Formal Reliability Analysis of Combinational Circuits using Theorem Proving

Osman Hasan[a,*], Jigar Patel[b], Sofiène Tahar[b]

[a]*School of Electrical Engineering and Computer Science*
National University of Science and Technology
Sector H-12, Islamabad, Pakistan
[b]*Department of Electrical and Computer Engineering,*
Concordia University
1455 de Maisonneuve W., Montreal, Quebec, H3G 1M8, Canada

Abstract

Reliability analysis of combinational circuits has become imperative these days due to the extensive usage of nanotechnologies in their fabrication. Traditionally, reliability analysis of combinational circuits is done using simulation or paper-and-pencil proof methods. But, these techniques do not ensure accurate results and thus may lead to disastrous consequences when dealing with safety-critical applications. In this paper, we mainly tackle the accuracy problem of these traditional reliability analysis approaches by presenting a formal reliability analysis framework based on higher-order-logic theorem proving. We present the higher-order-logic formalization of the notions of fault and reliability for combinational circuits and formally verify the von-Neumann fault models for most of the commonly used logic gates, such as, AND, NOT, OR, etc. This formal infrastructure is then used along with a computer program, written in C++, to automatically reason about the reliability of any combinational circuit within a higher-order-logic theorem prover (HOL). For illustration purposes, we utilize the proposed framework to analyze the reliability of a few benchmark combinational circuits.

Keywords: Formal Verification, Higher-order logic, HOL, Probability Theory, Reliability Analysis, Theorem Proving

*Corresponding Author
Email addresses: osman.hasan@seecs.nust.edu.pk (Osman Hasan), ji_p@ece.concordia.ca (Jigar Patel), tahar@ece.concordia.ca (Sofiène Tahar)

1. Introduction

Reliability analysis involves the usage of probabilistic 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. This information is in turn utilized to design more reliable and secure systems. The reliability analysis of combinational circuits has been conducted since their early introduction [20, 22]. However, the ability to efficiently analyze the reliability of combinational circuits has become very challenging nowadays because of their growing sizes and complexity and the inherent variability in the nanoscale fabrication processes.

Traditionally, simulation has been the most commonly used computer based reliability analysis technique for combinational circuits, e.g., see [18, 11, 12]. Most simulation based reliability analysis software provide a programming environment for defining functions that approximate random variables for probability distributions. The sources of error and the input patterns in combinational circuits are random quantities and are thus modeled by these functions and the system is analyzed using computer simulation techniques [7], such as the Monte Carlo Method [19], 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 expectation 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 coupled with the usage of computer arithmetic, the reliability analysis results attained by the simulation approach can never be termed as 100% accurate.

The accuracy of reliability analysis results has become imperative these days because of the extensive usage of hardware systems in safety critical areas, like medicine, military and transportation, where an erroneous analysis could even result in the loss of human lives. Formal methods are capable of conducting accurate system analysis and thus overcome the above mentioned limitations of simulation [10]. The main principle behind formal 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 [2] and higher-order-logic theorem

proving [8]. 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 verification approach that allows us to mathematically reason about system properties by representing the behavior of a system in higher-order logic.

We believe that due to the recent developments in the formalization of probability theory concepts in higher-order-logic [17, 14], we are now at the stage where we can handle the reliability analysis of a variety of combinational circuits in a higher-order-logic theorem prover with reasonable amount of modeling and verification efforts. The main motivation of using a higher-order-logic theorem prover for this purpose is the ability to formally analyze a broader range of combinational circuits and reliability properties by leveraging upon the high expressiveness of the underlying logic. But, this option involves two main challenges. The first one is that we need a foundational infrastructure to be able to formally specify and reason about the reliability of erroneous behavior of logical gates, which is unpredictable in nature, in logical terms. Whereas, the second one is related to the inherent nature of the higher-order-logic theorem proving, i.e., the tedious user efforts involved in interactively reasoning about the reliability properties of the system in hand. The second point mentioned here is one of the major limitations associated with the theorem proving approach and is the biggest reason why theorem proving has not been widely accepted as a verification tool in the industry.

This paper tackles both of the above mentioned challenges and, to the best of our knowledge, presents the first automatic theorem proving based approach for the reliability analysis of combinational circuits. The proposed approach is primarily inspired by the probabilistic gate models (PGM) method [11, 12, 25]. The main idea behind this approach is to formally represent the erroneous behavior of all the basic logical gates (AND, OR, NOT etc.) in terms of the probabilities of obtaining $True$ or a logical 1 at their inputs. These expressions, also referred to as the von-Neumann error models for combinational gates, can then be used to evaluate the reliability of a combinational circuit that is essentially a structure composed of the basic logical gates. We have also developed a C++ program that translates a combinational circuit, expressed in the hardware description language VHDL, to its corresponding logical description, writes the reliability theorem and generates its proof script, based on a rich library of formally verified theorems corresponding to the PGMs, that we have developed in this work. This kind

of a setting makes the approach automatic, which is an attractive feature for the microelectronic design engineers, who are usually not comfortable in working with pure formal verification based approaches or logical reasoning.

The definitions and theorems, related to the von-Neumann error models for the basic gates and the generic reliability expression of a combinational circuit, exhibit random and probabilistic behaviors, due to the random nature of gate-faults. Therefore, they have been formally defined by building upon the methodology for higher-order-logic formalization of probabilistic algorithms given in [17]. Since this formalization has been done using the HOL theorem prover [9], the proposed work has also been done using HOL.

To illustrate the practical effectiveness and demonstrate the utilization of the proposed framework, we use it to assess the reliabilities of a comparator, a full adder and five benchmarks, i.e., LGSynth'91-C17, LGSynth'91-Majority, LGSynth'91-Parity, ISCAS-85-74283 (4-bit Adder) and ISCAS-85-C6288 (16x16 multiplier). The comparator is a simple combinational circuit and is used to illustrate the working of the proposed automated framework. The simulation based PGM approach [12] was used to assess the reliability of the full adder circuit and therefore we assess its reliability using the proposed approach to highlight the accuracy of our results. The four benchmarks LGSynth'91-C17, LGSynth'91-Majority, LGSynth'91-Parity, and ISCAS-85-74283 have been analyzed to demonstrate the applicability of the approach to analyze real-world problems. We report some statistics, like the size of the circuit and the analysis time, for these benchmarks. Finally, the ISCAS-85-C6288 benchmark has been picked up due to its comparatively larger size, i.e., approximately 2400 gates. Instead of modeling this circuit at the gate level, we illustrate how the proposed approach can be used to model the given circuit using full adder cells, which significantly reduces the size of the model and thus demonstrates the scalability of the proposed approach towards hierarchical designs.

We have already presented some of the ideas and formalization related to the formalization of von-Neumann error models for the basic gates in a workshop [15]. The current paper is an extension to that work as we present further formalization details. Likewise, the idea of automatically reasoning about the reliability of combinational circuits is novel. The extensive case studies, presented in the current paper, is also one of the major extensions to our first paper [15] in the area.

The rest of the paper is organized as follows. Section 2 provides a review of the related work. In Section 3, we present a brief overview of the HOL the-

orem prover and the theorem proving based probabilistic analysis. Sections 4 and 5 present the description of the core formalizations of this paper that allow us to automatically conduct the reliability analysis of combinational circuits, i.e., the formalization of von-Neumann models for the basic logical gates and the generic reliability expression, respectively. Then, we illustrate the proposed framework using the comparator circuit example in Section 6. The case studies are given in Section 7. Finally, Section 8 concludes the paper.

2. Related Work

A number of reliability analysis approaches for combinational circuits have been recently proposed that tend to somewhat meet the accuracy and scalability challenges. The first worth mentioning approach is based on representing the erroneous behavior of a gate as a matrix, referred to as the probabilistic transfer matrix (PTM) [18]. Depending on the interconnections of the gates, their PTMs are then utilized to attain the erroneous behavior of the whole circuit as a relatively large PTM by performing matrix operations like dot or tensor products [18]. The PTM of the whole circuit can then be used with the input and output probabilities of the combinational circuit to compute its reliability. Since the PTM evaluation is based on the exhaustive listing of all input and output probabilities, a circuit with i inputs and j outputs is represented by a PTM with $2^{(i+j)}$ entries. Thus, as the circuits grow bigger in size, their PTMs require a significant amount of memory for storage and computational time for their reliability evaluation. Algebraic decision diagrams have been utilized to minimize these requirements but still the scalability remains a big issue for the PTM based approach. A similar approach, but a lot more efficient in terms of space and time complexity than the PTM based approach, has been proposed in [11, 12, 25] that calls for developing von-Neumann models, called the probabilistic gate models (PGMs), for unreliable logic gates. These models are used to analytically analyze the reliability for a single output and an input pattern. Such a capability has been found to be particularly useful for the reliability modeling of certain critical paths in a circuit.

Both of the above mentioned techniques have been utilized to analyze the reliability of many frequently used and some benchmark combinational circuits. Thus, as far as conducting the analysis of the large combinational circuits is concerned, these techniques are quite efficient but in terms of the

accuracy of the results, the analysis cannot be termed as 100% accurate. The main reason behind that is the fact that the analysis in these approaches is primarily based either on paper-and-pencil proof methods or simulation [7]. The paper-and-pencil proof methods have always some risk of an erroneous analysis due to the lengthy nature of computations involved in the case of conducting reliability analysis of present age combinational circuits coupled with the human-error factor. Whereas, due to the reasons mentioned in the previous section, the results from computer simulations cannot be termed as accurate. The proposed higher-order-logic theorem proving approach is primarily based on the PGM method but due to its inherent soundness overcomes the inaccuracy limitation of the simulation based PGM approach. We also provide the reliability results automatically, just like the simulation approach, and a variety of circuits can be analyzed as will be illustrated in Section 7 of this paper. As far as the scalability of the approach is concerned, the proposed is very similar to the simulation based PGM method.

Given the dire need of accuracy in the area of reliability analysis of combinational circuits, probabilistic model checking [1, 24], which enables analyzing systems with random or unpredictable behaviors, has been recently used in this domain as well [3, 4]. More specifically, reliability-redundancy trade-offs for NAND multiplexing have been evaluated and the reliability of some fixed bit adders has been assessed. Due to the inherent nature of model checking, the worst case space and time complexity for the reliability analysis of a combinational circuit with i inputs and j inputs is $O(2^{(i+j)})$. This limits the applicability of probabilistic model checking approach for such an analysis due to its well-known state-space explosion problem [6]. Similarly, to the best of our knowledge, it has not been possible to precisely reason about most of the commonly used reliability related statistical quantities, such as averages and variances, using probabilistic model checking so far. The proposed approach tends to overcome the limitations of probabilistic model checking as well in the reliability analysis of combinational circuits domain. First of all it is not a state-based approach and thus does not suffer from the associated problems like the state-space explosion. Secondly, due to the high expressiveness of higher-order logic, it can be used to reason about any property, including the statistical ones, that can be expressed in a closed mathematical form.

The foremost criteria for implementing a theorem proving based reliability analysis framework is to be able to formalize and verify random variables in higher-order logic. Hurd's PhD thesis [17] can be considered a pioneering

work in this regard as it presents a methodology for the formalization and 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 [17]. In fact, some of the commonly used discrete random variables along with their verification, based on the corresponding Probability Mass Function (PMF) properties has been presented in [17]. Building upon Hurd's formalization [17], verification of the sampling algorithms of a few continuous random variables based on their Cumulative Distribution Function (CDF) properties has been reported as well [14]. For comparison purposes, it is frequently desirable to summarize the characteristic of the distribution of a random variable by a single number, such as its expectation or variance, rather than an entire function. For example, it is easier to compare the reliability of two implementations of the full adder circuit based on the expected values of their radiabilities rather than the probabilities of their failures. Hurd's formalization was extended with a formal definition of expectation in [14]. This definition is then utilized to formalize and verify the expectation and variance characteristics associated with discrete random variables that attain values in positive integers only. All of this formalization can play a pioneering role in the proposed theorem proving based reliability analysis framework. In particular, we built upon Hurd's approach [17] to interactively reason about the reliability properties of combinational circuits in [15].

3. Preliminaries

In this section, we give a brief introduction to the HOL theorem prover and present an overview of conducting probabilistic analysis using the HOL theorem proving. The intent is to introduce the main ideas along with some notation that is going to be used in the rest of the paper.

3.1. HOL Theorem Prover

HOL is an interactive theorem prover which is capable of conducting proofs in higher-order logic. It utilizes the simple type theory of Church [5] along with Hindley-Milner polymorphism [21] to implement higher-order logic. HOL has been successfully used as a verification framework for both software and hardware as well as a platform for the formalization of pure mathematics.

In order to ensure secure theorem proving, the logic in the HOL system is represented in the strongly-typed functional programming language ML [23]. An ML abstract data type is used to represent higher-order-logic theorems and the only way to interact with the theorem prover is by executing ML procedures that operate on values of these data types. The HOL core consists of only 5 basic axioms and 8 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 HOL theorem prover includes many proof assistants and automatic proof procedures [13] to assist the user in directing the proof. The user interacts with a proof editor and provides it with the necessary tactics to prove goals while some of the proof steps are solved automatically by the automatic proof procedures.

In order to facilitate reutilization of verified theorems, HOL allows its users to store a collection of valid HOL types, constants, axioms and theorems as a HOL theory file in computers. Once stored, HOL theories can be loaded in the HOL system and the corresponding definitions and theorems can be utilized right away. Thus, HOL theories allow us to build upon existing results in an efficient way without going through the tedious process of regenerating these results using the basic axioms and primitive inference rules. Various mathematical concepts have been formalized and saved as HOL theories by the HOL users. Out of this useful library of HOL theories, we utilized the theories of Booleans, lists, sets, positive integers, *real* numbers, measure and probability in this paper. In fact, one of the primary motivations of selecting the HOL theorem prover for our work was to benefit from these built-in mathematical theories.

Table 1 provides the mathematical interpretations of some frequently used HOL symbols and functions, which are inherited from existing HOL theories and will be used in this paper.

3.2. Probabilistic Analysis in HOL

The foremost criteria for implementing a theorem proving based reliability analysis framework is to be able to formalize random variables in higher-order logic and verify their probabilistic properties. Random variables are fundamentally probabilistic functions that can be modeled in higher-order logic as deterministic functions with access to an infinite Boolean sequence $\mathbb{B}^\infty$; a source of infinite random bits [17]. These deterministic functions make random choices based on the result of popping the top most bit in the infinite

HOL Symbol	Standard Symbol	Meaning
$\wedge$	*and*	Logical *and*
$\vee$	*or*	Logical *or*
$\neg$	*not*	Logical *negation*
::	*cons*	Adds a new element to a list
++	*append*	Joins two lists together
`hd L`	*head*	Head element of list L
`tl L`	*tail*	Tail of list L
`mem a L`	*member*	True if a is a member of list L
(a, b)	a x b	A pair of two elements
`fst`	fst (a, b) = a	First component of a pair
`snd`	snd (a, b) = b	Second component of a pair
λx.t	$\lambda x.t$	Function that maps x to $t(x)$
{x|P(x)}	$\{x \| P(x)\}$	Set of all x such that $P(x)$

Table 1: HOL Symbols and Functions

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 programs. Thus, a random variable which takes a parameter of type α and ranges over values of type β can be represented by the function.

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

Consider the following formalization of the Bernoulli($\frac{1}{2}$) random variable that returns 1 or 0 with equal probability $\frac{1}{2}$:

```
bit = (λs.if shd s then 1 else 0,stl s)
```

where s is the infinite Boolean sequence and `shd` and `stl` are the sequence equivalents of the list operation *'head'* and *'tail'*. The probabilistic programs can also be expressed in the more general state-transforming monad where the states are the infinite Boolean sequences.

```
∀ a s.  unit a s = (a,s)
∀ f g s.  bind f g s = g (fst (f s)) (snd (f s))
```

The HOL functions `fst` and `snd` return the first and second components of their argument, which is 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

```
bit_monad = bind sdest (λb.  if b then unit 1 else unit 0)
```

where `sdest` gives the head and tail of a sequence as a pair (*shd* `s`, *stl* `s`) and (λ`x.t`) denotes the lambda abstraction function in HOL that maps its argument x to $t(x)$.

Hurd [17] also presents some formalization of the 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 prove probabilistic properties for random variables such as the following Probability Mass Function (PMF) property can be verified for the function `bit`.

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

where the function `fst` selects the first component of a pair and $\{x|C(x)\}$ represents a set of all x that satisfy the condition C in HOL.

The measurability and independence of a probabilistic function are important concepts in probability theory. A property `indep`, called *strong function independence*, is introduced in [17] such that if $f \in$ `indep`, then f will be both measurable and independent. It has been shown in [17] that a function is guaranteed to preserve *strong function independence*, if it accesses the infinite Boolean sequence using only the `unit`, `bind` and `sdest` primitives. All reasonable probabilistic programs preserve *strong function independence*, and these extra properties are a great aid to verification.

The above approach has been successfully used to formalize most of the commonly used random variables and verify them based on their corresponding probability distribution properties. In this paper, we utilize the model for the Bernoulli random variable, formalized as the function `bern_rv`, and verified using the following PMF relation [17]:

Lemma 1: *PMF of Bernoulli(p) Random Variable*
```
∀ p.  0 ≤ p ∧ p ≤ 1 ⇒ ℙ {s | fst (bern_rv p s)} = p
```

The function `bern_rv` for the Bernoulli(p) random variable models an experiment with two outcomes; *True* and *False*, whereas p represents the probability of obtaining a *True*.

4. Formalization of a Faulty Gate

The foremost step in the proposed theorem proving based reliability analysis framework is to formally express the behavior of a faulty gate or component using the von-Neumann model [11].

Definition 1: *von-Neumann Faulty Component*
```
(rv_list [] = (unit [])) ∧
∀ h t.  (rv_list (h::t) =
  bind h (λx.  bind (rv_list t) (λy.  unit (x::y))))

∀ f P e.  faulty_comp f P e =
  bind (bern_rv e) (λx.  bind (rv_list P)
    (λy.  unit (if x then ¬(f y) else (f y))))
```

The first function `rv_list` accepts a list of random variables and returns the corresponding list of the same random variables such that the outcome of each one of these random variables is independent of the outcomes of all the others. This is done by recursively using the remaining portion of the infinite Boolean sequence of each random variable to model its subsequent random variable in the list using the monadic functions `unit` and `bind`. The second function `faulty_comp` accepts three variables. The first one is a function `f` that represents the Boolean logic functionality of the given component with data type *bool list* $\rightarrow$ *bool*, where the *bool list* represents the list of Boolean values corresponding to the inputs of the component and the return type *bool* corresponds to the output of the component. The second input to the function `faulty_comp` is a list of Boolean random variables `P`, which corresponds to the values available at the input of the component. Whereas, the third input is the probability `e` of error occurrence in the component. The function `faulty_comp` returns a Boolean value corresponding to the output of the component with parameters `f` and `e`, when its inputs are modeled by calling the random variables in the list of random variables `P` independently.

It is important to note here that the output of such a faulty component is an unpredictable quantity, which is dependant on the error probability `e` and the input random variable list `P`. Therefore, this function is formally modeled using the approach explained in Section 3. Another point worth mentioning here is that we have used the function `bern_rv` for the Bernoulli random variable to model the random behavior associated with the error occurrence in the component. Therefore, the function `faulty_comp` basically models the erroneous behavior of a component based on the von-Neumann model [11], which assumes that the component flips its output with a probability `e`, given that the input and output lines function correctly.

Next, we verify a general expression for the probability of obtaining a $True$ or a logical 1 at the output of the von-Neumann model of a component, which is very closely related to the gate reliability in the PGM approach [11].

Theorem 1: *Probability of a True output in a Faulty Component*

```
⊢ ∀ e f P. (0 ≤ e) ∧ (e ≤ 1) ∧
  (∀x.  mem x P ⇒ x ∈ indep) ⇒
ℙ {s | fst (faulty_comp f P e s)} =
    e (1 - ℙ {s |f (fst (rv_list P s))}) +
    (1 - e) (ℙ {s | f (fst (rv_list P s))})
```

The theorem is verified under the assumption that the error probability of the component `e` is bounded in the closed interval $[0, 1]$ and every member of the random variable list `P` is measurable, i.e., $\in$ `indep`. The right-hand-side (RHS) of the theorem represents the given probability in terms of the probability of obtaining a $True$ from an error-free component, which is much easier to reason about. The HOL proof of this theorem is based on the independence of the error occurrence, the PMF of the Bernoulli random variable, given in Lemma 1, and some basic probability axioms.

Theorem 1 can now be used to formally reason about the probability of obtaining a logical 1 from any logical gate that may exhibit a faulty behavior. In order to be able to automatically reason about the reliability of digital circuits, we now verify this probability for some of the commonly used gates. Note that all of these theorems are verified under the assumption that `e` lies in the interval $[0, 1]$ and all the concerned random variables are measurable, i.e., they $\in$ `indep`.

4.1. AND and NAND Gates

For the sake of generality, first consider an N-input AND-gate. Its Boolean functionality can be formally defined as follows:

Definition 2: *N-Bit AND Gate*

```
and_gate [] = True ∧
∀ h t.  and_gate (h::t) = h ∧ (and_gate t)
```

The function `and_gate` accepts a list of Boolean values and recursively returns the logical conjunction of these values. The theorem corresponding to the probability of obtaining a *True* from this component can be expressed as follows:

Theorem 2 *Probability of True in a N-Bit AND Gate*

```
⊢ ∀ e P. (0 ≤ e ≤ 1)∧
  (∀x.  mem x P ⇒ x ∈ indep) ⇒
ℙ {s | fst (faulty_comp and_gate P e s)} =
    e (1 - prob_rv_list_mul P) +
    (1 - e) (prob_rv_list_mul P)
```

The function `prob_rv_list_mul` recursively returns the multiplication of the probabilities of each random variable being equal to *True* in the given list of random variables as follows:

```
(prob_rv_list_mul [] = 1) ∧
∀ h t.  (prob_rv_list_mul (h::t) =
  ℙ {s | fst (h s)} * (prob_rv_list_mul t))
```

The proof of Theorem 2 is based on Theorem 1 along with the fact that the probability of obtaining a logical 1 at the output of an error-free AND-gate is equal to the product of the probabilities of obtaining all logical 1's at its inputs. The result of Theorem 2 is generic and can be specialized for any AND-gate with a specific number of inputs. For example, the theorem for a 2 input AND-gate is as follows:

Theorem 3: *Probability of True in a 2-Bit AND Gate*

```
⊢ ∀ x1 x2 e.  (0 ≤ e ≤ 1) ∧
  (x1 ∈ indep) ∧ (x2 ∈ indep) ⇒
ℙ {s | fst (faulty_comp and_gate [x1;x2] e s)} =
    (ℙ {s | fst (x1 s)})(ℙ {s | fst (x2 s)}) +
    e (1 - 2(ℙ{s | fst (x1 s)})(ℙ{s | fst (x2 s)}))
```

where x1 and x2 are Boolean random variables and [x1;x2] is a list containing these two random variables, which represent the inputs of the 2-input AND-gate. Theorem 3 allows us to evaluate the probability of obtaining a logical 1 at the output of a 2-input AND-gate, which may contain an error, if we know the probabilities of obtaining a logical 1 at both of its inputs individually.

The NAND function is basically the complement of AND and thus an N-input NAND-gate can be modeled as follows:

Definition 3: *N-Bit NAND Gate*

```
∀ l.  nand_gate l = ¬(and_gate l)
```

The types of the function nand_gate are the same as the ones of and_gate. The theorem for the probability of obtaining a $True$ from the NAND gate can be expressed as follows:

Theorem 4 *Probability of True in a N-Bit NAND Gate*

```
⊢ ∀ e P. (0 ≤ e ≤ 1)∧
  (∀x.  mem x P ⇒ x ∈ indep) ⇒
ℙ {s | fst (faulty_comp nand_gate P e s)} =
    e (prob_rv_list_mul P) +
    (1 - e) (1 - prob_rv_list_mul P)
```

The proof of Theorem 4 is based on Theorem 2 along with the complement law of probability ($\mathbb{P}(\bar{A}) = 1 - \mathbb{P}(A)$). Specializing the result of Theorem 4 for the case of two inputs we get

Theorem 5: *Probability of True in a 2-Bit NAND Gate*

```
⊢ ∀ x1 x2 e.  (0 ≤ e ≤ 1) ∧
  (x1 ∈ indep) ∧ (x2 ∈ indep) ⇒
ℙ {s | fst (faulty_comp nand_gate [x1;x2] e s)} =
    (1 - e) +
    (2e - 1) (ℙ {s | fst (x1 s)})(ℙ {s | fst (x2 s)})
```

4.2. OR and NOR Gates

Besides the AND and NAND gates, other widely used logical gates are the OR and NOR gates. Following the approach for the formalization of AND gate, the OR gate can be modeled as the following recursive function.

Definition 4: *N-Bit OR Gate*

```
or_g [] = False ∧
∀ h t.  or_gate (h::t) = h ∨ (or_gate t)
```

Similarly, the NOR gate can be modeled as the complement of the OR gate as follows:

Definition 5: *N-Bit NOR Gate*

```
∀ l.  nor_gate l = ¬(or_gate l)
```

The theorem for the probability of obtaining a *True* from the NOR gate can be expressed as follows:

Theorem 6: *Probability of True in a N-Bit NOR Gate*

```
⊢ ∀ e P. (0 ≤ e ≤ 1) ∧
  (∀x.  mem x P ⇒ x ∈ indep) ⇒
ℙ {s | fst (faulty_comp nor_gate P e s)} =
    e (1 - prob_neg_rv_list_mul P) +
    (1 - e) (prob_neg_rv_list_mul P)
```

Where the function `prob_neg_rv_list_mul` recursively returns the multiplication of the probabilities of each random variable being equal to *False* in the given list of random variables as follows:

```
(prob_neg_rv_list_mul [] = 1) ∧
∀ h t.  (prob_neg_rv_list_mul (h::t) =
  ℙ {s | ¬ fst (h s)} (prob_neg_rv_list_mul t))
```

The proof of Theorem 6 is done using induction on the variable `P` and it involves reasoning based on Theorem 1 along with the independence of the random variables involved. Now, Theorem 6 can be used to verify the relation for the probability of obtaining a *True* output for the n-bit OR gate using Definition 5 and the complement law of probability ($\mathbb{P}(\bar{A}) = 1 - \mathbb{P}(A)$).

Theorem 7: *Probability of True in a N-Bit OR Gate*

```
⊢ ∀ e P. (0 ≤ e ≤ 1) ∧
  (∀x.  mem x P ⇒ x ∈ indep) ⇒
ℙ {s | fst (faulty_comp or_gate P e s)} =
    1 - (e (1 - prob_neg_rv_list_mul P) +
    (1 - e) (prob_neg_rv_list_mul P))
```

Based on Theorems 6 and 7, the corresponding theorems for OR and NOR gates with any specified number of inputs can also be verified in a very straightforward way. For example, we verified the theorem for the 2-input NOR gate as follows:

Theorem 8: *Probability of True in a 2-Bit NOR Gate*

```
⊢ ∀ x1 x2 e.  (0 ≤ e ≤ 1) ∧
  (x1 ∈ indep) ∧ (x2 ∈ indep) ⇒
ℙ {s | fst (faulty_comp nor_gate [x1;x2] e s)} =
    (1 - (ℙ {s | fst (x1 s)}) - (ℙ {s | fst (x2 s)}) +
    (1 - 2 e) (ℙ {s | fst (x1 s)}) (ℙ {s | fst (x2 s)}) +
    e (2 (ℙ {s | fst (x1 s)}) + 2 (ℙ {s | fst (x2 s)}) - 1))
```

4.3. Interconnect and the NOT Gate

For reliability assessment of a combinational circuit, we treat the interconnect between two logical gates as a gate as well. This way, we can include the effect of interconnect failures in our reliability evaluations. Ideally, the job of the interconnect is to pass an incoming signal as-is to the output and thus its functionality can be formally modeled as follows:

Definition 6: *Interconnect*

```
∀ l.  xconnect l = hd l
```

The function `xconnect` accepts a list of Boolean inputs, in order to be consistent with our other logical gate models, and returns the head of that list.

The inverter, or the NOT gate, is very similar to our interconnect model except the fact that it provides the logical negation of the input signal. This functionality can be formally modeled in terms of the interconnect functionality as follows:

Definition 7: *NOT Gate*

```
∀ l.  not_gate l = ¬(xconnect l)
```

Like the previous gate models, we are interested in the expression for the probability of getting a logical 1 at the output. The corresponding theorem for the Interconnect is as follows:

Theorem 9: *Probability of True in the Interconnect*

```
⊢ ∀ x1 e.  (0 ≤ e ≤ 1) ∧
  (x1 ∈ indep) ⇒
ℙ {s | fst (faulty_comp xconnect [x1] e s)} =
    (ℙ {s | fst (x1 s)}) +
    e (1 - 2 (ℙ {s | fst (x1 s)}))
```

The proof is based on Theorem 1 along with the definition of the function `xconnect`. Similarly, using the above theorem, the definition of the NOT gate along with the complement law of probability, we verified the following theorem for the NOT gate.

Theorem 10: *Probability of True in the NOT Gate*

```
⊢ ∀ x1 e.  (0 ≤ e ≤ 1) ∧
  (x1 ∈ indep) ⇒
ℙ {s | fst (faulty_comp not_gate [x1] e s)} =
    1 - (ℙ {s | fst (x1 s)}) -
    e + 2 e (ℙ {s | fst (x1 s)})
```

4.4. Exclusive-OR (XOR) Gate

The XOR gate is also considered to be a basic logic gate and is very frequently used in a variety of combinational circuits. The functionality of a 2-input XOR gate can be expressed using our formalization approach as follows:

Definition 8: *2-Input XOR Gate*

```
∀ l.  xor_gate l = (hd l) ∧ ¬(hd (tl l)) ∨
                   ¬(hd l) ∧ (hd (tl l))
```

The function `xor_gate` accepts a list of Boolean values and uses the list operations *head*(*hd*) and *tail*(*tl*) to access its two top most elements to compute the respective output. Based on this function, we verified the expression for the probability of getting a *True* at the output of a faulty XOR gate.

Theorem 11: *Probability of True in a 2-Bit XOR Gate*

```
⊢ ∀ x1 x2 e.  (0 ≤ e ≤ 1) ∧
  (x1 ∈ indep) ∧ (x2 ∈ indep) ⇒
ℙ {s | fst (faulty_comp nor_gate [x1;x2] e s)} =
    (ℙ {s | fst (x1 s)}) + (ℙ {s | fst (x2 s)}) -
```

```
  2 (ℙ {s | fst (x1 s)}) (ℙ {s | fst (x2 s)}) +
  e (4 (ℙ {s | fst (x1 s)}) (ℙ {s | fst (x2 s)}) -
    2 (ℙ {s | fst (x1 s)}) - 2 (ℙ {s | fst (x2 s)}) + 1)
```

The proof is based on the definition of the XOR gate, Theorem 1, the independence of error random variables and some basic probability axioms.

4.5. Majority Gate

Majority gate returns a *True* if and only if more than half of its inputs are *True*. Besides the basic logic gates covered so far, we also present a formalization of a 3-input Majority gate along with the verification of its probability for obtaining a logical 1 at the output under erroneous conditions. The main motivation for this choice is to be able to assess the reliability of a full-adder circuit that is based on the majority gate and has been analyzed previously using the PGM approach. We present the formal reliability of this full-adder circuit in Section 7 of this paper and then compare our results with the ones obtained using the informal PGM approach.

Instead of modeling the majority gate using its behavioral description, we model it using its logical model as follows:

Definition 9: *3-Input Majority Gate*

```
∀ l.  majority_gate l =
    ((hd l) ∧ (hd (tl l)) ∧ ¬(hd (tl (tl l)))) ∨
    ((hd l) ∧ (hd (tl (tl l))) ∧ ¬(hd (tl l))) ∨
    ((hd (tl l)) ∧ (hd (tl (tl l))) ∧ ¬(hd l)) ∨
    ((hd l) ∧ (hd (tl l)) ∧ (hd (tl (tl l))))
```

The function `majority_gate` accepts a list of Boolean values and uses the list operations *head*(*hd*) and *tail*(*tl*) to access its three top most elements to compute the respective output. Based on this function, we verified the expression for the probability of getting a *True* at the output of a faulty majority gate in HOL as follows:

Theorem 12: *Probability of True in a 3-Bit Majority Gate*

```
⊢ ∀ x1 x2 x3 e.  (0 ≤ e ≤ 1) ∧
  (x1 ∈ indep) ∧ (x2 ∈ indep) ∧ (x3 ∈ indep) ⇒
ℙ {s | fst (faulty_comp majority_gate [x1;x2;x3] e s)} =
    (ℙ {s | fst (x1 s)}) (ℙ {s | fst (x2 s)}) +
```

```
(ℙ {s | fst (x1 s)}) (ℙ {s | fst (x3 s)}) +
(ℙ {s | fst (x2 s)}) (ℙ {s | fst (x3 s)}) -
2 (ℙ {s | fst (x1 s)}) (ℙ {s | fst (x2 s)})
     (ℙ {s | fst (x3 s)}) +
e ( 4 (ℙ {s | fst (x1 s)}) (ℙ {s | fst (x2 s)})
     (ℙ {s | fst (x3 s)}) -
  2 (ℙ {s | fst (x1 s)}) (ℙ {s | fst (x2 s)}) -
  2 (ℙ {s | fst (x1 s)}) (ℙ {s | fst (x3 s)}) -
  2 (ℙ {s | fst (x2 s)}) (ℙ {s | fst (x3 s)}) + 1)
```

Like other similar theorems considered so far, the proof of Theorem 12 is based on the definition of the majority gate, Theorem 1, the independence of error random variables and some basic probability theory axioms.

The theorems corresponding to the probability of obtaining a logical 1 from the faulty models of basic logic gates, verified in this section, are summarized in Table 2. In this table, the probability of an input x_i being equal to 1, i.e., ℙ{s|`fst`(`xi s`)}, is represented as X_i. These results are exactly the same as the relations verified using paper-and-pencil proof methods in [11], which reassures us of the correctness of our definitions. It is important to note that the left-hand-side (LHS) of all these theorems is a function of the probability of obtaining a logical 1 at each input of the corresponding gates and the error probability e. This means that the probability of obtaining a logical 1 for a combinational circuit with erroneous gates can be evaluated based on these theorems by simple rewriting. This fact plays a vital role in the automatic reasoning about the reliability of combinational circuits as will be seen in the coming sections.

5. Formalization of Combinational Circuit Reliability

The next step in the proposed reliability analysis framework is to formally define the reliability of a combinational circuit. Reliability of a system or component is defined as the probability that it performs its intended function. Based on this definition, the reliability for a logical gate or a combinational circuit can be represented as the probability that it produces the error free result [11]. This can be formally expressed using the function `faulty_comp`, given in Definition 1, as follows:

Gate	Theorem
2-input AND Gate	$\mathbb{P}\{s \mid fst(faulty_comp\ and_gate[X_1; X_2]e\ s)\}$ $= X_1X_2 + e(1 - 2X_1X_2)$
2-input NAND Gate	$\mathbb{P}\{s \mid fst(faulty_comp\ nand_gate[X_1; X_2]e\ s)\}$ $= (1 - e) + (2e - 1)X_1X_2$
2-input NOR Gate	$\mathbb{P}\{s \mid fst(faulty_comp\ nor_gate[X_1; X_2]e\ s)\}$ $= 1 - X_2 - X_1 + X_1X_2(1 - 2e) + e(2X_1 + 2X_2 - 1)$
Interconnect	$\mathbb{P}\{s \mid fst(faulty_comp\ xconnect[X]e\ s)\} = X + e(1 - 2X)$
NOT Gate	$\mathbb{P}\{s \mid fst(faulty_comp\ not_gate[X]e\ s)\} = 1 - X - e + 2eX$
2-input XOR Gate	$\mathbb{P}\{s \mid fst(faulty_comp\ xor_gate[X_1; X_2]e\ s)\}$ $= X_2 + X_1 - 2X_1X_2 + e(4X_1X_2 - 2X_2 - 2X_1 + 1)$
3-input Majority Gate	$\mathbb{P}\{s \mid fst(faulty_comp\ majority_gate[X_1; X_2; X_3]e\ s)\}$ $= X_1X_2 + X_1X_3 + X_2X_3 - 2X_1X_2X_3 +$ $e(4X_1X_2X_3 - 2X_1X_2 - 2X_1X_3 - 2X_2X_3 + 1)$

Table 2: Probability of Output 1 for commonly used Faulty Gates

Definition 10: *Reliability*

```
∀ f L e.  reliability f L e =
  ℙ {s | fst(faulty_comp f (L e) e s) =
       fst (faulty_comp f (L 0) 0
         (snd (faulty_comp f (L e) e s)))}
```

The function `reliability` accepts three parameters, whereas, just like the function `faulty_comp`, the variables `f` and `e` represent the Boolean logic functionality of the given component and the probability of error occurrence in the component, respectively. The third variable `L` represents a function that accepts an error probability as a *real* number and returns a list of Boolean random variables with the same type as the variable `P` in the function `faulty_comp`. The LHS term in the set represents the output of the component while considering the effect of error, by using `e` as the input to the function `L`, and the RHS term represents the error free output of the given component, as the input of the function `L` is 0. This way, the function `reliability` returns the desired reliability of the component with functionality `f` and error probability `e`. It is important to note that the re-

maining portion of the infinite Boolean sequence from the LHS side term is used to model randomness in the RHS term in order to ensure probabilistic independence between them.

Building upon the above definition of reliability and using some probability theoretic reasoning, we formally verified the following alternative expression for reliability of a component. This is the same expression that has been used to assess the reliability of logical circuits in the PGM approach [25].

Theorem 13: *Alternate Expression for Reliability*

```
∀ f L e.  0 ≤ e ≤ 1 ∧
  (∀x.  mem x (L e) ⇒ x ∈ indep) ∧
  (∀x.  mem x (L 0) ⇒ x ∈ indep) ∧ ⇒
  (reliability f L e =
    ℙ {s | fst (faulty_comp f (L e) e s)}
    ℙ {s | fst (faulty_comp f (L 0) 0 s)} +
    (1 - ℙ {s | fst (faulty_comp f (P e) e s)})
    (1 - ℙ {s | fst (faulty_comp f (P 0) 0 s)}))
```

The theorem is verified under the assumptions that the error probability is bounded in the interval $[0, 1]$ and all random variables in the lists `L e` and `L 0` are measurable.

The main advantage of Theorem 13 is that it can be used to evaluate the reliability of a logical gate or a combinational circuit in terms of the probability of attaining a logical 1 at its output. This is a very useful result in terms of automatically reasoning about the reliability in a theorem prover since we have already verified the relations, given in Table 2, for finding the probability of attaining a logical 1 at the output for most of the commonly used logical gates. This infrastructure, i.e., the theorems given in Table 2 and Theorem 13, is based on the formally verified results in the HOL theorem prover and hence the results attained by formally building on top of it can be regarded as 100% accurate unlike all the available combinational logic reliability analysis approaches.

6. Proposed Reliability Analysis Framework

The above mentioned formalization can be used to reason about reliability properties of combinational circuits in an interactive way. The user input

is required in translating the circuit model to its higher-order-logic counterpart, writing the reliability theorem and finally to interactively verify it in the HOL theorem prover. In this paper, we propose a framework, illustrated in Figure 1, that allows us to tackle these tasks automatically. The proposed framework accepts the VHDL model of the combinational circuit, the output port name in the circuit, the error probability and a combination of circuit's input values for which the reliability needs to be analyzed. Whereas, it returns the accurate reliability of the given circuit under the given conditions without any user interaction. The reliability problem is first translated to its corresponding higher-order-logic proof goal by a C++ translator module. This goal is then automatically verified based on the already verified von-Neumann error models for the basic logical gates along with a formally verified generic expression for reliability of combinational circuits. It is important to note that the initial verification of the von-Neumann error models for the basic logical gates and the generic reliability expression is not automatic and is one of the main contributions of this paper. But based on these formally verified results, we can automatically reason about the reliability of any combinational circuit in a theorem prover. Since the analysis is performed in the sound core of a theorem prover, the reliability analysis results can be termed as 100% accurate.

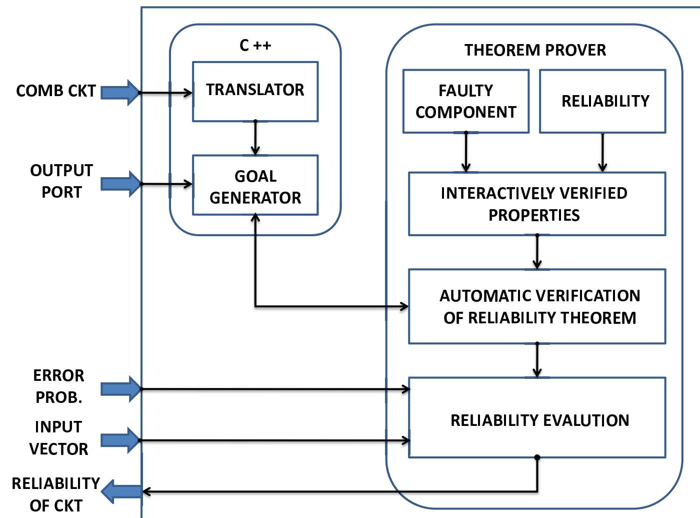

Figure 1: Proposed Reliability Analysis Framework

For illustrating the role of each module in the proposed framework, consider the example of assessing the comparator circuit given in Figure 2.

The proposed framework accepts the concurrent VHDL model of the circuit, the output node for which the circuit reliability needs to be assessed, the probability of error and an input pattern (00, 01, 10 or 11 in the case of the comparator circuit corresponding to the two inputs A and B). The framework has a built-in Translator, written in C++, that converts the concurrent VHDL model of the given circuit to its corresponding higher-order-logic description using the function `faulty_comp`, explained in Section 5. The output of the Translator in the case of analyzing the output O1 or O3 for an input pattern (`pA`,`pB`) is given below.

```
and_gate
  (λx.[(faulty_comp xconnect [bern_rv pA] x);
       (faulty_comp xconnect [(faulty_comp nand_gate
          [(faulty_comp xconnect [bern_rv pA] x);
           (faulty_comp xconnect [bern_rv pB]) x])x])x])
```

The interconnect between a primary input port and a gate or between two gates has been modeled using the function `xconnect` in the above expression. This way, we consider the reliability impact of the interconnect as well in our reliability computations. The function `and_gate` above corresponds to the AND-gate in Figure 2, the output of which is the one that we are interested in finding the reliability for. It is a two input gate and its list of random variables, which corresponds to the inputs of the gate, contains two random variables. The first input is coming from a primary port and therefore we use the Bernoulli random variable function `bern_rv` with input probability `pA` of getting a logical 1 at this input in the input random variables list. Thus, ensuring that if `pA` is 1 then the probability of getting a logical 1 at this input is 1 and if `pA` is 0 then the probability of getting a logical 1 is 0. The second input of the AND-gate is coming from a 2-input NAND-gate, for which the inputs are in turn connected to the primary ports A and B via the interconnect and these connections can be observed in the input random variable list for the function `nand_gate` in the output of the Translator.

The second C++ module, i.e., the Goal Generator given in Figure 1, utilizes the output of the Translator to first generate the following goal:

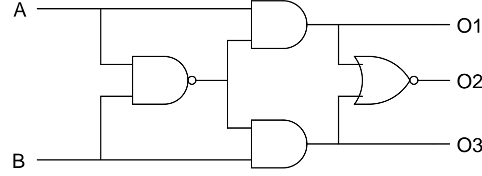

Figure 2: A 2-bit Comparator

```
∀ pA pB e.  0 ≤ e ≤ 1 ∧
  0 ≤ pA ∧ pA ≤ 1 ∧ 0 ≤ pB ∧ pB ≤ 1 ⇒
    (reliability and_gate
  (λx.[(faulty_comp xconnect [bern_rv pA] x);
       (faulty_comp xconnect [(faulty_comp nand_gate
          [(faulty_comp xconnect [bern_rv pA] x);
           (faulty_comp xconnect [bern_rv pB]) x])x])x]) e = Z)
```

The LHS of the proof goal represents the reliability of the given comparator circuit, using the function `reliability` given in Definition 10, and the RHS is set to an arbitrary real number `Z`. At this point, the goal is fed to the HOL theorem prover and is simplified using the theorems given in Table 2 and Theorem 13. Once the most simplified form is obtained, the expression is fed back to the Goal Generator module, which replaces the real number Z by the simplified expression and generates the following new proof goal:

Theorem 14: *Reliability for Comparator Output O1/O3*

```
∀ pA pB e.  (0 ≤ e ≤ 1) ∧
  (0 ≤ pA ≤ 1) ∧ (0 ≤ pB ≤ 1) ⇒
    (reliability and_gate
  (λx.[(faulty_comp xconnect [bern_rv pA] x);
       (faulty_comp xconnect [(faulty_comp nand_gate
          [(faulty_comp xconnect [bern_rv pA] x);
           (faulty_comp xconnect [bern_rv pB]) x])x])x]) e =
(pA(1 - e + (2e - 1)(pA + e(1 - 2pA))(pB + e(1 - 2pB))) + e(
1 - 2pA(1 - e + (2e - 1)(pA + e(1 - 2pA))(pB + e(1 - 2pB)))))
  (pA(1 - (pApB))) +
(1 - (pA(1 - e + (2e - 1)(pA + e(1 - 2pA))(pB + e(1 - 2pB)))+e(
1 - 2pA(1 - e + (2e - 1)(pA + e(1 - 2pA))(pB + e(1 - 2pB))))))
  (1 - pA(1 - (pApB))))
```

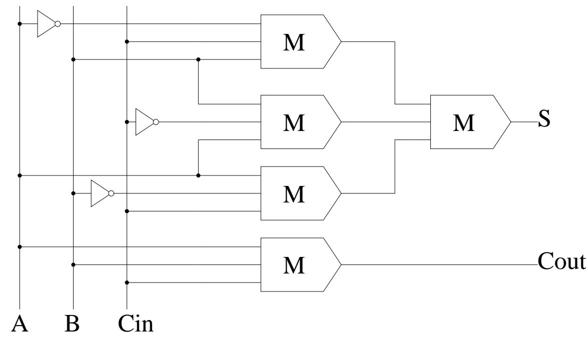

Figure 3: Majority Gate based Full Adder

The new goal is now fed to HOL and this time is automatically verified using the theorems given in Table 2 and Theorem 13. The distinguishing feature of the above theorem is its generic nature, i.e., it is true for all values of `e`, `pA` and `pB`. In other words, once this theorem is verified it can be readily used to evaluate the reliability of outputs $O1$ or $O3$ for any values of `e`, `pA` and `pB`.

Next, we illustrate the practical effectiveness of our approach by providing the reliability analysis of some interesting real-world problems.

7. Experimental Results

We first present the analysis of a full adder circuit and provide a comparison of our results with the ones obtained using the informal PGM approach. We then analyze a few benchmark circuits to illustrate the usefulness of the proposed approach. This will be followed by the analysis of the benchmark ISCAS-85-C6288, where we illustrate the scalability of the proposed approach for hierarchical designs.

7.1. Comparison with the Classical PGM Approach: Full Adder Circuit

We now assess the reliability of a majority gate based full adder, given in Figure 3. The overall reliability of the full adder can be assessed by multiplying the individual reliabilities of the two outputs since both of them are independent. Just like the comparator circuit, analyzed in the last section, a generic expression for the reliability of the full adder circuit is formally verified in HOL automatically. This generic expression is then used to obtain the reliability values for a set of different allowable values for the error

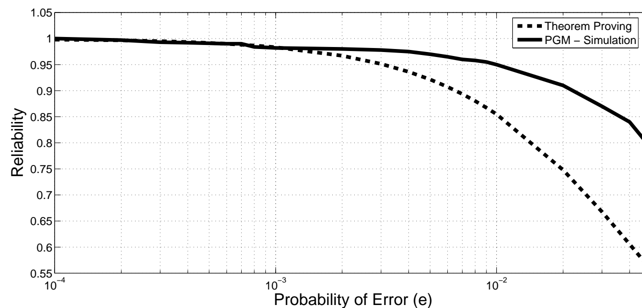

Figure 4: Reliability for Majority Gate based Full Adder

probability e and the results are summarized as the dotted line in Figure 4. Reliability analysis for the same full adder circuit was done in [12] using the simulation based PGM approach and Figure 4 also presents those results as the solid line. The reliability results of the two approached are clearly different and the difference gets more prominent as the probability of gate error e increases beyond 10^{-3}. The results obtained from the proposed approach exactly match the ones from paper-and-pencil based analysis and thus can be regarded as 100% accurate. Thus the differences in the simulation based PGM results can be attributed to the usage of approximate random variable models and the inherent nature of simulation. It is important to note that the simulation discrepancies are clearly visible even for such a small reliability analysis problem. Obviously, the scale of the discrepancies would increase as the number of gates in the circuits are increased. These results clearly indicate the importance of the proposed formal reliability analysis approach, which is capable of addressing the inaccuracies of the traditional simulation based approach and thus can prove to be quite useful for the reliability analysis of combinational circuits used in safety-critical domains.

7.2. Practical Effectiveness: Benchmark Circuits

In order to demonstrate the practical effectiveness of our approach, we now present the reliability analysis of some benchmark combinational circuits. Just like the comparator circuit, we first automatically verify generic reliability expressions for the benchmarks the LGSynth'91-C17, LGSynth'91-Majority, LGSynth'91-Parity, and ISCAS-85-74283 in HOL. These formally verified reliability expressions are then evaluated for the case when all their inputs are set to logical 1's and gate error probability e is equal to 0.05

Benchmark	Circuit Name	No. of gates	No. of inputs	No. of outputs	Name of outputs	Reliability	Time(s)
LGSynth'91	C17	6	5	2	O1	0.8788	3.66
					O2	0.6972	4.17
	majority	13	5	1	F	0.8644	230.42
	Parity	15	16	1	Q	0.5235	462.61
ISCAS'85 74X series	4-bit Adder (74283)	36	9	5	C	0.9477	27.22
					S0	0.6998	26.00
					S1	0.7075	57.02
					S2	0.7101	214.43
					S3	0.6844	240.27

Table 3: Reliability with all inputs set to $True$ and gate error $e = 0.05$

and the results are summarized in Table 3. The experiments were run on a Unix workstation with Sparc-v9 processor operating at 1015 MHz with 4096 Megabytes of memory. The successful handling of these reliability analysis problems clearly demonstrates the practical effectiveness of the proposed approach. Due to the inherent soundness of our approach, these results can be regarded as 100% accurate. This accuracy is the main motivation of the proposed approach and to the best of our knowledge, cannot be achieved by any other existing reliability analysis approach. It is also important to note that these results have been obtained automatically and no user guidance was required during this process, which is also a distinguishing feature of our approach when compared to other higher-order-logic theorem proving based analysis frameworks. Another worth mentioning point here is that the times reported against the reliability computations in Table 3 may seem a bit high for the given benchmarks. The reason for these somewhat larger times is that they include the verification time for the generic reliability theorems like Theorem 14 for the benchmark circuits. This means that once these relations have been verified, the times for evaluating reliabilities for other input combinations and/or error probabilities for the same benchmarks would be almost negligible since the same theorems can be reutilized.

7.3. Scalability for Hierarchical Circuits: ISCAS-85-C6288

ISCAS-85-C6288 is a 16x16 multiplier that is hierarchically constructed using 240 full and half adder cells. An implementation with full adder cells only is given in Figure 5. The gate count for this benchmark is approximately 2400. The formal reliability analysis approach, outlined in Figure 1, do not

scale very well to a flattened gate-level netlist of ISCAS-85-C6288 due to its high gate count and the complex interconnectivity and thus we end up having either memory problems or very long proof times. Similar problems are also encountered in the other state-of-the-art reliability analysis tools based on PTM, PGM and probabilistic model checking (PMC) approaches and thus they cannot handle the analysis of ISCAS-85-C6288 or other similar or larger sized combinational circuits. However, the proposed approach is flexible enough to cater for such hierarchical designs in a hierarchical way. The main idea is to leverage upon the independence of faults between gates and construct formal von-Neumann fault models of the frequently used modules in the hierarchical design just like we built the models of basic logic gates in Section 4. Based on this idea, we can construct formal von-Neumann models of the full-adder cells of the ISCAS-85-C6288 benchmark and then formally analyze its reliability by modeling it as a structure of 240 full adder cells. This way, we considerably reduce the size of the generic reliability expression that needs to be verified and thus the associated memory requirements, which in turn allows us to assess the reliability of the benchmark.

For illustration purposes, we now apply the idea outlined above for formally analyzing the reliability of output $P2$ of the ISCAS-85-C6288 benchmark. The reason for picking up a lower order output is to be able to express the formalization in a compact way and thus facilitate understanding of the approach. Though, the same method can be applied to assess the reliability equations for higher order output bits as well, but their expressions would obviously be much longer. The first step in this regard is to formally verify the probability for having a logical '1' at the sum and carry outputs of the faulty ISCAS-85-C6288 full adder cell, given in Figure 6. The corresponding HOL theorem for the sum output is as follows:

Theorem 15: *Probability of True in the ISCAS-85-C6288 Full Adder*

```
⊢ ∀ ac ar br ci e.  (0 ≤ e ≤ 1) ∧
  (ac ∈ indep) ∧ (ar ∈ indep) ∧
    (br ∈ indep) ∧ (ci ∈ indep) ⇒
ℙ {s | fst (faulty_comp nor_gate
          (full_adder_sum_iscas85_list ac ar br ci e) e s)} =
    1 - (1 - ℙ {s | fst (ar s)})(ℙ {s | fst (br s)}) +
    e (1 - 2 ℙ {s | fst (ar s)})(ℙ {s | fst (br s)}) -
    ...
```

The RHS of the expression has been truncated due to its relatively large

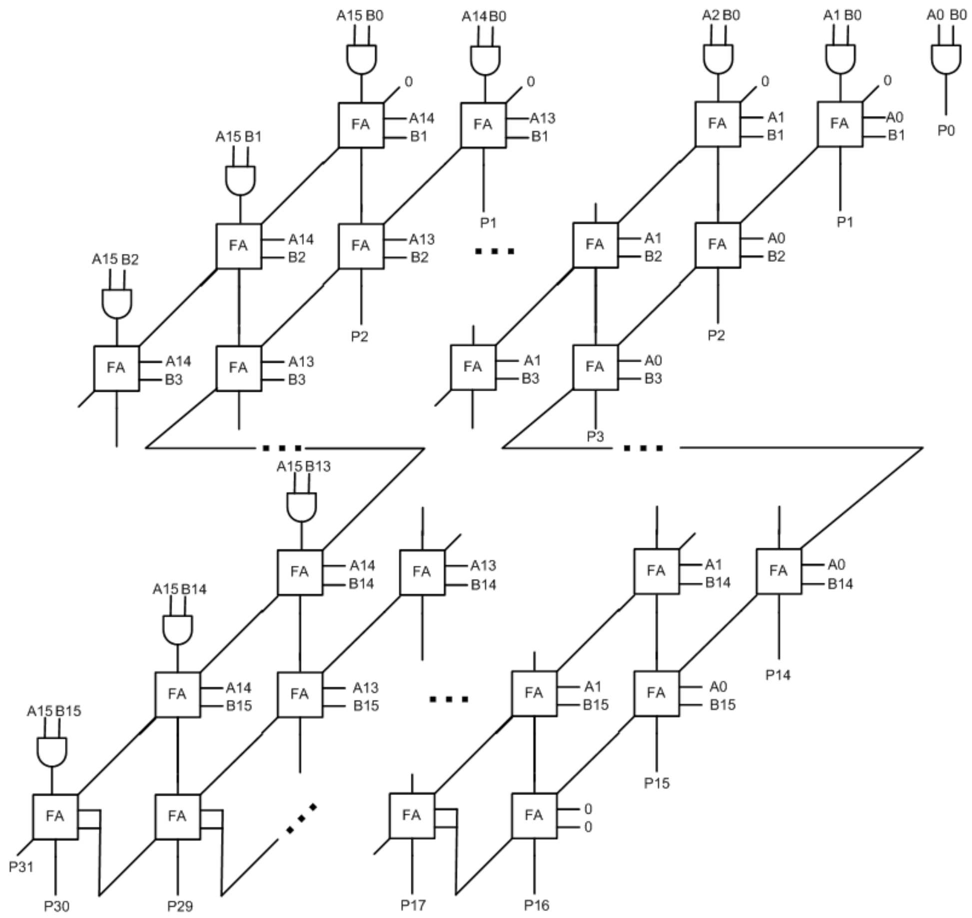

Figure 5: ISCAS-85-C6288 Benchmark

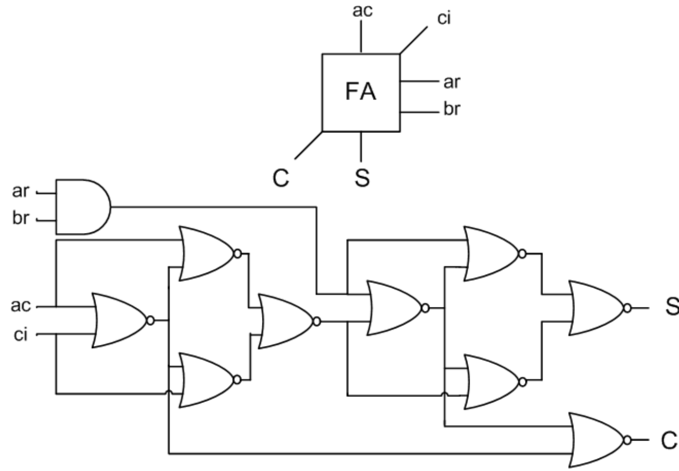

Figure 6: ISCAS-85-C6288 Full Adder Cell

size. The approach described to model the comparator circuit has been again adopted to model the full adder circuit above but in order to facilitate the reusability of the full adder cell, the details of the gates accept the last NOR gate have been represented as the function `full_adder_sum_iscas85_list`, which is defined as follows:

Definition 11: *ISCAS-85-C6288 Full Adder List*

```
∀ ac ar br ci e.  full_adder_sum_iscas85_list ac ar br ci e =
  [(faulty_comp nor_gate
   [(faulty_comp and_gate [ar; br] e); (faulty_comp nor_gate
   [(faulty_comp and_gate [ar; br] e); (faulty_comp nor_gate
    [(faulty_comp nor_gate[ac;
     (faulty_comp nor_gate [ac; ci] e)] e);
     (faulty_comp nor_gate
  [(faulty_comp nor_gate[ac; ci] e); ci] e)] e)] e)] e);
   (faulty_comp nor_gate
    [(faulty_comp nor_gate[(faulty_comp and_gate[ar; br] e);
    (faulty_comp nor_gate
     [(faulty_comp nor_gate[ac;
     (faulty_comp nor_gate[ac; ci] e)] e);
     (faulty_comp nor_gate
  [(faulty_comp nor_gate[ac; ci] e); ci] e)] e)] e) ;
```

```
(faulty_comp nor_gate
 [(faulty_comp nor_gate[ac;
  (faulty_comp nor_gate [ac; ci] e)] e);
  (faulty_comp nor_gate
 [(faulty_comp nor_gate[ac; cin] e); ci] e)] e)] e)]
```

It is important to note that we have kept the inputs as generic random variables in the above definition and theorem just like what was done in the case of basic logic gates in Section 4. This allows us to use the probability expression of Theorem 15 to assess the reliability of a circuit where the full adder is used as the first (Bernoulli random variables would be used to model the input random variables) or an intermediate cell (the output of the previous full adder cell would be to model input random variable). The above theorem and definition have been written manually as our C++ module can only handle flattened gate level netlists but we were able to verify Theorem 15 automatically using the proposed framework.

Now, the ISCAS-85-C6288 benchmark can be modeled in terms of Definition 11 and its reliability can be formally reasoned about using Theorem 15. This way, we avoid working at the gate level and hence the large definitions and theorems. The formally verified theorem corresponding to the probability of getting a logical '1' at output $P2$ of the ISCAS-85-C6288 is given below:

Theorem 16: *Probability of getting a True at output P2 of ISCAS-85-C6288*

```
⊢ ∀ pA0 pA1 pA2 pB0 pB1 pB2 e.  (0 ≤ e ≤ 1) ∧
  (0 ≤ pa0 ≤ 1) ∧ (0 ≤ pa1 ≤ 1) ∧
  (0 ≤ pa2 ≤ 1) ∧ (0 ≤ pb0 ≤ 1) ∧
  (0 ≤ pb1 ≤ 1) ∧ (0 ≤ pb2 ≤ 1) ∧ ⇒
ℙ {s | fst (faulty_comp nor_gate
  (full_adder_sum_iscas85_list
    (full_adder_sum_iscas85_list
      (faulty_comp and_gate [(bern_rv pA2); (bern_rv pB0)] e)
      (bern_rv pA1)
      (bern_rv pB1)
      (λs.  (F, s)) e) e)
    (bern_rv pA0)
    (bern_rv pB2)
```

```
  (faulty_comp nor_gate
    (full_adder_carry_iscas85_list
      (faulty_comp and_gate [(bern_rv pA1); (bern_rv pB0)] e)
      (bern_rv pA0)
      (bern_rv pB1)
      (λs.  (F, s)) e) e) e) e s} =
  1 - (1 - (pA0 pB2 + e (1 - 2 pA0 pB2)) -
  (1 - (pA0 pB2 + e (1 - 2 pA0 pB2)) -
  (1 - (1 - ℙ {s | fst ((faulty_comp nor_gate
(full_adder_sum_iscas85_list
  (full_adder_sum_iscas85_list
    (faulty_comp and_gate [(bern_rv pA2); (bern_rv pB0)] e)
    (bern_rv pA1)
    (bern_rv pB1)
    (λs.  (F, s)) e) e) s)} -
  ...
```

The RHS expression above has been again truncated due to its large size. The above theorem can now be used along with Theorem 13 to obtain a generic expression for the desired reliability, which can in turn be used to evaluate the reliability just like the previous examples. With this result, we demonstrated the scalability of the proposed approach towards analyzing the reliability of hierarchical circuits. This is another distinguishing feature of the proposed approach besides its accuracy. Though, we had to compromise the automatic nature of the analysis to achieve the above result. However, most of the user interaction was required in the formalization step as all the verification was almost done automatically.

Based on the above examples, comparing the proposed approach to the other existing approaches [11, 18], it can be observed that our approach does not rely on paper-and-pencil proof methods or simulation, which are the major sources of error in the PTM and PGM based approaches. Whereas, PMC based reliability analysis [3, 4] is based on formal methods but is severely limited by the state-space-explosion problem. For example, Table 1 in [4] reports on the state-space-explosion problem in the reliability analysis of adder circuits with more than 4 inputs using PMC. Similarly, the analysis done based on the PMC approach does not provide generic results, like the proposed theorem proving based approach does, and thus the whole analysis needs to be repeated if some parameter changes. For example, the reliability

of combinational circuits have been assessed in [4] for only a specific set of values of error probabilities. Henceforth, the scalability and the generic nature of the above examples clearly demonstrate the usefulness of the proposed approach compared to PMC.

8. Conclusions

The paper presents the first theorem proving based automatic framework for the reliability analysis of combinational circuits. Due to the formal nature of the approach, the reliability results are 100% accurate; a feature that is very useful for the analysis of combinational circuits that are used in safety-critical applications. The proposed framework is primarily based on the PGM method as we present the formalization of combinational gate faults and the notion of reliability for a combinational logic gate. Due to the undecidable nature of the underlying higher-order logic, these results had to be interactively verified in HOL. This part consumed around 120 man hours and is composed of approximately 2000 lines of HOL code. These formally verified theorems are then leveraged upon to automatically assess the reliability of combinational circuits using some C++ modules.

The paper also presents some interesting case studies to support the formal automatic reliability analysis approach and we have been able to evaluate the reliability for combinational circuits with up to 36 gates. With a little bit of user interaction, i.e., 16 man hours, we were also able to analyze a hierarchical combinational circuit of about 2400 gates by expressing the circuit at the module level, using full adders, then the gate-level. In a similar way, by compromising upon the automatic nature of the approach, we can formally verify reliabilities of very large circuits, in terms of gate counts, at the module level. This fact makes the proposed approach quite scalable when compared to other existing reliability analysis approaches for combinational circuits. Besides the accuracy and scalability, another distinguishing feature of the proposed approach is its generic nature. It verifies generic reliability expressions for combinational circuits that can be simply instantiated to evaluate the reliabilities of the given circuit under different inputs and error probabilities. To the best of our knowledge, no other publicly available reliability analysis approach provides these kind of features and thus the proposed approach is very promising for the microelectronic design community where the accurate reliability analysis of combinational circuits is a major issue.

This work opens the doors of many new areas in the direction of theorem proving based reliability analysis of combinational circuits. First of all, one of our ongoing projects is to analyze some statistical aspects, such as average or variance, of reliability of combinational circuits by building on top of our generic higher-order-logic based reliability analysis approach [16] instead of evaluating the reliability for individual input patterns. Similarly, in order to ensure 100% accurate results from the proposed reliability assessment framework, given in Figure 1, we are planning to formally verify the functional correctness of the C++ blocks, which have been verified using informal testing techniques so far. Another potential extension worth mentioning here is to lift the independence of signals assumption from the analysis presented in this paper. The independence assumption simplifies the analysis but leads to an abstract modeling of the real scenario, where the signals are somehow correlated. Such correlations can be integrated in the models presented in this paper using mathematical concepts of joint and conditional probabilities based on the approach presented in [25].

References

[1] Baier, C., Haverkort, B., Hermanns, H., Katoen, J.. Model Checking Algorithms for Continuous time Markov Chains. IEEE Transactions on Software Engineering 2003;29(4):524–541.

[2] Berard, B., Bidoit, M., Finkel, A., Laroussinie, F., Petit, A., Petrucci, L., Schnoebelen, P., McKenzie, P.. Systems and Software Verification: Model-Checking Techniques and Tools. Springer, 2001.

[3] Bhaduri, D., Shukla, S., Graham, P., Gokhale, M.. NANOPRISM: A Tool for Evaluating Granularity versus Reliability Trade-offs in Nano Architectures. In: Great Lakes Symp. VLSI,. ACM; 2004. p. 109–112.

[4] Bhaduri, D., Shukla, S., Graham, P., Gokhale, M.. Scalable Techniques and Tools for Reliability Analysis of Large Circuits. IEEE Trans Circuits and Systems 2007;54(11):2447–2460.

[5] Church, A.. A Formulation of the Simple Theory of Types. Journal of Symbolic Logic 1940;5:56–68.

[6] Clarke, E., Grumberg, O., Peled, D.. Model Checking. The MIT Press, 2000.

[7] Devroye, L.. Non-Uniform Random Variate Generation. Springer-Verlag, 1986.

[8] Gordon, M.. Mechanizing Programming Logics in Higher-0rder Logic. In: Current Trends in Hardware Verification and Automated Theorem Proving. Springer; 1989. p. 387–439.

[9] Gordon, M., Melham, T.. Introduction to HOL: A Theorem Proving Environment for Higher-Order Logic. Cambridge University Press, 1993.

[10] Hall, A.. Realising the Benefits of Formal Methods. J Universal Computer Science 2007;13(5):669–678.

[11] Han, J., Taylor, E., Gao, J., Fortes, J.. Faults, Error Bounds and Reliability of Nanoelectronic Circuits. In: Application-Specific Systems, Architecture Processors. IEEE Computer Society; 2005. p. 247–253.

[12] Han, J., Taylor, E., Gao, J., Fortes, J.. Reliability Modelling of Nanoelectronic Circuits. In: Conference on Nanotechnology. IEEE; 2005. p. 104–107.

[13] Harrison, J.. Formalized Mathematics. Technical Report 36; Turku Centre for Computer Science, Finland; 1996.

[14] Hasan, O.. Formal Probabilistic Analysis using Theorem Proving. PhD Thesis; Concordia University; Montreal, QC, Canada; 2008.

[15] Hasan, O., Patel, J., Tahar, S.. On the Accurate Reliability Analysis of Combinational Circuits using Theorem Proving. In: Proc. North-Eastern Workshop on Circuits and Systems. IEEE; 2010. p. 273–276.

[16] Hasan, O., Tahar, S., Abbasi, N.. Formal Reliability Analysis using Theorem Proving. IEEE Transactions on Computers 2010;59(5):579–592.

[17] Hurd, J.. Formal Verification of Probabilistic Algorithms. PhD Thesis; University of Cambridge; Cambridge, UK; 2002.

[18] Krishnaswamy, S., Viamontes, G.F., Markov, I., Hayes, J.. Accurate Reliability Evaluation and Enhancement via Probabilistic Transfer Matrices. In: Design, Automation and Test in Europe. IEEE Computer Society; 2005. p. 282–287.

[19] MacKay, D.. Introduction to Monte Carlo Methods. In: Learning in Graphical Models, NATO Science Series. Kluwer Academic Press; 1998. p. 175–204.

[20] Mathur, F.. On Reliability Modeling and Analysis of Ultra-reliable Fault-Tolerant Digital Systems. IEEE Trans Computers 1971;20(11):1376– 1382.

[21] Milner, R.. A Theory of Type Polymorphism in Programming. Journal of Computer and System Sciences 1977;17:348–375.

[22] Ogus, R.. The Probability of a Correct Output from a Combinational Circuit. IEEE Trans Computers 1975;24(5):534–544.

[23] Paulson, L.. ML for the Working Programmer. Cambridge University Press, 1996.

[24] Rutten, J., Kwaiatkowska, M., Normal, G., Parker, D.. Mathematical Techniques for Analyzing Concurrent and Probabilisitc Systems. volume 23 of *CRM Monograph Series*. American Mathematical Society, 2004.

[25] Taylor, E., Han, J., Fortes, J.. Towards the Accurate and Efficient Reliability Modeling of Nanoelectronic Circuits. In: Nanotechnology Conference. 2006. p. 395–398.