Efficient Probabilistic Fault Tree Analysis of Safety Critical Systems via Probabilistic Model Checking

Marwan Ammar*, Ghaith Bany Hamad†, Otmane Ait Mohamed* and Yvon Savaria†
*Concordia University, Montréal, Canada
Email: {m_amma, ait}@ece.concordia.ca
†École Polytechnique Montréal, Montreal, Canada
Email:{ghaith.bany-hamad, yvon.savaria}@polymtl.ca

***Abstract*—The cost and complexity involved in the development of critical systems encourage the use of reliability assessment techniques as early in the design cycle as possible. Existing techniques often lack the capacity to perform a comprehensive and exhaustive analysis on complex redundant architectures, leading to less than optimal risk evaluation. This paper addresses these weaknesses by 1) proposing a new probabilistic modeling of Fault Tree gates and their composition as Markov Decision Processes; 2) developing a new formal-based technique to perform an in-depth verification of the system's reliability. This technique makes use of the expressiveness of fault trees and the power of probabilistic model checking in order to investigate the best Triple Modular Redundancy partitioning and configuration of a system. The presented approach greatly improves the overall scalability with respect to other techniques, while also improving the accuracy of the results. For example, we can provide probabilistic failure rates for a chain of 100 redundant components in little over one second.**

I. Introduction

Ensuring the proper functionality of safety-critical systems is of utmost importance and, as such, dealing with faults at early stages of development has become increasingly important. To this end, many techniques for fault analysis and fault tolerance have been developed. In this context, Fault Trees (FTs) have gained widespread acceptance for quantitative safety analysis. FTs are top-down graphical representations of various combinations of lower level events that may cause the system to reach a top level failure (i.e., system failure). Fault Tree Analysis (FTA) can provide insightful information to designers regarding the reliability of their systems, such as how is their system most likely to fail and what are the most efficient ways to make it safer. Traditionally, FTA is performed by converting the system's FT into a Boolean function and simulating that function with different low level component failure rates [1], [2], [3], [4], [5], [6], [7]. However, these approaches are costly in time and resources. This is mainly due to the fact that their modeling of FT is limited to the Boolean representation, which may exponentially increase the resource requirements to reach the desired results.

Redundant architectures exploiting Triple Modular Redundancy (TMR) are broadly used to obtain fault tolerant safety-critical systems. In the literature, the utilization of formal methods to analyze these architectures is rather limited. In [8] the behavior of a single TMR is formalized through Communicating Sequential Processes (CSPs). The work in [9] uses Uppaal model checker to analyze a timed automata model of a TMR system design. Both techniques are limited to a single system and do not consider multi-staged TMRs. In [10], [11], the fault tree of a redundant system is modeled and analyzed through Satisfiability Modulo Theories (SMTs), giving an estimation of the reliability gain across different TMR configurations. The main problem of these approaches is the amount of redundancy required to perform the analysis. Furthermore, their proposed modeling has the same limitation as previous approaches because they also rely on boolean representation.

Due to the increased cost and physical size constraints, the main challenge of implementing fault tolerance through TMRs is knowing where in the system to apply the redundancy and knowing which TMR configuration is the most beneficial to the overall reliability. In order to address the above mentioned issues, a new formal probabilistic FTA methodology is proposed. This work is distinct in the following ways:

1) A new probabilistic modeling of FTs is introduced. This approach consists of building a library of FT gates, constructed by modeling the behavior of each FT gate as a probabilistic automaton (PA). Subsequently, a Markov Decision Process (MDP) model of a system's FT is be generated through the parallel composition, by instantiating and connecting the gates that compose that FT. 2) An iterative formal probabilistic analysis of the MDP model of the FT is proposed. In this analysis, a Component Contribution Investigation (CCI) is performed. The CCI consists in the evaluation of the contribution of the failure of each subcomponent to the systems failure. Through our iterative analysis, we identify the most critical components in the system, and the failure rate mitigation obtained from applying different types of TMR to those components.

The proposed methodology is experimentally evaluated on different system architectures, including the ones analyzed in [10], [11], demonstrating clear advantages. Our methodology is more scalable and provides approximately 70-fold speed-up compared to [11]. For example, the technique proposed in [10] fails to analyze a chain of 10 TMR components. The technique proposed in [11] is able to analyze a chain of 140 TMR components in 110 seconds, while our approach can analyze the same chain in less than 1.5 seconds. Our methodology significantly widens the possible analyses of TMR architectures. It also allows evaluating the best TMR

configurations (like in [10], [11]) and to determine the optimal TMR partitioning as well as the impact of TMR at different FT levels.

The rest of this paper is structured as follows. In Section II, the most relevant related works are discussed. In Section III, we present the proposed methodology. In Section IV, we describe the experiments validating the proposed method and report the main results that were obtained. In Section V, we draw some conclusions and discuss possible future work.

II. Related Work

Fault tree analysis (FTA) is a widely used method for risk assessment, mainly in the area of avionics, nuclear, and chemical industries [1]. FTA follows a deductive approach that allows finding the origins of undesirable events. In the context of FTA, a general event is known as the *top level event* (TLE), from which the fault tree branches out vertically. The top event is defined as the failing point of a system in operating conditions, whether those conditions are considered normal or abnormal. Basic events are the ones at the very bottom of the fault tree, called *low level events* (LLEs), independent from any other events. A quantitative FTA is performed by assigning fault probabilities to the bottom events which will dynamically be distributed through the rest of the tree. The relationship between fault tree events is expressed using relational constructs called *logic gates*, such as *AND* and the *OR* gates.

In [4], [12] quantitative and qualitative FTA techniques are proposed, respectively. These techniques consist of converting the FT into a format compatible with Shannon's decomposition (i.e., boolean formulas). Thereafter, the proposed analysis is performed on the FT boolean formula to generate fault configurations that can cause system failure. The problem with these approaches lies in the boolean representation which limits their expressiveness of the system's behavior. For instance, it is not possible to trace the source of the error without performing step-by-step comparison between the faulty and the reference models of the system. The works in [5] and [7] focus on the analysis of dynamic FTs. The FTs are represented as cause-effect diagrams known as *Bayesian networks*. The analysis was performed with the Galileo tool [13], which performs FTA with the DIFTree algorithms.

A model-based safety assessment approach to analyze redundant components is proposed in [10]. In this technique, SMT solvers are utilized to generate all fault configurations that can lead to a system failure. This technique is an improvement (in terms of speed and accuracy) over traditional reliability analyses [14], [8], which utilize manual algorithms. However, the problem with the approach proposed in [10] lies in the overhead to generate the model. This is mainly due to the need to incorporate a reference and a redundant model of each subcomponent. Moreover, the redundant model of the subcomponents must be duplicated, with one added multiplexer between all duplicated items, to allow fault injection. Thus, this modeling approach is very costly, with serious scalability issues due to the complexity and size of the final SMT model. According to [10], with such modeling approach, even the most efficient SMT solvers cannot handle a chain of more than 10 TMR components.

An attempt to address the scalability issues of [10] is presented in [11], where the complexity is reduced through predicate abstraction. This technique replaces the reference and the redundant models with their abstracted versions to allow larger systems to be modeled. However, even if the behavior of the models is preserved, another problem arises with [10], [11]. Generating all possible fault configurations that may lead to system failure is impractical. According to [10], a chain of 100 TMR components can have $2^{6\times100}$ possible fault configurations. An SMT solver can only generate a subset of these possible fault configurations (i.e. the obtained system failure rate will be underestimated).

III. Proposed Method

In this section, the proposed probabilistic fault tree analysis of redundant architectures is introduced. The flow chart of the proposed method is shown in Fig. 1. We start from a system-level model, in which a system is composed of interconnected hardware components. This system-level model must be provided by the designer in a general-purpose modeling language, such as SysML [15]. The failure rate of each component is characterized from the system-level specification. From the SysML model, a fault tree of the system is automatically synthesized using the tool proposed in [16]. Subsequently, a formal MDP model of the system's FT is obtained through the parallel composition of the PAs of the FT gates. A library of the PAs of the FT gates is modeled a priori and each gate can be instantiated as needed. In this work, the composition of the MDP model of the FT is done automatically through PRISM [17]. The next step is to exhaustively analyze the generated MDP model in order to evaluate the maximum probability of the TLE (i.e., compute the system's failure rate), in the presence of all LLEs. This is done by performing probabilistic model checking of the following property: *"What is the maximum probability that eventually TLE will be true (i.e. system will fail)"*. According to the specification and based on the criticality of the system, a threshold for an acceptable failure rate is defined. If the computed failure rate is above the expected threshold, then an exhaustive analysis of the fault tree is required. In this case, we perform a Component Contribution Investigation (CCI), which characterizes the impact of the failure of each subcomponent on the system failure. This is done by verifying the following property: *"What is the maximum probability that if component A fails, the TLE will eventually fail"*

For each system, a Component Contribution Matrix (CCM) is created. The CCM is a library that records the impact of the failure of each subcomponent to the system's failure probability. Subsequently, the proposed methodology investigates the effects of applying TMR on the critical components (based on the system's CCM). If the system's failure rate is above a desired threshold, the CCI is repeated and the CCM is updated. The previous steps are repeated until the system satisfies its

specification. In the next subsections, we explain in detail the proposed modeling and analysis.

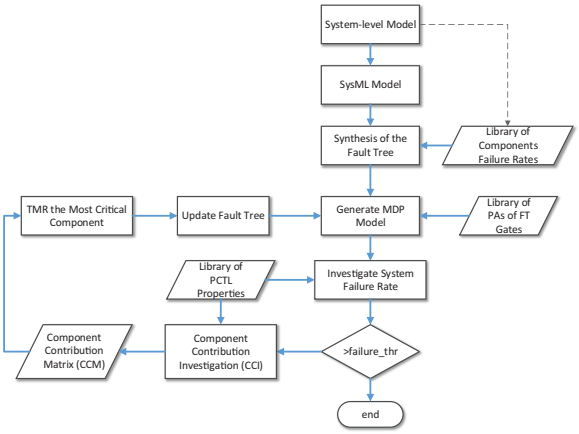


Fig. 1: Main Steps of the Proposed Method

A. Probabilistic Modeling of Fault Trees

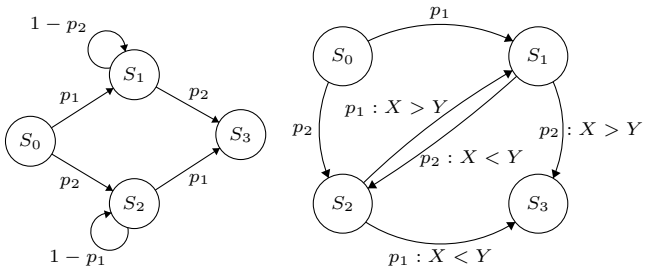
In fault trees, the set of events that might lead to a top level event failure is defined in a top-down manner and connected using FT gates. In order to accurately model the fault dependencies in FT, our approach focuses on the formalization and modeling of the probabilistic behavior of FT gates. The general probabilistic model (automaton) of any FT gate is expressed in Definition 1.

Definition 1: Given a FT gate with a set of inputs *X* and an output *Z*, connected through a certain logic (such as AND, OR). The probabilistic automaton (PA) of this gate can be formally defined as a finite transition system as follows:

- S, a finite set of states;
- $\overline{s}$ is the initial state $\overline{s} = S_0$;
- $\delta \subseteq (S \times S)$, a set of transitions between the states;
- Fr, a set of components failure rates;
- $\pi : \delta(S \times S) \rightarrow (\lambda, 1-\lambda)$, a transition function assigning probability $\lambda \in Fr$ for each transition.

Example: Given two input events *X* and *Y* and their output *Z*, connected through an *AND* gate, output *Z* becomes true if and only if *X* and *Y* are true. Two examples of the PAs of one static and one dynamic gates are depicted in Fig. 2. For the static AND gate (see Fig. 2(a)), starting from an initial state S_0(X=0, Y=0, Z=0) the next state can either be S_1(X=1, Y=0, Z=0) or S_2(X=0, Y=1, Z=0), with probabilities *p1* and *p2*, respectively. At this point, the system can either move from S_1 or S_2 to S_3(X=1, Y=1, Z=1), with probability *p2* or *p1*, respectively, signifying fault propagation, or stay in S_1 or S_2, with probabilities *1-p1* or *1-p2*.

A dynamic gate is modeled in the same manner, but with an added degree of complexity due to the priority constraint. A *Priority AND* (*PAND*) gate model (see Fig. 2(b)) starts at state S_0(X=0, Y=0, Z=0). The model may move to S_1(X=1, Y=0, Z=0) with probability *p1*, or to S_2(X=0, Y=1, Z=0) with probability *p2*. If the model is at S_1 and the priority of *X* is higher than the priority of *Y*, and event *Y* happens, the next state will be S_3(X=1, Y=1, Z=1), with probability *p2*. If the model is at S_1 and the priority of *Y* is higher than the priority of *X*, and event *Y* happens, then the next state will be S_2, with probability *p2*. Lastly, if the model is at state S_2 and the priority of *Y* is higher than the priority of *X*, and event *X* happens, the model moves to S_3, with probability *p1*. If the model is at S_2 and the priority of *X* is higher than the priority of *Y*, and event *X* happens, the model will move to state S_1 with probability *p1*.



(a) 2-input AND gate (b) 2-input PAND gate

Fig. 2: Example of Fault Tree Gate Automata

Due to space constrains, only the PAs of the *AND* and *PAND* gates are shown in Fig. 2. However, a list of all the FT gates that we have modeled and a brief description of their behavior can be seen in Table I. After the PAs of the gates in the FT are modeled, the MDP of the FT is constructed. In this work, the MDP definition developed in [18] is adapted, in Definition 2.

Definition 2: An MDP is defined as a finite transition system $(S, \overline{s}, A, P, L, \textbf{Steps})$ as follows:

- S, a finite set of states;
- $\overline{s}$ is the initial state $\overline{s} = S_0$;
- A is a set of actions (i.e., LLEs);
- AP is a set of atomic propositions;
- $L : S \rightarrow 2^{AP}$ is a labelling function that provides, to each state $s \in S$, a set $L(s)$ of atomic propositions;
- **Steps** : $S \times Act \rightarrow Dist(S)$ is the (partial) transition probability function, with Dist(S) denoting the set of all discrete probability distributions over S.

The MDP, as defined previously, is a stochastic system where all the decisions are made in a non-deterministic manner. At each state, a non-deterministic choice is done from a finite set of possible actions by parallel synchronization ($\|_S$) of the PAs ($M_{MDP} = \{PA_1 \|_S PA_2 \|_S \dots \|_S PA_n\}$). Subsequently, the function **Steps** randomly chooses the successor state according to the probability distribution **Steps**(s, a).

The choice of MDP over other types of Markov models, such as the Discrete-Time Markov Chain (DTMC), lies in the fact that real life aerospace systems may experience nondeterministic behavior due to the concurrency between components operating in parallel. Concurrency can be found when an event leads to several different errors (i.e., impacts several subcomponents) in the system. A DTMC model, in this case, divides the probability of failure equally over all

the affected subcomponents. However, this behavior may not be realistic. The MDP implementation of the aforementioned event can result in a completely random distribution choice, that could reflect more accurately a real environment. An example of this can be seen in Fig. 3, where the effects of non-deterministic (i.e., MDP) versus probabilistic (i.e., DTMC) distribution on the TLE failure rate ($1 - R_{sys}$) of the solar array system of the DHF-3 satellite [19] is investigated. The solar array FT can be seen in Fig. 3(a). It can be observed from the results (shown in 3(b)) that the choice of model (MDP or DTMC) does not influence the computed failure rates when the events only affect a single subcomponent (e.g., *X2* and *X7*). On the other hand, an event that affects several subcomponents (e.g., event *X3* directly affects subcomponents *X15*, *X17*, *X18*, and *X21*) will have its impact on the system's failure underestimated in the DTMC model, as demonstrated in [20].

It is important to mention that our probabilistic modeling approach takes full advantage of the concept of FT modularity [1], which means that a fault tree of a big system can be broken down into smaller sub trees. These smaller fault trees can then be analyzed separately, and the result of this analysis is fed back to the original FT.

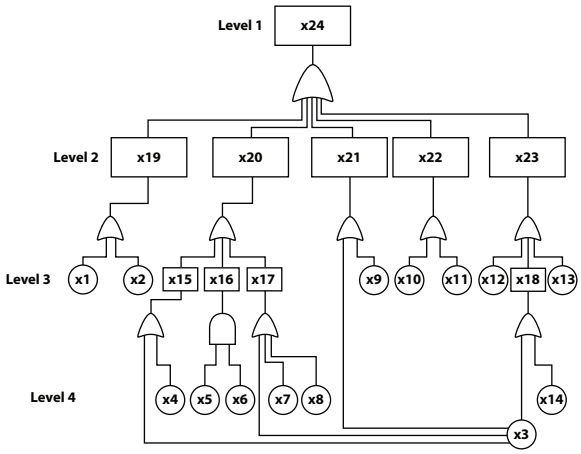
B. Modeling of TMR

The assessment of the impact of the optimal redundant architecture configuration is of great importance to the development of safety critical systems. Redundancy is commonly applied to components deemed essential to the system's correct functionality. TMR is a broadly diffused architectural pattern for component redundancy [21], [22], [23], [24], which consists in triplicating a critical component and feeding each copy's output to a majority voter. The voter evaluates each component's output and returns the value computed by the majority. Applying TMR to the whole system is very costly and might not be required based on the system's criticality level. Moreover, the efficiency of a TMR is dependent on the reliability of each component and TMR voter, as well as on the TMR configuration.

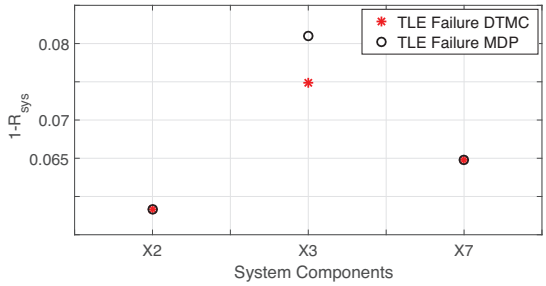
The proposed methodology determines the optimal TMR partitioning (where the TMR is required in the system) and the best TMR configuration to be used. Starting from identifying the best partition, the most critical component, according to the system's CCM, is singled out. Following this, the systems failure rate is evaluated for each TMR configuration. This process is repeated for all critical components until the desired system failure rate is achieved, as shown in Fig. 1.

TABLE I: Modeled FT Gates

Gate	OutputCondition
AND Gate	All inputs are true
OR Gate	At least one input is true
Inhibit Gate	In the presence of a trigger event, one other input must become true
Functional Dependency	When the trigger event occurs, the dependent basic events are forced to occur
Combination Gate	At least *n* inputs are true
Exclusive OR Gate	Exactly one input is true
Priority AND (Sequence Enforcing) Gate	The inputs must become true in a specific sequence
Spare Gate	The primary input becomes true, followed by the spare inputs, in sequence



(a) Solar Array Fault Tree



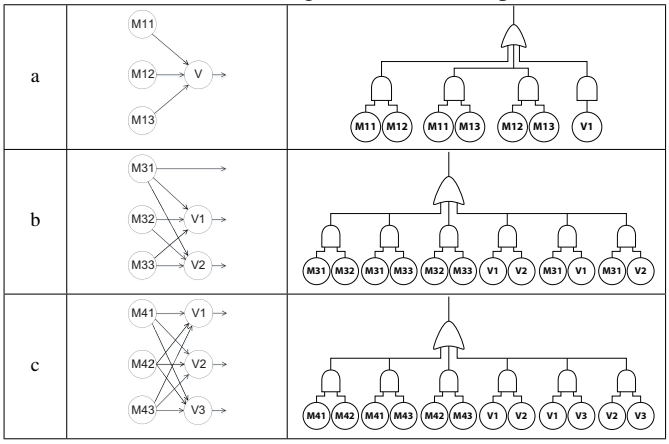
(b) MDP versus DTMC

Fig. 3: Modeling of FT as MDP Versus DTMC

Through our modeling, component redundancy can be applied in two ways: 1) the sub FT, of which the critical component is the TLE, is triplicated within the system FT and the new analysis is performed. 2) the sub FT of the critical component is extracted from the system FT, triplicated, and analyzed separately. The numerical results of the analysis are then fed back to the system FT, as *transfer in* events. The first way involves increasing the size of the MDP model and, consequentially, the state space. However, the second way reduces the size of the model and increases the efficiency of the verification, while retaining accuracy. This is achievable through FT modularity [1], combined with the proposed probabilistic modeling, which allows FTs to be divided into sub-FTs without losing any properties. Therefore, throughout this work, the second method is applied.

In this work, all TMR configurations were analyzed, but for space constraints we only present the most commonly used ones. Table II shows the main three TMR configurations and their equivalent fault trees. The 1-voter and 2-voters TMR configurations can have different sub-configurations depending on how these TMRs feed their output to the next stage.

TABLE II: TMR Configurations and Respective FTs



For instance, a 1-voter TMR can have its output driven by only the voter, or by a combination of the voter output and the copies of the component. The number of outputs of a TMR and their origin are taken into account when building the equivalent FTs. As can be seen in the 2-voters TMR example, element *M31* has more impact to the TLE than elements *M32* and *M33*, due to the fact that it is the singled-out element. Another modeling option that impacts the analysis of redundant systems is the uniformity of failure rates of the TMR copies of the component. Failure rates can be uniform or non-uniform, meaning that the triplicated components can have either the same or different failure rates. The same principle applies to TMR arrangement, where a 2-voters TMR chain can feature homogeneous or non-homogeneous arrangement of voters at different stages, as seen in Fig. 4. The variation in the system reliability due to all the above mentioned TMR design options is discussed in detail in Section IV.

IV. Experimental Results

The proposed methodology is fully automated on top of the well known PRISM probabilistic model checker, a free, open source probabilistic symbolic model checker, based on the reactive modules formalism [25]. The probabilistic automata of the fault tree gates (similar to Fig. 2) are modeled as PRISM modules. Each module is composed of a set of commands. A PRISM command is a tuple $cmd = (act, guard, rate, action)$ following the format `[<act>] <guard> → <rate>: <action>`, where:

- *act* is an action label used for synchronization of the different levels of an FT;
- *guard* is a predicate over the inputs of the FT gates;
- *rate* is the probability of occurrence of a FT event;
- *action* is a set of *n* updates that will translate into transitions in the FT.

Afterwards, an MDP model of the system is automatically generated by PRISM. The failure rate of the system is investigated by verifying a set of Probabilistic Computation Tree Logic (PCTL) properties over the MDP model. The maximum probability of a system failure can be obtained by verifying the following property:

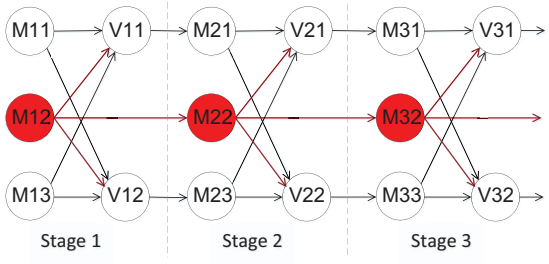
$$\textbf{PCTL 1}: P_{max} =? \ [(F \ TLE = 1)]$$

The contribution of the failure of a component to the system failure (i.e., CCI) is obtained by verifying the following property:

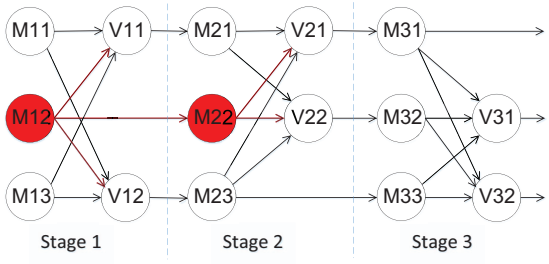
$$\textbf{PCTL 2}: P_{max} =? \ [(F \ comp = 1) \& \ (F \ TLE = 1)]$$

The experiments have been performed on a machine with an Intel Core I5-4200U CPU and 8 GB of RAM. To show the applicability of our approach, different sizes of TMR chains are analyzed. This experiment is done with chains of length *n*, with 1, 2 and 3 voters, considering different TMR combinations. The length of the chain varies from 10 to 100 components. Fig. 5(a) shows the CPU time consumed to construct the MDP model and to perform the proposed analysis. Due to the efficiency of the proposed modeling, we were able to analyze a chain of 100 TMR components in little over 1 second. Fig. 5(b) brings the memory consumption for chains of different sizes. Fig. 5(c) depicts the relationship between the total number of states in the model and the length of the chain. As evidenced by the results, the resource consumption is small and grows linearly with the size of the chain. For example, a chain of 100 components is verified in less than 1 second and consumes less than 0.02 MB of memory. This is achieved through our efficient modeling approach, which virtually eliminates the problem of state explosion [26].

The second experiment consists in investigating the impact of the TMR configurations on system reliability. The analyses are performed on networks of 10 components (after applying TMR) connected in series, where we vary the failure rate of components and voters. In this experiment, we consider



(a) Homogeneous 2-voter TMR Chain



(b) Non-homogeneous 2-voter TMR Chain

Fig. 4: Example of Different TMR Arrangements

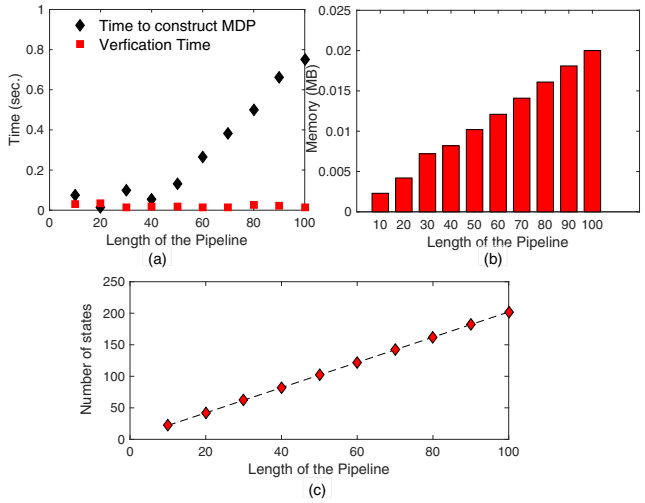


Fig. 5: Results of Investigating the Proposed Methodology's Scalability

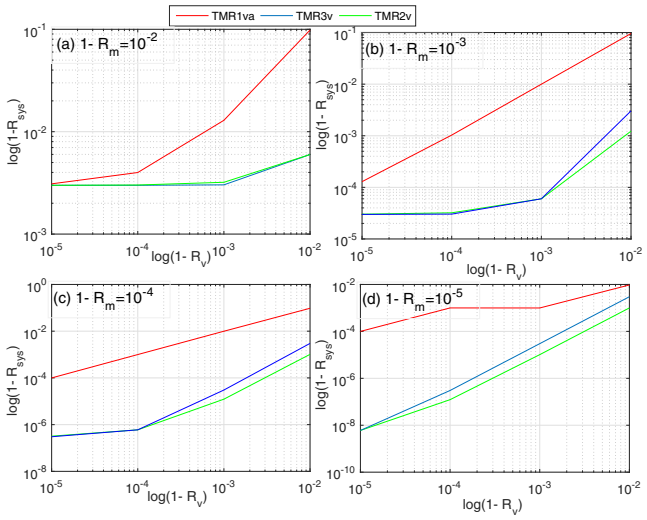


Fig. 6: TMR Chain with Uniform Failure Rates

a uniform failure distribution (i.e. the triplicated version of each component have the same failure rate). Moreover, it is assumed that all the components feature the same TMR configuration. The results, depicted in Fig. 6, lead to the following observations: 1) when the failure rate of the voter $(1-R_v)$ is high and the failure rate of the component $(1-R_m)$ is low (see Fig. 6 (c,d)), the best TMR configuration to improve system reliability $(1-R_{sys})$ is the one which employs two voters. This is mainly because that configuration reduces the impact of the failure of the voters by taking into consideration the low failure rate of the components. 2) When the failure rate of the voter is low and the failure rate of the components is high (see Fig. 6 (a)), the best TMR configuration is one voter, as the reliability provided is very similar to the other configurations but without the area and power overhead resultant from voter redundancy. From Fig. 6 (a,b) we can conclude that when the components have high failure rates, the performance of two and three voters configurations is similar, and they outperform one voter configuration as the failure rate of the voter increases.

We further investigate TMR chains, considering non-uniform failure distributions (i.e., the triplicated components have different failure rates). To perform this analysis, it is assumed that the triplicated components are physically separated, thus they may be affected differently by environmental conditions (i.e. extreme heat, radiation, etc.). The analysis performed consist of randomly increasing the failure rate of one component (treated as a critical component) within each TMR.

Through the analysis of Fig. 7 (TMR3v) it is evidenced that the TMR with three voters is the best choice for nearly all situations. This happens because all components are "shielded" by the voters at all stages of the chain, thus the occurrence of a component failure is always masked. Three-voter TMRs are, however, the most costly option in terms of area and power. From the results of our analysis of TMRs of two voters, depicted in Fig. 7 (TMR2v), it can be observed that the homogeneous chain of two-voter TMRs is particularly sensitive to non-uniform failure conditions, especially if the critical component is the singled out one. This is due to the fact that a failure in the singled out component feeds directly into the next singled out component until it reaches the system outputs and cause the system failure. This phenomenon can be seen in Fig. 4(a), where the failure of *M12* carries over to *M22*, and subsequently to *M32*, and so on. Our results demonstrate that this weakness can be mitigated with the use of non-homogeneous chains of two-voter TMRs. As seen in Fig. 4(b), the configuration of the voters is changed from stage to stage. A fault originated in *M12* propagates to *M22* but it is masked by the voters *V21* and *V22*. This is evidenced by Fig. 7 (TMR2v2), where significant improvement can be seen when compared to homogeneous chains.

Lastly, we perform analyses on the HSCOM subsystem of the HERMES Cubesat Satellite, responsible for primary high speed communications [27] and on the solar array system of the DFH-3 satellite. The fault tree of the HSCOM subsystem is shown in Fig. 8. Due to the lack of exact component failure rates in the design specification, all bottom events are assigned

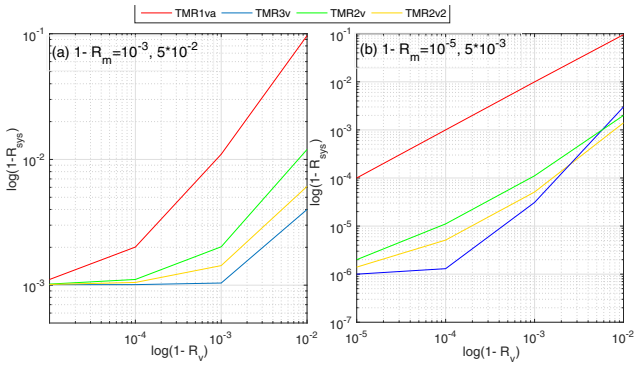


Fig. 7: Variation in the System Failure Rate due to Non-uniform Distribution

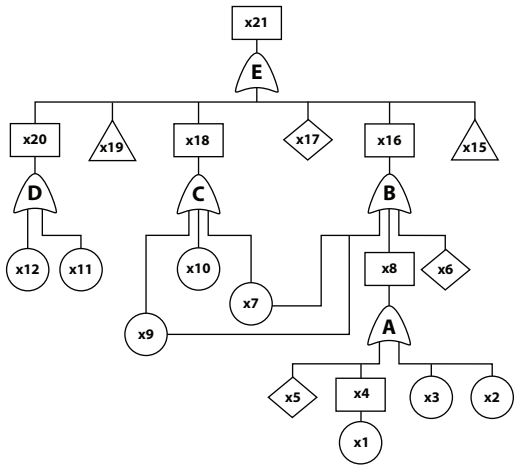


Fig. 8: Hermes Cubesat HSCOM Fault Tree

the same failure rate of 5.0×10^{-3}. *Transfer in* events (X15 and X19) have probabilities of failure equal to 1×10^{-2} each. Subsequently, a CCI is performed on the model and the results are reported in the CCM.

Fig. 9(a) presents a comparison of the impact of TMR (1 and 3 voters) on the different components of the system. It can be observed that the most critical component in the system is *X16*, so we further investigate it. *X16* is the top event of a sub FT composed by gates *B* and *A*. We will call the sub fault tree as *BA*. Upon further inspection on *BA*, it was observed that the most critical component in *BA* is *X8*. Thus a TMR is applied to *X8*. Fig. 9(b) contrasts the original system failure rate (no TMR) with the updated failure rate after applying TMR to each component in the system. To show the versatility of the approach, a similar set of experiments is performed on the solar array FT of the DFH-3 satellite, shown in Fig. 3(a). The solar array experiment follows the same steps described in the HSCOM experiment, with the difference that we use the actual failure rates of the FT components, obtained from [19]. A comparative result of the effectiveness of TMR on each component of the solar array is presented in Fig. 10(a). The impact of applying TMR on each component to the failure rate of the system is depicted in Fig. 10(b). It can be observed through the analysis that the impact of TMR on a multi-level structure FT is not straightforward, when compared to linear structures (chains of TMRs). This is mainly because the impact is not only dependant on the failure rates of components and voters, but also on the structure of the FT. This means that the same TMR configurations may have different levels of impact when applied to different FTs of different systems.

Through our analysis, it is also observable that within the same sub tree, triplicating lower level events is more efficient, compared to higher level events. This can also be seen in Fig. 9(b), where the impact of applying TMR on *X8* or on *X16* is practically the same. This is very insightful, as the area and power overhead resulting from triplicating *X8* is much smaller than the overhead generated from triplicating *X16* (see Fig. 8).

Throughout the experiments it is observed that although the

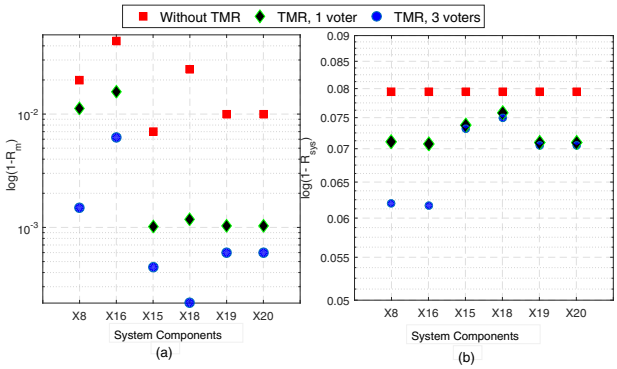


Fig. 9: Hermes HSCOM Results

implementation of TMR usually impacts the system positively, the choice of the best TMR configuration is heavily dependant on the characteristics of the system and on the environment where the system is deployed. Fig. 6 shows that, in controlled environments (i.e., the triplicated versions of the component have the same failure rates), two-voters TMR provides the best reliability. In unknown environments that might influence the system in unpredictable ways (i.e., the triplicated versions of the component have different failure rates), the three voters TMR configuration is always the best choice, assuming that the failure rate of the voters is always smaller than the failure rate of the components.

V. Conclusion

In this paper, an efficient probabilistic modeling and analysis of fault trees is proposed. A new probabilistic approach for modeling fault trees was developed. These model are used to construct the MDP model of system fault tree. Next, the efficient PMC (i.e., PRISM) is utilized to perform the proposed exhaustive analysis. The proposed technique is extended to perform an iterative probabilistic analysis of TMR architectures. This technique is used to investigate the best TMR partitioning, configurations, and the impact of implementing TMR at the different levels of the system's FT. Our results demonstrate that the proposed methodology has great potential as it is more scalable, orders of magnitude faster, and it provides better quality results when compared with contemporary techniques [10], [11]. For instance, the proposed technique

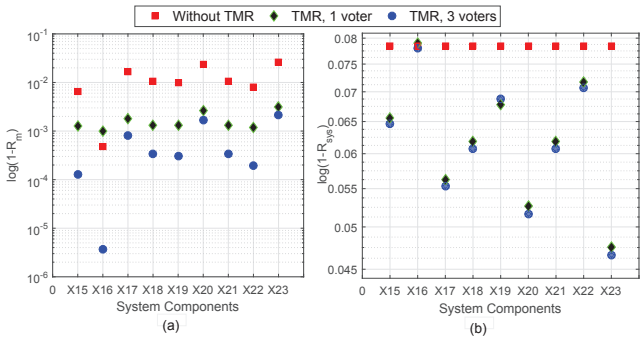


Fig. 10: Solar Array TMR Results

is able to analyze a chain of 140 TMR components in less than 1.5 seconds. Also, our proposed methodology provides a more accurate system failure rate estimation in the presence of TMRs. In contrast, other techniques ([10], [11]) can only investigate a subset of all possible fault configurations of large systems.

REFERENCES

[1] W. Vesely, F. Goldberg, N. Roberts, and D. Haasl, "Fault tree handbook (nureg-0492)," in *http://www.hq.nasa.gov/office/codeq/doctree/fthb.pdf*, 1981.

[2] K. Durga Rao, V. Gopika, V. Sanyasi Rao, H. Kushwaha, A. Verma, and A. Srividya, "Dynamic fault tree analysis using monte carlo simulation in probabilistic safety assessment," in *Reliability Engineering and System Safety Volume 94, Issue 4*, April 2009, p. 872883.

[3] M. Bozzano, A. Cimatti, and F. Tapparo, "Symbolic fault tree analysis for reactive systems," in *5th International Symposium on Automated Technology for Verification and Analysis (ATVA)*, November 2007, pp. 162–176.

[4] R. Sinnamon and J. Andrews, "Improved accuracy in quantitative fault tree analysis," in *Proceedings of the 12th Advances in Reliability Technology Symposium, Manchester, UK*, 1996.

[5] H. Boudali and J. Duga, "A new bayesian network approach to solve dynamic fault trees," in *Reliability and Maintainability Symposium, 2005. Proceedings. Annual*. IEEE, 2005, pp. 451–456.

[6] M. Bouissou and J.-L. Bon, "A new formalism that combines advantages of fault-trees and markov models: Boolean logic driven markov processes," *Reliability Engineering & System Safety*, vol. 82, no. 2, pp. 149–163, 2003.

[7] H. Boudali, P. Crouzen, and M. Stoelinga, "Dynamic fault tree analysis using input/output interactive markov chains," in *Dependable Systems and Networks, 2007. DSN'07. 37th Annual IEEE/IFIP International Conference on*. IEEE, 2007, pp. 708–717.

[8] T. Lanfang, T. Qingping, and L. Jianli, "Specification and verification of the triple-modular redundancy fault tolerant system using csp," in *DEPEND 2011, The Fourth International Conference on Dependability*, 2011, pp. 14–17.

[9] M. Zhang, Z. Liu, C. Morisset, and A. P. Ravn, "Design and verification of fault-tolerant components," in *Methods, Models and Tools for Fault Tolerance*. Springer, 2009, pp. 57–84.

[10] M. Bozzano, A. Cimatti, and C. Mattarei, "Automated analysis of reliability architectures," in *18th International Conference on Engineering of Complex Computer Systems (ICECCS)*, July 2013, pp. 198 – 207.

[11] ——, "Efficient analysis of reliability architectures via predicate abstraction," in *9th International Haifa Verification Conference (HVC)*, November 2013, pp. 279–294.

[12] R. Sinnamon and J. Andrews, "Improved efficiency in qualitative fault tree analysis," in *Proceedings of the 12th Advances in Reliability Technology Symposium, Manchester, UK*, 1996.

[13] K. J. Sullivan, J. B. Dugan, and D. Coppit, "The galileo fault tree analysis tool," in *Fault-Tolerant Computing, 1999. Digest of Papers. Twenty-Ninth Annual International Symposium on*. IEEE, 1999, pp. 232–235.

[14] M. Hamamatsu, T. Tsuchiya, and T. Kikuno, "On the reliability of cascaded tmr systems," in *Dependable Computing (PRDC), 2010 IEEE 16th Pacific Rim International Symposium on*. IEEE, 2010, pp. 184–190.

[15] S. Friedenthal, A. Moore, and R. Steiner, *A practical guide to SysML: the systems modeling language*. Morgan Kaufmann, 2014.

[16] F. Mhenni, N. Nguyen, and J.-Y. Choley, "Automatic fault tree generation from sysml system models," in *Advanced Intelligent Mechatronics (AIM), 2014 IEEE/ASME International Conference on*. IEEE, 2014, pp. 715–720.

[17] M. Kwiatkowska, G. Norman, and D. Parker, "PRISM 4.0: Verification of probabilistic real-time systems," in *Proc. 23rd International Conference on Computer Aided Verification (CAV'11)*, vol. 6806, 2011, pp. 585–591.

[18] ——, "Advances and challenges of probabilistic model checking," in *Communication, Control, and Computing (Allerton), 2010 48th Annual Allerton Conference on*. IEEE, 2010, pp. 1691–1698.

[19] J. Wu, S. Yan, and L. Xie, "Reliability analysis method of a solar array by using fault tree analysis and fuzzy reasoning petri net," in *Acta Astronautica Volume 69, Issues 1112*, December 2011, p. 960968.

[20] M. Ammar, K. A. Hoque, and O. Ait Mohamed, "Formal analysis of fault tree using probabilistic model checking: A solar array case study," pp. 1–6.

[21] J. A. Abraham and D. P. Siewiorek, "An algorithm for the accurate reliability evaluation of triple modular redundancy networks," *Computers, IEEE Transactions on*, vol. 100, no. 7, pp. 682–692, 1974.

[22] P. A. Lee and T. Anderson, *Fault tolerance: principles and practice*. Springer Science & Business Media, 2012, vol. 3.

[23] M. Favalli and C. Metra, "Tmr voting in the presence of crosstalk faults at the voter inputs," *Reliability, IEEE Transactions on*, vol. 53, no. 3, pp. 342–348, 2004.

[24] D. D. Thaker, R. Amirtharajah, F. Impens, I. Chuang, and F. T. Chong, "Recursive tmr: Scaling fault tolerance in the nanoscale era," *Design & Test of Computers, IEEE*, vol. 22, no. 4, pp. 298–305, 2005.

[25] R. Alur and T. A. Henzinger, "Reactive modules," *Formal Methods in System Design*, vol. 15, no. 1, pp. 7–48, 1999.

[26] A. Valmari, "The state explosion problem," in *Lectures on Petri nets I: Basic models*. Springer, 1998, pp. 429–528.

[27] F. Bidner, "Fault tree analysis of the hermes cubesat," *University of Colorado at Boulder, USA*, 2010.