

**Dynamic Dependability Analysis using HOL
Theorem Proving with Application in
Multiprocessor Systems**

Yassmeen Elderhalli

A Thesis

in

The Department

of

Electrical and Computer Engineering

Presented in Partial Fulfillment of the Requirements

For the Degree of

Doctor of Philosophy (Electrical and Computer Engineering) at

Concordia University

Montréal, Québec, Canada

October 2019

© Yassmeen Elderhalli, 2019

CONCORDIA UNIVERSITY
SCHOOL OF GRADUATE STUDIES

This is to certify that the thesis prepared

By: Yassmeen Farouk Said Elderhalli

Entitled: Dynamic Dependability Analysis using HOL Theorem Proving with
Application in Multiprocessor Systems

and submitted in partial fulfillment of the requirements for the degree of

Doctor Of Philosophy (Electrical and Computer Engineering)

complies with the regulations of the University and meets the accepted standards with respect to originality and quality.

Signed by the final examining committee:

_____ Chair
Dr. Rene Witte

_____ External Examiner
Dr. William M. Farmer

_____ External to Program
Dr. Rajagopalan Jayakumar

_____ Examiner
Dr. Yan Liu

_____ Examiner
Dr. Mohammed Reza Soleymani

_____ Thesis Supervisor
Dr. Sofiène Tahar

Approved by _____
Dr. Rastko Selmic, Graduate Program Director

December 9, 2019

Dr. Amir Asif, Dean
Gina Cody School of Engineering & Computer Science

ABSTRACT

Dynamic Dependability Analysis using HOL Theorem Proving with Application in Multiprocessor Systems

Yassmeen Elderhalli, Ph.D.

Concordia University, 2019

Dynamic dependability analysis has become an essential step in the design process of safety-critical systems to ensure the delivery of a trusted service without failures. Dependability usually encompasses several attributes, such as reliability and availability. A dynamic dependability model is created using one of the dependability modeling techniques, such as Dynamic Fault Trees (DFTs) and Dynamic Reliability Block Diagrams (DRBDs). Several analysis methods, including paper-and-pencil or simulation, exist for analyzing these models to ascertain various dependability related parameters. However, their results cannot be always trusted since they may involve some approximations, truncations or even errors. Formal methods, such as model checking and theorem proving, can be used to overcome these inaccuracy limitations due to their inherent soundness and completeness. However, model checking suffers from state-space explosion if the state space is large. While, theorem proving was used only for the static dependability analysis without considering the system dynamics.

In order to conduct the formal dependability analysis of systems that exhibit dynamic failure behaviors within a theorem prover, these models need to be captured formally, where their structures, operators and properties are properly formalized. In this thesis, we provide a complete framework for the formal dependability analysis of systems modeled as DFTs and DRBDs in the HOL4 higher-order logic theorem

prover. We provide the formalization of DFT gates and verify important simplification theorems based on well-known DFT algebra. In addition, our framework allows both qualitative and quantitative DFT analyses to be conducted using theorem proving. We use this formalization to formally verify the DFT rewrite rules, that are used by automated DFT analysis tools, to ascertain their correctness. Due to the lack of a DRBD algebra that allows the analysis using a theorem prover, in this thesis, we develop and formalize a novel algebra that includes operators and simplification theorems to formalize traditional RBD structures, such as the series and parallel, besides the DRBD spare construct. We formally verify their reliability expressions, which allows conducting both the qualitative and quantitative analyses of a given system. Leveraging upon the complementary nature of DFTs and DRBDs, our proposed framework provides the possibility of formally converting one model to the other, which allows reasoning about both the success and failure of a given system. Our framework provides generic expressions of probability of failure and reliability that are independent of the failure distribution of an arbitrary number of system components, which cannot be obtained using other formal tools, such as model checking. In order to demonstrate the usefulness of the proposed framework, we formally model and analyze the dependability of the terminal, broadcast and network reliability of shuffle-exchange networks, which are multistage interconnections networks that are used to connect the elements of multiprocessor systems. Conducting a sound analysis with generic expressions is essential in these systems, where it is required to accurately capture and analyze the failure behavior.

To my father, my mother, my sister and brothers

ACKNOWLEDGEMENTS

First of all, I am profoundly grateful to my supervisor, Dr. Sofiène Tahar, for his guidance, support and encouragement throughout my Ph.D. studies. His deep expertise in the field of formal methods has greatly boosted my Ph.D. He showed deep confidence in my work and never hesitated to travel and present it when I could not.

I am indebted to Dr. Osman Hasan for providing me with his support, technical feedback, encouragement and advice. I have learned a lot from his deep knowledge and immense experience. His valuable comments and our discussions have helped strengthen this work. Many thanks to Muhammed Qasim for his valuable comments about his HOL4 theories and Muhammad Umair Siddique for his suggestions.

I would like to express my gratitude to Dr. William Farmer for accepting to be my external Ph.D. thesis examiner. I am also grateful to Dr. Rajagopalan Jayakumar, Dr. Reza Soleymani and Dr. Yan Liu for serving on my advisory thesis committee.

I am very thankful to Dr. Asim Al-Khalili for his continuous encouragement and support. He was always available when I needed his advice. Many thanks to my friends and colleagues at the Hardware Verification Group (HVG), specially Hassnaa El-derhalli, Mbarka Soualhia, Mahmoud Masadeh and Saif Najmeddin, for being kind, supportive and helpful. Their company has made my journey at HVG unforgettable.

I am eternally grateful to my father, my sister, Hassnaa, my brothers, Omar and Mohamed, for their unconditional love and support. I would not have been able to reach this stage of my life without them.

TABLE OF CONTENTS

LIST OF TABLES	xi
LIST OF FIGURES	xiv
LIST OF ACRONYMS	xvi
1 Introduction	1
1.1 Motivation	1
1.2 Dynamic Fault Trees	5
1.3 Dynamic Reliability Block Diagrams	9
1.4 Framework for Formal Dynamic Dependability Analysis	13
1.5 Thesis Contributions	16
1.6 Thesis Organization	18
2 Preliminaries	20
2.1 HOL4 Theorem Proving	20
2.2 Probability Theory	21
2.3 Lebesgue Integral	29
3 Formal Qualitative Analysis of Dynamic Fault Trees	31
3.1 Methodology	31
3.2 Identity Elements and Temporal Operators	33
3.3 Formalization of FT Gates	35
3.3.1 AND and OR Gates	36
3.3.2 Priority AND Gate (PAND)	37
3.3.3 Functional DEpendency Gate (FDEP)	38
3.3.4 Spare Gates	39

3.4	Formal Verification of the Simplification Theorems	42
3.5	Formal Qualitative Analysis of DFT Examples	43
3.6	Formal Qualitative Analysis Case Studies	46
3.6.1	Qualitative Analysis of DBW	47
3.6.2	Qualitative Analysis of CAS	49
3.7	Summary	52
4	Formal Quantitative Analysis of Dynamic Fault Trees	53
4.1	Methodology	53
4.2	Probabilistic Model of DFT Gates	55
4.2.1	Probabilistic Model of AND Gate	57
4.2.2	Probabilistic Model of OR and FDEP Gates	58
4.2.3	Probabilistic Model of PAND Gate and Before Operator	60
4.2.4	Probabilistic Model of Spare Gates	69
	Cold Spare Gate	70
	Warm Spare Gate	75
	Spare Gates with a Shared Spare	78
4.3	Formal Quantitative Analysis of DFT Examples	82
4.4	Formal Quantitative Analysis Case Studies	84
4.4.1	Formal Quantitative Analysis of DBW	87
4.4.2	Formal Quantitative Analysis of CAS	90
4.5	Summary	94
5	Formal Verification of DFT Rewrite Rules	97
5.1	DFT Rewrite Rules	98
5.1.1	Rewrite Framework	99

5.1.2	Rewrite Rules	100
	General Rewrite Rules	101
	Rewrite Rules for PAND gates	102
5.1.3	Non-structural Rules	103
	Removing BEs	103
	Merging BEs	104
5.2	HOL Formalization of n -ary DFT Gates	104
5.3	Verification of Rewrite Rules	107
	5.3.1 General Rewrite Rules	107
	5.3.2 Rewrite Rules for PAND Gates	115
	5.3.3 Non-Structural Rules	118
5.4	Summary	119
6	Formal Analysis of Dynamic Reliability Block Diagrams	120
6.1	Methodology	121
6.2	DRBD Event	122
6.3	Identity Elements and Operators	124
6.4	Simplification Theorems	131
6.5	Spare Construct	131
6.6	DRBD Structures	136
6.7	Formal DBW DRBD Analysis	145
6.8	Formal Equivalence of DFT-DRBD Algebras	146
6.9	Summary	151
7	Formal Dependability Analysis of Shuffle-exchange Networks	153
7.1	Overview	153

7.2	Terminal Reliability Analysis of Shuffle-exchange Networks	158
7.2.1	DFT Analysis of SEN and SEN+	158
7.2.2	DRBD Analysis of SEN and SEN+	166
7.3	Broadcast Reliability Analysis of Shuffle-exchange Networks	171
7.3.1	DFT Analysis of SEN and SEN+	172
7.3.2	DRBD Analysis of SEN and SEN+	174
7.4	Network Reliability Analysis of Shuffle-exchange Networks	177
7.4.1	DFT Analysis of SEN and SEN+	178
7.4.2	DRBD Analysis of SEN and SEN+	187
7.5	Equivalence of SEN DFT and DRBD Models	196
7.6	Summary	199
8	Conclusions and Future Work	200
8.1	Conclusions	200
8.2	Future Work	203
	Bibliography	205
	Biography	215

List of Tables

2.1	Some HOL Symbols	22
3.1	DFT Gates Mathematical Expressions	36
3.2	Examples of Formally Verified Simplification Theorems	43
4.1	Failure Rates for the DBW System ($\times 10^{-7}$)	89
4.2	Failure Rates of CAS ($\times 10^{-6}$)	93
6.1	Formally Verified DRBD Simplification Theorems	132
6.2	Mathematical and Reliability Expressions of DRBD Structures	138
6.3	Verified Equivalence of DFT and DRBD Algebras	149
6.4	Comparison of Formal Analysis Efforts of DBW	151

List of Figures

1.1	Some DFT Elements	6
1.2	Dynamic DRBD Constructs	10
1.3	Overview of the Proposed Framework	14
3.1	Formal DFT Qualitative Analysis Methodology	32
3.2	DFT Examples	44
3.3	DFT of Drive-by-wire System	47
3.4	DFT of Cardiac Assist System	49
4.1	Formal DFT Quantitative Analysis Methodology	54
4.2	Probability of Failure of CPAND, AND-FDEP and WSP-OR	85
4.3	Probability of Failure of the Drive-by-wire System	89
4.4	Probability of Failure of the Cardiac Assist System	93
5.1	Subsumption of OR Gates by AND Gates [66, Rewrite Rule 8]	99
5.2	Example Application of Rewrite Rule	100
5.3	Left-flattening of Gates [66, Rewrite Rule 5]	101
5.4	AND/PAND Gate with $\text{CONST}(\perp)$ Successor [66, Rewrite Rule 13]	102
5.5	Conflicting PAND Gates with Independent Successors [66, Rewrite Rule 19]	102

5.6	Example Application of Non-structural Rules	103
6.1	Formal DRBD Analysis Methodology	122
6.2	Two-Block Series and Parallel DRBDs	126
6.3	Spare Construct	133
6.4	DRBD Structures	137
6.5	DRBD of Drive-by-Wire System	145
6.6	Reliability of DBW System	147
6.7	Integrated Framework for Formal DFT-DRBD Analysis using HOL4 . .	147
7.1	Overview of Multiprocessor System Architecture	154
7.2	An 8×8 SEN	156
7.3	An 8×8 SEN+	156
7.4	DFT of SEN	158
7.5	DFT of SEN+ Terminal Connection	162
7.6	Probability of Failure of the Terminal Connection of a 128×128 SEN+ with and without Spares	165
7.7	DRBD of SEN	166
7.8	Terminal Reliability DRBD of SEN+	168
7.9	Terminal Reliability of 128×128 SEN+ with and without Spares . . .	171
7.10	DFT of Broadcast SEN+	172
7.11	Probability of Failure of the Broadcast of a 128×128 SEN+	175
7.12	Broadcast DRBD Model of SEN+	175
7.13	Broadcast Reliability of a 128×128 SEN+	177
7.14	DFT of SEN Network with Multiple Spares	178
7.15	DFT of SEN+ Network	180
7.16	DFT of SEN+ with Multiple Spares	185

7.17	The Probability of Failure of the Network of a 128×128 SEN+	188
7.18	DRBD of SEN Network	188
7.19	DRBD of SEN+ Network	190
7.20	DRBD of SEN+ Network with Multiple Spares	193
7.21	The Network Reliability of a 128×128 SEN+	196

LIST OF ACRONYMS

AE	Almost Everywhere
BC	Brake Control
BDD	Binary Decision Diagram
BE	Basic Event
BS	Brake Sensor
CAS	Cardiac Assist System
CDF	Cumulative Distribution Function
CPAND	Cascaded PAND
CPN	Colored Petri Nets
CPU	Central Processing Unit
CS	Crossbar Switch
CSP	Cold SPare
CTMC	Continuous Time Markov Chain
DAG	Directed Acyclic Graph
DBW	Drive-by-wire
DFT	Dynamic Fault Tree
DRBD	Dynamic Reliability Block Diagram
EF	Engine Failure
ET	Event Tree
FDEP	Functional DEPEndency
FT	Fault Tree
FTA	Fault Tree Analysis
HOL	Higher-order Logic

HSP	Hot SPare
LSH	Load SHaring
MC	Markov Chain
MIN	Multi-stage Interconnection Network
MTTF	Mean-Time-To-Failure
PAND	Priority-AND
PDF	Probability Density Function
PIE	Principle of Inclusion and Exclusion
PMC	Probabilistic Model Checker
RBD	Reliability Block Diagram
SDEP	State DEPEndency
SEN	Shuffle-exchange Network
SFT	Static Fault Tree
SS	System Supervisor
SSD	Solid State Drive
TF	Throttle Failure
TS	Throttle Sensor
VOT	Voting
WSP	Warm SPare

Chapter 1

Introduction

1.1 Motivation

A man who lacks reliability is utterly useless.

- Confucius (551–479 BC)

The recent decades witnessed huge technological advancements, which took part in almost all aspects of our lives. While some of the applications that are related to technology can somehow tolerate errors in their results, there are other applications where errors can lead to financial losses, disasters and in the worst case losses in human lives. Crises such as the Toyota global recall [1] increased and necessitated the need to measure system dependability, which is the ability of a system to provide a trusted service [2]. Dependability analysis is also required in other types of systems, such as data centers, where the loss of financial or personal information cannot be tolerated.

Dependability generally consists of several attributes such as reliability, availability, maintainability, safety and confidentiality. Some of these concepts can be quantified and measured such as reliability and availability. *Reliability* can be defined as the probability that a certain system when subject to specific conditions will deliver its correct function in a given period of time. This means that the probability that a system will not encounter any failure for a specific period can be calculated and analyzed based on certain requirements. On the other hand, *availability* is the probability that the system or a component of a system will provide its service at a certain moment of time [2]. Dependability is commonly modeled using combinatorial models, such as fault trees (FTs) [3], reliability block diagrams (RBDs) [4] and event trees (ETs) [5]. However, these traditional modeling techniques cannot capture the dynamic failure behavior of systems and thus cannot truly model many real-world systems that exhibit sequential failures and dependencies among system components.

A *dynamic dependability*, on the other hand, generally captures the dependent failure and repair sequences that the regular combinatorial dependability models cannot represent. Dynamic dependability analysis is considered as an important step in the design process of any system that possesses a dynamic behavior, especially safety-critical systems. The appropriate modeling technique is chosen to estimate the system dynamic dependability. These modeling techniques include Dynamic Fault Trees [3] and Dynamic Reliability Block Diagrams [6]. A *Dynamic Fault Tree* (DFT) is a graphical representation of the sources of faults that cause the failure of a system represented as the top event of the DFT. It utilizes dynamic gates to model the failure dependencies of the basic events. *Dynamic Reliability Block Diagrams* (DRBDs) have been introduced as an extension to traditional RBDs to capture the dynamic success paths of a given system.

Traditionally, dependability models are analyzed using paper-and-pencil based proof methods or using simulation. The former provides a flexible way to model and analyze systems. However, it is prone to human error. On the other hand, simulation provides an easy and automated method to conduct the analysis, which justifies its common use in analyzing a wide range of applications. For example, in [7], the reliability of Solid State Drives (SSDs) of two different configurations are compared using simulation. However, due to the high computational cost of simulation, only part of the space could be analyzed, and thus the results cannot be termed as accurate or complete.

Formal methods, such as model checking [8] and theorem proving [9] have been used for the analysis of dependability models to overcome the inaccuracy limitations of the above-mentioned techniques. For example, the PRISM model checker [10] has been used in the reliability analysis of many applications, such as defect-tolerant systems [11]. More recently, the STORM model checker [12] has been used in the safety analysis of a vehicle guidance system [13] using DFTs. Although probabilistic model checkers provide an automatic way to conduct the analysis of dependability models, the state space explosion problem often limits its scope especially when analyzing complex systems. Moreover, the reduction algorithms embedded in these tools are usually not formally verified, which questions the accuracy of the reduced models. More importantly, probabilistic model checkers inherently assume the failures to be exponentially distributed for system components [14], and thus cannot capture, for example, the aging factor of these components.

Since the formalization of the probability theory in higher-order logic (HOL) [15,

16, 17, 18], theorem proving has also been used for dependability analysis. For example, some properties for continuous random variables were employed to formally reason about some system reliability properties, such as Mean-Time-To-Failure (MTTF) [19]. Moreover, some reliability theory elements were formally verified and used in the formal analysis of a reconfigurable memory array with stuck-at and coupling faults [20]. In [21] and [22], two frameworks for the analysis of FTs and RBDs, respectively, using theorem proving were proposed and used to formally analyze some real-world applications, like an air traffic management system [22] and a solar array for a satellite system [23]. However, this HOL formalization cannot handle or be extended to verify the dynamic properties of DFTs and DRBDs. Generally, using a theorem prover in the analysis allows having verified generic expressions of dependability that are independent of the distribution of system components. Accordingly, the results are not limited to exponential distributions.

Taking into account the concerns mentioned above about dynamic dependability analysis of systems, in particular safety-critical ones, there is a dire need to have an accurate framework for modeling and analysis of dynamic dependability models. In this thesis, we aim to provide a formal framework for the accurate dynamic dependability analysis of systems modeled as DFTs and DRBDs. Mainly, the idea is to build a mathematical model for the dynamic dependability of the system using either DFTs or DRBDs based on the system description and requirements, and then to utilize the expressiveness and sound nature of HOL theorem proving to perform the analysis of such models to provide generic expressions of dependability. It is worth mentioning that such generic expressions cannot be provided using model checking, which adds to the importance of the proposed framework as the first of its kind in providing sound generic expressions of dynamic dependability models that are formally verified. As

an illustration of the usefulness of the proposed framework, we apply it in verifying some rewrite rules that are used in automatic DFT analysis tools, like the STORM model checker, to reduce DFT models, which demonstrates the soundness of these rules. Furthermore, we utilize this framework to conduct the formal dependability analysis of two safety-critical systems from the medical and automotive domains to reason about their dynamic failure behaviors. Furthermore, we apply our framework for the dependability analysis of real-world systems, namely multiprocessor networks, where the increased number of processing elements requires having a sophisticated interconnection network that must be on one hand efficient with low cost and on the other hand more reliable [24]. In particular, we propose to formally model and conduct the dynamic dependability analysis of shuffle-exchange networks (SENs) [25] using DFTs and DRBDs to reason about their behaviors. SENs are multistage interconnection networks (MINs) [26], which are widely used in multiprocessor systems to establish communication between system nodes, including processors, memories and I/O peripherals.

In this thesis, we use the HOL4 theorem prover [27] to formalize DFTs and DRBDs and perform the above-mentioned analysis. We choose HOL4 as we are using existing theories (libraries), such as probability [17] and Lebesgue integral [28], to build and verify the underlying foundations of these dependability models.

1.2 Dynamic Fault Trees

Dynamic fault trees (DFTs) [3] are introduced to model the failure dependencies among system components that cannot be captured using traditional FTs, i.e., static fault trees (SFTs). A DFT is a graphical representation of the sources of failure of a given system. The modeling starts with an undesired top event that represents



Figure 1.1: Some DFT Elements

the failure of the whole system or a subsystem. Inputs of the DFT represent basic events that contribute to the occurrence (failure) of the top event. The relationships and dependencies among these basic events are modeled using DFT gates, such as Priority-AND (PAND) gate.

DFTs are directed acyclic graphs (DAG) with typed nodes (AND, OR, etc.). Successors of a node v in the DAG are *inputs* of v . Some commonly used DFT elements are shown in Figure 1.1. Nodes without inputs are *basic events* (BE, Figure 1.1(a)) that represent atomic components, which can fail according to a failure distribution. Special cases of BEs are *constant failed* elements ($\text{CONST}(\top)$, Figure 1.1(b)) that always fail and *constant fail-safe* elements ($\text{CONST}(\perp)$, Figure 1.1(c)), that can never fail. DFT *gates* are nodes with inputs and are used to model the state dependencies and redundancies among system components. Some commonly used DFT gates include SFT gates (AND, OR) as well as the PAND DFT gate. The output event of the AND gate (Figure 1.1(d)) fails when both input events fail. The OR gate (Figure 1.1(e)) requires that at least one of its input events fails for the output event to fail. The PAND gate (Figure 1.1(f)) acts in a similar way to the AND gate, i.e., it requires that both input events fail. However, an additional condition is needed, where the inputs should fail in sequence, usually from left to right. There are also other DFT gates that are used to model the dynamic behavior in systems, like the Functional-DEpendency (FDEP) and spare gates.

Fault tree analysis (FTA) can be generally carried out qualitatively or quantitatively [29]. In the *qualitative* analysis, the combinations and sequences of basic events that contribute to the occurrence of the top event (failure of the system) are identified. The sets that include these combinations are called cut sets, while the cut sequences determine the required sequences of failure of the basic events. These combinations and sequences represent the cut sets and cut sequences [3], respectively. In the *quantitative* analysis, attributes, such as the MTTF and the probability of failure, can be evaluated based on the failure distribution of the basic events and their relationships. Dynamic FTA has been commonly conducted using a DFT algebra [30] or by analyzing the corresponding Continuous Time Markov Chain (CTMC) of the given DFT [29]. In the former method, an algebra similar to the ordinary Boolean algebra is defined with some temporal operators and simplification properties that allow the structure function of the top event to be reduced. Based on this structure function, both the qualitative and quantitative analyses can be carried out, where the probability of failure of the DFT's top event can be expressed based on the failure distribution of the basic events. On the other hand, the given DFT can be converted into its equivalent CTMC and then this CTMC is analyzed to find the probability of failure of the top event [29]. Complex systems can generate CTMCs with a large state space. This can be handled by applying a modularization approach, where the DFT is divided into static and dynamic parts. The static FT can be analyzed using one of the conventional methods, such as binary decision diagrams (BDDs) [3]. The dynamic part can then be analyzed by converting it to its corresponding CTMC. This kind of modularization is implemented in the Galileo tool [31].

Dynamic FTA can be conducted analytically to manually generate probability of failure expressions. The cut sets and sequences are identified and then the probabilistic

principle of inclusion and exclusion (PIE) is applied to provide the probability of failure expression [30]. However, the results of this manual manipulation are prone to human error. Simulation, on the other hand, can provide a scalable and automated alternative to conduct the FTA. For example, in [32], DFT analysis is performed by combining the DFT algebra of [33] and Monte Carlo Simulation [34]. There also exist other tools for DFT analysis, such as BlockSim [35], Möbius [36] and isograph [37]; however, as mentioned earlier, their analysis results cannot be termed as complete nor accurate due to the sampling nature of their simulation method.

In order to overcome the limitations of simulation in terms of inaccuracies and completeness, formal methods can be used in the dynamic FTA. The DFTCalc tool [38] analyzes DFTs using an Input/Output Interactive Markov Chain, an extension of CTMCs, which is built based on a compositional aggregation technique [39]. Probabilistic model checking (PMCs) has been utilized to perform the analysis of DFTs. For example, the STORM model checker is used to conduct the quantitative analysis of DFTs in the form of probability of failure and MTTF [40]. However, generic expressions of probability of failure cannot be obtained based on this kind of analysis. Moreover, the failure distributions of the inputs are assumed to be exponential due to the state based nature of PMCs.

On the other hand, HOL theorem proving has been used in [21] to formalize SFTs and reason about their properties. However, this formalization cannot handle the dynamic aspects of real-world systems that are captured using DFTs. Furthermore, it cannot be extended to model and analyze the dynamic behavior of systems, and thus a new formalization is needed.

Due to the high expressive nature of HOL, in [41], we proposed a methodology to

conduct the DFT’s qualitative analysis using the HOL theorem prover and the quantitative analysis using the STORM model checker. We provided the formalization of DFTs based on the algebraic approach [30]. In the algebraic approach, identity elements and temporal operators are defined to express the structure function of a DFT event. Several simplification properties are introduced that facilitate the reduction of this function. However, the arithmetic foundation of this approach was not formally verified, which puts a question mark on the soundness of the reported results. In [41], we provided the formalization of the DFT gates, operators and their simplification theorems. This allows the qualitative analysis to be performed within the sound core of a HOL theorem prover. In addition, we proposed to use the STORM model checker to find the probability of failure of the formally verified reduced DFT structure function. However, we cannot obtain generic expressions of probability of failure based on this methodology as a PMC is involved in the quantitative analysis. Moreover, our definitions in [41] cannot handle the DFT probabilistic analysis. Therefore, in this thesis, we propose new definitions of DFT gates and temporal operators, which allow us to conduct qualitative as well as quantitative analyses of DFTs in the form of generic expressions of probability of failure in a theorem prover.

1.3 Dynamic Reliability Block Diagrams

A dynamic reliability block diagram (DRBD) models the paths of success in a given system using system components as blocks that are connected in the traditional series, parallel, series-parallel and parallel-series structures. The connections between system blocks are modeled using connectors (lines) to create one or more paths from the DRBD input to its output. These paths represent the required working blocks (system components) for the system to have a successful operation. The modeled system fails

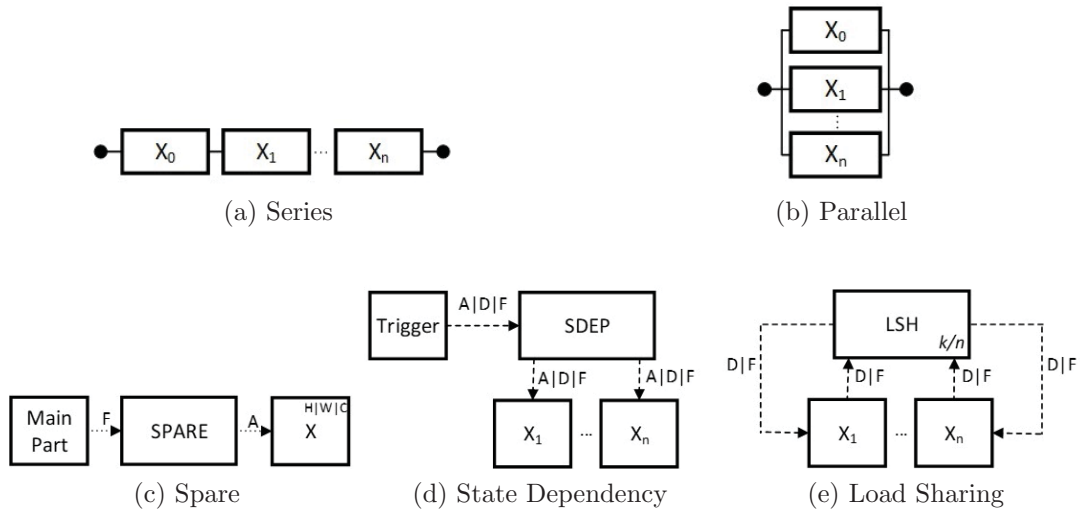


Figure 1.2: Dynamic DRBD Constructs

when components fail in such a manner that leads to the disconnection of all the paths between the input and the output. Additional constructs are used to model the dynamic dependencies among system blocks.

The main dynamic DRBD structures and constructs are shown in Figure 1.2 [42]. In DRBDs, the blocks can be connected in series, parallel or deeper nested structures. For the series structure, shown in Figure 1.2(a), all system blocks should function properly in order to maintain a successful behavior. On the other hand, at least one of the blocks of the parallel structure in Figure 1.2(b) must work to have a successful system behavior. These two structures can be connected in a hierarchical manner to model complex systems. The *spare* construct (Figure 1.2(c)) is used to model spare parts in systems, similar to the DFT spare gate. The *state dependencies* (Figure 1.2(d)) are used to model the effect of activation(A)/deactivation(D)/failure(F) among system components. In Figure 1.2(d), the A/D/F of the trigger will cause the state dependency controller (SDEP) to signal the A/D/F of components $X_1 \dots X_n$. Finally, the *load sharing* (LSH) construct is used to model the effect of sharing the

same load among several components on the failure effect of the overall system. For example, the LSH in Figure 1.2(e) models a load that is shared among n components. It is required that at least k out of these n components to be working in order for the functionality of the system to be successful. Therefore, the D/F of some of these components may cause the D/F of the rest of the components. The last two constructs enable modeling more realistic scenarios in system reliability that include the effect of A/D of one component on the rest of the components. This behavior cannot be captured using DFTs [43] as they can only capture the failure effect of one system component on the rest of the components without considering the A/D effect.

Due to the dynamic nature of DRBDs, they can be analyzed by converting them into a state-space model, i.e, a Markov chain. Then, the resultant Markov chain can be analyzed using one the of the traditional techniques, including analytical methods or simulation, such as Monte Carlo simulation. There exist some tools that provide the DRBD analysis, such as Möbius [36], isograph [37] and BlockSim [35], which provide a graphical user interface to model DRBDs and conduct the analysis either analytically or using discrete event simulation. As mentioned previously, complex systems can generate Markov chains with a large number of states, which hinders the analysis process. Decomposition can be applied to divide the DRBD into a dynamic part that can be solved using Markov chains and a static part that can be analyzed using static RBD analysis techniques [44]. This decomposition would reduce the state space, but such simulation based analysis cannot provide accurate and complete results. The formal semantics of DRBDs have been introduced in [45] using the Object-Z formalism [46]. Then, this DRBD is converted into a Colored Petri net (CPN) [47], where it can be analyzed using existing Petri net tools. An algorithm to automatically convert a DRBD into a CPN is also proposed in [48]. However,

since the given DRBD is converted into a CPN, only state-based properties can be analyzed. In addition, generic expressions of reliability cannot be obtained, which represents our target in this thesis. HOL theorem proving has been used for the analysis of traditional RBDs [22]. However, there is no support for DRBD analysis using a HOL theorem prover that can handle the analysis of real-world systems that exhibit dynamic behavior.

In system engineering, it is important to be able to analyze DRBDs qualitatively in order to identify the sources of system vulnerability, and quantitatively in order to evaluate the system reliability. However, to the best of our knowledge, so far there exists no algebra that mathematically models a given DRBD and enables expressing its function based on basic components like the DFT algebra [30]. Using such algebra in the reliability analysis will result in simpler and fewer proof steps than the DFT-based algebraic analysis [30], since the probabilistic PIE will not be invoked. In this PhD thesis, we propose a new algebraic approach for DRBD analysis that allows a DRBD expression to be used for both qualitative and quantitative analyses. We introduce new operators to mathematically model the dynamic behavior in DRBD structures and constructs. In particular, we use these operators to model a DRBD spare construct as well as traditional series, parallel, series-parallel and parallel-series structures. Moreover, we provide simplification theorems that allow the structure of a given DRBD to be reduced. This DRBD structure can be then analyzed to obtain a generic expression of the system reliability. The reliability expressions obtained using this approach are generic and independent of the distribution and density functions that represent the system components. Although basic operators, such as OR and AND, were introduced in [44], they are only useful to model parallel and series constructs of dependent components. Moreover, there is no general mathematical

expression that would allow reasoning about the behavior of DRBDs. In addition, the DRBD constructs of [44] are quite complex, which complicates modeling large systems. Therefore, in this thesis, we use the constructs proposed in [45] as they are much simpler, which facilitates defining the new algebra to model various new DRBD constructs. Leveraging upon the expressive nature of HOL, we formally verify the soundness of the proposed DRBD algebra using HOL theorem proving. Although the formalization development can be conducted using many theorem provers, we choose the HOL4 theorem prover, as our existing formalization of DFT algebra can be useful since our proposed DRBD algebra is compatible with the DFT's.

It is worth mentioning that a given DFT can be converted into its equivalent DRBD and vice-versa, which allows reasoning about both the success and failure of a given system using one model. This requires the conversion of each DFT gate into its equivalent DRBD construct or structure. For example, the DFT spare gate can be modeled using the DRBD spare construct [43].

1.4 Framework for Formal Dynamic Dependability Analysis

As mentioned earlier, there exist many techniques that can be invoked to analyze the dynamic dependability of systems. However, none of them provides an accurate, scalable and expressive framework for dynamic dependability modeling and analysis, which represents important features in analyzing the dynamic failure behavior of systems, specially safety critical ones. Therefore, the objective of this thesis is to provide a framework for the dynamic dependability analysis of systems modeled as DFTs and DRBDs. The proposed framework is depicted in Figure 1.3.

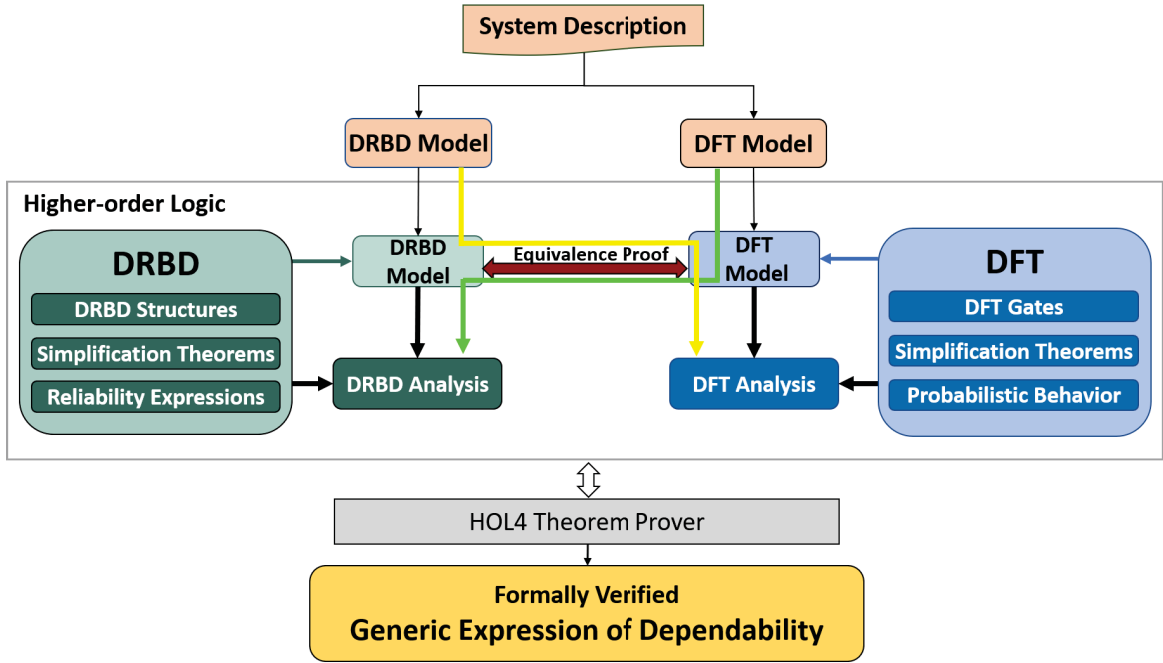


Figure 1.3: Overview of the Proposed Framework

This framework provides verified generic expressions of dependability using HOL4 theorem prover of DFTs and DRBDs. The analysis starts by having a system description, which is assumed to be correct, with some dependability requirements, such as a certain expression of reliability. This system can be modeled either using a DFT or DRBD model according to its description. We create formal DFT or DRBD models utilizing our library of formalized DFT gates and DRBD constructs. This library also includes the DFT and DRBD simplification theorems and verified probabilistic behavior as well as the reliability expressions. The formal DFT and DRBD models can then be analyzed qualitatively or quantitatively. In the former, the sources of vulnerabilities of the system are verified by identifying the cut sets and cut sequences. In the latter, we prove generic failure and reliability expressions

of DFT and DRBD based systems, respectively. It is worth mentioning that unlike model checking approaches, the formally verified generic expressions of DFT and DRBD are independent of the probability distributions of the system components. HOL4 was used in the development of the formalization of static dependability models, i.e., FTs [21] and RBDs [22]. Therefore, in this thesis, we choose to follow the same path and use HOL4 as this would facilitate using some of the developed work, such as the probabilistic PIE theory [23]. In addition, we build our theories utilizing some of the existing theories in HOL4 such as the theories on measure, Lebesgue integral [49] and probability [28]. This proposed framework allows conducting the dynamic dependability analysis of many real-world systems to provide generic expressions. It is important to note that the DFT part of the proposed framework is primarily based on the formalization of the algebraic approach presented in [30]. However, a distinguishing feature of our formalization is that it allows us to conduct computer based proofs of the probability of failure expressions for DFT gates within the sound environment of a theorem prover software. These proofs are either unavailable in [30], or we are able to conduct them in a simpler manner. In addition, we explicitly define DFT events that are used to provide the set of time to perform the probabilistic analysis. Moreover, as we are providing the formalization in a theorem prover, datatypes should be carefully handled to capture both the behavior of DFT gates and the probability of their failure. These details are not provided in [30], which signifies the importance of the proposed methodology. Since this framework integrates both DFT and DRBD algebras, it provides the capability of formally converting one dependability model to another based on the equivalence proof of both algebras. This means that the DRBD model can be converted to a DFT to model the failure instead of the success, then this model is analyzed using the DFT algebra. Similarly, the DFT model can be analyzed

by converting it to its counterpart DRBD model. It is worth noting that based on the system description, the analysis can be conducted at different levels of abstraction in a hierarchical (modular) manner.

As an application of the formalized DFT algebra, we formally verify within HOL the DFT rewrite rules that are used in other DFT analysis tool, such as STORM. This demonstrates the applicability and generality of our DFT formalization. Furthermore, we apply our framework in the formal DFT analysis of two safety-critical systems: a cardiac assist system (CAS), and a drive-by-wire (DBW) system. Similarly, we perform the analysis of the DRBD of the DBW. Finally, we illustrate the usefulness of the entire framework by conducting the formal dependability analysis of shuffle-exchange networks, which are widely used in multiprocessor systems. We use both DFT and DRBD models to perform the analysis and utilize their equivalence to show the possibility of conducting the analysis in both directions.

1.5 Thesis Contributions

The main contribution of this thesis is to develop a framework for the formal dynamic dependability analysis of DFT and DRBD models using HOL theorem proving. This framework represents an alternative approach to other less rigorous ones, such as simulation and paper-and-pencil. To accomplish this objective, we develop libraries for each dependability model. Each library has the formalized mathematical foundations for each algebra, including the DFT gates definitions and the DRBD structures. In addition, these libraries include the formally verified probabilistic behavior of each model besides the equivalence proof of both algebras. Below, we provide the list of contributions of this work along with the related publications available in the Biography section at the end of the thesis:

- Formalization of a DFT algebra in HOL, which includes operators and gates definitions in addition to the verification of the simplification theorems. The gates and operators are defined based on the time-of-failure of the inputs and the outputs, which are modeled as random variables of failure that return extended-real numbers. We verify the simplification theorems based on these definitions and using the properties of extended-real numbers [Bio-Jr2, Bio-Tr3].
- Formal qualitative analysis of DFTs in HOL. We use the simplification theorems to reduce the structure function of a given DFT. Then, using the HOL4 theorem prover, we formally verify the cut sets and cut sequences. Moreover, we formally conduct the qualitative analysis of the DBW and CAS systems [Bio-Jr1].
- Formal verification of the probabilistic failure behavior of the DFT gates using HOL theorem proving, which allows conducting the formal quantitative analysis of a given DFT in HOL. Building this theory requires using the properties of the Lebesgue integral and the probability theory. We perform the probabilistic analysis of the DBW and CAS systems to provide generic expressions of probability of failure that are independent of the failure distributions of the input events [Bio-Jr2, Bio-Tr4].
- Formal verification of DFT rewrite rules that are useful in reducing a given DFT. These rewrite rules are used in automated DFT analysis tools, such as the STORM model checker, which represents the first step towards the formal verification of these tools. We extend the definitions of DFT gates to n-ary gates, which are required to formalize these rewrite rules [Bio-Cf2].
- Development of a novel DRBD algebra that allows expressing the structure of a DRBD with spare constructs. We introduce DRBD operators and simplification

theorems, similar to the DFT algebra, to enable reducing a given DRBD [Bio-Cf1, Bio-Tr3].

- The formalization of the new DRBD algebra besides modeling the spare construct and several DRBD structures, such as series and parallel structures. We formally verify the reliability expressions of the DRBD structures and the spare construct to allow conducting the reliability analysis of a given DRBD in a theorem prover [Bio-Cf1, Bio-Tr3].
- Formal verification of the equivalence of DFT and DRBD that allows the bidirectional path between both models. This enables formally analyzing the success and failure behaviors of systems modeled using either model [Bio-Tr2].
- Formal verification of the terminal, broadcast and network reliability analysis of several versions of generic shuffle-exchange networks, which are widely used in the interconnection of multiprocessor systems. We perform the formal analysis using both DFT and DRBD dependability models and illustrate the utilization of their equivalence [Bio-Tr1].

1.6 Thesis Organization

The rest of the thesis is structured as follows: In Chapter 2, we provide some preliminaries that are required for the understanding of the rest of this thesis. This includes an overview of theorem proving and the HOL4 theorem prover. In addition, we provide a summary of the required theories that are needed to develop the proposed framework.

In Chapter 3, we present the formalization of DFT operators and gates based on the algebraic approach. Then, we formally verify a set of simplification theorems

that are used to conduct the formal qualitative analysis.

In Chapter 4, we provide the verification details of the probabilistic failure analysis of each DFT gate. Furthermore, we present the formal quantitative analysis of the CAS and DBW systems in the form of generic expressions of probability of failure.

As an application to our DFT formalization, in Chapter 5, we provide the formalization of the rewrite rules of DFTs, which enable the reduction of DFT models in automated DFT analysis tools. Moreover, we present the HOL formalization of n-ary gates that are required to model the rewrite rules.

In Chapter 6, we introduce the proposed DRBD algebra including DRBD operators, spare construct and structures. Furthermore, we provide the simplification theorems that are used to simplify the structure of a given DRBD. We present the formalization details of the DRBD algebra and the verified reliability expressions. Finally, we verify the equivalence of the DFT and DRBD algebras, which enables the analysis of a given system using both models.

We use the proposed framework in Chapter 7 to provide the dynamic dependability analysis of MINs of multiprocessor systems, particularly the terminal, broadcast and network reliability of shuffle-exchange networks. We provide DFT and DRBD models of these systems and verify generic expressions of probability of failure and reliability for different system scenarios. Finally, we conclude the thesis in Chapter 8 and provide some future work directions.

Chapter 2

Preliminaries

In this chapter, we provide some preliminaries that are required for the understanding of the rest of the thesis. We describe the HOL4 theorem prover and its probability and Lebesgue integral theories. In addition, we introduce some definitions in HOL4 that are required for the formalization of DFTs and DRBDs.

2.1 HOL4 Theorem Proving

Theorem proving is one of the formal methods techniques that uses a computerized program, i.e., a theorem prover, to carry out mathematical proofs of theorems based on deductive reasoning. The level of expressiveness of these theorems depends on the type of logic used, like first-order logic and higher-order logic (HOL). There are several HOL theorem provers that are available, such as HOL4 [27], Isabelle [50] and Coq [51], which vary in the availability of the supported libraries.

HOL4 is an interactive theorem prover, which is capable of verifying a wide range of hardware and software systems as well as mathematical expressions constructed

in HOL. Being an interactive tool, HOL4 requires the guidance of the verification engineer to complete the verification process. In order to verify certain properties of a system, a mathematical model for this system should be created first, then based on this model, HOL4 can be used to verify several system properties in the form of theorems. This makes HOL4 an expressive platform for the verification of any system that can be described mathematically. The main characteristic of HOL theorem proving is its soundness, i.e., only valid proof goals can be proved. The core of HOL4 consists only of four axioms and eight inference rules. Soundness is assured as any new theorem should be verified based on these axioms and rules, or based on previously proven theorems. In addition, no approximation is involved in the models, as their behavior, such as the failure in the case of DFTs, is captured in mathematical terms. These features make HOL4 suitable for carrying out the DFT based analysis of safety-critical systems that require sound verification results. The term *formalization* means to mathematically model the behavior of a system in an appropriate logic. A *proof goal* consists of a list of assumptions of type Boolean and a conclusion. For example, “ $\forall (x:\text{real}). 0 < x \Rightarrow 0 < x^{-1}$ ” is a proof goal, which can be formally verified as a theorem in HOL4. A *theory* in HOL4 is a collection of definitions, constants and theorems that can be included in the working environment to be used in verifying other proof goals. Table 2.1, lists some of the used symbols in HOL.

2.2 Probability Theory

The probability theory is formalized based on the measure theory in HOL4 [28]. A measurable space is represented as a pair $(\mathcal{X}, \mathcal{A})$, where \mathcal{X} represents a space and \mathcal{A} a set of measurable sets. The functions `space` and `subsets` are defined in HOL to return the \mathcal{X} and \mathcal{A} , respectively, of a measurable space $(\mathcal{X}, \mathcal{A})$. A measure is

Table 2.1: Some HOL Symbols

HOL Symbol	Meaning
\wedge	Logical <i>and</i>
\vee	Logical <i>or</i>
$::$	Appends a new element to a list
$++$	Joins two lists
HD L	Head of a list L
TL L	Tail of list L
(a, b)	A pair with elements a and b
FST	First element of a pair
SND	Second element of a pair
$\lambda x. f x$	Lambda abstracted function f
$\{x \mid P x\}$	Set of elements x that satisfy $P(x)$
FINITE s	set s is finite
A DIFF B	$A - B$
e INSERT s	insert element e in set s
s DELETE e	A set that has all elements of s except e
DISJOINT A B	sets A and B are disjoint

generally a function that designates a certain number to a set, which represents the size of this set [17]. It is defined as the triplet $(\mathcal{X}, \mathcal{A}, \mu)$, where \mathcal{X} represents the space, \mathcal{A} represents the measurable sets and finally μ represents the measure. Three functions, `m_space`, `measurable_sets` and `measure`, are defined in HOL to return the space (\mathcal{X}) , measurable sets (\mathcal{A}) and measure (μ) of a measure space, respectively [52]. A probability space is defined as a measure space, with the added condition that the probability measure for the entire space is equal to 1.

Random variables are formalized as measurable functions that map events from the probability space to some other σ - algebra space s . For a collection of subsets of a space (\mathcal{X}) , (\mathcal{A}) is a σ - algebra on (\mathcal{X}) if it contains the empty set, and is closed under countable unions and complements within the space (\mathcal{X}) [17]. Random variables are defined in HOL4 as in [17]:

Definition 2.1.

$$\vdash \forall X p s. \text{ random_variable } X p s \Leftrightarrow$$

$$\text{prob_space } p \wedge X \in \text{measurable } (p_space \ p, \text{ events } p) \ s$$

where `prob_space p` ensures that `p` is a probability space with `p_space` as its space and `events` as its measurable sets. `X ∈ measurable (p_space p, events p) s` ensures that `X` belongs to the set of measurable functions from the probability space `p` to the σ -algebra space `s` [52]. Measurable spaces `s` and `(p_space p, events p)` are ensured to be σ -algebra spaces using the `measurable` function.

The probability distribution of a random variable X represents the probability that the random variable X belongs to a set A . This is equivalent to finding the probability of the event $\{X \in A\}$, which can also be represented using the preimage as $X^{-1}(A)$. The probability distribution is defined in HOL4 as in [17]:

Definition 2.2.

$$\vdash \forall p X. \text{ distribution } p X = (\lambda s. \text{ prob } p (\text{PREIMAGE } X \ s \cap p_space \ p))$$

where `s` is a set of elements of the space that the random variable X maps to. For a random variable that maps the probability space (p) into another measurable space, the push forward measure is a measure that uses the space and subsets of the measurable space as its space and measurable sets and uses the distribution of the random variable as its measure part [18]. In general, the push forward measure for any measurable function X from measure M to measure N can be expressed as:

Definition 2.3.

$$\vdash \forall M N f. \text{ distr } M N f =$$

$$(\text{m_space } N, \text{ measurable_sets } N,$$

$$\lambda A. \text{ measure } M (\text{PREIMAGE } f \ A \cap \text{m_space } M))$$

A density measure is used to define a density function, f , over the measure space M as in [53]:

Definition 2.4.

$\vdash \forall M f. \text{ density } M f =$
 $(\text{m_space } M, \text{ measurable_sets } M,$
 $\lambda A. \text{ pos_fn_integral } M (\lambda x. f x * \text{ indicator_fn } A x))$

where `pos_fn_integral` represents the Lebesgue integral of positive functions as will be described in the following section.

The cumulative distribution function (CDF) of a random variable X is usually used when we are interested in finding the probability that the random variable is less than or equal to a certain value. It is formally defined for real values as in [23]:

Definition 2.5.

$\vdash \forall p X t. \text{ CDF } p X t = \text{ distribution } p X \{y \mid y \leq (t:\text{real})\}$

It is worth mentioning that the CDF can be defined for extended-real (`extreal`) random variables as well, where `extreal` is a HOL data-type containing the real numbers plus $\pm\infty$. However, in our formalization we will use the CDF of real random variables, as it is required to integrate their density functions over the real line.

When dealing with multiple random variables, the probabilistic *Principle of Inclusion and Exclusion* (PIE) provides a very interesting relationship between the probability of the union of different events. It can be expressed as:

$$Pr\left(\bigcup_{i=1}^n A_i\right) = \sum_{t \neq \{\}, t \subseteq \{1, 2, \dots, m\}} (-1)^{|t|+1} Pr\left(\bigcap_{j \in t} A_j\right) \quad (2.1)$$

It has been formally verified in HOL4 as follows [23]:

Theorem 2.1.
 $\vdash \forall p L.$

$$\text{prob_space } p \wedge (\forall x. \text{MEM } x L \Rightarrow x \in \text{events } p) \Rightarrow$$

$$(\text{prob } p (\text{union_list } L = \text{sum_set } \{t \mid t \subseteq \text{set } L \wedge t \neq \{\}\})$$

$$(\lambda t. -1 \text{ pow } (\text{CARD } t+1) * \text{prob } p (\text{BIGINTER } t))$$

where L is the list of events that we are interested in expressing the probability of their union.

In order to be able to handle multiple random variables, a pair measure (often called binary product measure) is required to be able to model joint distribution measures. This pair measure can be also used in a nested way to model the joint distribution measure of multiple random variables. The pair measure is defined as the product of two measures. It was initially formalized in Isabelle/HOL [18] and was then ported to HOL4 [53]. The space and the measurable sets of this pair measure are generated using the Cartesian product of the spaces and the measurable sets of the participating measures, while the measure part is defined using the Lebesgue integral.

Since there are real and extended-real data-types in HOL4, there exist two Borel spaces, one over the real line (`bore1`) [54] and the second over the extended-real line (`Bore1`) [49]. The Lebesgue-Borel measure is required to integrate over the real line. In particular, we need the Lebesgue-Borel measure in this work to integrate the density functions of the random variables over the real line. The Lebesgue-Borel measure is a measure defined over the real line, which uses the real line as its space and the Borel sets as its measurable sets. The Lebesgue-Borel measure is defined in HOL4 as `lbore1`, which uses the real borel sigma algebra (`bore1`) generated by the open sets of the real line as well as the Lebesgue measure [54].

The independence of random variables is an important property when dealing with multiple random variables. In general, for any two random variables X and Y , the probability of the intersection of their events is equal to the multiplication of the probability of the individual events. The independence of random variables is defined as `indep_vars` [53]:

Definition 2.6.

$$\begin{aligned} \vdash \text{indep_vars } p \ M \ X \ ii = & \\ (\forall i. \ i \in ii \Rightarrow & \\ \text{random_variable } (X \ i) \ p & \\ (\text{m_space } (M \ i), \text{measurable_sets } (M \ i))) \wedge & \\ \text{indep_sets } p & \\ (\lambda i. \ \{\text{PREIMAGE } f \ A \cap \text{p_space } p \mid & \\ (f = X \ i) \wedge A \in \text{measurable_sets } (M \ i)\}) \ ii & \end{aligned}$$

where p is the probability space and M is the measure space that the random variable X maps to. In this case, M and X are indexed by a number from the set of numbers ii , which gives the possibility of defining the independence for multiple random variables that map from the probability space to different spaces. The function `indep_vars` defines the independence by first ensuring that the group of input functions X are random variables and that their event sets are independent using `indep_sets`. Using `indep_sets`, the probability of the intersection of any sub-group of events of the random variables is equal to the multiplication of the probability of the individual events.

Using `indep_vars`, the independence of two random variables is defined as in [53]:

Definition 2.7.

$$\begin{aligned} \vdash \text{indep_var } p \ M_x \ X \ M_y \ Y = \\ \text{indep_vars } p \ (\lambda i. \ \text{if } i = 0 \ \text{then } M_x \ \text{else } M_y) \\ (\lambda i. \ \text{if } i = 0 \ \text{then } X \ \text{else } Y) \ \{x \mid (x = 0) \vee (x = 1)\} \end{aligned}$$

We define several functions that facilitate handling our formalization. The first function is `measurable_CDF`, which is defined as:

Definition 2.8.

$$\begin{aligned} \vdash \forall p \ X. \ \text{measurable_CDF } p \ X = \\ (\lambda x. \ \text{CDF } p \ X \ x) \in \text{measurable borel Borel} \end{aligned}$$

This function ensures that the CDF of random variable X is measurable from the `borel` space to the `Borel` space. In other words, it ensures that the CDF is measurable from the real line to the extended-real line. This implies that the domain for this CDF is the real line and the range is the extended-real line.

We define another function, `cont_CDF`, which ensures that the CDF is continuous. It is formally defined as:

Definition 2.9.

$$\vdash \forall p \ X. \ \text{cont_CDF } p \ X = \forall z. \ (\lambda x. \ \text{real } (\text{CDF } p \ X \ x)) \ \text{cont1 } z$$

where the function `real` typecasts the value of CDF from extended-real to real datatype, and `cont1` ascertains that the function is continuous over all values in its domain. It is worth mentioning that X is a real valued random variable. However, the CDF returns extended-real. As the continuity of functions is defined in HOL4 for real valued functions, it is required to typecast the value of the CDF from extended-real to real. In addition, since the values of the CDF range from 0 to 1, as it represents a probability, this function is the same in both cases but with different datatypes. Therefore, if

the function is continuous in the extended-real, then it is continuous using the real data-type. Furthermore, later we will use extended-real random variables, therefore, it is required to typecast their values using the `real` function.

Next, we define a function, `rv_gt0_ninfinite`, to ensure that the input random variables of a DFT can only have the range $[0, +\infty)$:

Definition 2.10.

```

⊢ (rv_gt0_ninfinite [] = T) ∧
  (rv_gt0_ninfinite (h::t) = (∀ s. 0 ≤ h s ∧ h s ≠ PosInf) ∧
  (rv_gt0_ninfinite t))

```

Finally, we define a function, `den_gt0_ninfinite` to ensure the proper values for the marginal, joint and conditional density functions:

Definition 2.11.

```

⊢ ∀ f_xy f_y f_cond.
  den_gt0_ninfinite f_xy f_y f_cond ⇔
  ∀ x y.
    0 ≤ f_xy (x,y) ∧ 0 < f_y y ∧ f_y y ≠ PosInf ∧ 0 ≤ f_cond y x

```

where `f_xy` is the joint density function, `f_y` is the marginal density function, and finally `f_cond` is the conditional density function of X given Y. This function can be used to assign the mentioned conditions to other functions and not necessarily only the density functions.

2.3 Lebesgue Integral

The Lebesgue integral is defined in HOL4 using positive simple functions, which are measurable functions defined as a linear combinations of indicator functions of measurable sets representing a partition of the space X [28]. A positive simple function, g , can be represented using the triplet (s, a, x) as [28]:

$$\forall t \in X, g(t) = \sum_{i \in s} x_i \mathbf{1}_{a_i}(t), \quad x_i \geq 0 \quad (2.2)$$

where s is a finite set of partition tags, x_i is a sequence of positive **extreal** numbers, a_i is a sequence of measurable sets and $\mathbf{1}_{a_i}$ is the indicator function of measurable set a_i and is defined as in [28]:

Definition 2.12.

$\vdash \forall A. \text{indicator_fn } A = (\lambda x. \text{ if } x \in A \text{ then } 1 \text{ else } 0)$

The Lebesgue integral is first defined for positive simple functions and then extended for positive functions for measure M as `pos_fn_integral M ($\lambda x. f x$)` [28], where f is the positive function that we are integrating. In this work, we are integrating the density functions over the real line. Therefore, we will use the Lebesgue-Borel measure with the Lebesgue integral. The Lebesgue-Borel (`lborel`) measure is a measure defined over the real line. As with any measure, `lborel` should have a space and measurable sets. For `lborel`, the real line represents its space and the borel sets represent the `lborel` measurable sets [54]. For example, the Lebesgue integral of the positive function f from 0 to t ($\int_0^t f(x) dx$) can be expressed formally as: `pos_fn_integral lborel ($\lambda x. f x * \text{indicator_fn } \{x' \mid 0 \leq x' \wedge x' \leq t\} x$)`, where the indicator function is used here to set the interval that we are integrating over. Throughout

this thesis, we will use a mix of standard and HOL math notation to facilitate the readability of the text.

It is usually required that the probability of an event for a random variable to be expressed using the integration of the random variable's distribution. This is verified in HOL4 as [17]:

Theorem 2.2.

$\vdash \forall X p s A.$

$$\begin{aligned} & \text{random_variable } X p s \wedge A \in \text{subsets } s \Rightarrow \\ & (\text{distribution } p X A = \\ & \quad \text{integral (space } s, \text{ subsets } s, \text{ distribution } p X)(\text{indicator_fn } A)) \end{aligned}$$

In the above theorem, X can be a continuous or a discrete random variable. However, in our DFT formalization, we are only interested in continuous random variables as they represent the time of failure of system components.

Chapter 3

Formal Qualitative Analysis of Dynamic Fault Trees

In this chapter, we present our formalization of DFT gates and operators based on the algebraic approach, which are required in conducting the formal DFT qualitative analysis. We apply this formalization to qualitatively analyze two case studies to obtain formally reduced cut sets and cut sequences.

3.1 Methodology

The proposed methodology for conducting the formal qualitative analysis of DFTs in HOL is depicted in Figure 3.1. The analysis starts by a system description that can be used to build a DFT model with some dependability requirements, which are related to the qualitative assessments of DFTs. A formal DFT model of the given system is developed in HOL, which requires the formal definitions of DFT gates. These gates include the AND, OR, FDEP, PAND and spare gates with shared

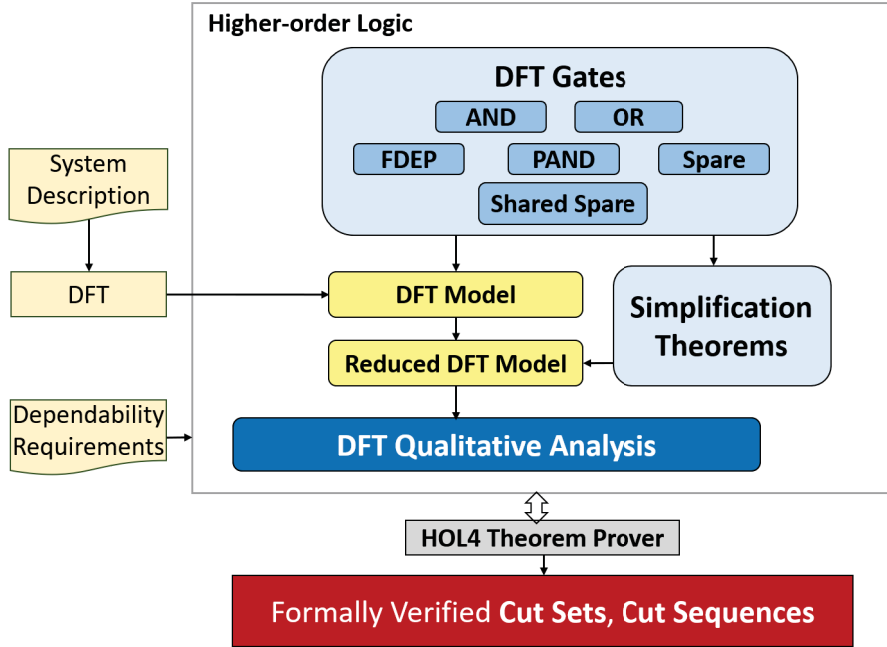


Figure 3.1: Formal DFT Qualitative Analysis Methodology

spares. Then, the structure function of the DFT’s top event has to be reduced and we propose to formally verify this reduction based on a library of generic formally verified simplification theorems. This ensures that the reduced formal DFT model corresponds to the original DFT model. This verified reduced structure function is then used in the qualitative analysis to produce a reduced form of the cut sets and cut sequences that satisfy the requirements. The cut sets can be defined as a group of sets, where each set has the inputs that their combined failure leads to the occurrence of the top event. The cut sequences, on the other hand, is a group of lists, where each list has a certain sequence of inputs that their failure in this particular sequence leads to the failure of the top event.

It is worth mentioning again that this methodology formalizes the mathematical foundations of the algebraic approach presented in [30], where the DFT events are treated based on their time of occurrence (failure of corresponding components d). This allows us to conduct the qualitative analysis using the sound core of HOL theorem

prover. Therefore, it is required first to have a library of formalized DFT operators, gates and simplification theorems to be used in the formal DFT qualitative analysis. In [30], it is assumed that system components are non-repairable, i.e., the components are not repaired after failure.

3.2 Identity Elements and Temporal Operators

Similar to ordinary Boolean algebra, the DFT algebraic approach defines identity elements that are important in the simplification process of the DFT [30]. The DFT identity elements are: the *ALWAYS* element representing an event that always occurs from time 0 (constant failed element $\text{CONST}(\top)$), i.e., $\text{ALWAYS} = 0$, and the *NEVER* element, which describes an event that never occurs (constant fail-safe element $\text{CONST}(\perp)$), i.e., $\text{NEVER} = +\infty$. We formally define these elements as:

Definition 3.1. *Always Element*

$$\vdash \text{ALWAYS} = (\lambda \mathbf{s}. \quad (0:\text{extreal}))$$

Definition 3.2. *Never Element*

$$\vdash \text{NEVER} = (\lambda \mathbf{s}. \quad \text{PosInf})$$

where PosInf represents $+\infty$ in HOL4. We define the time of failure of the events as lambda abstracted functions that accept an arbitrary data-type that represents an element from the probability space and return the time. So that they can be later treated as random variables. For example, the time of failure of a component is a random variable X and can be expressed in lambda abstraction form as $(\lambda \mathbf{s}. \quad X \ \mathbf{s})$.

Temporal operators are also required to model the DFT gates in the algebraic approach [30]. These operators are: *Before* (\triangleleft), *Simultaneous* (Δ) and *Inclusive Before* (\trianglelefteq). Each one of these operators accepts two inputs, which can be subtrees

or basic events that represent faults of system components. The time of occurrence of the output event of each operator equals the time of occurrence of the first input event if a certain condition is satisfied, otherwise the output can never occur. The output event of the *Before* operator occurs (fails) when the first input event (left) is less than the time of occurrence of the second input (right), otherwise it can never occur. It is mathematically expressed as [30]:

$$d(X \triangleleft Y) = \begin{cases} d(X), & d(X) < d(Y) \\ +\infty, & d(X) \geq d(Y) \end{cases} \quad (3.1)$$

We formally define the *Before* operator in HOL as lambda abstracted function:

Definition 3.3. *Before Operator*

$\vdash \forall X Y.$

`D_BEFORE X Y = ($\lambda s.$ if X s < Y s then X s else PosInf)`

where X and Y represent the time of occurrence of events X and Y, respectively.

The time of failure of the *Simultaneous* operator is equal to the time of occurrence of one of the events, only if their time of failure is equal, otherwise, it can never occur. It can be expressed as [30]:

$$d(X \Delta Y) = \begin{cases} d(X), & d(X) = d(Y) \\ +\infty, & d(X) \neq d(Y) \end{cases} \quad (3.2)$$

It is worth mentioning that if the inputs of the *Simultaneous* operator are basic events with continuous failure distributions, then the output of this operator can never fail [30]. This is because the time of failure is continuous, and the possibility that two system components fail at the same time is negligible. As a consequence, it is assumed in the algebraic approach that any two *different* basic events can never fail

at the same time. This can be expressed for basic failure events of the inputs of the given DFT as in [30]:

$$d(X\Delta Y) = NEVER \quad (3.3)$$

We formally define the *Simultaneous* operator as:

Definition 3.4. *Simultaneous Operator*

$\vdash \forall X Y.$

`D_SIMULT X Y = ($\lambda s.$ if X s = Y s then X s else PosInf)`

Finally, the *Inclusive Before* combines the behavior of both the *Simultaneous* and the *Before* operators, i.e., if the first input event (left) occurs before or at the same time as the second input event (right), then the output event occurs with a time equals to the time of occurrence of the first input event. This is expressed as [30]:

$$d(X \leq Y) = \begin{cases} d(X), & d(X) \leq d(Y) \\ +\infty, & d(X) > d(Y) \end{cases} \quad (3.4)$$

We formally define this operator in HOL as:

Definition 3.5. *Inclusive Before Operator*

$\vdash \forall X Y.$

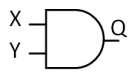
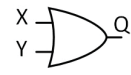
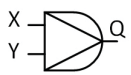
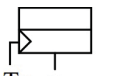
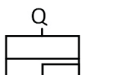
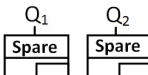
`D_INCLUSIVE_BEFORE X Y = ($\lambda s.$ if X s \leq Y s then X s else PosInf)`

3.3 Formalization of FT Gates

The mathematical expressions of the FT gates; static and dynamic, are listed in Table 3.1. In [30], the DFT gates are defined using the temporal operators and the definitions of the AND and OR. However, in this work, we present mathematical

definitions for the gates based on the time of failure of the output event, then we verify the equivalence of these definitions with the definitions presented in [30].

Table 3.1: DFT Gates Mathematical Expressions

Gates	Mathematical Expressions
 AND	$d(X \cdot Y) = \max(d(X), d(Y))$
 OR	$d(X + Y) = \min(d(X), d(Y))$
 PAND	$d(Q_{PAND}) = \begin{cases} d(Y), & d(X) \leq d(Y) \\ +\infty, & d(X) > d(Y) \end{cases}$
 FDEP	$d(X_{Tr}) = \min(d(X), d(Tr))$
 Spare	$d(Q_{CSP}) = \begin{cases} d(X), & d(Y) < d(X) \\ +\infty, & d(Y) \geq d(X) \end{cases}$
	$d(Q_{HSP}) = \max(d(Y), d(X))$
	$d(Q_{WSP}) = d(Y \cdot (X_d \triangleleft Y) + X_a \cdot (Y \triangleleft X_a) + Y \Delta X_a + Y \Delta X_d)$
 Shared Spare	$d(Q_1) = d(X \cdot (Z_d \triangleleft X) + Z_a \cdot (X \triangleleft Z_a) + X \cdot (Y \triangleleft X))$

3.3.1 AND and OR Gates

The AND (\cdot) and OR ($+$) are considered as operators and as static gates as well. They can be modeled based on the time of occurrence of their output events. For the AND gate, the output occurs when both of its input events occur and the time of

occurrence of the output is modeled as the maximum time of occurrence of both input events [30]. For the OR gate, the output occurs once one of its input events occurs. Therefore, we formalize it as the minimum time of occurrence of the inputs [30]. We formally define the AND and OR in HOL as:

Definition 3.6. *AND Gate/Operator*

$$\vdash \forall X Y. D_AND X Y = (\lambda s. \max (X s) (Y s))$$

Definition 3.7. *OR Gate/Operator*

$$\vdash \forall X Y. D_OR X Y = (\lambda s. \min (X s) (Y s))$$

where `max` and `min` are the HOL4 functions that return the maximum and the minimum of their input arguments, respectively. It is important to notice that we define the AND and OR gates as lambda abstracted functions that accept two inputs that are also functions. This enables defining the inputs later as random variables to represent the time of failure function of system components. This also applies to the formal definitions of the rest of DFT gates.

3.3.2 Priority AND Gate (PAND)

The PAND gate, shown in Table 3.1, captures the sequence of occurrence (failure) of its inputs. The output event of this gate occurs if all input events occur in a certain sequence (conventionally from left to right). In Table 3.1, we provide the mathematical definition of the PAND gate. We formalize this expression in HOL as:

Definition 3.8. *PAND Gate*

$$\vdash \forall X Y. PAND X Y = (\lambda s. \text{if } X s \leq Y s \text{ then } Y s \text{ else PosInf})$$

The behavior of the PAND gate can also be represented using the temporal operators as defined in [30]:

$$Q = Y \cdot (X \leq Y) \quad (3.5)$$

We verify the above relationship in HOL4 as follows:

Theorem 3.1.

$$\vdash \forall X Y. \text{ PAND } X Y = \text{ D_AND } Y (\text{ D_INCLUSIVE_BEFORE } X Y)$$

This result ascertains that the behavior of PAND gate is correctly captured in our formal definition.

3.3.3 Functional DEpendency Gate (FDEP)

The FDEP is used to model the dependencies in the failure behavior between the system components. In other words, it is used when the failure of one component triggers the failure of another. For the FDEP gate, shown in Table 3.1, event X can occur if it is triggered by the failure of Tr or if it occurs by itself. As a result, the occurrence time of X_{Tr} (triggered X) equals the minimum time of occurrence of Tr and X . From the FDEP definition, we can notice that its behavior is equivalent to the behavior of the OR gate, which is similar to what is proposed in [55, 56] We formally define the FDEP as:

Definition 3.9. *FDEP Gate*

$$\vdash \forall X Tr. \text{ FDEP } X Tr = (\lambda s. \min (X s) (Tr s))$$

In [30], the behavior of the FDEP gate is represented using the temporal operators as:

$$X_{Tr} = Tr + (X \leq Tr) \quad (3.6)$$

We verify the above relationship in HOL4 as follows:

Theorem 3.2.

$$\vdash \forall X \text{ Tr. } \text{FDEP } X \text{ Tr} = \text{D_OR } \text{Tr} (\text{D_INCLUSIVE_BEFORE } X \text{ Tr})$$

3.3.4 Spare Gates

Modeling spare parts in real systems is necessary when analyzing the probability of failure of the overall system, as these spares are used to replace the main parts after their failure. The main part Y of the spare gate, shown in Table 3.1, is replaced by the spare part X after a failure occurs. The spare gate has three variants depending on the type of the spare:

- *Cold SPare Gate (CSP)*: The spare part can only fail while it is active.
- *Hot SPare Gate (HSP)*: The spare part can fail in both the active and the dormant states with the same probability.
- *Warm SPare Gate (WSP)*: The spare part can fail in both the dormant and active states with different probabilities.

While manipulating the structure function of the DFT, it is required to distinguish between the two states of the spare part, i.e., the active state and the dormant state. Therefore a different variable is assigned to each state. For example, for the spare gate in Table 3.1, the variable X is assigned X_d and X_a for the dormant and active states, respectively [30]. This is required in case of a WSP gate, where the spare part has two different states. Recall that in the case of a CSP gate, it is not necessary to use these subscripts, since the spare part in the CSP gate does not work in the dormant state. Therefore, the active state only affects the DFT behavior and is included in the expressions. In the HSP gate, the spare part has the same behavior for both states and no subscript is required to distinguish between these two.

It can be noticed from the definition of the WSP gate that the output of the spare occurs in two cases: if the spare fails in its dormant state, then the main part fails or the main part fails then the spare is activated and then it fails in its active state. The last two terms in the WSP definition cover the possibility that the spare and the main part fail at the same time. This can happen if the main part and the spare are functionally dependent on the same trigger. The WSP represents the general case for the spare gates, while the CSP and HSP represent special cases of the WSP, where the spare cannot fail or is fully functioning in its dormant state. We have defined mathematical expressions for both the CSP gate for basic events and the HSP gate to facilitate using their expressions in DFT analysis. However, as will be seen shortly, we have verified that the behavior of our expressions is equivalent to a WSP under certain conditions. For the CSP gate, the output occurs if the main part fails then the spare is activated and then the spare fails while it is active. Since the spare part of the HSP has the same failure distribution in both of its states, the output of the HSP occurs when both inputs (main and spare) fail. Therefore, its behavior is equivalent to an AND gate.

We formally define in HOL the three variants of the spare gate as:

Definition 3.10. *CSP Gate*

$$\vdash \forall X Y. \text{CSP } Y \ X = (\lambda s. \text{ if } Y \ s < X \ s \text{ then } X \ s \text{ else PosInf})$$

Definition 3.11. *HSP Gate*

$$\vdash \forall X Y. \text{HSP } Y \ X = (\lambda s. \text{ max } (Y \ s)(X \ s))$$

Definition 3.12. *WSP Gate*

$$\begin{aligned} &\vdash \forall Y X_a X_d. \\ &\quad \text{WSP } Y X_a X_d = \\ &\quad \text{D_OR} \\ &\quad (\text{D_OR} \\ &\quad (\text{D_OR } (\text{D_AND } Y (\text{D_BEFORE } X_d Y)) \\ &\quad (\text{D_AND } X_a (\text{D_BEFORE } Y X_a))) \\ &\quad (\text{D_SIMULT } Y X_a)) (\text{D_SIMULT } Y X_d) \end{aligned}$$

Then, we formally verify that the WSP gate is equivalent to an HSP gate when the spare part in its dormant state is equal to its active state.

Theorem 3.3. $\vdash \forall X Y. \text{WSP } Y X X = \text{HSP } Y X$

Moreover, we formally verify that the WSP gate is equivalent to a CSP gate, if the spare part cannot fail in its dormant state. We formally verify this as:

Theorem 3.4.

$$\begin{aligned} &\vdash \forall X_a X_d Y. (X_d = \text{NEVER}) \wedge \\ &(\forall s. \text{ALL_DISTINCT } [Y \ s; X_a \ s]) \Rightarrow \text{WSP } Y X_a X_d = \text{CSP } Y X_a \end{aligned}$$

where $X_d = \text{NEVER}$ indicates that the spare part cannot fail in its dormant state, and ALL_DISTINCT ensures that the inputs cannot fail at the same time. This is because we defined the CSP gate for basic events. As can be seen from the above theorem, the CSP gate only deals with the active state of the spare, therefore, when dealing with a CSP there is no need to use the subscript.

In some real-world applications, a spare part can replace one of two main parts. This case is represented using shared spare gates, as shown in Table 3.1 [41]. The expression of the output Q_1 of the first gate is listed in Table 3.1 [30]. This expression

implies that the output Q_1 of this gate occurs in three different situations: (i) if the main part Y fails, then the spare fails while it is active (X_a), (ii) if the spare part fails in its dormant state X_d , then the main part fails, or (iii) if the second main part (of the other gate) Z fails before Y , and thus the spare is not available to replace Y when it fails. We use the DFT operators to formally model the behavior of this gate:

Definition 3.13. *Shared Spare*

$\vdash \forall Y Z X_a X_d.$

`shared_spare Y Z X_a X_d =`

`D_OR`

`(D_OR (D_AND Y (D_BEFORE X_d Y))`

`(D_AND X_a (D_BEFORE Y X_a)))`

`(D_AND Y (D_BEFORE Z Y)))`

3.4 Formal Verification of the Simplification

Theorems

In the DFT algebraic approach, many simplification theorems exist and are used to reduce the structure function of the top event [30]. In [41], we verified over 80 simplification theorems. However, these theorems were based on our old definitions of the DFT gates and operators that do not support probabilistic analysis. We verify all these theorems for the new definitions, presented in this work. These simplification theorems range from simple ones, such as commutativity of the AND, OR and Simultaneous operator, to more complex ones that include combinations of all the operators. Table 3.2 includes some of these verified properties. The verification details of these theorems as well as the gates definitions are available at [57].

Table 3.2: Examples of Formally Verified Simplification Theorems

DFT Algebra Theorems	HOL Theorems
$X + Y = Y + X$	$\vdash \forall X Y. D_OR\ X\ Y = D_OR\ Y\ X$
$X \cdot NEVER = NEVER$	$\vdash \forall X. D_AND\ X\ NEVER = NEVER$
$X \triangleleft (Y + Z) = (X \triangleleft Y) \cdot (X \triangleleft Z)$	$\vdash \forall X Y Z. D_BEFORE\ X\ (D_OR\ Y\ Z) = D_AND\ (D_BEFORE\ X\ Y)\ (D_BEFORE\ X\ Z)$
$X \trianglelefteq (Y + Z) = (X \trianglelefteq Y) \cdot (X \trianglelefteq Z)$	$\vdash \forall X Y Z.$ $D_INCLUSIVE_BEFORE\ X\ (D_OR\ Y\ Z) =$ $D_AND\ (D_INCLUSIVE_BEFORE\ X\ Y)$ $(D_INCLUSIVE_BEFORE\ X\ Z)$
$(X \trianglelefteq Y) + (X \Delta Y) = X \trianglelefteq Y$	$\vdash \forall X Y. D_OR\ (D_INCLUSIVE_BEFORE\ X\ Y)$ $(D_SIMULT\ X\ Y) = D_INCLUSIVE_BEFORE\ X\ Y$

3.5 Formal Qualitative Analysis of DFT

Examples

In this section, we illustrate, using three small DFT examples (Figure 3.2), the application of the proposed framework to qualitatively analyze a given DFT.

Using our proposed methodology, we are able to build a HOL formal DFT model and formally verify the reduction of the given DFT examples as expressed in the following three theorems. We assume that all inputs are basic events, i.e., they cannot fail at the same time. This condition can be relaxed if we are modeling a system with a common cause of failure for the inputs.

Theorem 3.5. *Reduced CPAND*

$$\vdash \forall X Y Z. (\forall s. ALL_DISTINCT [X\ s; Y\ s; Z\ s]) \Rightarrow$$

$$(PAND\ (PAND\ Z\ Y)\ X =$$

$$D_AND\ X\ (D_AND\ (D_BEFORE\ Z\ Y)\ (D_BEFORE\ Y\ X)))$$

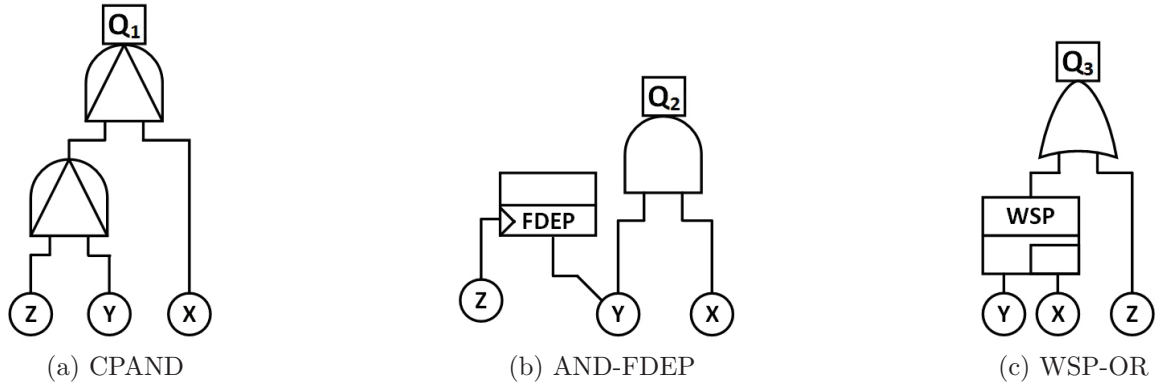


Figure 3.2: DFT Examples

Theorem 3.6. *Reduced AND-FDEP*

$$\vdash \forall X Y Z. X \cdot (\text{FDEP } Y Z) = \text{D_OR } (\text{D_AND } X Y) (\text{D_AND } X Z)$$

Theorem 3.7. *Reduced WSP-OR*

$$\vdash \forall Y X_a X_d Z.$$

$$(\forall s. \text{ALL_DISTINCT } [Y \ s; X_a \ s; X_d \ s; Z \ s]) \Rightarrow$$

$$(\text{D_OR } (\text{WSP } Y X_a X_d) Z =$$

$$\text{D_OR } (\text{D_AND } X_a (\text{D_BEFORE } Y X_a)) (\text{D_OR } (\text{D_AND } Y (\text{D_BEFORE } X_d Y)) Z)$$

As mentioned previously, each theorem can have a list of required conditions and a conclusion. For Theorem 3.5, the condition ensures that all random variables that represent system components are not equal, i.e., these random variables represent basic events. This is accomplished using the HOL4 function `ALL_DISTINCT`. The left hand side of the conclusion of this theorem represents the formal DFT expression for the given DFT, while the right hand side represents the verified reduced structure function. This also applies to Theorems 3.6 and 3.7. Using these verified reduced expressions, one can determine the cut sets and cut sequences to conduct the qualitative analysis. For example, the CPAND has only one sequence that can cause

the occurrence of the top event which is $[Z; Y; X]$. Similarly, the AND-FDEP DFT has only two cut sets $\{X; Y\}$ and $\{X; Z\}$. Finally, the WSP-OR DFT has two cut sequences $[Y; X_a]$ and $[X_d; Y]$ and one cut set represented by the single element $\{Z\}$. This indicates that using our proposed methodology, we have been able to formally conduct the qualitative analysis and determine the cut sets and cut sequences of a given DFT.

In order to emphasize on the importance of formally verifying the underlying math of the algebraic approach and the significance of knowing the required conditions for the analysis results to be valid, we provide more details regarding a flaw in one of the algebraic approaches. In [58], a simple algebra is introduced that provides definitions and simplification theorems for DFTs. We have been able to identify an error in one of the simplification theorems, which is the distributivity property of the sequence operator over the OR operator, i.e.,:

$$A.(B + C) = A.B + A.C \quad (3.7)$$

where $.$ and $+$ represent the sequence and OR operators, respectively. The sequence operator indicates that its output occurs if the input events occur in sequence from left to right, i.e., the time of occurrence of the left input event is less than that of the right input event. While the OR operator is represented by the minimum time of occurrence for the input events. Now, assuming that the time of occurrence of the input events A , B , and C are d_A , d_B and d_C , respectively, the left hand side of Equation (3.7) occurs only if $d_A < \min(d_B, d_C)$. While, the right hand side of the same equation occurs if $d_A < d_B$ or $d_A < d_C$. This property fails to hold when $d_A > \min(d_B, d_C)$ but at the same time $d_A < \max(d_B, d_C)$, i.e., d_A falls in the middle between d_B and d_C . In this particular case, the left hand side of Equation (3.7) will

not occur because d_A is not less than the minimum of d_B and d_C , however, one of the terms of the right hand side occurs. For this property to hold, it is required to have the condition $d_A < \min(d_B, d_C)$. We have been able to identify this flaw using theorem proving, as the property was not verifiable for the aforementioned case unless this particular condition is added. As a consequence, using this property without the required condition in any application, including the application part of the mentioned paper [58], would lead to erroneous results, which is serious specially for safety-critical systems that cannot tolerate any error in the analysis. These findings emphasize on the importance of having a rigorous formal framework for DFT analysis, which is essential for safety-critical systems. In the next section, we will apply our methodology on two real-world case studies.

3.6 Formal Qualitative Analysis Case Studies

In this section, we apply our methodology for conducting the qualitative analysis of two safety-critical systems, i.e., a drive-by-wire (DBW) system to control the brakes and throttle systems of modern vehicles [59] and a cardiac assist system (CAS) that provides care to patients with heart failure [60]. The analysis of these systems should be carefully conducted as any error may lead to the loss of life in extreme cases.

In order to conduct the formal qualitative analysis using our framework, we provide generic steps that can be followed:

1. Express the structure function of the top event of the DFT using the DFT algebra.
2. Simplify the structure function and formally verify that the simplified and original functions are equal using the simplification theorems.

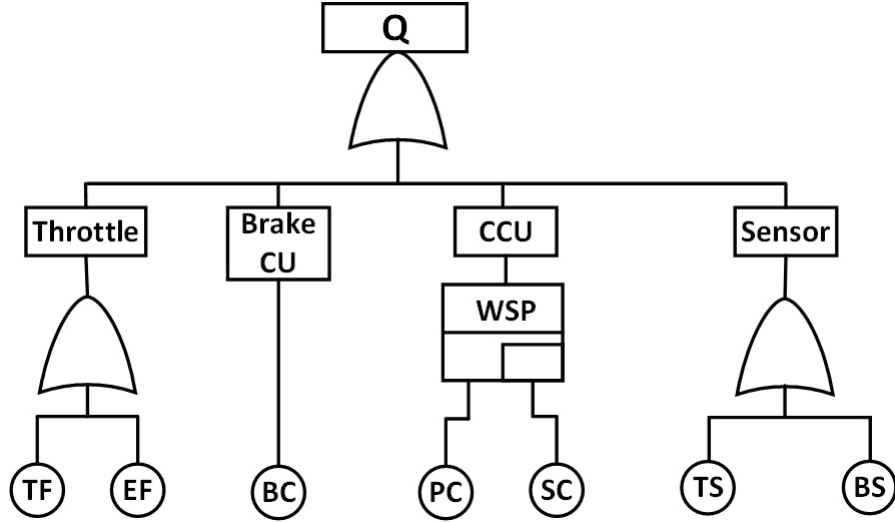


Figure 3.3: DFT of Drive-by-wire System

3. Obtain a reduced form of the cut sets and cut sequences.

We demonstrate the utilization of these steps in the formal qualitative analysis of the DBW and CAS systems.

3.6.1 Qualitative Analysis of DBW

The DFT of the drive-by-wire (DBW) system is shown in Figure 3.3 [59]. We chose to analyze the brake and the throttle parts of this system, which consists of the following parts: the brakes control unit (BC), the throttle failure (TF), two sensors; the brake sensor (BS) and the throttle sensor (TS), the engine failure (EF) and finally the primary central control (PC) unit with its spare part (SC_d and SC_a for both the dormant and active states, respectively). We model the spare part of the central control unit as a warm spare, as this is the general case for the spare. In addition, this is the most convenient way to model it as the spare control unit can be working in the sleep mode, and it will only be activated after the main unit fails.

We proceed with the analysis of the DBW system following the steps outlined in

our proposed methodology. We first start by verifying the reduction of the structure function. The reduced structure is:

$$Q_{DBW} = TF + EF + BC + SC_a \cdot (PC \triangleleft SC_a) + PC \cdot (SC_d \triangleleft PC) + TS + BS \quad (3.8)$$

We formally verify this reduced form as:

Theorem 3.8. *Reduced DBW*

$\vdash \forall BS \ TS \ PC \ SC_a \ SC_d \ BC \ EF \ TF .$

$(\forall s . \text{ ALL_DISTINCT}$

$[BS \ s; \ TS \ s; \ PC \ s; \ SC_a \ s; \ SC_d \ s; \ BC \ s; \ EF \ s; \ TF \ s]) \Rightarrow$

$(D_OR$

$(D_OR \ (D_OR \ (D_OR \ TF \ EF) \ (WSP \ PC \ SC_a \ SC_d)) \ BC)$

$(D_OR \ TS \ BS)) =$

D_OR

$(D_OR$

$(D_OR$

$(D_OR$

$(D_OR \ (D_OR \ TF \ EF) \ BC)$

$(D_AND \ SC_a \ (D_BEFORE \ PC \ SC_a)))$

$(D_AND \ PC \ (D_BEFORE \ SC_d \ PC))) \ TS) \ BS)$

From this expression, we can find a reduced form of the *cut sequences*:

$$[PC, SC_a] , [SC_d, PC]$$

which means that the top event can fail due to two different sequences of input failures.

The first one is the failure of the main control unit (PC) followed by the failure of

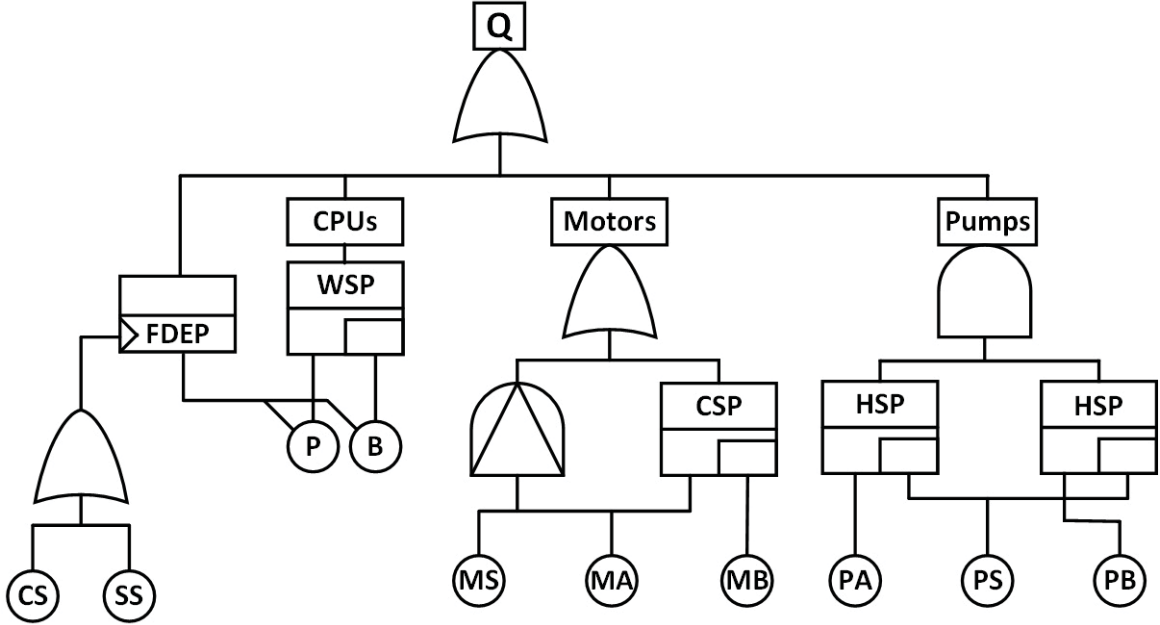


Figure 3.4: DFT of Cardiac Assist System

the spare in its active state (SC_a). The second sequence is when the spare part fails in its dormant state (SC_d) followed by the failure of the main control unit.

In a similar way, a reduced form of the *cut sets* can be extracted from the reduced top event expression as:

$$\{TF\}, \{EF\}, \{BC\}, \{TS\}, \{BS\}$$

3.6.2 Qualitative Analysis of CAS

The DFT for the cardiac assist system (CAS) is shown in Figure 3.4 [60]. The system consists of three sub-systems: pumps, motors and CPUs. There are two main pumps PA and PB . After the failure of one of these pumps, a shared spare PS replaces the failed one. There are two motors MA and the spare MB and a switch MS . The motor sub-system fails if MS then MA fail in sequence or if MA and the spare MB fail. Finally, there is one main CPU P and its spare B . Both CPUs are functionally dependent on the union of a crossbar switch (CS) and the system supervisor (SS).

We consider here different variations of spare gates, to make this case study more general and inclusive to all the formalized gates, as shown in Figure 3.4. A simplified version of this DFT, where we assumed that all spare gates are HSPs gates, was verified in [61]. However, the variations that we assume here for the spares allow modeling and verifying the probability of failure of the given system, while the independence of some of the events does not hold anymore, which makes it a more general case.

We start first by verifying a reduced version of the structure function of the top event to enable us to perform the qualitative analyses on a reduced function. The reduced form of the structure function can be expressed as:

$$Q_{CAS} = CS + SS + MA \cdot (MS \triangleleft MA) + MB \cdot (MA \triangleleft MB) + B_a \cdot (P \triangleleft B_a) + P \cdot (B_d \triangleleft P) + PA \cdot PB \cdot PS \quad (3.9)$$

We formally verify this reduction in HOL4 as:

Theorem 3.9. *Reduced CAS*

$\vdash \forall PA PB PS MS MA MB B_a B_d CS SS P.$

$(\forall s. \text{ALL_DISTINCT}$

$[\text{MA } s; \text{MS } s; \text{PA } s; \text{PB } s; \text{PS } s; \text{MB } s; \text{P } s; \text{B}_d s; \text{B}_a s; \text{CS } s; \text{SS } s] \wedge$

$(\text{D_BEFORE } B_a P = \text{NEVER}) \Rightarrow$

$(\text{D_OR}$

$(\text{D_AND } (\text{shared_spare } PA PB PS PS)$

$(\text{shared_spare } PB PA PS PS))$

$(\text{D_OR}$

$(\text{D_OR } (\text{PAND } MS MA) (\text{CSP } MA MB))$

$$\begin{aligned}
& (\text{WSP } (\text{FDEP } (\text{D_OR } \text{CS } \text{SS}) \text{P}) (\text{FDEP } (\text{D_OR } \text{CS } \text{SS}) \text{B}_a) \\
& \quad (\text{FDEP } (\text{D_OR } \text{CS } \text{SS}) \text{B}_d))) = \\
& \text{D_OR} \\
& \quad (\text{D_OR} \\
& \quad \quad (\text{D_OR} \\
& \quad \quad \quad (\text{D_OR} \\
& \quad \quad \quad \quad (\text{D_OR } \text{CS } \text{SS}) (\text{D_AND } \text{MA } (\text{D_BEFORE } \text{MS } \text{MA}))) \\
& \quad \quad \quad \quad (\text{D_AND } \text{MB } (\text{D_BEFORE } \text{MA } \text{MB}))) \\
& \quad \quad \quad \quad (\text{D_AND } \text{B}_a (\text{D_BEFORE } \text{P } \text{B}_a))) \\
& \quad \quad \quad (\text{D_AND } \text{P } (\text{D_BEFORE } \text{B}_d \text{P}))) (\text{D_AND } \text{P}_a (\text{D_AND } \text{PB } \text{PS})))
\end{aligned}$$

where ALL_DISTINCT ensures that the inputs do not occur at the same time, and (D_BEFORE B_a P = NEVER) ascertains that the spare part B in its active state cannot fail before P . This ensures the proper behavior of the WSP gate. It is worth mentioning that such a condition is not required for the HSP gate as the spare part exhibits the same failure behavior in both of its states.

It is important to mention that since the inputs of the WSP gate are functionally dependent on the union of CS and SS , we use (FDEP (CS+SS) P), (FDEP (D_OR CS SS) B_a) and (FDEP (D_OR CS SS) B_d) for the main, the spare in its active state and the spare in its dormant state, respectively.

From the verified reduced top event, we can conclude a reduced form of *cut sequences* as follows:

$$[MS; MA], [MA; MB], [P; B_a], [B_d; P]$$

Moreover, a reduced form of the *cut sets* is deducted as:

$$\{CS\}, \{SS\}, \{PA, PB, PS\}$$

3.7 Summary

In this chapter, we provided the formalization of the DFT operators and gates based on the DFT algebraic approach. Furthermore, we formally verified several simplification theorems that are utilized to reduce the structure function of a given DFT. We used this reduced format to conduct the qualitative analysis and obtain a reduced form of the cut sets and cut sequences. We provided the details of a flaw in a published DFT algebra, which further highlights the importance of our formalization. Finally, we conducted the formal qualitative analysis of two case studies. One of the main challenges faced during our formalization is choosing the proper data-type that can be used to capture the behavior of each gate and at the same time allows the verification of the simplification theorems. In the next chapter, we provide the verification details of the probabilistic behavior of each gate to enable performing the quantitative analysis formally within a theorem prover.

Chapter 4

Formal Quantitative Analysis of Dynamic Fault Trees

In this chapter, we introduce a methodology for the formal DFT quantitative analysis. We provide the verification details of the probabilistic failure expressions of DFT gates. Based on these verified expressions and the proposed methodology, we perform the quantitative analysis of two case studies to verify generic expressions of probability of failure that are independent of the failure distributions of systems components.

4.1 Methodology

The DFT formal quantitative analysis methodology is depicted in Figure 4.1. The analysis starts with a system description that is used to create a DFT model. Based on the formalized DFT gates, a formal DFT model is created. Then we use the simplification theorems to verify a reduced DFT model, i.e., a reduced structure function. The quantitative analysis is conducted by utilizing this reduced structure

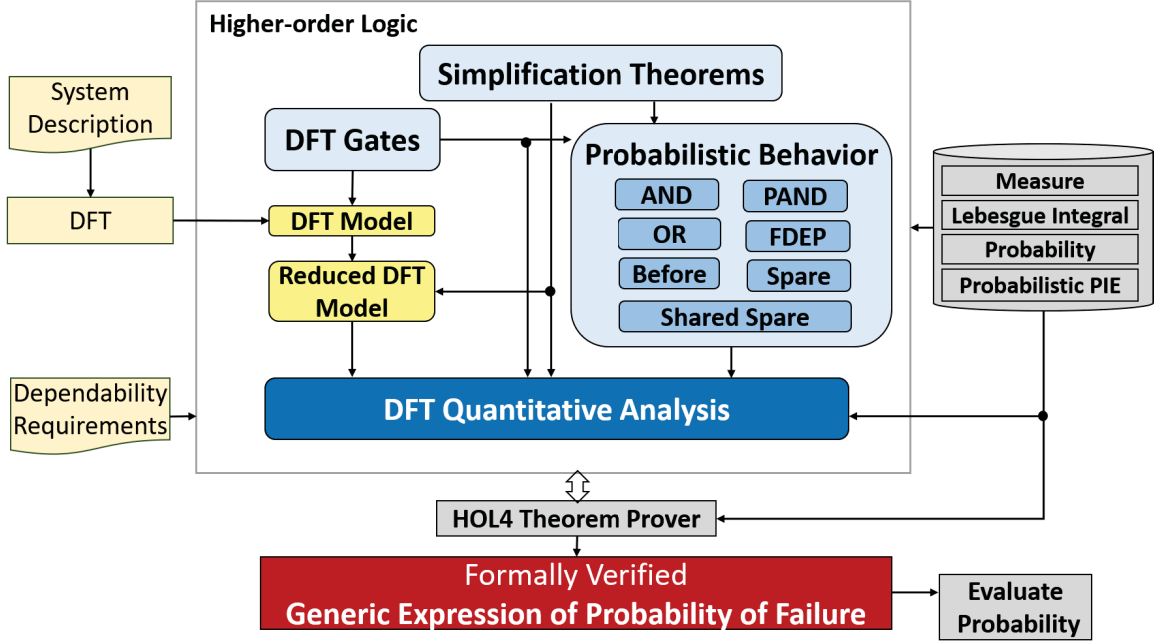


Figure 4.1: Formal DFT Quantitative Analysis Methodology

function to generate formally verified generic expressions of probability of failure. This last step requires having a library of verified probabilistic failure expressions of DFT gates and operators, like AND and PAND gates. Creating this library requires some existing HOL4 libraries, such as the measure, Lebesgue integral, probability and probabilistic PIE theories. We provide probability of failure expressions that are generic by using universally quantified probability density and distribution functions. The distribution and density functions and the variables in these generic expressions can be instantiated and evaluated in any other tool, such as MATLAB [62], to evaluate the probability of failure of a given system. A set of probabilistic expressions for the gates and operators was proposed in [30]. However, these expressions were not formally verified, and thus they cannot be fully trusted for the formal DFT analysis as such. Furthermore, there are some missing gaps in the paper-and-pencil proofs available in [30] that we were able to fill using our formalization, which is built on top of the Lebesgue integral and probability theories. In [30], there is no direct description

on how to build the DFT analysis based on the above-mentioned theories. Besides this, we also had to use different strategies for some proofs. All these differences will be highlighted throughout this chapter. We present a summary of the challenges that we faced during the formalization of the probabilistic failure behavior of DFT gates at the end of this chapter.

4.2 Probabilistic Model of DFT Gates

The foremost requirement for formally conducting the probabilistic analysis of DFTs in a HOL theorem prover is to have verified expressions for the probability of failure of DFT gates. Therefore, it is required to formally model and verify the probability of failure expression for each DFT gate. We assume that the basic events of the DFT are independent. However, in some cases these events can be dependent; in particular in the case of CSP and WSP, where the failure of the main part affects the operation and failure of the spare part. We handle this by first introducing the probabilistic behavior of the gates for independent events, and then we present the probabilistic behavior of the *WSP* and the *CSP* gates, which deal with dependent events.

Assuming that we are interested in finding the probability of failure until time t , the following four expressions can be used to express the probability of any DFT gate with independent basic events [30]:

$$Pr\{X \cdot Y\}(t) = F_X(t) \times F_Y(t) \quad (4.1a)$$

$$Pr\{X + Y\}(t) = F_X(t) + F_Y(t) - F_X(t) \times F_Y(t) \quad (4.1b)$$

$$Pr\{Y \cdot (X \triangleleft Y)\}(t) = \int_0^t f_Y(y) F_X(y) dy \quad (4.1c)$$

$$Pr\{X \triangleleft Y\}(t) = \int_0^t f_X(x)(1 - F_Y(x)) dx \quad (4.1d)$$

where F_X and F_Y represent the CDFs of the random variables X and Y , respectively, and f_X and f_Y represent their corresponding probability density functions (PDFs).

Equation (4.1a) represents the probability of the AND and HSP gates, which results from the probability of intersection of two independent events. Equation (4.1b) describes the probability of the OR and FDEP gates, which corresponds to the probability of union of two independent events. Equation (4.1c) represents the probability of having two basic events occurring in sequence one *after* the other until time t , i.e., $Pr(X < Y)$ until time t or $Pr(X < Y \wedge Y \leq t)$, which is the failure probability of the PAND for basic events. Finally, the probability of the *Before* operator is represented by Equation (4.1d), which is the probability of having event X occurring *before* event Y until time t , i.e., $Pr(X < Y \wedge X \leq t)$. The difference between the last two events (*before* and *after*) is that in the *before* event, we are just interested in finding the probability of failure of X until time t with the condition that X fails before Y . So, it is not necessary that Y fails. While in the *after* event, we find the probability of failure of Y until time t with the condition that Y fails after X . So, it is required that both X and Y fail in sequence.

Since the probability is applied for sets that belong to the events of the probability space, we define a `DFT_event` that satisfies the condition that the input function

is less than or equal to time t , which represents the moment of time until which we are interested in finding the probability of failure. Without this `DFT_event`, there is no possible way to apply the probability directly to DFT gates. We first need to create the `DFT_event` for the time-to-failure function of the output event of any gate or DFT, then apply the probability to it.

Definition 4.1. *DFT Event*

$$\vdash \forall p \ X \ t. \ \text{DFT_event } p \ X \ t = \{s \mid X \ s \leq \text{Normal } t\} \cap p_space \ p$$

where `Normal` typecasts the type of t from `real` to `extreal`, p represents the probability space and X represents the time-to-failure function.

We formally verify the equivalence between the probability of the `DFT_event` of an extended real function and its equivalent CDF of the real version of the function as:

Theorem 4.1.

$$\vdash \forall X \ p \ t. \ (\forall s. \ X \ s \neq \text{PosInf} \wedge 0 \leq X \ s) \Rightarrow \\ (\text{CDF } p \ (\text{real } \circ X) \ t = \text{prob } p \ (\text{DFT_event } p \ X \ t))$$

where `real` is the mirror opposite to the typecasting `Normal` operator. This typecasting is required as the `DFT_event` is defined for `extreal` data-type, and the CDF is defined for real random variables only. Therefore, it is required to ensure that the input function does not equal $+\infty$ and is greater than or equal to 0 since it represents the time of failure of a system component.

4.2.1 Probabilistic Model of AND Gate

To formally verify Equation (4.1a), we verify the equivalence of the DFT event of the AND gate to the intersection of two events:

Lemma 4.1.

$\vdash \forall p \ t \ X \ Y.$

$$\text{DFT_event } p \ (\text{D_AND } X \ Y) \ t = \text{DFT_event } p \ X \ t \cap \text{DFT_event } p \ Y \ t$$

Based on the independence of random variables and using Theorem 4.1, we formally verify Equation (4.1a) in HOL4 as:

Theorem 4.2.

$\vdash \forall p \ t \ X \ Y. \text{rv_gt0_ninfinity } [X; Y] \wedge$

$$\text{indep_var } p \ \text{lborel } (\text{real } \circ X) \ \text{lborel } (\text{real } \circ Y) \Rightarrow$$

$$(\text{prob } p \ (\text{DFT_event } p \ (\text{D_AND } X \ Y) \ t) =$$

$$\text{CDF } p \ (\text{real } \circ X) \ t * \text{CDF } p \ (\text{real } \circ Y) \ t$$

where `indep_var` ensures the independence of the random variables, X and Y , over the Lebesgue-Borel (`lborel`) measure [53]. `rv_gt0_ninfinity` is required since we are dealing with the real versions of the random variables. It is a logical condition, since any real-world component will eventually fail, so we are interested only in dealing with the time of failure that is not $+\infty$.

In Theorem 4.2, the random variables are typecasted as real-valued, using the operator `real`, to function over the Lebesgue-Borel (`lborel`) measure. `lborel` is purposely used here to facilitate the Lebesgue integration over the real line when expressing the probabilities of the *before* and *after* events. Theorem 4.2 represents the probability of the AND gate and the *HSP* gate, since the behavior of the *HSP* is equivalent to the behavior of the AND gate.

4.2.2 Probabilistic Model of OR and FDEP Gates

To formally verify Equation (4.1b), we verify the equivalence of the DFT event of the OR as the union of two events:

Lemma 4.2.

$$\vdash \forall p \ t \ X \ Y.$$

$$\text{DFT_event } p \ (\text{D_OR } X \ Y) \ t = \text{DFT_event } p \ X \ t \cup \text{DFT_event } p \ Y \ t$$

We formally verify Equation (4.1b) based on the probabilistic PIE, the independence of random variables and using Theorem 4.1 as:

Theorem 4.3.

$$\vdash \forall p \ t \ X \ Y. \text{rv_gt0_ninfinity } [X; Y] \wedge$$

$$\text{All_distinct_events } p \ [X;Y] \ t \wedge$$

$$\text{indep_var } p \ \text{lborel } (\text{real } \circ X) \ \text{lborel } (\text{real } \circ Y) \Rightarrow$$

$$(\text{prob } p \ (\text{DFT_event } p \ (\text{D_OR } X \ Y) \ t) =$$

$$\text{CDF } p \ (\text{real } \circ X) \ t + \text{CDF } p \ (\text{real } \circ X) \ t -$$

$$\text{CDF } p \ (\text{real } \circ X) \ t \times \text{CDF } p \ (\text{real } \circ Y) \ t)$$

where `All_distinct_events` asserts that the event sets are not equal. We formally define it as:

Definition 4.2.

$$\vdash \text{All_distinct_events } p \ L \ t =$$

$$\text{ALL_DISTINCT } (\text{MAP } (\lambda x. \text{DFT_event } p \ x \ t) \ L)$$

where `ALL_DISTINCT` is a HOL4 predicate, which ensures that the elements of its input list are not equal and `MAP` is a function that applies the input function $(\lambda x. \text{DFT_event } p \ x \ t)$ to all the elements in the list `L` and returns a list. This condition is required for the probabilistic PIE.

Theorem 4.3 provides the probability of the OR gate as well as the FDEP gate, since the behavior of the FDEP is equivalent to the OR gate.

It is worth noting that in [30], Equations (4.1a) and (4.1b) were just presented without any information on how to link them to the definitions of the AND and OR gates. We should recall that the AND and OR gates are defined as the maximum and minimum of their operands. Looking at these definitions does not give any knowledge about how the probability of the AND gate is equivalent to the probability of the intersection or how the probability of the OR gate is equal to the probability of the union. However, using our formalization and utilizing our formal definition of `DFT_event`, we are able to verify that the `DFT_event` of the AND gate is equal to the intersection of the input events and that the `DFT_event` of the OR gate is equal to the union of the input events. Based on this, we can ensure that the probability of the AND and OR gates are represented using Equations (4.1a) and (4.1b), respectively.

4.2.3 Probabilistic Model of PAND Gate and Before Operator

In this section, we present the formalization details of the probabilistic model of the PAND gate and the before operator.

We verify Equations (4.1c) and (4.1d) as Theorems 4.4 and 4.5, respectively.

Theorem 4.4.

$\vdash \forall X Y p f_y t.$

$$\begin{aligned} & \text{rv_gt0_ninfinity } [X; Y] \wedge 0 \leq t \wedge \text{prob_space } p \wedge \\ & \text{indep_var } p \text{ lborel } (\text{real } \circ X) \text{ lborel } (\text{real } \circ Y) \wedge \\ & \text{distributed } p \text{ lborel } (\text{real } \circ Y) f_y \wedge (\forall y. 0 \leq f_y y) \wedge \\ & \text{cont_CDF } p (\text{real } \circ X) \wedge \\ & \text{measurable_CDF } p (\text{real } \circ X) \Rightarrow \\ & (\text{prob } p (\text{DFT_event } p (Y \cdot (X < Y)) t) = \end{aligned}$$

$$\begin{aligned} & \text{pos_fn_integral lborel} \\ & (\lambda y. \text{fy } y * \\ & (\text{indicator_fn } \{w \mid 0 \leq w \wedge w \leq t\} y * \\ & \text{CDF } p \text{ (real o } X) y))) \end{aligned}$$

Theorem 4.5.

$\vdash \forall X Y p \text{fy } t.$

$$\begin{aligned} & \text{rv_gt0_ninfinity } [X; Y] \wedge 0 \leq t \wedge \text{prob_space } p \wedge \\ & \text{indep_var } p \text{ lborel (real o } X) \text{ lborel (real o } Y) \wedge \\ & \text{distributed } p \text{ lborel (real o } X) \text{fx} \wedge (\forall x. 0 \leq \text{fx } x) \wedge \\ & \text{measurable_CDF } p \text{ (real o } Y) \Rightarrow \\ & (\text{prob } p \text{ (DFT_event } p \text{ (} X < Y) t) = \\ & \text{pos_fn_integral lborel} \\ & (\lambda x. \text{fx } x * \\ & (\text{indicator_fn } \{u \mid 0 \leq u \wedge u \leq t\} x * \\ & (1 - \text{CDF } p \text{ (real o } Y) x)))) \end{aligned}$$

where `pos_fn_integral` is the Lebesgue integral for positive functions [28], `fy` and `fx` are the PDF of random variables of the real version of the functions `Y` and `X`, respectively. `cont_CDF` is required in Theorem 4.4 as we need to prove that $Pr(X \leq t)$ and $Pr(X < t)$ are equal, and this is not valid unless the CDF function is continuous (`cont`).

Verifying Theorems 4.4 and 4.5 is not a straightforward task due to the involvement of Lebesgue integration. To the best of our knowledge, this is the first time that these proofs are formally verified in a theorem prover, where we are able to identify the exact steps to reach the final form of Theorems 4.4 and 4.5. In addition, in [30], Equation (4.1c) is presented without any proof, while a proof is presented for Equation

(4.1d) that is based mainly on the probability of disjoint events and utilizes derivatives to reach the final expression. However, we have been able to verify the same expression of Equation (4.1d), but following a different and simpler proof, which is similar to the proof of Equation (4.1c) to reach the final form of Theorem 4.5 without using derivatives. We first prove the probability of sets of real random variables in the form of integration before extending the proofs to extended real functions.

Proof Strategy for Theorem 4.4

To verify Theorem 4.4, we first express the event set that corresponds to the integration in Equation (4.1c) as:

$$(X, Y)^{-1}\{(u, w) \mid u < w \wedge 0 \leq w \wedge w \leq t\} \quad (4.2)$$

Then, we verify that the probability of this set can be written using integration as in Equation (4.1c). Therefore, we verify the relationship between the distribution and the integration of positive functions using the push forward measure (`distr`):

Theorem 4.6.

$\vdash \forall X \text{ p } M \text{ A}.$

```

measure_space M  $\wedge$ 
random_variable X p (m_space M, measurable_sets M)  $\wedge$ 
A  $\in$  measurable_sets M  $\Rightarrow$ 
(distribution p X A =
  pos_fn_integral (distr p M X) (indicator_fn A))

```

It is worth mentioning that this theorem can be used in the verification process of other applications and not only for DFT analysis. We use Theorem 4.6 to verify

the relationship between the probability and the integration of the joint distribution F_{XY} of two independent random variables as:

$$Pr(X, Y)^{-1}(A) = \int \mathbf{1}_A dF_{XY} \quad (4.3)$$

We formalize this relationship in HOL4 and use a property, which converts the distribution of a pair measure of independent measures into the pair measure of the individual distributions [53], to split the integral of joint distributions into two integrals of the individual distributions ($\int \int \mathbf{1}_A dF_X dF_Y$). In order to reach the final form of Equation (4.1c), we express it in the form of two integrals:

$$\int_0^t f_Y(y) \times F_X(y) dy = \int_0^t \int_{-\infty}^y f_Y(y) \times f_X(x) dx dy \quad (4.4a)$$

$$= \int_0^t f_Y(y) \left(\int_{-\infty}^y f_X(x) dx \right) dy \quad (4.4b)$$

The problem in Equations (4.4a) and (4.4b) lies in the fact that the outer integral is a function of the inner integral, i.e., for the inner integral we are integrating until y which is the variable of the outer integral. To be able to handle this formally, we verify that the indicator function of the set in Equation (4.2) can be written in the form of the multiplication of two indicator functions, where one is a function of the other.

Lemma 4.3.

$\vdash \forall x y t.$

$$\text{indicator_fn } \{(u,w) \mid u < w \wedge 0 \leq w \wedge w \leq t\}(x,y) =$$

$$\text{indicator_fn } \{w \mid 0 \leq w \wedge w \leq t\} y * \text{indicator_fn } \{u \mid u < y\} x$$

In order to use the above-mentioned lemma and the set on the left hand side, we need to verify that this set is measurable in the two dimensional borel space, i.e., the set belongs to the measurable sets of `pair_measure lborel lborel`. This property can be verified based on the fact that a countable union of measurable sets is also measurable. We verify this fact on the rational numbers \mathbb{Q} as follows:

Theorem 4.7.

$\vdash \forall m s.$

$$\text{measure_space } m \wedge (\forall n. n \in \text{Q_set} \Rightarrow s \ n \in \text{measurable_sets } m) \Rightarrow \\ \text{BIGUNION (IMAGE } s \ \text{Q_set)} \in \text{measurable_sets } m$$

where m in our case is `pair_measure lborel lborel`. This theorem is generic and can be used in other contexts than DFTs.

The purpose of using the set of rational numbers is that we need a countable set that can be used to express the set in Lemma 4.3 as the union of borel rectangles. We verify this in HOL4 as:

Lemma 4.4.

$\vdash \forall t. \text{BIGUNION}$

$$\{\{u \mid u < \text{real } q\} \times \{w \mid \text{real } q < w \wedge 0 \leq w \wedge w \leq t\} \mid \\ q \in \text{Q_set}\} = \\ \{(u,w) \mid u < w \wedge 0 \leq w \wedge w \leq t\}$$

Besides this, we also verify a lemma that the sets in the union of Lemma 4.4 are measurable sets in the `pair_measure lborel lborel` as:

Lemma 4.5.

$$\vdash \forall t q. \{u \mid u < \text{real } q\} \times \{w \mid \text{real } q < w \wedge 0 \leq w \wedge w \leq t\} \in \\ \text{measurable_sets (pair_measure lborel lborel)}$$

We can use the proof steps of the previous lemmas to verify the same properties for similar sets, which is essential for other gates expressions. This facilitates dealing with other events in the future, by following the steps in our proof.

By using the above lemmas, we can verify that the original set is a measurable set in the `pair_measure lborel lborel` as:

Lemma 4.6.

$$\vdash \forall t. \{ (u, w) \mid u < w \wedge 0 \leq w \wedge w \leq t \} \in$$

$$\text{measurable_sets (pair_measure lborel lborel)}$$

We use Lemmas 4.3 and 4.6 to verify that the expression given in Equation (4.4b) is equal to $\int_A dF_X dF_Y$, where A is the set that specifies the boundaries of the integral. We verify this in HOL4 using the push forward measure as:

Lemma 4.7.

$$\vdash \forall X Y p t.$$

$$\text{prob_space } p \wedge \text{indep_var } p \text{ lborel } X \text{ lborel } Y \Rightarrow$$

$$(\text{pos_fn_integral}$$

$$(\text{pair_measure (distr } p \text{ lborel } X)(\text{distr } p \text{ lborel } Y))$$

$$(\lambda(x, y). \text{ indicator_fn}\{ (u, w) \mid u < w \wedge 0 \leq w \wedge w \leq t \}(x, y) =$$

$$\text{pos_fn_integral (distr } p \text{ lborel } Y)$$

$$(\lambda y. \text{ indicator_fn } \{ w \mid 0 \leq w \wedge w \leq t \} y *$$

$$\text{pos_fn_integral(distr } p \text{ lborel } X)$$

$$(\lambda x. \text{ indicator_fn } \{ u \mid u < y \} x)))$$

where `pair_measure (distr p lborel X) (distr p lborel Y)` represents the joint distribution of the push forward measures of random variables X and Y over the borel space.

We verify several essential properties for CDF in order to prove that the inner integral of Lemma 4.7 is equal to $F_X(y)$ or formally to $(\text{CDF } p \ X \ y)$. In order to have the PDF of random variable Y in the integral, we assume that the random variable Y has a PDF by defining a density measure for Y . We ported the following definition, `distributed`, from Isabelle/HOL[18], where f in this definition is the PDF of random variable X , and the measure part of the density measure is the integral of this PDF. Using this definition, the integral of f is equal to the distribution of the random variable X .

Definition 4.3. *distributed*

$\vdash \forall p \ M \ X \ f.$

`distributed` $p \ M \ X \ f \Leftrightarrow$

$X \in$

`measurable`(`m_space` p ,`measurable_sets` p)

(`m_space` M ,`measurable_sets` M) \wedge

$f \in \text{measurable}(\text{m_space } M, \text{measurable_sets } M) \text{ Borel} \wedge$

$\text{AE } M \ \{x \mid 0 \leq f \ x\} \wedge (\text{distr } p \ M \ X = \text{density } M \ f)$

where `density` is the density measure, and $\text{AE } M \ \{x \mid 0 \leq f \ x\}$ ensures that the PDF f is almost everywhere (AE) positive over the measure M . We also use a theorem that replaces the integration with respect to the density measure by the PDF with respect to the original measure (`lborel` in our case) [18]. In addition to the previously verified theorems, we also prove some additional properties, such as a sigma finite measure for the push forward measure over the borel space (`sigma_finite_measure` (`distr` p `lborel` X)). We also verify that the space generated by the pair measure of two distributions over the borel space is a sigma algebra (`sigma_algebra` (`m_space` (`pair_measure` (`distr` p `lborel` X)(`distr` p `lborel` Y)), `measurable_sets`

(pair_measure (distr p lborel X)(distr p lborel Y))). Moreover, we verify that the space generated by the space and the measurable sets of the pair measure of lborel is also a sigma algebra (sigma_algebra (m_space (pair_measure lborel lborel), measurable_sets (pair_measure lborel lborel))). Finally, we prove that the set of the left-hand side of Equation (4.1c) is equal to the set that corresponds to the integration of the right-hand side of the same equation as:

Lemma 4.8.

$\vdash \forall p \ t \ X \ Y.$

$$\begin{aligned} & \text{rv_gt0_ninfinity } [X; Y] \wedge 0 \leq t \Rightarrow \\ & (\text{DFT_event } p \ (Y \cdot (X < Y)) \ t = \\ & \text{PREIMAGE } (\lambda x. \ (\text{real } (X \ x), \ \text{real } (Y \ x))) \\ & \{(u, w) \mid u < w \wedge 0 \leq w \wedge w \leq t\} \cap p_space \ p \end{aligned}$$

Based on all the above mentioned lemmas, we are able to verify the original goal for Equation (4.1c) as in Theorem 4.4.

Proof Strategy for Theorem 4.5

For the verification of Theorem 4.5, we follow almost the same steps for the previous proof. We start by first writing the event set for the integration as:

$$(X, Y)^{-1}\{(u, w) \mid 0 \leq u \wedge u \leq t \wedge u < w\} \tag{4.5}$$

Then, we describe the indicator function of this set as the multiplication of two indicator functions as:

Lemma 4.9.

$\vdash \forall x y t.$

$$\begin{aligned} & \text{indicator_fn } \{(u,w) \mid 0 \leq u \wedge u \leq t \wedge u < w\}(x,y) = \\ & \text{indicator_fn } \{u \mid 0 \leq u \wedge u \leq t\} x * \text{indicator_fn } \{w \mid x < w\} y \end{aligned}$$

Similar to the procedure explained previously for the set of the after event in Lemmas 4.4, 4.5 and 4.6, we verify that the set of the before event is a measurable set in the `pair_measure lborel lborel`.

Finally, we rewrite Equation (4.1d) as:

$$\begin{aligned} Pr\{X \triangleleft Y\}(t) &= \int_0^t \int_x^\infty f_X(x) f_Y(y) dy dx \\ &= \int_0^t f_X(x) \left(\int_x^\infty f_Y(y) dy \right) dx \end{aligned} \tag{4.6}$$

We verify some additional properties for the CDF in order to complete the proof. For example, we verify that $\int_x^\infty f_Y(y) dy$ is equal to $1 - F_Y(x)$. Similarly, we also formally verify that the event of the left-hand side of Equation (4.1d) is equal to the set that corresponds to the integration of the right-hand side of the same equation. We use the set in Equation (4.5) to verify this as:

Lemma 4.10.

$\vdash \forall p t X Y.$

$$\begin{aligned} & \text{rv_gt0_ninfinity } [X; Y] \wedge 0 \leq t \Rightarrow \\ & (\text{DFT_event } p (X \triangleleft Y) t = \\ & \text{PREIMAGE } (\lambda s. (\text{real } (X s), \text{real } (Y s))) \\ & \{ (u,w) \mid 0 \leq u \wedge u < w \wedge u \leq t \} \cap \text{p_space } p) \end{aligned}$$

Based on all these verified theorems, we are able to formally verify Theorem 4.5.

So far, we presented the formal verification of the probabilistic behavior of:

- The AND gate using Theorem 4.2.
- The probability of the OR and FDEP gates using Theorem 4.3 (since they are equivalent).
- The probability of the PAND gate for basic events using Theorem 4.4.
- The probability of the *Before* operator using Theorem 4.5.

There is no probability of failure for the *Simultaneous* operator as it is eliminated for basic events according to Equation (3.3). This implies that the probability of the *Inclusive Before* operator is equal to the probability of the *Before* operator for basic events.

It is worth mentioning that the inputs of these gates can be dependent in case of having a common cause of failure. In this case the probability of intersection should be handled using conditional probabilities, in a similar manner to the spare gate as will be explained in the sequel.

4.2.4 Probabilistic Model of Spare Gates

As mentioned in Chapter 3, the behavior of the HSP gate is equal to the behavior of the AND gate. Therefore, Theorem 4.2 can be used to express the probability of the HSP as long as the main and spare parts are independent. The CSP and WSP gates require different strategies to handle their probability of failure, as will be explained in this section.

Cold Spare Gate

The probabilistic behavior of the CSP requires dealing with dependent events, as the failure of the main part affects the behavior of the spare part. Therefore, it is required to approach the proof in a different manner.

The failure distribution of the spare part of a *CSP* gate is affected by the failure time of the main part, as the cold spare starts working after the failure of the main part. Hence, the failure distribution of the spare part is dependent on the failure of the main part. The probability of failure for the output event of a CSP with Y as the main part and X as the spare part is given by [30]:

$$Pr(Q_{CSP})(t) = \int_0^t \left(\int_v^t f_{(X_a|Y=v)}(u) du \right) f_Y(v) dv \quad (4.7)$$

where $f_{(X_a|Y=v)}$ is the conditional probability density function for the spare part in its active state (X_a) given that the main part (Y) has failed at time v . As mentioned previously, the subscript of X_a can be omitted, since the spare part of the CSP gate does not work in its dormant state and we are only concerned with the active state, so using X directly with CSP means that we are dealing with the active state and not the dormant one. It can be seen from Equation (4.7) that the failure distribution of the spare part is affected by the failure of the main part. Hence, these two input events are not independent, and we cannot utilize the previously verified relationships to verify the probabilistic behavior of the CSP gate.

We verify Equation (4.7) as:

Theorem 4.8.

$\vdash \forall p \ X \ Y \ f_{xy} \ f_y \ f_{cond} \ t.$

$rv_gt0_ninfinity \ [X; \ Y] \wedge 0 \leq t \wedge$

```

(∀ y.
  cond_density lborel lborel p
  (real o X) (real o Y) y f_xy f_y f_cond) ∧
prob_space p ∧ den_gt0_ninfinity f_xy f_y f_cond ⇒
(prob p (DFT_event p (CSP Y X) t) =
  pos_fn_integral lborel
  (λy.
    indicator_fn {u | 0 ≤ u ∧ u ≤ t} y * f_y y *
    pos_fn_integral lborel
    (λx. indicator_fn {w | y < w ∧ w ≤ t} x * f_cond y x )))

```

where p is the probability space, f_{xy} is the joint density function for X and Y , f_y is the marginal density function for Y , $cond_density$ defines the conditional density function (f_cond) for X given that ($Y = y$) and $den_gt0_ninfinity$ ensures the proper values for the density functions as mentioned in Section 2.2.

Notice that the spare part in the CSP is used without any subscript, i.e., it is used as X , since the spare has only one state in the CSP, which is the active state. Therefore, there is no need to use any subscript to distinguish between the dormant and the active states. As with the previous theorems, we need to use the typecast operator `real` with the random variables, since the random variables are of type `extreal` and the integral over the `lborel` requires real random variables.

In [30], a proof has been introduced for the above expression, which is based mainly on the total expectation theorem [63]. However, we have been able to conduct the same proof in a simpler manner based on conditional density functions as explained below.

Proof Strategy for Theorem 4.8 (CSP Gate)

In order to verify Theorem 4.8, we formalize the conditional density function as [64]:

Definition 4.4.

```

⊢ ∀ M1 M2 p X Y y f_xy f_y f_cond.
  cond_density M1 M2 p X Y y f_xy f_y f_cond ⇔
  random_variable X p (m_space M1, measurable_sets M1) ∧
  random_variable Y p (m_space M2, measurable_sets M2) ∧
  distributed p (pair_measure M1 M2) (λx. (X x, Y x)) f_xy ∧
  distributed p M2 Y f_y ∧ (f_cond y = (λx. f(x,y) / f_y y))

```

where p is the probability space, $M1$ and $M2$ are the measure spaces that the random variables X and Y map to, respectively (we will use `lborel` in our case), f_xy is the joint density function for X and Y , f_y is the marginal density function of Y and finally, f_cond is the conditional density function of X given ($Y = y$).

The conditional density function definition ensures that X and Y are random variables with the joint density function f_xy and the marginal density function f_y . It is noticed from the definition of the conditional density function f_cond that it is a function of x only, and it can have different variants depending on the value of Y that we are conditioning at, i.e., y . This is why f_cond takes y as a parameter.

From Definition 4.4, we formally verify the following relationship between the conditional density, the joint density and the marginal density functions, given that $f_Y(y) \neq 0$:

$$f_{XY}(x, y) = f_{X|Y=y}(x) \times f_Y(y) \quad (4.8)$$

The above equation can be formalized in HOL4 as:

Theorem 4.9.

$$\vdash \forall M1 M2 p X Y f_cond x y f_xy f_y.$$

$$(\forall y. f_y y \neq 0 \wedge f_y y \neq \text{PosInf} \wedge f_y y \neq \text{NegInf}) \wedge$$

$$\text{cond_density } M1 M2 p X Y y f_xy f_y f_cond \Rightarrow$$

$$(f_xy (x,y) = f_cond y x * f_y y)$$

The condition $f_y y \neq 0$ is required, as this function will be used in the denominator of the conditional density and it cannot equal to 0. In addition, since we are dealing with extended-real numbers, $f_y y$ cannot equal infinity. This theorem is applicable to any conditional density function that satisfies the given conditions.

The second step in verifying the expression of the CSP is by verifying that the probability of the joint random variables is equal to the iterated integrals of the joint density function. This can be expressed as:

$$Pr(X, Y)^{-1}(A) = \int \int \mathbf{1}_A \times f_{XY}(x, y) dx dy \quad (4.9)$$

We use Theorem 4.6 to verify this in HOL4 as:

Theorem 4.10.

$$\vdash \forall p X Y f_xy A.$$

$$\text{distributed } p (\text{pair_measure lborel lborel}) (\lambda x. (X x, Y x)) f_xy \wedge$$

$$\text{prob_space } p \wedge (\forall x. 0 \leq f_xy x) \wedge$$

$$A \in \text{measurable_sets } (\text{pair_measure lborel lborel}) \Rightarrow$$

$$(\text{prob } p (\text{PREIMAGE } (\lambda x. (X x, Y x)) A \cap p_space p) =$$

$$\text{pos_fn_integral lborel}$$

$$(\lambda y. \text{pos_fn_integral lborel}$$

$$(\lambda x. \text{indicator_fn } A (x,y) * f_xy (x,y))))$$

Then, we express the probability of the joint random variables using the conditional density function as:

$$Pr(X, Y)^{-1}(A) = \int \int \mathbf{1}_A \times f_{(X|Y=y)}(x) \times f_Y(y) dx dy \quad (4.10)$$

which we verify in HOL4, using Theorems 4.9 and 4.10, as:

Theorem 4.11.

```

⊢ ∀ p X Y f_xy f_y f_cond A.
  (∀ y. cond_density lborel lborel p X Y y f_xy f_y f_cond) ∧
  prob_space p ∧ (∀ x. 0 ≤ f_xy x) ∧
  (∀ y. 0 < f_y y ∧ f_y y ≠ PosInf) ∧
  A ∈ measurable_sets (pair_measure lborel lborel) ⇒
  (prob p (PREIMAGE (λx. (X x, Y x)) A ∩ p_space p) =
   pos_fn_integral lborel
    (λy.
      pos_fn_integral lborel
        (λx. indicator_fn A (x,y) * f_cond y x * f_y y )))

```

In order to be able to reach the final form of Equation (4.7), we need first to express the event set that corresponds to the integration in Equation (4.7) as:

$$(X, Y)^{-1}\{(x, y) \mid y < x \wedge x \leq t \wedge 0 \leq y \wedge y \leq t\} \quad (4.11)$$

We verify in HOL4 that this set corresponds to the `DFT_event` of the CSP gate as:

Lemma 4.11.

```

⊢ ∀ X Y p t.
  rv_gt0_ninfinity [X; Y] ∧ 0 ≤ t ⇒

```

```

(DFT_event p (CSP Y X) t =
  PREIMAGE (\s. (real (X s), real (Y s)))
    {(x,y) | y < x ∧ x ≤ t ∧ 0 ≤ y ∧ y ≤ t} ∩ p_space p)

```

In addition, we verify that the event set in Lemma 4.11 is measurable in `pair_measure lborel lborel`. Finally, we verify that the indicator function of the set in Lemma 4.11 can be expressed as the multiplication of two indicator functions to determine the boundaries of the iterated integrals in Equation (4.7) as:

Lemma 4.12.

$\vdash \forall x y t.$

```

indicator_fn {(w,u) | u < w ∧ w ≤ t ∧ 0 ≤ u ∧ u ≤ t} (x,y) =
indicator_fn {w | y < w ∧ w ≤ t} x *
indicator_fn {u | 0 ≤ u ∧ u ≤ t} y

```

Using all these verified theorems and lemmas, we formally verify Theorem 4.8.

Warm Spare Gate

Similar to the CSP, the failure of the main part of the WSP gate affects the behavior of the spare part. Thus, we need to deal with dependent events.

For the WSP gate with two basic events, the output fails in two cases. Case 1 is when the main part fails, then the spare fails in its active state (this case is similar to the CSP case). Case 2 is when the spare part fails in its dormant state, then the main part fails with no spare to replace it. In the latter case, the failure distribution of the spare part in its dormant state is independent of the main part. Hence, we can use the previously verified expressions for this case. The probability expression for a WSP with X as the spare part (X_a for the active state and X_d for the dormant state) and Y as the main part is expressed as [30]:

$$Pr(Q_{WSP})(t) = \int_0^t \left(\int_v^t f_{(X_a|Y=v)}(u) du \right) f_Y(v) dv + \int_0^t f_Y(u) F_{X_d}(u) du \quad (4.12)$$

where F_{X_d} is the CDF of X in its dormant state. The first part of Equation (4.12) represents the probability of a CSP and the second part represents the probability when the spare fails before the main part. For the second part, Y and X_d are considered to be independent as the failure of one of them does not affect the failure of the second and hence we can use Equation (4.1c) for this case.

We verify Equation (4.12) as:

Theorem 4.12.

$\vdash \forall p \ Y \ X_a \ X_d \ t \ f_y \ f_xy \ f_cond.$

$prob_space \ p \wedge (\forall s. \ ALL_DISTINCT \ [X_a \ s; \ X_d \ s; \ Y \ s]) \wedge$

$DISJOINT_WSP \ Y \ X_a \ X_d \ t \wedge rv_gt0_ninfinitiy \ [X_a; \ X_d; \ Y] \wedge 0 \leq t \wedge$

$(\forall y.$

$\quad cond_density \ lborel \ lborel \ p$

$\quad (real \ o \ X_a)(real \ o \ Y) \ y \ f_xy \ f_y \ f_cond) \wedge$

$\quad den_gt0_ninfinitiy \ f_xy \ f_y \ f_cond \wedge$

$\quad indep_var \ p \ lborel \ (real \ o \ X_d) \ lborel \ (real \ o \ Y) \wedge$

$\quad cont_CDF \ p \ (real \ o \ X_d) \wedge$

$\quad measurable_CDF \ p \ (real \ o \ X_d) \Rightarrow$

$(prob \ p \ (DFT_event \ p \ (WSP \ Y \ X_a \ X_d) \ t) =$

$\quad pos_fn_integral \ lborel$

$\quad (\lambda y.$

$\quad \quad indicator_fn \ \{u \mid 0 \leq u \wedge u \leq t\} \ y \ * \ f_y \ y \ *$

$\quad \quad pos_fn_integral \ lborel$

```

      (λx. indicator_fn {w | y < w ∧ w ≤ t} x * f_cond y x ))+
pos_fn_integral lborel
  (λy. f_y y *
    (indicator_fn {u | 0 ≤ u ∧ u ≤ t} y *
      CDF p (real o X_d) y )))

```

In the WSP, we need to distinguish between the two states, i.e., active and dormant, hence the usage of X_a and X_d . The condition $\text{DISJOINT_WSP } Y \ X_a \ X_d \ t$ indicates that until time t , the spare part X can only fail in one of its states. It is assumed that the spare part in the dormant (X_d) state is independent of the main part Y since the failure of the spare part in its dormant state is not affected by the failure of the main part.

Proof Strategy for Theorem 4.12 (WSP Gate)

For the verification of Theorem 4.12, it is evident that the probability expression involves the probability of the CSP gate in addition to the probability of the *after* expression of Theorem 4.4. Therefore, we choose to verify that the event of the WSP for basic events is equivalent to the union of two sets as:

Lemma 4.13.

$\vdash \forall p \ Y \ X_a \ X_d \ t.$

$$\begin{aligned}
& (\forall s. \ 0 \leq Y \ s) \wedge \\
& (\forall s. \ \text{ALL_DISTINCT } [X_a \ s; X_d \ s; Y \ s]) \Rightarrow \\
& (\text{DFT_event } p \ (\text{WSP } Y \ X_a \ X_d) \ t = \\
& \quad \{s \mid Y \ s < X_a \ s \wedge X_a \ s \leq \text{Normal } t \wedge \\
& \quad \quad 0 \leq Y \ s \wedge Y \ s \leq t\} \cap \text{p_space } p \cup \\
& \quad \{s \mid X_d \ s < Y \ s \wedge Y \ s \leq \text{Normal } t\} \cap \text{p_space } p)
\end{aligned}$$

Then, we verify that the above two sets are disjoint. As a consequence, the probability of the original set is equivalent to the sum of the probabilities of the disjoint sets. Based on this, we verify that the probability of the first set ($\{\mathbf{s} \mid Y \mathbf{s} < X_a \mathbf{s} \wedge X_a \mathbf{s} \leq \text{Normal } \mathbf{t} \wedge 0 \leq Y \mathbf{s} \wedge Y \mathbf{s} \leq \mathbf{t}\} \cap \text{p_space } \mathbf{p}$) is equal to the probability of the CSP gate, which will result in the first term in the addition of the conclusion of Theorem 4.12. We also verify that the probability of the second set in Lemma 4.13 ($\{\mathbf{s} \mid X_d \mathbf{s} < Y \mathbf{s} \wedge Y \mathbf{s} \leq \text{Normal } \mathbf{t}\} \cap \text{p_space } \mathbf{p}$) is expressed using Theorem 4.4, which will result in the second term of the addition of the conclusion of Theorem 4.12. As a result, we have the probability of the WSP as in Theorem 4.12.

In this section, we formally verified the probabilistic behavior of the DFT gates: AND, OR, HSP, FDEP, PAND, CSP and WSP besides the formalization of expressions for $Pr(X < Y \wedge Y \leq t)$ and $Pr(X < Y \wedge X \leq t)$. These verified properties are generic, i.e., universally quantified for all distribution and density functions, and can be used to formally verify the probability of failure expression of any DFT. The HOL4 proof script for this verification as well as the gate definitions is available at [57].

Spare Gates with a Shared Spare

The spare gate with shared spare is formally defined in Chapter 3. It is worth mentioning that the definition in [30] does not allow the simultaneous failures of the main parts and thus we use the same constraint.

Q_1 of the spare gate with a shared spare, shown in Table 3.1, is represented as a sum of disjoint products in order to express its probability. This is accomplished by introducing the complement of an input event to be able to create the disjoint events. Thus Q_1 can be expressed as [65]:

$$\begin{aligned}
Q_1 = & X_a \cdot (Y \triangleleft Z) \cdot (Z \triangleleft X_a) + Z \cdot (Y \triangleleft X_a) \cdot (X_a \triangleleft Z) + X_a \cdot (Y \triangleleft X_a) \cdot \bar{Z} + \\
& Z \cdot (X_d \triangleleft Y) \cdot (Y \triangleleft Z) + Y \cdot (X_d \triangleleft Y) \cdot \bar{Z} + Y \cdot (Z \triangleleft Y)
\end{aligned}
\tag{4.13}$$

where \bar{Z} indicates the event when Z cannot happen. We formally verify this as:

Theorem 4.13.

$\vdash \forall X_a X_d Y Z p t.$

$$\begin{aligned}
& \text{rv_gt0_ninfinity } [X_a; X_d; Y; Z] \wedge \\
& (\forall s. \text{ ALL_DISTINCT } [Y \ s; X_a \ s; X_d \ s; Z \ s]) \wedge \\
& \text{DISJOINT_WSP } Y \ X_a \ X_d \ t \wedge \\
& \text{DISJOINT_WSP } Z \ X_a \ X_d \ t \wedge \\
& (\forall s. ((Z \triangleleft X_d) \cdot (X_d \triangleleft Y)) \ s = \text{NEVER } s) \wedge \\
& (\forall s. ((Y \triangleleft X_d) \cdot (X_d \triangleleft Z)) \ s = \text{NEVER } s) \wedge \\
& (\forall s. ((X_a \triangleleft Y) \cdot (X_a \triangleleft Z)) \ s = \text{NEVER } s) \Rightarrow \\
& (\text{DFT_event } p \ Q_1 \ t = \\
& \text{DFT_event } p \ (X_a \cdot (Y \triangleleft Z) \cdot (Z \triangleleft X_a)) \ t \cup \\
& \text{DFT_event } p \ (Z \cdot (Y \triangleleft X_a) \cdot (X_a \triangleleft Z)) \ t \cup \\
& \text{DFT_event } p \ (X_a \cdot (Y \triangleleft X_a)) \ t \cap \\
& (p_space \ p \ \text{DIFF } \text{DFT_event } p \ Z \ t) \cup \\
& \text{DFT_event } p \ (Z \cdot (X_d \triangleleft Y) \cdot (Y \triangleleft Z)) \ t \cup \\
& \text{DFT_event } p \ (Y \cdot (X_d \triangleleft Y)) \ t \cap \\
& (p_space \ p \ \text{DIFF } \text{DFT_event } p \ Z \ t) \cup \\
& \text{DFT_event } p \ (Y \cdot (Z \triangleleft Y)) \ t)
\end{aligned}$$

The first two conditions ensure that the time of occurrence of any event is always greater than or equal to 0 but not equal to $+\infty$ and are not equal. While the remaining conditions are required to ensure the proper behavior of the spare gates. For instance, the first two conditions mean that until time t , the spare part can fail in either the active or the dormant state. While the last two conditions indicate that the spare part cannot fail after any of the main parts while it is dormant. Since after the failure of one of the main parts, the spare part will be activated (working in the active state) and in case it fails it will be in the active state and not the dormant state. Similarly, the spare part cannot fail in its active state before the failure of both main parts, as it will be in its dormant state.

The difference between the expression in Equation (4.13) and the verified expression in Theorem 4.13 is that we formally verified the DFT event of Equation (4.13) based on the DFT event of the inputs. We decided to deal with the sets of the input events as there is no gate that can exhibit the behavior of Z in the algebraic approach. This is due to the fact that in the algebraic approach, we are dealing with extended-real numbers and it is impossible to implement a NOT gate using extended-reals. This means that instead of ANDing with Z , we intersect with the complement of $\text{DFT_event } p \ Z \ t$, i.e., $\text{space} - \text{DFT_event } p \ Z \ t$ or more formally $p_space \ p \ \text{DIFF } \text{DFT_event } p \ Z \ t$. As a result, instead of verifying Equation (4.13), we verified that the event of the left hand side is equal to the union of the events of the six products on the right hand side, and whenever we encounter $(\cdot \ Z)$ we use $(\cap p_space \ p \ \text{DIFF } \text{DFT_event } p \ Z \ t)$.

Since Q_1 is represented as the sum of disjoint products, the probability of Q_1 is expressed as the sum of the probabilities of the individual products as given in Equation (4.14) [65].

$$\begin{aligned}
Pr(Q_1)(t) = & \int_0^t \left(\int_y^t \left(\int_y^x f_Z(z) dz \right) f_{(X_a|Y=y)}(x) dx \right) f_Y(y) dy + \\
& \int_0^t \left(\int_0^z \left(\int_y^z f_{(X_a|Y=y)}(x) dx \right) f_Y(y) dy \right) f_Z(z) dz + \\
& (1 - F_Z(t)) \int_0^t \left(\int_y^t f_{(X_a|Y=y)}(x) dx \right) f_Y(y) dy + \tag{4.14} \\
& \int_0^t \left(\int_0^z f_Y(y) F_{X_d}(y) dy \right) f_Z(z) dz + \\
& (1 - F_Z(t)) \int_0^t f_Y(y) F_{X_d}(y) dy + \int_0^t f_Y(y) F_Z(y) dy
\end{aligned}$$

We verified that these products are disjoint to be able to sum the individual probabilities. Some of these probabilistic expressions utilize our existing verified expressions for DFT gates, while the rest requires handling three iterated integrals when dealing with conditional density functions in addition to verifying the probability of a complement of a DFT event.

We have been able to verify Equation (4.14), but as the final form of our verified theorem for Equation (4.14) is quite long, we will explain some details about the proof and the theorem here; the complete theorem can be accessed from [57]. Since this theorem combines many previous formalized expressions, it requires the conditions for those expressions, such as having a conditional density of X_a given $Y = y$, having a density function for Y and Z . Also, the CDF of Z is measurable and continuous, besides the obvious conditions such as $0 \leq t$. The proof of the first of the six terms in Equation (4.14) is quite similar to the proof of the CSP gate. However, in this case, we are dealing with three iterated integrals which makes things a bit complex, since each time we need to prove that the single integral and the double iterated integrals are measurable. In addition, the independence of the random variables that correspond to the input events should be handled appropriately, i.e., Z should be

independent of the joint random variables of (Y, X_a) . The proof of the second term is conducted in a similar way to the first term since it consists of three iterated integrals with conditional density function. However, the density function lies this time in the inner integral. The proof of the third term is primarily based on proving that $Pr(\text{p_space } p \text{ DIFF DFT_event } p \text{ Z } t) = 1 - F_Z(t)$. The proof of the fifth term also requires the same result. The fourth term corresponds to finding the probability of a cascaded PAND gate for three inputs (this will be explained in a following section). Finally, the last term corresponds to the probability of the after event.

4.3 Formal Quantitative Analysis of DFT

Examples

In this section, we conduct the quantitative analysis of the DFT examples presented in Section 3.5. We verify the probability of failure for the CPAND as in Theorem 4.14. The verification steps are similar to the *after* event. However, we are dealing now with three inputs instead of two. Hence, X is assumed to be independent of the joint random variable (Y, Z) using `indep_CPAND` in Theorem 4.14, where it is defined for X over the `lborel` measure and (Y, Z) over the two dimensional `lborel`.

Theorem 4.14. *Probability of CPAND*

$\vdash \forall p \ X \ Y \ Z \ t \ f_y \ f_x.$

$$\begin{aligned} & \text{prob_space } p \wedge 0 \leq t \wedge \text{indep_CPAND } X \ Y \ Z \ p \wedge \\ & \text{rv_gt0_ninfinity } [X; Y; Z] \wedge (\forall s. \text{ALL_DISTINCT } [X \ s; Y \ s; Z \ s]) \wedge \\ & \text{distributed } p \ \text{lborel } (\lambda x. \text{real } (X \ x)) \ f_x \wedge \\ & \text{distributed } p \ \text{lborel } (\lambda x. \text{real } (Y \ x)) \ f_y \wedge \\ & (\forall y. \ 0 \leq f_y(y)) \wedge \end{aligned}$$

$$(\forall x. 0 \leq f_x(x)) \wedge \text{cont_CDF } p (\lambda x. \text{ real } (Z \ x)) \ x)) \Rightarrow$$

$$(\text{prob } p (\text{DFT_event } p (Q_1) \ t) = \int_0^t (\int_0^x f_Y(y) F_Z(y) \ dy) f_X(x) \ dx)$$

We verify the probability of failure of AND-FDEP DFT as:

Theorem 4.15. *Probability of AND-FDEP*

$\vdash \forall X \ Y \ Z \ p \ t.$

$$\text{rv_gt0_ninfinity } [X; Y; Z] \wedge \text{All_distinct_events } p [X \cdot Y; X \cdot Z] \ t \wedge$$

$$\text{indep_vars3 } X \ Y \ Z \ p \Rightarrow$$

$$(\text{prob } p (\text{DFT_event } p (Q_2) \ t) =$$

$$F_X(t) \times F_Y(t) + F_X(t) \times F_Z(t) - F_X(t) \times F_Y(t) \times F_Z(t))$$

The main idea of this proof is to replace the FDEP gate by an OR gate as they are equivalent. Then, the probability of the union of two events, each of which is the intersection of two basic events, $\{X; Y\}$ and $\{X; Z\}$ is verified. These two events represent the cut sets of the AND-FDEP DFT. For this proof, it is required to ensure that the random variables are independent using `indep_vars3` and that the events of the two cut sets of the DFT are not equal using `All_distinct_events`.

Finally, we verify the probability of the top event of the WSP-OR DFT as in Theorem 4.16. The top event is composed of the union of the WSP event and the basic event Z . Hence, the final form of the probability is the probability of the union of two events; the WSP and Z . Therefore, it is required to include the conditions needed for expressing the probability of the WSP event in the list of assumptions, and ensure that the WSP event is independent of event Z using `indep_var_set_WOR`.

Theorem 4.16. *Probability of WSP-OR*

$\vdash \forall Y \ X_a \ X_d \ Z \ p \ t \ f_{xy} \ f_y \ f_{X_a|Y}.$

$$\text{DISJOINT_WSP } Y \ X_a \ X_d \ t \wedge$$

$$\begin{aligned}
& (\forall s. \text{ ALL_DISTINCT } [Y \ s; X_a \ s; X_d \ s; Z \ s]) \wedge \\
& \text{All_distinct_events } p \ [WSP \ Y \ X_a \ X_d; Z] \ t \wedge \\
& \text{rv_gt0_ninfinity } [Y; X_d; X_a; Z] \wedge 0 \leq t \wedge \\
& (\forall y. \text{ cond_density lborel lborel } p \\
& \quad (\text{real } o \ X_a) \ (\text{real } o \ Y) \ y \ f_{xy} \ f_y \ f_{x_a|y}) \wedge \\
& \text{den_gt0_ninfinity } f_{xy} \ f_y \ f_{x_a|y} \wedge \text{cont_CDF } p \ (\text{real } o \ X_d) \wedge \\
& \text{measurable_CDF } p \ (\text{real } o \ X_d) \wedge \\
& \text{indep_var_set_WOR } Y \ X_a \ X_d \ Z \ p \ t \Rightarrow \\
& (\text{prob } p \ (\text{DFT_event } p \ (Q_3) \ t) = \\
& \quad \int_0^t \left(\int_y^t f_{(X_a|Y=y)}(x) \ dx \right) f_Y(y) \ dy + \int_0^t f_Y(y) F_{X_d}(y) \ dy + F_Z(t) - \\
& \quad \left(\int_0^t \left(\int_y^t f_{(X_a|Y=y)}(x) \ dx \right) f_Y(y) \ dy + \int_0^t f_Y(y) F_{X_d}(y) \ dy \right) \times F_Z(t)
\end{aligned}$$

After having the verified generic expressions for the probability of failure of the three examples, these expressions can be used to evaluate the probability of failure for any integrable distribution functions that represent the failure distribution of the system components. For example, assuming exponential distributions for the inputs with failure rates: 2×10^{-2} , 3×10^{-3} , and 1×10^{-2} for X , Y and Z , respectively, we can evaluate the probability of failure using MATLAB [62] until 400 working hours with a dormancy factor of 0.1. The results are shown in Figure 4.2.

4.4 Formal Quantitative Analysis Case Studies

In this section, we present the verification details of the formal quantitative analysis of the DBW and CAS systems.

We provide generic steps that can be followed in order to use our DFT formalization to conduct the formal quantitative analysis of DFTs in the form of generic expressions of failure probabilities. These steps are:

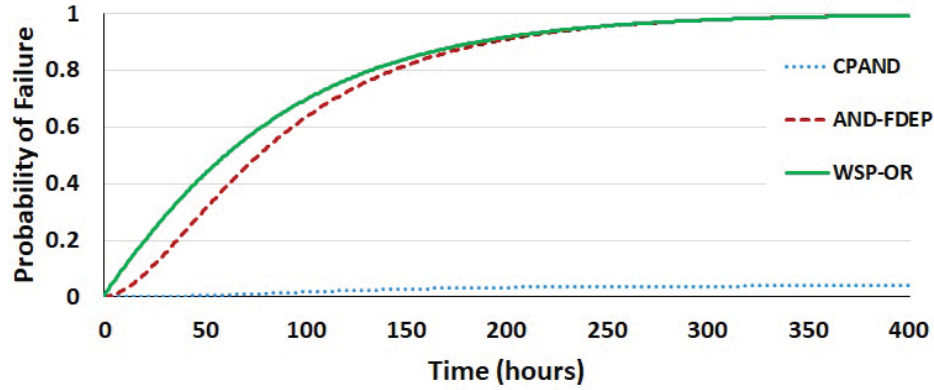


Figure 4.2: Probability of Failure of CPAND, AND-FDEP and WSP-OR

1. Determine the structure function of the top event of the DFT.
2. Simplify the structure function and formally verify that the simplified version is equal to the original function obtained in step (1).
3. Create the `DFT_event` of the structure function.
4. Express the `DFT_event` of the top event as the union of multiple input events.
5. Apply the probabilistic PIE to the union of events generated in the previous step, then simplify the result of the PIE. This will result in having the summation of the probabilities of the intersection of the different events that contribute to the failure of the top event of the DFT.
6. Replace each term in the result of the PIE by its probabilistic expression based on the verified expressions in Section 4.2 for each gate and operator.

Step (5) requires proving many lemmas that are necessary for manipulating the result of the PIE. In order to facilitate the analysis, we verified several generic properties that can be used to reduce the manual interaction of the reliability engineer in the theorem proving related tasks. For example, for any group of independent random variables, we verified that the probability of the preimage of any two random variables out of the original set equals to the multiplication of the individual probabilities as:

Theorem 4.17.

$$\begin{aligned} & \vdash \forall p \ M \ X \ ii \ A \ s \ t. \ s \neq t \wedge \text{prob_space } p \wedge \\ & \quad \text{indep_vars } p \ M \ X \ ii \wedge \{s; t\} \subset ii \wedge \\ & \quad (\forall i. \ i \in \{s; t\} \Rightarrow A \ i \in \text{measurable_sets } (M \ i)) \Rightarrow \\ & \quad (\text{prob } p \\ & \quad \quad (\text{PREIMAGE } (X \ s) \ (A \ s) \cap \text{p_space } p \cap \\ & \quad \quad (\text{PREIMAGE } (X \ t) \ (A \ t) \cap \text{p_space } p)) = \\ & \quad \text{prob } p \ (\text{PREIMAGE } (X \ s) \ (A \ s) \cap \text{p_space } p) * \\ & \quad \text{prob } p \ (\text{PREIMAGE } (X \ t) \ (A \ t) \cap \text{p_space } p)) \end{aligned}$$

The formal DFT analysis now requires proving the required conditions for this property to hold only. As an example, consider that we have a group of 10 random variables, and we need to prove that the probability of the preimages of the 6th and the 8th random variables equals the multiplication of their individual probabilities. Therefore, in Theorem 4.17, $s = 6$, $t = 8$ and ii equals the set of numbers from 0 – 9. We just need to verify the following properties for this proof:

- $6 \neq 8$.
- $\{6; 8\} \subset \{0; 1; 2; 3; 4; 5; 6; 7; 8; 9\}$.
- The sets of the preimages are measurable.

These requirements can be easily verified using various built-in arithmetic tactics in HOL4. Similarly, we verified the same property for up to 10 random variables out of a group of independent random variables. These properties are very helpful in the verification process of the probabilistic analysis of DFTs, in particular when applying the probabilistic PIE. We also verified several additional properties that allow the direct usage of the PIE in its final form with a system that can be represented as

the union of six elements as the behavior of both case studies can be represented as the union of six events. However, our formalization can be extended easily to verify larger systems, as the flow of the proofs will remain the same but will extend to a larger number of inputs. We illustrate the utilization of the previous steps to perform the formal DFT analysis of the DBW and CAS to provide generic expressions for the probability of failure of the top events.

4.4.1 Formal Quantitative Analysis of DBW

In order to perform the formal quantitative analysis of the DBW system, we use the verified reduced DFT model of this system of Section 3.6.1 We choose to use a single event for the WSP as this will reduce the intermediate steps required to reach our final goal for the probabilistic expression and would result in expressing the top event as the union of six events. We verify that the `DFT_event` of the DBW is equal to the union of six events as:

Lemma 4.14. *DBW Union of Events*

$\vdash \forall BS TS PC SC_a SC_d BC EF TF p t.$

`DFT_event p ((TF + EF) + WSP PC SC_a SC_d + BC + (TS + BS)) t =`

`union_list`

`[DFT_event p TF t; DFT_event p EF t; DFT_event p (WSP PC SC_a SC_d) t;`

`DFT_event p BC t; DFT_event p TS t; DFT_event p BS t]`

We apply the probabilistic PIE to perform the formal quantitative analysis of the top event, by incorporating the existing verified properties. We verify the probabilistic failure expression of the DBW as Theorem 4.18. In the following, we are presenting the formalization in mixed formal and standard math notations to make the results more readable.

Theorem 4.18. *Probability of Failure of DBW*

$\vdash \forall BS TS PC SC_a SC_d BC EF TF p t$

$f_{PC} f_{(SC_a|PC)} f_{SC_aPC} \cdot 0 \leq t \wedge$

$All_distinct_events p [TF; EF; BC; WSP PC SC_a SC_d; BS; TS] t \wedge$

$rv_gt0_ninfinity [BS; TS; PC; SC_a; SC_d; BC; EF; TF] \wedge$

$DISJOINT_WSP PC SC_a SC_d t \wedge$

$(\forall y. cond_density lborel lborel p$

$(real \circ SC_a) (real \circ PC) y f_{SC_aPC} f_{PC} f_{(SC_a|PC)}) \wedge$

$den_gt0_ninfinity f_{SC_aPC} f_{PC} f_{(SC_a|PC)} \wedge cont_CDF p (real \circ SC_d) \wedge$

$measurable_CDF p (real \circ SC_d) \wedge$

$indep_vars_sets_drive [BS; TS; PC; SC_a; SC_d; BC; EF; TF] p t \Rightarrow$

$(prob p (DFT_event p Q_{DBW} t) =$

$$F_{TF}(t) + F_{EF}(t) + F_{BC}(t) + \int_0^t f_{PC}(pc) \times \int_{pc}^t f_{(SC_a|PC=pc)}(sc_a) dsc_a dpc +$$

$$\int_0^t f_{PC}(pc) \times F_{SC_d}(pc) dpc + F_{BS}(t) + F_{TS}(t) - \dots + \dots -$$

$F_{TF}(t) \times F_{EF}(t) \times F_{BC}(t) \times$

$$\left[\left(\int_0^t f_{PC}(pc) \times \left(\int_{pc}^t f_{(SC_a|PC=pc)}(sc_a) dsc_a \right) dpc \right) + \right.$$

$$\left. \int_0^t f_{PC}(pc) \times F_{SC_d}(pc) dpc \right] \times F_{BS}(t) \times F_{TS}(t)$$

where `All_distinct_events` ensures that all event sets are distinct. As listed earlier, since the events of the WSP are disjoint, we used the WSP event directly to reduce the proof steps. It is necessary that all random variables representing the input events to be positive or equal to 0, since they represent the time of failure. This condition is added by `rv_gt0_ninfinity`. It is also required to ensure the proper behavior of the WSP by adding the condition `DISJOINT_WSP PC SC_a SC_d t`, which ascertains that the events of the WSP are disjoint, i.e., until time t , the spare part can fail in one of

Table 4.1: Failure Rates for the DBW System ($\times 10^{-7}$)

TF	EF	BC	PC	SC	TS	BS
1	4	5	2	3	1	2

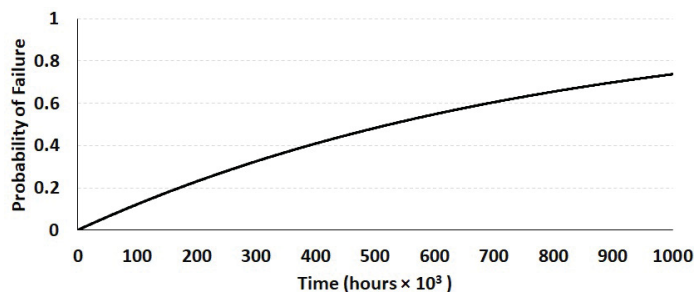


Figure 4.3: Probability of Failure of the Drive-by-wire System

its states only. A conditional density $f_{(SC_a|PC)}$ of SC_a given that $PC = pc$ is defined using `cond_density`. The function `indep_vars_sets_drive` adds the condition that the input events and their sets are independent, and finally we need to ensure that the CDF of SC_d is continuous and measurable. It is worth mentioning that since the union list of the DBW system has six events, applying the PIE results in the generation of 63 different terms, and a truncated version of the final expression is given above.

The reliability engineer working on the analysis of the DBW system just needs to ensure that the mentioned conditions hold in order to use the results of the analysis. After formally ensuring that the probability of failure expression is correct, this expression can be used to evaluate the probability of failure using any tool with any distribution and density functions that satisfy the listed conditions. Assuming exponential distributions for the inputs with failure rates as listed in Table 4.1 [59], we evaluated the probability of failure using MATLAB until 1,000,000 working hours with dormancy factor of 0.5, as shown in Figure 4.3. The proof script for the DBW system is around 4950 lines long.

4.4.2 Formal Quantitative Analysis of CAS

In a similar way to the DBW system, we use the formally verified reduced DFT model of the CAS in Section 3.6.2 to perform the probabilistic analysis. We verify that the `DFT_event` of the CAS equals the union of events as:

Lemma 4.15. *CAS Union of Events*

$\vdash \forall PA PB PS MS MA MB CS SS P B_a B_d p t.$

`DFT_event p`

$(CS + SS + MA \cdot (MS \triangleleft MA) + MB \cdot (MA \triangleleft MB) +$
 $B_a \cdot (P \triangleleft B_a) + P \cdot (B_d \triangleleft P) + PA \cdot PB \cdot PS) t =$

`union_list`

`[DFT_event p CS t; DFT_event p SS t; DFT_event p (MA \cdot (MS \triangleleft MA)) t;`

`DFT_event p (MB \cdot (MA \triangleleft MB)) t;`

`DFT_event p (B_a \cdot (P \triangleleft B_a) + P \cdot (B_d \triangleleft P)) t;`

`DFT_event p (PA \cdot PB \cdot PS) t]`

Verifying a generic expression of the probability of failure for the CAS requires dealing with different conditions of independence for the input events, where we considered different configurations for the spare gates in the CAS from [61]. In particular, the outputs of the PAND and the CSP gates are no longer independent because of having *MA* in common. Therefore, it is required to use conditional probabilities to verify the probability of intersection that results from applying the probabilistic PIE. We verify the probability of failure of this system in HOL4:

Theorem 4.19. *Probability of Failure of CAS*

$\vdash \forall CS SS MA MS MB P B_a B_d PA PB PS p t f_{MA} f_{B_a P} f_P f_{B_a | P} f_{MBMA} f_{MB | MA} f_{MS}.$

$0 \leq t \wedge \text{prob_space } p \wedge (B_a \triangleleft P = \text{NEVER}) \wedge \text{DISJOINT_WSP } P B_a B_d t \wedge$

$$\begin{aligned}
& \text{ALL_DISTINCT_RVg } [PA; PB; PS; MS; MA; MB; CS; SS; P; B_a; B_d] \text{ p } t \wedge \\
& \text{indep_vars_setsg } [PA; PB; PS; MS; MA; MB; CS; SS; P; B_a; B_d] \text{ p } t \wedge \\
& (\forall y. \text{ cond_density lborel lborel p} \\
& \quad (\text{real o } B_a) (\text{real o } P) y f_{B_a|P} f_P f_{B_a|P}) \wedge \\
& (\forall y. \text{ cond_density lborel lborel p} \\
& \quad (\text{real o } MB) (\text{real o } MA) y f_{MB|MA} f_{MA} f_{MB|MA}) \wedge \\
& \text{den_gt0_ninfinity } f_{B_a|P} f_P f_{B_a|P} \wedge \text{den_gt0_ninfinity } f_{MB|MA} f_{MA} f_{MB|MA} \wedge \\
& \text{cont_CDF p (real o } B_d) \wedge \text{measurable_CDF p (real o } B_d) \wedge \\
& (\forall z. 0 \leq f_{MS}(z)) \wedge (\forall x. f_{MB|MA}(x) \neq \text{PosInf}) \wedge \\
& \text{distributed p lborel } (\lambda x. \text{ real (MS } x)) f_{MS} \wedge \\
& \text{cont_CDF p (real o MS)} \wedge \text{measurable_CDF p (real o MS)} \Rightarrow \\
& (\text{prob p (DFT_event p } Q_{CAS} \text{ t)} = \\
& F_{CS}(t) + F_{SS}(t) + \int_0^t f_{MA}(ma) \times F_{MS}(ma) dma + \\
& \int_0^t f_{MA}(ma) \times \left(\int_{ma}^t f_{MB|MA=ma}(mb) dmb \right) dma + \\
& \left(\int_0^t f_P(pp) \times \left(\int_{pp}^t f_{B_a|P=pp}(b_a) db_a \right) dpp + \int_0^t f_P(pp) \times F_{B_d}(pp) dpp \right) + \\
& F_{PA}(t) \times F_{PB}(t) \times F_{PS}(t) - \dots + \dots - \\
& F_{CS} \times F_{SS} \times \int_0^t f_{MA}(ma) \times F_{MS}(ma) \times \left(\int_{ma}^t f_{MB|MA=ma}(mb) dmb \right) \times \\
& \left[\int_0^t f_P(pp) \times \left(\int_{pp}^t f_{B_a|P=pp}(b_a) db_a \right) dpp + \int_0^t f_P(pp) \times F_{B_d}(pp) dpp \right] \times \\
& F_{PA}(t) \times F_{PB}(t) \times F_{PS}(t)
\end{aligned}$$

where $(B_a \triangleleft P = \text{NEVER}) \wedge \text{DISJOINT_WSP } P \ B_a \ B_d \ t$ are required to ensure that the spare part B_a cannot fail before the main part P and that the events of the WSP are disjoint, i.e., until time t , the spare part can fail in either the dormant or the active states. `ALL_DISTINCT_RVg` is a predicate required to assert that the inputs and their

event sets are not equal and that the inputs are greater than or equal to 0 but not equal to $+\infty$. `indep_vars_setsg` ensures the independence of the random variables and the event sets. It is also required to define conditional density functions for $f_{B_a|P}$ and $f_{MB|MA}$ using `cond_density`. `den_gt0_ninfinity` ensures the proper values for the joint, marginal and conditional density functions that are used with `cond_density`. For example, the conditional density functions cannot be equal to 0. In addition, the density functions cannot equal $+\infty$. It is also required to ensure that the CDFs of random variables B_a and MS are continuous and measurable using `cont_CDF` and `measurable_CDF`, respectively. `distributed p lborel (\lambda x. real (MS x)) f_MS` is used to indicate that MS has a density function f_{MS} . It is worth mentioning again that the usage of the function `real` is required here as the random variables return `extreal`, while they need to be used with the Lebesgue-Borel measure, which is defined over the real line. The first six elements of the conclusion of Theorem 4.19 represent the probability of the individual terms of the union list of Lemma 4.15, which result from applying the probabilistic PIE. While the rest of the elements represent the probability of the intersection of all combinations of the events. The last term represents the probability of the intersection of the six elements of the CAS.

As with the DBW system, we assume exponential distributions for the inputs of the cardiac assist system with the failure rates listed in Table 4.2 [65]. We evaluated the probability of failure for this generic expression using MATLAB with a dormancy factor of 0.5 for the spare part MB until 400,000 working hours, as shown in Figure 4.4.

We have illustrated in this section the application of our proposed methodology to conduct the formal failure analysis of the DBW and the CAS systems. We have created the HOL formal DFT models for these systems and verified a reduced form of the structure functions utilizing the verified simplification theorems. We then

Table 4.2: Failure Rates of CAS ($\times 10^{-6}$)

CS	SS	P	B	MS	MA	MB	PA	PS	PB
1	2	4	4	1	5	5	5	5	5

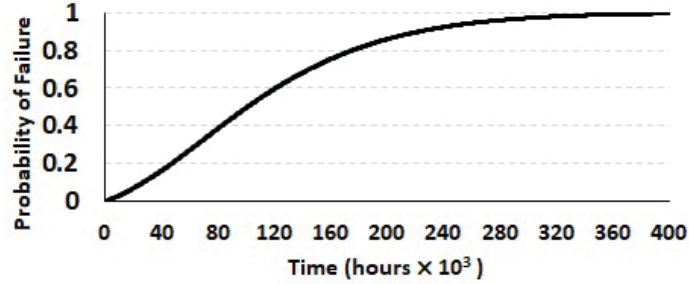


Figure 4.4: Probability of Failure of the Cardiac Assist System

conducted the qualitative and the probabilistic analyses to generate formally verified expressions of probability of failure. Building upon the expressive and sound nature of HOL theorem proving, generic intermediate lemmas are verified that are valid for the analysis of systems similar to the DBW and the CAS systems. Leveraging upon the current formalization of DFTs, the existing lemmas and theorems can be extended to analyze more complex systems. In addition, the results obtained using our methodology, in particular the generic expressions, cannot be obtained formally using a PMC. Moreover, our proposed methodology overcomes the vulnerability of the paper-and-pencil analysis results due to human errors, as it inherits the soundness of HOL theorem proving. Although, providing the formalization of this methodology is costly in terms of time and lines of script, the results obtained are usable by the reliability engineer without the need to go through all the steps of the formalization. The reliability engineer only needs to use the results of the theorems after ensuring that all the required conditions hold, which provides him/her with a formal proof that the analysis results can apply to his system if the conditions are met.

4.5 Summary

In this chapter, we provided a methodology to conduct the formal quantitative analysis of DFTs within a theorem prover. Furthermore, we explained the verification steps of the probabilistic failure expressions of DFT gates that are required in the analysis process. We applied our methodology to perform the quantitative analysis of the DBW and CAS systems to obtain generic expressions of probability of failure.

In the following, we summarize the main challenges that we faced during our formalization of the DFT gates, which allows us to formally analyze DFTs in a theorem prover.

The first challenge is resolving the data-types issue. The problem in the data-types is that the gates and operators are defined as functions that return `extreal`. This is mainly required because we need to model $+\infty$ that represents the NEVER condition. However, this data-type cannot be used to represent random variables over the `lborel` measure. Any random variable defined from a probability space to the `lborel` measure should return `real` data-type. This is required because we need to integrate the density and distribution functions over the real line. Therefore, we need random variables that return `extreal` to model the gates but at the same return `real` to be used with `lborel`. We resolved this issue by using `extreal` to model the gates, but when conducting the probabilistic analysis, we use the real version of the random variable (`real o X`).

Secondly, after modeling the DFT and expressing the structure function of the top event using the DFT gates and operators, it is required to conduct the probabilistic failure analysis of the top event. However, the structure function cannot be used directly since it is a time-to-failure function, not a set. Furthermore, in [30], there is no

clear information on how to create the DFT event and link it to the structure function of the DFT top event or any other event in the fault tree. Using our formalization, we have been able to clearly and formally define a `DFT_event` that is used to create the set of moments of time until the time of failure t , as explained in Definition 4.1.

Thirdly, the probabilities of the AND and OR gates are directly presented in [30] as the probability of the intersection and union (Equations (4.1a) and (4.1b), respectively). However, the AND and the OR gates are defined using the maximum and minimum of their input operands, respectively. There is no information in [30] on how the AND and OR gates are related to the intersection and union of the input events. Using our formalization, we have been able to verify the relationship between the AND and the interaction of the input events utilizing our defined `DFT_event`. In a similar way, we verified the relationship between the OR gate and the union of the input events.

Another contribution is represented by introducing a formal proof in a theorem prover for the probability of failure of the PAND and Before operator, which are represented by $Pr(X < Y)$ in both forms, i.e., $Pr(X < Y \wedge Y \leq t)$ and $Pr(X < Y \wedge X \leq t)$. As mentioned earlier, the first proof of these ($Pr(X < Y \wedge Y \leq t)$) is not provided in [30], while the second one ($Pr(X < Y \wedge X \leq t)$) is presented in a different manner that involves derivatives. In our formalization, we presented, for the first time, the formal proof for $Pr(X < Y)$ in both its formats, i.e., $Pr(X < Y \wedge Y \leq t)$ that represents the probability of the PAND gate for basic events; and $Pr(X < Y \wedge X \leq t)$ that represents the probability of the before operator. In addition, we presented a formal proof for the probability of the *WSP* and *CSP* gates based on conditional density functions, which we defined, while the proof of these gates is presented in [30] based on the law of total expectation.

Finally, while performing all of these formalizations and proofs in HOL, we identified several missing assumptions or conditions that were required to ensure the correctness of the theorems. For example, ensuring the proper values for the input random variables that represent the time-to-failure functions of the system components. These important assumptions were either unavailable in [30] or are not explicitly presented as a requirement in the final form of the theorems in [30].

It is important to highlight that the main benefit of having the formalization of DFT in higher-order logic is that it enables conducting the formal DFT analysis within the sound environment of a theorem prover, which is very useful in the context of safety-critical systems.

Chapter 5

Formal Verification of DFT

Rewrite Rules

As mentioned in Chapter 1, probabilistic model checkers, such as STORM, have been widely used for the probabilistic analysis of DFTs via Markov chains. For example, STORM supports the analysis of DFTs, among other probabilistic models, and allows the verification of the probability of failure and the MTTF of the top event of a given DFT. The scalability of this analysis can be significantly improved by using efficient DFT rewriting rules, as presented by Junges *et al.* [66], that facilitate simplifying a DFT before analysis. The simplification of the DFT is achieved by transforming the underlying graph of the DFT according to the rewrite rules. Experimental evaluation in [66] showed that rewriting heavily improves the performance of the DFT analysis. For example, while originally 68% of the 183 DFTs in [66] could be solved within 2 hours, applying the rewriting beforehand allowed one to solve 95% of the DFTs. Moreover, the total analysis time was reduced from 41 to 18 hours when using rewriting. Simplifying DFTs by rewriting enables the analysis of DFTs that could not be

analyzed before, and can lead to speed-ups and memory savings of up to two orders of magnitude [66].

The rewrite rules are generic for n -ary gates and can be implemented in any tool that supports DFT analysis. Proving the correctness of the rewrite rules as done in [67] is a manual and error-prone process. To the best of our knowledge, a rigorous, mechanically checkable proof of correctness of these rewriting rules has not been done. Thus, their usage in a formal analysis raises soundness concerns especially when dealing with the analysis of safety-critical systems, like transportation or healthcare.

In this chapter, we propose to use our HOL DFT formalization presented in Chapter 3, including the simplification theorems, to verify the DFT rewriting rules of [66] using the HOL4 theorem prover. This requires extending the DFT gates definitions for an arbitrary number of inputs and defining the VOT gate. Verifying these rewrite rules provides the assurance of their correctness and thus adds the confidence to tools that exploit these rules in their DFT analysis.

5.1 DFT Rewrite Rules

In the following, we recap the rewrite rules for DFTs as in [66]. The simplification of DFTs is performed by graph rewriting [68] on the underlying graph of the DFT. We represent a DFT as a labeled graph by extending the induced graph with labels encoding the type of the DFT element and the ordering of the inputs. The graph transformation on the labeled graph is performed by applying a chain of rewrite rules.

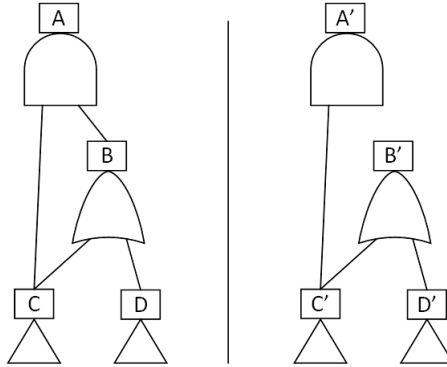


Figure 5.1: Subsumption of OR Gates by AND Gates [66, Rewrite Rule 8]

5.1.1 Rewrite Framework

A rewrite rule is specified by two (sub-) DFTs: the left-hand side capturing the (sub-) DFT before applying the rewrite rule and the right-hand side depicting the resulting (sub-)DFT after the graph rewrite. An example of a rewrite rule is given in Figure 5.1. The rule depicts the subsumption of OR gates by AND gates.

A rewrite rule can be applied whenever a (sub-)DFT can be matched with the left-hand side of the rule. Elements represented by a triangle in the rewrite rule match every gate type. Matched elements might have additional ingoing and outgoing edges not matched by the rewrite rule. These edges are retained during the rewriting step. Applying a rewrite rule replaces the matched part with the right-hand side of the rule. All non-matched parts remain unchanged during the rewriting step. Note that in general, rewrite rules might lead to inconsistent graphs with dangling edges or DFTs that are no longer well-formed (e.g., cyclic DFTs). In these cases, the rewrite rule cannot be applied. It is important to note also that most of the rewrite rules can also be applied from right to left.

An example application of the given subsumption rule is depicted in Figure 5.2. Figure 5.2(a) depicts the original DFT used as input. The subsumption rule from

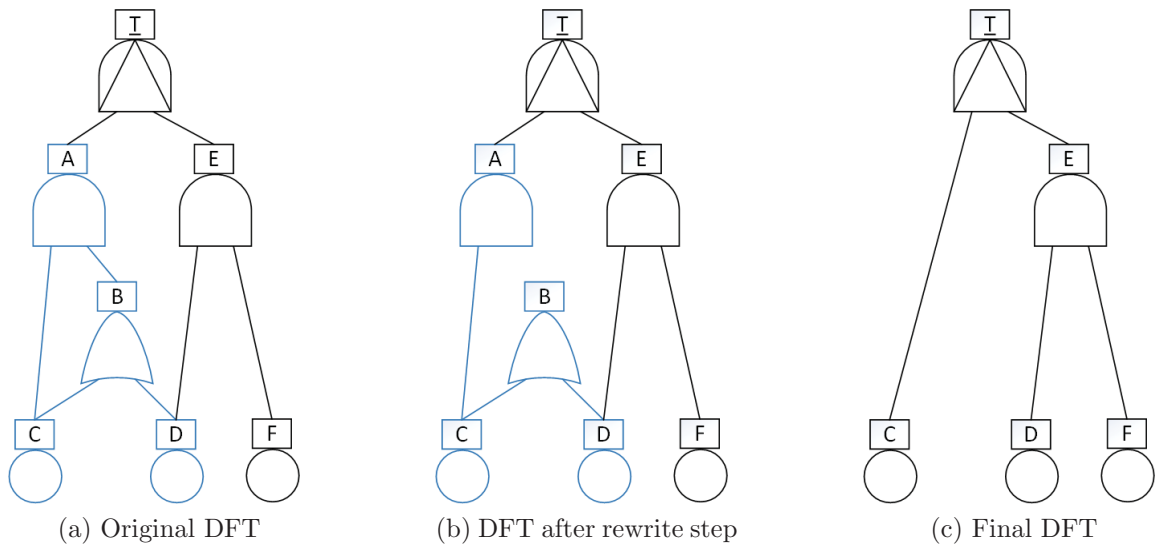


Figure 5.2: Example Application of Rewrite Rule

Figure 5.1 can be applied and the matched sub-DFT is highlighted in blue. Applying the rule removes the connection between AND gate A and OR gate B and yields the rewritten DFT in Figure 5.2(b). Further simplification by applying additional rewrite rules results in the final DFT in Figure 5.2(c). Using the rewrite rules leads to a simpler DFT, which is considerably smaller—and easier to understand.

During rewriting multiple rules might be applicable for the current DFT or different sub-DFTs match the left-hand side of a rewrite rule. The sequence of rewrite steps is chosen by a rewrite strategy. As the rewrite framework is not confluent, the strategy heavily influences the size of the resulting DFTs and a heuristic approach is used.

5.1.2 Rewrite Rules

In the following, we consider 22 rules of the 29 rewrite rules given in [66]. Of the remaining 7 rules, one rule gives the Shannon expansion for VOT_k gates, which deals

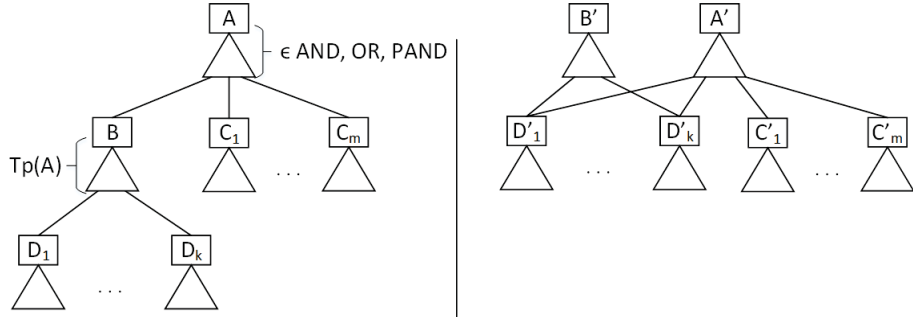


Figure 5.3: Left-flattening of Gates [66, Rewrite Rule 5]

with variables as Boolean, whereas generally DFTs, as formalized in HOL, treat variables as real numbers representing time-to-failure functions. The other 6 rules apply to FDEPs and spares; both gate types are not considered in this chapter. We recap a selection of the rewrite rules and use the same rule enumeration as in [66, Sect. 5.3].

General Rewrite Rules

The first rewrite rules 1-7 consider structural identities such as commutativity of static gates, removal of gates with a single successor or no predecessor, and left-flattening of gates. As an example, the rewrite rule for left-flattening is given in Figure 5.3. The rule can only be applied if the top element of the (sub-)DFT is an AND, OR or PAND gate, and the first input is of the same gate type as the top element ($Tp(B) = Tp(A)$). Applying the left-flattening rule adds the inputs of B as first inputs of A . Gate B is not removed as it might still have connections to other parts of the DFT.

Rules 8-10 capture standard axioms from Boolean algebra on the static gates such as subsumption of OR gates by AND gates (cf. Figure 5.1).

DFTs containing constant failed $CONST(\top)$ or constant fail-safe $CONST(\perp)$ events can lead to large simplifications as often complete sub-DFTs can be evaluated to constant. Rules 11-14 specifically consider constant elements and as an example, we present the rewrite rule for AND/PAND gates with $CONST(\perp)$ inputs in Figure 5.4.

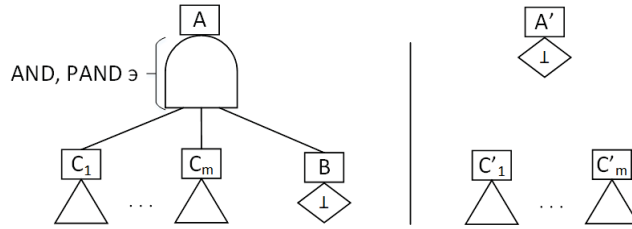


Figure 5.4: AND/PAND Gate with $\text{CONST}(\perp)$ Successor [66, Rewrite Rule 13]

If at least one of the inputs of an AND/PAND gate is fail-safe, it is impossible for the gate to fail and therefore it can be set to fail-safe as well.

Encoding of VOT gates by OR/AND gates is given in the rewrite rules 15-16.

Rewrite Rules for PAND gates

So far, the rewrite rules mostly captured simplifications of static gates, which are based on the corresponding properties in Boolean algebra. The remaining rules 18-23 consider PAND gates where the order of failures is crucial. As an example, consider the rewrite rule for conflicting PAND gates with independent successors in Figure 5.5. PAND gate D_1 requires that input B fails strictly before C or simultaneously with C . If C fails strictly before B , D_1 becomes fail-safe. D_2 requires the opposite behavior. If both elements B and C are independent, they will not fail simultaneously. Thus,

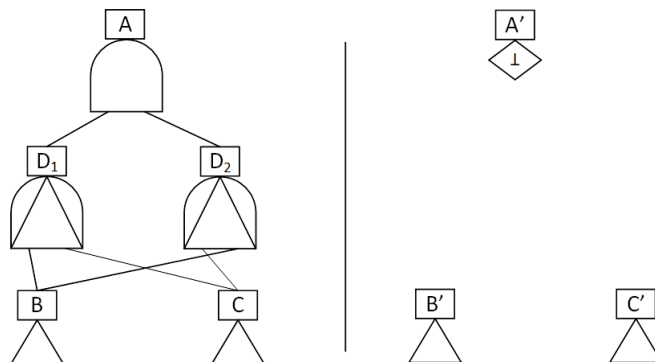


Figure 5.5: Conflicting PAND Gates with Independent Successors [66, Rewrite Rule 19]

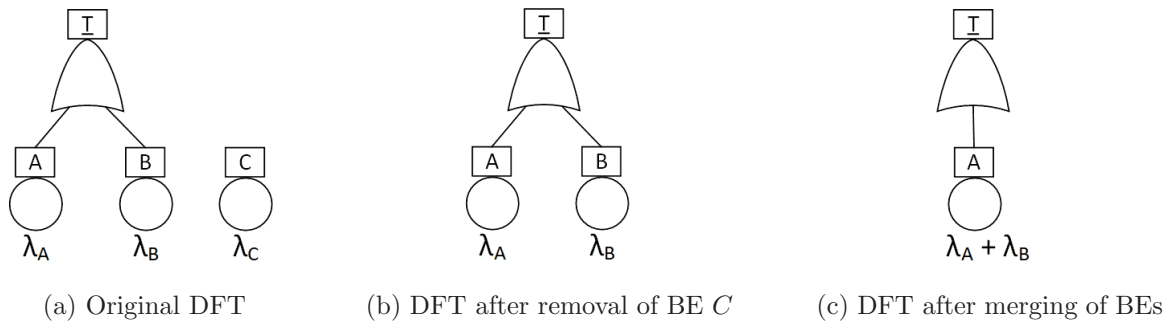


Figure 5.6: Example Application of Non-structural Rules

either PAND gate D_1 or D_2 will become fail-safe. As the PAND gates can never both fail, A is fail-safe and can be replaced by $\text{CONST}(\perp)$.

Note that the rewrite rule can only be applied if B and C are independent—and at most one input is $\text{CONST}(\top)$. Otherwise, a common cause failure can let both B and C fail simultaneously, both PAND gates fail and A fails as well. The independence assumption in this rewrite rule is a *context restriction*, which prevents the application of the rule for certain DFTs.

5.1.3 Non-structural Rules

There are two additional rules that are not present in the rewrite framework as they go beyond structural rules and are not captured by graph transformations.

Removing BEs The BEs that have no connection to other DFT elements (and are not the top level element) are called *dispensable*. Dispensable BEs can be removed from the DFT as they do not influence the analysis results. An example is given in Figure 5.6. In the original DFT in Figure 5.6(a), BE C is dispensable and can be removed yielding the DFT in Figure 5.6(b).

Merging BEs In our analysis, we are only interested in the reliability of the top level element. The state of other elements is not important for this analysis. Thus, we can simplify a DFT by merging multiple BEs into a single BE. Consider the example DFT in Figure 5.6(b). Both BEs A and B have an exponential failure distribution with failure rates λ_A and λ_B , respectively. The failure distribution of an OR gate is the minimum over its inputs and is exponentially distributed as well. Thus, we can replace multiple BEs A_1, \dots, A_n under an OR gate by a single BE A' with failure rate $\lambda_{A'} = \sum_{i=1}^n \lambda_{A_i}$. In our example, merging both BEs leads to the final DFT in Figure 5.6(c). The resulting OR gate with a single input can be simplified further by applying the rewrite framework.

After presenting the details of DFT rewrite rules, in the sequel, we present our efforts in formally verifying them using HOL theorem proving. For some of these rules, such as Rule 5, it is necessary to formally model DFT gates for an arbitrary number of inputs. Therefore, in the next section, we introduce the new HOL definitions of n -ary gates.

5.2 HOL Formalization of n -ary DFT Gates

In order to handle DFT gates with an arbitrary number of inputs, we extend the definitions of DFT gates of Chapter 3 by utilizing lists to represent the arbitrary number of inputs. In other words, the input of an n -ary gate is a list of arbitrary size of time-to-failure functions that represent inputs of a DFT gate.

We formally hence define the n -ary AND gate as:

Definition 5.1. *n_AND*

$\vdash \forall L. \text{n_AND } L = \text{FOLDR } (\lambda a b. \text{ D_AND } a b) \text{ ALWAYS } L$

where `FOLDR` is used to apply a binary (2-input) function over a list from right to left. The function in our case here is the binary `D_AND` that accepts two inputs and returns their result of the DFT AND operation between them. `FOLDR` requires including an element that is used to apply the function to the last element of the input list. We use `ALWAYS` in this case as it is the identity element of the AND and does not affect its behavior. `L` represents the list of inputs to be ANDed. For example, `n_AND [X; Y; Z]` equals `D_AND X (D_AND Y (D_AND Z ALWAYS))`.

In a similar manner, we formally define the n -ary OR as:

Definition 5.2. *n_OR*

$\vdash \forall L. \text{n_OR } L = \text{FOLDR } (\lambda a b. \text{ D_OR } a b) \text{ NEVER } L$

`D_OR` is the function used with `FOLDR` in this definition. We use `NEVER` in this case as it is the identity element for the OR, i.e., `NEVER` will not affect the behavior of the OR gate. It is worth mentioning that `FOLDL` can be used with these definitions as well, since the order of applying the OR and AND gates does not matter if it starts from the left or from the right.

We formally define the n -ary PAND gate as:

Definition 5.3. *n_PAND*

$\vdash \forall L. \text{n_PAND } L = \text{FOLDL } (\lambda a b. \text{ P_AND } a b) \text{ ALWAYS } L$

This is similar to the previous definitions. However, since the PAND gate requires that the input events fail from left to right, we use `FOLDL` in this case. We use `ALWAYS` as it does not affect the behavior of the PAND gate, i.e., for any input `X` that is greater than or equal to 0, `PAND ALWAYS X = X`.

The VOT_k (k out of n) gate can be defined using the `n_OR` and `n_AND` gates. Firstly, we need to get the combinations that lead to the failure of the VOT gate. For

example, a (2/3) VOT gate requires having all possible pairs out of the three inputs. Therefore, we first need to get all the possible k elements of the input list. We define `k_out` that accepts a list and a number k , which identifies the number of elements to be retrieved from the input list.

Definition 5.4. *k_out*

$$\vdash \forall k L. \text{k_out } k L = \{s \mid s \subseteq (\text{set } L) \wedge (\text{CARD } s = k)\}$$

where `set L` returns a set with the elements in list L , and `CARD` is a HOL function that returns the cardinality (number of elements) of a given set. This definition basically returns a set of sets, where the inner sets are subsets of `set L`. This means that these inner subsets contain elements from the input list L . The added condition is that the cardinality of each of these sets equals k . As a result, we get all possible combinations of the input list that have k elements.

We use `k_out` to define the VOT gate by ANDing the elements of each inner set, then ORing the result of this ANDing. We need to recall that the `n_AND` and `n_OR` accept inputs as lists not sets. Therefore, we apply a function that converts a set into a list (`SET_TO_LIST`). We formally define the VOT gate as:

Definition 5.5. *VOT*

$$\vdash \forall k L. \text{k_out_n_gate } k L = \\ \text{n_OR (MAP } (\lambda a. \text{ n_AND (SET_TO_LIST } a)) \text{ (SET_TO_LIST (k_out } k L))}$$

where `SET_TO_LIST` is a HOL4 function that accepts a set and returns a list of the elements of this set. `MAP` is used to map a function over a list and returns a list of the mapped elements. In this definition, we first convert the outer set of `k_out` to a list using `SET_TO_LIST (k_out k L)`. Then, we apply `n_AND` to each element of this list using `MAP` and convert each inner set to a list. Finally, the `n_OR` is applied to the result

of the MAP, i.e., the result will be the OR of ANDs and each AND has only k elements of the input list. We verify several properties for `k_out` and the VOT gate, such as the finiteness of the inner and outer sets, besides other properties that are useful in the verification of the DFT rewriting rules. The HOL4 script can be accessed from [69].

5.3 Verification of Rewrite Rules

We list the verification details of some of the rewrite rules described in Section 5.1. The details of verifying the rest of the rules can be accessed from [69].

5.3.1 General Rewrite Rules

The structural rewrite rules 1-5 and 7 are verified based on the definitions of n -ary gates and some list and extreal number theories properties, whereas rule 6 is implemented implicitly in the DFT formalization.

Commutativity of Static Gates (Rule 1)

Theorem 5.1.

$$\vdash \forall L_1 L_2. \text{ PERM } L_1 L_2 \Rightarrow (\text{n_AND } L_1 = \text{n_AND } L_2)$$

Theorem 5.2.

$$\vdash \forall L_1 L_2. \text{ PERM } L_1 L_2 \Rightarrow (\text{n_OR } L_1 = \text{n_OR } L_2)$$

Theorem 5.3.

$$\vdash \forall L_1 L_2 k. \text{ PERM } L_1 L_2 \Rightarrow (\text{k_out_n_gate } k L_1 = \text{k_out_n_gate } k L_2)$$

The commutativity property indicates that the order of the inputs of any static gate will not affect its behavior, i.e., the time of failure for the output of the gate

remains the same. We use the permutation of two lists ($\text{PERM } L_1 \ L_2$) to add the condition that L_1 and L_2 have the same inputs but with different orders. We verify the commutativity of the n_AND and n_OR gates using induction, FOLDR definition and some properties of the 2-input AND and OR gates, defined in Chapter 3, such as associativity and commutativity. The proof of the commutativity property for the VOT gate is mainly based on the following lemma:

Lemma 5.1.

$$\vdash \forall L_1 \ L_2 \ k. \ \text{PERM } L_1 \ L_2 \Rightarrow (\text{k_out } k \ L_1 = \text{k_out } k \ L_2)$$

which states that the sets returned by k_out are the same for two lists that have the same elements with different orders.

Gate with a Single Successor (Rule 3)

Theorem 5.4.

$$\vdash \forall x. \ \text{rv_gt0 } [x] \Rightarrow (\text{n_AND } [x] = x)$$

Theorem 5.5.

$$\vdash \forall x. \ \text{n_OR } [x] = x$$

Theorem 5.6.

$$\vdash \forall x. \ \text{rv_gt0 } [x] \Rightarrow (\text{k_out_n_gate } 1 \ [x] = x)$$

Theorem 5.7. $\vdash \forall x. \ \text{rv_gt0 } [x] \Rightarrow (\text{n_PAND } [x] = x)$

For the static gates and the n_PAND gate, if the input list consists of only one element, then the output fails once the single input fails. The function rv_gt0 ensures that the inputs of the gates are greater than or equal to 0, which is valid as we are dealing with time-to-failure functions. We recursively define rv_gt0 as:

Definition 5.6. rv_gt0

$$(rv_gt0 [] = T) \wedge (\forall h t. \quad rv_gt0 (h::t) = (\forall s. \quad 0 \leq h s) \wedge rv_gt0 t)$$

For n_AND and n_OR , rule 3 is verified based on some properties of the D_AND and D_OR gates. For VOT gate, we use the VOT ($1/n$) property (Theorem 5.25) that replaces the VOT gate with the n_OR gate. Finally, we verify rule 3 for n_PAND using its definition and some list and extreal numbers properties.

*Left Flattening of AND/OR/PAND Gates (Rule 5)***Theorem 5.8.**

$$\vdash \forall L_1 L_2.$$

$$rv_gt0 (L_1 ++ L_2) \Rightarrow (n_AND (n_AND L_2::L_1) = n_AND (L_2 ++ L_1))$$

Theorem 5.9.

$$\vdash \forall L_1 L_2. \quad n_OR (n_OR L_2::L_1) = n_OR (L_2 ++ L_1)$$

Theorem 5.10.

$$\vdash \forall L_1 L_2.$$

$$rv_gt0 (L_1 ++ L_2) \Rightarrow (n_PAND (n_PAND L_2::L_1) = n_PAND (L_2 ++ L_1))$$

In order to verify Theorem 5.8, we first verify the n_AND append property that would split the AND of two appended lists as:

Lemma 5.2.

$$\vdash \forall L_1 L_2.$$

$$rv_gt0 (L_1 ++ L_2) \Rightarrow (n_AND (L_1 ++ L_2) = D_AND (n_AND L_1)(n_AND L_2))$$

where $++$ is a list operator used to append two lists. We verify Theorem 5.8 by first rewriting $n_AND L_2::L_1$ as $[n_AND L_2]++L_1$, where $::$ is a list operator used to add an

element to a list, which in the considered case is $n_AND\ L_2$. Then, we use Lemma 5.2 to rewrite the left hand side of Theorem 5.8 to $D_AND\ (n_AND\ [n_AND\ L_2])\ (n_AND\ L_1)$ and use Theorem 5.4 to verify Theorem 5.8. In a similar way, we verify Theorem 5.9 by verifying a lemma for appending two lists with n_OR as:

Lemma 5.3.

$$\vdash \forall L_1\ L_2. \quad n_OR\ (L_1\ ++\ L_2) = D_OR\ (n_OR\ L_1)\ (n_OR\ L_2)$$

For the left-flattening property of the n_PAND gate, we first verify a lemma that $rv_gt0\ L \Rightarrow \forall s. \quad 0 \leq n_PAND\ L\ s$, which states that the output of the n_PAND gate is greater than or equal to 0 if the inputs follow the same condition. Theorem 5.10 is then verified based on the previous lemma, induction on the list argument and some P_AND and list properties.

Identical Leftmost Successors of AND, OR or PAND Gates (Rule 7)

Theorem 5.11.

$$\vdash \forall x\ L. \quad n_AND\ (x::x::L) = n_AND\ (x::L)$$

Theorem 5.12.

$$\vdash \forall x\ L. \quad n_OR\ (x::x::L) = n_OR\ (x::L)$$

Theorem 5.13.

$$\vdash \forall x\ L. \quad rv_gt0\ [x] \Rightarrow (n_PAND\ (x::x::L) = n_PAND\ (x::L))$$

Theorems 5.11 and 5.12 are verified based on the definitions of n_AND and n_OR with the associativity and idempotence of D_AND and D_OR gates. Theorem 5.13 requires verifying that the output of a 2-input $PAND$ gate (P_AND defined in Chapter 3) with an input that already failed ($ALWAYS$) as the left input fails with the failure of the second (right) input.

Lemma 5.4.

$$\vdash \forall X. (\forall s. 0 \leq X \ s) \Rightarrow (\text{P_AND ALWAYS } X = X)$$

Finally, we verify the idempotence property of the P_AND gate.

Lemma 5.5.

$$\vdash \forall X. \text{P_AND } X \ X = X$$

Subsumption of OR Gates by AND Gates (Rule 8)

Theorem 5.14.

$$\vdash \forall X \ Y. \text{D_AND } X \ (\text{D_OR } X \ Y) = X$$

Subsumption of AND Gates by OR Gates (Rule 9)

Theorem 5.15.

$$\vdash \forall X \ Y. \text{D_OR } X \ (\text{D_AND } X \ Y) = X$$

Distributing OR Gates over AND Gates (Rule 10)

Theorem 5.16.

$$\vdash \forall X \ Y \ Z. \text{D_OR } (\text{D_AND } X \ Y) \ (\text{D_AND } Y \ Z) = \text{D_AND } (\text{D_OR } X \ Z) \ Y$$

We verify rules 8-10 that are concerned with the standard axioms of Boolean algebra based on basic properties of D_AND and D_OR gates, such as the commutativity and distributivity of the AND over the OR.

OR Gates with Fail-Safe (NEVER) Successors (Rule 11)

Theorem 5.17.

$$\vdash \forall L_1 L_2. \text{ n_OR } (L_1 \text{ ++ } [\text{NEVER}] \text{ ++ } L_2) = \text{ n_OR } (L_1 \text{ ++ } L_2)$$

OR Gates with Already Failed (ALWAYS) Successors (Rule 12)

Theorem 5.18.

$$\vdash \forall L_1 L_2. \text{ rv_gt0 } (L_1 \text{ ++ } L_2) \Rightarrow (\text{ n_OR } (L_1 \text{ ++ } [\text{ALWAYS}] \text{ ++ } L_2) = \text{ ALWAYS})$$

Rewrite rules 11-14 deal with scenarios that include fail-safe (NEVER) or CONST(\perp), and failed (ALWAYS) or CONST(\top).

For Theorem 5.17, we use Lemma 5.3 and the definition of n_OR with the property stating that $\forall X. \text{ D_OR } X \text{ NEVER} = X$. We verify Theorem 5.18 based on Lemma 5.3 and the definition of n_OR along with the following lemma:

Lemma 5.6.

$$\vdash \forall X. (\forall s. 0 \leq X \ s) \Rightarrow (\text{ D_OR } X \text{ ALWAYS} = \text{ ALWAYS})$$

Then, we verify that the output of the n_OR is greater than or equal to 0 if the inputs are all greater than or equal to 0. Theorem 5.18 is then verified using the previous lemmas and some properties of the D_OR gate.

AND Gate with a Fail-Safe (NEVER) Successor (Rule 13)

Theorem 5.19.

$$\vdash \forall L_1 L_2. \text{ rv_gt0 } (L_1 \text{ ++ } L_2) \Rightarrow (\text{ n_AND } (L_1 \text{ ++ } [\text{NEVER}] \text{ ++ } L_2) = \text{ NEVER})$$

Theorem 5.20.

$$\vdash \forall L. \text{ rv_gt0 } L \Rightarrow (\text{ n_PAND } (L \text{ ++ } [\text{NEVER}]) = \text{ NEVER})$$

Theorem 5.21.

$$\vdash \forall L. \text{rv_gt0 } L \Rightarrow (\text{n_PAND } (\text{NEVER}::L) = \text{NEVER})$$

Theorem 5.22.

$$\vdash \forall L_1 L_2. \text{rv_gt0 } (L_1 ++ L_2) \Rightarrow (\text{n_PAND } (L_1 ++ [\text{NEVER}] ++ L_2) = \text{NEVER})$$

We verify Theorem 5.19 using Lemma 5.2 and some properties for the `D_AND`, such as the commutativity property and `ANDing` with `NEVER`.

We verify this rule for the `PAND` gate by verifying two cases. Firstly, we verify that the output of the `PAND` cannot fail if the `NEVER` input is the rightmost input (Theorem 5.20). This is mainly verified based on some list properties to manipulate `rv_gt0` along with the left flattening property of the `PAND` (Theorem 5.10). Similarly, we verify the second case when the left most input of the `PAND` gate is fail-safe (Theorem 5.21). Finally, we verify a generic property, where the fail-safe input can be at any position (Theorem 5.22).

AND Gate with a Failed (ALWAYS) Element as Successor (Rule 14)

Theorem 5.23.

$$\vdash \forall L. \text{rv_gt0 } L \Rightarrow (\text{n_AND } (\text{ALWAYS}::L) = \text{n_AND } L)$$

Theorem 5.24.

$$\vdash \forall L. \text{rv_gt0 } L \Rightarrow (\text{n_PAND } (\text{ALWAYS}::L) = \text{n_PAND } L)$$

Theorem 5.23 is verified using the definition of the `n_AND` gate with the property that the output of the gate is greater than or equal to 0 if the inputs satisfy the same condition. We verify Theorem 5.24 based on the definition of the `n_PAND` and the idempotence property of the `PAND` gate.

The VOT gate can behave as an OR gate, when $k = 1$ (Rule 15), and as an AND gate, when k equals the number of its inputs (Rule 16). The verification details of these rules are listed below.

Voting (1/n) is an OR Gate (Rule 15)

Theorem 5.25.

$\vdash \forall L. \text{ALL_DISTINCT } L \wedge \text{rv_gt0 } L \Rightarrow (\text{k_out_n_gate } 1 \ L = \text{n_OR } L)$

As mentioned previously, the voting gate is defined as the OR of a list and each element in the list is the AND of another list of k elements. In order to verify Theorem 5.25, we need to use the commutativity property of the `n_OR` gate (Theorem 5.2), i.e., we need to verify that the list of the `n_OR` in the voting gate definition (`MAP` $(\lambda a. \text{n_AND } (\text{SET_TO_LIST } a)) (\text{MAP } (\lambda a. \{a\}) \ L)$) and the input list L possess the permutation property when $k = 1$. Therefore, we first verify that the list generated from `k_out 1 L` is the permutation of the list `MAP` $(\lambda a. \{a\}) \ L$. We need to recall that `MAP` $(\lambda a. \{a\}) \ L$ generates another list that has all elements from the input list L but as sets. Then, we verify that the list generated from applying the `n_AND` to the list of `k_out 1 L` is the permutation of applying `n_AND` to `MAP` $(\lambda a. \{a\}) \ L$. We also verify the following property:

Lemma 5.7.

$\vdash \forall L. \text{rv_gt0 } L \Rightarrow$

$\text{PERM } (\text{MAP } (\lambda a. \text{n_AND } (\text{SET_TO_LIST } a)) (\text{MAP } (\lambda a. \{a\}) \ L)) \ L$

Finally, we use these verified properties of permutation and the commutativity property of `n_OR` to verify Theorem 5.25.

Voting (n/n) is an AND Gate (Rule 16)

Theorem 5.26.

$\vdash \forall L. \text{ALL_DISTINCT } L \Rightarrow (\text{k_out_n_gate } (\text{LENGTH } L) L = \text{n_AND } L)$

Theorem 5.26 is used when k equals the length of the input list ($\text{LENGTH } L$), i.e., $\text{VOT } (n/n)$, and n is the number of inputs of the gate. In this case, the VOT gate acts as an AND gate. We verify this by first rewriting using the VOT gate and k_out definitions. Then, we verify that $\{\text{s} \mid \text{s} \subseteq \text{set } L \wedge (\text{CARD } \text{s} = \text{LENGTH } L)\} = \{\text{set } L\}$. This way the original expression of the VOT gate can be reduced to $\text{n_OR } [\text{n_AND } (\text{SET_TO_LIST } (\text{set } L))]$. Then, we verify that $\text{PERM } L (\text{SET_TO_LIST } (\text{set } L))$, which means that the original list and the list generated from the set of the original list are the permutation of each other. This is a consequence of using $\text{set } L$ in the formal definition of the VOT gate, which requires the added condition that the elements in the original list are distinct, i.e., they are not equal or repeated. This condition is added using the HOL predicate $\text{ALL_DISTINCT } L$. Finally, we verify Theorem 5.26 using the commutativity property of the AND (Theorem 5.1) and the definition of n_OR .

5.3.2 Rewrite Rules for PAND Gates

Rules 18-23 deal with PAND gates that require considering the order of the inputs.

Representing AND Gate using OR and PAND Gates (Rule 18)

Theorem 5.27.

$\vdash \forall X Y. \text{D_AND } X Y = \text{D_OR } (\text{P_AND } X Y) (\text{P_AND } Y X)$

Conflicting PAND Gates with Independent Successors (Rule 19)

Theorem 5.28.

$$\vdash \forall X Y. (\forall s. \text{ALL_DISTINCT } [X \ s; Y \ s]) \Rightarrow \\ (\text{D_AND } (\text{P_AND } X \ Y) (\text{P_AND } Y \ X) = \text{NEVER})$$

We verify Theorems 5.27 and 5.28 based on the definitions of D_AND, D_OR and P_AND gates and some properties of extreal numbers. Note that the added condition for rule 19 is that the inputs are distinct (ALL_DISTINCT), i.e., they cannot fail simultaneously. This results from the fact that the inputs are independent (there is no common cause of failure) and they possess continuous failure distributions. Therefore, rule 19 cannot be applied unless this context restriction is ensured using this assumption.

PAND Gate with a PAND Successor (Rule 20)

Theorem 5.29.

$$\vdash \forall B \ C_1 \ C_2 \ L. \text{rv_gt0 } (L \ ++ \ [B; \ C_1; \ C_2]) \Rightarrow \\ (\text{n_PAND } ([B; \ \text{P_AND } \ C_1 \ C_2] \ ++ \ L) = \\ \text{D_AND } (\text{P_AND } \ C_1 \ C_2) (\text{n_PAND } ([B; \ C_2] \ ++ \ L)))$$

We verify Theorem 5.29 based on manipulating the input lists and the PAND appended with a single element lemma, which we verify as:

Lemma 5.8.

$$\vdash \forall x \ L. \text{rv_gt0 } L \Rightarrow (\text{n_PAND } (L \ ++ \ [x]) = \text{P_AND } (\text{n_PAND } L) \ x)$$

Based on Lemma 5.8 and list induction and manipulation, we verify that the left-hand-side of Theorem 5.29 equals: P_AND(D_AND(P_AND C1 C2)(n_PAND (B::C2::L))) x, where x is the additional element generated through induction.

Then, we verify a property stating that the time of failure of the PAND gate should be greater than or equal to the failure time of any of its inputs, since it is required that the failure occurs from left to right.

PAND Gate with a First OR Successor (Rule 21)

Theorem 5.30.

$\vdash \forall X Y L. \text{rv_gt0 } [X; Y] \Rightarrow$

$(\text{n_PAND } (\text{D_OR } X Y :: L) = \text{D_OR } (\text{n_PAND } (X :: L)) (\text{n_PAND } (Y :: L)))$

To verify Theorem 5.30, we first apply induction to the input argument and rewrite using the rule of `n_PAND` with a single successor. Then, we use the definitions of the `P_AND`, `n_PAND` and some simplification theorems, such as `P_AND ALWAYS X = X`. Using some list properties, such as applying a function to two appended lists using `FOLDL` (we need to recall that the definition of `n_PAND` is based on `FOLDL`), we reach a point where the whole goal can be verified using the following lemma:

Lemma 5.9. $\vdash \forall X Y Z. \text{P_AND } (\text{D_OR } X Y) Z = \text{D_OR } (\text{P_AND } X Z) (\text{P_AND } Y Z)$

PAND Gate with ALWAYS as Non-First Successor (Rule 23)

Theorem 5.31.

$\vdash \forall L_1. L_1 \neq [] \wedge (\forall x. \text{MEM } x L_1 \Rightarrow \forall s. 0 < x s) \Rightarrow$

$\forall L_2. \text{n_PAND } (L_1 ++ [\text{ALWAYS}] ++ L_2) = \text{NEVER}$

Theorem 5.31 shows that if the inputs to the left of the input that already failed (`ALWAYS`) do not fail from the beginning, i.e., their time of failure is greater than 0, then the output of the `n_PAND` can never fail. Therefore, we add the condition that the inputs to the left (list `L1`) are greater than 0 using $\forall x. \text{MEM } x L_1 \Rightarrow \forall s. 0$

$< x \ s$. We verify Theorem 5.31 using induction over list L_1 . After some basic list and extreal theory based reasoning, we reach the step for the left-hand-side:

$\text{FOLDL } (\lambda a \ b. \ \text{P_AND } a \ b)$

$(\text{P_AND } (\text{FOLDL}(\lambda a \ b. \ \text{P_AND } a \ b) \ h \ L_1) \ \text{ALWAYS}) \ L_2$

where h is the appended element that results from induction. We verify that $\text{P_AND } (\text{FOLDL}(\lambda a \ b. \ \text{P_AND } a \ b) \ h \ L_1) \ \text{ALWAYS} = \text{NEVER}$, which can be done if the first input of the P_AND is greater than 0. We verify the following property:

Lemma 5.10.

$\vdash \forall s \ L. (\forall x. \ \text{MEM } x \ L \Rightarrow \forall s. \ 0 < x \ s) \Rightarrow$

$\forall h. \ 0 < h \ s \Rightarrow 0 < \text{FOLDL } (\lambda a \ b. \ \text{P_AND } a \ b) \ h \ L \ s$

This lemma basically means that if we have a list of inputs and an additional element, h , that are greater than 0, then the result of applying P_AND using FOLDL is also greater than 0. Using this lemma, the left hand side is reduced to $\text{FOLDL } (\lambda a \ b. \ \text{P_AND } a \ b) \ \text{NEVER} \ L_2$. Finally, we use the following lemma to verify the Theorem 5.31.

Lemma 5.11.

$\vdash \forall L. \ \text{FOLDL } (\lambda a \ b. \ \text{P_AND } a \ b) \ \text{NEVER} \ L = \text{NEVER}$

This lemma indicates that if we apply P_AND to a list of inputs with an element NEVER at the beginning, then the output equals NEVER .

5.3.3 Non-Structural Rules

The BEs that are not connected to the given DFT can be safely removed. This is already implicitly embedded in the current DFT formalization, as we are verifying the rewrite rules by proving that the time of failure before and after rewriting remains

the same. Therefore, if the BEs are not connected to the DFT, this means that they are not affecting the time of failure of the top element and thus they can be removed in the verification process. Since DFT gates are modeled as time-to-failure functions, merging BEs is also already embedded in the DFT formalization. For example, the OR gate is modeled using the `min` function. This means that the inputs of the OR gate are merged and the output of the OR gate can be replaced with the `min` function.

We illustrate the usage of the verified rules on the example of Figure 5.2:

Theorem 5.32.

$$\vdash \forall c d f. \text{ P_AND } (\text{D_AND } c (\text{D_OR } c d)) (\text{D_AND } d f) = \text{P_AND } c (\text{D_AND } d f)$$

5.4 Summary

As an application of our DFT formalization and analysis framework, in this chapter, we presented the formal definitions and proofs of the rewriting rules in [66], which we believe is a novel contribution as details about how to mathematically conduct these proofs are not available in [66]. In fact, in [66], the correctness of the rewrite rules is described implicitly based on the behavior of DFT gates rather than on their formal mathematical models as presented in this work. It is worth noting that our formal definitions and verified lemmas allowed verifying several DFT rewriting rules that can be used with tools that simplify DFTs prior to the analysis. In addition, verifying these rules represent the first step towards formally verifying other DFT tools, such as STORM.

Chapter 6

Formal Analysis of Dynamic Reliability Block Diagrams

In this chapter, we introduce our novel DRBD algebra that allows conducting both the qualitative and quantitative analyses based on the structure of the DRBD. We propose new DRBD operators, similar to the DFT algebra, to capture the dynamic dependencies among system components. We use these operators to model the three variants of the spare construct, i.e., hot spare HSP, cold spare CSP, and warm spare WSP. Furthermore, we model the series, parallel, series-parallel and parallel-series structures using our newly introduced operators. We propose several simplification theorems to enable reducing the structure function of a given DRBD. We express the reliability of the DRBD spare construct and structures to provide generic expressions of distribution and density functions. We formalize this algebra to ensure its soundness and allow the analysis within a theorem proving environment. Our ultimate goal is to develop a formally verified algebra that follows the traditional reliability expressions of the series and parallel structures in an easily extensible manner and at the same

time can capture the dynamic behavior of real-world systems. Our formalization differs from and overcomes the formalization of traditional RBDs presented in [22] in the sense that it can formally express the structure function of a DRBD using the introduced DRBD operators. In addition, it can formally model and analyze DRBD spare constructs. Furthermore, we model the traditional RBD structures, i.e., series, parallel and deeper structures in a way similar to the mathematical models available in the literature, which makes it easily understood and followed by reliability engineers that are not familiar with HOL theorem proving. We illustrate the usefulness of the proposed developments in conducting the formal analysis of the DBW system. Finally, we verify the equivalence of both DFT and DRBD algebras, which enables the analysis in both directions.

6.1 Methodology

Figure 6.1 depicts the proposed methodology to formally conduct the DRBD qualitative and quantitative analyses using HOL4. Similar to the DFT analysis, the formal DRBD analysis starts with a system description that is interpreted into a DRBD model and some reliability requirements that should be met. A formal model is created using the DRBD structures and constructs, such as the series, parallel and spares. Then, based on the simplification theorems, the DRBD structure is reduced to enable the qualitative analysis in the form of cut sets and cut sequences. The quantitative analysis of the DRBD is performed using the verified reliability expressions of the DRBD structures. As a result, a generic reliability expression of the modeled system is formally verified using HOL4. Since the DFT and the DRBD encompass complementary behavior, we utilize our DFT library to formalize DRBDs, i.e., the DFT theory is the parent of the DRBD, besides the measure, Lebesgue integral and

probability theories.

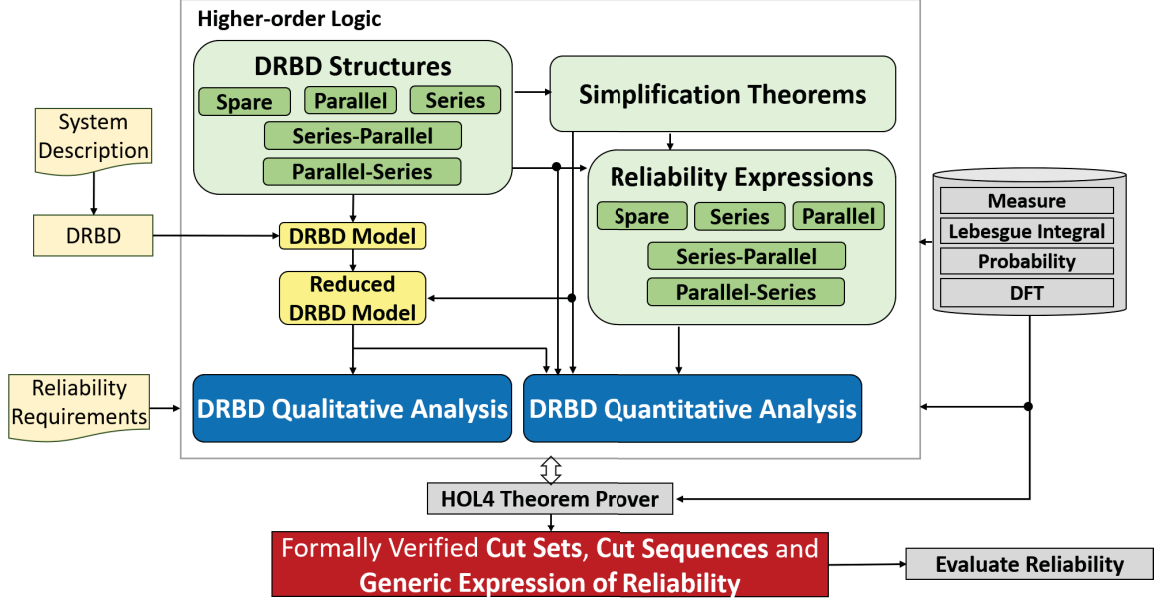


Figure 6.1: Formal DRBD Analysis Methodology

6.2 DRBD Event

Throughout this work, we assume that system components or blocks are represented by random variables that in turn represent their time-to-failures. In addition, we assume that system components are non-repairable, i.e., we are interested in expressing the reliability of the system considering that the failed components will not be repaired. It is worth mentioning that our proposed algebra follows the general lines for the DFT algebra [30], which allows DFTs conversion into DRBDs for conducting their analysis as well.

The reliability of a single component, which time-to-failure function is represented by random variable X , is mathematically defined as [4]:

$$R_X(t) = Pr\{s \mid X(s) > t\} = 1 - Pr\{s \mid X(s) \leq t\} = 1 - F_X(t) \quad (6.1)$$

where $F_X(t)$ is the CDF of X .

We call $\{s \mid X(s) > t\}$ a *DRBD event* as it represents the set that we are interested in finding the probability of until time t :

$$event(X, t) = \{s \mid X(s) > t\} \quad (6.2)$$

In our formalization, we define the inputs, or the random variables representing the time to failure of system components, as lambda abstracted functions with a return datatype of extended-real, which represents real numbers besides $\pm\infty$.

We formally define the DRBD event of Equation (6.2) as:

Definition 6.1. *DRBD Event*

$$\vdash \forall p \ X \ t. \ DRBD_event \ p \ X \ t = \{s \mid Normal \ t < X \ s\} \cap p_space \ p$$

where `Normal` typecasts the real value of `t` from real to extended-real. This type conversion is required since we need real-valued random variables. However, we need to deal with the extended-real data-type to model the `NEVER` element. Therefore, we define the time-to-failure functions to return extended-real and typecast the values from extended-real to real using the function `real` and vice versa using `Normal`. This is similar to our approach of the DFT formalization.

We formally define the reliability as the probability of the DRBD event according to Equation (6.1):

Definition 6.2. *Reliability*

$$\vdash \forall p \ X \ t. \ Rel \ p \ X \ t = prob \ p \ (DRBD_event \ p \ X \ t)$$

We verify the relationship between the reliability and the CDF of Equation (6.1) as:

Theorem 6.1.

$$\vdash \forall p \ X \ t. \quad \text{rv_gt0_ninfinity} \ [X] \ \wedge \\ \text{random_variable} \ (\text{real} \circ X) \ p \ \text{borel} \ \Rightarrow \\ (\text{Rel} \ p \ X \ t = 1 - \text{CDF} \ p \ (\text{real} \circ X) \ t)$$

where `real` typecasts the values of the random variable from extended-real to real as the CDF is defined for real-valued random variables, `random_variable (real o X) p borel` ensures that `(real o X)` is a random variable over the real line represented by the `borel` space, and `rv_gt0_ninfinity` ensures that the random variable is greater than or equal to 0 and not equal to $+\infty$, which means that the time of failure of any component cannot be negative or $+\infty$. Theorem 6.1 is verified based on the fact that the `DRBD_event` and the set of the CDF are the complement of each other. Therefore, the probability of one of them equals one minus the other. For the rest of the work, we will denote `CDF p (real o X) t` by $F_X(t)$ to facilitate the understanding of the theorems.

6.3 Identity Elements and Operators

Similar to the identity elements of ordinary Boolean algebra and DFT algebra [30], we introduce two identity elements, i.e., `ALWAYS` and `NEVER`, that represent two states of any system block. The `ALWAYS` element represents a system component that always fails, i.e., it fails from time 0. While the `NEVER` element represents a component that never fails, i.e., the time of its failure is $+\infty$. These identity elements play an important role in the reduction process of the structure functions of DRBDs, as will be introduced in the following sections.

$$ALWAYS = 0 \tag{6.3}$$

$$NEVER = +\infty \tag{6.4}$$

We formally define these elements as:

Definition 6.3. *DRBD ALWAYS*

$$\vdash \text{R_ALWAYS} = (\lambda s. (0:\text{extreal}))$$

Definition 6.4. *DRBD NEVER*

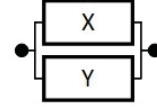
$$\vdash \text{R_NEVER} = (\lambda s. \text{PosInf})$$

We introduce operators to model the relationship between the various blocks in a DRBD. These operators can be divided into two categories: 1) The AND and OR operators that are not concerned with the dependencies among system components. 2) Temporal operators, i.e., *After*, *Simultaneous* and *Inclusive After*, that can capture the dependencies between system components. It is worth mentioning that DRBDs are concerned with modeling the several paths of success of a given system. Therefore, if we are concerned in knowing the success behavior of a DRBD until time t , it means that we are interested in knowing how the system would not fail until time t . As a result, we can use the time-to-failure random variables in modeling the time-to-failure of a given DRBD, i.e., its structure function. It is assumed that for any two system components that possess continuous failure distribution functions, the possibility that these components fail at the same time can be neglected.

In [44], AND and OR operators were introduced to model the parallel and series constructs between dependent components only without providing any mathematical model to these operators. We propose to use the AND (\cdot) and OR ($+$) operators to model series and parallel blocks in a DRBD, respectively without any restriction. We provide a mathematical model for each operator based on the time of failure of its input to be used in the proposed algebra. The AND operator models the series connection



(a) Series DRBD



(b) Parallel DRBD

Figure 6.2: Two-Block Series and Parallel DRBDs

between two or more system blocks, as shown in Figure 6.2(a). For example, the DRBD in Figure 6.2(a) will continue to work only if component X and component Y are working. Once one of these blocks stops working, then there will be no connection between the input and the output of the DRBD and thus the system will no longer work. We model the AND operator as the minimum time of its input arguments. Similarly, the OR operator models the connection between parallel components in a DRBD. For example, the DRBD in Figure 6.2(b) will continue to work if X is working or Y is working. All the components in a parallel structure should fail for this DRBD to fail. Therefore, we model the OR operator as the maximum time of failure of its input arguments, which represents the time of failure of basic system blocks or sub-DRBDs. This approach facilitates using these operators to model even more complex structures.

We define the AND and OR operators as:

$$X \cdot Y = \min (X, Y) \tag{6.5}$$

$$X + Y = \max (X, Y) \tag{6.6}$$

We formally define these operators as:

Definition 6.5. *DRBD AND*

$$\vdash \forall X Y. \text{R_AND } X Y = (\lambda s. \min (X \text{ s}) (Y \text{ s}))$$

Definition 6.6. *DRBD OR*

$$\vdash \forall X Y. R_OR X Y = (\lambda s. \max (X s) (Y s))$$

If X and Y are independent, then the reliability of the systems, shown in Figure 6.2, can be expressed as:

$$R_{(X \cdot Y)}(t) = R_X(t) \times R_Y(t) \quad (6.7)$$

$$R_{(X+Y)}(t) = 1 - ((1 - R_X(t)) \times (1 - R_Y(t))) \quad (6.8)$$

To reach these expressions, it is required first to express the DRBD events as the intersection and union for the AND and OR operators, respectively, as:

$$event ((X \cdot Y), t) = event (X, t) \cap event (Y, t) \quad (6.9)$$

$$event ((X + Y), t) = event (X, t) \cup event (Y, t) \quad (6.10)$$

We verify their reliability expressions as in Theorems 6.2 and 6.3, respectively.

Theorem 6.2.

$$\begin{aligned} \vdash \forall p X t. \quad & rv_gt0_ninfinity [X;Y] \wedge \\ & indep_var p lborel (real \circ X) lborel (real \circ Y) \Rightarrow \\ & (Rel p (X \cdot Y) t = Rel p X t * Rel p Y t) \end{aligned}$$

Theorem 6.3.

$$\begin{aligned} \vdash \forall p X t. \quad & rv_gt0_ninfinity [X;Y] \wedge \\ & indep_var p lborel (real \circ X) lborel (real \circ Y) \Rightarrow \\ & (Rel p (X + Y) t = 1 - (1 - Rel p X t) * (1 - Rel p Y t)) \end{aligned}$$

We verify Theorem 6.2 by first rewriting using Definition 6.2. Then, we prove that `DRBD_event` of the AND operator equals the intersection of the individual events,

as in Equation (6.9). Utilizing the independence of the real-valued random variables $\text{real} \circ X$ and $\text{real} \circ Y$, the probability of intersection of their events equals the product of the probability of the individual events. Since X and Y are greater than 0 and are not equal to $+\infty$, based on the function `rv_gt0_ninfinity`, the events in the probability space that correspond to X and Y are equal to the ones that correspond to $\text{real} \circ X$ and $\text{real} \circ Y$. As a result, the `DRBD_events` of X and Y are independent. Hence, the probability of their intersection equals the product of the probability of the individual events, i.e., their reliability. Theorem 6.3 is verified in a similar way. However, we prove that the `DRBD_event` of the OR operator equals the union of the individual events, as in Equation (6.10). We verify that this union of events equals to the complement of the intersection of the complements of the individual events. Now, Theorem 6.3 can be proven using the independence of random variables.

We extend the definition of the AND and OR operators to n-ary operators, `nR_AND` and `nR_OR`, that can be used to represent the relationship between an arbitrary number of elements. We formally define n-ary AND (`nR_AND`) as:

Definition 6.7. *nR_AND*

$\vdash \forall X \ s. \ \text{nR_AND } X \ s = \text{ITSET } (\lambda e \ \text{acc}. \ \text{R_AND } (X \ e) \ \text{acc}) \ s \ \text{R_NEVER}$

where `ITSET` is the HOL function to iterate over sets. This definition applies the `R_AND` over the elements of X indexed by the numbers in s . `R_NEVER` is the identity element of the `R_AND` operator.

Similarly, we formally define n-ary OR (`nR_OR`) as:

Definition 6.8. *nR_OR*

$\vdash \forall X \ s. \ \text{nR_OR } X \ s = \text{ITSET } (\lambda e \ \text{acc}. \ \text{R_OR } (X \ e) \ \text{acc}) \ s \ \text{R_ALWAYS}$

where `R_ALWAYS` is the identity element of the `R_OR` operator. The reliability of these

two operators would be similar to the reliability of the series and parallel structures, respectively, as will be described in the following section.

In order to model the dynamic behavior of systems in DRBDs, we introduce new temporal operators: *after* (\triangleright), *simultaneous* (Δ), and *inclusive after* (\trianglerighteq). The *after* operator represents a situation where it is required to model a component that continues to work after the failure of another. The time of failure of the after operator equals the time of failure of the last component, which is required to fail. However, if the required sequence does not occur, then the output can never fail, i.e., the time of failure equals $+\infty$.

$$X \triangleright Y = \begin{cases} X, & X > Y \\ +\infty, & X \leq Y \end{cases} \quad (6.11)$$

The behavior of the simultaneous operator is similar to the one introduced in the DFT algebra [30]. The output of this operator fails if both its inputs fail at the same time, otherwise it can never fail.

$$X \Delta Y = \begin{cases} X, & X = Y \\ +\infty, & X \neq Y \end{cases} \quad (6.12)$$

Finally, the inclusive after operator encompasses the behavior of both the after and simultaneous operators, i.e., it models a situation where it is required that one component continues to work after another one or fail at the same time, otherwise it can never fail.

$$X \trianglerighteq Y = \begin{cases} X, & X \geq Y \\ +\infty, & X < Y \end{cases} \quad (6.13)$$

We formally define these temporal operators as:

Definition 6.9. *DRBD After*

$\vdash \forall X Y. R_AFTER X Y = (\lambda s. \text{if } Y \ s < X \ s \text{ then } X \ s \text{ else PosInf})$

Definition 6.10. *DRBD Simultaneous*

$\vdash \forall X Y. R_SIMULT X Y = (\lambda s. \text{if } X \ s = Y \ s \text{ then } X \ s \text{ else PosInf})$

Definition 6.11. *DRBD Inclusive After*

$\vdash \forall X Y.$

$R_INCLUSIVE_AFTER X Y = (\lambda s. \text{if } Y \ s \leq X \ s \text{ then } X \ s \text{ else PosInf})$

In the case of dealing with basic components, the inclusive after will behave in a similar way as the after operator. Therefore, their probabilities can be expressed for independent random variables in the same way as:

$$R_{(X \triangleright Y)}(t) = 1 - \int_0^t f_X(x) \times F_Y(x) dx \quad (6.14)$$

where f_X is the PDF of X and F_Y is the CDF of Y .

Finally, we verify this expression utilizing our formalization in Section 4.2.3:

Theorem 6.4.

$\vdash \forall X Y p f_x t. \text{rv_gt0_ninfinity } [X; Y] \wedge 0 \leq t \wedge$
 $\text{indep_var } p \text{ lborel } (\text{real } \circ X) \text{ lborel } (\text{real } \circ Y) \wedge$
 $\text{distributed } p \text{ lborel } (\text{real } \circ X) f_x \wedge (\forall x. 0 \leq f_x \ x) \wedge$
 $\text{cont_CDF } p \text{ (real } \circ Y) \wedge \text{measurable_CDF } p \text{ (real } \circ Y) \Rightarrow$
 $(\text{Rel } p \ (X \triangleright Y) \ t = 1 - \int_0^t f_x(x) \times F_Y(x) \ dx)$

The proof of this theorem is based on $Pr(Y < X < t) = \int_0^t f_X(x) \times F_Y(x) dx$, which has been verified in Section 4.2.3 using the properties of the Lebesgue integral and independence of random variables. The DRBD *after* operator represents a

situation where the system continues to work until two components fail in sequence. Thus, the above expressions allow us to verify the reliability expression of the *after* operator, as the DRBD and DFT events complement one another.

6.4 Simplification Theorems

We introduce several simplification properties to reduce the structure function of a DRBD. These simplification properties range from simple ones, such as the associativity and idempotence of the operators, to more complex theorems. The idea of these properties is to reduce the algebraic expressions based on the time of failure. For example, $X \cdot ALWAYS = ALWAYS$ means that if a component in a series structure is not working, i.e., always fails, then the series structure is not working as well. Similarly, $X + NEVER = NEVER$ means that if a component in a parallel structure cannot fail, then the whole parallel structure cannot fail as well. $X + Y = Y + X$, $X \cdot Y = Y \cdot X$ and $X \Delta Y = Y \Delta X$ represent the commutativity property for the OR, AND and simultaneous operators, respectively. An example of a more complex theorem is $X \triangleright (Y \cdot Z) = (X \triangleright Y) \cdot (X \triangleright Z)$. We formally verify these theorems in order to perform the reduction of a given DRBD using HOL. Table 6.1 lists these simplification theorems that we developed and verified using the proposed algebra.

6.5 Spare Construct

The spare construct, shown in Figure 6.3 [45], is introduced in DRBDs to model situations where a spare part is activated and replaces the main part, after its failure, by introducing a spare controller to activate the spare [45]. Depending on the failure behavior of the spare part, we can have three variants, i.e., hot, warm and cold

Table 6.1: Formally Verified DRBD Simplification Theorems

Simplification Theorems
$\vdash \forall X. (\forall s. 0 \leq X s) \Rightarrow (X \cdot \text{R_ALWAYS} = \text{R_ALWAYS})$
$\vdash \forall X Y Z. (X \cdot Y) \cdot Z = X \cdot (Y \cdot Z)$
$\vdash \forall X Y. X \cdot Y = Y \cdot X$
$\vdash \forall X. X \cdot X = X$
$\vdash \forall X. X \cdot \text{R_NEVER} = X$
$\vdash \forall X. (\forall s. 0 \leq X s) \Rightarrow (X + \text{R_ALWAYS} = X)$
$\vdash \forall X Y Z. (X + Y) + Z = X + (Y + Z)$
$\vdash \forall X Y. X + Y = Y + X$
$\vdash \forall X. X + X = X$
$\vdash \forall X. X + \text{R_NEVER} = \text{R_NEVER}$
$\vdash \forall X Y. X + (X \cdot Y) = X$
$\vdash \forall X Y Z. X \text{ rhd } (Y \triangleright Z) = ((X \triangleright Y) + (X \triangleright Z)) (Y \triangleright Z)$
$\vdash \forall X Y. (X \triangleright Y) + (Y \triangleright X) = \text{R_NEVER}$
$\vdash \forall X Y Z. X \triangleright (Y \cdot Z) = (X \triangleright Y) \cdot (X \triangleright Z)$
$\vdash \forall X Y Z. X \cdot (Y + Z) = (X \cdot Y) + (X \cdot Z)$
$\vdash \forall X Y Z. X + (Y \cdot Z) = (X + Y) \cdot (X + Z)$
$\vdash \forall X Y. X \trianglerighteq Y = (X \triangleright Y) \cdot (X \triangle Y)$
$\vdash \forall X Y Z. X \triangleright (Y + Z) = (X \triangleright Y) + (X \triangleright Z)$
$\vdash \forall X Y. X \triangle Y = Y \triangle X$

$(H|W|C)$ spares. The hot spare possesses the same failure behavior in both its active and dormant states. The cold spare cannot fail in its dormant state and is only activated after the failure of the main part. The failure behavior of the warm spare in the dormant state is attenuated by a dormancy factor from the active state. In order to distinguish between the dormant and active states of the spare, just like the DFT algebra [30], we use two different symbols to model the spare part of the DRBD spare construct, one for the dormant state and the other for the active one. For the



Figure 6.3: Spare Construct

spare construct of Figure 6.3, the spare X is represented by X_a and X_d for the active and dormant states, respectively. After the failure (F) of the main part Y , X will be activated (A) by the spare controller. We model the structure function of the spare construct (Q_{spare}) using the DRBD operators based on the description of its behavior:

$$Q_{spare} = (X_a \triangleright Y) \cdot (Y \triangleright X_d) \quad (6.15)$$

Thus, we need two conditions to be satisfied in order for the spare to work. The first one is that the active state of the spare will continue to work after the failure of the main part ($X_a \triangleright Y$). The second condition is that the main part will continue to work after the failure of the spare in its dormant state ($Y \triangleright X_d$). However, since the spare part can only fail in one of its states (X_a, X_d) but not both as it is non-repairable, only one of the terms in Equation (6.15) affects the behavior and the other term can never fail, i.e., it fails at $+\infty$.

We formally define the warm spare (WSP) as:

Definition 6.12. *DRBD WSP*

$$\vdash \forall Y X_a X_d. \text{ R_WSP } Y X_a X_d = (X_a \triangleright Y) \cdot (Y \triangleright X_d)$$

Since the spare construct of the DRBD and the spare gate of the DFT exhibit complementary behavior, i.e., the DRBDs consider the success and the DFTs consider the failure, we can use the probability of failure of the spare DFT gate [30] to find the reliability of the spare DRBD construct. It is assumed that the dormant spare and

the main part are independent since the failure of one does not affect the failure of the other. However, the failure of the active spare is affected by the time of failure of the main part, since it will be activated after the failure of the main part. We express the reliability of the spare as:

$$R_{\text{spare}}(t) = 1 - \int_0^t \int_y^t f_{(X_a|Y=y)}(x) f_Y(y) dx dy - \int_0^t f_Y(y) F_{X_d}(y) dy \quad (6.16)$$

where $f_{(X_a|Y=y)}$ is the conditional density function of X_a given that Y failed at time y .

We use our formalization of the probability of failure of the warm spare gate of Section 4.2.4 to verify the reliability of the WSP construct:

Theorem 6.5.

$$\begin{aligned} & \vdash \forall p \ Y \ X_a \ X_d \ t \ f_Y \ f_{X_a Y} \ f_{X_a|Y}. \ 0 \leq t \wedge \\ & (\forall s. \ \text{ALL_DISTINCT} [X_a \ s; X_d \ s; Y \ s]) \wedge \text{DISJOINT_WSP} \ Y \ X_a \ X_d \ t \wedge \\ & \text{rv_gt0_ninfinity} [X_a; X_d; Y] \wedge \text{den_gt0_ninfinity} \ f_{X_a Y} \ f_Y \ f_{X_a|Y} \wedge \\ & (\forall y. \ \text{cond_density} \ \text{lborel} \ \text{lborel} \ p \\ & \quad (\text{real} \circ X_a)(\text{real} \circ Y) \ y \ f_{X_a Y} \ f_Y \ f_{X_a|Y}) \wedge \\ & \text{indep_var} \ p \ \text{lborel} \ (\text{real} \circ X_d) \ \text{lborel} \ (\text{real} \circ Y) \wedge \\ & \text{cont_CDF} \ p \ (\text{real} \circ X_d) \wedge \text{measurable_CDF} \ p \ (\text{real} \circ X_d) \Rightarrow \\ & (\text{Rel} \ p \ (\text{R_WSP} \ Y \ X_a \ X_d) \ t) = \\ & 1 - \left(\int_0^t f_Y(y) * \left(\int_y^t f_{(X_a|Y=y)}(x) \ dx \right) dy + \int_0^t f_Y(y) F_{X_d}(y) dy \right) \end{aligned}$$

where `ALL_DISTINCT` ensures that the main and spare parts cannot fail at the same time, `DISJOINT_WSP Y Xa Xd t` ensures that until time t , the spare can only fail in one of its states and `den_gt0_ninfinity` ascertains the proper values of the density functions; joint (f_{XY}), marginal (f_Y) and conditional ($f_{X_a|Y}$). Theorem 6.5 is verified by first defining a conditional density function $f_{X_a|Y}$ for random variables (`real o Xa`)

and (`real o Y`). This is required as the failure of the spare part is affected by the time of failure of the main part. Therefore, we need to define this conditional density function then prove the expression based on the probability of failure of the DFT spare gate, which is verified based on the properties of the Lebesgue integral.

Equations (6.15) and (6.16) represent the general behavior of the spare, i.e., the warm spare. The cold and hot spares represent special cases of the warm spare and can be expressed as:

$$Q_{coldspare} = X_a \triangleright Y \quad (6.17)$$

$$Q_{hotspare} = X + Y \quad (6.18)$$

In Equation (6.18), the spare part X has the same behavior in both states and thus there is no need to use any subscript to distinguish both states.

We formally define the DRBD CSP as:

Definition 6.13. *DRBD CSP*

$\vdash \forall Y X. \text{R_CSP } Y X = (\lambda s. \text{ if } Y s < X s \text{ then } X s \text{ else PosInf})$

This definition means that the CSP construct will continue to work until the main part fails then the spare part is activated and fails in its active state. It is worth noting that since the spare part has only one state that affects the behavior of the CSP, which is the active state, we do not use any subscript with the active state, as the dormant state has no effect here in the behavior.

Finally, we define the hot spare construct (HSP) as:

Definition 6.14. *DRBD HSP*

$\vdash \forall Y X. \text{R_HSP } Y X = (\lambda s. \text{ max } (Y s) (X s))$

The reliability expression of Equation (6.18) can be expressed using the reliability of the OR operator. Therefore, we can use Theorem 6.2 to express the reliability

of the HSP construct. The reliability of the cold spare construct can be expressed as:

$$R_{cold\ spare}(t) = 1 - \int_0^t \int_y^t f_{(X_a|Y=y)}(x) f_Y(y) dx dy \quad (6.19)$$

We verify the reliability of the CSP construct based on the probability of failure of the CSP gate as:

Theorem 6.6.

$$\begin{aligned} \vdash \forall p\ X\ Y\ f_{XY}\ f_Y\ f_{X|Y}\ t. \quad & 0 \leq t \wedge \\ & rv_gt0_ninfinity\ [X; Y] \wedge den_gt0_ninfinity\ f_{XY}\ f_Y\ f_{X|Y} \wedge \\ & (\forall y. \quad cond_density\ lborel\ lborel\ p \\ & \quad (real\ o\ X)(real\ o\ Y)\ y\ f_{XY}\ f_Y\ f_{X|Y}) \wedge \\ (Rel\ p\ (R_CSP\ Y\ X)\ t) = & 1 - (\int_0^t f_Y(y) * (\int_y^t f_{(X|Y=y)}(x)\ dx)\ dy) \end{aligned}$$

The conditions required for this theorem are similar to the ones of Theorem 6.5, as the WSP exhibits the behavior of the CSP if the main part fails before the spare.

6.6 DRBD Structures

Beside the dynamic DRBD constructs, system components are represented as blocks that can be connected in series, parallel, series-parallel and parallel-series fashion, as shown in Figure 6.4 [4]. Each block in Figure 6.4 represents either a simple system component or one of the DRBD dynamic constructs.

The series structure (Figure 6.4(a)) represents a collection of blocks that are connected in series. The system continues to work until the failure of one of these blocks. We define a series structure that represents the intersection of all events of the blocks in this structure as in Table 6.2, where X_i represents the i^{th} block in the series structure and n is the number of blocks. Interestingly, any block in our

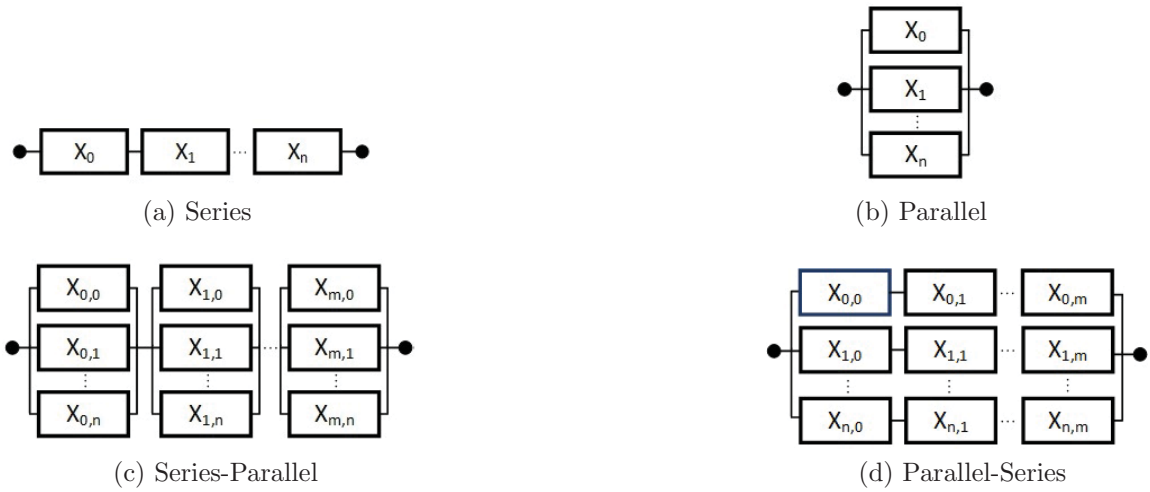


Figure 6.4: DRBD Structures

proposed algebra can represent a basic system component or a complex structure, such as a spare construct. Moreover, since we are dealing with the events, we can use the ordinary reliability expressions for the series structure assuming the independence of the individual blocks. The parallel structure (Figure 6.4(b)) represents a system that continues to work until the failure of the last block in the structure. The behavior of the parallel structure can be expressed using the OR operator. We represent the parallel structure as the union of the individual events of the blocks. The series-parallel structure (Figure 6.4(c)) represents a series structure, where the blocks of the series structure are parallel structures. The structure function of this structure can be expressed using an AND of OR operators. Table 6.2 lists the model for this structure with its reliability expression, where n is the number of blocks in the parallel structure and m is the number of parallel structures that are connected in series. The parallel-series structure (Figure 6.4(d)) represents a group of series structures that are connected in parallel. Its structure function can be expressed using an OR of AND operators.

Table 6.2: Mathematical and Reliability Expressions of DRBD Structures

Structure	Math. Model	Reliability Expression
Series	$\bigcap_{i=1}^n (\text{event } (X_i, t))$	$\prod_{i=1}^n R_{X_i}(t)$
Parallel	$\bigcup_{i=1}^n (\text{event } (X_i, t))$	$1 - \prod_{i=1}^n (1 - R_{X_i}(t))$
Series-Parallel	$\bigcap_{i=1}^m \bigcup_{j=1}^n (\text{event } (X_{(i,j)}, t))$	$\prod_{i=1}^m (1 - \prod_{j=1}^n (1 - R_{X_{(i,j)}}(t)))$
Parallel-Series	$\bigcup_{i=1}^n \bigcap_{j=1}^m (\text{event } (X_{(i,j)}, t))$	$1 - (\prod_{i=1}^n (1 - \prod_{j=1}^m (R_{X_{(i,j)}}(t))))$

We formally define the series structure as:

Definition 6.15. *DRBD Series Structure*

$$\vdash \forall Y \ s. \ \text{DRBD_series } Y \ s = \bigcap_{i \in s} (Y \ i)$$

We define the series structure as a function that accepts a group of sets, Y , that are indexed by the numbers in set s and returns the intersection of these sets.

The parallel structure is defined in a similar way but it returns the union of the sets rather than the intersection. We formally define it as:

Definition 6.16. *DRBD Parallel Structure*

$$\vdash \forall Y \ s. \ \text{DRBD_parallel } Y \ s = \bigcup_{i \in s} (Y \ i)$$

The group of sets, Y , in both structures, represents a family of events, i.e., Y will be instantiated later with DRBD events. The reliability expressions of the series and parallel structures are given in Table 6.2. We verify these expressions as:

Theorem 6.7.

$$\begin{aligned} &\vdash \forall p \ X \ t \ s. \ s \neq \{\} \wedge \text{FINITE } s \wedge \\ &\text{indep_sets } p \ (\lambda i. \{\text{rv_to_event } p \ X \ t \ i\}) \ s \Rightarrow \\ &(\text{prob } p \ (\text{DRBD_series } (\text{rv_to_event } p \ X \ t) \ s) = \\ &\text{Normal } (\prod_{i \in s} (\text{real } (\text{Rel } p \ (X \ i) \ t)))) \end{aligned}$$

Theorem 6.8.

$$\begin{aligned} &\vdash \forall p \ X \ t \ s. \ s \neq \{\} \wedge \text{FINITE } s \wedge \\ &\quad \text{indep_sets } p \ (\lambda i. \{\text{rv_to_event } p \ X \ t \ i\}) \ s \Rightarrow \\ &\quad (\text{prob } p \ (\text{DRBD_parallel } (\text{rv_to_event } p \ X \ t) \ s) = \\ &\quad 1 - \text{Normal } (\prod_{i \in s} (\text{real } (1 - \text{Rel } p \ (X \ i) \ t)))) \end{aligned}$$

where $s \neq \{\} \wedge \text{FINITE } s$ ensures that the set of indices, s , is nonempty and finite.

The reliability of the series structure is verified based on the independence of the input events using `indep_sets`, which ensures that for the probability space p , the given group of sets $(\lambda i. \{\text{rv_to_event } p \ X \ t \ i\})$ indexed by the numbers in set s are independent. The family of sets $(\lambda i. \{\text{rv_to_event } p \ X \ t \ i\})$ represents the DRBD events of the group of time-to-failure functions, X . This is defined as:

Definition 6.17. *rv_to_event*

$$\vdash \forall p \ X \ t. \ \text{rv_to_event } p \ X \ t = (\lambda i. \text{DRBD_event } p \ (X \ i) \ t)$$

The function `rv_to_event` enables us to create the group of `DRBD_event` of time-to-failure functions of system blocks (X). Based on the independence of these sets and the definition of the series structure (intersection of sets), we verify that the probability of the series structure is equal to the product of the reliability of the individual blocks $(\text{Rel } p \ (X \ i) \ t)$, where $i \in s$. The product function (Π) in HOL4 returns a real value and the probability returns `extreal`, therefore, it is required to typecast the product function to `extreal` using `Normal`. Similarly, the product function finds the product of real-valued functions, therefore, it is required to typecast the reliability function (Rel) to real using the `real` function. The parallel structure is verified in a similar way. We replace the parallel structure (the union of events) with the complement of the intersection of the complements of the events. Then, we verify that the probability of this complement equals one minus the probability of the

intersection of the complements. This requires the added condition that all DRBD events created using `rv_to_event` belong to the events of the probability space `p`, which is an embedded condition in `indep_sets` definition.

In order to express the series and parallel structures using DRBD operators, we verify that these structures are equal to the DRBD events of the `nR_AND` and `nR_OR`, respectively:

Theorem 6.9.

$$\vdash \forall p \ X \ t \ s. \text{FINITE } s \wedge s \neq \{\} \Rightarrow \\ (\text{DRBD_event } p \ (\text{nR_AND } X \ s) \ t = \text{DRBD_series } (\text{rv_to_event } p \ X \ t) \ s)$$

Theorem 6.10.

$$\vdash \forall p \ X \ t \ s. \text{FINITE } s \wedge 0 \leq t \Rightarrow \\ (\text{DRBD_event } p \ (\text{nR_OR } X \ s) \ t = \text{DRBD_parallel } (\text{rv_to_event } p \ X \ t) \ s)$$

We verify Theorems 6.9 and 6.10 by inducting on set `s` using `SET_INDUCT_TAC` that will create two subgoals to be solved; one for the empty set and another one for inserting an element to a finite set. Furthermore, we use the fact that the DRBD events of the AND and OR operators equal the intersection and the union of the individual events, respectively. For Theorem 6.10, an additional condition is required, $0 \leq t$, to be able to manipulate the sets and reach the final form of the theorem.

Interestingly, these structures can be easily extended to model and verify more complex structures, such as two-level structures, i.e., series-parallel and parallel series structures. We formally verify the reliability of the series-parallel structure as:

Theorem 6.11.

$$\vdash \forall p \ X \ t \ s \ J. \\ \text{indep_sets } p \ (\lambda i. \{\text{rv_to_event } p \ X \ t \ i\}) \ (\bigcup_{j \in J} (s \ j)) \wedge$$

$$\begin{aligned}
& (\forall i. i \in J \Rightarrow s\ i \neq \{\} \wedge \text{FINITE } (s\ i)) \wedge \\
& \text{FINITE } J \wedge J \neq \{\} \wedge \text{disjoint_family_on } s\ J \Rightarrow \\
& (\text{prob } p \\
& \quad (\text{DRBD_series} \\
& \quad \quad (\lambda j. \text{DRBD_parallel } (\text{rv_to_event } p\ X\ t) (s\ j))\ J) = \\
& \text{Normal} \\
& \quad (\prod_{j \in J} (1 - \prod_{i \in (s\ j)} (\text{real } (1 - \text{Rel } p\ (X\ i)\ t))))))
\end{aligned}$$

We formally verify the reliability of the parallel-series structure as:

Theorem 6.12.

$\vdash \forall p\ X\ t\ s\ J.$

$$\begin{aligned}
& \text{indep_sets } p\ (\lambda i. \{\text{rv_to_event } p\ X\ t\ i\}) \left(\bigcup_{j \in J} (s\ j) \right) \wedge \\
& (\forall i. i \in J \Rightarrow s\ i \neq \{\} \wedge \text{FINITE } (s\ i)) \wedge \\
& \text{FINITE } J \wedge J \neq \{\} \wedge \text{disjoint_family_on } s\ J \Rightarrow \\
& (\text{prob } p \\
& \quad (\text{DRBD_parallel } (\lambda j. \text{DRBD_series } (\text{rv_to_event } p\ X\ t) (s\ j))\ J) = \\
& \quad 1 - \text{Normal } (\prod_{j \in J} (1 - \prod_{i \in (s\ j)} (\text{real } (\text{Rel } p\ (X\ i)\ t))))))
\end{aligned}$$

The main idea in building these two-level structures is to partition the family of blocks into distinct groups, where we use a set, J , to index these partitions, i.e., it includes the number of groups in the first top level. Then, for each group in this top level, we have another set, $\{s\ j \mid j \in J\}$, that includes the indices of the blocks in the second level, i.e. the subgroups. For example, consider the parallel-series structure of Figure 6.4(d), if $n = m = 1$, then the outer parallel structure has two series structures, where each series structure has two blocks. Thus, $J = \{0; 1\}$. For each $j \in J$, we have a certain set $s\ j$ that has the indices of the blocks in the inner

series structure. Thus, $\mathbf{s} = (\lambda j. \text{ if } j = 0 \text{ then } \{0;1\} \text{ else } \{2;3\})$. The same concept is applied to the series-parallel structure. Therefore, the structure of the DRBD can be determined based on the given sets of indices.

We verify Theorems 6.11 and 6.12 by extending the proofs of the series and parallel structures. However, it is required to deal with the intersection of unions in case of the series-parallel structure and the union of intersections in case of parallel-series structure. Therefore, we need to extend the independence of sets properties to include the independence of union and intersection of partitions of the events. We verify these properties as:

Theorem 6.13.

$$\begin{aligned} &\vdash \forall p \ \mathbf{s} \ J \ Y. \text{ indep_sets } p \ (\lambda i. \{Y \ i\}) \bigcup_{j \in J} (\mathbf{s} \ j) \wedge J \neq \{\} \wedge \\ &\quad (\forall i. i \in J \Rightarrow \text{countable } (\mathbf{s} \ i)) \wedge \text{FINITE } J \wedge \\ &\quad \text{disjoint_family_on } \mathbf{s} \ J \Rightarrow \\ &\quad \text{indep_sets } p \ (\lambda j. \{\bigcup_{i \in \mathbf{s} \ j} (Y \ i)\}) \ J \end{aligned}$$

Theorem 6.14.

$$\begin{aligned} &\vdash \forall p \ \mathbf{s} \ J \ Y. \text{ indep_sets } p \ (\lambda i. \{Y \ i\}) \bigcup_{j \in J} (\mathbf{s} \ j) \wedge J \neq \{\} \wedge \\ &\quad (\forall i. i \in J \Rightarrow \text{countable } (\mathbf{s} \ i) \wedge \mathbf{s} \ i \neq \{\}) \wedge \text{FINITE } J \wedge \\ &\quad \text{disjoint_family_on } \mathbf{s} \ J \wedge (\forall i. i \in \bigcup_{j \in J} (\mathbf{s} \ j) \Rightarrow Y \ i \subset \text{m_space } p) \Rightarrow \\ &\quad \text{indep_sets } p \ (\lambda j. \{\bigcap_{i \in \mathbf{s} \ j} (Y \ i)\}) \ J \end{aligned}$$

where set J includes the indices of the partitions and \mathbf{s} has the indices of the individual blocks of each partition, `disjoint_family_on` ensures that the indices of the blocks in different partitions are disjoint and `indep_sets p (λi. {Y i}) ⋃j∈J (s j)` ensures the independence of the family of blocks $\{Y \ i\}$ where the indices of the individual blocks are given by the union of \mathbf{s} . In order to verify Theorems 6.13 and 6.14, we need the fact that the σ -algebras generated by $(\lambda j. \bigcup_{i \in \mathbf{s} \ j} \{Y \ i\})$ with index set J

are independent. Then, we verify that $\forall j. j \in J, \text{set } \{\bigcup_{i \in s_j} \{Y_i\}\}$ is a subset of the σ -algebra generated by $\bigcup_{i \in s_j} \{Y_i\}$. Finally, based on these intermediate verified steps and the definition of `indep_sets`, we are able to verify these theorems.

In order to verify the reliability of the series-parallel structure, we need to ensure the independence of the individual blocks. Therefore it is required to combine the indices of all blocks into a single set using $\bigcup_{j \in J} (s_j)$ to be used with `indep_sets`. To be able to use the reliability of the series structure in this proof, we use Theorem 6.13 to verify the independence of the unions of partitions of events. This means verifying that the parallel structures are independent, i.e., the probability of intersection of these parallel structures equals the product of the reliability of the parallel structures. Finally, several assumptions related to sets $\{s_i \mid i \in J\}$ and J are required, which include that these sets are finite and nonempty. Finally, it is required that every block has a unique index, which is ensured using `disjoint_family_on`. The reliability of the parallel-series structure is verified in a similar manner based on the reliability of the parallel structure. We verify the independence of the intersection of partitions of events rather than the union using Theorem 6.14. In addition, it is required that all DRBD events belong to the events of the probability space.

We extend the reliability of the two-level series-parallel structure to verify the reliability of a more nested structure, i.e., series-parallel-series-parallel, as:

Theorem 6.15.

$\vdash \forall p \ X \ t \ s \ L \ A \ J.$

`indep_sets` p $(\lambda i. \{\text{rv_to_event } p \ X \ t \ i\})$ $(\text{nested_BIGUNION } s \ L \ A \ J) \wedge$
`sets_finite_not_empty` $s \ L \ A \ J \Rightarrow$
 $(\text{prob } p$
 $(\text{DRBD_series } (\lambda j.$

DRBD_parallel ($\lambda a.$

DRBD_series ($\lambda l.$

DRBD_parallel (rv_to_event p X t) (s l)) (L a)) (A j)) J) =

Normal

$$\left(\prod_{j \in J} \left(1 - \prod_{a \in (A \ j)} \left(1 - \prod_{l \in (L \ a)} \left(1 - \prod_{i \in (s \ l)} (\text{real } (1 - \text{Rel } p (X \ i) \ t))) \right) \right) \right) \right)$$

For this four-level nested structure, we have four sets (indexed sets) that determine the structure of the DRBD, which are: J , A , L and s . This is similar to the two-level nested structure but with a deeper hierarchy. Therefore, in order to combine the indices of all the individual blocks in the DRBD in a single set, we define `nested_BIGUNION s L A J` to union the elements of all $s \ i$, where $i \in L \ a$, $a \in A \ j$ and $j \in J$. This is done in a hierarchical manner and can be extended easily to deeper levels. We use the previously mentioned function to ensure that all the individual events belong to the probability events and are independent as well. Moreover, it is required to ensure that the sets are finite, disjoint and nonempty, just like the series-parallel structure. We combine these set-related conditions using the function `sets_finite_not_empty`. Finally, we verify Theorem 6.15 within two main steps. The first step is to verify the reliability of the outer series-parallel, which requires verifying the independence of the intersection of union of partition of the DRBD blocks, i.e., the inner series-parallel structures are independent. The second step is to verify the reliability of the inner series-parallel structures, which can be done based on some set manipulation. This theorem can be used to verify even deeper structures, which would require verifying the independence of more nested structures. We use Theorem 6.15 to verify the reliability of the series-parallel-series structure as it represents a special case of the series-parallel-series-parallel, where each of the innermost parallel

structures has only one block. Our formalization follows the natural definitions of parallel and series structures. Moreover, our verified lemmas of independence allow verifying deeper structures, which makes our formalization flexible and applicable to model the most complex systems. The proof script of the DRBD algebra is available at [70]. In the following section, we utilize our formalization in the verification of the reliability of the DBW system.

6.7 Formal DBW DRBD Analysis

To demonstrate the applicability of our proposed DRBD algebra, we present the formal reliability analysis of the DBW system [59] to verify generic expressions that are independent of the failure distribution of the system components, i.e., we can use different types of distributions to model the failure of system components as long as they satisfy the required conditions, such as the continuity.

The DRBD of the DBW system, shown in Figure 6.5, models the successful behavior. Similar to our DFT analysis of the DBW, we provide the analysis of the throttle and brake subsystems. The throttle subsystem continues to work as long as the throttle (TF) and the engine (EF) are working. In addition, the system successful

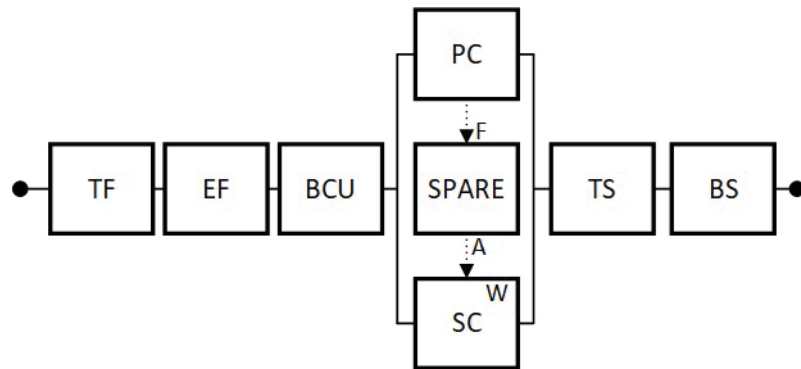


Figure 6.5: DRBD of Drive-by-Wire System

operation requires the operation of the brake control unit (BCU). The system includes a primary control unit (PC) with a warm spare (SC) that replaces the main part after failure. Finally, the system needs the operation of the throttle sensor (TS) and the brake sensor (BS). The DRBD of this system is modeled as a series structure with a spare construct. We express the structure function of this DRBD using our operators:

$$Q_{DBW} = TF \cdot EF \cdot BCU \cdot (R_WSP \ PC \ SC_a \ SC_d) \cdot TS \cdot BS$$

Since this is a series DRBD, then the cut sets and cut sequences are easily determined using all the blocks of the system.

Then, we verify the DBW reliability as:

Theorem 6.16.

$$\vdash \forall p \ TF \ EF \ BCU \ PC \ SC_a \ SC_d \ TS \ BS \ t.$$

$$DBW_set_req \ p \ TF \ EF \ BCU \ PC \ SC_a \ SC_d \ TS \ BS \ t \Rightarrow$$

$$(\text{prob } p \ (\text{DRBD_event } p \ Q_{DBW} \ t) =$$

$$\text{Rel } p \ TF \ t \ * \ \text{Rel } p \ EF \ t \ * \ \text{Rel } p \ BCU \ t \ * \ \text{Rel } p \ (R_WSP \ PC \ SC_a \ SC_d) \ t \ * \$$

$$\text{Rel } p \ TS \ t \ * \ \text{Rel } p \ BS \ t)$$

where `DBW_set_req` ensures the proper conditions for the independence of the blocks in the DBW system.

In Figure 6.6, we evaluate, using MATLAB, the reliability of the DBW system assuming exponential distributions for the system components with failure rates as given in the figure and a dormancy factor of 0.5.

6.8 Formal Equivalence of DFT-DRBD Algebras

The proposed framework integrating DFT and DRBD algebras is depicted in Figure 6.7. As mentioned in Section 1.4, the proposed methodology can be utilized to

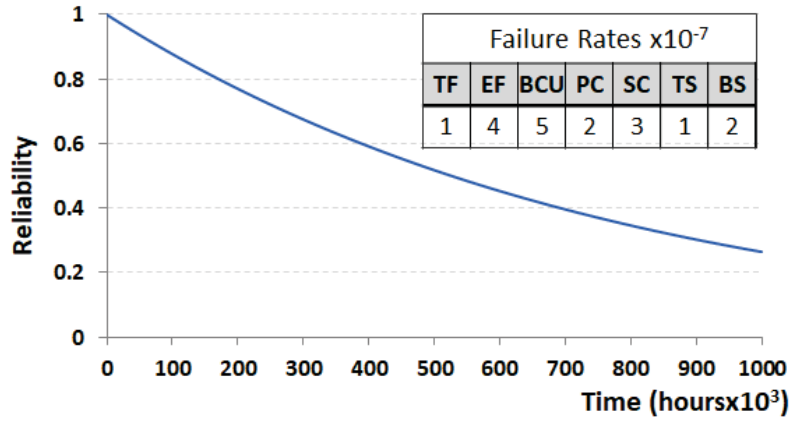


Figure 6.6: Reliability of DBW System

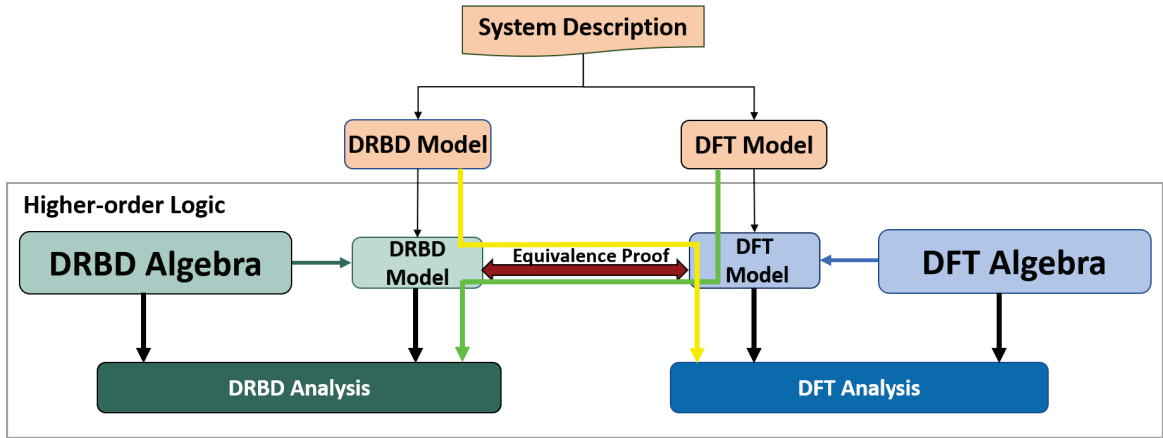


Figure 6.7: Integrated Framework for Formal DFT-DRBD Analysis using HOL4

conduct both DFT and DRBD analyses using the HOL formalized algebras and allows formally converting a DFT model into its corresponding DRBD based on the equivalence of both algebras. The analysis starts by a given system description that can be modeled as a DFT or DRBD. Formal models of the given system can be created based on the HOL formalized algebras. The DRBD model can be analyzed as described in this chapter, where a DRBD event is created and its reliability is verified based on the available verified theorems of DRBD algebra. On the other hand, a DFT model can be analyzed using the formalized DFT algebra presented in Chapters 3 and 4.

Furthermore, the DRBD model can be converted to a DFT to model the failure instead of the success, then this model is analyzed using the DFT algebra. Similarly, the DFT model can be analyzed by converting it to its counterpart DRBD model to analyze the success.

In order to handle the DFT analysis using DRBD algebra and the DRBD analysis using the DFT algebra, it is required to be able to represent the DRBD of the corresponding DFT gates using the DRBD algebra and vice-versa (the equivalence proof in Figure 6.7). According to [43], the OR, AND and FDEP gates can be represented using series, parallel and series RBDs, respectively. Therefore, they can be modeled using AND and OR operators, while the spare gate corresponds to the spare construct. Finally, the PAND gate can be expressed using the inclusive after operator ($Y \supseteq X$). However, we need to formally verify this equivalence to ensure its correctness. In Table 6.3, we provide the theorems of equivalence of DFT gates and DRBD operators and constructs, where `D_AND`, `D_OR`, `n_OR`, `n_AND` `FDEP`, `P_AND` and `WSP` are the names of the AND, OR, n -ary OR, n -ary AND, FDEP, PAND and spare DFT gates in our HOL formalization, respectively. `R_WSP`, `nR_OR`, `nR_AND` are the names of the spare DRBD construct, n -ary DRBD OR and n -ary DRBD AND operators, respectively, in our formalized DRBD. `ALL_DISTINCT [Y Xa Xd]` ensures that the inputs cannot fail at the same time. The proof script of these verified theorems is available at [71].

We need to recall that the DFT n -ary gates accept lists of random variables. Whereas the DRBD n -ary operators accept an indexed group of random variables with their indices in another set. Therefore, we use `(MAP X (SET_TO_LIST s))` to create a list of random variables from the group of indexed random variables.

In order to use these verified expressions in Table 6.3, we need to verify that the `DRBD_event` and the `DFT_event` possess complementary sets in the probability space.

Table 6.3: Verified Equivalence of DFT and DRBD Algebras

DFT Gate	DRBD Operator/Construct	Verified Theorem
AND	OR	$\vdash \forall X Y. D_AND X Y = R_OR X Y$
OR	AND	$\vdash \forall X Y. D_OR X Y = R_AND X Y$
n_AND	nR_OR	$\vdash \forall X s. FINITE s \Rightarrow$ $(n_AND (MAP X (SET_TO_LIST s))) =$ $nR_OR X s)$
n_OR	nR_AND	$\vdash \forall X s. FINITE s \Rightarrow$ $(n_OR (MAP X (SET_TO_LIST s))) =$ $nR_AND X s)$
FDEP	AND	$\vdash \forall X Y. FDEP X Y = R_AND X Y$
PAND	Inclusive After	$\vdash \forall X Y. P_AND X Y =$ $R_INCLUSIVE_AFTER Y X$
Spare	Spare	$\vdash \forall X_a X_d Y. (\forall s.$ $ALL_DISTINCT [Y s; X_a s; X_d s]) \Rightarrow$ $(WSP Y X_a X_d = R_WSP Y X_a X_d)$

We formally verify this as:

Theorem 6.17.

$$\vdash \forall p X t. \text{prob_space } p \wedge (DFT_event \ p \ X \ t) \in \text{events } p \Rightarrow$$

$$(\text{prob } p \ (DRBD_event \ p \ X \ t) = 1 - \text{prob } p \ (DFT_event \ p \ X \ t))$$

where the conditions ensure that p is a probability space and that the DFT event belongs to the events of the probability space. This theorem can be verified also if we ensure that the DRBD event belongs to the probability space. This theorem means that for the same time-to-failure function, the DRBD and DFT events are the complements of each other. This way, we can analyze DFTs using the DRBD algebra and vice-versa.

Based on the verification results obtained in Table 6.3, DFT gates can be formally represented using DRBDs. We show that the amount of effort required by the reliability engineer to formally analyze DFTs by analyzing its counterpart DRBD is less than that of analyzing the original DFT model. In Chapter 4, a DFT is formally analyzed using the DFT algebra by expressing the DFT event of the structure function as the union of the individual DFT events. Then, the probabilistic PIE is utilized to formally verify the probability of failure of the top event. The number of terms in the final result equals $2^n - 1$, where n is the number of individual events in the union of the structure function. Therefore, in the verification process, it is required to verify at least $2^n - 1$ expressions if the PIE is to be used. On the other hand, verifying a DRBD would require verifying a single expression for each nested structure.

As an example, consider the reliability analysis of the DBW system. Analyzing the DFT of this system required verifying 63 subgoals as the top event is composed of the union of six different events. While analyzing the DRBD of the DBW system required verifying only one main subgoal to be manipulated to reach the final goal. Table 6.4 provides a comparison of the size of the script, the required time to develop it and the number of goals to be verified. Based on these observations, analyzing the reliability of the DBW using the DRBD required 1/24 of the time needed by the DFT based on the probabilistic PIE. These results show that it is more convenient to analyze the DRBD of a system rather than its DFT if the algebraic approach and the probabilistic PIE are to be used. The only added step will be to formally verify that the DFT and DRBD are the complements of each other, which is straightforward utilizing the theorems in Table 6.3. Therefore, we verify this as:

Table 6.4: Comparison of Formal Analysis Efforts of DBW

	# of subgoals	# of lines in the script	required time
DFT	63	4850	24 hours
DRBD	1	150	1 hour

Theorem 6.18.

$\vdash \forall p \text{ TF EF BCU PC SC}_a \text{ SC}_d \text{ TS BS } t .$

$\text{prob_space } p \wedge \text{DBW_events_p } p \text{ TF EF BCU PC SC}_a \text{ SC}_d \text{ TS BS } t \Rightarrow .$

$(\text{prob } p (\text{DRBD_event } p \text{ F}_{\text{DBW}} t) = 1 - \text{prob } p (\text{DFT_event } p \text{ Q}_{\text{DBW}} t))$

where `DBW_events_p` ensures that the DBW DFT events are in the events of the probability space. Thus, we can use the DRBD reliability expression (Theorem 6.16) to verify the probability of failure of the DFT, which results in a reduction in the analysis efforts.

6.9 Summary

In this chapter, we proposed a new algebra to analyze DRBDs. We developed the HOL formalization of this algebra in HOL4, which ensures its correctness and allows conducting the analysis within a theorem prover. Furthermore, this algebra provides formalized generic expressions of reliability that cannot be verified using other formal tools. This HOL formalization is the first of its kind that takes into account the system dynamics by providing the HOL formal model of spare constructs and temporal operators. The proposed algebra is compatible with the reliability expressions of traditional RBDs as demonstrated by the reliability expressions of the series and parallel structures. It also facilitates extending the verified reliability expressions to model complex systems using nested structures. We demonstrated the usefulness of this formalized algebra by formally conducting the analysis of the DBW system

to verify a generic expression of its reliability, which is independent of the failure probability distribution of system components. Finally, we verified the equivalence of the DFT and DRBD algebras and their gates and constructs, which allows verifying DFT models using the DRBD algebra and vice-versa.

One of the main challenges that we faced is the lack of a DRBD algebra that enables the analysis based on the structure function of the DRBD. Having this algebra and the reliability expressions of the DRBD structures and constructs enables the development of a framework, using a theorem prover, to formally conduct the analysis. Thus, we proposed this novel DRBD algebra, including the temporal operators, simplification theorems, which allows expressing the structure of the DRBD based on system blocks. Another challenge is the formalization of this algebra in HOL, particularly the traditional structures. We needed to formally define these structures in a manner that is easily understood by users that are not familiar with theorem proving, but at the same time to be compatible with the dynamic aspects that are captured using the rest of the definitions, like the spare construct. Furthermore, we needed to create these formal definitions to be compatible with existing theories in HOL4, such as the probability and the Lebesgue integral, in particular the independence of random variables that is of great importance to be able to extend these structures to model more nested ones and verify their reliability expressions.

Chapter 7

Formal Dependability Analysis of Shuffle-exchange Networks

7.1 Overview

With the ongoing demands for intensive processing applications, multiprocessor systems represent one of the solutions that satisfies such demand. Nowadays, such systems are feasible due to their reduced cost and thus it is possible to have systems of hundreds of processors. Multiprocessor systems allow parallel computing, where tasks are executed in parallel with the possibility of interacting with one another when required. This parallel execution highly impacts the overall system performance, such as throughput. However, memory and I/O peripheral resources are shared among processors and thus an efficient data routing among system nodes is necessary to maintain high system performance, reliability and low cost. This is of a great importance, particularly with scientific applications, where a huge number of processors

are used, i.e., large-scale multiprocessor systems [72]. Therefore, a dedicated interconnection network is used to connect processors and memory modules, as depicted in Figure 7.1 [72].

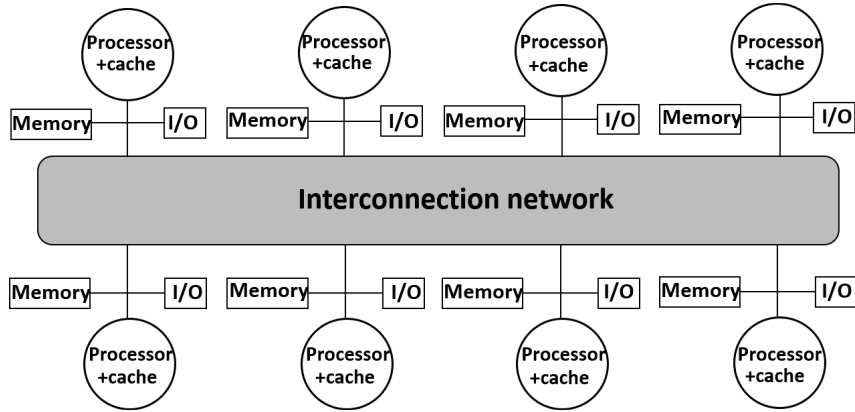


Figure 7.1: Overview of Multiprocessor System Architecture

The complexity of interconnection networks ranges from simple networks, such as time-shared bus to crossbar switching. The former has a negative impact on the system performance, while the latter has much higher cost as there exists a separate link between each pair of nodes in the systems. For example, for a system of N nodes, i.e., N inputs and N outputs, it is required to have N^2 links or switching elements between each input and output.

Multistage interconnection networks (MINs) are introduced to reduce the number of required switching elements and hence, reduce the cost while providing better performance than shared-bus networks. The main idea of MINs is to have multiple small stages of crossbar switches that are connected between sources (inputs) and destinations (outputs), which results in a much reduced number of used switching elements. The number of paths available between each input and output determines the category of the MIN. A single-path MIN has only one path to route information

between each source-destination pair. A shuffle-exchange network (SEN) is an example of such type of networks. Each stage has $N/2$ switching elements, where N is the number of inputs and outputs of the network. Usually the switching elements are of size 2×2 to reduce the cost. The number of stages required to establish the single-path MIN is $\log_2 N$, which is lower than crossbar networks. An 8×8 SEN is shown in Figure 7.2, where only a single path is available for each input-output pair. However, the reliability of single-path MINs and SENs depends on the switching elements and thus a fault in any of these switches cannot be tolerated.

Enhancing the reliability of MINs is of great importance in order to maintain high system performance. Therefore, redundant switching elements are used to ensure that the network is able to provide the required switching even after the failure of some of these elements [73, 74]. Multiple-path MINs are used to increase the fault tolerance and hence the network reliability. SEN+ is a SEN, where an additional stage is added to provide two paths between each input-output pair, as shown in Figure 7.3. However, even with the additional path, the failure of some switches can lead to the failure of the connection in some situations. Spare parts have been used in [75] to replace switches after failure. However, the analysis was not conducted formally to ensure its correctness.

Studying the reliability of SENs has been an active research area [76, 77, 78, 79]. The reliability of MINs are commonly analyzed using simulation or analytically. For example, in [80], Monte Carlo simulation is used to analyze the reliability of SENs. However, as mentioned previously, simulation cannot provide accurate results due to its sampling based nature. Although CTMCs can analytically solve the reliability of MINs [24], they cannot be used with large-scale systems since the state space grows exponentially with the increase in the number of system components. On the

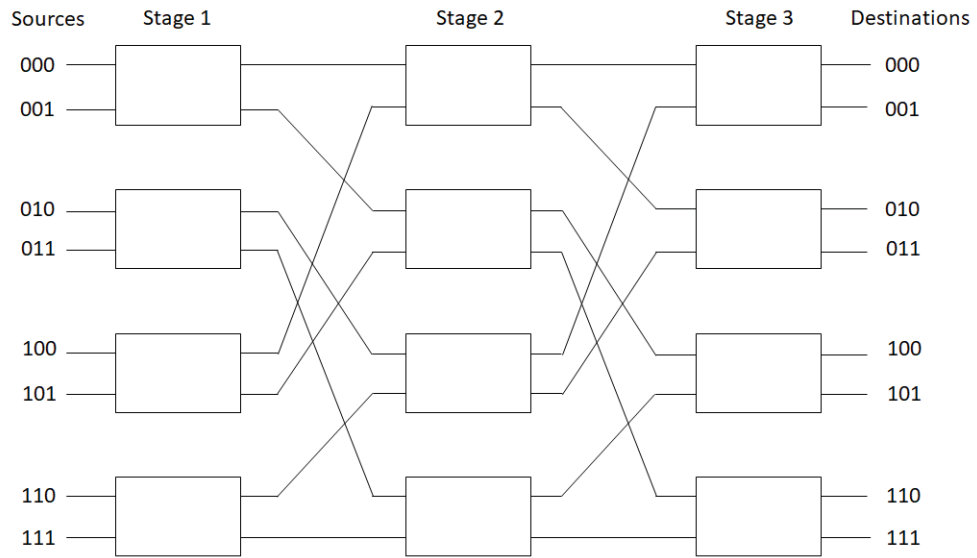


Figure 7.2: An 8×8 SEN

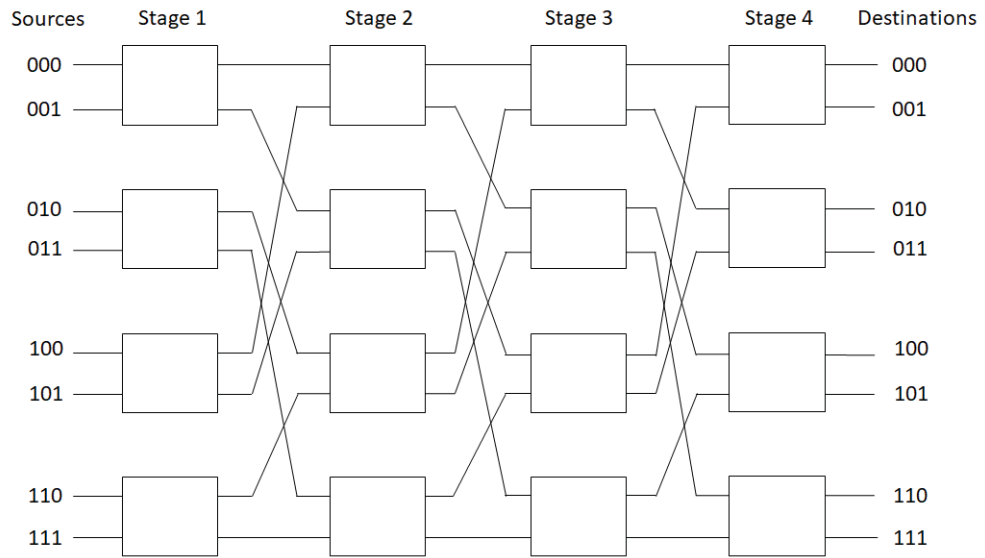


Figure 7.3: An 8×8 SEN+

other hand, when the complexity of the network increases, reliability bounds provides estimate values for the MIN reliability [81, 82]. RBDs have been also used in the analysis of MINs with single and multiple paths. For example, in [25], the reliability of SEN, SEN+ and SEN+2 (a SEN with two additional stages) is modeled using traditional RBDs. Generic expressions of success rates of the switching elements are provided analytically assuming that all these elements have the same failure rates. However, these generic expressions are not formally verified, which may raise questions about its accuracy. Furthermore, dynamic dependencies among system components, like warm spares, are not considered or modeled.

Based on the previous discussion, accurate modeling and analysis of these networks is necessary to capture the dynamic behavior as this will provide the design engineers with some measures that can help enhancing the performance of the entire multiprocessor system. To the best of our knowledge, dynamic dependability analysis using formal methods has not been used with MINs. Therefore, we propose to add spare switches to replace the critical ones after failure and conduct the analysis of MINs, particularly SENs using our formal dependability framework.

Since the reliability of MINs affects the performance of the overall multiprocessor system, it is required to accurately model and analyze their reliability. In this thesis, we use both DRBDs and DFTs to model the dynamic reliability of these networks, particularly SEN and SEN+, and conduct the analysis using our framework. In this chapter, we formally verify the terminal, broadcast and network reliability of SEN and SEN+ in HOL and provide generic expressions of reliability and probability of failure.

7.2 Terminal Reliability Analysis of Shuffle-exchange Networks

The terminal reliability is the reliability of the connection between a given source and destination, i.e., the probability of having a reliable connection between one source-destination pair. We analyze the terminal reliability of the SEN and SEN+ using both DFT and DRBD models.

7.2.1 DFT Analysis of SEN and SEN+

We model the sources of failure of both SEN and SEN+ using DFTs. We use n -ary gates, which enable verifying expressions of the probability of failure for generic number of system components.

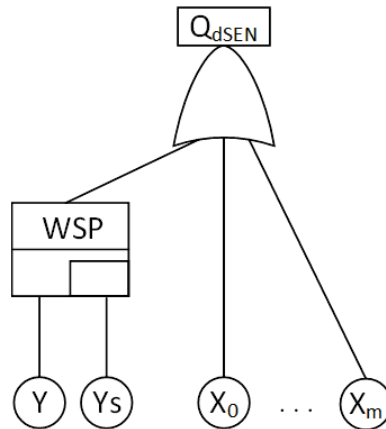


Figure 7.4: DFT of SEN

Figure 7.4 shows the DFT model of the SEN system. Since SENs are single path MINs, the failure of any of the switches in the path between a given source and destination leads to losing the connection. Therefore, adding spare parts will lower the probability of failure. For illustration purposes, we use a spare part to replace the

main switch Y after failure. The DFT consists of an n -ary OR gate, which means that the failure of any of the switches, interrupts the connection between the source and the destination.

Since the top event is an n -ary OR gate, we need first to verify that the DFT_event of the n -ary OR is equal to the union of the individual events, as:

Theorem 7.1.

$$\begin{aligned} \vdash \forall p \ X \ t \ s. \ \text{FINITE } s \Rightarrow \\ (\text{DFT_event } p \ (\text{n_OR } (\text{MAP } X \ (\text{SET_TO_LIST } s)))) \ t = \\ \bigcup_{i \in s} \{\text{rv_to_devent } p \ X \ t \ i\} \end{aligned}$$

where s is a set of numbers that has the indices of the system components. X is a group of random variables that represent the time-to-failure of the switches in the system. We need to recall that n_OR accepts a list of random variables as an argument. Therefore, we create this list using $\text{MAP } X \ (\text{SET_TO_LIST } s)$. rv_to_devent , in Theorem 7.1, is similar to the rv_to_event of the DRBD, but it creates DFT events. It is defined as:

Definition 7.1. *rv_to_devent*

$$\vdash \forall p \ X \ t. \ \text{rv_to_devent } p \ X \ t = (\lambda i. \ \text{DFT_event } p \ (X \ i) \ t)$$

This way, we can use this function to create a group of DFT events for a set of indexed random variables. Then, we verify the probability of the n -ary OR gate in a way similar to the probability of the DRBD parallel structure, which is defined as the union of events.

Theorem 7.2.

$$\begin{aligned} \vdash \forall p \ X \ t \ s. \ s \neq \{\} \wedge \text{FINITE } s \\ \text{indep_sets } p \ (\lambda i. \ \{\text{rv_to_devent } p \ X \ t \ i\}) \ s \wedge \end{aligned}$$

$$\begin{aligned}
& (\forall i. i \in s \Rightarrow \text{rv_gt0_ninfinitiy } [X \ i]) \Rightarrow \\
& (\text{prob } p \ (\text{DFT_event } p \ (\text{n_OR } (\text{MAP } X \ (\text{SET_TO_LIST } s)))) \ t) = \\
& 1 - \text{Normal } \left(\prod_{i \in s} (\text{real } (1 - F_{X_i}(t))) \right)
\end{aligned}$$

In Theorem 7.2, it is required that the set of indices, s , to be nonempty and to be finite, which is a realistic condition as in any system the number of components is finite. The last condition of Theorem 7.3, ensures that the random variables of X are greater than or equal to 0 and not equal to $+\infty$, which is required to be able to use the CDF of the random variable using Theorem 4.1.

We express the structure function of the DFT of SEN as:

$$\begin{aligned}
Q_{\text{dSEN_Terminal}} = \text{n_OR } (\text{MAP } (\lambda i. \quad & \text{if } i = 0 \text{ then WSP } Y \ Y_{s_a} \ Y_{s_d} \\
& \text{else } X \ i) \ (\text{SET_TO_LIST } \{0\} \cup L)) \tag{7.1}
\end{aligned}$$

We notice that the structure of the DFT is defined using the indices in $\{0\} \cup L$. 0 is the index of the spare gate and L has the indices of the rest of the switches in the system.

Finally, we verify the probability of failure of this top event as:

Theorem 7.3.

$$\begin{aligned}
& \vdash \forall p \ X \ Y \ Y_{s_a} \ Y_{s_d} \ t \ L. \\
& \text{DISJOINT } \{0\} \ L \wedge \text{FINITE } L \wedge L \neq \{\} \wedge \\
& \text{indep_sets } p \ (\lambda i. \ \{\text{event_set } [(\text{DFT_event } p \ (\text{WSP } Y \ Y_{s_a} \ Y_{s_d}) \ t, \ 0]) \\
& \quad (\text{rv_to_devent } p \ X \ t) \ i\}) \ (\{0\} \cup L) \wedge \\
& (\forall i. i \in L \Rightarrow \text{rv_gt0_ninfinitiy } [X \ i]) \wedge \\
& (\text{prob } p \ (\text{DFT_event } p \ Q_{\text{dSEN_Terminal}} \ t) =
\end{aligned}$$

1-

(1- prob p (DFT_event p (WSP Y Ys_a Ys_d) t)) *
 Normal ($\prod_{i \in L} (\text{real}(1-F_{x_i}(t)))$)

where `DISJOINT {0} L` ensures that the indices of the elements are unique. While `FINITE L ^ L ≠ {}` ascertain that set L , which has the indices, is finite and not empty. Finally, the independence of the events is added using `indep_sets`. Theorem 7.3 can be further rewritten using Theorem 4.12. However, the required conditions of the latter should be satisfied, such as the continuity of the distributions. Since we need a group of indexed sets in `indep_sets`, we define a function `event_set` that accepts a list of pairs each of which is composed of a DFT event with its index. This function also accepts the remaining blocks of the DFT that have their indices embedded in a set (that can be generic of any size).

In `SEN+`, an additional path is added to increase the redundancy in the system. Therefore, for the connection between a given source and a destination to be broken, it is required that these two paths must be disconnected. The DFT of the `SEN+` is shown in Figure 7.5, where two spares are added to replace the main switches Y and Z after failure. Switch Y is the input switch connected to the source and switch Z is connected to the destination. This DFT is composed of three levels of OR of AND of OR gates. Therefore, in order to verify the probability of the top event, we need first to verify that the `DFT_event` of the n -ary AND gate is equal to the intersection of the input events. We formally verify this in HOL as:

Theorem 7.4.

$\vdash \forall p X t s. \text{FINITE } s \wedge s \neq \{\} \wedge 0 \leq t \Rightarrow$
 $(\text{DFT_event } p$
 $(n_AND (\text{MAP } X (\text{SET_TO_LIST } s))) t = \bigcap_{i \in s} \{\text{rv_to_devent } p X t i\})$

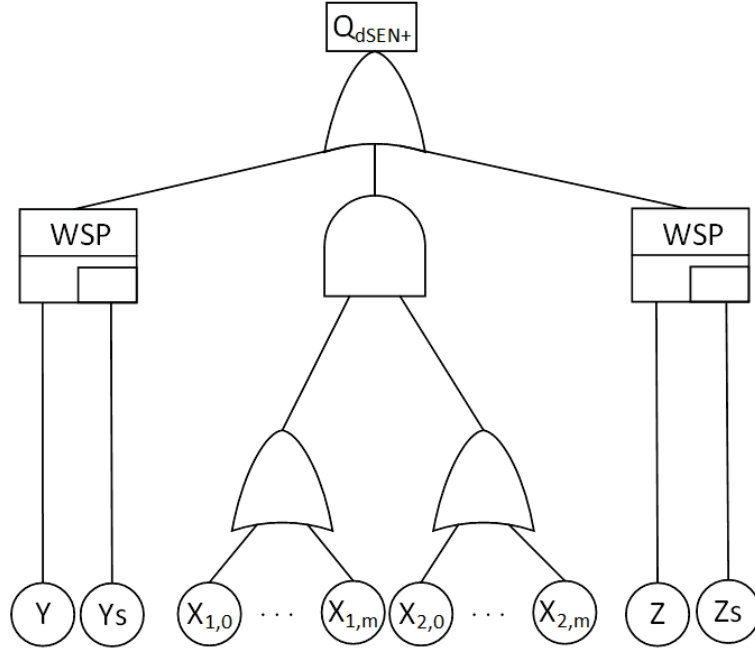


Figure 7.5: DFT of SEN+ Terminal Connection

Then, we verify the probability of failure of the top event of the AND gate as:

Theorem 7.5.

$$\begin{aligned}
 & \vdash \forall p \ X \ t \ s. \ \text{FINITE } s \wedge s \neq \{\} \wedge 0 \leq t \wedge \\
 & \text{indep_sets } p \ (\lambda i. \ \{\text{rv_to_devent } p \ X \ t \ i\}) \ s \\
 & (\forall i. \ i \in s \Rightarrow \text{rv_gt0_ninfinity } [X \ i]) \Rightarrow \\
 & (\text{prob } p \\
 & \quad (\text{DFT_event } p \\
 & \quad \quad (\text{n_AND } (\text{MAP } X \ (\text{SET_TO_LIST } s)))) \ t) = \\
 & \text{Normal } (\prod_{i \in s} (\text{real } (F_{X_i}(t))))
 \end{aligned}$$

The first three conditions are needed to be able to use Theorem 7.4, while `indep_sets` ensures the independence of the events.

We use Theorems 7.2 and 7.5 to verify the probability of OR of AND of OR, which is required for the probability of the top event. We express the top event of the

DFT of Figure 7.5, Q_{dSEN+} as:

$$\begin{aligned}
Q_{dSEN+_{Terminal}} = & \text{n_OR} \left(\text{MAP} \left(\lambda i. \text{if } i = 0 \text{ then WSP } Y \ Y_{s_a} \ Y_{s_d} \right. \right. \\
& \text{else if } i = 1 \text{ then} \\
& \quad \left(\left(\text{n_OR} \left(\text{MAP } X \left(\text{SET_TO_LIST } L1 \right) \right) \right) \cdot \right. \\
& \quad \left. \left(\text{n_OR} \left(\text{MAP } X \left(\text{SET_TO_LIST } L2 \right) \right) \right) \right) \\
& \left. \text{else WSP } Z \ Z_{s_a} \ Z_{s_d} \right) \left(\text{SET_TO_LIST } \{0; 1; 2\} \right)
\end{aligned} \tag{7.2}$$

where $\{0; 1; 2\}$ indicates that the OR gate has three inputs with indices 0 for the first spare, 1 for the AND of ORs, and 2 for the second spare. $L1$ and $L2$ has the indices of the switches in the two redundant paths (for the two lower ORs).

The DFT top event can be expressed using union and intersection of events, which can be quite useful in reusing the existing theorems of probability of union of intersections and intersection of unions. We verify this relationship as:

Theorem 7.6.

$$\begin{aligned}
& \vdash \forall p \ Y \ Y_{s_a} \ Y_{s_d} \ Z \ Z_{s_a} \ Z_{s_d} \ X \ L1 \ L2 \ t. \\
& \text{FINITE } L1 \ \wedge \ \text{FINITE } L2 \ \wedge \\
& \text{disjoint_family_on} \left(\text{ind_set} \left[\{0\}; L1; L2; \{3\} \right] \right) \{0; 1; 2; 3\} \Rightarrow \\
& \left(\text{DFT_event } p \left(Q_{dSEN+_{Terminal}} \right) \ t = \right. \\
& \text{BIGUNION} \\
& \quad \left\{ \text{BIGINTER} \right. \\
& \quad \quad \left\{ \text{BIGUNION} \right. \\
& \quad \quad \quad \left\{ \text{event_set} \right. \\
& \quad \quad \quad \quad \left[\left(\text{DFT_event } p \left(\text{WSP } Y \ Y_{s_a} \ Y_{s_d} \right) \ t, 0 \right); \right.
\end{aligned}$$

$$\begin{aligned}
& (\text{DFT_event } p \text{ (WSP } Z \text{ } Z_{S_a} \text{ } Z_{S_d}) \text{ } t, 3)] \\
& (\text{rv_to_devent } p \text{ } X \text{ } t) \text{ } i \text{ } | \\
& i \in \text{ind_set } [\{0\}; L1; L2; \{3\}] \text{ } a \text{ } | \\
& a \in \text{ind_set } [\{0\}; \{1; 2\}; \{3\}] \text{ } j \text{ } | \\
& j \in \{0; 1; 2\})
\end{aligned}$$

Finally, we verify the probability of failure of $Q_{\text{dSEN+}}$:

Theorem 7.7.

$$\begin{aligned}
& \vdash \forall p \text{ } X \text{ } Y \text{ } Y_{S_a} \text{ } Y_{S_d} \text{ } Z \text{ } Z_{S_a} \text{ } Z_{S_d} \text{ } t \text{ } L1 \text{ } L2. \quad 0 \leq t \wedge \\
& \text{SEN_set_req } p \text{ } L1 \text{ } L2 \text{ (ind_set } [\{0\}; L1; L2; \{3\}]) \\
& \text{(ind_set } [\{0\}; \{1; 2\}; \{3\}]) \text{ } \{0; 1; 2\} \\
& \text{(event_set } [(\text{DFT_event } p \text{ (WSP } Y \text{ } Y_{S_a} \text{ } Y_{S_d}) \text{ } t, 0); \\
& \quad (\text{DFT_event } p \text{ (WSP } Z \text{ } Z_{S_a} \text{ } Z_{S_d}) \text{ } t, 3)] \\
& \quad (\text{rv_to_devent } p \text{ } X \text{ } t)) \wedge \\
& (\forall i. \quad i \in (L1 \cup L2) \Rightarrow \text{rv_gt0_ninfinity } [X \text{ } i]) \Rightarrow \\
& (\text{prob } p \text{ (DFT_event } p \text{ } Q_{\text{dSEN+_Terminal}} \text{ } t) = \\
& 1 - \\
& (1 - \\
& \quad \text{prob } p \text{ (DFT_event } p \text{ (WSP } Y \text{ } Y_{S_a} \text{ } Y_{S_d}) \text{ } t)) * \\
& \quad (\text{Normal} \\
& \quad (1 - \\
& \quad \quad (1 - \prod_{i \in L1} (\text{real } (1 - F_{X_i}(t)))) * \\
& \quad \quad (1 - \prod_{i \in L2} (\text{real } (1 - F_{X_i}(t)))) * \\
& \quad (1 - \text{prob } p \text{ (DFT_event } p \text{ (WSP } Z \text{ } Z_{S_a} \text{ } Z_{S_d}) \text{ } t))))))
\end{aligned}$$

where `SEN_set_req` ensures the required conditions of the input sets including that the sets are finite and nonempty. It also ensures the independence of the input events

over the probability space. We also define `ind_set` that accepts a list of sets and returns a group of indexed sets. This is required to be able to create the hierarchy of the DFT using sets.

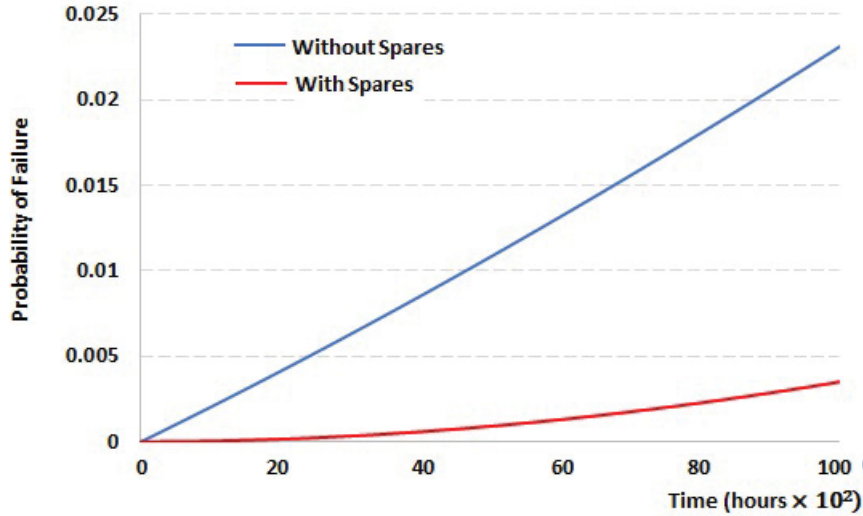


Figure 7.6: Probability of Failure of the Terminal Connection of a 128×128 SEN+ with and without Spares

In order to use the above generic probability of failure expressions on a concrete instance of SEN+, we evaluate the probability of failure of the terminal connection of a 128×128 SEN+, where each OR gate of the first level of Figure 7.5 has 6 inputs. We assume that the failure rate of each switching element is 1×10^{-5} . We evaluate the probability of failure for the SEN+ system without and with spare parts with a dormancy factor of 0.1, as shown in Figure 7.6. This result shows that considering the spares in the analysis leads to having more reliable and realistic system than the traditional FTs.

7.2.2 DRBD Analysis of SEN and SEN+

For SENs (single-path MIN), the terminal reliability is modeled as a series RBD. For illustration purposes, we use a spare part to replace the first input switch, and thus increase the reliability. The DRBD of the modified SEN is shown in Figure 7.7, where Y is the main switch that will be replaced by Y_s after failure and the series structure has $m + 1$ elements.

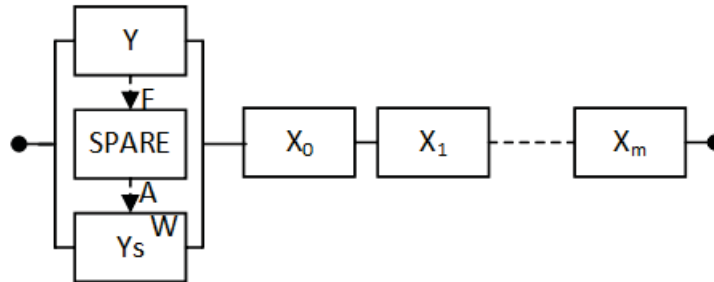


Figure 7.7: DRBD of SEN

Using the proposed algebra in Chapter 6, we express the structure function of the SEN DRBD as:

$$Q_{\text{SEN.Termina1}} = \text{nR_AND} (\lambda i. \text{ if } i = 0 \text{ then R_WSP } Y \ Y_{S_a} \ Y_{S_d} \text{ else } X \ i) \ \{0\} \cup L \quad (7.3)$$

where X is a group of indexed time-to-failure functions that represent the blocks of the series structure and L is a set with their indices. L can be instantiated with any group of numbers, which makes this function generic to represent the reliability model of any SEN with any size.

Then, we verify that the DRBD_event of Q_{SEN} can be represented using the series parallel structures as:

Theorem 7.8.

$$\vdash \forall p X Y Ys_a Ys_d t L.$$

$$\text{DISJOINT } \{0\} L \wedge \text{FINITE } L \wedge L \neq \{\} \Rightarrow$$

$$(\text{DRBD_event } p \ Q_{\text{SEN_Terminal}} t =$$

$$\text{DRBD_series}$$

$$(\lambda i. \ \text{event_set}$$

$$[(\text{DRBD_event } p \ (\text{R_WSP } Y \ Ys_a \ Ys_d) \ t, 0)]$$

$$(\text{rv_to_event } p \ X \ t) \ i) \ (\{0\} \cup L) \)$$

where DISJOINT ensures that all sets are disjoint. We use `event_set` and `ind_set` to create the events, similar to the DFTs. Since we are dealing with a series structure, we only need to specify the hierarchy of the architecture in one direction using $\{0\} \cup L$. We verify Theorem 7.8 using Theorem 6.9 and some set-related theorems.

Based on Theorem 7.8, we verify a generic expression for the reliability of the SEN system:

Theorem 7.9.

$$\vdash \forall p X Y Ys_a Ys_d t L.$$

$$\text{DISJOINT } \{0\} L \wedge \text{FINITE } L \wedge L \neq \{\} \wedge$$

$$\text{indep_sets } p \ (\lambda i. \ \{\text{event_set } [(\text{DRBD_event } p \ (\text{R_WSP } Y \ Ys_a \ Ys_d) \ t, 0)]$$

$$(\text{rv_to_event } p \ X \ t) \ i\}) \ (\{0\} \cup L) \Rightarrow$$

$$(\text{prob } p \ (\text{DRBD_event } p \ Q_{\text{SEN_Terminal}} t) =$$

$$\text{Rel } p \ (\text{R_WSP } Y \ Ys_a \ Ys_d) \ t * \text{Normal } (\prod_{l \in L} (\text{real } (\text{Rel } p \ (X \ l) \ t))))$$

In a similar manner, the SEN+ is modeled as a series-parallel-series structure [25]. To further enhance the reliability, we use spare constructs as shown in Figure 7.8, where Y and Z are the main single switches that are connected to the

source and destination with their spares Y_s and Z_s , respectively. The parallel structure in the middle represents the reliability model of the two alternative paths between the source and the destination. Therefore, this DRBD consists of a series of two spare constructs and one parallel structure that consists of two series structures.

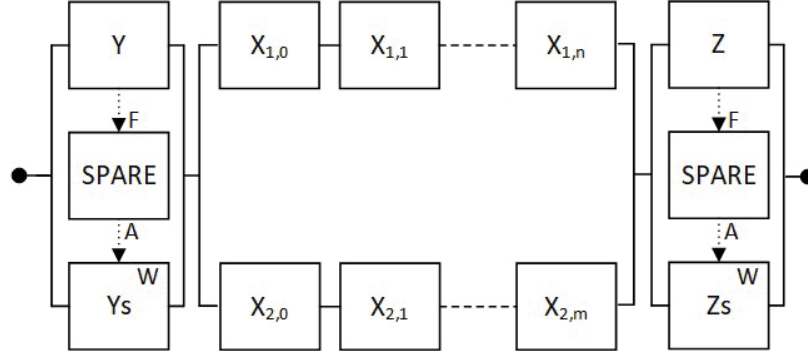


Figure 7.8: Terminal Reliability DRBD of SEN+

Using our DRBD operators, we formally express the structure function of this DRBD as:

$$\begin{aligned}
 Q_{\text{SEN+_Terminal}} = \text{nR_AND} (\lambda i. \quad & \text{if } i = 0 \text{ then R_WSP } Y \ Ys_a \ Ys_d \\
 & \text{else if } i = 1 \text{ then } ((\text{nR_AND } X \ L1) + (\text{nR_AND } X \ L2)) \\
 & \text{else R_WSP } Z \ Zs_a \ Zs_d) \ {0; 1; 2}
 \end{aligned}
 \tag{7.4}$$

Thus, the outer series structure is expressed using the `nR_AND` operator over the set $\{0; 1; 2\}$ as this structure has three different structures; i.e., two spare constructs and one parallel structure. In order to re-utilize the verified expressions of reliability, it is required to express this DRBD using the series and parallel structures. Therefore, we verify that the DRBD event of the $Q_{\text{SEN+}}$ is equal to a nested series-parallel-series structure as:

Theorem 7.10.

$$\begin{aligned}
& \vdash \forall p \ X \ Y \ Y_{s_a} \ Y_{s_d} \ Z \ Z_{s_a} \ Z_{s_d} \ t \ L1 \ L2. \\
& \text{disjoint_family_on} \ (\text{ind_set} \ [\{0; 3\}; L1; L2]) \ \{0;1;2\} \wedge \\
& \text{FINITE } L1 \wedge \text{FINITE } L2 \wedge L1 \neq \{\} \wedge L2 \neq \{\} \Rightarrow \\
& (\text{DRBD_event } p \ \mathbb{Q}_{\text{SEN+Terminal}} \ t = \\
& \quad \text{DRBD_series} \ (\lambda j. \\
& \quad \quad \text{DRBD_parallel} \ (\lambda a. \\
& \quad \quad \quad \text{DRBD_series} \ (\lambda i. \\
& \quad \quad \quad \quad \text{event_set} \\
& \quad \quad \quad \quad \quad [(\text{DRBD_event } p \ (\text{R_WSP } Y \ Y_{s_a} \ Y_{s_d}) \ t, 0); \\
& \quad \quad \quad \quad \quad (\text{DRBD_event } p \ (\text{R_WSP } Z \ Z_{s_a} \ Z_{s_d}) \ t, 3)]) \\
& \quad \quad \quad \quad (\text{rv_to_event } p \ X \ t) \ i) \\
& \quad \quad \quad \text{ind_set} \ [\{0\}; L1; L2; \{3\}] \ a)) \\
& \quad \text{ind_set} \ [\{0\}; \{1; 2\}; \{3\}] \ j)) \ \{0; 1; 2\})
\end{aligned}$$

where $\text{disjoint_family_on} \ (\text{ind_set} \ [\{0; 3\}; L1; L2]) \ \{0;1;2\}$ ensures that the sets $\{0;3\}$, $L1$ and $L2$ are disjoint, i.e., each switch has a unique index. Since we are dealing with a series-parallel-series structure, we need three sets to identify the hierarchy of this nested structure. Set $\{0;1;2\}$ in Theorem 7.10 indicates that the outer series structure has three elements, i.e., three parallel structures. $\text{ind_set} \ [\{0\}; \{1;2\}; \{3\}]$ indicates that the first parallel structure has only one series structure with index 0, the second parallel structure has two series structures with indices 1 and 2, and the third parallel structure has only one series structure with index 3. Finally, $\text{ind_set} \ [\{0\}; L1; L2; \{3\}]$ implies that the first series structure has only one element with index 0, the second and third series structures have an arbitrary number of blocks indexed by $L1$ and $L2$. The last series structure has one element

with index 3. We verify Theorem 7.10 using Theorem 6.9 and the equivalence of the event of the OR with the union of events besides some set-related theorems.

Based on Theorem 7.10, we verify a generic expression for the reliability of the SEN+ system:

Theorem 7.11.

$\vdash \forall p X Y Y_{s_a} Y_{s_d} Z Z_{s_a} Z_{s_d} t L1 L2.$

$$\begin{aligned}
& \text{SEN_set_req } p L1 L2 (\text{ind_set } [\{0\}; L1; L2; \{3\}]) \\
& (\text{ind_set } [\{0\}; \{1; 2\}; \{3\}]) \{0; 1; 2\} \\
& (\text{event_set } [(\text{DRBD_event } p (\text{R_WSP } Y Y_{s_a} Y_{s_d}) t, 0); \\
& \quad (\text{DRBD_event } p (\text{R_WSP } Z Z_{s_a} Z_{s_d}) t, 3)]) \\
& (\text{rv_to_event } p X t)) \Rightarrow \\
& (\text{prob } p (\text{DRBD_event } p Q_{\text{SEN_Terminal}} t) = \\
& \text{Rel } p (\text{R_WSP } Y Y_{s_a} Y_{s_d}) t * \text{Rel } p (\text{R_WSP } Z Z_{s_a} Z_{s_d}) t * \\
& (1 - \\
& \quad (1 - \text{Normal } (\prod_{l \in L1} (\text{real } (\text{Rel } p (X l) t)))) * \\
& \quad (1 - \text{Normal } (\prod_{l \in L2} (\text{real } (\text{Rel } p (X l) t)))))
\end{aligned}$$

where `SEN_set_req` is the same function that we use with DFTs. We first rewrite the goal using Theorem 7.10, then we use the reliability of the series-parallel-series to verify the final expression. The reliability of the spare constructs can be further rewritten using Theorem 6.5 given that the required conditions are ensured, such as the continuity of the CDFs. It can be noticed that the DRBD and the DFT models possess the same hierarchy represented by the sets of indices, which makes it easy to be used when going from one model to the other.

Similar to the DFT analysis, we evaluate the terminal reliability of a 128×128 SEN+, where each inner series structure of Figure 7.8 has 6 blocks. We assume that

the failure rate of each switching element is 1×10^{-5} . We evaluate the reliability for the SEN+ system without and with spare parts with a dormancy factor of 0.1, as shown in Figure 7.9.

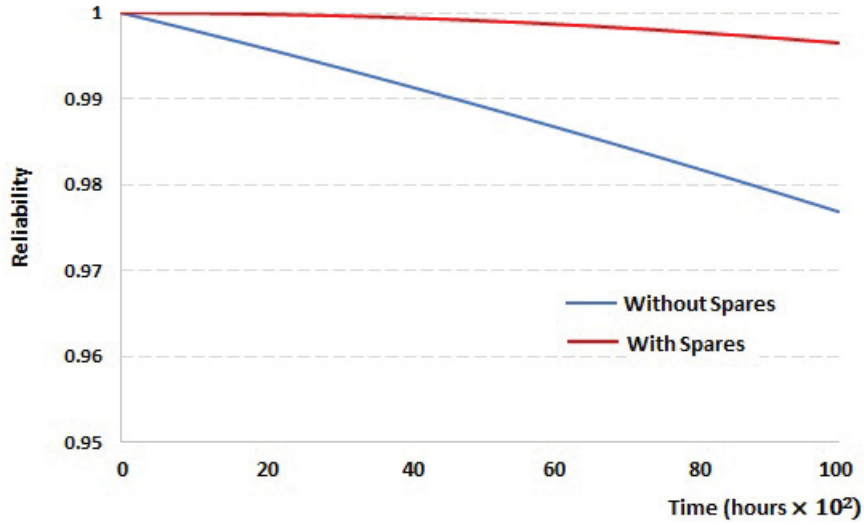


Figure 7.9: Terminal Reliability of 128×128 SEN+ with and without Spares

7.3 Broadcast Reliability Analysis of Shuffle-exchange Networks

The broadcast reliability represents the probability of having a working connection between one source and all destinations. This is required when one of the processors in the system needs to transmit information to all destinations in the network. We present in this section, the broadcast reliability of the SEN and SEN+ using both DFT and DRBD models.

7.3.1 DFT Analysis of SEN and SEN+

Since in SENs there exists a single path between each source and destination, it is required to have a successful transmission through all these paths for a proper broadcast. Therefore, the DFT can be modeled using an OR gate. We further lower the probability of failure by adding an additional spare gate, as shown in Figure 7.4. However, the number of DFT inputs, which represent the switches, varies between the terminal and broadcast reliability models. For example, consider an 8×8 SEN. The number of inputs for the terminal DFT is 3, i.e., $\log_2 8$, while the broadcast DFT requires seven inputs, i.e., $\sum_{i=1}^{\log_2 8} (\frac{8}{2^i})$ [25]. Therefore, we can also use Theorem 7.3 for the broadcast, since this theorem is verified for any number of system blocks with their indices in the set s . This highlights the importance of having generic verified expressions for any number of system blocks, which enables the re-utilization of the theorems in different contexts.

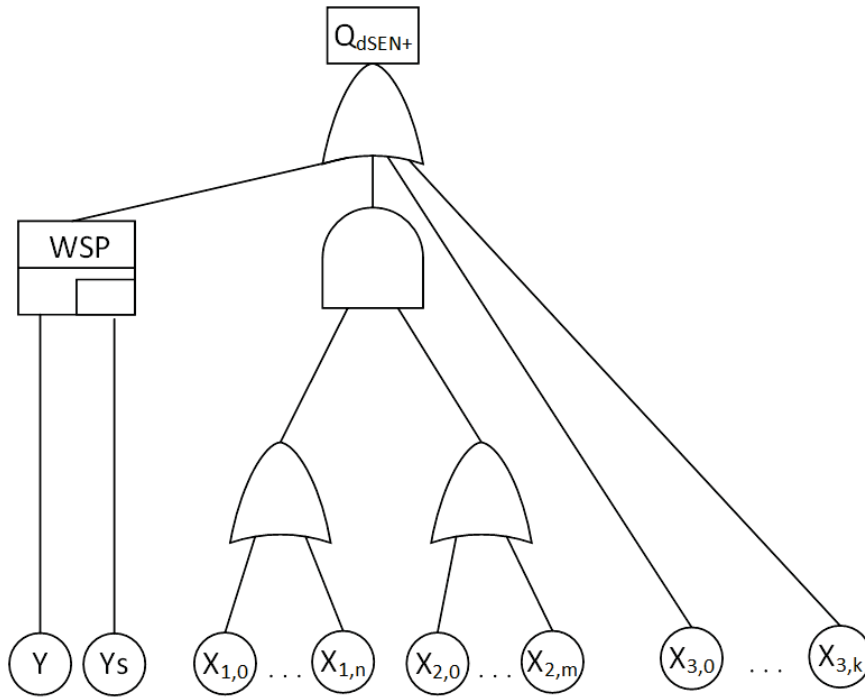


Figure 7.10: DFT of Broadcast SEN+

The DFT model of the broadcast SEN+ is shown in Figure 7.10. Its top event is modeled using an OR gate that is connected to a spare gate for the input switch, AND of OR to model the two alternative paths and finally, the rest of the destination switches in order to have a proper broadcast transmission.

We formally express the structure function of the top event as:

$$\begin{aligned}
Q_{dSEN+_{Broadcast}} = & \text{n_OR} \left(\text{MAP} \left(\lambda i. \quad \text{if } i = 0 \text{ then WSP } Y \ Y_{s_a} \ Y_{s_d} \right. \right. \\
& \quad \text{else if } i = 1 \text{ then} \\
& \quad \quad \left(\left(\text{n_OR} \left(\text{MAP } X \left(\text{SET_TO_LIST } L1 \right) \right) \right) \cdot \right. \\
& \quad \quad \quad \left. \left(\text{n_OR} \left(\text{MAP } X \left(\text{SET_TO_LIST } L2 \right) \right) \right) \right) \\
& \quad \quad \text{else } \left(\text{n_OR} \left(\text{MAP } X \left(\text{SET_TO_LIST } L3 \right) \right) \right) \\
& \left. \left(\text{SET_TO_LIST } \{0; 1; 2\} \right) \right)
\end{aligned} \tag{7.5}$$

The hierarchy of the DFT is divided using the sets of indices. We need to recall that $\text{MAP } X \left(\text{SET_TO_LIST } L1 \right)$, $\text{MAP } X \left(\text{SET_TO_LIST } L2 \right)$ and $\text{MAP } X \left(\text{SET_TO_LIST } L3 \right)$ are used to create the lists of the group of random variables for the n -ary gates. $L1$ and $L2$ has the indices of the switches in the two alternative paths, i.e., the inputs of the two lower OR gates in the DFT of Figure 7.10, while $L3$ has the indices of the remaining inputs of the top OR gate. The set $\{0; 1; 2\}$ indicates that the top OR gate has three inputs, which is similar to the terminal DFT model.

We use this structure function to verify the probability of failure of the top event:

Theorem 7.12.

$\vdash \forall p \ X \ Y \ Y_{s_a} \ Y_{s_d} \ t \ L1 \ L2 \ L3 \ s.$

$$\begin{aligned}
& \text{SEN_broad_set_req } p \ L1 \ L2 \ L3 \left(\text{ind_set } [\{0\}; L1; L2; L3] \right) \\
& \left(\text{ind_set } [\{0\}; \{1; 2\}; \{3\}] \right) \{0; 1; 2\}
\end{aligned}$$

$$\begin{aligned}
& (\text{event_set } [(DFT_event \text{ p } (WSP \text{ Y } Y_{s_a} \text{ Y}_{s_d}) \text{ t}, 0); \\
& \quad (\text{rv_to_devent } \text{ p } \text{ X } \text{ t})) \wedge 0 \leq \text{ t } \wedge \\
& (\forall \text{ i. } \text{ i } \in (L1 \cup L2 \cup L3) \Rightarrow \text{rv_gt0_ninfinity } [\text{X } \text{ i}]) \Rightarrow \\
& (\text{prob } \text{ p } (DFT_event \text{ p } Q_{dSEN+_Broadcast} \text{ t}) = \\
& 1 - \\
& (1 - \\
& \quad \text{prob } \text{ p } (DFT_event \text{ p } (WSP \text{ Y } Y_{s_a} \text{ Y}_{s_d}) \text{ t})) * \\
& \quad (\text{Normal} \\
& \quad (1 - \\
& \quad \quad (1 - \prod_{i \in L1} (\text{real } (1 - F_{X_i}(\text{t})))) * \\
& \quad \quad (1 - \prod_{i \in L2} (\text{real } (1 - F_{X_i}(\text{t})))) * \\
& \quad \quad \text{Normal } (\prod_{i \in L3} (\text{real } (1 - F_{X_i}(\text{t}))))))
\end{aligned}$$

where `SEN_broad_set_req` ascertains the conditions required for the sets such as finiteness. It also ensures the independence of the events.

Figure 7.11 shows the evaluation results of the probability of failure of the DFT of Figure 7.10 for a 128×128 SEN+. This SEN+ has 63 inputs for each first level OR gate and the top level OR gate has 66 inputs. As with the terminal SEN+, we assume that the failure rate of each switching element is 1×10^{-5} with a dormancy factor of 0.1.

7.3.2 DRBD Analysis of SEN and SEN+

Similar to the DFT SEN broadcast model, we can use the model in Figure 7.7. However, as mentioned previously, the number of the blocks is different. Therefore, we can also use Theorem 7.9 for the broadcast reliability, since this theorem is verified for any number of system blocks using set s .

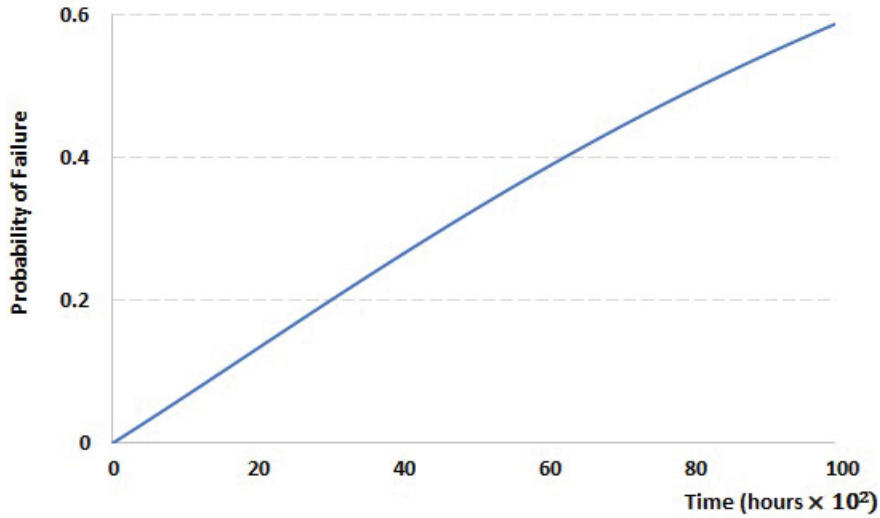


Figure 7.11: Probability of Failure of the Broadcast of a 128×128 SEN+

The DRBD of the SEN+ is depicted in Figure 7.12. The first block (with the spare) represents the input switch that is connected directly to the source. The failure of this switch will interrupt the broadcast transmission. Therefore, we add a spare part to replace it after failure.

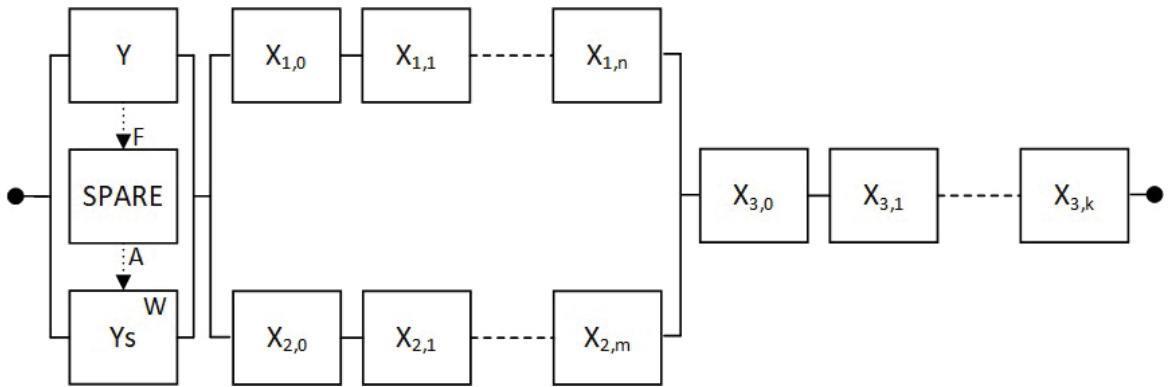


Figure 7.12: Broadcast DRBD Model of SEN+

The series structure on the right side of the figure models the switches of all destinations, as they are all receiving the transmission. Finally, the parallel-series structure in the middle, represents the two alternative paths that are available for each broadcast transmission. For example, for the SEN+ shown in Figure 7.3, the

number of switches connected to the destinations are four, while each one of the alternative paths has three switches.

In order to formally verify the reliability of the broadcast of the SEN+, we first express it using our operators as:

$$\begin{aligned}
Q_{\text{SEN+_Broadcast}} = \text{nR_AND } (\lambda i. \quad & \text{if } i = 0 \text{ then R_WSP } Y \text{ } Y_{s_a} \text{ } Y_{s_d} \\
& \text{else if } i = 1 \text{ then } ((\text{nR_AND } X \text{ } L1) + \\
& \hspace{15em} (\text{nR_AND } X \text{ } L2)) \\
& \text{else } (\text{nR_AND } X \text{ } L3)) \text{ } (\{0; 1 \ 2\})
\end{aligned} \tag{7.6}$$

where $L1$ and $L2$ are the sets that have the indices of the inner series structures of the parallel-series structure in the middle. The set $\{0; 1; 2\}$ indicates that the outer series structure consists of three main components. The first spare construct has index 0, while the parallel-series structure has index 1. Finally, the series structure on the left side of Figure 7.12 has index 2, and $L3$ has the indices of the blocks in this series structure. We verify the reliability of this DRBD as:

Theorem 7.13.

$\vdash \forall p \ X \ Y \ Y_{s_a} \ Y_{s_d} \ t \ L1 \ L2 \ L3.$

$\text{SEN_broad_set_req } p \ L1 \ L2 \ (\text{ind_set } [\{0\}; L1; L2; L3])$

$(\text{ind_set } [\{0\}; \{1; 2\}; \{3\}]) \ \{0; 1; 2\}$

$(\text{event_set } [(\text{DRBD_event } p \ (\text{R_WSP } Y \ Y_{s_a} \ Y_{s_d}) \ t), 0];$

$(\text{rv_to_event } p \ X \ t)) \Rightarrow$

$(\text{prob } p \ (\text{DRBD_event } p \ Q_{\text{SEN+_Broadcast}} \ t)) =$

$\text{Rel } p \ (\text{R_WSP } Y \ Y_{s_a} \ Y_{s_d}) \ t \ * \ \text{Normal } (\prod_{i \in L3} (\text{real } (\text{Rel } p \ (X \ 1) \ t))) \ *$

$(1 - (1 - \text{Normal } (\prod_{l \in L1} (\text{real } (\text{Rel } p \ (X \ 1) \ t)))) \ *$

$(1 - \text{Normal } (\prod_{l \in L2} (\text{real } (\text{Rel } p \ (X \ 1) \ t))))))$

We evaluate the broadcast reliability, in Figure 7.13, of a 128×128 SEN+, where each inner series structure of Figure 7.12 has 63 blocks and the series structure on the right hand side of the figure has 64 blocks. We use the same failure rates of 1×10^{-5} for each switching element with a dormancy factor of 0.1.

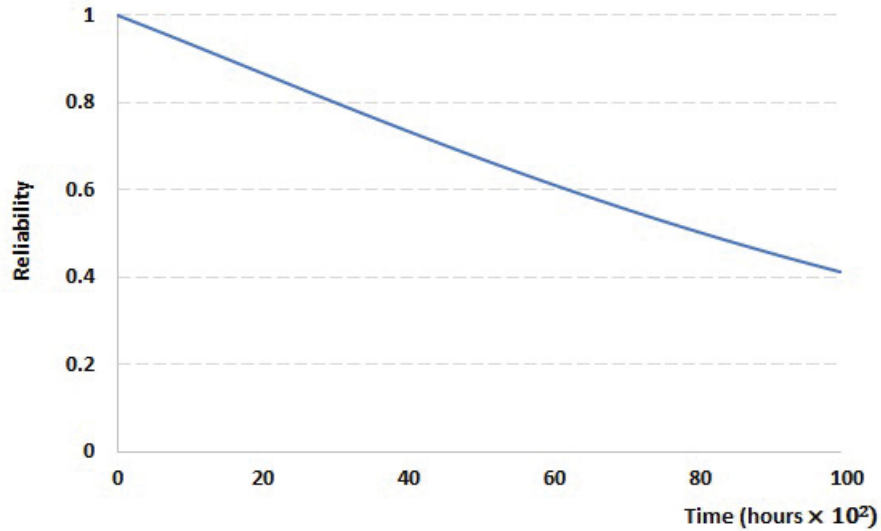


Figure 7.13: Broadcast Reliability of a 128×128 SEN+

7.4 Network Reliability Analysis of Shuffle-exchange Networks

According to [25], the network reliability of SENs can be defined as the reliability of all connections between sources (inputs) and destinations (outputs). In other words, we are looking at the reliability of the overall network. This is usually modeled using RBDs. In this section, we use both DFT and DRBD models in different scenarios to model the reliability of the network.

7.4.1 DFT Analysis of SEN and SEN+

In the SEN, it is required that all switching elements must work properly in order to maintain a successful behavior of the network. Thus, the system fails with the failure of any of the switching elements. The behavior can be further enhanced by using spares. The DFT of the SEN network can be modeled as in Figure 7.4. However, to further enhance the system reliability, the reliability engineer may suggest to use more spares to replace the switching elements. Therefore, we present a generic model, where the number of switching elements that have spares is generic, as shown in Figure 7.14. This model can be also used with both the terminal and broadcast models, when more spares are required.

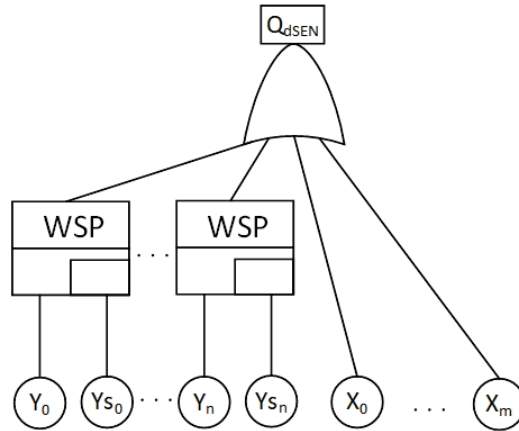


Figure 7.14: DFT of SEN Network with Multiple Spares

We express the DFT top event of Figure 7.14 using the DFT operators as:

$$\begin{aligned}
 Q_{dsEN_Network} = n_OR \ (MAP \ (\lambda i. \ \text{if } i \in L1 \ \text{then } WSP \ (Y \ i) \ (Y_{s_a} \ i) \ (Y_{s_d} \ i) \\
 \text{else } X \ i) \ (SET_TO_LIST \ (L1 \cup \ L2)))
 \end{aligned}
 \tag{7.7}$$

We verify the probability of failure of the top event in a similar way to Theorem 7.3, as:

Theorem 7.14.

$\vdash \forall p X Y Y_{s_a} Y_{s_d} t L1 L2.$

$DISJOINT L1 L2 \wedge FINITE L1 \wedge L1 \neq \{\} \wedge$

$FINITE L2 \wedge L2 \neq \{\} \wedge$

$(\forall i. i \in L2 \Rightarrow rv_gt0_ninfinity [X i]) \wedge$

$indep_sets p$

$(\lambda i.$

$\{rv_to_devent p$

$(\lambda i. i \in L1 \text{ then WSP } (Y i) (Y_{s_a} i) (Y_{s_d} i) \text{ else } X i)$

$t i\})(L1 \cup L2) \Rightarrow$

$(\text{prob } p (\text{DFT_event } p Q_{\text{dSEN_Network}} t) =$

$1 -$

Normal

$(\prod_{i \in L1}$

$(\text{real}(1 - \text{prob } p (\text{DFT_event } p (\text{WSP } (Y i) (Y_{s_a} i) (Y_{s_d} i)) t)))) *$

$\text{Normal } (\prod_{i \in L2} (\text{real}(1 - F_{X_i}(t))))))$

where Y , Y_{s_a} and Y_{s_d} are groups of indexed random variables that represent the main and spare switches. Theorem 7.14 provides a generic scenario for the SEN, where $L1$ and $L2$ can be instantiated with any number of distinct indices that represent the system switches, with and without spares.

The DFT model of the SEN+ network is shown in Figure 7.15. It consists of a spare gate for one of the switches in the input stage. The rest of the input switches ($X_{1,0} - X_{1,r}$) are connected directly to the n -OR gate of the top event. Therefore, the

failure of any of these switches leads to the failure of the network. The series of ANDs and ANDs of ORs are used to model the two available paths. Finally, all destination switches ($X_{4,0} - X_{4,k}$) are required to function and thus they are all connected to the output OR gate. This DFT is composed of three levels; OR of ANDs of ORs, and thus we can use the theorems of union of intersections of unions to verify its probability of failure if the sets of indices are handled properly.

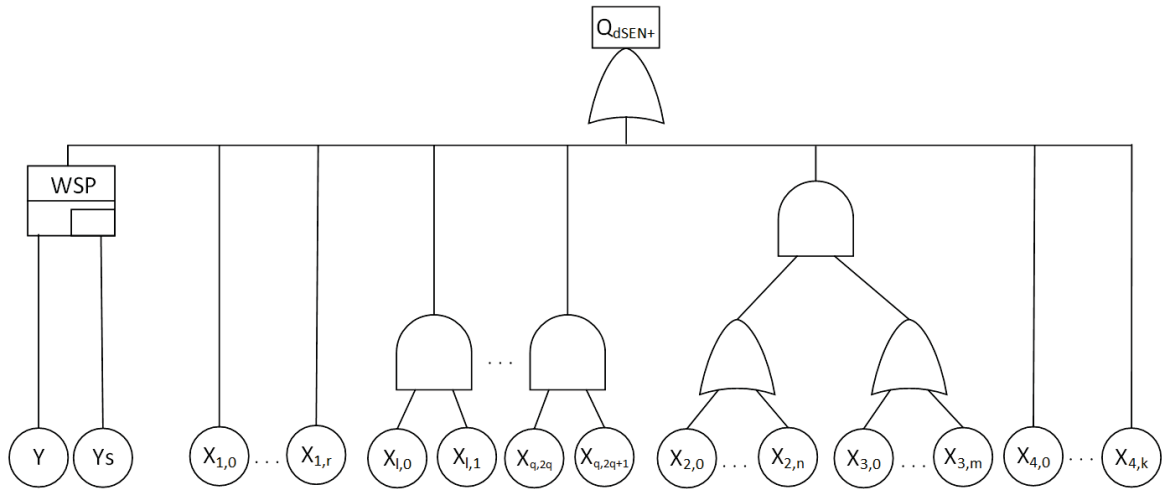


Figure 7.15: DFT of SEN+ Network

We first express the top event using the DFT operators as:

$$\begin{aligned}
Q_{dSEN_Network} = & \\
& n_OR \\
& (MAP \\
& \quad (\lambda i. \text{ if } i = 0 \text{ then WSP } Y \ Y_{S_a} \ Y_{S_d} \\
& \quad \quad \text{else if } i = 1 \text{ then } n_OR \ (MAP \ X \ (SET_TO_LIST \ L1)) \\
& \quad \quad \text{else if } i = 3 \text{ then } (n_OR \ (MAP \ X \ (SET_TO_LIST \ L2))) \cdot \\
& \quad \quad \quad (n_OR \ (MAP \ X \ (SET_TO_LIST \ L3))) \\
& \quad \quad \text{else if } i = 4 \text{ then } n_OR \ (MAP \ X \ (SET_TO_LIST \ L4)) \\
& \quad \quad \text{else } (X \ (2 * i)) \cdot (X \ (2 * i + 1))) \\
& \quad (SET_TO_LIST \ (\{0; 1; 3; 4\} \cup L)))
\end{aligned} \tag{7.8}$$

where the spare gate is assigned index 0. The second group of switches has index 1, while the indices of these switches, $X_{1,0} - X_{1,r}$, are in set L1. They are represented as $n_OR \ (MAP \ X \ (SET_TO_LIST \ L1))$. The output of the AND of ORs is assigned index 3 and is modeled as $(n_OR \ (MAP \ X \ (SET_TO_LIST \ L2))) \cdot (n_OR \ (MAP \ X \ (SET_TO_LIST \ L3)))$, which is similar to both the terminal and broadcast models. The group of switches, $X_{4,0} - X_{4,k}$, has index 4 and is represented using $n_OR \ (MAP \ X \ (SET_TO_LIST \ L4))$. Thus, we have the indices $\{0; 1; 3; 4\}$ for the outer groups in the DFT. However, the last part of the DFT, which is the series of ANDs in the middle of Figure 7.15, has a generic number of AND gates and cannot be assigned a specific index. Therefore, we use set L to get a unique index for the output of each AND gate. We use this unique number to create the indices of the inputs of each AND gate. For example, for an index j in set L, we create two indices for the inputs of the AND gate

as $(2*j)$ and $(2*j+1)$. This is modeled as $(X (2 * i)) \cdot (X (2 * i + 1))$ and set L is used with the set of indices in the outer level as $(\text{SET_TO_LIST} (\{0; 1; 3; 4\} \cup L))$. It is important to highlight that the indices of the individual inputs should be unique.

We then verify that the DFT_event of $\mathbb{Q}_{\text{dSEN_Network}}$ is equal to the union of intersection of union of events as in the following theorem:

Theorem 7.15.

$\vdash \forall p \ L1 \ L2 \ L3 \ L4 \ L \ X \ Y \ Ys_a \ Ys_d \ t.$

FINITE $L1 \wedge L1 \neq \{\}$ \wedge FINITE $L2 \wedge L2 \neq \{\}$ \wedge FINITE $L3 \wedge$
 $L3 \neq \{\}$ \wedge FINITE $L4 \wedge L4 \neq \{\}$ \wedge FINITE $L \wedge$
DISJOINT $\{0; 1; 3; 4\} \ L \wedge$
 $(\forall i. \ i \in L \Rightarrow \text{DISJOINT} \{2 * i; 2 * i + 1\} \ \{0; 1; 2; 3; 4\}) \wedge$
disjoint_family_on
(ind_set
 $[\{0\}; L1; L2; L3; L4; \{2 * i \mid i \in L\} \cup \{2 * i + 1 \mid i \in L\}])$
 $\{0; 1; 2; 3; 4; 5\} \Rightarrow$
 $(\text{DFT_event } p \ (\mathbb{Q}_{\text{dSEN_Network}}) \ t =$
BIGUNION
{BIGINTER
{BIGUNION
[event_set $[(\text{DFT_event } p \ (\text{WSP } Y \ Ys_a \ Ys_d) \ t, 0)]$
 $(\text{rv_to_devent } p \ X \ t) \ i \mid$
 $i \in \text{if } a \in \{2 * i \mid i \in L\} \cup \{2 * i + 1 \mid i \in L\} \text{ then } \{a\}$
else ind_set $[\{0\}; L1; L2; L3; L4] \ a\} \mid$
 $a \in \text{if } j \in L \text{ then } \{2 * j; 2 * j + 1\}$

```

else ind_set [{0}; {1}; {}; {2; 3}; {4}] j |
j ∈ {0; 1; 3; 4} ∪ L}

```

where the conditions are required to ensure that the sets are finite, nonempty and that at each level of the DFT the indices are unique. It is clear from the theorem how the hierarchy of the DFT is structured using the sets. For example, “if $j \in L$ then $\{2 * j; 2 * j + 1\}$ else ind_set $[\{0\}; \{1\}; \{\}; \{2; 3\}; \{4\}] j$ ” determines the indices of the second level of the DFT (the ORs) based on the value of j in the outer level. The first part “if $j \in L$ then $\{2 * j; 2 * j + 1\}$ ” is for the series of ANDs, while “else ind_set $[\{0\}; \{1\}; \{\}; \{2; 3\}; \{4\}] j$ ” is for the rest of the parts in the second level. Although some of the parts of the DFT have no intermediate OR gates, like the spare, we implicitly assume that there are OR gates with single inputs to maintain the consistency. The indices of the second level indicates the indices of the output of these gates. This can be obvious for the AND of ORs in Figure 7.15, where the OR gates have indices 2 and 3. We use an empty set ($\{\}$) in the indices of the second level due to the fact that there is no index 2 in the outer level, and thus we assigned an empty set in the second level for this index.

We verify the probability of failure of $Q_{\text{dSEN_Network}}$ as:

Theorem 7.16.

$\vdash \forall p L1 L2 L3 L4 L X Y Y_{\text{sa}} Y_{\text{sd}} t.$

SEN_network_set_req p L1 L2 L3 L4 L

($\lambda i.$

if $i \in \{2 * i \mid i \in L\} \cup \{2 * i + 1 \mid i \in L\}$ then $\{i\}$

else ind_set $[\{0\}; L1; L2; L3; L4] i$)

($\lambda j.$

if $j \in L$ then $\{2 * j; 2 * j + 1\}$

```

    else ind_set [{0}; {1}; {}; {2; 3}; {4}] j)
  ({0; 1; 3; 4} ∪ L)
  (event_set [(DFT_event p (WSP Y Ysa Ysd) t,0)]
    (rv_to_devent p X t)) ∧
(∀ i.
  i ∈ L1 ∪ L2 ∪ L3 ∪ L4 ∪
    {2 * i | i ∈ L} ∪ {2 * i + 1 | i ∈ L} ⇒
  rv_gt0_ninfinity [X i]) ⇒
(prob p
  (DFT_event p (QdSEN_Network) t) =
  1 -
  (1 - prob p (DFT_event p (WSP Y Ysa Ysd) t)) *
  Normal (∏l∈L1 (real (1 - FXl(t)))) *
  (1 -
  (1 - Normal (∏l∈L2 (real (1 - FXl(t)))))) *
  (1 - Normal (∏l∈L3 (real (1 - FXl(t)))))) *
  Normal (∏l∈L4 (real (1 - FXl(t)))) *
  Normal (∏j∈L (1 - real (FX2*j(t) * FX2*j+1(t)))))

```

where `SEN_network_set_req` ensures all the required conditions for the sets to be finite, nonempty and distinct. It also ensures the independence of the input events. It accepts all the sets of the indices of the three levels. The second condition (`rv_gt0_ninfinity [X i]`) ascertains that each element in the group of random variables of `X` that have their indices in $L1 \cup L2 \cup L3 \cup L4 \cup \{2 * i \mid i \in L\} \cup \{2 * i + 1 \mid i \in L\}$ are greater than or equal to 0 but not equal $+\infty$. This condition is required to be able to use the CDF of the random variables.

where Y , Y_{s_a} and Y_{s_d} are indexed random variables that represent the main and spare parts for each spare gate. We choose to use the same hierarchy of Figure 7.15, where we assign index 0 for the first spare and the rest of the spares have their indices in set $L1$. In addition, the model of these additional spares is embedded within X as will be explained shortly.

We verify the probability of failure of the top event as:

Theorem 7.17.

$\vdash \forall p \ L1 \ L2 \ L3 \ L4 \ L \ X \ Y \ Y_{s_a} \ Y_{s_d} \ t.$

$SEN_network_set_req \ p \ L1 \ L2 \ L3 \ L4 \ L$

$(\lambda i.$

$\text{if } i \in \{2 * i \mid i \in L\} \cup \{2 * i + 1 \mid i \in L\} \text{ then } \{i\}$

$\text{else ind_set } [\{0\}; L1; L2; L3; L4] \ i)$

$(\lambda j.$

$\text{if } j \in L \text{ then } \{2 * j; 2 * j + 1\}$

$\text{else ind_set } [\{0\}; \{1\}; \{\}; \{2; 3\}; \{4\}] \ j)$

$(\{0; 1; 3; 4\} \cup L)$

$(\lambda i.$

$\text{event_set } [(DFT_event \ p \ (WSP \ (Y \ 0) \ (Y_{s_a} \ 0) \ (Y_{s_d} \ 0)) \ t, 0)]$

$(rv_to_devent \ p \ X \ t) \ i) \wedge$

$(\forall \ i.$

$i \in L1 \cup L2 \cup L3 \cup L4 \cup \{2 * i \mid i \in L\} \cup \{2 * i + 1 \mid i \in L\} \Rightarrow$

$rv_gt0_ninfinity \ [X \ i]) \wedge$

$(\forall \ i. \ i \in L1 \Rightarrow (X \ i = WSP \ (Y \ i) \ (Y_{s_a} \ i) \ (Y_{s_d} \ i))) \Rightarrow$

$(\text{prob } p$

$(DFT_event \ p \ (Q_{dSEN_Network2}) \ t) =$

1 -

Normal

$$\begin{aligned}
& \left(\prod_{l \in (\{0\} \cup L1)} \left(\text{real} \left(1 - \text{prob} \ p \left(\text{DFT_event} \ p \left(\text{WSP} \ (Y \ l) \ (Ys_a \ l) \ (Ys_d \ l)) \ t \right) \right) \right) \right) * \\
& (1 - \\
& \quad (1 - \text{Normal} \left(\prod_{l \in L2} \left(\text{real} \left(1 - F_{X_1}(t) \right) \right) \right)) * \\
& \quad (1 - \text{Normal} \left(\prod_{l \in L3} \left(\text{real} \left(1 - F_{X_1}(t) \right) \right) \right)) * \\
& \quad \text{Normal} \left(\prod_{l \in L4} \left(\text{real} \left(1 - F_{X_1}(t) \right) \right) \right) * \\
& \quad \text{Normal} \left(\prod_{j \in L} \left(1 - \text{real} \left(F_{X_{2*j}}(t) * F_{X_{2*j+1}}(t) \right) \right) \right)
\end{aligned}$$

where the conditions are similar to Theorem 7.16. However, we add the condition that $(\forall i. i \in L1 \Rightarrow (X \ i = \text{WSP} \ (Y \ i) \ (Ys_a \ i) \ (Ys_d \ i)))$, which adds the additional spare gates. This way, we can use Theorem 7.16 to verify Theorem 7.17. Set $\{0\} \cup L1$ is used to provide the indices of the spares, including the first one with index 0.

We evaluate the probability of failure of the network DFT, shown in Figure 7.16, for a 128×128 SEN+. The DFT of this SEN has 32 AND gates in the first level. Each OR gate in the first level has 160 inputs. Furthermore, we assume that all the 64 input switches have spares. Figure 7.17 shows the evaluated result of the probability of failure, where the failure rates of each switching element is 1×10^{-5} with a dormancy factor of 0.1.

7.4.2 DRBD Analysis of SEN and SEN+

Similar to the DFT models, we start first with the network reliability model of the SEN. Since it is a single path, it can be modeled using the series DRBD of Figure 7.7. Thus, we can use Theorem 7.9 to provide a generic expression for its reliability. We

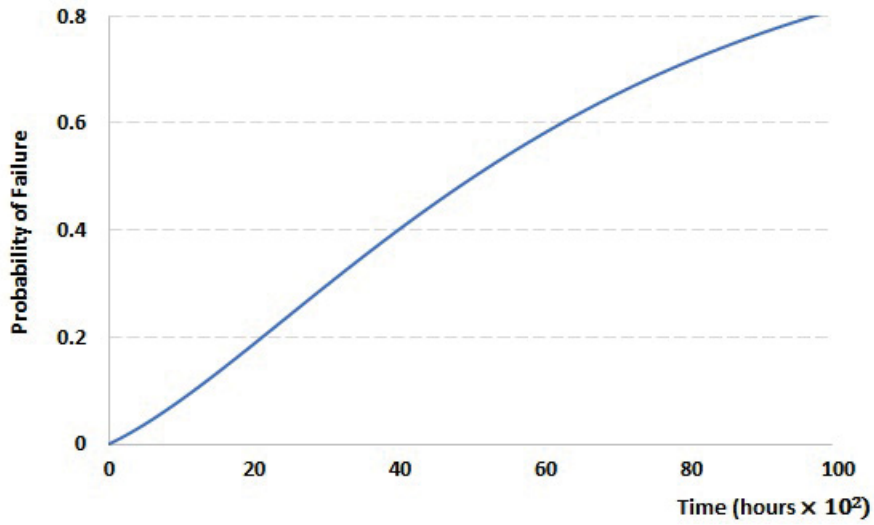


Figure 7.17: The Probability of Failure of the Network of a 128×128 SEN+

provide a generic model in Figure 7.18, where additional spares are used. This provides a general case where we can choose how many switches can be replaced with spares.

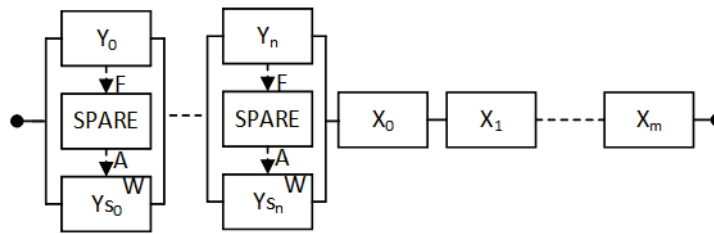


Figure 7.18: DRBD of SEN Network

We express the structure function of this DRBD using our DRBD operators as:

$$\begin{aligned}
 Q_{sSEN_Network} = nR_AND (\lambda i. \quad & \text{if } i \in L1 \text{ then } R_WSP (Y \ i) (Y_{S_a} \ i) (Y_{S_d} \ i) \\
 & \text{else } X \ i) (L1 \cup L2)
 \end{aligned}
 \tag{7.10}$$

where L1 and L2 provide the indices of the blocks in the series structure for the spare

constructs and the remaining blocks, respectively.

Similar to the proof steps of Theorem 7.11, we verify the reliability of the SEN network as:

Theorem 7.18.

$\vdash \forall p X Y Y_{s_a} Y_{s_d} t L1 L2.$

$DISJOINT L1 L2 \wedge FINITE L1 \wedge L1 \neq \{\} \wedge$

$FINITE L2 \wedge L2 \neq \{\} \wedge$

$indep_sets p$

$(\lambda i. \{if i \in L1 then DRBD_event p (R_WSP (Y i) (Y_{s_a} i) (Y_{s_d} i)) t$
 $else (rv_to_event p X t) i\}) (L1 \cup L2) \Rightarrow$

$(prob p (DRBD_event p Q_{SEN_Network} t) =$

$Normal$

$(\prod_{i \in L1}$

$(real (Rel p (R_WSP (Y i) (Y_{s_a} i) (Y_{s_d} i)) t))) *$

$Normal (\prod_{i \in L2} (real (Rel p (X i) t))))$

The DRBD of the SEN+ network is modeled in Figure 7.19, where only one of the switches of the input stage can be replaced by a spare. This DRBD is composed of a series-parallel-series structure. The indices of each level can be treated in a similar manner to the DFT.

We express the structure function using the operators with the same sets of indices of the DFT as:

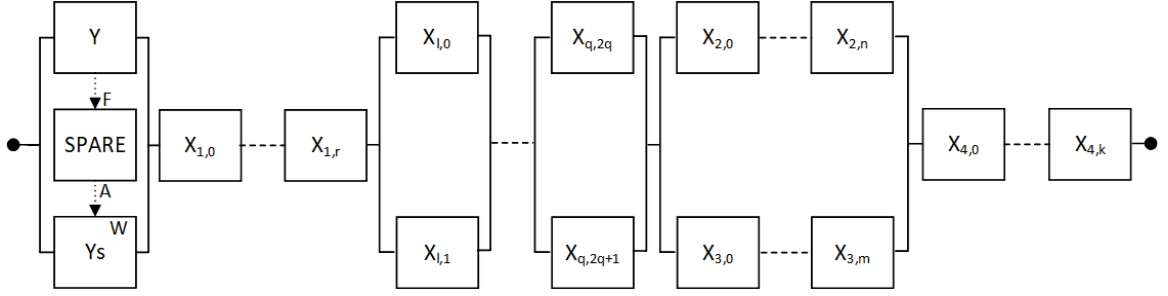


Figure 7.19: DRBD of SEN+ Network

$$Q_{\text{SEN_Network}} = \text{nR_AND}$$

($\lambda i.$

if $i = 0$ then $\text{R_WSP } Y \ Y_{\text{Sa}} \ Y_{\text{Sd}}$

else if $i = 1$ then $\text{nR_AND } X \ L1$

else if $i = 3$ then $(\text{nR_AND } X \ L2) + (\text{nR_AND } X \ L3)$

else if $i = 4$ then $\text{nR_AND } X \ L4$

else $(X \ (2 * i)) + (X \ (2 * i + 1))$

($\{0; 1; 3; 4\} \cup L$))

(7.11)

Then, we verify that the DRBD_event of this structure can be expressed as a series-parallel-series structure as:

Theorem 7.19.

$\vdash \forall p \ L1 \ L2 \ L3 \ L4 \ L \ X \ Y \ Y_{\text{Sa}} \ Y_{\text{Sd}} \ t.$

$\text{FINITE } L1 \wedge L1 \neq \{\} \wedge \text{FINITE } L2 \wedge L2 \neq \{\} \wedge \text{FINITE } L3 \wedge$

$L3 \neq \{\} \wedge \text{FINITE } L4 \wedge L4 \neq \{\} \wedge \text{FINITE } L \wedge$

$\text{DISJOINT } \{0; 1; 3; 4\} \ L \wedge$

$(\forall i. \ i \in L \Rightarrow \text{DISJOINT } \{2 * i; 2 * i + 1\} \ \{0; 1; 2; 3; 4\}) \wedge$

```

disjoint_family_on
  (ind_set
    [{0}; L1; L2; L3; L4;
     {2 * i | i ∈ L} ∪ {2 * i + 1 | i ∈ L}])
  {0; 1; 2; 3; 4; 5} ⇒
(DRBD_event p
  (QSEN_Network) t =
DRBD_series
  (λj.
    DRBD_parallel
      (λa.
        DRBD_series
          (λi.
            event_set
              [(DRBD_event p (R_WSP Y Ys_a Ys_d t,0)]
               (rv_to_event p X t) i)
            ((λi.
              if i ∈ {2 * i | i ∈ L} ∪ {2 * i + 1 | i ∈ L} then {i}
              else ind_set [{0}; L1; L2; L3; L4] i) a))
          ((λj.
            if j ∈ L then {2 * j; 2 * j + 1}
            else ind_set [{0}; {1}; {}; {2; 3}; {4}] j) j))
          ({0; 1; 3; 4} ∪ L))

```

Finally, we verify the reliability of the DRBD as:

Theorem 7.20.

$\vdash \forall p L1 L2 L3 L4 L X Y Ys_a Ys_d t.$

SEN_network_set_req p L1 L2 L3 L4 L

($\lambda i.$

if $i \in \{2 * i \mid i \in L\} \cup \{2 * i + 1 \mid i \in L\}$ then $\{i\}$

else ind_set $[\{0\}; L1; L2; L3; L4]$ i)

($\lambda j.$

if $j \in L$ then $\{2 * j; 2 * j + 1\}$

else ind_set $[\{0\}; \{1\}; \{\}; \{2; 3\}; \{4\}]$ j)

($\{0; 1; 3; 4\} \cup L$)

(event_set $[(DRBD_event\ p\ (R_WSP\ Y\ Ys_a\ Ys_d)\ t, 0)]$

(rv_to_event p X t)) \Rightarrow

(prob p

(DRBD_event p (Q_{SEN_Network}) t) =

Rel p (R_WSP Y Ys_a Ys_d) t *

Normal ($\prod_{l \in L1}$ (real (Rel p (X l) t)))) *

(1 -

(1 - Normal ($\prod_{l \in L2}$ (real (Rel p (X l) t)))) *

(1 - Normal ($\prod_{l \in L3}$ (real (Rel p (X l) t)))) *

Normal ($\prod_{l \in L4}$ (real (Rel p (X l) t)))) *

Normal

($\prod_{j \in L}$

(1 -

real

$$((1 - \text{Rel } p (X (2 * j)) \tau) * (1 - \text{Rel } p (X (2 * j + 1)) \tau))))$$

It is worth mentioning that the conditions of the sets are similar to Theorem 7.16 of the DFT.

Finally, we provide a generic model to have any number of spares that can replace the input switches as shown in Figure 7.20. We choose to use the same indices of Figure 7.19 in order to reuse the verified theorems.

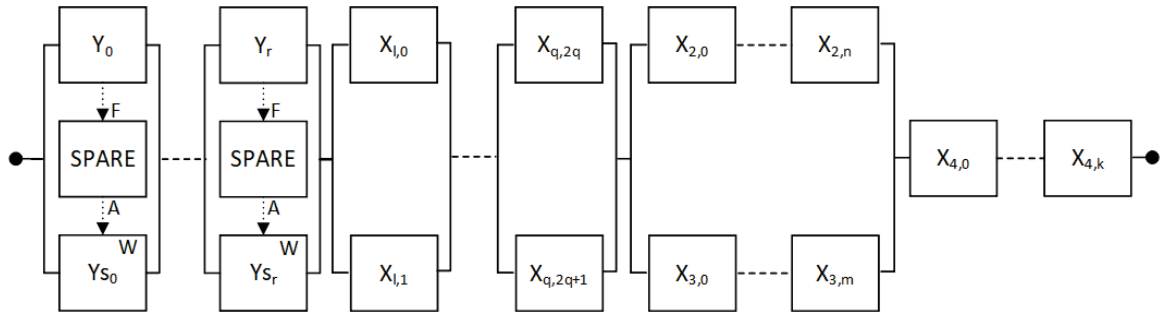


Figure 7.20: DRBD of SEN+ Network with Multiple Spares

We express the structure of the DRBD of Figure 7.20 as:

$$\begin{aligned}
Q_{\text{SEN_Network2}} &= \text{nR_AND} \\
&(\lambda i. \\
&\quad \text{if } i = 0 \text{ then R_WSP } (Y \ 0) \ (Y_{s_a} \ 0) \ (Y_{s_d} \ 0) \\
&\quad \text{else if } i = 1 \text{ then nR_AND } X \ L1 \\
&\quad \text{else if } i = 3 \text{ then (nR_AND } X \ L2) + (\text{nR_AND } X \ L3) \\
&\quad \text{else if } i = 4 \text{ then nR_AND } X \ L4 \\
&\quad \text{else } (X \ (2 * i)) + (X \ (2 * i + 1))) \\
&(\{0; 1; 3; 4\} \cup L)
\end{aligned} \tag{7.12}$$

where $(Y \ 0)$, $(Y_{s_a} \ 0)$ and $(Y_{s_d} \ 0)$ are indexed groups of random variables that represent the main parts and their spares.

Finally, we use Theorem 7.20 to verify the reliability of this DRBD as:

Theorem 7.21.

$$\begin{aligned}
&\vdash \forall p \ L1 \ L2 \ L3 \ L4 \ L \ X \ Y \ Y_{s_a} \ Y_{s_d} \ t. \\
&\text{SEN_network_set_req } p \ L1 \ L2 \ L3 \ L4 \ L \\
&(\lambda i. \\
&\quad \text{if } i \in \{2 * i \mid i \in L\} \cup \{2 * i + 1 \mid i \in L\} \text{ then } \{i\} \\
&\quad \text{else ind_set } [\{0\}; L1; L2; L3; L4] \ i) \\
&(\lambda j. \\
&\quad \text{if } j \in L \text{ then } \{2 * j; 2 * j + 1\} \\
&\quad \text{else ind_set } [\{0\}; \{1\}; \{\}; \{2; 3\}; \{4\}] \ j) \\
&(\{0; 1; 3; 4\} \cup L) \\
&(\text{event_set } [(\text{DRBD_event } p \ (\text{R_WSP } (Y \ 0) \ (Y_{s_a} \ 0) \ (Y_{s_d} \ 0))) \ t, 0])
\end{aligned}$$

$$\begin{aligned}
& (\text{rv_to_event } p \text{ X } t)) \wedge \\
(\forall i. i \in L1 \Rightarrow (X i = \text{R_WSP } (Y i) (Y_{s_a} i) (Y_{s_d} i))) \Rightarrow \\
& (\text{prob } p \\
& \quad (\text{DRBD_event } p \text{ (Q}_{\text{SEN_Network2}}) t) = \\
& \quad \text{Normal} \\
& \quad (\prod_{l \in (\{0\} \cup L1)} \\
& \quad \quad (\text{real } (\text{Rel } p \text{ (R_WSP } (Y l) (Y_{s_a} l) (Y_{s_d} l)) t))) * (1 - \\
& \quad (1 - \text{Normal } (\prod_{l \in L2} (\text{real } (\text{Rel } p \text{ (X l) t)))) * \\
& \quad (1 - \text{Normal } (\prod_{l \in L3} (\text{real } (\text{Rel } p \text{ (X l) t)))))) * \\
& \quad \text{Normal } (\prod_{l \in L4} (\text{real } (\text{Rel } p \text{ (X l) t)))) * \\
& \quad \text{Normal} \\
& \quad (\prod_{j \in L} \\
& \quad \quad (1 - \\
& \quad \quad \quad \text{real} \\
& \quad \quad \quad ((1 - \text{Rel } p \text{ (X (2 * j)) t) * \\
& \quad \quad \quad (1 - \text{Rel } p \text{ (X (2 * j + 1)) t))))))
\end{aligned}$$

We evaluate the network reliability of a 128×128 SEN+ as shown in Figure 7.21. In Figure 7.20, there are 32 parallel structures that are connected in series. The DRBD has 64 spare constructs, while there are 160 blocks in the inner series structures. Finally, the series structure on the right hand side of Figure 7.20 has 64 blocks. We assume that the failure rates of each switching element is 1×10^{-5} with a dormancy factor of 0.1.

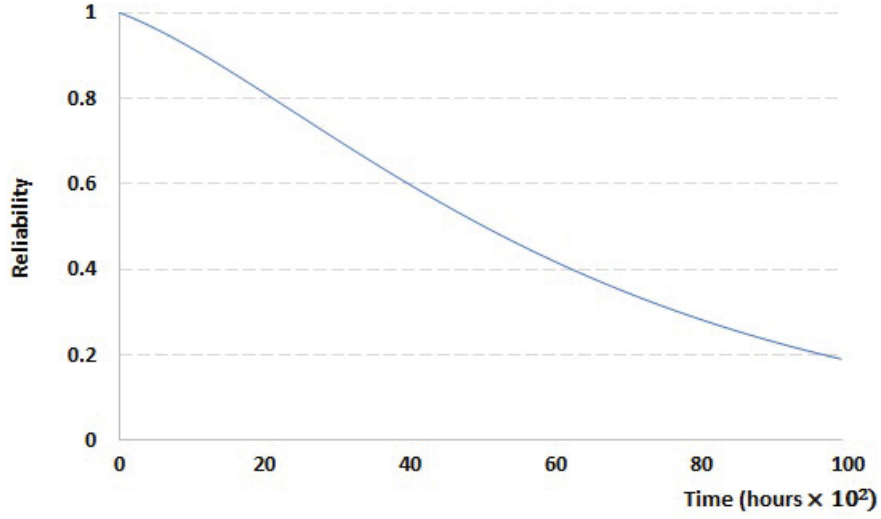


Figure 7.21: The Network Reliability of a 128×128 SEN+

7.5 Equivalence of SEN DFT and DRBD Models

To illustrate the utilization of the DFT-DRBD equivalence part of the proposed thesis methodology, we formally verify the equivalence of the DRBD and the complement of the DFT events for both terminal and broadcast reliability of SEN and SEN+. The equivalence of the network models can be conducted in a similar manner. Proving this equivalence allows verifying the probability of one model and directly use the equivalence proof to provide the probability of the other model.

We verify the equivalence of the DRBD and DFT models of the terminal reliability of both SEN and SEN+ as:

Theorem 7.22. *Terminal/Broadcast SEN*

$\vdash \forall p \ X \ Y \ Y_{s_a} \ Y_{s_d} \ t \ L.$

$\text{FINITE } L \wedge (\forall s. \text{ ALL_DISTINCT } [Y \ s; Y_{s_a} \ s; Y_{s_d} \ s]) \Rightarrow$

$(\text{DRBD_event } p$

$\text{ (nR_AND$

```

      (λi.
        if i = 0 then R_WSP Y Ysa Ysd
        else X i) {0} ∪ L) t =
p_space p DIFF
DFT_event p
(n_OR
  (MAP
    (λi.
      if i = 0 then WSP Y Ysa Ysd
      else X i)
    (SET_TO_LIST ({0} ∪ L)))) t)

```

Theorem 7.23. *Terminal SEN+*

$\vdash \forall p \ X \ Y \ Y_{s_a} \ Y_{s_d} \ Z \ Z_{s_a} \ Z_{s_d} \ t \ L1 \ L2.$

FINITE L1 \wedge FINITE L2 \wedge

$(\forall s. \text{ALL_DISTINCT } [Y \ s; Y_{s_a} \ s; Y_{s_d} \ s; Z \ s; Z_{s_a} \ s; Z_{s_d} \ s]) \Rightarrow$

(DRBD_event p

(nR_AND

(λi.

if i = 0 then R_WSP Y Y_{s_a} Y_{s_d}

else if i = 1 then ((nR_AND X L1) + (nR_AND X L2))

else R_WSP Z Z_{s_a} Z_{s_d}) {0; 1; 2}) t =

p_space p DIFF

DFT_event p

(n_OR

```

(MAP
  (λi.
    if i = 0 then WSP Y Ysa Ysd
    else if i = 1 then
      ((n_OR (MAP X (SET_TO_LIST L1))) .
        (n_OR (MAP X (SET_TO_LIST L2))))
    else WSP Z Zsa Zsd) (SET_TO_LIST {0; 1; 2})) t)

```

In a similar manner, we verify the equivalence of the DRBD and DFT models of the SEN+ broadcast reliability as:

Theorem 7.24. *Broadcast SEN+*

$\vdash \forall p X Y Y_{s_a} Y_{s_d} t L1 L2 s.$

$FINITE L1 \wedge FINITE L2 \wedge FINITE L3 \wedge$

$(\forall s. ALL_DISTINCT [Y s; Y_{s_a} s; Y_{s_d} s]) \Rightarrow$

$(DRBD_event p$

$(nR_AND$

$(\lambda i.$

$if i = 0 then R_WSP Y Y_{s_a} Y_{s_d}$

$else if i = 1 then ((nR_AND X L1) + (nR_AND X L2))$

$else (nR_AND X L3)) (\{0; 1\ 2\}) t =$

$p_space p DIFF$

$DFT_event p$

$(n_OR$

$(MAP$

$(\lambda i.$

```

if i = 0 then WSP Y YSa YSd
else if i = 1 then
    ((n_OR (MAP X (SET_TO_LIST L1))) .
    (n_OR (MAP X (SET_TO_LIST L2))))
else (n_OR (MAP X (SET_TO_LIST L3))))
({0; 1 2})) t)

```

It is worth mentioning that Theorem 7.22 can be used for the equivalence of the DRBD-DFT models of the SEN in both the terminal and broadcast since they both share the same structure.

Based on these theorems, we can use one model to verify the probability of the other model using the probability of the complement.

7.6 Summary

In this chapter, we presented the formal dynamic dependability analysis of SEN and SEN+ MINs that form a critical part in the routing process of multiprocessor systems. We provided generic expressions of reliability and probability of failure that are independent of the failure distributions. Furthermore, we verified these expressions for an arbitrary number of system blocks that can be instantiated later to a certain number without the need to repeat the verification process. For instance, we evaluated the reliability and probability of failure using MATLAB for a specific number of system components based on these generic expressions. It is worth mentioning that such sound generic results cannot be obtained using simulation or model checking as the state space should be defined in advance. The proof script of the verification of SEN and SEN+ is available at [83] and it took around 80 hours to be developed.

Chapter 8

Conclusions and Future Work

8.1 Conclusions

In this thesis, we proposed a framework to formally conduct the dynamic dependability analysis of systems modeled as dynamic fault trees (DFTs) and dynamic reliability block diagrams (DRBDs) using HOL theorem proving. This framework overcomes the limitations of existing techniques, such as simulation and model checking, in terms of soundness and scalability, which allows reasoning about generic properties of a system without specifying particular models.

We provided the formalization of a well-known DFT algebra to ensure the soundness of its mathematical foundations. We formalized the DFT gates and operators and verified several simplification theorems based on the properties of extended-real numbers. This allows obtaining a reduced form of the structure function of a given DFT, which enables having formally verified cut sets and cut sequences to qualitatively analyze a given DFT. We reported our findings of a flaw in one of the published DFT algebras, which further emphasize on the importance of formally validating the

correctness of the mathematical foundations of such algebras, specially if they are to be used in the context of safety-critical systems analysis. Furthermore, we formally verified several probabilistic expressions, based on the HOL4 probability and Lebesgue integral theories, that allow reasoning about the probabilistic behavior of DFT gates and thus performing the DFT quantitative analysis. Furthermore, based on our DFT formalization, we formally verified, using the HOL4 theorem prover, the DFT rewrite rules that are utilized within other DFT analysis tools, such as the STORM model checker. These rewrite rules cover general n -ary OR, AND and Voting gates, as well as the PAND gate.

On the other hand, we developed a novel DRBD algebra, similar to the DFT's, and introduced several operators and simplification theorems to mathematically model traditional RBD structures as well as the dynamic spare construct. We provided the formalization of this novel algebra in HOL and verified the reliability expressions of its structures. Our formalization allows modeling and verifying the most complex structures thanks to its scalable definitions and theorems. For example, we verified several theorems that are concerned with the independence of union and intersection of events in order to verify the reliability of nested structures. Furthermore, our formalization overcomes and outperforms the existing formalization of the static RBD as it allows reasoning about the dynamic behavior of the system that the previous formalization cannot perform.

Due to the complementary modeling nature of DFTs and DRBDs, i.e., DFTs model the failure and DRBDs model the success, our framework allows formally conducting DFT analysis of a system modeled as a DRBD and vice-versa. This means that we are able to reason about both failure and success of a system represented using only one of the models.

The framework developed in this thesis provides generic expressions of probability of failure and reliability that are independent of the failure distribution of system component. This is of great importance as it overcomes the limitation of other formal tools, such as model checking, where the analysis is only limited to exponential distributions that cannot properly capture the dynamic behavior, such as aging. Furthermore, our expressions are verified for an arbitrary number of system components, which is useful in modeling a generic form of the system without the need to specify instances of it, such as multistage interconnection networks (MINs) of multiprocessor systems.

We demonstrated the strengths of our framework by formally conducting the dynamic dependability analysis of multistage interconnection networks (MINs), particularly shuffle-exchange networks (SEN) and SEN+ (an SEN with an additional stage). These SENs connect the processing, memory, and peripherals elements of multiprocessor systems. In particular, we provided the formal dynamic terminal reliability analysis of SEN and SEN+ using both DFT and DRBD models. Moreover, we formally verified the broadcast and network reliability analysis of both types of networks using DRBDs and DFTs. We verified generic expressions for arbitrary number of switching elements that can be instantiated later according to the size of the network without the need to repeat the entire verification process. This highlights the importance and the applicability of our framework in modeling and verifying large-scale systems.

8.2 Future Work

Dynamic dependability analysis is gaining a growing interest as it is able to model more realistic properties of real-world systems. Building a formal framework that utilizes HOL theorem proving is necessary to obtain sound modeling and analysis results. The framework proposed in this thesis represents the foundation of a more general and complete one that allows the analysis of various dependability properties. Based on the formalizations presented in this thesis work, we propose several extensions that can further enrich the available formalization:

- Our proposed framework allows having generic expressions of probability of failure and reliability that can be instantiated using any distribution and density functions that satisfies the required conditions. However, evaluating these expressions when dealing with the Lebesgue integral requires verifying the equivalence of the Lebesgue and Riemann integrals in HOL4. This will further strengthen the proposed framework by adding the possibility of performing and evaluating the expressions based on specific instances of the given system.
- In the current formalization, we have been able to deal with iterated (multiple) integrals with a predetermined number, i.e., the number of integrals is specific and known. For example, we have been able to deal with two iterated integrals that require integrating over a pair measure. However, sometime it is required to have an undetermined number of iterated integrals, i.e., n iterated integrals that require the product of n measures. This can be useful in verifying the probability of failure of n -ary PAND gate and similar expressions. Extending the measure and integral theories in HOL4 to support product measures and iterated integrals will be quite helpful not only within the context of dependability

analysis, but also in reasoning about other probabilistic systems.

- Since HOL4 is an interactive HOL theorem prover, the verification process requires providing interactively most of the proof steps. Furthermore, many proofs have a common pattern and the same steps had to be repeated for each of these proofs, making the verification process tedious. Machine learning can be quite helpful in speeding up and automating the verification process. In [Bio-Cf3], we proposed a road map for conducting the DFT based analysis with the help of machine learning techniques. Particularly, we proposed using TacticToe approach implemented in [84] that automates the selection of the proper tactics to prove a goal in HOL4. This will allow end-users that are unfamiliar with theorem proving to benefit from our DFT and DRBD formalization to provide sound analysis.
- In our proposed DRBD algebra, we have modeled the behavior of the spare construct. However, DRBDs also have other constructs such as load sharing that model the effect of sharing a load among system components on the system reliability and state dependencies that take into consideration the activation and deactivation effect on system reliability. Another open direction for our framework is to develop mathematical models for these constructs and formally verify their reliability expressions, which enables analyzing various systems.
- Dynamic dependability can be modeled and analyzed using CTMCs that capture the dynamic behavior as state machines. Formalizing CTMCs in HOL4 to conduct both the transient and steady state analysis represents another future direction that can be used in the context of dynamic dependability analysis, specially verifying properties, such as the system reliability and availability.

Bibliography

- [1] A Chronology of How the World's Largest and Most Profitable Automaker Drove into a PR Disaster. <http://www.motortrend.com/news/toyota-recall-crisis/>, 2010.
- [2] A. Avizienis, J.C. Laprie, B. Randell, and C. Landwehr. Basic Concepts and Taxonomy of Dependable and Secure Computing. *IEEE Transactions on Dependable and Secure Computing*, 1(1):11–33, 2004.
- [3] E. Ruijters and M. Stoelinga. Fault Tree Analysis: A Survey of the State-of-the-art in Modeling, Analysis and Tools. *Computer Science Review*, 15-16:29 – 62, 2015.
- [4] O. Hasan, W. Ahmed, S. Tahar, and M. S. Hamdi. Reliability Block Diagrams based Analysis: A Survey. In *International Conference of Numerical Analysis and Applied Maths*, volume 1648, page 850129. AIP, 2015.
- [5] I. A. Papazoglou. Mathematical Foundations of Event Trees. *Reliability Engineering & System Safety*, 61(3):169–183, 1998.
- [6] S. Distefano and L. Xing. A New Approach to Modeling the System Reliability: Dynamic Reliability Block Diagrams. In *Reliability and Maintainability Symposium*, pages 189–195. IEEE, 2006.

- [7] Y. Li, P. P. C. Lee, and J. C. S. Lui. Stochastic Analysis on RAID Reliability for Solid-State Drives. In *IEEE International Symposium on Reliable Distributed Systems*, pages 71–80, 2013.
- [8] C. Baier and J.P. Katoen. *Principles of Model Checking*. MIT Press, 2008.
- [9] M. J. Gordon and T. F. Melham. *Introduction to HOL: A Theorem Proving Environment for Higher-order Logic*. Cambridge University Press, 1993.
- [10] PRISM. <http://www.prismmodelchecker.org/>, 2019.
- [11] G. Norman, D. Parker, M. Kwiatkowska, and S. Shukla. Evaluating the Reliability of NAND Multiplexing with PRISM. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, 24(10):1629–1637, 2005.
- [12] C. Dehnert, S. Junges, J.P. Katoen, and M. Volk. A Storm is Coming: A Modern Probabilistic Model Checker. In *Computer Aided Verification*, LNCS 10427, pages 592–600. Springer, 2017.
- [13] M. Ghadhab, S. Junges, J.P. Katoen, M. Kuntz, and M. Volk. Model-based Safety Analysis for Vehicle Guidance Systems. In *Computer Safety, Reliability, and Security*, LNCS 10488, pages 3–19. Springer, 2017.
- [14] M. Kwiatkowska, G. Norman, and D. Parker. Quantitative Analysis with the Probabilistic Model Checker PRISM. *Electronic Notes in Theoretical Computer Science*, 153(2):5–31, 2006.
- [15] J. Hurd. *Formal Verification of Probabilistic Algorithms*. PhD thesis, University of Cambridge, UK, 2002.

- [16] O. Hasan. *Formal Probabilistic Analysis using Theorem Proving*. PhD thesis, Concordia University, Montreal, QC, Canada, 2008.
- [17] T. Mhamdi. *Information-theoretic Analysis using Theorem Proving*. PhD thesis, Concordia University, Montreal, QC, Canada, 2012.
- [18] J. Hölzl. *Construction and Stochastic Applications of Measure Spaces in Higher-Order Logic*. PhD thesis, Technische Universität München, Germany, 2012.
- [19] N. Abbasi, O. Hasan, and S. Tahar. Formal Lifetime Reliability Analysis using Continuous Random Variables. In *International Workshop on Logic, Language, Information, and Computation*, pages 84–97. Springer, 2010.
- [20] O. Hasan, S. Tahar, and N. Abbasi. Formal Reliability Analysis using Theorem Proving. *IEEE Transactions on Computers*, 59(5):579–592, 2010.
- [21] W. Ahmed and O. Hasan. Formalization of Fault Trees in Higher-order Logic: A Deep Embedding Approach. In *Dependable Software Engineering: Theories, Tools, and Applications*, LNCS 9984, pages 264–279. Springer, 2016.
- [22] W. Ahmed, O. Hasan, and S. Tahar. Formalization of Reliability Block Diagrams in Higher-order Logic. *Journal of Applied Logic*, 18:19–41, 2016.
- [23] W. Ahmed and O. Hasan. Towards Formal Fault Tree Analysis using Theorem Proving. In *Intelligent Computer Maths.*, LNCS 9150, pages 39–54. Springer, 2015.
- [24] S. Rajkumar and N.K. Goyal. Review of Multistage Interconnection Networks Reliability and Fault-tolerance. *IETE Technical Review*, 33(3):223–230, 2016.

- [25] F. Bistouni and M. Jahanshahi. Analyzing the Reliability of Shuffle-exchange Networks using Reliability Block Diagrams. *Reliability Engineering & System Safety*, 132:97–106, 2014.
- [26] S.C. Kothari. Multistage Interconnection Networks for Multiprocessor Systems. In *Advances in computers*, volume 26, pages 155–199. Elsevier, 1987.
- [27] HOL4. <https://hol-theorem-prover.org/>, 2019.
- [28] T. Mhamdi, O. Hasan, and S. Tahar. Formalization of Entropy Measures in HOL. In *Interactive Theorem Proving*, LNCS 6898, pages 233–248. Springer, 2011.
- [29] M. Stamatelatos, W. Vesely, J.B. Dugan, J. Fragola, J. Minarick, and J. Railsback. *Fault Tree Handbook with Aerospace Applications*. NASA Office of Safety and Mission Assurance, 2002.
- [30] G. Merle. *Algebraic Modelling of Dynamic Fault Trees, Contribution to Qualitative and Quantitative Analysis*. PhD thesis, ENS, France, 2010.
- [31] K. J. Sullivan, J. B. Dugan, and D. Coppit. The Galileo Fault Tree Analysis Tool. In *IEEE Symposium on Fault-Tolerant Computing*, pages 232–235, 1999.
- [32] G. Merle, J.M. Roussel, J.J. Lesage, V. Perchet, and N. Vayatis. Quantitative Analysis of Dynamic Fault Trees Based on the Coupling of Structure Functions and Monte Carlo Simulation. *Quality and Reliability Engineering International*, 32(1):7–18, 2016.
- [33] G. Merle, J. M. Roussel, J. J. Lesage, and A. Bobbio. Probabilistic Algebraic Analysis of Fault Trees with Priority Dynamic Gates and Repeated Events. *IEEE Transactions on Reliability*, 59(1):250–261, 2010.

- [34] C.Z. Mooney. *Monte Carlo Simulation*. Sage, 1997.
- [35] BlockSim. <https://www.reliasoft.com/products/reliability-analysis/blocksim>, 2019.
- [36] Möbius. <https://www.mobius.illinois.edu/>, 2019.
- [37] isograph. <https://www.isograph.com/>, 2019.
- [38] F. Arnold, A. Belinfante, F. Van der Berg, D. Guck, and M. Stoelinga. DFTCalc: A Tool for Efficient Fault Tree Analysis. In *Computer Safety, Reliability, and Security*, LNCS 8153, pages 293–301. Springer, 2013.
- [39] H. Boudali, P. Crouzen, and M. Stoelinga. Dynamic Fault Tree Analysis using Input/Output Interactive Markov Chains. In *IEEE Dependable Systems and Networks*, pages 708–717, 2007.
- [40] M. Volk, S. Junges, and J.P. Katoen. Fast Dynamic Fault Tree Analysis by Model Checking Techniques. *IEEE Transactions on Industrial Informatics*, 14(1):370–379, 2018.
- [41] Y. Elderhalli, O. Hasan, W. Ahmad, and S. Tahar. Formal Dynamic Fault Trees Analysis Using an Integration of Theorem Proving and Model Checking. In *NASA Formal Methods*, LNCS 10811, pages 139–156. Springer, 2018.
- [42] H. Xu, L. Xing, and R. Robidoux. Drbd: Dynamic Reliability Block Diagrams for System Reliability Modelling. *International Journal of Computers and Applications*, 31(2):132–141, 2009.

- [43] S. Distefano and A. Puliafito. Dynamic Reliability Block Diagrams vs Dynamic Fault Trees. In *Reliability and Maintainability Symposium*, pages 71–76. IEEE, 2007.
- [44] S. Distefano. *System Dependability and Performances: Techniques, Methodologies and Tools* . PhD thesis, University of Messina,, Italy, 2005.
- [45] H. Xu and L. Xing. Formal Semantics and Verification of Dynamic Reliability Block Diagrams for System Reliability Modeling. In *Software Engineering and Applications*, pages 155–162, 2007.
- [46] G. Smith. *The Object-Z Specification Language*, volume 1. Springer Science & Business Media, 2012.
- [47] K. Jensen. A Brief Introduction to Coloured Petri Nets. In *International Workshop on Tools and Algorithms for the Construction and Analysis of Systems*, pages 203–208. Springer, 1997.
- [48] R. Robidoux, H. Xu, L. Xing, and M. Zhou. Automated Modeling of Dynamic Reliability Block Diagrams using Colored Petri Nets. *IEEE Transactions On Systems, Man and Cybernetics*, 40(2):337, 2010.
- [49] T. Mhamdi, O. Hasan, and S. Tahar. On the Formalization of the Lebesgue Integration Theory in HOL. In *Interactive Theorem Proving*, LNCS 6172, pages 387–402. Springer, 2010.
- [50] Isabelle. <https://isabelle.in.tum.de/>, 2019.
- [51] Coq. <https://coq.inria.fr/>, 2019.

- [52] T. Mhamdi, O. Hasan, and S. Tahar. Formalization of Measure Theory and Lebesgue Integration for Probabilistic Analysis in HOL. *ACM Transactions on Embedded Computing Systems*, 12(1):13, 2013.
- [53] M. Qasim. Formalization of Normal Random Variables. Master’s thesis, Concordia University, Montreal, QC, Canada, 2016.
- [54] M. Qasim, O. Hasan, M. Elleuch, and S. Tahar. Formalization of Normal Random Variables in HOL. In *Intelligent Computer Mathematics*, LNCS 9791, pages 44–59. Springer, 2016.
- [55] H. Boudali and J.B. Dugan. A Continuous-time Bayesian Network Reliability Modeling, and Analysis Framework. *IEEE Transactions on Reliability*, 55(1):86–97, 2006.
- [56] G. Merle, J.M. Roussel, and J.J. Lesage. Improving the Efficiency of Dynamic Fault Tree Analysis by Considering Gate FDEP as Static. In *European Safety and Reliability Conference*, pages pp–845. Taylor & Francis, 2010.
- [57] Y. Elderhalli. DFT Formal Analysis: HOL4 Script, Concordia University, Canada, http://hvg.ece.concordia.ca/code/hol/DFT_method/index.php (2019).
- [58] J. Ni, W. Tang, and Y. Xing. A Simple Algebra for Fault Tree Analysis of Static and Dynamic Systems. *IEEE Transactions on Reliability*, 62(4):846–861, 2013.
- [59] A. Altby and D. Majdandzic. Design and implementation of a fault-tolerant drive-by-wire system. Master’s thesis, Chalmers University of Technology, Sweden, 2014.

- [60] H. Boudali, P. Crouzen, and M. Stoelinga. A Rigorous, Compositional, and Extensible Framework for Dynamic Fault Tree Analysis. *IEEE Transactions on Dependable and Secure Computing*, 7:128–143, 2010.
- [61] Y. Elderhalli, W. Ahmad, O. Hasan, and S. Tahar. Probabilistic Analysis of Dynamic Fault Trees using HOL Theorem Proving. *Journal of Applied Logics*, 2631(3):469, 2019.
- [62] MATLAB 2017a, The MathWorks, Natick, 2017.
- [63] P. Billingsley. *Probability and Measure*. John Wiley & Sons, 2012.
- [64] H. Bauer. *Probability Theory*. Walter de Gruyter, 1996.
- [65] G. Merle, J.M. Roussel, and J.J Lesage. Quantitative Analysis of Dynamic Fault Trees based on the Structure Function. *Quality and Reliability Engineering International*, 30(1):143–156, 2014.
- [66] S. Junges, D. Guck, J.P. Katoen, A. Rensink, and M. Stoelinga. Fault Trees on a Diet: Automated Reduction by Graph Rewriting. *Formal Aspects of Computing*, 29(4):651–703, 2017.
- [67] Sebastian Junges. Simplifying Dynamic Fault Trees by Graph Rewriting, 2015. Master Thesis, RWTH Aachen University.
- [68] H. Ehrig, K. Ehrig, U. Prange, and G. Taentzer. *Fundamentals of Algebraic Graph Transformation*. Monographs in Theoretical Computer Science. An EATCS Series. Springer, 2006.
- [69] Y. Elderhalli. DFT Rewriting Rules: HOL4 Script, Concordia University, Canada, <http://hvg.ece.concordia.ca/code/hol/DFT-rewrite/index.php>, 2019.

- [70] Y. Elderhalli. DRBD Formal Analysis: HOL4 Script, Concordia University, Canada, <http://hvg.ece.concordia.ca/code/hol/DRBD/index.php>, 2019.
- [71] Y. Elderhalli. DFT-DRBD Formal Equivalence: HOL4 Script, Concordia University, Canada, <http://hvg.ece.concordia.ca/code/hol/DFT-DRBD-eq.zip>, (2019).
- [72] J.L. Hennessy and D.A. Patterson. *Computer Architecture: A Quantitative Approach*. Elsevier, 2011.
- [73] R. Aggarwal and L. Kaur. On Reliability Analysis of Fault-tolerant Multistage Interconnection Networks. *International Journal of Computer Science and Security*, 2(4):01–08, 2008.
- [74] V.P Kumar and S.M Reddy. Fault-tolerant Multistage Interconnection Networks for Multiprocessor Systems. In *Concurrent Computations*, pages 495–523. Springer, 1988.
- [75] M. Jeng and H.J. Siegel. A Fault-Tolerant Multistage Interconnection Network for Multiprocessor Systems Using Dynamic Redundancy. In *International Conference on Distributed Computing Systems*, pages 70–77. IEEE, 1986.
- [76] N.A.M. Yunus and M. Othman. Reliability Evaluation for Shuffle Exchange Interconnection Network. *Procedia Computer Science*, 59:162–170, 2015.
- [77] F. Bistouni and M. Jahanshahi. Determining the Reliability Importance of Switching Elements in the Shuffle-exchange Networks. *International Journal of Parallel, Emergent and Distributed Systems*, 34(4):448–476, 2019.
- [78] N.A.M. Yunus, M. Othman, Z.M. Hanapi, and Y.L. Kweh. Evaluation of Replication Method in Shuffle-Exchange Network Reliability Performance. In *Advances in Data and Information Sciences*, pages 271–281. Springer, 2019.

- [79] D.K. Panda, R.K. Dash, A.K. Mishra, and S.K. Mohapatra. Reliability Evaluation and Analysis of Multistage Interconnection Networks. *International Journal of Pure and Applied Mathematics*, 119(14):1729–1737, 2018.
- [80] I. Gunawan. Reliability Prediction of Distributed Systems using Monte Carlo Method. *International Journal of Reliability and Safety*, 7(3):235–248, 2013.
- [81] I. Gunawan. Redundant Paths and Reliability Bounds in Gamma Networks. *Applied Mathematical Modelling*, 32(4):588–594, 2008.
- [82] N.A.M. Yunus, M. Othman, Z.M. Hanapi, and K.Y. Lun. Reliability Review of Interconnection Networks. *IETE Technical Review*, 33(6):596–606, 2016.
- [83] Y. Elderhalli. Shuffle-exchange Network Formal Dependability Analysis: HOL4 Script, Concordia University, Canada, <http://hvg.ece.concordia.ca/code/hol/SEN/index.php>, (2019).
- [84] T. Gauthier, C. Kaliszyk, and J. Urban. TacticToe: Learning to reason with HOL4 Tactics. In *Logic for Programming, Artificial Intelligence and Reasoning*, volume 46, pages 125–143, 2017.

Biography

Education

- **Concordia University:** Montreal, Quebec, Canada
Ph.D., Electrical & Computer Engineering, (Jan. 2017 - Dec. 2019)
- **New York Institute of Technology:** Amman, Jordan
M.Sc, Electrical and Computer Engineering, (Aug. 2007 - Oct. 2008)
- **Al Ahliyya Amman University:** Amman, Jordan
B.Sc, Computer Engineering (Oct. 2002 - Feb. 2007)

Work History

- **Concordia University:** Montreal, Quebec, Canada
Research Assistant, Electrical and Computer Engineering (2017-2019)
- **Al Ahliyya Amman University:** Amman, Jordan
Lecturer, Computer Engineering (2009-2015)
- **Al Ahliyya Amman University:** Amman, Jordan
Lab Engineer, Computer Engineering (2007-2009)

Publications

• Journal Papers

- **Bio-Jr1** Y. Elderhalli, O. Hasan, and S. Tahar. “A Methodology for the Formal Verification of Dynamic Fault Trees Using HOL Theorem Proving”, *IEEE Access*, Vol. 7, No. 1, December 2019, pp. 136176-136192.
- **Bio-Jr2** Y. Elderhalli, W. Ahmad, O. Hasan, and S. Tahar. “Probabilistic Analysis of Dynamic Fault Trees using HOL Theorem Proving”, *Journal of Applied Logics*, Vol. 6, No. 3, May 2019, pp. 467-509.

• Refereed Conference Papers

- **Bio-Cf1** Y. Elderhalli, O. Hasan, and S. Tahar. “A Formally Verified Algebraic Approach for Dynamic Reliability Block Diagrams”, In *International Conference on Formal Engineering Methods.*, LNCS 11852, pages 253-269. Springer, 2019.
- **Bio-Cf2** Y. Elderhalli, M. Volk, O. Hasan, J.P. Katoen and S. Tahar: “Formal Verification of Rewriting Rules for Dynamic Fault Trees” In: *Software Engineering and Formal Methods*, LNCS 11724, pages 513-531. Springer, 2019.
- **Bio-Cf3** Y. Elderhalli, O. Hasan and S. Tahar. “Using Machine Learning to Minimize User Intervention in Theorem Proving based Dynamic Fault Tree Analysis”; In: *Artificial Intelligence and Theorem Proving*, 2019.
- **Bio-Cf4** Y. Elderhalli, O. Hasan, W. Ahmad, and S. Tahar. “Formal Dynamic Fault Trees Analysis Using an Integration of Theorem Proving

and Model Checking”. In *NASA Formal Methods*, LNCS 10811, pages 139-156. Springer, 2018.

• Technical Reports

- **Bio-Tr1** Y. Elderhalli, O. Hasan, and S. Tahar, Dynamic Dependability Analysis of Shuffle-exchange Networks using HOL Theorem Proving, Technical Report, Department of Electrical and Computer Engineering, Concordia University, October 2019. <https://arxiv.org/abs/1910.11203>
- **Bio-Tr2** Y. Elderhalli, O. Hasan, and S. Tahar, Integrating DFT and DRBD Formalizations in HOL4, Technical Report, Department of Electrical and Computer Engineering, Concordia University, October 2019. <https://arxiv.org/abs/1910.08875>
- **Bio-Tr3** Y. Elderhalli, O. Hasan, and S. Tahar, A Formally Verified HOL Algebra for Dynamic Reliability Block Diagrams, Technical Report, Department of Electrical and Computer Engineering, Concordia University, August 2019. <https://arxiv.org/abs/1908.01930>
- **Bio-Tr4** Y. Elderhalli, W. Ahmad, O. Hasan, and S. Tahar. Formal Probabilistic Analysis of Dynamic Fault Trees in HOL4, Technical Report, Department of Electrical and Computer Engineering, Concordia University, July 2018. <https://arxiv.org/abs/1807.11576>
- **Bio-Tr5** Y. Elderhalli, O. Hasan, W. Ahmad and S. Tahar. Dynamic Fault Trees Analysis Using an Integration of Theorem Proving and Model Checking, Technical Report, Department of Electrical and Computer Engineering, Concordia University, December 2017. <https://arxiv.org/abs/1712.02872>