, we found the reason that it has the smallest dimension of the routing network but the highest logic density as indicated by x, y and w. At the same time, the values in the SER quadruplets for many bits in “des” are high and close to each other, which limits the design freedom of our IPV approach.

We also listed the runtime for the two approaches in Table 5.1 and 5.2. Note that the time listed does not include the fault simulation time for SER quadruplets, which is relatively small comparing to the time consumed by ILP approach. The ILP approach consumes much longer runtime than its SA counterpart. We also notice that the ILP approach may result in a sub-optimal solution (marked by *), because the solver cannot solve the optimal solution in a limited time. In contrast, the SA based algorithm can find the same optimal results for the circuits that can be exactly solved by the ILP but with much shorter runtime. It also outperforms the ILP approaches in the circuits with sub-optimal results by ILP. Therefore, we show that the SA based algorithm is highly effective, both in solution quality and runtime.

Circuit	IPD+IPF Reduction		IPV+IPF Reduction		IPD+IPV Reduction	
	LUT(%)	Chip(%)	LUT(%)	Chip(%)	LUT(%)	Chip(%)
ex5p	62.00	17.95	7.00	48.44	66.47	48.68
apex4	66.63	15.60	13.29	75.17	78.21	79.49
misex3	75.82	33.02	26.12	68.59	82.07	67.93
alu4	63.81	20.58	22.53	55.36	68.78	51.53
ex1010	59.03	14.59	9.24	80.51	76.57	85.84
apex2	85.26	15.76	12.52	70.41	85.69	74.68
seq	79.31	25.59	19.26	69.47	82.51	71.67
des	31.87	7.80	10.13	10.83	28.85	9.26
spla	70.94	20.49	14.48	87.69	85.80	92.44
pdc	70.64	24.90	13.05	86.51	88.41	91.79
Average	66.53	19.63	14.76	65.30	74.34	67.33

Table 5.3: SER Reduction by combined algorithms of IPF+IPD, IPF+IPV, and IPD+IPV.

5.5 Combined IPD + IPF + IPV Experimental Results

Finally, we evaluate the improvement by combining different in-place resynthesis algorithms for robustness. In this section, consider IPD, IPV, and also IPF [FNH11], a recently published in-place technique. As resynthesis-based techniques, the three algorithms are performed within LUTs after placement and routing and preserve the circuit functionality without invoking physical resyntheses.

Then, several combinations of the three algorithms are evaluated to investigate the interactions between them to further boost their fault mitigation capability. We evaluated the combinations of IPD+IPF, IPV+IPF, IPD+IPV, and IPD+IPV+IPF, where the algorithms are applied on the circuit as indicated by their order, and the results are shown in Table 5.3.

For IPD+IPF, results show that the two algorithms are not orthogonal. First, the interaction between them degrades the SER reduction on LUTs more than with an individual IPD, from about 74% down to 67% on average. This is because IPF may reduce the on/off set SER difference on LUTs, which is an indicator

Circuit	IPD+IPV+IPF Reduction	
	LUT(%)	Chip(%)
ex5p	62.00	52.55
apex4	66.63	79.79
misex3	75.82	74.12
alu4	63.81	59.87
ex1010	59.03	86.37
apex2	85.26	78.29
seq	79.31	76.78
des	31.87	12.49
spla	70.94	92.71
pdc	70.64	92.38
Average	66.53	70.53

Table 5.4: SER Reduction by IPD+IPV+IPF

provided of IPD discussed in Section 4.4. The "on" (resp. "off") set is the CRAM bit set with logic "1" (resp. "0"). In general, a higher on/off set SER difference indicates more potential improvement that IPD can provide. We further plotted the on/off set SER differences for the ten circuits in Figure 5.10, where most of the differences are reduced after applying IPF, and thus the SER reductions by IPD+IPF are degraded. Second, compared with individual IPF, an extra SER reduction of several percent (about 5% on average) is observed on the chip level after applying IPD on IPF. This is due to IPD further reducing the errors on LUTs, which in turn improves chip reliability. Third, both algorithms present limited improvement on the chip level (less than 20% on average), since neither of them considers interconnect fault explicitly. This experiment reveals that interconnect is more important in fault mitigation, and in order to develop more advanced fault mitigation techniques in the future, the errors on LUTs and interconnects should be tuned together to improve the circuit fault tolerance to the greatest extent.

For the combined algorithms of IPV+IPF and IPD+IPV, results indicate that the SER on the chip level can be reduced respectively by around 65% and 67% on average, which means reductions of 2.88× and 3.06× can be achieved. This is due to IPV explicitly considering soft errors on interconnect. In addition, the IPV algorithm is completely orthogonal with IPD. That is, the SER reduction of LUT comes from IPD, while reduction of interconnect comes from IPV, because there is no interaction or coupling between the two algorithms. For the combination of IPV+IPF, IPV keeps the LUT SER reduction by IPF, while IPF helps to reduce the interconnect errors by another 4% on average (from 61.3% to 65.3%). This slight improvement is due to the implicit fault reduction by IPF, as previously explained.

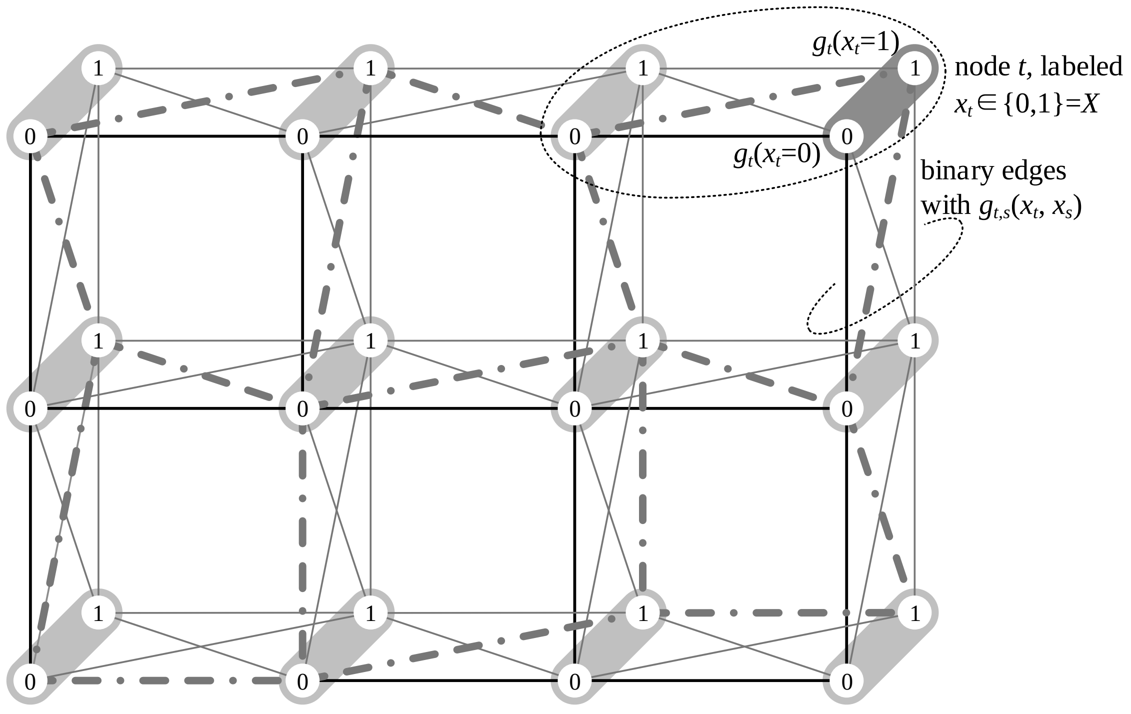
For the combination of IPD+IPV+IPF, the experiment results also confirm our understanding of the three algorithms. That is, (1) since IPV has no improvement on LUT fault, the SER reduction on LUT is the same as that of IPD+IPF; (2) since the interconnect fault is explicitly considered by IPV, the SER reduction on the chip level is higher than IPD+IPF; (3) as IPF implicitly helps to reduce interconnect fault in fan-in cones for each LUT, the SER reduction is higher than IPD+IPV in all cases. Although this combination covers soft error mitigation both on LUTs and interconnects, their improvements are not orthogonal, because the current combination simply neglects the interaction between them, for example, IPD with IPF, IPV with IPF, which overlaps in optimization. This reveals that in order to improve the fault tolerance on the chip level, future mitigation algorithms should be concerned not only for the soft errors in LUTs and interconnects, but also their interactions.

5.6 Summary

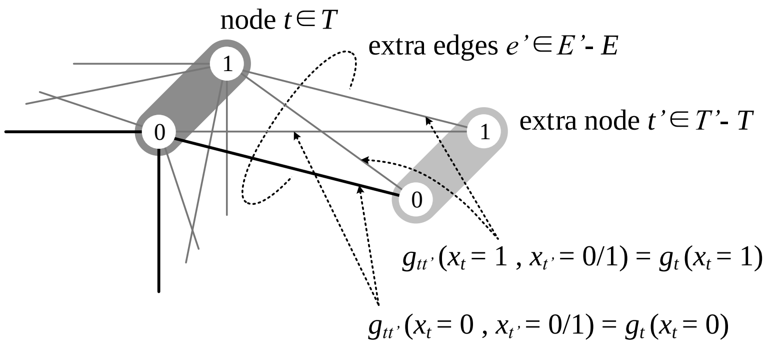
This chapter proposed an effective yet efficient approach to mitigate the SEU fault on FPGA interconnects. By targeting the routing multiplexers that are the dominant routing elements in the modern unidirectional routing architecture, we took the feature of fault masking to reduce the interconnect fault to the greatest extent, which involves nearly no cost in the already placed and routed FPGA applications. Further, we formulated the problem and proved it to be NP-Hard. Then, based on the locality of the IPV problem, we developed two approaches, a binary ILP technique to look for a globally optimized solution and a SA-based search for fast result.

As demonstrated in the experiments, our approach can significantly reduces the interconnect fault in terms of SER ranging from 1.2× to 17.2×, with an averaged 6× for 4-input LUT with a cluster size of 4 and about 5.4× for 6-input LUT with a cluster size of 8. At the same time, the fault on circuit level can be greatly reduced as well, nearly 4× on average. The SA based algorithm outperforms the ILP both in solution optimality and runtime. Comparing with the previous IPD algorithm mitigating the fault on LUTs, our IPV technique enables shorter runtime but higher improvement at chip level. In addition, we observed that combining the IPD and IPV can lead to a SER reduction up to 5.3× on chip level.

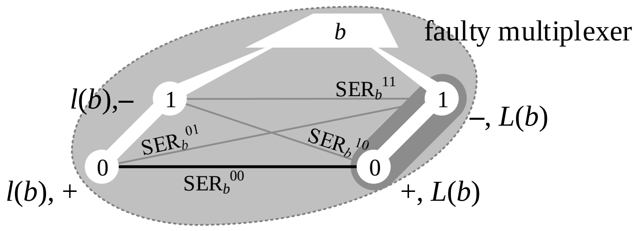
It is important to note that, our approach can be in-placed performed on the placed and routed circuit, and thus involves nearly no cost in the standard FPGA design flow. In the future, we will improve the speed of ILP approach and the optimality of SA algorithm. At the same time, we will apply our approach to sequential circuits by calculating bit SER for each CRAM bit after identifying the sequential feedbacks in the circuit. Besides, we will further extend our IPV approach to mitigating the fault simultaneously on LUT and interconnect by considering their inherent interactions, such that the SEU fault in FPGA can be mitigated to the great extend.


$g_t(x_t{=}1)$
node t, labeled $x_t \in \{0,1\}=X$
$g_t(x_t{=}0)$
binary edges with $g_{t,s}(x_t, x_s)$

(a)


node $t \in T$
extra edges $e' \in E'$- E
extra node $t' \in T'$- T
$g_{tt'}(x_t = 1 , x_{t'} = 0/1) = g_t(x_t = 1)$
$g_{tt'}(x_t = 0 , x_{t'} = 0/1) = g_t(x_t = 0)$

(b)



(c)

Figure 5.8: Illustrations of the binary Max-Sum labeling problem (a) in the context of our fault impact minimization problem (c) with the reduction in (b).

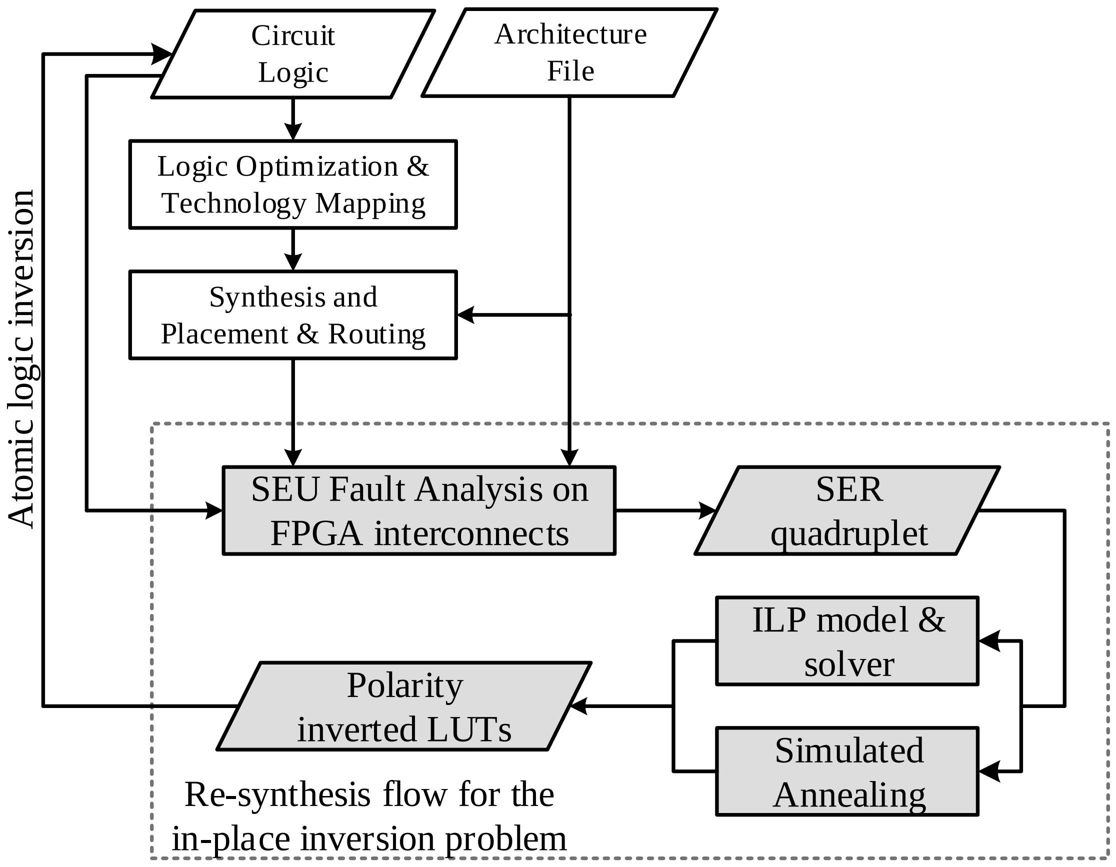

Circuit Logic
Architecture File
Logic Optimization & Technology Mapping
Synthesis and Placement & Routing
SEU Fault Analysis on FPGA interconnects
SER quadruplet
ILP model & solver
Simulated Annealing
Polarity inverted LUTs
Re-synthesis flow for the in-place inversion problem
Atomic logic inversion

Figure 5.9: The overall flow of IPV resynthesis algorithm.

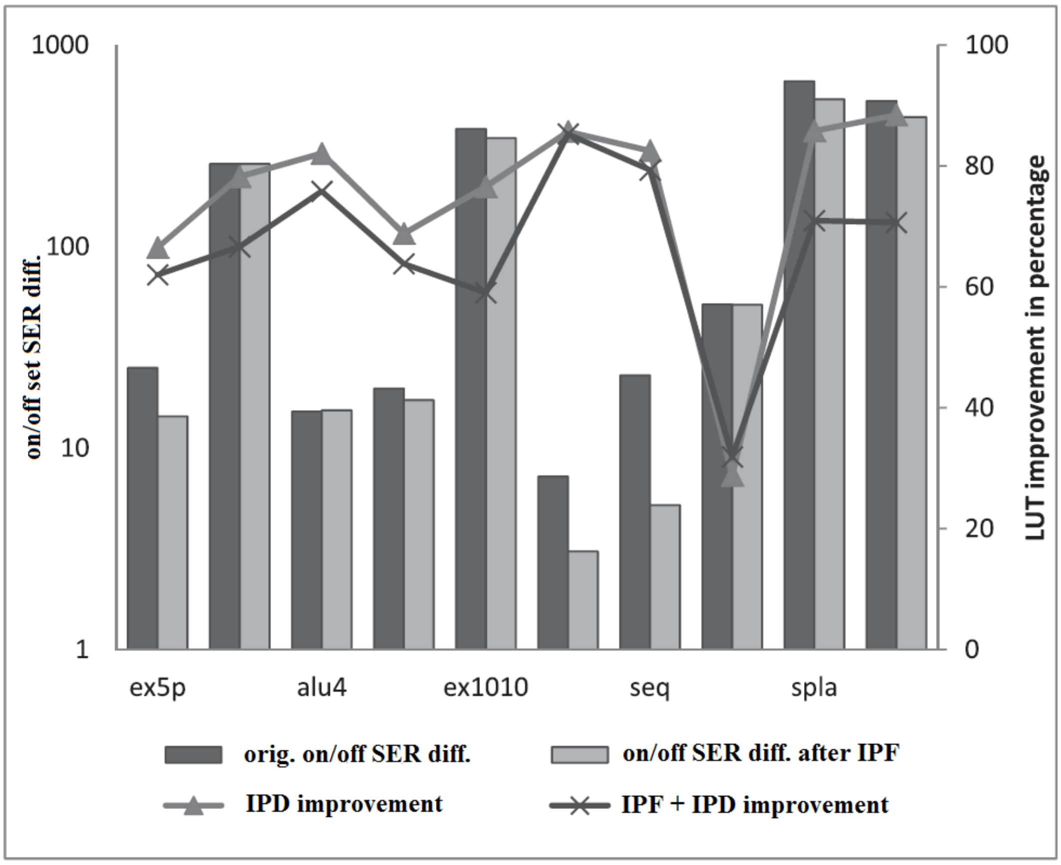


Figure 5.10: The on/off set criticality differences versus LUT criticality improvements by IPD and IPD+IPF.

CHAPTER 6

Soft Error and Leakage Mitigations for FPGAs using In-Place LUT Polarity Inversion

FPGA designs are suffering from an increasing leakage power dissipation. There are two main sources of the leakage power [AN06, HHL12], i.e. sub-threshold leakage and gate leakage. The sub-threshold leakage comes from a non-zero current between the source and drain of an off-state CMOS transistor. With shrinking feature size and smaller supply voltages, threshold voltage has to be lowered to mitigate the performance degradation but aggravate the leakage problem. The gate leakage comes from tunneling current through the gate oxide of a transistor, which is increasing because gate oxides are thinned to improve transistor drive strength in modern manufacturing process. It is observed that over 40% total power could be leakage, as reported by the previous studies in [KNC02].

Interestingly, in FPGA structures, routing resources have been demonstrated as the most vulnerable resources to SEU, as well as the dominant contributor of the total FPGA power. More than 90% of SEU induced soft errors are from routing as reported in Section 2.3, and over 60% of the leakage are dissipated on routing [RP04, TL03]. A common reason for this is that routing are the most dominant resource and occupies the largest area in FPGAs. The large routing resources lead to the most configuration bits in a modern SRAM-based FPGA, typically millions of bits, which naturally become the focus for both SEU mitigation and leakage reduction techniques.

In this chapter, we propose a joint optimization for the soft error and leakage on the VPR academic FPGA architecture by the in-place LUT polarity inversion (IPV) technique. We formulate the soft error mitigation and leakage reduction problems, and reveal that the two problems have similar structures but may be contradictive in their objectives. Then, we propose a Simulated Annealing (SA) based joint optimization approach for the two problems at the same time. For the 10 largest combinational MCNC benchmark circuits, our experimental results show that the proposed co-optimization algorithm reduces soft errors and leakage by 50% and 33%, respectively.

The remainder of this chapter is organized as follows. Section 6.1 reviews the background and the previous works on leakage mitigation. Section 6.2 formulates the optimization problems for SEU and leakage, and proposes simulated annealing based co-optimization algorithm. Section 6.3 shows the experimental results and Section 6.4 concludes this chapter.

6.1 Background and Motivation

6.1.1 Previous Works on Leakage Reduction in FPGAs

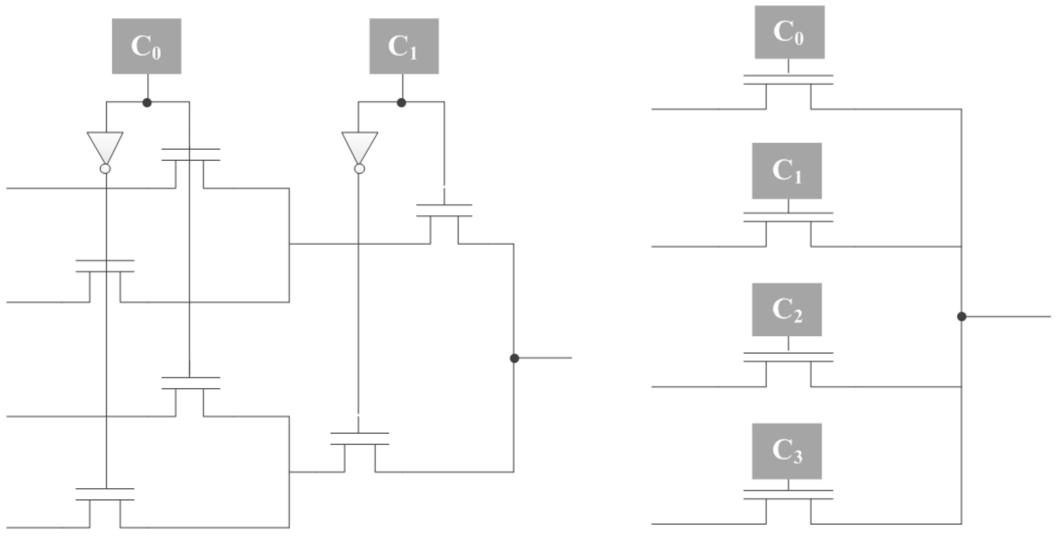
Leakage reduction in FPGAs has been addressed in the literature targeting on different leakage sources. For example, to reduce the leakage from the active paths, multiple or dynamic voltage adjustment techniques are used in [KR02, MFM02]. High supply voltage is applied to resources on critical paths for high performance, while lower supply voltage is applied on non-critical paths for low power, and gated voltage is applied to the sleep resources to save power. For standby leakage power, high threshold sleep transistors are used because high threshold voltage effectively reduce the leakage power in [WCJ98, KR02, CLC04]. In [AME02, Sak02], sleep transistors are on when circuit is active and are turned off when the circuit is in standby mode. However, these methods are subject to the tradeoff between performance and power, and FPGA architectural changes are required which also introduces additional cost.

Apart from the voltage adjustment, other approaches to leakage reduction in FPGAs include considering the dynamic leakage characteristics due to internal logic states at runtime. For example, in [Gat, HN97], specific input vectors are identified and applied onto inputs to minimize leakage power when circuit is in standby mode. To reduce leakage from MUXes in FPGA interconnect, the work in [AN06] proposes to reassign logic polarity and invert configuration bits in LUTs based on signal static probability. In addition, off-path leakage optimization is also proposed in [Zhu07], which further considers the off-path leakage due to different input vectors and performs optimization during routing stage. These works apply similar techniques to the SER reduction technique IPV as proposed in Chapter 5, which directly motivates us to find SER and leakage co-optimization solutions for FPGAs.

6.1.2 IPV for FPGA Leakage Reduction

Because an FPGA structure is pre-fabricated before a design is implemented, those circuit resources not utilized in the design will consumes unnecessary leakage power. As mentioned in the introduction, the primary source of leakage power consumption comes from FPGA routing fabrics. In modern FPGA architecture, interconnections between logic elements are typically constructed by MUXes as depicted in Figure 6.1, where (a) shows an encoded MUX style and (b) shows the decoded (non-encoded) style.

In fact, the leakage strongly depends on the two signals that are connected to the drain and source of a NMOS pass transistor in a MUX. As reported in [HHL12], the leakage of the on-path transistor (the connected pass transistor to route the selected input signal of the MUX as in Figure 6.2) is negligible compared to those transistors in the off-paths (the disconnected pass transistors as configured in the



(a) encoded (b) decoded

Figure 6.1: 4-to-1 multiplexor structure.

design). Specifically, there are two cases when significant leakage will occur, that is, when $I2$ carries a different signal, or $I2$ is not connected to any signal, i.e. open connection.

To mitigate the leakage of routing MUXes, [AN06] proposed a LUT polarity assignment method based on static probability, where signals with static probability less than 0.5 are inverted if possible. For the case that $I1$ or $I2$ is open, the larger the static probability of the other signal is, the less leakage is dissipated. This is due to the fact that FPGAs are usually designed to pull up the unconnected nets to logic '1' by specially designed circuitry, such as half-latch [GCJ03]. Consequently, assume that $I2$ is open and is pulled up to logic '1', higher static probability of signal $I1$ leads to lower possibility that a voltage different exists between source and drain of the off-path transistor. However, when $I1$ and $I2$ are two different signals, static probability method is less effective and may increase leakage for some cases. For example, assuming that $I1$ and $I2$ are connected to the signals as in Figure 6.3(a) and 6.3(b) shows, where the static probabilities of $I1$ and $I2$ are 0.75 and 0.4, respectively. Because the static probability of $I2$ is

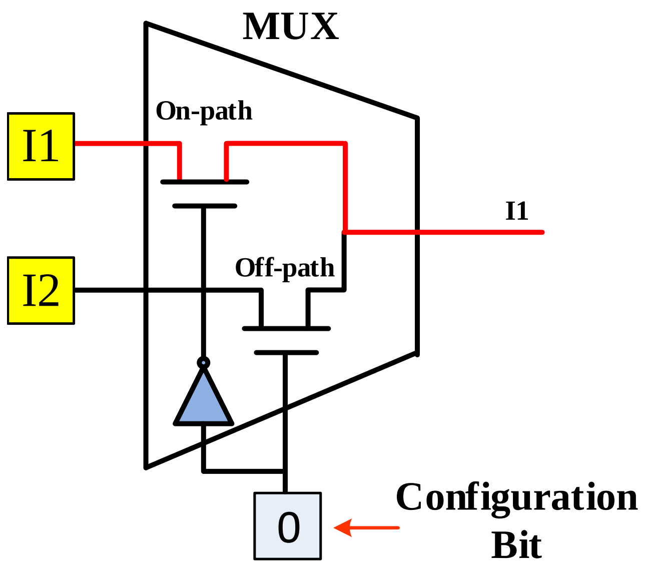


Figure 6.2: Off-path leakage in a decoded multiplexor structure.

less than 0.4, it is inverted as suggested by [AN06]. However, as illustrated in Figure 6.3(c), the leakage is increased after the signal probability method.

In fact, the static signal probability method is more effective when the pin utilization rate of MUXes are low because in this case, more constant logic '1' signal can be found due to the unconnected MUX input pins. On the other hand, this indicates that when a large design is mapped, the signal probability method is less effective.

6.2 Co-optimization of SER and Leakage in FPGAs

In this section, we first present the formulations of IPV for SER and leakage optimization, and show that the two objectives can be contradictory, which makes it difficult to optimize both criteria at the same time. Then, we propose an efficient SA based co-optimization algorithm for SER and leakage.

6.2.1 Formulation of SER and leakage co-optimization problem

As discussed in the previous section, the leakage power primarily arises when the logic value of the selected signal is different from that of the unselected signal on an off-path transistor. Therefore, the leakage of an off-path transistor t can be calculated by

$$Leakage_t = P_{ij}(01) \times L_{01} + P_{ij}(10) \times L_{10} \tag{6.1}$$

where i is the selected signal of the MUX, j is the input signal of t, $P_{ij}(xy)$ is the probability that signal i is x and signal j is y, and L_{xy} is the corresponding leakage. Note that when the off-path transistor has the same logic value at the input and the output, the leakage can be ignored because the leakage is minimal. Therefore, given Eq. 6.1, the total off-path leakage can be calculated by the summation of all of the off-path transistors as follows.

$$Total\_leakage_{off-path} = \sum_{t \in M_{active}} Leakage_t \tag{6.2}$$

where M_{active} is the set of all active MUXes of the implemented design on the FPGA.

SEU induced SER on a MUX can be estimated using similar method as leakage, but with the addition of the observability of the mux. Because SER is the probability that a soft error is generated and propagated to the circuit primary outputs, the SER of a MUX CRAM bit can be estimated by the probability that the corresponding pseudo fan-in signal pair (i,j) have different logic values, D_{ij}, multiplied by the observability, O_b, from the MUX to the circuit primary outputs, given as follows.

$$SER_b = D_{ij} \times O_b \tag{6.3}$$

Similarly, the total SER can be calculated as follows.

$$Total\_SER_{mux} = \frac{\sum_{b \in C_{mux}} SER_b}{|C_{mux}|} \tag{6.4}$$

where C is all of the CRAM bits controlling those active MUXes used by the design.

In fact, D_{ij} can be calculated by $P_{ij}(01) + P_{ij}(10)$. The existing methods proposed in [AN06, HHL12] generally reduce the overall hamming distance between signals, and thus reduce the overall leakage. However, due to the complicated dynamic characteristics of the observability, optimization for leakage and SER may contradict to each other, resulting in degraded optimization quality. There are two factors contributing to the contradiction. First, the sets of signal pairs for SER and leakage differ. The signal pairs for SER are those pairs controlled by CRAM bits, while the pairs for leakage depend on off-path transistors. Second, observability has a wide impact on SER. Particularly, observability is usually the dominant factor of SER, depending on the circuit structure. Therefore, polarity assignment becomes a complicated and difficult problem when both objectives are involved.

6.2.2 Simulated Annealing based Co-Optimization for Leakage and SER

From the discussion above, it is clearly that optimization for leakage and SER separately do not make joint benefits. As a result, the reduction of the leakage power and SER need to be considered together during optimization. In this section, we propose a simulated annealing based optimization algorithm with linear combination of leakage and SER objectives as the cost function as follows.

$$\frac{Total\_SER_{mux}}{Total\_SER_{mux}\_norm} + \frac{Total\_leakage_{off-path}}{Total\_leakage_{off-path}\_norm} \tag{6.5}$$

Because the metrics of SER and leakage are in different magnitudes, they are normalized by $Total\_SER_{mux}\_norm$ and $Total\_leakage_{off-path}\_norm$ in the combined objective function, where $Total\_SER_{mux}\_norm$ and $Total\_leakage_{off-path}\_norm$ are typical SER and leakage values from the experiments.

The solution space of the SA based co-optimization algorithm is the logic polarity assignment for all the LUTs in the circuit. Starting from the initial circuit with positive logic polarities for all the LUTs, the new solution can be generated by switching a random LUT to its opposite logic polarity at each move. The objective function of the new assignment is evaluated after each move, by the co-optimization cost function in Eq. 6.5.

The update on $Total\_leakage_{off-path}$ after an LUT is inverted can be efficiently calculated in the following fashion. When an LUT is inverted, the leakage power of the off-path transistors in its driven MUX are updated accordingly, with the leakage primitive values provided in previous studies, e.g. [AN06]. Given a transistor t of an active MUX with signal pari i and j, as presented in Eq. 6.1, suppose that i is inverted, $P_{\ ij}(xy)$ equals to $P_{ij}(\ xy)$. Therefore, the leakage can be updated by

$$\begin{aligned} Leakage_t = & P_{\ ij}(01) \times L_{01} + P_{\ ij}(10) \times L_{10} && (6.6)\\ & P_{ij}(11) \times L_{01} + P_{ij}(00) \times L_{10} && (6.7) \end{aligned}$$

As the observability of MUX is not involved here, the $Total\_leakage_{off-path}$ can be directly calculated by the linear combination of the leakage primitives within minimum runtime.

On the other hand, the update of SER is slightly complicated due to observability. However, leveraging the precalculated SER quadruplet proposed in Section 5.3, $Total\_SER_{mux}$ can be quickly updated similar to leakage. Therefore, the objective of the co-optimization problem (Eq. 6.5) can updated efficiently with LUT polarity inversion, which significantly reduces the runtime of the SA based algorithm.

Starting from the given netlist of a circuit, the overall co-optimization algorithm flow is similar to IPV for SER reduction presented in Section 5.3. The algorithm flow first applies logic optimization and technology mapping onto LUTs. The mapped circuit is packed into logic blocks, then placed and routed by physical design tools. Then, based on the efficient SER and leakage update method proposed above, we start the SA-based algorithm to determine the reassignment of logic polarities for all the LUTs. The result that yields the best objective value is selected, and then back-annotated to the initial circuit by the atomic logic inversion operations to complete our proposed re-synthesis flow.

6.3 Experimental results for SER and leakage co-optimization

We experimentally verify the proposed method by using the 10 largest combinational MCNC benchmark circuits. We first perform logic optimization and technology mapping to 6-input LUTs using Berkeley ABC tool [ABC]. Next, the mapped circuits are placed and routed by VPR tool set [LKJ11] with a cluster size of 8 using a minimum dimension of the FPGA to implement each circuit. Then, the SA based co-optimization algorithm is applied and verified for SER and leakage. In addition, we also apply the static probability based IPV algorithm to the benchmark circuits separately for the comparison purpose.

Figure 6.4 shows improvements on SER by both algorithms, where the improvements are calculated by the ratio of SER compared to the original circuit.

On average, the proposed co-optimization significantly reduces the SER by 52%, while the static probability method can only reduce SER by 10%. We observe that the observability values lie in a wide range from 1.0 to 1e-9. Therefore, according to Eq. 6.3, reducing the hamming distance with low corresponding observability only has minimal impact on SER. Furthermore, when the hamming distances of those pseudo fan-in pairs with large observability values are increased, SER rises significantly even if the overall hamming distance is reduced.

In addition, the average hamming distance between signals and the SER ratios after the signal probability based method are shown in Figure 6.5. The hamming distance between signals can be reduced by approximately 40% on average. From the experiment, we show that hamming distance serves as a good indicator for leakage optimization. As far as we investigate, although SER can also be reduced with lower hamming distance values for most cases, it is not a good heuristic because the observability is involved in SER calculation. For example, there are cases that the SER gets increased on the "des" circuit by 18%, even though the hamming distance is reduced by 13%.

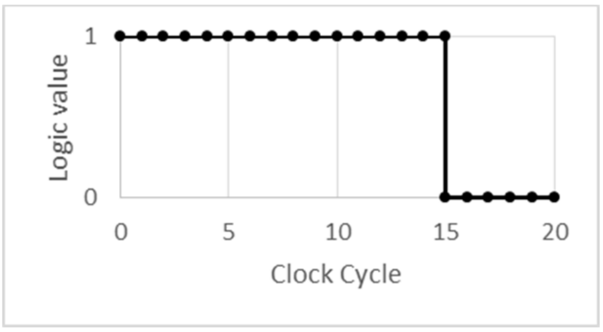
For leakage, the proposed method can achieve slightly better leakage reduction compared to the static probability method. The static probability method on average reduces leakage by 29%. As the leakage does not involve the observability, both the signal probability and co-optimization approaches can well predict the leakage, although the proposed co-optimizatoin can further reduce leakage by another 1%-5% for all of the benchmark circuits tested and achieves an averaged total leakage reduction by 33%.

From the experimental results, we show that without careful consideration of the soft error impact, SER can hardly be improved and may become even worse. While providing the same level or slightly better leakage reduction, from 10% (signal probability based) to 52% (co-optimization based), the proposed co-optimization algorithm demonstrates significant improvement on SER values.

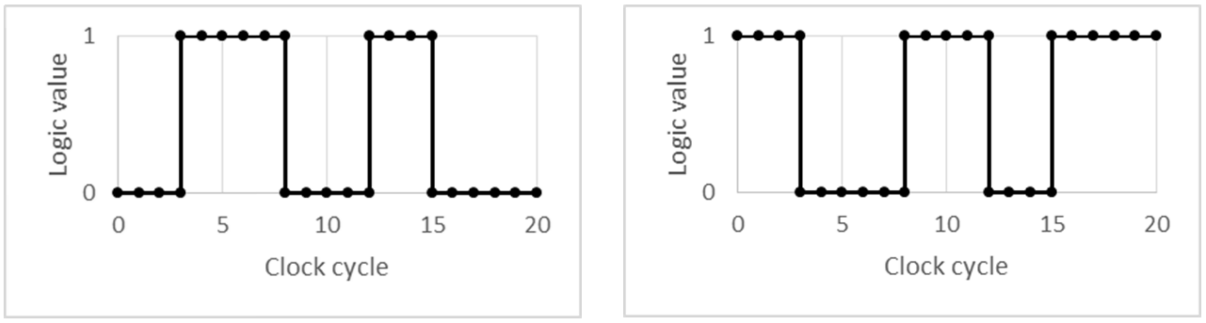
Particularly, our method reduces SER by 84% on “ex1010” circuit, which is 4.45× improvement compared to the static probability method.

6.4 Summary

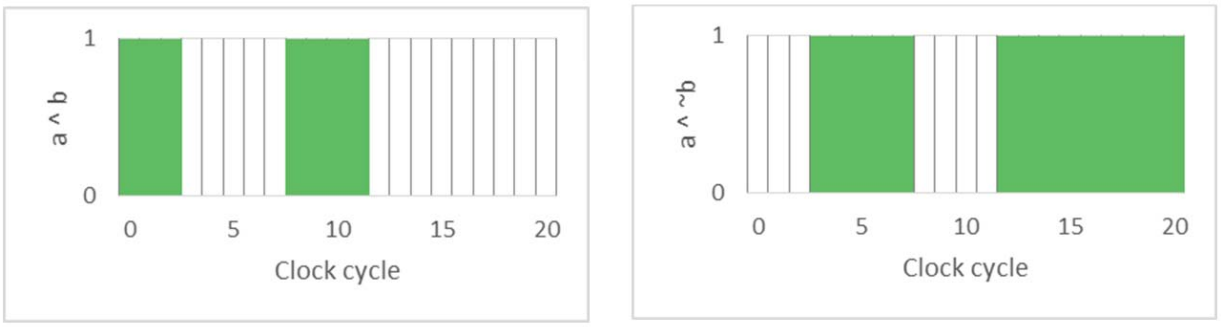
In this paper, the contradictions and difficulties of mitigations for leakage and SEU induced soft error are identified. We propose an efficient simulated annealing based co-optimization algorithm considering soft errors and leakage mitigations at the same time. Our experimental results show 52% soft error rate reduction and 33% leakage reduction on average on academic VPR FPGA architecture.



(a) signal I1 with static probability of 0.75



(b) signal I2 with static probability of 0.4 and I2



(c) leakage comparison

Figure 6.3: An example of inversion using static probability method that increases leakage.

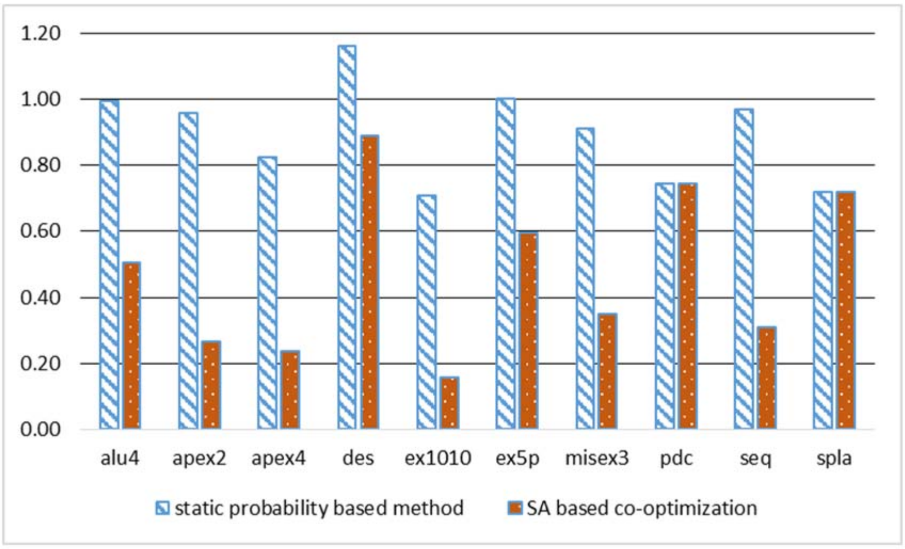


Figure 6.4: SER ratio for IPV based on static probability and the proposed SA based co-optimization algorithm.

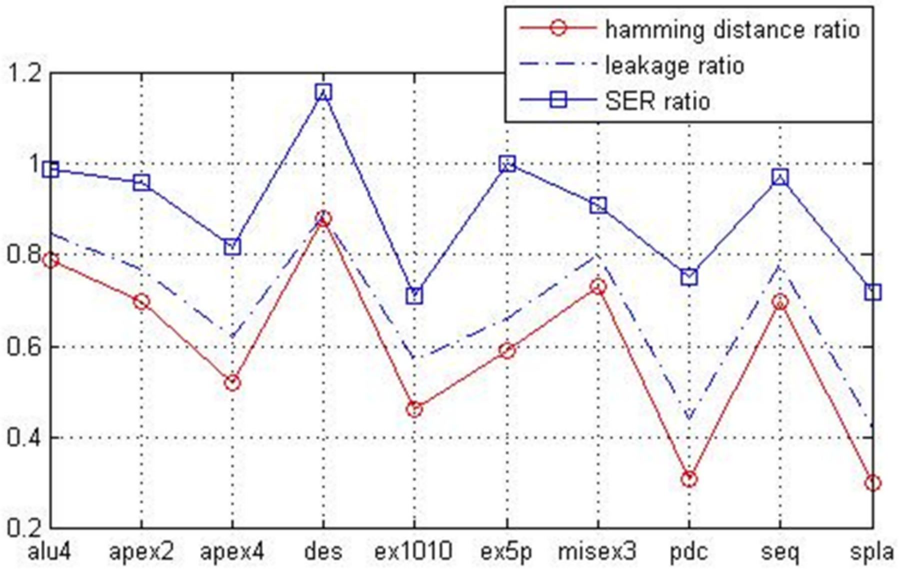


Figure 6.5: Average hamming distance, leakage, and SER ratio after applying LUT polarity inversion based on signal probability.

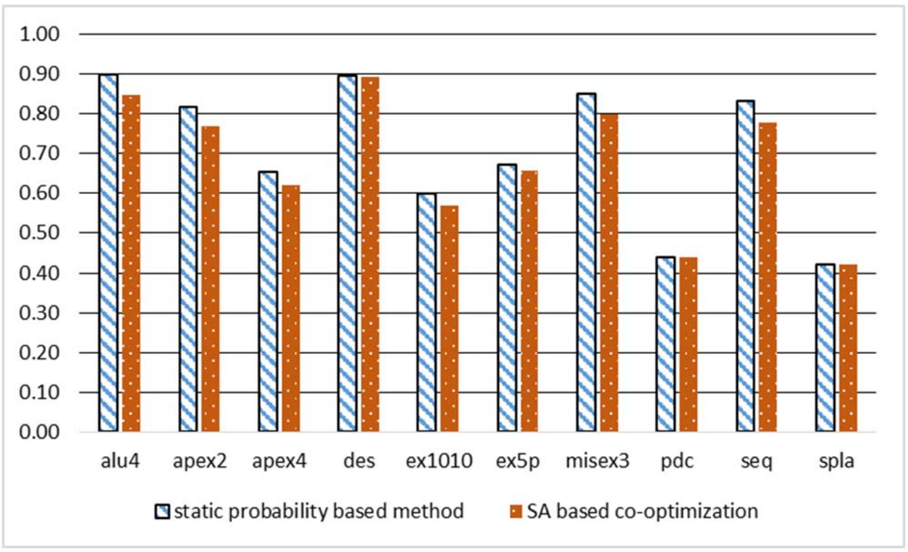


Figure 6.6: Leakage ratio for IPV based on static probability and the proposed SA based co-optimization algorithm.

CHAPTER 7

Validation of In-Place LUT Polarity Inversion for Soft Error Mitigation on Xilinx Virtex FPGAs

The objective of this chapter is to develop and validate IPV for industrial FPGAs. However, the physical architectures of industrial FPGAs are different from the academic VPR FPGA architecture which is assumed in the original IPV presented in Chapter 5. Therefore, a new soft error model and an extended IPV algorithm need to be developed in order to apply IPV to industrial FPGAs. In this work, we use a Xilinx Virtex xcv300 FPGA as our case study. We first analyze the impact of SEU on the interconnect of the xcv300 leveraging an in-field interconnect SEU injection tool developed and build a soft error model based on signal probability. Next, we propose an extended IPV 2.0 and its resynthesis flow integrated into Xilinx ISE design tool set. To validate, we map circuits to an xcv300 FPGA by ISE followed by IPV 2.0, and compare the soft error rates (SERs) with circuits without IPV 2.0. As an in-place re-synthesis approach, it does not change placement and routing in standard FPGA synthesis flow, and therefore has no impact on performance, area, and design closure.

The remainder of the paper is organized as follows. Section 7.1 proposes an SEU emulation methodology to calculate SER of an industrial FPGA. Section 7.2 presents the study of SEU impact on Xilinx Virtex FPGAs and the soft error modeling. Then in section 7.3, we introduce the proposed IPV 2.0 resynthesis algorithm and its integration to ISE. Section 7.4 presents the experimental results. Section 7.5 summarizes this chapter.

7.1 An SEU Emulation for Soft Error Rate Evaluation

To verify the SEU robustness of commercial FPGAs, fault simulation and emulation are the most popular techniques. However, most of the existing studies focused on the FPGA logic components. To investigate the fault characteristic and evaluate SER of the SEU on the commercial FPGA interconnect architecture, SEU emulations are used in our study. Before describing the proposed SEU emulation method, we first present an SEU injection tool via JTAG FPGA configuration interface.

Modern FPGAs feature advanced technique of dynamic and partial reconfiguration (DPR) capability. Dynamic reconfiguration enables us to make changes to the circuit design implemented on the FPGA by modifying the configuration memory (CRAM) data during runtime, without halting the FPGA operation. In addition, partial reconfiguration is another key feature that allows us to modify a small portion of the design without performing a full FPGA configuration. Combine the above two reconfiguration features, it allows us to flip a bit in the CRAM during runtime. We then develop an SEU injection tool based on DPR via the JTAG FPGA configuration interface. To inject an SEU into the FPGA CRAM, we first read back a frame from CRAM, where a frame is the minimum size of the data that can be read from or write to the Xilinx FPGA CRAM. Then, the frame is modified by flipping a bit. Finally, the modified frame data is written back to the CRAM to complete the SEU injection process. Note that this fault injection technique can be applied via other FPGA configuration interfaces, such as SelectMAP and the Internal Configuration Access Port (ICAP). We select JTAG because it is the most viable interface that is design for debugging purpose.

Leveraging the proposed JTAG SEU injection tool, we develop an FPGA SEU emulation flow as Figure 7.1 illustrates. The emulation iteratively injects SEU into the FPGA and performs circuit output analyses to characterize the fault behavior. During each iteration, a random SEU address is selected. An SEU location contains the frame address plus the bit offset in the frame. Next, the SEU is injected using the JTAG SEU injection tool discussed above. After an SEU is injected, the circuit primary outputs are analyzed and compared with the golden values for a predefined monitoring time period. Then, the comparison result is recorded for later fault behavior characterization. Note that for each time we start a new SEU to be injected to a different location, we perform full FPGA reprogramming in order to guarantee a clean experimental setup.

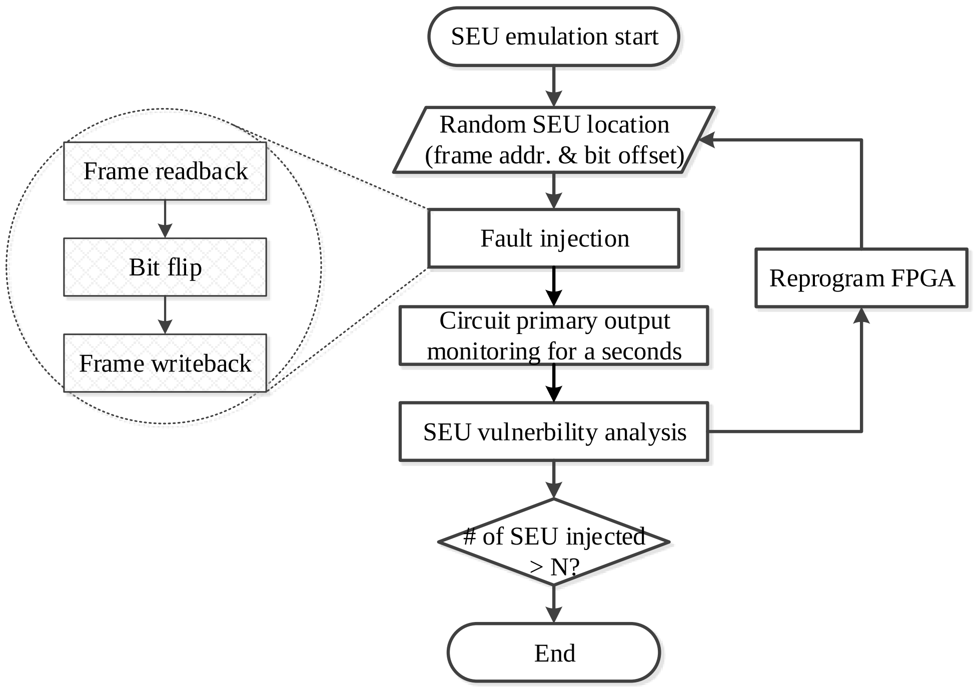

SEU emulation start
Random SEU location (frame addr. & bit offset)
Fault injection
Circuit primary output monitoring for a seconds
SEU vulnerbility analysis
# of SEU injected > N?
End
Reprogram FPGA
Frame readback
Bit flip
Frame writeback

Figure 7.1: An FPGA emulation flow for SEU analysis.

7.2 SEU Study for Xilinx Virtex Interconnect Architecture

Without loss of generality, with respect to the logic value changed, the SEU induced soft errors can be grouped into three categories: (1) switch-to-one, (2) switch-to-zero, and (3) always-inverting faults. A switch-to-one fault is the undesired logical transition from logic '0' to '1'. On the contrary, a switch-to-zero fault changes the logic value from '1' to '0', respectively. Finally, an always-inverting fault upsets the logic value disregarding the logic value carried on the net.

In the following we implement a circuit specific designed for our SEU evaluation purpose and perform two complementary experiments to quantitatively calculate the number of switch-to-zero, switch-to-one, and always-inverting faults caused by SEU on the interconnect.

For switch-to-zero faults, we implement a randomly generated large circuit netlist that is composed of AND gates only. Then, each of the primary inputs is connected to logic '1', such that every internal net and the primary output holds logic '1'. The circuit outputs are monitored to record any transition from '1' to '0'. Because logic '0' is the dominant input value of an AND gate, when a fault is raised in the circuit, only the logic transitions from logic '1' to '0', i.e., switch-to-zero or always-inverting fault is always propagated to the circuit primary output and captured.

Similarly, to capture the switch-to-one faults, we implement the exactly same circuit netlist, with identical placement and routing but replacing all AND gates with OR gates and connecting all of the primary inputs to logic '0'. Such circuit holds logic '0' on all nets in the circuit and is only sensitive to switch-to-one or always-inverting faults.

Next, we generate 10 different pairs of such complementary AND/OR circuit netlist, with 1K gates (LUTs) randomly placed and routed for each pair. Note

	Quantity	Percentage
Switch-to-one	872	82.41%
Switch-to-zero	182	17.20%
Always-inverting	4	0.39%

Table 7.1: Summary of the quantities of switch-to-one, switch-to-zero, and always-inverting faults.

that the primary concern of the circuit size is the routing utilization. Such circuit networks with 1K gates (LUTs) after random placement on a xcv300 FPGA result in complex routing networks that are complicated enough to characterize the interconnect fault behaviors. Then, 5000 SEU locations are randomly selected for each pair. Each SEU is injected to both the AND-network and the OR-network of a pair, with one SEU injection at a time, and we monitor and record the faults observed using the proposed emulation flow.

The emulation results are summarized in Table 7.1, where we show the average number of each type of faults from the 10 circuit pairs and their percentage. Approximately, we observe that only 21% of the injected have functional impact on the circuits. Moreover, the probability of inverting fault is very low to be negligible. The emulation results from real Virtex FPGA convince us that the majority of the SEU on Virtex FPGA interconnect causes switch-to-one faults.

7.3 IPV2.0 for Xilinx Virtex FPGAs

In this section, we first propose a new soft error model based on static signal probability for Xilinx Virtex FPGA interconnect architecture. Then, we propose IPV2.0 algorithm integrated into Xilinx ISE design flow.

Base on the investigation in Section 7.2, the SER of an interconnect CRAM bit is determined by the static probability of the net that the CRAM bit is controlling, which is given by

$$SER_b = P_{st0} \times SP_{Net(b)} \times O_{b0} + P_{st1} \times (1 - SP_{Net(b)}) \times O_{b1} \tag{7.1}$$

where P_{st0} and P_{st1} denote the probabilities of switch-to-zero and switch-to-one faults, $SP_{Net(b)}$ denotes the static probability of the net that CRAM bit b is controlling, and O_{b0} and O_{b1} are the observabilities of logic '0' and '1' of the net. The inverting fault is ignored because its probability is relatively tiny. According to Table 7.1, since $P_{st1} >> P_{st0}$, a net with low static probability value is more vulnerable due to switch-to-one faults and the SER can be approximated by:

$$SER_b = P_{st1} \times (1 - SP_{Net(b)}) \times O_{b1} \tag{7.2}$$

Eq. 7.2 leads to a conclusion that a higher signal probability results in a lower SER value. Therefore, we develop a signal probability based IPV algorithm for Virtex FPGAs, with an objective to increase the overall static probability. In essence, when a net has its signal probability less than 0.5, it is suggested that the net to be inverted. Therefore, the algorithm inverts the logic polarities of those nets whose signal probability is below 0.5 to increase the overall signal probability and thus mitigates SER of the design.

Next, we present an IPV2.0 resynthesis flow integrated into the Xilinx ISE toolchain depicted in Figure 7.2. The proposed IPV2.0 resynthesis contains two major steps: (1) static probability calculation and (2) LUT inversion. To obtain static probability information, we perform simulation on the post-routed circuit netlist and calculate static probabilities of net by analyzing the simulation waveform. Next, to apply net polarity inversion to the post-routed circuit, we develop an IPV resynthesis tool based on Xilinx Design Language (XDL). The circuit is converted to the XDL format and static probability based IPV resynthesis is applied. Finally, the modified circuit netlist with IPV technique is converted to generate the design bitstream of the FPGA.

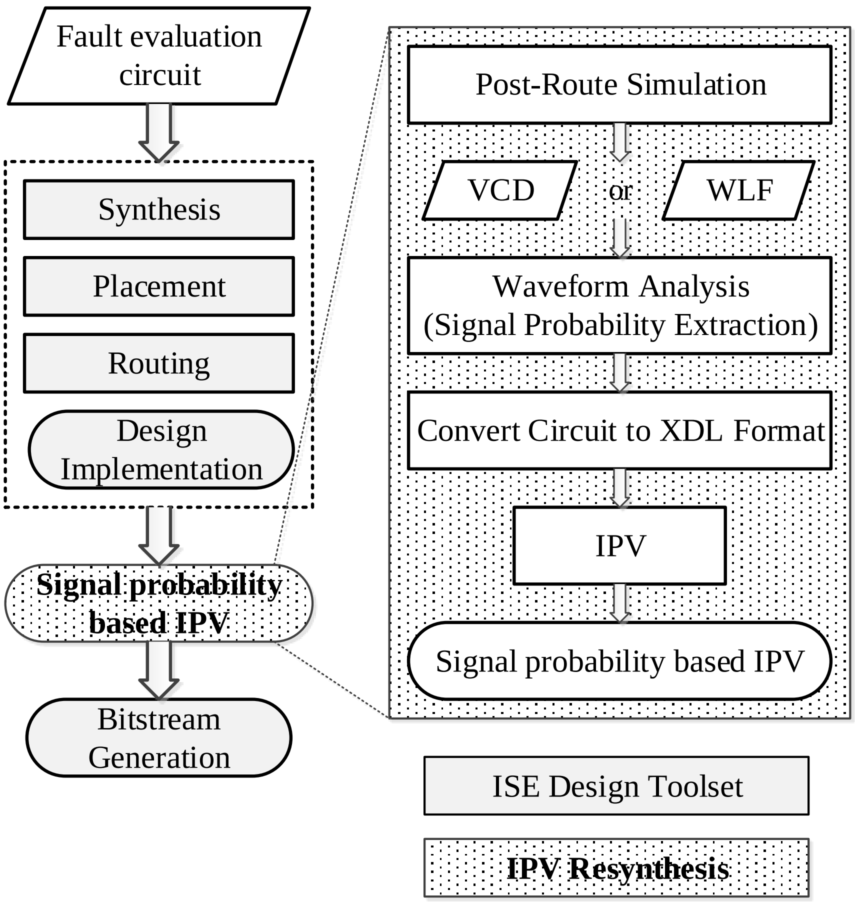


Figure 7.2: Signal probability based IPV resynthesis flow for Xilinx Virtex FPGAs.

7.4 Experimental results

The IPV resynthesis for Virtex FPGA discussed above is applied and verified using the 10 largest combinational MCNC benchmark circuits implemented on an xcv300 FPGA chip.

To evaluate the number of soft errors caused by SEU, we implement an SEU evaluation platform as illustrated in Figure 7.3. Each benchmark circuit is duplicated where one copy serves as the golden case for comparison purpose and the other copy serves as the circuit under test (CUT). The CUT is placed within a specific region that is isolated from any other circuit module, such that an SEU can be injected to the CUT precisely. Both the inputs of the golden circuit and the CUT are given by the random input vectors generated by a Linear Feedback Shift Register (LFSR). Finally, the outputs from the golden circuit and the CUT are connected to the monitoring module for comparison, where we perform exclusive or to detect errors and send the number of error detected to a host PC for SER analyses.

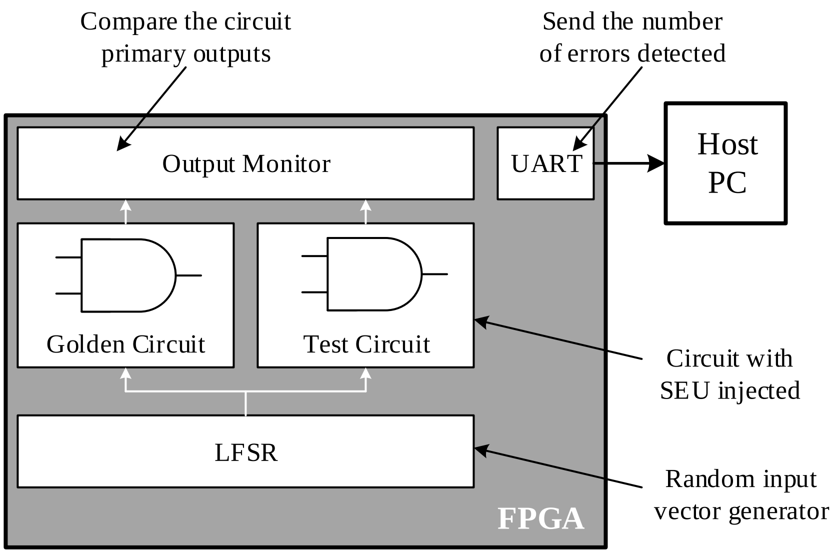

Compare the circuit primary outputs
Send the number of errors detected
Output Monitor
UART
Host PC
Golden Circuit
Test Circuit
Circuit with SEU injected
LFSR
Random input vector generator
FPGA

Figure 7.3: SEU emulation platform for soft error evaluation.

For each benchmark circuit, we randomly select 5000 SEU locations that are used for interconnect configuration within the placement region of the CUT and inject one SEU at a time. Then, each SEU is evaluated using 100k random input vectors by the emulation flow proposed in Figure 7.1 to calculate the number of errors that can be observed at the circuit primary outputs for SER estimation. Then, to evaluate the SER improvement, IPV is applied to the CUT and the same SEU emulation is performed.

The experimental results are summarized in Figure 7.4, depicting the total number of errors that are detected. According to Eq. 2.2, the SER value is proportional to the total number of errors. Therefore, the SER improvement ratio can be calculated by the total number of errors of the circuit with IPV2.0 divided by that of the original circuit. The results show that IPV2.0 achieves on average 2× SER improvement approximately. Particularly, the results show that 5× SER improvement can be achieved on the “ex1010” circuit.

In terms of runtime, the runtime of signal probability based IPV2.0 algorithm is less than a minute on an Intel Xeon w3540 quad core CPU with 24G memory for all of the benchmark circuit tested. The majority runtime of the entire resynthesis flow is consumed by the post-route simulation to calculate the signal probability values for each net. However, several vector-less based techniques to calculate signal probability and switching activity have been proposed and widely used in both academic research and industrial tools such as [CB10] and [AR12]. Leveraging those analysis techniques, it leads to a very low design closure overhead of the proposed IPV resynthsis flow.

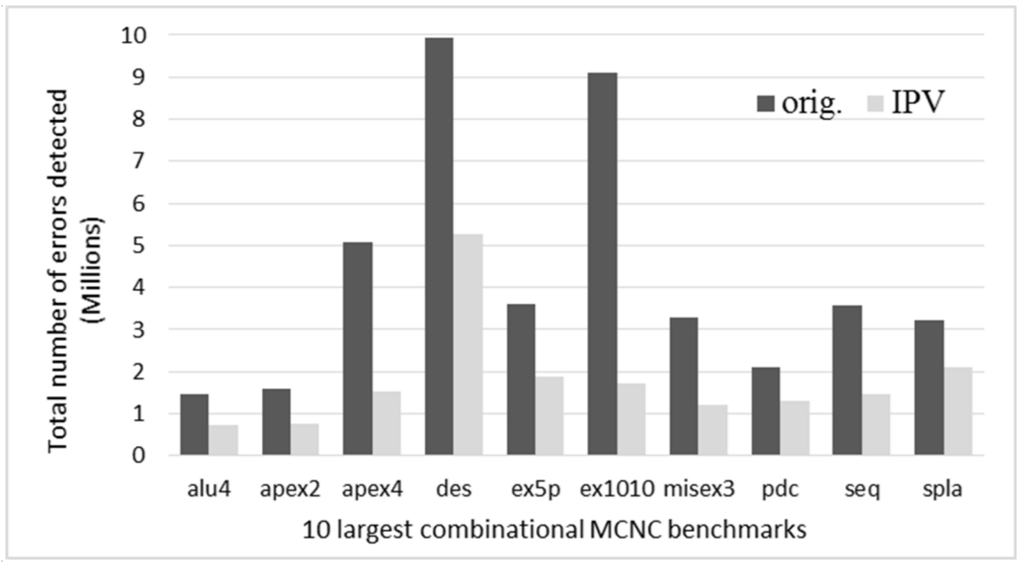


Figure 7.4: SEU emulation platform for soft error evaluation.

7.5 Summary

In this chapter, the contradictions and difficulties of mitigation for SEU induced soft error and leakage are identified. We propose an efficient simulated annealing based co-optimization algorithm considering soft errors and leakage mitigations at the same time. Our experimental results show 52% soft error rate reduction and 33% leakage reduction on average on academic VPR FPGA architecture. In addition, an efficient soft error model is proposed for Xilinx Virtex FPGAs and a modified IPV algorithm is proposed. Then, validation of IPV on a Virtex xcv300 FPGA is presented in this paper. Using SEU emulation, the results show that IPV2.0 can achieve 50% soft error reduction on average, i.e. $2\times$ MTTF improvement.

CHAPTER 8

Conclusions and Future Work

8.1 Conclusion and Future work

In this dissertation we proposed a robust design methodology for FPGA. We first developed a SEU soft error analysis framework in Chapter 2 to quantitatively evaluate the FPGA reliability against soft errors. To improve FPGA reliability, we first provided a system level study and presented a novel CRAM scrubbing technique in Chpater 3. In Chapter 4, to improve reliability for FPGA logic components, we proposed a local remapping technique leveraging the underutilized modern FPGA architecture features. From Chapter 5 to Chapter 7, we proposed an LUT polarity re-assignment technique to reduce the soft error impact and leakage power consumption on FPGA routing components explicitly. Then, the polarity re-assignment algorithm was applied to an industrial FPGA to validate the effectiveness on soft error mitigation.

In Chapter 2, a comprehensive SEU soft error evaluation framework for SRAM-based FPGAs was proposed. Unlike most of the existing soft error evaluation methods targeting the logic level, the proposed framework quantitatively measures the impact of soft errors to circuit functional failure at the physical design level. The proposed framework is capable of quantifying the soft error induced functional failures for exact FPGA configuration bits in various circuit elements, such as LUTs, connection boxes, switch boxes, and local routing multiplexers. Leveraging the proposed framework, we evaluated different FPGA architectures characterized by CLB sizes, LUT sizes, and routing structures. Most important of all, the proposed framework helps to identify critical components of a design implemented on an FPGA, which enables development of efficient robust techniques and their validation.

To mitigate the soft error impact, we first developed a novel robust technique at the system level. In Chapter 3, we proposed heterogeneous CRAM scrubbing technique (HCS). For the first time we proposed a MTTF estimation method for FPGA CRAM scrubbing using a stochastic modeling of system failures. While the conventional CRAM scrubbing technique does not consider the failure sensitivity of a memory bit, HCS rewrites critical CRAM bits more aggressively to improve the overall system MTTF. Based on the information of failure sensitivity, we proposed a dynamic programming based algorithm that efficiently solves HCS problem. By a system level experiment on an H.264/AVC decoder, we showed that HCS achieves 60% MTTF improvement over the conventional CRAM scrubbing technique.

Next, we continue to improve FPGA robustness at physical design level. To improve robustness of functions implemented on FPGA logic components, we proposed in-place decomposition technique (IPD). Leveraging the under utilized decomposable LUT and built-in carry chain/adder features, IPD decomposes functions into smaller subfunctions by decomposable LUTs and converges subfunctions by built-in carry chain/adder. In doing this, not only the overall LUT CRAM bit utilization is reduced but also the error masking capability is improved. We showed that IPD on average improves MTTF of LUT by 4.55× and 3.33× under 0% and 20% carry chain/adder utilization rates.

On the other hand, we improved robustness of FPGA routing components by proposing an in-place LUT polarity inversion technique (IPV) in Chapter 5. By leveraging error masking at FPGA routing multiplexors, we proposed to re-assign LUT polarities to optimize the overall soft error rate. As demonstrated in the experiments, the approach significantly reduces the interconnect fault in terms of SER by nearly 4× on average. In addition, we observed that combining the IPD and IPV can lead to a SER reduction up to 5.3× at chip level. Furthermore, we presented a SER and leakage power co-optimization algorithm for IPV in Chapter 6 and showed that it reduces leakage power by 30% while maintaining a similar level of SER reduction. Lastly, in Chapter 7, a validation of IPV on an industrial FPGA architecture is presented, where a 2× SER reduction is demonstrated.

There are several future directions for this research. First, a future direction is to explore logic and physical synthesis to improve memory scrubbing efficiency. For example, although we have validated the combination of the two in-place techniques proposed, the interaction between memory scrubbing and the two in-place techniques has not yet been studied and will be an interesting future work. Another interesting future direction is to explore architecture impact on the proposed techniques. For example, in Chapter 6 and 7, we have shown the SER improvement for IPV based on static signal probability on academic VPR and Xilinx Virtex FPGA architectures. While IPV based on static signal probability achieves 2× SER reduction on Xilinx Virtex architecture, its improvement on VPR FPGA architecture is less than 20%. Therefore, the proposed techniques need to be architecture-aware and more efficient algorithms need to be developed. Lastly, validation on industrial FPGA architectures and exploiting advanced architecture features will also be an interesting future work. In the future, by identifying sequential feedbacks, we can also apply our approach to sequential circuits. Furthermore, commercial architectures will be modeled to make this framework more general for architectural and synthesis algorithm evaluation with respect to SEU fault in FPGAs.

Our SEU fault evaluation framework provides detailed information for identifying the most critical configuration bits or circuit elements to develop new fault mitigation algorithms. We envision that our fault evaluation framework will be used to cast more useful insights for the design of more robust FPGA circuits, architectures, and better synthesis algorithms.

References

[ABC] "ABC: A system for sequential synthesis and verification." In *Berkeley Logic Synthesis and Verification Group.*

[acta] "HiRel SX-A Family FPGAs Datasheet." In *http://www.actel.com.*

[Actb] "Radiation Hardened FPGAs Datasheet." In *http://www.actel.com.*

[AKA08] Taneem Ahmed, Paul D Kundarewich, Jason H Anderson, Brad L Taylor, and Rajat Aggarwal. "Architecture-specific packing for virtex-5 FPGAs." In *Proceedings of the 16th international ACM/SIGDA symposium on Field programmable gate arrays*, pp. 5–13. ACM, 2008.

[alt] "Altera Stratix IV Features." In *http://www.altera.com.*

[AME02] Mohab Anis, Mohamed Mahmoud, Mohamed Elmasry, and Shawki Areibi. "Dynamic and leakage power reduction in MTCMOS circuits using an automated efficient gate clustering technique." In *Proceedings of the 39th annual Design Automation Conference*, pp. 480–485. ACM, 2002.

[AN06] Jason Helge Anderson and Farid N Najm. "Active leakage power optimization for FPGAs." *Computer-Aided Design of Integrated Circuits and Systems, IEEE Transactions on*, **25**(3):423–437, 2006.

[AR12] PJ Anju and SR Ramesh. "Toggle rate estimation technique for FPGA circuits considering spatial correlation." In *Computing Communication & Networking Technologies (ICCCNT), 2012 Third International Conference on*, pp. 1–7. IEEE, 2012.

[AT05] Ghazanfar Asadi and Mehdi B Tahoori. "Soft error rate estimation and mitigation for SRAM-based FPGAs." In *Proceedings of the 2005 ACM/SIGDA 13th international symposium on Field-programmable gate arrays*, pp. 149–160. ACM, 2005.

[AT07] Hossein Asadi and Mehdi Baradaran Tahoori. "Analytical techniques for soft error rate modeling and mitigation of FPGA-based designs." *Very Large Scale Integration (VLSI) Systems, IEEE Transactions on*, **15**(12):1320–1331, 2007.

[ATM07] Hossein Asadi, Mehdi B Tahoori, Brian Mullins, David Kaeli, and Kevin Granlund. "Soft error susceptibility analysis of SRAM-based FPGAs in high-performance information systems." *Nuclear Science, IEEE Transactions on*, **54**(6):2714–2726, 2007.

[BBB04a] M Bellato, Paolo Bernardi, D Bortolato, A Candelori, M Ceschia, Alessandro Paccagnella, Maurizio Rebaudengo, M Sonza Reorda, Massimo Violante, and P Zambolin. "Evaluating the effects of SEUs affecting the configuration memory of an SRAM-based FPGA." In *Proceedings of the conference on Design, automation and test in Europe-Volume 1*, p. 10584. IEEE Computer Society, 2004.

[BBB04b] M Bellato, Paolo Bernardi, D Bortolato, A Candelori, M Ceschia, Alessandro Paccagnella, Maurizio Rebaudengo, M Sonza Reorda, Massimo Violante, and P Zambolin. "Evaluating the effects of SEUs affecting the configuration memory of an SRAM-based FPGA." In *Proceedings of the conference on Design, automation and test in Europe-Volume 1*, p. 10584. IEEE Computer Society, 2004.

[BCD11] Cinzia Bernardeschi, Luca Cassano, and Andrea Domenici. "Failure probability of SRAM-FPGA systems with Stochastic Activity Networks." In *Design and Diagnostics of Electronic Circuits & Systems (DDECS), 2011 IEEE 14th International Symposium on*, pp. 293–296. IEEE, 2011.

[BCD12] Cinzia Bernardeschi, Luca Cassano, Andrea Domenici, and Luca Sterpone. "Accurate simulation of SEUs in the configuration memory of SRAM-based FPGAs." In *Defect and Fault Tolerance in VLSI and Nanotechnology Systems (DFT), 2012 IEEE International Symposium on*, pp. 115–120. IEEE, 2012.

[Bid10] N Bidokhti. "SEU concept to reality (allocation, prediction, mitigation)." In *Reliability and Maintainability Symposium (RAMS), 2010 Proceedings-Annual*, pp. 1–5. IEEE, 2010.

[BPP08] Melanie Berg, C Poivey, D Petrick, D Espinosa, Austin Lesea, KA LaBel, M Friendlich, H Kim, and Anthony Phan. "Effectiveness of internal versus external SEU scrubbing mitigation strategies in a Xilinx FPGA: Design, test, and analysis." *Nuclear Science, IEEE Transactions on*, **55**(4):2259–2266, 2008.

[BR97] Vaughn Betz and Jonathan Rose. "VPR: A new packing, placement and routing tool for FPGA research." In *Field-Programmable Logic and Applications*, pp. 213–222. Springer, 1997.

[CB10] Tomasz S Czajkowski and Stephen D Brown. "Decomposition-based vectorless toggle rate computation for FPGA circuits." *Computer-Aided Design of Integrated Circuits and Systems, IEEE Transactions on*, **29**(11):1723–1735, 2010.

[Cha09] Ken Chapman. "SEU Strategies for Virtex-5 Devices." In *http://www.xilinx.com/*, 2009.

[CLC04] Luca Ciccarelli, Andrea Lodi, and Roberto Canegallo. "Low leakage circuit design for FPGAs." In *Custom Integrated Circuits Conference, 2004. Proceedings of the IEEE 2004*, pp. 715–718. IEEE, 2004.

[CM10] Jason Cong and Kirill Minkovich. "LUT-based FPGA technology mapping for reliability." In *Proceedings of the 47th Design Automation Conference*, pp. 517–522. ACM, 2010.

[CO05] In Suk Chong and Antonio Ortega. "Hardware testing for error tolerant multimedia compression based on linear transforms." In *Defect and Fault Tolerance in VLSI Systems, 2005. DFT 2005. 20th IEEE International Symposium on*, pp. 523–531. IEEE, 2005.

[CR06] Adrian Cosoroaba and Frédéric Rivoallon. "Achieving higher system performance with the Virtex-5 family of FPGAs." *Xilinx WP245 V*, **1**, 2006.

[CVR03] M Ceschia, M Violante, M Sonza Reorda, A Paccagnella, P Bernardi, M Rebaudengo, D Bortolato, M Bellato, P Zambolin, and A Candelori. "Identification and classification of single-event upsets in the configuration memory of SRAM-based FPGAs." *Nuclear Science, IEEE Transactions on*, **50**(6):2088–2094, 2003.

[CX11] Jason Cong and Bingjun Xiao. "mrFPGA: A novel FPGA architecture with memristor-based reconfiguration." In *Nanoscale Architectures (NANOARCH), 2011 IEEE/ACM International Symposium on*, pp. 1–8. IEEE, 2011.

[DeH96] Andre DeHon. "Reconfigurable architectures for general-purpose computing." 1996.

[DH06] Asbjoern Djupdal and Pauline C Haddow. "Yield enhancing defect tolerance techniques for FPGAs." In *MAPLD International Conference*. Citeseer, 2006.

[DR08] Joanne E Degroat and C Ramswamy. "Error Detection and Correction-A novel technique implementing Dual Rail Logic and Rollback recovery Architecture." In *Aerospace and Electronics Conference, 2008. NAECON 2008. IEEE National*, pp. 89–91. IEEE, 2008.

[FAL12] JB Ferron, L Anghel, and R Leveugle. "Towards low-cost soft error mitigation in SRAM-based FPGAs: A case study on AT40K." In *Circuits and Systems (LASCAS), 2012 IEEE Third Latin American Symposium on*, pp. 1–4. IEEE, 2012.

[FHH09] Zhe Feng, Yu Hu, Lei He, and Rupak Majumdar. "IPR: in-place reconfiguration for FPGA fault tolerance." In *Proceedings of the 2009 International Conference on Computer-Aided Design*, pp. 105–108. ACM, 2009.

[FNH11] Zhe Feng, Jing Naifeng, Yu Hu, and Lei He. "IPF: In-place X-Filing to Mitigate Soft Errors in SRAM-based FPGAs." In *International Conference on Field Programmable Logic and Applications(FPL)*, pp. 482–485, 2011.

[Gat] Modifying Gates. "Runtime Mechanisms for Leakage Current Reduction in CMOS VLSI Circuits.".

[GB07] Shahin Golshan and Elaheh Bozorgzadeh. "Single-event-upset (SEU) awareness in FPGA routing." In *Proceedings of the 44th annual Design Automation Conference*, pp. 330–333. ACM, 2007.

[GCJ03] Paul Graham, Michael Caffrey, D Eric Johnson, Nathan Rollins, and Michael Wirthlin. "SEU mitigation for half-latches in Xilinx Virtex FPGAs." *Nuclear Science, IEEE Transactions on*, **50**(6):2139–2146, 2003.

[GCZ03] Paul Graham, Michael Caffrey, J Zimmerman, P Sundararajan, E Johnson, and Cameron Patterson. "Consequences and categories of SRAM FPGA configuration SEUs." In *Proceedings of the 6th Military and Aerospace Applications of Programmable Devices and Technologies International Conference (MAPLD'2003) C*, volume 6, p. 2003, 2003.

[Geo00] Varghese George. *Low energy field-programmable gate array.* PhD thesis, Citeseer, 2000.

[GKB11] Shahin Golshan, Hessam Kooti, and Elaheh Bozorgzadeh. "SEU-Aware High-Level Data Path Synthesis and Layout Generation on SRAM-Based FPGAs." *Computer-Aided Design of Integrated Circuits and Systems, IEEE Transactions on*, **30**(6):829–840, 2011.

[GYM05] Haixia Gao, Yintang Yang, Xiaohua Ma, and Gang Dong. "Testing for resistive shorts in FPGA interconnects." In *Quality of Electronic Design, 2005. ISQED 2005. Sixth International Symposium on*, pp. 159–163. IEEE, 2005.

[HAW05] Olivier Héron, Talal Arnaout, and H-J Wunderlich. "On the reliability evaluation of SRAM-based FPGA designs." In *Field Programmable Logic and Applications, 2005. International Conference on*, pp. 403–408. IEEE, 2005.

[HFH08] Yu Hu, Zhe Feng, Lei He, and Rupak Majumdar. "Robust FPGA resynthesis based on fault-tolerant Boolean matching." In *Computer-Aided Design, 2008. ICCAD 2008. IEEE/ACM International Conference on*, pp. 706–713. IEEE, 2008.

[HHL12] Keheng Huang, Yu Hu, Xiaowei Li, Bo Liu, Hongjin Liu, and Jian Gong. "Off-path leakage power aware routing for SRAM-based FPGAs." In *Design, Automation & Test in Europe Conference & Exhibition (DATE), 2012*, pp. 87–92. IEEE, 2012.

[HN97] Jonathan P Halter and Farid N Najm. "A gate-level leakage power reduction method for ultra-low-power CMOS circuits." In *Custom Integrated Circuits Conference, 1997., Proceedings of the IEEE 1997*, pp. 475–478. IEEE, 1997.

[HSW09] Jonathan Heiner, Benjamin Sellers, Michael Wirthlin, and Jeff Kalb. "FPGA partial reconfiguration via configuration scrubbing." In *Field Programmable Logic and Applications, 2009. FPL 2009. International Conference on*, pp. 99–104. IEEE, 2009.

[ITA12] Yoshihiro Ichinomiya, Kohei Takano, Motoki Amagasaki, Morihiro Kuga, Masahiro Iida, and Toshinori Sueyoshi. "Accelerated evaluation of SEU failure-in-time using frame-based partial reconfiguration." In *Field-Programmable Technology (FPT), 2012 International Conference on*, pp. 220–223. IEEE, 2012.

[ITY10] Eishi Ibe, Hitoshi Taniguchi, Yasuo Yahagi, K-i Shimbo, and Tadanobu Toba. "Impact of scaling on neutron-induced soft error in SRAMs from a 250 nm to a 22 nm design rule." *Electron Devices, IEEE Transactions on*, **57**(7):1527–1538, 2010.

[JCC08] Stephen Jang, Billy Chan, Kevin Chung, and Alan Mishchenko. "WireMap: FPGA technology mapping for improved routability." In *Proceedings of the 16th international ACM/SIGDA symposium on Field programmable gate arrays*, pp. 47–55. ACM, 2008.

[JCG03] Eric Johnson, Michael Caffrey, Paul Graham, Nathan Rollins, and Michael Wirthlin. "Accelerator validation of an FPGA SEU simulator." *Nuclear Science, IEEE Transactions on*, **50**(6):2147–2157, 2003.

[JHM10] Manu Jose, Yu Hu, Rupak Majumdar, and Lei He. "Rewiring for robustness." In *Proceedings of the 47th Design Automation Conference*, pp. 469–474. ACM, 2010.

[JMA12] Hadi Jahanirad, Karim Mohammadi, and Pejman Attarsharghi. "Single fault reliability analysis in FPGA implemented circuits." In *Quality*

Electronic Design (ISQED), 2012 13th International Symposium on, pp. 49–56. IEEE, 2012.

[KAJ12] Uli Kretzschmar, Armando Astarloa, Jaime Jimenez, Mikel Garay, and J Del Ser. "Fast and Accurate Single Bit Error Injection into SRAM Based FPGAs." In *Field Programmable Logic and Applications (FPL), 2012 22nd International Conference on*, pp. 675–678. IEEE, 2012.

[KNC02] James Kao, Siva Narendra, and Anantha Chandrakasan. "Subthreshold leakage modeling and reduction techniques." In *Proceedings of the 2002 IEEE/ACM international conference on Computer-aided design*, pp. 141–148. ACM, 2002.

[KPM07] Smita Krishnaswamy, Stephen M Plaza, Igor L Markov, and John P Hayes. "Enhancing design robustness with reliability-aware resynthesis and logic simulation." In *Computer-Aided Design, 2007. ICCAD 2007. IEEE/ACM International Conference on*, pp. 149–154. IEEE, 2007.

[KR02] Chris H Kim and Kaushik Roy. "Dynamic V¡ sub¿ TH¡/sub¿ scaling scheme for active leakage power reduction." In *Design, Automation and Test in Europe Conference and Exhibition, 2002. Proceedings*, pp. 163–167. IEEE, 2002.

[KSC05a] F Lima Kastensmidt, Luca Sterpone, Luigi Carro, and M Sonza Reorda. "On the optimal design of triple modular redundancy logic for SRAM-based FPGAs." In *Proceedings of the conference on Design, Automation and Test in Europe-Volume 2*, pp. 1290–1295. IEEE Computer Society, 2005.

[KSC05b] F Lima Kastensmidt, Luca Sterpone, Luigi Carro, and M Sonza Reorda. "On the optimal design of triple modular redundancy logic for SRAM-based FPGAs." In *Proceedings of the conference on Design, Automation and Test in Europe-Volume 2*, pp. 1290–1295. IEEE Computer Society, 2005.

[LBN10] Uros Legat, Anton Biasizzo, and Franc Novak. "Automated SEU fault emulation using partial FPGA reconfiguration." In *Design and Diagnostics of Electronic Circuits and Systems (DDECS), 2010 IEEE 13th International Symposium on*, pp. 24–27. IEEE, 2010.

[Le] Robert Le. "Soft Error Mitigation Using Prioritized Essential Bits." In *http://www.xilinx.com.*

[LHM10] Ju-Yueh Lee, Yu Hu, Rupak Majumdar, Lei He, and Minming Li. "Fault-tolerant resynthesis with dual-output LUTs." In *Proceedings of the 2010 Asia and South Pacific Design Automation Conference*, pp. 325–330. IEEE Press, 2010.

[LKJ11] Jason Luu, Ian Kuon, Peter Jamieson, Ted Campbell, Andy Ye, Wei Mark Fang, Kenneth Kent, and Jonathan Rose. "VPR 5.0: FPGA cad and architecture exploration tools with single-driver routing, heterogeneity and process scaling." *ACM Transactions on Reconfigurable Technology and Systems (TRETS)*, **4**(4):32, 2011.

[LLH10] Samuel Luckenbill, Ju-Yueh Lee, Yu Hu, Rupak Majumdar, and Lei He. "RALF: Reliability Analysis for Logic FaultsXAn exact algorithm and its applications." In *Design, Automation & Test in Europe Conference & Exhibition (DATE), 2010*, pp. 783–788. IEEE, 2010.

[LLT04] Guy Lemieux, Edmund Lee, Marvin Tom, and Anthony Yu. "Directional and single-driver wires in FPGA interconnect." In *Field-Programmable Technology, 2004. Proceedings. 2004 IEEE International Conference on*, pp. 41–48. IEEE, 2004.

[LSR11] Shih-Fu Liu, Gabriele Sorrenti, Pedro Reviriego, Fabio Casini, Juan Antonio Maestro, and Monica Alderighi. "Increasing reliability of FPGA-based adaptive equalizers in the presence of single event upsets." *Nuclear Science, IEEE Transactions on*, **58**(3):1072–1077, 2011.

[LSR12] S Liu, G Sorrenti, P Reviriego, F Casini, JA Maestro, M Alderighi, and H Mecha. "Comparison of the Susceptibility to Soft Errors of SRAM-Based FPGA Error Correction Codes Implementations." *Nuclear Science, IEEE Transactions on*, **59**(3):619–624, 2012.

[LV62] Robert E Lyons and Wouter Vanderkulk. "The use of triple-modular redundancy to improve computer reliability." *IBM Journal of Research and Development*, **6**(2):200–209, 1962.

[MB05] Alan Mishchenko and Robert K Brayton. "SAT-based complete don't-care computation for network optimization." In *Design, Automation and Test in Europe, 2005. Proceedings*, pp. 412–417. IEEE, 2005.

[MBJ11] Alan Mishchenko, Robert Brayton, Jie-Hong R Jiang, and Stephen Jang. "Scalable don't-care-based logic optimization and resynthesis." *ACM Transactions on Reconfigurable Technology and Systems (TRETS)*, **4**(4):34, 2011.

[McC09] John McCollum. "ASIC versus antifuse FPGA reliability." In *Aerospace conference, 2009 IEEE*, pp. 1–11. IEEE, 2009.

[McM05] K McMillan. "Dont-care computation using k-clause approximation." *Proc. IWLS05*, pp. 153–160, 2005.

[MER05] Shubhendu S Mukherjee, Joel Emer, and Steven K Reinhardt. "The soft error problem: An architectural perspective." In *High-Performance Computer Architecture, 2005. HPCA-11. 11th International Symposium on*, pp. 243–247. IEEE, 2005.

[MFM02] Steven M Martin, Krisztian Flautner, Trevor Mudge, and David Blaauw. "Combined dynamic voltage scaling and adaptive body biasing for lower power microprocessors under dynamic workloads." In *Proceedings of the 2002 IEEE/ACM international conference on Computer-aided design*, pp. 721–725. ACM, 2002.

[mos] "mosek." In *http://www.mosek.com.*

[Muk11] Shubu Mukherjee. *Architecture design for soft errors.* Morgan Kaufmann, 2011.

[NSB12] Mahtab Niknahad, Oliver Sander, and Juergen Becker. "Fine grain fault toleranceXA key to high reliability for FPGAs in space." In *Aerospace Conference, 2012 IEEE*, pp. 1–10. IEEE, 2012.

[PCC08] Brian Pratt, Michael Caffrey, James F Carroll, Paul Graham, Keith Morgan, and Michael Wirthlin. "Fine-grain SEU mitigation for FPGAs using partial TMR." *Nuclear Science, IEEE Transactions on*, **55**(4):2274–2280, 2008.

[PCG08] Brian H Pratt, Michael P Caffrey, Derrick Gibelyou, Paul S Graham, Keith Morgan, and Michael J Wirthlin. "TMR with More Frequent Voting for Improved FPGA Reliability." In *ERSA*, pp. 153–158, 2008.

[RCS05] E Syam Sundar Reddy, Vikram Chandrasekhar, Milagros Sashikánth, V Kamakoti, and Narayanan Vijaykrishnan. "Detecting SEU-caused routing errors in SRAM-based FPGAs." In *VLSI Design, 2005. 18th International Conference on*, pp. 736–741. IEEE, 2005.

[RP04] Arifur Rahman and Vijay Polavarapuv. "Evaluation of low-leakage design techniques for field programmable gate arrays." In *Proceedings of the 2004 ACM/SIGDA 12th international symposium on Field programmable gate arrays*, pp. 23–30. ACM, 2004.

[RPD07] Leonard Rockett, Dinu Patel, Steven Danziger, Brian Cronquist, and JJ Wang. "Radiation hardened FPGA technology for space applications." In *Aerospace Conference, 2007 IEEE*, pp. 1–7. IEEE, 2007.

[RRV02] Maurizio Rebaudengo, M Sonza Reorda, and Massimo Violante. "Simulation-based analysis of SEU effects on SRAM-based FPGAs." In *Field-Programmable Logic and Applications: Reconfigurable Computing Is Going Mainstream*, pp. 607–615. Springer, 2002.

[Sak02] Takayasu Sakurai. "Minimizing power across multiple technology and design levels." In *Proceedings of the 2002 IEEE/ACM international conference on Computer-aided design*, pp. 24–27. ACM, 2002.

[SAS10a] Sudipta Sarkar, Anubhav Adak, Virendra Singh, Kewal Saluja, and Masahiro Fujita. "SEU tolerant SRAM for FPGA applications." In *Field-Programmable Technology (FPT), 2010 International Conference on*, pp. 491–494. IEEE, 2010.

[SAS10b] Sudipta Sarkar, Anubhav Adak, Virendra Singh, Kewal Saluja, and Masahiro Fujita. "SEU tolerant SRAM for FPGA applications." In *Field-Programmable Technology (FPT), 2010 International Conference on*, pp. 491–494. IEEE, 2010.

[SB04] Sunil Shukla and Neil W Bergmann. "Single bit error correction implementation in CRC-16 on FPGA." In *Field-Programmable Technology, 2004. Proceedings. 2004 IEEE International Conference on*, pp. 319–322. IEEE, 2004.

[SBT91] Hamid Savoj, Robert K Brayton, and Hervé J Touati. "Extracting local don't cares for network optimization." In *Computer-Aided Design, 1991. ICCAD-91. Digest of Technical Papers., 1991 IEEE International Conference on*, pp. 514–517. IEEE, 1991.

[SCW09] Alastair M Smith, George A Constantinides, S Wilton, and P Cheung. "Concurrently optimizing FPGA architecture parameters and transistor sizing: Implications for FPGA design." In *Field-Programmable Technology, 2009. FPT 2009. International Conference on*, pp. 54–61. IEEE, 2009.

[SRK04] Praveen K Samudrala, Jeremy Ramos, and Srinivas Katkoori. "Selective triple modular redundancy (STMR) based single-event upset (SEU) tolerant synthesis for FPGAs." *Nuclear Science, IEEE Transactions on*, **51**(5):2957–2969, 2004.

[SV06] Luca Sterpone and Massimo Violante. "A new reliability-oriented place and route algorithm for SRAM-based FPGAs." *Computers, IEEE Transactions on*, **55**(6):732–744, 2006.

[TL03] Tim Tuan and Bocheng Lai. "Leakage power analysis of a 90nm FPGA." In *Custom Integrated Circuits Conference, 2003. Proceedings of the IEEE 2003*, pp. 57–60. IEEE, 2003.

[VSC04] Massimo Violante, Luca Sterpone, M Ceschia, D Bortolato, P Bernardi, M Sonza Reorda, and A Paccagnella. "Simulation-based analysis of SEU effects in SRAM-based FPGAs." *Nuclear Science, IEEE Transactions on*, **51**(6):3354–3359, 2004.

[WCJ98] Liqiong Wei, Zhanping Chen, Mark Johnson, Kaushik Roy, and Vivek De. "Design and optimization of low voltage high performance dual threshold CMOS circuits." In *Proceedings of the 35th annual Design Automation Conference*, pp. 489–494. ACM, 1998.

[WCW09] Lei Wang, Lei Chen, Zhiping Wen, Huabo Sun, and Shuo Wang. "A novel high-density single-event upset hardened configurable sram applied to fpga." In *Reconfigurable Computing and FPGAs, 2009. ReConFig'09. International Conference on*, pp. 1–5. IEEE, 2009.

[Wer07] Tomas Werner. "A linear programming approach to max-sum problem: A review." *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, **29**(7):1165–1179, 2007.

[Xila] "QPRO XQR4000XL Radiation Hardened FPGAs Datasheet." In *http://www.xilinx.com.*

[Xilb] "QPRO XQR4000XL Radiation Hardened FPGAs Datasheet." In *http://www.xilinx.com.*

[Xilc] "Virtex-7 FPGA Family Product Table." In *http://www.xilinx.com/publications/prod_mktg/Virtex7-Product-Table.pdf.*

[Xild] "Xilinx." In *http://www.xilinx.com/.*

[ZCW11] Xinrui Zhang, Liguang Chen, Liyun Wang, Jian Wang, and Jinmei Lai. "The design and verification of SEU-hardened configurable DFF." In *ASIC (ASICON), 2011 IEEE 9th International Conference on*, pp. 937–940. IEEE, 2011.

[Zhu07] Kai Zhu. "Post-route LUT output polarity selection for timing optimization." In *Proceedings of the 2007 ACM/SIGDA 15th international symposium on Field programmable gate arrays*, pp. 89–96. ACM, 2007.

[ZIO10] Qian Zhao, Yoshihiro Ichinomiya, Yasuhiro Okamoto, Motoki Amagasaki, Masahiro Iida, and Toshinori Sueyoshi. "A robust reconfigurable logic device based on less configuration memory logic cell." In *Field-Programmable Technology (FPT), 2010 International Conference on*, pp. 162–169. IEEE, 2010.

[ZP10] P. Zicari and S. Perri. "A Fast Carry Chain Adder for Virtex-5 FPGAs." In *MELECON*, pp. 304–308, 2010.