Accurate Reliability Analysis of Combinational Circuits using Theorem Proving

Osman Hasan, Jigar Patel and Sofiène Tahar

Department of Electrical and Computer Engineering,
Concordia University, Montreal, Canada
Email: {o_hasan, ji_p, tahar}@ece.concordia.ca

Technical Report

September, 2009

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 reliability analysis by presenting a formal reliability analysis approach that is based on higher-order-logic theorem proving. The distinguishing feature of the approach is that, despite being based on higher-order logic that is undecidable, it is capable of automatically evaluating the precise reliability of combinational circuits. The paper summarizes the formalization infrastructure that is fundamental to our approach. For illustration purposes, it also presents the reliability analysis of a few benchmark combinational circuits.

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 [12, 13]. Nowadays, the ability to efficiently analyze the reliability of combinational circuits has become very challenging since they are being fabricated at the nanoscale level and are thus not only humongous in size but are also more prone to errors because of the inherent variability in the fabrication processes.

A number of reliability analysis approaches for combinational circuits have been recently proposed that tend to somewhat meet the above mentioned 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) [11]. 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 arithmetic operations. The PTM of the whole circuit can now 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 in this approach. A more efficient approach, in terms of space and time complexity, than the PTM method, has been proposed in [8] that calls for developing von-Neumann models, called the probability gate models (PGMs), for unreliable logic gates and use these models 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 extensively to analyze the reliability of many combinational circuits. Thus, as far as conducting the analysis of the present age 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% precise. 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. 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 in computer simulations, the fundamental idea is to approximately answer a query by analyzing a large number of samples and thus by its inherent nature the results cannot be termed as accurate.

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, like medicine, military and transportation, where an erroneous analysis could even result in the loss of human lives. Therefore, traditional techniques like simulation or paper-and-pencil proofs should not be relied upon for the reliability analysis of combinational circuits that are supposed to be used in safety-critical domains. Formal methods [7] are capable of conducting precise system analysis and thus overcome the above mentioned limitations of traditional approaches. 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 [3] and higher-order-logic theorem proving [5]. 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.

Given the dire need of accuracy in the area of reliability analysis of combinational circuits, probabilistic model checking, which enables analyzing systems with random or unpredictable

behaviors, has been recently used for their analysis as well [1, 2]. 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 [4]. 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.

We believe that due to its high expressiveness nature, higher-order-logic theorem proving, which is the other commonly used formal method, can be utilized to overcome the above mentioned limitations of probabilistic model checking in the domain of accurate reliability analysis of combinational circuits. Though, this solution mainly involves two main challenges. The first one is that we need a foundational infrastructure to be able to formally specify and reason about the 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 user efforts involved in interactively reasoning about the reliability properties of the system in hand, which could be very tedious at times. 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 the above mentioned challenges and, to the best of our knowledge, presents the first theorem proving based approach for the reliability analysis of combinational circuits. The proposed infrastructure, illustrated in Figure 1, is primarily inspired by the PGM method. It accepts the VHDL model of the combinational circuit, the output port name in the circuit for which the reliability needs to be computed, the error probability and a combination of its input values. Whereas, it returns the precise 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++ module. This goal is then automatically verified based on some already verified properties in a theorem prover and is thus 100% precise. The theorem proving infrastructure, which enables the automatic verification, is based on the formal definitions of a faulty component and reliability of combinational circuits in higher-order-logic. These definitions exhibit random and probabilistic behaviors, due to the random nature of gate-faults, and thus have been formally defined by building upon the methodology for higher-order-logic formalization of probabilistic algorithms given in [10]. These definitions are then utilized to interactively verify key properties associated with the reliability evaluation of combinational circuits in the PGM approach. These properties include a generalized form of von-Neumann equation, which allows us to evaluate the probability of getting a logical 1 for any combinational gate, and a generic expression that allows us to evaluate the reliability of a combinational circuit in terms of the probabilities of getting a logical 1 at its output or any one of its intermediate nodes. Due to their generic nature, the formally verified properties can be used to automatically verify the reliability requirement goal for any combinational circuit under the given conditions.

To illustrate the practical effectiveness of the proposed infrastructure, we utilize it to assess the reliability of a simple comparator and four benchmarks, i.e., LGSynth'91-C17, LGSynth'91-Majority, LGSynth'91-Parity, and ISCAS-85-74283 (4-bit Adder), in this paper.

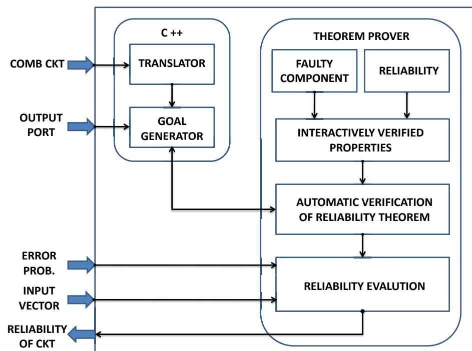

Figure 1: Proposed Reliability Analysis Infrastructure

We used the HOL theorem prover [6], which is based on higher-order logic, for this work. 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 main motivation behind choosing HOL for our work is that the probabilistic analysis of algorithms infrastructure that we build upon is developed in HOL.

The rest of the paper is organized as follows. In Section 2, we present an overview of the infrastructure for the probabilistic analysis of algorithms. Section 3 presents the core of this paper, where we present the formalization and verification details regarding the reliability properties that allow us to automatically conduct the reliability analysis of combinational circuits. The experimental results along with a more detailed explanation of the proposed tool, shown in Figure 1, are given in Section 4. Finally, Section 5 concludes the paper.

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 [10]. 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 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 $t(x)$. Now, by formalizing a probability space of infinite Boolean sequences in higher-order logic, where the probability function $\mathbb{P}$ maps from sets of infinite Boolean sequences to real numbers between 0 and 1, we can formally prove probabilistic properties for random variables in a theorem prover [10]. For example, the following Probability Mass Function (PMF) property can be verified for the function `bit`.

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

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 variables, formalized as the function `ber_rv`, and verified using the following PMF relation [10]:

Lemma 1: *PMF of Bernoulli(p) Random Variable*

```
∀ p.  0 ≤ p ∧ p ≤ 1 ⇒
  ℙ {s | fst (ber_rv p s)} = p
```

The function `ber_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*.

3 HOL RELIABILITY ANALYSIS INFRASTRUCTURE

In this section, we describe the formalization and verification of our HOL definitions and theorems, which lead to the automatic and precise reliability evaluation of combinational circuits in the proposed infrastructure.

As illustrated in Figure 1, the first step in the proposed theorem proving based reliability analysis infrastructure is to formally express the behavior of a faulty component.

Definition 1: *von-Neumann Faulty Component*

```
∀ 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))))
```

where ¬ denotes the logical negation in the above defnition. The function `rv_list` accepts a list of random variables and returns the 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. The function `faulty_comp` accepts three variables, i.e., 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, a list of Boolean random variables `P`, which corresponds to the values available at the input of the component, and the probability `e` of error occurrence in the component. Whereas, 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 dependent on the error probability `e` and the input random variable list `P`. Therefore, this function is formally modeled using the infrastructure explained in Section 2. Another worth mentioning point here is that we have used the Bernoulli random variable function `bern_rv` to model the random behavior associated with the error occurrence. This way, the function `faulty_comp` models the erroneous behavior of a component based on the von-Neumann model [8], 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-Newmann model of a component.

Theorem 1: *General Expression for Gate Reliability*

```
∀ e f P. (0 ≤ e ≤ 1) ⇒
  (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 $[0, 1]$. 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 is primarily based on the independence of error occurrence and PMF of the Bernoulli random variable, given in Lemma 1.

Theorem 1 can now be used to formally reason about the probability of obtaining a logical 1 from any logical gate. For illustration purposes, consider an N-input AND-gate for which the Boolean functionality can be formally defined as follows:

Definition 2: *N-Bit AND Gate*

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

The function and_g 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 output in N-Bit AND Gate*

```
∀ e P. (0 ≤ e ≤ 1) ⇒
  (ℙ {s|fst(faulty_comp and_g P e s)} =
    e (1 - prob_rv_list_mul P) +
    (1 - e) (prob_rv_list_mul P))
```

where the function prob_rv_list_mul returns the multiplication of the probabilities of each random variable being equal to *True* in the given list of random variables. 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 output in 2-Bit AND Gate*

```
∀ x1 x2 e.  (0 ≤ e ≤ 1) ⇒
ℙ{s|fst(faulty_comp and_g[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 if we know the probabilities of obtaining a logical 1 at both of its inputs individually.

The probability of obtaining a logical 1 from any logical gate be verified in a similar way as described above. The formally verified theorems corresponding to such probabilities for some commonly used 2-input logical gates are given in Table I, which are verified under the assumption that e lies in the interval $[0, 1]$. 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 play a vital role in reasoning about the reliability of combinational circuits as will be seen next.

The next step in the proposed reliability analysis infrastructure 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 [8]. This can be formally expressed using the function faulty_comp, given in Definition 1, as follows:

Definition 3: *Reliability*

```
∀ f L e.  rel 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)))}
```

Gate	Theorem
NAND	$\mathbb{P}\{s \mid fst(faulty_comp\ nand_g[X_1; X_2]e\ s)\}$ $= (1-e) + (2e-1)X_1X_2$
NOR	$\mathbb{P}\{s \mid fst(faulty_comp\ nor_g[X_1; X_2]e\ s)\}$ $= 1 - X_2 - X_1 + X_1X_2(1-2e)+$ $e(2X_1 + 2X_2 - 1)$
Interconnect	$\mathbb{P}\{s \mid fst(faulty_comp\ xconnect[X]e\ s)\}$ $= X + e(1-2X)$
Inverter	$\mathbb{P}\{s \mid fst(faulty_comp\ not_g[X]e\ s)\}$ $= 1 - X - e + 2eX$
XOR	$\mathbb{P}\{s \mid fst(faulty_comp\ xor_g[X_1; X_2]e\ s)\}$ $= X_2 + X_1 - 2X_1X_2+$ $e(4X_1X_2 - 2X_2 - 2X_1 + 1)$
Majority	$\mathbb{P}\{s \mid fst(faulty_comp\ maj_g[X_1; X_2]e\ s)\}$ $= X_1X_2 + X_1X_3 + X_2X_3 - 2X_1X_2X_3+$ $e(4X_1X_2X_3 - 2X_1X_2 - 2X_1X_3 - 2X_2X_3 + 1)$

Table 1: Probability of Output equal to 1 for Commonly used Gates

The function `rel` 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` is 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 function `rel` returns the desired reliability of the component with functionality `f` and error probability `e`. The left-hand-side (LHS) term in the set represents the output of the component while considering the effect of error and the the RHS term represents the error free output of the given component. It is important to note that the remaining 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 independent 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 [14].

Theorem 5: *Alternate Expression for Reliability*

```
∀ f L e.  0 ≤ e ≤ 1 ⇒
  (rel 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 main advantage of the above expression 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 I, for finding the probability of attaining a logical 1 at the output for most of the commonly used gates. This infrastructure, i.e., the theorems given in Table I and Theorem 5, is based on the formally verified results in the HOL theorem prover and hence the results attained by building on top it can be regarded as 100% precise unlike all the available combinational logic reliability analysis approaches.

4 EXPERIMENTAL RESULTS

In order to demonstrate the utilization and practical effectiveness of our approach, we now present the some examples. First consider the comparator circuit of Figure 2. The proposed infrastructure, given in Figure 1, 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 infrastructure 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 3. 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_g (λx.  [ber_rv pA; (faulty_comp nand_g
      [ber_rv pA;ber_rv pB] x)])
```

The function `and_g` 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 `ber_rv` with input probability `pA` of getting a logical 1 at this input for its 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 and these connections can be observed in the input random variable list for the function `nand_g` in the output of the Translator.

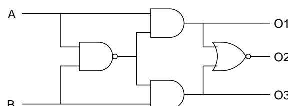

Figure 2: A 2-bit Comparator

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

```
∀ pA pB e.  0 ≤ e ≤ 1 ∧
  0 ≤ pA ∧ pA ≤ 1 ∧ 0 ≤ pB ∧ pB ≤ 1 ⇒
    (rel and_g
  (λx.  [ber_rv pA; (faulty_comp nand_g
      [ber_rv pA;ber_rv pB] x)]) e = Z)
```

The LHS of the proof goal represents the reliability of the given comparator circuit, using the function `rel` given in Definition 3, 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 I and Theorem 5. Once the most simplified form is obtained in HOL, the expression is fed back to the Goal Generator module, which replaces the real number Z by the simplified expression and generates a new proof goal, which in the case of the comparator circuit is as follows:

Theorem 6: *Reliability for Comparator Output o1/o3*

```
∀ pA pB e.  (0 ≤ e ≤ 1) ∧
  (0 ≤ pA ≤ 1) ∧ (0 ≤ pB ≤ 1) ⇒
    (rel and_g
    (λx.  [ber_rv pA; (faulty_comp nand_g
      [ber_rv pA;ber_rv pB] x)]) e =
(pA(1-e+(2e-1)pApB)+
  e(1-2pA(1-e+(2e-1)pApB)))(
pA(1-(pApB)))+(1-(pA(1-e+(2e-1)pApB)+e(1
-2pA(1-e+(2e-1)pApB))))(1-pA(1-(pApB))))
```

The new goal is now fed to HOL and this time is automatically verified using the theorems given in Table I and Theorem 5. 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`. Thus, we evaluate the expression on the RHS of the above subgoal in ML for the given values of these variables and the reliabilities for the input combinations 00, 01, 10 and 11 with error probability equal to 0.1 were found to be 0.9, 0.9, 0.7624 and 0.6984, respectively.

Just like the comparator circuit, we assessed the reliabilities of 4 benchmark circuits, given in Table II, with all inputs set to logical 1's and gate error probability e = 0.05. The experiments were run on a Linux workstation with Sparc-v9 processor operating at 1015 MHz with 4096 Megabytes of memory. First of all these reliability results are based on formally verified theorems in a theorem prover and thus their integrity cannot be doubted, i.e., they are 100% precise. 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. Secondly, the 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 II include the verification of generic reliability theorems like Theorem 6 for the benchmark circuits. This means that the times for evaluating reliabilities for other input

Benchmark	Circuit	Gates	Inputs	Outputs	Output Name	Reli- ability	Time (sec)
LGSynth91	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 2: Reliability of Benchmark combinational Circuits

combinations for the same benchmarks now would be almost negligible since they would utilize the same theorems as well.

Comparing the proposed approach to the other existing approaches [8, 11], 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. Similarly, the proposed approach somewhat overcomes the scalability issue since there is no simulation involved. Also, the powerful induction proofs can be utilized to analyze the reliability of generic circuits. For example, if the reliability or the probability of having a 1 at the output of a full-adder circuit is evaluated, then it can be built upon to formally assess the reliability of an N-bit adder using the powerful induction methods in a theorem prover. The other formal reliability analysis approach, i.e., the one based probabilistic model checking (PMC), do not enjoy such benefits and its usage is limited by the state-space-explosion problem. For example, Table 1 in [2] provides more details regarding the well known 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 [2] for only a specific set of values of error probabilities.

5 CONCLUSIONS

The paper presents the first theorem proving based infrastructure for the reliability analysis of combinational circuits. Due to the formal nature of the approach, the reliability results are 100% accurate and thus can be very useful for the analysis of combinational circuits that are used in safety critical applications. The results of Section 3 of the paper form the main core of the proposed infrastructure and were 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 then in turn can be used to automatically assess the reliability of any combinational circuit, which has been illustrated in the paper by analyzing some combinational circuit benchmarks.

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 the average or expected reliability of circuits by building on top of the higher-order-logic formalization of expectation theory [9] instead of evaluating the reliability for individual input patterns. The work presented in this paper can also be extended upon to model the source of the signal dependencies [14], i.e., fanouts, feedbacks etc., to further improve upon the accuracy of the results.

References

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

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

[3] E. Clarke, O. Grumberg, and D. Long. Verification Tools for Finite State Concurrent Systems. In *A Decade of Concurrency-Reflections and Perspectives*, volume 803 of *LNCS*, pages 124–175. Springer, 1993.

[4] E.M. Clarke, O. Grumberg, and D.A. Peled. *Model Checking*. The MIT Press, 2000.

[5] M.J.C. Gordon. Mechanizing Programming Logics in Higher-0rder Logic. In *Current Trends in Hardware Verification and Automated Theorem Proving*, pages 387–439. Springer, 1989.

[6] M.J.C. Gordon and T.F. Melham. *Introduction to HOL: A Theorem Proving Environment for Higher-Order Logic*. Cambridge University Press, 1993.

[7] A. Gupta. Formal Hardware Verification Methods: A Survey. *Formal Methods in System Design*, 1(2-3):151–238, 1992.

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

[9] O. Hasan and S. Tahar. Using Theorem Proving to Verify Expectation and Variance for Discrete Random Variables. *Journal of Automated Reasoning*, 41(3–4):295–323, 2008.

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

[11] S. Krishnaswamy, G. F. Viamontes nd I.L. Markov, and J.P. Hayes. Accurate Reliability Evaluation and Enhancement via Probabilistic Transfer Matrices. In *Design, Automation and Test in Europe*, pages 282–287. IEEE Computer Society, 2005.

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

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

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