The Journal of Systems & Software 170 (2020) 110766

Contents lists available at ScienceDirect

The Journal of Systems & Software

journal homepage: www.elsevier.com/locate/jss

# Reliability analysis of dynamic fault trees with spare gates using conditional binary decision diagrams

Siwei Zhou [a], Jianwen Xiang [a,*], W. Eric Wong [b,*]

[a] School of Computer Science and Technology, Wuhan University of Technology, Wuhan 430070, PR China
[b] Department of Computer Science, University of Texas at Dallas, Richardson, TX 75080-3021, United States of America

ARTICLE INFO

*Article history:*
Received 11 September 2019
Received in revised form 19 July 2020
Accepted 28 July 2020
Available online 14 August 2020

*Keywords:*
Conditional binary decision diagram (CBDD)
System reliability
Dynamic fault tree
Conditional state transformation
Dependable computing

ABSTRACT

Dynamic fault trees (DFTs) with spare gates have been used extensively in reliability analysis. The traditional approach to DFTs is Markov-based that may suffer from problems like state–space explosion. Algebraic-structure-based methods consume long computation time caused by the inclusive/exclusive formula. Recently, some combinatorial solutions have been applied to DFTs such as sequential binary decision diagrams (SBDD) and algebraic binary decision diagrams (ABDD). They analyze systems by the minimal cut sequence (MCQ) based on sequence-dependence. We propose an analytical method based on conditional binary decision diagrams (CBDD) for combinatorial reliability analysis of non-repairable DFTs with spare gates. A detectable component state is mined to describe the sequence-dependent failure behaviors between components in the spare gate. Minimal cut set (MCS) instead of MCQ is used for qualitative analysis to locate faults via the component state. Compared to Markov-based methods, our method can generate system reliability result with any arbitrary time-to-failure distribution for system components. Different from SBDD and ABDD, specific operation rules are proposed to eliminate inconsistencies and reduce redundancies when building a CBDD. For quantitative analysis, the CBDD simplifies computation via using the sum of disjoint products. Case studies are presented to show the advantage of using our method.

© 2020 Elsevier Inc. All rights reserved.

## 1. Introduction

System reliability is one of the measures of dependability in systems engineering (Commission et al., 2016). Fault Tree Analysis (FTA) is a traditional reliability analysis method that is suitable for system dependable computing. In fault trees, the dynamic fault tree (DFT) (Dugan et al., 1992; Ruijters and Stoelinga, 2015) is an extension of the static fault tree (SFT) (Vesely et al., 2002). The DTF considers functional-dependent failure and sequence-dependent failure. Compared to the static fault tree, the dynamic fault tree includes several dynamic gates such as functional-dependency (FDEP) gate, spare gate, priority-and (PAND) gate, and sequence-enforcing (SEQ) gate. As a result of dynamic behaviors like function-dependency and sequence-dependency, the reliability analysis of high-reliability systems in the critical field becomes complicated. To maintain the desired high-reliability for the system, a high degree of redundancy, dynamic redundancy management, and spares are usually required. Compared to redundancy, spares can flexibly adjust its working state by adopting three kinds of spare gates to balance its required performance and energy consumption. The spare gate (Fig. 1) is widely known as a common design technique for achieving the fault-tolerant system, which can keep the system working despite hardware failures or software errors that may cause the entire system to fail. The spare gate consists of primary components and spare components. According to the state of spare components, the spare gate has three different types: hot spare (HSP [Fig. 1(a)]) gate, warm spare (WSP [Fig. 1(b)]) gate, and cold spare (CSP [Fig. 1(c)]) gate (Dugan et al., 1992; Dugan and Doyle, 1996; Misra, 2008). For considering energy consumption, the HSP gate is the most expensive of the three types of spare gates since the primary component and spare components are both in a working state. The hot spare component can be placed into service immediately when the current primary component (an initial primary component or an activated spare component) fails. It is usually used in applications whose failure resume time is minimal such as A/V switches, computers, network printers, and hard drive data backup systems. In contrast, the CSP gate is the most economical of the three types of spare gates since its spare component is always in an unpowered state before the current primary component failure activates the cold spare component. The cold spare component requires a long time to replace the faulty component in the CSP gate. Hence, the CSP gate is typically applied in places where power is limited such as satellites and conventional submarines. The WSP gate is

* Corresponding authors.
*E-mail addresses:* zhousiwei@whut.edu.cn (S. Zhou), jwxiang@whut.edu.cn (J. Xiang), ewong@utdallas.edu (W.E. Wong).

https://doi.org/10.1016/j.jss.2020.110766
0164-1212/© 2020 Elsevier Inc. All rights reserved.

**Acronyms and abbreviations**

| | |
|---|---|
| FTA | Fault tree analysis |
| SFT | Static fault tree |
| DFT | Dynamic fault tree |
| CFT | Conditional fault tree |
| BDD | Binary decision diagram |
| MDD | Multiple-valued decision diagram |
| CBDD | Conditional binary decision diagram |
| MCQ | Minimal cut sequence |
| MCS | Minimal cut set |
| HSP | Hot spare |
| WSP | Warm spare |
| CSP | Cold spare |

**Assumptions**

| | |
|---|---|
| (1) | The system is non-repairable. |
| (2) | All the spare components supply service in a specified order (from left to right in the spare gate). |

**Notation**

| | |
|---|---|
| $\theta_X$ | Component $X$. |
| $X$ | Failure of component $X$. |
| $\Diamond$ | Boolean operator (and/or). |
| $+, \cdot, \neg$ | Basic Logic OR, AND, Negation |
| $\triangleleft$ | Temporal non-inclusive BEFORE (BF) operator |
| $\rightarrow$ | Precedence order of component failures |

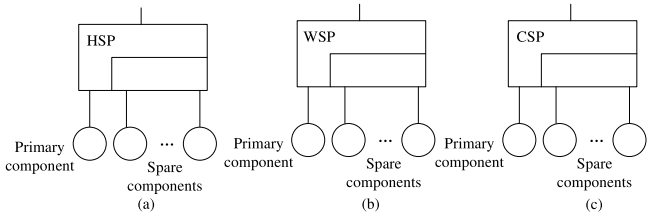



**Fig. 1.** Three types of the spare gate. (a) HSP, (b) WSP and (c) CSP.

a compromise solution between the HSP gate and the CSP gate since the warm spare component is in a dormant (standby) state that is power-on but not actively operating. Compared to the HSP gate, the WSP gate consumes much less power before the warm spare component is activated. Compared to the CSP gate, the WSP gate has a shorter response time to let the spare component replace the faulty component and restore the system. The WSP gate is commonly used in sensor networks, disk management systems, and vehicle management systems.

In the DFT reliability analysis, the spare gate is one of the most complicated cases. It includes sequence-dependent failure behaviors and different failure probabilities in the same spare component in different states. Spare gate fires when all components including the primary component and the spare components fail (shared spare component may be occupied rather than fail). In the CSP gate, all the spare components may fail only after the current primary component fails. Note that cold spare component is not considered to have failed before it is activated (Dugan et al., 1992). In this case, the cold spare component has two failure probabilities: one is 0 before it is activated and the other is $Pr_{C_a}$ after it is activated. Unlike the cold spare component, however, the warm spare component has no sequential fault restriction if specific failure states are not concerned; it also has two failure probabilities: one is $Pr_{W_d}$ in the dormant state and the other is $Pr_{W_a}$ in the working state. In the HSP gate, the hot spare component failure is neither sequence-dependent nor has different probabilities. Hence, if we only consider failure behaviors, an HSP gate with non-shared spares is equivalent to a logic AND gate, which has no sequence-dependent failure behaviors. As a result, the spare gate in this paper mainly involves CSP and WSP gates except for Fig. 2 (hot spare mode). Current approaches to reliability analysis of dynamic fault trees (DFTs) with spare gates are Markov-based methods (Misra, 2008), simulation-based methods (Long, 2002; Merle et al., 2016), Bayesian-network-based methods (Boudali and Dugan, 2005; Kabir et al., 2014) and algebraic-structure-based methods (Merle et al., 2011a,b). Markov-based methods may suffer from the state–space explosion problem when the scale of DFTs is large (Ruijters and Stoelinga, 2015). However, it can achieve the reliability analysis of large DFT via reducing state–space for cases mentioned in Volk et al. (2016), Volk et al. (2018). Also, it is only suitable for the exponential time-to-failure probability distribution of basic events. Simulation-based methods eliminate the exponential time-to-failure distribution restriction, but they cannot offer accurate results. Bayesian-network-based methods have similar computation complexity troubles as the Markov-based methods. With the help of temporal logic, algebraic-structure-based methods (Merle et al., 2011a) use a symbol $\triangleleft$ to denote a sequential relationship "before". Algebraic-structure-based methods can handle any arbitrary time-to-failure probability distribution but they may require using the I/E (Inclusive/Exclusive) formula for reliability computation. However, they will relate to a huge amount of computation when minimal cut sequences (MCQs) are too numerous. In recent years, BDD-based methods are used for DFTs analysis. Converting the fault tree to binary decision diagram (BDD) is an efficient method to find MCS (Akers and B, 1978). After converting, the BDD has exponential complexity in the worst case, but it also has linear complexity in the best case (Ruijters and Stoelinga, 2015).

We mine a detectable component state in spare gates. According to this specific state, we create a certain conditioning event that implies sequence-dependent behaviors in spare gates. Thus, the DFT described in our paper is a fault tree that consists of spare gates and static logic gates (logic AND gate and/or logic OR gates). Our proposed method also is a combinatorial solution involving an extended BDD.

Firstly, we convert the DFT to the conditional fault tree (CFT) by using some conditioning events that relate to the status of the component. Secondly, according to the proposed rules, a system CBDD model corresponding to the CFT is built. At last, a system CBDD model can be evaluated by translating paths from the top to terminal "1" to algebraic expressions of sequence-dependent.

Note that, the DFT described in our paper is a fault tree that consists of spare gates and static logic gates (logic AND gate and/or logic OR gates). We do not consider voting gates separately since it can be replaced by the combination of logic OR and AND gates (Ruijters and Stoelinga, 2015).

The remainder of this paper is organized as follows. Section 2 presents recent related work regarding reliability analysis using BDDs and multiple-valued decision diagrams (MDDs). Section 3 introduces some concepts of static transform and basics of BDD. Section 4 presents the conditional state transformation. Section 5 shows the construction of the CBDD. How to use a CBDD for reliability analysis based on DFTs with spare gates is presented in Section 6. In Section 7, three practical DFT case studies are illustrated for the reliability analysis using CBDD in detail. Section 8 concludes the paper.

## 2. BDD Related work

Recently, many BDD-based methods were used in reliability analysis or related field. A new and efficient BDD-based method was utilized for performability analysis of *k*-to-*l*-out-of-*n* computing systems (Mo et al., 2018), An improved BDD can reduce memory consumption and computation time by combining truncation with modularization, which quickly obtains an accuracy probability in the fault tree analysis if a proper truncation probability is found (Deng et al., 2015). A BDD assisted with a dynamic labeling method is proposed for non-coherent fault tree analysis. The dynamic labeling can be used to reduce the number of intersections to be calculated for the determination of prime implicants (Matuzas and Contini, 2015). A BDD combined with an incremental method is used for the quantification of sequences of linked fault trees, adopting a reduction procedure that can be used individually to each fault tree defined in the sequence (Ibáñez-Llano et al., 2010). However, sequence-dependent behaviors are not considered. An analytical and combinatorial method based on sequential BDD (SBDD) was proposed for the analysis of non-repairable standby systems (Xing et al., 2012; Tannous et al., 2011) and it was improved by creating a heuristic variable index to keep the scale of resultant cut sequences as small as possible (Ge et al., 2015). However, the SBDD cannot eliminate sequence-dependent at prime events level, and it removes invalid nodes after the final SBDD has been built. The SBDD (Rauzy, 2011) that was inspired by Minato's Zero-Suppressed BDD (Minato, 1993) used a new data structure to encode sets of sequences of basic events. All rules related to sequence algebras can be operated based on the proposed data structure. Nevertheless, it is similar to algebraic-structure-based methods and allows complex sequence-operation rules in the SBDD. Also, the SBDD is not developed based on a well-defined temporal or sequential algebra. Hence, the SBDD is complicated when conducting both qualitative and quantitative analysis of a large-scale DFT. An algebraic BDD (ABDD) on algebraic-structure-based methods was applied for the analysis of DFTs (Jiang et al., 2018), but it eliminates the invalid path after the final ABDD has been completed and relates to complex operations with temporal logical failure behaviors as well. MDD, which is an extended version of BDD, has mainly been used for multiple-state system reliability analysis. The MDD inherits all the advantages of BDD, such as no restriction on the time-to-failure distribution for the basic event in the DFT and reliability computed by the sum of disjoint product (SDP). A generalized MDD was proposed for the reliability analysis of fault-tolerant systems (Xiang et al., 2016). A multi-state MDD was presented for the analysis of multi-state systems with multi-state components (Xing and Dai, 2009). A new MDD-based analysis technique was proposed for the reliability analysis of network systems with dependent propagation effects (Mo et al., 2016). An MDD is proposed to evaluate cold standby systems via supplementing a mark to the activated event (Zeng, 2019). Some MDDs were used for the analysis of warm standby systems and cold standby systems (Zhai et al., 2013, 2015a,b) based on fault coverage models. An efficient MDD-based method was proposed for the reliability analysis of binary-state phased-mission systems by using independent multi-valued variables to encode component failure behavior across phases (Peng et al., 2014). However, they conduct reliability analysis based on coverage models or multi-state systems rather than the DFT model. An efficient MDD-based DFT analysis approach for computing the reliability measures of large dynamic subtrees was proposed (Mo, 2014). This method can relieve the state–space explosion problem by identifying whether subtree components relate dynamic failure behaviors, but it still has the exponential limitation on basic events in dynamic gates.

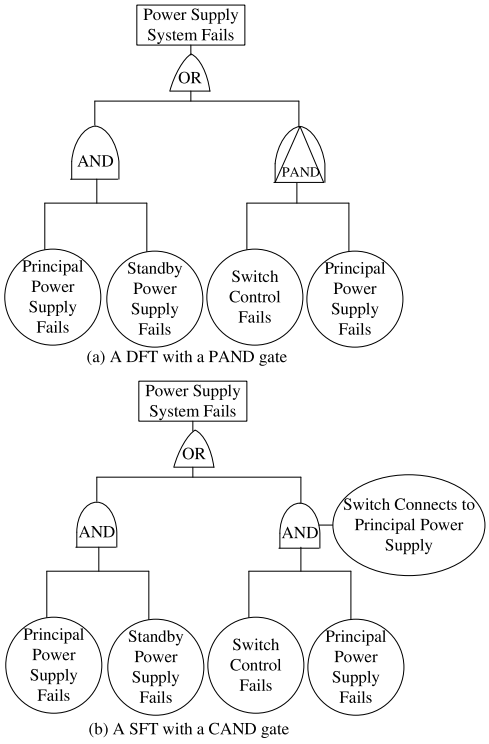
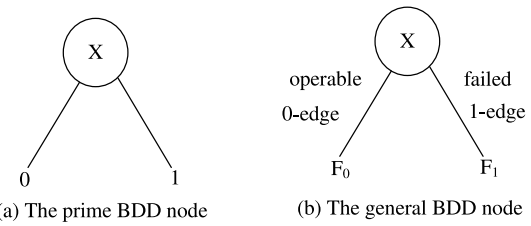



**Fig. 2.** A simple example of the static transformation (Xiang et al., 2013).

The above methods either cannot be directly available to DFTs or involve state–space explosions or complex sequence-dependent failure behaviors. Unfortunately, sequence-dependent failure behaviors may cause a permutation problem due to cut sequence searching in DFTs, and a permutation problem usually involves a factorial complexity problem (Dixon and Mortimer, 1996). In this paper, an analytical and combinatorial method is proposed for reliability analysis of non-repairable DFTs with spare gates while addressing problems of the existing approaches. The proposed method is based on conditional binary decision diagram (CBDD), an extended version of BDD (Bryant, 1986; Rauzy, 1993).

It is easy to understand that searching cut sets in SFTs is a mathematical combinatorial problem. The complexity of combinatorial problems is much less than that of permutation problems (Biggs and White, 1979). To reduce the complexity, a static conditional transformation of spare gates is proposed. A new conditional fault tree (CFT) is built with the static conditional transform, and the minimal cut set (MCS) replaces the MCQ. In the CFT, permutation problems are changed into combinatorial problems, since sequence-dependent failure behaviors are converted into conditional states in DFTs with spare gates. In this paper, a novel BDD (CBDD) based on the CFT is propounded for reliability analysis of DFTs with spare gates.

Note that, in this paper, different from the state based on state–space methods, the conditional state is located in the component (basic event) level rather than the gate (or system) level. We only focus on the entire system reliability evaluation and assume that the time to failure of the system components satisfies the continuous probability distribution function (PDF). The PDF of the system components can be estimated by methods of statistical inference (ALLEN AO, 1990) if it is unknown or partially unknown.




(a) The prime BDD node (b) The general BDD node

**Fig. 3.** Two BDD nodes: (a) the general BDD node (b) the prime BDD node.

## 3. Static transform & binary decision diagram

A transformation from dynamic PAND gates to static AND gates with some dependent conditioning events has been presented in Xiang et al. (2013). The fault tree conditioning event relates to some specific conditions or restrictions that are used to any logic gate. With the help of the transformation, the MCQ of a DFT with PAND gates is converted into the MCS of a conditional SFT. After the transformation, the conditional AND gate is called the CAND gate. It can represent the priority relations that cannot be modeled by the PAND gate. A simple example of the transformation is shown in Fig. 2. In the CAND gate, the conditioning event "Switch Connects to Principal Supply" indicates the switch fails before the principal supply fails. In other words, it implies the sequence-dependent failure behaviors between components. The specific conditioning event can be used to achieve static transformation for PAND gates (Xiang et al., 2013). If we only consider failure behaviors, in most cases, the SEQ gate is likely to be replaced by a special case of CSP gates with non-shared spare components (Ruijters and Stoelinga, 2015; Manian et al., 1999) and the FDEP gate can be translated into a logic OR gate (Merle et al., 2011b). In some special scenarios (Boudali et al., 2010; Junges et al., 2016), it cannot achieve translation rules regarded as SEQ gates and FDEP gates. For example, an SEQ gate with a logical AND gate as the second child or an FDEP gate whose two dependent events are the inputs in a PAND gate. However, the static transformation-related complete solution of spare gates is not proposed.

BDD was first used as a new algorithm for fault tree analysis in Rauzy (1993). Unlike other methods of fault tree reliability analysis, BDD requires less memory and computation time. It is a rooted acyclic graph based on the Shannon decomposition as follows:

$$f = x \cdot f_{x=1} + \neg x \cdot f_{x=0} = ite(x, F_1, F_0) \tag{1}$$

In Eq. (1), $f$ denotes a Boolean expression and $x$ denotes a Boolean variable in $f$. $F_1$ and $F_0$ equal $f_{x=1}$ and $f_{x=0}$, respectively. $f_{x=1}$ and $f_{x=0}$ represent $f$ evaluated as $x$ being one and zero, respectively. *ite* represents the concise *if* − *then* − *else* format. The BDD has two branches, labeled by 0-edge and 1-edge, representing the operable state and failed state, respectively. The two BDD terminal nodes are logic value 0 and value 1. The general BDD node and prime BDD node are shown in Fig. 3.

An SFT can be converted to the BDD by recursively using the following operation rules set out in Bryant (1986). Let $\Diamond$ represent any logic operation (AND/OR). $g$ and $h$ represent two Boolean expressions corresponding to the traversed sub STFs, and then we have $g \Diamond h = ite(x, G_1, G_0) \Diamond ite(y, H_1, H_0) =$

$$\begin{cases} ite(x, G_1 \Diamond H_1, G_0 \Diamond H_0) & index(x) = index(y); \\ ite(x, G_1 \Diamond H, G_0 \Diamond H) & index(x) < index(y); \\ ite(y, G \Diamond H_1, G \Diamond H_0) & index(x) > index(y); \end{cases}$$

where *index* represents the Boolean variable order based on the heuristic method.

If BDDs are suitable for static fault trees, the DFT also could be converted to a certain BDD after static transformation. Thus, our combinatorial solution addressing this is presented in the following sections.

## 4. Conditional state transformation

A component in dynamic gates has some detectable static states when it is in different conditions such as operable, failure, and other states. In spare gates, a spare component has two states: working state and standby state. Precisely, the standby state can be divided into two states, un-power state and dormant state, which are respectively located in the CSP gate and the WSP gate. The spare component in WSP gates can fail in the dormant state. When a primary component fails, a spare component is activated and replaces the failed primary component. At this point, the spare component is working before it fails. The spare component in the working state plays a role of the primary component. It can be replaced by the other spare component as well. Generally speaking, the spare component can replace a replaceable component. The replaceable component refers to a primary component or a working spare component in the spare gate. Also, we consider that if there is an available spare component, a replaceable component will be replaced when it fails, and a spare component is operational when it replaces a component.

### 4.1. Conditioning event

A particular event may occur in the spare gate, which can determine the replacement behaviors between components. Here we introduce a "replacement" symbol *rep* to represent this conditioning event, which exists only in spare gates. $rep(\theta_1, \theta_2)$ denotes the conditioning event that a component $\theta_1$ replaces a component $\theta_2$, where $\theta_1$ can only be a spare component while $\theta_2$ is a replaceable component (either a primary component $\theta_P$ or a working spare component $\theta_S$). Accordingly, $\neg rep(\theta_1, \theta_2)$ denotes that the replacement between $\theta_1$ and $\theta_2$ has never happened. It is easy to understand that $rep(\theta_S, \theta_S)$ represents the conditioning event that $\theta_S$ never replaces any other component, as it is in the dormant state at all time. In this case, $rep(\theta_S, \theta_S) \cdot S$ denotes the warm spare component $\theta_S$ fails in the dormant state. $\neg rep(\theta_S, \theta_S)$ denotes $\theta_S$ replaces a certain component. It implies $\theta_S$ is activated by a replaceable component. In the algebraic-structure-based method (Merle et al., 2011a), $S_a$ denotes a spare component $\theta_S$ fails after it replaces a component, and $S_d$ denotes a spare component $\theta_S$ fails while no replacing any other components. As opposed to $S_a$ and $S_d$, the *rep* (conditioning event) does not imply the spare component $\theta_S$ failure.

### 4.2. Conditional fault tree

Thanks to *rep*, the combination of component states can be considered to supersede the sequence-dependent behaviors between components in the spare gate. Based on the specific *rep*, if we only consider the non-shared spare, the WSP gate or the CSP gate can be converted into a kind of conditional SFT called CFT, as shown in Fig. 4. For the shared spare WSP gate and shared spare CSP gate, primary components could be affected by each other due to the occupation of the shared spare component. The conversion of shared spare gates (WSP gate and CSP gate) is shown in Fig. 5. For Fig. 5, it is easy to observe that the occupation of a shared spare is symmetrical for TE1 and TE2. Hence, if TE1 and TE2 are combined with a logic OR gate, we only consider the failure of $P_1$ and $P_2$ for the logic OR gate, but not which one will be replaced first. For $m$ primary components and $n$ shared spare components, each primary component corresponds to a spare

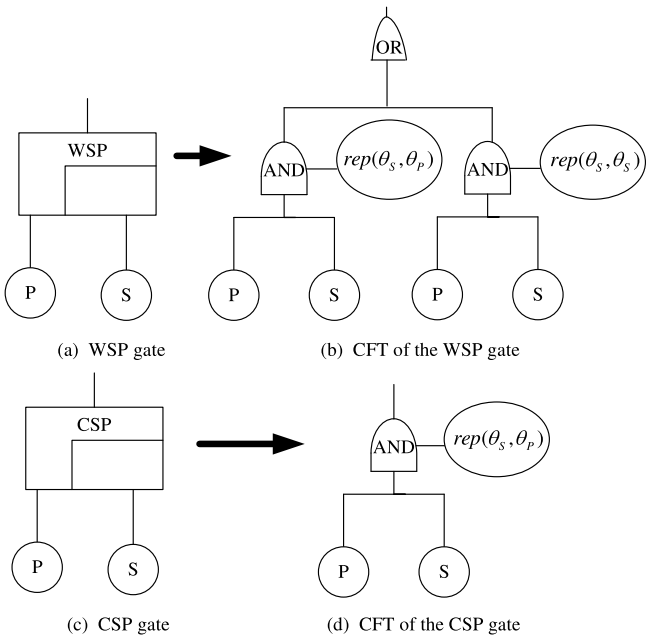



**Fig. 4.** Converting the DFT into the CFT.

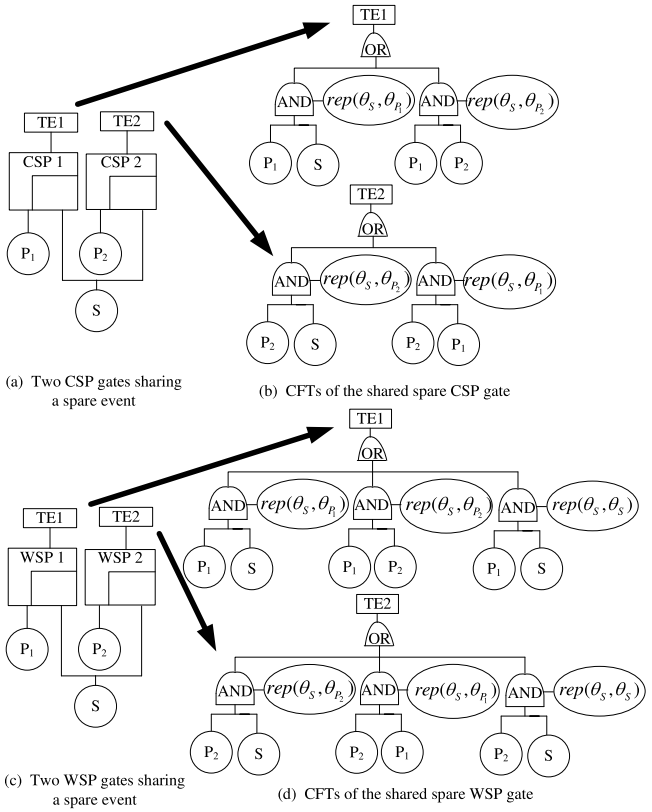



**Fig. 5.** Converting the shared spare SP gates into the CFT.

gate, and we can deduce that one of the shared spare SP gates is bound to happen if $n+1$ components fail. Furthermore, it can be obtained that one of the shared SP gates will happen if $n+1$ primary components fail and $m > n$.

Compared with the sequence-dependent failure behavior, the static transformation expression of spare gates can describe both the location and failure state of basic events. Thus, it can more accurately describe the failure process of the spare gate. Furthermore, a combination of primary events (Vesely et al., 2002) and conditioning events, instead of primary event permutation, describes the dynamic failure behavior of DFTs with spare gates.

### 4.3. Minimal cut set vs minimal cut sequence

As a result of conditional states replacing the sequence-dependent failure behaviors, MCQ is no longer suitable for qualitative analysis of a CFT. Hence, MCS is considered to be used in qualitative analysis. However, basic events in the MCS are typically primary events (Vesely et al., 2002) which are assumed to be independent. Obviously, the conditioning events in the CFT are dependent events. According to the definition of basic events, the conditioning event is a type of basic events (Vesely et al., 2002). The MCS is the smallest combination of basic events that result in the top event. Thus, MCS can also be applied to CFTs, but the inconsistency and redundancy caused by conditioning events need to be eliminated. The replacement can only happen when the replaceable component fails. The following rules can be introduced to solve the redundancy problem:

$$P \cdot rep(\theta_S, \theta_P) = rep(\theta_S, \theta_P) \tag{2}$$

$$S' \cdot rep(\theta_S, \theta_{S'}) = rep(\theta_S, \theta_{S'}) \tag{3}$$

Eqs. (2) and (3) indicate if a replacement happens and if the corresponding replaceable component was bound to fail.

If a replaceable component is operational, then it is impossible to be replaced (it refers to Eqs. (4) and (5)). For a spare component, "never replace" and "replace" cannot happen at the same time, which refers to Eq. (6). It is also impossible for a spare component to "never replace" and "be replaced" simultaneously, which refers to Eqs. (7) and (8). A spare cannot replace more than one replaceable component at the same time as well, which refers to Eqs. (9) and (11). For the same reason, a replaceable component cannot be replaced twice or more simultaneously, which refers to Eqs. (10) and (12). Based on the above analysis, the following rules can be introduced to solve the inconsistency problem:

$$\neg P \cdot rep(\theta_S, \theta_P) = 0 \tag{4}$$

$$\neg S' \cdot rep(\theta_S, \theta_{S'}) = 0 \tag{5}$$

$$rep(\theta_S, \theta_P) \cdot rep(\theta_S, \theta_S) = 0 \tag{6}$$

$$rep(\theta_S, \theta_{S'}) \cdot rep(\theta_S, \theta_S) = 0 \tag{7}$$

$$rep(\theta_S, \theta_{S'}) \cdot rep(\theta_{S'}, \theta_{S'}) = 0 \tag{8}$$

$$rep(\theta_S, \theta_P) \cdot rep(\theta_S, \theta_{P'}) = 0 \tag{9}$$

$$rep(\theta_S, \theta_P) \cdot rep(\theta_{S'}, \theta_P) = 0 \tag{10}$$

$$rep(\theta_S, \theta_{S'_1}) \cdot rep(\theta_S, \theta_{S'_2}) = 0 \tag{11}$$

$$rep(\theta_{S_1}, \theta_{S'}) \cdot rep(\theta_{S_2}, \theta_{S'}) = 0 \tag{12}$$

where $\theta_P$ and $\theta_{P'}$ are distinct primary components in the shared spare gate. $\theta_{S'_1}$ and $\theta_{S'_2}$ are distinct working spare components in the same spare gate. $\theta_{S_1}$ and $\theta_{S_2}$ are distinct spare components in the same spare gate. In this paper, the conditioning event only refers to the relationship between replaceable components and spare components in spare gates. However, the conditioning event is not a primary event (the primary component failure), though it can imply the replaceable component failure. The spare gate only fires when all components fail. Hence, the conditioning event cannot appear in the MCS singly. It combines with at least one primary event. For the spare gates shown in Fig. 4, the MCS is presented as follows:

$$CSP : rep(\theta_S, \theta_P) \cdot S$$

$$WSP : rep(\theta_S, \theta_P) \cdot S + P \cdot rep(\theta_S, \theta_S) \cdot S$$

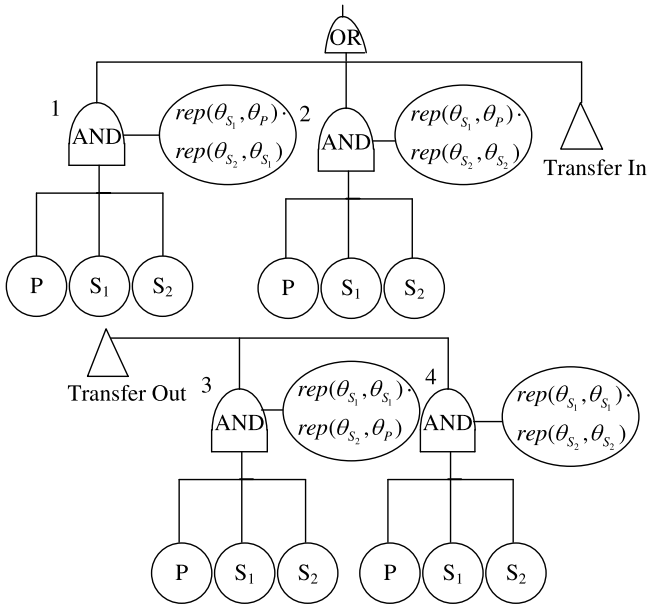



**Fig. 6.** CFT of the WSP gate with two spare components.

As we know, the CSP gate can be regarded as a special case of the WSP gate if the failure of the spare component in the dormant is not considered. Thus, the following part of this paper focuses on WSP gates. For a WSP gate, the MCS is the product of all primary events in the WSP gate if the specific failure status of the spare component is not distinguished. However, the failure probability of a spare component is different before or after the replacement. In quantitative analysis, it will be divided into two sub-events. For the convenience of engineering applications, these differences will be reflected in the MCS. For example, a WSP gate with two spare components has a primary components $\theta_P$ and two spare components $\theta_{S_1}$ and $\theta_{S_2}$. It can be converted into a CFT, as shown in Fig. 6. Four MCSs of the CFT are as follows:

$$
\begin{aligned}
&rep(\theta_{S_1},\theta_P)\cdot rep(\theta_{S_2},\theta_{S_1})\cdot S_2\\
&+rep(\theta_{S_1},\theta_P)\cdot rep(\theta_{S_2},\theta_{S_2})\cdot S_1\cdot S_2\\
&+rep(\theta_{S_1},\theta_{S_1})\cdot rep(\theta_{S_2},\theta_P)\cdot S_1\cdot S_2\\
&+P\cdot rep(\theta_{S_1},\theta_{S_1})\cdot rep(\theta_{S_2},\theta_{S_2})\cdot S_1\cdot S_2
\end{aligned}
$$

If MCQs are used for the above example, the six MCQs are as below:

$$
\begin{aligned}
&(P\lhd S_{1_a}\lhd S_{2_a})\cdot S_{2_a}\\
&+(P\lhd S_{2_d}\lhd S_{1_a})\cdot S_{1_a}\\
&+(S_{2_d}\lhd P\lhd S_{1_a})\cdot S_{1_a}\\
&+(S_{1_d}\lhd P\lhd S_{2_a})\cdot S_{2_a}\\
&+(S_{1_d}\lhd S_{2_d}\lhd P)\cdot P\\
&+(S_{2_d}\lhd S_{1_d}\lhd P)\cdot P
\end{aligned}
$$

The symbol $\lhd$ denotes a sequential relationship "before". For example, if $A$ and $B$ are two basic events in a DFT, then $A\lhd B$ denotes that only when the event $A$ occurs before event $B$, the event $(A\lhd B)$ will occur.

For the WSP gate with two spare components, it is evident that the quantity of MCSs is less than that of MCQs. This is because some MCSs can imply two MCQs. $P\cdot rep(\theta_{S_1},\theta_{S_1})\cdot rep(\theta_{S_2},\theta_{S_2})\cdot S_1\cdot S_2$ implies $(S_{1_d}\lhd S_{2_d}\lhd P)\cdot P$ and $(S_{2_d}\lhd S_{1_d}\lhd P)\cdot P$, and $rep(\theta_{S_1},\theta_P)\cdot rep(\theta_{S_2},\theta_{S_2})\cdot S_1\cdot S_2$ implies $(P\lhd S_{2_d}\lhd S_{1_a})\cdot S_{1_a}$ and $(S_{2_d}\lhd P\lhd S_{1_a})\cdot S_{1_a}$.

The general non-shared spare case is the WSP gate with $n$ spare components. In this case, the WSP gate firing will be satisfied if:

(i) All the components fail.

(ii) Each spare component can only fail in one of the two different states simultaneously. One is the failure without replacement, and the other is the failure after replacement.

According to the inconsistency rules (9) and Eq. (11), it is impossible for a spare component to replace two different replaceable components at the same time. Hence, a spare component has a unique replacement failure (failing after replacement) in one MCS. The MCS that leads to the WSP gate firing is related to a simple combination of the primary component failure state and the spare component failure state. A primary component has only one failure state, and each spare component has two possible failure states. In total, $n$ spare components have $2^n$ possible failure states. Therefore, the quantity of combinations of possible failure states in the non-shared general case is $1\cdot 2^n=2^n$. Correspondingly, the quantity of MCSs is also $2^n$. Compared to the MCS, MCQ is a sequence permutation with all primary events (one primary component failure and $n$ spare components failure) in the WSP gate. In the general non-shared spare case, the quantity of MCQs is $P_{n+1}^{n+1}$:

$$P_{n+1}^{n+1}=(n+1)\cdot n\cdot(n-1)\cdot(n-2)...\cdot 1>2^n,\ \ where\ n>1$$

In the qualitative analysis, the quantity of MCSs is less than that of MCQs if the WSP gate has two or more spare components. Also, as the quantity of spare components increases, the difference is more significant.

The general shared spare case is $m$ WSP gates with $n$ shared spare components. In this case, a single WSP gate firing will be satisfied if:

(i) Its primary component fails.

(ii) All the spare components are exhausted. Each shared spare component can only be consumed in one of three different states at the same time. One is the failure without replacement, one is the failure after replacement, and the third is the replacement in the not-self WSP gate.

Each shared spare component has two failed states that are the same as those in the non-shared WSP gate. However, it is different from the case of non-shared WSP gates. The shared spare component has one more state (occupied state) that replaces the replaceable component in the not-self WSP gate. The possible quantity of occupied states is related to the number of primary components in the shared WSP gate. In the general shared case, the quantity of primary components is $m$, and then the shared spare component has $m-1$ possible occupation states. Therefore, the shared spare component has $m+1$ $((m-1)+2)$ possible consumed states. In total, $n$ spare components have $(m+1)^n$ possible consumed states. Similarly, combined with the primary component failure state, the quantity of MCSs of a single WSP gate in general shared case is $1\cdot(m+1)^n=(m+1)^n$. If $m=1$, it is converted into $2^n$, which is a general case of non-shared spare. In other words, $(m+1)^n$ can also solve the general non-shared spare case. Thus, it is considered a general formula for a WSP gate failure. For the general shared spare case, if $n+1$ components fail, one of the WSP gates is bound to fire. Additionally, the probability of occurrence of each WSP gate, in this case, is the same. Hence, the quantity of MCQs of a single WSP gate in a general shared case is $P_{n+m}^{n+1}/m$.

$$
\begin{aligned}
P_{n+m}^{n+1}/m=&(m+n)\cdot(m+n-1)...(m+1)>(m+1)^n\\
&where\ n>1.
\end{aligned}
$$

For example, two WSP gates (with two primary components $\theta_{P_1}$, $\theta_{P_2}$) share two spare components $(\theta_{S_1},\theta_{S_2})$. Nine MCSs can cause one of the WSP gates to fail (the WSP gate with $\theta_{P_1}$), as follows:

$$
\begin{aligned}
&The\ quantity\ of\ MCSs: (2+1)^2=9\\
&rep(\theta_{S_1},\theta_{P_1})\cdot rep(\theta_{S_2},\theta_{S_1})\cdot S_2
\end{aligned}
$$

$$+ rep(\theta_{S_1}, \theta_{S_1}) \cdot rep(\theta_{S_2}, \theta_{P_1}) \cdot S_1 \cdot S_2$$
$$+ rep(\theta_{S_1}, \theta_{P_1}) \cdot rep(\theta_{S_2}, \theta_{S_2}) \cdot S_1 \cdot S_2$$
$$+ P_1 \cdot rep(\theta_{S_1}, \theta_{S_1}) \cdot rep(\theta_{S_2}, \theta_{S_2}) \cdot S_1 \cdot S_2$$
$$+ P_1 \cdot rep(\theta_{S_1}, \theta_{P_2}) \cdot rep(\theta_{S_2}, \theta_{S_1})$$
$$+ rep(\theta_{S_1}, \theta_{P_2}) \cdot rep(\theta_{S_2}, \theta_{P_1}) \cdot S_2$$
$$+ rep(\theta_{S_1}, \theta_{P_1}) \cdot rep(\theta_{S_2}, \theta_{P_2}) \cdot S_1$$
$$+ P_1 \cdot rep(\theta_{S_1}, \theta_{P_2}) \cdot rep(\theta_{S_2}, \theta_{S_2}) \cdot S_2$$
$$+ P_1 \cdot rep(\theta_{S_1}, \theta_{S_1}) \cdot rep(\theta_{S_2}, \theta_{P_2}) \cdot S_1$$

In the same condition, the quantity of MCQs is twelve.

$$\textit{The quantity of MCQs} : P_4^3/2 = 12$$
$$(P_1 \lhd S_{1_a} \lhd S_{2_a}) \cdot S_{2_a} + (P_1 \lhd S_{2_d} \lhd S_{1_a}) \cdot S_{1_a}$$
$$+ (S_{1_d} \lhd P_1 \lhd S_{2_a}) \cdot S_{2_a} + (S_{1_d} \lhd S_{2_d} \lhd P_1) \cdot P_1$$
$$+ (S_{2_d} \lhd P_1 \lhd S_{1_a}) \cdot S_{1_a} + (S_{2_d} \lhd S_{1_d} \lhd P_1) \cdot P_1$$
$$+ (P_2 \lhd S_{2_d} \lhd P_1) \cdot P_1 + (P_2 \lhd P_1 \lhd S_{2_a}) \cdot S_{2_a}$$
$$+ (S_{1_d} \lhd P_2 \lhd P_1) \cdot P_1 + (S_{2_d} \lhd P_2 \lhd P_1) \cdot P_1$$
$$+ (P_2 \lhd S_{1_a} \lhd P_1) \cdot P_1 + (P_1 \lhd P_2 \lhd S_{1_a}) \cdot S_{1_a}$$

Typically, in mathematical theory, the complexity of the combination problem is smaller than that of the permutation problem (Lucey and Lucey, 2002). Generally speaking, the quantity of MCSs of a single WSP gate is less than that of MCQs whether it is a WSP gate that has shared or non-shared spare components if the spare components are more than one. The more spare components and the more shared WSP gates, the higher the difference. Furthermore, when the WSP gate is combined with the static gate, the difference will become more significant as well. Additionally, our MCS is more intuitive and straightforward than the MCQ for describing dynamic behaviors in spare gates since *rep* can directly denote replacement between components.

## 5. Conditional binary decision diagram

Our BDD is an extension of the traditional BDD, as it involves *s*-dependent nodes that are converted by conditioning events *rep*. Thus, to distinguish other BDDs, our BDD is called Conditional BDD (CBDD). The node in traditional BDD is considered to be *s*-independent, but the node in CBDD may not be since the conditioning node is converted from the conditioning event. In CBDD, conditioning nodes are not the same as sequential nodes in SBDD (Ge et al., 2015) and ABDD (Jiang et al., 2018), since they cannot be represented alone and have different dependent relation. Hence, it is assumed that all of the nodes in CBDD are *s*-independent. However, the dependent relation between CBDD nodes will be considered in the quantitative analysis. Similar to the traditional BDD, an ordering strategy of nodes is always needed since the order of input variables is also heavily relevant to the size of the CBDD. However, the solution in Rauzy (1993) cannot be directly used to the CBDD due to conditioning nodes. Also, more operational rules will be introduced to eliminate some redundancy and inconsistency.

A few solutions were proposed for an ordering strategy of BDD nodes such as depth-first-based methods (Bouissou, 1996), heuristics-based methods (Mo et al., 2013), neural network-based methods (Bartlett and Andrews, 2002), and neighbor-first-based methods (Sun and Du, 2008). However, there is no generic solution for the ordering strategy of BDD nodes. In this paper, we choose a neighbor-first-based method called progressive neighbor first ordering (PNFO) since it can reduce the number of BDD nodes as much as possible. PNFO evaluates logical relationships among the basic events in the CFT and is not influenced by the way the CFT has been constructed. Also, it focuses on repeated events and gives the ordering priority to their neighbor events (Sun and Du, 2008).

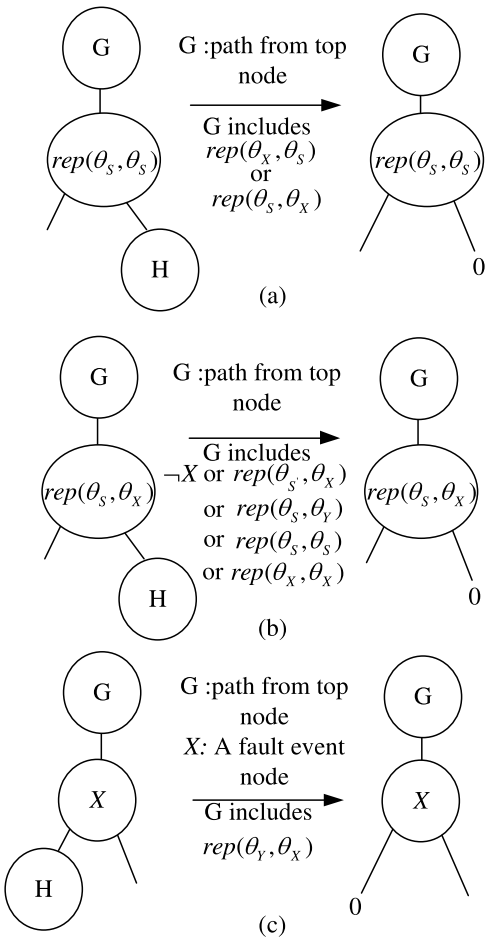



**Fig. 7.** Operation rules for inconsistency for CBDD generation.

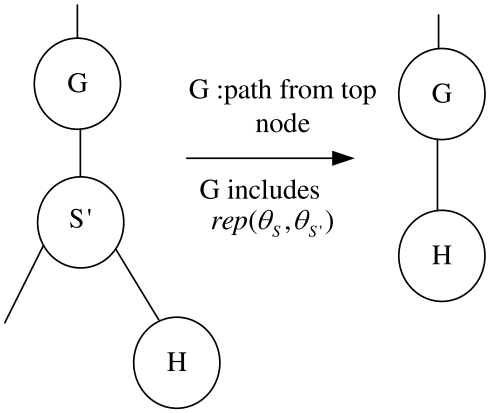



**Fig. 8.** Simplification rule for CBDD generation.

To eliminate the inconsistency problem caused by conditioning nodes during the logical operation between CBDDs, several rules are introduced based on Eq. (4) to Eq. (12), as shown in Fig. 7.

As the CBDD is built, simplification rules are available to ensure that the final CBDD is minimal based on the index order of chosen nodes. Besides simplification rules of traditional BDD, there is a simplification rule of CBDD that is introduced as follows:

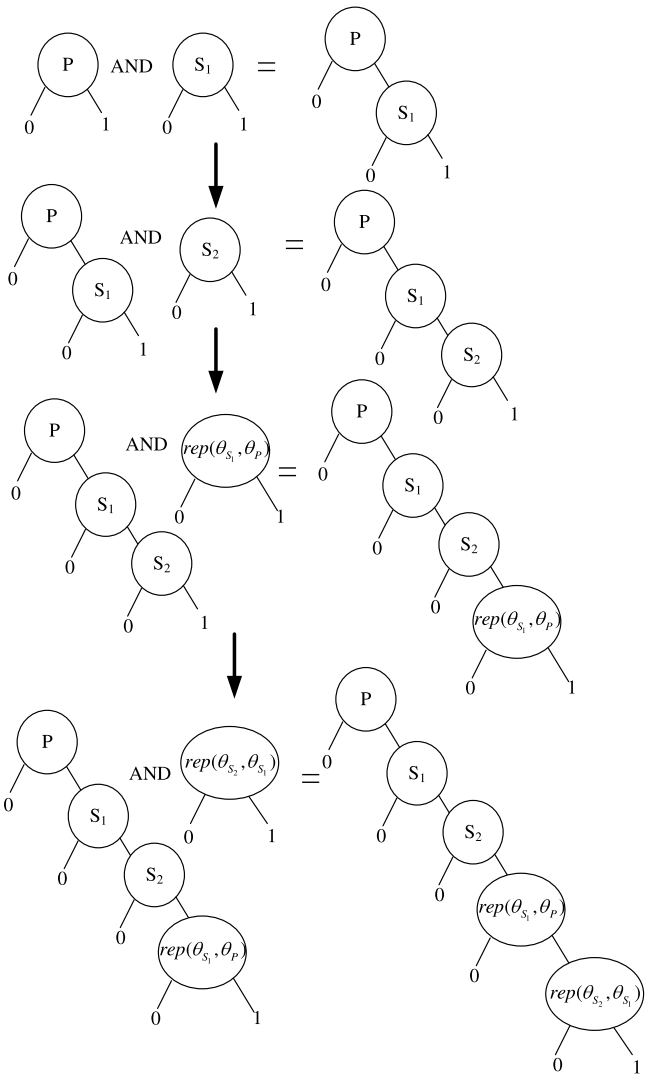



**Fig. 9.** No. 1 gate CBDD generation.

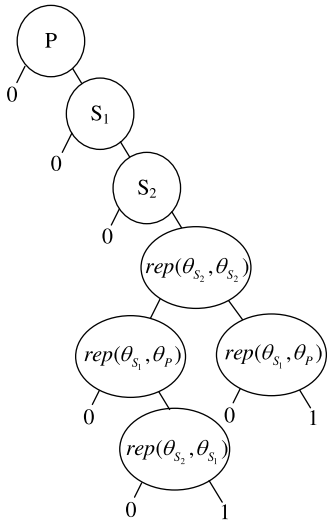



**Fig. 10.** Sub_ CBDD generated by No.1 gate CBDD logic OR operation No.2 gate CBDD.

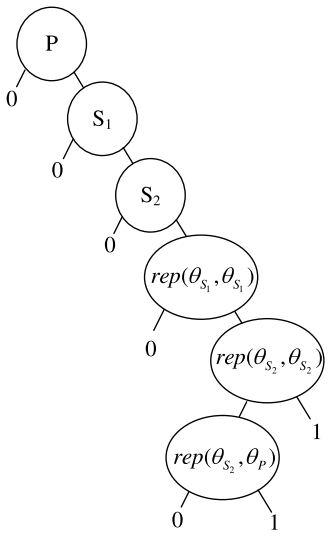



**Fig. 11.** Sub-CBDD generated by No.3 gate CBDD logic OR operation No. 4 gate CBDD.

- *Simplification Rule*. Replace the redundant node S' to its 1-edge sub_node $H$, if the node S' and the conditioning node $rep(\theta_S, \theta_{S'})$ are on the same path, as shown in Fig. 8.

We illustrate how to construct the CBDD using a WSP gate. The CFT shown in Fig. 6 consists of four logic AND gates with conditioning events. Based on the PNFO mentioned above, the index order of nodes is set to $P < S_1 < S_2 < rep(\theta_{S_1}, \theta_{S_1}) < rep(\theta_{S_2}, \theta_{S_2}) < rep(\theta_{S_1}, \theta_P) < rep(\theta_{S_2}, \theta_{S_1}) < rep(\theta_{S_2}, \theta_P)$. The CBDD of the CFT is generated as follows:

The CBDD of No. 1 gate in Fig. 6 is built as shown in Fig. 9. Similarly, CBDDs of No. 2 to No. 4 gates are constructed. These are basic logical operations without inconsistency. Fig. 10 shows the sub_CBDD generated by logical operation between CBDDs of No. 1 gate and No. 2 gate.

The same operation for sub_BDD generated via No. 3 gate CBDD merging with No. 4 gate CBDD is shown in Fig. 11.

The CBDD of the CFT is generated by a logic OR operation between two sub_CBDDs shown in Figs. 10 and 11. However, when the structure of the CBDD is complicated, the inconsistencies problem will emerge. For example, there is an inconsistency elimination operation (Fig. 7(b)) during logical operation between two sub_CBDDs, as shown in Fig. 12. The red dotted circle indicates the removed inconsistent node by replacing it to terminal node '0'.

Finally, the CBDD of the CFT after inconsistency elimination operations (Fig. 7(b)) is shown in Fig. 13.

Typically, inputs of spare gates are basic events (Dugan et al., 1992). However, if a basic event is both a primary event and a spare event, for example, a spare gate, the replacement becomes unclear (Boudali et al., 2010). Our proposed solution also can analyze this case if the definition of *rep* makes a corresponding extension. An example of DFT with specific spare gates is shown in Fig. 14 (Volk et al., 2018). To handle this DFT, the $\theta_1$ and $\theta_2$ in $rep(\theta_1, \theta_2)$ are allowed to denote a subtree, respectively.

WSP gates in Fig. 14 are converted to logic AND gates with the specific conditioning event. Once either $\theta_A$ or $\theta_B$ fails, the logic OR gate connected as the primary component will happen. Hence, we consider $\theta_A$ or $\theta_B$ as a primary component respectively during the conversion of CFT. The CFT of Fig. 14 is shown in Fig. A.1 (in the Appendix). We can build the CBDD based on the CFT shown in Fig. A.1. The steps are similar to the CBDD construction of Fig. 6. The final CBDD of the WSP gate with subtrees inputs is shown

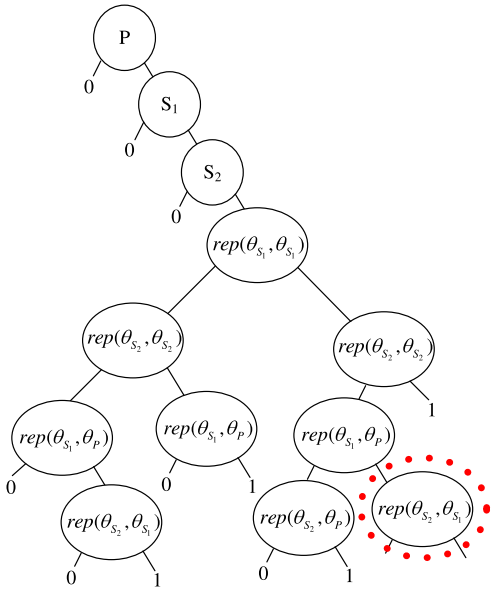



**Fig. 12.** CBDD of the CFT before elimination inconsistency and simplification.

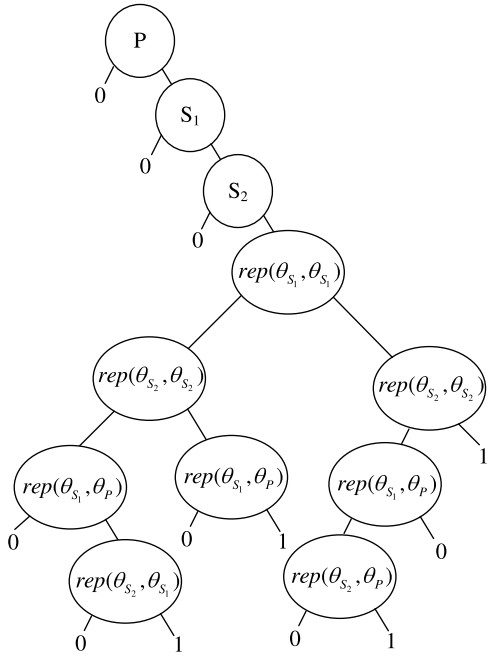



**Fig. 13.** Final CBDD of the CFT.

as Fig. 15 based on the index order selected by PNFO. The index order is set to $C < D < E < A < B < rep(\theta_D, \theta_C) < rep(\theta_D, \theta_D) < rep(T, T) < rep(T, \theta_A) < rep(T, \theta_B)$. The evaluation based on the CBDD is presented in Appendix A.

Note that our extension cannot be used for SP gates with arbitrary subtrees. For example, if both primary components and spare components are dependent events in an FDEP gate, then once the trigger is activated, spare races (more than two primary components simultaneously compete for the same spare component) may cause the non-deterministic behavior (Junges et al., 2016). Moreover, some complex nested SP gates in (Junges et al., 2016) also need carefully identifying replacement behaviors regarded to activation.

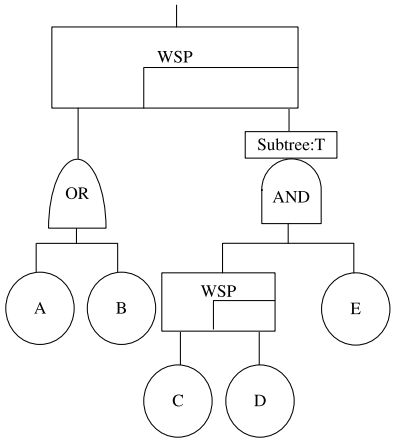



**Fig. 14.** WSP gate with subtrees inputs (Volk et al., 2018).

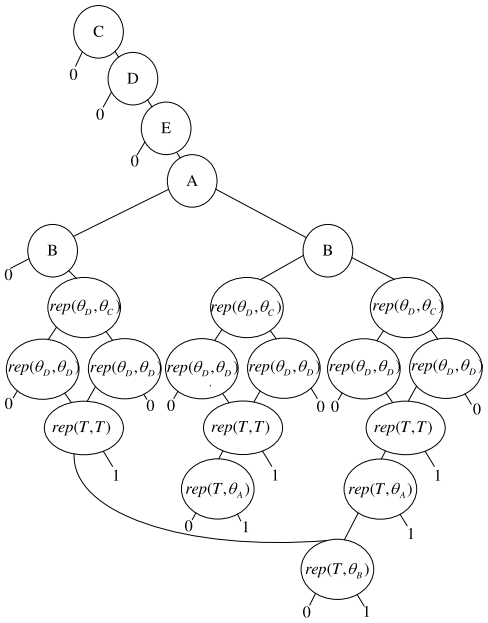



**Fig. 15.** CBDD of WSP gate with subtrees inputs.

## 6. Reliability analysis using conditional binary decision diagram

A general three-step process can design the CBDD based on dynamic fault trees with spare gates: CFT conversion, system CBDD model generation, and system CBDD model evaluation, which are described in the following:

**Step 1 — CFT Conversion**

The replacement behavior of the spare gate in the DFT is extracted, and the replacement behavior is transformed into the conditioning event after analysis. Then, the spare gate is converted into one or more logic AND gates with conditional events. Also, the occupation of shared spare components is considered (mentioned in Section 4.2). If some condition is satisfied, it may simplify conditioning AND gates to a logic AND gate. According to the description in Section 4.2, we assume that $m$ SP gates (each SP gate only has one primary component) share $n$ spare components, and $m$ SP gates are connected by a logical OR gate where $m > n$, then any combination of $n+1$ primary components failure (in total $C_m^{n+1}$) will cause the logical OR gate to occur. Consequently, we can use logical AND gates instead of partial

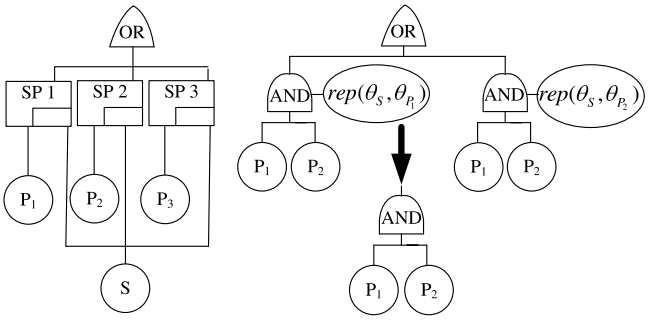



**Fig. 16.** A simplification for conditioning AND gates to a logical AND gate.

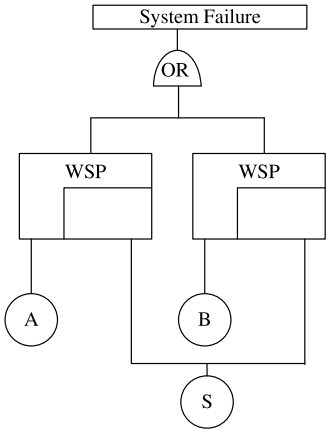



**Fig. 17.** DFT for the hard disk system (Tannous et al., 2011).

conditioning AND gates in this scenario. For example, three SP gates ($\theta_{P_1}$, $\theta_{P_2}$, and $\theta_{P_3}$) connected by a logical OR gate share one spare component ($\theta_S$), as shown in Fig. 16. A logical AND gates with two primary components (any combination of two primary components among three) can replace two conditioning AND gates, as shown in Fig. 16. Similarly, there two other logical AND gates ($P_1P_3$ and $P_2P_3$) that can be obtained, in total there are three ($C_3^2 = 3$). After that, the DFT can be converted into the CFT.

**Step 2 — System CBDD Model Generation**

Construct the CBDD based on the CFT using the process mentioned in Section 5. First, build sub-CBDDs related to separate logic gates. Then, new sub-CBDDs are generated by combining two sub-CBDDs via logical operations that correspond to the static logical gate in the CFT. Loop this process until the final CBDD is presented. In these operations, pay attention to avoiding contradiction and redundancy by using elimination redundancy and inconsistency rules mentioned in Figs. 7 and 8.

**Step 3 — System CBDD Model Evaluation**

Collect all paths from the top node to terminal node 1 in the CBDD. After eliminating negation events, inconsistency, and redundancy, all the paths are converted to corresponding algebraic-structure-based expressions (Merle, 2010). Finally, the results are obtained by using an integral-based method.

## 7. Case studies

In this section, three systems are illustrated to use and verify the proposed solution in detail.

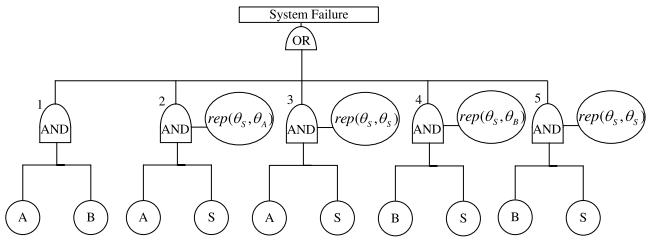



**Fig. 18.** CFT for the hard disk system.

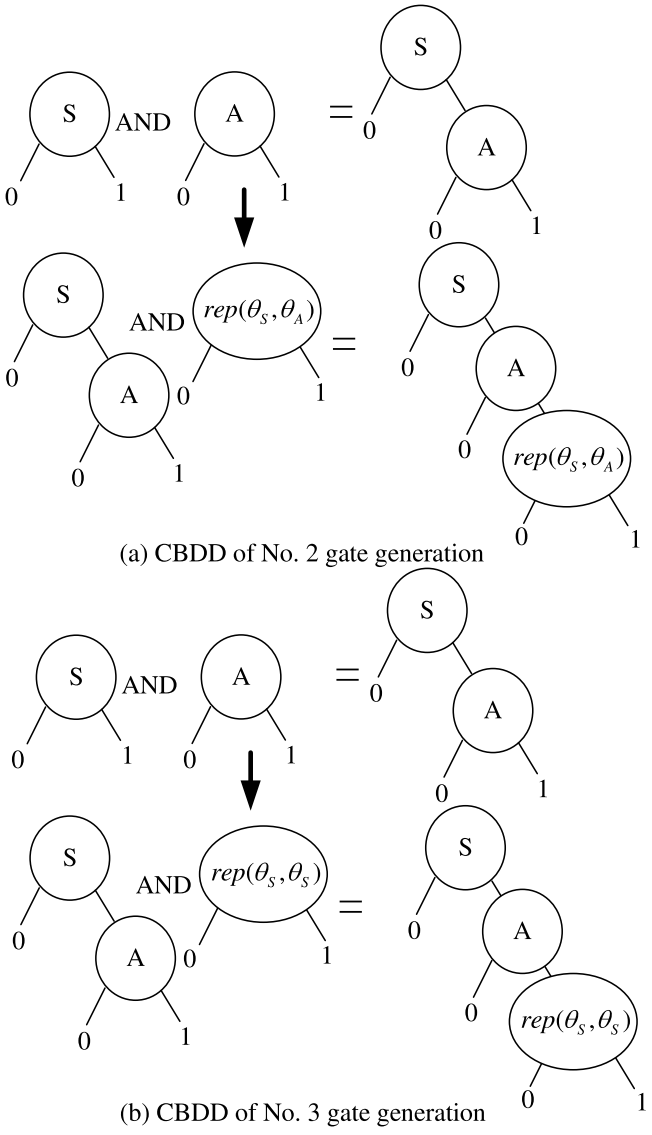



**Fig. 19.** Two conditional AND gate CBDDs generation.

### 7.1. Case study I

A hard disk system (Tannous et al., 2011) is shown in Fig. 17. The $\theta_A$ and $\theta_B$ are primary hard disks (primary components) sharing the same warm spare disk (spare component) $\theta_S$ in the hard disk system.

**Step 1 — CFT Conversion**

In this system, any WSP gate failure can lead to an entire system failure. The shared spare component failure and occupation (replaces the other primary component) are able to cause WSP

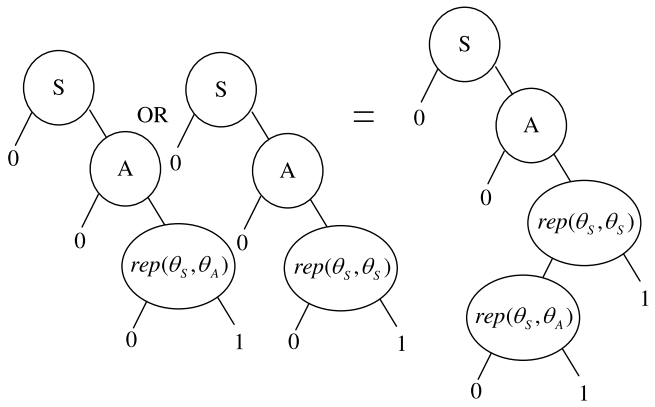



**Fig. 20.** Sub-CBDD of the hard disk system generation.

gate failure when the primary component fails. There are three conditioning events: $rep(\theta_S, \theta_A)$, $rep(\theta_S, \theta_B)$, and $rep(\theta_S, \theta_S)$. The DFT of the hard disk system needs to be converted into the CFT, as shown in Fig. 18. As mentioned in Section 4.2, two primary components share one spare components ($2 > 1$), and one of two WSP gates will occur if both of the two primary components fail. Because $\theta_S$ can only replace one primary component, when $\theta_A$ and $\theta_B$ both fail, one of two WSP gates is bound to fire, and the WSP gate firing will cause the entire system to fail as well. Hence, the No. 1 gate in Fig. 18 is constructed as a logic AND gate. Due to the logic OR gate in the top, when any one of the WSP gates fires individually, the entire system will also fail. Each WSP gate can fire in two conditions. One condition is when the primary component fails and then the spare component replaces the primary component and finally fails, which corresponds to No. 2 gate and No. 4 gate. The other condition is when the spare component fails first and then the primary component fails, which corresponds to No. 3 gate and No. 5 gate. No. 2 gate and No. 4 gate include conditioning events related to replacement. No. 3 gate and No. 5 gate include conditioning events related to non-replacement. Any gate firing can result in an entire system failure.

**Step 2 — System CBDD Model Generation**

According to PNFO, the index order of variables corresponding to nodes of CBDD is $S < A < B < rep(\theta_S, \theta_S) < rep(\theta_S, \theta_A) < rep(\theta_S, \theta_B)$, and all components start to work (or are in the dormant state) simultaneously. The CFT for the hard disk system includes one logic AND gate and four conditional AND gates. In the system CFT model, the occurrence of any one of the conditional AND gates or the logic AND gate can result in the top event happening.

Firstly, construct CBDDs of five gates numbered 1 to 5 as shown in Fig. 18. The No. 1 gate is a logic AND gate. The No. 2 gate and No. 3 CBDD model generation are shown as Fig. 19. Similarly, No. 4 gate and No. 5 gate are built as well.

Secondly, merge the CBDDs of five gates.

(i) Perform a logic OR operation between two CBDDs related to No. 2 gate and No. 3 gate as a sub-CBDD of the hard disk system, as shown in Fig. 20.

(ii) The same operation is used for CBDDS of No. 4 gate and No. 5 gate, and the result is the other sub-CBDD of the hard disk system.

(iii) The sub-CBDD generated in (i) does a logic OR operation with the CBDD of No. 1 gate, and it can generate the third sub-CBDD of the system.

(iv) The final CBDD can be obtained by a logic OR operation between the sub-CBDD generated in (ii) and the sub-CBDD generated in (iii), as shown in Fig. 21.

Five MCSs can be found by the system CBDD as well as from the CFT directly:

(1) $A \cdot B$ (absorbs $S \cdot A \cdot B$ )

(2) $S \cdot rep(\theta_S, \theta_B)$

(3) $S \cdot B \cdot rep(\theta_S, \theta_S)$

(4) $S \cdot rep(\theta_S, \theta_A)$

(5) $S \cdot A \cdot rep(\theta_S, \theta_S)$

However, they need to be converted to the sequence-dependent model of the algebraic structure to compute failure probability via the I/E-based method. Obviously, it is more complicated than the SDP-based method.

**Step 3 — System CBDD Model Evaluation**

The CBDD can implicitly represent the SDP. Each disjoint product (DP) corresponds to a path from the root node to terminal node '1' like that in the BDD. As a result of all the paths being disjointed, we can compute the entire system probability via the sum of probabilities of all the paths from the root to the terminal node '1'. To evaluate the system CBDD model, probabilities of all the paths from the root node to the terminal node '1' must be calculated. There are six paths shown as the logic operation result in Fig. 21, which can lead to an entire system failure. The symbol $\Rrightarrow$ denotes a path between two CBDD nodes.

(1) $\neg S \Rrightarrow A \Rrightarrow B$

(2) $S \Rrightarrow \neg A \Rrightarrow B \Rrightarrow rep(\theta_S, \theta_S)$

(3) $S \Rrightarrow \neg A \Rrightarrow B \Rrightarrow \neg rep(\theta_S, \theta_S) \Rrightarrow rep(\theta_S, \theta_B)$

(4) $S \Rrightarrow A \Rrightarrow \neg B \Rrightarrow rep(\theta_S, \theta_S)$

(5) $S \Rrightarrow A \Rrightarrow \neg B \Rrightarrow \neg rep(\theta_S, \theta_S) \Rrightarrow rep(\theta_S, \theta_A)$

(6) $S \Rrightarrow A \Rrightarrow B$

Two paths (1) and (6) in Eq. (13) can be considered as a path $A \Rrightarrow B$. The system failure probability can be expressed as:

$$\begin{aligned} U_{system} = & Pr\{S \cdot \neg A \cdot B \cdot rep(\theta_S, \theta_S) \\ & + S \cdot \neg A \cdot B \cdot \neg rep(\theta_S, \theta_S) \cdot rep(\theta_S, \theta_B) \\ & + S \cdot A \cdot \neg B \cdot rep(\theta_S, \theta_S) \\ & + S \cdot A \cdot \neg B \cdot \neg rep(\theta_S, \theta_S) \cdot rep(\theta_S, \theta_A) \\ & + A \cdot B\} \end{aligned} \tag{13}$$

From Eq. (13), all the disjunctive terms are mutually exclusive. Actually, they are DPs. According to the inclusion–exclusion principle, Eq. (13) can be deduced as follows:

$$\begin{aligned} U_{system} = & Pr\{S \cdot \neg A \cdot B \cdot rep(\theta_S, \theta_S)\} \\ & + Pr\{S \cdot \neg A \cdot B \cdot \neg rep(\theta_S, \theta_S) \cdot rep(\theta_S, \theta_B)\} \\ & + Pr\{S \cdot A \cdot \neg B \cdot rep(\theta_S, \theta_S)\} \\ & + Pr\{S \cdot A \cdot \neg B \cdot \neg rep(\theta_S, \theta_S) \cdot rep(\theta_S, \theta_A)\} \\ & + Pr\{A \cdot B\} \end{aligned} \tag{14}$$

In this case, some negation conditioning events can be replaced as follows:

$$\neg rep(\theta_S, \theta_A) = \neg A + A \cdot rep(\theta_S, \theta_S) + A \cdot rep(\theta_S, \theta_B) \tag{15}$$

$$\neg rep(\theta_S, \theta_B) = \neg B + B \cdot rep(\theta_S, \theta_S) + B \cdot rep(\theta_S, \theta_A) \tag{16}$$

$$\neg rep(\theta_S, \theta_S) = rep(\theta_S, \theta_A) + rep(\theta_S, \theta_B) \tag{17}$$

According to Eq. (15) to Eq. (17) and the elimination redundancy and inconsistency rules (in Section 4.3), Eq. (14) can be reduced as follows:

$$\begin{aligned} U_{system} = & Pr\{A \cdot B\} \\ & + Pr\{S \cdot A \cdot \neg B \cdot rep(\theta_S, \theta_S)\} \\ & + Pr\{S \cdot \neg B \cdot rep(\theta_S, \theta_A)\} \\ & + Pr\{S \cdot \neg A \cdot B \cdot rep(\theta_S, \theta_S)\} \end{aligned}$$

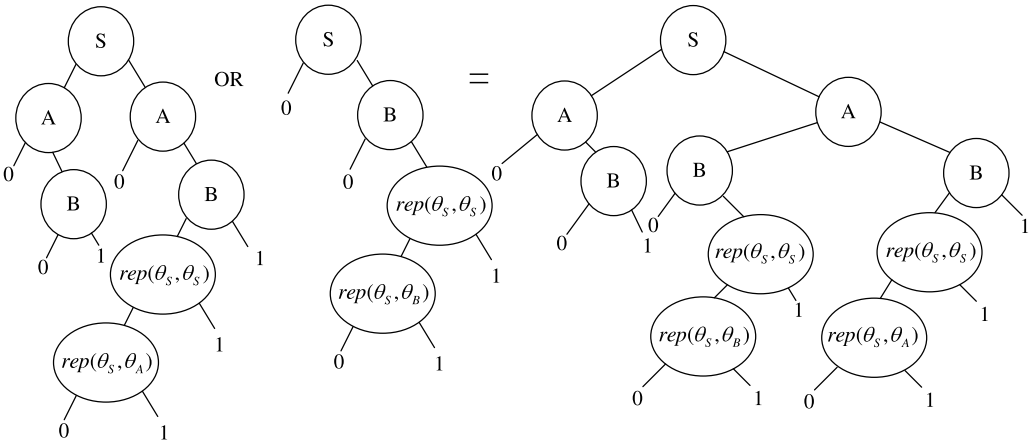



**Fig. 21.** CBDD of the hard disk system generation.

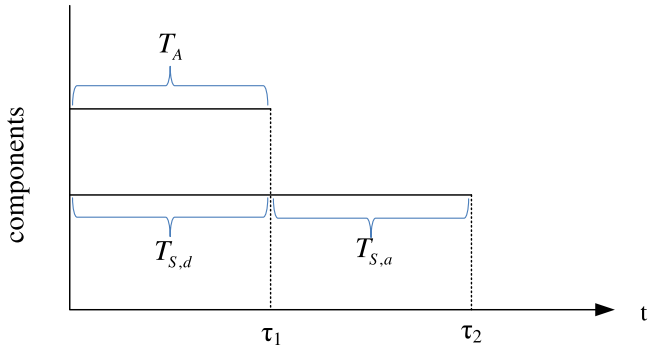



**Fig. 22.** Time relationship for $T_A$, $T_{S,d}$, and $T_{S,a}$.

$$+ Pr\{S \cdot \neg A \cdot rep(\theta_S, \theta_B)\} \tag{18}$$

$Pr\{A \cdot B\} = Pr\{A\} \cdot Pr\{B\}$ can be easily deduced.

$Pr\{S \cdot A \cdot \neg B \cdot rep(\theta_S, \theta_S)\} = (1 - Pr\{B\}) \cdot Pr\{S \cdot A \cdot rep(\theta_S, \theta_S)\}$. $Pr\{S \cdot A \cdot rep(\theta_S, \theta_S)\}$ is the failure probability of the WSP gate consisting of the primary component $\theta_A$ and the spare component $\theta_S$ in the condition of non-replacement. Non-replacement implies $\theta_S$ failed in the dormant state. $S \cdot A \cdot \neg B \cdot rep(\theta_S, \theta_S)$ represents $\theta_A$ fails and $\theta_B$ does not fail in the case that $\theta_S$ fails in a dormant state. It implies the spare component $\theta_S$ fails first ($\theta_S$ fails before $\theta_A$ and $\theta_B$), $\theta_A$ fails after $\theta_S$, and $\theta_B$ does not fail. Thus, $S \cdot A \cdot rep(\theta_S, \theta_S)$ can be converted into a sequence-dependent model of algebraic structure function as $(S_d \triangleleft A) \cdot (S_d \triangleleft B) \cdot A$. Then, $S \cdot A \cdot \neg B \cdot rep(\theta_S, \theta_S)$ can be converted into $\neg B \cdot (S_d \triangleleft A) \cdot (S_d \triangleleft B) \cdot A$. According to the sequence-dependent model of algebraic determination of the structure-function deduction rules (Merle, 2010), it can be deduced as follows:

$$\begin{aligned}
&\neg B \cdot (S_d \triangleleft A) \cdot (S_d \triangleleft B) \cdot A \\
&= \neg B \cdot (S_d \triangleleft A) \cdot ((S_d \triangleleft B) \cdot B + S_d \cdot \neg B) \cdot A \\
&= \neg B \cdot (S_d \triangleleft A) \cdot S_d \cdot \neg B \cdot A \\
&= \neg B \cdot (S_d \triangleleft A) \cdot A
\end{aligned} \tag{19}$$

Here, $T_A$ is set as a random variable ($r.v$) that represents the time-to-failure of the primary component $\theta_A$. Similarly, $T_B$ refers to the primary component $\theta_B$. $T_{Sd}$ and $T_{Sa}$ are set as the time to failure $r.v$ of warm spare component $\theta_S$ before and after it is activated, respectively. The relationship between $T_A$, $T_{S,d}$, and $T_{S,a}$ is shown in Fig. 22. For ease of computation, it is assumed that all components start to work at $t = 0$. Let $f_A(t)$, $f_B(t)$, $f_{S,d}(t)$, and $f_{S,a}(t)$ denote the probability density function of $T_A$, $T_B$, $T_{S,d}$, and $T_{S,a}$ respectively. The calculation similar to $(S_d \triangleleft A) \cdot A$ has been discussed many times (Merle et al., 2011a; Merle, 2010). Assuming that $\theta_S$ fails at $\tau_1$, then the probability that $\theta_A$ fails after $\theta_S$ ($\theta_A$ fails after $\tau_1$) is:

$$Pr\{T_A > \tau_1\} = \int_{\tau_1}^{t} f_A(\tau_2) d\tau_2 \tag{20}$$

Hence, the probability that $\theta_S$ fails before $\theta_A$ and $\theta_A$ fails is given as follows:

$$Pr\{(S_d \triangleleft A) \cdot A\} = \int_0^t \int_{\tau_1}^t f_A(\tau_2) f_{S,d}(\tau_1) d\tau_2 d\tau_1 \tag{21}$$

Consequently, the probability of $S \cdot A \cdot \neg B \cdot rep(\theta_S, \theta_S)$ is:

$$\begin{aligned}
&Pr\{S \cdot A \cdot \neg B \cdot rep(\theta_S, \theta_S)\} \\
&= Pr\{\neg B\} \cdot Pr\{(S_d \triangleleft A) \cdot A\} \\
&= (1 - Pr\{B\}) \cdot Pr\{(S_d \triangleleft A) \cdot A\} \\
&= \left(1 - \int_0^t f_B(\tau_3) d\tau_3\right) \cdot \int_0^t \int_{\tau_1}^t f_A(\tau_2) f_{S,d}(\tau_1) d\tau_2 d\tau_1
\end{aligned} \tag{22}$$

$Pr\{S \cdot \neg B \cdot rep(\theta_S, \theta_A)\} = Pr\{\neg B\} \cdot Pr\{S \cdot rep(\theta_S, \theta_A)\}$. $Pr\{S \cdot rep(\theta_S, \theta_A)\}$ is the failure probability of the WSP gate consisting of the primary component $\theta_A$ and the spare component $\theta_S$ in the condition that replacement occurs. Replacement indicates $\theta_S$ fails in the activated state. It implies that $\theta_A$ fails first (before $\theta_B$ and $\theta_S$ fail), and $\theta_S$ fails after the failure of component $\theta_A$. Thus, $S \cdot \neg B \cdot rep(\theta_S, \theta_A)$ can be converted into a sequence-dependent model of algebraic structure function as $\neg B \cdot (A \triangleleft B) \cdot (A \triangleleft S_a) \cdot S_a$. It can also be deduced as follows:

$$\begin{aligned}
&\neg B \cdot (A \triangleleft B) \cdot (A \triangleleft S_a) \cdot S_a \\
&= \neg B \cdot ((A \triangleleft B) \cdot B + A \cdot \neg B) \cdot (A \triangleleft S_a) \cdot S_a \\
&= \neg B \cdot A \cdot \neg B \cdot (A \triangleleft S_a) \cdot S_a \\
&= \neg B \cdot (A \triangleleft S_a) \cdot S_a
\end{aligned} \tag{23}$$

First, the probability that the primary component $\theta_A$ fails at $\tau_2$ and the spare component $\theta_S$ fails after $\theta_A$, meaning that $\theta_S$ survives $\tau_2$ (the probability of $\theta_S$ non-occurrence in a dormant state from time 0 to time $\tau_2$):

$$\begin{aligned}
&Pr\{S\ survives\ \tau_2\} = 1 - Pr\{S\ fails\ before\ \tau_2\} \\
&= 1 - \int_0^{\tau_2} f_{S,d}(\tau_1) d\tau_1
\end{aligned} \tag{24}$$

$$\begin{aligned}
&Pr\{S\ fails\ after\ \tau_2 | S\ survives\ \tau_2\} \\
&= \int_{\tau_2}^t f_{S,a}(t_1) dt_1
\end{aligned}$$

$$= \int_0^{t-\tau_2} f_{S,a}(t_1-\tau_2)d(t_1-\tau_2) = \int_0^{t-\tau_2} f_{S,a}(\tau_1)d\tau_1 \tag{25}$$

Secondly, the probability that $\theta_A$ fails at $\tau_2$ and then $\theta_S$ fails after $\theta_A$ equals the probability that $\theta_A$ fails, $\theta_S$ survives $\tau_2$ and $\theta_S$ fails after $\tau_2$:

$$Pr\{(A \lhd S_a)\cdot S_a\} = \int_0^t \int_0^{t-\tau_2} f_{S,a}(\tau_1) f_A(\tau_2)\left(1-\int_0^{\tau_2} f_{S,d}(\tau_1)d\tau_1\right)d\tau_1 d\tau_2 \tag{26}$$

Finally, the $Pr\{S\cdot\neg B\cdot rep(\theta_S,\theta_A)\}$ is:

$$\begin{aligned} &Pr\{S\cdot\neg B\cdot rep(\theta_S,\theta_A)\} \\ &= Pr\{\neg B\cdot(A\lhd S_a)\cdot S_a\} \\ &= \left(1-\int_0^t f_B(\tau_3)d\tau_3\right) \\ &\cdot \int_0^t\int_0^{t-\tau_2} f_{S,a}(\tau_1)f_A(\tau_2)\left(1-\int_0^{\tau_2} f_{S,d}(\tau_1)d\tau_1\right)d\tau_1 d\tau_2 \end{aligned} \tag{27}$$

The structure of $S\cdot A\cdot\neg B\cdot rep(\theta_S,\theta_S)$ and $S\cdot\neg B\cdot rep(\theta_S,\theta_A)$ are symmetrical to $S\cdot\neg A\cdot B\cdot rep(\theta_S,\theta_S)$ and $S\cdot\neg A\cdot rep(\theta_S,\theta_B)$, respectively. A system failure can be caused by the failure of the WSP gate, which consists of the primary component $\theta_B$ and the spare component $\theta_S$. Hence, they can be analyzed in a similar way with the following results:

$$\begin{aligned} &Pr\{\neg A\cdot B\cdot rep(\theta_S,\theta_S)\cdot S\} = Pr\{\neg A\cdot(S_d\lhd B)\cdot B\} \\ &= Pr\{\neg A\}\cdot Pr\{(S_d\lhd B)\cdot B\} \\ &= (1-Pr\{A\})\cdot Pr\{(S_d\lhd B)\cdot B\} \\ &= \left(1-\int_0^t f_A(\tau_2)d\tau_2\right)\cdot\int_0^t\int_{\tau_1}^t f_B(\tau_3)f_{S,d}(\tau_1)d\tau_3 d\tau_1 \end{aligned} \tag{28}$$

$$\begin{aligned} &Pr\{\neg A\cdot rep(\theta_S,\theta_B)\cdot S\} \\ &= Pr\{\neg A\cdot(B\lhd S_a)\cdot S_a\} \\ &= \left(1-\int_0^t f_A(\tau_2)d\tau_2\right) \\ &\cdot\int_0^t\int_0^{t-\tau_3} f_{S,a}(\tau_1)f_B(\tau_3)\left(1-\int_0^{\tau_3} f_{S,d}(\tau_1)d\tau_1\right)d\tau_1 d\tau_3 \end{aligned} \tag{29}$$

According to Eq. (18), Eq. (22), Eq. (27), Eqs. (28) and (29), the unreliability of the system is:

$$\begin{aligned} U_{system} = &\left(\int_0^t f_A(\tau_2)d\tau_2\right)\cdot\left(\int_0^t f_B(\tau_3)d\tau_3\right) \\ &+\left(1-\int_0^t f_B(\tau_3)d\tau_3\right)\cdot\int_0^t\int_{\tau_1}^t f_A(\tau_2)f_{S,d}(\tau_1)d\tau_2 d\tau_1 \\ &+\left(1-\int_0^t f_B(\tau_3)d\tau_3\right) \\ &\cdot\int_0^t\int_0^{t-\tau_2} f_{S,a}(\tau_1)f_A(\tau_2)\left(1-\int_0^{\tau_2} f_{S,d}(\tau_1)d\tau_1\right)d\tau_1 d\tau_2 \\ &+\left(1-\int_0^t f_A(\tau_2)d\tau_2\right)\cdot\int_0^t\int_{\tau_1}^t f_B(\tau_3)f_{S,d}(\tau_1)d\tau_3 d\tau_1 \\ &+\left(1-\int_0^t f_A(\tau_2)d\tau_2\right) \\ &\cdot\int_0^t\int_0^{t-\tau_3} f_{S,a}(\tau_1)f_B(\tau_3)\left(1-\int_0^{\tau_3} f_{S,d}(\tau_1)d\tau_1\right)d\tau_1 d\tau_3 \end{aligned} \tag{30}$$

To verify the results, the Markov-based method, SBDD, and ABDD are used to analyze the same system. Assume that $\theta_A$, $\theta_B$, and $\theta_S$ fail exponentially with constant rates. Let $f_A(\tau_2) = a_A e^{-a_A\tau_2}$, $f_B(\tau_3) = a_B e^{-a_B\tau_3}$, $f_{S,a}(\tau_1) = a_S e^{-a_S\tau_1}$ and $f_{S,d}(\tau_1) = d_S e^{-d_S\tau_1}$. Use the parameter values of $a_A = 0.002/day$, $a_B = 0.0015/day$, $a_S = 0.0025/day$ (replacement), and $d_S = 0.001/day$ (non-replacement). The Markov model generated in the Markov solution is shown in Fig. 23. As a result, there are five different differential equations, as shown in Eq. . According to the assumption mentioned above, the initial state (at time 0) is state 1.

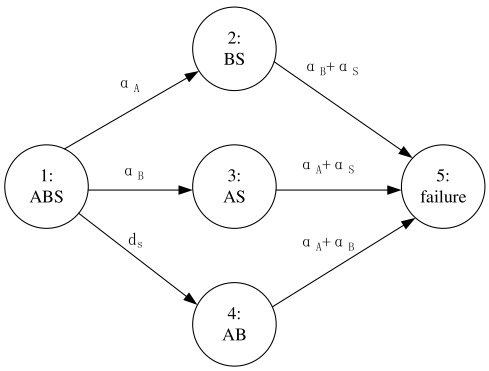



**Fig. 23.** Markov model of the hard disk system.

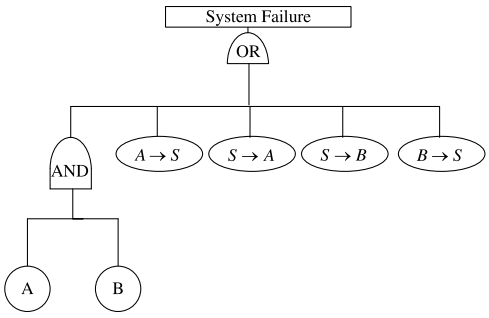



**Fig. 24.** Fault tree after conversion by SBDD (Tannous et al., 2011).

$$\begin{aligned} &P_1(0)=1 \ \ and \ \ P_i(0)=0,\ i\neq 1 \\ &\frac{d}{dt}P_1(t) = -(a_A+a_B+d_s)P_1(t) \\ &\frac{d}{dt}P_2(t) = a_A P_1(t)-(a_B+a_S)P_2(t) \\ &\frac{d}{dt}P_3(t) = a_B P_1(t)-(a_A+a_S)P_3(t) \\ &\frac{d}{dt}P_4(t) = d_S P_1(t)-(a_A+a_B)P_4(t) \\ &\frac{d}{dt}P_5(t) = (a_B+a_S)P_2(t)+(a_A+a_S)P_3(t) \\ &\qquad +(a_A+a_B)P_4(t) \end{aligned} \tag{31}$$

The converted trees are shown as Figs. 24 and 25 and they correspond to SBDD and ABDD, which are shown as Fig. 26 and Fig. 27, respectively. In Figs. 24 and 26, the symbol "→" denotes the precedence order of component failures, for example, "$S \rightarrow A$" means that a basic event $S$ fails before a basic event $A$ fails (Tannous et al., 2011). The SBDD and ABDD (only compute 7 valid paths ) use the SDPs-based method to calculate the system unreliability (Tannous et al., 2011; Jiang et al., 2018).

We compute the system unreliability results by using the proposed solution, the Markov solution, the SBDD solution, and the ABDD solution for three different mission times (t=300, 500, and 900 days, respectively). All equations (integrals and differentials) are programmed via Matlab 2016 and get an exact match, as shown in Table 1.

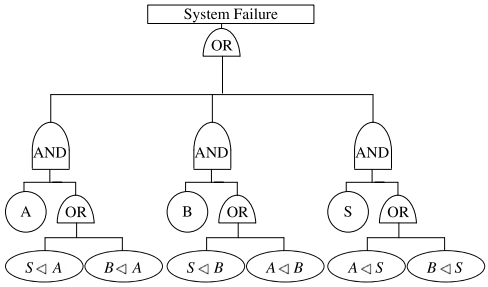



**Fig. 25.** Fault tree after conversion by ABDD (Jiang et al., 2018).

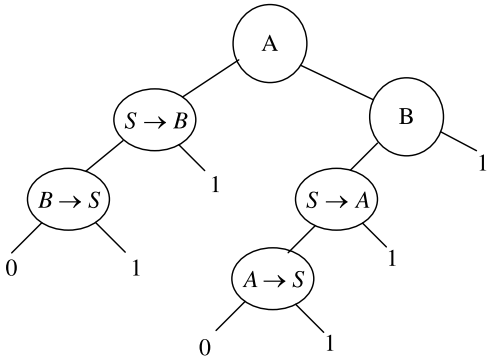



**Fig. 26.** Final SBDD of the hard disk system (Tannous et al., 2011).

**Table 1**
Results for the hard disk system analysis.

| Method | t (days) | | |
|---|---|---|---|
| | 300 | 500 | 900 |
| Markov | 0.365588 | 0.627432 | 0.894022 |
| SBDD (Exponential) | 0.365588 | 0.627432 | 0.894022 |
| ABDD (Exponential) | 0.365588 | 0.627432 | 0.894022 |
| CBDD (Exponential) | 0.365588 | 0.627432 | 0.894022 |
| CBDD (Weibull) | 0.137563 | 0.479438 | 0.942553 |

The time-to-failure distributions of Weibull also can be applied to the system components. The following parameter values are used for computation: Weibull (shape: $k = 1.8$, scale: $a_A = 0.002/day$, $a_B = 0.0015/day$, $a_S = 0.0025/day$, and $d_S = 0.001/day$, $f_A(\tau_2) = k(a_A^k)\tau_2^{k-1}e^{-(a_A\tau_2)^k}$). Results of the Weibull distribution used in CBDD with different mission times are shown in Table 1. It is obvious that our proposed method is not limited to exponential distribution. We can calculate the disk system unreliability when the system components follow other time-to-failure distributions, such as Weibull. Compared to SBDD and ABDD, CBDD avoids sequence-dependent (or temporal logic) nodes in the formal and eliminates inconsistencies in the process of building the CBDD. As opposed to SBDD and ABDD, all nodes of CBDD can be located in the basic event level. Moreover, our approach is more efficient than using the I/E-based method (Dugan and Doyle, 1996; Liu et al., 2007) on MCSs/MCQs directly since it only computes an expression of 5 integrals rather than an expression of 31 ($2^5 - 1$, 5 MCSs) terms (each term is also an integral) including our 5 integrals.

### 7.2. Case study II

The DFT with spare gates example that we are going to use is the DFT of a Vehicle Management System (VMS) (Vesely et al., 2002) in the application, as shown in Fig. 28. The system consists of three WSP gates, a logic AND gate, and a logic OR gate. Three

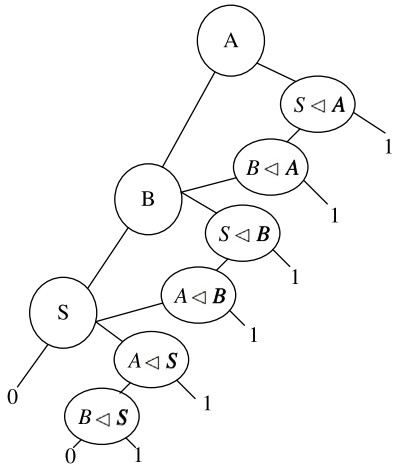



**Fig. 27.** The final ABDD of the hard disk system including invalid paths (Jiang et al., 2018).

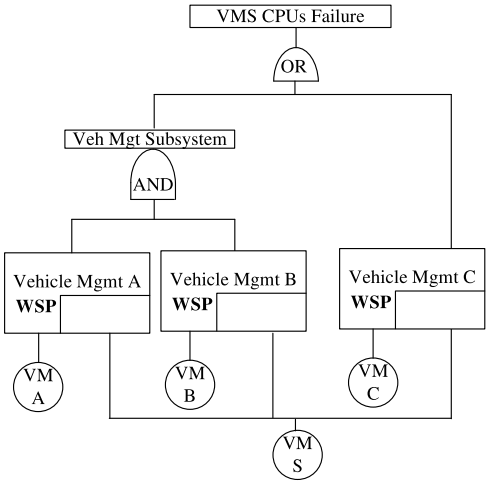



**Fig. 28.** An example DFT for VMS (Vesely et al., 2002).

WSP gates share the same spare component $\theta_S$ in the VMS. In the system, a vehicle management subsystem is a logic AND gate that is composed of two WSP gates (vehicle management A and vehicle management B). Finally, the system consists of the vehicle management subsystem and the WSP gate (vehicle management C) via logic OR gate.

**Step 1 — CFT Conversion**

When WSP gates are used, the assignment of the spare component as well as the determination of the failure rate for each spare component is included. The failure of vehicle management C can directly lead to an entire system failure, which can be caused by two cases. One case is both $\theta_C$ and $\theta_S$ (before or after replacement) fail, and the other case is when $\theta_S$ replaces either $\theta_A$ or $\theta_B$, then $\theta_C$ fails. The subsystem failure caused by the failure of two WSP gates can also result in the entire system failure. The combination between $A$, $B$, $S$, $rep(\theta_S, \theta_A)$, $rep(\theta_S, \theta_B)$, and $rep(\theta_S, \theta_S)$ can deduce three cut sets to lead to a subsystem failure. Another set that causes subsystem failure is the combination of $A$, $B$, $C$, and $rep(\theta_S, \theta_C)$. In the VMS, three WSP gates share a spare $\theta_S$. According to the analysis of converting from shared spare SP gates to CFTs in Section 4.2, one of three WSP gates will fail after two primary components fail. However, two of three primary components failing may not cause a system failure because two WSP gates are combined with a logic AND gate. For example, $C$

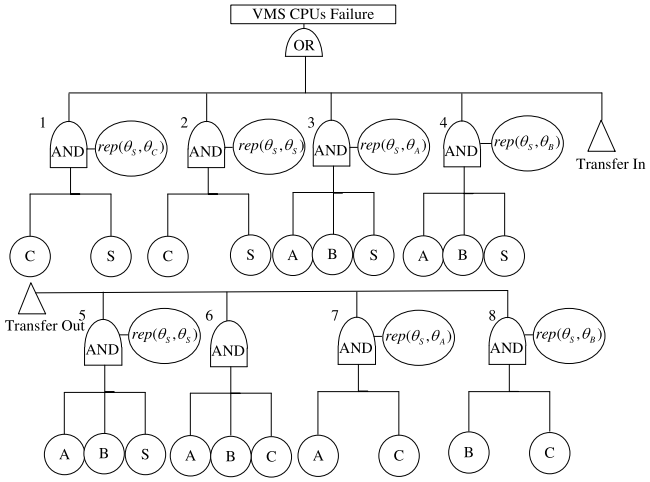



**Fig. 29.** The CFT for VMS.

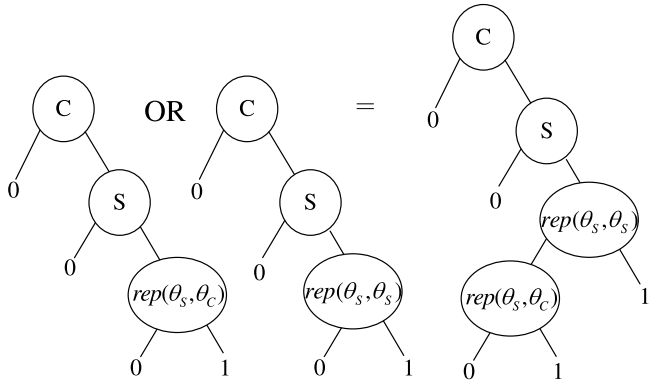
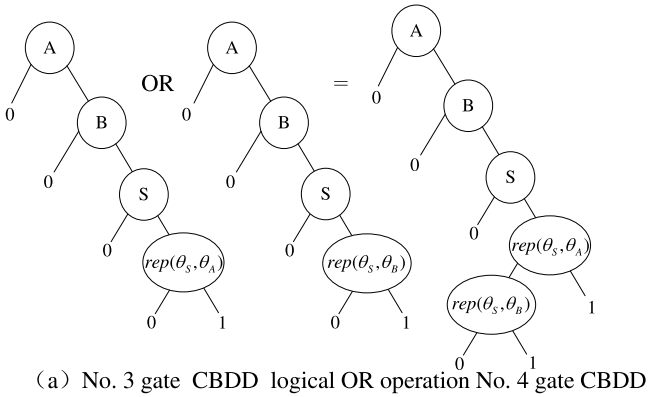



**Fig. 30.** CBDD of Vehicle Mgmt C generation.

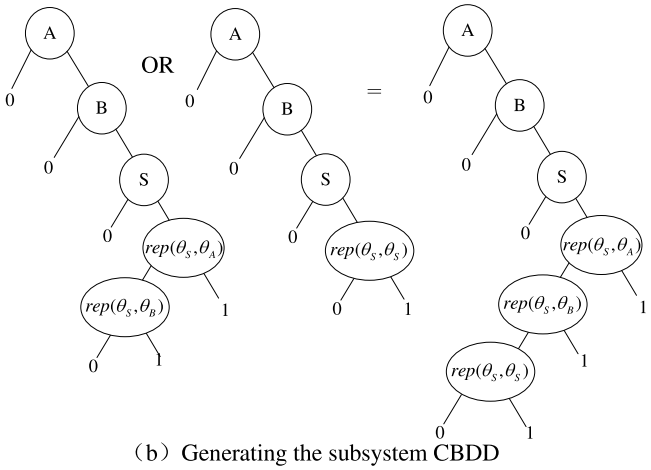
failed before *B* fails, *C* failed before *A* fails, and both *A* and *B* fail. If all three primary components fail, two of three WSP gates will fail since only one spare component is available. Any two WSP gates fail will cause the top event to happen. Thus, the combination of *A*, *B*, and *C* can directly lead to entire system failure. Hence, the combination of *A*, *B*, and *C* is chosen. According to the analysis above, the DFT can be converted into the CFT, as shown in Fig. 29. The CFT is made up of seven conditional AND gates and one logic AND gate. In this case study, it is not easy to directly obtain a logical AND gate with three primary components ($\theta_A$, $\theta_B$, and $\theta_C$) by the rule mentioned in Section 4.2. It is required to separately analyze the two WSP gates connected by a logical AND gate then consider the replacement between the spare component and three primary components to translate the logical AND gate.

### Step 2 — System CBDD Model Generation

Set the index order of nodes as $A < B < C < S < rep(\theta_S, \theta_A) < rep(\theta_S, \theta_B) < rep(\theta_S, \theta_S) < rep(\theta_S, \theta_C)$ based on the PNFO when two CBDDs are merged by logical operations. According to the VMS CFT, the system failure is caused by any one of the eight gates (seven conditional AND gates and one logic AND gate) that are numbered from 1 to 8 from left to right. All gates from 1 to 8 can be easily constructed as shown in Fig. 19. The primary events *C*, *S*, conditioning event $rep(\theta_S, \theta_C)$, and conditioning event $rep(\theta_S, \theta_S)$ can cause the No. 1 gate and No. 2 gate to fire (Vehicle Mgmt C), which will lead to VMS system failure. The related CBDD is built as shown in Fig. 30. The subsystem failure is caused by three primary events (*A*, *B*, and *S*) and three conditioning events ($rep(\theta_S, \theta_A)$, $rep(\theta_S, \theta_B)$ and $rep(\theta_S, \theta_S)$), and the subsystem CBDD




(a) No. 3 gate CBDD logical OR operation No. 4 gate CBDD




(b) Generating the subsystem CBDD

**Fig. 31.** The subsystem CBDD generation.

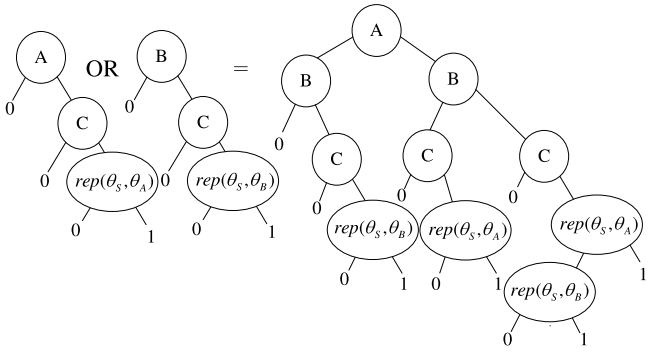



**Fig. 32.** Sub-CBDD of VMS generated by No. 7 gate CBDD logic OR operation No. 8 gate CBDD.

is built as shown in Fig. 31. Merge No. 7 gate CBDD with No. 8 gate CBDD by logic OR operation, as shown in Fig. 32. Similarly, do logic OR operation between these CBDDs including the CBDD of Vehicle Mgmt C, subsystem CBDD, No. 6 gate CBDD, and the CBDD in Fig. 32 via logic OR operation. During the operation, some inconsistencies can be solved by inconsistency elimination rules shown in Fig. 7. The Final CBDD of VMS is shown in Fig. 33.

### Step 3 — System CBDD Model Evaluation

Assume that all components start to work (or are in the dormant state) at the same time in the beginning. The fourteen paths that can lead to the VMS failure are from the root node to a terminal node '1' in the VMS CBDD, as shown in Fig. 33.

(1)$\neg A \Rrightarrow \neg B \Rrightarrow C \Rrightarrow S \Rrightarrow \neg rep(\theta_S, \theta_S) \Rrightarrow rep(\theta_S, \theta_C)$

(2)$\neg A \Rrightarrow \neg B \Rrightarrow C \Rrightarrow S \Rrightarrow rep(\theta_S, \theta_S)$

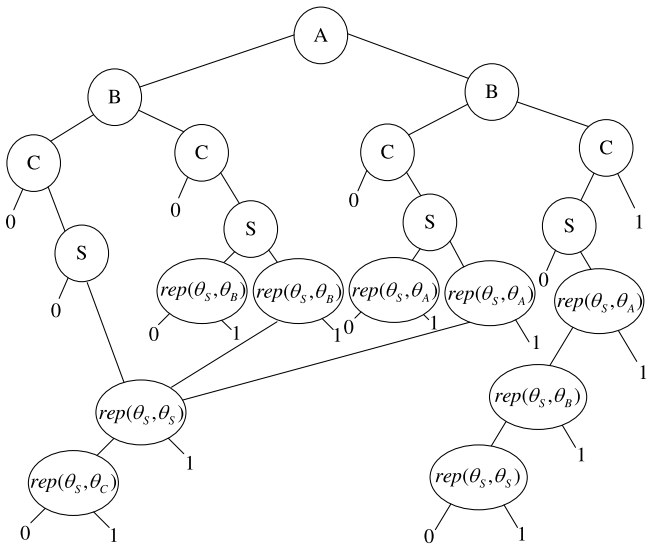



**Fig. 33.** CBDD of VMS.

(3)$\neg A \Rightarrow B \Rightarrow C \Rightarrow \neg S \Rightarrow rep(\theta_S, \theta_B)$
(4)$\neg A \Rightarrow B \Rightarrow C \Rightarrow S \Rightarrow rep(\theta_S, \theta_B)$
(5)$\neg A \Rightarrow B \Rightarrow C \Rightarrow S \Rightarrow \neg rep(\theta_S, \theta_B) \Rightarrow \neg rep(\theta_S, \theta_S) \Rightarrow rep(\theta_S, \theta_C)$
(6)$\neg A \Rightarrow B \Rightarrow C \Rightarrow S \Rightarrow \neg rep(\theta_S, \theta_B) \Rightarrow rep(\theta_S, \theta_S)$
(7)$A \Rightarrow \neg B \Rightarrow C \Rightarrow \neg S \Rightarrow rep(\theta_S, \theta_A)$
(8)$A \Rightarrow \neg B \Rightarrow C \Rightarrow S \Rightarrow rep(\theta_S, \theta_A)$
(9)$A \Rightarrow \neg B \Rightarrow C \Rightarrow S \Rightarrow \neg rep(\theta_S, \theta_A) \Rightarrow \neg rep(\theta_S, \theta_S) \Rightarrow rep(\theta_S, \theta_C)$
(10)$A \Rightarrow \neg B \Rightarrow C \Rightarrow S \Rightarrow \neg rep(\theta_S, \theta_A) \Rightarrow rep(\theta_S, \theta_S)$
(11)$A \Rightarrow B \Rightarrow \neg C \Rightarrow S \Rightarrow \neg rep(\theta_S, \theta_A) \Rightarrow \neg rep(\theta_S, \theta_B) \Rightarrow rep(\theta_S, \theta_S)$
(12)$A \Rightarrow B \Rightarrow \neg C \Rightarrow S \Rightarrow \neg rep(\theta_S, \theta_A) \Rightarrow rep(\theta_S, \theta_B)$
(13)$A \Rightarrow B \Rightarrow \neg C \Rightarrow S \Rightarrow rep(\theta_S, \theta_A)$
(14)$A \Rightarrow B \Rightarrow C$

Based on the binary logic of VMS CBDD, the mutually exclusive relation can be easily found in these paths. Hence, the probability of the VMS failure is shown as follows:

$$\begin{aligned}Pr\{VMS\} =& Pr\{\neg A \cdot \neg B \cdot C \cdot S \cdot \neg rep(\theta_S, \theta_S) \cdot rep(\theta_S, \theta_C)\}+\\ & Pr\{\neg A \cdot \neg B \cdot C \cdot S \cdot rep(\theta_S, \theta_S)\}+\\ & Pr\{\neg A \cdot B \cdot C \cdot \neg S \cdot rep(\theta_S, \theta_B)\}+\\ & Pr\{\neg A \cdot B \cdot C \cdot S \cdot rep(\theta_S, \theta_B)\}+\\ & Pr\{\neg A \cdot B \cdot C \cdot S \cdot \neg rep(\theta_S, \theta_B) \cdot \neg rep(\theta_S, \theta_S) \cdot rep(\theta_S, \theta_C)\}+\\ & Pr\{\neg A \cdot B \cdot C \cdot S \cdot \neg rep(\theta_S, \theta_B) \cdot rep(\theta_S, \theta_S)\}+\\ & Pr\{A \cdot \neg B \cdot C \cdot \neg S \cdot rep(\theta_S, \theta_A)\}+\\ & Pr\{A \cdot \neg B \cdot C \cdot S \cdot rep(\theta_S, \theta_A)\}+\\ & Pr\{A \cdot \neg B \cdot C \cdot S \cdot \neg rep(\theta_S, \theta_A) \cdot \neg rep(\theta_S, \theta_S) \cdot rep(\theta_S, \theta_C)\}+\\ & Pr\{A \cdot \neg B \cdot C \cdot S \cdot \neg rep(\theta_S, \theta_A) \cdot rep(\theta_S, \theta_S)\}+\\ & Pr\{A \cdot B \cdot \neg C \cdot S \cdot \neg rep(\theta_S, \theta_A) \cdot \neg rep(\theta_S, \theta_B) \cdot rep(\theta_S, \theta_S)\}+\\ & Pr\{A \cdot B \cdot \neg C \cdot S \cdot \neg rep(\theta_S, \theta_A) \cdot rep(\theta_S, \theta_B)\}+\\ & Pr\{A \cdot B \cdot \neg C \cdot S \cdot rep(\theta_S, \theta_A)\}+\\ & Pr\{A \cdot B \cdot C\}\end{aligned} \tag{32}$$

Consider the entire logical relation in the VMS. Negation conditioning events are converted into the following basic events:

$$\neg rep(\theta_S, \theta_A) = \neg A + A \cdot rep(\theta_S, \theta_B) + A \cdot rep(\theta_S, \theta_C)+ A \cdot rep(\theta_S, \theta_S) \tag{33}$$

$$\neg rep(\theta_S, \theta_B) = \neg B + B \cdot rep(\theta_S, \theta_A) + B \cdot rep(\theta_S, \theta_C)+ B \cdot rep(\theta_S, \theta_S) \tag{34}$$

$$\neg rep(\theta_S, \theta_C) = \neg C + C \cdot rep(\theta_S, \theta_A) + C \cdot rep(\theta_S, \theta_B)+ C \cdot rep(\theta_S, \theta_S) \tag{35}$$

$$\neg rep(\theta_S, \theta_S) = rep(\theta_S, \theta_A) + rep(\theta_S, \theta_B) + rep(\theta_S, \theta_C) \tag{36}$$

According to Eq. (33) to Eq. (36), replace negation conditioning events and remove inconsistencies. Furthermore, to reduce calculations, $\neg A \cdot B \cdot C \cdot \neg S \cdot rep(\theta_S, \theta_B)$ and $\neg A \cdot B \cdot C \cdot S \cdot rep(\theta_S, \theta_B)$ are merged into $\neg A \cdot C \cdot rep(\theta_S, \theta_B)$. The same operate from $\neg A \cdot \neg B \cdot C \cdot S \cdot rep(\theta_S, \theta_S)$ and $\neg A \cdot B \cdot C \cdot S \cdot \neg rep(\theta_S, \theta_B) \cdot rep(\theta_S, \theta_S)$ to $\neg A \cdot C \cdot S \cdot rep(\theta_S, \theta_S)$. Similarly, $\neg B \cdot C \cdot rep(\theta_S, \theta_A)$ and $\neg A \cdot S \cdot rep(\theta_S, \theta_C)$ can also be obtained. Finally, fourteen products are reduced to ten products, as shown in the following:

$$\begin{aligned}Pr\{VMS\} =& Pr\{\neg A \cdot C \cdot S \cdot rep(\theta_S, \theta_S)\}+\\ & Pr\{\neg A \cdot S \cdot rep(\theta_S, \theta_C)\}+\\ & Pr\{\neg A \cdot C \cdot rep(\theta_S, \theta_B)\}+\\ & Pr\{A \cdot \neg B \cdot C \cdot S \cdot rep(\theta_S, \theta_S)\}+\\ & Pr\{A \cdot \neg B \cdot S \cdot rep(\theta_S, \theta_C)\}+\\ & Pr\{\neg B \cdot C \cdot rep(\theta_S, \theta_A)\}+\\ & Pr\{A \cdot B \cdot \neg C \cdot S \cdot rep(\theta_S, \theta_S)\}+\\ & Pr\{A \cdot \neg C \cdot S \cdot rep(\theta_S, \theta_B)\}+\\ & Pr\{B \cdot \neg C \cdot S \cdot rep(\theta_S, \theta_A)\}+\\ & Pr\{A \cdot B \cdot C\}\end{aligned} \tag{37}$$

Similarly, for the purpose of verification, Markov, SBDD and ABDD analysis are performed on the same VMS case study. Assume that all the component failures in the VMS follow an exponential distribution with parameter $a$. The warm spare component can fail either before or after the primary component with failure rate $d_S$ and $a_S$, respectively. Similar to the assumption in Section 5, the failure PDF are $f_{S,d}(\tau_1) = d_S e^{-d_S \tau_1}$ ($\theta_S$ before replacement), $f_{S,a}(\tau_1) = a_S e^{-a_S \tau_1}$ ($\theta_S$ after replacement), $f_{A_a}(\tau_2) = a_S e^{-a_A \tau_2}$ ($\theta_A$), $f_{B_a}(\tau_3) = a_S e^{-a_B \tau_3}$ ($\theta_B$), and $f_{C_a}(\tau_4) = a_S e^{-a_C \tau_4}$ ($\theta_C$), respectively. The unreliability of the VMS is calculated by an expression with ten integral terms shown in Eq. (B.1) (in Appendix B).

The Markov model of the VMS example is shown in Fig. B.1. According to Fig. B.1, we have to solve the Markov chain with 11 states and 21 transitions. As a result, 11 different differential equations are shown in the Eq. (B.2).

The converted fault tree of VMS prepares to generate the SBDD is shown in Fig. B.2. The SBDD (Tannous et al., 2011) of VMS is shown in Fig. B.3 under the index order $(C \rightarrow S) < (S \rightarrow C) < (A \rightarrow C) < (B \rightarrow C) < (A \rightarrow S) < (S \rightarrow A) < (B \rightarrow S) < (S \rightarrow B) < A < B < C$. Different from Fig. 33, the path from top to terminal node 1 in Fig. B.3 is not necessarily able to be used for qualitative analysis since some of them may be invalid (for example, $\neg(C \rightarrow S) \Rightarrow \neg(S \rightarrow C) \Rightarrow \neg(A \rightarrow C) \Rightarrow \neg(B \rightarrow C) \Rightarrow (A \rightarrow S) \Rightarrow \neg(B \rightarrow S) \Rightarrow \neg(S \rightarrow B) \Rightarrow A \Rightarrow B \Rightarrow C$). Each path needs to be analyzed to identify its correctness.

**Table 2**
Analysis results for the VMS.

| Method | t (days) | | |
|---|---|---|---|
| | 300 | 500 | 900 |
| CBDD | 0.466659 | 0.747732 | 0.956783 |
| Markov | 0.466659 | 0.747732 | 0.956783 |
| SBDD | 0.466659 | 0.747732 | 0.956783 |
| ABDD | 0.466659 | 0.747732 | 0.956783 |

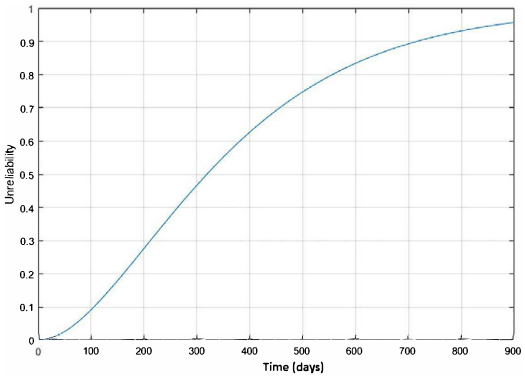



**Fig. 34.** VMS unreliability time distribution (components exponential failure distribution).

The converted fault tree of VMS for building the ABDD is shown in Fig. B.4. However, the ABDD does not eliminate inconsistencies during generation (It handles invalid paths after ABDD generation). As a result, the ABDD of VMS seems to be more complex than both the CBDD and the SBDD, as shown in Fig. B.5. Its index order is $C < S < A < B < C < (S \triangleleft C) < (A \triangleleft C) < (B \triangleleft C) < (C \triangleleft S) < (A \triangleleft S) < (S \triangleleft B) < (B \triangleleft S) < (B \triangleleft A) < (S \triangleleft A) < (A \triangleleft B)$. Remove inconsistencies or reduce some paths when calculating paths from the top to terminal node 1 after the final ABDD is constructed.

Setting parameter values of $d_S = 0.001/day$, $a_S = 0.0025/day$, $a_A = 0.002/day$, $a_B = 0.0025/day$, and $a_C = 0.003/day$, we use Matlab 2016 to compute the system unreliability results from the proposed method, the Markov approach (the initial state (at time 0) is state 1), the SBDD method, and the ABDD method, which match for three different mission times accurately, as shown in Table 2. The VMS unreliability time distribution under the condition of components exponential failure distribution is shown in Fig. 34.

However, the I/E based method can also be used to compute the VMS unreliability via evaluating an expression of $(2^n - 1)$ terms (Dugan and Doyle, 1996) based on $n$ MCQs in general. For our example, the I/E-based method involves at least 8 MCSs deduced from Fig. 29. According to the I/E-based formula, an expression of $(2^8 - 1) = 255$ terms (each term involves integrals) is required to evaluate the VMS unreliability if internal mutually exclusive is not considered. Compared to 255 terms, our proposed method, which only computes an expression of 10 integrals (they are included in 255 terms), is more efficient than the I/E-based method in the VMS example.

### 7.3. Case study III

We will illustrate a more complex example than the previous two case studies. A multiprocessor computing system (Montani et al., 2006) is shown in Fig. 35. The DFT of the multiprocessor computing system involves a logic AND gate, three logic OR gates, four WSP gates including two shared spare component WSP gates, and a (one-short) PDEP gate (Portinale et al., 2010).

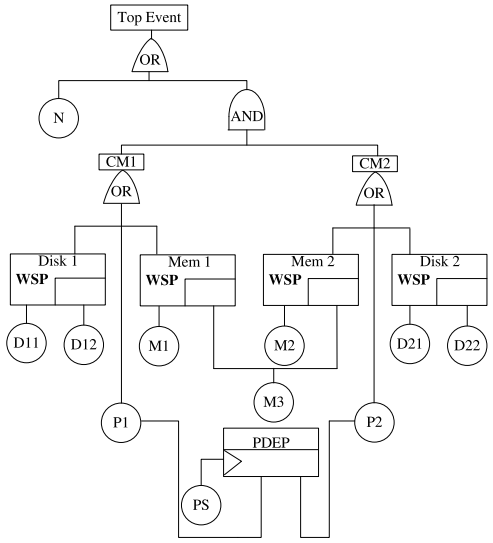



**Fig. 35.** DFT model of the multiprocessor computing system..

The function of the system consists of two computing modules ($CM1$ and $CM2$). $CM1$ includes two disks (a primary disk $\theta_{D11}$ and a spare disk $\theta_{D12}$), processor $\theta_{P1}$, and memory $\theta_{M1}$. Similarly, the structure of $CM2$ is the same as $CM1$. $\theta_{M1}$ in $CM1$ shares a spare memory $\theta_{M3}$ with $\theta_{M2}$ in $CM2$. Moreover, a power supply $\theta_{PS}$ supports both $\theta_{P1}$ and $\theta_{P2}$, and all data from both two computing modules transit through the Bus $\theta_N$.

**Step 1 — CFT Conversion**

To WSP gates, we can directly convert them to logic AND gates with specific conditioning events. For a one-short PDEP gate that is a special case of FDEP, the trigger event leads the dependency events to fail with a probability $P \leq 1$. Here, we set $P = 1$ as the same as Montani et al. (2006). Hence, we can treat this PDEP gate as a FDEP gate. In this condition, once the failure of $\theta_{PS}$ cases both $\theta_{P1}$ and $\theta_{P2}$ fail, which leads the system to fail. The CFT of the multiprocessor computing system is shown in Fig. 36.

**Step 2 — System CBDD Model Generation**

A similar operation with the previous two case studies is used to build the sub_CBDD of each gate. Then, merge sub_CBDDs by using Boolean operation rules. During the operation, eliminate potential inconsistencies and simplify. The index of variables is $N < PS < P1 < M1 < M3 < M2 < rep(\theta_{M3}, \theta_{M1}) < rep(\theta_{M3}, \theta_{M3}) < rep(\theta_{M3}, \theta_{M2}) < D11 < D12 < rep(\theta_{D12}, \theta_{D11}) < rep(\theta_{D12}, \theta_{D12}) < P2 < D21 < D22 < rep(\theta_{D21}, \theta_{D22}) < rep(\theta_{D22}, \theta_{D22})$. The final CBDD under the index determined by PNFO is shown in Fig. 37.

**Step 3 — System CBDD Model Evaluation**

There are fifty-nine paths from the top node N to terminal nodes '1', which corresponds to fifty-nine computing items for the system failure. The negation conditioning events are converted into the following basic events:

$$\neg rep(\theta_{M3}, \theta_{M1}) = \neg M1 + M1 \cdot rep(\theta_{M3}, \theta_{M2}) + M1 \cdot rep(\theta_{M3}, \theta_{M3}) \quad (38)$$

$$\neg rep(\theta_{M3}, \theta_{M2}) = \neg M2 + M2 \cdot rep(\theta_{M3}, \theta_{M1}) + M2 \cdot rep(\theta_{M3}, \theta_{M3}) \quad (39)$$

$$\neg rep(\theta_{M3}, \theta_{M3}) = rep(\theta_{M3}, \theta_{M1}) + rep(\theta_{M3}, \theta_{M2}) \quad (40)$$

$$\neg rep(\theta_{D12}, \theta_{D11}) = \neg D11 + D11 \cdot rep(\theta_{D12}, \theta_{D12}) \quad (41)$$

$$\neg rep(\theta_{D12}, \theta_{D12}) = rep(\theta_{D12}, \theta_{D11}) \quad (42)$$

$$\neg rep(\theta_{D22}, \theta_{D21}) = \neg D21 + D21 \cdot rep(\theta_{D22}, \theta_{D22}) \quad (43)$$

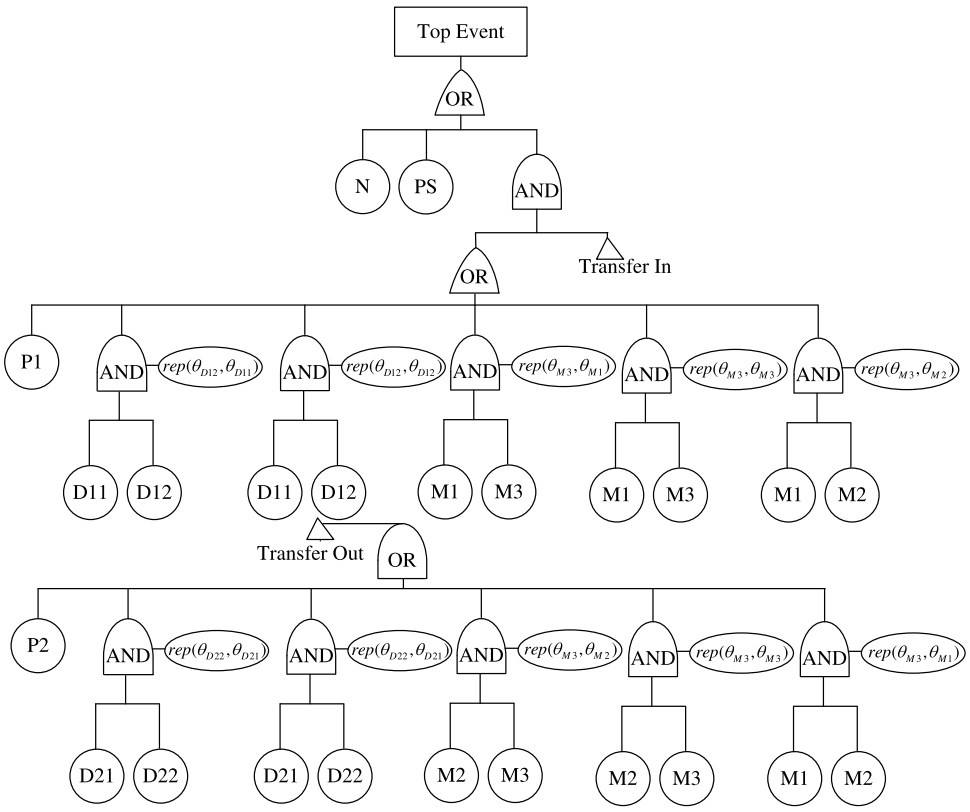



**Fig. 36.** CFT model of the multiprocessor computing system.

$$\neg rep(\theta_{D22},\theta_{D22}) = rep(\theta_{D22},\theta_{D21}) \tag{44}$$

For fifty-nine calculation items, according to Eq. (38) to Eq. (44), replace negation conditioning events and remove inconsistencies. Obviously, there are six products that can be reduced as three products. $Pr\{\neg N \cdot \neg PS \cdot \neg P1 \cdot M1 \cdot M2 \cdot M3 \cdot rep(\theta_{M3},\theta_{M1})\}$ and $Pr\{\neg N \cdot \neg PS \cdot P1 \cdot M1 \cdot M2 \cdot M3 \cdot rep(\theta_{M3},\theta_{M1})\}$ can be merged into $Pr\{\neg N \cdot \neg PS \cdot M1 \cdot M2 \cdot M3 \cdot rep(\theta_{M3},\theta_{M1})\}$. Similarly, we can obtain $Pr\{\neg N \cdot \neg PS \cdot M1 \cdot M2 \cdot M3 \cdot rep(\theta_{M3},\theta_{M3})\}$ from $Pr\{\neg N \cdot \neg PS \cdot \neg P1 \cdot M1 \cdot M2 \cdot M3 \cdot rep(\theta_{M3},\theta_{M3})\}$ and $Pr\{\neg N \cdot \neg PS \cdot P1 \cdot M1 \cdot M2 \cdot M3 \cdot rep(\theta_{M3},\theta_{M3})\}$. $Pr\{\neg N \cdot \neg PS \cdot M1 \cdot M2 \cdot M3 \cdot rep(\theta_{M3},\theta_{M2})\}$ is from $Pr\{\neg N \cdot \neg PS \cdot \neg P1 \cdot M1 \cdot M2 \cdot M3 \cdot rep(\theta_{M3},\theta_{M2})\}$ and $Pr\{\neg N \cdot \neg PS \cdot P1 \cdot M1 \cdot M2 \cdot M3 \cdot rep(\theta_{M3},\theta_{M2})\}$. All assumptions are the same as the previous two case studies. We also assume each component in the multiprocessor computing system failure follows an exponential distribution with parameter $a$. Setting parameter values of $a_N = 2 \times 10^{-9}/day$, $a_{PS} = 6 \times 10^{-6}/day$, $a_{P1} = a_{P2} = 5 \times 10^{-7}/day$, $a_{D11} = a_{D21} = 8 \times 10^{-5}/day$, $a_{D12} = a_{D22} = 8 \times 10^{-5}/day(\theta_{D12}$ or $\theta_{D22}/day$ after replacement), $d_{D12} = a_{D22} = 4 \times 10^{-5}/day(\theta_{D12}$ or $\theta_{D22}$ before replacement), $a_{M1} = a_{M2} = 3 \times 10^{-8}/day$, $a_{M3} = 3 \times 10^{-8}/day(\theta_{M3}$ after replacement), and $d_{M3} = 1.5 \times 10^{-8}/day(\theta_{M3}$ before replacement). Finally, we also are using Markov (212 states), SBDD, and ABDD to verify the result of our CBDD relating to the multiprocessor computing system. The system unreliability results computed by Matlab from different methods match for three different mission times accurately, as shown in Table 3.

35 MCQs/MCSs (32 MCQs, 3 MCSs) can be deduced from Fig. 36. If we use the I/E-based method to compute the unreliability, we have to evaluate an expression of $(2^{35}-1)$ terms (including integrals terms) if internal mutually exclusive is not considered. Compared to $(2^{35}-1)$ terms, our proposed method, which only computes an expression including 33 integrals terms and 3 non-integrals terms. However, even if we consider mutually exclusive between 32 MCQs, the number of terms based on the I/E-based method is much more than ours. Hence, the proposed method is more efficient than the I/E-based method in the multiprocessor computing system example.

**Table 3**
Analysis results for the multiprocessor computing system.

| Method | t (days) | | |
|---|---|---|---|
| | 500 | 1000 | 1500 |
| CBDD | 0.002998 | 0.006009 | 0.009072 |
| Markov | 0.002998 | 0.006009 | 0.009072 |
| SBDD | 0.002998 | 0.006009 | 0.009072 |
| ABDD | 0.002998 | 0.006009 | 0.009072 |

## 8. Conclusion

In this paper, we have presented an efficient reliability analysis method of DFTs with spare gates based on CBDD via conditional transformation. CBDD can be applicable to both spare systems (DFTs model with spare gates) and static systems (SFT models) with any arbitrary component time-to-failure distributions and different component failure parameter values. We proved that our MCS (with conditioning events) is more intuitive and efficient than MCQ in reliability analysis of spare gates. Additionally, our MCS can help engineers locate component faulty by conditional events needless to search current or historical records associated with fault components. We demonstrated that the MCS could be explained by sequence-dependent failure behaviors used in algebraic-structure-based methods (Merle et al., 2011a,b). Compared to SBDD (Xing et al., 2012; Tannous et al., 2011) and ABDD (Jiang et al., 2018), our proposed solution avoids sequence-dependent (or temporal logical) nodes and uses MCSs instead of MCQs to analyze system reliability. Although they

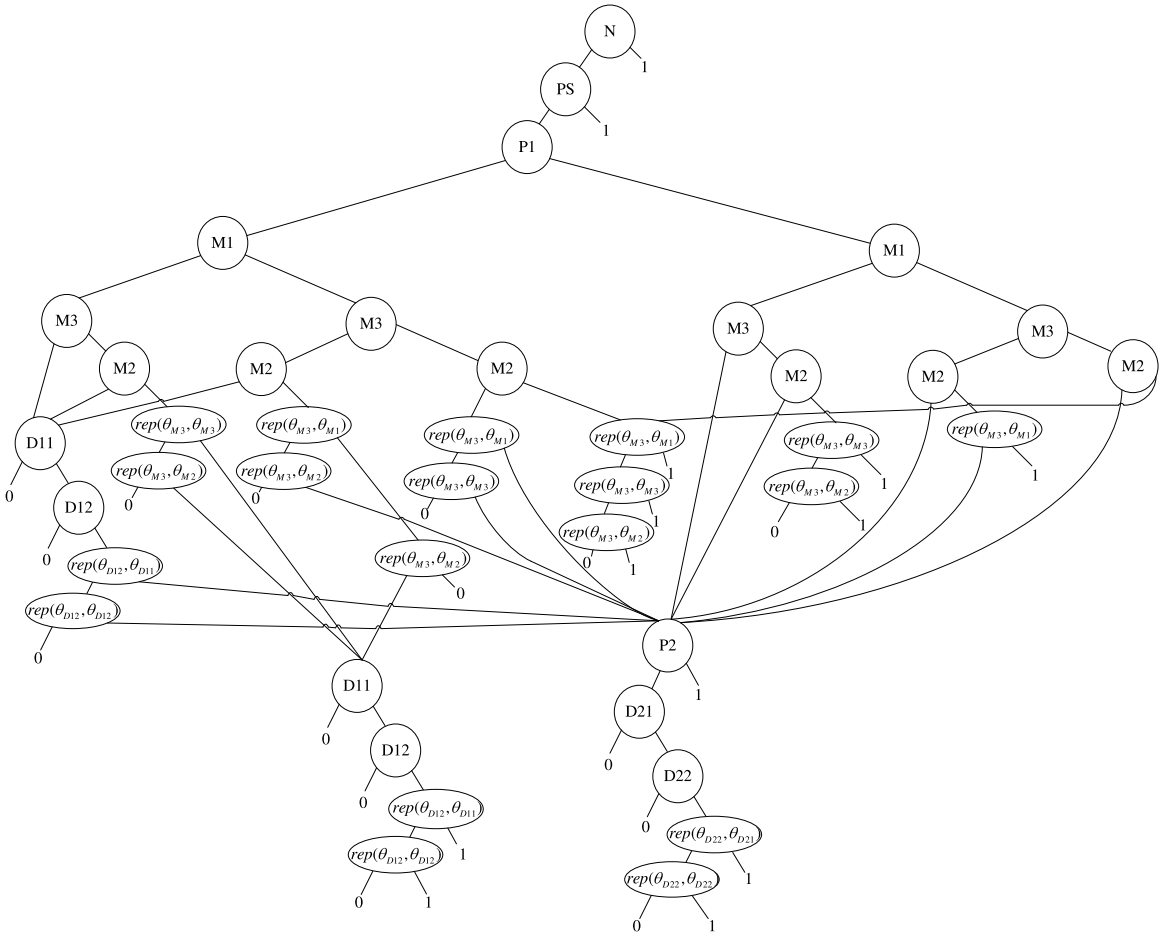



**Fig. 37.** CBDD of the multiprocessor computing system..

are converted to temporal logic to compute failure probability via integrals, CBDD replaces MCQs with MCSs by transferring sequence-dependent behaviors to static conditional status. Based on the data collected in our case studies, our proposed method is more efficient than the MCQ combining I/E-based method. At last, the quantitative analysis based on the CBDD has been presented by means of a hard disk system, VMS examples, and multiprocessor computing system example.

In future work, our proposed method will be expanded to DFTs including other dynamic gates (PAND gates and SEQ gates) reliability analysis. Also,we will focus on some scenarios (including the complex shared spare SP gate and other cases in Junges et al. (2016)) which we have not implemented the automation yet. Finally, we will develop an integrated solution based on CBDDs for reliability analysis of dynamic fault trees including monotonic or non-monotonic and repairable or non-repairable.

## Declaration of competing interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

## Acknowledgments

This work was partially supported by the National Natural Science Foundation of China (Grant No. 61672398 and 61806151), the Defense Industrial Technology Development Program (Grant No. JCKY2018110C165), the Hubei Provincial Natural Science Foundation of China (Grant No. 2017CFA012), and US National Science Foundation (Award No. 1822137 and 1757828). It also received financial support from the China Scholarship Council. The paper was prepared while Siwei Zhou visited the University of Texas at Dallas from September 2017 to September 2019.

## Appendix A. Analysis of WSP gate with subtrees inputs based on the CBDD

Based on Fig. 15 , there are fourteen cut sets (paths from top to terminal "1"). However, some cut sets can be combined as one. Cut sets after removing negation conditioning events can be converted into a sequence-dependent model of algebraic structure-function. This can help users gain a better understanding of the dependency between basic events. The negation conditioning events can be replaced as follows:

$$\neg rep(\theta_D, \theta_C) = \neg C + C \cdot rep(\theta_D, \theta_D)$$
$$\neg rep(\theta_D, \theta_D) = rep(\theta_D, \theta_C)$$
$$\neg rep(T, T) = rep(T, \theta_A) + rep(T, \theta_B)$$
$$\neg rep(T, \theta_A) = \neg A + rep(T, \theta_B) + rep(T, T)$$
$$\neg rep(T, \theta_B) = \neg B + rep(T, \theta_A) + rep(T, \theta_B)$$

Eight merged paths replacing fourteen are used for evaluation, shown as follows:

$$1\ Pr\{C \cdot D \cdot E \cdot\ \neg A \cdot B \cdot \neg rep(\theta_D, \theta_C) \cdot rep(\theta_D, \theta_D)$$

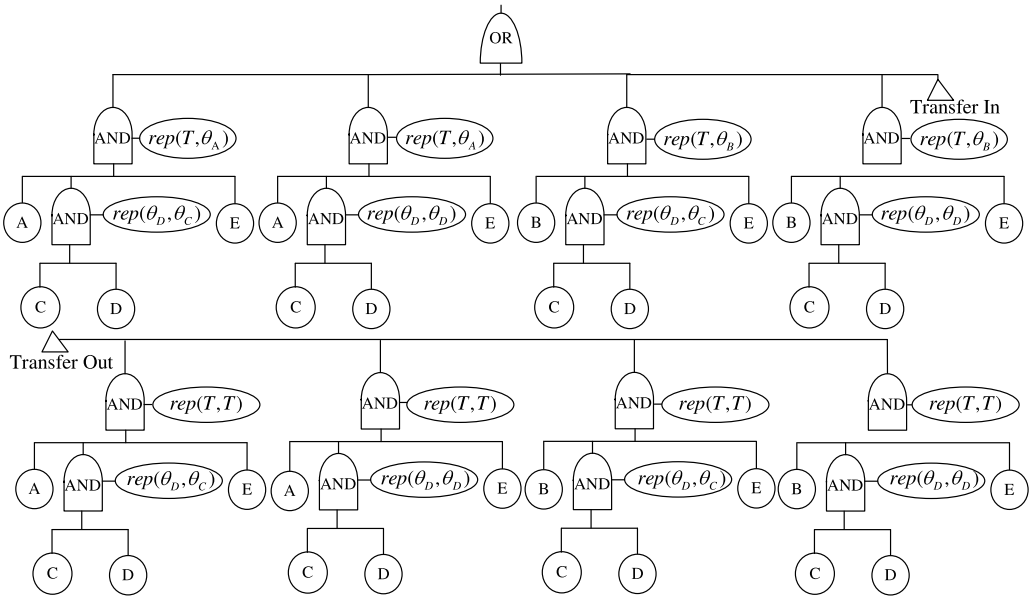



**Fig. A.1.** CFT of WSP gate with subtrees inputs.

$\cdot \neg rep(T,T) \cdot rep(T,\theta_B) + C \cdot D \cdot E \cdot\ A \cdot B \cdot \neg rep(\theta_D,\theta_C)$
$\cdot rep(\theta_D,\theta_D) \cdot \neg rep(T,T) \cdot \neg rep(T,\theta_A) \cdot rep(T,\theta_B)\}$
$= Pr\{C \cdot D \cdot E \cdot B \cdot rep(\theta_D,\theta_D) \cdot rep(T,\theta_B)\}$
$= Pr\{(B \lhd ((D_d \lhd C) \cdot C \cdot E)) \cdot ((D_d \lhd C) \cdot C \cdot E) \cdot (B \lhd A)\}$
$= Pr\{(B \lhd C) \cdot (E \lhd B) \cdot (D_d \lhd C) \cdot (B \lhd A) \cdot C\}$
$+ Pr\{(B \lhd E) \cdot (D_d \lhd C) \cdot (B \lhd A) \cdot C \cdot E\}$

2 $Pr\{C \cdot D \cdot E \cdot\ \neg A \cdot B \cdot \neg rep(\theta_D,\theta_C) \cdot rep(\theta_D,\theta_D)$
$\cdot rep(T,T) + C \cdot D \cdot E \cdot\ A \cdot B \cdot \neg rep(\theta_D,\theta_C)$
$\cdot rep(\theta_D,\theta_D) \cdot rep(T,T)\}$
$= Pr\{C \cdot D \cdot E \cdot B \cdot rep(\theta_D,\theta_D) \cdot rep(T,T)\}$
$= Pr\{(((D_d \lhd C) \cdot C \cdot E) \lhd A)$
$\cdot (((D_d \lhd C) \cdot C \cdot E) \lhd B) \cdot B\}$
$= Pr\{(D_d \lhd C) \cdot (C \lhd A) \cdot (E \lhd A) \cdot (C \lhd B)$
$\cdot (E \lhd B) \cdot (B \lhd A)\}$
$+ Pr\{(D_d \lhd C) \cdot (C \lhd A) \cdot (E \lhd A) \cdot (C \lhd B)$
$\cdot (E \lhd B) \cdot (A \lhd B) \cdot B\}$

3 $Pr\{C \cdot D \cdot E \cdot\ \neg A \cdot B \cdot rep(\theta_D,\theta_C) \cdot \neg rep(\theta_D,\theta_D)$
$\cdot \neg rep(T,T) \cdot rep(T,\theta_B) + C \cdot D \cdot E \cdot\ A \cdot B \cdot rep(\theta_D,\theta_C)$
$\cdot \neg rep(\theta_D,\theta_D) \cdot \neg rep(T,T) \cdot \neg rep(T,\theta_A) \cdot rep(T,\theta_B)\}$
$= Pr\{C \cdot D \cdot E \cdot B \cdot rep(\theta_D,\theta_C) \cdot rep(T,\theta_B)\}$
$= Pr\{(B \lhd ((C \lhd D) \cdot C \cdot E)) \cdot ((C \lhd D) \cdot C \cdot E) \cdot (B \lhd A)\}$
$= Pr\{(E \lhd B) \cdot (B \lhd D_a) \cdot (C \lhd D_a) \cdot (B \lhd A) \cdot D_a\}$
$+ Pr\{(B \lhd E) \cdot (B \lhd D_a) \cdot (C \lhd D_a) \cdot (B \lhd A) \cdot D_a \cdot E\}$
$+ Pr\{(C \lhd D_d) \cdot (D_d \lhd B) \cdot (B \lhd E) \cdot (B \lhd A) \cdot C \cdot E\}$

To $(C \lhd D_d) \cdot (D_d \lhd B) \cdot (B \lhd E) \cdot (B \lhd A) \cdot C \cdot E$ replaces the primary $\theta_C$ and subtree $T$ replaces primary $\theta_B$ ($\theta_C$ activates $\theta_D$ and $\theta_B$ actives $T$), but $\theta_D$ fails before $T$ replaces $\theta_B$ ($T$ is activated). Hence, $\theta_D$ fails in the dormant state. The same case is in calculation formula 7.

Although primary $\theta_C$ fails and spare $\theta_D$ replaces it ($\theta_C$ activate $\theta_D$), all components in the subtree $T$ (Fig. 14) remain in the dormant state since primary $\theta_A$ or $\theta_B$ is not replaced by the subtree $T$. Thus, the $\theta_D$ remains in the dormant state because the $T$ is not activated (Boudali et al., 2010). The same case is considered in the following calculation formulas (4 and 8).

4 $Pr\{C \cdot D \cdot E \cdot\ \neg A \cdot B \cdot rep(\theta_D,\theta_C) \cdot \neg rep(\theta_D,\theta_D)$
$\cdot rep(T,T) + C \cdot D \cdot E \cdot\ A \cdot B \cdot rep(\theta_D,\theta_C)$
$\cdot \neg rep(\theta_D,\theta_D) \cdot rep(T,T)\}$
$= Pr\{C \cdot D \cdot E \cdot B \cdot rep(\theta_D,\theta_C) \cdot rep(T,T)\}$
$= Pr\{(((C \lhd D_d) \cdot D_d \cdot E) \lhd A)$
$\cdot (((C \lhd D_d) \cdot D_d \cdot E) \lhd B) \cdot B\}$
$= Pr\{(C \lhd D_d) \cdot (D_d \lhd A) \cdot (E \lhd A) \cdot (D_d \lhd B)$
$\cdot (E \lhd B) \cdot (B \lhd A)\}$
$+ Pr\{(C \lhd D_d) \cdot (D_d \lhd A) \cdot (E \lhd A) \cdot (D_d \lhd B)$
$\cdot (E \lhd B) \cdot (A \lhd B) \cdot B\}$

5 $Pr\{C \cdot D \cdot E \cdot\ A \cdot \neg B \cdot \neg rep(\theta_D,\theta_C) \cdot rep(\theta_D,\theta_D)$
$\cdot \neg rep(T,T) \cdot rep(T,\theta_A) + C \cdot D \cdot E \cdot\ A \cdot B \cdot \neg rep(\theta_D,\theta_C)$
$\cdot rep(\theta_D,\theta_D) \cdot \neg rep(T,T) \cdot rep(T,\theta_A)\}$
$= Pr\{C \cdot D \cdot E \cdot B \cdot rep(\theta_D,\theta_D) \cdot rep(T,\theta_A)\}$
$= Pr\{(A \lhd ((D_d \lhd C) \cdot C \cdot E)) \cdot ((D_d \lhd C) \cdot C \cdot E) \cdot (A \lhd B)\}$
$= Pr\{(A \lhd C) \cdot (E \lhd A) \cdot (D_d \lhd C) \cdot (A \lhd B) \cdot C\}$
$+ Pr\{(A \lhd E) \cdot (D_d \lhd C) \cdot (A \lhd B) \cdot C \cdot E\}$

6 $Pr\{C \cdot D \cdot E \cdot\ A \cdot \neg B \cdot \neg rep(\theta_D,\theta_C) \cdot rep(\theta_D,\theta_D)$
$\cdot rep(T,T)\}$
$= Pr\{C \cdot D \cdot E \cdot A \cdot \neg B \cdot rep(\theta_D,\theta_D) \cdot rep(T,T)\}$
$= Pr\{(((D_d \lhd C) \cdot C \cdot E) \lhd A) \cdot A \cdot \neg B\}$
$= Pr\{(D_d \lhd C) \cdot (C \lhd A) \cdot (E \lhd A) \cdot A \cdot \neg B\}$

7 $Pr\{C \cdot D \cdot E \cdot\ A \cdot \neg B \cdot rep(\theta_D,\theta_C) \cdot \neg rep(\theta_D,\theta_D)$
$\cdot \neg rep(T,T) \cdot rep(T,\theta_A) + C \cdot D \cdot E \cdot\ A \cdot B \cdot rep(\theta_D,\theta_C)$
$\cdot \neg rep(\theta_D,\theta_D) \cdot \neg rep(T,T) \cdot rep(T,\theta_A)\}$
$= Pr\{C \cdot D \cdot E \cdot A \cdot rep(\theta_D,\theta_C) \cdot rep(T,\theta_A)\}$
$= Pr\{(A \lhd ((C \lhd D) \cdot C \cdot E)) \cdot ((C \lhd D) \cdot C \cdot E) \cdot (A \lhd B)\}$
$= Pr\{(E \lhd A) \cdot (A \lhd D_a) \cdot (C \lhd D_a) \cdot (A \lhd B) \cdot D_a\}$
$+ Pr\{(A \lhd E) \cdot (A \lhd D_a) \cdot (C \lhd D_a) \cdot (A \lhd B) \cdot D_a \cdot E\}$

$$+ Pr\{(C \lhd D_d) \cdot (D_d \lhd A) \cdot (A \lhd E) \cdot (A \lhd B) \cdot C \cdot E\}$$

$$8\ Pr\{C \cdot D \cdot E \cdot A \cdot \neg B \cdot rep(\theta_D, \theta_C) \cdot \neg rep(\theta_D, \theta_D) \cdot rep(T, T)\}$$

$$= Pr\{C \cdot D \cdot E \cdot B \cdot rep(\theta_D, \theta_C) \cdot rep(T, T)\}$$

$$= Pr\{(((C \lhd D_d) \cdot D_d \cdot E) \lhd A) \cdot A \cdot \neg B\}$$

$$= Pr\{(C \lhd D_d) \cdot (D_d \lhd A) \cdot (E \lhd A) \cdot A \cdot \neg B\}$$

Each calculation formula from 1 to 8 can be computed by the multiple integral (Merle, 2010). Because the above eight calculation formulas are mutually exclusive, directly summing them can be used for the calculation of the failure probability for the DFT shown in Fig. 14.

## Appendix B. Figures and equations relating to case study II

The following is a calculation of the unreliability of the VMS.

$$\begin{aligned}
U_{VMS} = {} & \left(1 - \int_0^t f_A(\tau_2)d\tau_2\right) \cdot \left(\int_0^t \int_{\tau_1}^t f_{S,d}(\tau_1) f_C(\tau_4) \right. \\
& \left. \times \left(1 - \int_0^{\tau_1} f_B(\tau_3)d\tau_3\right) d\tau_4 d\tau_1\right) + \\
& \left(1 - \int_0^t f_A(\tau_2)d\tau_2\right) \\
& \cdot \left(\int_0^t \int_0^{t-\tau_4} f_C(\tau_4) f_{S,a}(\tau_1) \left(1 - \int_0^{\tau_4} f_{S,d}(\tau_1)d\tau_1\right)\right. \\
& \left. \times \left(1 - \int_0^{\tau_4} f_B(\tau_3)d\tau_1\right) d\tau_1 d\tau_4\right) + \\
& \left(1 - \int_0^t f_A(\tau_2)d\tau_2\right) \\
& \cdot \left(\int_0^t \int_{\tau_3}^t f_B(\tau_3) f_C(\tau_4) \left(1 - \int_0^{\tau_3} f_{S,d}(\tau_1)d\tau_1\right) d\tau_4 d\tau_3\right) + \\
& \left(1 - \int_0^t f_B(\tau_3)d\tau_3\right) \cdot \left(\int_0^t \int_{\tau_1}^t \int_{\tau_1}^t f_{S,d}(\tau_1) f_A(\tau_2) f_C(\tau_4) d\tau_4 d\tau_2 d\tau_1\right) + \\
& \left(1 - \int_0^t f_B(\tau_3)d\tau_3\right) \\
& \cdot \left(\int_0^t \int_0^{t-\tau_4} \int_{\tau_4}^t f_C(\tau_4) f_{S,a}(\tau_1) f_A(\tau_2) \right. \\
& \left. \times \left(1 - \int_0^{\tau_4} f_{S,d}(\tau_1)d\tau_1\right) d\tau_2 d\tau_1 d\tau_4\right) + \\
& \left(1 - \int_0^t f_B(\tau_3)d\tau_3\right) \\
& \cdot \left(\int_0^t \int_{\tau_2}^t f_A(\tau_2) f_C(\tau_4) \left(1 - \int_0^{\tau_2} f_{S,d}(\tau_1)d\tau_1\right) d\tau_4 \tau_2\right) + \\
& \left(1 - \int_0^t f_C(\tau_4)d\tau_4\right) \cdot \left(\int_0^t \int_{\tau_1}^t \int_{\tau_1}^t f_{S,d}(\tau_1) f_A(\tau_2) f_B(\tau_3) d\tau_3 d\tau_2 d\tau_1\right) + \\
& \left(1 - \int_0^t f_C(\tau_4)d\tau_4\right) \\
& \cdot \left(\int_0^t \int_0^{t-\tau_3} \int_{\tau_3}^t f_B(\tau_3) f_{S,a}(\tau_1) f_A(\tau_2) \right. \\
& \left. \times \left(1 - \int_0^{\tau_3} f_{S,d}(\tau_1)d\tau_1\right) d\tau_2 d\tau_1 d\tau_3\right) + \\
& \left(1 - \int_0^t f_C(\tau_4)d\tau_4\right) \\
& \cdot \left(\int_0^t \int_0^{t-\tau_2} \int_{\tau_2}^t f_A(\tau_2) f_{S,a}(\tau_1) f_B(\tau_3) \right. \\
& \left. \times \left(1 - \int_0^{\tau_2} f_{S,d}(\tau_1)d\tau_1\right) d\tau_3 d\tau_1 d\tau_2\right) + \\
& \left(\int_0^t f_A(\tau_2)d\tau_2\right) \cdot \left(\int_0^t f_B(\tau_3)d\tau_3\right) \cdot \left(\int_0^t f_C(\tau_4)d\tau_4\right) \qquad \text{(B.1)}
\end{aligned}$$

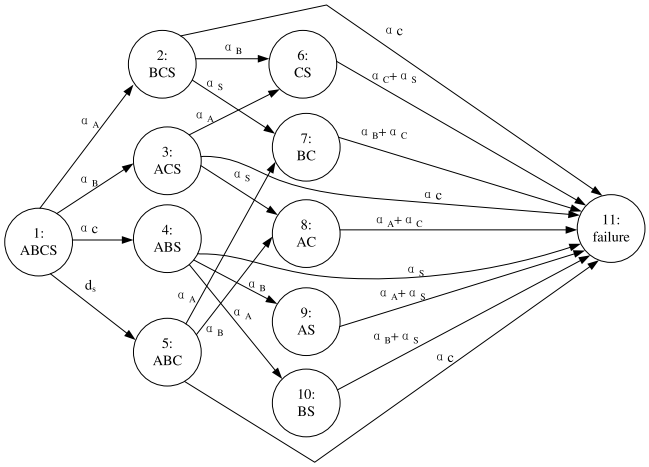



**Fig. B.1.** Markov model of the VMS example.

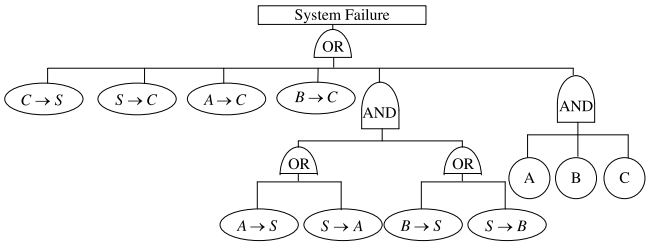



**Fig. B.2.** The converted fault tree of VMS for SBDD.

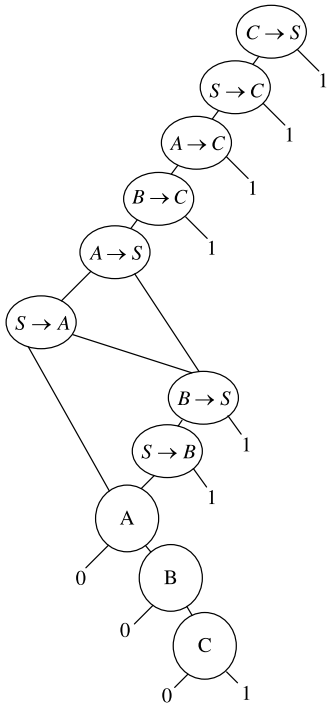



**Fig. B.3.** SBDD of VMS.

The Markov model of the VMS is shown in Fig. B.1. The following are differential equations relating to the Markov chain of the VMS.

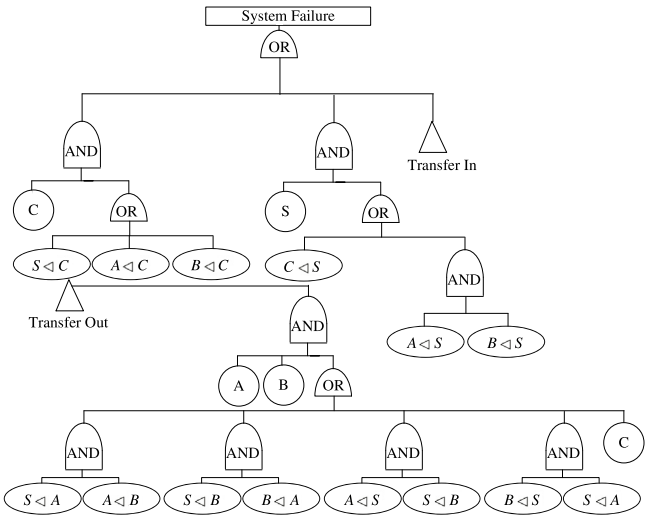



**Fig. B.4.** The converted fault tree of VMS for ABDD.

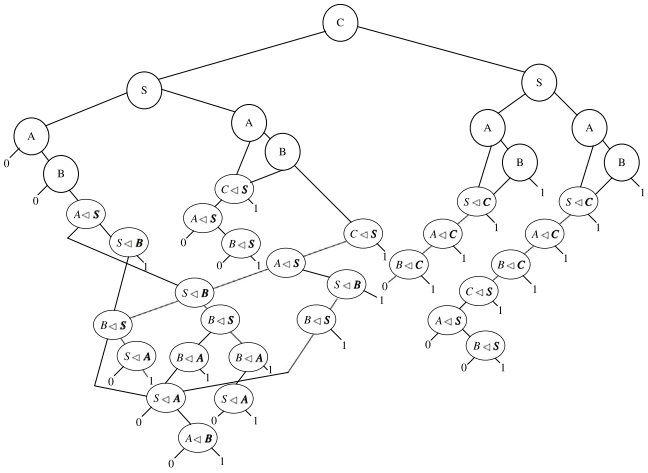


**Fig. B.5.** Complicated ABDD of VMS.

$$P_1(0) = 1 \ \ and \ \ P_i(0) = 0, \ \ i \neq 1$$

$$\frac{\mathrm{d}}{\mathrm{d}t}P_1(t) = -(a_A + a_B + a_C + d_s)P_1(t)$$

$$\frac{\mathrm{d}}{\mathrm{d}t}P_2(t) = a_A P_1(t) - (a_B + a_C + a_S)P_2(t)$$

$$\frac{\mathrm{d}}{\mathrm{d}t}P_3(t) = a_B P_1(t) - (a_A + a_C + a_S)P_3(t)$$

$$\frac{\mathrm{d}}{\mathrm{d}t}P_4(t) = a_C P_1(t) - (a_A + a_B + a_S)P_4(t)$$

$$\frac{\mathrm{d}}{\mathrm{d}t}P_5(t) = d_S P_1(t) - (a_A + a_B + a_C)P_5(t)$$

$$\frac{\mathrm{d}}{\mathrm{d}t}P_6(t) = a_B P_2(t) + a_A P_3(t) - (a_A + a_S)P_6(t)$$

$$\frac{\mathrm{d}}{\mathrm{d}t}P_7(t) = a_S P_2(t) + a_A P_5(t) - (a_B + a_C)P_7(t)$$

$$\frac{\mathrm{d}}{\mathrm{d}t}P_8(t) = a_S P_3(t) + a_B P_5(t) - (a_A + a_C)P_8(t)$$

$$\frac{\mathrm{d}}{\mathrm{d}t}P_9(t) = a_B P_4(t) - (a_A + a_S)P_9(t)$$

$$\frac{\mathrm{d}}{\mathrm{d}t}P_{10}(t) = a_A P_4(t) - (a_B + a_S)P_{10}(t)$$

$$\begin{aligned}\frac{\mathrm{d}}{\mathrm{d}t}P_{11}(t) = {} & a_C P_2(t) + a_C P_3(t) + a_S P_4(t) + a_C P_5(t) \\ & + (a_C + a_S)P_6(t) + (a_B + a_C)P_7(t) \\ & + (a_A + a_C)P_8(t) + (a_A + a_S)P_9(t) \\ & + (a_B + a_S)P_{10}(t)\end{aligned} \tag{B.2}$$

## References


Akers, B, S., 1978. Binary decision diagrams. IEEE Trans. Comput. C-27 (6), 509–516.

ALLEN AO, P., 1990. Statistics and Queuing Theory with Computer Science Applications, Vol. 2. Academic Press.

Bartlett, L.M., Andrews, J.D., 2002. Choosing a heuristic for the "fault tree to binary decision diagram" conversion, using neural networks. IEEE Trans. Reliab. 51 (3), 344–349.

Biggs, N.L., White, A.T., 1979. Permutation Groups and Combinatorial Structures, Vol. 33. Cambridge University Press.

Boudali, H., Crouzen, P., Stoelinga, M., 2010. A rigorous, compositional, and extensible framework for dynamic fault tree analysis. IEEE Trans. Dependable Secure Comput. 7 (2), 128–143.

Boudali, H., Dugan, J.B., 2005. A discrete-time Bayesian network reliability modeling and analysis framework. Reliab. Eng. Syst. Saf. 87 (3), 337–349.

Bouissou, M., 1996. An ordering heuristic for building binary decision diagrams from fault-trees. In: Proceedings of 1996 Annual Reliability and Maintainability Symposium, Jan 1996, pp. 208–214.

Bryant, R.E., 1986. Graph-based algorithms for boolean function manipulation. Comput. IEEE Trans. 100 (8), 677–691.

Commission, I.E., et al., 2016. Electropedia del 192 Dependability. Tech. Rep., 192-01-22 Dependability.

Deng, Y., Wang, H., Guo, B., 2015. BDD algorithms based on modularization for fault tree analysis. Prog. Nucl. Energy 85, 192–199.

Dixon, J.D., Mortimer, B., 1996. Permutation Groups, Vol. 163. Springer Science & Business Media.

Dugan, J.B., Bavuso, S.J., Boyd, M.A., 1992. Dynamic fault-tree models for fault-tolerant computer systems. IEEE Trans. Reliab. 41 (3), 363–377.

Dugan, J.B., Doyle, S.A., 1996. New results in fault-tree analysis. In: Reliability and Maintainability Symposium. pp. 568–573.

Ge, D., Lin, M., Yang, Y., Zhang, R., Qiang, C., 2015. Quantitative analysis of dynamic fault trees using improved Sequential Binary Decision Diagrams. Reliab. Eng. Syst. Saf. 142, 289–299.

Ibáñez-Llano, C., Rauzy, A., Meléndez, E., Nieto, F., 2010. A reduction approach to improve the quantification of linked fault trees through binary decision diagrams. Reliab. Eng. Syst. Saf. 95 (12), 1314–1323.

Jiang, W., Zhou, S., Ye, L., Zhao, D., Tian, J., Wong, W.E., Xiang, J., 2018. An algebraic binary decision diagram for analysis of dynamic fault tree. In: 2018 5th International Conference on Dependable Systems and their Applications. DSA, IEEE, pp. 44–51.

Junges, S., Guck, D., Katoen, J.-P., Stoelinga, M., 2016. Uncovering dynamic fault trees. In: 2016 46th Annual IEEE/IFIP International Conference on Dependable Systems and Networks. DSN, IEEE, pp. 299–310.

Kabir, S., Walker, M., Papadopoulos, Y., 2014. Reliability analysis of dynamic systems by translating temporal fault trees into Bayesian networks. In: Model-Based Safety and Assessment. Springer, pp. 96–109.

Liu, D., Zhang, C., Xing, W., Li, R., Li, H., 2007. Quantification of cut sequence set for fault tree analysis. In: International Conference on High Performance Computing and Communications. Springer, pp. 755–765.

Long, W., 2002. On the quantitative analysis of sequential failure logic using Monte Carlo method for different distributions. In: Proceedings of Probabilistic Safety Assessment and Management, 2002. pp. 391–396.

Lucey, T., Lucey, T., 2002. Quantitative Techniques. Cengage Learning EMEA.

Manian, R., Coppit, D.W., Sullivan, K.J., Bechta Dugan, J., 1999. Bridging the gap between systems and dynamic fault tree models. In: Annual Reliability and Maintainability. Symposium. 1999 Proceedings (Cat. No.99CH36283). pp. 105–111.

Matuzas, V., Contini, S., 2015. Dynamic labelling of BDD and ZBDD for efficient non-coherent fault tree analysis. Reliab. Eng. Syst. Saf. 144, 183–192.

Merle, G., 2010. Algebraic Modelling of Dynamic Fault Trees, Contribution to Qualitative and Quantitative Analysis (Ph.D. dissertation). École normale supérieure de Cachan-ENS Cachan.

Merle, G., Roussel, J.M., Lesage, J.J., 2011a. Algebraic determination of the structure function of Dynamic Fault Trees. Reliab. Eng. Syst. Saf. 96 (2), 267–277.

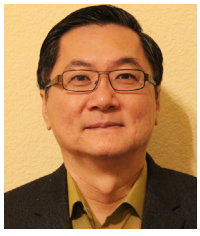
Merle, G., Roussel, J.-M., Lesage, J.-J., 2011b. Dynamic fault tree analysis based on the structure function. In: Reliability and Maintainability Symposium (RAMS), 2011 Proceedings-Annual. IEEE, pp. 1–6.

Merle, G., Roussel, J.-M., Lesage, J.-J., Perchet, V., Vayatis, N., 2016. Quantitative analysis of dynamic fault trees based on the coupling of structure functions and Monte Carlo simulation. Qual. Reliab. Eng. Int. 32 (1), 7–18.

Minato, S.-i., 1993. Zero-suppressed BDDs for set manipulation in combinatorial problems. In: Proceedings of the 30th International Design Automation Conference. ACM, pp. 272–277.
Misra, K.B., 2008. Handbook of Performability Engineering. Springer Science & Business Media.
Mo, Y., 2014. A multiple-valued decision-diagram-based approach to solve dynamic fault trees. IEEE Trans. Reliab. 63 (1), 81–93.
Mo, Y., Xing, L., Dugan, J.B., 2018. Performability analysis of k-to-l-out-of-n computing systems using binary decision diagrams. IEEE Trans. Dependable Secure Comput. 15 (1), 126–137.
Mo, Y., Xing, L., Zhong, F., Zhang, Z., 2016. Reliability evaluation of network systems with dependent propagated failures using decision diagrams. IEEE Trans. Dependable Secure Comput. 13 (6), 672–683.
Mo, Y., Zhong, F., Liu, H., Yang, Q., Cui, G., 2013. Efficient ordering heuristics in binary decision diagram–based fault tree analysis. Qual. Reliab. Eng. Int. 29 (3), 307–315.
Montani, S., Portinale, L., Bobbio, A., Codetta-Raiteri, D., 2006. Automatically translating dynamic fault trees into dynamic Bayesian networks by means of a software tool. In: First International Conference on Availability, Reliability and Security, No. 6, ARES'06. pp. 804–809.
Peng, R., Zhai, Q., Xing, L., Yang, J., 2014. Reliability of demand-based phased-mission systems subject to fault level coverage. Reliab. Eng. Syst. Saf. 121, 18–25.
Portinale, L., Raiteri, D.C., Montani, S., 2010. Supporting reliability engineers in exploiting the power of Dynamic Bayesian Networks. Internat. J. Approx. Reason. 51 (2), 179–195.
Rauzy, A., 1993. New algorithms for fault trees analysis. Reliab. Eng. Syst. Saf. 40 (3), 203–211.
Rauzy, A.B., 2011. Sequence algebra, sequence decision diagrams and dynamic fault trees. Reliab. Eng. Syst. Saf. 96 (96), 785–792.
Ruijters, E., Stoelinga, M., 2015. Fault tree analysis: a survey of the state-of-the-art in modeling, analysis and tools. 15–16 (feb.-may), 29–62.
Sun, Y., Du, S.-g., 2008. A novel ordering method of binary decision diagram. J. Syst. Manag. 2.
Tannous, O., Xing, L., Dugan, J.B., 2011. Reliability analysis of warm standby systems using sequential BDD. In: Reliability and Maintainability Symposium (RAMS), 2011 Proceedings - Annual, pp. 1–7.
Vesely, W., Stamatelatos, M., Dugan, J., Fragola, J., Minarick, III, J., Railsback, J., 2002. Fault Tree Handbook with Aerospace Applications Version 1.1. NASA Office of Safety and Mission Assurance, NASA HQ.
Volk, M., Junges, S., Katoen, J.-P., 2016. Advancing dynamic fault tree analysis - GetSuccinct state spaces fast and synthesise failure rates. In: Computer Safety, Reliability, and Security. Springer International Publishing, pp. 253–265.
Volk, M., Junges, S., Katoen, J., 2018. Fast dynamic fault tree analysis by model checking techniques. IEEE Trans. Ind. Inf. 14 (1), 370–379.
Xiang, J., Machida, F., Tadano, K., Yanoo, K., Sun, W., Maeno, Y., 2013. A static analysis of dynamic fault trees with priority-and gates. In: Dependable Computing (LADC), 2013 Sixth Latin-American Symposium on. IEEE, pp. 58–67.
Xiang, J., Zhou, S., Ye, L., Xiong, S., Wong, W.E., 2016. A generalized multiple-valued decision diagram for reliability analysis of fault-tolerant systems. In: 2016 Third International Conference on Trustworthy Systems and their Applications, TSA, Sept 2016, pp. 36–41.
Xing, L., Dai, Y., 2009. A new decision-diagram-based method for efficient analysis on multistate systems. IEEE Trans. Dependable Secure Comput. 6 (3), 161–174.
Xing, L., Tannous, O., Dugan, J.B., 2012. Reliability analysis of nonrepairable cold-standby systems using sequential binary decision diagrams. IEEE Trans. Syst. Man Cybern. A 42 (3), 715–726.
Zeng, T., 2019. Reliability analysis of cold standby systems based on supplement events using multiple-valued decision diagrams. Int. J. Perform. Eng. 15 (8), 2062–2070.
Zhai, Q., Peng, R., Xing, L., Yang, J., 2015a. Reliability of demand-based warm standby systems subject to fault level coverage. Appl. Stoch. Models Bus. Ind. 31 (3), 380–393.
Zhai, Q., Xing, L., Peng, R., Yang, J., 2015b. Multi-valued decision diagram-based reliability analysis of *k*-out-of-*n* cold standby systems subject to scheduled backups. IEEE Trans. Reliab. 64 (4), 1310–1324.
Zhai, Q., Yang, J., Wang, M., Qiu, Y., Peng, R., 2013. Reliability modeling of warm standby systems subject to fault level coverage based on multi-valued decision diagram. In: Quality, Reliability, Risk, Maintenance, and Safety Engineering (QR2MSE), 2013 International Conference on. IEEE, pp. 50–53.

**Siwei Zhou** received his M.S. in System Analysis and Integration from Hubei University in 2009. He was a visiting Ph.D. student, supported by China Scholarship Council, in Computer Science Department at the University of Texas at Dallas from September 2017 to September 2019. He is currently a Ph.D. candidate at the Wuhan University of Technology. His research is focused on system reliability engineering and dependable computing.

**Jianwen Xiang** received his B.S. and M.S. degrees in Computer Science from Wuhan University in 1997 and 2000, respectively. He received his Ph.D. in Computer Science from Japan Advanced Institute of Science and Technology (JAIST) in 2005. He is currently a full professor of the School of Computer Science and Technology of Wuhan University of Technology, and he was a research scientist of NEC Corporation. His research interests include reliability engineering, formal methods, and software engineering.



**W. Eric Wong** received his M.S. and Ph.D. in Computer Science from Purdue University. He is a full professor and the founding director of the Advanced Research Center for Software Testing and Quality Assurance, an NSF-sponsored I/UCRC site, in Computer Science Department at the University of Texas at Dallas (UTD). He also has an appointment as a guest researcher with National Institute of Standards and Technology (NIST), an agency of the US Department of Commerce. Prior to joining UTD, he was with Telcordia Technologies (formerly Bellcore) as a senior research scientist and the project manager in charge of Dependable Telecom Software Development. In 2014, he was named the IEEE Reliability Society Engineer of the Year. Professor Wong's research focuses on helping practitioners improve the quality of software while reducing the cost of production. In particular, he is working on software testing, debugging, risk analysis/metrics, safety, and reliability. Professor Wong is the Editor-in-Chief of IEEE Transactions on Reliability and the Founding Steering Committee Chair of the IEEE International Conference on Software Quality, Reliability, and Security (QRS).