Automated Reliability Analysis of Redundancy Architectures Using Statistical Model Checking

Hongbin He[1], Hongyu Kuang[1], Lin Yang[1], Feng Yang[1], Qiang Wang[1(✉)], and Weipeng Cao[2]

[1] National Key Laboratory of Science and Technology on Information System Security, Systems Engineering Institute, Academy of Military Sciences, Beijing, China
18513688908@163.com

[2] College of Computer Science and Software Engineering, Shenzhen University, Shenzhen, China
caoweipeng@szu.edu.cn

Abstract. Reliability is a fundamental property for mission and safety-critical systems, and adopting redundancy architectures is a common and prominent practice to increase system reliability. This paper proposes a novel approach for the modeling and quantitative reliability analysis of redundancy architectures based on the SBIP framework. Our approach supports modeling the nominal system behavior and the system faults in a unified formal model, which can be further integrated into the rigorous component-based system design paradigm advocated by BIP. We also propose two categories of metrics for formal reliability evaluation of redundancy architectures in terms of whether the system can operate correctly or provide reduced functionalities in the presence of faults. We take a computation unit as the running example and apply the proposed approach to analyze static redundancy and dynamic redundancy, which are Triple Module Redundancy architecture and Cold Standby architecture respectively. The experimental results show that our approach can accurately model various redundancy architectures and provide a comprehensive analysis of reliability and related properties in an automated manner. Moreover, our approach can be easily extended to a wide range of fault types and behaviors.

Keywords: Model based system design · Model based reliability analysis · Statistical model checking · Redundancy architecture

1 Introduction

Reliability is the desired ability for mission and safety-critical systems, which generally speaking characterizes the ability of the system to continue to operate the intended functions correctly even in the presence of faults [2,18]. Among all the possible approaches to increase the system reliability, a common and prominent practice in reliability engineering is to replicate the components carrying out critical functions and encapsulate them in architectural redundancy patterns so that the single point of failure can

H. He and H. Kuang—Contribute equally to this work.

G. Memmi et al. (Eds.): KSEM 2022, LNAI 13370, pp. 463–476, 2022.
https://doi.org/10.1007/978-3-031-10989-8_37

be avoided and when faults occur in a limited number of critical components, they can be identified and excluded upon reconfiguration without compromising the overall functionality of the system. The most well-known redundancy architectural patterns are static Triple Modular Redundancy (TMR) and dynamic Cold Standby, which have been widely used in the practical development of mission and safety-critical systems [23].

Despite the practical needs of designing and analyzing safety-critical systems, the reliability analysis of systems built with redundancy architectures, in general, is a difficult task due to the lack of specific techniques addressing both modeling and automated analysis. Previous works mostly use fault tree analysis (FTA) [16] and rely on a substantial amount of labor effort for the reliability verification and analysis. In [15] the authors analyze the reliability of cascaded TMR with "paper-and-pencil" techniques. However, such approaches cannot be generalized to cover a wide range of architectural patterns. Some other works [9,10] propose to use SMT techniques to perform automated analysis of redundancy architectures and it improves the overall scalability concerning the monolithic case, obtaining speed-ups of various orders of magnitude. However, they do not consider the system's dynamic behavior or integrate it with the process of model-based system design.

In order to automate the reliability evaluation of different redundancy architectures, we follow the methodology of model-based design and safety analysis [21], which has been promoted as an increasingly prominent approach for the development of safety-critical systems. In model-based design, various development activities such as simulation, verification, testing, and code generation are based on a unified model, describing system behavior and architectures. In the model-based safety analysis, the model of the system behavior is further extended and augmented by taking into account the faulty behavior of software and hardware components. We then can analyze the reliability properties based on the extended system in the presence of faults. The main advantage of this methodology is that the system and safety engineers work off a common, unambiguous model of the system leading to tighter integration between the systems and safety engineering processes. The common model ensures that reliability analysis results are relevant and up-to-date as the system architecture evolves and allows reliability assessment early in the system design process. Additionally, it supports the exploration of different architectures and design choices by automatically determining which choices will increase reliability. Ideally, computational tools such as model checkers can automate many reliability analysis activities, leading to more accurate and complete reliability analyses while reducing manual effort.

In this work, we leverage the SBIP framework. SBIP [22] is a stochastic extension of BIP (Behavior-Interaction-Priority) [3,4,7,17] with an emphasis on formal modeling and statistical analysis of safety-critical systems exhibiting stochastic behaviors. BIP is a component-based system design framework advocating the rigorous design methodology for complex hardware/software mixed system design [3,24]. The concept of rigorous system design can be understood as a formal, accountable and coherent process for deriving correct-by-construct system implementations from high-level specifications. The essential safety properties of the design are guaranteed at the earliest possible design phase by applying algorithmic verification to the system model. Then the system implementation is automatically generated by a sequence of property preserving model transformations, progressively refining the model with details specific to

the target platforms. The BIP framework provides a well-defined modeling language and an associated toolbox to realize the rigorous system design flow. The modeling language allows the construction of composite components from atomic components through the layered application of interactions and priorities. The BIP toolbox supports both verifications of high-level system designs [5] and automatic model transformation and code generation of low-level implementations from high-level system designs. In practice, BIP has been actively used in several applications [1, 19].

To this end, the contributions of this work can be summarized as follows.

(1) We propose a novel approach for the modeling and quantitative reliability analysis of redundancy architectures based on the SBIP framework. Our approach supports modeling the nominal system behavior and the system faults in a unified formal model, which can be further integrated into the rigorous component-based system design paradigm.
(2) We propose two categories of metrics to evaluate the reliability and related properties of redundancy architectures, in terms of whether the system is able to operate correctly or provide reduced functionalities in the presence of faults. All the properties are specified as formulas in bounded LTL and automatically analyzed using the statistical model checker for BIP [22].
(3) We further take a computation unit as the running example and apply the proposed approach to the analysis of two widely used redundancy architectures, i.e., the static triple duplication redundancy and the dynamic cold standby redundancy. The experimental results show that our approach can accurately model various redundancy architectures and provide a comprehensive analysis of reliability and related properties in an automated manner.

The remainder of this paper is organized as follows. Section 2 describes the related works. In Sect. 3, we carry out a study with the computation unit and build a formal model of CU including both nominal and faulty behavior within the SBIP framework. Then we develop the formal model of TMR and Cold Standby architectures and analyze their reliability using SBIP. Finally, Sect. 4 concludes the paper and points out our future work.

2 Related Work

COMPASS (Correctness, Modeling, and Performance of Aerospace Systems) [8] is an international research effort aiming to ensure system-level correctness, safety, dependability, and performability of onboard computer-based aerospace systems. COMPASS uses a System-Level Integrated Modeling (SLIM) language for modeling and specifying hardware/software systems, and the COMPASS toolset supports for timed failure propagation graphs, non-deterministic models, the newly developed statistical model checking and requirement formalization approaches.

The Altarica language [6] can also formally specify the behavior of systems when faults occur. An Altarica model can be assessed be employing complementary tools such as fault tree generator and model-checker. FSAP/NuSMV-SA [11, 12] is a toolset for safety and reliability analysis. FSAP/NuSMV-SA supports failure mode definition and model extension through automatic failure injection.

The work in [14] proposes a model-based approach using dynamic fault trees for the safety analysis of vehicle guidance systems. Its flexibility could support new partitions and architectural changes being accommodated automatically. The fault tree analysis in [9,10] aims at evaluating characteristics of redundancy architectures without considering the behaviors of components. The work in [20] proposes an intelligent fault diagnosis method based on an improved domain adaptation method. In [25], a module based on redundancy is designed within the formalism of timed automata and formally analyzed using the UPPAAL model checker. The work in [13] presents an approach based BIP framework for the rigorous design of FDIR components. It leverages the statistical model checking to check the requirement satisfaction and the code generation feature of the BIP compiler. Our work differs from the previous work in that we mainly focus on the reliability analysis of redundancy architectures. The work in [15] presents an ad-hoc algorithm that can analyze the reliability of computational chains based on Triple Modular Redundancy with one voter. Differently, we provide ways to make quantitative analyses reliability of redundancy architectures in full automation.

3 Formal Modeling and Reliability Analysis in SBIP

This section introduces a simple computation unit (CU) that will be used as a running example to illustrate the concepts and our approach throughout. Firstly, we build a formal model of CU including both nominal and faulty behavior within the SBIP framework. Then we build the formal model of TMR and Cold Standby architectures.

3.1 A Computation Unit Example

The computation unit is a common module in many embedded systems, for instance, the detection system in the space field. Its function is to receive the instructions of the upper main control system, then obtain the data from the lower sensor, process the data, and return the processing results to the main control system.

We consider an abstract model of CU, which receives data from some components as its input and computes an output to the other components. Let `cu_input` be the input and `cu_output` be the output of the CU. To simplify the model, we assume that the value of `cu_output` ranges from -1 to 1, where 1 indicates the correct computation, -1 indicates the incorrect computation and 0 is the cleared output. Initially, the value of `cu_output` is 1. An impulse generator is used to first issue an edge impulse xms every `T` time unit to force the CU to read its input, compute and place the result in `cu_output`, and then clear the output. Figure 1 displays a system with one fault-free CU, one Impulse generator and their connector modeled in SBIP. In the Impulse generator model, the Impulse generator automaton models the impulse generator periodically produces edge impulses. A clock x is declared to record the time between sending of two edge impulses. Starting from the initial state `S0`, it generates a signal through port xms in each T time unit that synchronizes with the port xms of the CU automaton, then generates a signal through port $clear$ that synchronizes with the port $clear$ of the CU automaton. In the CU model, a variable `cu_output` is declared to represent the computing result of CU. Starting from the initial state `Good`, it receives a signal through

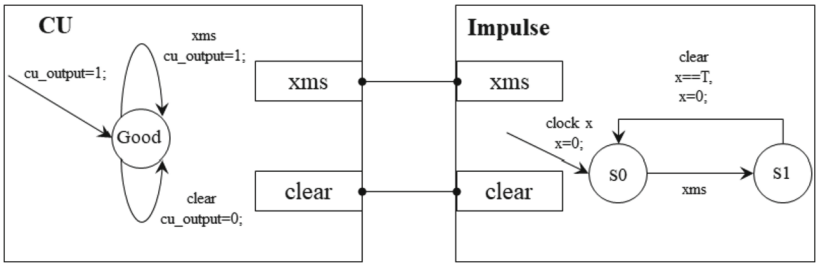


Fig. 1. Fault-free CU model

port xms that computes an output and receives a signal through port $clear$ that clears an output.

We consider that following three kinds of faults that may occur in a CU.

- $FAULT_0$: CU enters a deadlock state.
- $FAULT_1$: CU enters an error state in which it computes incorrect results.
- $FAULT_2$: CU enters a livelock state and executes only internal actions without any outputs.

The fault-affected CU model is shown in Fig. 2. Initially, CU works well and stays in location $Good$. The self-loop in fault-free location $Good$ models the scenario when a synchronization impulse xms occurs, and the CU outputs a correct result. From location $Good$, the CU automaton selects which fault may occur in a stochastic manner. We use exponential distribution to approximate the failure occurrence model for the faults mentioned above in this work. In SBIP, we define a stochastic port with a gamma density function, i.e., its scheduling time is sampled with respect to a gamma function with alpha 1 and beta 100 to indicate the probability of the occurring fault. If $FAULT_0$ occurs, it moves to the error location $Error0$ and the automaton deadlocks. Since the xms is defined as a broadcast channel, the Impulse automaton can still execute it when the CU stays in location $Error0$. The Impulse automaton can still execute the xms. The location $Error1$ is reached from location $Good$ when $FAULT_1$ occurs. The model outputs incorrect data when the signal xms is issued in this location. Similarly, if $FAULT_2$ occurs, the CU goes to location $Error2$, in which it fails to output data when the signal xms is issued. We also add fault transitions from location $Error1$ to $Error0$ and $Error2$, because a $FAULT_1$ may be followed by one of the other faults.

An auxiliary variable `cu_fault` indicates whether the CU is faulty. It is initially 0, representing that the CU is in location $Good$. When a fault occurs in the CU, the value of `cu_fault` becomes 1.

3.2 Formal Model of Triple Modular Redundancy Architecture

The **TMR** model consists of three replicated CUs, one Impulse generator and one Voter (see Fig. 3). Since the Impulse generator and Voter are not redundant in this architecture, it is prone to single-point failures. However, its complexity is usually significantly lower than that of the three modules, and this safety gap is tolerated. So we assume that no faults occur in the Impulse generator or the Voter.

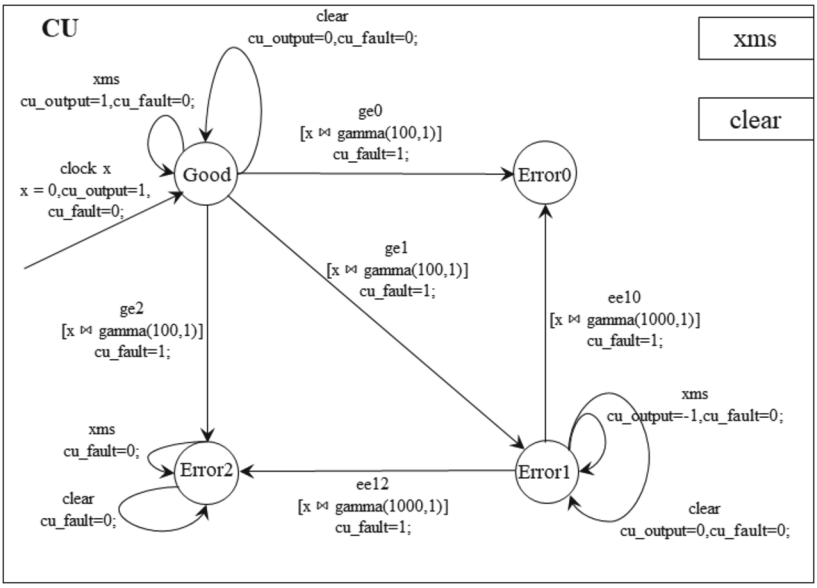


Fig. 2. A SBIP model of fault-effected CU

In the Impulse generator model (see Fig. 4), the automaton generates a synchronization signal through port xms to trigger the three CUs to process their inputs and start the computation. Then it generates a signal through port $x5ms$ to trigger the Voter to check on the outputs of CUs after CU_PERIOD time units. Finally, a $clear$ is sent after $VOTER\_PERIOD$ time units to trigger the three CUs to clear their outputs.

In **TMR**, the CU model (see Fig. 5) is similar to fault-effected CU, but it has two additional ports. The port $fault$ of `CUi` synchronizes with the port $faulti$ of the Voter automaton, and the port $output$ of `CUi` synchronizes with the port $outputi$ of the Voter automaton. In **TMR**, we define the system as in a degraded state when one CU is faulty and the other two CUs are fault-free.

In this model, a Voter is used to detect the faults (see Fig. 6). When it receives the $x5ms$ signal, it calls two functions: `fault_check`() and `vote`(). Function `fault_check`() evaluates whether any CU is faulty. It gets the value of variable cu_fault of `CUi` through port $faulti$. If it is 1, then Voter treats `CUi` as faulty. Otherwise, it is fault-free. Function `vote`() computes the voting algorithm. It compares results from the three CUs and outputs the majority value to the buffer `voter_output`. If the results of the three CUs are different, it computes the value of `CU0`.

3.3 Formal Model of Dynamic Cold Standby Redundancy Architecture

We build the cold standby redundancy architecture model (see Fig. 7), which contains one Impulse generator, one Primary CU, one Backup CU, and one Switch. The Impulse generator and Primary CU are similar to that of the **TMR** model, but all ports of Primary CU are connected to the port $primary$ of Switch.

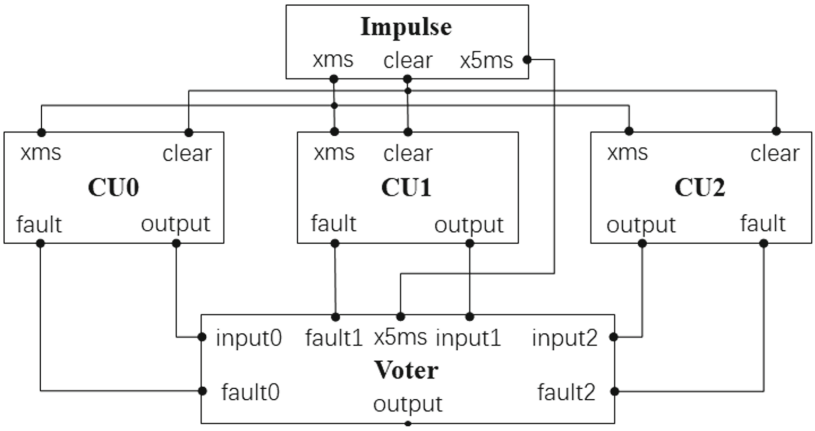


Fig. 3. Triple Modular Redundancy model

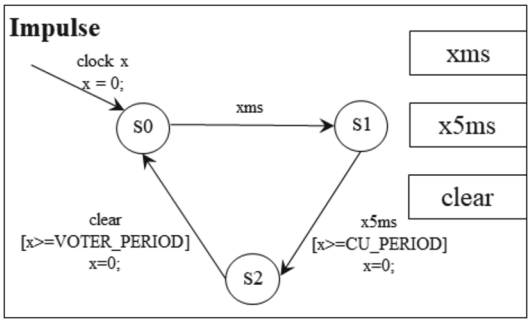


Fig. 4. A SBIP model of Impulse generator

The Switch automaton is shown in Fig. 9. The Switch initiates in the state `S0` and sets variable `is_fault` to 0, which means the Primary CU is fault-free. Moreover, the Switch only interacts with the Primary CU in state `S0`. The port $isswitch$ connects with the port $fault$ of the Primary CU. The port $isswitch$ is active and the state `S0` transfers to state `S1` when the condition `is_faulty` $==$ `1` holds. Then the Switch interacts with the Backup CU in the state `S1`.

The Backup CU automaton is shown in Fig. 8. The initial state is in `Start`, and it transfers to state `Good` when the port $backup$ is active. Moreover, the function of Backup CU is similar to Primary CU after it locates the state `Good`. It may still locate in state `Good` in the next state, or transfer to state `Errori`. All ports of the Backup CU except port $backup$ are connected to the port $backup$ of Switch.

3.4 Formal Reliability Analysis Using SBIP

As stated above, reliability refers to the capability of the system to operate correctly in the presence of faults. The system is in a fault-free state if the system safely operates, and the system is in a degraded state if the system still safely operates but provides

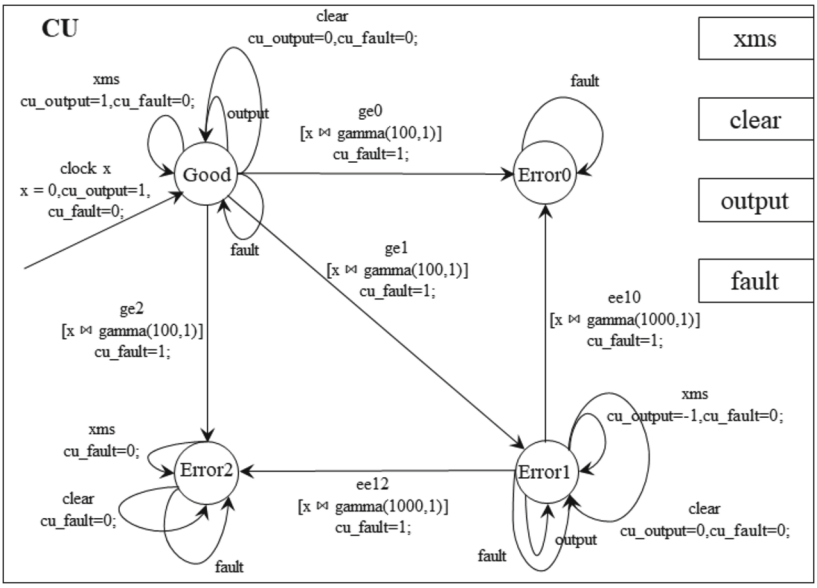


Fig. 5. A SBIP model of TMR's CU

reduced functionality. Furthermore, the system is in a failure state if the function of the system, parts or components are lost or abnormal, resulting in its inability to perform the expected function. We propose two categories of metrics for formal reliability evaluation of the system in terms of whether the system can operate correctly or provide reduced functionalities in the presence of faults.

We use the following four measures to analyze the reliability of the system. `Reliability` means the probability of the system being in a fault-free state. `Failures In Time` (`FIT`) means the average probability of the system being in a fault-free state during the considered operational lifetime. `Degraded Availability` (`DA`) means the probability of the system being in a degraded state. Moreover, `Full Function Availability` (`FFA`) means the probability of the system being in a fault-free and degraded state.

`Reliability` is obtained by first computing the time-bounded probability to reach a state where the system has failed and then complementing this value. To obtain the average failure-probability, the complement of the reliability is scaled with the lifetime (to determine the failures in time, `FIT`). `DA` is obtained by computing the time-bounded reachability probability for reaching a degraded state. `FFA` is obtained by computing the complement of the time-bounded reachability probability for reaching a failed or degraded state.

The formal definition of the measures is given in Table 1. The detailed BLTL specifications of the measures estimated on the Triple Modular Redundancy model and dynamic redundancy model are followed.

In order to simplify writing, we assume that the cf_i represents that the i-th CU is faulty, i.e., variable `cu_fault` of the i-th CU equals to 1, and $\overline{cf_i}$ represents the i-th CU is fault-free, i.e., variable `cu_fault` of the i-th CU equals to 0. The ϕ_T represents

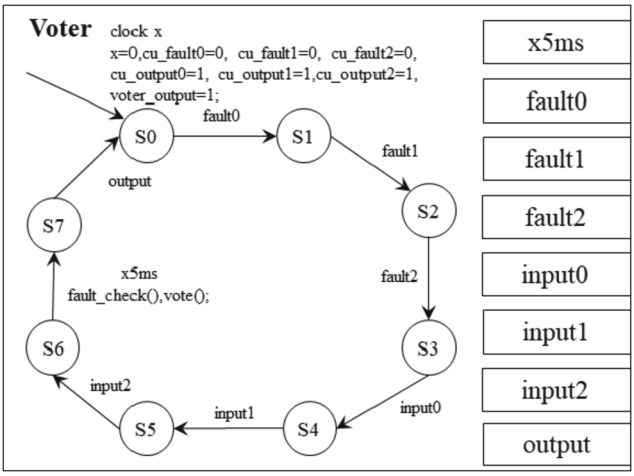


Fig. 6. A SBIP model of TMR's Voter

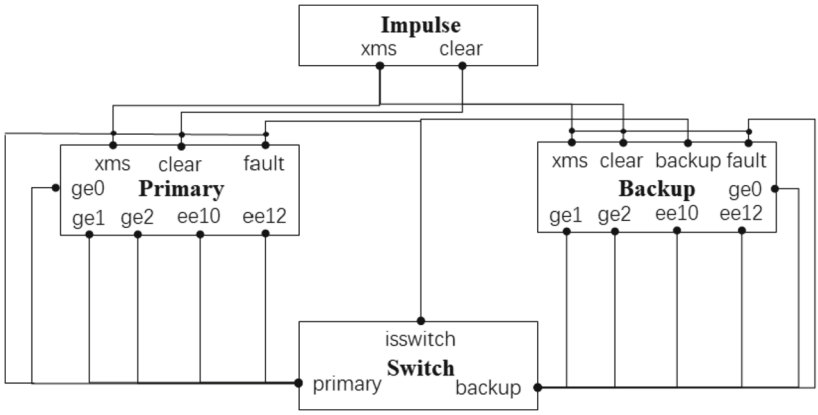


Fig. 7. Cold Standby model

the detailed BLTL specification of the Triple Modular Redundancy model, and the ϕ_D represents the detailed BLTL specification of the dynamic redundancy model.

- For `Reliability`:

$$\begin{aligned}\phi_T(t) = 1 - \mathbf{F}[0,t]((cf_0 \wedge cf_1 \wedge cf_2) \vee (\overline{cf_0} \wedge cf_1 \wedge cf_2) \\ \vee (cf_0 \wedge \overline{cf_1} \wedge cf_2) \vee (cf_0 \wedge cf_1 \wedge \overline{cf_2}))\end{aligned} \tag{1}$$

$$\phi_D(t) = 1 - \mathbf{F}[0,t](cf_0 \wedge cf_1) \tag{2}$$

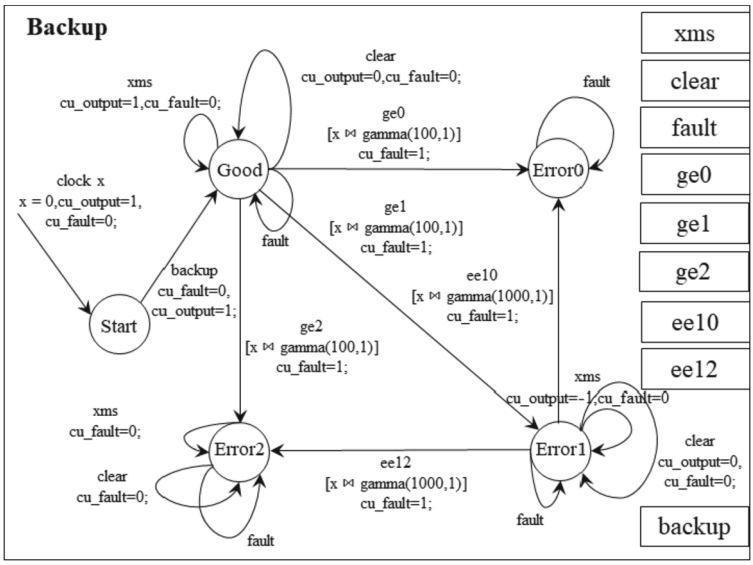


Fig. 8. A SBIP model of Backup CU

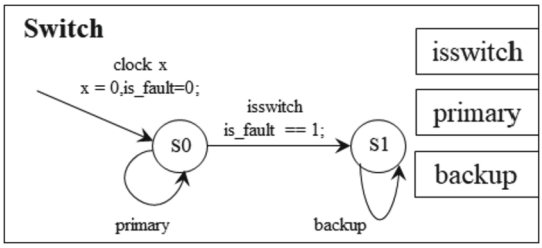


Fig. 9. A SBIP model of Switch

- For `FIT`:

$$\phi_T(t) = \frac{1}{lifetime}(1 - \mathbf{F}[0, lifetime]((cf_0 \wedge cf_1 \wedge cf_2) \vee (\overline{cf_0} \wedge cf_1 \wedge cf_2) \vee (cf_0 \wedge \overline{cf_1} \wedge cf_2) \vee (cf_0 \wedge cf_1 \wedge \overline{cf_2}))) \tag{3}$$

$$\phi_D(t) =) = \frac{1}{lifetime}(1 - \mathbf{F}[0, lifetime](cf_0 \wedge cf_1)) \tag{4}$$

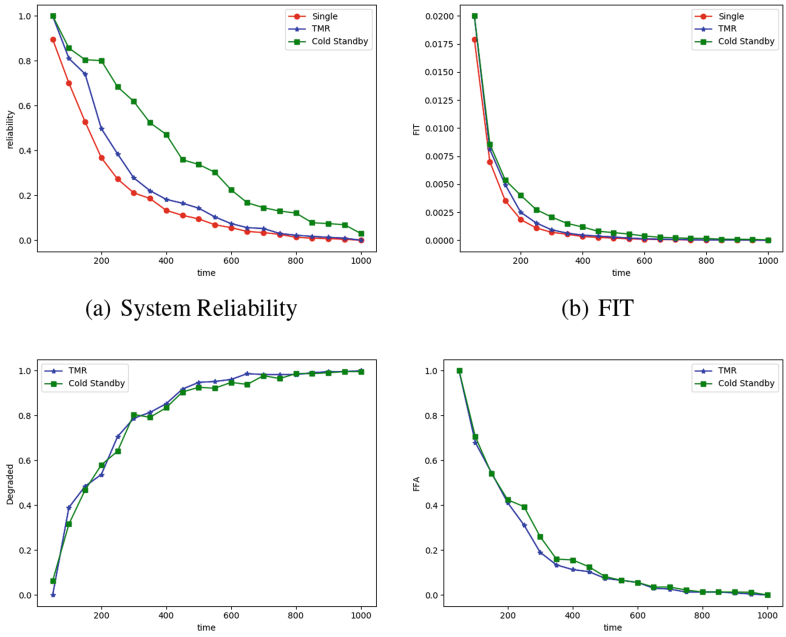
- For `DA`:

$$\phi_T(t) = \mathbf{F}[0, t]((\overline{cf_0} \wedge \overline{cf_1} \wedge cf_2) \vee (\overline{cf_0} \wedge cf_1 \wedge \overline{cf_2}) \vee (cf_0 \wedge \overline{cf_1} \wedge \overline{cf_2})) \tag{5}$$

$$\phi_D(t) = \mathbf{F}[0, t](cf_0 \wedge \overline{cf_1}) \tag{6}$$

Table 1. Definition of measures

	Measure	Model checking queries
System	Reliability	$1 - P(\mathbf{F}^t\ \texttt{failed})$
	FIT	$\frac{1}{lifetime} \cdot (1 - P(\mathbf{F}^{lifetime}\ \texttt{failed}))$
Degradation	DA	$P(\mathbf{F}^t\ \texttt{degraded})$
	FFA	$1 - P(\mathbf{F}^t\ (\texttt{failed} \vee \texttt{degraded}))$



(a) System Reliability (b) FIT

(c) DA (d) FFA

Fig. 10. Results of experimental evaluations using statistical model checking

- For FFA:

$$\begin{aligned}\phi_T(t) = 1 - \mathbf{F}[0,t]((cf_0 \wedge cf_1 \wedge cf_2) \vee (\overline{cf_0} \wedge cf_1 \wedge cf_2) \\ \vee(cf_0 \wedge \overline{cf_1} \wedge cf_2) \vee (cf_0 \wedge cf_1 \wedge \overline{cf_2}) \vee (\overline{cf_0} \wedge \overline{cf_1} \wedge cf_2) \\ \vee(\overline{cf_0} \wedge cf_1 \wedge \overline{cf_2}) \vee (cf_0 \wedge \overline{cf_1} \wedge \overline{cf_2}))\end{aligned} \tag{7}$$

$$\phi_D(t) = 1 - \mathbf{F}[0,t]((cf_0 \wedge cf_1) \vee (cf_0 \wedge \overline{cf_1})) \tag{8}$$

In the experimental evaluations, probability estimation algorithm is utilized. It enables to compute the probability p for S to satisfy ϕ. Give a *precision* δ, this algorithm computes a value for p' such that $|p' - p| \leq \delta$ with confidence $1 - \alpha$. We use probability estimation with (α = 0.2, δ = 0.2) for all the analyses and rely on the parametric exploration to analyze specifications ϕ_T(t) and ϕ_D(t).

Figure 10(a) shows a comparison of the reliability of a single CU, **TMR**, and Cold Standby model. The x-axis represents the time steps and the y-axis represents the probability of property. We observe that **TMR** is slightly more reliable than single CU and Cold Standby is more reliable than single CU and **TMR**. Because three CUs in **TMR** are executed concurrently and the backup CU is activated after the primary CU is faulty, the reliability of Cold Standby is higher at the beginning of a period. The result of Fig. 10(b) shows that the FIT between **TMR** and single CU are nearly the same, and the FIT of Cold Standby is greater than the others. Figure 10(c) indicates that the differences between DA of **TMR** and Cold Standby are marginal. Figure 10(d) shows that the FFA of Cold Standby is Slightly higher than **TMR** at the beginning of a period, and then the FFA of Cold Standby is similar to **TMR**.

4 Conclusions and Future Work

We propose a novel approach to automated reliability analysis of redundancy architectures based on the BIP framework and illustrate it on Triple Modular Redundancy and Cold Standby. The reliability analysis uses the statistical model checking tool SBIP. We propose a relatively complete set of metrics for reliability, including system reliability metrics and degradation reliability metrics, and use it to evaluate the reliability of redundancy architectures. We apply our approach to a computation unit example and experimental results show that our approach can automatically model various redundancy architectures and perform a comprehensive analysis of reliability in practical redundancy systems. Moreover, our models are scalable and our approach is open to additional - models that can be easily extended to further types of fault rates, behavior types, etc. In the future, we will integrate qualitative analysis of reliability into our approach such that qualitative and quantitative analysis can be integrated into the rigorous component-based system design paradigm. Moreover, we can broaden the evaluation architecture and guide the system design according to the evaluation results.

Acknowledgment. This work was supported by National Natural Science Foundation of China (Grant No. 62106150).

References

1. Abdellatif, T., Bensalem, S., Combaz, J., De Silva, L., Ingrand, F.: Rigorous design of robot software: a formal component-based approach. Robot. Auton. Syst. **60**(12), 1563–1578 (2012)
2. Avizienis, A., Laprie, J.C., Randell, B., Landwehr, C.: Basic concepts and taxonomy of dependable and secure computing. IEEE Trans. Dependable Secure Comput. **1**(1), 11–33 (2004)
3. Basu, A., et al.: Rigorous component-based system design using the BIP framework. IEEE Softw. **28**(3), 41–48 (2011)
4. Basu, A., Bensalem, S., Bozga, M., Bourgos, P., Sifakis, J.: Rigorous system design: the BIP approach. In: Kotásek, Z., Bouda, J., Černá, I., Sekanina, L., Vojnar, T., Antoš, D. (eds.) MEMICS 2011. LNCS, vol. 7119, pp. 1–19. Springer, Heidelberg (2012). https://doi.org/10.1007/978-3-642-25929-6_1

5. Bensalem, S., Bozga, M., Nguyen, T., Sifakis, J.: Compositional verification for component-based systems and application. IET Softw. **4**(3), 181–193 (2010)
6. Bieber, P., Bougnol, C., Castel, C., Christophe Kehren, J.-P.H., Metge, S., Seguin, C.: Safety assessment with Altarica. In: Jacquart, R. (ed.) Building the Information Society. IIFIP, vol. 156, pp. 505–510. Springer, Boston, MA (2004). https://doi.org/10.1007/978-1-4020-8157-6_45
7. Bliudze, S., et al.: Formal verification of infinite-state BIP models. In: Finkbeiner, B., Pu, G., Zhang, L. (eds.) ATVA 2015. LNCS, vol. 9364, pp. 326–343. Springer, Cham (2015). https://doi.org/10.1007/978-3-319-24953-7_25
8. Bozzano, M., Bruintjes, H., Cimatti, A., Katoen, J.-P., Noll, T., Tonetta, S.: COMPASS 3.0. In: Vojnar, T., Zhang, L. (eds.) TACAS 2019. LNCS, vol. 11427, pp. 379–385. Springer, Cham (2019). https://doi.org/10.1007/978-3-030-17462-0_25
9. Bozzano, M., Cimatti, A., Mattarei, C.: Efficient analysis of reliability architectures via predicate abstraction. In: Bertacco, V., Legay, A. (eds.) HVC 2013. LNCS, vol. 8244, pp. 279–294. Springer, Cham (2013). https://doi.org/10.1007/978-3-319-03077-7_19
10. Bozzano, M., Cimatti, A., Mattarei, C.: Formal reliability analysis of redundancy architectures. Formal Aspects Comput. **31**(1), 59–94 (2019). https://doi.org/10.1007/s00165-018-0475-1
11. Bozzano, M., Villafiorita, A.: Improving system reliability via model checking: the FSAP/NuSMV-SA safety analysis platform. In: Anderson, S., Felici, M., Littlewood, B. (eds.) SAFECOMP 2003. LNCS, vol. 2788, pp. 49–62. Springer, Heidelberg (2003). https://doi.org/10.1007/978-3-540-39878-3_5
12. Bozzano, M., Villafiorita, A.: The FSAP/NuSMV-SA safety analysis platform. Int. J. Softw. Tools Technol. Transf. **9**(1), 5–24 (2007). https://doi.org/10.1007/s10009-006-0001-2
13. Dragomir, I., Bensalem, S.: Rigorous design of FDIR systems with BIP. Electron. Commun. Eur. Assoc. Softw. Sci. Technol. (ECEASST) **77** (2019). https://doi.org/10.14279/tuj.eceasst.77.1107, https://researchr.org/publication/DragomirB19
14. Ghadhab, M., Junges, S., Katoen, J.P., Kuntz, M., Volk, M.: Safety analysis for vehicle guidance systems with dynamic fault trees. Reliab. Eng. Syst. Saf. **186**, 37–50 (2019)
15. Hamamatsu, M., Tsuchiya, T., Kikuno, T.: On the reliability of cascaded TMR systems. In: 2010 IEEE 16th Pacific Rim International Symposium on Dependable Computing, pp. 184–190 (2010)
16. Hiraoka, Y., Murakami, T., Yamamoto, K., Furukawa, Y., Sawada, H.: Method of computer-aided fault tree analysis for high-reliable and safety design. IEEE Trans. Reliab. **65**(2), 687–703 (2016)
17. Konnov, I., Kotek, T., Wang, Q., Veith, H., Bliudze, S., Sifakis, J.: Parameterized systems in BIP: design and model checking. In: Proceedings of the 27th International Conference on Concurrency Theory (CONCUR 2016), p. 30-1. Schloss Dagstuhl-Leibniz-Zentrum fuer Informatik (2016)
18. Laprie, J.: Dependable computing and fault tolerance: concepts and terminology. In: Twenty-Fifth International Symposium on Fault-Tolerant Computing 1995, 'Highlights from Twenty-Five Years', p. 2 (1995)
19. Lekidis, A., Stachtiari, E., Katsaros, P., Bozga, M., Georgiadis, C.K.: Model-based design of IoT systems with the BIP component framework. Softw. Pract. Exp. **48**(6), 1167–1194 (2018)
20. Li, Y., Song, Y., Jia, L., Gao, S., Li, Q., Qiu, M.: Intelligent fault diagnosis by fusing domain adversarial training and maximum mean discrepancy via ensemble learning. IEEE Trans. Ind. Inf. **17**(4), 2833–2841 (2021)
21. Lu, K.L., Chen, Y.Y.: Model-based design, analysis and assessment framework for safety-critical systems, Virtual, Taipei, Taiwan, pp. 25–26 (2021)

22. Mediouni, B.L., Nouri, A., Bozga, M., Dellabani, M., Legay, A., Bensalem, S.: $\mathcal{S}$BIP 2.0: statistical model checking stochastic real-time systems. In: Lahiri, S.K., Wang, C. (eds.) ATVA 2018. LNCS, vol. 11138, pp. 536–542. Springer, Cham (2018). https://doi.org/10.1007/978-3-030-01090-4_33
23. Schnellbach, A.: Fail-operational automotive systems. Ph.D. thesis, Graz University of Technology (2016)
24. Sifakis, J.: System design automation: challenges and limitations. Proc. IEEE **103**(11), 2093–2103 (2015)
25. Zhang, M., Liu, Z., Morisset, C., Ravn, A.P.: Design and verification of fault-tolerant components. In: Butler, M., Jones, C., Romanovsky, A., Troubitsyna, E. (eds.) Methods, Models and Tools for Fault Tolerance. LNCS, vol. 5454, pp. 57–84. Springer, Heidelberg (2009). https://doi.org/10.1007/978-3-642-00867-2_4