Applying Formal Verification to Early Assessment of FPGA-based Aerospace Applications: Methodology and Experience

Khaza Anuarul Hoque*, Otmane Ait Mohamed* and Yvon Savaria**

*Department of Electrical and Computer Engineering
Concordia University, Montreal, QC, Canada
**Department of Electrical Engineering
Polytechnique Montréal, Montreal, QC, Canada
Email: {k_hoque,ait}@ece.concordia.ca
yvon.savaria@polymtl.ca

***Abstract*—SRAM-based Field Programmable Gate Arrays (FPGAs) have been used in the aerospace application for more than a decade. Unfortunately, a significant disadvantage of these devices is their sensitivity to radiation effects that can cause bit flips in memory elements and ionisation induced faults in semiconductors, commonly known as Single Event Upsets (SEUs). An early dependability analysis on SRAM FPGA-based safety-critical application will enable the designers to develop a more reliable and robust design complying with design requirements, such as the DO-254 standard. We propose a methodology based on probabilistic model checking, to analyze the dependability and performability properties of such designs to guide design decisions. Probabilistic model checking is a well known formal verification technique, and the main advantage is that the analysis is exhaustive, which results in numerically exact answers to the temporal logic queries that contrast with discrete-event simulations. In the proposed methodology, starting from the high-level description of a system, a Markov (reward) model is constructed from the extracted Control Data Flow Graph (CDFG). Various dependability and performability related properties are then verified automatically using the PRISM model checker tool.**

I. Introduction

SRAM-based FPGAs offer increasingly attractive options for aerospace applications due to field programmability, the absence of non-recurring engineering costs, low manufacturing costs and so many other advantages. However, these devices are sensitive to Single Event Upsets (SEUs) [1]. Dependability (reliability, availability, and safety) and performability are major concerns in safety-critical applications that are common in the aerospace industry. An early analysis of dependability and performability impacts due to SEU allow the designers to develop more reliable and efficient system, and may reduce the overall cost associated with the design effort. Our work aims at achieving these goals.

In this paper, we propose a means by which formal verification techniques, in particular, probabilistic model checking [2] can be applied at early design stages to analyze the dependability and performability of such reconfigurable systems. The aim of this paper is to present the overall methodology for high-level design analysis and to summarize our learning from recent modeling results.

Probabilistic model checking is widely used to verify the systems whose behavior is stochastic in nature. Probabilistic model checkers, for example, PRISM [3], have the ability to verify exact solutions for probabilistic properties in an automated manner. Probabilistic model checking is mainly based on the construction and analysis of a probabilistic model, typically a Markov chain or a Markov process. These models are constructed in an exhaustive fashion, and hence these models explore all possible states that might occur in a system. This contrasts with discrete-event simulations, in which approximate results are generated by averaging results from a large number of random samples.

To analyze a design at high-level, we start from its Control Data Flow Graph (CDFG) [4] representation, obtained from a high-level description expressed using a language such as C++. Depending on the goal of analysis (the methodology currently can handle 3 different goals), possible failures of a system and their mitigation strategies are then modeled with the PRISM modeling language. The failure rates of the components are obtained from a worst-case component characterization library [5]. Given the design requirements, properties related to dependability and performability are automatically evaluated using the PRISM tool to guide design decisions. It is worth mentioning that, for this work, the assumptions are: (1) The SEU may only cause SBUs (Single Bit Upsets) in the design. (2) The design may employ periodic blind scrubbing. So when we mention scrubbing in this paper, we refer to periodic blind scrubbing [6]. (3) The systems are modeled using the concepts of Continuous-Time Markov chain (CTMC) (the *design option* analysis branch in the methodology uses a Markov Reward Model [7] which is also a variant of CTMC). (4) The target FPGA platform for our analysis is the Xilinx Virtex-5. This is the technology for which the available characterization library was produced.

The remainder of the paper is organized as follows. Section 2 reviews the preliminaries. Section 3 presents the state-of-the-art techniques and description of the proposed methodology. We present our learning, and experience from this project in section 5 and section 6 concludes the paper with future research directions.

II. Preliminaries

A. FPGA and Single Event Upsets

In a reconfigurable FPGA, the configuration memory is a collection of bits commonly known as a bitstream. Bitstream bits set the values of the LUT, flip-flop and memory initialization values, and states of switches and connection boxes that route signals through the FPGA. Therefore, interaction with high-energy radiated particles that are common in the aerospace environment, such as protons, neutrons, and heavy ions, may corrupt the FPGA configuration. The effects of these particles on electronics are collectively known as Single-Event Effects (SEE) and there are several types of SEE that are relevant to FPGAs. Single-Event Upsets (SEUs) occur when one or more bits in configuration memory changes state due to a radiation event. If only one bit is affected, then it is called a Single-Bit Upset (SBU). If more than one bit is affected, then it called an MBU.

B. Scrubbing and TMR

A scrubbing technique [6] is a single algorithm used in the system to mitigate configuration-memory upsets. A scrubbing strategy is composed of at least one fault correction technique and optionally, a fault detection technique. Blind scrubbing is a very popular scrubbing strategy with no detection technique and does not interrupt the system operation [1]. Scrubbing can be done at a specified rate meaning that there might be a period of time between the moment the upset occurs and the moment when it is repaired. That is why another form of mitigation is required, such as a redundancy-based solution known as TMR [8]. TMR is a technique for enhancing the reliability, in which each module in a circuit or the whole system is triplicated. A majority vote (two out of three) is taken on the TMR outputs to determine the final module output.

C. Probabilistic Model Checking

Model checking [9] is a well established formal verification technique to verify the correctness of finite-state systems. Given a formal model of the system to be verified in terms of labelled state transitions and the properties to be verified in terms of temporal logic, the model checking algorithm exhaustively and automatically explores all the possible states in a system to verify if the property is satisfiable or not. If not, a counterexample is generated. *Probabilistic model checking* deals with systems that exhibit stochastic behaviour, such as fault-tolerant systems. Probabilistic model checking is based on the construction and analysis of a probabilistic model of the system, typically a Markov chain. In this report, we focus on the continuous-time Markov chains (CTMCs) and Markov reward models [10], widely used for reliability and performance analysis.

A CTMC comprises a set of states S and a transition rate matrix $\mathbf{R} : S \times S \to \mathbb{R}_{\geq 0}$. The rate $\mathbf{R}(s, s')$ defines the delay before which a transition between states s and s' takes place. If $\mathbf{R}(s, s') \neq 0$ then the probability that a transition between the states s and s' might take place within time t can be defined as $1 - e^{-\mathbf{R}(s,s') \times t}$. No transitions will take place if $\mathbf{R}(s, s') = 0$. Exponentially distributed delays are suitable for modelling component lifetimes and inter-arrival times.

In the model-checking approach to performance and dependability analysis, a model of the system under consideration is required together with a desired property or performance/dependability measure. In case of stochastic modelling, such models are typically CTMCs, while properties are usually expressed in some form of extended temporal logic such as Continuous Stochastic Logic (CSL) [11], a stochastic variant of the well-known Computational Tree Logic (CTL) [9]. Below are a number of illustrative examples with their natural language translation:

1. $P_{\geq 0.95}[\Diamond\ complete]$ - "The probability of the system eventually completing its execution successfully is at least 0.95".

2. $shutdown \Rightarrow P_{\geq 0.81}[\neg\ fail\ U^{\leq 500}\ up]$ - "Once a shutdown has occurred, with probability 0.81 or greater, the system will successfully recover within 500 hours and without any further failures occurring".

Additional properties can be specified by adding the notion of rewards. Each state (and/or transition) of the model is assigned a real-valued reward, allowing queries such as:

$R = [\Diamond\ success]$ - "What is the expected reward accumulated before the system successfully terminates?".

Rewards can be used to specify a wide range of measures of interest, for example, the number of correctly delivered packets or the time that the system is operational. Of course, conversely, the rewards can be considered as costs, such as power consumption, expected number of failures, etc.

III. State-of-the-art and Proposed Methodology

There are three possible ways to analyze SEU sensitivity in FPGA based designs: 1) hardware testing such as particle beams and laser testing 2) fault injection emulation or simulation; and 3) analytical techniques. These three types of techniques are complementary, and they are applied at different product design steps. Hardware testing techniques are the most accurate of all. But such techniques require finished implementations. Also they may introduce physical damages to the device under test. Therefore they are very costly. Fault injection is a popular method. However, in fault injection test time grows with the number of possible test cases. In contrast, analytical methods tend to be relatively

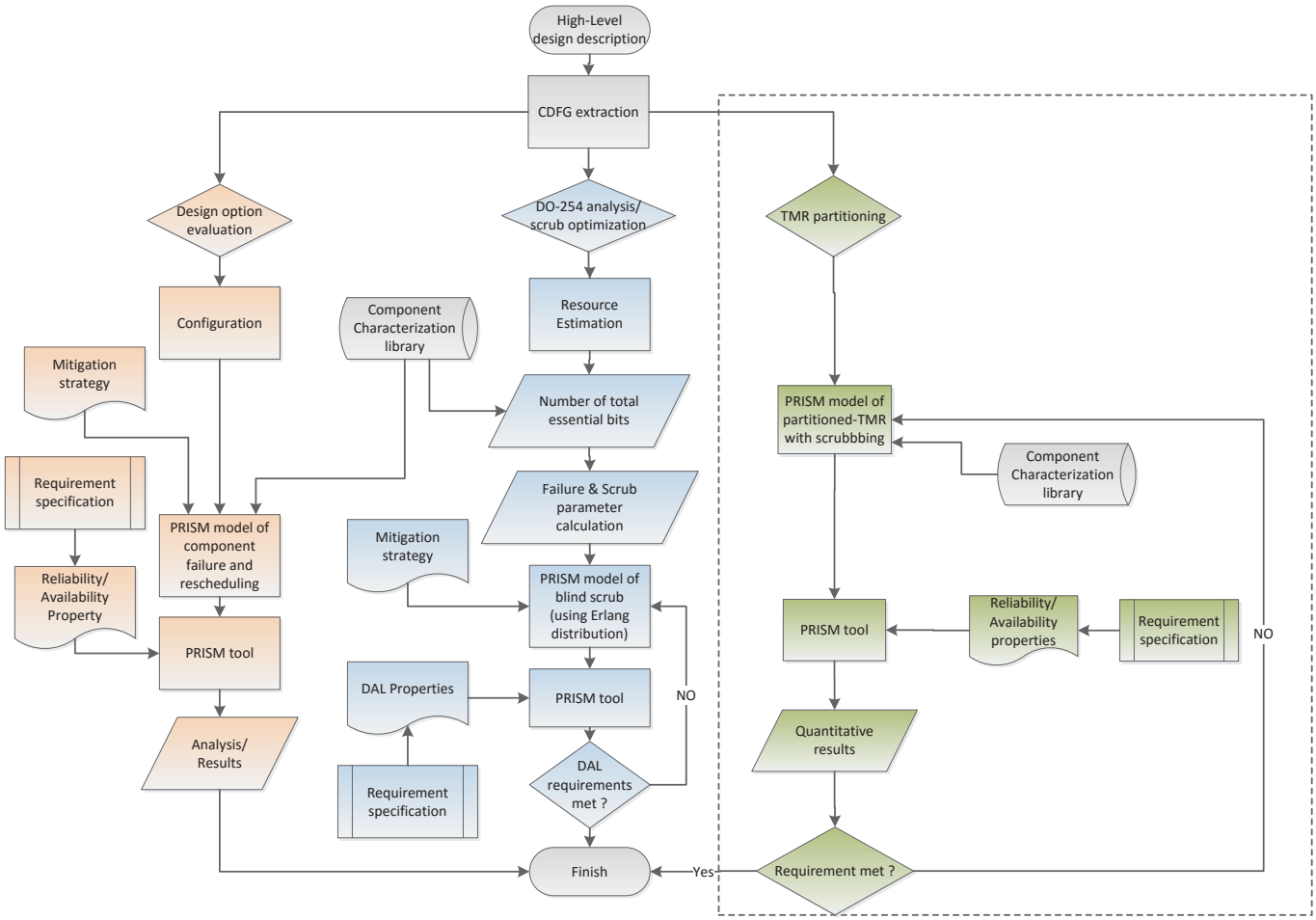

Fig. 1: Proposed Methodology for High-level Dependability Analysis

less accurate in some aspects. Still, they can provide a much better controllability and observability. Also analytical methods enable quick estimation of SEU susceptibility, with no risk of damaging the devices [12]. Moreover, they can capture features of the true test conditions that would be very hard to accurately reproduce when bombarding the circuit or while performing fault injection. Analytical estimation traditionally provides information at an earlier stage in the design cycle compared to the other two techniques.

Early analysis on a design can help designers to take decisions before finalizing an implementation, such as adopting proper mitigation technique(s) that will satisfy requirements and constraints. This may reduce the overall effort, time and cost of the design. In our proposed methodology, we address three issues in the SRAM-based FPGA design: (1) Design options analysis (system can be designed in different ways with different number of components allocated, and there is a need to decide on design options), (2) DO-254 [13] verification (Design Assurance Level analysis) with scrub optimization (where relevant questions relate to scrubbing parameters and frequency), and (3) TMR partitioning for reliability improvement (that can lead to optimizing the number of partitions in relation to the scrub frequency for instance). The first two issues were addressed as separate entities in our earlier works [14], [15]. The modeling and analysis of TMR partitioning is currently in progress and the initial results can be found in [16].

In contrast with previous contributions, in this paper, for the first time, we put all the branches of the method together and describe all parts from an overall perspective - as a single entity. Fig. 1 shows the proposed methodology. For ease of understanding the figure has been color coded. The gray boxes means that the information obtained from these boxes is shared between different branches of the methodology. The boxes in same colors mean that they belong to the specific branch, with similar color in the methodology. For example, the boxes in blue are part of the *DO-254 analysis/scrub optimization* branch. The rectangle (with dashed lines) represents the work-in-progress portion of the methodology. We start from the data flow graph of the application extracted from the high-level description of the design. Once the CDFG is extracted, depending on the target analysis, one of the three branches can be chosen. We describe the steps in the methodology as follows:

A. CDFG extraction:

The CDFG is used to represent the functionality of a selected C/C++ model, which is composed of arithmetic or logical operations and capable of representing all behaviors present in that algorithm. We use GAUT [17] to extract the CDFG from a high-level design description expressed using C/C++. The C/C++ model can be written by the designer or can be generated automatically in case the design is modeled in Matlab/Simulink.

B. Design options evaluation

In this branch of the methodology, we use probabilistic model checking to evaluate the trade-off between available *design options* in terms of dependability, performability, and area. The main intention is to demonstrate that in some cases, a redundancy-based solution might not be the best choice as one may expect. Alternatively, for those cases, rescheduling in conjunction with scrubbing can be a good option. High-level synthesis algorithms such as forced-directed list scheduling [18] can generate different CDFGs depending on components availability. Switching to another CDFG allows recovering from a failure while a system can continue its operation, possibly with a lower throughput. We model such a system with rescheduling, redundant components and scrubbing in PRISM and then analyze the results that support our claim. Steps of this branch of the methodology are described as follows; however, the detail can be found in [14].

C. Configuration:

The *configuration* of a system is the allocation of components that implements a CDFG. Each *design option* represents a *configuration*. So a *configuration* represents the initial set of available components before any failure has occurred. To analyze a *configuration*, we model it with the PRISM modeling language.

D. Characterization library:

As the upset rates λ is highly dependent on device process technology, architecture, and orbits of interest, so this parameter is different for each device family. We use CREME96 with radiation cross sections from [19] to find per bit upset rate λ_{bit} for Xilinx Vitex-5 in ISS HEO and LEO orbit. The failure rate for a component in the design (or the whole design) can be calculated using the equation: $\lambda_{design} = \lambda_{bit} \times Number\ of\ critical\ bits\ in\ the\ component(design)$. Note that we use the characterization library to obtain the failure rate of the components for the CTMC model, and the methodology is generic enough to be used with a different characterization library, with more precise and accurate data, without any major changes. The details of the characterization library used in our methodology can be found in [14], [5].

E. Modeling of dependability and performability:

For each of the design options (*configurations*), a Markov chain dependability model is built by including the possible component failures, fault coverage and possible recovery (rescheduling/scrubbing/redundancy). Using the high-level synthesis technique, each state of the CTMC model is augmented with associated performance and area rewards. The cumulative reward of this single Markov Reward Model (MRM[7]) is then used to evaluate the corresponding design option with respect to the metric of interest.

F. DO-254 analysis with scrub optimization:

More frequent scrub means fewer chances of accumulating SEUs. However, more frequent scrub requires more power, and many space missions set a need for low power requirement. Hence, in such cases, frequent scrubbing might not be a good option. This branch of the methodology is dedicated to verification of the DO-254 Design Assurance Level (DAL), and also the verification of high-availability requirements while optimizing the scrub frequency. We analyze the effect of scrub rate variation on the design to suggest the lowest possible scrub frequency that will meet the availability requirements and also to assess reliability enhancements that can be obtained with a suitably designed architecture leveraging techniques such as TMR. Detail of this part of the work can be found in [15]; however, steps of this branch are described as follows:

G. Resource estimation:

Resources estimation is based on the extracted CDFG. We analyze each of the nodes in the graph to compute resources required to implement an application on a specific FPGA target platform. Estimated resources depend on the style of implementation of the design. For example, a full parallel application of a CDFG will require a maximum number of resources with maximum speedup. However, depending on the area, performance and power requirement, the CDFG might require scheduling to deal with the constraints.

H. Failure and Scrub Parameter calculation:

Once the Markov model is built, we need to populate the model for further analysis. To calculate the failure rate we use the information from the *resource estimation* step and the information from the *characterization library* which states the number of configuration bits required to implement a specific resource in a target FPGA. Three different types of parameters are required, namely (1) Environmental parameters (2) Target system parameters and (3) Mitigation parameters. Details of these parameter estimations can be found in [15].

I. Modeling of Blind-Scrub and TMR using Erlang:

Maintenance actions such as scrubbing cannot generally be modeled by simple exponential distribution within the Markov process. For example, a significant repair or periodic inspection time does not follow the exponential distribution. Therefore, we have to develop an approximation methodology to allow the Markov processes to model significant holding times. The concept of a phase-type distribution [20], [21] can be used to approximate a time delay until absorption to one of the states in the Markov chain, commonly known as the Erlang process. In other words, non-exponential holding

time distributions can be approximated by inserting multiple intermediate states between the two conventional degradation states. We use Erlang distribution to model blind scrubbing in the FPGA using PRISM modeling language.

J. Optimization of TMR partitioning and scrubbing

The aim of this part of the methodology is to optimize the number of partitions in a TMRed system and scrub frequency depending on design requirement. Previous works in this area [22] address this issue either via fault injection or by assuming equal sized partitions [23]. In contrast, our methodology handles this at early design stages and can deal with both equal and non-equal sized partitions.

K. Modeling of Partitioned TMR and analysis:

Depending on the number of partitions defined by the user (the user can also choose the uneven size of each partition), each module (of partitioned TMR) in each partition may have one or more nodes (nodes in the CDFG represents a basic operation such as add, multiply, etc, so the number of nodes in each module defines the partition size, also known as window size). The *characterization library* provides us the estimate for the failure rate of each basic component. Using this approach, the failure rate calculation of each module becomes a straight forward approach, e.g. the failure rate of a TMR module in a specific partition is equal to the total failure rate of the components allocated to that module. Based on the failure rate of each module and the user defined scrub rate, a PRISM model is then built. The PRISM model checker then automatically converts the PRISM model to equivalent CTMC representation. Different reliability and availability properties are then verified to check if the design meets the reliability and availability requirements given the number of partitions with a scrub rate. The PRISM model checker provides the quantitative results. If the requirements are not met, the number of partitions or the scrub frequency is then modified, and the analysis is performed again. By analyzing the resulted graphs from the analysis, the designer can choose the appropriate number of partitions with adequate scrub frequency to meet the design requirements (the decision is of course design specific).

IV. Discussion

A. Lessons learned

We have learned some interesting lessons from the application of our proposed methodology. We evaluated four different design options in [14]. We observed that, from the reliability, availability and safety point of view, the redundancy-based solution works well. However, we also notice that extra reliability provided by the redundancy is not always useful to suppress the additional area overhead. If the designer is aimed to optimize overall reliability, area, and performance, then in such cases rescheduling with scrubbing is good enough to serve as a fault recovery and repair mechanism. This is an important observation since it guides the designer to choose a proper design option depending on his needs. DO-254 verification and Scrub optimization branch showed how to verify the high-availability requirements and the design assurance level compliance at high-level while minimizing the scrub frequency. Modeling results revealed that it is possible to find an appropriate scrub interval (slowest scrub rate) to save power while meeting the dependability requirements instead of choosing the highest scrub frequency. Note that, frequent scrubbing requires more power. That is why optimization of scrub for power-aware applications (such as deep space missions) plays a major role in the design. We also analyzed some early results that indicate how TMR partitioning and scrubbing trade-off can be evaluated at early design stages. A major observation from this analysis is when the scrub interval is smaller (frequent scrub), the number of partitions plays a major role increasing the reliability of a system. However, even for a delayed scrub, the improvement is noticeable enough. The more the number of partitions (which means smaller modules), the less frequent scrub will be required to meet a target reliability. The fewer number of partitions (larger module size) will require more frequent scrub to meet a target reliability requirement. For availability, the number of partitions is important for the cases where the scrub interval is long. For the case of frequent scrubbing, the number of partitions increases the availability to a minimal level, but for longer scrub intervals the availability improvement with the increased number of partitions is quite significant. All these three different kinds of analysis using our methodology can provide important guidelines to the designer at an early design stages. Such guidelines can help the designer not only to adopt proper mitigation (depending on design requirement) but also will add more confidence in the design.

B. Scalability

To demonstrate the applicability of our approach, we already applied our methodology on a DSP benchmark circuits [15], [14]. PRISM model checker includes multiple model checking engines, many of which are based on symbolic implementations (using binary decision diagrams and their extensions). These engines enable the probabilistic verification of models of up to 10^{10} states. PRISM also features a variety of advanced techniques such as abstraction refinement and symmetry reduction. It is worth mentioning that it also supports approximate/statistical model checking through a discrete event simulation engine. So considering the capability PRISM model checker, it is also possible to analyze larger systems using our methodology. However, since the PRISM modeling from CDFG is not fully automated, this restricts us to do so. In the future, we will work to overcome this limitation.

V. Conclusion

We presented a methodology for high-level dependability analysis of SEU sensitive designs. Our analysis based on the proposed methodology is capable of evaluating a design at an early stage regarding the required scrub interval, the number of TMR partitions and also can evaluate design alternatives via rescheduling. The first two part of the methodology presented

in this paper has already been implemented successfully [14], [15], and the contribution regarding the TMR partitioning is still in progress. For the future work, we would like to extend the methodology to analyze adaptive mitigation techniques, e.g. runtime adoption of different mitigation schemes depending on the predicted failure rate.

VI. Acknowledgments

This research work is a part of the AVIO-403 project financially supported by the Consortium for Research and Innovation in Aerospace in Quebec (CRIAQ), Fonds de Recherche du Québec - Nature et Technologies (FRQNT) and the Natural Sciences and Engineering Research Council of Canada (NSERC). The authors would also like to thank Bombardier Aerospace, MDA Space Missions and the Canadian Space Agency (CSA) for their technical guidance and financial support.

References

[1] M. Berg, C. Poivey, D. Petrick, D. Espinosa, A. Lesea, K. Label, M. Friendlich, H. Kim, and A. Phan, "Effectiveness of Internal Versus External SEU Scrubbing Mitigation Strategies in a Xilinx FPGA: Design, Test, and Analysis," *Nuclear Science, IEEE Transactions on*, vol. 55, no. 4, pp. 2259–2266, 2008.

[2] C. Baier, J.-P. Katoen *et al.*, *Principles of model checking*. MIT press Cambridge, 2008, vol. 26202649.

[3] M. Kwiatkowska, G. Norman, and D. Parker, "PRISM 4.0: verification of probabilistic real-time systems," in *Computer aided verification*. Springer, 2011, pp. 585–591.

[4] K. Kavi, B. Buckles, and U. N. Bhat, "A formal definition of data flow graph models," *Computers, IEEE Transactions on*, vol. C-35, no. 11, pp. 940–948, 1986.

[5] C. Thibeault, Y. Hariri, S. R. Hasan, C. Hobeika, Y. Savaria, Y. Audet, and F. Z. Tazi, "A library-based early soft error sensitivity analysis technique for SRAM-based FPGA design," *J. Electronic Testing*, vol. 29, no. 4, pp. 457–471, 2013.

[6] A. S. C. Carmichael, M. Caffrey, "Correcting Single-Event Upsets through Virtex Partial Configuration, Xilinx corporation, 2010."

[7] V. G. Kulkarni, *Modeling and analysis of stochastic systems*. London, UK, UK: Chapman & Hall, Ltd., 1995.

[8] C. Carmichael, "Triple module redundancy design techniques for virtex FPGAs (XAPP197 v1.0.1), Xilinx corporation, 2006."

[9] E. M. Clarke, E. A. Emerson, and A. P. Sistla, "Automatic verification of finite-state concurrent systems using temporal logic specifications," *ACM Transactions on Programming Languages and Systems*, vol. 8, pp. 244–263, 1986.

[10] W. J. Stewart, *Introduction to the numerical solution of Markov Chains*. Princeton University Press, 1994.

[11] C. Baier, J.-P. Katoen, and H. Hermanns, "Approximate symbolic model checking of continuous-time markov chains (extended abstract)," 1999.

[12] L. Sterpone, M. Violante, R. Sorensen, D. Merodio, F. Sturesson, R. Weigand, and S. Mattsson, "Experimental Validation of a Tool for Predicting the Effects of Soft Errors in SRAM-Based FPGAs," *Nuclear Science, IEEE Transactions on*, vol. 54, pp. 2576–2583, Dec 2007.

[13] P. S. Miner, V. A. Carreño, M. Malekpour, and W. Torres, "A case-study application of RTCA DO-254: design assurance guidance for airborne electronic hardware," in *Digital Avionics Systems Conference, 2000. Proceedings. DASC. The 19th*, vol. 1. IEEE, 2000, pp. 1A1–1.

[14] K. A. Hoque, O. Ait Mohamed, Y. Savaria, and C. Thibeault, "Early Analysis of Soft Error Effects for Aerospace Applications Using Probabilistic Model Checking," in *Formal Techniques for Safety-Critical Systems*, ser. Communications in Computer and Information Science. Springer International Publishing, 2014, vol. 419, pp. 54–70.

[15] ——, "Probabilistic model checking based DAL analysis to optimize a combined TMR-blind-scrubbing mitigation technique for FPGA-based aerospace applications," in *Formal Methods and Models for Codesign, 2014 Twelfth ACM/IEEE International Conference on*, Oct 2014, pp. 175–184.

[16] K. A. Hoque, "Early Dependability Analysis of FPGA-Based Space Applications Using Formal Verification," Ph.D. dissertation, Concordia University, 2016.

[17] P. Coussy, C. Chavet, P. Bomel, D. Heller, E. Senn, and E. Martin, "GAUT: A high-level synthesis tool for DSP applications," in *High-Level Synthesis*, P. Coussy and A. Morawiec, Eds. Springer Netherlands, 2008, pp. 147–169.

[18] P. G. Paulin and J. P. Knight, "Force-directed scheduling for the behavioral synthesis of ASICs," *Computer-Aided Design of Integrated Circuits and Systems, IEEE Transactions on*, vol. 8, pp. 661–679, 1989.

[19] H. Quinn, K. Morgan, P. Graham, J. Krone, and M. Caffrey, "Static proton and heavy ion testing of the Xilinx Virtex-5 device," in *Radiation Effects Data Workshop, 2007 IEEE*, vol. 0, July 2007, pp. 177–184.

[20] T. Osogami and M. Harchol-Balter, "Closed form solutions for mapping general distributions to quasi-minimal ph distributions," *Performance Evaluation*, vol. 63, no. 6, pp. 524–552, 2006.

[21] K. A. Hoque, O. Ait Mohamed, and Y. Savaria, "Towards An Accurate Reliability, Availability and Maintainability Analysis Approach for Satellite Systems Based on Probabilistic Model Checking," in *Design, Automation, and Test in Europe*. IEEE, 2015.

[22] F. L. Kastensmidt, L. Sterpone, L. Carro, and M. S. 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*. IEEE Computer Society, 2005, pp. 1290–1295.

[23] B. H. Pratt, M. P. Caffrey, D. Gibelyou, P. S. Graham, K. Morgan, and M. J. Wirthlin, "TMR with More Frequent Voting for Improved FPGA Reliability," in *ERSA*, 2008, pp. 153–158.