Received December 11, 2018, accepted February 21, 2019, date of publication March 4, 2019, date of current version March 20, 2019.

Digital Object Identifier 10.1109/ACCESS.2019.2902796

Towards an Accurate Probabilistic Modeling and Statistical Analysis of Temporal Faults via Temporal Dynamic Fault-Trees (TDFTs)

MARWAN AMMAR[1], GHAITH BANY HAMAD[1], OTMANE AIT MOHAMED[1], (Member, IEEE), AND YVON SAVARIA[2], (Fellow, IEEE)

[1]Electrical and Computer Engineering Department, Concordia University, Montreal, QC H3G 1M8, Canada
[2]Groupe de Recherche en Microelectronique et Microsystemes, Polytechnique Montréal, Montreal, QC H3T 1J4, Canada

Corresponding author: Marwan Ammar (m_amma@ece.concordia.ca)

ABSTRACT Fault tree (FT) is a standardized notation for representing relationships between a system's reliability and the faults and/or the events associated with it. However, the existing FT fault models are only capable of portraying permanent events in the system. This is a major hindrance since these models fail to reflect accurately the other classes of faults, such as soft-faults, which are often temporary events that usually disappear after the source of the interference is no longer present. This paper proposes a new fault tree modeling paradigm, to capture the impact of temporal events in systems, called *temporal dynamic fault trees (TDFTs)*. TDFTs are utilized to model the characteristics and dependencies between different temporal events, soft-faults, and permanent faults. These features are integrated into the proposed probabilistic models of the temporal gates, which are modeled as *priced-timed automata*. This paper also proposes a new FT analysis methodology, based on *statistical model checking*, designed to circumvent the state-explosion problem that is inherent to other model-checking approaches. The proposed analysis is able to evaluate the impact of temporal faults in systems, as well as to estimate the reliability and availability of the system over extended periods of time. The experiments reported in this paper demonstrate the versatility and scalability of the proposed approach. For instance, the results display the impact that temporal events may have in a digital system. Our observations indicate that while regular soft-fault analyses tend to underestimate metrics such as system reliability, TDFT analysis shows remarkable consistency with radiation testing, with differences of under 2%, in the conducted analysis.

INDEX TERMS Fault tree, temporal events, radiation effects, single-event effects, statistical model-checking, formal verification, system-level analysis, reliability, availability.

I. INTRODUCTION

Fault tree analysis (FTA) is a prominent fault diagnosis technique which has gained widespread acceptance for quantitative safety analysis. A fault tree consists of a diagram which represents the failure of the top level event (TLE) according to the failure of basic events based on the relationships between them. The objective of FTA is to provide insightful information to designers regarding the reliability of their systems by identifying the ways in which the system is most likely to fail and thus showing the most efficient ways to make the system safer. In the literature, FTs are often classified as either static or dynamic, based on the dependency relationships between their respective components and events. Examples of such dependencies can be event priority, sequence, spare behavior, etc. With the introduction of dynamic gates [1]–[3], the relationship between the events and components of a fault tree have changed into a dynamic one, in which the outcome becomes dependent on the order and the number of occurrences of basic events. Further development into dynamic gates has led to the development of fault trees with temporal requirements [4]–[6]. These fault trees extend the dynamic gates to include tighter sequence requirements for the events. However, the inclusion of temporal constrains in these approaches greatly limits the analysis process. Traditionally, FTA is performed through simulation

The associate editor coordinating the review of this manuscript and approving it for publication was Engang Tian.

techniques [7], with tools based on techniques such as *Monte-Carlo simulation*, *time-sequential simulation*, and *discrete event simulation* [8], [9]. Analysis of large systems using these techniques can often be overwhelming for the tool since simulation-based analysis relies on input space sampling. This means that unless all possible points are sampled, there exists a possibility that an error is not detected by the analysis.

In recent years, many formal-based approaches for the analysis of dynamic fault trees have emerged [7], [10]–[12]. These approaches (detailed in Section II) solve most of the limitations of simulation techniques, achieving fast and efficient FTA. However, current FTA approaches are not suitable for safety-critical analysis, since current FT modeling techniques cannot capture sequences of actions (such as how many times has event 1 occurred before event 2) and state history. Moreover, the binary representation (working or failed state) of FTs is not adequate for systems with complex state-spaces. For example, conventional fault tree events start as inactive and may become active according to a probability rate. Once an event becomes active, it will remain active throughout the rest of the analysis. This behavior makes it impossible to represent the occurrence of transient faults (i.e., faults that may appear or disappear from the system due to the effects of external sources of interference, such as radiation and heat) with existing fault-tree gates. This is due to the absence of suitable fine-grain management of time in conventional fault-tree structures [13].

This paper proposes the modeling of a new type of dynamic fault trees with strict behavioral and temporal requirements, hence referred to as *Temporal Dynamic Fault Trees (TDFTs)*. TDFTs are primarily targeted towards the analysis of temporal events, such as radiation effects and heat. However, the TDFT gates are flexible enough to be configured for the analysis of most FT systems. The work in this paper is distinct from the literature in the following ways: 1) The proposed TDFTs are fault trees that capture temporal events, state history and sequences of actions. We present and explain the models of each TDFT gate, formulated as Priced-Timed Automata (PTA). Thereafter, the complete model of the fault tree is obtained through the parallel composition and synchronization of the PTAs of all the required gates. 2) Each of the gates in the fault tree is extended with a unique clock which, along with a global system clock, enables precise time tracking and management. This allows the proposed model to accurately represent temporal faults, which are only active for a certain amount of time [14]. It also enables the verification of other temporal properties, such as the time required for a specific event to manifest itself in the system. 3) An analysis methodology is introduced, utilizing FT modularity and sequential hypothesis testing in order to decrease time and resource requirements while maintaining a high confidence level for the results. This is combined with *Statistical Model-Checking* (SMC), which provides statistical evidence for the satisfaction or violation of the specification.

The aforementioned distinctions are demonstrated through a comparative study, derived from verifying the estimated system reliability obtained with conventional FTs versus the new TDFT model. The proposed TDFT analysis is experimentally evaluated on different scenarios in order to assess its impact on the different types of fault tree gates and to demonstrate its versatility. Finally, the proposed methodology is used to analyze the 7-stage integer pipeline of a Leon-3 microprocessor. The obtained results are compared with radiation and simulation tests from the literature [15]. Our results demonstrate that regular FTA methods are inadequate for the analysis of systems that are exposed to temporal faults. The rest of this paper is structured as follows: In Section II, some of the most relevant related works are briefly discussed. In Section III, we explain a few preliminary concepts that are key to the development of this work. Section IV introduces the proposed modeling of the TDFT gates, and the SMC-based analysis methodology. In Section V, several experiments are presented. These experiments have the goal of demonstrating the main differences and advantages introduced by the proposed methodology. Finally, in Section VI, we draw some conclusions and discuss future works.

II. RELATED WORKS

FTA is an extremely important field of research. Being the focus of many groups during the past decade, FTA has evolved into one of the de facto techniques for early analysis of critical systems. Walker and Papadopoulos [6] first suggested extending static FTs with Priority-AND, Priority-OR, and Simultaneous-AND gates. These gates enable a fault tree to enforce temporal dependencies between events. The works in [11] and [12] present very robust formal approaches to DFT analysis, based on *Input/Output-Interactive Markov Chains* (I/O-IMCs) and stochastic model checking. The work in those papers apply modularization approaches and compositional aggregation techniques, along with heavy reduction algorithms, to formally analyze DFTs with model checking. These techniques are built around the *Galileo* formalism [16], which greatly limits the expressiveness of the analyzed models. For example, neither approach is able to handle self-repairable systems or any dependencies between gates or events that have not been pre-programmed in the tools.

In recent years, several techniques have been proposed to extend DFTs in ways to offer more flexible temporal dependencies between its faults and events. Schilling [17] proposes an extension of the conventional boolean FTA in order to take sequence dependencies into account for qualitative and probabilistic analyses without state-space transformations. This allows modeling of sequences of events in all levels of the fault tree. The analysis of uncertainty over time is the focus in [18]. This work uses the Pandora tool and fuzzy logic in order to predict and capture different sequences of dependent dynamic events over time. The authors use their method to combine probabilistic data and fuzzy set theory with Pandora TFTs to enable dynamic analysis of complex systems with limited or absent exact quantitative data. Peng *et al.* [19] use a timed FT extension method applied to a railway maintenance system in order to identify which faults are likely to occur first and, therefore, must be eliminated more urgently. This method can estimate the time required for railway maintenance and thereby improve maintenance efficiency, and reduce risks.

All these techniques have in common the attempt to introduce new dependencies between the different components and events of dynamic fault trees. However, they lack the expressiveness required for a robust analysis of the impact of failures over time in a dynamic environment. The work presented in this paper further augments fault trees, by combining and extending dynamic, repairable and temporal fault trees. The proposed fault tree models are able to capture the randomness of fault testing, where a significant event (i.e., radiation) may occur non-deterministically. Furthermore, this event may be permanent, intermittent, or it may be active for a variable unknown amount of time.

III. PRELIMINARIES

A. THE UPPAAL FORMALISM

UPPAAL is a toolbox for verification of real-time systems, represented by a network of timed automata, extended with integer variables, structured data types, and channel synchronization. For the efficient analysis of probabilistic performance properties, UPPAAL-SMC proposes to work with *Statistical Model Checking* (SMC). SMC works by monitoring some simulations of the system, and then use statistical results (including *sequential hypothesis testing* or *Monte Carlo simulations*) to decide whether the system satisfies some property with a sufficient degree of confidence. The modeling formalism of UPPAAL-SMC is based on a stochastic interpretation and an extension of the *Timed Automata* (TA) formalism used in the classical model checking version of UPPAAL. For individual TA components, the stochastic interpretation replaces the non-deterministic choices between multiple transitions enabled by probabilistic choices (that may or may not be user-defined). Similarly, the nondeterministic choices of time delays are refined by probability distributions, which at the component level are given either uniform distributions in cases with time-bounded delays or exponential distributions in cases of unbounded delays [20].

An illustrative example of the UPPAAL formalism is given by the PTA in Fig. 1. In this example and through the rest of this paper, the weight annotations on locations and edges are ignored and defaulted to "1". For Fig. 1, the delay distribution determined by the upper and lower paths to the END state is given by sums of uniform distributions, where $X \geq 2$ (green label) is the guard of the transition (i.e., minimum time), and $X \leq 4$ (purple label) is the invariant distribution (i.e., maximum time delay) of the transition. The stochastic choice that determines which path will be taken is represented by a forked transition, where each path is weighted accordingly. In the example, the weights of each path are either $\frac{1}{6}$ or $\frac{5}{6}$. Finally, an update may be performed during each transition (blue labels). Therefore, the END location in the example is reachable within the interval $X = [4, 12]$.

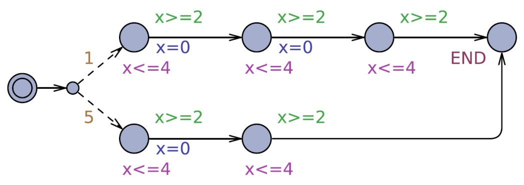

FIGURE 1. Illustrative example of the UPPAAL formalism. Reproduced from [20].

B. FAULT TREE ANALYSIS

The fault tree analysis (FTA) method is a widely used method for risk assessment, mainly in the area of avionics, nuclear and chemical industries [1]. FTA follows a deductive approach, which means that it starts from an undesirable general event in order to find what circumstances may lead to that event. In the context of FTA, the general event is known as the *top event*, 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. Bottom events are occurrences that may lead to a component failure. As such, fault tree models are characterized as graphical representations of system failures, in terms of the system's components. Standard fault tree models are defined as combinatorial models, composed by static gates (mainly *AND* and *OR* gates) and basic events. Combinatorial models can only handle combinations of events and not the order of occurrence of such events, thus are not able to represent complex systems adequately. Dynamic fault trees (DFTs) extend standard fault trees to allow the representation of more complex relationships between basic events, such as functional dependencies, priority, and order of occurrence.

IV. PROPOSED TDFT MODELING AND ANALYSIS METHODOLOGY

A constant among existing FTA techniques is that the failure of a basic event cannot be reverted (i.e., when a basic event changes from the normal state to the fail state, it will remain in the fail state forever) [21]. However, it has been noted in the literature that basic events (i.e., events that may lead to the failure of a component) are not always permanent. This fact may heavily impact the results of fault analyses, especially in systems exposed to nondeterministic environmental interferences, such as radiation, heat, and cosmic rays [22]–[24]. In a previous work related to malfunction in pacemakers exposed to ionizing radiation [25], we have demonstrated that these radiation-induced malfunctions vary from a simple temporary abnormality to complete system malfunction. The analyses in [25] have indicated that a simple temporary bit-flip (which, according to technical reports, is a common occurrence) may cause the pacemaker to behave erroneously and even endanger the life of a patient, in some cases.

The approach proposed in this paper overcomes this FTA limitation by introducing the sensitivity to *Temporal Basic Events* (TBE) to the fault tree gates (i.e, events that may appear for a limited amount of time and then disappear if certain conditions are met). This enables a more accurate representation of the environmental hazards that the system may be exposed to, as well as for the tracking of faults which may only manifest after thousands of cycles. Moreover, the proposed models are capable of estimating the probability of fault occurrence in systems exposed to temporal events of varying duration.

In order to accurately model the fault dependencies in FTs, our approach focuses on the formalization and modeling of the probabilistic behavior of FT gates and events over time. In this section, we show the modeling of temporal FT gates. The general probabilistic model (automaton) of a FT gate is expressed in Definition 1, adapted from [26].

Definition 1: Given a TDFT gate with a set of inputs Y and an output Z, connected through a certain logic (such as AND). The priced-timed automaton (PTA) of this gate can be formally defined by a tuple $A = (L, L_0, \chi, Act, P, \mathcal{L})$*, where:*

- L *is a finite number of states.*
- L_0 *is the initial state.*
- χ *is a finite set of clocks.*
- Act *is a finite set of actions over* L*.*
- *inv:* $L \rightarrow \zeta(Y)$ *is an invariant condition.*
- P *is a probabilistic transition function* $L \times \zeta(Y) \times Dist(2^Y \times L)$*.*
- $\mathcal{L} : L \rightarrow 2^{AP}$ *is a labeling function assigning atomic propositions to different states.*

In the PTA defined above, a state $(l, \nu) \in L \times \mathbb{R}^{\chi}_{\geq 0}$ is characterized such that $\nu \models inv(l)$. In any state (l, ν), there is a nondeterministic choice of either making a discrete transition or letting time pass. A discrete transition can be made according to any $(l, g, p) \in P$, with current state l being enabled and zone g is satisfied by the current clock valuation ν. The probability of moving to location l' and resetting all clocks in Y to 0 is given by $p(Y, l')$. The option of letting time pass is available only if the invariant condition $inv(l)$ is satisfied while time elapses. Based on this definition, following we explain in detail the proposed models for each TDFT gate.

A. PROPOSED PROBABILISTIC MODEL OF THE TEMPORAL AND GATE

The probabilistic *AND* gate can be modeled as a temporal gate because of the tight dependency between the output and the inputs. As stated previously, the output is only generated if all inputs occur. However, in the case of temporal faults, input events need to happen at the same time to cause the top level fault. For example, let us imagine a component that fails in the presence of external heat (event x) AND radiation (event y). Let us assume a scenario where external heat is applied to the component for a certain amount of time, after which the heat source is dissipated. Let us also assume that the component has time to return to its regular temperature before it is affected by external radiation. In this case, since both events have happened at different points in time, the requirements for component failure have not been met.

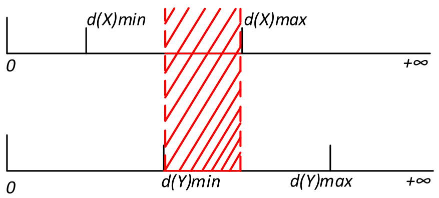

FIGURE 2. Possible time window of a TAND output.

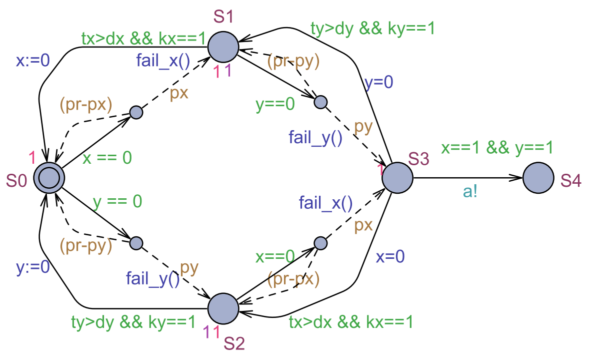

FIGURE 3. Example of a 2-Input leaf TAND gate.

In the proposed Temporal AND gate (*TAND*), each of the basic events connected to the gate (X and Y in this example) is tied to two attributes: the probability of the event occurring and the duration of the event. The derivation of the TAND rule is the following:

- **Duration of the Event**: It is assumed that the time duration of a basic event is a random variable d selected from an interval $[0, \ldots, N]$, where $N \in \mathbb{R} \geq 0$. After an amount of time equal to d has passed, the basic event ceases to exist in the system. The basic event may re-occur according to the specified probability rate, in which case the duration of the event may be different.
- **Temporal Condition**: As is the case with regular *AND* gates, the output Z occurs only if all the inputs (X and Y) occur. However, in the *TAND* gate, the output is only generated when the duration of both inputs intersect. In other words, both inputs must be active at the same time. As illustrated in Fig. 2, the time interval in which the output of the *TAND* gate may be generated is given by:

$$Z = [d(X)_{min}, d(X)_{max}] \bigcap [d(Y)_{min}, d(Y)_{max}] \quad (1)$$

where $[d(X)_{min}, d(X)_{max}]$ and $[d(Y)_{min}, d(Y)_{max}]$ are the intervals in which events X and Y are active.

Fig. 3 shows a possible configuration of a *Leaf TAND* gate with two inputs. A *leaf* gate (or *bottom* gate) is any gate that has only basic events as inputs. An example of the possible flow of a *Leaf TAND* gate with two inputs is as follows: State *S0* signifies the absence of faults. From there, the model can transition through two symmetrical paths, where either event may happen (i.e., x or y). Let us assume that event y occurs, with probability py. In this case, the automaton moves to state *S2* and an update is performed by the function *fail_y*(), which

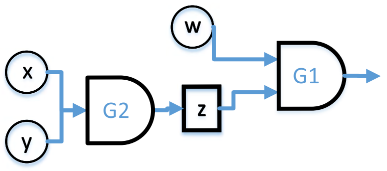

FIGURE 4. Example of a simple fault tree.

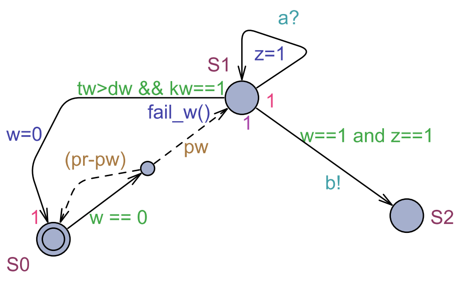

FIGURE 5. Example of a 2-Input variant TAND gate.

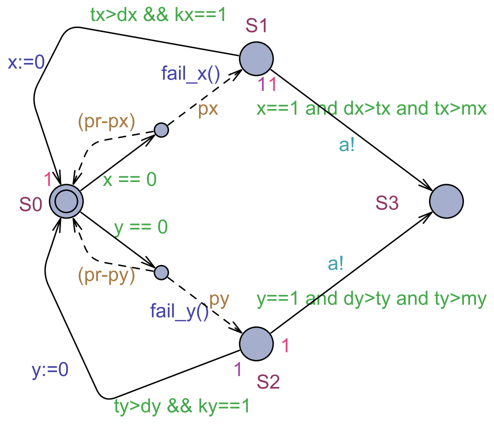

FIGURE 6. Example of a 2-Input leaf TOR gate.

resets the clock *ty*, determines the type of the event *ky* (*ky == 0* signifies permanent fault, and *ky == 1* signifies temporal fault) and the maximum duration *dy* (in time units) of the event. At this point, the automaton may return to state *S0*, if *ky == 1* and *ty > dy*, or it may proceed to state *S3*, with probability *px*. If event *x* happens, the update *fail_x*() resets clock *tx* and sets the value of *kx*, moving the automaton to state *S3*. In this scenario, state *S3* may transition back to state *S2*, if *tx > dx* and *kx == 1*, or it may transition to state *S4*, broadcasting the output to the next gate. The same flow applies to the top path of the automaton, when event *x* occurs first.

To demonstrate how a FT is constructed with the proposed automata, a simple example is given. Let us consider the FT shown in Fig. 4. This FT has instances of two possible configurations of the 2-Input TAND gate. Gate *G2* is a *leaf TAND* gate, since all of its inputs (*x* and *y*) are basic events. The automaton of this gate flows as described above (Fig. 3), producing output *z*. Let us now consider the gate *G1* from Fig. 4. Gate *G1* is a variant of the *leaf TAND* gate, employed in cases where one of the inputs of the gate is also the output of another gate. The PTA of gate *G1* is shown in Fig. 5. Although gate *G1* receives two inputs, *w* and *z*, the input *z* is not an event, but rather the output of gate *G2*. Therefore, gate *G1* only receives one basic event as input (event *w*). The flow of this automaton is the following: if *w == 0* (i.e., event *w* is inactive), then *w* may happen with probability *pw*. If event *w* happens, an update is performed by the function *fail_w*(), which resets the clock *tw*, determines the type *kw* and the maximum duration *dw* of event *w*. The transition takes the system to state *S1*. In state *S1*, the automaton will wait for event *z*, which can be received through a synchronization channel (channel *a*, in this example). If the system is in state *S1* and event *z* occurs (i.e., *w == 1* and *z == 1*), then a transition to state *S2* takes place. Alternatively, if the system is in state *S1* and the conditions *kw == 1* (i.e., temporal fault) and *tw > dw* (i.e., duration of the fault has expired) are met, then the automaton goes back to state *S0*. When state *S2* is reached, the output of the gate is produced and communicated to the next level of the FT via synchronization.

For conciseness, the discussions for the other gates will focus on the *leaf* variant, since it displays the full functionality of the gate and any required variants can be derived from the *leaf* automaton. It must also be noted that all behavioral patterns in the proposed timed automata are enforced with the use of variables, either through guards, updates, synchronization, or declarations, as exemplified above. Therefore, our use of invariants in the models has the sole goal of allowing the automaton to stay in any state indefinitely (i.e., invariant set to 1). This is important, since the propagation delay of the different inputs and gates throughout the fault tree is unknown, especially when temporal and permanent faults coexist in the same fault tree. Our experiments have shown that wrong results and often verification errors are generated when the automata are not allowed to hold a state indefinitely. Similarly, our experiments have shown that the presence of exponential exit rates is required in states where a clock is manipulated. Our results have suggested that different values of exponential exit rates have little impact on the verification, as long as the value is consistent across all states. For this reason, we have adopted the policy of setting the value of the exponential exit rate to 1 in all states where it is required.

B. PROPOSED PROBABILISTIC MODEL OF THE TEMPORAL OR GATE

The *OR* gate is modeled as a temporal gate to better represent the propagation delay that exists in real systems. The concept of the proposed leaf Temporal OR (*TOR*), shown in Fig. 6, is relatively simple, compared with other temporal gates: assuming two temporal events (*x* and *y*) connected to a *TOR* gate, where each event has a given probability of failure, the occurrence of either event can generate an output in the gate. However, due to the propagation delay of the modeled component (represented in Fig. 6 by the variables *mx* and *my*), the output is not generated immediately. Therefore, the output is only generated if the duration of the fault is longer than the propagation delay, but smaller than the maximum duration of the fault (variables *dx* and *dy*). The output of the gate is communicated to the next layer of the FT through a synchronization channel. The derivation of the TOR gate follows the same rules as the TAND gate with regards to the duration of the events. The temporal condition for this gate is given by the equation:

$$Z = [(d(X)_{min}, d(X)_{max}] \mid [d(Y)_{min}, d(Y)_{max}] \qquad (2)$$

C. PROPOSED PROBABILISTIC MODEL OF THE TEMPORAL FDEP GATE

The Functional Dependency (FDEP) gate is a dynamic gate composed of a trigger input event and one or more dependent basic events. If the trigger event occurs, the dependent events automatically become unavailable. While the trigger event is not in a failed state, the dependent events behave like regular events (i.e., the dependent events may fail independently from the trigger event). The FDEP gate does not have a direct output, however, a functional dependency may be attached to any other gate in the fault tree, altering its behavior. The behavior of the probabilistic FDEP assumes that a dependent event *y* (such as the output of a gate) may happen with probability *py*. However, if the trigger event *x* occurs, with probability *px*, then event *y* is forced to happen as well.

The main issue with the regular modeling of the FDEP gate is that the temporary occurrence of the trigger event compromises the system permanently. Let us take the example of another electrical component. Let us consider the absence of electricity to power up the component as the trigger event and the failure of the component as the dependent event. It may be the case that an external interference causes the trigger event to occur, and therefore causes the component to cease function. However, it may also be the case that the trigger event disappears after a certain amount of time (i.e., the power is restored). In this scenario, the system may still function. The proposed Temporal FDEP (*TFDEP*) gate (Fig. 7), models the behavior of a temporal trigger event *x*. From state *S0*, event *x* may occur with probability *px*, moving the automaton to State *S1*. From state *S1*, the automaton may go back to state *S0* if $tx > dx$ and $kx == 1$. In this case, event *x* has no observable effect in the FT. Alternatively, from state *S1*, the automaton may transition to state *S2*. This transition sends a message (*a!*) which immediately causes all other events that are associated to the *TFDEP* gate to fail. Furthermore, from state *S2*, if the conditions $tx > dx$ and $kx == 1$ are satisfied, the automaton may move back to state *S0*. When this transition takes place, another message is sent *(b!)*, which may override the effects of the first message *(a!)*. The temporal condition of an TFDEP dependency relationship is given by the equation:

$$Z = [d(X)_{min}, d(X)_{max}] \mid [d(Y)_{min}, d(Y)_{max}] \bigcap [d(W)_{min}, d(W)_{max}] \qquad (3)$$

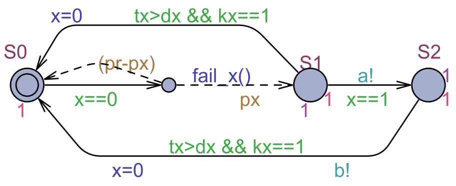

FIGURE 7. Example of a TFDEP gate.

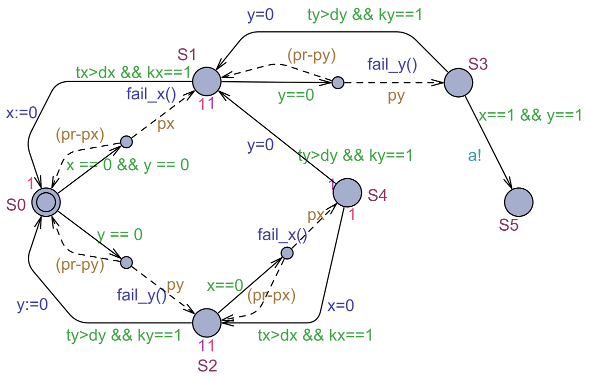

FIGURE 8. Example of a 2-Input leaf TPAND gate.

where *x* is the trigger event, and *y* and *w* are two dependent events.

D. PROPOSED PROBABILISTIC MODEL OF THE TEMPORAL PAND GATE

As with the other FT gates, the PAND gate logic may also be susceptible to temporary faults. To illustrate this, let us consider the case of a backup system with two components in a standby configuration, with component *A* being the primary component and component *B* in standby. If component *A* fails, then the electrical switch (which can also fail) activates component *B*. Thus, the system will fail if component *A* fails and the switch fails, or if component A fails then component *B* fails subsequently. In this scenario, the events must occur in the specified order. However, it is possible for the switch or component *B* to fail without causing a system failure, if component *A* never fails. Moreover, it is possible that the electrical switch fails to switch due to a temporary fault. In this case, the switching action might take place immediately after the temporary fault exits the system.

In the proposed *leaf* variant of a Temporal PAND (*TPAND*) gate with two inputs (Fig. 8), starting from state *S0* and assuming temporary event *x* is the primary event, a failure only happens if event *x* fails before event *y*. The failure of event *x* moves the automaton to state *S1*. From state *S1*, event *x* may disappear from the system, returning to state *S0*, or event *y* can happen, with probability *py*. The occurrence of event *y* transitions the automaton to state *S3*, where event *y* may disappear (returning to state *S1*) or the output may be generated, which transitions the model to state *S5*. If event *y* happens first, the model moves to state *S2*. From state *S2*, event *y* may disappear, which causes a return to state *S0*,

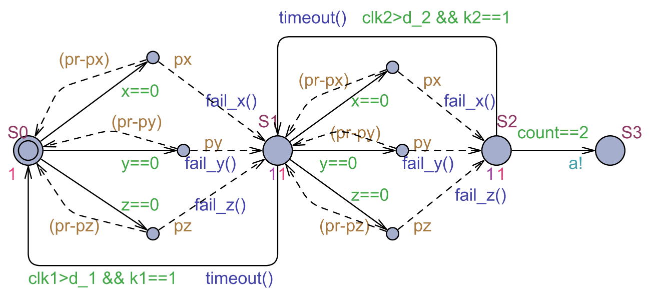

FIGURE 9. Example of a 2-of-3 leaf TComb gate.

or event x may occur, which causes a transition to state *S4*. Finally, from state *S4*, if event y expires, the automaton moves to state *S1*. Alternatively, if event x expires, the automaton moves back to state *S2*. As with the other gates, the output of the TPAND is communicated to the next gate via synchronization message (message *a!*, for example). The temporal condition for the output of the TPAND gate is given by the equation:

$$Z = [d(X)_{min}, d(X)_{max}] \bigcap [d(Y)_{min}, d(Y)_{max}] \iff (d(X)_{min} \leq d(Y)_{min}) \quad (4)$$

E. PROPOSED PROBABILISTIC MODEL OF THE TEMPORAL COMB GATE

The Combinational gate (COMB) is a special case of the AND gate. A COMB gate is composed by three or more inputs and one output. The output occurs if M-of-N inputs occur. In other words, the combination gate allows the designer to specify the number of failures within a group of inputs that is required for the top level event to occur. By observing the examples given in the previous gates, especially the TAND and TPAND gates, it becomes clear how the combinational gate can be augmented into Temporal Combinational (*TCOMB*) gate. A possible configuration of a *leaf* 2-of-3 model of the TCOMB gate is illustrated in Fig. 9. As usual, state *S0* of the automaton signifies *no faults*. From state *S0*, three symmetrical paths may be taken, representing the probabilities of failure of the three inputs of the gates. Let us take event x as an example. From *S0*, event x may happen with probability px. This transition triggers the function $fail\_x()$. This function is responsible for a plethora of variable updates. If the variable *count* is equal to zero, which is the case since this is the first transition in the model, function $fail\_x()$ will set the clock associated to state *S1* (*clk1*) to zero, the variable d_1 will be set to the maximum duration of event x, and variable *k1* will be set to the value of kx. Finally, function $fail\_x()$ will increment the variable *count* and set variable x to 1. After all these updates, the automaton is in state *S1*. From this state, event x may expire if the conditions $clk1 > d_1$ and $k1 == 1$ are met. In this case, the function *timeout()* takes place. If function *timeout()* occurs from state *S1* (i.e., $count == 1$), all variables are reset, returning the automaton to state *S0*. Alternatively, from state *S1*, maintaining the assumption that event x has already occurred, event y can happen, with probability py, or event z can happen, with probability pz. Let us assume that event z takes place. In this case, the automaton calls the update function $fail\_z()$. Since variable *count* is equal to one, the update function will set the clock *clk2* to zero, the variable d_2 will be set to the value of dz, and the value of *k2* will be set to the value of kz. The function also increments the variable *count* and sets variable z to one. With the automaton in state *S2* and variable *count* equal to 2, the transition to state *S3* may happen, broadcasting the output through channel a, in this example. However, from state *S2*, if the conditions $clk2 > d\_2$ and $k2 == 1$ are met, the automaton may transition back to state *S1*, which triggers the update function *timeout()*. If *timeout()* is called from state *S2* (i.e., $count == 2$), the most recent event is reset (in the case of this example, z), and the variable *count* is decremented.

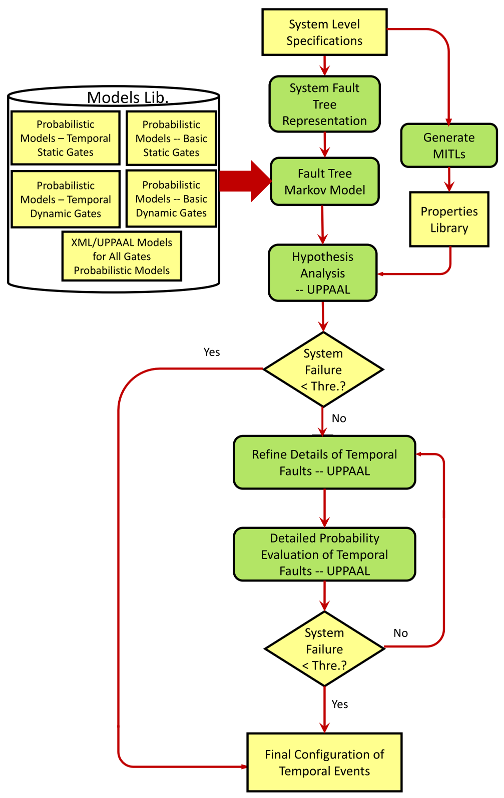
System Level Specifications
Models Lib.
Probabilistic Models – Temporal Static Gates
Probabilistic Models -- Basic Static Gates
Probabilistic Models – Temporal Dynamic Gates
Probabilistic Models -- Basic Dynamic Gates
XML/UPPAAL Models for All Gates Probabilistic Models
System Fault Tree Representation
Fault Tree Markov Model
Generate MITLs
Properties Library
Hypothesis Analysis -- UPPAAL
Yes
System Failure < Thre.?
No
Refine Details of Temporal Faults -- UPPAAL
Detailed Probability Evaluation of Temporal Faults -- UPPAAL
System Failure < Thre.?
No
Yes
Final Configuration of Temporal Events

FIGURE 10. Main steps of the proposed methodology.

The temporal condition used for the output of the TCOMB gate is a variant of the one used in the TAND gate. Therefore the temporal condition is omitted to avoid redundancy.

F. PROPOSED ANALYSIS METHODOLOGY

This subsection introduces the proposed analysis methodology. The flow chart of the proposed methodology is shown in Fig. 10. We start from a system-level specification model, in which a system is composed of interconnected components. This specification model must be provided by the designer in a general-purpose modeling language, such as *SysML* [27]. The failure rate of each component is characterized from the system specification. From the SysML model,

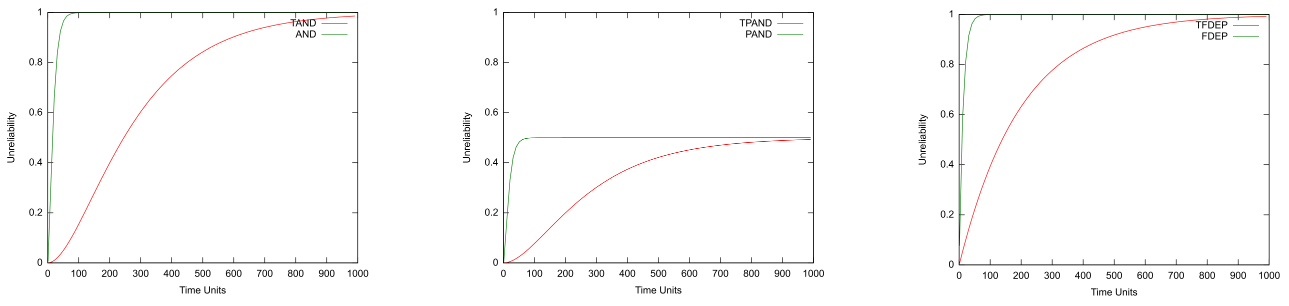

FIGURE 11. Comparison of the estimated unreliability over time of TDFTs and DFTs. (a) TAND vs. AND gate. (b) TPAND vs. PAND gate. (c) TFDEP vs. FDEP gate.

a fault tree of the system is obtained using an automatic synthesis tool, such as the one proposed in [28]. Subsequently, a formal PTA model of the system's FT is obtained through the parallel composition of the PTA models of the TDFT gates. These gate models exist in a UPPAAL gate library (shown as *Models lib.* in Fig. 10) that can be imported into any statistical model-checking tool with support to UPPAAL's XML format. The next step is to evaluate if the reliability of the system under analysis is within the acceptable threshold defined by the specification. This evaluation is performed in UPPAAL-SMC by using Wald's sequential hypothesis testing [29]. This test computes a proportion r among n runs that satisfy the defined property. Given two possible hypothesis, a and b, the value of r will eventually cross $\log(\beta \div (1-\alpha))$ or $\log((1-\beta) \div \alpha)$ with probability 1, where α and β are the probabilities of accepting hypothesis a or b, respectively [20]. The properties generated for the hypothesis testing and for the model checking steps are in the form of full weighted *Metric Interval Temporal Logic (MITL)* queries. An example of such query is given below:

$$Pr[bound; N](max : expr) \tag{5}$$

where *bound* defines the constraint on the number of runs. N gives the number of runs explicitly, and *expr* is the expression to evaluate. It is worth noting that these properties are analyzed for a certain confidence interval, which controls the number of iterations processed by the tool. If the probability of failure in the system is within the allowed threshold, no further analysis is conducted. However, if the probability of failure is above the allowed threshold, the proposed methodology moves into the refinement evaluation step. The goal of this step is to identify if there exists a certain configuration of the fault tree under analysis that satisfies the evaluated query. To this end, first, the critical path of the TDFT is identified (i.e., the sub-tree that has the highest probability of failure). Then, the leaf gates of the critical path are tested with different instances of temporal events with different durations. For example, a gate that has only permanent faults may be changed so that one or more of its events may become temporal. Thereafter, these temporal events may be tested over different duration intervals.This analysis can generate a very a detailed quantitative report, that has the objective of showing the system designers of the exact vulnerabilities of the system, and which parameters have a greater impact to the system's criticality.

V. EXPERIMENTAL RESULTS

In this section, the results of the analyses of the failure probabilities of different fault tree gates and systems with and without the presence of temporal faults are presented and discussed. Furthermore, different experiments are presented, with the purpose of demonstrating a different aspect of the proposed TDFT methodology. The analyses have been performed on UPPAAL-SMC version 4.1.19, running on a machine with an AMD Ryzen 1800X CPU and 32 GB of RAM.

A. UNRELIABILITY EVALUATION OVER TIME

It is important to illustrate the behavioral difference between a TDFT gate and a regular FT gate. This experiment evaluates the probability of failure of a single TDFT gate due to temporal faults and the progression of this failure over time compared with a regular FT gate. Fig. 11 illustrates the behavior of different TDFT gates and their corresponding regular gates. The failure rate of all basic events in this experiment is assumed to be equal to 0.1. Fig. 11(a) shows the results obtained from the AND gate and from the TAND gate. Fig. 11(b) shows the unreliability progression of the PAND and TPAND gates. Finally, Fig. 11(c) shows the results obtained by computing the unreliability of the FDEP and TFDEP gates, connected to a regular AND gate through a trigger relationship. It can be observed that the unreliability probabilities obtained with the regular FT gates are greatly overestimated. This happens because, as discussed previously, basic events that occur in a regular gate are permanent. However, basic events occurring in a temporal gate may have a limited duration, in which case a failure is only triggered if the required events happen during the same time window.

B. SCALABILITY OF THE PROPOSED TDFT ANALYSIS

One of the major weaknesses of regular FTA methods based on model-checking is the size limitation that is imposed on the analysis. For example, probabilistic model checking of FTs over time is very demanding and often cannot be handled

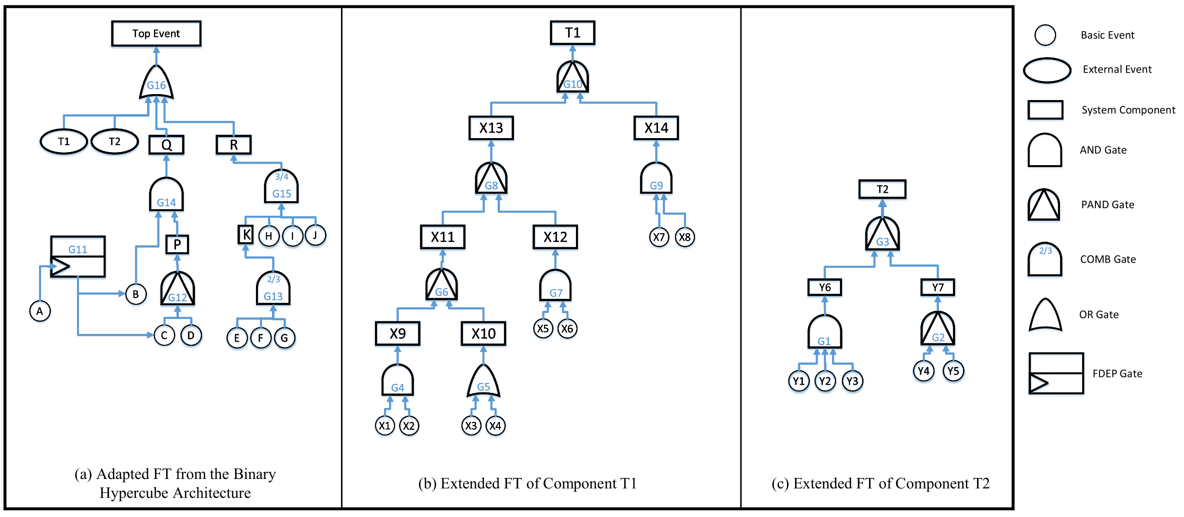

FIGURE 12. Modular FT analysis of the binary hypercube architecture. (a) Adapted FT from the binary hypercube architecture. (b) Extended FT of component T1. (c) Extended FT of component T2.

by model-checking tools such as PRISM, MRMC and Storm. The proposed TDFT method circumvents this limitation in two distinct ways: 1) As previously mentioned (Fig. 10), the proposed TDFT analysis takes advantage of a technique called Statistical Hypothesis Testing (SHT) [29]. SHT is a method of statistical inference where two statistical data sets are compared, or a data set obtained by sampling is compared against a synthetic data set from an idealized model. In other words, the model may be evaluated against a query (i.e., is the estimated availability of component $x \geqslant 0.9$?). The outcome of this evaluation is either true or false. This method may be used to guide the analysis process by minimizing the expected resource consumption. 2) Unlike most model-checking techniques, the proposed TDFT methodology applies full FT modularity without the need to partition the original model into its sub-trees. This can be done by simply editing a command line, responsible for instantiating the models, in the UPPAAL system declarations.

The main objective of the second experiment is to demonstrate the applicability of the proposed approach on large systems. This experiment is also used to showcase the discrepancy between the results obtained with the proposed TDFTs against regular probabilistic FTA analysis, in scenarios where temporal faults are considered. The fault trees used in this experiment can be seen in Fig. 12. Fig. 12(a) shows a top-level fault tree model that is adapted from the case study developed in Dugan *et al.* [30]. Components *T1* (Fig. 12(b)) and *T2* (Fig. 12(c)) are external events to the top-level FT. This experiment is conducted in three separate steps. In each step, two different models of the fault trees are built. The first model (DFT model) uses regular probabilistic FT gates, while the second model (proposed TDFT model) uses the temporal gates proposed in this paper. Both FT models are analyzed separately. In the case of the TDFT model, whenever an event is triggered in a temporal gate, a random choice is made to determine if the event is permanent or temporal. This is done to test the scalability of the models, as temporal events

TABLE 1. Estimated availability of component T1 after 100 seconds. Failure rate of basic events is assumed to be 0.05. Temporal events are assumed to last up to 3 seconds.

	Temporal Gate	Components						
		X9	X10	X11	X12	X13	X14	T1
Regular FTA	n/a	0.0135	0.0001	0.833	0.0001	0.9023	0.0135	0.9652
TDFT	G4	0.9328	0.0001	0.997	0.0135	0.9994	0.0135	0.99997
	G7	0.0135	0.0001	0.833	0.933	0.9896	0.0135	0.99893
	G9	0.0135	0.0001	0.833	0.0135	0.902	0.932	0.998

are more complex to be resolved. In the case of the DFT model, all events are considered permanent upon occurrence. It is worth mentioning that for the purposes of this paper, the results of the different iterations of the hypothesis analyses are omitted, for clarity.

The first step in this experiment is the analysis of component *T1*. The extended FT of component *T1* is shown in Fig. 12(b). To illustrate the impact that temporal events may have in the analysis, this goal of this experiment is to consider a single temporal gate in the FT and to compute the impact of that temporal gate on the estimated availability of the system. Table 1 shows the estimated availability at the different system components when the events of gates *G4, G7, or G9* are considered temporal. In this experiment, it is assumed that the failure rate of the basic events (*X1 - X8*) is 0.05, the duration of temporal basic events is 3 seconds, and the estimated availability is computed over a period of 100 seconds. For example, let us consider the cases where gates *G4 or G7* are temporal. It can be seen in Fig. 12(b) that gate *G4* impacts the results of components *X9, X11, X13, and T1*. On the other hand, gate *G7* impacts components *X12, X13, and T1*. For each case, the results in the table quantify the impact of these gates on each of the components, compared to analyses that only consider permanent events. For example, if we were to classify the system in the terms of the *Five Nines* standard [31], and assuming that the input events of gate *G4* are temporal, a regular FTA would classify this system (availability of *T1*) as *One Nine (1N)*, whereas

TABLE 2. Estimated availability of component T2 after 100 seconds. Failure rate of basic events is assumed to be 0.05. Temporal events are assumed to last up to 3 seconds.

	Temporal Gate	Components		
		Y6	Y7	T2
Regular FTA	n/a	0.088	0.534	0.885
TDFT	G1	0.9986	0.534	0.99934
	G2	0.0879	0.9964	0.9973

TABLE 3. Estimated availability of the binary hypercube system after 100 seconds. Temporal events are assumed to last up to 3 seconds.

	Temporal Gate	Components				
		P	Q	K	R	TLE
DFTCalc	n/a	0.998	0.9993	0.996	0.99994	0.851
Regular FTA	n/a	0.998	0.9994	0.996	0.99993	0.862
TDFT	G11	0.99992	0.99996	0.996	0.99993	0.9984
	G12	0.99996	0.99995	0.996	0.99993	0.9986
	G13	0.998	0.9994	0.99998	0.99996	0.991
	G14	0.998	0.9994	0.996	0.9999991	0.9985

the TDFT analysis would rightfully classify it as *Four Nines (4N)*. This means that in this hypothetical scenario, the regular FTA would greatly underestimate the availability rating of this system. This, in turn, would mean that the design team would have to spend more resources than necessary in order to increase the rating of the system to the *Five Nines (5N)* standard (i.e., availability = 0.999995, or the system is available for 99.999% of the time).

Next, we perform a similar analysis on the extended FT of component *T2*, shown in Fig. 12(c). For this experiment, the assumptions used for component *T1* also apply. The results are presented in Table 2, for the regular and temporal analyses. A different analysis is performed for each of the lower-level gates to assess the impact of temporal events on those gates as well as their impact on the top event. In the last step of this experiment, we utilize the partial results in Tables 1 and 2 to analyze the fault tree of the Hypercube system (in Fig. 12(a)). The failure rates for the basic events of this fault tree (*A* - *J*) are assumed to be equal to the failure rate of *T1*.

Table 3 shows an estimation of the availability of the different components of *T1*, after 100 seconds. In the table, *Regular FTA* shows the results obtained with regular FT analysis. For comparison and validation, the row *DFTCalc* of the table shows the results obtained through FTA using the DFTCalc tool [11], which is a well-known tool for dynamic fault tree analysis. In the *TDFT analysis* rows, we report the results obtained with the proposed TDFT models, with both temporal and permanent faults considered. Similarly to the previous steps of the analysis, each of the lower-level gates was analyzed individually to evaluate the effect of temporal events, as well as their impact to the top-level event. The results shown, for *Regular FTA* and *DFTCalc*, demonstrate a near parity between the values obtained. However, the table also shows that the presence of even a single source of temporal faults may drastically alter the estimated results of the analysis. This demonstrates the importance of the proposed

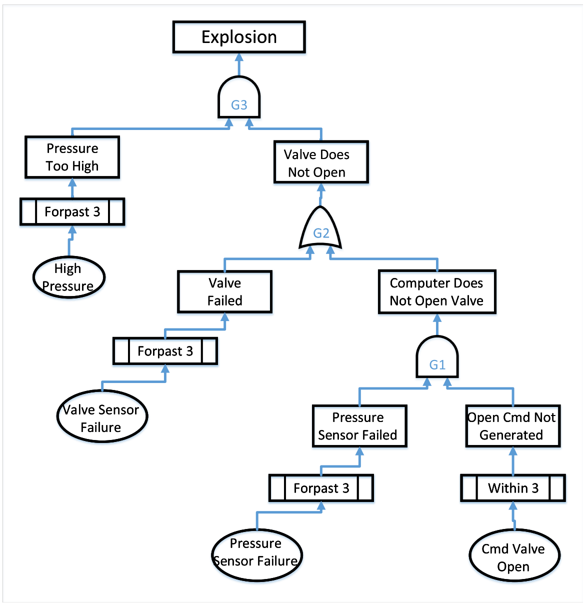

FIGURE 13. Temporal fault tree of the pressure chamber case-study.

methodology in environments where temporal faults may occur, since this difference cannot be detected with regular FTA. It is noticeable that the type and location of the temporal gate may drastically change the results obtained at the TLE. For example, gate *G11* directly impacts the outcome of gates *G12, G14 and G16*. In other words, increasing or decreasing the availability rating associated to gate *G11* has direct effects on the outputs of all other gates that *G11* is connected to. Therefore, as seen in Table 3, the effects of a single gate may ripple through the FT generating significant differences in the estimated results.

C. COMPARISON BETWEEN TDFTs AND TEMPORAL FAULT TREES (TFTs)

As previously discussed in Section II, other techniques have tried to integrate temporal constraints to fault tree analysis. One of the most expressive techniques in the literature is the Temporal Fault Trees (TFTs) formalism, introduced in [32]. The analysis in [32] assigns time constraints to the propagation of the bottom events of the tree. The examples of such time constraints, shown in Fig. 13, are *Forpast* and *Within*. *Forpast* indicates that the event must be active for a minimum amount of time before propagating (e.g., *Forpast* 3 indicates that the bottom event must be active for 3 units of time before propagating in the system). Similarly, *Within* indicates that the bottom event must occur within a certain time-frame (e.g, *Within* 3 signifies that the event must happen before 3 time units have passed, in order to propagate in the system). In this subsection, we present a direct comparison between the TDFT and TFT techniques, by adapting the Pressure Chamber fault tree (Fig. 13) from [32] and comparing the obtained results. The system depicted in the figure shows a series of events and conditions that may lead to an explosion in the system. Starting from the bottom-most events, the

TABLE 4. Estimated reliability of the pressure chamber system after 100 seconds. (Fault=X in the table refers to the time duration of the fault, with X being units of time.)

	(1 - Probability of Explosion)				
	Fault=1	Fault=3	Fault=5	Fault=7	Fault=10
Proposed TDFT	0.99997	0.9975	0.9911	0.9827	0.9635
TFT	0	0.69	0.69	0.69	0.69
Regular FTA	0.06	0.06	0.06	0.06	0.06

failure of the *pressure sensor* for over 3 units of time together with the absence of an *open valve command* within the same 3 time units, generates an error where the computer was supposed to open the pressure valve but fails to do so (gate *G1*). Alternatively, the system may experience a *valve sensor failure*, which, if it lasts more than 3 time units, generates a *valve failure*. If the valve fails or if the computer fails to open the valve, an error is generated, since the valve did not open to release the building pressure (gate *G2*). If the valve cannot be opened and the system experiences high pressure in the chamber for over 3 time units, an explosion occurs (gate *G3*).

For the TDFT analysis, the conditions *Forpast* and *Within* have been modeled with *leaf* TOR gates, where the conditions *Forpast* and *Within* are enforced by adjusting the guards of the output transition of the *TOR* gate. Gates *G1, G2 and G3* are regular TDFT gates. The comparison of the results of the TFT analysis with the proposed TDFT analysis are presented in Table 4. From the table, it can be seen that while the TFT technique is definitely an improvement over regular FTA, the results obtained by the former are rather limited. In the example (Fig. 13), the expected duration of the fault events is 3 seconds. Therefore, if the duration of the faults is less than 3 seconds, the probability of failure is equal to zero. Furthermore, if the duration of the faults is equal or greater than 3 seconds, the probability of failure is always the same. Based on this fact. The table shows that unlike other approaches, the results provided by the TDFT analysis can provide a distinct estimation for each considered fault duration.

D. FAILURE ESTIMATION OF THE SPARC V8 ARCHITECTURE WITH TDFTs

Having established the differences between the proposed TDFT analysis and the regular probabilistic FTA in the previous experiments, the final experiment demonstrates the importance of the proposed approach to fault analysis. This is done by analyzing the fault tree of the integer pipeline of the *Leon-3* processor and comparing the obtained results to radiation testing and simulation. In order to obtain the FT of the Leon-3 integer pipeline, we have adopted an analytical approach for the generation of fault trees for complex systems, known as the *Behaviour-Based Method* [33]. This approach considers faults as behaviours, and fault-tree gates as operations on those behaviours. By applying this technique to the Leon-3 7-stage pipeline, and based on the

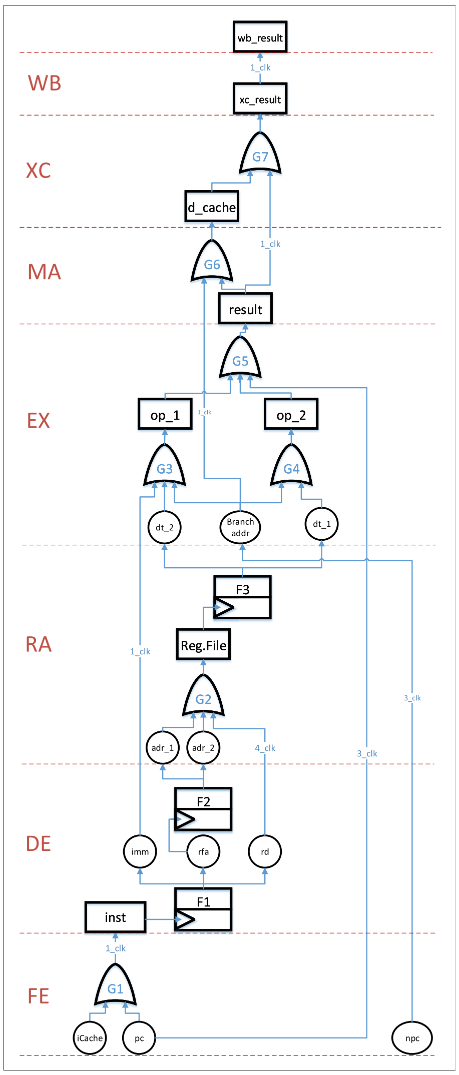

FIGURE 14. DFT of the 7-stage integer pipeline of the SPARC-V8 architecture.

structural information available in the *SPARC V8* architecture manual [34], [35], the fault tree of the integer pipeline is constructed, as shown in Fig. 14. The fault tree is divided into 7 levels, each representing a stage of the pipeline. For this experiment, the probabilities used for the failure rates in the model are derived from the cross-section values reported in [15].

The goals of this experiment are first to determine the probability of a crash error in the processor pipeline, which raises a trap exception. Secondly, to determine the probability of each type of trap error generated over a period of time. To this end, the probabilities of soft-error events utilized in our model are derived from the cross-section values obtained through the radiation bombardment of a LEON3 design conducted and published in [15]. Furthermore, the work in [15] also contains a fault-injection simulation experiment. These results have been used in this paper for set-up and validation purposes.

In this analysis, the proposed FT model has been evaluated through over 3800 iterations, reaching a confidence

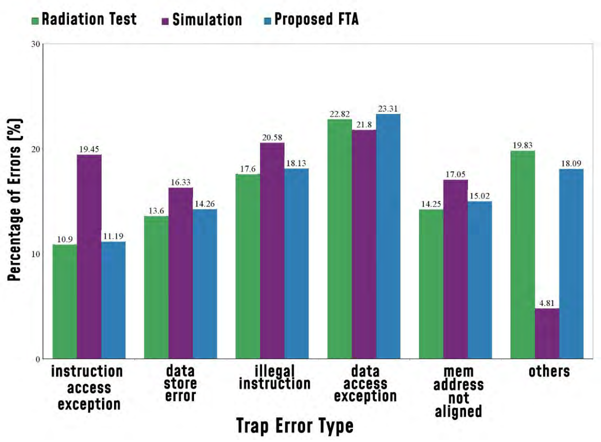

FIGURE 15. **Probability of trap exceptions in different approaches. Simulation and radiation test results are reproduced from [15].**

level of 95%. The confidence level can be further increased if more iterations are considered. Each iteration computes the estimated probability of system failure, assuming multiple soft-errors may happen at any given time. Through our results, we estimate that approximately 41% of all errors originating from soft-faults were captured as trap exceptions. Based on the modeled fault tree, and following the functionality described in the SPARC-V8 manual [34], the proposed model is able to estimate the probability of occurrence of the following types of trap exceptions in the SPARC-V8 pipeline:

- **instruction_access_exception:** A blocking error exception causes the instruction to be unavailable.
- **data_store_error:** An error exception that occurs during a data store to memory.
- **illegal_instruction:** An attempt to execute an instruction with an invalid opcode.
- **data_access_exception:** An error exception that occurs on a load/store data access.
- **mem_address_not_aligned:** A load/store operation that generates an improper memory address, according to the instruction.
- **Others:** All other types of trap exceptions.

Out of the 41% of soft-faults captured as trap exceptions, Fig. 15 shows the probability of each exception type to occur. In order to validate the proposed model, we have compared the obtained probabilities (*Proposed FTA*) to the ones reported in [15] (*Radiation Test* and *Simulation*). It can be seen that the values obtained with the *proposed FTA* are consistent with the values of the *radiation test*.

The proposed analysis can also be used to assess other metrics, such as the estimated time before a failure, the failure rate over time, and the impact of soft-errors on different components to the vulnerability of the system. Table 5 shows an estimation of the probability of trap exceptions over time, the probability of detected errors over time, and the probability of undetected errors over time. Although relatively small, the probability of undetected errors in the system may represent a serious issue in certain conditions, where the system is

TABLE 5. **SPARC-V8 probability of failure over time.**

	Prob. of TE (million hours)	Prob. of Error (million hours)	Prob. of Undetected Errors (million hours)
SPARC-V8 Pipeline	0.027	0.056	0.004

expected to operate without maintenance. Our analysis shows that the biggest contributors to the occurrence of undetected errors are the nPC register (22.5 % of cases), the rfa register (19.7 % of cases), and the d_cache (17.3 % of cases).

VI. CONCLUSION

This paper presents a new modeling and analysis approach to accurately compute the availability of systems exposed to temporal faults. Regular probabilistic fault tree models calculate the probability of failure under the assumption that every fault is permanent. However, in the real world, sources of interference (such as heat and radiation) can be intermittent. Therefore, their impact on the behavior of digital circuits (especially self-repair systems) may be only temporary. TDFTs are introduced to capture such phenomena, providing an unprecedented level of precision and customization to fault tree analysis of soft-faults. The results presented in this paper illustrate the versatility of the proposed methodology and the level of accuracy obtained in comparison with other FTA approaches. Future work includes the further extension of TDFT gates in order to analyze latent faults and rare events in systems affected by temporal faults.

REFERENCES

[1] W. E. Vesely *et al.*, "Fault tree handbook," Division Syst. Rel. Res., U.S. Nucl. Regulatory Commission, Washington, DC, USA, Tech. Rep. NUREG-0492, 1981. [Online]. Available: https://www.nrc.gov/docs/ML1007/ML100780465.pdf

[2] H. Boudali, P. Crouzen, and M. Stoelinga, "Dynamic fault tree analysis using input/output interactive Markov chains," in *Proc. IEEE/IFIP Int. Conf. Dependable Syst. Netw.*, Jun. 2007, pp. 708–717.

[3] M. Walker *et al.*, "Synthesis and analysis of temporal fault trees with PANDORA[1]: The time of Priority AND gates," *Nonlinear Anal. Hybrid Syst.*, vol. 2, no. 2, pp. 368–382, Jun. 2008.

[4] P. G. Wijayarathna and M. Maekawa, "Extending fault trees with an AND-THEN gate," in *Proc. Int. Symp. Softw. Rel. Eng.*, Oct. 2000, pp. 283–292.

[5] M. Walker *et al.*, "Compositional temporal fault tree analysis," *Computer Safety, Reliability, and Security*. Berlin, Germany: Springer, 2007, pp. 106–119.

[6] M. Walker *et al.*, "A hierarchical method for the reduction of temporal expressions in pandora," in *Proc. 1st Workshop Dyn. Aspects Dependability Models Fault-Tolerant Syst.*, Apr. 2010, pp. 7–12.

[7] E. Ruijters and M. Stoelinga, "Fault tree analysis: A survey of the state-of-the-art in modeling, analysis and tools," *Comput. Sci. Rev.*, vols. 15–16, pp. 29–62, Mar. 2015.

[8] K. D. Rao, V. Gopika, V. V. S. Rao, H. S. Kushwaha, A. K. Verma, and A. Srividya, "Dynamic fault tree analysis using monte carlo simulation in probabilistic safety assessment," *Rel. Eng. Syst. Saf.*, vol. 94, no. 4, pp. 872–883, Apr. 2009.

[9] J. Faulin *et al.*, *Simulation Methods for Releaty Availability Complex System*. New York, NY, USA: Springer, 2010.

[10] C. Dehnert *et al.*, "A storm is coming: A modern probabilistic model checker," in *Proc. Int. Conf. Comput. Aided Verification*, Sep. 2017, pp. 592–600.

[11] F. Arnold *et al.*, "Dftcalc: A tool for efficient fault tree analysis," in *Proc. Int. Conf. Comput. Saf., Rel., Secur.*, 2013, pp. 293–301.

[12] M. Volk *et al.*. (2016). "Advancing dynamic fault tree analysis." [Online]. Available: https://arxiv.org/abs/1604.07474

[13] L. H. Mutuel, "Single event effects mitigation techniques report," Thales Avionics, Seattle, WA, USA, Tech. Rep. DOT/FAA/TC-15/62, 2016. [Online]. Available: https://www.faa.gov/aircraft/air_cert/design_approvals/air_software/media/TC-15-62.pdf

[14] S. Mukherjee, *Archit. Design for Soft Errors*. Burlington, MA, USA: Morgan Kaufmann, 2011.

[15] C. Bottoni *et al.*, "Heavy ions test result on a 65nm sparc-v8 radiation-hard microprocessor," in *Proc. IEEE Int. Rel. Phys. Symp.*, Aug. 2014, pp. 58–75.

[16] K. J. Sullivan *et al.*, "The galileo fault tree analysis tool," in *Proc. 29th Annu. Int. Symp. Fault-Tolerant Comput.*, Jun. 1999, pp. 232–235.

[17] S. J. Schilling. (2015). "Contribution to temporal fault tree analysis without modularization and transformation into the state space." [Online]. Available: https://arxiv.org/abs/1505.04511

[18] S. Kabir, M. Walker, Y. Papadopoulos, E. Rüde, and P. Securius, "Fuzzy temporal fault tree analysis of dynamic systems," *Int. J. Approx. Reasoning*, vol. 77, pp. 20–37, Oct. 2016.

[19] Z. Peng *et al.*, "Risk assessment of railway transportation systems using timed fault trees," *Qual. Rel. Eng. Int.*, vol. 32, no. 1, pp. 181–194, Feb. 2016.

[20] A. David *et al.*, "Uppaal SMC tutorial," *Int. J. Softw. Tools Technol. Transf.*, vol. 17, no. 4, pp. 397–415, Aug. 2015.

[21] E. Cheshmikhani and H. R. Zarandi, "Probabilistic analysis of dynamic and temporal fault trees using accurate stochastic logic gates," *Microelectron. Rel.*, vol. 55, no. 11, pp. 2468–2480, Nov. 2015.

[22] F. L. Kastensmidt *et al.*, *Fault-Tolerance Techniques for SRAM-Based FPGAs*, vol. 32. New York, NY, USA: Springer, 2006.

[23] P. Adell *et al.*, "Analysis of single-event transients in analog circuits," *IEEE Trans. Nucl. Sci.*, vol. 47, no. 6, pp. 2616–2623, Dec. 2000.

[24] A. H. Johnston, G. M. Swift, T. F. Miyahira, and L. D. Edmonds, "A model for single-event transients in comparators," *IEEE Trans. Nucl. Sci.*, vol. 47, no. 6, pp. 2624–2633, Dec. 2000.

[25] G. B. Hamad *et al.*, "New insights into soft-faults induced cardiac pacemakers malfunctions analyzed at system-level via model checking," *IEEE Access*, vol. 6, pp. 62107–62119, 2018.

[26] M. Kwiatkowska *et al.*, "Performance analysis of probabilistic timed automata using digital clocks," *Formal Methods Syst. Des.*, vol. 29, no. 1, pp. 33–78, Aug. 2006.

[27] S. Friedenthal *et al.*, *A Practical Guide to SysML: System Modeing Language*. Burlington, MA, USA: Morgan Kaufmann, 2014.

[28] F. Mhenni *et al.*, "Automatic fault tree generation from sysml system models," in *Proc. Int. Conf. Adv. Intell. Mechatron.*, 2014, pp. 715–720.

[29] A. Wald, "Sequential tests of statistical hypotheses," *Ann. Math. Statist.*, vol. 16, no. 2, pp. 117–186, 1945.

[30] J. Bechta Dugan, S. J. Bavuso, and M. A. Boyd, "Dynamic fault-tree models for fault-tolerant computer systems," *IEEE Trans. Rel.*, vol. 41, no. 3, pp. 363–377, Sep. 1992.

[31] F. Piedad and M. Hawkins, *High Availability: Design, Technical Processes*. New York, NY, USA:Prentice Hall, 2001.

[32] G. K. Palshikar, "Temporal fault trees," *Inf. Softw. Technol.*, vol. 44, no. 3, pp. 137–150, 2002.

[33] A. Rae *et al.*, "A behaviour-based method for fault tree generation," in *Proc. Int. Syst. Saf. Conf., Syst. Saf. Soc.*, 2004, pp. 289–298.

[34] CORPORATE SPARC International, Inc., *The SPARC Architecture Manual: Version 8*. Upper Saddle River, NJ, USA: Prentice-Hall, 1992.

[35] M. Daněk *et al.*, "The leon3 processor," in *Exploring Fine-Grain Multi-Threading FPGAs*. New York, NY, USA: Springer, 2013, pp. 9–14.

MARWAN AMMAR received the B.S. degree in computer science and information systems from the Regional University of the State of Rio Grande do Sul, Ijui, Brazil, in 2008, and the M.S. degree in electrical and computer engineering from Concordia University, Montreal, QC, Canada, in 2011. He is currently pursuing the Ph.D. degree in electrical and computer engineering with Concordia University. His research interests include high-level modeling and analysis, formal verification techniques, and radiation effects on digital circuits.

GHAITH BANY HAMAD received the B.Sc. degree in electrical and computer engineering from Hashemite University, Jordan, in 2009, the M.A.Sc. degree in electrical and computer engineering from Concordia University, Montreal, Canada, in 2011, and the Ph.D. degree from the Electrical Engineering Department, Polytechnique Montréal, Montreal, Canada, in 2017. His research interests include multi-level reliability analysis, cyber-physical systems, non-functional formal verification, and radiation effects.

OTMANE AIT MOHAMED (M'01) is currently a Professor with the Department of Electrical and Computer Engineering, Concordia University. His research interests include hardware verification, early reliability analysis of cyber physical systems, and radiation effects on microelectronics systems.

He is a member of the Concordia Hardware Verification Group, the Concordia Cyber Security Center, and the Concordia Institute of Aerospace Design & Innovation, and an Executive Member of Microsystems Strategic Alliance of Quebec (ReSMiQ). He is also a long-lasting member of ACM, and a Licensed Engineer with QC, Canada. He was the General Chair and the Program Chair for several conferences, such as IEA-AIE'2018, CIIA'2018, and TOPHOL'2012. He also served as a Reviewer for several related conferences and journals.

YVON SAVARIA (S'77–M'86–SM'97–F'08) received the B.Ing. and M.Sc.A. degrees in electrical engineering from École Polytechnique Montreal, in 1980 and 1982, respectively, and the Ph.D. degree in electrical engineering from McGill University, in 1985.

He has carried work in several areas related to microelectronic circuits and microsystems such as testing, verification, validation, clocking methods, defect and fault tolerance, effects of radiation on electronics, high-speed interconnects and circuit design techniques, CAD methods, reconfigurable computing and applications of microelectronics to telecommunications, aerospace, image processing, video processing, radar signal processing, and digital signal processing acceleration. Since 1985, he has been with Polytechnique Montréal, where he is currently a Professor with the Department of Electrical Engineering. He is also involved in several projects that relate to aircraft embedded systems, radiation effects on electronics, asynchronous circuits design and test, green IT, wireless sensor networks, virtual networks, machine learning, computational efficiency, and application specific architecture design. He holds 16 patents, has published 140 journal papers and 440 conference papers, and he was the Thesis Advisor of 160 graduate students who completed their studies.

He was a member of CMC Microsystems Board. He is a member of the Regroupement Stratégique en Microélectronique du Québec, of the Ordre des Ingénieurs du Québec. He also received a Synergy Award of the Natural Sciences and Engineering Research Council of Canada, in 2006. He was the Program Co-Chairman of NEWCAS'2018. He received a Tier 1 Canada Research Chair, in 2001, on design and architectures of advanced microelectronic systems that he held until 2015. He has been working as a Consultant or was sponsored for carrying research by Bombardier, CNRC, Design Workshop, DREO, Ericsson, Genesis, Gennum, Huawei, Hyperchip, ISR, Kaloom, LTRIM, Miranda, MiroTech, Nortel, Octasic, PMC-Sierra, Technocap, Thales, Tundra, and VXP.