Stochastic Computational Models for Gene Regulatory Networks and Dynamic Fault Tree Analysis

by

Peican Zhu

A thesis submitted in partial fulfillment of the requirements for the degree of

Doctor of Philosophy

in

Computer Engineering

Department of Electrical and Computer Engineering

University of Alberta

Abstract

Originally proposed in the 1960s, stochastic computation uses random binary bit streams to encode signal probabilities. Stochastic computation enables the implementation of basic arithmetic functions using simple logic elements. Here, the application of stochastic computation is extended to the domain of gene network models and the fault-tree analysis of system reliability.

Initially, context-sensitive stochastic Boolean networks (CSSBNs) are developed to model the effect of context sensitivity in a genetic network. A CSSBN allows for a tunable tradeoff between accuracy and efficiency in a simulation. Studies of a simple p53-Mdm2 network reveal that random gene perturbation has a greater effect on the steady state distribution (SSD) compared to context switching activities. Secondly, stochastic multiple-valued networks (SMNs) are investigated to evaluate the effect of noise in a WNT5A network. Lastly, asynchronous stochastic Boolean networks (ASBNs) are proposed for investigating various asynchronous state updating strategies in a gene regulatory network (GRN). The dynamic behavior of a T helper network is investigated and the SSDs found by using ASBNs show the robustness of attractors of the network. In a long term, these results may help to accelerate drug discovery and develop effective drug intervention strategies for some genetic diseases.

As another application of stochastic computation, the reliability analysis of dynamic fault trees (DFTs) is further pursued. Stochastic computational models are proposed for the priority AND (PAND) gate, the spare gate and probabilistic common cause failures (PCCFs). Subsequently, a phased-mission system (PMS) is analyzed by using a DFT to model each phase's failure conditions. The accuracy of a stochastic analysis increases with the length of random binary bit streams in stochastic computation. In addition, non-exponential failure distributions and repeated events are readily handled by the stochastic computational approach. The accuracy, efficiency and scalability of the stochastic approach are demonstrated by several case studies of DFT analysis.

Preface

This dissertation makes original contributions in the areas of gene network models and dynamic fault tree analysis. Specifically, stochastic computational models have been developed for a fast analysis of genetic networks and system reliability.

The context sensitivity in a gene network is first considered in the modeling of context-sensitive stochastic Boolean networks (CSSBNs). Chapter 3 is devoted to this work and has been published as: P. Zhu, J. Liang, and J. Han, "Gene Perturbation and Intervention in Context-Sensitive Stochastic Boolean Networks," BMC Systems Biology, 8-60, 2014. I fully developed the initial stochastic models proposed by J. Liang, carried out the network studies, performed the statistical analysis, and drafted the manuscript. Dr. J. Han participated in the network studies, supervised the research, and revised the manuscript.

Stochastic multiple-valued networks (SMNs) are then developed to model the theoretical probabilistic multiple-valued networks (PMNs). The work is presented in Chapter 4 and has been published as: P. Zhu, and J. Han, "Stochastic Multiple-Valued Gene Networks," IEEE Transactions on Biomedical Circuits and Systems, 8(1): 42-53, 2014. I developed the stochastic models, carried out the network studies, performed the statistical analysis and drafted the manuscript. Dr. J. Han participated in the studies, revised the manuscript, and helped in the entire process.

In Chapter 5, asynchronous stochastic Boolean networks (ASBNs) are presented to consider the stochasticity and asynchronicity in the analysis of gene networks. This work has been published as: P. Zhu, and J. Han, "Asynchronous Stochastic Boolean Networks as Gene Network Models," Journal of Computational Biology, 21(10): 760-770, 2014. I developed the asynchronous stochastic models, carried out the GRN studies, performed the statistical analysis and drafted the manuscript. Dr. J. Han participated in discussions and revised the manuscript.

In the analysis of dynamic fault trees (DFTs), stochastic computational models are first proposed for priority AND (PAND) gate, as presented in Chapter 6. This work has been published as: P. Zhu, J. Han, L. Liu, and M. J. Zuo, "A Stochastic Approach for the Analysis of Fault Trees with Priority AND Gates," IEEE Transactions on Reliability, 63(2): 480-494, 2014. I developed the stochastic models for the PAND gate, carried out the case studies, and drafted the manuscript. Dr. J. Han revised the manuscript with the help of Dr. L. Liu and Dr. M. J. Zuo. All authors participated in the discussions.

Next, stochastic computational models are developed for a DFT analysis with spare gates and probabilistic common cause failures (PCCFs). This work is presented in Chapter 7 and has been published as: P. Zhu, J. Han, L. Liu, and F. Lombardi, "A Stochastic Approach for the Analysis of Dynamic Fault Trees with Spare Gates under Probabilistic Common Cause Failures," IEEE Transactions on Reliability, PP(99): 1-15, 2015. I developed the stochastic models for spare gates, carried out the statistical analysis and drafted the manuscript. Dr. J. Han revised the manuscript with the help of Dr. L. Liu and Dr. F. Lombardi. All authors participated in the discussions.

Finally, a phased-mission system (PMS) consisting of H phases is analyzed by H fault trees with each of them modeling the failure conditions of a phase. This work is presented in Chapter 8 and has been submitted for publication in IEEE Transactions on Reliability as: P. Zhu, J. Han, L. Liu, and F. Lombardi, "Reliability Evaluation of Phased-Mission Systems using Stochastic Computation." I proposed the stochastic models for a general PMS, carried out the statistical analysis and prepared the draft manuscript. Dr. J. Han, Dr. L. Liu, and Dr. F. Lombardi revised the manuscript and participated in the discussions.

To my family and friends

Acknowledgements

This dissertation could not have been completed without the help of many people. First of all, I greatly appreciate my supervisor, Dr. Jie Han. During my four years' Ph.D. study, he has always been patient to provide guidance in the research, especially in determining the research topics and discussions of the research problems. He has also been very helpful to revise and edit the papers by providing valuable comments. He has encouraged me to communicate with other researchers by providing various means and support. It was my pleasure to work with him; many thanks to his insights and valuable discussion, both for my personal and academic development.

Then I would like to give special thanks to Dr. Guohui Lin and Dr. Lukasz Kurgan, my supervisory committee members, as well as, Dr. Jie Chen, Dr. Hasan Uludag, and Dr. Jie Han, the candidacy and thesis committee members. Their suggestions and critical comments helped me to improve the dissertation. Many thanks to Dr. Fangxiang Wu for being the external committee member.

Furthermore, I am deeply thankful to all the co-authors of my publications: Jinghang Liang, Hao Chen, Zhixi Yang, Dr. Leibo Liu, Dr. Yangming Guo, Dr. Mingjian Zuo, Dr. Hamidreza Montazeri Aliabadi, and Dr. Fabrizio Lombardi. Without their insight, help and support, the manuscripts could not have been completed or published.

I also want to thank my colleagues who are not mentioned above: Cong Liu, Ran Wang, Honglan Jiang, Siting Liu, Yidong Liu, and Xiaogang Song. Many thanks to your support, kind help and friendship.

Furthermore, I would like to extend my sincere appreciation to my friends and classmates for your support and encouragement. And last but not the least, I want to express my profound gratitude to my family for their unconditional care, love and support.

Table of Contents

List of Figures

List of Tables

List of Acronyms

ARO	all genes updated in a random order
ASBN	asynchronous stochastic Boolean network
BDDs	binary decision diagrams
BN	Boolean network
CCF	common cause failure
cdf	cumulative density function
CSP	cold spare gate
CSPBN	context-sensitive probabilistic Boolean network
CSSBN	context-sensitive stochastic Boolean network
DA-PBN	deterministic-asynchronous probabilistic Boolean network
DFT	dynamic fault tree
FDEP	functional dependency gate
FPGA	field programmable gate array
FTA	fault tree analysis
GAP	gene activity profile
GRN	gene regulatory network
MC	Monte Carlo
MRO	*m* genes updated in a random order
MUX	multiplexer
PAND	priority AND gate
PBN	probabilistic Boolean network
PCCF	probabilistic common cause failure
pdf	probability density function
PMN	probabilistic multiple-valued network
PMS	phased-mission system
RMG	randomly *m* genes

ROG	randomly one gene
SBDDs	sequential binary decision diagrams
SBN	stochastic Boolean network
SC	stochastic computation
SEQ	sequence enforcing gate
SIN	stochasticity in node
SMN	stochastic multiple-valued network
SPP	stochasticity with propensity parameters
SSD	steady state distribution
STM	state transition matrix
UAV	unmanned autonomous vehicle
WSP	warm spare gate

Chapter 1

Introduction

The correct functioning of a biological system involves an extraordinary integrated process. Diverse biological functions are regulated through the interactions among genes, proteins and other molecules in a cell [1]. It has become increasingly difficult to obtain insights about a biological system through the investigation of single genes. Instead, a systems or network-based approach often provides additional information that would otherwise be difficult to obtain from biochemical experiments [2]. Such an approach could further help to gain a better understanding of the mechanisms of diseases and may help to speed up the process of drug discovery and development.

Cell survival and numerous cellular functions are enabled by the amounts and the temporal pattern of genes' products; a gene regulatory network (GRN) governs the genes' expression and product levels [3]. In a GRN, however, gene expressions are affected by intrinsic and extrinsic noise [1]. A major source of the noise is the stochastic fluctuations in regulatory interactions [4]. This indicates the necessity to consider noise in the study of GRNs.

Various methods have been proposed to model the interactions among genes; these include logical models [5], continuous models using differential equations [6]-[9] and stochastic models at the single-molecule level [10][11]. The detailed stochastic simulation algorithms and differential equations-based approaches provide a more accurate simulation of a biological system, however they also require more detailed knowledge about many parameters *a priori*, such as the kinetic rate constants [3][12]. In the so-called approximate stochastic simulation algorithms [13][14], accuracy is sacrificed to improve the algorithmic efficiency; however a long simulation time is still

required, especially in the modeling of large genetic networks. Logical models have widely been used to gain insights into the biological behavior of GRNs. As a classic logical model, Boolean networks (BNs) have been widely used to quantitatively model the interactions among genes [5][15]-[17]. In a BN, discrete values of 0 and 1 are utilized to indicate a gene's expression level, which is referred to as a gene state. The state is obtained by comparing the expression level of a gene to a threshold expression level (usually a typical expression level of most genes in a network). For instance, a state of 1 (indicating the expression of a gene) is given if the expression level is higher than the threshold; otherwise, a value of 0 is obtained. Probabilistic Boolean networks (PBNs) have been proposed to consider noise in a BN model [18]-[20]. In a PBN, the next state of a gene is determined by its current state and a Boolean update function. If the Boolean function is randomly selected, a PBN is referred to as an instantaneous PBN [18]. Using a PBN, the dynamics of a GRN can be investigated and steady state distributions (SSDs) can be derived. In such a network, some states may be associated with diseases such as cancer. Hence, external stimuli can be introduced to deliberately change the states of certain genes to guide a network into a desired state. This process is referred to as gene intervention [19].

As an application of BN, logic circuits have been used to simulate genetic networks [21]. Recently, circuit diagnosis techniques have been utilized to identify the most vulnerable molecules in cellular networks [22]. Stochastic logic has been demonstrated in several biological applications [23][24]. Initially proposed for reliable circuit design, stochastic computation uses random binary bit streams to compute probabilities [25]-[28]. This technique enables effective implementations of arithmetic operations using standard logic elements. The accuracy and efficiency of stochastic computation have been discussed for circuit reliability evaluation [29] and gene network analysis [30]. In [30], stochastic Boolean networks (SBNs) were proposed for a fast implementation of an instantaneous PBN. The SBN approach can recover biologically-proven regulatory behaviors as shown in [31] and [32].

While the temporal evolution of genetic networks can be modeled as a Markov chain of static (but random) Boolean networks at a sequence of time points [33], the required computational complexity presents a significant challenge in evaluating even the static network at each time. In fact, this computational problem exists in a wide class of applications including the evaluation of system reliability that involves dynamic fault trees (DFTs).

Fault tree analysis (FTA) was first proposed in 1962 for evaluating a system's failure probability, the probability that a system fails during a specified mission time [34]. Failures can be disastrous for systems such as chemical plants, nuclear reactors, airplane and computer systems, or costly for systems such as online sales or commercial servers. FTA has developed rapidly, and gained much attention in many applications, especially in the analysis of large safety-critical systems [35]-[38]. In a traditional FTA, dynamic behaviors, such as sequence-dependent, functionally dependent and priority relationships, cannot be modeled properly [39][40]. To account for these dynamic behaviors, DFT analysis has been proposed by incorporating additional dynamic gates such as the priority AND (PAND) and spare gates. A phased-mission system (PMS) undergoes different scenarios for which the failure criteria vary during the mission time. Thus, the system topology of a PMS is usually modeled by a fault tree to indicate the combination of component failures [41]. Hence, the technique of FTA has been applied to evaluate the reliability of a PMS.

Several observations that motivate the development of this dissertation are presented as follows:

1. A combination of Boolean update functions is referred to as a context in a gene network and each update function contributes in determining the next state of a gene. The genetic interactions are inherently context dependent, that is, certain regulatory functions are active in some cellular states, but inactive in others [42]. A context

remains unchanged until a switching occurs. This switching of contexts, possibly caused by external stimuli, is considered to occur randomly in a network. As a general model, a context-sensitive probabilistic Boolean network (CSPBN) considers the feature of context dependence in a gene network model [43]. The analysis of CSPBNs, however, presents a great challenge due to its large computational complexity. As a result, current CSPBN analysis was limited to networks with no more than 15 genes [18][44]. A simulation-based method such as MC simulation [44] requires a large sample size and thus a long run time to meet an accuracy requirement, due to the slow convergence typically encountered in a random sampling-based method [45].

2. In a BN, the Boolean simplification may incur an accuracy loss in the modeling of complex biological networks such as a random Boolean network [46][47]. To address this issue, an approach using multiple-valued variables introduces an increased level of granularity and can be more accurate in the modeling of a GRN [48]-[51]. As a general model, probabilistic multiple-valued networks (PMNs) consider a gene's state at multiple levels in a GRN [33]. However, the application of a PMN analysis is even more severely limited due to an increased computational complexity.
3. The Boolean models usually consider a synchronous update of all genes' states in a network. In biology, this is not necessarily the case because the expression of a gene is seldom an instantaneous process. Instead it may require a few milliseconds or even up to a few seconds [52]; this makes a synchronous model less realistic. In fact, an analysis of the stability of attractors has shown that some attractors may disappear in a BN model with gene perturbation [53]. This indicates that these attractors may be the result of artifacts due to the synchronous updating rules.
4. The reliability analysis of a fault tree becomes challenging when dynamic behaviors and complex failure distributions are considered. For instance, the derivation of analytical expressions becomes cumbersome with the increase of network size. It is also difficult to analyze a system with non-exponentially distributed failure events using Markov models. In addition, the basic components of a system are often subject to common cause failures (CCFs) caused by, for examples, earthquakes,

sudden changes in the environment, design errors, or incorrect operations in practice [54]. As a result, the analysis of DFTs has become increasingly complex.

5. The reliability analysis of a PMS is more challenging than a single-phased system because various factors, such as different system topologies and the dependency between different phases due to the presence of common components, must be considered. This limits the efficiency of existing approaches in the reliability evaluation of a PMS.

1.1. Thesis Statements and Aims

Motivated by the aforementioned observations, we aim at performing a more efficient analysis of gene networks and dynamic fault trees. The following thesis statements are addressed in this dissertation.

1. The challenges in both analytical and simulation approaches such as the CSPBN and MC simulation call for a more efficient analysis of gene network models that considers context sensitivity in a GRN.
2. The increased computational complexity in a PMN analysis requires more efficient computational model that deal with multiple-valued states of genes in a GRN.
3. An asynchronous model is potentially more accurate in discovering the dynamic behavior of a GRN. However, usually a larger number of transitory states are required for deriving an attractor with an asynchronous update strategy. Thus, new gene network models are desired for a fast analysis of the dynamics in an asynchronous genetic network.
4. Due to the limitations in the current methodologies for a DFT analysis, it becomes imperative to develop more efficient methods to analyze the dynamic behaviors in a fault tree.
5. Because of the special features of a PMS, it becomes important to extend the computational models for a general DFT analysis to the reliability analysis of a PMS. A PMS may contain common components at different phases and each common component may have a different failure rate at each phase.

The following objectives are proposed for addressing the thesis statements:

1. **To develop a stochastic computational model that considers context sensitivity in the modeling of GRNs.** In this model, the effects of external perturbation, gene intervention and context-switching activities must be jointly considered and effectively analyzed.
2. **To develop a stochastic computational model that incorporates the feature of multiple values into the analysis of gene activities in a GRN.** This model utilizes multiple-valued logic gates in the simulation of interactions of genes with multiple-valued states.
3. **To develop a stochastic computational model that analyzes the genes' state updating behavior in an asynchronous manner.** The effects of both stochasticity and asynchronicity must be considered in this model.
4. **To develop new stochastic models for the analysis of DFTs consisting of PAND gates and spare gates under the effect of CCFs.** In these models, the dynamic behaviors such as priority and sequence dependency are analyzed by using stochastic computation.
5. **To develop stochastic models for the fast analysis of the reliability of a PMS.** These models make it possible to analyze the reliability of a PMS with common components that fail with different failure rates at different phases.

This dissertation is aimed at advancing the state of the art by extending the application of stochastic computation to the domain of gene network models and the fault-tree analysis of system reliability.

1.2. Thesis Outline

The remainder of this dissertation is organized as follows. Chapter 2 reviews the fundamentals of stochastic computation, the theory of PBNs, the structure of SBNs and the fundamentals of DFT analysis. Chapter 3 presents the first contribution of this dissertation: gene perturbation and intervention in CSSBNs. Chapter 4 describes the second contribution of this dissertation: stochastic multiple-valued gene networks.

Chapter 5 introduces the third contribution of this dissertation: asynchronous stochastic Boolean networks as gene network models. Chapter 6 proposes a stochastic model of the PAND gate for a fast analysis of DFTs; this is the fourth contribution of this dissertation. The fifth contribution of this dissertation is presented in Chapter 7, i.e., a stochastic model of the spare gate for a fast analysis of DFTs under probabilistic common cause failures (PCCFs). The stochastic analysis of a PMS is presented in Chapter 8 as the sixth contribution of this dissertation. Chapter 9 concludes this dissertation and provides a discussion of future work. Contents of this dissertation are based on the publications in the publication list with permissions whenever applicable.

Chapter 2

Background and Related Work

2.1. Stochastic Computation

In stochastic computation, real numbers or probabilities are represented by random binary bit sequences that are readily processed by stochastic logic. Signal probabilities are encoded into random binary bit streams by setting a proportional number of bits to a specific value, i.e., one or zero. Hence, signal probabilities are typically encoded as the proportion of the mean number of ones in a bit stream. **Fig. 2.1** illustrates a stochastic encoding example. The binary bit streams are then processed by stochastic logic. Thus, Boolean logic operations are transformed into probabilistic computations in the real domain based on the following rules (where A, B and C are binary input and output signals, while a, b and c are the signal probabilities of A, B and C) [55][56]:

- Boolean "NOT," or $B = \bar{A}$, which corresponds to $b = 1 - a$;
- Boolean "AND," or $C = AB$, which corresponds to $c = a \cdot b$;
- Boolean "OR," or $C = A + B$, which corresponds to $c = a + b - a \cdot b$.

The complement of a probability can be computed by an inverter and the multiplication of probabilities can be implemented by an AND gate for independent inputs. A multiplexer computes a weighted sum of its input probabilities, with the weights given by the selection inputs. **Fig. 2.2** shows several commonly-used logic gates for stochastic computation. Examples of computation and encoding using a sequence length of 10 bits are shown in **Fig. 2.2(a)** through **(d)**; a longer sequence length is usually required in a practical application, as shown in **Fig. 2.2(e)**. For the 2-to-1 multiplexer of **Fig. 2.2(h)**, the output takes the value of one of the two inputs when the control bit is zero or one. When stochastic sequences are used as input and control signals, this multiplexer selects one of the inputs as the output according to the distributions (and thus the probabilities) of zeros and ones in the control sequence.

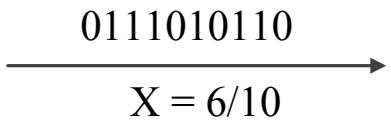


Fig. 2.1. A stochastic encoding example with a sequence length of 10 bits.

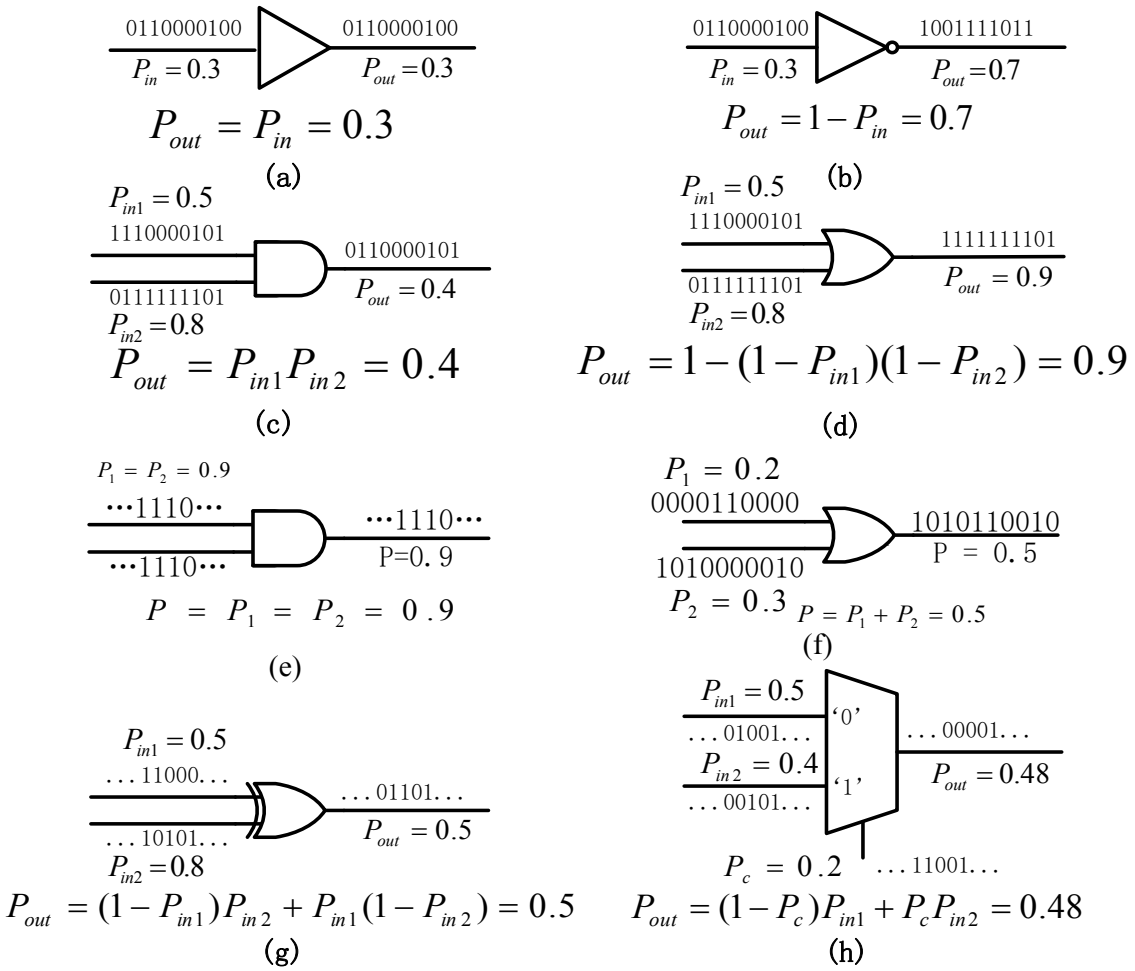


Fig. 2.2. Logic gates for stochastic computation: (a) a buffer; (b) an inverter; (c) an AND gate; (d) an OR gate; (e) an AND with totally dependent inputs; (f) an exclusive OR gate; (g) an XOR gate; and (h) a 2-to-1 multiplexer. Stochastic logic performs a probabilistic analysis by encoding probabilities into random value streams as proportional numbers of different values.

Stochastic computation has the advantages of hardware simplicity and fault tolerance [29][57]. However, inevitable random fluctuations occur in the computation of probabilities. The random fluctuations can be reduced by increasing the bit-stream length

or by using deterministic bit-streams [58][59]. Conventionally, Bernoulli sequences are utilized as random binary bit streams in stochastic computation (i.e., each bit is randomly generated according to a probability). Due to inevitable stochastic fluctuations in stochastic computation, the number of ones in the output sequence is not deterministic but probabilistic. Hence, the output of a stochastic analysis follows approximately a Gaussian distribution when long random binary bit streams are used [29]. However, the use of non-Bernoulli sequences as random permutations of fixed numbers of ones and zeros for initial input probabilities leads to a faster convergence of the result, as stated in the following lemma.

Lemma 1. (*Theorem* 1 in [29]) Compared to the case when Bernoulli sequences are used to encode initial input probabilities, the use of large non-Bernoulli sequences as random permutations of fixed numbers of ones and zeros results in an output sequence with the same mean number of ones, and a smaller variance for an AND gate with statistically independent inputs.

Lemma 1 leads to the conclusion that, to meet a specific accuracy requirement, a smaller sequence length is required by using the non-Bernoulli sequences compared to the use of Bernoulli sequences for encoding initial input probabilities of an AND gate [29].

It is trivial to show that *Lemma* 1 is also applicable to an inverter, thus any logic network (as combinations of inverters and AND gates) can be more accurately evaluated by using the non-Bernoulli sequences as initial input probabilities.

In this type of non-Bernoulli sequences, the numbers of ones and zeros are computed from a specified probability, and then they are randomly permuted to encode the probability. This is a faster process compared to the generation of Bernoulli sequences, because fewer pseudo-random numbers need to be generated. The non-Bernoulli sequences contain deterministic numbers of ones and zeros, so there is no variation in the initial sequences. Therefore, the use of non-Bernoulli sequences as initial inputs results in less variation in the stochastic computation process of a network, thus it

produces more accurate results than the case when Bernoulli sequences are used as initial inputs. Nonetheless, different initial sequences produce different results, but the results approximately follow a Gaussian distribution [29].

When the inputs of a gate are correlated, the output is also determined by the correlation between the input signals. However, signal correlation (usually caused by the re-convergence of signals) is handled readily in stochastic computation. This feature is a particularly favorable property for handling the repeated input events in a complex DFT.

2.2. Probabilistic Boolean Networks

In a cell, biological functions are implemented through the interactions among genes, proteins and other molecules. However, gene networks are noisy due to the effect of stochastic fluctuations in genetic interactions [4]. Various methods have been proposed to model gene regulatory networks (GRNs) [3]. As a classic logical model, Boolean networks (BNs) have been widely used to qualitatively model the interactions among genes by Boolean logic operations [5][15]-[17]. Probabilistic Boolean networks (PBNs) have been proposed to consider noise in a BN model [18]-[20].

For a network of n genes, a PBN is defined by G (V, F), where $V = \{x_1, x_2, \cdots, x_n\}$, a set of binary-valued nodes, $F = (F_1, F_2, \cdots, F_n)$, a list of sets of update functions: $F_i = \left\{f_1^{(i)}, f_2^{(i)}, \cdots, f_{l(i)}^{(i)}\right\}$ and $l(i)$ is the number of possible update functions for gene i, $i = 1,2, \cdots, n$ [18]-[20]. A node x_i represents the state of gene i; x_i = 1 (or 0) indicates that gene i is (or not) expressed. The set F_i contains the rules that determine the next state of gene i. Each $f_{j(i)}^{(i)}: \{0,1\}^n \rightarrow \{0,1\}$ for $1 \leq j(i) \leq l(i)$ is a mapping or predictor function determining the state of gene i.

Due to the stochastic behavior, the next state of gene i is determined by all the $l(i)$ update functions in F_i, i.e., $f_1^{(i)}, f_2^{(i)}, \cdots, f_{l(i)}^{(i)}$ with probabilities $c_1^{(i)}, c_2^{(i)}, \cdots, c_{l(i)}^{(i)}$. A PBN is independent if the update functions from F_i are independent, i.e., the selection of Boolean update functions for gene i has no influence on the selection of Boolean

update functions for gene j $(i \neq j)$ [60]. For an instantaneous PBN of n genes, there are a total number of $\prod_{i=1}^{n} l(i)$ possible BNs, each of which is a possible realization of the genetic network. Each BN can be considered as a possible realization of the genetic network. In the j th context, assume the network function is given by $\boldsymbol{f}_j = (f_{j(1)}^{(1)}, f_{j(2)}^{(2)}, \cdots, f_{j(n)}^{(n)})$, where each $f_{j(i)}^{(i)}: \{0,1\}^n \rightarrow \{0,1\}$ for $1 \leq j(i) \leq l(i)$ is a mapping or predictor function determining the state of gene i. The probability that the jth context is selected, is obtained as $C_j = \prod_{i=1}^{n} c_{j(i)}^{(i)}$ for j = 1, 2, ⋯, k, where k is the number of contexts, each of which is a possible realization of the genetic network and referred to as a context.

At time t, the state of a genetic network can be described by a vector, i.e., $\boldsymbol{x}(t) = (x_1(t), x_2(t), \cdots, x_n(t))$, where the state of a gene i at time t is given by $x_i(t)$, $x_i(t) \in \{0,1\}$ for $i \in \{1,2,\cdots,n\}$. A network state is also referred to as a gene activity profile (GAP). A GAP is also given as a decimal index, i.e., $d = \sum_{j=1}^{n} 2^{n-j} x_j(t) + 1$.

2.3. Stochastic Boolean Networks

Recently, stochastic models have been proposed for a fast computation of the state transition matrix (STM) and steady state distribution (SSD) of an instantaneous PBN, referred to as stochastic Boolean networks (SBNs) [30].

2.3.1. Stochastic Boolean Networks without Perturbation

For a PBN, the next states of genes are updated by a set of Boolean functions, each of which is selected with certain probabilities. In an SBN, random binary bit sequences are used to represent those probabilities. The Boolean update functions are selected by a stochastic multiplexer controlled by proper sequences. A structure of an SBN for a single gene is shown in **Fig. 2.3**.

As shown in **Fig. 2.3**, the next state of gene i is determined by a total number of $l(i)$ Boolean functions and the selection of functions is implemented by a multiplexer for

gene i. Thus $m = \lceil log_2(l(i)) \rceil$ bits are required to control the multiplexer. For one gene, the number of possible Boolean functions is usually small, between 1 and 4 for 93% of genes [61][62]. Thus, one or two bits are sufficient to control the multiplexer in an SBN. Hence, an update function is selected with probability $c_{j(i)}^{(i)}$ for gene i in an SBN. This SBN structure accurately implements the probabilistic functions of a PBN if all the genes are considered.

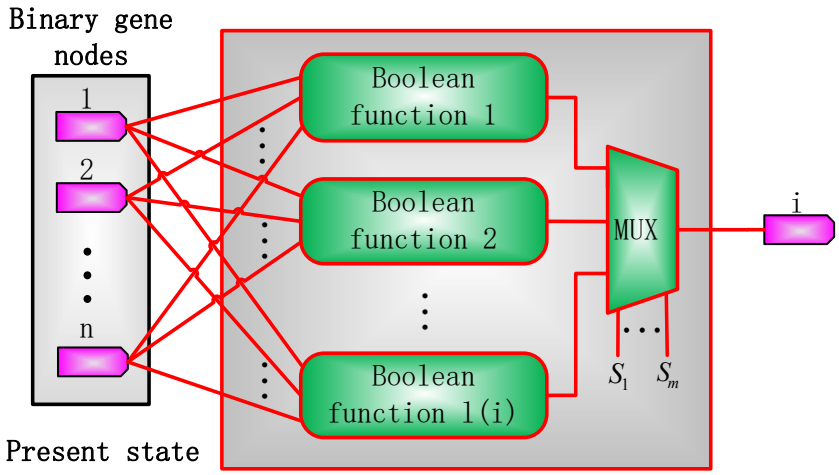


Fig. 2.3. A stochastic Boolean network (SBN) without perturbation for a single gene [30].

2.3.2. Stochastic Boolean Networks with Perturbation

In a PBN with perturbation, a gene may change its value with a small probability p during transition. Let $\boldsymbol{x} = (x_1, x_2, \ldots, x_n)$ and $\boldsymbol{\gamma}$ represent the current state of an n-gene network at time t and the random perturbation vector respectively, then the next state $\boldsymbol{x}'$ is given by [19]:

$$\boldsymbol{x}' = \begin{cases} \boldsymbol{x} \oplus \boldsymbol{\gamma} & with\ 1-(1-p)^n \\ \boldsymbol{f}_k(\boldsymbol{x}) & with\ (1-p)^n \end{cases} \tag{2.1}$$

where $\oplus$ indicates a modulo 2 additions and $\boldsymbol{f}_k(\cdot)$ is the kth BN at time t.

A general model of SBNs with perturbation is shown in **Fig. 2.4**. As can be seen, the addition modulo 2 of the perturbation vector and the present state is implemented by XOR gates. Either a Boolean function works or a perturbation occurs is determined by

the output sequence of an n-input OR gate. If there is no perturbation, the output sequence of the OR gate contains all zeros; thus the next state is given by the original SBN without perturbation. If perturbation occurs, the next state is determined by the output sequence of the stochastic OR gate. Hence, the SBN in **Fig. 2.4** accurately implements the function of (2.1).

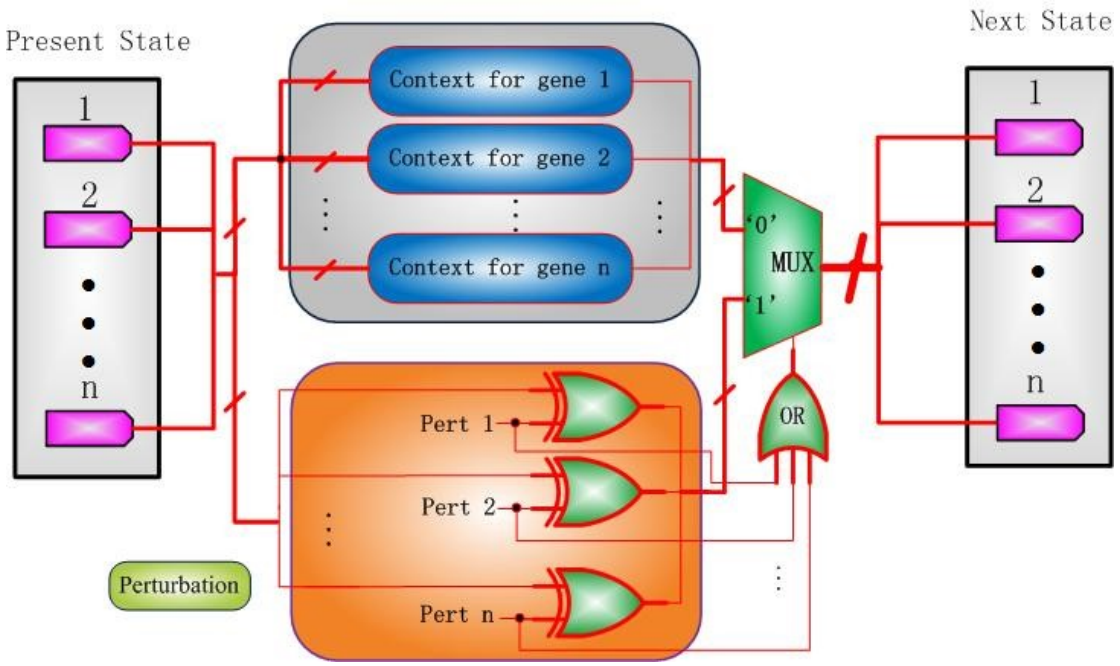


Fig. 2.4. A stochastic Boolean network (SBN) for a network of n genes with perturbation [30].

In an SBN, the complexity to compute the STM is $O(nL2^n)$, where L is the minimum sequence length required for obtaining an evaluation accuracy. For a network with a large number of genes, L is significantly smaller than *N,* the number of Boolean networks. By using a time-frame expanded structure of the SBN, the SSD can be evaluated (the technique of time-frame expansion is presented in detail later). The SBN approach can recover biologically-proven regulatory behaviors, such as the oscillatory dynamics of a simplified p53-Mdm2 network [31] and the dynamic attractors in a T cell immune response network [32]. The interested reader is referred to [30] for more detail.

2.3.3. Steady State Distribution Analysis

The state transition of a PBN can be described by an STM as:

$$\boldsymbol{A} = \begin{bmatrix} Pr(1|1) & Pr(2|1) & \cdots & \cdots & Pr(2^n|1) \\ Pr(1|2) & Pr(2|2) & \cdots & \cdots & Pr(2^n|2) \\ \cdots & \cdots & \cdots & \cdots & \cdots \\ \cdots & \cdots & \cdots & \cdots & \cdots \\ Pr(1|2^n) & Pr(2|2^n) & \cdots & \cdots & Pr(2^n|2^n) \end{bmatrix}, \tag{2.2}$$

where each entry is a conditional (transition) probability that the genes transfer from a given present state into a next state with the state indicated by a decimal index.

The STM $\boldsymbol{A}$ can be obtained by $\boldsymbol{A} = \sum_{j=1}^{N} c_j \boldsymbol{A_j}$, where c_j is the probability that the jth BN is selected and $\boldsymbol{A_j}$ is the STM due to the jth BN. Hence, the complexity of computing $\boldsymbol{A}$ is $O(nN2^{2n})$ [64]. In an open genome system, random inputs under external stimuli can incur random gene perturbation [19]. Due to a perturbation, the state of a gene flips from 1 to 0 (or vice versa). A perturbed STM can be derived for a genetic network under gene perturbation [20]. Any PBN with perturbation will reach a steady state in a long run due to the property of an aperiodic and irreducible homogeneous Markov chain [64].

Given an initial state distribution $\boldsymbol{x}^{(0)}$, let $\boldsymbol{x}^{(m)}$ and $\boldsymbol{x}^{(m+1)}$ be the state distributions after m and $m+1$ transitions respectively. Assume that the STM of the network is given by $\boldsymbol{A}$, then the state transitions from $\boldsymbol{x}^{(m)}$ to $\boldsymbol{x}^{(m+1)}$ are described by:

$$\boldsymbol{x}^{(m+1)} = \boldsymbol{x}^{(m)} \cdot \boldsymbol{A}, \tag{2.3}$$

If $\left\|\boldsymbol{x}^{(m+1)} - \boldsymbol{x}^{(m)}\right\|_{\infty}$ is used to compute the maximum absolute value of the summed difference of each row in $\boldsymbol{x}^{(m)}$ and $\boldsymbol{x}^{(m+1)}$, the condition for reaching a steady state is given by [63]:

$$\left\|\boldsymbol{x}^{(m+1)} - \boldsymbol{x}^{(m)}\right\|_{\infty} < \varepsilon, \tag{2.4}$$

where ε indicates a threshold for determining whether the steady state has been reached or not. If (2.4) is met, a genetic network is considered to have reached a steady state, i.e., $\boldsymbol{x}^{(\infty)} = \boldsymbol{x}^{(m)}$; thus $\boldsymbol{x}^{(m)}$ is considered as the stationary distribution.

In order to measure the disparity of the obtained matrices and steady states, the norms $\|\cdot\|_1$, $\|\cdot\|_2$, and $\|\cdot\|_\infty$ are used to measure the differences of the STMs or SSDs obtained by various methods. $\|\cdot\|_1$ and $\|\cdot\|_\infty$ indicate the maximum absolute values of the summed differences of the columns and rows respectively, while $\|\cdot\|_2$ measures the average difference of all entries. For a vector $\boldsymbol{x}$, the norms are defined as $\|\boldsymbol{x}\|_1 = \sum_{i=1}^{n}|x_i|$, $\|\boldsymbol{x}\|_2 = \sqrt{\sum_{i=1}^{n}|x_i|^2}$, and $\|\boldsymbol{x}\|_\infty = max_{1\le i\le n}|x_i|$, where n is the number of elements in the vector $\boldsymbol{x}$.

Due to the large size of the STM for a PBN of a large number of genes, it becomes difficult to evaluate the SSD using the STM-based analysis. However, the SSD can be evaluated through an iterative simulation of stochastic model in the temporal domain (or the so-called time-frame expansion technique [30]). An illustration is presented in **Fig. 2.5**. By this technique, an iterative structure of the SBN is used to simulate the temporal evolution of a GRN. The required number of iterations is determined by the number of state transitions before reaching a steady state. As an alternative to an STM-based analysis, the simulation of stochastic models provides flexibility in achieving a tunable accuracy-efficiency tradeoff by using stochastic sequences of different lengths.

For an SSD analysis, random bit sequences are first generated for the initial input signals and the control bits of multiplexers. Then, the sequences propagate through the iterative SBN structure. This process is equivalent to multiplying the initial input probabilities with the powers of the STM, as given by (2.3). Given a threshold, if (2.4) is met, the system is considered to have reached a steady state. The SSD is then obtained from the final output sequence of the time-frame expanded SBN structure. As shown in [30], a time-frame expanded SBN provides an alternative and fast means to estimate the SSD of a PBN.

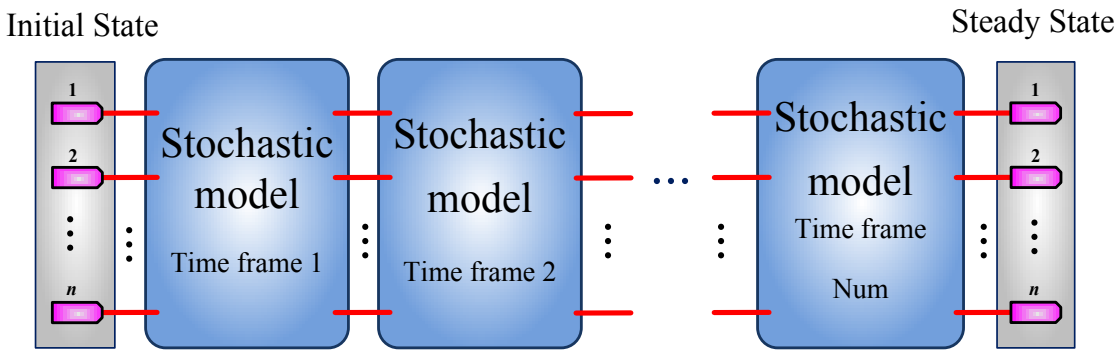


Fig. 2.5. A general time-frame expanded architecture. n: the number of genes in a genetic network. Num: the number of time frames required for deriving a steady state distribution (SSD).

2.4. Dynamic Fault Tree Analysis

Over the last few decades, fault tree analysis (FTA) has been widely applied to the analysis of various systems, including chemical plants, nuclear reactors, airplane controllers, and computers [35]. The so-called dynamic fault tree (DFT) has been developed to simulate the dynamic behavior of a system. This has been accomplished by incorporating several additional dynamic gates, such as the priority AND gate (PAND), the sequence enforcing gate (SEQ), the standby or spare gate (Spare), and the functional dependency gate (FDEP) [65][66].

For systems with perfect fault coverage, the FDEP can be treated as an OR gate [67][68]; the SEQ gate can be regarded as a special case of a cold spare gate (CSP) [69]. Furthermore, a hot spare gate (HSP) is logically equivalent to an AND gate [70]. Hence, the study of PAND and Spare (mainly CSP and WSP) gates becomes essential for a DFT analysis. For a PAND gate, an input indicates the firing of a basic event that occurs in a predetermined order, and the output indicates whether a failure occurs [37][71]. The spare gate is usually used to model a standby system consisting of two types of modules: the primary (or online) modules and the standby modules. When the primary module fails, standby modules are used to replace the faulty modules to keep the system functional or operational. Hence, the spare gate fires (i.e., fails) if both of the modules fail.

A phased-mission system (PMS) undergoes different phases of the mission time and the failure criteria vary at each phase. The mission can fail in any of these phases and the PMS must be evaluated to obtain the failure probability of each phase. The PMS achieves an overall mission success only if every phase successfully completes the task. Hence, the overall mission failure is obtained by a logic OR of the failures of all phases [72]. The topology of each phase can be described by a fault tree (or a dynamic fault tree if dynamic behaviors must be considered). Thus, FTA can be used to evaluate the reliability of a PMS.

2.5. Summary

This chapter reviews the fundamentals of stochastic computation, the theory of PBNs and the structures of SBNs. The use of non-Bernoulli sequences of fixed counts of zeros and ones is also introduced as an effective method to improve the accuracy and efficiency of stochastic computation. As another application of stochastic computation, the fundamentals of the DFT analysis are also reviewed, including the definitions of PAND gates, spare gates and a PMS.

Chapter 3

Gene Perturbation and Intervention in Context-Sensitive Stochastic Boolean Networks

In a gene regulatory network (GRN), gene expressions are affected by noise, and stochastic fluctuations exist in the interactions among genes. These stochastic interactions are context dependent, thus it becomes important to consider noise in a context-sensitive manner in a network model. As a logical model, context-sensitive probabilistic Boolean networks (CSPBNs) accounts for molecular and genetic noise in the temporal context of gene functions, so the study of CSPBNs have provided insights into the understanding of gene perturbation and intervention. Analytical expressions have been derived for analyzing CSPBNs [73]; however, this method is only applicable to the steady state analysis of a network with small perturbation and switching probabilities. The method in [61] ignores some Boolean networks (BNs) with very small probabilities for reducing the size of the state transition matrix (STM) and thus provides a faster but approximate solution for computing the steady state distribution (SSD) of a CSPBN. Stochastic logic has been demonstrated in several biological applications [23][24]. Hence, in this chapter, a novel structure of context-sensitive stochastic Boolean networks (CSSBNs) is proposed for modeling the stochasticity in a GRN. The proposed CSSBN model enables a fast simulation of CSPBNs. The results in this chapter have been published as [74].

The novelty of this chapter is as follows:

- Based on the original stochastic Boolean networks (SBNs) in [30], new SBN models are developed to evaluate the effect of gene perturbation and intervention in a context-sensitive environment in a GRN. These models are referred to as CSSBNs.
- A CSSBN analysis provides meaningful insights into the dynamics of the p53-Mdm2 network in a context-switching environment.
- The CSSBN models are used to predict the SSD of genes in a glioma network.

3.1. Context-Sensitive Probabilistic Boolean Networks

In a context-sensitive PBN (CSPBN) with k contexts, a network function is defined as $\boldsymbol{f}_j = (f_j^{(1)}, f_j^{(2)}, \cdots, f_j^{(n)})$, where $f_j^{(i)}: \{0,1\}^n \rightarrow \{0,1\}$ is the predictor function for gene i, i = 1, 2, ⋯, n, in context j (j = 1, 2, ⋯, k) [43][75]. In the jth context, assume the network function is given by $\boldsymbol{f}_j = (f_{j(1)}^{(1)}, f_{j(2)}^{(2)}, \cdots, f_{j(n)}^{(n)})$, where each $f_{j(i)}^{(i)}: \{0,1\}^n \rightarrow \{0,1\}$ for $1 \leq j(i) \leq l(i)$ is a mapping or predictor function determining the state of gene i. The probability that the jth context is selected, is obtained as $C_j = \prod_{i=1}^{n} c_{j(i)}^{(i)}$ for $j = 1, 2, \cdots, k$, where k is the number of contexts in a context-sensitive network [61]. The state of a gene is updated in the selected context; thus the next state depends on both the present state and the selected context.

In a context-sensitive network, a context may remain for certain time until a random event occurs. A context switching usually occurs with probability q. For $q = 1$, the CSPBN becomes an instantaneous PBN. For $q < 1$, if a new context is to be selected, it is randomly chosen from the set of network functions: $\{\boldsymbol{f}_1, \boldsymbol{f}_2, \cdots, \boldsymbol{f}_k\}$, with a set of context selection probabilities: $\{C_1, C_2, \cdots, C_k\}$ [75]. If noise is considered in a CSPBN, it is often referred to as a perturbation, by which a gene flips its state with a probability p ($p \neq 0$). It has been shown that a PBN with perturbation is an ergodic Markov chain in that all the states are connected in the PBN [19]. The transition probability for any two states is determined by the values of p and q pairs. Following [73], one of four mutually exclusive events occurs at time t in a CSPBN with perturbation:

- $\emptyset_1$: The predictor functions in the currently selected context are applied to update the gene expressions and this context remains for the next transition.
- $\emptyset_2$: The predictor functions in the currently selected context are applied to update the states of the genes and then a new context is selected for the next transition.
- $\emptyset_3$: A random perturbation occurs and the currently selected context remains for the next transition.
- $\emptyset_4$: A random perturbation occurs and a new context is selected for the next

transition.

The effect of a switching order, i.e., whether a network switches its context before or after its state transition, is considered in [76], whereas in this dissertation, we focus on the transition rules, i.e., the network function is applied first and then the context switches.

A gene activity profile (GAP) is defined as a vector for describing the state of a network at time t, $\boldsymbol{x}(t) = (x_1(t), x_2(t), \cdots, x_n(t))$, where $x_i(t) \in \{0,1\}$ for $i = 1, 2, \cdots, n$. The state of a CSPBN can be represented as a combination of a context and a GAP, i.e., $\boldsymbol{S} = (context\ j, GAP\ i)$. By this definition, the number of states in a CSPBN increases from 2^n to $2^n k$ for a network with n genes and k contexts. A GAP can also be represented by its decimal index, i.e., $d = \sum_{j=1}^{n} 2^{n-j} x_j(t) + 1$. A state of a CSPBN is then given by $\boldsymbol{S} = (f_i, d)$ for $i \in \{1, 2, \cdots, k\}$ and $d \in \{1, 2, \cdots, 2^n\}$, where d is the decimal index of a GAP. For convenience, a state (f_i, d) is referred to as (i, d) in the following analysis.

In a CSPBN with a perturbation probability p and a switching probability q, the transition probability for any two states $\boldsymbol{a}$ and $\boldsymbol{b}$ is given by [76]:

$$P(\boldsymbol{S}_{t+1} = \boldsymbol{b} | \boldsymbol{S}_t = \boldsymbol{a}) = \{(1-p)^n f_{r_1,x_1,x_2} + (1-p)^{n-h} p^h s(h)\} \cdot \tag{3.1}$$

$$[\left(1 - q + q c_{r_1}\right) g(a,b) + q c_{r_2} (1 - g(a,b))],$$

where

$$f_{r_1,x_1,x_2} = \begin{cases} 1, & if\ x_1\ directly\ transitions\ to\ x_2\ in\ context\ r_1 \\ 0, & otherwise \end{cases} \tag{3.2}$$

$$g(\boldsymbol{a},\boldsymbol{b}) = \begin{cases} 1, & if\ r_1 = r_2 = r \\ 0, & otherwise \end{cases} \tag{3.3}$$

$$s(h)=\begin{cases}0, & if\ h=0\\ 1, & otherwise\end{cases} \tag{3.4}$$

and h is the Hamming distance between two GAPs with decimal indices x_1 and x_2, $x_1, x_2 \in \{1, 2, \cdots, 2^n\}$. This distance indicates the number of genes with different expressions in the two GAPs x_1 and x_2.

The STM of a CSPBN without perturbation is of the size $2^n \cdot k \times 2^n \cdot k$, given by [76]:

$$A=\begin{bmatrix}(1-q+qc_1)\boldsymbol{P}_1 & qc_2\boldsymbol{P}_1 & \cdots & c_k\boldsymbol{P}_1\\ \cdots & \cdots & \cdots & \cdots\\ qc_1\boldsymbol{P}_k & qc_2\boldsymbol{P}_k & \cdots & +qc_k)\boldsymbol{P}_k\end{bmatrix} \tag{3.5}$$

and the STM of a CSPBN with perturbation is given by:

$$A=\begin{bmatrix}(1-q+qc_1)\tilde{\boldsymbol{P}}_1 & \cdots & \cdots & c_k\tilde{\boldsymbol{P}}_1\\ \cdots & \cdots & \cdots & \cdots\\ qc_1\tilde{\boldsymbol{P}}_\kappa & \cdots & \cdots & \cdots\tilde{\boldsymbol{P}}_\kappa\end{bmatrix}, \tag{3.6}$$

where $\boldsymbol{P}_i$ and $\tilde{\boldsymbol{P}}_i$ denote the STMs for the Boolean network i, $i \in \{1, 2, \cdots, k\}$, without and with perturbation respectively.

The study of PBNs has focused on the analysis of SSDs under gene perturbation and intervention. A Markov Chain Monte Carlo (MCMC) method is used in [44] to analyze the long run behavior of a PBN; however, this method generally requires a large number of simulations to reach a steady state, due to the slow convergence typically encountered in a Monte Carlo method [45]. An analysis is performed in [63] for finding the SSD of a PBN through the computation of the STM. However, the application of an analytical approach is generally limited to small networks due to the exponential increase

in the size of an STM with gene numbers. The analysis of CSPBNs presents even a greater challenge due to its significantly increased computational complexity. In [43][77][78], gene intervention is investigated for avoiding undesirable states associated with certain diseases (such as cancer). Due to external stimuli, the STM is changed by external control variables, so desirable states can be obtained with larger probabilities in the SSD. In a context-sensitive network with n genes and k contexts, however, a $(2^n \cdot k) \times (2^n \cdot k)$ [76] (or $2^n \times 2^n$ [61]) matrix is required for an accurate (or approximate) analysis of the SSD; this results in a computational complexity of $O(nk^2 2^{2n})$ (or $O(nk2^n)$) for an accurate (or approximate) computation of the STM. Hence, the application of gene network analysis was limited to those of less than 15 genes, due to the large computational complexity required.

3.2. Context-Sensitive Stochastic Boolean Networks without Perturbation

In a CSPBN, the selection of a context is determined by the context switching probability. This probability indicates the likelihood to maintain the current context i or to select a new context from the k contexts (including the currently selected context i). Based on the present state and the selected context, a gene's state is updated. If no perturbation occurs in an n-gene CSPBN with a switching probability q, the transition probability from state $(\boldsymbol{s}, y)$ to $(\boldsymbol{r}, x)$ is given by [76]:

$$p((r,x)\,|\,(s,y)) = \begin{cases} f_{s,y,x}(1-q+qC_s) & \text{if } r = s \\ f_{s,y,x}(qC_r) & \text{if } r \neq s \end{cases} \tag{3.7}$$

where

$$f_{s,y,x} = \begin{cases} 1, & \text{if } y \text{ directly transitions to } x \text{ in context } s \\ 0, & \text{otherwise} \end{cases} \tag{3.8}$$

$\boldsymbol{s}$ and $\boldsymbol{r}$ denote the sth and rth contexts, y and x represent two GAPs (in decimal indices), and C_s and C_r indicate the probability of selecting the sth and rth contexts respectively.

An SBN structure is proposed in [30] to implement an instantaneous PBN. For a CSPBN, a CSSBN is constructed to consider the switching of contexts, as shown in **Fig. 3.1**. In this CSSBN model, the probabilistic switching is implemented using a multiplexer in stochastic computation and the switching probability q is encoded as a random binary sequence Q that serves as the control sequence of a 2-to-1 multiplexer (MUX). If the jth bit in the sequence Q is 1, a new context will be selected for the next transition. Otherwise, the current context will remain. The selection probability of the new context is determined by the original and current context selection probabilities. As shown in the lower section of **Fig. 3.1**, this process is implemented by 2-to-1 multiplexers with the original and current context selection sequences as inputs and Q as the control sequence. The selection probability of the new context is then obtained as encoded in the output stochastic sequences of the multiplexers. If q = 0, the CSSBN functions as a fixed BN. If q = 1, the CSSBN is simplified to an instantaneous SBN.

If a switching does occur, the context selection process is implemented by another multiplexer for choosing one from the k contexts, according to predefined selection probabilities. As a network function is a combination of each gene's predictor function, a combination of the m control sequences of $S_1 \sim S_m$ is used to encode the predefined selection probabilities; this in turn determines the selection probability of each new context. For a CSPBN with n genes, the CSSBN needs to be run for each of the 2^n input states and sequences need to be generated for the m control signals of the multiplexer.

For a switching probability of q in the proposed CSSBN, the currently selected context i remains at time t with a probability of q and switches to one of the k contexts with a probability of $1-q$. If the network transitions from GAP y to x in the sth context (i.e., $f_{s,y,x} = 1$), then context $\boldsymbol{s}$ will remain with probability $1 - q + qC_s$

at the next time step. Otherwise, the network moves into a new context $\boldsymbol{r}$ with probability qC_r. From this analysis, it can be seen that the CSSBN in **Fig. 3.1** computes the transition probability from state $(\boldsymbol{s}, y)$ to $(\boldsymbol{r}, x)$ as given by (3.7). This indicates that the proposed CSSBN model accurately implements the function of a CSPBN.

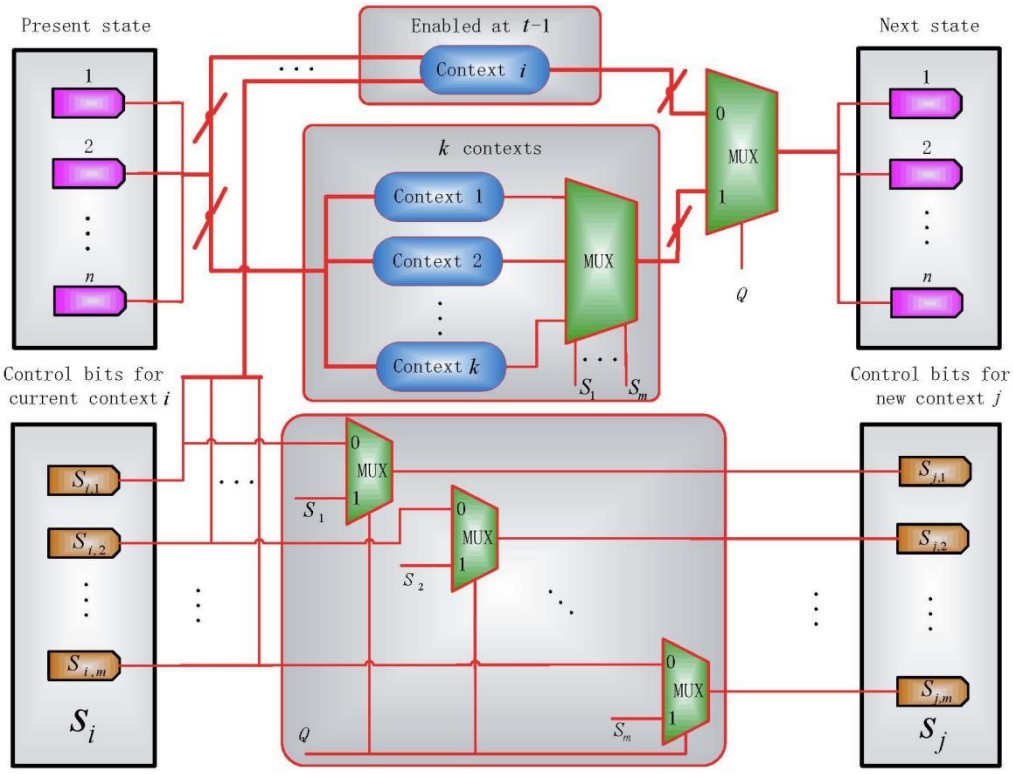


Fig. 3.1. A context-sensitive stochastic Boolean network (CSSBN) without perturbation (at time t). The multiplexer (MUX) with control sequences $S_1 \sim S_m$ probabilistically determine the selection of a network function for context i, while the multiplexer with control sequence Q determines whether a switch of contexts occurs. The selection probabilities of the new context are computed by the 2-to-1 MUX with the original and current context selection sequences as inputs and Q as the control sequence.

3.3. Context-Sensitive Stochastic Boolean Networks with Perturbation

In a PBN with random perturbation, a gene may change its state with a probability p during a state transition. Following [18], the effect of perturbation is considered to flip

a gene's state. Assume in an n-gene CSPBN, the current GAP at time t is given by $\boldsymbol{x} = (x_1, x_2, \cdots, x_n)$ and $\boldsymbol{\gamma}$ is the perturbation vector, the GAP at $t+1$, $\boldsymbol{x'}$, is given by:

$$\boldsymbol{x}' = \begin{cases} \boldsymbol{x} \oplus \boldsymbol{\gamma} & with\ a\ probability\ of\ 1-(1-p)^n \\ \boldsymbol{f}_j(\boldsymbol{x}) & with\ a\ probability\ of\ (1-p)^n \end{cases} \tag{3.9}$$

where $\oplus$ is the addition modulo 2 and $\boldsymbol{f}_j(\cdot)$ is the network function for the jth context at time t.

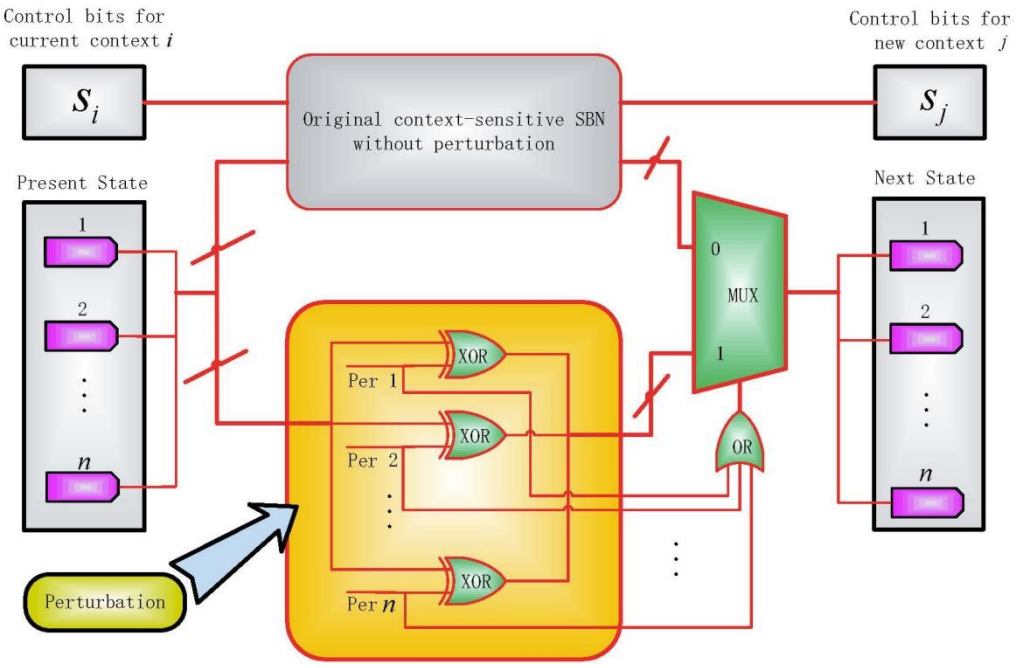


Fig. 3.2. A context-sensitive stochastic Boolean network (CSSBN) with perturbation. A perturbation network is implemented by the XOR logic of the perturbation vector and the present state. The probability that either a new context works or a perturbation occurs is given by the output sequence of an n-input OR gate, which in turn determines the selection of a new context (without perturbation) or a perturbed network by a bus (or multiple-bit) multiplexer (MUX).

To account for the effect of perturbation, a CSSBN with perturbation is constructed, as shown in **Fig. 3.2**. In this CSSBN, XOR gates are used to implement the addition modulo 2 of the perturbation vector and the present state, while an n-input OR gate is

used to compute the probability that a perturbation occurs. The output of the OR gate is then used as the control sequence of a bus (or multiple-bit) multiplexer to decide the selection of sequences with or without perturbation. If the output sequence of the OR gate contains all zeros, which means that there is no perturbation, then the next state is given by the predictor functions in the currently selected context in the original CSSBN without perturbation; otherwise, the next state is determined by the perturbation effect. A stochastic analysis of the function of the CSSBN with perturbation shows that the next state of the network is given by:

$$\boldsymbol{x}' = (\boldsymbol{x} \oplus \boldsymbol{\gamma}) \cdot (1 - (1-p)^n) + \boldsymbol{f}_j(\boldsymbol{x})(1-p)^n \tag{3.10}$$

which is equivalent to (3.9). This indicates that a CSPBN with perturbation can be accurately implemented by a CSSBN with perturbation.

3.4. Intervention in Context-Sensitive Stochastic Boolean Networks

In contrast to perturbation, gene intervention refers to the process of deliberately changing the states of some genes to guide a network into a desired state [18]. External stimuli are applied to a network to avoid undesirable states that might be associated with certain diseases. For an effective intervention, control policies are developed for different intervention strategies; as several genes may affect the state of the target gene, a single gene that is most influential on the network state is usually identified as the control gene, due to its simplistic biological implications [43]. The STM is then changed by an external intervention [43][77][78].

In a CSPBN, the context information is usually hidden and thus difficult to obtain in practice. However, gene expressions can readily be observed, so a GAP within all possible contexts is considered to be undesirable if the GAP is an undesirable state of the

network [79]. Hence, gene intervention refers to changing the state of a network as represented by a GAP.

For an n-gene network, a vector $\boldsymbol{s} = (s_1, s_2, \cdots, s_n)$ with $s_i \in \{0, 1\}$ for any $i \in \{1, 2, \cdots, n\}$, is defined as the control gene vector, that is, if s_i = 1, then gene i is selected as a control gene [80]. An intervention vector is defined as $\boldsymbol{u} = (u_1, u_2, \cdots, u_{2^n})$, $u_j \in \{0,1\}$ for any $j \in \{1,2, \cdots, 2^n\}$, where u_j = 1 (or 0) indicates a flipping (or remaining) of the state of a control gene at GAP *j*. If gene i is selected to be a control gene, for example, then s_i = 1. The expression level of the control gene i is then determined by its current state and the status of $\boldsymbol{u}$. If u_j = 1, i.e., $s_i u_j$ = 1, the state of control gene i is flipped by the external intervention when GAP *j* emerges; otherwise, the state of control gene i remains unchanged and thus the current state is preserved in GAP j [81][82]. An intervention vector can be obtained by various methods; in this chapter, a control policy using the SSD [83] is used to obtain $\boldsymbol{u}$.

The state of a network under intervention, i.e., a GAP, is determined by the control signal, the state prior to the intervention and the intervention vector. Let $\hat{\boldsymbol{x}}_t$ and $\boldsymbol{x}_t$ be the GAPs before and after intervention at time t; if the gth gene is the control gene (i.e., s_g = 1), the state of the network under intervention is given by [75]:

$$\boldsymbol{x}_t = \left(\hat{\boldsymbol{x}}_t \oplus \boldsymbol{s}_g\right) \cdot 1(u(\hat{\boldsymbol{x}}_t) = 1) + \hat{\boldsymbol{x}}_t \cdot 1(u(\hat{\boldsymbol{x}}_t) = 0) \tag{3.11}$$

where $\boldsymbol{s}_g$ is a vector of n bits with 0 at every bit except the gth bit. 1(·) is an indicator function: $1(u(\hat{\boldsymbol{x}}_t)) = 1$ if $u(\hat{\boldsymbol{x}}_t) = 1$ and $1(u(\hat{\boldsymbol{x}}_t)) = 0$ otherwise. When $u(\hat{\boldsymbol{x}}_t) = 0$, the GAP $\hat{\boldsymbol{x}}_t$ is maintained; otherwise, the GAP $\hat{\boldsymbol{x}}_t$ is an undesirable state, for which the control bit is to be toggled, thereby improving the occurring probability of a desirable state.

The CSSBN model under intervention is given in **Fig. 3.3**. If gene i is selected as the control gene, i.e., s_i = 1, the output of the AND gate is determined by the state of $\boldsymbol{u}$.

If state j is an undesirable state, i.e., $u_j = 1$, the state of the control gene i is toggled. This is implemented by an XOR gate: if $s_i u_j = 1$, the state of gene i, given by the output of an XOR gate, is flipped at the state, or GAP, j; otherwise, the state is not changed by the intervention. From this analysis, the state of a network after intervention is given by:

$$\boldsymbol{x}_t = \begin{cases} \hat{\boldsymbol{x}}_t \oplus \boldsymbol{s}_i, & \text{if } u(\hat{\boldsymbol{x}}_t) = 1 \\ \hat{\boldsymbol{x}}_t, & \text{otherwise} \end{cases} \tag{3.12}$$

which is equivalent to (3.11). This indicates that the CSSBN model in **Fig. 3.3** accurately implements the process of external gene intervention.

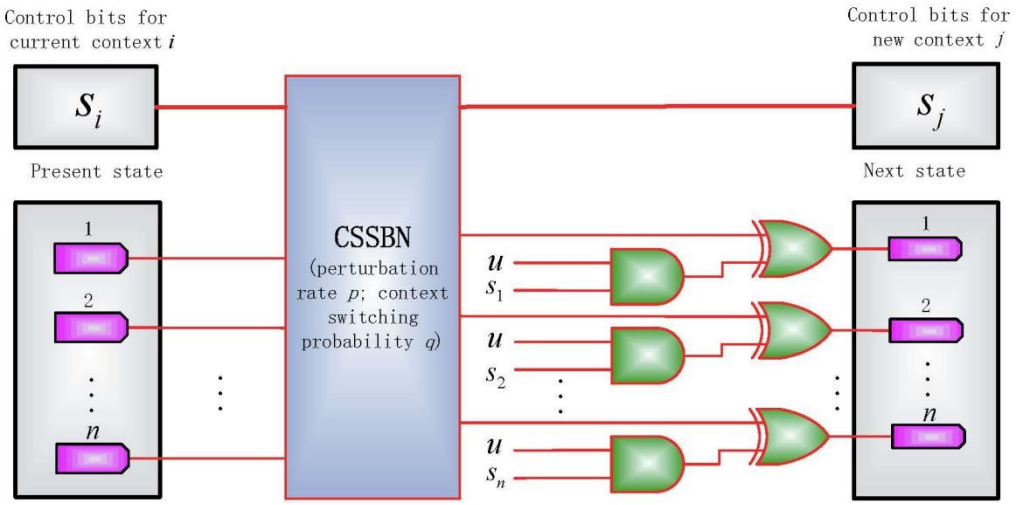


Fig. 3.3. A context-sensitive stochastic Boolean network (CSSBN) for external gene intervention. p: perturbation rate; q: context switching probability. s_i indicates whether gene i is selected as a control gene and u is the derived intervention vector for all states or gene activity profiles (GAPs) of a network.

3.5. State Transition Matrix and Steady State Distribution Analysis

As discussed previously, the state of a CSPBN can be represented as $\boldsymbol{S}$ = (*context*, *x*) by considering the selected context and GAP information. Using a CSSBN, the STM can

be obtained through the statistics encoded in the output sequences. Given an input state, each simulation of a CSSBN produces the transition probabilities from this input to all output states, i.e., the row in the STM for this input. For a CSPBN with n genes and k contexts, the CSSBN needs to be run for each of the $2^n k$ input states and an $O(n)$ number of sequences need to be generated for the control signals of the multiplexers.

As in [30], a factor, L, is used to account for the computational overhead required by using a longer stochastic sequence. Therefore, a complexity of $O(nLk2^n)$ results for computing the STM of a context-sensitive network. In a CSSBN, the required minimum sequence length is typically on a polynomial order of the numbers of genes and contexts, as shown in the simulation results in Table 3.1. Since the number of possible BNs, k, generally increases exponentially with the number of predictor functions for each gene, the complexity of using a CSSBN to compute the STM, i.e., $O(nLk2^n)$, is smaller than $O(nk^2 2^{2n})$ of an accurate analysis for a CSPBN [76]. As indicated by the shorter average run time in Table 3.1, this difference becomes significant for a network with a large number of gene states. In Table 3.1, the simulation results are also provided for CSSBNs with a single context, i.e., $k = 1$. In this case, the CSSBN degenerates into a deterministic BN, for which an analytical approach is faster, due to the use of random binary bit sequences in the stochastic approach. However, the CSSBN approach is faster in computing the STM for a gene network with a large number of states compared to the CSPBN approach.

In general, a $(2^n \cdot k) \times (2^n \cdot k)$ STM is required for an accurate CSPBN analysis [76] while a $2^n \times 2^n$ STM is needed by an approximate method [61]. As the number of genes increases in a network, a matrix-based analysis becomes infeasible due to a significant increase in the required computational resources. The analysis of the SSD is even more challenging for a CSPBN. For a CSSBN, however, the STM can be effectively computed. Furthermore, the SSD can be evaluated through an iterative simulation in the temporal domain (or the so-called time-frame expansion technique [30]). By this technique, an iterative structure of the CSSBN is used to simulate the temporal evolution of a GRN.

The required number of iterations is determined by the number of state transitions before reaching a steady state.

Table 3.1. Minimum sequence length and average run time in computing the state transition matrix (STM) for context-sensitive stochastic Boolean networks (CSSBNs).

n	k	Num	CSSBN (Norm 2 = 0.05)			CSSBN (Norm 2 = 0.03)			CSPBN [76]	
			L (bits)	Avg. time (s)	Standard deviation	L (bits)	Avg. time (s)	Standard deviation	Avg. time (s)	Standard deviation
2	1	4	40	1.48×10^{-3}	4.80×10^{-4}	100	1.81×10^{-3}	4.80×10^{-4}	4.90×10^{-4}	4.60×10^{-4}
2	4	16	900	5.53×10^{-3}	1.67×10^{-3}	2000	1.12×10^{-2}	3.60×10^{-4}	1.25×10^{-2}	2.51×10^{-3}
2	16	64	1000	0.64	0.42	6000	1.03	0.11	9.59×10^{-2}	9.01×10^{-3}
3	1	8	140	4.46×10^{-3}	1.32×10^{-3}	340	8.12×10^{-3}	9.40×10^{-4}	1.57×10^{-3}	4.50×10^{-4}
4	1	16	250	1.44×10^{-2}	2.87×10^{-3}	750	3.68×10^{-2}	2.68×10^{-3}	4.43×10^{-3}	2.70×10^{-4}
4	**16**	**256**	**8000**	**1.43**	**1.84×10^{-2}**	**20000**	**3.29**	**3.83×10^{-2}**	**1.66**	**0.10**
5	1	32	420	4.60×10^{-2}	2.61×10^{-3}	1200	0.12	3.14×10^{-3}	1.86×10^{-2}	2.14×10^{-3}
5	**32**	**1024**	**8000**	**10.74**	**1.35×10^{-2}**	**25000**	**35.54**	**4.34×10^{-2}**	**24.07**	**8.39×10^{-2}**
6	1	64	600	9.99×10^{-2}	4.58×10^{-3}	1700	0.34	1.97×10^{-3}	6.66×10^{-2}	7.05×10^{-3}
6	**64**	**4096**	**24000**	**74.51**	**1.21**	**60000**	**195.79**	**4.87**	**531.07**	**6.36**
7	1	32	1000	0.53	2.76×10^{-2}	2800	1.34	6.38×10^{-2}	0.26	7.19×10^{-3}
7	**16**	**2048**	**8000**	**18.89**	**0.43**	**25000**	**50.09**	**0.27**	**221.92**	**1.66**
8	1	256	1400	1.46	2.79×10^{-2}	3500	3.68	0.16	1.05	1.30×10^{-2}
8	**4**	**1024**	**4000**	**5.51**	**5.17×10^{-2}**	**12500**	**15.67**	**0.12**	**37.41**	**2.21**
9	1	512	2600	5.36	7.64×10^{-2}	8000	16.27	0.14	4.07	6.89×10^{-2}
9	**4**	**2048**	**8000**	**23.20**	**0.40**	**18000**	**45.92**	**0.79**	**72.44**	**2.62**
10	1	1024	4000	18.48	0.13	11000	49.24	0.37	17.51	0.13
10	**4**	**4096**	**10000**	**59.94**	**0.41**	**30000**	**162.69**	**1.13**	**619.31**	**13.41**

n: the number of genes; k: the number of contexts. Perturbation and context switch probabilities: $p = 0.1$ and $q = 0.8$ (for $k > 1$). L: the required minimum sequence length for a given accuracy measured by Norm 2. Num: the number of states for the CSSBN. The results that at least one case of the CSSBN study is faster than the context-sensitive probabilistic Boolean network (CSPBN), are highlighted in bold.

In a time-frame expanded CSSBN, random binary bit streams are generated for predictor function selection, gene perturbation and context switching probabilities. As in [30], non-Bernoulli sequences are used for encoding initial probabilities. These sequences then propagate through the CSSBNs and the statistics encoded in the output sequences are used to obtain the SSD. Compared to the use of Bernoulli sequences, as shown in [29], the use of the non-Bernoulli sequences produces more accurate results for a given sequence length or requires a shorter sequence length for a desired evaluation accuracy. As an alternative to an STM-based analysis, the stochastic simulation of CSSBNs provides flexibility in achieving a tunable accuracy-efficiency tradeoff by using stochastic sequences of different lengths. Hence, the proposed CSSBN approach is potentially useful in the analysis of large GRNs.

3.6. Results and Discussion

3.6.1. Simulation of a p53-Mdm2 Network

As a tumour suppressor gene, p53 plays an important role in preventing the development and progression of tumour cells [84][85]. In a p53 network, signaling pathways are triggered by external stimuli. For example, DNA damages activate pathways that involve the genes p53 and Mdm2 (**Fig. 3.4**) [31][86]. It has been shown that the expression level of the p53 protein is reversely related with that of the Mdm2 gene, which leads to an oscillatory behavior of the p53-Mdm2 network [31][86]. Various Boolean models have been developed to simulate the dynamics of a p53 network [87]-[89]. In this section, the two-gene PBN model developed in [30] for the p53-Mdm2 network is used to illustrate the applicability of context-sensitive stochastic Boolean networks (CSSBNs). This (instantaneous) PBN of the two genes p53 and Mdm2 consists of $V = \{x_1, x_2\}$ and the predictor function sets $F_1 = \{f_1^{(1)}, f_2^{(1)}, f_3^{(1)}, f_4^{(1)}\}$ and $F_2 = \{f_1^{(2)}, f_2^{(2)}, f_3^{(2)}, f_4^{(2)}\}$. The truth table for the state transitions of this PBN is given as Table 3.2 [30].

In Table 3.2, the present states of p53 and Mdm2 are indicated by x_1 and x_2 respectively. For a given predictor function, a logical 1 or 0 results as the next state of

each gene. Based on the K-map analysis [90], the predictor function can be implemented by a combination of logic gates. The probability that each function is selected is given in the bottom row. In [30], an SBN of the p53-Mdm2 network is proposed. Based on the general CSSBN model in **Fig. 3.1**, a CSSBN for the p53-Mdm2 network is shown in **Fig. 3.5**. In this model, a network function (or a BN) is selected at time $t-1$ for context i and at time t, the network remains at context i with a probability of q or switches to one of the k contexts ($k = 16$ for this network) with a probability of $1-q$. This context switching behavior is modeled by a 2-to-1 MUX for each gene and the switching probability q is encoded in the control bit sequence Q. At the same time, the selection probabilities of the new context are determined by the 2-to-1 multiplexers with the original and current context selection sequences as inputs and Q as the control sequence, as shown in the upper section of **Fig. 3.5**.

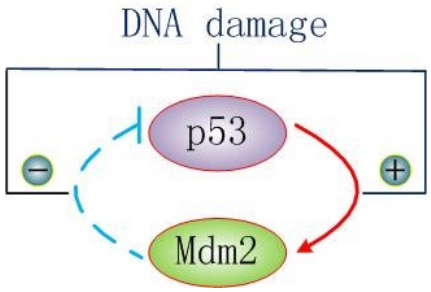


Fig. 3.4. The p53-Mdm2 network (as in [30] and adapted from [31]).

Table 3.2. Truth table of a probabilistic Boolean network (PBN) for the p53-Mdm2 network: x_1, x_2 are the present states of p53 and Mdm2.

x_1x_2	$f_1^{(1)}$	$f_2^{(1)}$	$f_3^{(1)}$	$f_4^{(1)}$	$f_1^{(2)}$	$f_2^{(2)}$	$f_3^{(2)}$	$f_4^{(2)}$
00	1	1	1	0	0	0	0	1
01	1	1	0	0	0	0	1	1
10	0	0	1	1	1	1	0	0
11	0	1	1	1	1	0	0	0
$c_j^{(i)}$	0.5	0.4	0.09	0.01	0.5	0.4	0.09	0.01

An internal entry indicates whether a logical 1 or 0 would result at the next state of a gene from a selected Boolean update function. $c_j^{(i)}$ indicates the transition probability by an update function $f_j^{(i)}$ [30].

The context-sensitive p53-Mdm2 network has 2 genes and 16 contexts. If both context and GAP are considered, there are $2^2 \times 16 = 64$ states for the p53-Mdm2 CSSBN.

Given the transition probability for each predictor function (in Table 3.2), the probability for selecting each context can readily be computed for independent functions, as shown in Table 3.3.

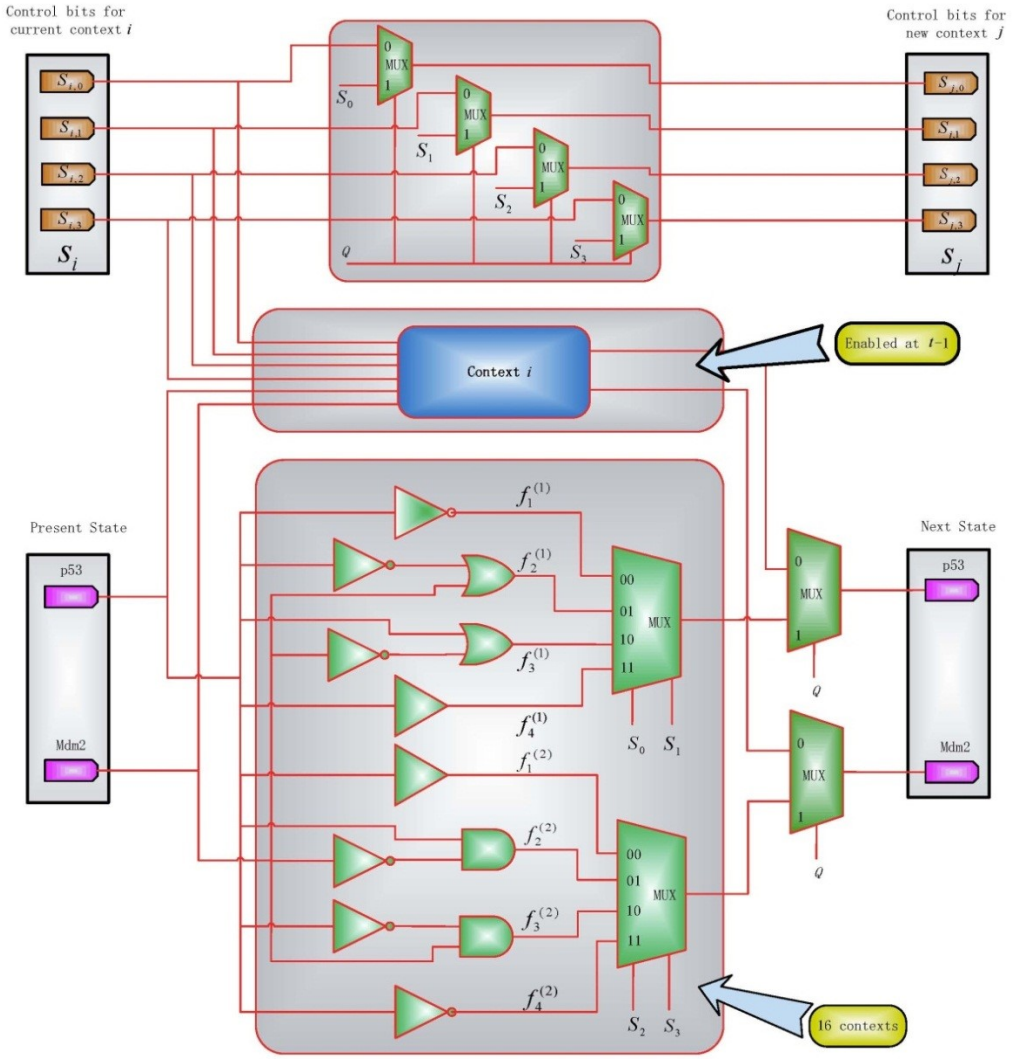


Fig. 3.5. A context-sensitive stochastic Boolean network (CSSBN) for the p53-Mdm2 network.

With random perturbation to the genes, a CSSBN with perturbation is constructed as shown in **Fig. 3.6**. For this two-gene network, a two-input multiplexer is used for each gene to probabilistically select a perturbed state or the original CSSBN state without perturbation.

Table 3.3. The network function and selection probability for each context in the p53-Mdm2 network.

Context	S2S3, S0S1	Combination	Selection probability	Context	S2S3, S0S1	combination	Selection probability
1	00,00	$f_1^{(2)} f_1^{(1)}$	0.25	9	10,00	$f_3^{(2)} f_1^{(1)}$	0.045
2	00,01	$f_1^{(2)} f_2^{(1)}$	0.2	10	10,01	$f_3^{(2)} f_2^{(1)}$	0.036
3	00,10	$f_1^{(2)} f_3^{(1)}$	0.045	11	10,10	$f_3^{(2)} f_3^{(1)}$	0.0081
4	00,11	$f_1^{(2)} f_4^{(1)}$	0.005	12	10,11	$f_3^{(2)} f_4^{(1)}$	0.0009
5	01,00	$f_2^{(2)} f_1^{(1)}$	0.2	13	11,00	$f_4^{(2)} f_1^{(1)}$	0.005
6	01,01	$f_2^{(2)} f_2^{(1)}$	0.16	14	11,01	$f_4^{(2)} f_2^{(1)}$	0.004
7	01,10	$f_2^{(2)} f_3^{(1)}$	0.036	15	11,10	$f_4^{(2)} f_3^{(1)}$	0.0009
8	01,11	$f_2^{(2)} f_4^{(1)}$	0.004	16	11,11	$f_4^{(2)} f_4^{(1)}$	0.0001

The control bits to select a predictor function in Fig. 3.6 are also listed. S0S1: the control bits for p53; S2S3: the control bits for Mdm2.

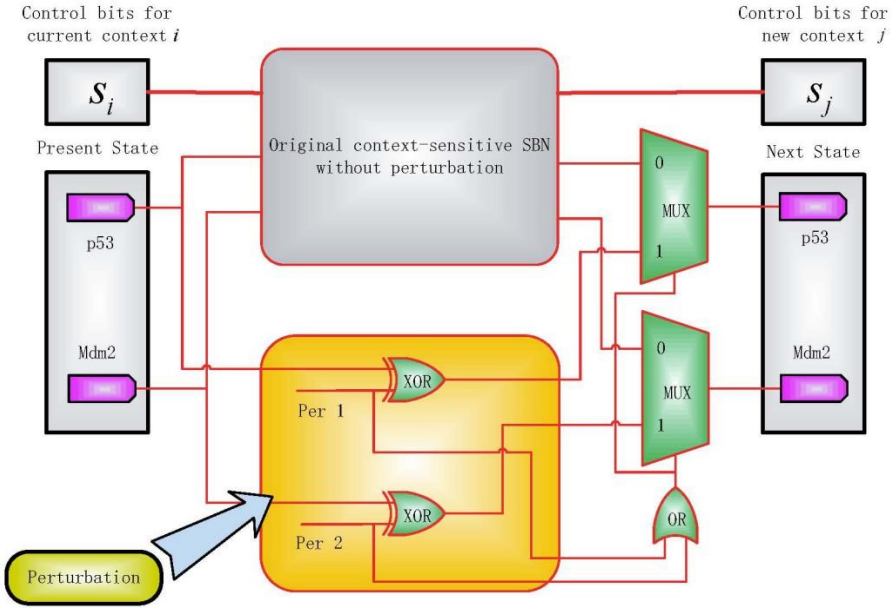


Fig. 3.6. A context-sensitive stochastic Boolean network (CSSBN) with perturbation for the p53-Mdm2 network.

The CSSBN is then used to obtain the STM for the p53-Mdm2 network. The norms $\|\cdot\|_1$, $\|\cdot\|_2$, and $\|\cdot\|_\infty$ are used to measure the differences of the STMs obtained for the CSSBN and CSPBN. Assume $\boldsymbol{A}_{CSSBN}$ and $\boldsymbol{A}_{CSPBN}$ are the obtained STMs for the CSSBN and CSPBN respectively. Let $\Delta\boldsymbol{A} = \boldsymbol{A}_{CSSBN} - \boldsymbol{A}_{CSPBN}$; the norms of the differences of the computed matrices ($\|\Delta\boldsymbol{A}\|_1$, $\|\Delta\boldsymbol{A}\|_2$ and $\|\Delta\boldsymbol{A}\|_\infty$) are then shown in Table 3.4 for different values of the switching probability q and perturbation rate p. The average run time is also shown for using the CSSBN.

Table 3.4. Differences in the state transition matrices (STMs) obtained using the context-sensitive stochastic Boolean network (CSSBN) with perturbation, compared to the results by using the analytical context-sensitive probabilistic Boolean network (CSPBN) approach in [76].

	L (bits)	$q = 1$ $p = 0$	$q = 0.99$ $p = 0$	$q = 0.5$ $p = 0$	$q = 0.8$ $p = 0.01$
$\|\Delta \boldsymbol{A}\|_1$	1k	0.1820	0.2067	0.2320	0.3469
	10k	0.0754	0.0754	0.0605	0.0890
	100k	0.0291	0.0239	0.0306	0.0210
$\|\Delta \boldsymbol{A}\|_2$	1k	0.0642	0.0754	0.0706	0.1012
	10k	0.0258	0.0259	0.0207	0.0297
	100k	0.0101	0.0091	0.0097	0.0080
$\|\Delta \boldsymbol{A}\|_\infty$	1k	0.0382	0.0413	0.0530	0.0888
	10k	0.0144	0.0145	0.0164	0.0249
	100k	0.0050	0.0055	0.0064	0.0074
Avg. time (s)	1k	6.01×10^{-2}	5.47×10^{-2}	5.34×10^{-2}	6.23×10^{-2}
	10k	0.32	0.32	0.39	0.32
	100k	2.88	2.85	2.85	2.92

The average run time of 20 simulations using the CSSBN is also shown for different perturbation rates, p, and context switching probabilities, q. L: sequence length.

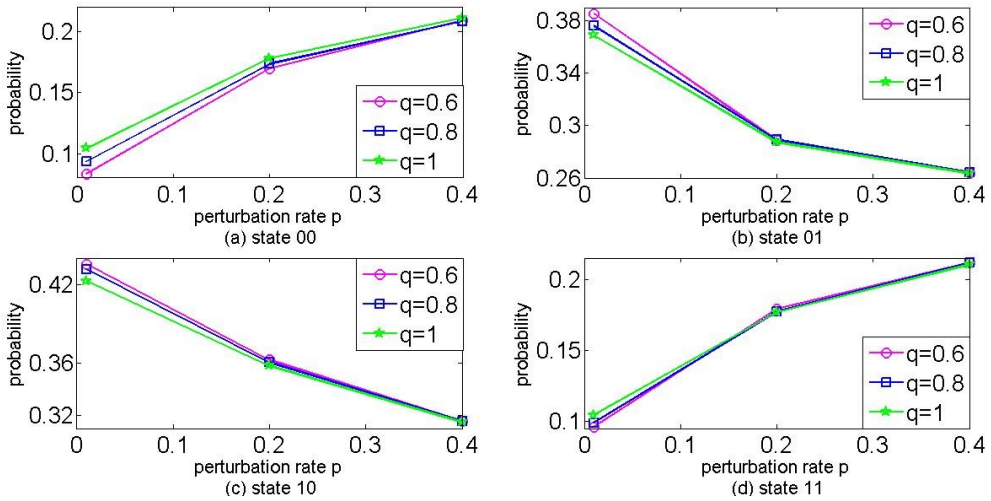


Fig. 3.7. The steady state distribution (SSD) of the p53-Mdm2 network for different perturbation rate, p, and context switching probability, q. The steady-state probabilities are shown for (a) state 00, (b) state 01, (c) state 10, and (d) state 11. (L = 500k bits)

As revealed in Table 3.4, the difference in the STMs computed using the CSSBN and an analytical CSPBN approach is significantly reduced by increasing the sequence

length L. However, the inaccuracies, due to the inherent stochastic fluctuations in stochastic computation, are generally small and thus negligible. Hence, the proposed CSSBN model can be used to compute the STM of a CSPBN.

A steady state analysis is further performed on the proposed CSSBN. For the p53-Mdm2 network, there are four states or GAPs. The probability of each GAP is given by the sum of the probabilities for all contexts. The simulation results for the four GAPs with respect to different p and q values are shown in **Fig. 3.7** (using a sequence length of 500k bits).

As can be seen in **Fig. 3.7**, while the SSD is determined by both the perturbation and switching probabilities, the perturbation rate has a greater effect on the final distribution of the steady state compared to the switching probability. The SSD changes drastically with the increase of the perturbation rate, whereas the effect of context switching is rather limited for a given perturbation rate. As $p, q \in [0, 1]$, several p and q pairs are chosen for further analysis. The difference in the SSDs obtained by using the time-frame expanded CSSBN technique, the approximate [61] and accurate analysis [76] are shown in Table 3.5.

As revealed in Table 3.5, the CSSBN approach can compute the SSD more accurately than the approximate analysis. In fact, the difference in the results between the CSSBN and the accurate analysis is negligible when reasonably long stochastic sequences are used. With the STM obtained for a CSSBN, an SSD is evaluated and the results are very close to those obtained using the CSPBN approach, as shown in **Fig. 3.8**. A further analysis shows that the relative error is limited to less than approximately 0.2% for the CSSBN approach.

It can be seen that the SSD can be evaluated by the CSSBN model faster compared to the accurate analytical approach. It has been shown that a PBN with perturbation evolves as an ergodic Markov chain [18]. For a larger perturbation probability p, there is an increased randomness in the network activities, thus the steady states of the network are more evenly distributed [76]. **Fig. 3.7** further shows that context switching has little

Table 3.5. Differences in steady state distributions (SSDs) computed using the context-sensitive stochastic Boolean network (CSSBN) model, compared to the results by using approximate [61] and accurate analysis [76].

			$p = 0.01$ $q = 0.9$	$p = 0.1$ $q = 0.8$	$p = 0.3$ $q = 0.9$
$\|\Delta \boldsymbol{SSD}_{AP-AC}\|_1$			0.0228	0.0214	0.0041
$\|\Delta \boldsymbol{SSD}_{CSSBN-AC}\|_1$	L (bits)	10k	0.0094	0.0099	0.0073
		50k	0.0061	0.0060	0.0057
		100k	0.0047	0.0056	0.0040
$\|\Delta \boldsymbol{SSD}_{AP-AC}\|_2$			0.0118	0.0118	0.0025
$\|\Delta \boldsymbol{SSD}_{CSSBN-AC}\|_2$	L (bits)	10k	0.0055	0.0057	0.0041
		50k	0.0036	0.0036	0.0030
		100k	0.0028	0.0031	0.0025
$\|\Delta \boldsymbol{SSD}_{AP-AC}\|_\infty$			0.0074	0.0081	0.0020
$\|\Delta \boldsymbol{SSD}_{CSSBN-AC}\|_\infty$	L (bits)	10k	0.0047	0.0045	0.0027
		50k	0.0030	0.0029	0.0021
		100k	0.0020	0.0022	0.0017

$\Delta \boldsymbol{SSD}_{AP-AC}$: the difference between the results obtained by using the approximate and accurate analysis. $\Delta \boldsymbol{SSD}_{CSSBN-AC}$: the difference between the results obtained by using the CSSBN approach and the accurate analysis. p: perturbation rate, q: context switching probability, L: sequence length.

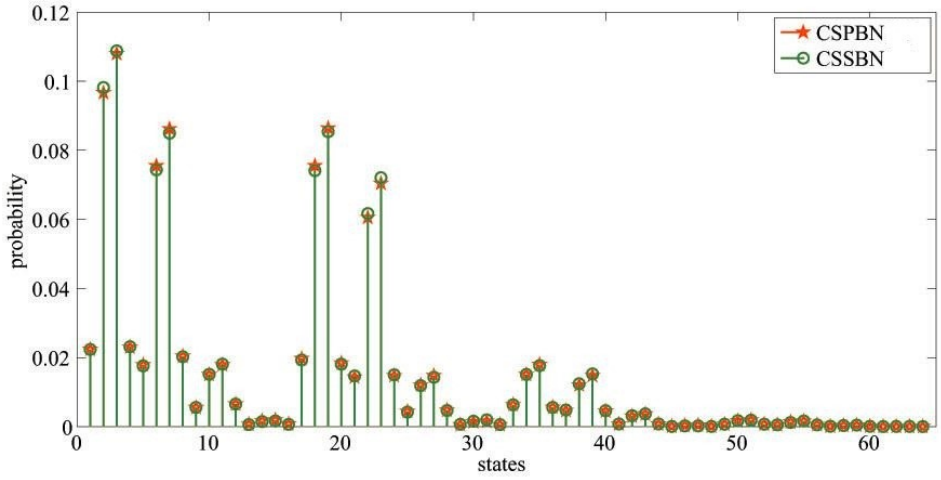


Fig. 3.8. Accuracy comparison of the proposed context-sensitive stochastic Boolean network (CSSBN) approach and the accurate analytical approach [76] for $p = 0.01$, $q = 0.9$ and $L = 10$k bits. The relative error is generally less than 0.2% for the CSSBN approach.

impact on the SSD of the network with perturbation. This is due to the fact that the context switching activity only affects the selection probability of a context, but not the predictor functions.

3.6.2. Experiments on a Glioma Network

A network in [44] obtained from a human glioma gene expression data set [91] is further used to illustrate the efficiency of the CSSBN model and the time-frame expansion technique. Based on this data set, a PBN was inferred by a method using the coefficients of determination (CODs) and an SSD analysis was performed in [63]. An approximate method was developed in [61] as an extension to estimate the SSD of a CSPBN with perturbation. The gene TOP2A was not considered in either study as it is an input gene with an in-degree of zero. In our study, the network setting is considered the same as in [61][63] with the gene TOP2A removed; this leads to a total of 2^{14} GAPs. **Fig. 3.9** shows a detailed structure of the considered glioma network with double (or single) - headed arrows indicating the bi (or uni) - directional relationships of gene pairs. The selection probabilities for the predictor functions are shown in Table 3.6 [61] and the Boolean functions for each gene are obtained through an analysis of the data in [63].

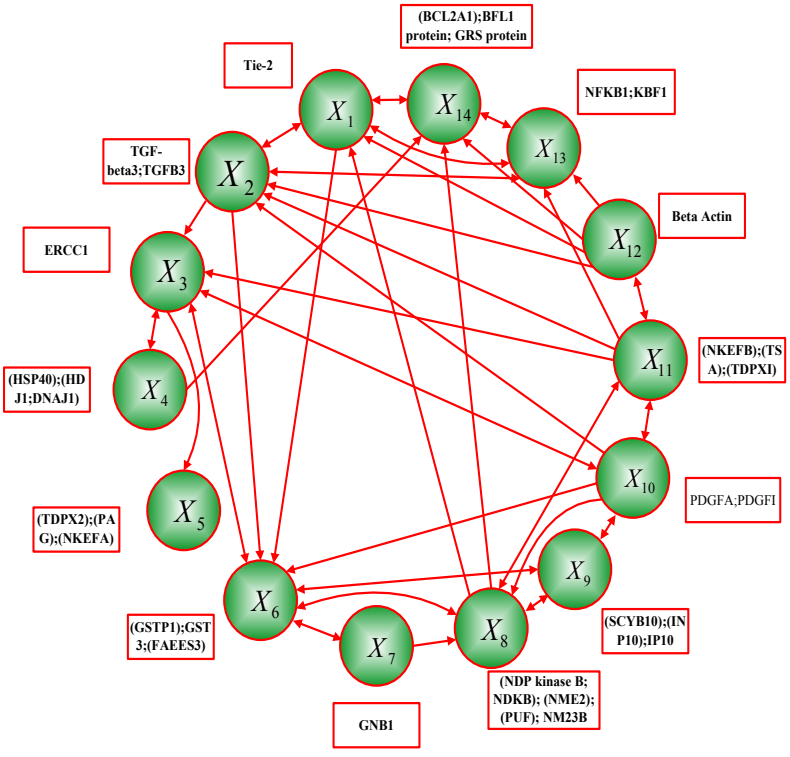


Fig. 3.9. A Glioma network (adapted from [61] and [63]).

In Table 3.6, each column for $\boldsymbol{f}_i$, $i \in \{1, 2, \cdots, 14\}$, contains the selection probabilities for the Boolean update functions of gene i, with the value in the jth row as the probability for $f_j^{(i)}$, thus the probabilities sum to 1 in each column. Table 3.6 also shows that six genes have only a single predictor function; only the state of gene 6 is determined by three predictor functions, while the states of the other genes are determined by two predictor functions.

With the network topology in Fig. 3.9 and the selection probabilities of update functions in Table 3.6, simulation results can be obtained for certain initial input signal probabilities and the perturbation and context-switching probabilities.

Table 3.6. Selection probabilities of the Boolean functions, $\boldsymbol{f}_i$, $i \in \{1, 2, \cdots, 14\}$, for each gene in the glioma network in Fig. 3.10 [61][63] .

$\boldsymbol{f}_1$	$\boldsymbol{f}_2$	$\boldsymbol{f}_3$	$\boldsymbol{f}_4$	$\boldsymbol{f}_5$	$\boldsymbol{f}_6$	$\boldsymbol{f}_7$
0.8560	0.2768	0.6759	1.0000	1.0000	0.0263	1.0000
0.1440	0.7232	0.3241			0.4983	
					0.4754	
$\boldsymbol{f}_8$	$\boldsymbol{f}_9$	$\boldsymbol{f}_{10}$	$\boldsymbol{f}_{11}$	$\boldsymbol{f}_{12}$	$\boldsymbol{f}_{13}$	$\boldsymbol{f}_{14}$
0.0857	0.5595	0.0751	0.8508	1.0000	0.8697	0.6004
0.9143	0.4405	0.9249	0.1492		0.1303	0.3996

3.6.2.1. Steady State Analysis

For the 14-gene glioma network, there are a total of 384 contexts, as can be determined from Table 3.6. It requires an STM with $2^{14} \times 384 = 6291456$ columns and rows for an accurate analysis. This makes it infeasible to estimate the steady states of a CSPBN using a matrix-based analysis. The approximate analysis in [61] would require the computation of an STM of the size $2^{14} \times 2^{14}$. Thus, it is difficult in general to use an analytical approach to evaluate a large network, due to the demanding computational resources required. However, a CSSBN model can be constructed for the glioma network; this CSSBN is based on the constituent SBN, as shown in **Fig. 3.10**. With the CSSBN, the SSD can be estimated using the aforementioned time-frame expansion technique with a greatly reduced complexity. The obtained SSD is then compared with that obtained from the approximate analysis in [61]. In this dissertation, a network is considered to

have reached a steady state if the discrepancy between two adjacent iterations is smaller than a required threshold ε (by (2.14)) or the number of simulation iterations has reached a maximum constant value.

The state or GAP of the glioma network can be represented by a binary vector as $(x_1,\ x_2,\ \cdots,\ x_{14}),\ x_i \in \{0,1\}$ for $i \in \{1,2,\ \cdots,14\}$, or its decimal index. For example, the state (01001100011011) can be represented as state 4892. The SSDs of the context-sensitive glioma network with all 16384 states, obtained using the CSSBN approach and the approximate analysis [61], are shown in **Fig. 3.11**.

The norms of the differences of SSDs, obtained by using CSSBN with different sequence lengths and the approximate method in [61], are shown in Table 3.7.

Table 3.7. Norms of the differences in the computed steady state distributions (SSDs) and average run time for the glioma network.

		$p = 0.01\ q = 0.9$			$p = 0.1\ q = 0.9$		
L (bits)		50k	200k	1M	50k	200k	1M
$\|\Delta \boldsymbol{SSD}_{AP-CSSBN}\|_1$		0.2314	0.1323	0.0861	0.4251	0.2104	0.0939
$\|\Delta \boldsymbol{SSD}_{AP-CSSBN}\|_2$		0.0055	0.0032	0.0023	0.0045	0.0023	0.0010
$\|\Delta \boldsymbol{SSD}_{AP-CSSBN}\|_\infty$		0.0013	8.01×10^{-4}	4.13×10^{-4}	3.05×10^{-4}	2.16×10^{-4}	7.85×10^{-5}
Avg. time (s)	CSSBN	10.11	311.55	1377.50	10.07	315.65	1380.70
	Approximate	21421.00			21417.00		

A maximum of 300 iterations are used for obtaining the SSD. p: perturbation rate; q: context switching probability; L: sequence length. $\Delta \boldsymbol{SSD}_{AP-CSSBN}$: the difference between the SSDs obtained by the approximate analysis [61] and the context-sensitive stochastic Boolean network (CSSBN) approach.

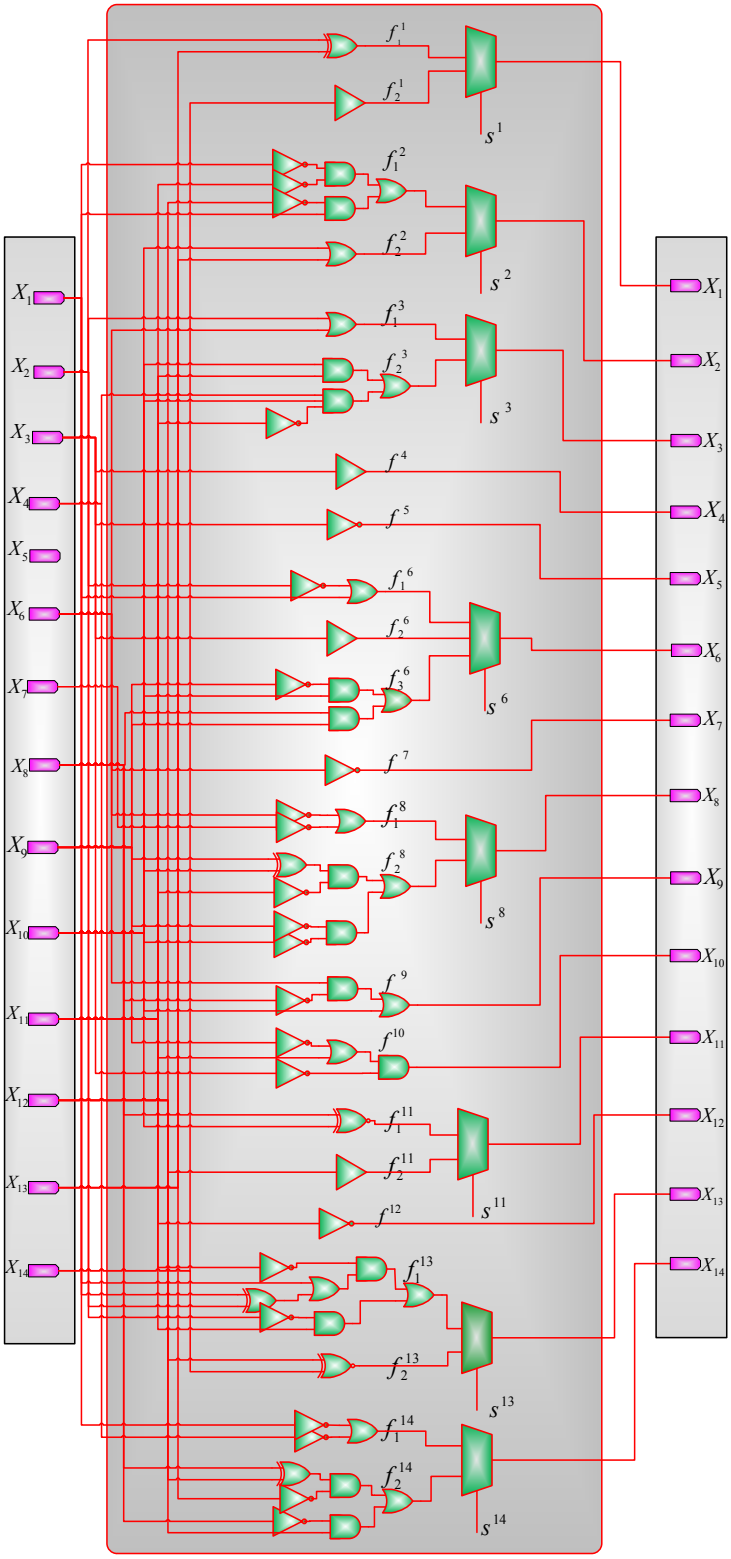


Fig. 3.10. A stochastic Boolean network (SBN) for the glioma network as a basis for the context-sensitive SBN (CSSBN) model with 384 contexts.

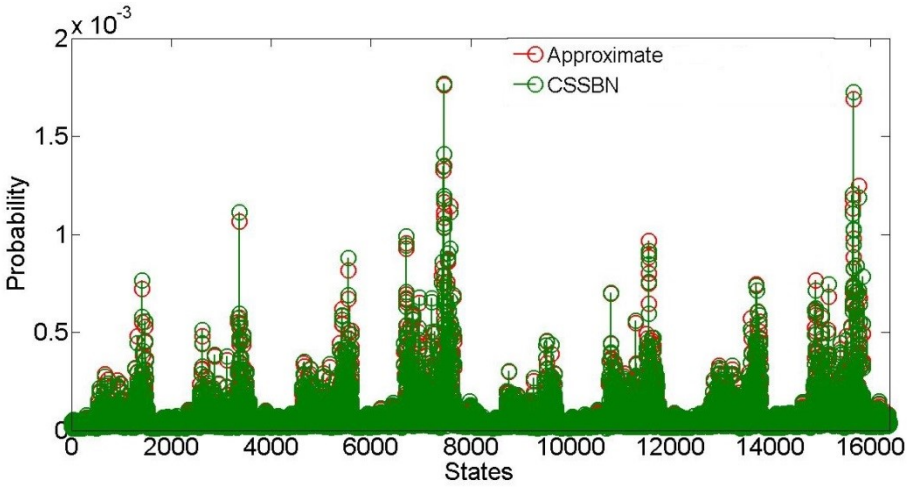


Fig. 3.11. The steady state distributions (SSDs) of the glioma network obtained using the context-sensitive stochastic Boolean network (CSSBN) time-frame expansion technique and the approximate analysis [61]. L = 800k bits.

As shown in Table 3.7, the CSSBN time-frame expansion technique effectively evaluates the SSD of the glioma network and produces results comparable to those obtained by the approximate analysis [61]. Evaluation accuracy is further improved by using longer stochastic sequences with yet a shorter runtime compared to the approximate method. Since it is difficult, if not impossible, to compute the STM or SSD of a large GRN by using an accurate or approximate analysis, the CSSBN time-frame expansion approach provides an alternative means to estimate the SSD of a large network with a tunable accuracy by using stochastic sequences of different lengths.

3.6.2.2. Intervention Analysis

In an *n*-gene network, if gene X_i is the target gene, the states of all genes can be divided into a set of desirable states, D, and a set of undesirable states, U, by the expression level of the target gene [83]. As can be seen in **Fig. 3.10**, gene X_{14}, the (BCL2A1);BFL1 protein;GRS protein is one of the most influential genes that interacts closely with others, thus it is chosen as the target gene. The desirable and undesirable states are then separated by the expression level of X_{14}. Assume that the inactivation of X_{14} is preferred; the cumulative probabilities of the desirable and undesirable states are

given by $\sum_{X_{14}=0}\pi_x$ and $\sum_{X_{14}=1}\pi_y$, respectively, where π_x and π_y are elements in the desirable and undesirable SSDs respectively.

As discussed previously, a GRN can be intervened by applying external stimuli to minimize the likelihood of being in an undesirable state. Various methods have been proposed for deriving control vectors for an effective intervention [75][83]; in this study the SSD algorithm proposed in [83] is used to determine an intervention vector. For simplicity, 16 most significant contexts that account for a total selection probability of 57.27% are chosen from the total 384 contexts of the glioma network for an intervention analysis. For these 16 contexts, the expression levels of gene X_5 and X_7 have no effect on the states of other genes, so these two genes are removed; this yields a simplified 12-gene glioma network. The cumulative distributions of the desirable steady states of the context-sensitive 12-gene glioma network are shown in Table 3.8 for using a different gene as the control gene.

Table 3.8. Cumulative distributions of the desirable states with a different gene selected as the control gene for the simplified 12-gene context-sensitive glioma network, with perturbation rate p = 0.001 and context switching probability q = 0.99.

gene	X_1	X_2	X_3	X_4	X_6	X_8	X_9	X_{10}	X_{11}	X_{12}	X_{13}
$\sum_{x_{14}=0}\pi_x$	70.91%	62.93%	52.61%	45.90%	48.98%	54.78%	53.76%	61.61%	57.23%	56.99%	61.78%

The cumulative probability of the desirable states with no intervention is 45.97%. Sequence length L = 800k bits.

When a different gene is selected as the control gene, the effect of intervention varies with respect to improving the probabilities of desirable states in the SSD. For the glioma network, as revealed in Table 3.8, gene X_1, Tie-2, is the most effective for maximizing the percentage of the desirable states. The cumulative distribution of the desirable states is increased from 45.97% to 70.91% by an intervention using Tie-2 as the control gene. Also revealed in the table is that an intervention via gene X_4, (HSP40);(HDJ1;DNAJ1), has nearly no impact on the distribution of the desirable states, as the percentage of the desirable states has not been changed much by the intervention

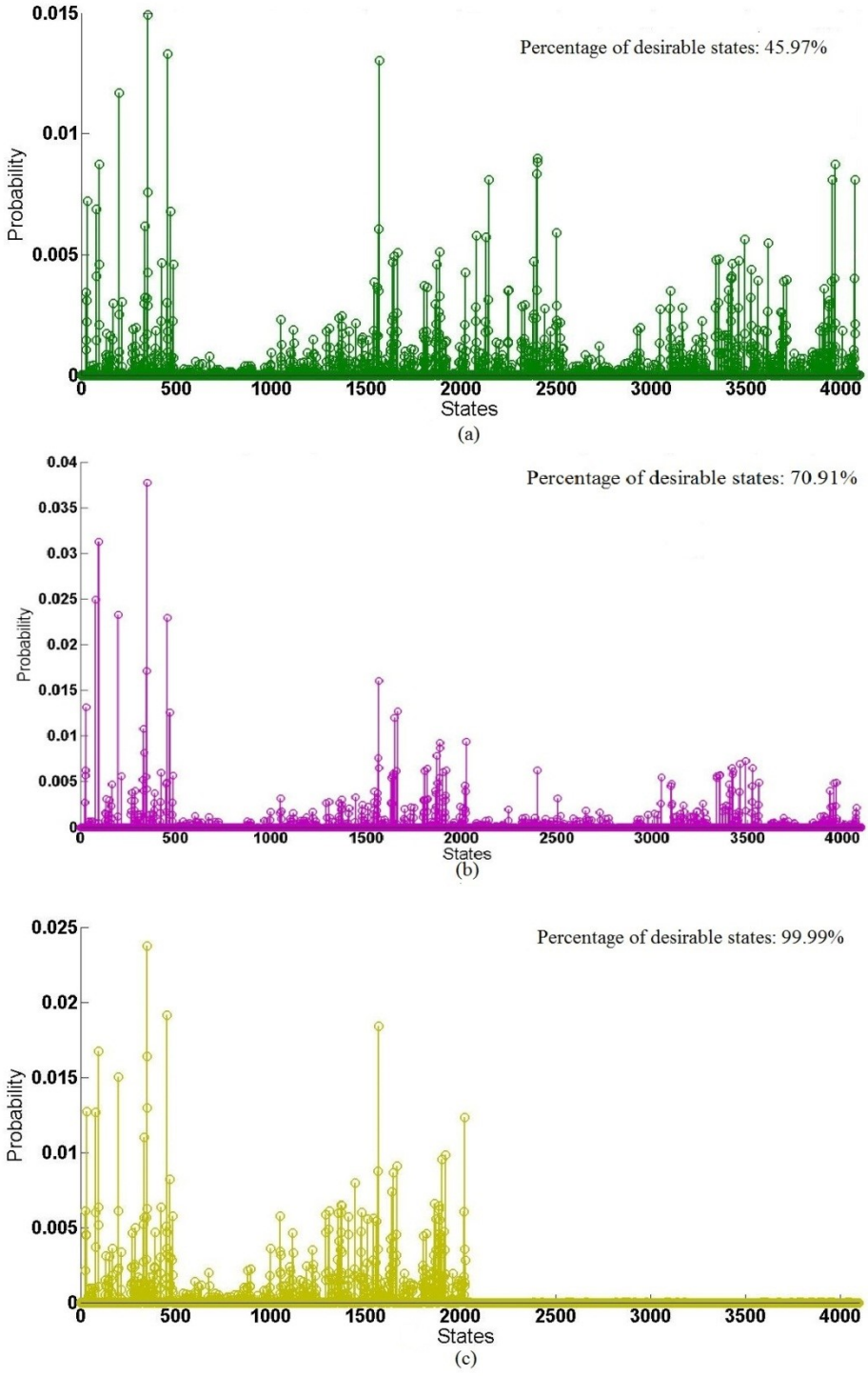


Fig. 3.12. Steady state distributions (SSDs) of the 12-gene context sensitive glioma network (p =0.001, q = 0.99). Obtained by: (a) no intervention, (b) an indirect intervention via gene X_1, Tie-2, and (c) a direct intervention via the target gene X_{14}, the (BCL2A1);BFL1 protein;GRS protein.

(i.e., 45.90% vs. 45.97%). A modest improvement in the desirable state distribution is obtained by an intervention with another gene as the control gene. For any target gene, this process can be applied to find the most significant gene that maximizes the probabilities of desirable states.

The effects of intervention can also be seen in **Fig. 3.12**, where three different scenarios are considered: (a) no intervention, (b) an indirect intervention via gene X_1, i.e., Tie-2, and (c) a direct intervention via the target gene X_{14}, the (BCL2A1);BFL1 protein;GRS protein. The cumulative probabilities of the desirable states are 45.97% for no intervention, 70.91% and 99.99% for the two intervention strategies. As revealed in these results, a direct intervention of the target gene is perhaps optimal for avoiding the undesirable states and almost certainly taking the network into a desirable state. However, when an intervention on the target gene is not possible, an intervention through Tie-2 is the most effective means to maximize the probability of the down-regulation of the target gene X_{14}.

3.7. Summary

CSSBNs are proposed as a fast approach to modeling the effects of gene perturbation and intervention in GRNs. In a CSSBN, the STM can be computed with a complexity of $O(nLk2^n)$, where n is the number of genes in a CSPBN, k is the number of contexts and L is a factor determined by the stochastic sequence length. This result is an improvement compared to the previous result of $O(nk^2 2^{2n})$ for an accurate analysis. The use of non-Bernoulli stochastic sequences further increases computational efficiency and allows for a tunable tradeoff between accuracy and efficiency. A steady state analysis using a time-frame expansion technique has shown a significant speedup and produced more accurate results than an approximate analysis in the computation of the SSD.

CSSBNs are constructed for the analysis of gene perturbation in a p53-Mdm2 network and gene intervention in a glioma network. It has been shown that SSD changes drastically with the increase of the perturbation rate, whereas the effect of context switching is rather limited for a given perturbation rate. Therefore, random gene

perturbation may have a greater effect on the final distribution of the steady state compared to context switching activities. By predicting the SSD, the CSSBN approach can further evaluate the effectiveness of external gene intervention. A case study of the glioma network shows that the CSSBNs are useful in modeling the effects of gene perturbation and intervention in a complex context-sensitive GRN. This will eventually help drug discovery for an effective drug intervention therapy.

Chapter 4

Stochastic Multiple-Valued Gene Networks

For complex biological networks, the Boolean simplification may incur an accuracy loss, whereas an approach using multiple-valued variables introduces an increased level of granularity [48]-[51]. For examples, three states of the protein p53 are considered in [88][89] and multiple-valued gene nodes are analyzed in a T-helper network [51]. Moreover, deterministic multiple-valued networks are analyzed in [92]. Multiple-valued networks have also been studied in chemical reactions [93] and cognitive sciences [94]. To further exploit the simplicity of logical models, stochastic multiple-valued networks (SMNs) are proposed for modeling the effects of noise and gene perturbation in a gene regulatory network (GRN). An SMN enables a fast simulation of a probabilistic multiple-valued network (or PMN), as an extension of a probabilistic Boolean network (PBN). The results in this chapter have been published in [95][96].

The novelty of this chapter is as follows:

- Stochastic multiple-valued logic is generalized from Boolean logic to consider gene states that are not limited to binary values.
- A stochastic multiple-valued network model, referred to as SMNs, is developed for modeling the effect of noise and gene perturbation in a GRN. The SMNs provide a finer granularity in the modeling of GRNs.
- SMNs are constructed to investigate the dynamics of a p53-Mdm2 network and a WNT5A network with ternary gene states.

4.1. Probabilistic Multiple-Valued Networks

A multiple-valued network of n genes is defined by $G(V,F)$, with a node set $V=\{x_1,x_2,\cdots,x_n\}$ and a list of sets of predictor functions $F=(F_1,F_2,\cdots,F_n)$ [18]. If the state of gene i is quantized into k levels, then $x_i\in\{0,1,\cdots,k-1\}$ for $i\in\{1,2,\cdots,n\}$. For $k=2$, a network is referred to as a PBN, where V is a set of

binary-valued nodes; for $k = 3$, it is considered as a ternary network [33]. For a k-valued network of n genes, hence, there are a total of k^n network states or gene activity profiles (GAPs). A GAP is also given as a decimal index. For a multiple-valued network of n genes, a GAP is indexed by:

$$d = \sum_{i=1}^{n} x_i(t) \cdot k^{i-1} + 1, \tag{4.1}$$

where x_i is the state of the ith gene, $i \in \{1, 2, \cdots, n\}$, and k indicates the discretization level.

For gene i ($i \in \{1, 2, \cdots, n\}$), the set of predictor functions is given by $F_i = \{f_1^{(i)}, f_2^{(i)}, \cdots, f_{l(i)}^{(i)}\}$, with each predictor function $f_{j(i)}^{(i)}: \{0, 1, \cdots, k-1\}^n \to \{0, 1, \cdots, k-1\}$, where $l(i)$ is the number of possible predictor functions for gene i and $l(i)$ is usually a small number [61][62]. Due to the stochastic behavior, the next state of gene i is determined by all of its predictor functions in F_i, i.e., $f_1^{(i)}$, $f_2^{(i)}$, ... , $f_{l(i)}^{(i)}$ with probabilities $c_1^{(i)}$, $c_2^{(i)}$, $\cdots$, $c_{l(i)}^{(i)}$. If the predictor functions are independent, there are $N = \prod_{i=1}^{n} l(i)$ possible realizations of the network, each of which is referred to as a context. Assume that the jth context is represented as $\boldsymbol{f}^j = (f_{j(1)}^{(1)}, f_{j(2)}^{(2)}, \cdots, f_{j(n)}^{(n)})$, where each $f_{j(i)}^{(i)}: \{0, 1, \cdots, k-1\}^n \to \{0, 1, \cdots, k-1\}$, for $1 \le j(i) \le l(i)$, is a predictor function of gene i; the next state of a gene is determined by both the present state and the selected context.

A multiple-valued network can be modeled by a Markov chain [33], so the next state of gene i, x_i ($x_i \in \{0, 1, \cdots, k-1\}$ (in a k-valued network) is given by:

$$x_i^{(t+1)} = \begin{cases} 0 & with\ C_i^0(\boldsymbol{S}^{(t)}) = Pr\left(x_i^{(t+1)} = 0 \middle| \boldsymbol{S}^{(t)}\right) \\ 1 & with\ C_i^1(\boldsymbol{S}^{(t)}) = Pr\left(x_i^{(t+1)} = 1 \middle| \boldsymbol{S}^{(t)}\right) \\ & \vdots \\ k-1 & with\ C_i^{k-1}(\boldsymbol{S}^{(t)}) = Pr\left(x_i^{(t+1)} = k-1 \middle| \boldsymbol{S}^{(t)}\right) \end{cases}. \tag{4.2}$$

where $C_i^0(\boldsymbol{S}^{(t)}) + C_i^1(\boldsymbol{S}^{(t)}) + \cdots + C_i^{k-1}(\boldsymbol{S}^{(t)}) = 1$. Thus, the transition probability from the network state (or GAP) $\boldsymbol{S}^{(t)}$ at time t to $\boldsymbol{S}^{(t+1)}$ at $t+1$ is given by:

$$Pr(\boldsymbol{S}^{(t)} \to \boldsymbol{S}^{(t+1)}) = \prod_{i=1}^{n} C_i^{x_i^{(t+1)}}. \tag{4.3}$$

Using the decimal indices of GAPs by (4.1), the state transition of a ternary network is described by the STM as follows:

$$\boldsymbol{A} = \begin{bmatrix} Pr(1|1) & Pr(2|1) & \cdots & \cdots & Pr(3^n|1) \\ Pr(1|2) & Pr(2|2) & \cdots & \cdots & Pr(3^n|2) \\ \cdots & \cdots & \cdots & \cdots & \cdots \\ \cdots & \cdots & \cdots & \cdots & \cdots \\ Pr(1|3^n) & Pr(2|3^n) & \cdots & \cdots & Pr(3^n|3^n) \end{bmatrix}. \tag{4.4}$$

In $\boldsymbol{A}$, each entry indicates the conditional probability that the network transitions from a present state into a next state. For N realizations of the network, $\boldsymbol{A}$ can be obtained as $\boldsymbol{A} = \sum_{j=1}^{N} P_j \boldsymbol{A}_j$, where P_j ($P_j = \prod_{i=1}^{n} c_{j(i)}^{(i)}$) is the probability that the jth realization of the network emerges and $\boldsymbol{A}_j$ is the STM resulting from the jth realization [18]. Hence, the STM can be derived for a multiple-valued network with a complexity of $O(nNk^{2n})$, where N is the number of possible realizations of the network and k is the quantization level of the gene state.

External stimuli cause random gene perturbation that makes the dynamics of a network an ergodic Markov chain [19]. In an ergodic Markov chain, all states are communicated and thus a steady state distribution (SSD) exists in a network. Since a perturbed gene has $k-1$ possible states, there are $(k-1)^{n_0}$ states for n_0 perturbed genes ($n_0 \in \{1, 2, \cdots, n\}$); hence, each of the perturbed states in $\boldsymbol{S}^{(t+1)}$ is selected with a probability of $[1/(k-1)]^{n_0}$. The event that no gene is perturbed, occurs with a probability of $(1-p)^n$. Hence, $\boldsymbol{S}^{(t+1)}$ is determined by the selected context if no perturbation exists, i.e. $Pr\{\boldsymbol{S}^{(t)} \to \boldsymbol{S}^{(t+1)}\} = \prod_{l=1}^{n} C_l^{x_l^{(t+1)}}$. If n_0 genes are perturbed, $\boldsymbol{S}^{(t)} \to \boldsymbol{S}^{(t+1)}$ occurs with probability $p^{n_0} \cdot (1-p)^{n-n_0} \cdot [1/(k-1)]^{n_0}$. Following [33],

the state transition probability from $\boldsymbol{S}^{(t)}$ to $\boldsymbol{S}^{(t+1)}$ in a perturbed k-valued network is given by:

$$Pr\{\boldsymbol{S}^{(t)} \to \boldsymbol{S}^{(t+1)}\} = \left(\prod_{i=1}^{n} C_i^{x_i^{(t+1)}}\right) \cdot (1-p)^n + p^{n_0} \cdot (1-p)^{n-n_0} \cdot p_0^{n_0} \cdot 1[\boldsymbol{S}^{(t)} \neq \boldsymbol{S}^{(t+1)}], \tag{4.5}$$

with

$$n_0 = \sum_{i=1}^{n} 1(x_i^{(t)} \neq x_i^{(t+1)}), \tag{4.6}$$

$$p_0 = 1/(k-1), \tag{4.7}$$

where p is the perturbation rate, n_0 is the number of perturbed genes, p_0 is the probability that a gene will change to a new state if perturbed, and $1(\cdot)$ is an indicator function: $1[\boldsymbol{S}^{(t)} \neq \boldsymbol{S}^{(t+1)}] = 1$ if $\boldsymbol{S}^{(t)} \neq \boldsymbol{S}^{(t+1)}$ and $1[\boldsymbol{S}^{(t)} \neq \boldsymbol{S}^{(t+1)}] = 0$ otherwise. Using (4.5), a perturbed STM or perturbation matrix [30][61] can be obtained for further analysis of the SSD.

When gene expressions are discretized by multiple-valued variables, they are not only affected by the presence of activating or repressing proteins, but also by the absence of a protein [97]. PMNs have been studied in [50] and [33], respectively, for providing insights into the long run behaviors of a network with noise. For a k-valued network of n genes with N network functions, a $k^n \times k^n$ matrix is required for an accurate analysis of the SSD, resulting in a complexity of O(nNk^{2n}) in the computation of the STM. This also requires a memory usage in the order of at least O(k^{2n}). Since the size of an STM increases exponentially with the number of genes, the analysis of a large network with a higher quantization level presents even a greater challenge. This prevents the use of an accurate analysis in the evaluation of large networks. For a network with an increased number of genes, a Markov chain Monte Carlo (MCMC) method is often used

to estimate the SSD of a PBN [44] and its multiple-valued extension, PMNs [33]. An MCMC simulation is considered to produce an accurate result when a sufficient number of simulations are performed to produce a stable output; however, this number is usually required to be very large, due to the slow convergence of the MCMC method [45], thus incurring a very long simulation time.

4.2. Stochastic Multiple-Valued Logic

To reduce the inaccuracy incurred in a binary analysis, multiple-valued logic has been considered to generalize the stochastic analysis [95]. Stochastic computation is also applicable to the probabilistic analysis of multiple-valued signals. For a k-valued signal, the probability of each value is given in a vector $P = [p_{k-1}, p_{k-2}, \cdots, p_1, p_0]$, with $\sum_{i=0}^{k-1} p_i = 1$. This probability vector can be encoded into a multiple-valued stochastic sequence. An example is shown in **Fig. 4.1** for a ternary signal.

$$\text{“0121012102”} \qquad \text{for} \qquad \begin{cases} P(0) = 0.3 \\ P(1) = 0.4 \\ P(2) = 0.3 \end{cases}$$

Fig. 4.1 The stochastic encoding of a ternary signal *A*. $A \in \{0, 1, 2\}$; $L = 10$ bits.

Multiple-valued logic includes the buffer, inverter, MIN (minimum), MAX (maximum) and rotator, some of which are defined as follows [92]:

(1) A multiple-valued buffer:

$$\mathrm{BUF}(A) = A,$$

(2) A multiple-valued inverter:

$$\mathrm{INV}(A) = (k-1) - A,$$

(3) A multiple-valued rotator $\emptyset$:

$$\emptyset(A) = \begin{cases} A+1 & A \neq k-1 \\ 0 & A = k-1 \end{cases},$$

The following new logic operators are further defined:

(4) A multiple-valued equal or larger (EL) operator:

$$EL(A \geq a) = MAX(A, a),$$

(5) A multiple-valued equal or smaller (ES) operator:

$$ES(A \leq a) = MIN(A, a).$$

Several ternary stochastic processing elements are shown in **Fig. 4.2**, including a buffer, an inverter, an EL operator, an ES operator, a MIN, a MAX, a rotate gate and a 4-to-1 multiplexer.

For the ternary MIN logic, if the two inputs are independent with probabilities A = [0.3 0.4 0.3] and B = [0.5 0.4 0.1], the output probabilities are expected to be $p(2) = p_A(2) \cdot p_B(2) = 0.3 \times 0.1 = 0.03$, $p(0) = p_A(0) + p_B(0) - p_A(0) \cdot p_B(0) = 0.65$ and $p(1) = 1 - p(0) - p(2) = 0.32$. This function can be implemented by the ternary MIN gate, as shown in **Fig. 4.2(e)**, using stochastic sequences. For a sequence length of 10,000 bits, the output sequence is expected to have approximately 6,500 0's, 3,200 1's and 300 2's. For the ternary rotate logic, if the input's signal probability is given by A = [0.3 0.4 0.3], the output's signal probability is expected to be $p(0) = p_A(2) = 0.3$, $p(1) = p_A(0) = 0.3$ and $p(2) = p_A(1) = 0.4$. This function can be implemented by the ternary rotate gate with the use of stochastic sequences (**Fig. 4.2(h)**).

For the 4-to-1 multiplexer logic in **Fig. 4.2(i)**, its output is determined by its control bits $'ef'$. It takes the value of input a for $ef = 00$, b for $ef = 01$, c for $ef = 10$, or d for $ef = 11$. Similarly, a stochastic multiplexer takes one of the inputs as its output according to the distributions of control bits (i.e., 00 01 10 11). Thus, the selection probabilities are encoded in the random sequences of the control bits. As in stochastic computation, SMNs employ random streams of multiple values to encode probabilities

and computation is performed by stochastic logic. However, these numbers are not deterministic but probabilistic, due to inherent stochastic fluctuations. For stochastic Boolean networks (SBNs), it has been shown that the effect of the fluctuation can be significantly reduced through the use of non-Bernoulli sequences as initial input probabilities [30]. Here, stochastic sequences of random permutations of fixed numbers of the multiple values, hereafter referred to as randomly permuted sequences, are used for encoding initial input probabilities. The use of randomly permuted sequences reduces the amount of stochastic fluctuations in a network. It will be shown that the effect of fluctuation is negligible when a reasonable sequence length is used in the simulation.

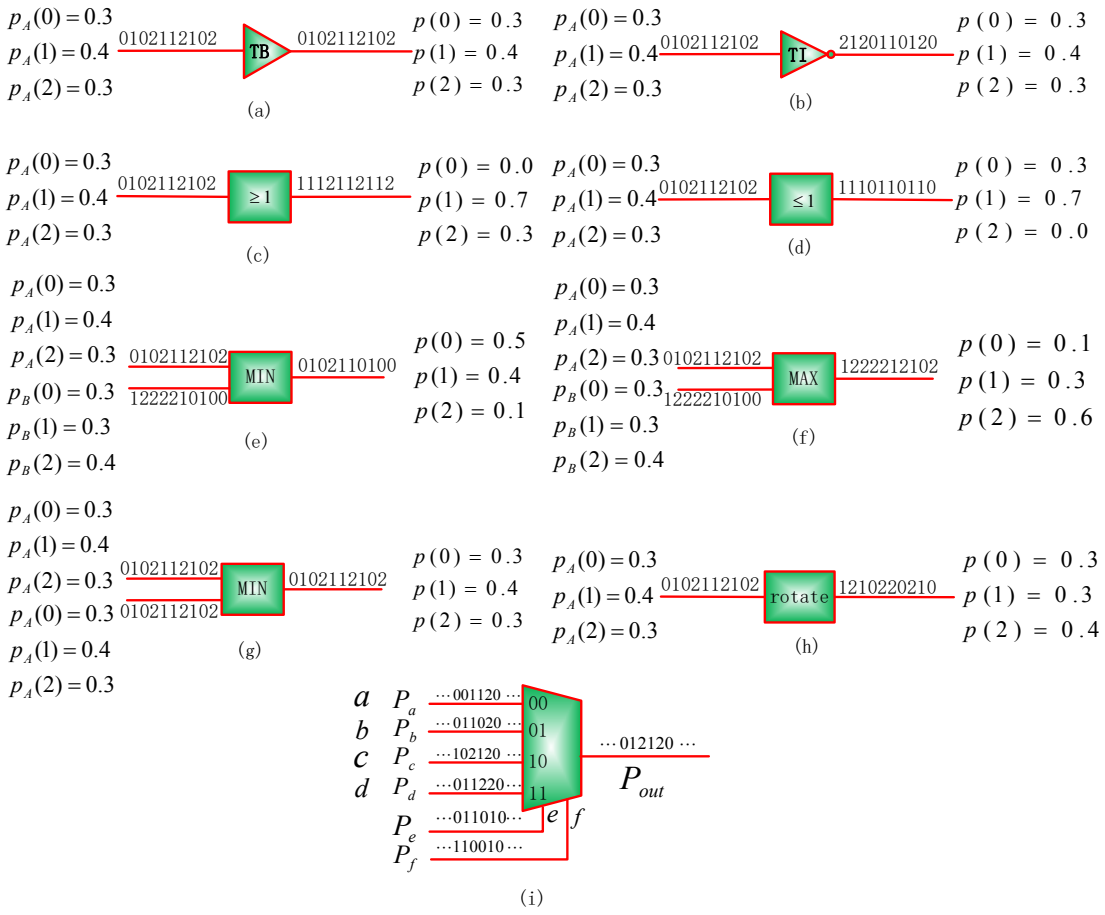


Fig. 4.2. Stochastic logic: (a) a ternary buffer (TB); (b) a ternary inverter (TI); (c) an equal or larger (EL) operator; (d) an equal or smaller (ES) operator; (e) a ternary minimum (MIN) with independent inputs; (f) a ternary maximum (MAX) with independent inputs; (g) a ternary MIN with totally dependent inputs; (h) a ternary rotate gate; (i) a 4-to-1 multiplexer. A probabilistic computation is performed through stochastic logic operations by encoding signal probabilities into random sequences.

4.3. Stochastic Multiple-Valued Networks without Perturbation

In the general case that multiple quantization levels are considered, an SMN can be constructed to model a multiple-valued gene network. As discussed previously, the next state of a gene is determined by the present state of its input genes and a set of predictor functions according to their occurring probabilities. In an SMN, these probabilities are represented by randomly permuted multiple-valued sequences and the selection of the predictor functions is implemented by a multiple-input multiplexer with properly generated control sequences. A structure of the SMN for a single gene is shown in **Fig. 4.3**.

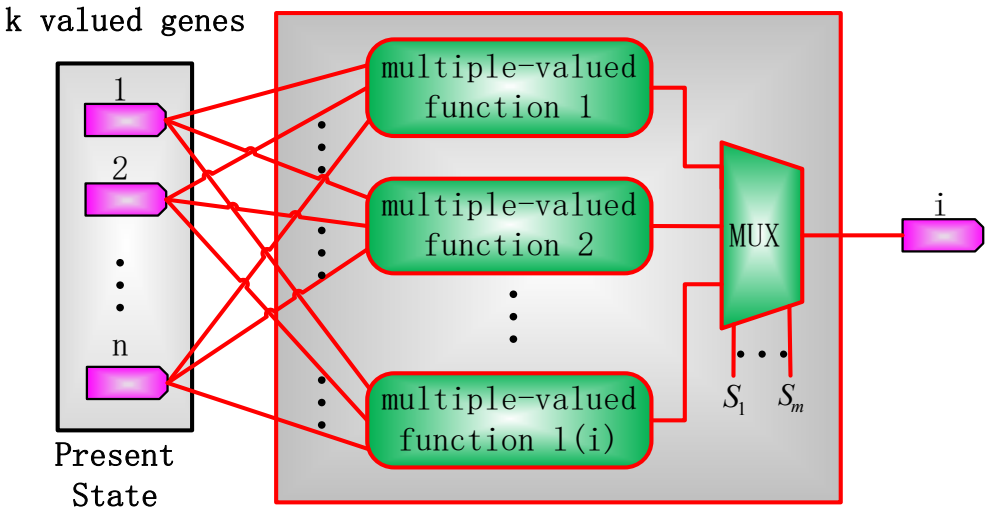


Fig. 4.3. A stochastic multiple-valued network (SMN) without perturbation (for a single gene *i*). The control sequences $S_1 \sim S_m$ of the multiplexer (MUX) probabilistically determine the selection of the multiple-valued functions.

If the next state of gene i is determined by $l(i)$ predictor functions, the number of control bits of the multiplexer is given by $\lceil log_2(l(i)) \rceil$. By a multiplexer with control bits $S_1 \sim S_m$, a function is selected in the jth BN for gene i with probability $c_{j(i)}^{(i)}$. Assume that a network transfers from state $\boldsymbol{S}^{(t)}$ to $\boldsymbol{S}^{(t+1)}$ in a context (or network function), then the transition probability for $\boldsymbol{S}^{(t)} \rightarrow \boldsymbol{S}^{(t+1)}$ is given by the probability of selecting this context. This indicates that when all the genes are considered, the SMN model in **Fig. 4.3** accurately implements the function of (4.3).

4.4. Stochastic Multiple-Valued Networks with Perturbation

Under external stimuli, a gene's state can be perturbed by a small chance during a transition [19]. In a k-valued network of genes, a perturbation flag vector $\boldsymbol{\gamma}$ is used to indicate whether a gene is to be perturbed. Assume that the network goes from state $\boldsymbol{S}^{(t)}$ to $\boldsymbol{S}^{(t+1)}$ under perturbation. If each gene is to be perturbed with a probability p, the probability that the next state is totally determined by a network function (i.e., no perturbation occurs) is $(1-p)^n$. When a perturbation occurs, the state of the perturbed gene transitions to a different state: this new state is determined by the present state and the value in the perturbation flag vector $\boldsymbol{\gamma}$. Without the loss of generality, a set of transition rules can be determined, as shown in Table 4.1 for a ternary network. The set of rules in Table 4.1 can be implemented by sum and modulo operations; for $\boldsymbol{S}^{(t)} = (0,0,0,1,1,1,2,2,2)$ and $\boldsymbol{\gamma} = (0,1,2,0,1,2,0,1,2)$, as an example, the next state is given by $\boldsymbol{S}^{(t+1)} = modulo\left(\left(\boldsymbol{S}^{(t)} + \boldsymbol{\gamma}\right), 3\right) = (0,1,2,1,2,0,2,0,1)$. Hence, the perturbation in a ternary network can be implemented by the sum and modulo operations. For a network of higher levels, similar operations can be implemented for the perturbation (although not discussed in detail), while for a Boolean network, this operation is simplified to an XOR gate.

Table 4.1. State transition rules for a gene in a ternary network under perturbation.

Current State (x)	perturbation (γ)	Next State ($x' = modulo(x+\gamma, 3)$)
0	1	1
	2	2
1	1	2
	2	0
2	1	0
	2	1

For an SMN, therefore, if $\boldsymbol{S}^{(t)} = (x_1, x_2, \cdots, x_n)$ is the GAP or state of the network at time t; the next state $\boldsymbol{S}^{(t+1)}$ is given by:

$$\boldsymbol{S}^{(t+1)} = \begin{cases} modulo(sum(\boldsymbol{S}^{(t)}, \boldsymbol{\gamma}), k) & with\ 1-(1-p)^n, \\ \boldsymbol{f}_j(x_1, x_2, \cdots, x_n) & with\ (1-p)^n, \end{cases} \tag{4.8}$$

where p is the perturbation rate for each gene and $\boldsymbol{f}_j(\cdot)$ is the jth realization of the network at time t. (4.8) indicates that no perturbation occurs, i.e., $\gamma_i = 0$ for any $i \in \{1, 2, \cdots, n\}$, with a probability of $(1-p)^n$. In this case, the next state $\boldsymbol{S}^{(t+1)}$ is determined by the selected context (or the network function). If gene i is perturbed, γ_i in $\boldsymbol{\gamma}$ is assigned to be m $(m \neq 0)$ with a probability of $1/(k-1)$; the gene's state x_i is then changed from j to m $(m \neq j)$ with a probability of $1/(k-1)$ [33]. This state transition under perturbation is then implemented by the function of $modulo(sum(\boldsymbol{S}^{(t)}, \boldsymbol{\gamma}), k)$. In a network of n genes, if n_0 genes are to be perturbed, this indicates that the perturbation flag vector $\boldsymbol{\gamma}$ contains n_0 non-zero values and $n - n_0$ zeros. For each zero, the current state of the corresponding gene remains, as shown in the aforementioned example. For the n_0 non-zero values, a different set of values leads to a different next state of the perturbed genes. For random gene perturbation, each set occurs with a probability of $[1/(k-1)]^{n_0}$, so the network transition from the present state to a particular next state, i.e., $\boldsymbol{S}^{(t)} \rightarrow \boldsymbol{S}^{(t+1)}$, occurs with a probability of $p^{n_0} \cdot (1-p)^{n-n_0} \cdot [1/(k-1)]^{n_0}$. Since a perturbed state is considered to be different from the present state, i.e. $\boldsymbol{S}^{(t+1)} \neq \boldsymbol{S}^{(t)}$, under perturbation, the probability of the state transition of (4.8) is given by (4.5).

To account for the perturbation effect, a modified SMN is shown in **Fig. 4.4**. The probability that the multiple-valued network is left without perturbation or that a perturbation takes effect, is determined by the output of an n-input MAX gate. In the SMN in **Fig. 4.4**, gene perturbation is considered as follows. Since a random gene perturbation probabilistically changes the state of a gene, the modules of sum and modulo k operations are used to implement the perturbation function (of the perturbation vector and the genes' current states). The jth perturbation vector, Per_j, is a sequence consisting of a number of i's, $i = 0, 1, \cdots, (k-1)$; for instance, if an L-bit sequence Per_i is used

to indicate the perturbation rate p in a ternary network and let $M = L \cdot p$, then there are $L - M$ 0's, $M/2$ 1's and $M/2$ 2's in the sequence.

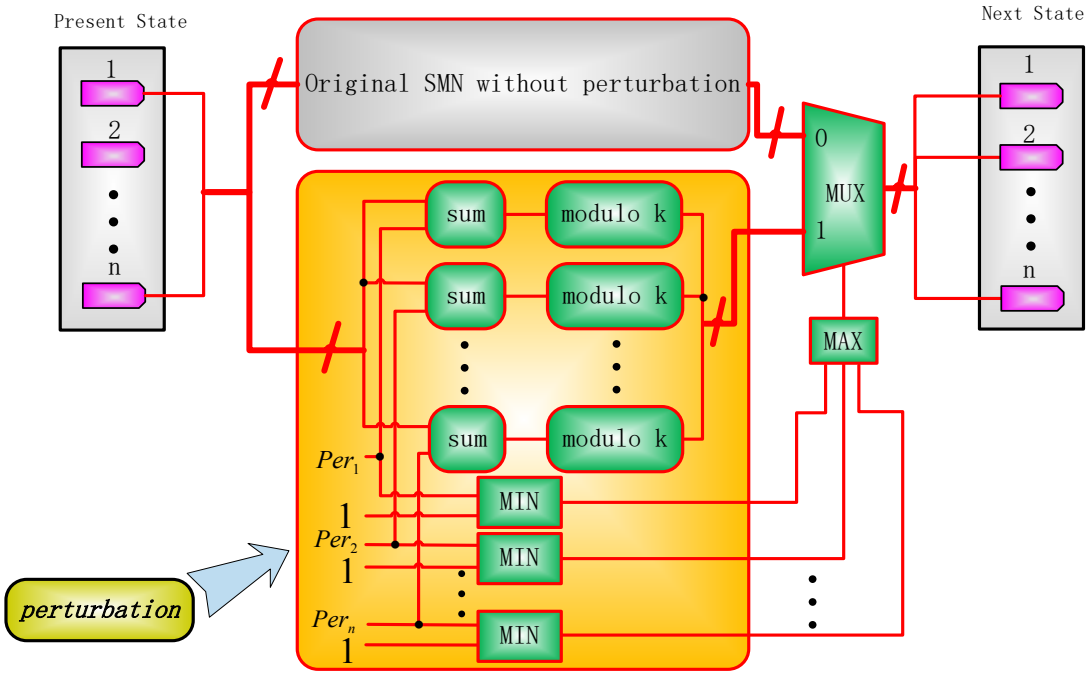


Fig. 4.4. A stochastic multiple-valued network (SMN) structure with perturbation. Gene perturbation is implemented by the sum and modulo k functions of the perturbation vector and the present state.

This indicates that if a gene at state i is perturbed, the new state can be any $j(j \neq i)$ with an equal probability of $1/(k-1)$. Hence, if n_0 genes are perturbed, a perturbed state is chosen with a probability of $[1/(k-1)]^{n_0}$. The probability that either an original multiple-valued function works or a perturbation occurs (by (4.8)) is implemented by the output sequence of an n-input MAX gate. This sequence is then used as the control sequence of a bus (or multiple-bit) multiplexer. If no perturbation occurs, the perturbation vectors ($'Per_1'$, $'Per_2'$, ⋯, $'Per_n'$ in **Fig. 4.4**) consist of all zeros, and thus the output sequence of the MAX gate will contain all zeros. The next state is subsequently given by the original SMN without perturbation; otherwise, the next state is determined by the perturbation probability encoded in the output sequence of the MAX gate. From this analysis, it can be seen that the SMN model computes the function of (4.8)

and thus computes the transition probability of (4.5). This indicates that it accurately implements a probabilistic multiple-valued network with perturbation.

4.5. State Transition Matrix and Steady State Distribution Analysis

In the simulation of an SMN, each input combination results in output sequences that contain information about the transition probability from this input to every output (or next state). For a deterministic input (i.e. the present state), the proportions of the numbers of the next states encoded in the output sequences return the statistics as the transition probabilities in a row in the STM. Hence, all the transition probabilities for this input can be generated in a single run. For a PMN with k levels and n genes, the SMN needs to be run for each of the k^n input states and an $O(n)$ number of sequences need to be generated for the control signals of the multiplexers.

The accuracy in the computed state transition probabilities is determined by the length of the stochastic sequences. Since longer sequences are usually required in a larger network for achieving an evaluation accuracy, a factor, L, is used here to account for the computational overhead required by using a longer stochastic sequence. For a k-valued network of n genes, a complexity of O(nLk^n) results for computing the STM at a desired accuracy. As shown in the simulation results in Table 4.2, the required minimum sequence length increases slower with the numbers of genes than the number of possible networks, N, which generally increases exponentially with the number of genes in a network. Therefore, the complexity of using an SMN to compute the STM, i.e., O(nLk^n), is smaller than O(nNk^{2n}) of an accurate analysis [33]. This difference becomes significant for a large network, as indicated by the shorter average run time in Table 4.2.

In a network with a large number of genes, a matrix-based analysis becomes cumbersome because of the size of the involved matrices. A steady state analysis becomes even more challenging. Using an SMN, however, the STM can be accurately computed. The SSD can be evaluated by using the so-called time-frame expansion

technique [30]. By this technique, the temporal evolution of a multiple-valued network is simulated using a spatially iterative structure of the SMN. The number of iterations is determined by the number of state transitions before reaching a steady state.

Table 4.2. Minimum sequence length and average run time required in computing the state transition matrix (STM) of ternary stochastic multiple-valued networks (SMNs), compared to those obtained by a Markov chain analysis [33].

n	N	Num of states	SMN (Norm 2 = 0.04)			SMN (Norm 2 = 0.02)			Markov chain analysis [33]	
			L	Avg. time (s)	Std.	L	Avg. time (s)	Std.	Avg. time (s)	Std.
2	4	9	0.26k	2.76×10^{-3}	4.40×10^{-4}	1k	6.66×10^{-3}	9.70×10^{-4}	3.44×10^{-3}	1.90×10^{-4}
3	8	27	0.9k	2.32×10^{-2}	1.31×10^{-3}	3.6k	8.36×10^{-2}	4.19×10^{-3}	2.35×10^{-2}	1.68×10^{-3}
4	16	81	1.6k	0.15	5.06×10^{-3}	6k	0.53	4.16×10^{-3}	0.20	6.44×10^{-3}
5	32	243	2.7k	0.93	3.97×10^{-2}	10k	3.38	1.38×10^{-2}	1.61	1.22×10^{-2}
6	64	729	4.2k	4.67	2.43×10^{-2}	17k	15.07	3.06×10^{-2}	12.39	0.36
7	128	2187	6k	24.27	0.17	24k	74.78	0.62	119.08	5.92
8	256	6561	10k	136.93	3.03	34k	438.99	11.95	1003.70	37.18

n: the number of genes; N: possible number of networks; perturbation rate $p = 0.1$; L: required minimum sequence length. Accuracy of the SMN approach is measured by norm 2 between the STMs obtained by the Markov chain analysis and the SMN approach. An equal number of predictor functions are randomly generated for each gene.

A general multiple-valued network (with any k) can be analyzed by the time-frame expanded SMN approach. The simulation results in Table 4.3 reveal that, while the SMN approach takes longer time than a Markov chain analysis [33] for small networks, it becomes faster in the analysis of large networks. Although the evaluation accuracy slightly decreases with the increase of the discretization level, k, a better accuracy is obtained when longer stochastic sequences are used.

Table 4.3. Average run time (Avg. time) in computing the steady state distribution (SSD) of stochastic multiple-valued networks (SMNs), compared to the use of a Markov chain analysis [33]. Accuracy of the SMN approach is measured by norm 2 between the SSDs, i.e. $\|\Delta \boldsymbol{SSD}\|_2$, obtained by the Markov chain analysis [33] and the SMN approach.

n	N	k	Time-frame expanded SMN approach									Markov
			L	Avg. time (s)	$\|\Delta \boldsymbol{SSD}\|_2$	L	Avg. time (s)	$\|\Delta \boldsymbol{SSD}\|_2$	L	Avg. time (s)	$\|\Delta \boldsymbol{SSD}\|_2$	[33] Avg.
2	4	3	1k	1.79×10^{-2}	0.0255	10k	0.19	0.0096	100k	1.39	0.0058	1.63×10^{-2}
		4		1.74×10^{-2}	0.0282		0.18	0.0105		1.37	0.0072	2.05×10^{-2}
		5		2.01×10^{-2}	0.0306		0.16	0.0125		1.63	0.0091	1.65×10^{-2}
		6		1.99×10^{-2}	0.0312		0.18	0.0130		1.54	0.0106	4.96×10^{-2}
3	8	3	1k	3.31×10^{-2}	0.0283	10k	0.27	0.0099	100k	2.62	0.0049	4.37×10^{-2}
		4		2.97×10^{-2}	0.0303		0.27	0.0104		2.49	0.0054	9.40×10^{-2}
		5		2.81×10^{-2}	0.0310		0.28	0.0116		2.54	0.0064	0.38
		6		2.94×10^{-2}	0.0323		0.27	0.0121		2.46	0.0073	1.05
4	16	3	1k	3.31×10^{-2}	0.0293	10k	0.37	0.0098	100k	3.21	0.0045	0.24
		4		3.95×10^{-2}	0.0302		0.36	0.0100		2.82	0.0046	1.63
		5		3.36×10^{-2}	0.0312		0.31	0.0107		2.84	0.0054	8.52
		6		3.26×10^{-2}	0.0317		0.31	0.0110		2.86	0.0059	35.33

The steady state is considered to have been reached in 30 iterations. n: the number of genes; N: possible number of networks; k: the discretization level of a gene network (all genes are assumed to have the same discretization level); perturbation rate $p = 0.1$; L: sequence length used in the simulation.

The memory usage of the SMN approach is further investigated and compared to that of the Markov chain analysis [33]. As shown in the simulation results in Table 4.4, the Markov chain analysis requires less memory than SMN for small networks with a low

quantization level, k, whereas the required memory outgrows that of the SMN approach in the analysis of a larger network with a larger k. In fact, the required memory by the Markov chain analysis increases exponentially with the number of genes and depends heavily on k, because of the increased size of transition matrices in an analysis. On the other hand, the memory required by the time-frame expanded SMN approach is mainly determined by the sequence length (L) and number of genes (n), while the quantization level (k) has little impact. It is also shown that the Markov chain analysis incurs a significantly longer run time than the SMN approach in the analysis of networks with larger n and k. Although a constant sequence length (30k) is used for the simulation results in Table 4.4, further simulations using different sequence lengths show a similar pattern. As reported in the Results and Discussion section, these features make the SMN approach faster than an analytical Markov chain approach while producing more accurate results compared to the Monte Carlo method in the analysis of large gene networks.

Table 4.4. Required memory usage in computing the steady state distribution (SSD) of multiple-valued networks by the Markov chain analysis [33] and time-frame expanded stochastic multiple-valued network (SMN) approach, given by Mem_{MCA} (M Byte) and Mem_{SMN} (M Byte) respectively. Average (Avg.) time (s) is also provided.

	$k=3$	$k=4$	$k=5$	$k=6$	$k=3$	$k=4$	$k=5$	$k=6$
	$n=2$				$n=3$			
Mem_{MCA}	2.60×10^{-3}	6.80×10^{-3}	1.56×10^{-2}	3.14×10^{-2}	1.83×10^{-2}	9.68×10^{-2}	0.36	1.08
Avg. time	1.63×10^{-2}	2.05×10^{-2}	1.64×10^{-2}	4.96×10^{-2}	4.37×10^{-2}	9.39×10^{-2}	0.38	1.05
Mem_{SMN}	5.98	5.98	5.98	5.98	8.73	8.73	8.73	8.73
Avg. time	0.47	0.45	0.46	0.47	0.69	0.72	0.70	0.69
	$n=4$				$n=5$			
Mem_{MCA}	0.15	1.51	8.97	38.49	1.36	24.04	223.64	1384.30
Avg. time	0.24	1.63	8.52	35.33	1.40	21.86	200.21	1272.67
Mem_{SMN}	11.47	11.47	11.48	11.48	14.22	14.23	14.24	14.27
Avg. time	0.93	0.93	0.92	0.93	1.17	1.18	1.20	1.29

50 iterations are performed in each simulation. n: the number of genes; k: the discretization level of a gene network; perturbation rate $p = 0.1$; sequence length: L = 30k. Two predictor functions are randomly generated for each gene.

4.6. Results and Discussion

4.6.1. A Multiple-Valued p53-Mdm2 Network

p53 is a tumour suppressor gene that plays an important role in preventing the development and progression of tumour cells [84][85]. External stimuli such as DNA damages can activate signaling pathways that involve the genes p53 and Mdm2. The dynamic behavior of a p53 network has been studied by using various Boolean models [87][88] and an oscillatory behavior of the p53 and Mdm2 has been observed [31][86].

A four-node network has been analyzed in [88][89] with "DNA damage" as one of the nodes. As DNA damage (such as double strand breaks) is one of the major factors that activate the p53 network [31][85][86], a three-node network that excludes the DNA damage as an external factor, as shown in **Fig. 4.5**, is considered in this section for an application of the SMN model. Let X_1 denotes the gene p53, cytoplasmic p53 and nucleic p53 (i.e. protein p53), and X_2 and X_3 denote the cytoplasmic Mdm2 and nucleic Mdm2, respectively. As protein p53 activates the cytoplasmic Mdm2 that has a positive effect on the nuclear Mdm2. Thus, protein p53 promotes nucleic Mdm2 indirectly through the cytoplasmic Mdm2. At the same time, p53 down-regulates nucleic Mdm2 by directly inhibiting the nuclear translocation of p53 [88][89].

Based on these interactions, an SMN for the p53 network is established as follows: $V = \{X_1, X_2, X_3\}$, where X_1 has ternary values, each of which indicates a different concentration level of the p53 protein (low, medium and high) [88], while X_2 and X_3 are binary nodes, with the ternary functional sets $F_1 = \{f_1^{(1)}, f_2^{(1)}\}$, $F_2 = \{f_1^{(2)}, f_2^{(2)}\}$, and $F_3 = \{f_1^{(3)}, f_2^{(3)}\}$. Given their truth tables [89], these functions can be implemented by multiple-valued logic gates. For the gene node X_2 (i.e. cytoplasmic Mdm2), for example, the state transitions are shown in the first and last columns in Table 4.5. These transitions can be implemented by an ES operator and two rotate gates, as shown in **Fig. 4.6**. The intermediate states during the transitions are shown in Table 4.5.

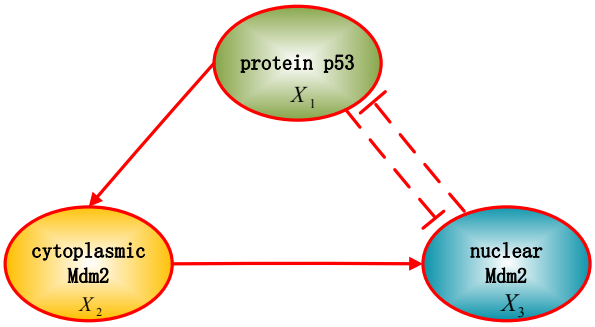


Fig. 4.5. The multiple-valued p53-Mdm2 network under DNA damage (adapted from [88][89]).

Table 4.5. State transitions of X_2.

X_1	X^1 (≥1)	X^2 (rotate)	X_2
0	1	2	0
1	1	2	0
2	2	0	1

Table 4.6. Truth table for X_1 [89].

X_3	X_1	X_1
0	0	1
0	1	2
0	2	2
1	0	0
1	1	0
1		1

Table 4.7. Truth table for X_3 (adapted from [89]).

X_1	X_2	X_3
0	0	0
0	1	1
1	0	0
1	1	1
2	0	0
2	1	1

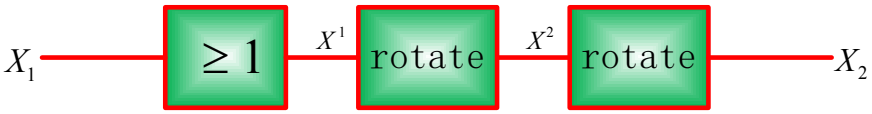


Fig. 4.6. A stochastic multiple-valued network for gene X_2 (cytoplasmic Mdm2).

Similarly, the implementation functions for the other genes X_1 and X_3 can be determined from their truth tables as well (in Tables 4.6 and 4.7 respectively).

While the state transition in [89] is dependent on the current state and the state after transition, random state transitions are considered in this work, as in [18]-[20][33]. Under this assumption, the present state is transitioned into a next state with a transition

probability when perturbation occurs. The selection probabilities are shown in Table 4.8 for the predictor functions.

Table 4.8. The selection probabilities of the predictor functions for the multiple-valued p53-Mdm2 network.

f^1	f^2	f^3
0.95	0.95	0.95
0.05	0.05	0.05

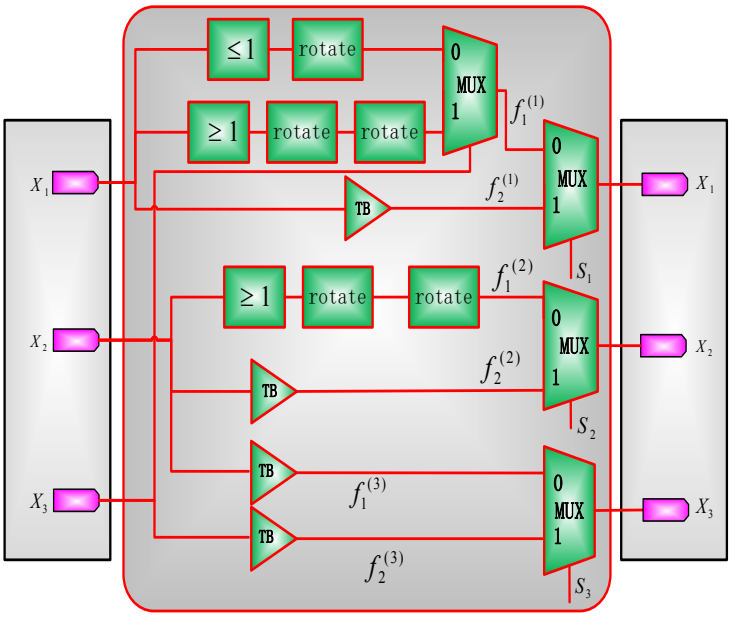


Fig. 4.7. A stochastic multiple-valued network (SMN) for the p53-Mdm2 network under DNA damage.

For the p53-Mdm2 network in **Fig. 4.5**, an SMN can be constructed for implementing its functions, as shown in **Fig. 4.7**. For this three-gene network, a two-input multiplexer is used for each gene to probabilistically select a function with the selection probability encoded in the control sequence. For the update functions, $f_1^{(i)}$ $(i \in \{1,2,3\})$ is for the state transition due to interactions with other genes or the change of the current state, while $f_2^{(j)}$ $(j \in \{1,2,3\})$ indicates the preservation of the current state. In this model, the effect of asynchronicity [98] is implicitly considered at each step of the state updating process. For each input state, the output sequences are read out and decoded into (transition) probabilities.

The p53 SMN model is used to compute the STM for this network, which is compared to the STM obtained by a Markov chain analysis. Let $\boldsymbol{A}_{SMN}$ and $\boldsymbol{A}_{MCA}$ be the STMs obtained by the SMN and a Markov chain analysis; the difference between these two matrices is then given by $\Delta\boldsymbol{A} = \boldsymbol{A}_{SMN} - \boldsymbol{A}_{MCA}$. For the multiple-valued p53-Mdm2 network with no perturbation, we obtain $\|\Delta\boldsymbol{A}\|_1$ = 0.0049, $\|\Delta\boldsymbol{A}\|_2$ = 0.0023 and $\|\Delta\boldsymbol{A}\|_\infty$ = 0.0021 by using a sequence length of 10,000 values for the SMN.

The STM of the p53-Mdm2 network under perturbation can similarly be computed using an SMN with perturbation (by implementing the SMN in **Fig. 4.7** into that of **Fig. 4.4**). The STMs obtained by different approaches are illustrated in **Fig. 4.8**, while the norms of the differences, $\|\Delta\boldsymbol{A}\|_1$, $\|\Delta\boldsymbol{A}\|_2$ and $\|\Delta\boldsymbol{A}\|_\infty$, are shown in Table 4.9 for using different sequence lengths. The average run time is also provided for both approaches.

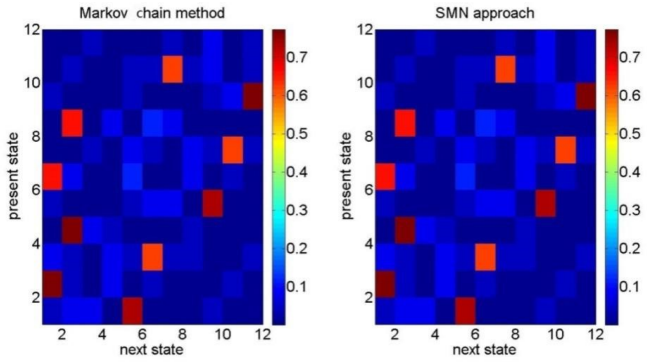


Fig. 4.8. State transition matrices (STMs) obtained by Markov chain method [33] and stochastic multiple-valued network (SMN) approach for the dynamic p53-Mdm2 network. Sequence length: L = 10,000 values; perturbation rate: p = 0.1.

As revealed in Table 4.9, the difference between the STMs computed using the SMN model and the Markov chain analysis decreases with the increase of sequence length L. For the same accuracy requirement, as can be seen, a larger sequence length is needed for a higher perturbation rate. This relationship between the sequence length and perturbation rate is further shown in **Fig. 4.9**. However, the computational inaccuracy due to the inherent stochastic fluctuation in stochastic computation is generally small and

negligible. Hence, the proposed SMN model can compute the STM of a PMN with or without perturbation.

Table 4.9. Norms of the difference between the state transition matrices (STMs) obtained by Markov chain analysis (MCA) and stochastic multiple-valued network (SMN) for the p53-Mdm2 network, $\Delta \boldsymbol{A}_{MCA-SMN}$.

		$p = 0$		
L (bits)		1,000	10,000	100,000
$\|\Delta \boldsymbol{A}_{MCA-SMN}\|_1$		0.0091	0.0049	7.6500×10^{-4}
$\|\Delta \boldsymbol{A}_{MCA-SMN}\|_2$		0.0091	0.0023	8.1496×10^{-4}
$\|\Delta \boldsymbol{A}_{MCA-SMN}\|_\infty$		0.0183	0.0021	0.0016
Avg. time (s)	MCA	5.22×10^{-3}		
	SMN	6.8×10^{-2}	0.59	5.73
		$p = 0.1$		
L (bits)		1,000	10,000	100,000
$\|\Delta \boldsymbol{A}_{MCA-SMN}\|_1$		0.0368	0.0097	0.0030
$\|\Delta \boldsymbol{A}_{MCA-SMN}\|_2$		0.0210	0.0061	0.0016
$\|\Delta \boldsymbol{A}_{MCA-SMN}\|_\infty$		0.0401	0.0105	0.0032
Avg. time (s)	MCA	1.54×10^{-2}		
	SMN	5.94×10^{-2}	0.65	5.97

p: perturbation rate; $p = 0.1$; L: sequence length.

A probabilistic network with random perturbation evolves as an ergodic Markov chain [19], because the non-zero perturbation rate makes all the states in the network connected. Hence, a steady state exists in a network with perturbation. The SSDs for the p53 network under DNA damage are obtained by different approaches, as shown in **Fig. 4.10**.

As shown in **Fig. 4.10**, all approaches produce similar SSDs. In fact, the difference between the results by the SMN and the accurate Markov chain analysis is negligible when reasonably long stochastic sequences are used (such as those of 10,000 values). Using the STM computed by an SMN approach results in very close values of SSD compared to the rigorous Markov analysis.

A further analysis shows that the relative error is less than approximately 0.2% for the stochastic approach. Individual gene expressions are shown in **Fig. 4.11** for a single simulation of 30 transitions. It can be seen that the likely expression levels of p53 and

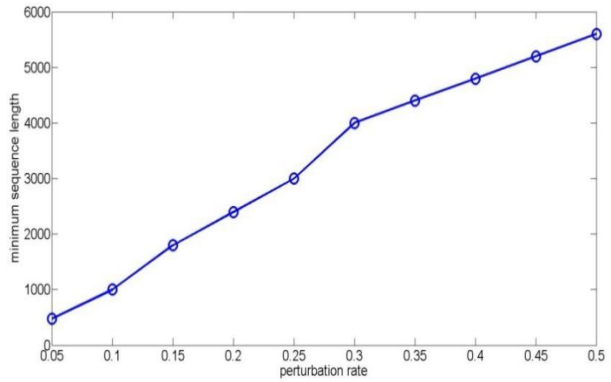


Fig. 4.9. The relationship between the minimum sequence length required in the process of computing the state transition matrix (STM) (with an accuracy requirement of $\|\cdot\|_2 = 0.02$) and the perturbation rate for the multiple-valued p53-Mdm2 network.

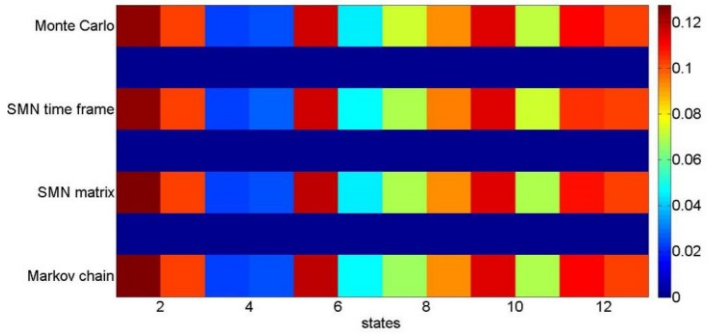


Fig. 4.10. Steady state distributions (SSDs) for the multiple-valued p53 network after 30 state transitions with an initial state of 000. The X-axis indicates the network state, the Y-axis is for the different approaches and the color bar on the right shows the values of the SSDs. Perturbation rate: $p = 0.1$ and the sequence length or simulation runs: $L =$ 10,000 values.

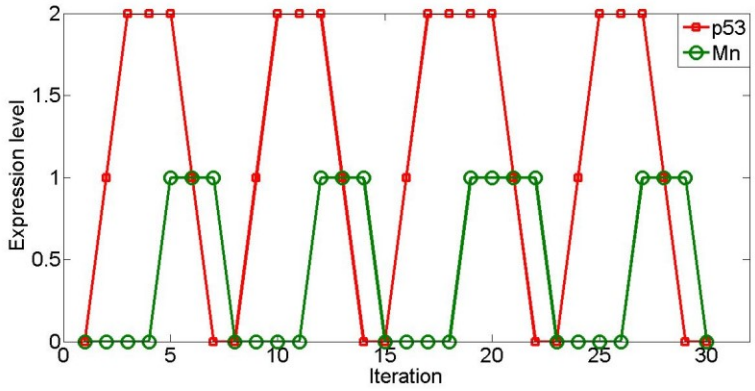


Fig. 4.11. Individual gene expressions for the p53 network generated from a single simulation of 30 iterations with an initial state of 011. X-axis indicates the iteration number and Y-axis shows the expression level of p53 or nuclear Mdm2.

nuclear Mdm2 follow an oscillatory pattern as analytically [89] and experimentally [99] shown previously.

4.6.2. Application on a WNT5A Network

Next, a WNT5A network [33] is used to illustrate the efficiency of the SMN model and the time-frame expansion technique. A ten-gene network is derived from the predictive relationships in Table 4.10. The selection probabilities of predictor functions are also given in Table 4.10 (estimated from [33]). **Fig. 4.12** shows a detailed structure of the network with double (or single) - headed arrows indicating the bi (or uni) - directional relationships of gene pairs. While the number of output arcs varies, every node (or gene) has three input arcs in **Fig. 4.13**.

Table 4.10. The selection probability of the predictor functions for 10 genes (estimated from [33]).

Target	Predictor f_1	Select probability	Predictor f_2	Select probability	Predictor f_3	Select Probability
pirin	WNT5A	0.6	STC2	0.2	HADHB	0.2
WNT5A	pirin	0.6	S100P	0.2	RET-1	0.2
S100P	WNT5A	0.33	RET-1	0.33	Synuclein	0.34
RET-1	pirin	0.43	WNT5A	0.24	S100P	0.33
MMP-3	S100P	0.43	RET-1	0.25	HADHB	0.32
PHO-C	MART-1	0.33	Synuclein	0.33	STC-2	0.34
MART-1	pirin	0.44	WNT5A	0.28	MMP-3	0.28
HADHB	pirin	0.3	WNT5A	0.4	MMP-3	0.3
Synuclein	pirin	0.25	S100P	0.25	MART-1	0.5
STC2	pirin	0.35	WNT5A	0.3	PHO-C	0.35

For the 10-gene ternary WNT5A network, it requires a STM of $3^{10} = 59049$ columns and rows for an accurate analysis. This makes it difficult, if not impossible, to estimate the steady state of an SMN using a matrix-based analysis. In general, it is difficult to analyze a large gene network, due to its excessive computational overhead. An MC method has been used in [33] for evaluating the SSD of a network with perturbation. However, the MC method is very time consuming due to the slow convergence typically encountered in an MC simulation.

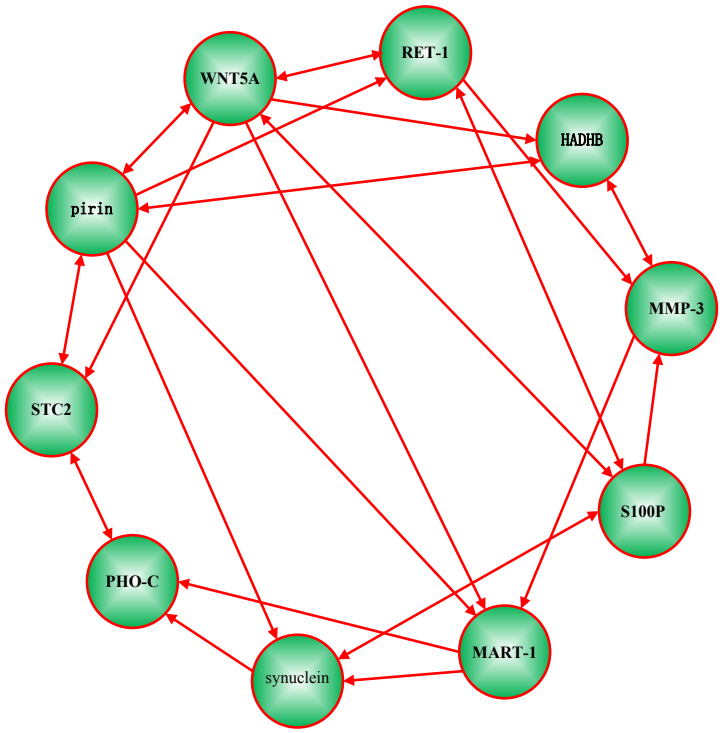


Fig. 4.12. A ternary WNT5A network with gene interactions (adapted from [33]).

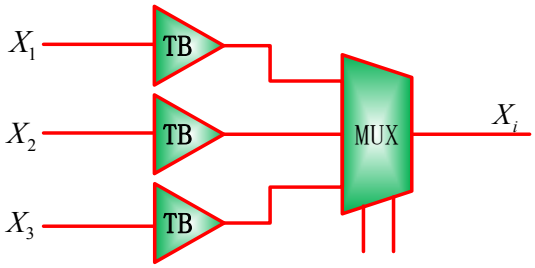


Fig. 4.13. A stochastic multiple-valued network (SMN) module for gene *i* in the ternary WNT5A network, with the predictor function implemented by a ternary buffer. Let $\boldsymbol{G}_i = (X_1, X_2, X_3)$ be the input vector for gene i; the input vector for each of the genes in the ternary WNT5A network is given by: $\boldsymbol{G}_{WNT5A} = (pirin, S100P, RET-1)$; $\boldsymbol{G}_{pirin} = (WNT5A, STC2, HADHB)$; $\boldsymbol{G}_{RET-1} = (pirin, WNT5A, S100P)$; $\boldsymbol{G}_{HADHB} = (pirin, WNT5A, MMP-3)$; $\boldsymbol{G}_{MMP-3} = (S100P, RET-1, HADHB)$; $\boldsymbol{G}_{S100P} = (WNT5A, RET-1, Synuclein)$; $\boldsymbol{G}_{MART-1} = (pirin, WNT5A, MMP-3)$; $\boldsymbol{G}_{Synuclein} = (pirin, S100P, MART-1)$; $\boldsymbol{G}_{PHO-C} = (MART-1, Synuclein, STC2)$; $\boldsymbol{G}_{STC2} = (pirin, WNT5A, PHO-C)$.

However, an SMN model can be constructed for the ternary WNT5A network, as shown in **Fig. 4.13**. For this SMN, the SSD can be estimated using the aforementioned time-frame expansion technique and compared with the MC simulation [33]. By the

time-frame expansion technique, the temporal operation of an SMN is laid out into a series of identical SMN modules in the spatial domain. The required iterations of the SMN are determined by the number of state transitions before reaching a steady state. As in [63], a steady state is considered to have been reached if the discrepancy between two adjacent simulations is smaller than a threshold or the number of simulations has reached a maximum value. The state or GAP of the WNT5A network can be represented by a ternary vector as $(x_1,\ x_2,\ \cdots,\ x_{10})$, or its decimal index. The SSDs of the network with all of the 59049 states, obtained using the SMN and the MC method [33], are shown in **Fig. 4.14**.

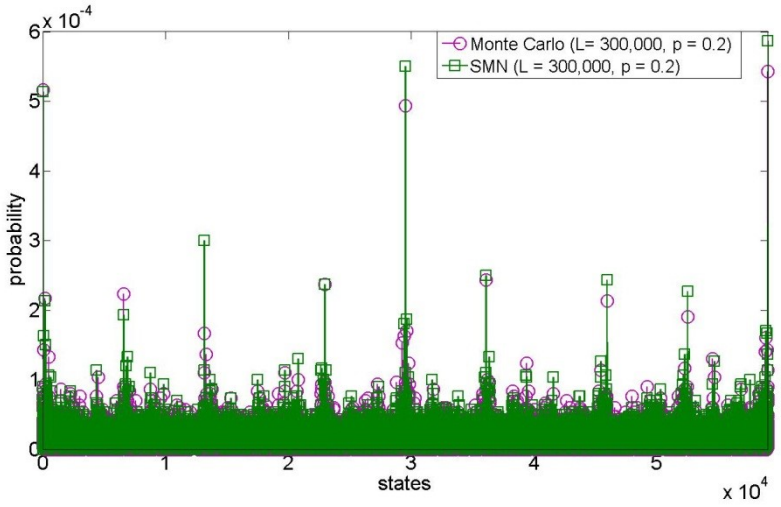


Fig. 4.14. Steady state distributions (SSDs) of the ternary WNT5A network using the stochastic multiple-valued network (SMN) model and Monte Carlo (MC) simulation with perturbation rate $p = 0.2$ and sequence length or simulation runs $L = 300,000$ values.

The norms of the differences of the SSDs obtained using the time-frame expanded SMN approach with different sequence lengths and the MC method are shown in Table 4.11. As can be seen, the time-frame expanded SMN technique effectively evaluates the SSD of the WNT5A network and produces very accurate results compared to the Monte Carlo simulation [33]. The average run time reveals the efficiency of the SMN approach. This is because the use of randomly permuted sequences results in a faster convergence than in the MC simulation. The use of longer stochastic sequences further improves the accuracy of evaluation and remains faster by several orders of magnitude than the MC

method. Albeit at a higher memory cost than the MC simulation (shown in Table 4.11), the SMN approach requires much less memory than an accurate approach such as a Markov chain analysis (shown in Table 4.4). Since it is difficult to compute the STM or SSD of a large GRN by using an accurate analysis, a time-frame expanded SMN provides an alternative method to evaluate the SSD of a large network with a tunable tradeoff between accuracy and efficiency by using stochastic sequences of different lengths.

Table 4.11. Norms of the difference between the steady state distribution (SSDs) obtained by the time-frame expanded stochastic multiple-valued network (SMN) technique and Monte Carlo (MC) simulation for the ternary WNT5A network with perturbation rate $p = 0.2$. The average run time is also shown.

Num/L		3k	30k	300k	3000k
$\|\boldsymbol{SSD}_{MC} - \boldsymbol{SSD}_{SMN}\|_1$		1.8827	1.3291	0.4915	0.1605
$\|\boldsymbol{SSD}_{MC} - \boldsymbol{SSD}_{SMN}\|_2$		0.0258	0.0082	0.0026	8.5342×10^{-4}
$\|\boldsymbol{SSD}_{MC} - \boldsymbol{SSD}_{SMN}\|_\infty$		1.0000×10^{-3}	2.6667×10^{-4}	1.3333×10^{-4}	5.6333×10^{-5}
Avg. time (s)	MC	98.48	981.16	9731.04	97336.50
	SMN	0.49	4.24	58.93	673.93
Required memory (M Byte)	MC	2.71	10.01	51.21	599.61
	SMN	9.81	40.94	373.30	3696.50

L: sequence length for the stochastic approach; Num: number of simulation runs for the MC method; $\boldsymbol{SSD}_{MC}$ and $\boldsymbol{SSD}_{SMN}$ respectively denote the steady state distributions obtained by the MC simulation and the time-frame expanded SMN technique; a maximum number of 50 iterations is applied to the steady state evaluation.

4.7. Summary

As a generalization of SBNs, SMNs are proposed as a fast approach to modeling the effects of noise in GRNs. In an SMN, the STM can be computed with a complexity of $O(nLk^n)$, where n is the number of genes in a network, k is the quantization level of a gene's state and L is a factor determined by the stochastic sequence length. Since L increases slower with n than the number of network functions N, this result is an improvement compared to the previous result of $O(nNk^{2n})$ for an accurate analysis. The use of randomly permuted sequences further increases computational efficiency and allows for a tunable tradeoff between accuracy and efficiency. A steady state analysis using a time-frame expansion technique has shown a significant speedup compared to a Markov chain analysis and produced very accurate results compared to MC simulation.

SMNs are constructed for the analysis of a multiple-valued p53-Mdm2 network and a ternary WNT5A network under gene perturbation. Simulations of the SMNs have revealed the oscillatory dynamics of the p53-Mdm2 network with random gene perturbation. The SMN approach can also predict the SSD of the WNT5A network with gene perturbation. Hence, the SMNs are useful in evaluating the effects of gene perturbation and, potentially, helpful in drug discovery for an intervention-based gene therapy.

Chapter 5

Asynchronous Stochastic Boolean Networks as Gene Network Models

For simplicity, the Boolean models usually consider a synchronous update of all genes' states in a gene regulatory network (GRN) [20][100][101]. However, this assumed synchronicity may not be appropriate, because each gene may require a different period of time for changing its state, as in asynchronous networks [102], and stochastic networks with time delays [103] and parameter uncertainties [104]. In this chapter, asynchronous stochastic Boolean networks (ASBNs) are proposed for investigating various asynchronous state updating strategies in a GRN. As in stochastic computation, ASBNs use randomly permutated stochastic sequences to encode probability. A GRN is considered to be subject to noise and external perturbation, which are investigated by several stochasticity models. In [105][106], stochasticity is modeled as the perturbation of a gene's state that occurs with certain probability; however, this stochasticity in node (SIN) model over-expresses the effect of noise. Both stochasticity and asynchronicity are considered in the state evolution of a GRN. As a case study, ASBNs are utilized to investigate the dynamic behavior of a T helper network. It is shown that ASBNs are fast in evaluating the steady state distributions (SSDs) of the network with random gene perturbation. The SSDs found by using ASBNs show the robustness of the attractors of the T helper network, when various stochasticity and asynchronicity models are considered to investigate its dynamic behavior. The results in this chapter have been published in [107].

The novelty of this chapter is as follows:

- ASBNs are proposed for a fast modeling of GRNs with asynchronous state updating properties.
- Stochastic analyses are carried out for the asynchronous models of SIN and stochasticity with propensity parameters (SPP).

- The obtained SSDs can be used to estimate the attractors of, especially, a large gene network. This is shown by the study of a T helper network of 23 genes. The robustness of the attractors for the T helper network is demonstrated for a number of synchronous and asynchronous state update strategies.

5.1. Asynchronous Gene Regulatory Network

A network model is presented in **Fig. 5.1** for a deterministic synchronous update strategy. If the predict function F_i, $i \in \{1, 2, \cdots, n\}$, consists of several update functions with different selection probabilities, this network becomes a (synchronous) probabilistic network [20][33]. Otherwise, a deterministic network is obtained if F_i only consists of one implementation function.

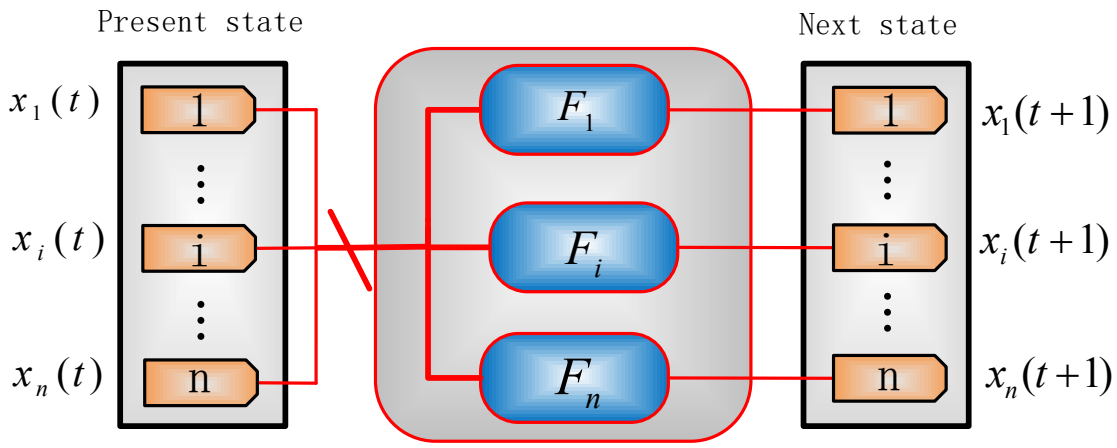


Fig. 5.1. A general synchronous gene network model. F_i denotes the update function for gene i, $x_i(t)$ and $x_i(t+1)$ are the states of gene i at time t and $t+1$ respectively, $i \in \{1, 2, \cdots, n\}$. n is the number of genes in the investigated network.

When asynchronicity is considered, a deterministic-asynchronous probabilistic Boolean network (DA-PBN) assumes that the state of each gene is independently updated according to its own updating period [20]. Whether the state of gene *i* is updated or not at a time is indicated by a binary variable c (either 1 or 0 respectively). The next state of gene i, $x_i(t+1)$, is then given by:

$$x_i(t+1) = \begin{cases} f^{(i)}\big(x_1(t), \cdots, x_n(t)\big) & if\ c = 1 \\ x_i(t) & otherwise \end{cases}, \tag{5.1}$$

where $f^{(i)}$ is the predictor function in the DA-PBN for gene i; c is a binary variable that determines whether a gene is to be updated or not.

Furthermore, a different number of genes can be selected in an asynchronous update process. The genes can be updated at the same time in one time step or following certain updating order. In [46][102], only one gene is randomly selected at a time for updating its state, while in [98], m gene nodes are randomly chosen for a simultaneous state update in one time step, where m is a randomly generated number. Even in one time step, however, the update of m genes is likely to be asynchronous and the state of a gene depends on the updating order of the inputs of its predictor function. In [108], all the gene nodes are asynchronously updated one by one, so an updated state will subsequently affect the other components' states, even within the same time step. Hence, an asynchronous update model is more realistic in modeling biological behaviors. Since the next state of a network is determined by the updated gene and the remaining genes, an updated gene state is determined by the current state vector and the corresponding predictor function [102].

5.2. Stochastic Computational Models for Asynchronous Update Strategies

In a biological system, some cells may update their states immediately while it may take longer for other cells to respond. This phenomenon corresponds to a variable reaction time or rate for each cell. To model this asynchronous process, a 2-to-1 multiplexer (MUX) is used to determine whether a gene's state is updated or not, as determined by the value of the control bit S_i, shown in **Fig. 5.2**. If the control bit S_i is 1, where $i \in \{1, 2, \cdots, n\}$ and n is the number of genes, then the ith gene's state is determined by the input genes' states and the predictor function F_i (i.e. $F_i\big(x_1(t), \cdots, x_n(t)\big)$). If $S_i = 0$, the gene remains at its current state, indicated by the

application of the buffer in **Fig. 2.2** (i.e. $x_i(t+1) = x_i(t)$). Hence, the stochastic architecture in **Fig. 5.2** accurately implements the function of (5.1).

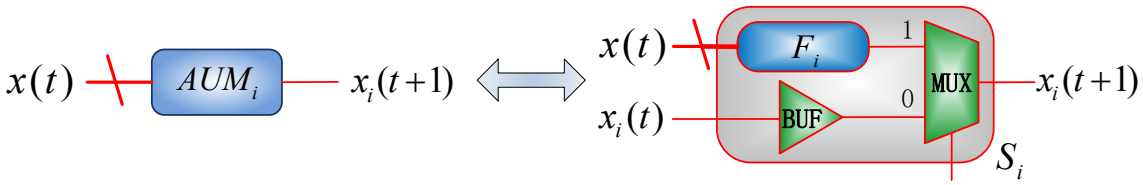


Fig. 5.2. An asynchronous update module (AUM) for gene i, referred to as AUM_i, consists of a 2-to-1 multiplexer (MUX) with a control bit S_i. F_i is the predictor function for gene i. $x(t) = (x_1(t), \cdots, x_i(t), \cdots, x_n(t))$ with $x_i(t)$ being the state of gene i at time t, while $x_i(t+1)$ indicates the next state of gene i, $i \in \{1,2, \cdots, n\}$. n is the number of genes in the investigated network.

Stochastic architectures for asynchronous networks can be constructed with the control sequences of the MUX that are generated according to the updating rules. The state transition matrices (STMs) are usually applied to derive the steady state distribution (SSD) by an iterative evaluation, especially for synchronous models. However, a matrix-based analysis becomes cumbersome to perform due to the size of STMs for a large network; an analysis using STMs becomes even more challenging for asynchronous networks. Hence, the SSD is evaluated by using the so-called time-frame expansion technique [30]. By this technique, the temporal evolution of an asynchronous network is simulated using a spatially iterative structure of the asynchronous network model.

Different strategies of updating one gene or m genes synchronously or in a random order are presented as follows:

- The ROG (randomly one gene) model [102]: At each time step, only one gene node is randomly selected for a state update, while the remaining nodes stay at their present states.
- The RMG (randomly m genes) model [98]: At each time step, m gene nodes are randomly selected and synchronously updated, while the remaining nodes preserve

their present values. Here, m can be either a random number or a fixed value. If $m = 1$, the RMG model is simplified to the ROG model.

- The ARO (all genes updated in a random order) model [108]: At each time step, all gene nodes are updated; however, the state update of each gene occurs one by one in a random order. The update is carried out immediately and a state change will affect the change of other genes' states, even within the same time step.
- The MRO (m genes updated in a random order) model: At each time step, only m gene nodes are randomly selected for updating their states and each state update occurs in a random order, while the other genes remain at their present states. If $m = 1$, this is simplified to the ROG model. If $m = n$, the result will be the same as the ARO model.

The general architectures are similar for the ROG and RMG models; the only difference lies in the control sequences $\{S_1,\ S_2,\ \cdots,\ S_n\}$ for the 2-to-1 MUXs, where S_i indicates the control sequence for gene $i \in \{1, 2, \cdots, n\}$. For the control sequences of asynchronous update modules (AUMs) in the ROG model, if the kth bit in the control sequence S_i is 1, i.e., $S_{i,k} = 1$, then $S_{j,k} = 0$ for $j \neq i$, $i, j \in \{1, 2, \cdots, n\}$, $k \in \{1, 2, \cdots, L\}$ with $\sum_{i=1}^{n} S_{i,k} = 1$ as only one gene is selected for updating its state, where L indicates the length of the control sequences. In contrast, the requirement of $\sum_{i=1}^{n} S_{i,k} = m$ must be met for the RMG model as m gene nodes are randomly selected for updating the states using the control sequences.

The generation of the stochastic control sequences for the AUMs is explained next with an example of a 5-gene network using the ROG and RMG models. Let the number of genes to be updated for the RMG model, i.e. m, be 2. For each column of the generated sequence matrices, only one gene is updated while two gene nodes are randomly selected to be refreshed for $j \in \{1, 2, \cdots, L\}$ for the RMG model. At certain time step, let the generated control sequences be C_{ROG} and C_{RMG} for the ROG and RMG models respectively, as illustrated in **Fig. 5.3**.

For the ARO and MRO models, similarly, the evolution of an asynchronous network is simulated by the time-frame expansion technique. In the ARO model, as all genes are updated in a random order, each time step is divided into n sub-steps and only one gene is chosen to be updated at each sub-step while the other genes maintain their present states. Assume at time t, only one gene is updated at each sub-step (n sub-steps in total); hence, after the evolution of n sub-steps, all the genes are updated, followed by the evolution at time $t+1$. The ARO model is implemented with only one AUM with a control bit of 1 at each sub-step for the gene to be updated (the AUM is simplified to a buffer as the control bit is 0 for the other genes). If m genes are to be updated in a different order, there will be n-m buffers (or AUMs with a control bit of 0) for the n sub-steps. This becomes an MRO model.

$$C_{ROG} = \begin{bmatrix} S_1 \\ S_2 \\ S_3 \\ S_4 \\ S_5 \end{bmatrix} = \begin{matrix} \\ \\ \\ \\ \\ \end{matrix} \overset{j}{\begin{bmatrix} \cdots & 0 & \cdots \\ \cdots & 0 & \cdots \\ \cdots & 1 & \cdots \\ \cdots & 0 & \cdots \\ \cdots & 0 & \cdots \end{bmatrix}} \qquad C_{RMG} = \begin{bmatrix} S_1 \\ S_2 \\ S_3 \\ S_4 \\ S_5 \end{bmatrix} = \begin{matrix} \\ \\ \\ \\ \\ \end{matrix} \overset{j}{\begin{bmatrix} \cdots & 0 & \cdots \\ \cdots & 1 & \cdots \\ \cdots & 0 & \cdots \\ \cdots & 1 & \cdots \\ \cdots & 0 & \cdots \end{bmatrix}}$$

(a) (b)

Fig. 5.3. An illustrative example of the generated control sequences at a time step for a network of five genes. (a) The control sequences for the randomly one gene (ROG) model. (b) The control sequences for the randomly m genes (RMG) model. S_i indicates the control sequence for gene i, $i \in [1,5]$; j is for the jth bit in each sequence, which can be considered as the jth trial of the L experiments and L indicates the length of the stochastic sequences and $j \in \{1,2,\cdots,L\}$. $S_{i,j} = 0$ indicates that the control bit is 0 for the AUM in Fig. 5.2, so the state of gene i remains, while $S_{i,j} = 1$ means that the next state is determined by the predictor function F_i. For the RMG model, the number of genes to be updated for the jth trail is assumed to be two, thus $\sum_{i=1}^{n} S_{i,j} = 2$ in (b). n is the number of genes in the investigated network.

For the ARO model, at each sub-step only one gene is updated while the others preserve their present states. If $S_{i,k} = 1$, $i \in \{1, 2, \cdots, n\}$, $k \in \{1, 2, \cdots, L\}$, the ith gene's state is updated by the predictor function; otherwise, the current state remains. For the MRO model, the total number of 1's for the n sub-steps at time step i is equal to the number of genes to be updated for the MRO model. All genes remain at their present states if the control bits for the AUMs are 0 at a time step. The updating order is indicated by the different position of 1's in the generated sequences.

$$C_{ARO} = \overset{t}{\left[\cdots \quad C_{ARO_t} \quad \cdots\right]}$$

$$C_{ARO_t} = \begin{array}{c} \\ x_1 \\ x_2 \\ x_3 \\ x_4 \\ x_5 \end{array} \begin{array}{c} \begin{array}{ccccc} t_1 & t_2 & t_3 & t_4 & t_5 \end{array} \\ \begin{bmatrix} 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 1 & 0 \\ 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 \end{bmatrix} \end{array}$$

(a)

$$C_{MRO} = \overset{t}{\left[\cdots \quad C_{MRO_t} \quad \cdots\right]}$$

$$C_{MRO_t} = \begin{array}{c} \\ x_1 \\ x_2 \\ x_3 \\ x_4 \\ x_5 \end{array} \begin{array}{c} \begin{array}{ccccc} t_1 & t_2 & t_3 & t_4 & t_5 \end{array} \\ \begin{bmatrix} 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 \end{bmatrix} \end{array}$$

(b)

Fig. 5.4. An illustrative example of the generated control sequences at a time step for a network of five genes. (a) The control sequences for the all genes updated in a random order (ARO) model. (b) The control sequences for the m genes updated in a random order (MRO) model. t is for the tth time step and t_1, ⋯, t_5 are the sub-steps at time i. C_{ARO_t} and C_{MRO_t} are the generated control sequences at the sub-steps of time t. An entry of 0 indicates the state of gene i remains while a value of 1 indicates that the next state is determined by the predictor functions. The number of genes to be updated at time t is three (in a random order), thus $\sum C_{MRO_t} = 3$ in (b).

An example of the random updating order for the AUMs at time i is illustrated for a network of five genes. Following the previous analysis, time step i is divided into five sub-steps. The control sequences of C_{ARO} and C_{MRO} for the ARO and MRO models are shown in **Fig. 5.4(a)** and **(b)** respectively.

The updating order of the ARO model at time t is given by the control sequences in **Fig. 5.4(a)**, i.e., $x_1 \rightarrow x_4 \rightarrow x_5 \rightarrow x_3 \rightarrow x_2$, while the updating order of the MRO

model at time step i is $x_3 \rightarrow x_5 \rightarrow x_1$, as given by the control sequences in **Fig. 5.4(b)**. For genes x_2 and x_4, the present states at time t remain the same.

5.3. Asynchronous Stochastic Boolean Networks

Any gene node in a GRN can change the expression level due to noise or a gene perturbation that occurs with certain probability. The perturbation probability can be encoded as a perturbation flag vector γ with each element indicating whether a gene is to be perturbed or not. When a perturbation occurs, the state of the perturbed gene is determined by the present state and the perturbation flag vector. In the stochasticity in node (SIN) model [109], therefore, the effect of perturbation can be modeled by an XOR gate [30], by which the gene state at time t will be flipped with a probability of p, as shown in **Fig. 5.5(a)**.

In the SIN model, the correlation between a gene node's current expression level and the probability of changing the expression value is not taken into consideration. Even if the expression level of the input node for an update function guarantees the activation or degradation, there still exists a probability that the process will not occur due to stochasticity. Hence, a stochasticity model is proposed in [89] with propensity parameters,

$$F = \{f_i, p_i^{\uparrow}, p_i^{\downarrow}\}_{i=1}^{n}, \tag{5.2}$$

where f_i indicates the update function for node x_i, $0 \leq p_i^{\uparrow} \leq 1$ and $0 \leq p_i^{\downarrow} \leq 1$ denote the activation propensity and degradation propensity, and n is the number of genes. A model of stochasticity with propensity parameters (SPP) is described by [89]:

$$x_i \rightarrow f_i(x) = \begin{cases} p_i^{\uparrow} & x_i < f_i(x) \\ p_i^{\downarrow} & x_i > f_i(x) \\ 1 & x_i = f_i(x) \end{cases} \tag{5.3}$$

In order to implement the function of (5.3), a multiplexer is used here to simulate the behavior of stochasticity with parameters. A general structure is shown in **Fig. 5.5(b)**. For the architecture in **Fig. 5.5(b)**, the output is determined by the value in the control sequence S. S consists of ternary values $\{0, 1, 2\}$, which indicate the three cases of $S_i(t) < S_{f_i}(t)$, $S_i(t) = S_{f_i}(t)$ and $S_i(t) > S_{f_i}(t)$ for any $i \in \{1, 2, \cdots, n\}$. $S_i(t)$ and $S_i(t+1)$ are the stochastic sequences encoding the probabilities of gene i at time t and $t+1$ respectively. $S_{f_i}(t)$ is the stochastic sequence encoding the expected probability for the state of gene i at time t. This probability is calculated from $f_i(x(t))$, where f_i indicates the update function for gene i and $x(t) = \{x_1(t), \cdots, x_i(t), \cdots, x_n(t)\}$ with $x_i(t)$ being the state of gene i at time t. $S_{p_i^{\uparrow}}$ and $S_{p_i^{\downarrow}}$ represent the stochastic sequences encoding the propensity parameters.

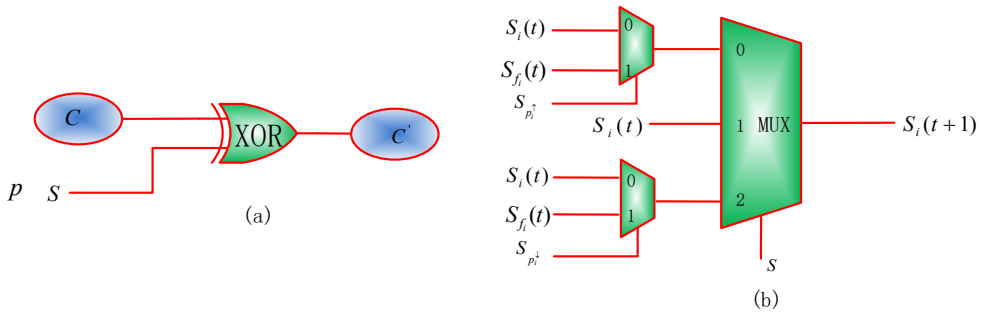


Fig. 5.5. Stochastic architectures for the models of stochasticity in node (SIN) and stochasticity with propensity parameter (SPP). (a) The stochastic architecture of the SIN model for a binary gene node [30]. (b) The stochastic architecture of the SPP model for updating the state of a gene.

Let $(S_i(t))_j$, $(S_{f_i}(t))_j$ and $(S_i(t+1))_j$ be the jth value in $S_i(t)$, $S_{f_i}(t)$ and $S_i(t+1)$. For a ternary case, if $(S_i(t))_j = (S_{f_i}(t))_j$, the output $(S_i(t+1))_j$ is the same as $(S_i(t))_j$, because the transition $(S_i(t))_j \to (S_{f_i}(t))_j$ occurs with a probability of 1. However, if $(S_i(t))_j < (S_{f_i}(t))_j$, $(S_i(t+1))_j$ is determined by the output of a 2-to-1 multiplexer with $S_i(t)$ and $S_{f_i}(t)$ as input sequences and $S_{p_i^{\uparrow}}$ as the control sequence. If $(S_i(t))_j = 0 < (S_{f_i}(t))_j = 1$, for example, then $(S_i(t+1))_j = 1$ with a probability of $p_i^{\uparrow}$. If $(S_i(t))_j > (S_{f_i}(t))_j$, then $(S_i(t+1))_j = (S_{f_i}(t))_j$ with a

probability of $p_i^{\downarrow}$ or $(S_i(t+1))_j = \left(S_i(t)\right)_j$ with a probability of $1-p_i^{\downarrow}$. These functions are implemented by the 2-to-1 multiplexers in the model in **Fig. 5.5(b)**. Let a and b be the values in $(S_i(t))_j$ and $(S_{f_i}(t))_j$ respectively; the next state of the gene i is obtained as:

$$S_i(t+1) = \begin{cases} S_i(t), \ if \ a = b \\ S_{f_i}(t) \ with \ p_i^{\uparrow}, or \ S_i(t) \ with \ 1-p_i^{\uparrow}, if \ a < b \\ S_{f_i}(t) \ with \ p_i^{\downarrow}, or \ S_i(t) \ with \ 1-p_i^{\downarrow}, \ if \ a > b \end{cases} \quad (5.4)$$

which is equivalent to the function of (5.3). This shows that the stochastic model in **Fig. 5.5(b)** accurately computes the function of the SPP model.

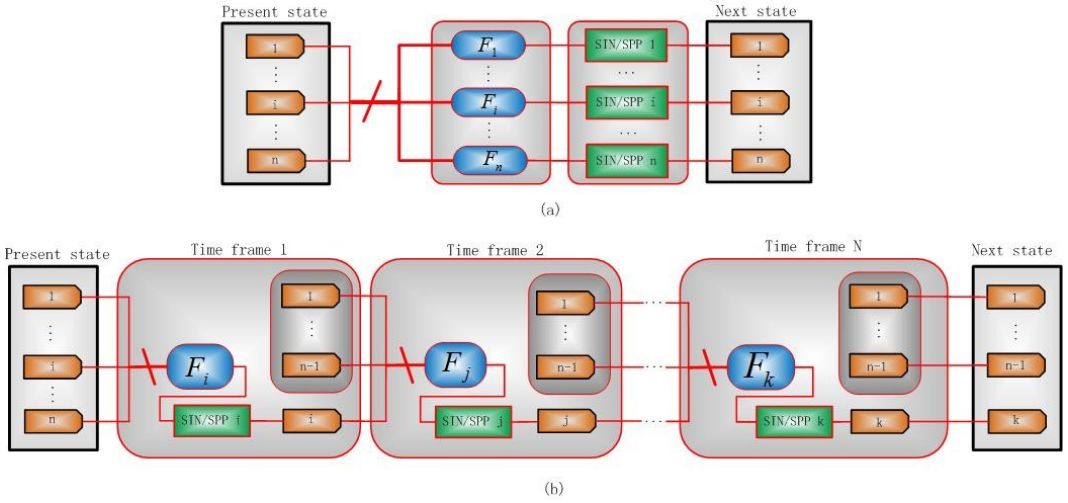


Fig. 5.6. Synchronous and asynchronous stochastic Boolean networks (ASBNs) for different stochasticity models. (a) A synchronous SBN. (b) An ASBN for the randomly one gene (ROG) model. F_i: predictor function for gene i, $i \in \{1,2,\cdots,n\}$. n is the number of genes in the investigated network. The detailed structures and inputs of the SIN and SPP modules are shown in Fig. 5.5 (a) and (b) respectively.

In order to analyze the long run behavior of gene networks with different stochasticity effects, synchronous and asynchronous stochastic Boolean networks (ASBNs) are presented in **Fig. 5.6** for the SIN and SPP models. The synchronous network is obtained by implementing the stochasticity models (either the SIN in **Fig. 5.5(a)** or the SPP in **Fig. 5.5(b)**) into the architecture in **Fig. 5.1**. For simplicity, only the

randomly one gene (ROG) model is illustrated in **Fig. 5.6(b)** for an ASBN. For randomly m gene (RMG) or a different updating order, the stochastic networks can be obtained by applying a stochasticity model (either the SIN in **Fig. 5.5(a)** or the SPP in **Fig. 5.5(b)**) into the stochastic architecture in **Fig. 5.5**. Although discussed within the context of BNs, multiple-valued networks such as those in [95] can also be used to construct an asynchronous network. In a stochastic network, randomly permuted sequence and perturbation vectors are generated for different signal probabilities. With these stochastic sequences as inputs to the stochastic architectures, the stochastic behavior of a gene network can be easily analyzed.

5.4. Results and Discussion

A T helper network of 23 genes with 35 regulatory interaction relationships (**Fig. 5.7**) [110] is considered to show the robustness of the attractors under the various asynchronous update schemes. The functional molecules or molecular complexes are listed as TCR, $NFAT$, $IFN-\beta$, $IFN-\beta R$, $L-18$, $IL-18R$, $IRAK$, $SOCS1$, $IL-12$, $IL-12R$, $STAT4$, $T-bet$, $IFN-\gamma$, $IFN-\gamma R$, $JAK1$, $STAT1$, $IL-4$, $IL-4R$, $STAT6$, $GATA3$, $IL-10$, $IL-10R$, $STAT3$.

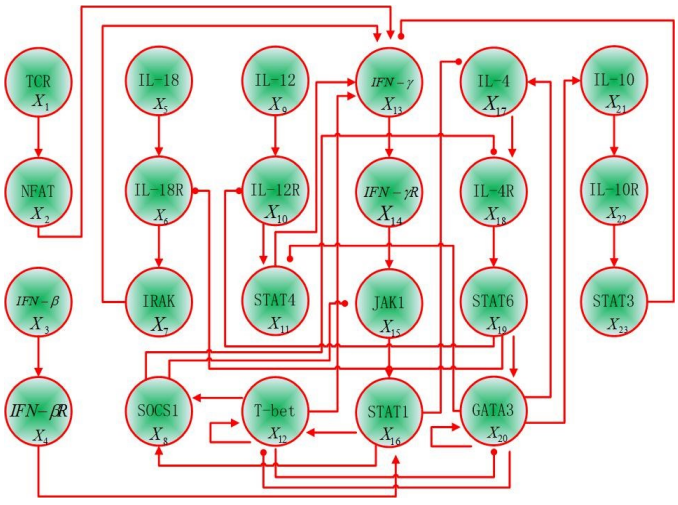


Fig. 5.7. A T helper network. A positive regulatory interaction is indicated by a regular arrow, while a negative interaction is represented by a blunted arrow [110].

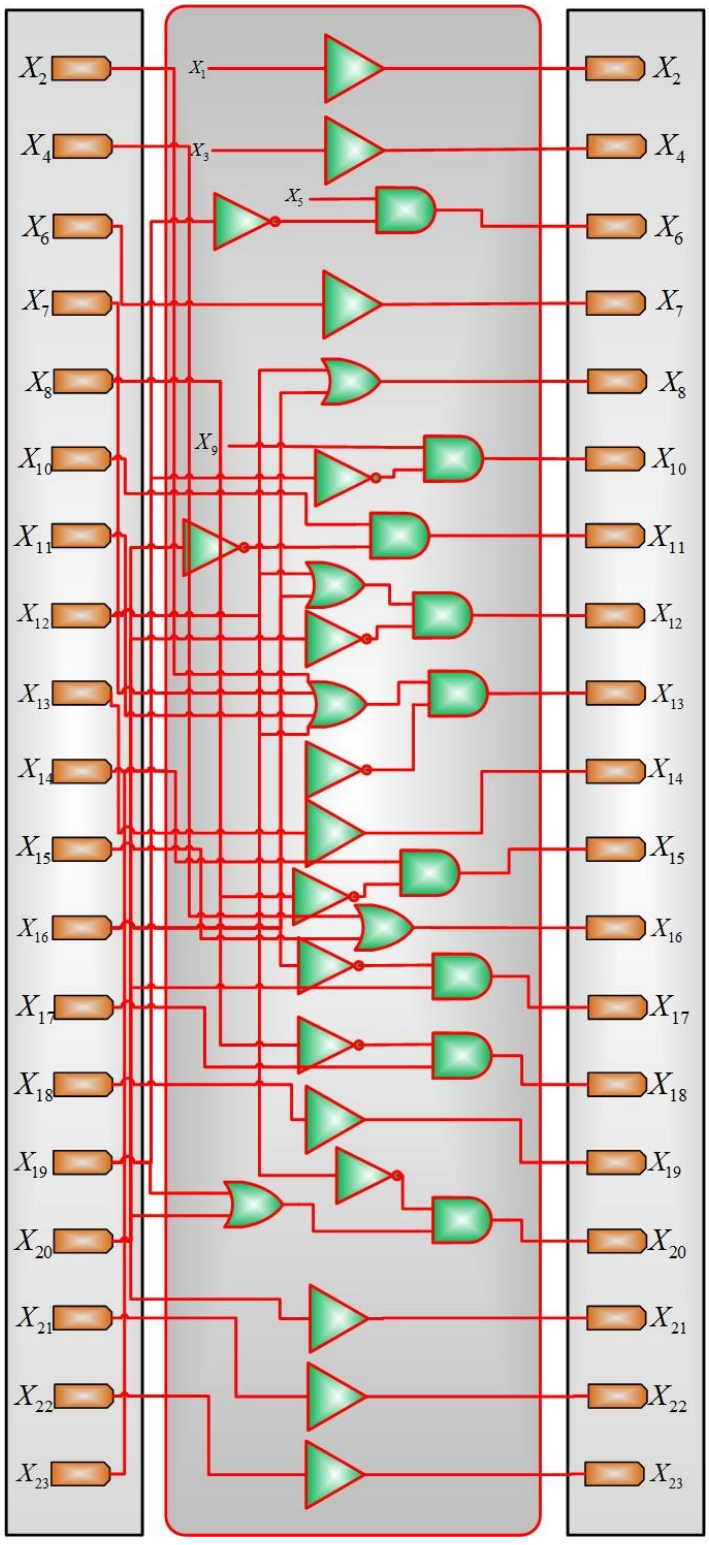


Fig. 5.8. A deterministic BN model for the 23-gene T helper network. Here $X_1 = 0$, $X_3 = 0$, $X_5 = 0$ and $X_9 = 0$. X_i, $i \in \{1, 2, \cdots, 23\}$, indicates a gene node in Fig. 5.7.

There is no input for four gene nodes, namely, $IL-12$, $IFN-\beta$, $IL-18$ and TCR, in the T helper network. As in [110], hence, these four elements are considered to be consistently at state 0. The network in **Fig. 5.7** can be implemented by the BN model shown in **Fig. 5.8**.

The cell population may fall into different subgroups on exposure to input stimuli – this is referred to as a cellular differentiation [111]. For the 23-gene T helper network in **Fig. 5.7**, $IL-2$ functions as a key cytokine in the $Th0$ to $Th1$ differentiation process while a large perturbation of $IL-4$ forces the network to switch from state $Th0$ to $Th2$, as illustrated by **Fig. 5.9**.

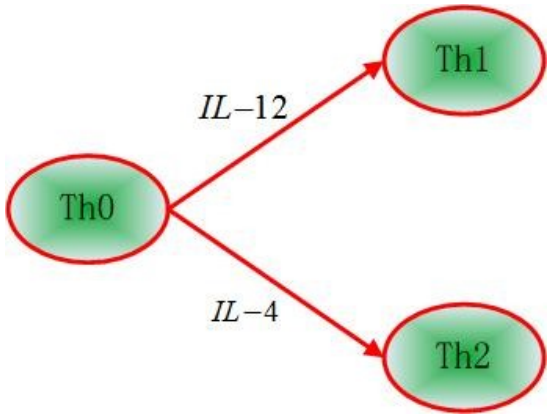


Fig. 5.9. Differentiation of T helper (adapted from [111]). $Th1$ cells express $IFN-\gamma$ and $T-bet$, while $Th2$ expresses $GATA3$ and $IL-4$.

In the state space of the T helper network, the basins draining to different attractors are not evenly distributed; the distribution of initial states leading to the attractors is shown in Table 5.1. As revealed in Table 5.1, a random initial state evolves into $Th1$ with the largest probability for each state update strategy. The attractor $Th0$ is reached with the lowest probability. For the asynchronous update strategies, the probability of reaching $Th0$ is even lower than that for the synchronous state update.

The effect of perturbation on different gene nodes has been shown by experiments of the differentiation of T helper cells [112][113]. The stochastic simulations using synchronous and asynchronous update strategies correctly identify the attractors for a

wild type T helper network, in which no perturbation exists. Table 5.2 shows the steady states found by the stochastic approach for several scenarios of the wild type (no gene suppression or overexpression; the initial states are randomly generated with a probability of 0.5), $IL-12$ overexpressed and $IL-4$ overexpressed. These steady states are good estimates of the attractors found in [102][114].

Table 5.1. Steady states of the T helper network found by the stochastic approach.

Perturbed genes	Active genes in steady states								attractors
Wild type	All genes are inactive								$Th0$
	$IFN-\gamma$	$T-bet$	$SOCS1$	$IFN-\gamma R$					$Th1$
	$IL-10$	$IL-10R$	$GATA3$	$STAT3$	$STAT6$	$IL-4$	$IL-4R$		$Th2$
$IL-12$ over expressed	$IFN-\gamma$	$T-bet$	$SOCS1$	$IFN-\gamma R$	$IL-12$	$IL-12R$	$STAT4$		$Th1$
	$IL-10$	$IL-10R$	$GATA3$	$STAT3$	$IL-12$	$STAT6$	$IL-4$	$IL-4R$	$Th2$
$IL-4$ over expressed	$IFN-\gamma$	$T-bet$	$SOCS1$	$IFN-\gamma R$	$IL-4$				$Th1$
	$IL-10$	$IL-10R$	$GATA3$	$STAT3$	$STAT6$	$IL-4$	$IL-4R$		$Th2$

Table 5.2. The distribution of initial states (in percentages) leading to the attractors of a wild type T helper cell: 200,000 states are randomly chosen for simulation.

Wild type	$Th0$ (%)	$Th1$ (%)	$Th2$ (%)
Synchronous	6.86	58.65	36.89
ROG	3.75	50.07	46.19
RMG (m = 10)	2.94	51.42	45.65
RMG (m = 21)	5.32	54.51	40.17
ARO	3.06	50.70	46.25
MRO (m = 10)	3.79	50.21	46.01

In [109], at most one perturbation is considered to occur at a time. In this work, k time frames are considered with the stochasticity models (k = 5 in **Fig. 5.10**). If the initial state of the T helper network is $Th0$, after the constant activation of $IL-12$, the state will transition into $Th1$. As shown in **Fig. 5.10(b)** and **(c)**, the state remains even after the inactivation of $IL-12$ if no perturbation occurs. For the SIN model, the $Th0$

cells derive into $Th1$ and $Th2$, while a few cells remain at their current states. The almost equally-likely cellular differentiation of steady states has been analyzed in [97]. However, it has been shown that a $Th0$ cell cannot transfer into a $Th2$ cell [114]. Hence, the SIN model could be the main reason for this discrepancy, because $Th1$ and $Th2$ are evolved from $Th0$ with nearly equal probability, as revealed in **Fig. 5.10(d)-(f)** for the synchronous and RMG models.

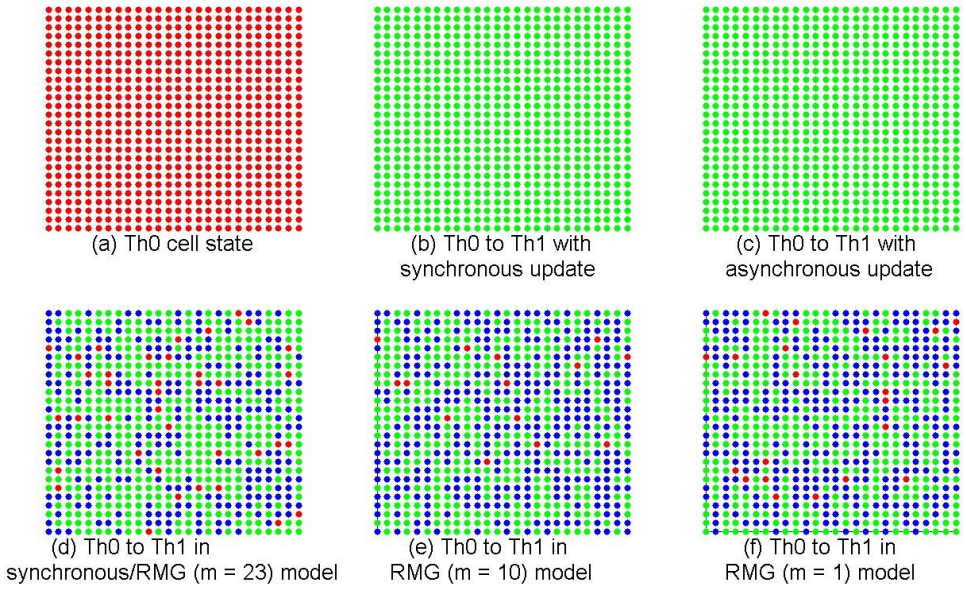


Fig. 5.10. Transitions between attractors during the differentiation process of the T helper network with an external stimulus of $IL-12$ for the synchronous and randomly m genes (RMG) models. Each dot indicates a T helper cell and each cell is independent of the neighboring cells. A red dot represents an undifferentiated cell at $Th0$, a green dot indicates the cell state of $Th1$ and a blue dot indicates the state of $Th2$. (a) All cells are in the initial $Th0$ state. (b) and (c) The transitions from $Th0$ to $Th1$ by synchronous and asynchronous updating rules (without perturbation to the network). (d), (e) and (f) The transitions from $Th0$ for the SIN model with perturbation probability of 0.5 for each node: the percentage of the corresponding nodes indicates the percentage of the cells deriving into an attractor. For the first five time frames, each of the gene nodes is subject to the effect of SIN. The obtained transition probabilities from $Th0$ to different attractors are similar to the values in Table 5.1, because it is equivalent to setting the initial state as the wile type by having each of the nodes under SIN for several time frames.

For the SPP model, the next state of a gene is closely related with the activity of other nodes, the predictor function and the present states. For the SPP model, nearly all

the nodes of $Th0$ differentiate into $Th1$ cells under the dosage of $IL-12$. Similar results can be obtained for the dosage of $IL-4$ and for the other asynchronous update strategies. The instability of $Th0$ is in agreement with the experimental results in [111].

The above simulation results can be reproduced with the network topology, updating functions, the selection probabilities of the updating functions, initial input signal probabilities, and the parameter values related with perturbation and the asynchronous updating strategies.

5.5. Summary

In this chapter, ASBNs are proposed for an effective modeling of GRNs. Various asynchronous updating rules are considered, including the models of ROG, RMG, ARO and MRO. In an ASBN, if the control bits for an AUM are all zeros, an asynchronous network becomes the same as a synchronous update network. Stochastic analyses are carried out for the asynchronous models of SIN and SPP. The expected long run behavior of a GRN is studied by the stochastic approach for various asynchronous state update strategies.

For an asynchronous gene network, an accurate analysis becomes difficult due to the limitation of computational resources, especially when the number of genes or the quantization level of gene states increases. In the stochastic approach, a simulation-based time-frame expansion technique is used for a fast analysis of the SSD of a network. The obtained SSDs can be used to estimate the attractors of, especially, a large network. This is shown by the study of a T helper network of 23 genes. The robustness of the attractors for the T helper network is demonstrated for a number of synchronous and asynchronous state update strategies. The effect of different asynchronous updating rules on the distribution of initial states leading to an attractor is investigated for the T helper network. It is shown that the model of SPP accurately reveal the state transition from $Th0$ to $Th1$ under the dosage of $IL-12$.

Chapter 6

A Stochastic Approach for the Analysis of Fault Trees with Priority AND Gates

In a dynamic fault tree (DFT) analysis, several dynamic gates, including the priority AND (PAND) gate, the sequence enforcing gate (SEQ), the standby or spare gate (Spare), and the functional dependency gate (FDEP) are frequently used to model the dynamic behaviors in a DFT [65][66][115]. In this chapter, a stochastic computational approach is proposed for a fast analysis of the top event's failure probability in a DFT with PAND gates. The results in this chapter have been published in [116].

The novelty of this chapter is as follows:

- A stochastic model is proposed for the analysis of a two-input PAND gate in a DFT. This model is then used in a successive cascading structure for the analysis of a general multiple-input PAND gate. For a DFT with PAND gates, a stochastic approach using the proposed models provides a fast analysis of the DFT with good scalability compared to an accurate or algebraic approach.
- The use of non-Bernoulli sequences of random permutations of fixed numbers of ones and zeros as initial input event probabilities makes the stochastic approach faster and more accurate than Monte Carlo simulation.
- Repeated events are correctly and readily handled in a DFT analysis, because signal correlation is maintained in the random binary bit streams and in the propagation of the stochastic sequences in a fault tree analysis.

6.1. Motivation

Various methodologies using Markov [117][118] and Bayesian [119][120] models have been proposed for evaluating the dependability of a fault tree. Due to the inevitable state-space explosion problem, these approaches incur a large complexity for the analysis

of complex systems. Moreover, the evaluation of a large DFT using a state-space based method becomes difficult when a basic event's failure behavior is not exponentially distributed.

In [39], an Inclusion-Exclusion method is proposed for an exact analysis of a DFT that contains PAND gates and repeated events. However, this method is limited to the analysis of systems with exponentially distributed failure events; in addition, detailed information on the minimal cut set is usually required in advance. In [40], an integral-based analysis is proposed for handling any probability distribution; however, an analytical expression is generally difficult to derive as a function of the basic events' failure distributions. Several approaches have been developed to simplify the process of deriving an exact analytical expression. These approaches include those using binary decision diagrams (BDDs) [121], sequential binary decision diagrams (SBDDs) [70][122], and an algebraic analysis [115][123]. In particular, the SBDD approach has been applied to the analysis of the PAND gate [124]. Monte Carlo (MC) simulation [125][126] has been widely used to evaluate complex DFTs; however, a long run time and a large sample size are needed to meet an accuracy requirement. Generally, it is challenging to accurately evaluate the reliability of a DFT with dynamic gates such as PAND gate. A stochastic approach has been proposed in [127] for the evaluation of a system's reliability. In particular, the serial and parallel implementations of stochastic computation are considered, and a speedup in analysis is obtained by a parallel implementation in field programmable gate arrays (FPGAs). In [127], the PAND gate is modeled as a three-input AND gate, and a sequential event is considered to be a basic event. In a general case, however, the input of a PAND gate is not limited to a basic event. Thus, improved stochastic models can be developed for a fast analysis of DFTs.

6.2. Background

The background presented here is also applicable to the DFT analysis in Chapters 7 and 8.

6.2.1. Assumptions

Some assumptions are as follows:

(1) For the PAND gate, the failure of the gate occurs if the inputs fail in a predetermined order [37][71]. Without the loss of generality, the predefined order is assumed to be from left to right in this dissertation, unless otherwise noted.

(2) The quantization level of a basic event is denoted by a binary variable x, $x \in \{0, 1\}$, with 0 indicating no fault;

(3) All basic events are fault-free at the beginning of the mission time;

(4) The basic events are non-repairable [37]. This means that if a basic event fails, the variable that indicates the status of the basic event, takes 1. Let $Ft(a)$ be the failure time of a basic event a; the status variable of a is 1 for time $t > Ft(a)$ and 0 otherwise. A generic timing diagram for a non-repairable basic event a is shown in **Fig. 6.1**.

(5) The probability density function (*pdf*) and cumulative density function (*cdf*) of an exponential distribution are given by:

$$f(t) = \lambda\, e^{-\lambda t}, \tag{6.1}$$

and

$$F(t) = \int_0^t f(t)\, dt = 1 - \, e^{-\lambda t}, \tag{6.2}$$

where t is the specified mission time and λ is the (constant) failure rate of a basic event for an exponential distribution.

(6) The failure probability of a basic event in a selected time interval $[t_i, t_i + \Delta t]$ is considered constant at the value in the beginning of the time interval, i.e., the failure

probability is given by $p = F(t_i)$ for any time in this time interval. For simplicity, the time interval $[t_i, t_i + \Delta t]$ is referred to as time i in Chapters 6 and 7.

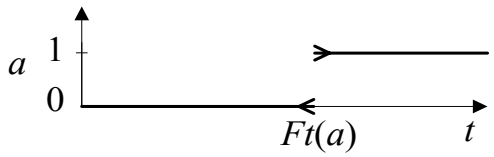


Fig. 6.1. A timing diagram for a non-repairable basic event a [37][115], where a value of 0 indicates no fault, while 1 means the event has failed and $Ft(a)$ is the failure time of the basic event a.

6.2.2. Discretization

Assume that the mission time t is divided into M equal time intervals, i.e., $\Delta t = t/M$. Due to the nature of discretization, a failure probability is estimated more precisely at time t with a larger M. However, a longer run time is required as more stochastic sequences need to be generated. Hence, M is determined by a tradeoff between accuracy and efficiency. With a reasonable M, the discretization provides a relatively accurate estimation of the continuous failure probability of a basic event.

6.2.3. Generation of Non-Bernoulli Sequences

Assume that the failure probabilities for the two adjacent time intervals, $[t_i - \Delta t, t_i]$, and $[t_i, t_i + \Delta t]$, are given by $F(t_i - \Delta t)$, and $F(t_i)$ respectively. If we use non-Bernoulli sequences of L bits, as a random permutation of a fixed number of zeros and ones, then the number of ones in these sequences for the two probabilities are given by:

$$\begin{cases} N(t_i - \Delta t) = L \cdot F(t_i - \Delta t), \\ N(t_i) = L \cdot F(t_i). \end{cases} \tag{6.3}$$

The difference of the number of ones is then:

$$\Delta N = N(t_i) - N(t_i - \Delta t) = L \cdot [F(t_i) - F(t_i - \Delta t)]. \quad (6.4)$$

Further assume that the non-Bernoulli sequence for the probability in $[t_i - \Delta t, t_i]$ is given by $S(t_i - \Delta t)$. Then the sequence $S(t_i)$ for the probability in $[t_i, t_i + \Delta t]$ can be obtained by randomly assigning ΔN ones to replace the zeros in $S(t_i - \Delta t)$. Because the ones in $S(t_i - \Delta t)$ are a subset of those in $S(t_i)$, we obtain:

$$S(t_i - \Delta t)\ AND\ S(t_i) = S(t_i - \Delta t). \quad (6.5)$$

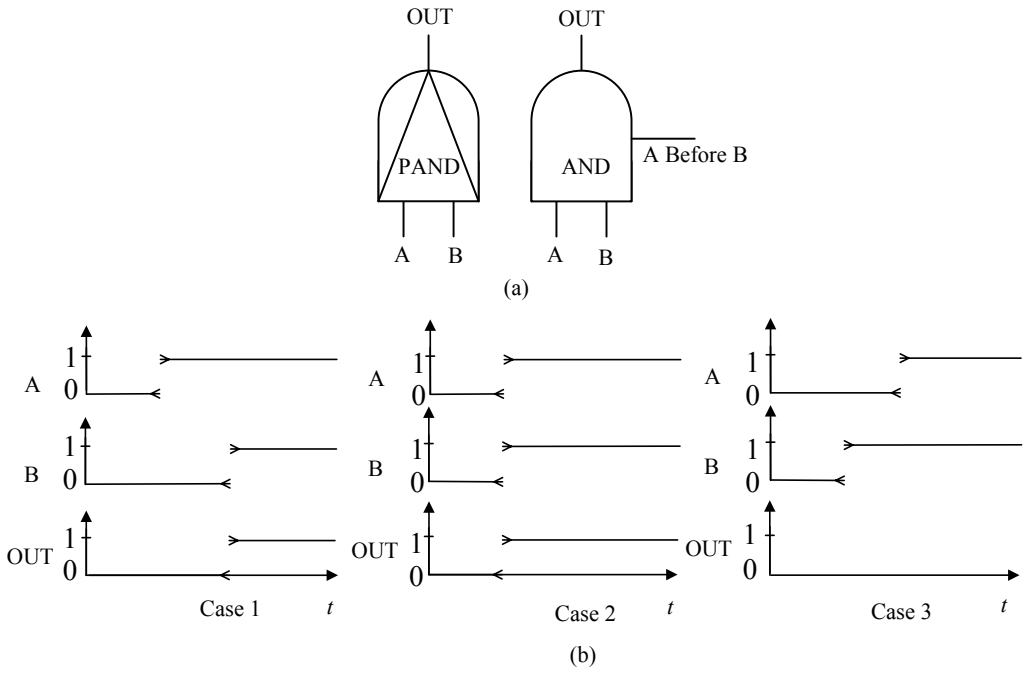


Fig. 6.2. (a) Symbols for a two-input priority AND (PAND) gate [37][71]; (b) The expected behaviour of the two-input PAND gate for an inclusive condition (adapted from [115]) where 1 and 0 indicate a faulty, and fault-free event respectively.

6.3. Priority AND Gate

A PAND gate is a special type of AND gate for which an input indicates the firing of a basic event that occurs in a predetermined order, and the output indicates whether a failure occurs [37][71]. The operational principles of a two-input PAND gate, as well as its symbols, are shown in **Fig. 6.2** for an inclusive condition [115]. By an inclusive

condition, if the two inputs of the PAND gate fail simultaneously, the output fails at the same time as the inputs.

As shown in **Fig. 6.2**, the output of the PAND gate is 1 (i.e., it fails) when the basic event A fails before B or A and B fail at the same time; otherwise, the output of the PAND gate is 0, i.e., fault free. Let $Ft(A)$ and $Ft(B)$ be the failure time of basic events A and B respectively; the failure time of the PAND gate's output, $Ft(OUT)$, is given by:

$$Ft(OUT) = \begin{cases} Ft(B), & if\ Ft(A) < Ft(B) \\ Ft(A)\ or\ Ft(B), & if\ Ft(A) = Ft(B) \\ \infty, & if\ Ft(A) > Ft(B) \end{cases} \tag{6.6}$$

6.4. Stochastic Priority AND Model

6.4.1. A Two-Input Priority AND Gate Model

Let A_{i-1}, and B_{i-1} respectively indicate the states of basic events A, and B at time $i-1$, and A_i, and B_i for the states at time i. If both A and B fail at time i, i.e., $A_{i-1}B_{i-1} = 00$ and $A_iB_i = 11$, then the failure time of the basic events A and B is given by:

$$Ft(A) = Ft(B) = i \cdot \Delta t. \tag{6.7}$$

Then, the failure time of the PAND gate's output is given by $Ft(OUT) = Ft(A) = Ft(B) = i \cdot \Delta t$, due to the model considered in Case 2 in **Fig. 6.2(b)**.

If $A_{i-1}B_{i-1} = 10$ and $A_iB_i = 11$, the basic event B fails at time i while A fails before time i. The failure time of the basic event B is then:

$$Ft(B) = i \cdot \Delta t. \tag{6.8}$$

The relationship between the failure times of the basic events A and B is given by:

$$Ft(A) < Ft(B). \tag{6.9}$$

Thus, $Ft(OUT) = Ft(B)$, due to (6.6) and the model considered in Case 1 in **Fig. 6.2(b)**.

For the other possible scenario, i.e., the basic event A fails after B, the top event of the PAND gate would not fail, i.e., with a failure time of infinity, due to the model considered in Case 3 in **Fig. 6.2(b)**.

Because the basic events are non-repairable, the state of the two-input PAND gate's output event is affected by the gate's output at the previous time, hence the output of the PAND gate at time i, OUT_i, is determined by three factors:

1) The current states of the input basic events A and B at time i, A_i and B_i;

2) The inverted state of basic event B at time $i-1$, $NOT(B_{i-1})$; and

3) The output of the PAND gate at time $i-1$, OUT_{i-1}.

Hence, the output of the PAND gate at time i is given by:

$$OUT_i = OUT_{i-1} + A_i \cdot B_i \cdot NOT(B_{i-1}) \tag{6.10}$$

A stochastic logic model can be constructed to determine the failure of the two-input PAND gate, as shown in **Fig. 6.3**.

According to the assumptions in Section 6.2.1, all basic events are fault free at the beginning of the mission time; thus, the input signals of the model in **Fig. 6.3** are zeros. In **Fig. 6.3(a)**, if the basic event A fails before time i, $E_i = 1$ if $B_{i-1} = 0$, and $B_i = 1$. Then, at time i, $OUT_i = 1$. However, if $OUT_{i-1} = 1$, which indicates that A fails

before B or both events fail simultaneously at time $i-1$, then $B_{i-1}=1$, and $E_i=0$. Because OUT_{i-1} and E_i cannot be 1 at the same time, either $OUT_{i-1}=1$ or $E_i=1$ results in $OUT_i=1$. Otherwise, the state of the top event remains zero. From this analysis, it can be seen that the stochastic PAND model in **Fig. 6.3** computes (6.10), thus it accurately implements the function of the PAND gate.

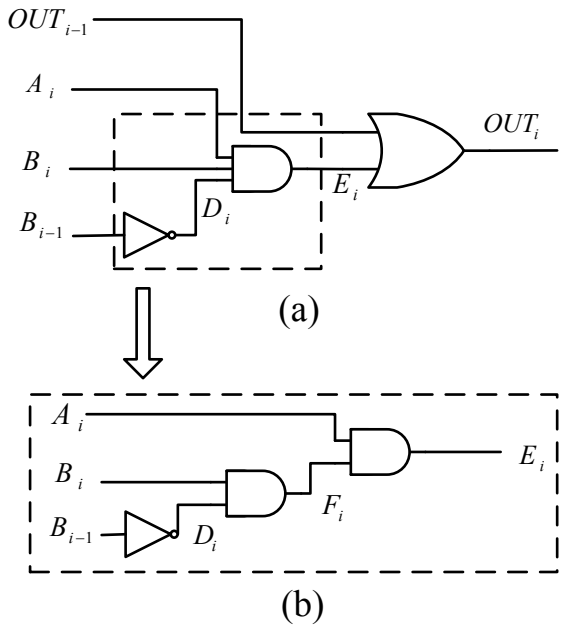


Fig. 6.3. (a) A stochastic logic model for a two-input priority AND (PAND) gate, and (b) the decomposition of the three-input AND gate in (a) into two-input AND gates.

6.4.2. Model Validation

To validate the proposed stochastic PAND model, the discretization of a continuous probability distribution and the generation of stochastic non-Bernoulli sequences are introduced next, followed by a theoretical proof.

6.4.2.1. Stochastic Model Validation

*Theorem 1: Compared to an accurate analysis method, a stochastic simulation of the two-input PAND gate model in **Fig. 6.3**, using large non-Bernoulli sequences of random permutations of fixed numbers of ones and zeros as initial input probabilities,*

produces the same increment in the failure probabilities of two adjacent time intervals when $\lambda\Delta t \to 0$.

Proof: Assume that the failure probabilities of the PAND gate at time i, and $i-1$ are given by $F((A \to B)_i)$, and $F((A \to B)_{i-1})$, respectively; we show that the failure probability of E_i in the stochastic model in **Fig. 6.3** is the same as the increment in the output failure probability of the PAND gate from time $i-1$ to i, i.e., $F(E_i) = F((A \to B)_i) - F((A \to B)_{i-1})$.

Given the basic events A, and B with the *pdfs* $f_A(t)$, and $f_B(t)$ respectively, the failure probability for the two-input PAND gate's output OUT (when both A and B fail or A fails before B, i.e., $A \to B$), is given by:

$$F(A \to B) = \int_0^t \int_{t_1}^t f_B(t_2) f_A(t_1) dt_2\, dt_1. \tag{6.11}$$

For an exponential distribution, (6.11) becomes:

$$F(A \to B) = \int_0^t \int_{t_1}^t \lambda_B e^{-\lambda_B t_2} \lambda_A e^{-\lambda_A t_1} dt_2\, dt_1, \tag{6.12}$$

which leads to the failure probability of the sequential event $A \to B$ as:

$$F(A \to B) = \frac{\lambda_A}{(\lambda_A+\lambda_B)}(1 - e^{-(\lambda_A+\lambda_B)t}) - e^{-\lambda_B t}(1 - e^{-\lambda_A t}). \tag{6.13}$$

Equation (6.13) can be obtained by using an analytical approach [40] or a probabilistic algebraic analysis [115].

By discretization, the failure probabilities of the sequential event $A \to B$ at time i, and $i-1$ are given by:

$$F((A \to B)_i) = \frac{\lambda_A}{(\lambda_A+\lambda_B)}(1 - e^{-(\lambda_A+\lambda_B)\cdot i\cdot \Delta t}) - e^{-\lambda_B \cdot i \cdot \Delta t}(1 - e^{-\lambda_A \cdot i \cdot \Delta t}), \tag{6.14}$$

and

$$F((A \to B)_{i-1}) = \frac{\lambda_A}{(\lambda_A+\lambda_B)}\left(1 - e^{-(\lambda_A+\lambda_B)\cdot (i-1)\cdot \Delta t}\right) - e^{-\lambda_B \cdot (i-1)\cdot \Delta t} +$$

$$e^{-(\lambda_A+\lambda_B)\cdot(i-1)\cdot\Delta t}, \tag{6.15}$$

respectively. Equation (6.14) can also be written as:

$$F((A \to B)_i) = \frac{\lambda_A}{(\lambda_A+\lambda_B)}\left(1 - e^{-(\lambda_A+\lambda_B)\cdot(i-1)\cdot\Delta t} \cdot e^{-(\lambda_A+\lambda_B)\cdot\Delta t}\right) - e^{-\lambda_B\cdot(i-1)\cdot\Delta t} \cdot e^{-\lambda_B\cdot\Delta t} + e^{-(\lambda_A+\lambda_B)\cdot(i-1)\cdot\Delta t} \cdot e^{-(\lambda_A+\lambda_B)\cdot\Delta t}. \tag{6.16}$$

Because $O\left((\lambda \cdot \Delta t)^i\right)$ for any $i \geq 2$ is negligible when $\lambda\Delta t \to 0$, applying a Taylor series expansion on (6.16) leads to:

$$F((A \to B)_i) = \left[\frac{\lambda_A}{(\lambda_A+\lambda_B)}\left(1 - e^{-(\lambda_A+\lambda_B)\cdot(i-1)\cdot\Delta t}\right) - e^{-\lambda_B\cdot(i-1)\cdot\Delta t} + e^{-\lambda_B\cdot(i-1)\cdot\Delta t} \cdot \lambda_B \cdot \Delta t + e^{-(\lambda_A+\lambda_B)\cdot(i-1)\cdot\Delta t} - e^{-(\lambda_A+\lambda_B)\cdot(i-1)\cdot\Delta t} \cdot \lambda_B \cdot \Delta t\right]. \tag{6.17}$$

From (6.15) and (6.17), the probability increment for two adjacent times is obtained as:

$$F(OUT_i) - F(OUT_{i-1}) = F((A \to B)_i) - F((A \to B)_{i-1}) = \lambda_B \cdot \Delta t \cdot e^{-\lambda_B\cdot(i-1)\cdot\Delta t} \cdot (1 - e^{-(\lambda_A)\cdot(i-1)\cdot\Delta t}). \tag{6.18}$$

Next, the stochastic analysis of the increased probability E_i between two adjacent time intervals is pursued. Let $F_c(\cdot)$, and $F_d(\cdot)$ indicate the *cdfs* for the continuous, and discretized distributions respectively. By applying the discretization process to the exponential distributions of the basic events (i.e., A and B), we have:

$$\begin{cases} F_c(A) = \int_0^t f_A(t)\, dt = 1 - e^{-\lambda_A\cdot t}, \\ F_d(A) = 1 - e^{-\lambda_A\cdot M\cdot\Delta t}, \end{cases} \tag{6.19}$$

and

$$\begin{cases} F_c(B) = \int_0^t f_B(t)\, dt = 1 - e^{-\lambda_B\cdot t}, \\ F_d(B) = 1 - e^{-\lambda_B\cdot M\cdot\Delta t}, \end{cases} \tag{6.20}$$

where M is the number of equally discretized time intervals Δt.

Hence, the input probabilities of A, and B at time i and $i-1$ are given by:

$$F(A_i) = \left(1 - e^{-\lambda_A \cdot i \cdot \Delta t}\right), \tag{6.21}$$

$$F(B_i) = \left(1 - e^{-\lambda_B \cdot i \cdot \Delta t}\right), \tag{6.22}$$

$$F(B_{i-1}) = \left(1 - e^{-\lambda_B \cdot (i-1) \cdot \Delta t}\right). \tag{6.23}$$

Let $S(A_i)$ be the stochastic sequence generated for the probability of the basic event A at time i; $S(B_i)$, and $S(B_{i-1})$ be the stochastic sequences for the basic event at time i, and $i-1$ respectively. In the model of **Fig. 6.3**, the inverter's output sequence, $S(D_i)$, is given by:

$$S(D_i) = NOT(S(B_{i-1})). \tag{6.24}$$

For the three-input AND gate in **Fig. 6.3(a)**, its output sequence is obtained as:

$$S(E_i) = S(A_i)\ AND\ S(B_i)\ AND\ S(D_i) = \tag{6.25}$$
$$S(A_i) AND\ (S(B_i) AND\ (NOT\ (S(B_{i-1})))).$$

Similar to (6.5), the probability encoded in the sequence $S(B_i) AND\ (NOT\ (S(B_{i-1})))$ is given by $F(B_i) - F(B_{i-1})$, i.e., the probability increment for the basic event B in two adjacent times.

By (6.22) and (6.23), this probability increment is thus:

$$F(B_i) - F(B_{i-1}) = e^{-\lambda_B \cdot (i-1) \cdot \Delta t} - e^{-\lambda_B \cdot i \cdot \Delta t}. \tag{6.26}$$

Considering $F(E_i)$ as the probability encoded in the sequence $S(E_i)$, together with (6.21) and (6.26), the probability increment in E_i is given by:

$$F(E_i) = F(A_i)(F(B_i) - F(B_{i-1})) = (1 - e^{-\lambda_A \cdot i \cdot \Delta t}) \cdot \left(e^{-\lambda_B \cdot (i-1) \cdot \Delta t} -\right.$$

$e^{-\lambda_B \cdot i \cdot \Delta t}\big)$. (6.27)

The application of a Taylor series expansion on (6.27) leads to a first-order approximation given by (6.18). This result shows that the proposed stochastic model accurately implements the function of a two-input PAND gate for exponentially distributed events, i.e.,

$$F(E_i) = F((A \to B)_i) - F((A \to B)_{i-1}). \quad (6.28)$$

Next, the proof of the theorem is pursued in the general case when the basic events are non-exponentially distributed. By an integral analysis, the failure probability of the two input PAND gate at time t is given by:

$$F((A \to B)_t) = \int_0^t \int_{t_1}^t f_B(t_2) f_A(t_1) dt_2\, dt_1 = \int_0^t \big(F_B(t) - F_B(t_1)\big) f_A(t_1) dt_1 = F_B(t) \int_0^t f_A(t_1)\, dt_1 - \int_0^t F_B(t_1)\, f_A(t_1) dt_1. \quad (6.29)$$

Similarly, this failure probability at time $t - \Delta t$ is given by:

$$F((A \to B)_{t-\Delta t}) = \int_0^{t-\Delta t} \int_{t_1}^{t-\Delta t} f_B(t_2) f_A(t_1) dt_2\, dt_1 = \int_0^{t-\Delta t} \big(F_B(t - \Delta t) - F_B(t_1)\big) f_A(t_1) dt_1 = F_B(t - \Delta t) \int_0^{t-\Delta t} f_A(t_1)\, dt_1 - \int_0^{t-\Delta t} F_B(t_1)\, f_A(t_1) dt_1 \quad (6.30)$$

The increment of the failure probabilities between t and $t - \Delta t$ is then

$$
\begin{aligned}
&F((A \to B)_t) - F((A \to B)_{t-\Delta t}) \\
&\quad = F_B(t) \int_0^t f_A(t_1)\, dt_1 - \int_0^t F_B(t_1)\, f_A(t_1) dt_1 \\
&\quad - F_B(t-\Delta t) \int_0^{t-\Delta t} f_A(t_1)\, dt_1 + \int_0^{t-\Delta t} F_B(t_1)\, f_A(t_1) dt_1 \\
&\quad = F_B(t) \int_0^{t-\Delta t} f_A(t_1)\, dt_1 + F_B(t) \int_{t-\Delta t}^{t} f_A(t_1)\, dt_1 \\
&\quad - F_B(t-\Delta t) \int_0^{t-\Delta t} f_A(t_1)\, dt_1 - \int_{t-\Delta t}^{t} F_B(t_1)\, f_A(t_1) dt_1
\end{aligned} \tag{6.31}
$$

When $\Delta t \to 0$, we have

$$\int_{t-\Delta t}^{t} F_B(t_1)\, f_A(t_1) dt_1 = \lim_{\Delta t \to 0}\{F_B(t-\Delta t) f_A(t-\Delta t)\Delta t\}, \tag{6.32}$$

and

$$\int_{t-\Delta t}^{t} f_A(t_1)\, dt_1 = \lim_{\Delta t \to 0}\{f_A(t-\Delta t)\Delta t\}. \tag{6.33}$$

In this case, (6.31) becomes

$$
\begin{aligned}
&\lim_{\Delta t \to 0}\{F((A \to B)_t) - F((A \to B)_{t-\Delta t})\} = \\
&\lim_{\Delta t \to 0}\{\big(F_B(t) - F_B(t-\Delta t)\big) \int_0^{t-\Delta t} f_A(t_1)\, dt_1 + F_B(t) f_A(t-\Delta t)\Delta t - \\
&F_B(t-\Delta t) f_A(t-\Delta t)\Delta t\}.
\end{aligned} \tag{6.34}
$$

When $\Delta t \to 0$,

$$
\begin{aligned}
&F((A \to B)_t) - F((A \to B)_{t-\Delta t}) \\
&\quad = \big(F_B(t) - F_B(t-\Delta t)\big)\left(\int_0^{t-\Delta t} f_A(t_1)\,dt_1 + \int_{t-\Delta t}^{t} f_A(t_1)\,dt_1\right) \\
&\quad = \big(F_B(t) - F_B(t-\Delta t)\big)\int_0^{t} f_A(t_1)\,dt_1
\end{aligned}
\tag{6.35}
$$

Because $F_A(t) = \int_0^t f_A(t_1)\,dt_1$, we obtain

$$
F((A \to B)_t) - F((A \to B)_{t-\Delta t}) = \big(F_B(t) - F_B(t-\Delta t)\big)F_A(t). \tag{6.36}
$$

The right hand side of (6.35) is the failure probability increment computed by the stochastic model of PAND in **Fig. 6.3**. This result proves *Theorem* 1 in the general case. □

6.4.2.2. Analysis of the Increment in Failure Probability

If $S(B_i)$, and $S(B_{i-1})$ are the non-Bernoulli sequences for the failure probabilities of the basic event B, $F(B_i)$, and $F(B_{i-1})$, at time i, and $i-1$ respectively, the mean number of elements equal to 1 in the non-Bernoulli sequence $S(B_{i-1})$ of L bits is then $L \cdot F(B_{i-1})$, and the variance is 0 (by the nature of the non-Bernoulli sequence). This result indicates that the use of non-Bernoulli sequences results in a deterministic initial value. Because there is no variation in the input signal of the inverter, the variance in the inverter's output sequence $S(D_i)$ is 0 as $S(D_i) = NOT(S(B_{i-1}))$. Hence, the mean and variance of the number of ones in the sequence $S(D_i)$ are given by:

$$
\mu = L \cdot (1 - F(B_{i-1})), \tag{6.37}
$$

and

$$
v = 0, \tag{6.38}
$$

respectively. In **Fig. 6.3(b)**, the first AND gate's output sequence $S(F_i)$ is given by

$S(F_i) = AND(S(B_i), NOT(S(B_{i-1})))$, where $S(B_i)$ is statistically dependent on $S(B_{i-1})$, as discussed previously. The mean, and variance of the number of elements equal to 1 in the first AND gate's output sequence are then given by:

$$\mu' = L \cdot (F(B_i) - F(B_{i-1})), \tag{6.39}$$

and

$$v' = 0, \tag{6.40}$$

respectively. (6.39) and (6.40) indicate that $S(F_i)$ is also a non-Bernoulli sequence.

For the basic event A, a non-Bernoulli sequence at time i, $S(A_i)$, is generated for the failure probability $F(A_i)$. For the last AND gate in **Fig. 6.3(b)**, the input sequences $S(F_i)$ and $S(A_i)$ are for two statistically independent signals. Per *Lemma* 1, therefore, the use of non-Bernoulli sequences produces a more accurate result at the output of the last AND gate in **Fig. 6.3(b)**, and thus at the output of the three-input AND gate in **Fig. 6.3(a)**, than using Bernoulli sequences.

If the expected probability of E_i is given by $z = N(E_i)/L$, where $N(E_i)$ indicates the number of ones in the sequence $S(E_i)$, through a combinatorial analysis and the application of Stirling's formula [128][129], the number of elements equal to 1 in the output stochastic sequence $S(E_i)$ of L bits follows approximately a Gaussian distribution, i.e.,

$$F(z) \sim \frac{1}{\sqrt{2\pi L}} \sqrt{\beta} e^{-\theta L}, \tag{6.41}$$

where

$$\beta \sim \frac{1}{F(A_i)(1 - F(A_i))(F(B_i) - F(B_{i-1}))(1 - (F(B_i) - F(B_{i-1})))}, \tag{6.42}$$

$$\theta \sim \frac{(z - F(A_i)(F(B_i) - F(B_{i-1})))^2}{2F(A_i)(1-F(A_i))(F(B_i)-F(B_{i-1}))(1-(F(B_i)-F(B_{i-1})))}, \tag{6.43}$$

as well as with a mean, and variance given by $L \cdot F(A_i)(F(B_i) - F(B_{i-1}))$ and $L \cdot F(A_i)(1 - F(A_i))(F(B_i) - F(B_{i-1}))(1 - (F(B_i) - F(B_{i-1})))$ respectively.

6.4.2.3. Generalization of the PAND Model

A multiple-input PAND gate can be converted to a successively cascaded model of two-input PAND gates. Take a three-input PAND as an example, as shown in **Fig. 6.4(a)**; its cascaded model is shown in **Fig. 6.4(b)**. Assume that the failure order of the three inputs is from left to right, i.e., $A \rightarrow B \rightarrow C$. Then, if the failures of the input events occur in this order, the output G is 1; otherwise, G is 0.

In the cascaded model in **Fig. 6.4(b)**, a 1 at the gate output G indicates that the intermediate event D fails before the basic event C, or both D and C fail at the same time. Because $D = 1$ is caused by the fact that the basic event A fails before B, or both A and B fail at the same time, the gate output $G = 1$ means that the sequential event $A \rightarrow B \rightarrow C$ occurs; thus the cascaded model implements the function of a three-input PAND gate. This model can be generalized for an arbitrary multiple-input PAND gate.

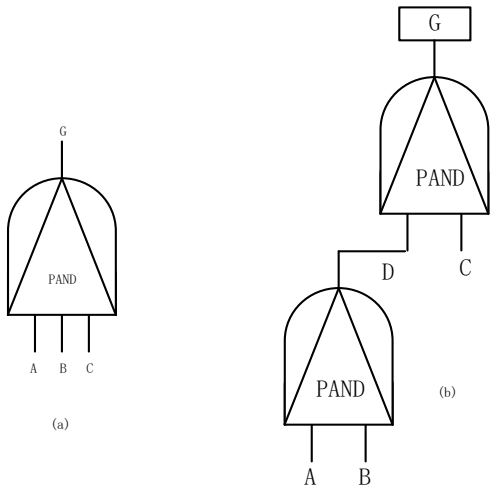


Fig. 6.4. (a) A three-input priority AND (PAND) gate, and (b) the successive cascading model of the three-input PAND gate in (a).

In summary, for a DFT with priority relationships, the stochastic two-input PAND model and the successive cascading model can be utilized in an FTA using the non-Bernoulli sequences generated for discretized probabilities of the basic events. The

failure probability of the top event is encoded in the statistics, i.e., the proportion of the number of ones in the output sequence of the stochastic analysis.

6.5. Case Studies and Validation Results

In this section, several case studies are presented to show the accuracy, efficiency and the ability of dealing with repeated basic events of the stochastic PAND model. Simulations are performed for both exponential and non-exponential distributions of basic events. The results are compared with those obtained by using accurate analysis and simulation-based approaches.

6.5.1. Validation of the Stochastic Priority AND Models

Example 6.1. For a two-input PAND gate and a three-input PAND gate, as shown in **Fig. 6.2(a)** and **Fig. 6.4(a)**, the failure probabilities of basic events are assumed to be exponentially distributed, with $\lambda_A = \lambda_B = \lambda_C = 0.01$. The mission time is 300 hours, and the time interval for discretization is one hour, i.e., $\Delta t = 1$ hour.

A quantitative analysis of the two-input PAND gate is first performed using the stochastic PAND model. The results are compared with those obtained by using the Monte Carlo (MC) [125] and analytical [40] methods, as shown in **Fig. 6.5**. In **Fig. 6.5** (and all subsequent figures and tables, wherever applicable), N is the number of simulation runs for the MC method, and L is the sequence length for the stochastic approach. It can be seen that the stochastic approach produces closer results than MC simulation to the accurate analysis.

Because a continuous failure distribution is discretized into M time intervals, the stochastic analysis results in a vector of the failure probability of the top event at every time interval, $\boldsymbol{F} = (F[1], F[2], \dots, F[M])$. Let $\boldsymbol{F}_S$, $\boldsymbol{F}_A$, and $\boldsymbol{F}_{MC}$ denote the failure probability vectors obtained by the stochastic approach, an accurate method [40], and the

MC method [125]. While an accurate result can be obtained by using an SBDD method [70][122], or an algebraic analysis [115], a direct integral method is used in this dissertation for an accurate analysis. Albeit very fast for a simple DFT analysis, such accurate analysis may become cumbersome in the evaluation of large DFTs. Further, let $\Delta \boldsymbol{F}_{MC-A}$ indicate the discrepancies between the failure probability vectors obtained from the MC method in [125] and the accurate analysis in [40], and let $\Delta \boldsymbol{F}_{S-A}$ indicate the discrepancies between the failure probability vectors obtained from the stochastic approach and the accurate analysis in [40]. The three norms, $\|\cdot\|_1$, $\|\cdot\|_2$, and $\|\cdot\|_\infty$, are then used to measure the differences of the failure probability vectors.

The results are shown in Table 6.1 for the two-input PAND gate with various sequence lengths for the stochastic approach. The average run time is also shown for comparing the efficiency. Unless otherwise noted, ten experiments are run in each case study for obtaining the norm values and average run time. As shown in Table 6.1, the smaller norm values and shorter run time indicate that the stochastic analysis using the non-Bernoulli sequences is more accurate and faster than the MC method.

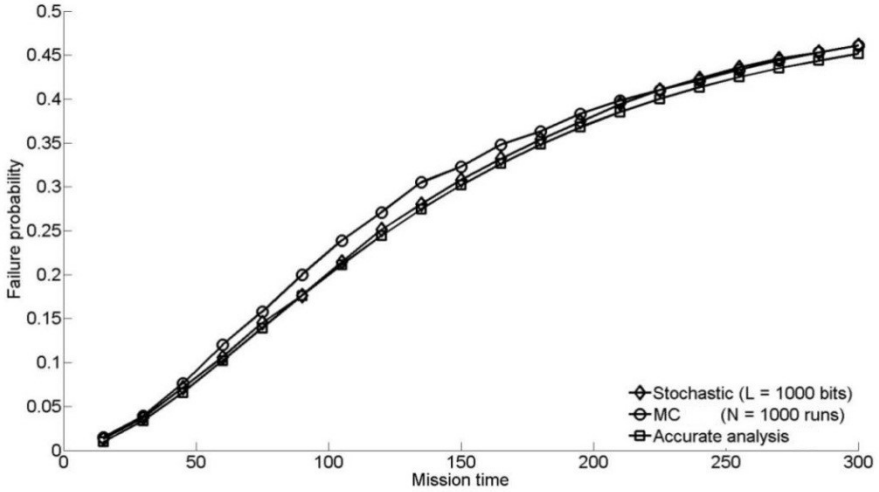


Fig. 6.5. The failure probabilities obtained by using the stochastic, Monte Carlo (MC) [125], and accurate [40] methods for the two-input priority AND (PAND) gate in Fig. 6.1(a).

The accuracy of the stochastic approach can further be improved by using longer stochastic sequences. As shown in **Fig. 6.6**, the stochastic approach can produce very close results to the accurate analysis [40] by using a large sequence length (e.g. 100k bits) for the two-input PAND gate.

Table 6.1. Accuracy and run time of the stochastic approach and Monte Carlo (MC) simulation [125], compared to an accurate analysis [40], for the two-input priority AND (PAND) gate in Fig. 6.1(a).

	N/L	1k	5k	10k	100k
MC [125] vs. Accurate analysis [40]	$\|\Delta \boldsymbol{F}_{MC-A}\|_1$	2.8665	1.4030	1.0888	0.9853
	$\|\Delta \boldsymbol{F}_{MC-A}\|_2$	0.1984	0.0940	0.0729	0.0686
	Avg. time for MC (s)	2.15	11.81	24.61	225.67
The stochastic approach vs. Accurate analysis [40]	$\|\Delta \boldsymbol{F}_{S-A}\|_1$	2.0604	0.9243	0.7085	0.6319
	$\|\Delta \boldsymbol{F}_{S-A}\|_2$	0.1338	0.0597	0.0455	0.0383
	Avg. time for stochastic (s)	3.39×10^{-2}	0.12	0.25	4.73

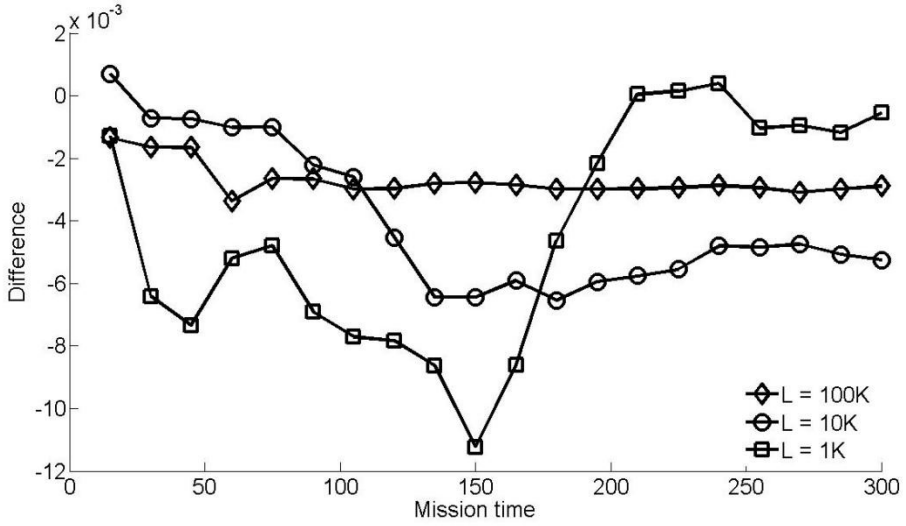


Fig. 6.6. The differences in the failure probability obtained by using the stochastic approach and an accurate analysis at different mission times for the two-input priority AND (PAND) gate.

Next, the failure probability of a three-input PAND gate is evaluated by using the successive cascading PAND model and the stochastic approach. Simulations are run for different sequence lengths, and the obtained failure probability vectors are compared with those obtained by an accurate analysis. As revealed in Table 6.2, the norms of the

differences of the computed failure probability vectors indicate that a stochastic analysis of the PAND model is more accurate and faster than an MC method. As shown in **Fig. 6.7**, moreover, the accuracy of the stochastic approach can further be improved by using longer stochastic sequences.

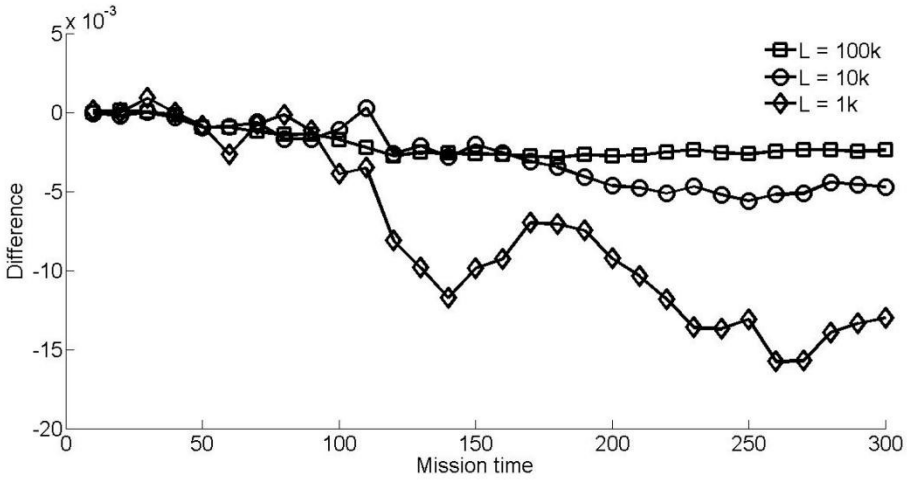


Fig. 6.7. The differences in the failure probability obtained by using the stochastic approach and an accurate analysis at different mission times for the three-input priority AND (PAND) gate.

Table 6.2. Accuracy and run time of the stochastic approach and Monte Carlo (MC) simulation [125], compared to an accurate analysis [40], for the three-input priority AND (PAND) gate in Example 6.1 (b).

Stochastic	L	$\|\Delta \boldsymbol{F}_{S-A}\|_1$	$\|\Delta \boldsymbol{F}_{S-A}\|_2$	$\|\Delta \boldsymbol{F}_{S-A}\|_\infty$	Avg. time (s)
	1k	1.5643	0.1071	0.0121	0.65
	10k	0.6408	0.0434	0.0045	6.88
	100k	0.4827	0.0313	0.0027	66.34
MC	N	$\|\Delta \boldsymbol{F}_{MC-A}\|_1$	$\|\Delta \boldsymbol{F}_{MC-A}\|_2$	$\|\Delta \boldsymbol{F}_{MC-A}\|_\infty$	Avg. time (s)
	1k	2.1041	0.1462	0.0156	5.05
	10k	0.7312	0.0512	0.0055	54.60
	100k	0.5007	0.0324	0.0029	558.07

6.5.2. A Dynamic Fault Tree with Repeated Events

A DFT with PAND gates and repeated events is analyzed next using the stochastic approach.

Example 6.2 (from [39]). A DFT consists of 5 logic gates (4 OR gates, 1 AND gate) and 2 dynamic gates (PANDs) with 9 basic events, as shown in **Fig. 6.8**. The failure rates

of the basic events are exponentially distributed with $\lambda_i = 0.01$ for $i = 1, 2, \ldots, 9$. The basic events e_2 and e_3 are repeated events. The maximum mission time is 300 hours.

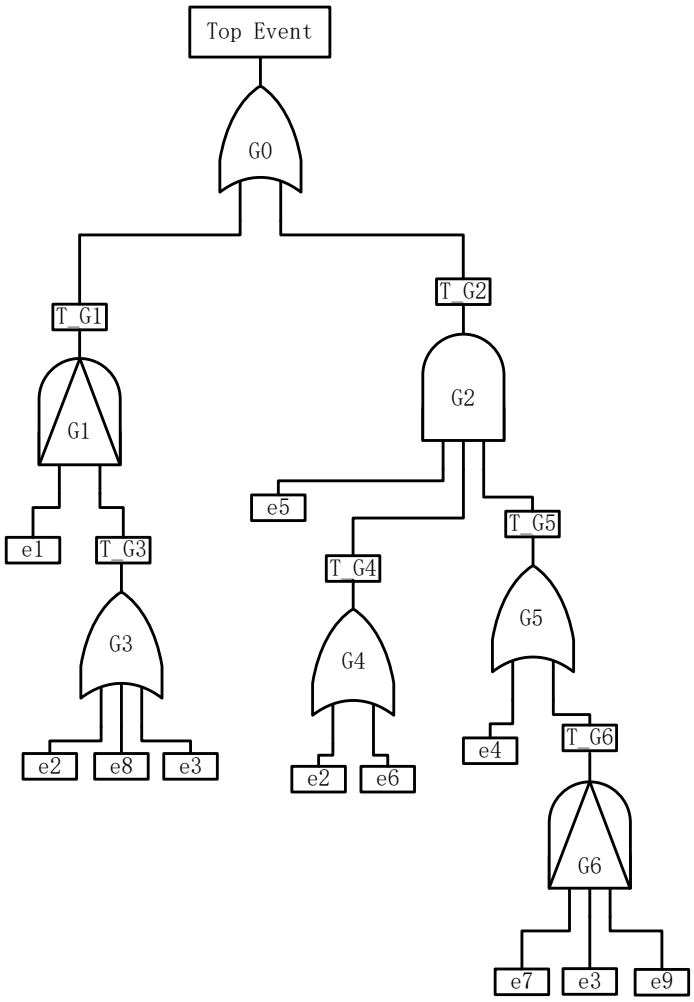


Fig. 6.8. Example 6.2: a dynamic fault tree (DFT) with repeated events e_2 and e_3 [39].

The simulation results by the stochastic approach with different sequence lengths and the MC method [125] with different numbers of simulations are shown in Table 6.3 for several mission times. As can be seen, the stochastic approach computes the failure probability of the top event with a better efficiency than the MC method. This indicates that the stochastic approach using the non-Bernoulli sequences as initial inputs can effectively evaluate the reliability of a dynamic system with repeated events. The accuracy improves with the increase of the length of the stochastic sequences.

Table 6.3. The top event's failure probability of the dynamic fault tree (DFT) in Fig. 6.8, with the total mission time of 300 hours.

t (hour)	Monte Carlo simulation [125] (N)				The stochastic approach (L)			
	1k	5k	10k	100k	1k	5k	10k	100k
50	0.1980	0.2104	0.2125	0.2165	0.2230	0.2134	0.2185	0.2172
100	0.4870	0.4988	0.4868	0.4952	0.5080	0.4954	0.4896	0.4953
150	0.6780	0.6996	0.6809	0.6886	0.7060	0.6786	0.6908	0.6909
200	0.8080	0.8156	0.8070	0.8121	0.8110	0.8118	0.8089	0.8092
250	0.8830	0.8890	0.8832	0.8860	0.8880	0.8878	0.8846	0.8868
300	0.9338	0.9336	0.9318	0.9314	0.9330	0.9328	0.9310	0.9315
Avg. time (s)	25.44	126.66	254.02	2543.30	2.74	13.63	27.03	196.60

6.5.3. DFTs with Non-Exponentially Distributed Events

The presence of a large number of basic events makes it very difficult to derive the top event's failure probability using an accurate analysis, because a large number of states need to be considered, and the complexity of an accurate analysis increases significantly with the number of basic events. It is also difficult to evaluate a PAND gate with intermediate events as inputs. The problem becomes even more challenging when the basic events' failures are not exponentially distributed. In this section, it is shown that these issues are effectively addressed by the stochastic approach, as illustrated by Examples 6.3.

Example 6.3 (from [125]): A DFT consists of a relatively large number of basic events, while the inputs of a PAND gate are two intermediate events, as shown in **Fig. 6.9**.

The failure probability of the top event can be obtained by the algebraic analysis in [115] as:

$$P\{(M \to N)\} = \int_0^t f_N(t_1) \cdot F_M(t_1)\, dt_1, \tag{6.44}$$

where

$$F_N(t_1) = 1 - \prod_{i\epsilon\{H,I,J,K,L\}}(1 - F_i(t_1)), \tag{6.45}$$

$$F_M(t_1) = \prod_{j\epsilon\{A,B,C,D,E\}} F_j(t_1). \tag{6.46}$$

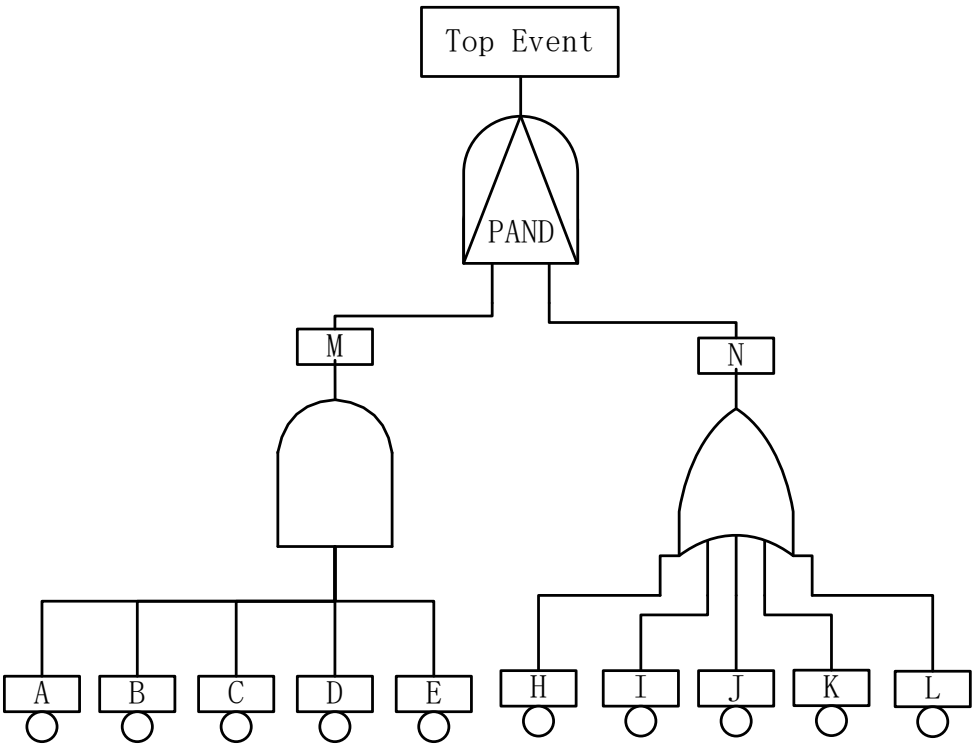


Fig. 6.9. Example 6.3, a dynamic fault tree (DFT) with intermediate events as the inputs of a priority AND (PAND) gate [125].

In a practical system, a non-exponential distribution may be required for a more accurate modeling of a basic event's failure. Although an exact result can be obtained by using an algebraic analysis, it becomes cumbersome for an algebraic analysis to accurately evaluate such systems due to the complexity involved in deriving a closed form of analytical expressions.

In this section, the Weibull distribution is considered to show that a DFT with non-exponentially distributed basic events can be handled by the stochastic approach. The *pdf* and *cdf* of the Weibull distribution is given by

$$f(t) = \frac{\alpha}{\lambda}(\frac{t}{\lambda})^{\alpha-1} e^{-(t/\lambda)^{\alpha}}, \quad (6.47)$$

and

$$F(t) = 1 - e^{-(t/\lambda)^{\alpha}}, \quad (6.48)$$

respectively, where α, and λ are the shape, and scale parameters of the Weibull distribution respectively.

Assume that, in the DFT in **Fig. 6.9**, the basic events J, K, L follow a Weibull distribution with $\alpha = 0.1$ and $\lambda = 20$, while the other basic events are exponentially distributed with failure rates given in Table 6.4 [125].

Table 6.4. The failure rates of the basic events in Example 6.3 [125].

Basic event	Failure rate	Basic event	Failure rate
A	0.011	B	0.012
C	0.013	D	0.014
E	0.015	H	0.0011
I	0.0012		

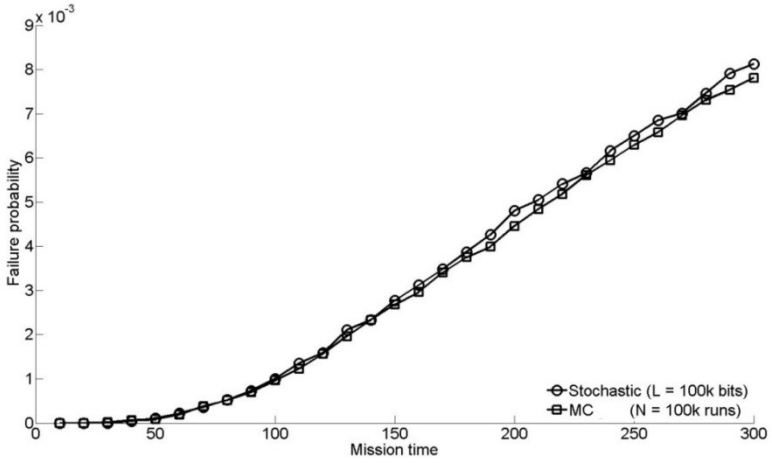


Fig. 6.10. The failure probability of the top event with non-exponentially distributed basic events.

For this system, the failure probability of the top event is plotted for a mission time of 300 hours, as shown in **Fig. 6.9**, for both the stochastic approach and the MC method

[125]. The norms of the differences of the failure probability vectors obtained by the stochastic and MC methods are $\|\cdot\|_1 = 0.0357$, $\|\cdot\|_2 = 0.0028$, and $\|\cdot\|_\infty = 4.5 \times 10^{-4}$. Because the encoding of a failure probability into a stochastic sequence is not limited to those of exponential distributions, a DFT with non-exponentially distributed basic events can be accurately evaluated by the stochastic approach, as shown in **Fig. 6.10**. Hence, the proposed stochastic approach is applicable to both exponential and non-exponential distributions in a DFT analysis.

6.5.4. A Fault Tree with Repeated and Non-Exponentially Distributed Events

Finally, a fault tree without dynamic gates, but with repeated events and non-exponentially distributed ones, is considered. This fault tree is developed from the DFT in **Fig. 6.9** by replacing the PAND gate with an AND gate and inserting a repeated event *E*, as Example 6.4 shown in **Fig. 6.11**. The failure rates of the exponentially-distributed basic events are assumed to be the same as those in Example 6.3, while the non-exponentially distributed events J, K, L follow a Weibull distribution with $\alpha = 0.5$ and $\lambda = 2$.

For this fault tree, the failure probability of the top event is plotted for a mission time of 300 hours, as shown in **Fig. 6.12**, for both the stochastic approach and the MC method [125]. A more detailed comparison is given in Table 6.5.

As revealed in **Fig. 6.11** and Table 6.5, a stochastic analysis of the fault tree is more accurate and faster than an MC method compared with the accurate analysis [40], as shown by the run time and norms of the differences in the failure probability vectors. Hence, a DFT with non-exponentially distributed basic events and repeated events can be evaluated by the stochastic approach.

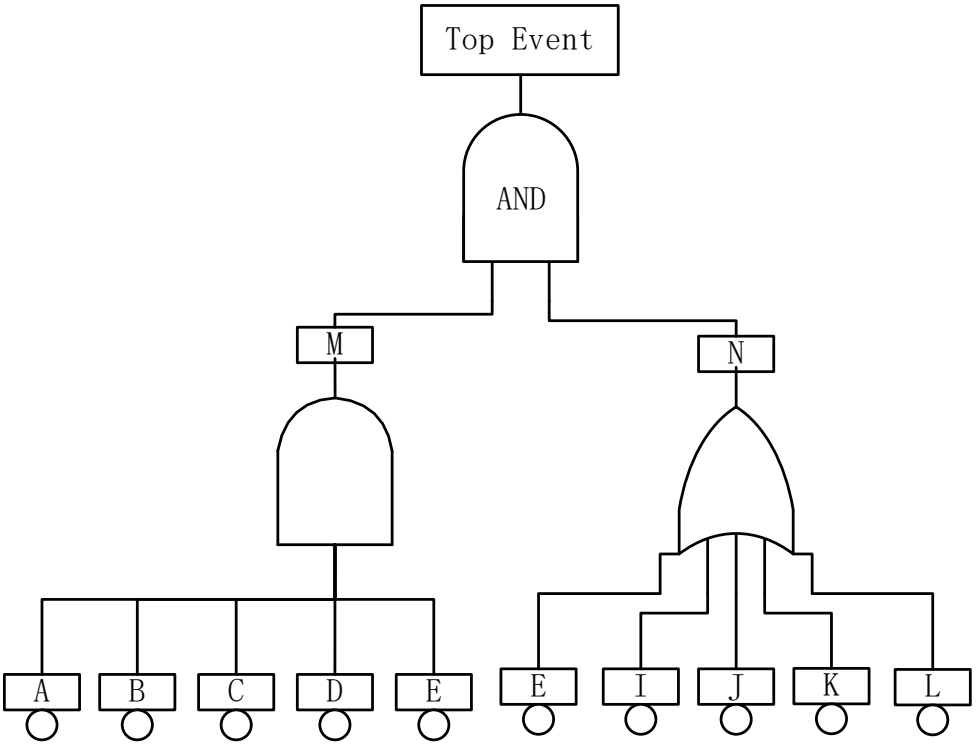


Fig. 6.11. Example 6.4, a fault tree with repeated events and non-exponentially distributed ones.

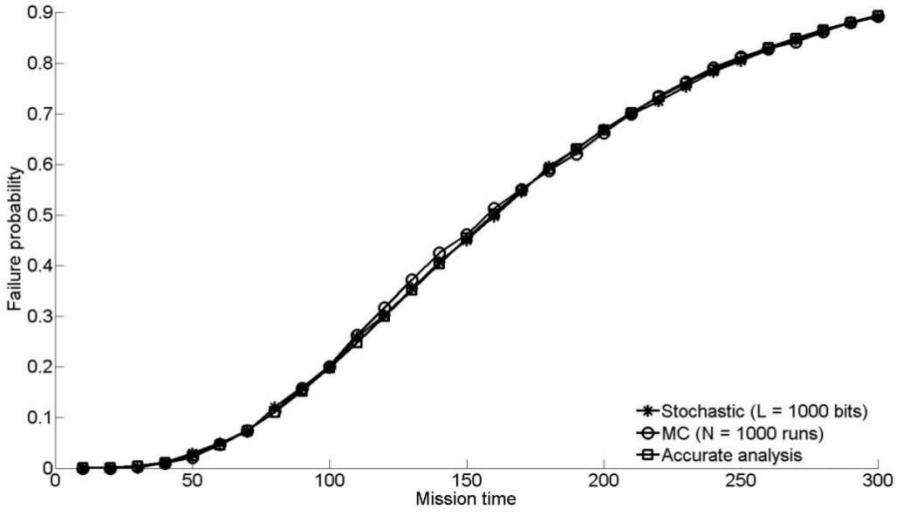


Fig. 6.12. The failure probability of the top event with non-exponentially distributed basic events.

Table 6.5. Accuracy comparison and run time of the stochastic approach and Monte Carlo (MC) simulation [125] for the dynamic fault tree (DFT) in Example 6.4.

Stochastic approach	L	$\|\Delta \boldsymbol{F}_{S-A}\|_1$	$\|\Delta \boldsymbol{F}_{S-A}\|_2$	$\|\Delta \boldsymbol{F}_{S-A}\|_\infty$	Avg. time (s)
	1k	0.9953	0.0747	0.0150	9.47×10^{-2}
	10k	0.4487	0.0348	0.0054	0.75
	100k	0.0967	0.0081	0.0017	8.71
MC simulation	N	$\|\Delta \boldsymbol{F}_{MC-A}\|_1$	$\|\Delta \boldsymbol{F}_{MC-A}\|_2$	$\|\Delta \boldsymbol{F}_{MC-A}\|_\infty$	Avg. time (s)
	1k	1.4800	0.1177	0.0203	8.31
	10k	0.5461	0.0436	0.0081	85.37
	100k	0.1373	0.0123	0.0027	969.38

6.6. Summary

A stochastic model is proposed for the analysis of a two-input PAND gate in a DFT. This model is then used in a successive cascading structure for the analysis of a general multiple-input PAND gate. For a DFT with PAND gates, a stochastic approach using the proposed models provides a fast analysis of the DFT compared to an accurate or algebraic approach. The use of non-Bernoulli sequences as initial input event probabilities makes the stochastic approach faster and more accurate than Monte Carlo simulation. The stochastic approach has the following features.

1) The failure probability of a basic event is not limited to an exponential distribution; any failure distribution can be analyzed by an appropriate sampling and coding into the stochastic non-Bernoulli sequences.

2) Repeated events are correctly and readily handled in a DFT analysis, because signal correlation is maintained in the random binary bit streams and the propagation of the stochastic sequences in a fault tree analysis.

3) The stochastic approach avoids the state-space explosion problem or the large computational complexity typically encountered in a Markov or analytical method, thus it is scalable for use in a general DFT analysis.

Chapter 7

A Stochastic Approach for the Analysis of Dynamic Fault Trees with Spare Gates under Probabilistic Common Cause Failures

The so-called spare gate is extensively used to model the dynamic behavior of redundant systems in the application of dynamic fault trees (DFTs). Additionally, the basic components of a system are often subject to common cause failures (CCFs) including those caused by earthquakes, sudden changes in the environment, design errors, and incorrect operations [54]. CCFs are sometimes closely related; for instance, floods are likely to be caused by hurricanes. These CCFs are referred to as *s*-dependent CCFs. Furthermore, the occurrence of a CCF is usually not deterministic, but probabilistic, thus referred to as a probabilistic CCF (PCCF) [130]. The probability of occurrence differs by components or conditions. The consideration of PCCFs in a DFT analysis is of great significance as the system's reliability is likely to be overestimated without incorporating PCCFs. However, it presents a great challenge to consider PCCFs in a DFT analysis using existing methods, such as an integration-based approach. In this chapter, stochastic computational models are proposed for a fast analysis of spare gates and PCCFs in a DFT. This work has been published as [131].

The novelty of this chapter is as follows:

- Stochastic computational models are proposed for analyzing a two-input spare gate and PCCFs in a DFT. The warm spare (WSP), and cold spare (CSP) gates are then analyzed in detail. The effect of PCCFs is taken into consideration, and a stochastic logic model is constructed for dependent PCCFs.

7.1. Spare Gate

A standby system modeled by a spare gate usually consists of two types of modules: the primary (or online) modules, and the standby modules. Standby modules are used to replace the faulty modules to keep the system functional or operational. Hence, the spare gate fires (i.e., fails) if both of the modules fail. Spare gates are divided into three categories, depending on the switching relationship of the primary and standby modules: the hot spare gate (HSP) [37][70], the cold spare gate (CSP) [37][70], and the warm spare gate (WSP) [37][132]-[134]. In an HSP system, a standby module is always powered and ready to replace a faulty primary module when a fault occurs. HSPs are typically used in systems in which a minimal reconfiguration time is required, e.g. a chemical process control system. In a CSP, however, the standby modules are usually not powered until it is necessary to replace a faulty module [135]. Hence, it is usually used in power consumption critical systems, such as a satellite system [136]. As a tradeoff between CSP and HSP, the standby modules in a WSP are powered initially, but with a lower failure rate. The failure rate of the standby module in a WSP changes when it is switched to replace a faulty module [132]. For WSP, usually less power is consumed in the standby state compared to HSP, and less initialization and recovery time are required compared to CSP [133][134].

Fig. 7.1 shows different types of spare gates. A spare gate models the sequential failure events of the primary online module P (with a failure rate of λ_P), and the standby module S. The failure rate is assumed to be $\alpha \cdot \lambda_S$ prior to the switching of the standby module to replace a faulty component. The standby module in operation is subject to a failure rate λ_S after switching. Hence, the spare gate can be classified by a different value of the factor α: if $\alpha = 1$, the gate is an HSP gate; if $0 < \alpha < 1$, it becomes a WSP gate; if $\alpha = 0$, it is a CSP gate. A generic failure rate switching diagram for a spare gate is shown in **Fig. 7.2**.

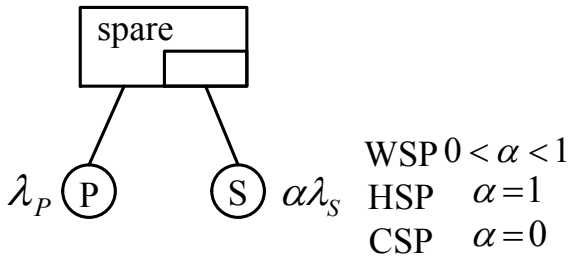


Fig. 7.1. A spare gate [37]. It is classified into different categories (WSP, HSP, and CSP) by the factor α, according to the failure behavior of the standby module.

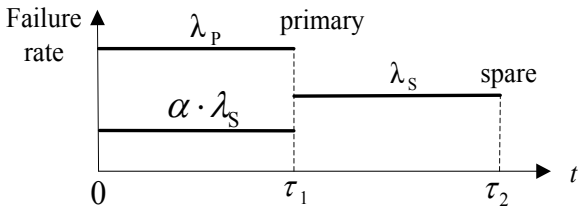


Fig. 7.2. A generic switching diagram for the failure in a spare gate [37][122].

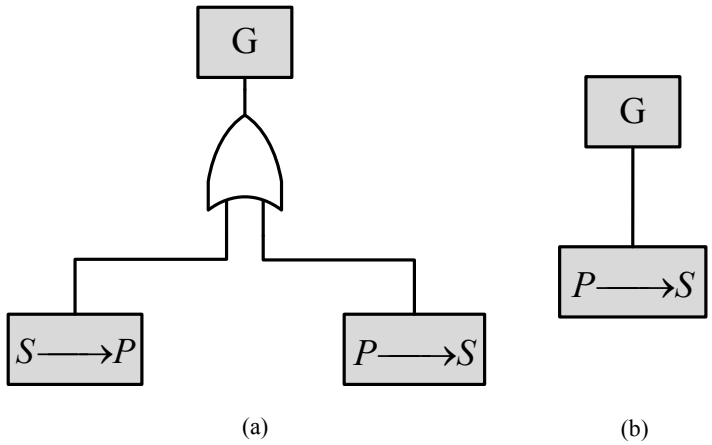


Fig. 7.3. The spare gate decomposition: (a) a combinational model for the spare gate, and (b) a simplified model for CSP. " → " indicates an inclusive precedence in a failure order.

HSP and CSP gates can be regarded as special cases of the WSP gate; the only difference lies in the value of the failure rate before and after the switching point. A spare gate can be converted into a combinational fault tree with two sequential components serving as inputs of an OR gate, as shown in **Fig. 7.3** [70][122].

In **Fig. 7.3**, the sequential event $S \rightarrow P$ indicates that both modules fail, and the standby module fails before the primary module does; while the sequential event $P \rightarrow S$ means that both modules fail, and the primary module fails before the standby module. The two sequential events cannot occur at the same time, thus they are mutually exclusive. Furthermore, it is impossible for the standby module of a CSP gate to fail because the failure rate of the standby module before switching is 0. Hence, the combinational model for a CSP gate can be simplified, as shown in **Fig. 7.3(b)**; this indicates that the output failure probability of a CSP gate is the same as the failure probability of the sequential event $P \rightarrow S$. The output failure probability of the spare gate in **Fig. 7.3(a)** is given by [122]:

$$U_{sys} = p(P \rightarrow S) + p(S \rightarrow P). \tag{7.1}$$

The probabilities of the two sequential failure events in (7.1) are given by:

$$p(P \rightarrow S) = \int_0^t \int_0^{t-\tau_2} f_P(\tau_2) f_{S,\lambda}(\tau_1)\,(1 - \int_0^{\tau_2} f_{S,\alpha\lambda}(\tau_1) d\tau_1) d\tau_2 d\tau_1, \tag{7.2}$$

and

$$p(S \rightarrow P) = \int_0^t \int_{\tau_1}^t f_P(\tau_2) f_{S,\alpha\lambda}(\tau_1)\, d\tau_2 d\tau_1, \tag{7.3}$$

where $f_A(t)$, $f_{S,\alpha\lambda}(t)$, and $f_{S,\lambda}(t)$, are the failure probability density functions (*pdfs*) for the primary module, the standby module before replacing the faulty primary module and after replacing the faulty primary module. For a *k*-out-of-*n* WSP system consisting of identical WSPs, a closed expression can be derived; however, it is only applicable to systems with identical input events [137].

7.2. Proposed Stochastic Models

In a redundant system, some modules are online or operational, while one or more

modules function as standby. They are therefore referred to as primary, and standby modules, respectively. Standby modules are critical for tolerating hardware failures or software errors; fault tolerance is achieved by removing the faulty primary module from the operation, and replacing it with a spare unit [136]. In a DFT, the primary and standby modules are considered as input events to a spare gate [70]. If the primary and standby components are not treated as basic events, or the input events include several standby components, it becomes cumbersome to obtain the failure probability through the use of existing approaches. Moreover, the components in a realistic DFT system may also suffer from common cause failures that occur either deterministically or probabilistically. This condition makes the distribution of the failure behavior even more complicated. Hence, a stochastic model is proposed in this paper for analyzing spare gates.

In this section, the discretization of a continuous probability distribution and the generation of the non-Bernoulli sequences are the same as the basic assumptions in Chapter 6.

7.2.1. Stochastic Models for the WSP and CSP Gates

Let S_{i-1}^P, and S_i^P denote the non-Bernoulli sequences generated for the failure probabilities of the primary module at two adjacent time intervals $i-1$, and i, i.e., F_{i-1}^P, and F_i^P, where F^P is the *cdf* for the failure of the primary module. As a primary module is non-repairable, the relationship of (6.5) must be met. For the jth bit in the non-Bernoulli sequence, the state of the primary module is given by $S_{i-1,j}^P$ and $S_{i,j}^P$ for the two consecutive time intervals; a state of 0, or 1 indicates that no fault occurs, or a fault occurs respectively. The state combination of the primary module at time $i-1$ and time i for the jth trial is represented by $S_{i-1,j}^P S_{i,j}^P$, where $S_{i-1,j}^P S_{i,j}^P \in \{00, 01, 11\}$, due to the non-reparability assumption. A trial is carried out by a bit or a combination of bits

in the stochastic sequences. For the WSP or CSP gate, the failure rate of the standby module varies before and after switching to replace the primary module (for CSP, the failure probability is 0 before switching). Hence, it is necessary to record the failure time of the primary module to determine the failure probability of the standby module. If $S_{k-1,j}^{P} = 0$, and $S_{k,j}^{P} = 1$, it indicates that the primary module fails at time k for the jth trial; hence, for WSP and CSP, the operational time of the standby module should be determined from the failure time of the primary module, i.e., $t_s = i - k$, where i is the present mission time, and k is the failure time of the primary module. Similarly, the operational time of the standby module can be determined for any other trial.

Let S_{i-1}^{S}, and S_{i}^{S} be the stochastic sequences generated for the failure probabilities of the standby module at two adjacent time intervals $i-1$, and i respectively. Then we discuss the following three different cases: $S_{i-1,j}^{P}S_{i,j}^{P} = 00$, $S_{i-1,j}^{P}S_{i,j}^{P} = 01$, and $S_{i-1,j}^{P}S_{i,j}^{P} = 11$.

- For $S_{i-1,j}^{P}S_{i,j}^{P} = 00$, the primary module does not fail at time i. For a WSP, if $S_{i-1,j}^{S} = 1$, then $S_{i,j}^{S} = 1$. If $S_{i-1,j}^{S} = 0$, the current state of the standby module for the jth trial is determined by the failure probability, i.e., the *cdf* $F_{i}^{S,\alpha\lambda}$ obtained from the failure rate of the standby module before it is switched to replace the primary module.
- For $S_{i-1,j}^{P}S_{i,j}^{P} = 01$, the primary module fails at time i, but it still functions at time $i-1$. The standby module is expected to replace the faulty primary module at the failure time. For a WSP, if $S_{i-1,j}^{S} = 0$, $S_{i,j}^{S}$ is determined by the failure *cdf* of the standby module before switching to replace the faulty module, $F_{i}^{S,\alpha\lambda}$.

In both of these two cases, $S_{i,j}^{S}$ remains 0 for a CSP as the standby module is not activated prior to the failure of the primary module, and is assumed to be fault free before

it is switched to replace the faulty primary module.

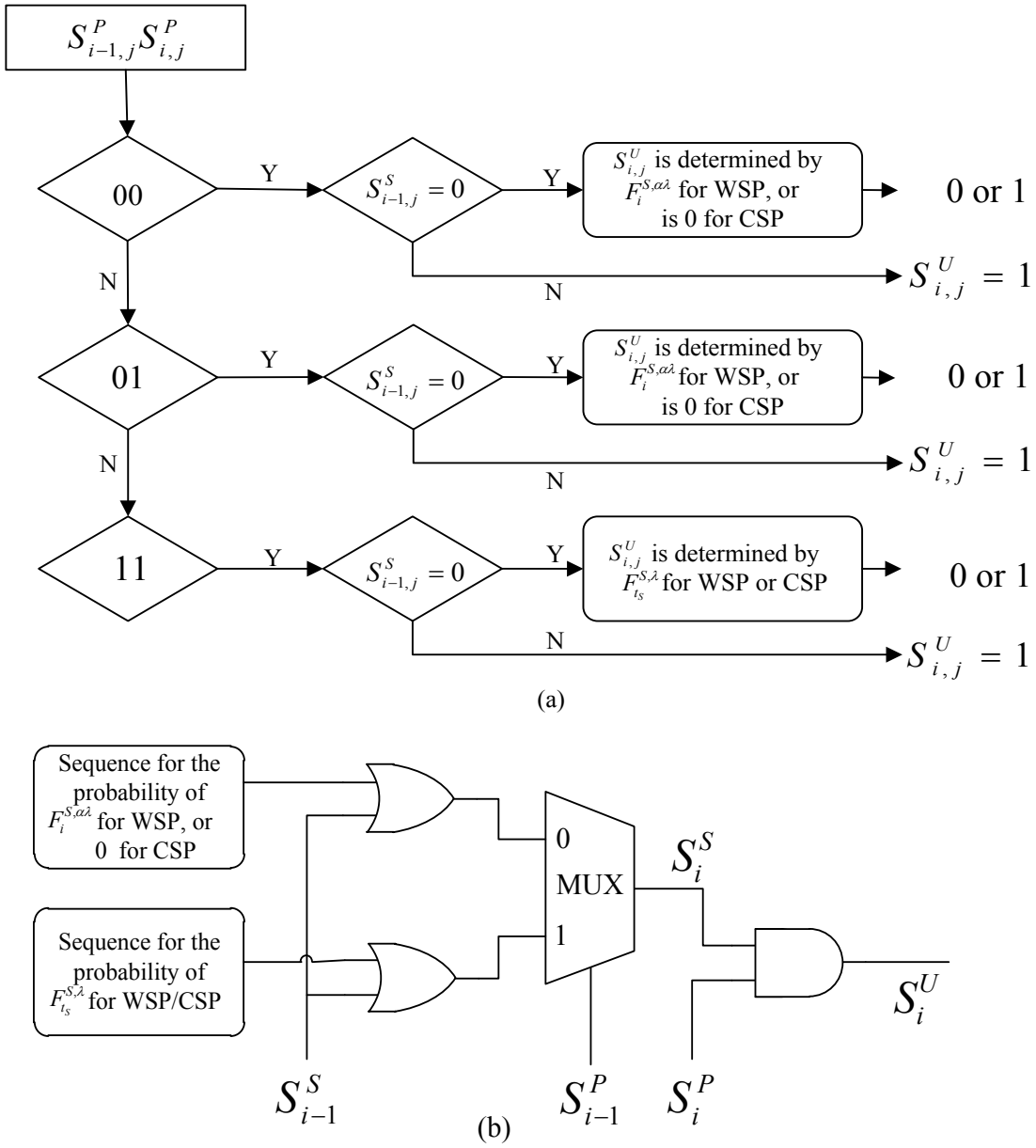


Fig. 7.4. (a) Flowchart for generating the stochastic sequences of the standby module, and (b) a general stochastic logic model for the spare gates (WSP and CSP). S_i^U denotes the output sequence for the spare gate.

- For $S^P_{i-1,j} S^P_{i,j} = 11$, the primary module has failed by time $i-1$. For the standby module, if $S^S_{i-1,j} = 1$, then $S^S_{i,j} = 1$. Otherwise, $S^S_{i-1,j}$ is determined for a WSP gate by the failure *cdf*, i.e. $F_{t_s}^{S,\lambda}$, obtained from the failure rate of the standby module after switching to replace the faulty primary module; while for a CSP, it is also

determined by $F_{t_s}^{S,\lambda}$, where t_s is the operational time of the standby module.

These processes are shown in the flowchart of **Fig. 7.4(a)**, which can be implemented by the stochastic architecture in **Fig. 7.4(b)**. These stochastic architectures model the sequential behavior of the WSP and CSP gates.

The proposed stochastic model is applied to evaluate the WSP in **Fig. 7.1**; the results are compared with those obtained by an accurate approach [122], as shown in Table 7.1. In Table 7.1, the failures of basic events are exponentially distributed with λ_P = 0.001, α = 0.6, and λ_S = 0.0025. $\boldsymbol{F}_A$, and $\boldsymbol{F}_S$ are the failure probability vectors for the accurate, and stochastic analysis respectively. $\Delta\boldsymbol{F}_{S-A}$ denotes the discrepancies of the two failure probability vectors, i.e., $\Delta\boldsymbol{F}_{S-A} = \boldsymbol{F}_S - \boldsymbol{F}_A$. The differences between the simulation results are measured by several norms.

Table 7.1. Evaluation of the stochastic WSP gate model for a mission time of 1,000 hours compared with an accurate approach [122]. The average simulation time for the stochastic approach is also provided.

Sequence length L (bits)	1k	10k	100k
$\|\Delta\boldsymbol{F}_{S-A}\|_1$	6.0512	2.1548	0.8354
$\|\Delta\boldsymbol{F}_{S-A}\|_2$	0.2424	0.0763	0.0163
$\|\Delta\boldsymbol{F}_{S-A}\|_\infty$	0.0189	0.0047	0.0025
Avg. time (s)	0.24	1.09	10.48

For the WSP gate with the same initial parameters of Table 7.1, the top event's failure probability is obtained for different mission times by the accurate and stochastic approaches (for a sequence length of 10k). For the WSP gate, the exact failure probabilities for 300, 600, and 1,000 hours (as computed by (7.1)) are 0.1175, 0.3173, and 0.5500 respectively, while the results obtained by the stochastic approach are 0.1153, 0.3136, and 0.5477 by using a sequence length of 10k bits. As shown in **Fig. 7.5**, the accuracy of the stochastic approach can be further improved with longer stochastic sequences. In general, the stochastic approach accurately computes the failure probability at a reasonable sequence length (e.g. 10k bits).

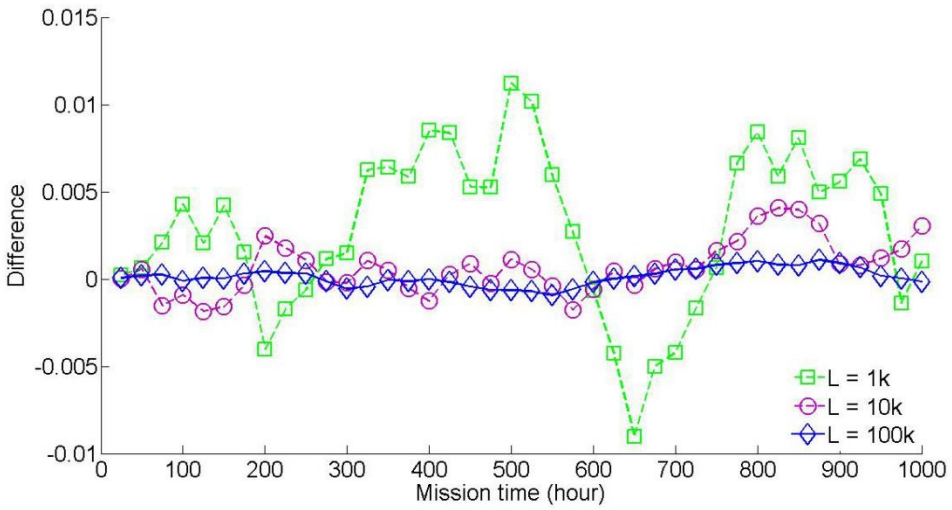


Fig. 7.5. The differences in the failure probabilities obtained by the stochastic approach and an accurate analysis for the WSP in Fig. 7.1.

7.2.2. Stochastic Models for CCFs and Majority Voters

The stochastic model for common cause failure (CCF) is presented next for analyzing a general DFT. This process is followed by an improved model that considers the CCF's probabilistic behavior, i.e., probabilistic CCFs (PCCFs). Finally, a stochastic majority voter is proposed.

7.2.2.1. Stochastic Model for CCFs

Generally, CCFs are usually modeled by two types of methods: explicit methods [138][139], and implicit methods [140][141]. In this Chapter, an explicit method is modeled by a stochastic approach, and the CCF is considered as a basic event. To model *s*-dependent CCFs, a multiplexer is used with the stochastic sequences as inputs, as shown in **Fig. 7.6(a)**. For a DFT [142], a hurricane occurs with a probability of $p(h) = 0.015$. As floods usually occur in conjunction with hurricanes, the dependent relationship between the hurricane and flood can be described by conditional probabilities. The occurrence of floods is usually conditional on the occurrence of

hurricanes, denoted as $p(b = f|h) = 0.55$, and $p(a = f|\bar{h}) = 0.035$. These conditional probabilities can be derived from available weather information [143].

The function computed by the 2-to-1 multiplexer in **Fig. 7.6 (a)** is given by

$$p(f) = p\big(a = f|\bar{h}\big) \cdot p\big(\bar{h}\big) + p(b = f|h) \cdot p(h), \tag{7.4}$$

where $f|\bar{h}$, and $f|h$ are the events of floods (f) conditional on the occurrence of a hurricane (h). The output of the multiplexer is determined by the value of the control bit. For the 2-to-1 multiplexer of **Fig. 7.6(a)**, one of the inputs is selected as the output according to the distributions of zeros and ones in the control sequence encoding the signal probability of h. For a sequence length of 10k bits, the input sequences for probabilities of $p(f|h) = 0.55$, and $p(f|\bar{h}) = 0.035$ consist of 5500, and 350 ones, respectively. If the random input sequences are independent, the output of the multiplexer is expected to be 0.0427 (by (7.4)), i.e. approximately 427 ones are expected in the output sequence for a sequence length of 10k bits. If multiple conditions are considered, for example to compute $p(ABC)$ based on $p(AB)$ and $p(C|AB)$, two conditions can first be combined, e.g., using an AND gate for a conjunction of the two events *A* and *B*; then this new condition can be used as the control input to a multiplexer for computing the joint probability $p(ABC)$. This process is shown in **Fig. 7.6(b)**. The computed result is however approximate due to the inevitable stochastic fluctuations inherent in the processing of the random binary bit streams. This is an important feature in stochastic computation as probabilistic values are propagated rather than deterministic ones.

It has been shown in [29] that, when the initial probabilities are encoded by non-Bernoulli sequences, the stochastic fluctuations are reduced compared with the use of Bernoulli sequences. For the previous example, the occurrence probability of floods is obtained by using a multiplexer with stochastic non-Bernoulli sequences; the mean and variance are reported in Table 7.2 for a number of simulations using different sequence

lengths. As shown by the simulation results, the evaluation accuracy is better for the stochastic approach with a smaller variance, and it can be improved with an increase of sequence length.

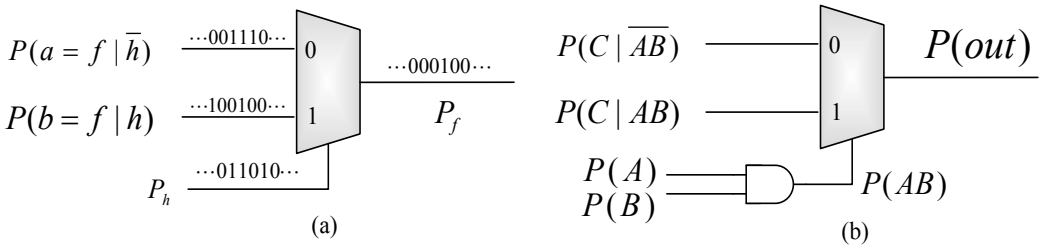


Fig. 7.6. (a) A stochastic multiplexer model for the s-dependency relationship between the two *s*-dependent common cause failures (CCFs) of flood (*f*) and hurricane (*h*), and (b) a stochastic model for computing the joint probabilities of multiple conditions.

Table 7.2. Mean, and variance of the occurrence probability of floods obtained by using the stochastic approach and Monte Carlo (MC) method for 1,000 experiments with different sequence lengths *L* (bits) or simulation runs *N*.

N/L		**1k**	**10k**	**100k**
Stochastic	Mean	4.26×10^{-2}	4.27×10^{-2}	4.27×10^{-2}
	Variance	3.89×10^{-6}	4.16×10^{-7}	3.83×10^{-8}
MC	Mean	4.28×10^{-2}	4.27×10^{-2}	4.27×10^{-2}
	Variance	5.59×10^{-5}	4.19×10^{-6}	4.01×10^{-7}

The stochastic approach effectively computes the occurrence probability of dependent CCFs as evidenced by the average run time in Table 7.2. Moreover, the variance is significantly reduced with an increase of sequence length. The use of a sequence length of 1k bits generates very accurate results, with a relative disparity (RD) of approximately 0.23%, compared to the analytical result of 0.0427 computed by (7.4). RD is defined as:

$$RD = (p - p_0)/p_0 \tag{7. 5}$$

where p, and p_0 are the probabilities obtained by using the stochastic approach, and an accurate analysis, respectively. For an increased sequence length, a smaller RD can be obtained by the stochastic approach.

7.2.2.2. A Stochastic Model for PCCF

A mechanical system can be subject to multiple CCFs, as denoted by CCF_1, CCF_2, ⋯, CCF_m. The failure of a dependent event affected by a specific CCF (say CCF_i) occurs with a probability, so the CCF is considered as a probabilistic CCF (PCCF). The occurrence probability of a PCCF is given by $\gamma_i = p(Dependent\ event\ fails|CCF_i\ occurs)$; γ_i may vary for different components affected by a CCF.

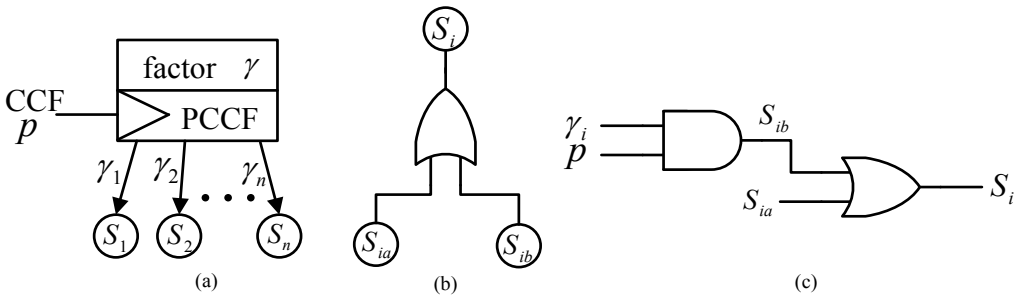


Fig. 7.7. (a) A probabilistic common cause failure (PCCF) gate [130], (b) a combinational model for the PCCF gate, and (c) proposed stochastic model for the PCCF gate.

In **Fig. 7.7(a)**, the CCF occurs as a trigger event with probability p; then, one or more dependent events affected by the trigger event will fail with a specific probability. For example, an event S_i occurs with probability γ_i if the trigger event occurs. Let S_{ia} and S_{ib} denote the failures of the basic event S_i without, and with considering the effect of a CCF; then, if S_{ia} or S_{ib} occurs, the output event S_i fails. Thus, the failure of the event S_i can be modeled as an OR gate with two input events S_{ia} and S_{ib} (as illustrated in **Fig. 7.7(b)**). The event S_{ib} occurs with probability $p \cdot \gamma_i$, where p is the occurrence probability of the CCF, and γ_i is the conditional failure probability of the dependent event S_i affected by the CCF. A stochastic model is proposed to implement the PCCF, as shown in **Fig. 7.7(c)**. The simulation results for this model are shown in Table 7.3 for $p(A) = 0.1$, and $p(CCF) = 0.01$; the probability of event A affected by

the CCF is given by $\gamma = 0.3$.

Table 7.3. Mean and variance of the simulated occurrence probability of a component *A* under a PCCF by applying the stochastic approach for 1,000 simulations. The average run time is also provided.

Sequence length *L* (bits)		**1k**	**10k**	**100k**
$p(A)$	Mean	0.1026	0.1027	0.1027
	Variance	1.94×10^{-6}	2.13×10^{-7}	1.96×10^{-8}
Avg. time (s)		6.42×10^{-4}	6.68×10^{-4}	7.14×10^{-2}

As revealed in Table 7.3, the proposed stochastic approach effectively computes the occurrence probability of PCCFs. The relative disparity (given by (7.5)) is approximately 0.0974% for a sequence length of 1,000 bits compared with the analytical result of 0.1027 [130]. Furthermore, the variance can be significantly reduced with a longer sequence length, thus the accuracy in the failure probability obtained by the stochastic approach increases with an increase of sequence length.

7.2.2.3. A Stochastic Model for Majority Voter

The stochastic structure for a 2-out-of-3 majority voter is shown in **Fig. 7.8**, as implemented by stochastic logic in **Fig. 7.8 (b)**. The analytical expression for the output probability of the 2-out-of-3 majority voter is given by [144]:

$$p(Out) = p(A) \cdot p(B) + p(A) \cdot p(C) + p(B) \cdot p(C) - 2 \cdot p(A) \cdot p(B) \cdot p(C). \qquad (7.6)$$

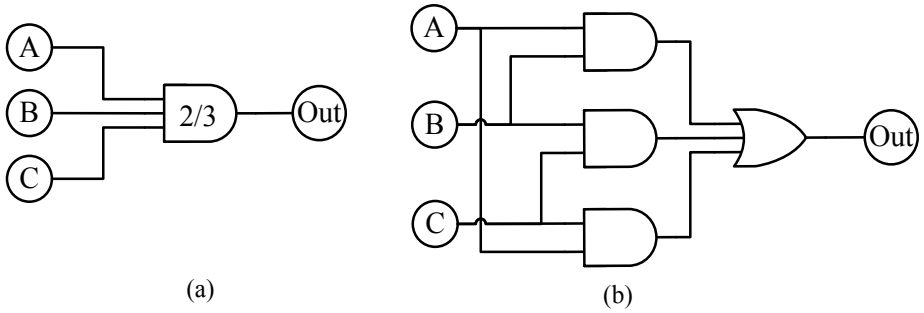


Fig. 7.8. (a) A 2-out-of-3 majority voter [144], and (b) a stochastic model for the 2-out-of-3 majority voter.

For a 2-out-of-3 majority voter with inputs' failure probabilities given in Table 7.4,

the output failure probability is 0.2880 by (7.6); the relative disparity (RD) is 0.069% (given by (7.5)) for the stochastic approach using sequences of 1k bits. For a 3-out-of-5 majority voter, the output probability is 0.1780 using the analysis of [144], while the RD is approximately 0.17% for the stochastic approach with $L = 1\text{k}$ bits. Thus, the error in the failure probability for the stochastic approach decreases with the increase of sequence length. Similar stochastic circuits can be constructed for majority gates with more than three inputs.

Table 7.4. Mean and variance of the failure probabilities of 2-out-of-3 and 3-out-of-5 majority voters, obtained by the stochastic approach. The average simulation time is also provided.

For the 2-out-of-3 voter, $p(A) = 0.3,\ p(B) = 0.6,\ p(C) = 0.2$			
Sequence length L (bits)	**1k**	**10k**	**100k**
Mean	0.2882	0.2880	0.2880
Variance	6.31×10^{-5}	5.43×10^{-6}	5.82×10^{-7}
Avg. time (s)	2.39×10^{-3}	2.26×10^{-3}	1.40×10^{-2}
For the 3-out-of-5 voter $p(A) = 0.2,\ p(B) = 0.4,\ p(C) = 0.5, p(D) = 0.1, p(E) = 0.4$			
Sequence length L (bits)	**1k**	**10k**	**100k**
Mean	0.1783	0.1781	0.1780
Variance	6.68×10^{-5}	6.27×10^{-6}	6.39×10^{-7}
Avg. time (s)	3.15×10^{-3}	2.77×10^{-2}	0.28

7.3. DFT Analysis Flow

Following the proposed stochastic models for the spare and PAND gates [116], the process for evaluating the top event's failure probability of a general DFT with PCCFs consists of the following steps.

Step 1: Replace the original spare gate with the proposed stochastic model for WSP and CSP in **Fig. 7.1(b)**.

Step 2: Substitute the original FDEP and PAND gates with the OR model [67][68], and the stochastic PAND model in [116], respectively; then a DFT with dynamic gates can be implemented by combinational logic.

Step 3: Encode the events' failure probabilities at different time steps into non-Bernoulli sequences.

Step 4: If PCCFs are considered in the DFT, an additional PCCF module is required for each of the basic events subject to PCCFs. Moreover, if the CCFs are *s*-dependent, a stochastic multiplexer is used to model the effect of the dependency.

Step 5: Derive the top event's failure probability at different time steps by propagating the non-Bernoulli sequences through the stochastic models.

7.4. Case Studies

In this section, several case studies are presented to show the efficiency and accuracy of the stochastic method, in comparison with the analytical method of [40], and the MC approach of [125]. Simulations are performed for DFTs with, and without PCCFs. Furthermore, the effect of dependent PCCFs is also analyzed. Non-exponential distributions of the basic events are also considered to show the capabilities of the stochastic approach to handle general cases. All simulations are run on a computer with a 3.10 GHz i3-2100 microprocessor, and a 6 GB memory.

Let the failure probability of a basic event B at time i be F_i^B, $i \in \{1, 2, \cdots, M\}$, obtained as the failure *cdf* for the basic event; then the failure probability of the DFT is given by:

$$F(i) = f(F_i^B), \tag{7. 7}$$

where $f(\cdot)$ indicates the logic operation determined by the system's topology. Hence, the failure probability vector for the entire mission time is given by a vector $\boldsymbol{F} = (F(1), F(2), \cdots, F(M))$, where M indicates the number of discretized intervals of the mission time. The failure probability vectors obtained using the stochastic, analytical [40],

and MC [125] approaches are then represented by $\boldsymbol{F}_S$, $\boldsymbol{F}_A$, and $\boldsymbol{F}_{MC}$ respectively. Hence, $\Delta\boldsymbol{F}_{S-A}$ indicates the disparity vector for the stochastic and analytical methods; $\Delta\boldsymbol{F}_{MC-A}$ represents the disparity vector for the MC and analytical approaches. Similarly, the norms, $\|\cdot\|_1$, $\|\cdot\|_2$ and $\|\cdot\|_\infty$, are used to measure the differences of these failure probability vectors.

7.4.1. HECS, with and without PCCFs

A DFT of the Hypothetical Example Computer System (HECS) (from [37], and shown in **Fig. 7.9**) is used to illustrate the efficiency and accuracy of the proposed stochastic method.

The correct operation of the HECS is determined by the states of different systems such as the processor (A1, A2, and A), memory, bus, and application interfaces [145]. The HECS will fail if any of the four subsystems fail. The computer of **Fig. 7.9** is modeled by a DFT in **Fig. 7.10(a)** as illustrated in [145]. Using the stochastic models of the dynamic gates, a complete stochastic system can be constructed for the HECS, as shown in **Fig. 7.10(b)**.

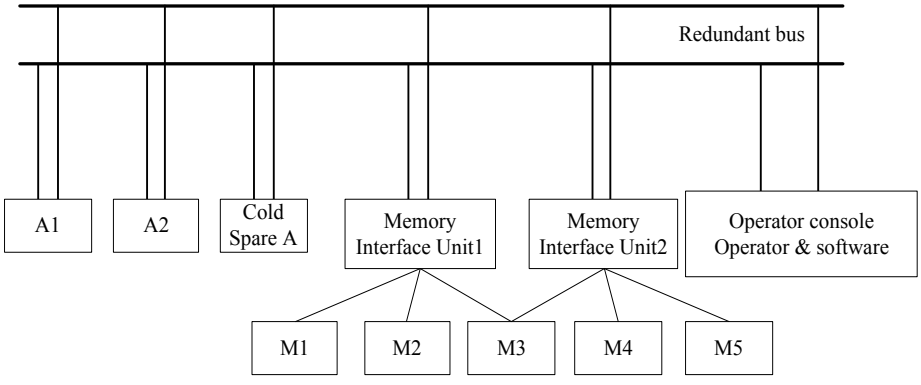


Fig. 7.9. The Hypothetical Example Computer System (HECS) [37].

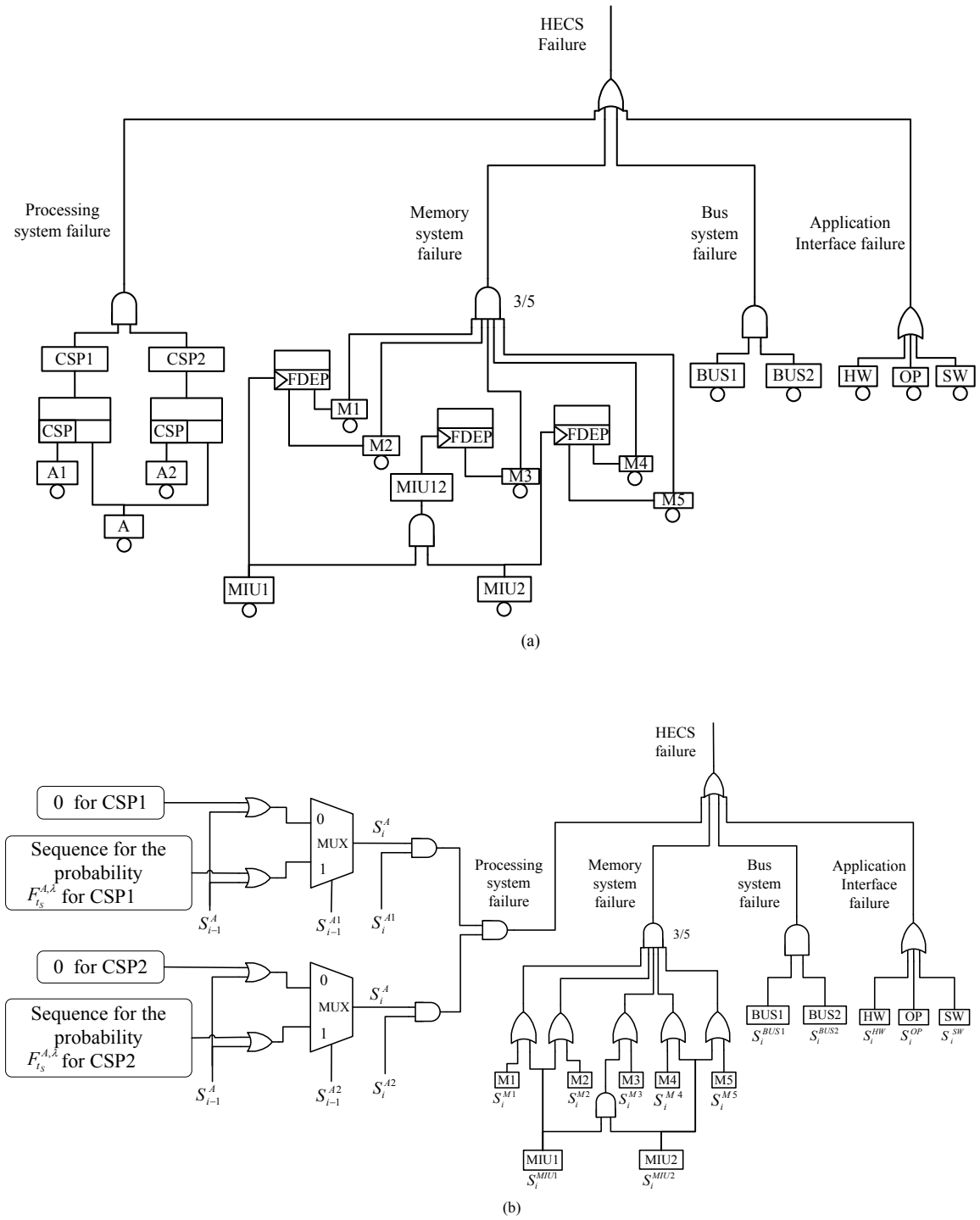


Fig. 7.10. (a) A dynamic fault tree (DFT) of Hypothetical Example Computer System (HECS) with CSP, FDEP, and static gates [145]; and (b) the stochastic model for the HECS with constituent dynamic gate models.

The mission time of the HECS is assumed to be 100 hours, and the failure

behaviors of the basic events in the HECS are considered to be exponentially distributed (the failure rates are shown in Table 7.5 [37][145]).

Table 7.5. The failure rates of the basic events in the Hypothetical Example Computer System (HECS) [37][145].

Basic event	Failure rate (h^{-1})
A1 A2 A	10^{-4}
M1M2 M3 M4 M5	6×10^{-5}
MIU1,MIU2	5×10^{-5}
BUS1,BUS2	10^{-6}
HW	5×10^{-5}
SW	3×10^{-2}
OP	10^{-3}

For a mission time of 100 hours, the difference in failure probabilities of the HECS and the average simulation time are shown in Table 7.6 for the different approaches. N, and L denote the number of simulation runs for the MC method, and the sequence length for the stochastic approach, respectively. The norms of the disparity vectors are presented for the stochastic and MC [125] approaches.

Table 7.6. Norms of the differences in the top event's failure probability vectors obtained by the proposed stochastic approach and Monte Carlo (MC) simulation for the dynamic fault tree (DFT) compared to accurate analysis. The average run time is also provided.

N/L		1k	10k	100k
$\|\Delta \boldsymbol{F}_{S-A}\|_1$		0.1749	0.0529	0.0168
$\|\Delta \boldsymbol{F}_{S-A}\|_2$		0.0225	0.0066	0.0023
$\|\Delta \boldsymbol{F}_{S-A}\|_\infty$		0.0064	0.0016	7.0872×10^{-4}
$\|\Delta \boldsymbol{F}_{MC-A}\|_1$		0.8199	0.2830	0.1024
$\|\Delta \boldsymbol{F}_{MC-A}\|_2$		0.1079	0.0386	0.0140
$\|\Delta \boldsymbol{F}_{MC-A}\|_\infty$		0.0364	0.0134	0.0049
Avg. time (s)	Accurate Analysis	1.74×10^{-3}		
	Stochastic	5.09×10^{-2}	0.39	3.96
	MC	0.14	1.26	12.59

As shown in Table 7.6, the proposed stochastic approach requires a shorter run time, and results in a smaller variance in the computed failure probability; hence, it is faster

and more accurate than the MC method. The evaluation accuracy can be further improved by increasing the sequence length. However, a tradeoff between precision and efficiency must be determined when selecting the sequence length. Although the accurate analysis results in the shortest run time, the significantly longer time required for deriving the analytical expressions is not included in the value reported in Table 7.6.

In practice, the failure distribution is not limited to an exponential distribution if other factors such as aging are taken into consideration. Therefore, a non-exponential distribution may be required for a more accurate model. A Weibull distribution is considered for a DFT with non-exponentially distributed basic events. The *pdf,* and *cdf* of a Weibull distribution are given by (6.47) and (6.48) in Chapter 6 respectively.

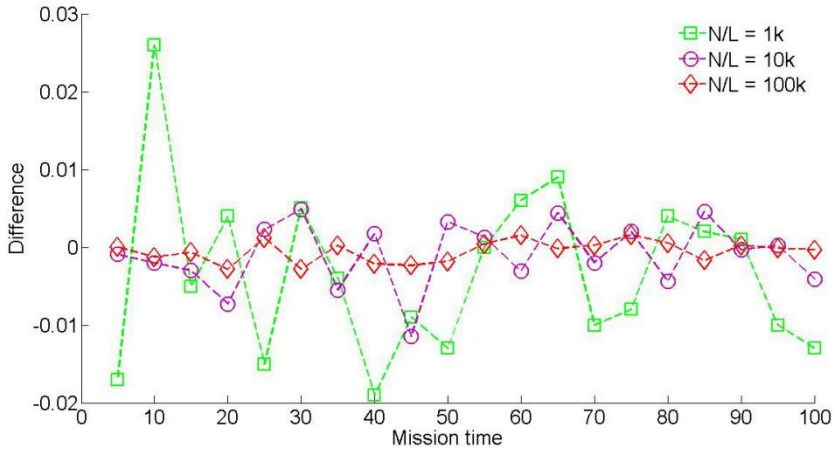


Fig. 7.11. Difference in the failure probabilities of the top event for the Hypothetical Example Computer System (HECS) for a mission time of 100 hours.

Assume that the basic events A1 and BUS1 follow a Weibull distribution with $\lambda = 2$, and $k = 0.1$, while the failures of the other basic events are exponentially distributed. Furthermore, assume that BUS1 and HW are subject to a CCF (i.e. $\gamma_i = 1$) with an occurrence probability of 0.1. **Fig. 7.11** reveals the difference of the top event's failure probabilities for a mission time of 100 hours, obtained by both the stochastic

approach and MC simulation [125]. It can be seen that the difference between the stochastic and MC approach decreases with the increase of the sequence length. For a sequence length (or simulation runs) of 10k bits, $\|\cdot\|_1$, $\|\cdot\|_2$, and $\|\cdot\|_\infty$ of the differences in the failure probability vectors obtained by the two approaches are 0.3087, 0.0387, and 0.0133 respectively. As revealed by these norm values, a DFT with non-exponentially distributed basic events subject to PCCFs can be effectively evaluated by the proposed stochastic approach with a reasonable sequence length.

7.4.2. DFT with Dependent PCCFs

A DFT with WSP, FDEP, and PAND gates is analyzed next to show the efficiency of the stochastic approach (**Fig. 7.12(a)**). Assume that h, l, and f denote the events of hurricanes, lightning strikes, and floods respectively. Table 7.7 shows the exponentially distributed failure rates of the components; non-exponential distributions will be dealt with subsequently. The occurrence probabilities of a hurricane, and a lightning strike are given as $p(h) = 0.015$ and $p(l) = 0.025$ respectively. The dependencies between the CCFs are modeled by the conditional occurrence probabilities of floods because of the occurrence of a hurricane, i.e., $p(f|h) = 0.55$, and $p(f|\bar{h}) = 0.035$. These probabilities are determined from weather information [143]. The probability that a component fails due to a CCF is assumed to be $\gamma_i = 0.8$ for $i = 1, 2, 3$ (where i indicates a different CCF, i.e., h, l, f, respectively). For this DFT, a stochastic model is constructed in **Fig. 7.12(b)** with the PAND model of [116], and the stochastic models in **Figs. 7.6** and **7.7** for considering the effects of PCCFs.

The average run time and the norms used to measure the differences in the failure probability vectors of the DFT obtained by the stochastic and MC [125] approaches are given in Table 7.8 for a mission time of 200 hours. Also shown are the failure probabilities by considering PCCFs for each of the modules.

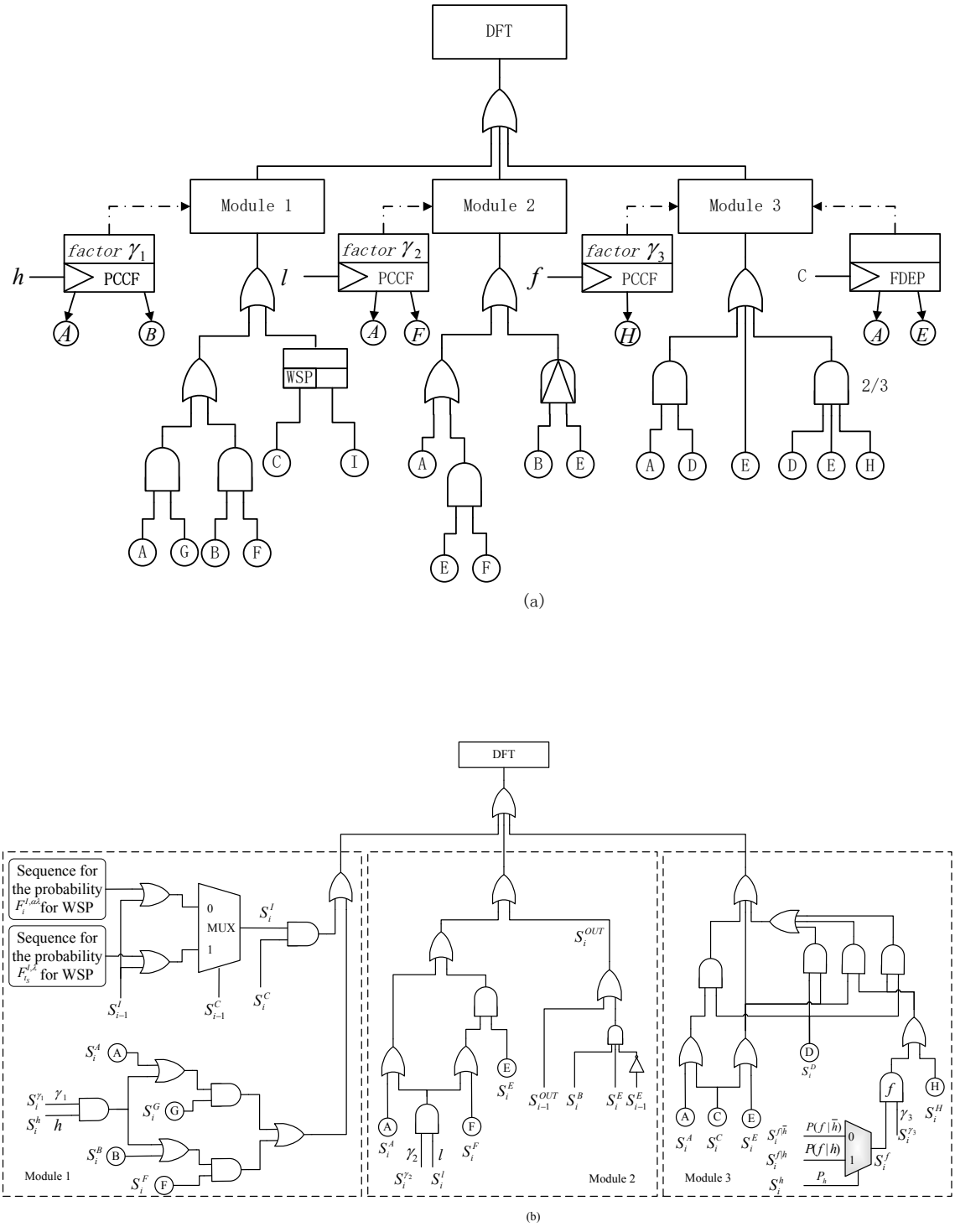


Fig. 7.12. (a) A dynamic fault tree (DFT) with *s*-dependent probabilistic common cause failures (PCCFs) (taken from [54]), and (b) a stochastic model for the DFT in (a).

In Monte Carlo simulation, the result follows approximately a Gaussian distribution for a large number of runs. The result of stochastic computation with a long sequence length is also approximately Gaussian distributed [29]. In this case, a parameter z_c can be used to determine the confidence interval of the simulated results [146]. The error in the computed result is then given by:

$$E = \frac{z_c}{\mu}\sqrt{\frac{v}{m}}, \tag{7.10}$$

where μ, and v are the accurate mean, and variance of the distribution of the results, and m is the number of simulations (or equivalently, the number of bits in a stochastic sequence). For a confidence level of 95%, z_c = 1.96. For the failure rates in Table 7.7 and for $\gamma_i = 0.8$, stochastic sequences of one million bits are used to find the accurate mean, and variance, as 0.8119, and 0.1527, respectively, for the stochastic approach. For a sequence length of 10k, the error is then obtained by (7.10) as 0.9434% (i.e. less than 1%) at a confidence level of 95%. As per (7.10), the error decreases with an increase of sequence length at a given confidence level. The required sequence length can thus be estimated by (7.10) for achieving a desired evaluation accuracy.

Table 7.7. Component failure rates (10^{-3}/hour).

Basic events	Failure rates	Basic events	Failure rates
A	1.5	B	1.0
C	4.0	D	1.0
E	2.0	F	1.0
G	3.0	H	2.0
I (spare)	1.0	I (working)	2.0

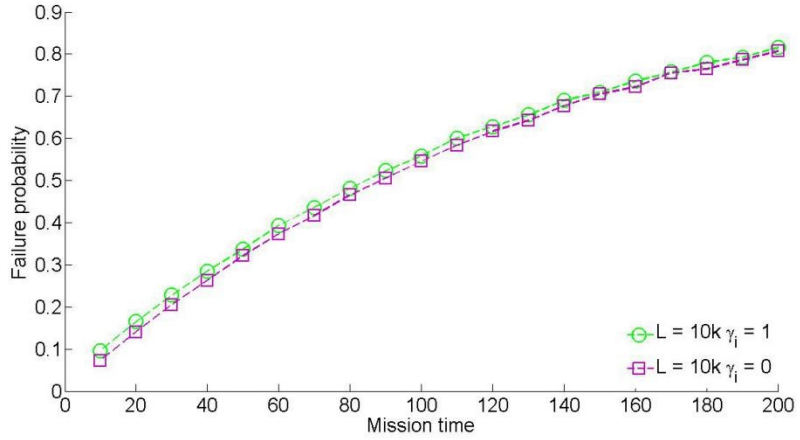
For this DFT, the top event's failure probability for a mission time of 200 hours is plotted in **Fig. 7.13** for two values of γ_i, where $\gamma_i,\ i = 1,\ 2,\ 3$, indicates the probability that the basic event is affected by a CCF. In **Fig. 7.13 (a)**, the failures of the basic events are assumed to be exponentially distributed; for $\gamma_i = 0$, i.e., when the effect of a CCF is not considered, the failure probability is underestimated compared to the case when the

occurrence of a CCF will definitely cause a failure of the basic event (i.e. when $\gamma_i = 1$). In the latter case, the failure probability is overestimated compared to the case when the probabilistic behavior of a CCF is considered, i.e., when $0 < \gamma_i < 1$ (for which the failure probabilities would lie between the values shown in **Fig. 7.13(a)**). Furthermore, assume that components A and D follow a Weibull distribution with $\lambda = 2$, and $k = 0.1$, while the failure rates of the other basic events remain at the values given in Table 7.7. **Fig. 7.13(b)** plots the differences between the failure probabilities obtained by the stochastic and MC approaches; the difference decreases with an increase of stochastic sequence length, or the number of simulation runs. For a sequence length (or simulation runs) of 10k bits, the norms, $\|\cdot\|_1$, $\|\cdot\|_2$, and $\|\cdot\|_\infty$, of the differences in the failure probability vectors of the stochastic and MC approaches are obtained as 0.8198, 0.0734, and 0.0136, respectively.

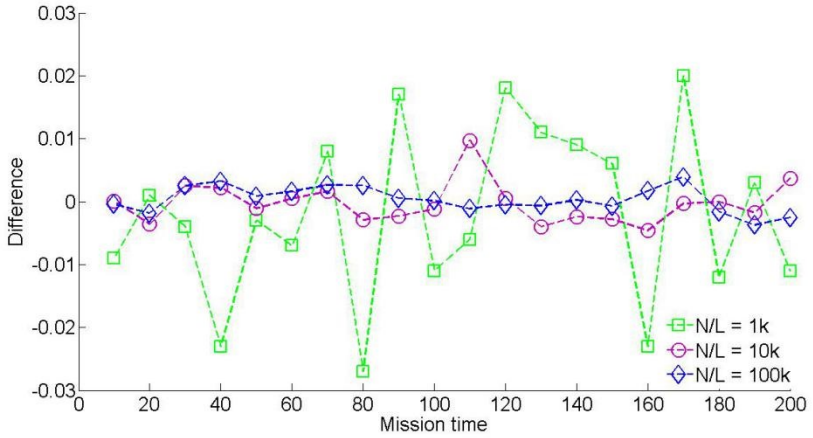
Table 7.8. Norms of the differences in the top event's failure probability vectors of the dynamic fault tree (DFT) in Fig. 7.12, and the average run time for the proposed stochastic approach and Monte Carlo (MC) simulation in [125].

Under CCFs (i.e. $\gamma_i = 1$) *N/L*		**1k**	**10k**	**100k**
$\|\Delta \boldsymbol{F}_{S-MC}\|_1$		2.4370	0.7457	0.2301
$\|\Delta \boldsymbol{F}_{S-MC}\|_2$		0.2227	0.0680	0.0205
$\|\Delta \boldsymbol{F}_{S-MC}\|_\infty$		0.0460	0.0148	0.0041
Avg. time (s)	Accurate	4.90×10^{-3}		
	Stochastic	0.22	2.06	16.28
	MC	0.38	3.55	36.66
Under PCCFs with $\gamma_i = 0.8$ *N/L*		**1k**	**10k**	**100k**
$\|\Delta \boldsymbol{F}_{S-MC}\|_1$		2.5580	0.8148	0.2498
$\|\Delta \boldsymbol{F}_{S-MC}\|_2$		0.2324	0.0722	0.0216
$\|\Delta \boldsymbol{F}_{S-MC}\|_\infty$		0.0450	0.0152	0.0042
Avg. time (s)	Accurate	6.10×10^{-3}		
	Stochastic	0.24	1.87	21.03
	MC	0.48	4.92	41.16

The DFT systems (inclusive of the spare gate, PAND gate, and FDEP gate) can be effectively evaluated by the proposed stochastic approach. The stochastic approach is



(a)



(b)

Fig. 7.13. Example 7.2: (a) the failure probability of the dynamic fault tree (DFT) subject to probabilistic common cause failures (PCCFs) for $\gamma_i = 0$, and $\gamma_i = 1$ (for basic events with exponentially distributed failures, using a sequence length of 10k bits); and (b) the difference of the failure probabilities of the DFT subject to PCCFs for $\gamma_i = 0.8$ (for basic events with non-exponentially distributed failures).

faster than a MC method [125] with an equivalent number of simulation runs. Furthermore, the accuracy of the proposed stochastic approach can be improved by increasing the sequence length. The required sequence length is determined as a tradeoff between precision and efficiency. It is also shown that the reliability of a DFT system decreases by considering the effects of PCCFs that widely occur in practice. Hence, if the failure of a certain component affected by PCCFs is not considered, then the reliability of

a DFT is likely to be overestimated. If the effect of PCCFs is considered to be deterministic in causing a failure, then the reliability of a DFT is underestimated.

7.5. Summary

In this chapter, stochastic models have been proposed for analyzing a two-input spare gate and PCCFs in a DFT; the WSP, and CSP gates have been analyzed in detail. For a DFT with spare gates, a stochastic approach using the proposed models provides a fast analysis of the DFT compared to an analytical approach. The use of non-Bernoulli sequences as initial input probabilities makes the stochastic approach faster and more accurate than Monte Carlo simulation. The effect of PCCF has been taken into consideration, and a stochastic logic model has been constructed for dependent PCCFs. The efficiency and accuracy of the proposed stochastic approach have been shown by the case studies of several benchmark systems.

Chapter 8

Reliability Evaluation of Phased-Mission Systems using Stochastic Computation

In this chapter, a stochastic computational approach is proposed for analyzing a phased-mission system (PMS). Initially, the topology of each phase in a PMS is modeled by either a static fault tree, or a dynamic fault tree (DFT). A stochastic computational model is then proposed for evaluating the output failure probability of a PMS as a function of the failure probabilities of its components. Due to the stochastic sequence's capability of preserving signal correlation, repeated common components at different phases (as frequently encountered in a PMS) are also analyzed. A PMS with dynamic gates can be analyzed using the proposed approach by utilizing stochastic models of priority AND (PAND) and functional dependency (FDEP) gates. Finally, due to the direct encoding of failure probabilities into non-Bernoulli sequences, any distributions can be analyzed by the stochastic model, as shown by several case studies. The contents presented in this chapter have been submitted for publication in IEEE Transactions on Reliability as [147].

The novelty of this chapter is as follows:

- A stochastic computational model is proposed for the analysis of a PMS. The effect of common failures at different phases is considered and various stochastic models have been developed to evaluate the dynamic behaviors in a PMS.
- The accuracy of the stochastic analysis is affected by the simulated sequence length in stochastic computation. However, the stochastic approach is more accurate than Monte Carlo simulation with an equivalent number of runs.
- The stochastic approach is applicable to any failure distribution of the basic

components.

8.1. Motivation

For a PMS, the mission time is usually decomposed into multiple non-overlapping phases. A PMS consisting of H phases is represented by H fault trees with each of them modeling the failure conditions of a phase. The fault tree for each phase can be constructed using stochastic logic gates. For the success of a PMS, all phases are required to be successfully and sequentially completed [148]. Many practical systems operate in this sequential manner, such as an aircraft flight, a nuclear power plant, aerospace and distributed computing systems [72][149]-[154]. For example, an aircraft mission of an unmanned autonomous vehicle (UAV) has a number of phases, including taxing, take-off, climbing to the required altitude, and cruising, descending and landing phases. The mission can fail in any of these phases and the PMS must be evaluated to obtain the failure probability of each phase. The PMS achieves an overall mission success only if every phase successfully completes the task. Hence, the overall mission failure is obtained by a logic OR of the failures of all phases [72].

Several approaches have been proposed to evaluate the reliability of a PMS. These approaches are mainly classified into two classes: analytical and simulation-based approaches [155][156]. The analytical approaches can be differentiated into three categories: combinatorial methods [154][157], state-space based methods [158]-[161] and phase modular methods [142][162][163]. A combinatorial method can handle any failure distribution and provide the exact failure probability using, for example, a binary decision diagram (BDD). However, this analysis is cumbersome when deriving the exact analytical expression as a function of the basic components' failure distributions due to the presence of a large number of basic components. The BDD based combinatorial approaches of [154] and [164] are only applicable to a PMS with static phases. Furthermore, dynamic relationships (such as functional dependency and priority relationships) usually exist among the basic components in a PMS. Hence, the reliability evaluation becomes even more challenging when dynamic gates are included in the PMS.

For a state-space based method, a large complexity is usually encountered when analyzing complex systems due to the state-space explosion problem. The phase-modular methods of [162][163] can model dynamic relationships, but a Markov chain analysis is required for the dynamic modules. The application of a Markov chain analysis is however limited when a basic component's failure behavior is not exponentially distributed. Simulation-based approaches can be found in [125][155][156] and Monte Carlo (MC) simulation [125] can also be used to evaluate the failure probability of a PMS. However, a long simulation time is often required for a simulation-based approach.

8.2. Preliminaries

8.2.1. A Phased-Mission System (PMS)

A PMS is usually defined as follows [72]:

1) It consists of multiple non-overlapping phases and the operation of phases is performed in a sequential order;
2) The topology of the system usually varies from phase to phase, i.e., different failure criteria apply to each phase;
3) All phases must be successfully completed for the mission to be successful.

A component can fail at any time during the mission time and the state of a component may be critical for a specific phase. Furthermore, the failure of a component resulting in the failure of the PMS may have occurred during previous phases. The mission can fail in any of the phases; hence, the evaluation of the PMS must calculate the failure probability occurred in each phase.

The system topology is usually modeled as a fault tree to show the combinations of component failures. Let $Q(\cdot)$ be the logic operation derived from the system topology; then, the structure function at phase h is denoted by $Q_h(\cdot)$. The technique of fault tree

analysis (FTA) [34][37] is utilized to calculate the failure probability during phase h as:

$$p_h = Q_h(S_i(h)), \tag{8. 1}$$

where $S_i(h)$ indicates the stochastic sequence generated based on the failure probability of component $B_i(h)$, i.e., $p_i(h)$ $B_i(h) \in \phi(h)$, $\phi(h)$ denotes the set of basic components at phase h, and p_h is the failure probability of the sub-system of phase h.

To compute the overall mission failure probability of the PMS, P, a conventional phase-OR model is adopted, as shown in Fig. 8.1. In this structure, the output failure probability of the PMS is determined by the stochastic sequences for the failure probabilities of different phases. As the failures are exclusive events at different phases, P is given by the sum of the failure probabilities of all phases [72], i.e.,

$$P = \sum_{h=1}^{H} p_h, \tag{8. 2}$$

where $h \in \{1, 2, \cdots, H\}$ and H is the number of phases of the PMS.

Then, the reliability of a PMS, r, i.e., the probability that all phases are successful, is given by:

$$r = 1 - P. \tag{8. 3}$$

8.2.2. Assumptions

The background of Section 6.1 in Chapter 6 is also applicable in this chapter. Additionally, two more assumptions are provided for the PMS.

- For a PMS, the state of a component at the beginning of a phase is the same as the state at the end of the previous phase [154];
- The duration of each phase in the PMS is known for the investigation.

8.3. Stochastic Models for Phased-Mission System

A PMS consisting of H phases is usually represented by H fault trees (**Fig. 8.1**). For a binary PMS, both the system and the components can only have two states: success or failure. For the success of the PMS, all phases are required to be successfully completed. The failure of any phase results in the failure of the PMS. Let the stochastic sequence $S(h)$ encode the failure probability of phase h, $p(h)$, and $S_j(h)$ denote the value of the jth bit. If $S_j(h) = 1$ (i.e., the failure of phase h), then $S_j(PMS) = 1$ where $S(PMS)$ is the stochastic sequence for the failure probability of the overall PMS. The calculation of the failure probability of the PMS (as in formula (8.2)) can be implemented by an OR gate; thereafter, the reliability of the PMS is obtained by inverting the stochastic sequence for the failure probability of the PMS.

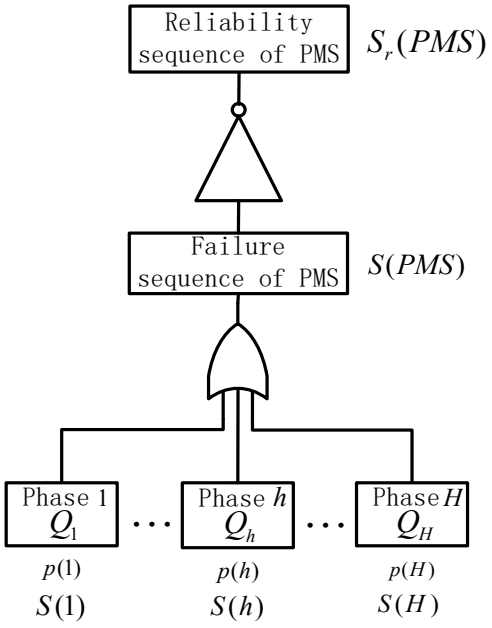


Fig. 8.1. A general structure of a phased-mission system (PMS) consisting of H phases with different system topologies. $S(h)$ is a stochastic sequence for the failure probability of phase h, $p(h)$. Q_h denotes the structural function at phase h, $(1 \le h \le H)$. H is the total number of phases. $S(PMS)$ denotes the stochastic sequence for the failure probability of the PMS. $S_r(PMS)$ denotes the stochastic sequence for the reliability of the PMS.

Each of the fault trees derived from the system topology (indicated by Q_h) is used to model the failure condition of a phase. **Fig. 8.2** illustrates a general fault tree structure of phase h, $h \in \{1, 2, \cdots, H\}$, for a PMS. Each phase has a different system topology, so let Q_h denote the structure function at phase h. If there are n basic components at phase h, let $A_j(h)$ indicate the jth component at phase h.

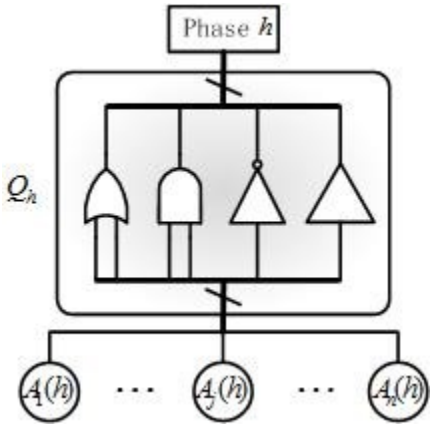


Fig. 8.2. A general fault tree structure of phase h for a phased-mission system (PMS) consisting of H phases with different system topology at each phase. $A_j(h)$ denotes the jth component at phase $h(1 \leq h \leq H)$. H is the number of phases in the PMS. n is the number of basic events in phase h.

The structure function at phase h, Q_h, is constructed using the stochastic logic gates according to the relationships between the basic components. Then, the output stochastic sequence for the failure probability of phase h is obtained by first determining the input sequences for the basic components. Let x_h denote the states of basic components at the end of phase h, i.e., $x_h = (x_1(h), x_2(h), \cdots, x_n(h))$, where $x_j(h)$ denotes the state of the basic component $A_j(h)$, $j \in \{1, 2, \cdots, n\}$. Then, $Q_h(x_h)$ gives the state of the system at the end of phase h. If $Q_h(x_h) = 0$, phase h is successfully completed. Otherwise, the system fails by the end of phase h.

In a general PMS, common components are often encountered in different phases. For example, the engine of a UAV must function in most phases. However, some of the

components might appear in all phases while some of them are just used in a few specific phases. The distribution of the common components is illustrated in **Fig. 8.3**, where $\phi(j)$ and $\phi(k)$ denote the basic component sets of phases j and k respectively, $j,k \in \{1,2,\cdots,H\}$. For any phases, say j and k, $\phi(j) \cap \phi(k) \neq \emptyset$ indicates that common components exist at phases j and k (as in Fig. 8.3(a)). If the failure can be masked and does not necessarily cause a system failure, it is referred to as case 1. If the failure of a common component directly results in the failure of a specific phase, it is referred to as case 3. As in **Fig. 8.3(b)**, $\phi(j) \cap \phi(k) = \emptyset$ indicates the case that there is no common component at phases j and k. This case is referred to as case 2 in this chapter.

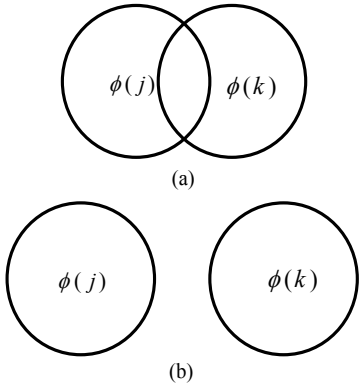


Fig. 8.3. Distribution of the common components for phases j and k, $j,k \in \{1,2,\cdots,H\}$. (a) Common components exist in phases j and k; (b) No common component at phases j and k. $\phi(j)$ or $\phi(k)$ denotes the basic component set of phase j or k. H is the number of phases in the PMS.

8.3.1. Case 1

Assume that component A_i exists in phase k and also in phase j, i.e., $A_i \in \phi(j) \cap \phi(k)$. **Fig. 8.4** shows an example of $A_i \in \phi(j) \cap \phi(k)$, where j and k are two consecutive or disjoint phases ($j < k$). In **Fig. 8.4** (and all subsequent figures, wherever applicable), $A_i(j)$ denotes the ith component at phase j with $i \in \{1,2,\cdots,m\}$, $j \in \{1,2,\cdots,k\}$ and $B_l(k)$ denotes the lth component at phase k with $l \in \{1,2,\cdots,n\}$. m and n are the total numbers of basic components for phase j and k respectively. If

the failure of the common component A_i does not directly cause the failure of phase j, the failure of A_i is masked by other components (for instance by an AND operation); so this component is not required to survive prior to phase k (as an example of case 1).

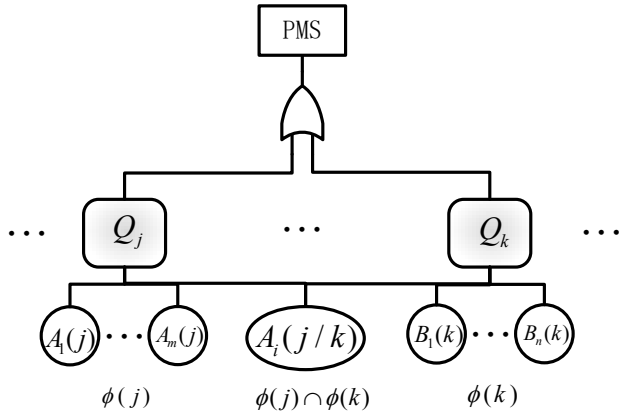


Fig. 8.4. Example of case 1: $\boldsymbol{\phi(j)\cap\phi(k)\neq\emptyset}$ and $\boldsymbol{A_i(j/k)\in\phi(j)\cap\phi(k)}$, where $\boldsymbol{A_i(j/k)}$ is one of the common components.

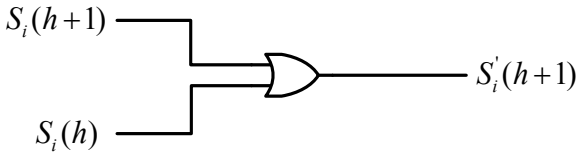


Fig. 8.5. A stochastic logic model for computing the failure probability of component A_i for cases 1 and 2.

For simplicity, let $j=h$ and $k=h+1$. Since, the failure of the component A_i may occur during phase $h+1$ or a phase before $h+1$, the stochastic sequence for the failure probability of A_i at phase $h+1$, $S_i'(h+1)$, is found by considering $S_i(h)$ (the stochastic sequence at the end of phase h) and $S_i(h+1)$ (the stochastic sequence for the failure probability of component B_i during $h+1$). Then the sequence $S_i'(h+1)$ is obtained using the stochastic model in **Fig. 8.5**.

Let $S_{i,j}(h)$, $S_{i,j}(h+1)$ and $S_{i,j}'(h+1)$ be the values of the jth bit of the corresponding stochastic sequences. If $S_{i,j}(h)=1$ (so component A_i fails at the end of

phase h), then $S'_{i,j}(h+1) = 1$ due to the assumption of non-reparability of the PMS. This corresponds to the case when component A_i fails before phase $h+1$. Otherwise, whether component A_i fails or not is determined by $S_{i,j}(h+1)$ for the failure probability during phase $h+1$, i.e., $p_{A_i}(h+1)$. This indicates that $S_{i,j}(h+1)$ is selected as the value of $S'_{i,j}(h+1)$ if $S_{i,j}(h) = 0$. This can be implemented by a stochastic OR gate as shown in **Fig. 8.5**.

8.3.2. Case 2

For case 2, there is no common component in any of the two phases j and k, i.e., $\phi(j) \cap \phi(k) = \emptyset$ (as in **Fig. 8.3(b)**); then whether the components of phase j or k fail or not cannot result in the failure of phase j or k (with $j < k$). **Fig. 8.6** shows an example of case 2. With the stochastic logic models for phases j and k (i.e., Q_j and Q_k), the failure probabilities of phases j and k are determined by the failure probabilities (encoded into non-Bernoulli sequences) of the input components of phases j and k respectively.

Again, let $j = h$ and $k = h+1$. The failure of phase k is determined by the failure of basic events in $\phi(k)$. For a basic component B_i, it may fail before or during phase $h+1$. Both scenarios need to be considered for the failure of phase k. Hence, the stochastic sequence for the failure probability of component B_i at phase $h+1$, $S'_i(h+1)$, is obtained by considering $S_i(h)$ and $S_i(h+1)$, the stochastic sequences for the failure probabilities of component B_i at the end of phase h and during phase $h+1$ respectively, are as follows. Then, the failure probability of B_i is found by using the stochastic model of **Fig. 8.5** (same as for case 1).

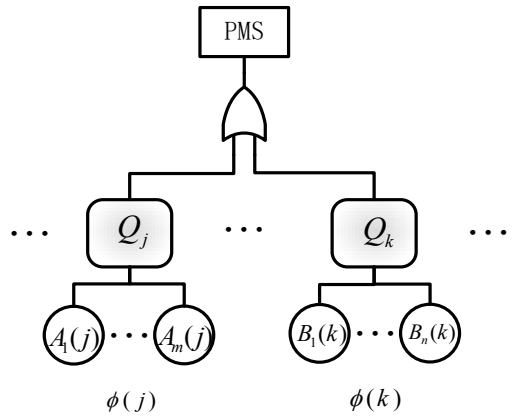


Fig. 8.6. Example of case 2 (corresponding to Fig. 8.3(b)). $\boldsymbol{\phi(j) \cap \phi(k) = \emptyset}$.

8.3.3. Case 3

For case 3, assume that component A_i appears in both phases j and k, i.e., $A_i \in \phi(j) \cap \phi(k)$. Let again $j = h$ and $k = h + 1$. The failure of A_i directly causes the failure of phase h, so the failure probability at the end of phase $h + 1$ is given by the failure probability during phase $h + 1$ with the failure before phase $h + 1$ excluded, i.e., $S_i'(h + 1) = S_i(h + 1)$ implemented by a buffer as in **Fig. 8.7**.

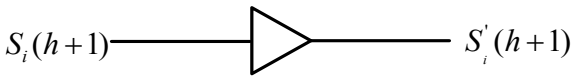


Fig. 8.7. A stochastic logic model for computing the failure probability for case 3 of component A_i.

Furthermore, if the failure probability for an intermediate time point during phase $h + 1$ is of interest, the following analysis is applicable. Let $\tau(h + 1)$ denote the intermediate mission time during phase $h + 1$; then, the intermediate failure probability of the overall PMS during phase $h + 1$ is derived by replacing the stochastic sequence $S_i(h + 1)$ (for $p_i(h + 1)$) with $S_i(\tau(h + 1))$ (for $p_i(\tau(h + 1))$).

Hence in the proposed approach, a stochastic model of the PMS can be constructed

for the relationship among basic components using stochastic logic gates. Two stochastic models (**Figs. 8.5** and **8.7**) are utilized to evaluate the effect of common components in different phases and to obtain the reliability of each phase from the failure probabilities of the basic components. The failure probability of the overall PMS is encoded in the statistics, i.e., the proportion of number of ones, in the output sequence of the stochastic analysis; the reliability of the PMS can then be determined by inverting the stochastic sequence indicating its failure probability (**Fig. 8.1**).

8.4. Phased-Mission System Evaluation Procedure

For a general PMS, the process of evaluating its overall reliability is as follows.

Step 1: Construct the PMS using stochastic logic gates;

Step 2: Determine the common components in different phases and find whether the failure of the common components can directly cause the failure of the phase being investigated;

Step 3: Compute the failure probabilities at different mission times based on the provided *pdfs* and *cdfs*;

Step 4: Encode the basic modules' failure probabilities at different time steps into non-Bernoulli sequences following the algorithm provided in Chapter 6;

Step 5: If a common component exists, the stochastic sequences must be generated using the stochastic model in **Fig. 8.5** or **Fig. 8.7**, as determined by whether a common component exists and whether the failure of the common component can result in the failure of the PMS;

Step 6: Derive the overall failure probability at different time steps by propagating

the non-Bernoulli sequences through the stochastic models. The reliability can be obtained by inverting the stochastic sequence indicating the failure probability of the PMS (using the model in **Fig. 8.1**).

In a DFT, the PAND gate can be utilized if required. This may occur for an input (to indicate the firing of a basic event in a predetermined order) and the output (to indicate whether a failure occurs [72]) as a priority relationship. A FDEP gate can further be utilized to model the behavior among the components. However, it is cumbersome to analyze a system using a combinatorial analysis. A stochastic model can instead be utilized to replace the dynamic gates with static gates. For systems with perfect fault coverage, the FDEP can be treated as an OR gate [67][68]. For the PAND gate in a PMS, the stochastic model of [116] can be used to model the priority relationship.

8.5. Case Studies

In this section, several case studies are presented to show the efficiency of the proposed stochastic method. The results are compared with the combinatorial analysis of [157] and the Monte Carlo (MC) simulation of [125]. The proposed stochastic approach is not limited to a particular failure distribution of the basic component. Hence, both exponential and non-exponential distributions are investigated to show the capability of the stochastic approach in handling the general cases. All simulations are run on a computer with a 3.10 GHz i3-2100 microprocessor and 6 GB memory.

The mission time τ is divided into H phases and each phase is further divided into M equal intervals. Let $p(i)$ be the failure probability of a basic component at time i, then the failure probability of an output event for the system is given by a vector $\boldsymbol{F} = (F(1), F(2), \cdots, F(M))$, i.e., $F(j) = Q(p(i))$, where $Q(\cdot)$ is the logic operation determined from the system topology. Let $\boldsymbol{F}_{Mc}$ and $\boldsymbol{F}_{S}$ denote the failure probability

vectors obtained by MC simulation and the stochastic approach respectively. Thus, $\Delta \boldsymbol{F}_{MC-S} = \boldsymbol{F}_{Mc} - \boldsymbol{F}_{S}$ denotes the difference between the failure probability vectors of the MC simulation and the stochastic approach. In the simulations of the following case studies (and for all related figures and tables as applicable), N denotes the number of simulation runs for the MC method, while L denotes the sequence length for the stochastic approach. Similar to [116], the norms, $\|\cdot\|_1$, $\|\cdot\|_2$ and $\|\cdot\|_\infty$, are calculated to measure the differences of failure probability vectors that reveal the accuracy of different approaches.

8.5.1. Example 1

A PMS with non-exponentially distributed basic events is analyzed first using the stochastic approach. It becomes challenging when the failures of basic components are not exponentially distributed for reliability evaluation. In this section, it is shown that this issue is effectively addressed by the stochastic approach. A Weibull distribution is considered for a PMS with non-exponentially distributed basic components. Furthermore, the reliability of the PMS at any time during the entire mission time can be found.

Example 1 deals with a non-repairable PMS consisting of four elements for a mission time of τ = 630 hours [157]. The system is successful only if the three phases perform without failure; the durations of the three phases are τ_1 = 160 hours, τ_2 = 200 hours and τ_3 = 270 hours. The structure of the stochastic PMS is shown in **Fig. 8.8**.

The parameters for each element of Example 1 in each phase are given in Table 8.1 [157]. Except for element 3 at phase 3 and element 4 at phase 1, which follow Weibull distributions, all other elements fail exponentially. The probability density function (*pdf*) and cumulative density function (*cdf*) of a Weibull distribution are given by (6.47) and (6.48) in Chapter 6 respectively.

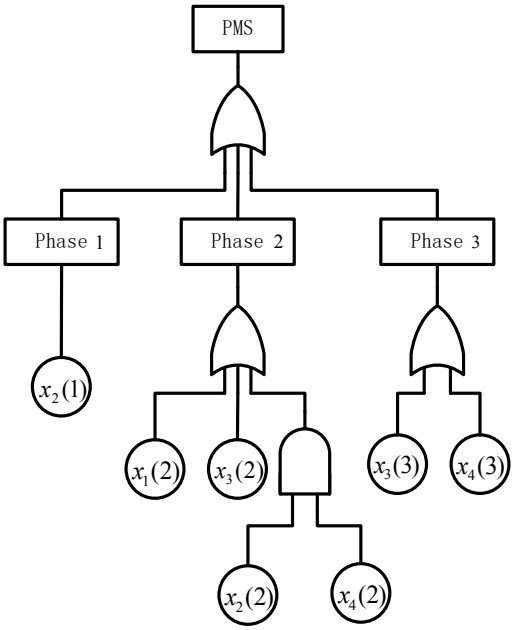


Fig. 8.8. A non-repairable phased-mission system (PMS) of Example 1 consisting of three phases and four components [157]. $x_i(j)$ denotes the ith component at phase j with $i \in \{1,2,3,4\}$ and $j \in \{1,2,3\}$.

Table 8.1. Input parameters for Example 1 [157] with exponentially and Weibull distributed basic components. $\boldsymbol{k}$ and $\boldsymbol{\lambda}$ denote the shape and scale parameters for the Weibull distribution.

Component	Phase 1	Phase 2	Phase 3
x_1	0.0002	0.0001	0.00015
x_2	0.0001	0.0001	0.0001
x_3	0.00025	0.0001	$1/\lambda = 0.0001,\ k = 2$
x_4	$1/\lambda = 0.0001,\ k = 1.5$	0.0002	0.0001

For the stochastic approach and the combinatorial analysis of [157], the reliabilities of the components at each phase and the reliability of the overall PMS are given in Table 8.2. For the same input parameters in Table 8.1, the mean and variance of the reliability of Example 1 obtained by using the stochastic, Monte Carlo (MC) and combinatorial approaches are also provided in Table 8.2.

Due to fluctuations in stochastic computation, the found reliability of the PMS is very close (or equal) to the probability obtained from the combinatorial analysis, as shown by the results in Table 8.2. For a reasonable sequence length, the stochastic

approach provides a very accurate result, and it is more accurate than MC simulation.

Table 8.2. Reliabilities of the phased-mission system (PMS) at different phases for Example 1 (sequence length here is 10k bits).

Element j	Methods	$h=1$	$h=2$	$h=3$
1	stochastic	0.9685	0.9803	0.9603
	combinatorial [157]	0.9685	0.9802	0.9603
2	stochastic	0.9841	0.9803	0.9735
	combinatorial [157]	0.9841	0.9802	0.9734
3	stochastic	0.9608	0.9803	0.9971
	combinatorial [157]	0.9608	0.9802	0.9971
4	stochastic	0.9980	0.9610	0.9734
	combinatorial [157]	0.9980	0.9608	0.9734
Reliability mean of 20 times	stochastic	0.8192		
	Monte Carlo	0.8196		
	combinatorial [157]	0.8188		
Variance of reliability for 20 times	stochastic	1.54×10^{-6}		
	Monte Carlo	1.85×10^{-5}		

For Example 1, the failure probability and reliability obtained by the stochastic approach are plotted in **Fig. 8.9** for a mission time of 630 hours.

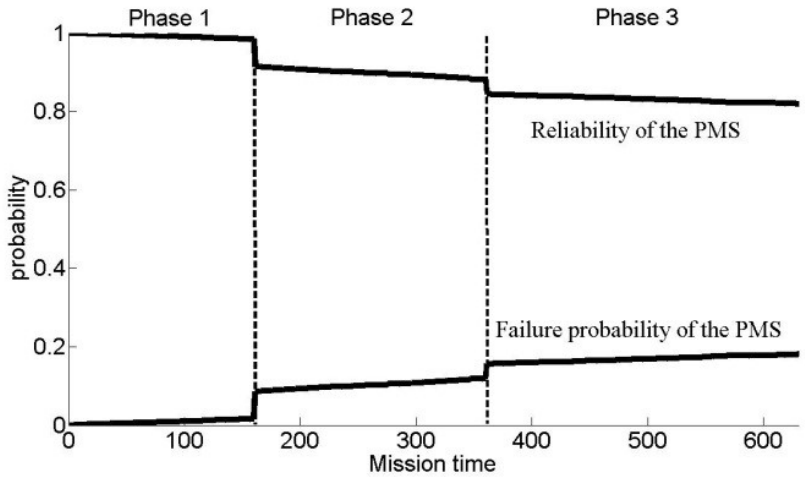


Fig. 8.9. Failure probability and reliability obtained by the stochastic approach for Example 1 with a sequence length of 10k bits.

Next, $\|\cdot\|_1$, $\|\cdot\|_2$, and $\|\cdot\|_\infty$ of the differences in the reliability vectors obtained by

the stochastic and MC approaches are shown in Table 8.3. As per the results in Table 8.3, the disparity in the failure probability vectors of the stochastic and MC approaches decreases, and thus, the stochastic fluctuations in both approaches decrease with the increase of the sequence length and the number of simulation runs. However, the stochastic approach is faster than MC simulation, as indicated by the smaller average run time for each sequence length or sample size (i.e., L/N in Table 8.3). The relationship between the average run time of the two approaches and the simulation runs/sequence length are further illustrated in **Fig. 8.10**.

Table 8.3. Norms of the differences in the failure probability vectors obtained by the proposed stochastic approach and Monte Carlo (MC) simulation for the phased-mission system (PMS) in Example 1.

N/L		1k	10k	100k
$\|\Delta \boldsymbol{F}_{MC-S}\|_1$		6.7200	0.8260	0.2717
$\|\Delta \boldsymbol{F}_{MC-S}\|_2$		0.3017	0.0384	0.0143
$\|\Delta \boldsymbol{F}_{MC-S}\|_\infty$		0.0300	0.0032	0.0016
Avg. time (s)	Stochastic	5.56×10^{-2}	0.34	7.68
	MC	0.29	2.42	23.30

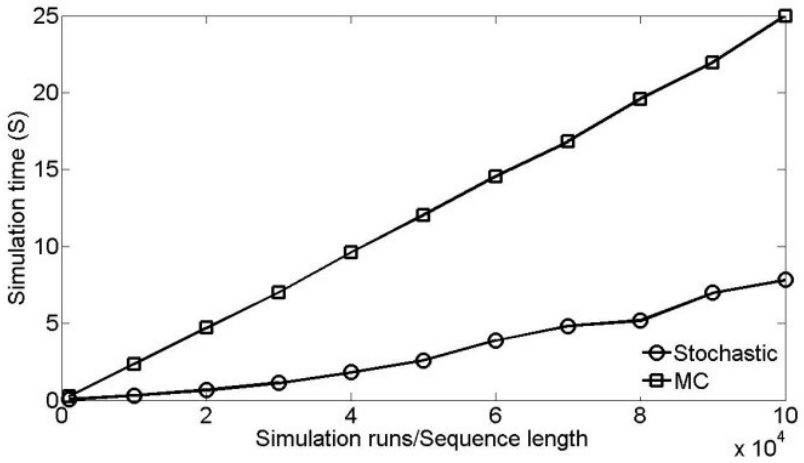


Fig. 8.10. The average run time for 10 simulation runs of Example 1 based on the stochastic approach and Monte Carlo (MC) simulation.

As shown by these simulation results, the stochastic approach can evaluate a PMS with non-exponentially distributed components at a high accuracy. As the encoding of a

failure probability into a stochastic sequence is not limited to exponential distributions, a PMS with non-exponentially distributed basic components can be effectively evaluated by the proposed stochastic approach using a reasonable sequence length. Hence, the proposed stochastic approach is applicable to both exponential and non-exponential distributions for a PMS analysis.

8.5.2. Example 2

Example 2 is taken from [54][142] and deals with the non-repairable PMS shown in **Fig. 8.11**. This PMS is simulated to assess the efficiency of the stochastic approach for a total mission time of 500 hours. The PMS consists of three consecutive non-overlapping phases and eight components. Furthermore, several common components exist in the PMS. The PMS contains a dynamic PAND gate in phase 2 and an FDEP gate in phase 3.

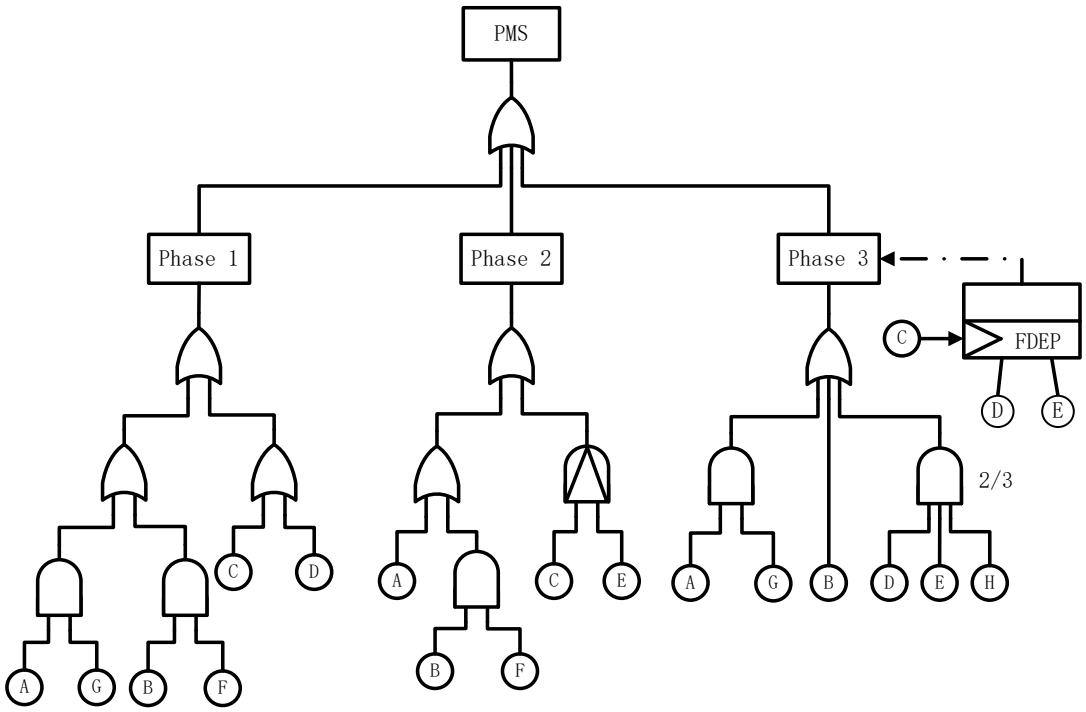


Fig. 8.11. A phased-mission system (PMS) consisting of three phases with a dynamic priority AND (PAND) gate in phase 2 and a functional dependency (FDEP) gate in phase 3 [54][142].

Table 8.4. Input failure parameters (10^{-3}/hour) for Example 2 with exponentially distributed basic components [54].

Basic components	Phase 1 τ_1 = 100 hours	Phase 2 τ_2 = 250 hours	Phase 3 τ_3 = 150 hours
A	1.0	2.0	1.5
B	0.5	1.0	1.0
C	3.0	2.0	1.0
D	3.0	2.0	1.0
E	1.5	1.0	2.0
F	1.0	1.0	1.0
G	2.0	1.0	1.0
H	1.0	2.0	2.0

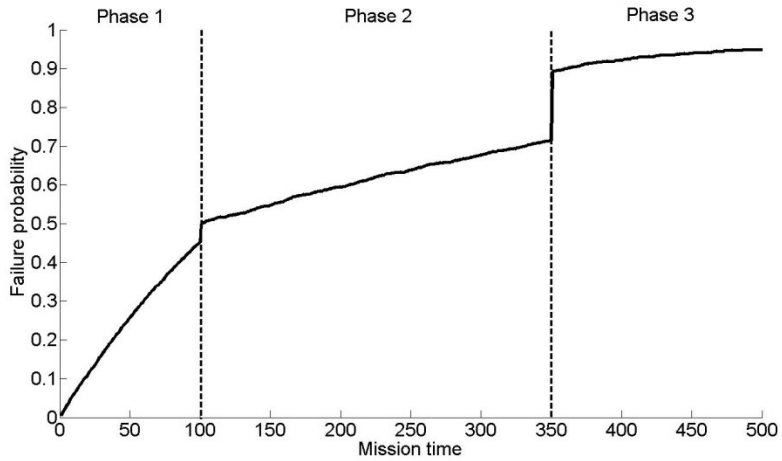
The stochastic model can be constructed for the PMS shown in **Fig. 8.11** with the adoption of stochastic logic gates. For systems with perfect fault coverage, the FDEP is treated as an OR gate [67][68]; moreover, the stochastic model of [116] is utilized for modeling the PAND gate. Table 8.4 shows the exponentially-distributed failure rates of the components at different phases. Assume the durations of the three phases are τ_1 = 100 hours, τ_2 = 250 hours, and τ_3 = 150 hours. Given the exponentially-distributed failure rates in Table 8.4, the failure probabilities for the components at different time points can be computed using (6.2) in Chapter 6.

Table 8.5. Norms of the differences in the failure probability vectors obtained by the proposed stochastic approach and Monte Carlo (MC) simulation [125] for the PMS of Example 2.

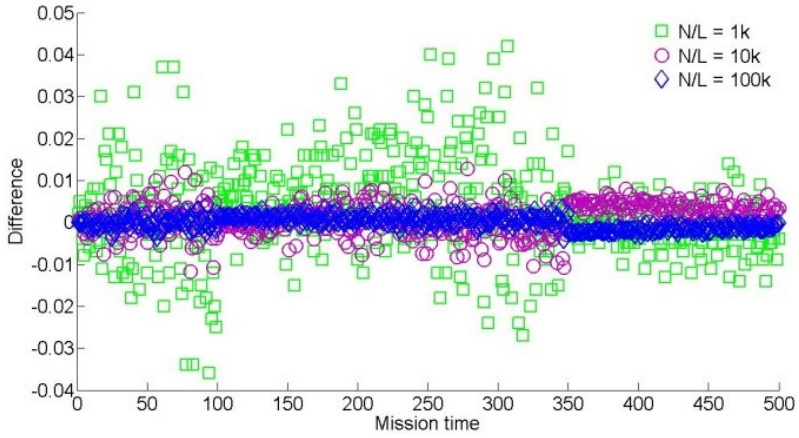
N/L		**1k**	**10k**	**100k**
$\|\Delta \boldsymbol{F}_{S-MC}\|_1$		6.0390	0.7980	0.2426
$\|\Delta \boldsymbol{F}_{S-MC}\|_2$		0.3123	0.0350	0.0157
$\|\Delta \boldsymbol{F}_{S-MC}\|_\infty$		0.0320	0.0040	0.0019
Avg. time (s)	Stochastic	0.84	6.11	63.51
	MC	2.61	26.19	237.33

Both the stochastic computational approach and MC simulation [125] are applied to find the failure probability/reliability of the PMS at any mission time. The accuracy of the stochastic approach is compared with the MC simulation. The norms of the disparity

vectors are given in Table 8.5 for the stochastic and MC approaches. The average run times of the stochastic approach and MC method are also shown.



(a)



(b)

Fig. 8.12. Failure probability of a phased-mission system (PMS) consisting of three stages for a mission time of 500 hours. (a) Failure probability and reliability obtained by the proposed stochastic approach with a sequence length of 10k bits (the durations of the three phases are τ_1 = 100 hours, τ_2 = 250 hours, and τ_3 = 150 hours). (b) Values of the difference between the failure probabilities for Example 1 obtained by the stochastic approach and Monte Carlo (MC) simulation.

As per the norms in Table 8.5, the proposed stochastic approach requires a shorter simulation time than the MC approach for the same value of sequence length/simulation

runs. Therefore, the stochastic approach is faster than the MC method, and provides a close result with a reasonable sequence length.

The failure probability for the PMS of Example 2 is plotted in **Fig. 8.12(a)** for a total mission time of 500 hours using the stochastic approach, while the difference between the obtained failure probability vectors for the stochastic and MC methods is illustrated in **Fig. 8.12(b)** for different sequence lengths or numbers of simulation runs. $\|\cdot\|_1$, $\|\cdot\|_2$, and $\|\cdot\|_\infty$ of the differences in the failure probability vectors obtained by the proposed stochastic and MC approaches are 0.7980, 0.0350 and 0.0040, respectively, for using a sequence length or sample size of 10k bits.

As can be seen from these results, the PMS (inclusive of static and dynamic gates, such as PAND and FDEP gates) can be easily evaluated by the proposed stochastic approach, using the stochastic PAND and FDEP models. Hence, a general PMS with dynamic behaviors can be evaluated by the stochastic approach. Moreover, the failures of common components in different phases can be effectively evaluated by the proposed stochastic models.

8.6. Summary

A stochastic model has been proposed for the analysis of a PMS. A PMS consisting of H phases is represented by H fault trees with each of them modeling the failure conditions of a phase. The fault tree for each phase can be constructed using stochastic logic gates. An OR gate model has been utilized to calculate the output stochastic sequence to indicate the failure probabilities of the H phases, i.e., for the entire system. Based on this analysis, the common components at different phases have been considered to determine whether their failure can cause the failure of the corresponding phase; different stochastic models have been proposed to compute the failure probabilities of the

components for each phase. If dynamic behaviors (such as functional dependency and priority relationships) are included in the relationships between components, stochastic models for dynamic gates, such as the PAND and the FDEP, can be utilized for stochastic analysis of a system. Hence, the stochastic model of a general PMS can be constructed with stochastic logic gates.

A general PMS has been evaluated using non-Bernoulli sequences as initial input probabilities. The accuracy of the stochastic analysis is affected by the simulated sequence length. In the case studies considered, it is shown that the accuracy of the stochastic approach is better than MC simulation with the same number of runs. Furthermore, the stochastic approach is capable of considering any failure distribution of the basic components; both exponential and Weibull distributions have been analyzed for the case studies considered.

Chapter 9

Conclusion and Future Work

9.1. Summary

This dissertation presents our work on the application of stochastic computational models on the analyses of gene regulatory networks (GRNs) and dynamic fault trees (DFTs).

Chapter 1 presents the motivation and major objectives of this dissertation. Chapter 2 introduces the background on stochastic computation, probabilistic Boolean networks (PBNs), stochastic Boolean networks (SBNs) and DFT analysis.

In Chapter 3, the context-sensitive stochastic Boolean network (CSSBN), is presented as an effective approach to modeling the effects of gene perturbation and intervention in a GRN. In a CSSBN, the state transition matrix (STM) can be computed with a reduced computational complexity compared to the use of a context sensitive probabilistic Boolean network (CSPBN). The dynamics of the p53-Mdm2 network and a glioma network are analyzed using the stochastic models, both in a context-switching environment with random gene perturbation. We have shown that random gene perturbation has a greater effect on the final distribution of the steady state compared to context switching activities.

In Chapter 4, stochastic multiple-valued networks (SMNs) are presented as a fast approach to modeling the effect of noise in a GRN. In an SMN, the use of randomly permuted sequences further increases computational efficiency and allows for a tunable tradeoff between accuracy and efficiency. A steady state analysis using a time-frame expansion technique has shown a significant speedup compared to either an accurate analysis or Monte Carlo (MC) simulation. It also produced more accurate results than MC simulation. Simulations of the SMNs have revealed the oscillatory dynamics of a

multiple-valued p53-Mdm2 network and have predicted the steady state distribution of a ternary WNT5A network with gene perturbation.

In Chapter 5, asynchronous stochastic Boolean networks (ASBNs) are introduced to model a gene network. Stochastic analyses are carried out for the asynchronous models of stochasticity in node (SIN) and stochasticity with propensity parameters (SPP). The dynamics of a GRN is studied under different asynchronous state update strategies. The ASBNs have been used to estimate the attractors of a T helper network of 23 genes and the robustness of the attractors have been demonstrated for the state-updating strategies in the T helper network. It has been shown that the SPP model accurately reveal the state transition from $Th0$ to $Th1$ under the dosage of $IL-12$.

In Chapter 6, a stochastic model is proposed for the analysis of a two-input priority AND (PAND) gate in a DFT. A successive cascading structure of this model is then constructed for the analysis of a general multiple-input PAND gate. For a DFT with PAND gates, a stochastic approach using the proposed models provides a fast analysis of the DFT compared to an analytical or algebraic approach. The failure probability of a basic event is not limited to an exponential distribution and repeated events are readily handled in the stochastic approach. It also avoids the state-space explosion problem or the large computational complexity typically encountered in Markov or other analytical methods, thus it is scalable for use in a general DFT analysis.

In Chapter 7, stochastic models are proposed for analyzing a two-input spare gate and probabilistic common cause failures (PCCFs) in a DFT. The warm spare (WSP), and cold spare (CSP) gates have been analyzed in detail. The effect of PCCFs has been taken into consideration and a stochastic logic model has been constructed for dependent PCCFs. The efficiency and accuracy of the proposed stochastic approach have been shown by the case studies of several benchmark systems.

Phased-mission systems (PMSs) are considered as another application of stochastic

computational models in Chapter 8. A PMS consisting of H phases is modeled by H fault trees, each of which can be constructed using stochastic logic gates. The common components in different phases have been considered to determine whether their failure can cause the failure of the corresponding phase. In the case studies considered, it is shown that the accuracy of the stochastic approach is higher than MC simulation with an equivalent number of runs.

9.2. Future Work

1) Signaling Pathway Analysis

The logical representation and analysis of a signaling pathway provides valuable insight into a biological system. The investigation of vulnerable components in such a pathway can contribute to the development of drug therapy addressing aberrations in that signaling pathway. The stochastic computational models are potentially applicable to analyzing the vulnerability of the components in multiple signaling pathways involved in different cancer cells. The inhibition/activation relationship among genes can be modeled by stochastic logic, thus a stochastic pathway can be constructed with logic gates. The computational results will be validated by experiments, where selected proteins will be silenced using siRNAs and the viability of the cells will be analyzed after silencing. We are currently collaborating with Dr. Hasan Uludag's group on this work.

2) System Reliability Evaluation

The stochastic computational approach is potentially applicable to several emerging problems in system reliability evaluation. For instance, the stochastic model for the 2-out-of-3 majority voter is presented in Chapter 7. This stochastic model can be generalized for k-out-of-n and weighted k-out-of-n voters. For a binary weighted k-out-of-n system [165], each component i is associated with a weight of w_i. The system functions if and only if the total weight of the functional components is greater than a threshold k. The accurate analysis of a large weighted k-out-of-n voter is

difficult because of the size of the problem. The current stochastic analysis is based on the non-repairable property, so the repair of faulty components [166] can be incorporated into a system reliability evaluation using binary or multiple-state fault trees [167][168]. The stochastic logic models could be readily used to evaluate the reliability of a network and to identify the vulnerable components that are critical to the robustness of a network.

3) Signal Correlation Reduction

Despite the recent development of stochastic computation (such as in [29][59][169]), inaccuracy remains a problem that limits its application. Application-dependent measures of the statistical similarity between the random bit streams are investigated in [169] and [170]. The impact of signal correlation on the accuracy of stochastic computation is analyzed in [171]. Different approaches are proposed to reduce signal correlation such as using the regeneration and isolation of stochastic numbers [171].

MC simulation is subject to a probabilistic error bound and the generation of random samples is usually expensive. The random fluctuation in MC simulation is reduced at a rate of $1/\sqrt{N}$, where N is the number of random samples. The stochastic approach using non-Bernoulli sequences of fixed number of zeros and ones is shown to be faster than MC simulation and requires a less expensive random number generation [29], however the error is likely to decrease at a rate of the same order as MC simulation. The Quasi Monte Carlo method uses carefully chosen and thus deterministic samples [172], so a smaller error bound is obtained for an improved accuracy. In general, what types of stochastic sequences to use remains a key issue to be investigated in future work, because a better sequence may indicate a shorter sequence length required to achieve a certain accuracy, thus significantly improving the efficiency of stochastic computation.

Publication List

Journals

1) **P. Zhu**, J. Liang, and J. Han, "Gene Perturbation and Intervention in Context-Sensitive Stochastic Boolean Networks," BMC System Biology, 8–60, 2014.
2) **P. Zhu**, and J. Han, "Asynchronous stochastic Boolean networks as gene network models," Journal of Computational Biology, **21**(10): 760-770, 2014.
3) **P. Zhu**, and J. Han, "Stochastic Multiple-Valued Gene Networks," IEEE Transactions on Biomedical Circuits and Systems, **8**(1):42–53, 2014.
4) **P. Zhu**, J. Han, L. Liu, and M. J. Zuo, "A Stochastic Approach for the Analysis of Fault Trees with Priority AND Gates," IEEE Transactions on Reliability, **63**(2): 480-494, 2014.
5) **P. Zhu**, J. Han, L. Liu, and F. Lombardi, "A Stochastic Approach for the Analysis of Dynamic Fault Trees with Spare Gates under Probabilistic Common Cause Failures," IEEE Transactions on Reliability, **PP**(99): 1-15, 2015.
6) J. Han, H. Chen, J. Liang, **P. Zhu**, Z. Yang, and F. Lombardi, "A Stochastic Computational Approach for Accurate and Efficient Reliability Evaluation," IEEE Transactions on Computers, **63**(6): 1336-1350, 2014.

Papers under review

1) **P. Zhu**, J. Han, L. Liu, and F. Lombardi, "Reliability Evaluation of Phased-Mission Systems Using Stochastic Computation," submitted to IEEE Transactions on Reliability, 2015.

Book chapter

1) **P. Zhu,** J. Liang, and J. Han, "Toward Intracellular Delivery and Drug Discovery:

Stochastic Logic Networks as Efficient Computational Models for Gene Regulatory Networks," a chapter in *Intracellular Delivery Vol. II, Fundamental Biomedical Technologies, Volume 7, 2014, pp 327-359. Springer Netherlands: Dordrecht.*

Bibliography

[1] M.B. Elowitz, A.J. Levine, E.D. Siggia, and P.S. Swain, "Stochastic Gene Expression in a Single Cell," *Science*, vol. 297, no. 5584, pp. 1183–1186, 2002.

[2] R. Iyengar, "Computational Biochemistry: Systems Biology Minireview Series," *Journal of Biology Chemistry*, vol. 284, pp. 5425-5426, 2009.

[3] G. Karlebach, and R. Shamir, "Modelling and Analysis of Gene Regulatory Networks," *Nature Reviews Molecular Cell Biology*, vol. 9, pp. 770-780, 2008.

[4] J.M. Pedraza, and A. van Oudenaarden, "Noise Propagation in Genetic Networks," *Science*, vol. 307, no. 5717, pp. 1965–1969, 2005.

[5] S.A. Kauffman, "Metabolic Stability and Epigenesis in Randomly Constructed Genetic Nets," *Journal of Theoretical Biology* 1969, vol. 22, no. 3, pp. 437–467.

[6] E. Klipp, "Systems Biology in Practice: Concepts, Implementation and Application," Wiley-VCH, Weinheim, Germany, 2005.

[7] K.C. Chen, *et al.*, "Integrative Analysis of Cell Cycle Control in Budding Yeast," *Molecular Biology of the Cell*, vol. 15, no. 8, pp. 3841–3862, 2004.

[8] S. Li, *et al.*, "A Quantitative Study of the Division Cycle of Caulobacter Crescentus Stalked Cells," *PLoS Computational Biology*, vol. 4, no. 1, pp. e9, 2008.

[9] L. Qian, H. Wang, and E. Dougherty, "Inference of Noisy Nonlinear Differential Equation Models for Gene Regulatory Networks using Genetic Programming and Kalman Filtering," *IEEE Transactions on Signal Processing*, vol. 56, no. 7, pp. 3327–3339, 2008.

[10] D.T. Gillespie, "A General Method for Numerically Simulating the Stochastic Time Evolution of Coupled Chemical Reactions," *Journal of Computational Physics*, vol. 22, pp. 403-434, 1976.

[11] D.T. Gillespie, "Exact Stochastic Simulation of Coupled Chemical Reactions," *Journal of Physical Chemistry*, vol. 81, pp. 2340–2361, 1977.

[12] H. de Jong, "Modeling and Simulation of Genetic Regulatory Systems: a Literature Review," *Journal of Computational Biology*, vol. 9, no. 1, pp. 67-103, 2002.

[13] M. Gibson, and J. Bruck, "Efficient Exact Stochastic Simulation of Chemical Systems with Many Species and Many Channels," *Journal of Physical Chemistry*, vol. 104, pp. 1876–1889, 1999.

[14] D.T. Gillespie, "Approximate Accelerated Stochastic Simulation of Chemically Reacting Systems," *Journal of Chemical Physics*, vol. 115, pp. 1716-1733, 2001.

[15] M. Davidich, and S. Bornholt, "Boolean Network Model Predicts Cell Cycle Sequence of Fission Yeast," *PLoS One*, vol. 3, no. 2, pp. e1672, 2008.

[16] C.E. Giacomantonio, and G.J. Goodhill, "A Boolean Model of the Gene Regulatory Network underlying Mammalian Cortical Area Development," *PLoS Computational Biology*, vol, 6, no. 9, pp. e1000936, 2010.

[17] S. Huang, "Gene Expression Profiling, Genetic networks, and Cellular States: an Integrating Concept for Tumorigenesis and Drug Discovery," *Journal of Molecular Medicine*, vol. 77, no. 6, pp. 469–480, 1999.

[18] I. Shmulevich, E.R. Dougherty, and W. Zhang, "From Boolean to Probabilistic Boolean Networks as Models of Genetic Regulatory Networks," *Proceedings of the IEEE*, vol. 90, no. 11, pp. 1778–1792, 2002.

[19] I. Shmulevich, E.R. Dougherty, and W. Zhang, "Gene Perturbation and Intervention in Probabilistic Boolean Networks," *Bioinformatics*, vol. 18, no. 10, pp. 1319-1331, 2008.

[20] I. Shmulevich, and E.R. Dougherty, "Probabilistic Boolean Networks: the Modeling and Control of Gene Regulatory Networks," Philadelphia, SIAM, 2009.

[21] H.H. McAdams, and L. Shapiro, "Circuit Simulation of Genetic Networks," *Science*, vol. 269, no. 5224, pp. 650-656, 1995.

[22] A. Abdi, M.B. Tahoori, and E.S. Emamian, "Fault Diagnosis Engineering of Digital Circuits can Identify Vulnerable Molecules in Complex Cellular Pathways," *Science Signal*, vol. 1, no. 42, pp. ra10, 2008.

[23] R. Adar, *et al*., "Stochastic Computing with Biomolecular Automata," *PNAS*, vol. 101, no. 27, pp. 9960–9965, 2004.

[24] Y. Benenson, *et al*., "An Autonomous Molecular Computer for Logical Control of Gene Expression," *Nature*, vol. 429, pp. 423–429, 2004.

[25] B.R. Gaines, "Stochastic computing systems," Advances in Information Systems, *Science*, vol. 2, pp. 37–172, 1969.

[26] W.J. Poppelbaum, C. Afuso, and J.W. Esch, "Stochastic Computing Elements and Systems," *Proceedings of AFIPS Fall Joint Computer Conference*, pp. 635-644, 1967.

[27] A. Alaghi, and J.P. Hayes, "Survey of Stochastic Computing," *ACM Transactions on Embedded Computing Systems*, vol. 12, no. 92, pp. 1-19, 2013.

[28] S.T. Ribeiro, "Random-pulse Machines," *IEEE Transactions on Electronic Computers*, vol. 16, no. 3, pp. 261–276, 1967.

[29] J. Han, *et al.*, "A Stochastic Computational Approach for Accurate and Efficient Reliability Evaluation," *IEEE Transactions on Computers*, vol. 63, no. 6, pp. 1336-1350, 2014.

[30] J. Liang, and J. Han, "Stochastic Boolean Networks: an Efficient Approach to Modeling Gene Regulatory Networks," *BMC Systems Biology*, vol. 6, pp. 113, 2012.

[31] G. Lahav, *et al.*, "Dynamics of the p53-Mdm2 Feedback Loop in Individual Cells," *Nature Genetics*, vol. 36, no. 2, pp. 147–150, 2004.

[32] S. Martin, Z. Zhang, A. Martino, and J.-L. Faulon, "Boolean Dynamics of Genetic Regulatory Networks Inferred from Microarray Time Series Data," *Bioinformatics*, vol. 23, no. 7, pp. 866-874, 2007.

[33] S. Kim, *et al.*, "Can Markov Chain Models Mimic Biological Regulation?" *Journal of Biological Systems*, vol. 10, no. 4, pp. 337–357, 2002.

[34] C.A. Ericson II, "Fault Tree Analysis – a History," *In Proceedings of the 17th International System Safety Conference*, pp. 16–21, 1999.

[35] N.G. Leveson, "Safeware: System Safety and Computers," Addison-Wesley, 1995.

[36] 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*, vol. 7, no. 2, pp. 128–143, 2010.

[37] M. Stamatelatos, and W. Vesely, "Fault Tree Handbook with Aerospace Applications," NASA Office of Safety and Mission Assurance, version 1.1, pp. 1–205, 2002.

[38] E.J. Henley, and H. Kumamoto, "Reliability Engineering and Risk Assesment," Englewood Cliffs: Prentice Hall, 1981.

[39] T. Yuge, and S. Yanagi, "Quantitative Analysis of a Fault Tree with Priority AND Gates," *Reliability Engineering & System Safety*, vol. 93, no. 11, pp. 1577–1583, 2008.

[40] S. Amari, G. Dill, and E. Howald, "A New Approach to Solve Dynamic Fault Trees," *In Annual IEEE reliability and maintainability symposium*, pp. 374–9, 2003.

[41] J.D. Esary, and H. Ziehms, "Reliability analysis of Phased-missions," Reliability and Fault-Tree Analysis, pp. 213-236 (Society for Industrial Applied Mathematics, Philadelphia, Pennsylvania), 1975.

[42] M. Dunlop, *et al*., "Regulatory Activity Revealed by Dynamic Correlations in Gene Expression Noise," *Nature Genetics*, vol. 40, pp. 1493–1498, 2008.

[43] R. Pal, A. Datta, M.L. Bittner, and E.R. Dougherty, "Intervention in Context-sensitive Probabilistic Boolean Networks," *Bioinformatics*, vol. 21, no. 7, pp. 1211–1218, 2005.

[44] I. Shmulevich, *et al*., "Steady-state Analysis of Genetic Regulatory Networks Modeled by Probabilistic Boolean Networks," *Comparative and Functional Genomics*, vol. 4, no. 6, pp. 601–608, 2003.

[45] J.S. Rosenthal, "Minorization Conditions and Convergence Rates for Markov Chain Monte Carlo," *Journal of the American Statistical Association*, vol. 90, no. 430, pp. 558–566, 1995.

[46] I. Harvey, and T. Bossomaier, "Time Out of Joint: Attractors in Asynchronous Random Boolean Networks," *Proceedings of the 4th European Conference on Artificial Life* (*ECAL*1997), (MIT Press), pp. 67–75, 1997.

[47] H. Kitano, "Foundations of Systems Biology," (MIT Press Cambridge, Massachusetts London), 2001.

[48] R. Thomas, and R. D'Ari, "*Biological Feedback*," CRC Press, 1990.

[49] M.K. Morris, J. Saez-Rodriguez, P.K. Sorger, and D.A. Lauffenburger, "Logic-based Models for the Analysis of Cell Signaling Networks," *Biochemistry*, vol. 49, no. 15, pp. 3216–3224, 2010.

[50] E. Dubrova, "Random Multiple-Valued Networks: Theory and Applications," *Proceedings of the 36th International Symposium on Multiple-Valued Logic* (*ISMVL '06*), pp. 27-33, 2006.

[51] A. Garg, L. Mendoza, I. Xenarios, and G. DeMicheli, "Modeling of Multiple Valued Gene Regulatory Networks," *In Proceedings of the 29th International Conference of the IEEE Engineering in Medicine and Biology Society (EMBS '07)*, pp. 1398-1404, 2007.

[52] F. Greil, B. Drossel, and J. Sattler, "Critical Kauffman Networks under Deterministic Asynchronous Update," *New Journal of Physics*, vol. 9, pp. 373, 2007.

[53] K. Klemm, and S. Bornholdt, "Stable and Unstable Attractors in Boolean Networks," *Physical Review E, Statistical, Nonlinear, and Soft Matter Physics*, vol. 72, pp. 055101, 2005.

[54] L. Xing, A. Shrestha, L. Meshkat, and W. Wang, "Incorporating Common-cause Failures into the Modular Hierarchical Systems Analysis," *IEEE Transactions on Reliability*, vol. 58, no. 1, pp. 10-19, 2009.

[55] K.P. Parker, and E.J. McCluskey, "Probabilistic treatment of general combinational networks," *IEEE Transactions on Computers*, vol. C-**24, no.** 6, pp. 668–670, 1975.

[56] I. Bahar, J.L. Mundy, and J. Chen, "A Probabilistic-based Design Methodology for Nanoscale Computation," *In Proceedings of International Conference on Computer Aided Design*, pp. 480–486, 2003.

[57] H. Chen, and J. Han, "Stochastic Computational Models for Accurate Reliability Evaluation of Logic Circuits," *In Proceedings of the 20th symposium on Great Lakes Symposium on VLSI (GLVLSI)*, Providence, RI, USA, pp. 61-66, 2010.

[58] D. Braendler, T. Hendtlass, and P. O'Donoghue, "Deterministic Bit-stream Digital Neurons," *IEEE Transactions on Neural Networks*, vol. 13, no. 6, pp. 1514-1525, 2002.

[59] P.K. Gupta, and R. Kumaresan, "Binary Multiplication with PN Sequences," *IEEE Transactions on Acoustics, Speech and Signal Processing*, vol. 36, no. 4, pp. 603-606, 1988.

[60] H. Lähdesmäki, *et al.*, "Relationships between Probabilistic Boolean Networks and Dynamic Bayesian Networks as Models of Gene Regulatory Networks," *Signal Processing*, vol. 86, no. 4, pp. 814-834, 2006.

[61] W. Ching, S. Zhang, M. NG, and T. Akutsu, "An Approximation Method for Solving the Steady-state Probability Distribution of Probabilistic Boolean Networks," *Bioinformatics*, vol. 23, no. 12, pp. 1511 – 1518, 2007.

[62] N. Guelzim, S. Bottani, P. Bourgine, and F. Kepes, "Topological and Causal Structure of the Yeast Transcriptional Regulatory Network," *Nature Genetics*, vol. 31, no. 1, pp. 60–63, 2002.

[63] S. Zhang, W. Ching, M. NG, and T. Akutsu, "Simulation Study in Probabilistic Boolean Network Models for Genetic Regulatory Networks," *International Journal of Data Mining and Bioinformatics*, vol. 1, no. 3, pp. 217–240, 2007.

[64] I. Shmulevich, E.R. Dougherty, S. Kim, and W. Zhang, "Probabilistic Boolean Networks: a Rule-based Uncertainty Model for Gene Regulatory Networks," *Bioinformatics*, vol. 18, no. 2, pp. 261-274, 2002.

[65] M.A. Boyd, "Dynamic Fault Tree Models: Techniques for Analyses of Advanced Fault Tolerant Computer Systems," Ph.D dissertation, Department of Computer Science, Duke University, 1991.

[66] J.B. Dugan, S.J. Bavuso, and M.A. Boyd, "Dynamic Fault Tree Models for Fault-tolerant Computer Systems," *IEEE Transactions on Reliability*, vol. 41, no. 3, pp. 363–377, 1992.

[67] A. Ejlali, and S. Miremadi, "FPGA-based Monte Carlo Simulation for Fault Tree Analysis," *Microelectronics Reliability*, vol. 44, no. 6, pp. 1017–1028, 2004.

[68] G. Merle, J.-M.Roussel, and J.-J. Lesage, "Improving the Efficiency of Dynamic Fault Tree Analysis by Considering Gates FDEP as Static," *In Proceedings of the European Safety & Reliability Conference 2010* (*ESREL2010*), pp. 845–851, 2010.

[69] S.J. Bavuso, "A Novel Solution-technique Applied to a Novel WAAS Architecture," *In Proceedings of Annual Reliability and Maintainability Symposium (RAMS'98)*, USA, pp. 229–234, 1998.

[70] L. Xing, O. Tannous, and J.B. Dugan, "Reliability Analysis of Nonrepairable Cold-standby Systems using Sequential Binary Decision Diagrams," *IEEE Transactions on Systems, Man, Cybernetics–Part A: Systems and Humans*, vol. 42, no. 3, pp. 715–726, 2012.

[71] J.B. Dugan, S.J. Bavuso, and M.A. Boyd, "Fault Trees and Sequence Dependencies," *In Proceedings of the Reliability and Maintainable Symposium*, pp. 286–93, 1990.

[72] R. Remenyte-Prescott, J.D. Andrews, and P.W.H. Chung, "An Efficient Phased-mission Reliability Analysis for Autonomous Vehicles," *Reliability Engineering & System Safety*, vol. 95, no. 3, pp. 226–35, 2010.

[73] M. Brun, E.R. Dougherty, and I. Shmulevich, "Steady-state Probabilities for Attractors in Probabilistic Boolean Networks," *Signal Processing*, vol. 85, no. 10, pp. 1993–2013, 2005.

[74] P. Zhu, J. Liang, and J. Han, "Gene Perturbation and Intervention in Context-Sensitive Stochastic Boolean Networks," *BMC Systems Biology*, vol. 8, pp. 60, 2014.

[75] B. Faryabi, *et al*., "Intervention in Context-sensitive Probabilistic Boolean Networks Revisited," *EURASIP Journal on Bioinformatics and Systems Biology*, vol. 2009, pp. 360864, 2009.

[76] R. Pal, "Context-sensitive Probabilistic Boolean Networks: Steady-state Properties, Reduction, and Steady-state Approximation," *IEEE Transactions on Signal Processing*, vol. 58, no. 2, pp. 879–890, 2010.

[77] A. Datta, A. Choudhary, M.L. Bittner, and E.R. Dougherty, "External Control in Markovian Genetic Regulatory Networks," *Machine Learning*, vol. 52, pp. 169–191, 2003.

[78] A. Datta, A. Choudhary, M.L. Bittner, and E.R. Dougherty, "External Control in Markovian Genetic Regulatory Networks: the Imperfect Information Case," *Bioinformatics*, vol. 20, no. 6, pp. 924–993, 2004.

[79] R. Pal, A. Datta, M.L. Bittner, and E.R. Dougherty, "Optimal Infinite Horizon Control for Probabilistic Boolean Networks," *IEEE Transactions on Signal Processing*, vol. 54, no. 6, pp. 2375–2387, 2006.

[80] B. Faryabi, A. Datta, and E.R. Dougherty, "On Approximate Stochastic Control in Genetic Regulatory Networks," *IET Systems Biology*, vol. 1, no. 6, pp. 361–368, 2007.

[81] R. Layek, A. Datta, R. Pal, and E.R. Dougherty, "Adaptive Intervention in Probabilistic Boolean Networks," *Bioinformatics*, vol. 25, no. 16, pp. 2042–2048, 2009.

[82] X. Qian, and E.R. Dougherty, "Effect of Function Perturbation on the Steady-state Distribution of Genetic Regulatory Networks: Optimal Structural Intervention," *IEEE Transactions on Signal Processing*, vol. 56, no. 10, pp. 4966-4976, 2008.

[83] X. Qian, I. Ivanov, N. Ghaffari, and E.R. Dougherty, "Intervention in Gene Regulatory Networks via Greedy Control Policies based on Long-run Behavior," *BMC Systems Biology*, vol. 3, pp. 16, 2009.

[84] R.A. Weinberg, "*The Biology of Cancer*," 1st edition. New York: Garland Science, 2006.

[85] B. Vogelstein, D. Lane, and A.J. Levine, "Surfing the p53 Network," *Nature*, vol. 408, no. 6810, pp. 307–310, 2000.

[86] E. Batchelor, A. Loewer, and G. Lahav, "The Ups and Downs of p53: Understanding Protein Dynamics in Single Cells," *Nature Reviews Cancer*, vol. 9, pp. 371-377, 2009.

[87] A. Ciliberto, B. Novak, and J.J. Tyson, "Steady States and Oscillations in the p53-Mdm2 Network," *Cell Cycle*, vol. 4, no. 3, pp. 486–493, 2005.

[88] W. Abou-Jaoude, D. Ouattara, and M. Kaufman, "From Structure to Dynamics: Frequency Tuning in the p53-mdm2 Network: I. Logical Approach," *Journal of Theoretical Biology*, vol. 258, no. 4, pp. 561–577, 2009.

[89] D. Murrugarra, *et al*., "Modeling Stochasticity and Variability in Gene Regulatory Networks," *EURASIP Journal on Bioinformatics and Systems Biology*, vol. 2012, pp. 5, 2012.

[90] K. Maurice, "The Map Method for Synthesis of Combinational Logic Circuits," *Transactions of the American Institute of Electrical Engineers part I* , vol. 72, no. 9, pp. 593–599, 1953.

[91] H. Ge, and M. Qian, "Boolean Network Approach to Negative Feedback Loops of the p53 Pathways: Synchronized Dynamics and Stochastic Limit Cycles," *Journal of Computational Biology*, vol. 16, no. 1, pp. 119–132, 2009.

[92] Z. Li, and D. Cheng, "Algebraic Approach to Dynamics of Multivalued Networks," *International Journal of Bifurcation and Chaos*, vol. 20, no. 3, pp. 561– 582, 2010.

[93] A. Adamatzky, "On Dynamically Non-trivial Three-valued Logics: Oscillatory and Bifurcatory Species," *Chaos, Solitons & Fractals*, vol. 18, no. 5, pp. 917–936, 2003.

[94] L.G. Volker, and M. Conrad, "The Role of Weak Interactions in Biological Systems: the Dual Dynamic Model," *Journal of Theoretical Biology*, vol. 193, no. 2, pp. 287–306, 1998.

[95] P. Zhu, and J. Han, "Stochastic Multiple-Valued Gene Networks," *IEEE Transactions on Biomedical Circuits and Systems*, vol. 8, no. 1, pp. 42-53, 2014.

[96] P. Zhu, J. Liang, and J. Han, "Toward Intracellular Delivery and Drug Discovery: Stochastic Logic Networks as Efficient Computational Models for Gene Regulatory Networks," a chapter in the book "Intracellular Delivery Vol. 2," Springer, 2014.

[97] L. Kadanoff，M. Aldana, and S.N. Coppersmith, "Boolean Dynamics with Random Couplings," *Springer Applied Mathematical Sciences Series, Special volume, Springer, New York*, pp. 23–89, 2003.

[98] C. Luo, and X. Wang, "Dynamics of Random Boolean Networks under Fully Asynchronous Stochastic Update Based on Linear Representation," *PLoS One*, vol. 8, no. 6, pp. e66491, 2013.

[99] N. Geva-Zatorsky, *et al*., "Oscillations and Variability in the p53 System," *Molecular Systems Biology*, vol. 2, no. 1, pp. 2: 2006.0033, 2006.

[100] A. Naldi, D. Thieffry, and C. Chaouiya, "Decision Diagrams for the Representation and Analysis of Logical Models of Genetic Networks," *Computational Methods in Systems Biology*, *Lecture Notes in Computer Science*, vol. 4695, pp. 233–247, 2007.

[101] E. Remy, *et al.*, "From Logical Regulatory Graphs to Standard Petri Nets: Dynamical Roles and Functionality of Feedback Circuits," *Transactions on Computer Systems Biology Springer Lecture Notes in Computer Science*, vol. 4230, pp. 56–72, 2006.

[102] A. Garg, *et al*., "Synchronous versus Asynchronous Modeling of Gene Regulatory Networks," *Bioinformatics*, vol. 24, no. 17, pp. 1917-1925, 2008.

[103] B.S. Chen, and P.W. Chen, "Robust Engineered Circuit Design Principles for Stochastic Biochemical Networks with Parameter Uncertainties and Disturbances," *IEEE Transactions on Biomedical Circuits and Systems*, vol. 2, no. 2, pp. 114-132, 2008.

[104] F.X. Wu, "Global and Robust Stability Analysis of Genetic Regulatory Networks with Time-varying Delays and Parameter Uncertainties," *IEEE Transactions on Biomedical Circuits and Systems*, vol. 5, no. 5, pp. 391–398, 2011.

[105] A.S. Ribeiro, and S.A. Kauffman, "Noisy Attractors and Ergodic Sets in Models of Gene Regulatory Networks," *Journal of Theoretical Biology*, vol. 247, no. 4, pp. 743–755, 2007.

[106] K. Willadsena, and J. Wiles, "Robustness and State-space Structure of Boolean Gene Regulatory Models," *Journal of Theoretical Biology*, vol. 249, no. 4, pp. 749–765, 2007.

[107] P. Zhu, and J. Han, "Asynchronous Stochastic Boolean Networks as Gene Network Models," *Journal of Computational Biology*, vol. 21, no. 10, pp. 771-783, 2014.

[108] M. Wu, X. Yang, and C. Chan, "A Dynamic Analysis of IRS-PKR Signaling in Liver Cells: a Discrete Modeling Approach," *PLoS One*, vol. 4, no. 12, pp. e8040, 2009.

[109] A. Garg, *et al*., "Modeling Stochasticity and Robustness in Gene Regulatory Networks," *Bioinformatics*, vol. 25, no. 12, pp. i101-9, 2009.

[110] L. Mendoza, and I. Xenarios, "A Method for the Generation of Standardized Qualitative Dynamical Systems of Regulatory Networks," *Theoretical Biology and Medical Modelling*, vol. 3, pp. 13, 2006.

[111] M. Kaern, *et al*., "Stochasticity in Gene Expression: from Theories to Phenotypes," *Nature Reviews Genetics*, vol. 6, no. 6, pp. 451–464, 2005.

[112] D. Agnello, *et al*., "Cytokines and Transcription Factors that Regulate T Helper Cell Differentiation: New Players and New Insights," *Journal of Clinical Immunology*, vol. 23, no. 3, pp. 147-161, 2003.

[113] C. Bergmann, J.L. Van Hemmen, and L.A. Segel, "Th1 or Th2: How an Appropriate T Helper Response can be Made," *Bulletin of Mathematical Biology*, vol. 63, no. 3, pp. 405–430, 2001.

[114] K.M. Murphy, and S.L. Reiner, "The Lineage Decisions on Helper T Cells," *Nature Reviews Immunology*, vol. 2, no. 12, pp. 933–944, 2002.

[115] 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*, vol. 59, no. 1, pp. 250-261, 2010.

[116] P. Zhu, J. Han, L. Liu, and M.J. Zuo, "A Stochastic Approach for the Analysis of Fault Trees with Priority AND Gates," *IEEE Transactions on Reliability*, vol. 63, no. 2, pp. 480-494, 2014.

[117] H. Boudali, P. Crouzen, and M. Stoelinga, "Dynamic Fault Tree Analysis through Input/Output Interactive Markov Chains," *In Proceedings of the International Conference on Dependable Systems and Networks (DSN)*, pp. 25–38, 2007.

[118] R. Manian, J.B. Dugan, D. Coppit, and K.J. Sullivan, "Combining Various Solution Techniques for Dynamic Fault Tree Analysis of Computer Systems," *In Proceedings of the 3rd IEEE International Symposium on High-Assurance Systems Engineering (HASE'98)*, pp. 21–28, 1998.

[119] A. Bobbio, L. Portinale, M. Minichino, and E. Ciancamerla, "Improving the Analysis of Dependable Systems by Mapping Fault Trees into Bayesian Networks," *Reliability Engineering & System Safety*, vol. 71, no. 3, pp. 249-260, 2001.

[120] H. Boudali, and J.B. Dugan, "A Discrete-time Bayesian Network Reliability Modeling and Analysis Framework," *Reliability Engineering & System Safety*, vol. 87, no. 3, pp. 337–349, 2005.

[121] L. Xing, "An Efficient Binary-decision-diagram-based Approach for Network Reliability and Sensitivity Analysis," *IEEE Transactions on Systems, Man, and Cybernetics*, vol. 38, no. 1, pp. 105-115, 2007.

[122] O. Tannous, L. Xing, and J.B. Dugan, "Reliability Analysis of Warm Standby Systems using Sequential BDD," *In Proceedings of the 57th Annual Reliability &Maintainability Symposium*, FL, USA, pp. 1-7, 2011.

[123] A. Rauzy, "Sequence Algebra, Sequence Decision Diagrams and Dynamic Fault Trees," *Reliability Engineering & System Safety*, vol. 96, no. 7, pp. 785-792, 2011.

[124] L. Xing, A. Shrestha, and Y. Dai, "Exact Combinatorial Reliability Analysis of Dynamic Systems with Sequence-Dependent Failures," *Reliability Engineering & System Safety*, vol. 96, no. 10, pp. 1375-1385, 2011.

[125] RK. Durga, *et al*., "Dynamic Fault Tree Analysis using Monte Carlo Simulation in Probabilistic Safety Assessment," *Reliability Engineering & System Safety*, vol. 94, no. 4, pp. 872–83, 2009.

[126] W. Long, T.L. Zhang, Y.F. Lu, and M. Oshima, " On the Quantitative Analysis of Sequential Failure Logic using Monte Carlo Method for Different Distributions," *In Proceedings of Probabilistic Safety Assessment Manage*, pp. 391–396, 2002.

[127] H. Aliee, and H.R. Zarandi, "A Fast and Accurate Fault Tree Analysis Based on Stochastic Logic Implemented on Field-Programmable Gate Arrays," *IEEE Transactions on Reliability*, vol. 62, no. 1, pp. 13-22, 2013.

[128] J. von Neumann, "Probabilistic Logics and the Synthesis of Reliable Organisms from Unreliable Components," *Automata Studies, Shannon C.E. & McCarthy J., eds.,* Princeton University Press, pp. 43-98, 1956.

[129] J. Han, "*Fault-Tolerant Architectures for Nanoelectronic and Quantum Devices*," Universal Press, Veenendaal, the Netherlands, 2004. A Ph.D. dissertation of the Delft University of Technology, 1-135. ISBN: 90-9018888-6.

[130] L. Xing and W. Wang, "Probabilistic Common-cause Failures Analysis," *In Proceedings of the 54th Annual Reliability & Maintainability Symposium*, pp. 354-358, 2008.

[131] P. Zhu, J. Han, L. Liu, and F. Lombardi, "A Stochastic Approach for the Analysis of Dynamic Fault Trees with Spare Gates under Probabilistic Common Cause Failures," *IEEE Transactions on Reliability*, **vol. PP, no.** 99, pp. 1-15, 2015.

[132] J.D. Andrews, and J.B. Dugan, "Dependency Modeling using Fault Tree Analysis," *In Proceedings of the 17th International System Safety Conference*, USA, Aug, 1999.

[133] J.B. Dugan, and S.A. Doyle, "New Results in Fault Tree Analysis," *Tutorial notes of Annual Reliability & Maintainability Symposium*, (Jan.) 1997.

[134] K.B. Misra (Editor), "Handbook of Performability Engineering," Springer-Verlag, London, ISBN: 978-1-84800-130-5, (Oct.) 2008.

[135] J.B. Dugan, S.J. Bavuso, and M.A. Boyd, "Dynamic Fault-tree Models for Fault-tolerant Computer Systems," *IEEE Transactions on Reliability*, vol. 41**,** no. 3, pp. 363–377, 1992.

[136] B.W. Johnson, "Design and Analysis of Fault Tolerant Digital Systems," Reading, MA: Addison-Wesley, 1989.

[137] J. She, and M. Pecht, "Reliability of a *k*-out-of-*n* Warm-standby System," *IEEE Transactions on Reliability*, vol. 41, no. 1, pp. 72-75, 1991.

[138] Y.S. Dai, M. Xie, K.L. Poh, and S.H. Ng, "A Model for Correlated Failures in N-version Programming," *IIE Transactions*, vol. 36, no. 12, pp. 1183–1192, 2004.

[139] K.N. Fleming, and A. Mosleh, "Common-cause Data Analysis and Implications in System Modeling," *In Proceedings of the International Topical Meeting on Probabilistic Safety Methods and Applications*, vol. 1, pp. 3.1–3.14, 1985.

[140] Z. Tang, H. Xu, and J.B. Dugan, "Reliability Analysis of Phased Mission Systems with Common Cause Failures," *In Proceedings of the 51st Annual Reliability and Maintainability Symposium*, pp. 313–318, 2005.

[141] J.K. Vaurio, "An Implicit Method for Incorporating Common-cause Failures in System Analysis," *IEEE Transactions on Reliability*, vol. 47, no. 2, pp. 173–180, 1998.

[142] L. Xing, L. Meshkat, and S. Donohue, "An Efficient Approach for the Reliability Analysis of Phased-mission Systems with Dependent Failures," *In Proceedings of the 8th International Conference on Probabilistic Safety Assessment and Management (PSAM8)*, New Orleans, LA, USA, May 14–19, 2006.

[143] L.B. Page, and J.E. Perry, "A Model for System Reliability with Common-cause Failures," *IEEE Transactions on Reliability*, vol. 38, no. 40, pp. 406–410, 1989.

[144] J. Han, E. Boykin, H. Chen, J. Liang, and J. Fortes, "On the Reliability of Computational Structures using Majority Logic," *IEEE Transactions on Nanotechnology*, vol. 10, no. 5, pp. 1099-1112, 2011.

[145] G. Merle, J.M. Roussel, and J.J. Lesage, "Dynamic Fault Tree Analysis based on the Structure Function," *In Annual reliability and maintainability symposium*, Lake Buena Vista, pp. 462-467, 2011.

[146] C.P. Robert, and G. Casella, "Monte Carlo Statistical Methods," Springer, 2004.

[147] P. Zhu, J. Han, L. Liu, and F. Lombardi, "Reliability Evaluation of Phased-mission Systems using Stochastic Computation," Submitted to *IEEE Transactions on Reliability* 2015.

[148] L. Xing, and S.V. Amari, "Reliability of Phased-Mission Systems," K. B. Misra, editor. *Handbook of performability engineering. Springer* 2008, pp. 349–68 Chapter 23.

[149] A.K. Somani, J.A. Ritcey, and S.H.L. Au, "Computationally Efficient Phased-mission Reliability Analysis for Systems with Variable Configurations," *IEEE Transactions on Reliability*, vol. 41, no. 4, pp. 504–11, 1992.

[150] L. Xing, "Reliability Evaluation of Phased-mission Systems with Imperfect Fault Coverage and Common-cause Failures," *IEEE Transactions on Reliability*, vol. 56, no. 1, pp. 58–68, 2007.

[151] A. Pedar, and V.V.S. Sarma, "Phased-mission Analysis for Evaluating the Effectiveness of Aerospace Computing Systems," *IEEE Transactions on Reliability*, vol. R-30, no. 5, pp. 429–37, 1981.

[152] H.S. Winokur J.R., and L.J. Goldstein, "Analysis of Mission-oriented Systems," *IEEE Transactions on Reliability*, vol. R-18, no. 4, pp. 144–8, 1969.

[153] J.L. Bricker, "A Unified Method for Analyzing Mission Reliability for Fault Tolerant Computer Systems," *IEEE Transactions on Reliability*, vol. R-22, no. 2, pp. 72–7, 1973.

[154] X. Zang, H. Sun, and K.S. Trivedi, "A BDD-based Algorithm for Reliability Analysis of Phased-Mission Systems," *IEEE Transactions on Reliability*, vol. 48, no. 1, pp. 50–60, 1999.

[155] R.E. Altschul, and P.M. Nagel, "The Efficient Simulation of Phased Fault Trees," *In Proceedings of the Annual Reliability and Maintainability Symposium*, pp. 292–296, January, 1987.

[156] F.A. Tillman, C.H. Lie, and C.L. Hwang, "Simulation Model of Mission Effectiveness for Military Systems," *IEEE Transactions on Reliability*, vol. R-27, no. 3, pp. 191–4, 1978.

[157] L. Xing, and L. Gregory, "BDD-based Reliability Evaluation of Phased-mission Systems with Internal/External Common-cause Failures," *Reliability Engineering & System Safety,* vol. 112, pp. 145-153, 2013.

[158] A. Bondavalli, S. Chiaradonna, FD. Giandomenico, and I. Mura, "Dependability Modeling and Evaluation of Multiple-phased Systems using DEEM," *IEEE Transactions on Reliability*, vol. 53, no. 4, pp. 509–22, 2004.

[159] J.B. Dugan, "Automated Analysis of Phased-mission Reliability," *IEEE Transactions on Reliability*, vol. 40, no. 1, pp. 45–52, 55, 1991.

[160] I. Mura, and A. Bondavalli, "Markov Regenerative Stochastic Petri Nets to Model and Evaluate Phased-mission Systems Dependability," *IEEE Transactions on Computers*, vol. 50, no. 12, pp. 1337–51, 2001.

[161] M.K. Smotherman, and K. Zemoudeh, "A Non-homogeneous Markov Model for Phased-mission Reliability Analysis," *IEEE Transactions on Reliability*, vol. 38, no. 5, pp. 585–90, 1989.

[162] L. Meshkat, L. Xing, S. Donohue, and Y. Ou, "An Overview of the Phase-modular Fault Tree Approach to Phased-mission System Analysis," *In Proceedings of the International Conference on Space Mission Challenges for Information Technology*, pp. 393–398, 2003.

[163] Y. Ou, and J.B. Dugan, "Modular Solution of Dynamic Multi-phase Systems," *IEEE Transactions on Reliability*, vol. 53, no. 4, pp. 499–508, 2004.

[164] L. Xing, and J.B. Dugan, "Analysis of Generalized Phased-mission System Reliability Performance, and Sensitivity," *IEEE Transactions on Reliability*, vol. 51, no. 2, pp. 199-211, 2002.

[165] J.S. Wu, and R.J. Chen, "An Algorithm for Computing the Reliability of a Weighted-*k*-out-of-*n* System," *IEEE Transactions on Reliability*, vol. R-43, no. 2, pp. 327–8, 1994.

[166] J.M. Lu, and X.Y. Wu, "Reliability Evaluation of Generalized Phased-mission Systems with Repairable Components," *Reliability Engineering & System Safety*, vol. 121, pp. 136-145, 2014.

[167] W. Li, and M.J. Zuo, "Reliability Evaluation of Multi-state Weighted *k*-out-of-*n* Systems," *Reliability Engineering & System Safety*, vol. 93, no. 1, pp. 160-167, 2008.

[168] Z. Tang, and J.B. Dugan, "BDD-based Reliability Analysis of Phased-mission Systems with Multimode Failures," *IEEE Transactions on Reliability*, vol. 55, no. 2, pp. 350–360, 2006.

[169] P. Jeavons, D.A. Cohen, and J. Shawe-Taylor, "Generating Binary Sequences for Stochastic Computing," *IEEE Transactions on Information Theory*, vol. 40, no. 3, pp. 716–720, 1994.

[170] S.-S. Choi, S.-H. Cha, and C.C. Tappert, "A Survey of Binary Similarity and Distance Measures," *Journal of Systemics, Cybernetics and informatics*, vol. 8, pp. 43-48, 2010.

[171] T.H. Chen, and J.P. Hayes, "Analyzing and Controlling Accuracy in Stochastic Circuits," *Computer Design (ICCD), 32rd IEEE International Conference on*, pp. 367-373, 2014.

[172] H. Niederreiter, "Random Number Generation and Quasi-Monte Carlo Methods," SIAM, Philadelphia, 1992.