Formal Reliability Analysis Using Theorem Proving

Osman Hasan, *Student Member*, *IEEE*, Sofiène Tahar, *Senior Member*, *IEEE*, and Naeem Abbasi, *Student Member*, *IEEE*

Abstract—Reliability analysis has become a tool of fundamental importance to virtually all electrical and computer engineers because of the extensive usage of hardware systems in safety and mission critical domains, such as medicine, military, and transportation. Due to the strong relationship between reliability theory and probabilistic notions, computer simulation techniques have been traditionally used to perform reliability analysis. However, simulation provides less accurate results and cannot handle large-scale systems due to its enormous CPU time requirements. To ensure accurate and complete reliability analysis and thus more reliable hardware designs, we propose to conduct a formal reliability analysis of systems within the sound core of a higher order logic theorem prover (HOL). In this paper, we present the higher order logic formalization of some fundamental reliability theory concepts, which can be built upon to precisely analyze the reliability of various engineering systems. The proposed approach and formalization is then utilized to analyze the repairability conditions for a reconfigurable memory array in the presence of stuck-at and coupling faults.

Index Terms—Formal models, performance and reliability, theorem proving, memory structures.

1 Introduction

Reliability analysis involves the application of various mathematical techniques for the prediction of reliability-related parameters, such as a system's resistance to failure and its ability to perform a required function under some given conditions. Reliability analysis relies heavily on the concepts of probability and statistics due to the enormous amount of random or unpredictable components associated with the reliability parameters of a system. Examples include failures due to fabrication faults and electromigration phenomena in System-on-Chips (SoCs). Moreover, these systems act upon and within complex environments that themselves have certain elements of unpredictability, such as corrosion, vibration, and temperature variations. Due to these random components, establishing the reliability of a system under all circumstances usually becomes impractically expensive. The engineering approach to analyze the reliability of a system with these kinds of unavoidable elements of randomness and uncertainty is to use probabilistic analysis. The main idea is to mathematically model the random and unpredictable elements of the given system and its environment by appropriate random variables. The probabilistic and statistical properties of these random variables are then used to judge the system's reliability regarding parameters of interest.

Today, simulation is the most commonly used computer-based reliability analysis technique. Most simulation softwares provide a programming environment for defining functions that approximate random variables for probability distributions. The random elements in a given system are modeled by these functions, and the system is analyzed using computer simulation techniques [1], such as the Monte Carlo Method [2], where the main idea is to approximately answer a query on a probability distribution by analyzing a large number of samples. Statistical quantities, such as average and variance, may then be calculated, based on the data collected during the sampling process, using their mathematical relations in a computer. Due to the inherent nature of simulation, the reliability analysis results attained by this technique can never be termed as 100 percent accurate. Moreover, simulation requires an enormous amount of CPU time for attaining meaningful estimates. We generally need to acquire hundreds of thousands of samples to estimate the desired probabilistic quantities and this fact makes the simulation approach impractical when each sample acquisition step involves extensive computations.

The accuracy of hardware system reliability analysis results has become imperative these days because of the extensive usage of these systems in safety-critical areas. One of the unfortunate incidents, related to the inaccurate reliability analysis of systems, is the loss of the Mars Polar Lander [3] in December 1999. The Mars Polar Lander; a $165 million NASA spacecraft launched to survey Martian conditions, is believed to be lost mainly because of its engine shutdown while it was still 40 meters above the Mars surface. The engine shutdown happened due to the vibrations caused by the deployment of the lander's legs, i.e., a random behavior that gave false indication that the spacecraft had landed. In order to avoid incidents like this, simulation should not be relied upon for the reliability analysis of systems that are supposed to be used in safety-critical domains.

Formal methods [4] are capable of conducting precise system analysis and thus overcome the above-mentioned limitations of simulation. The main principle behind formal

• *The authors are with the Department of Electrical and Computer Engineering, Concordia University, 1455 De Maisonneuve Blvd. West, EV005.139, Montreal, QC H3G 1M8, Canada.*
E-mail: {o_hasan, tahar, n_ab}@ece.concordia.ca.

Manuscript received 11 Nov. 2008; revised 4 Apr. 2009; accepted 2 Sept. 2009; published online 23 Oct. 2009.
Recommended for acceptance by C. Bolchini and D. Sciuto.
For information on obtaining reprints of this article, please send e-mail to: tc@computer.org, and reference IEEECS Log Number TCSI-2008-11-0569.
Digital Object Identifier no. 10.1109/TC.2009.165.

analysis of a system is to construct a computer-based mathematical model of the given system and formally verify, within a computer, that this model meets rigorous specifications of intended behavior. Two of the most commonly used formal verification methods are model checking [5] and higher order logic theorem proving [6]. Model checking is an automatic verification approach for systems that can be expressed as a finite-state machine. higher order logic theorem proving, on the other hand, is an interactive approach but is more flexible in terms of tackling a variety of systems.

Both model checking and theorem proving have been successfully used for the precise functional correctness of a broad range of hardware systems. On the other hand, due to the strong relationship between reliability theory and randomness, their usage for reliability analysis has been somewhat limited. The major limitations being the restricted system expressibility and the inability to precisely reason about statistical properties, such as variance and tail distribution bounds, in the case of model checking and the fear of huge proof efforts involved in reasoning about random components of a system in the case of theorem proving.

We believe that the recent developments in the formalization of probability theory concepts in higher order logic [7], [8] can be extended upon to conduct reliability analysis in a higher order logic theorem prover. The main objective of this paper is to minimize the interactive proof efforts for conducting reliability analysis in a theorem prover by presenting the higher order logic formalization of some core reliability theory concepts. More specifically, we present a formal definition of reliability, which can be used to formally express the reliability characteristic of a system or component in higher order logic, and the verification of Markov and Chebyshev's inequalities [9], which play a vital role in the formal estimation of failure probabilities in the reliability analysis of systems. These results can be built upon to reason about the reliability characteristics of a system in a higher order logic theorem prover and thus tend to minimize the associated modeling and verification efforts.

The utilization and effectiveness of the proposed approach for handling real-world reliability analysis problems, is demonstrated through the reliability analysis of reconfigurable memory arrays in the presence of stuck-at and coupling faults [10], which are two of the most commonly found faults in memory arrays. Stuck-at faults occur when a memory cell never changes its state, i.e., it is always stuck in one state. Whereas, a coupling fault occurs when a write operation in one cell changes the contents of another cell in the memory array. In order to ensure reliable operation of memory arrays, some redundancy is usually added to memory arrays during the design phase. This way even after fabrication, we can repair the memory faults by replacing the rows or columns of the memory array containing faulty memory cells with the available spare rows or columns. Memories fabricated with these spare rows and columns are usually termed as *reconfigurable memory arrays*. This technique poses an interesting solution to the memory faults problem but comes with a bigger design challenge of estimating the right number of spare rows and columns for meeting reliability specifications. If a combination of spare rows and columns exists such that all faults from the memory array can be eliminated then such a combination of spare rows and columns is called a *repair solution*, and the array is called *repairable*. The repairability problem of a reconfigurable memory array is similar to the vertex cover problem of the bipartite graph and is known to be an NP complete problem [11]. Thus, probabilistic techniques are usually utilized to obtain reasonable solutions. In this paper, the proposed reliability analysis approach is used for attaining precise solutions to the above-mentioned repairability problem.

The state-of-the-art reliability analysis approach for the repairability problem of reconfigurable memory arrays is simulation, which usually fails to produce precise results due to its inherent limitations and the large capacities of memory arrays. Since reconfigurable memory arrays are an integral component of essentially all SoC designs these days and hence are quite frequently used in safety-critical areas, the inaccuracies and inadequacies of simulation in this domain may even result in the loss of human lives. Therefore, the precise solutions obtained for the repairability problem, in this paper, not only indicate the practical usefulness of our approach but also address the above-mentioned safety problem.

The work described in this paper is done using the HOL theorem prover [12], which is an interactive higher order logic theorem prover. The HOL core consists of only five basic axioms and eight primitive inference rules, which are implemented as ML functions. Soundness is assured as every new theorem must be verified by applying these basic axioms and primitive inference rules or any other previously verified theorems/inference rules. The main motivation behind choosing HOL for our work is the availability of most of the mathematical theories like Booleans, natural and real numbers, sets, measure, and probability.

The rest of the paper is organized as follows: Section 2 provides some related work. Section 3 describes the proposed theorem-proving-based reliability analysis approach. The formalization of reliability theory fundamentals is given in Section 4. This is followed by the reliability analysis of reconfigurable memory arrays in Section 5. Finally, Section 6 concludes the paper.

2 Related Work

The recently emerged CMOS or non-CMOS nanotechnologies, which are used to develop most of the state-of-the-art electronics and computer-related equipment, are more prone to defects than their predecessors. Therefore, reliability analysis of nanoscale devices has become not only essential but, due to their large gate counts, very challenging as well. Many researchers around the world are trying to improve the quality of computer-based reliability methods. The ultimate goal is to come up with a reliability analysis framework that includes robust and accurate analysis methods, has the ability to perform analysis for large-scale designs, and is easy to use. Some of the existing approaches that allow us to tackle such reliability problems are presented in the following: For instance, the probabilistic transfer matrices (PTM) approach [13] fundamentally relies on matrix arithmetic operations for each gate entity to assess

the reliability of the whole circuit. It involves the complete enumeration of all possible input and output combinations, which can be very expensive in terms of computation when dealing with large designs. A similar but independently proposed technique is based on developing probabilistic models for unreliable logic gates [14], called probabilistic gate models (PGMs), and utilizing these individual models to analyze the reliability of the circuit. In order to somewhat overcome the runtime and scalability issues of the above-mentioned approaches, recently three algorithms, based on independent gate failure assumption, have been proposed in [15], and their effectiveness is illustrated by successfully analyzing the reliability of a few benchmark circuits. All of the above techniques are based on simulation and thus cannot provide 100 percent precise results due to the inherent nature of simulation as has been described in the previous section.

Formal methods are capable of addressing the inaccuracy issues of simulation and thus have also been explored to conduct reliability analysis. Probabilistic model checking has been applied to analyze the circuit reliability at the logic gate and block levels [16], [17]. Like traditional model checking, probabilistic model checking involves the construction of a precise state-based mathematical model of the given random system, which is then subjected to exhaustive analysis to verify if it satisfies a set of reliability properties formally expressed in temporal logic. Besides the accuracy of the results, other promising features of probabilistic model checking include the ability to perform the analysis automatically. On the other hand, probabilistic model checking is limited to systems that can only be expressed as probabilistic finite-state machines or Markov chains. Another major limitation of probabilistic model checking is state space explosion [18] due to which it is not scalable to large designs. Similarly, to the best of our knowledge, it has not been possible to precisely reason about most of the statistical quantities, such as variance and tail distribution bounds, using probabilistic model checking so far.

The higher order logic theorem-proving-based reliability analysis approach, utilized in this paper, tends to overcome the limitations of both the simulation and model checking. Due to the formal nature of the models and properties and the inherent soundness of the theorem-proving approach, reliability analysis carried out in this way is free from any approximation and precision issues. Similarly, the high expressibility of higher order logic allows us to analyze a wider range of systems without any modeling limitations, such as infinite state-space or the limitedness to Markovian chain models.

Hurd [7] developed a framework for the verification of probabilistic algorithms in the HOL theorem prover. Random variables are basically probabilistic algorithms and thus can be formalized and verified, based on their probability distribution properties, using the methodology proposed in [7]. In fact, building upon Hurd's formalization, most of the commonly used discrete [7] and continuous [8] random variables have been formalized in higher order logic and their corresponding probabilistic and statistical [8] properties have been verified using interactive theorem-proving techniques. In this paper, we utilize the above-mentioned formalization, available in the HOL theorem prover, to develop a generic theorem-proving-based reliability analysis approach, a novelty that to the best of our knowledge has not been presented in the open literature so far.

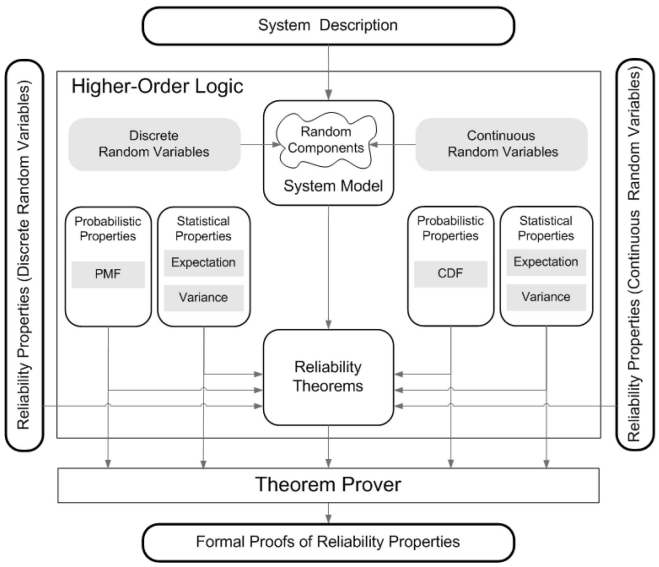

Fig. 1. Theorem-proving-based reliability analysis.

We utilize the proposed approach for the reliability analysis of memory arrays. Simulation techniques are very commonly used for such analysis [19], [20]. When memory sizes become large, analysis through simulation very quickly becomes computationally difficult to handle. Paper-and-pencil-based analytical analysis has been traditionally used for such cases. A memory array probability model represents either the occurrence of individual faults or the total number of faults as a random variable and thus allows reasoning about statistical properties of memory arrays. Questions, such as "given a certain fault distribution and number of faults can almost every reconfigurable memory array be repaired," or "with how many faults a memory array can almost never be repaired," are then answered [21], [22], [23] based on analytical reasonings. To the best of our knowledge, we were the first ones to utilize higher order logic theorem proving for tackling the repairability problem of stuck-at faults for reconfigurable memory arrays in [24]. That analysis has been extended in the current paper with the inclusion of coupling fault models and several new probabilistic and statistical properties and reliability conditions.

3 Proposed Methodology

A hypothetical model of the proposed reliability analysis approach is given in Fig. 1, with some of its most fundamental components depicted with shaded boxes. Like all traditional analysis problems, the starting point of reliability analysis is also a system description and some intended system properties and the goal is to check if the given system satisfies these given properties. For simplicity, we have divided system reliability properties into two categories, i.e., reliability properties related to discrete random variables and reliability properties related to continuous random variables.

The first step in the proposed approach is to construct a model of the given system in higher order logic. For this purpose, the foremost requirement is the availability of infrastructures that allow us to formalize all kinds of discrete and continuous random variables as higher order logic functions, which in turn can be used to represent the random components of the given system in its higher order logic model. The second step is to utilize the formal model of the system to express system reliability characteristics as higher order logic theorems. The prerequisite for this step is the ability to express probabilistic and statistical properties related to both discrete and continuous random variables in higher order logic. All probabilistic properties of discrete and continuous random variables can be expressed in terms of their *Probability Mass Functions* (PMFs) and *Cumulative Distribution Functions* (CDFs), respectively. Similarly, most of the commonly used statistical properties can be expressed in terms of the expectation and variance characteristics of the corresponding random variable. Thus, we require the formalization of mathematical definitions of PMF, CDF, expectation, and variance for both discrete and continuous random variables in order to be able to express the given system's reliability characteristics as higher order logic theorems. The third and the final step for conducting reliability analysis in a theorem prover is to formally verify the higher order logic theorems developed in the previous step using a theorem prover. For this verification, it would be quite handy to have access to a library of some preverified theorems corresponding to some commonly used properties regarding probability distribution functions, expectation, and variance. Hence, we can build upon such a library of theorems and thus speed up the verification process. The formalization details regarding the above-mentioned steps are briefly described now.

3.1 Discrete Random Variables

A random variable is called discrete if its range, i.e., the set of values that it can attain, is finite or at most countably infinite [25]. Discrete random variables can be completely characterized by their PMFs that return the probability that a random variable X is equal to some value x, i.e., $Pr(X = x)$. Discrete random variables are quite frequently used to model randomness in reliability analysis. For example, the Bernoulli random variable is widely used to model the fault occurrence in a component and the Binomial random variable may be used to represent the number of faulty components in a lot.

Discrete random variables can be formalized in higher order logic as deterministic functions with access to an infinite Boolean sequence $\mathbb{B}^\infty$; an infinite source of random bits with data type $(natural \rightarrow bool)$ [7]. These deterministic functions make random choices based on the result of popping the top most bit in the infinite Boolean sequence and may pop as many random bits as they need for their computation. When the functions terminate, they return the result along with the remaining portion of the infinite Boolean sequence to be used by other functions. Thus, a random variable that takes a parameter of type α and ranges over values of type β can be represented in HOL by the function

$$\mathcal{F} : \alpha \rightarrow B^\infty \rightarrow \beta \times B^\infty.$$

For example, a $Bernoulli(\frac{1}{2})$ random variable that returns 1 or 0 with probability $\frac{1}{2}$ can be modeled as

$$\vdash \texttt{bit} = \lambda \texttt{s}.\ (\texttt{if shd s then 1 else 0},\ \texttt{stl s}),$$

where the variable s, in the above definition, represents the infinite Boolean sequence and the functions `shd` and `stl` are the sequence equivalents of the list operations *"head"* and *"tail"*. A function of the form λx.t represent a lambda abstraction function in HOL that maps x to $t(x)$. The function `bit` accepts the infinite Boolean sequence and returns a pair with the first element equal to either 0 or 1 and the second element equal to the unused portion of the infinite Boolean sequence.

The random variables can also be expressed in the more general state-transforming monad where the states are the infinite Boolean sequences.

$$\vdash \forall\ \texttt{a s.unit a s} = (\texttt{a}, \texttt{s})$$
$$\vdash \forall\ \texttt{f g s.bind f g s} = \texttt{g fst(f s)snd(f s)},$$

where the HOL functions `fst` and `snd` return the first and second component of a pair, respectively. The `unit` operator is used to lift values to the monad, and the `bind` is the monadic analogue of function application. All monad laws hold for this definition, and the notation allows us to write functions without explicitly mentioning the sequence that is passed around, e.g., function bit can be defined as

$$\vdash \texttt{bit\_monad} = \texttt{bind sdest}$$
$$(\lambda \texttt{b. if b then unit 1 else unit 0}),$$

where `sdest s` returns the pair (*shd* `s,` *stl* `s`).

The work in [7] also presents the formalization of some mathematical measure theory in HOL, which can be used to define a probability function $\mathbb{P}$ from sets of infinite Boolean sequences to $real$ numbers between 0 and 1. The domain of $\mathbb{P}$ is the set $\mathcal{E}$ of events of the probability. Both $\mathbb{P}$ and $\mathcal{E}$ are defined using the Carathéodory's Extension theorem, which ensures that $\mathcal{E}$ is a σ-algebra: closed under complements and countable unions. The formalized $\mathbb{P}$ and $\mathcal{E}$ can be used to derive all the basic axioms of probability in the HOL theorem prover. Similarly, they can also be used to prove probabilistic properties for random variables. For example, we can formally verify the following probabilistic property for the function `bit`, defined above:

$$\vdash \mathbb{P}\{\texttt{s} \mid \texttt{fst}\ (\texttt{bit s}) = 1\} = \frac{1}{2},$$

where $\{x|C(x)\}$ represents a set of all elements x that satisfy the condition C in HOL.

The above-mentioned infrastructure can be utilized to formalize most of the commonly used discrete random variables and verify their corresponding PMF relations [7]. For example, the Binomial random variable, which models an experiment that counts the number of successes in m independent Bernoulli(p) trials with a success probability

equal to p, can be formalized in higher order logic as the following recursive function:

Definition 1. *Binomial(m,p) Random Variable*

$$\begin{aligned}\vdash\ &\forall\, \mathtt{p}.\ (\mathtt{prob\_bino\ 0\ p} = \mathtt{unit\ 0}) \wedge \\ &\forall\, \mathtt{p\ n}.\ \mathtt{prob\_bino}(\mathtt{n}+1)\ \mathtt{p} = \\ &\mathtt{bind}\ (\mathtt{prob\_bino\ n\ p}) \\ &(\lambda \mathtt{m}.\ \mathtt{bind}(\mathtt{prob\_bern\ p}) \\ &(\lambda \mathtt{b}.\ \mathtt{unit}\ (\mathtt{if\ b\ then}\ (\mathtt{m}+1)\ \mathtt{else\ m})))),\end{aligned}$$

where `prob_bern` *is the higher order logic function for the Bernoulli(p) random variable [7] and the* `bind` *and* `unit` *functions have already been defined above.*

We were also able to verify the correctness of the above definition by proving its PMF relation in HOL as follows:

Theorem 1. *PMF of Binomial(m,p) Random Variable*

$$\begin{aligned}\vdash\ &\forall\, \mathtt{m\ p\ n}.\ 0 \le \mathtt{p} \wedge \mathtt{p} \le 1 \\ &\Rightarrow \mathbb{P}\{\mathtt{s} \mid \mathtt{fst}\ (\mathtt{prob\_bino\ m\ p\ s}) = \mathtt{n}\} \\ &= \binom{\mathtt{m}}{\mathtt{n}} \mathtt{p}^{\mathtt{n}} (1-\mathtt{p})^{\mathtt{m}-\mathtt{n}}.\end{aligned}$$

3.2 Continuous Random Variables

A random variable is called continuous if it ranges over a continuous set of numbers [25]. A continuous set of numbers, sometimes referred to as an interval, contains all real numbers between two limits. Continuous random variables can be completely characterized by their CDFs that return the probability that a random variable X is exactly less than or equal to some value x, i.e., $Pr(X \le x)$. Continuous random variables are required to model various phenomenon in reliability analysis. For example, the exponential random variable is quite frequently used to model the time to failure of a component.

The sampling algorithms for continuous random variables are nonterminating and hence require a different formalization approach than discrete random variables, for which the sampling algorithms are either guaranteed to terminate or satisfy probabilistic termination, meaning that the probability that the algorithm terminates is 1. One approach to address this issue is to utilize the concept of the nonuniform random number generation [1], which is the process of obtaining arbitrary continuous random numbers using a Standard Uniform random number generator. The main advantage of this approach is that we only need to formalize one continuous random variable from scratch, i.e., the Standard Uniform random variable, which can be used to model other continuous random variables by formalizing the corresponding nonuniform random number generation method.

Based on the above approach, a methodology for the formalization of all continuous random variables for which the inverse of the CDF can be represented in a closed mathematical form is presented in [8]. The first step in this methodology is the formalization of the Standard Uniform random variable. The Standard Uniform random variable can be formalized using Hurd's approach for the formalization of discrete random variables, described in the last section, and the formalization of the mathematical concept of limit of a *real* sequence [27] as the following sampling algorithm:

$$\lim_{n\to\infty}\left(\lambda n.\ \sum_{k=0}^{n-1}\left(\frac{1}{2}\right)^{k+1} X_k\right), \tag{1}$$

where X_k denotes the outcome of the kth random bit; $True$ or $False$ represented as 1 or 0, respectively. The formalization details are outlined in [8].

The second step in the methodology for the formalization of continuous probability distributions is the formalization of the CDF and the verification of its classical properties. This is followed by the formal specification of the mathematical concept of the inverse function of a CDF. This formal specification, along with the formalization of the Standard Uniform random variable and the CDF properties, can be used to formally verify the correctness of the Inverse Transform Method (ITM) [1]. The ITM is a well-known nonuniform random generation technique for generating nonuniform random variables for continuous probability distributions for which the inverse of the CDF can be represented in a closed mathematical form. Mathematically, it can be expressed for a random variable X with CDF F using the Standard Uniform random variable U as follows:

$$Pr(F^{-1}(U) \le x) = F(x), \tag{2}$$

and its formal proof can be found in [8].

The formalized Standard Uniform random variable can now be used to formally specify any continuous random variable for which the inverse of the CDF can be expressed in a closed mathematical form as $X = F^{-1}(U)$. Whereas, the CDF of this formally specified continuous random variable, X, can be verified, based on simple arithmetic reasoning, using the formal proof of the ITM. For illustration purposes, consider the example of the exponential random variable, with the following CDF:

$$Pr(X \le x) = \begin{cases} 0, & \text{for } x \le 0, \quad (3a) \\ 1 - e^{-mx}, & \text{for } 0 < x. \quad (3b) \end{cases}$$

It can be expressed, using the above methodology, as the following higher order logic function.

Definition 2. *Exponential(m) Random Variable*

$$\begin{aligned}\vdash\ &\forall\, \mathtt{m\ s}.\ \mathtt{exp\_rv\ m\ s} = \\ &\left(\lambda \mathtt{x}.\ -\frac{1}{\mathtt{m}}\ \mathtt{ln}\ (1-\mathtt{x})\right)(\mathtt{std\_unif\_cont\ s}),\end{aligned}$$

where the HOL functions $(\lambda \mathtt{x}.\ -\frac{1}{\mathtt{m}}\ \mathtt{ln}\ (1-\mathtt{x}))$ *and* `std_unif_cont` *represent the inverse CDF of the exponential random variable and the Standard Uniform random variable, respectively.*

Now, the CDF of the exponential random variable, given in (3), can be expressed as the following theorem:

Theorem 2. *CDF for the Exponential Random Variable*

$$\begin{aligned}\vdash\ &\forall\, \mathtt{m\ x}.\ (0 < \mathtt{m}) \Rightarrow \\ &\mathtt{cdf}\ (\lambda \mathtt{s}.\ \mathtt{exp\_rv\ m\ s})\ \mathtt{x} = \\ &\mathtt{if\ x} \le 0\ \mathtt{then}\ 0\ \mathtt{else}\ (1 - \mathtt{e}^{-\mathtt{mx}}).\end{aligned}$$

The verification of Theorem 2 is based on the above methodology and requires very little user interaction, since it is based on the formally verified ITM and thus requires arithmetic reasoning only instead of the relatively complex reasoning based on probability theory principles.

3.3 Statistical Properties

In reliability analysis, statistical characteristics play a major role in decision making as they tend to summarize the probability distribution characteristics of a random variable in a single number. Due to their widespread interest, the computation of statistical characteristics has now become one of the core components of every modern reliability analysis framework.

Expectation provides the average of a random variable, where each of the possible outcomes of this random variable is weighted according to its probability [9]. The expectation for a function of a discrete random variable, which attains values in the positive integers only, is defined as follows [26]:

$$Ex\_fn[f(X)] = \sum_{n=0}^{\infty} f(n) Pr(X = n), \tag{4}$$

where X is the discrete random variable and f represents a function of the random variable X. The above definition only holds if the associated summation is convergent, i.e., $\sum_{n=0}^{\infty} f(n) Pr(X = n) < \infty$. The expression of expectation, given in (4), has been formalized in [8] as a higher order logic function using the formalization of the probability function $\mathbb{P}$, explained in Section 3.1 of this paper, as follows:

Definition 3. *Expectation of Function of a Discrete RV*

$$\vdash \forall \, \mathtt{f\ R}.\ \mathtt{expec\_fn\ f\ X} = \mathtt{suminf}(\lambda \mathtt{n}.\ (\mathtt{f\ n})\mathbb{P}\{\mathtt{s} \mid \mathtt{fst(X\ s)} = \mathtt{n}\}),$$

where the HOL function `suminf` *[27] represents the infinite summation of a real sequence* $\lim_{k\to\infty} \sum_{n=0}^{k} f(n)$.

The expected value of a discrete random variable can now be defined as a special case of Definition 3 when f is an identity function.

Definition 4. *Expectation of a Discrete RV*

$$\vdash \forall \, \mathtt{R}.\ \mathtt{expec\ R} = \mathtt{expec\_fn}\ (\lambda \mathtt{n}.\ \mathtt{n})\ \mathtt{X}.$$

In order to verify the correctness of the above definitions of expectation, they are utilized in [8] to formally verify the following classical expectation properties:

$$Ex\left[\sum_{i=1}^{n} X_i\right] = \sum_{i=1}^{n} Ex[X_i], \tag{5}$$

$$Ex[a + bX] = a + bEx[X]. \tag{6}$$

These properties not only verify the correctness of the above definitions but also play a vital role in verifying the expectation characteristics of discrete random components of probabilistic systems, as will be seen in Section 5 of this paper.

Variance of a random variable X describes the difference between X and its expected value and thus is a measure of its dispersion. It is defined for a discrete random variable, X, as follows:

$$Var[X] = Ex[(X - Ex[X])^2]. \tag{7}$$

The above definition of variance can be formalized in higher order logic by utilizing the formal definitions of expectation, given in Definitions 3 and 4 as follows:

Definition 5. *Variance of a Discrete Random Variable*

$$\vdash \forall \, \mathtt{X}.\ \mathtt{variance\ X} = \mathtt{expec\_fn}\ (\lambda \mathtt{n}.\ (\mathtt{n} - \mathtt{expec\ X})^2)\ \mathtt{X}.$$

Like the expectation definition, this definition was also formally verified to be correct by proving the following classical variance properties for it:

$$Var[X] = Ex[X^2] - (Ex[X])^2, \tag{8}$$

$$Var\left[\sum_{i=1}^{n} X_i\right] = \sum_{i=1}^{n} Var[X_i]. \tag{9}$$

The above-mentioned formalization allows us to reason about expectation and variance properties of any formalized discrete random variable that attains values in positive integers. For example, again consider the example of the Binomial(m, p) random variable, given in Definition 1. Its expectation (mp) and variance ($mp(1-p)$) expressions can be expressed as the following higher order logic theorems and verified using the above-mentioned infrastructure:

Theorem 3. *Expectation of Binomial(m,p) Random Variable*

$$\vdash \forall \, \mathtt{m\ p}.\ 0 \leq \mathtt{p} \wedge \mathtt{p} \leq 1 \Rightarrow \mathtt{expec}\ (\lambda \mathtt{s}.\ \mathtt{prob\_bino\ m\ p\ s}) = \mathtt{m\ p}.$$

Theorem 4. *Variance of Binomial(m,p) Random Variable*

$$\vdash \forall \, \mathtt{m\ p}.\ 0 \leq \mathtt{p} \wedge \mathtt{p} \leq 1 \Rightarrow \mathtt{variance}\ (\lambda \mathtt{s}.\ \mathtt{prob\_bino\ m\ p\ s}) = \mathtt{m\ p}\ (1 - \mathtt{p}).$$

The formalization and verification, presented in this section, can be utilized to formally reason about expectation and variance for positive integer valued discrete random variables. On the other hand, to the best of our knowledge, the formalization and verification of statistical properties, like expectation and variance, for continuous random variables is an open research issue as of now. This step requires a higher order logic formalization of an integration function that can also handle functions with domains other than real numbers. Lebesgue integration provides this feature and thus the higher order logic formalization of some portions of the Lebesgue integration theory [28] can be built upon for formalizing the mathematical concepts of expectation and variance for continuous random variables.

4 Reliability Theory Formalization

In this section, we present the higher order logic formalization of some fundamental and widely used reliability theory definitions and theorems. The main motivation behind this is to provide the users of the higher order logic theorem-proving-based reliability analysis approach with a generic infrastructure that can be built upon and thus minimize their interactive proof efforts associated with the third step of the proposed reliability analysis approach, described in the previous section.

In engineering reliability theory, reliability $R(t)$ of a system or component is defined as the probability that it performs its intended function until some time t [29].

$$R(t) = Pr(X > t). \tag{10}$$

The random variable X, in the above definition, models the time to failure of the system. Usually, this time to failure is modeled by the exponential random variable with parameter m that represents the failure rate of the system. The value of m depends on the drift of system's characteristics with time. This drift may occur because of several internal or external factors. For example, in the case of analyzing an SoC or some other hardware component the value of m would be effected due to electromigration, environmental effects, like corrosion, vibration, and temperature, and transient stresses such as electrostatic discharge and lightning.

Now, based on (10), we can formally define the reliability of a system or component in HOL using the formal definition of the exponential random variable, given in Definition 2, as follows:

Definition 6. *System Reliability*

$$\vdash \forall\, \mathtt{m\ t.rel\_sys\ m\ t} = \mathbb{P}\{\ \mathtt{s} |\ \mathtt{exp\_rv\ m\ s} > \mathtt{t}\}.$$

The function `rel_sys` accepts two parameters of type *real*, m, and t, which represent the failure rate and time, respectively. It returns the reliability of the given system at any time t based on the mathematical expression of (10).

We now formally verify the following useful alternate reliability expression [29]:

$$R(t) = e^{-mt}. \tag{11}$$

It can be formally expressed as follows:

Theorem 5. *Alternate System Reliability Expression*

$$\vdash \forall\, \mathtt{m\ t}.\ (\mathtt{0} \leq \mathtt{t}) \Rightarrow \mathtt{rel\_sys\ m\ t} = \mathtt{e}^{-\mathtt{mt}},$$

where the assumption $(\mathtt{0} \leq \mathtt{t})$ *ensures that time t can never be negative.*

Theorem 5 was verified using the complement law of probability ($\forall \mathtt{A}.\ \mathbb{P}(\bar{\mathtt{A}}) = 1 - \mathbb{P}(\mathtt{A})$) and the CDF theorem for the exponential random variable.

Equation (11) provides a very simple but useful means to determine the reliability of a component or system and is thus widely used in reliability analysis. For example, it has been used to assess the reliability of reconfigurable memory arrays in [30] and the reliability and fault tolerance of robotics in [31]. Thus, its verification in Theorem 5 allows us to tackle these kinds of reliability analysis within the sound core of a theorem prover.

In reliability analysis, while looking at the failure rates of a system, it is often the case that we are interested in the probability that a random variable assumes values that are far from its expectation. Instead of characterizing this probability by a distribution function, it is a common practice to rely upon bounds on this distribution, termed as tail distribution bounds, which are usually calculated using Markov and Chebyshev's inequalities [9].

The Markov's inequality gives an upper bound for the probability that a non-negative random variable X is greater than or equal to some positive constant

$$Pr(X \geq a) \leq \frac{Ex[X]}{a}. \tag{12}$$

Markov's inequality can be expressed in HOL, using the statistical-properties-related formalization presented in Section 3.3, for a measurable discrete random variable that attains values in positive integers only as follows:

Theorem 6. *Markov's Inequality*

$$\begin{aligned}\vdash \forall\, \mathtt{X\ a}.\ (\mathtt{0} < \mathtt{a}) \wedge \\ (\mathtt{summable}(\lambda \mathtt{n}.\ \mathtt{n}\ \mathbb{P}\{\mathtt{s} \mid \mathtt{fst}\ (\mathtt{X\ s}) = \mathtt{n}\})) \\ \Rightarrow \mathbb{P}\ \{\mathtt{s} \mid \mathtt{fst}\ (\mathtt{X\ s}) \geq \mathtt{a}\} \leq \frac{(\mathtt{expec\ X})}{\mathtt{a}},\end{aligned}$$

where a represents a real number and the predicate `summable` *[27] returns True if the infinite summation of its real sequence argument exists, i.e.,* $\exists x.\ \lim_{k\to\infty} \sum_{n=0}^{k} f(n) = x$.

Thus, the `summable` assumption in the above theorem states that the theorem is only valid for a random variable X with well-defined expectation. The HOL proof of Theorem 6 is based on some probability theory axioms and arithmetic reasoning and more details can be found in [8].

Markov's inequality gives the best tail bound possible, for a nonnegative random variable, using the expectation for the random variable only. This bound can be improved upon if more information about the distribution of the random variable is taken into account. Chebyshev's inequality is based on this principle and it presents a significantly stronger tail bound in terms of variance of the random variable

$$Pr(|X - Ex[X]| \geq a) \leq \frac{Var[X]}{a^2}. \tag{13}$$

The corresponding HOL theorem is as follows:

Theorem 7. *Chebyshev's Inequality*

$$\begin{aligned}\vdash \forall\, \mathtt{R\ a}.\ (\mathtt{0} < \mathtt{a}) \wedge (\mathtt{0} < \mathtt{variance\ X}) \wedge \\ (\mathtt{summable}\ (\lambda \mathtt{n}.\ \mathtt{n}\ \mathbb{P}\{\mathtt{s} \mid \mathtt{fst}\ (\mathtt{X\ s}) = \mathtt{n}\})) \wedge \\ (\mathtt{summable}\ (\lambda \mathtt{n}.\ \mathtt{n}^2\ \mathbb{P}\{\mathtt{s} \mid \mathtt{fst}\ (\mathtt{X\ s}) = \mathtt{n}\})) \\ \Rightarrow \mathbb{P}\ \{\mathtt{s} \mid \mathtt{abs}\ (\mathtt{fst}\ (\mathtt{X\ s}) - \mathtt{expec\ X}) \geq \mathtt{a}\} \\ \leq \frac{\mathtt{variance\ R}}{\mathtt{a}^2}.\end{aligned}$$

The *summable* assumptions ensure that the theorem is only valid for a random variable X with well-defined expectation and second moment values. The HOL proof of Theorem 7 is also based on some probability theory axioms and the proof details can be found in [8].

Due to the widespread interest in failure probabilities in reliability analysis, Markov and Chebyshev's inequalities are widely used in this domain. Thus, their formal verification is a significant step toward the development of a successful theorem-proving-based reliability analysis framework. In fact, we will utilize them for the repairability analysis presented in the next section.

5 Reconfigurable Memory Arrays

Embedded memory is the most dominant component in terms of silicon area of any SoC these days. Applications such as mobile communication devices, signal processing, and computer networks all require large amounts of memory. Extremely small memory cells and the fact that a significant amount of the chip area is taken by compact memories, makes them more prone to defects than standard logic. The defects in a memory can render the whole SoC useless. Even in mature fabrication processes where the defect densities tend to be small, throwing away of any chip is considered unacceptable because of its adverse effect on yield. Moreover, the fabrication defects that are not caught in the testing phase may also lead to devastating situations when the corresponding memories are used in safety-critical SoCs for domains, such as medicine, military, and transportation.

Reconfigurable memory arrays tend to increase the reliability of memory arrays in the presence of fabrication faults. The main idea is to add some redundancy to memory arrays during the design phase. This way even after fabrication, we can repair some of the memory faults by replacing the rows or columns containing faulty memory cells with the available spare rows or columns. Though, this solution comes with a bigger design challenge of solving the repairability problem, i.e., estimating the right number of spare rows and columns for meeting reliability specifications for a given memory array.

In this section, we analyze this repairability problem of reconfigurable memory arrays in the presence of stuck-at and coupling faults, which are two of the most commonly occurring fabrication faults, using the proposed reliability analysis approach. Our analysis is mainly inspired by the analytical model developed in [22] for the paper-and-pencil-based reliability analysis of reconfigurable memory arrays. We proceed by formally expressing a fault model for reconfigurable memory arrays in higher order logic. Our formalization utilizes precise random variable functions to express the random components in the model. This formalization is then utilized to formally verify two significant results regarding the repairability problem of reconfigurable memory arrays. First, we verify a relation that ascertains that a large square memory array is almost always repairable (with probability 1) if stuck-at and coupling faults are independent and identically distributed with specific probabilities. This condition is usually termed as the repairability condition. Second, we verify a bound on the stuck-at and coupling fault occurrence probabilities that will make reconfiguration of a large square memory array almost impossible (with probability 0). This condition, which is usually termed as the irrepairability condition, allows us to determine how large the probability of defects must be in order to make reconfiguration nearly impossible. Using the proposed approach, we are able to accurately analyze both of the above-mentioned repairability and irrepairability properties without any CPU time constraints, which clearly demonstrates its effectiveness for real-world reliability analysis problems.

5.1 Formal Stuck-at and Coupling Fault Model

In order to illustrate the formal stuck-at and coupling fault model, we first present a 6×6 memory array with one stuck-at and two coupling faults example, shown in Fig. 2a. The stuck-at fault is represented with a circle and the coupling faults are represented using a pair of squares. The two squares in the pair are connected by an arrow. The direction of the arrow is from the coupling cell (lightly shaded) to the coupled cell.

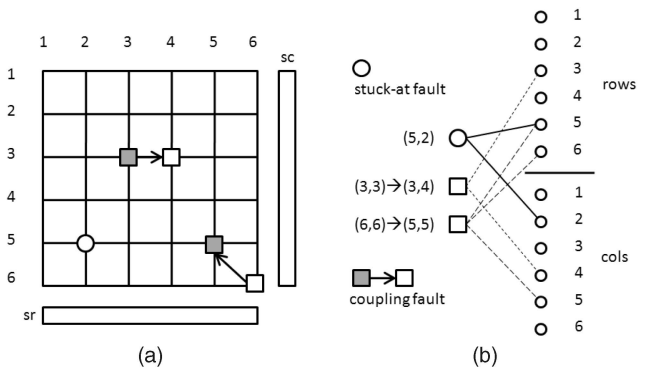

Fig. 2. Memory array model.

A coupling fault in the memory array can be repaired by replacing either the coupling cell or the coupled cell. Disabling the coupling or the aggressor cell must be done by replacing the row containing the coupling cell, whereas the coupled cell can be repaired by replacing either its row or column with a spare row or a spare column [22]. For example, the coupling fault $(6,6) \rightarrow (5,5)$ can be repaired by either replacing the sixth row with a spare row, or by replacing the fifth row or the fifth column with a spare row or a spare column, respectively (Fig. 2b). The degree of this fault node in the graph is three, since this fault node is connected to one column node (5) and two row nodes $(5,6)$. The second coupling fault, $(3,3) \rightarrow (3,4)$, on the other hand, has a degree of two since it is only connected to one row (3) and one column (4). A stuck-at fault, on the other hand, can be repaired by replacing either the row or column containing the fault by a spare row or a spare column, respectively [32]. Thus, the stuck-at fault at location $(5,2)$ can be repaired by either replacing the fifth row or second column of the array with a spare row or a spare column. In the graph model for the repairability problem, each fault node can have a degree of two or three for the coupling faults and a degree of two for the stuck-at faults.

We now generalize the above example. The reconfigurable memory array can be modeled as a bipartite graph (F, X, E). In this bipartite graph, F represents the set of nodes corresponding to faults in the memory array, $X = R \cup C$ is a set of nodes corresponding to rows (R) and columns (C) in the memory array, and E is the set of edges connecting various nodes of the sets F and X based on how these faults can possibly be repaired. It is important to note here that the number of elements in the set F and their identities is a random quantity as fault occurrence is an unpredictable event. Therefore, the probability that a node will be included in the set F depends on the probabilities p_s and p_c, corresponding to the occurrence of stuck-at and coupling faults, respectively. Also, the occurrence of stuck-at or coupling fault, and thus the inclusion of a node in the set F, is assumed to be independent and identically distributed in this model. Thus, the upper bound on the cardinality of the set F is $n^3 + n^2$ [22], where n represents the number of total

rows or columns of an $n \times n$ memory array. A repair solution exists if one can find a set of nodes say S in set X, which are less than or equal to in number of the available spare rows (sr) or columns (sc). The probability of repairability can now be defined as

$$\Pr(|F| \leq sr + sc), \tag{14}$$

where $|F|$ represents the cardinality of the set F. Equation (14) represents the probability of the event when the number of stuck-at and coupling faults $|F|$, a random quantity, is less than the total number of spare rows and columns $sr + sc$. We can express (14) in terms of the number of rows or columns of a square $n \times n$ reconfigurable memory array as

$$\Pr(|F| \leq (a+b)n), \tag{15}$$

where $a = \frac{sr}{n}$ and $b = \frac{sc}{n}$. The values of a and b are bounded in the real interval $[0, 1]$, since the number of spare rows and spare columns is usually a small fraction of the total number of rows and columns in the array and can never exceed it.

In this paper, our first goal is to formally verify that if the probabilities of stuck-at and coupling fault occurrence are given by the following expressions:

$$p_s = \frac{c_1}{n} - \frac{w(n)}{n\sqrt{n}}, \tag{16}$$

$$p_c = \frac{c_2}{n^2} - \frac{w(n)}{n^2\sqrt{n}}, \tag{17}$$

where $c_1 + c_2 = a + b$ and $w(n) \to \infty$ as $n \to \infty$, then the memory array is almost always repairable. The term almost always repairable in the above context means that the probability of repairability ($\Pr(|F| \leq sr + sc)$) tends to 1 as n becomes extremely large. The above expressions for the stuck-at and coupling fault occurrence probabilities have been initially proposed and analyzed using informal techniques in [22]. Our contribution in this paper is to formally verify the above argument using the HOL theorem prover.

The first step in the proposed probabilistic analysis approach is to construct a formal model of the system in higher order logic while representing its random component as formalized random variables. In the above-mentioned memory array model, our parameter of interest is the number of faults. The behavior of a stuck-at or coupling fault occurrence in the above model can be formally represented as a Bernoulli(p) random variable with $p = p_s$ and $p = p_c$, respectively. Now, under the assumption that the occurrence of stuck-at and coupling faults are independent and identically distributed, we can model the total number of these faults as Binomial(m, p) random variables, which model an experiment that counts the number of successes in m independent Bernoulli(p) trials, with their respective probabilities as follows:

Definition 7. *Number of Stuck-at Faults*

$$\begin{aligned}\vdash\ &\forall\ \mathtt{n}\ \mathtt{c1}\ \mathtt{w}.\ \mathtt{num\_of\_faults\_stuck}\ \mathtt{n}\ \mathtt{c1}\ \mathtt{w}\\ &= \mathtt{prob\_bino}\ \mathtt{n}^2\left(\frac{\mathtt{c1}}{\mathtt{n}} - \frac{\mathtt{w(n)}}{\mathtt{n}\sqrt{\mathtt{n}}}\right).\end{aligned}$$

Definition 8. *Number of Coupling Faults*

$$\begin{aligned}\vdash\ &\forall\ \mathtt{n}\ \mathtt{c2}\ \mathtt{w}.\ \mathtt{num\_of\_faults\_coupling}\ \mathtt{n}\ \mathtt{c2}\ \mathtt{w}\\ &= \mathtt{prob\_bino}\ \mathtt{n}^3\left(\frac{\mathtt{c2}}{\mathtt{n}^2} - \frac{\mathtt{w(n)}}{\mathtt{n}^2\sqrt{\mathtt{n}}}\right).\end{aligned}$$

The functions above accept three parameters each: the number of rows or columns of a square reconfigurable memory array as a $natural$ number n, the real numbers $c1$ or $c2$, respectively, that are related to the fractions of spare rows and columns as $c1 + c2 = a + b$, and the $real$ sequence w with data type ($natural \to real$). They utilize the Binomial random variable function `prob_bino`, given in Definition 1, to return the number of stuck-at and coupling faults, respectively, for the specific case of a square $n \times n$ memory array with the fault occurrence probabilities equal to the expressions, given in (16) and (17), respectively.

Now, the total number of the faults in the memory array can be formalized as the sum of the number of stuck-at and coupling faults as follows:

Definition 9. *Total Number of Faults*

$$\begin{aligned}\vdash\ &\forall\ \mathtt{n}\ \mathtt{c1}\ \mathtt{c2}\ \mathtt{w}.\ \mathtt{num\_of\_faults}\ \mathtt{n}\ \mathtt{c1}\ \mathtt{c2}\ \mathtt{w}\\ &= \mathtt{bind}\ (\mathtt{num\_of\_faults\_stuck}\ \mathtt{n}\ \mathtt{c1}\ \mathtt{w})\\ &\quad (\lambda \mathtt{x}.\ \mathtt{bind}\ (\mathtt{num\_of\_faults\_coupling}\ \mathtt{n}\ \mathtt{c2}\ \mathtt{w})\\ &\quad (\lambda \mathtt{y}.\ \mathtt{unit}\ (\mathtt{x} + \mathtt{y}))).\end{aligned}$$

The above function accepts four parameters, n, $c1$, $c2$, and w and returns the sum of stuck-at and coupling faults, generated by functions `num_of_faults_stuck` and `num_of_faults_coupling`, respectively, using the monadic functions `bind` and `unit`.

In the probabilistic analysis of very large memory arrays, it is often required to find when repairing a fault in a memory array becomes nearly impossible. In order to be able to answer such questions, we verify an irrepairability condition, according to which the memory array is almost never repairable if the probabilities of stuck-at and coupling fault occurrence are as follows:

$$p_s = \frac{c_1}{n}, \tag{18}$$

$$p_c = \frac{c_2}{n^2}, \tag{19}$$

where $c_1 + c_2 > \frac{-[a \ln a + (1-a)\ln(1-a)]}{(1-a)^2(1-b)}$ [22]. The term almost never repairable means that the probability of having 1 or more repair solutions using an rows tends to 0 as n becomes extremely large.

Our formal model of the memory array is also capable of capturing the number of repair solutions and thus the irrepairability condition. A repair solution ceases to exist if no combination of an spare rows can be used to repair all the stuck-at and coupling faults present in the memory array. Under the assumption that the occurrence of stuck-at and coupling faults are independent and identically distributed, we can model the number of repair solutions as a Binomial(m, p) random variable with m being equal to the total number of possible repair solutions for an $n \times n$ memory

array with an spare rows, i.e., $\binom{n}{an}$, and p being equal to the probability that a specific choice of an rows constitutes a repair solution. This probability can be expressed in terms of the stuck-at and coupling fault probabilities, p_s and p_c, as $(1-p_s)^{(n-an)(n-jn)}(1-p_c)^{(n-an)^2(n-jn)}$, where $j \leq b$ [22]. Thus, the number of repair solutions can be formalized in higher order logic as follows:

Definition 10. *Number of Repair Solutions*

$$\vdash \forall\, \mathtt{n\ a\ j\ c1\ c2.num\_of\_repsoln\ n\ a\ j\ c1\ c2}$$
$$= \mathtt{prob\_bino}\ \binom{\mathtt{n}}{\lfloor \mathtt{an} \rfloor} \left[\left(1-\frac{\mathtt{c1}}{\mathtt{n}}\right)^{(\mathtt{n}-\lfloor \mathtt{an} \rfloor)(\mathtt{n}-\lfloor \mathtt{jn} \rfloor)} \left(1-\frac{\mathtt{c2}}{\mathtt{n}^2}\right)^{(\mathtt{n}-\lfloor \mathtt{an} \rfloor)(\mathtt{n}-\lfloor \mathtt{an} \rfloor)(\mathtt{n}-\lfloor \mathtt{jn} \rfloor)}\right],$$

where $\lfloor x \rfloor$ denotes the floor of x and it returns the nearest integer that is less than x.

The floor function has been used since the binomial function $\binom{m}{n}$ requires the data type of both of its arguments m and n to be positive integers. It is important to note that in the paper-and-pencil analysis of the same problem [22], the floor function was missing in the binomial expression.

5.2 Repairability Condition

In this section, we utilize the function `num_of_faults` to formally verify a couple of statistical properties regarding the number of faults and the almost always repairability condition for an n x n reconfigurable memory array with stuck-at and coupling fault occurrence probabilities given by (16) and (17), respectively. These verification results play a vital role in designing reliable reconfigurable memory arrays.

For a memory array containing independent and identically distributed stuck-at and coupling faults with probabilities p_s and p_c, given by (16) and (17), respectively, the average number of faults is given by

$$Ex[|F|] = n^2\left(\frac{c1}{n} - \frac{w(n)}{n\sqrt{n}}\right) + n^3\left(\frac{c2}{n^2} - \frac{w(n)}{n^2\sqrt{n}}\right). \tag{20}$$

This property can be formally expressed using the formal definition of expectation and our formal definition of the number of faults as follows:

Theorem 8. *Average Number of Faults*

$$\begin{aligned}\vdash\ & \forall\, \mathtt{n\ a\ b\ c1\ c2\ w}.\\ & (0 \leq \mathtt{a}) \wedge (\mathtt{a} \leq 1) \wedge (0 \leq \mathtt{b}) \wedge (\mathtt{b} \leq 1)\\ & \wedge (\mathtt{c1} + \mathtt{c2} = \mathtt{a} + \mathtt{b}) \wedge (1 < \mathtt{n})\\ & \wedge (\forall\, \mathtt{n}.\ (0 < \mathtt{w(n)})\\ & \wedge (\mathtt{w(n)} < (\mathtt{min}\ \mathtt{c1}\sqrt{\mathtt{n}}\ \mathtt{c2}\sqrt{\mathtt{n}})))\\ & \Rightarrow \mathtt{expec}\ (\lambda \mathtt{s}.\ \mathtt{num\_of\_faults\ n\ c1\ c2\ w\ s})\\ & = \mathtt{n}^2\left(\frac{\mathtt{c1}}{\mathtt{n}} - \frac{\mathtt{w(n)}}{\mathtt{n}\sqrt{\mathtt{n}}}\right) + \mathtt{n}^3\left(\frac{\mathtt{c2}}{\mathtt{n}^2} - \frac{\mathtt{w(n)}}{\mathtt{n}^2\sqrt{\mathtt{n}}}\right)\end{aligned}$$

where the HOL function `min` returns the minimum value of its two real arguments.

The first four assumptions in the above theorem ensure that the fractions a and b are bounded by the interval $[0, 1]$ as has been described in the previous section. The relationship between $c1, c2$ and a, b is given in the fifth assumption. Whereas, the precondition $1 < n$ has been used in order to ensure that the given memory array has more than one cell. The last assumption is about the real sequence w and basically provides its upper and lower bounds. These bounds have been used in order to prevent the stuck-at and coupling fault occurrence probabilities p_s and p_c, given in (16) and (17), from falling outside their allowed interval $[0, 1]$. It is interesting to note that no such restriction on the sequence w was imposed in the paper-and-pencil-based analysis of the repairability problem given in [22]. This fact clearly demonstrates the strength of the proposed approach as it allowed us to highlight this corner case, which if ignored could lead to the invalidation of the whole analysis. The conclusion of Theorem 8 presents the mathematical relation given in (20).

We proceed with the verification of Theorem 8 by simplifying the left-hand side (LHS) of its proof goal using the linearity of expectation property, given in Definition 4, as follows:

$$\begin{aligned}&\mathtt{expec}\ (\lambda \mathtt{s.num\_of\_faults\_stuck\ n\ c1\ w\ s})\\ &\quad + \mathtt{expec}\ (\lambda \mathtt{s.num\_of\_faults\_couplin\ g\ n\ c2\ w\ s})\\ &= \mathtt{n}^2\left(\frac{(\mathtt{a}+\mathtt{b})}{\mathtt{n}} - \frac{\mathtt{w(n)}}{\mathtt{n}\sqrt{\mathtt{n}}}\right) + \mathtt{n}^3\left(\frac{\mathtt{c2}}{\mathtt{n}^2} - \frac{\mathtt{w(n)}}{\mathtt{n}^2\sqrt{\mathtt{n}}}\right).\end{aligned}$$

Next, we verify the following two theorems for the expectation of the number of stuck-at and coupling faults based on their definitions, given in Definitions 7 and 8, respectively, and the expectation of Binomial random variable, given in Theorem 3.

Theorem 9. *Average Number of Stuck-at Faults*

$$\begin{aligned}\vdash\ & \forall\, \mathtt{n\ c1\ w}.\ (1 < \mathtt{n})\\ & \wedge (\forall\, \mathtt{n}.\ (0 < \mathtt{w(n)}) \wedge (\mathtt{w(n)} < \mathtt{c1}\sqrt{\mathtt{n}}))\\ & \Rightarrow \mathtt{expec}\ (\lambda \mathtt{s}.\ \mathtt{num\_of\_faults\_stuck\ n\ c1\ w\ s})\\ & = \mathtt{n}^2\left(\frac{(\mathtt{a}+\mathtt{b})}{\mathtt{n}} - \frac{\mathtt{w(n)}}{\mathtt{n}\sqrt{\mathtt{n}}}\right).\end{aligned}$$

Theorem 10. *Average Number of Coupling Faults*

$$\begin{aligned}\vdash\ & \forall\, \mathtt{n\ c2\ w}.\ (1 < \mathtt{n})\\ & \wedge (\forall\, \mathtt{n}.(0 < \mathtt{w(n)}) \wedge (\mathtt{w(n)} < \mathtt{c2}\sqrt{\mathtt{n}}))\\ & \Rightarrow \mathtt{expec}\ (\lambda \mathtt{s}.\ \mathtt{num\_of\_faults\_coupling\ n\ c2\ w\ s})\\ & = \mathtt{n}^3\left(\frac{(\mathtt{a}+\mathtt{b})}{\mathtt{n}^2} - \frac{\mathtt{w(n)}}{\mathtt{n}^2\sqrt{\mathtt{n}}}\right).\end{aligned}$$

The above two theorems can now be used to conclude the HOL proof of Theorem 8.

The variance of the total number of faults for an $n \times n$ memory array, with the probabilities of stuck-at and coupling fault occurrence, given by (16) and (17), is given by

$$Var[|F|] = n^2(p_s)(1-p_s) + n^3(p_c)(1-p_c). \tag{21}$$

This property can be formally expressed using the formal definition of variance and the formal definition of the number of faults as follows:

Theorem 11. *Variance of the total Number of Faults*

$$\begin{aligned}\vdash \forall\, \mathtt{n\ a\ b\ c1\ c2\ w\ s}.\\ (0 \le \mathtt{a}) \wedge (\mathtt{a} \le 1) \wedge (0 \le \mathtt{b}) \wedge (\mathtt{b} \le 1)\\ \wedge (\mathtt{c1} + \mathtt{c2} = \mathtt{a} + \mathtt{b}) \wedge (1 < \mathtt{n})\\ \wedge (\forall\, \mathtt{n}.(0 < \mathtt{w(n)})\\ \wedge (\mathtt{w(n)} < (\mathtt{min\ c1}\sqrt{\mathtt{n}}\ \mathtt{c2}\sqrt{\mathtt{n}})))\\ \Rightarrow \mathtt{variance}\ (\lambda \mathtt{s.num\_of\_faults\ n\ c1\ c2\ w\ s})\\ = \mathtt{n}^2\left(\frac{\mathtt{c1}}{\mathtt{n}} - \frac{\mathtt{w(n)}}{\mathtt{n}\sqrt{\mathtt{n}}}\right)\left(1 - \left(\frac{\mathtt{c1}}{\mathtt{n}} - \frac{\mathtt{w(n)}}{\mathtt{n}\sqrt{\mathtt{n}}}\right)\right)\\ + \mathtt{n}^3\left(\frac{\mathtt{c2}}{\mathtt{n}^2} - \frac{\mathtt{w(n)}}{\mathtt{n}^2\sqrt{\mathtt{n}}}\right)\left(1 - \left(\frac{\mathtt{c2}}{\mathtt{n}^2} - \frac{\mathtt{w(n)}}{\mathtt{n}^2\sqrt{\mathtt{n}}}\right)\right).\end{aligned}$$

The HOL verification of Theorem 11 is based on the linearity of variance property, given in (8), and the variance characteristics of the Binomial random variable. The proof steps are very similar to the ones for Theorem 8.

A tail distribution bound of the number of faults for an $n \times n$ memory array, with the probabilities of stuck-at and coupling fault occurrence, given by (16) and (17), can be expressed as follows:

$$Pr(|F| \le (a+b)n) \ge 1 - \frac{n^2(p_s)(1-p_s) + n^3(p_c)(1-p_c)}{4n(w(n))^2}. \tag{22}$$

Whereas, the corresponding HOL theorem is as follows:

Theorem 12. *Tail Distribution Bound for the number of Faults*

$$\begin{aligned}\vdash\ \forall\, \mathtt{n\ a\ b\ c1\ c2\ w\ s}.\\ (0 \le \mathtt{a}) \wedge (\mathtt{a} \le 1) \wedge (0 \le \mathtt{b}) \wedge (\mathtt{b} \le 1)\\ \wedge (\mathtt{c1} + \mathtt{c2} = \mathtt{a} + \mathtt{b}) \wedge (1 < \mathtt{n})\\ \wedge (\forall\, \mathtt{n}.\ (0 < \mathtt{w(n)})\\ \wedge (\mathtt{w(n)} < (\mathtt{min\ c1}\ \sqrt{\mathtt{n}}\ \mathtt{c2}\ \sqrt{\mathtt{n}})))\\ \Rightarrow (\mathbb{P}\{\mathtt{s} \mid (\mathtt{fst}(\mathtt{num\_of\_faults\ n\ c1\ c2\ w\ s})) \le (\mathtt{a}+\mathtt{b})\ \mathtt{n}\}\\ \ge 1 - \left(\frac{\mathtt{variance}(\lambda \mathtt{s.\ num\_of\_faults\ n\ c1\ c2\ w\ s})}{4\mathtt{n}(\mathtt{w(n)})^2}\right).\end{aligned}$$

The proof of Theorem 12 involves the less-than-or-equal-to $(\forall \mathtt{x\ y\ z}.\ (\mathtt{x} \le \mathtt{y}) \wedge (\mathtt{y} \le \mathtt{z}) \Rightarrow (\mathtt{x} \le \mathtt{z}))$ transitive property of reals, the basic probability increasing axiom $(\forall \mathtt{A\ B}.\ \mathtt{A} \subseteq \mathtt{B} \Rightarrow \mathbb{P}(\mathtt{A}) \le \mathbb{P}(\mathtt{B}))$, the complement probability law $(\forall \mathtt{A}.\ \mathbb{P}(\bar{\mathtt{A}}) = 1 - \mathbb{P}(\mathtt{A}))$, the absolute value theorem $((|\mathtt{y} - \mathtt{x}| < \mathtt{d}) = (\mathtt{x} - \mathtt{d} < \mathtt{y} < \mathtt{x} + \mathtt{d}))$, along with the expectation and variance of total number of faults given in Theorems 8 and 11, respectively, and the Chebyshev's inequality, given in Theorem 7.

Finally, we use the statistical properties verified so far to analyze the repairability problem, i.e., an $n \times n$ reconfigurable memory array with the probabilities of stuck-at and coupling fault occurrence, given by (16) and (17), is almost always repairable.

$$\lim_{n\to\infty} \Pr(|F| \le (a+b)n) = 1. \tag{23}$$

The corresponding HOL theorem is as follows:

Theorem 13. *Repairability Condition*

$$\begin{aligned}\vdash \forall\, \mathtt{a\ b\ w}.\\ (0 \le \mathtt{a}) \wedge (\mathtt{a} \le 1) \wedge (0 \le \mathtt{b}) \wedge (\mathtt{b} \le 1)\\ \wedge (\mathtt{c1} + \mathtt{c2} = \mathtt{a} + \mathtt{b}) \wedge (1 < \mathtt{n})\\ \wedge (\forall\, \mathtt{n}.\ (0 < \mathtt{w(n)})\\ \wedge (\mathtt{w(n)} < (\mathtt{min\ c1}\ \sqrt{\mathtt{n}}\ \mathtt{c2}\ \sqrt{\mathtt{n}})))\\ \wedge \left(\mathtt{lim}\left(\lambda \mathtt{n}.\ \frac{1}{\mathtt{w(n)}}\right) = 0\right)\\ \Rightarrow (\mathtt{lim}\ (\lambda \mathtt{n}.\ \mathbb{P}\{\mathtt{s} \mid (\mathtt{fst}\ (\mathtt{num\_of\_faults\ n\ c1\ c2\ w\ s}))\\ \le (\mathtt{a}+\mathtt{b})\ \mathtt{n}\}) = 1)),\end{aligned}$$

where `lim M` *represents the HOL formalization of the limit of a real sequence* M *(i.e.,* $lim\ M = \lim_{n\to\infty} M(n)$*) [27].*

A new assumption $(\lim(\lambda\, \mathtt{n}.\, \frac{1}{\mathtt{w(n)}}) = 0)$ has been added that formally represents the intrinsic characteristic of *real* sequence w that it tends to infinity as its *natural* argument becomes extremely large.

The verification of Theorem 13 is based on the basic probability axiom $(\forall \mathtt{A}.\ \Pr(\mathtt{A}) \le 1)$, the transitivity property of less-than-or-equal-to for real numbers, and the fact that the term involving variance on the right-hand side of the inequality in Theorem 12 approaches zero as n becomes very large.

5.3 Irrepairability Condition

In this section, we utilize the function `num_of_repsoln` to formally verify the irrepairability condition for an $n \times n$ reconfigurable memory array. For a memory array containing independent and identically distributed stuck-at and coupling faults with probabilities p_s and p_c, given by (18) and (19), respectively, the average number of repair solutions is

$$Ex[U] = \binom{n}{an}\left(1 - \frac{c1}{n}\right)^{(n-an)(n-jn)}\left(1 - \frac{c2}{n^2}\right)^{(n-an)^2(n-jn)}, \tag{24}$$

where U represents the Bernoulli random variable that models the number of repair solutions. We formalized this property using the definitions of Bernoulli random variable, expectation, and the number of repair solutions.

Theorem 14. *Average Number of Repair Solutions*

$$\begin{aligned}\vdash \forall\, \mathtt{n\ a\ a1\ a2\ j\ c1\ c2}.\ \left(\mathtt{a} = \frac{\mathtt{a1}}{\mathtt{a2}}\right)\\ \wedge (0 \le \mathtt{a1}) \wedge (\mathtt{a1} \le \mathtt{a2}) \wedge (0 \le \mathtt{b}) \wedge (\mathtt{b} \le 1)\\ \wedge (0 \le \mathtt{j}) \wedge (\mathtt{j} \le \mathtt{b}) \wedge (0 < \mathtt{c1}) \wedge (\mathtt{c1}\\ < (\mathtt{na2})) \wedge (0 < \mathtt{c2}) \wedge (\mathtt{c2} < (\mathtt{na2})^2)\\ \Rightarrow \mathtt{expec}(\lambda \mathtt{s.num\_of\_repsoln}\ (\mathtt{na2})\ \mathtt{a\ j\ c1\ c2\ s})\\ = \binom{(\mathtt{na2})}{\lfloor \mathtt{a(na2)} \rfloor}\left[\left(1 - \frac{\mathtt{c1}}{\mathtt{na2}}\right)^{(\mathtt{na2} - \lfloor \mathtt{a(na2)} \rfloor)((\mathtt{na2}) - \lfloor \mathtt{j(na2)} \rfloor)}\right.\\ \left.\left(1 - \frac{\mathtt{c2}}{(\mathtt{na2})^2}\right)^{((\mathtt{na2}) - \lfloor \mathtt{a(na2)} \rfloor)((\mathtt{na2}) - \lfloor \mathtt{a(na2)} \rfloor)((\mathtt{na2}) - \lfloor \mathtt{j(na2)} \rfloor)}\right].\end{aligned}$$

The variable a, which is a constant and represents the ratio of the spare rows or columns to the total number of rows or columns in a square memory array, has been declared as a ratio of two positive integers $a1$ and $a2$. Similarly, we restrict the size of the square memory array, i.e., the number of rows or columns, to be only equal to a multiple of $a2$ and thus $na2$ has been used instead of n in the above theorem. These preconditions are used to ensure that the product of a and the total number of rows or columns should always be equal to an integer value that represents the number of spare rows. Besides the bounds on a, b, and j, bounds on the values of $c1$ and $c2$ have also been assumed in the above theorem. These bounds have been used in order to prevent the stuck-at and coupling fault occurrence probabilities p_s and p_c, given in (18) and (19), from falling outside their allowed interval $[0, 1]$. The conclusion of Theorem 14 formally presents the expectation relation of the number of repair solutions, given in (24). This theorem can be verified using the definition of the function `num_of_repsoln` and the expectation property for the Binomial random variable, given in Theorem 3, along with the fact that the probability of success for the Binomial random variable of the function `num_of_repsoln` lies in the interval $[0, 1]$.

In order to analyze the irrepairability condition, we are interested in the probability that 1 or more repair solutions exist. This probability has the following tail distribution bound for the case of an $n \times n$ memory array:

$$Pr(U > 0) < \left(2^{H(a)} e^{-c1(1-a)^2(1-b)} e^{-c2(1-a)^2(1-b)}\right)^n, \qquad (25)$$

where H is the binary entropy function [22].

$$H(x) = -\frac{-xln(x) - (1-a)ln(1-a)}{ln2}$$

The corresponding HOL theorem for the above tail distribution bound is as follows:

Theorem 15. *Tail Distribution Bound for Repair Solutions*

$$\begin{aligned}
\vdash \forall\, \mathtt{n\ a\ a1\ a2\ j\ c1\ c2}.\ & \left(\mathtt{a} = \frac{\mathtt{a1}}{\mathtt{a2}}\right) \\
& \wedge (0 \le \mathtt{a1}) \wedge (\mathtt{a1} \le \mathtt{a2}) \wedge (0 \le \mathtt{b}) \wedge (\mathtt{b} \le 1) \\
& \wedge (0 \le \mathtt{j}) \wedge (\mathtt{j} \le \mathtt{b}) \wedge (0 < \mathtt{c1}) \wedge (\mathtt{c1} < \mathtt{n}) \\
& \wedge (0 < \mathtt{c2}) \wedge (\mathtt{c2} < (\mathtt{na2})^2) \\
& \wedge \mathtt{c1} + \mathtt{c2} > \frac{-[\mathtt{a}\ln\mathtt{a} + (1-\mathtt{a})\ln(1-\mathtt{a})]}{(1-\mathtt{a})^2(1-\mathtt{b})} \\
\Rightarrow\ & (\mathbb{P}\{\mathtt{s} \mid \mathtt{fst}(\mathtt{num_of_repsoln\ na2\ a\ j\ c1\ c2\ s}) > 0\} \\
& < \left(\mathtt{e}^{-\mathtt{x}\ln(\mathtt{x})-(1-\mathtt{a})\ln(1-\mathtt{a})}\mathtt{e}^{-\mathtt{c1}(1-\mathtt{a})^2(1-\mathtt{b})}\mathtt{e}^{-\mathtt{c2}(1-\mathtt{a})^2(1-\mathtt{b})}\right)^{(\mathtt{na2})}).
\end{aligned}$$

We proceed with the verification of this theorem by splitting its proof goal into two subgoals as follows:

$$\begin{aligned}
& \mathbb{P}\{\mathtt{s} \mid \mathtt{fst}(\mathtt{num_of_repsoln\ na2\ a\ j\ c1\ c2\ s}) > 0\} \\
& \quad \le \mathtt{expec}(\lambda \mathtt{s}.\mathtt{num_of_repsoln\ na2\ a\ j\ c1\ c2\ s})
\end{aligned}$$

$$\begin{aligned}
& \mathtt{expec}(\lambda \mathtt{s}.\mathtt{num_of_repsoln\ na2\ a\ j\ c1\ c2\ s}) \\
& \quad < \left(\mathtt{e}^{-\mathtt{x}\ln(\mathtt{x})-(1-\mathtt{a})\ln(1-\mathtt{a})}\mathtt{e}^{-\mathtt{c1}(1-\mathtt{a})^2(1-\mathtt{b})}\mathtt{e}^{-\mathtt{c2}(1-\mathtt{a})^2(1-\mathtt{b})}\right)^{(\mathtt{na2})}.
\end{aligned}$$

The first subgoal can be verified using the Markov's inequality, verified in Theorem 6, as the set `{s|fst(num_of_repsoln na2 a j c1 c2 s) > 0}` is equivalent to the set `{s|fst(num_of_repsoln na2 a j c1 c2 s)≥1}`. Whereas, the second subgoal can be simplified using Theorem 14 as follows:

$$\begin{aligned}
& \binom{(\mathrm{na2})}{\lfloor \mathrm{an} \rfloor} \left(1 - \frac{\mathrm{c1}}{(\mathrm{na2})}\right)^{((\mathrm{na2})-\lfloor \mathrm{a(na2)} \rfloor)((\mathrm{na2})-\lfloor \mathrm{j(na2)} \rfloor)} \\
& \left(1 - \frac{\mathrm{c2}}{(\mathrm{na2})^2}\right)^{((\mathrm{na2})-\lfloor \mathrm{a(na2)} \rfloor)((\mathrm{na2})-\lfloor \mathrm{a(na2)} \rfloor)((\mathrm{na2})-\lfloor \mathrm{j(na2)} \rfloor)} \\
& \quad < \left(\mathrm{e}^{-\mathrm{xln(x)}-(1-\mathrm{a})\ln(1-\mathrm{a})}\mathrm{e}^{-\mathrm{c1}(1-\mathrm{a})^2(1-\mathrm{b})}\mathrm{e}^{-\mathrm{c2}(1-\mathrm{a})^2(1-\mathrm{b})}\right)^{(\mathrm{na2})}.
\end{aligned}$$

In order to verify the above inequality, we verified the following alternate definition of the exponential function:

Lemma 1. *Exponential Function*

$$\vdash \forall\, \mathtt{x}.\ \mathtt{lim}\left(\lambda \mathtt{n}.\ \left(1 + \frac{\mathtt{x}}{\mathtt{n}}\right)^{\mathtt{n}}\right) = \mathtt{e}^{\mathtt{x}},$$

using the formalized power-series-based definition of the exponential function, i.e., $e^x = \sum_{n=0}^{\infty} \frac{x^n}{n!}$, in HOL [27].

The formal proof of Lemma 1 involves the verification of L'Hopital's rule and the formal definitions of limit of a real sequence and limit of a function at a point along with some rigorous arithmetic reasoning. Once verified, Lemma 1 can be used along with the monotonically increasing property of the real sequence $(\lambda\, \mathrm{n}.\ (1 + \frac{\mathrm{x}}{\mathrm{n}})^{\mathrm{n}})$, when $|x| < n$, to prove the following useful result:

$$\vdash \forall\, \mathtt{n\ x}.\ |\mathtt{x}| < \mathtt{n} \Rightarrow \left(1 + \frac{\mathtt{x}}{\mathtt{n}}\right)^{\mathtt{n}} \le \mathtt{e}^{\mathtt{x}}.$$

The above relationship, along with some arithmetic reasoning, can now be used to discharge the remaining subgoal of Theorem 15. This also concludes the verification of our desired tail distribution bound.

Finally, we use the statistical properties verified so far to analyze the irrepairability property, i.e., a reconfigurable memory array with the probabilities of stuck-at and coupling fault occurrence, given by (18) and (19), is almost never repairable.

$$\lim_{n \to \infty} \Pr(U > 0) = 0. \qquad (26)$$

The corresponding HOL theorem is as follows:

Theorem 16. *Irrepairability Condition*

$$\begin{aligned}
\vdash \forall\, \mathtt{a\ a1\ a2\ j\ c1\ c2}.\ & \left(\mathtt{a} = \frac{\mathtt{a1}}{\mathtt{a2}}\right) \\
& \wedge (0 \le \mathtt{a1}) \wedge (\mathtt{a1} \le \mathtt{a2}) \wedge (0 \le \mathtt{b}) \wedge (\mathtt{b} \le 1) \\
& \wedge (0 \le \mathtt{j}) \wedge (\mathtt{j} \le \mathtt{b}) \wedge (0 < \mathtt{c1}) \wedge (0 < \mathtt{c2}) \\
& \wedge \mathtt{c1} + \mathtt{c2} > \frac{-[\mathtt{a}\ln\mathtt{a} + (1-\mathtt{a})\ln(1-\mathtt{a})]}{(1-\mathtt{a})^2(1-\mathtt{b})} \\
\Rightarrow\ & (\mathtt{lim}\ (\lambda \mathtt{n}.\ \mathbb{P}\ \{\mathtt{s} \mid \mathtt{fst} \\
& (\mathtt{num_of_repsoln\ na2\ a\ j\ c1\ c2\ s}) > 0\}) = 0).
\end{aligned}$$

The proof of Theorem 16 is very similar to Theorem 13. Though, in this case, we use the basic probability axiom $(\forall A.0 \le Pr(A))$ and the fact that the limit value of the tail

distribution bound of $Pr(U > 0)$, given in Theorem 15, is 0 since the expression is less than 1.

The above results clearly demonstrate the effectiveness of theorem-proving-based reliability analysis. Due to the formal nature of the models, the high expressiveness of higher order logic, and the inherent soundness of theorem proving, we have been able to verify generic properties of interest that are valid for any given memory array with 100 percent precision; a novelty which is not available in simulation. Similarly, we have been able to formally analyze properties that cannot be handled by model checking. The proposed approach is also superior to the paper-and-pencil-proof methods in a way as the chances of making human errors, missing critical assumptions, and proving wrongful statements are almost nil, since all proof steps are applied within the sound core of the HOL theorem prover. These additional benefits come at the cost of the time and effort spent, while formalizing the memory array and formally reasoning about its properties. But, the fact that we were building on top of already verified probability and reliability theory foundations, described in Sections 3 and 4, helped significantly in this regard as the memory analysis only consumed approximately 250 man-hours and 3,500 lines of HOL code.

6 Conclusions

In this paper, we utilized the probability theory formalized in higher order logic to construct a formal reliability analysis approach. The main idea behind this approach is to use formalized random variables to model systems and to verify the corresponding reliability characteristics in a theorem prover. We also formalized the definition of reliability and formally verified the Markov and Chebyshev's inequalities, which play a vital role in reliability analysis. Because of the formal nature of the models, the proposed reliability analysis is free of approximation and precision errors and due to the high expressive nature of higher order logic a wider range of systems can be analyzed. This makes the proposed approach very promising for the reliability analysis of safety critical and highly sensitive engineering and scientific applications.

The proposed approach was used to analyze the repairability problem of reconfigurable memory arrays in the presence of stuck-at and coupling faults. We first developed a higher order logic based formal stuck-at and coupling fault model for reconfigurable memory arrays, and based on this model we formally verified some key statistical properties and the repairability and irrepairability conditions. The formally verified expectation and variance properties and the Markov and Chebyshev's inequalities greatly helped us to speed up the analysis process. The results obtained are 100 percent precise and confirmed the results obtained via analytical approaches. Another distinguishing feature of these properties is their generic nature, i.e., they can be utilized to assess the reliability of any reconfigurable memory array. The successful handling of this real-world reliability analysis problem by the proposed approach clearly demonstrates its feasibility for other reliability analysis issues. To the best of our knowledge, this is the first study on using formal methods for the reliability analysis of reconfigurable memory arrays with both stuck-at and coupling faults.

The fundamentals associated with theorem-proving-based reliability analysis, presented in this paper, can certainly be applied to many other domains besides the illustrative example of memory arrays. An ongoing project in our research group is to utilize the formalized probability theory to analyze the reliability of Boolean logic circuits. The approach, mainly inspired from the probabilistic gate models (PGM) based reliability analysis [14], utilizes the formalized Bernoulli random variables to model the gate failure phenomena and the input arrival patterns at the logic gates. The reliability of a component can be formally defined in this case as the probability of having a correct output. The goal of this work is to formally verify probabilistic properties, by building on top of the infrastructure presented in Sections 3 and 4, associated with the reliability of commonly used logic circuits like decoders, multiplexors, and adders. The main benefits of conducting such analysis using theorem proving include the accuracy of the results and the generic nature of the properties.

The proposed approach is certainly not mature enough to handle all kinds of reliability problems. We do not have the formalization infrastructure to express and reason about statistical properties related to continuous random variables yet. Similarly, we also lack the formalization of other integration-theory-related reliability characteristics, such as mean time to failure (MTTF) and hazard rates [29]. Though, the higher order logic formalization of a domain independent integration theory, like the Lesbesgue's integration, can pave the path to resolve these bottlenecks. A major limitation of our approach is the associated user interaction, i.e., the user needs to guide the proof tools manually since we are dealing with higher order logic, which is known to be nondecidable. On the other hand, simulation is capable of handling all sorts of reliability analysis problems in an automated way but the solutions provided are not exact. Whereas, probabilistic model checking is capable of providing exact answers for a subset of reliability analysis problems. We believe that all these three techniques have to play together in order to form a successful reliability analysis approach. For example, an efficient approach would be to use simulation for the less critical parts of the analysis, model checking for the critical parts that it can handle, and theorem proving for the remaining critical parts.

References

[1] L. Devroye, *Non-Uniform Random Variate Generation.* Springer-Verlag, 1986.
[2] D. MacKay, "Introduction to Monte Carlo Methods," *Learning in Graphical Models,* pp. 175-204, Kluwer Academic Press, 1998.
[3] Mars Polar Lander, http://mpfwww.jpl.nasa.gov/msp98/, 2008.
[4] A. Gupta, "Formal Hardware Verification Methods: A Survey," *Formal Methods in System Design,* vol. 1, nos. 2/3, pp. 151-238, 1992.
[5] E. Clarke, O. Grumberg, and D. Long, "Verification Tools for Finite State Concurrent Systems," *A Decade of Concurrency-Reflections and Perspectives,* pp. 124-175, Springer, 1993.
[6] M. Gordon, "Mechanizing Programming Logics in Higher-Order Logic," *Current Trends in Hardware Verification and Automated Theorem Proving,* pp. 387-439, Springer, 1989.
[7] J. Hurd, "Formal Verification of Probabilistic Algorithms," PhD thesis, Univ. of Cambridge, 2002.

[8] O. Hasan, "Formal Probabilistic Analysis Using Theorem Proving," PhD thesis, Concordia Univ., 2008.
[9] P. Billingsley, *Probability and Measure.* John Wiley, 1995.
[10] A. Miczo, *Digital Logic Testing and Simulation.* Wiley Interscience, 2003.
[11] S. Kuo and W. Fuchs, "Efficient Spare Allocation for Reconfigurable Arrays," *IEEE Design and Test of Computers,* vol. 4, no. 1, pp. 24-31, Feb. 1987.
[12] M. Gordon and T. Melham, *Introduction to HOL: A Theorem Proving Environment for Higher-Order Logic.* Cambridge Univ. Press, 1993.
[13] S. Krishnaswamy, G.F. Viamonte, I.L. Markov, and J.P. Hayes, "Accurate Reliability Evaluation and Enhancement via Probabilistic Transfer Matrices," *Proc. Design, Automation and Test in Europe,* pp. 282-287, 2005.
[14] J. Han, E. Taylor, J. Gao, and J. Fortes, "Faults, Error Bounds and Reliability of Nanoelectronic Circuits," *Proc. Application Specific System Architectures and Processors,* pp. 247-253, 2005.
[15] M.R. Choudhury and K. Mohanram, "Reliability Analysis of Logic Circuits," *IEEE Trans. Computer-Aided Design of Integrated Circuits and Systems,* vol. 28, no. 3, pp. 392-405, Mar. 2009.
[16] D. Bhaduri and S. Shukla, "NANOPRISM: A Tool for Evaluating Granularity versus Reliability Trade Offs in Nano Architectures," *Proc. Great Lakes Symp. Very Large Scale Integration (VLSI),* pp. 109-112, 2004.
[17] G. Norman, D. Parker, M. Kwiatkowska, and S. Shukla, "Evaluating the Reliability of NAND Multiplexing with PRISM," *IEEE Trans. Computer-Aided Design of Integrated Circuits and Systems,* vol. 24, no. 10, pp. 1629-1637, Oct. 2005.
[18] E. Clarke, O. Grumberg, and D. Peled, *Model Checking.* The MIT Press, 2000.
[19] M. Nicolaidis, N. Achouri, and L. Anghel, "A Diversified Memory Built-in Self-Repair Approach for Nanotechnologies," *Proc. 22nd IEEE Very Large Scale Integration (VLSI) Test Symp.,* pp. 313-318, 2004.
[20] A. Sehgal, A. Dubey, E. Marinissen, C. Wouters, H. Vranken, and K. Chakrabarty, "Redundancy Modelling and Array Yield Analysis for Repairable Embedded Memories," *IEE Proc. Computers and Digital Techniques,* vol. 152, no. 1, pp. 97-106, 2005.
[21] W. Shi and W.K. Fuchs, "Probabilistic Analysis and Algorithms for Reconfiguration of Memory Arrays," *IEEE Trans. Computer-Aided Design of Integrated Circuits and Systems,* vol. 11, no. 9, pp. 1153-1160, Sept. 1992.
[22] C.P. Low and H.W. Leong, "Probabilistic Analysis of Memory Reconfiguration in the Presence of Coupling Faults," *Proc. IEEE Int'l Workshop Defect and Fault Tolerance in Very Large Scale Integration (VLSI) Systems,* pp. 157-166, 1992.
[23] D. Blough, "Performance Evaluation of a Reconfiguration-Algorithm for Memory Arrays Containing Clustered Faults," *IEEE Trans. Reliability,* vol. 45, no. 2, pp. 274-284, June 1996.
[24] O. Hasan, N. Abbasi, and S. Tahar, "Formal Probabilistic Analysis of Stuck-at Faults in Reconfigurable Memory Arrays," *Integrated Formal Methods,* pp. 277-291, Springer, 2009.
[25] R. Yates and D. Goodman, *Probability and Stochastic Processes: A Friendly Introduction for Electrical and Computer Engineers.* Wiley, 2005.
[26] A. Levine, *Theory of Probability.* Addison-Wesley, 1971.
[27] J. Harrison, *Theorem Proving with the Real Numbers.* Springer, 1998.
[28] S. Richter, "Formalizing Integration Theory, with an Application to Probabilistic Algorithms," Diploma thesis, Dept. of Informatics, Technische Universitat Munchen, 2003.
[29] K. Tridevi, *Probability and Statistics with Reliability, Queuing and Computer Science Applications.* Wiley Interscience, 2002.
[30] M. Choi, N. Park, and F. Lombardi, "Hardware-Software Co-Reliability in Field Reconfigurable Multi-Processor-Memory Systems," *Proc. Int'l Parallel and Distributed Processing Symp.,* pp. 170-184, 2002.
[31] J.R. Cavallaro and I.D. Walker, "A Survey of NASA and Military Standards on Fault Tolerance and Reliability Applied to Robotics," *Proc. AIAA/NASA Conf. Intelligent Robots in Field, Factory, Service, and Space,* pp. 282-286, 1994.
[32] M. Chang, W.K. Fuchs, and J.H. Patel, "Diagnosis and Repair of Memory with Coupling Faults," *IEEE Trans. Computers,* vol. 38, no. 4 pp. 493-500, Apr. 1989.

Osman Hasan received the BEng (Hons) degree from the N-W.F.P University of Engineering and Technology, Pakistan, in 1997, and the MEng and PhD degrees from Concordia University, Montreal, Quebec, Canada, in 2001 and 2008, respectively. He served as an ASIC design engineer from 2001 to 2003 in the industry prior to joining Concordia University in 2004 for his PhD. Currently, he is a postdoctoral fellow with the Hardware Verification Group, Concordia University, Montreal, Quebec, Canada. His current research interests include formal methods, higher order logic theorem proving, probabilistic analysis, and formal reliability analysis of systems. He is a student member of the IEEE.

Sofiène Tahar (M'96-SM'07) received the diploma degree in computer engineering from the University of Darmstadt, Germany, in 1990, and the PhD degree with distinction in computer science from the University of Karlsruhe, Germany, in 1994. Currently, he is a professor with the Department of Electrical and Computer Engineering, Concordia University, Montreal, Quebec, Canada. He is the founder and director of the Hardware Verification Group at Concordia University. From 2001 to 2006, he held a junior Concordia University research chair in Formal Verification of Microelectronics Systems. In 2007, he was appointed senior Concordia University research chair in Formal Verification of System-on-Chip. He has made contributions and published papers in the areas of formal hardware verification, microprocessor and system-on-chip verification, analog and mixed signal circuits verification, VLSI design automation, and probabilistic, statistical, and reliability analysis of systems. He is a professional engineer in the province of Quebec. He has been organizing and involved in program committees of various international conferences in the areas of formal methods and design automation. In 1998, he received the Canada Foundation for Innovation (CFI) Researcher Award. In 2007, he was named university research fellow upon receiving the University Research Award. He is a senior member of the IEEE.

Naeem Abbasi received the BSc degree from the University of Engineering and Technology, Lahore, Pakistan, in 1991, and the MSEE degree from Columbia University in New York in 1995. He is currently pursuing the PhD degree from Concordia University, Montreal, Quebec, Canada. His research interests include VLSI design, algorithms and architectures for DSP, formal methods, higher order logic theorem proving, and formal statistical analysis of circuits and systems. He is a student member of the IEEE.