Novel Probabilistic Combinational Equivalence Checking

Shih-Chieh Wu, Chun-Yao Wang, *Member, IEEE*, and Yung-Chih Chen

***Abstract*—Exact approaches to combinational equivalence checking, such as automatic test pattern generation-based, binary decision diagrams (BDD)-based, satisfiability-based, and hybrid approaches, have been proposed over the last two decades. Recently, we proposed another exact approach using signal probability. This probability-based approach assigns probability values to the primary inputs and compares the corresponding output probability of two networks via a probability calculation process to assert if they are equivalent. The shortcoming of all these exact approaches is that if two networks are too complex to be handled, their equivalence cannot be determined, even with tolerance. An approximate approach, named the probabilistic approach, is a suitable way to give such an answer for those large circuits. However, despite generally being more efficient than exact approaches, the probabilistic approach faces a major concern of a non zero aliasing rate, which is the possibility that two different networks have the same output probability/signatures. Thus, minimizing aliasing rate is substantial in this area. In this paper, we propose a novel probabilistic approach based on the exact probability-based approach. Our approach exploits proposed probabilistic equivalence checking architecture to efficiently calculate the signature of network with virtually zero aliasing rate. We conduct experiments on a set of benchmark circuits, including large and complex circuits, with our probabilistic approach. Experimental results show that the aliasing rate is virtually-zero, e.g., 10^{-6013}. Also, to demonstrate the effectiveness of our approach on error detection, we randomly inject errors into networks for comparison. As a result, our approach more efficiently detects the error than a commercial tool, Cadence LEC, does. Although our approach is not exact, it is practically useful. Thus, it can effectively complement exact methods to improve the efficiency and effectiveness of combination equivalence checking algorithms.**

***Index Terms*—Aliasing-free assignment, aliasing rate, combination equivalence checking (CEC), error detection, probabilistic approach, probabilistic equivalence checking (PEACH).**

I. Introduction

TRADITIONALLY, logic verification is carried out by pattern simulation. However, to exhaustively simulate all possible patterns is infeasible for practical designs with numerous inputs. Thus, formal combinational equivalence checking (CEC) methods are gaining popularity because they make it possible to guarantee the equivalence of two networks.

Existing exact approaches for formally verifying the equivalence of two networks can be classified into five categories [11]: 1) automatic test pattern generation (ATPG)-based [3], [25]; 2) binary decision diagrams (BDD)-based [7], [9], [15]; 3) satisfiability (SAT)-based [6], [23]; 4) hybrid [19], [20]; and 5) probability-based [26]. ATPG-based methods identify some internal gates of two networks and use them to construct a *miter* structure [3], as shown in Fig. 1, using ATPG [8] to examine whether or not the output of the miter stuck-at-0 fault is untestable. If the fault is untestable, no pattern exists to distinguish the two logic cones. Hence, these internal gates are equivalent, so one internal gate can be replaced by its equivalent gate to simplify the overall network. The efficiency of this approach relies on the capability of ATPG. If the test generation of the fault at the miter output is time-consuming or intractable, the approach becomes inefficient.

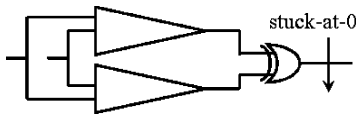

Fig. 1. Miter.

Reduced ordered binary decision diagrams (ROBDDs) [4] are a canonic representation of networks. Thus, two networks are equivalent if and only if the ROBDDs are isomorphic [5], [9]. BDD-based approaches encounter potential difficulties in ROBDDs construction. Certain circuits, such as multipliers with numerous inputs, cannot be represented by ROBDDs in any variable ordering [4].

Recently, Boolean SAT has been successfully used as an efficient and complete method for solving CEC problems [6], [23]. It transforms a combinational network to the conjunctive normal form (CNF) formula, which can be viewed as a set of clauses. The objective of the SAT-based approach is to prove propositional properties of the network [14]. However, the problem is that the SAT-based approach sometimes requires a large amount of time and backtracks to prove the network.

To improve the efficiency of CEC techniques, approaches combining BDD and SAT techniques are proposed in [19] and [20]. In general, these approaches partition the miter circuit into many blocks. The SAT engine is used in each partition of circuit and BDD engine is used for each cutpoint. Although these hybrid approaches are capable of solving verification instances that neither of both techniques was capable to solve, the CPU time needed is large.

Signal probability at the output of a network is considered as a signature function [2], [10] for the CEC problem. Signature functions are used to characterize properties of networks, e.g., the number of minterms in the onset of a network is a signature function. The signal probability, however, can be exact as well

Manuscript received August 30, 2006. This work was supported in part by the National Science Council of the Republic of China under Grant NSC 95-2220-E-007-020.

The authors are with the Department of Computer Science, National TsingHua University, HsinChu 300, Taiwan, R.O.C. (e-mail: wcyao@cs.nthu.edu.tw).

Digital Object Identifier 10.1109/TVLSI.2008.917397

if the input probability is an aliasing-free assignment [24]. The exact probability-based approach [26] assigns aliasing-free assignments at all primary inputs (PIs) of networks, then derives output probability by a probability calculation process from the PIs towards the primary outputs (POs). Two networks are equivalent if and only if their output probabilities are equal under the aliasing-free assignments. However, the disadvantage of these assignments is that they exponentially grow. Thus, this approach is not applicable to large circuits either.

The problem that all these exact methods share is that designers are unable to calculate the equivalence, even with tolerances, of any networks that are too complex for the methods to handle. For example, a commercial tool Cadence LEC probably returns *abort* after verification. From the viewpoint of verification engineers, we hope that at least one error, if it exists, can be quickly revealed for an effective verification process. Thus, the probabilistic approach can be used under this situation. An efficient probabilistic approach with virtually-zero aliasing rate balances the concerns of accuracy with the demands of computational effort and is a practical technique for verification.

Several probabilistic approaches have been proposed in [10] and [16]. Jain *et al.* [10] first propose an idea that simulates numeric numbers instead of binary numbers to obtain a signature of circuit for comparison. Although it considerably reduces the aliasing rate, e.g., 10^{-8} for a 64-input circuit, a large amount of complex multiplication operations are required when evaluating output signature. Also, the approach cannot directly apply the randomly generated numeric numbers into the circuit for simulation if signal correlation exists. It needs an *A-transform* manipulation to deal with the signal correlation issue. Meinel *et al.* [16] exploit a space-efficient $\oplus$-OBDD structure for probabilistic verification. Since $\oplus$-OBDD is not a canonical representation, two different functions with the same $\oplus$-OBDD representation are considered as an equivalent function. Although the aliasing rate is not large either, e.g., 10^{-10} for 100-input circuit with 10^7 nodes in $\oplus$-OBDD, unfortunately, this approach still faces memory explosion problem if verification instances are complex or large. Also, arithmetic operations are also required.

In this paper, we propose a novel probabilistic approach with probabilistic equivalence checking (PEACH) architecture for CEC problem. We also provide quantitative and theoretical analysis on aliasing rate based on this architecture. The virtually-zero aliasing rate significantly rises the confidence on our approach. Furthermore, our approach needs not complex arithmetic operations and can well deal with signal correlation issue such that numeric numbers can be directly applied into the circuits with/without reconvergent fan-outs.

We conduct experiments on a set of benchmark circuits, including large and complex circuits, with our probabilistic approach. Experimental results show that the aliasing rate is virtually zero, e.g., 10^{-6013}. Also, to demonstrate the effectiveness of our approach on error detection, we randomly inject errors into networks for comparison. As a result, our approach more efficiently detects the error than a commercial tool, Cadence LEC, does. Although our approach is not exact, it is practically useful. Thus, it can effectively complement exact methods to improve the efficiency and effectiveness of CEC algorithms.

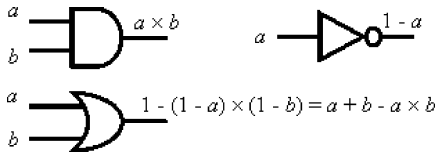

Fig. 2. Probability formula for primitive gates.

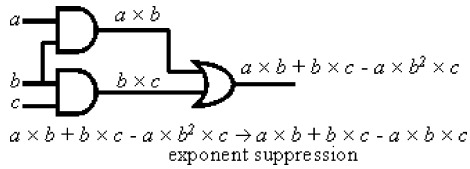

Fig. 3. Calculation of probability expression with exponent suppression.

The rest of this paper is organized as follows. Section II reviews the background of signal probability in a Boolean network and an exact probability-based approach [26]. Section III presents the PEACH architecture for CEC and analyzes the aliasing rate. Section IV shows the experimental results. Section V concludes this paper.

II. Background

This section first reviews the background of signal probability in a network and introduces the exact probability-based approach in Section II-B. It is the core technique of our probabilistic approach as presented in more detail in Section III. Here, we assume networks only consist of AND, OR, and NOT gates for simplicity. Complex gates can be decomposed into these gates.

A. Probability Expression of a Network

In this paper, we denote a gate in the network by an upper case letter and its probability of signal 1 by the corresponding lower case letter. The known probability formula for 2-input AND, OR, and NOT gates with independent signals are summarized in Fig. 2. The formula for AND, OR gates with more than two inputs can be extended from these formula.

Definition 1: Given gates s and d in the network, if more than one disjoint path exists from s to d, d is a reconvergent gate in the network [13], [22].

The *probability expression* of a network can be straightforwardly derived from the PIs to the POs by using these probability formula with an exponent suppression operation. The *exponent suppression* replaces the term x^m with x for every gate X in the expression [12], [18] due to a gate X is fully correlated with itself in the reconvergent gate.

For example, in Fig. 3, the probability expression at the output is originally $a\times b+b\times c-a\times b^2\times c$. After the exponent suppression, the probability expression becomes $a\times b+b\times c-a\times b\times c$ (b^2 is replaced by b). It is proven that the probability expression after the exponent suppression is unique for a network [12], [18]. Namely, the probability expression is a canonic representation.

Although probability expression is a canonic representation, deriving it for a large circuit is intractable. This is because $O(N\times 2^N)$ operations [12] are required to get the probability

X_3	X_2	X_1	x_3	x_2	x_1	prob. of minterm
0	0	0	$(1-\frac{1}{17})$	$(1-\frac{1}{5})$	$(1-\frac{1}{3})$	$\frac{128}{255}$
0	0	1	$(1-\frac{1}{17})$	$(1-\frac{1}{5})$	$\frac{1}{3}$	$\frac{64}{255}$
0	1	0	$(1-\frac{1}{17})$	$\frac{1}{5}$	$(1-\frac{1}{3})$	$\frac{32}{255}$
0	1	1	$(1-\frac{1}{17})$	$\frac{1}{5}$	$\frac{1}{3}$	$\frac{16}{255}$
1	0	0	$\frac{1}{17}$	$(1-\frac{1}{5})$	$(1-\frac{1}{3})$	$\frac{8}{255}$
1	0	1	$\frac{1}{17}$	$(1-\frac{1}{5})$	$\frac{1}{3}$	$\frac{4}{255}$
1	1	0	$\frac{1}{17}$	$\frac{1}{5}$	$(1-\frac{1}{3})$	$\frac{2}{255}$
1	1	1	$\frac{1}{17}$	$\frac{1}{5}$	$\frac{1}{3}$	$\frac{1}{255}$

Fig. 4. Probability of each minterm for 3-input functions assuming $x_1 = 1/3$, $x_2 = 1/5$, and $x_3 = 1/17$.

expression of an N-input network. Also, the number of terms in a probability expression is 2^N in the worst case [12].

B. Exact Probability-Based Approach

The exact probability-based approach [26] assigns numerical probabilities to PIs and evaluates the probabilities at POs for comparison where the assigned probability is aliasing-free [24]. Thus, it produces a unique output probability for each function. This output probability can be seen as a unique signature of a network.

Equation (1) is a recursive formula reported in [24] that produces aliasing-free probability assignments for an N-input network, where $1/\theta_i$ is the 1's probability of input variable X_i, $i = 1 \sim N-1$

$$\theta_{i+1} = (\theta_i - 1)^2 + 1 = {\theta_i}^2 - 2\theta_i + 2 \qquad i = 1 \sim N-1; \quad \theta_1 \geq 3; \text{ and } \theta_1 \in \mathrm{Z}^+. \tag{1}$$

To minimize memory usage in representing the probability of a gate, the assignment of $\theta_1 = 3$ is preferable. Thus, the aliasing-free assignment uses $\theta_1 = 3$ as the first assignment throughout this paper. The following example demonstrates why (1) results in a unique output probability for each function.

For a 3-input function, there are $2^{2^3} = 256$ distinct functions. If we set $\theta_1 = 3$, $x_1 = 1/3$; $\theta_2 = 5$, $x_2 = 1/5$; and $\theta_3 = 17$, $x_3 = 1/17$ according to (1), the probability of each minterm is shown in Fig. 4. The probability of each minterm acts as a weight which is similar to the weight of binary numeral system. The probability of each function is the summation of subset of these weights. Thus, each function has a unique probability and these probabilities are uniformly distributed among $0/255 \sim 255/255$.

Teslenko *et al.* [24] propose the aliasing-free assignments for CEC problem. But the work concludes that the assignments are of only theoretic interest. The corresponding CEC algorithm using these assignments is not proposed in [24]. Wu *et al.* then develop a CEC algorithm using these aliasing-free assignments in [26]. We will describe the algorithm in detail in the following.

Take Fig. 5 as an example. We can derive the probability expression of network_ori and network_opt. Both probability expressions are $a \times b + b \times c - a \times b \times c$; thus, they are equivalent networks. Next, instead of deriving probability expression, we introduce how to verify these two networks by using

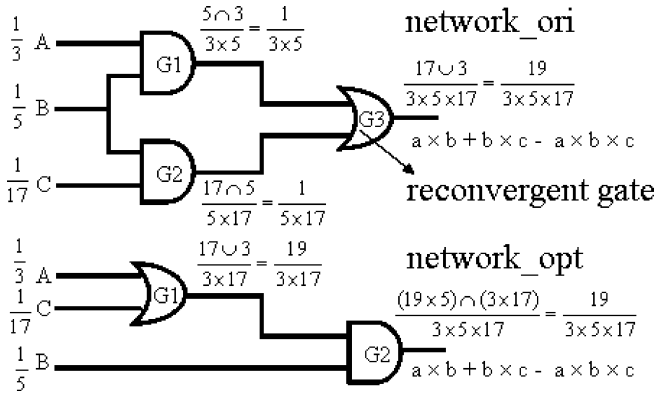

Fig. 5. Output probability evaluation process.

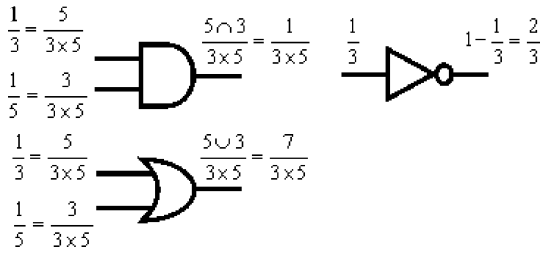

Fig. 6. Alternative operations for primitive gates.

aliasing-free assignments and get the output probability in a single-pass calculation. First, we assign the aliasing-free assignments $a = 1/3$, $b = 1/5$, and $c = 1/17$, to PIs A, B, and C in the network_ori and network_opt, respectively. As opposed to the original probability formula, these aliasing-free assignments use efficient alternative operations to calculate output probability, i.e., bitwise-AND ($\cap$) in an AND gate and bitwise-OR ($\cup$) in an OR gate. The probability evaluation process for these primitive gates are shown in Fig. 6. After finding a lowest common multiple denominator of two input probabilities in an AND/OR gate, we transform these two input probabilities to their equivalent probabilities with the same denominator. Then, the two new numerators conduct a bitwise-AND/bitwise-OR operation to obtain the numerator of output probability in an AND/OR gate. For example, to get the output probability of the AND gate in Fig. 6, we first transform 1/3 to $5/(3 \times 5)$ and 1/5 to $3/(3 \times 5)$. Then, the two new numerators conduct bitwise-AND operation, $5 \cap 3 = 001_2 = 1$, to obtain the output probability, $1/(3 \times 5)$. The calculation for the other gates can also be seen in Fig. 6. Applying these rules in Fig. 6 to gates in the networks, we can obtain correct output probabilities even though reconvergent gates existing in the networks [26]. For example, Fig. 5 shows two networks having the same output probability, $19/(3 \times 5 \times 17)$; thus, we can know they are indeed equivalent as mentioned.

An inherent disadvantage of aliasing-free assignments is that the assignments exponentially grow. The numerator of output probability may become too large to be represented. Wu *et al.* [26] report that the 24th assignment, θ_{24}, $2^{2^{24-1}} + 1 \approx 2^{2^{23}} \approx 10^{2525222}$, within a single fan-in cone is the maximal value it can support. Thus, this paper proposes an approximate approach for those large circuits.

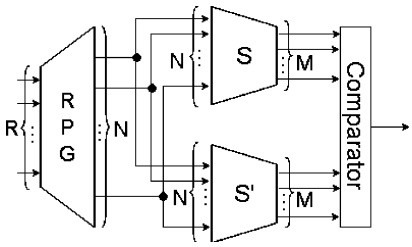

Fig. 7. PEACH architecture.

III. Probabilistic CEC and Its Analysis

This section first presents a verification architecture PEACH that is used for CEC and can provide a feature of configurable aliasing rates. Next, a detailed analysis on the aliasing rate with the PEACH is presented.

A. PEACH Architecture

The PEACH architecture is shown in Fig. 7. It contains three components: a random probability generator (RPG), a golden network (S), and a DUV (S'), and a comparator. S and S' are both N-input M-output networks, the RPG is an R-input N-output circuit, and the comparator has $2M$ inputs and one output. To verify the equivalence of networks S and S', we first apply R aliasing-free assignments to RPG's PIs. Next, we *randomly* select N output probabilities among $0/(2^{2^R}-1) \sim (2^{2^R}-1)/(2^{2^R}-1)$ and assign them to the PIs of S and S'. Using the probability evaluation process mentioned in Section II-B, we can get the output probabilities of S and S'. Then, the comparator makes a pairwise comparison of the output probabilities and reports if S and S' are equivalent or not. We use Example 3.1 to explain the process of equivalence checking using the PEACH architecture.

1) Example 3.1: Two networks S and S' as shown in Fig. 8 will be verified for equivalence. Assume the RPG has two PIs. (Note that this number is determined by designers.) We apply two aliasing-free assignments, 1/3 and 1/5, to the PIs of RPG. Hence, there are $2^{2^2} = 16$ ($0/15 \sim 15/15$) possible output probabilities out from the RPG. Since S and S' only have three inputs, we randomly select three of them, e.g., 1/15, 7/15, and 10/15. As earlier mentioned, the aliasing-free assignments have a property of *uniqueness*. Therefore, one output probability can represent one function. For example, output probability 1/15 represents the 2-input AND function, output probability 7/15 represents the 2-input OR function, and so on. These functions are also shown inside the RPG in Fig. 8. Next, we assign these selected probabilities, 1/15, 7/15, and 10/15 to the PIs A, B, and C of S and S', respectively. That is, $a = 1/15$, $b = 7/15$, and $c = 10/15$. After the probability calculation, the output probability of S is 3/15 and that of S' is 8/15. Thus, the comparator reports that S and S' are non-equivalent networks.

The PEACH architecture successfully verifies that networks S and S' are different in Example 3.1. However, if the randomly selected output probabilities are 0/15, 6/15, and 0/15 (repeated probability is possible), an identical output probability of S and S', 0/15, is obtained. This indicates that aliasing may occur in performing equivalence checking using the PEACH architecture.

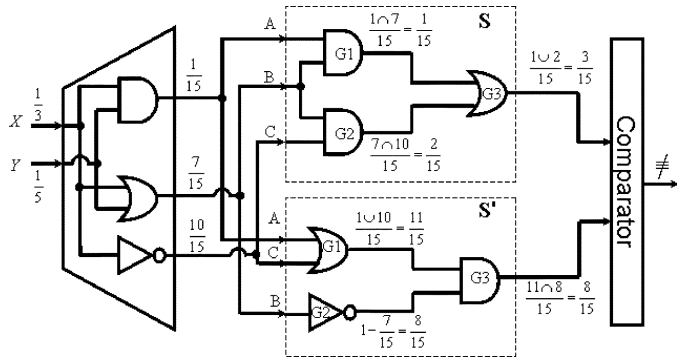

Fig. 8. Illustration of example 3.1.

Definition 2: Aliasing is the situation where two non-equivalent networks S and S' have an identical output probability under the same input assignments.

The manner for selecting the output probability of RPG connecting to the PIs of S and S' affects the magnitude of aliasing rate. Although a good selection of the output probability of RPG can lead to a lower aliasing rate, it requires extra computational effort. Thus, instead of carefully selecting the output probability of RPG, we analyze the aliasing rate with random selection.

As we mentioned, a major disadvantage of aliasing-free assignments is that the assignments exponentially grow. Thus, in practice the number of aliasing-free assignments is very limited using the exact approach. However, using the approximate approach with PEACH, the number of assignments reach at least 2^{2^R}, where R is the number of PIs in RPG. If the RPG has 10 PIs, there are $2^{2^{10}} \approx 10^{308}$ output probabilities that can be chosen as input probabilities of S and S'. Thus, this architecture makes verification possible on very large circuits. Although PEACH architecture can cause aliasing, we observe that if $R \geq 15$, the probability that aliasing occurs is virtually zero. The detailed analysis is presented in Section II-B.

B. Aliasing Rate Analysis

To analyze the probability of aliasing occuring, we define an L-transformation for the PEACH architecture. The L-transformation can transform an N-input M-output function (S) into an R-input M-output function (L). For example, network S in Fig. 9 is a 3-input 1-output network. Its function is $f(A,B,C) = A \cdot B + B \cdot C$. The 2-input RPG in Fig. 9 performs L-transformation on S. Thus, S is transformed into a new network L as shown in Fig. 9. The function of L is $f(X,Y) = (X \cdot Y) \cdot (X+Y) + (X+Y) \cdot \overline{X}$ due to $A = X \cdot Y$, $B = X + Y$, and $C = \overline{X}$. As a result, S and S' within the PEACH architecture are transformed into new networks L and L' by L-transformation, respectively. Note that the original objective is to determine the equivalence of S and S', but using PEACH, we can only determine the equivalence of L and L'. Next, we clarify the effect of this transformation on our original objective.

From the definition of aliasing in Definition 2, the aliasing rate can be formally defined as follows.

Definition 3: Given two networks S and S', S and S' are transformed into L and L' by L-transformation. The aliasing

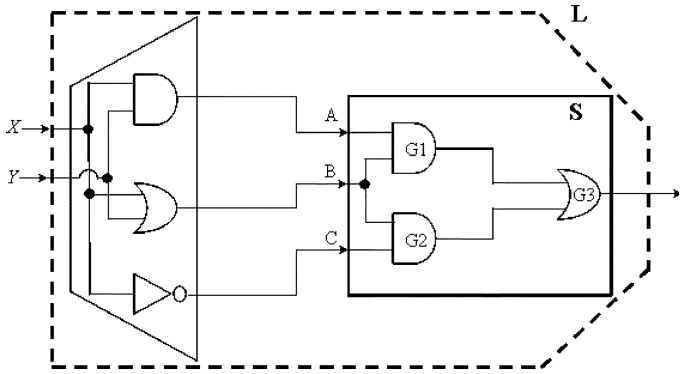

Fig. 9. An example of L-transformation.

C	B	A	c	b	a	prob. of minterm
0	0	0	$(1-\frac{3}{15})$	$(1-\frac{2}{15})$	$(1-\frac{1}{15})$	$\frac{12\cap 13\cap 14}{15}=\frac{12}{15}$
0	0	1	$(1-\frac{3}{15})$	$(1-\frac{2}{15})$	$\frac{1}{15}$	$\frac{12\cap 13\cap 1}{15}=\frac{0}{15}$
0	1	0	$(1-\frac{3}{15})$	$\frac{2}{15}$	$(1-\frac{1}{15})$	$\frac{12\cap 2\cap 14}{15}=\frac{0}{15}$
0	1	1	$(1-\frac{3}{15})$	$\frac{2}{15}$	$\frac{1}{15}$	$\frac{12\cap 2\cap 1}{15}=\frac{0}{15}$
1	0	0	$\frac{3}{15}$	$(1-\frac{2}{15})$	$(1-\frac{1}{15})$	$\frac{3\cap 13\cap 14}{15}=\frac{0}{15}$
1	0	1	$\frac{3}{15}$	$(1-\frac{2}{15})$	$\frac{1}{15}$	$\frac{3\cap 13\cap 1}{15}=\frac{1}{15}$
1	1	0	$\frac{3}{15}$	$\frac{2}{15}$	$(1-\frac{1}{15})$	$\frac{3\cap 2\cap 14}{15}=\frac{2}{15}$
1	1	1	$\frac{3}{15}$	$\frac{2}{15}$	$\frac{1}{15}$	$\frac{3\cap 2\cap 1}{15}=\frac{0}{15}$

Fig. 10. Probability of each minterm for 3-input functions assuming $a = 1/15, b = 2/15, c = 3/15$.

rate (ϵ) is defined as the probability of $S \neq S'$ and $L = L'$, and represented as $\mathrm{pr}(S \neq S' \cap L = L')$.

Both S and S' are N-input M-output networks. Since an N-input network has 2^{2^N} distinct functions, and S and S' can be any one of them,[1] we have

$$\mathrm{pr}(S = S') = \frac{1}{2^{2^N}}. \tag{2}$$

Both L and L' are R-input M-output networks. Since L and L' are transformed from S and S', the number of distinct functions of L and L' are not always 2^{2^R}. This number is determined by the selection of the output probabilities of RPG. We use Example 3.2 to explain this point.

1) Example 3.2: Assume the RPG is a 2-input 3-output circuit, and aliasing-free assignments 1/3 and 1/5 are assigned to RPG. We randomly select three output probabilities among $0/15 \sim 15/15$, e.g., 1/15, 2/15, and 3/15. Then, the probability of each minterm in S is as shown in Fig. 10. According to Fig. 10, we observe that only four probability values 0/15, 1/15, 2/15, and 12/15 appear as shown in the last column. Thus, the probabilities of networks are only these values or the summation of their subset. Consequently, some output probabilities of this 3-input network would not happen, such as 4/15, 5/15, 6/15, 7/15, 8/15, 9/15, 10/15, and 11/15. This implies that these 2-input networks L and L' can only have 8 distinct functions rather than 16 ($2^{2^2} = 16$) functions under these probability selections, 1/15, 2/15, and 3/15.

We denote the number of non zero probability minterms as $|C|$. For example, the probability 1/15, 2/15, and 12/15 in Fig. 10 are non zero probabilities, and $|C| = 3$. Note that these non zero probabilities will not repeat, since they come from minterms with different phases. Therefore, there are $2^{|C|}$ distinct functions of L and L' rather than 2^{2^R}. This is stated in Theorem 1.

Theorem 1: Given two N-input networks S and S', S and S' are transformed into R-input networks L and L' by L-transformation. If there are $|C|$ non-zero probability minterms, the number of distinct functions of networks L and L' is $2^{|C|}$.

[1]Even with structure similarity between two DUVs, a single error or multiple errors induced to S' could be in everywhere and cause S' be one of any possible functions. It is a risk in verification if we exclude the possibility that certain erroneous circuits occur. Thus, both previous probabilistic approaches [10], [16] are based on the same principle to analyze the aliasing rate.

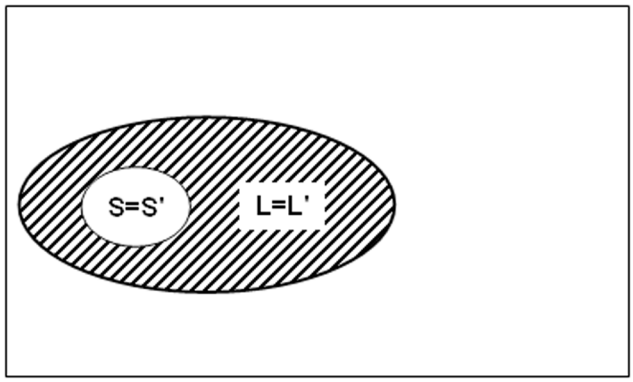

Fig. 11. Aliasing rate analysis.

Proof: Since there are $|C|$ non-zero probability minterms, we can choose i of them to form an output probability, $i = 0 \sim |C|$. Thus, there are $2^{|C|}$ combinations. This indicates that the number of distinct functions of networks L and L' is $2^{|C|}$. ■

According to Theorem 1, there are $2^{|C|}$ distinct functions, and L and L' can be any one of them. Thus, we have

$$\mathrm{pr}(L = L') = \frac{1}{2^{|C|}}. \tag{3}$$

If the network S is equivalent to the network S', then the network L is definitely equivalent to the network L'. This equivalence is asserted by observing the same output probability under the aliasing-free assignments at RPG's PIs and correct input/output correspondences between S and S'. This statement can be rewritten as the following one by the contrapositive law; *if the network L is not equivalent to the network L', then the network S is not equivalent to the network S'.* Thus, we obtain

$$\mathrm{pr}(S = S' \cap L \neq L') = 0. \tag{4}$$

Equation (4) means that the sample space of $S = S'$ is within that of $L = L'$. Their relation is illustrated in Fig. 11. The inner circle represents $S = S'$. The outer circle represents $L = L'$, and $L = L'$ completely covers $S = S'$.

R	Aliasing rate (ϵ)	
	Best case	Average case
10	10^{-308}	10^{-154}
11	10^{-616}	10^{-308}
12	10^{-1233}	10^{-616}
13	10^{-2466}	10^{-1233}
14	10^{-4932}	10^{-2466}
15	10^{-9864}	10^{-4932}

Fig. 12. Best and average case of ϵ for some R.

According to Fig. 11, the shadowed region represents the events that aliasing occurs. Thus, the aliasing rate (ϵ) defined in Definition 3, $\mathrm{pr}(S \neq S' \cap L = L')$, is obtained in

$$\epsilon = \mathrm{pr}(S \neq S' \cap L = L') = \mathrm{pr}(L = L') - \mathrm{pr}(S = S') = \frac{1}{2^{|C|}} - \frac{1}{2^{2N}}. \quad (5)$$

2) Example 3.3: Following Example 3.2, we get $N = 3$ and $|C| = 3$. Thus, $\mathrm{pr}(L = L') = 1/2^3 = 1/8$ and $\mathrm{pr}(S = S') = 1/2^{2^3} = 1/256$. Next, we examine the distribution of events in Fig. 11. The whole rectangle is the sample space and has $2^{2^3} \times 2^{2^3} = 256 \times 256 = 65\,536$ events. There are 256 ($65\,536 \times (1/256)$) events of $S = S'$ and 8192 ($65\,536 \times (1/8)$) events of $L = L'$. Thus, the shadowed region has $8192 - 256 = 7936$ events that represent aliasing.

In (5), if $|C| = 2^N$, $\epsilon = 0$; if $|C| < 2^N$, $\epsilon > 0$; and if $|C| > 2^N$, $\epsilon < 0$. However, the last case will not occur. This is because the maximal value of $|C|$ is determined by the values of R and N. If $R \leq N$, the maximal value of $|C|$ is 2^R due to an R-input network can have 2^{2^R} distinct functions at most. On the other hand, if $R > N$, the maximal value of $|C|$ is 2^N due to an N-input network can have at most 2^N minterms. Thus, we have Theorem 2.

Theorem 2: Given two N-input networks S and S', S and S' are transformed into R-input networks L and L' by L-transformation. Assume $|C|$ is the number of non-zero probability minterms in the L and L' networks, $|C| \leq 2^{\min\{R,N\}}$.

Proof:

(I) $R \leq N$: Since the input of RPG is aliasing-free assignment, the number of possible output probabilities (functions) at L and L' is 2^{2^R}. Thus, the number of non-zero probability minterms is 2^R at most, i.e., $|C| \leq 2^R$. Note that in this case, some different S and S' functions share the same output probability.

(II) $R > N$: Since the S and S' networks have N inputs, the number of minterms in S and S' is 2^N. If each minterm's probability is not zero and unique, the number of non-zero probability minterm is 2^N, i.e., $|C| \leq 2^N$.

By (I) and (II), $|C| \leq 2^{\min\{R,N\}}$. ■

3) Example 3.4: Assume S and S' are 3-input networks and the RPG in the PEACH has 2 PIs, i.e., $R = 2 < N = 3$. Since 2-input L network can have $2^{2^R} = 2^{2^2} = 16$ distinct functions at most, the maximal value of $|C|$ is $2^R = 2^2 = 4$. On the other hand, assume S and S' are 2-input networks and the RPG in the PEACH has three PIs, i.e., $R = 3 > N = 2$. Since 2-input S network can have $2^N = 2^2$ minterms at most, the maximal value of $|C|$ is $2^N = 2^2 = 4$.

From Theorem 2, the ϵ as shown in (5) is never negative. Furthermore, suppose that the value of $N \geq 25$,[2] the term $1/2^{2^N}$ is very close to zero and can be ignored. Thus, ϵ is only related to $|C|$. If $|C| = 2^{\min\{R,N\}}$, the ϵ is the least. We call this the best case. If $|C| = 2^{\min\{R,N\}}/2$ due to random selection on RPG's outputs, we call this the average case. Since R is configurable by designers, we show the ϵ value of these two cases for some R in Fig. 12. For example, if $R = 15$, $|C| = 2^{\min\{15,N\}} = 2^{15}$ in the best case, and ϵ is equal to $1/2^{2^{15}} = 2^{-2^{15}} \approx 10^{-9864}$. Note that Fig. 12 is a sample result of theoretical analysis on ϵ. The actual value of ϵ in the experiments will be reported in Section IV.

Since the networks S and S' have M outputs, each output has its own aliasing rate. Assume ϵ_i is the aliasing rate of the ith PO. The probability that aliasing does not occur in the whole network is $(1-\epsilon_1) \times (1-\epsilon_2) \times \cdots \times (1-\epsilon_M)$. Thus, we can obtain the overall aliasing rate shown in

$$\epsilon = 1 - (1-\epsilon_1) \times (1-\epsilon_2) \times \cdots \times (1-\epsilon_M). \quad (6)$$

When $R \geq 15$, ϵ_i will approach to zero, and $\Pi(\epsilon_i)$ is even smaller. Thus, (6) can be approximately rewritten as

$$\epsilon = \Sigma(\epsilon_i), i = 1 \sim M. \quad (7)$$

Equation (7) takes the summation of ϵ_i as the aliasing rate and this value will be reported in our experimental results.

Due to the manner of randomly selects RPG's output probabilities as the input probabilities of S and S', one may think our approach is random simulation-like. This is true only to some extent. The major difference between random simulation and ours is that we apply a random circuit generated by RPG instead of a random vector. This difference significantly enhances the efficiency and effectiveness of our approach as seen in the experimental results.

IV. Experimental Results and Analysis

Since previous probabilistic approaches [10], [16] report the experimental results on BDD size rather than aliasing rate and error detectability, we cannot compare with them directly. Thus, we design five experiments to demonstrate the effectiveness and efficiency of our probabilistic approach. The experiments are conducted within SIS [21] environment based on a preliminary implementation on a 1280 MHz Sun Blade 2500 workstation with 4 GB memory. The benchmarks are in BLIF format. Since we assume DUVs only consist of AND, OR, and NOT gates, we decompose complex gates in the benchmark into these primitive gates by mapping to the SIS library (22–1.genlib). We also use a free library GMP[3] to deal with the operation on large numbers needed in the program.

1) Experiment I: This experiment verifies two equivalent networks for a set of ISCAS'85 benchmarks. It is used to demonstrate the efficiency of our approach. We apply two restructuring methods on original circuits, *script.rugged* script and *red_removal* command in SIS. We compare our approach

[2]When $N < 25$, the exact probability approach can be applied.

[3][Online]. Available: http://www.swox.com/gmp/

TABLE I
COMPARISON AMONG [6], [20], AND OUR APPROACH FOR SCRIPT.RUGGED CIRCUITS

Circuit	Size	Original V.S. script.rugged circuits			
		Ours		SAT[6]	SAT+BDD[20]
		ϵ	Time(s)		
C432	286	10^{-9154}	0.065	2.14	1.47
C499	567	10^{-9822}	0.116	0.92	3.6
C880	423	10^{-9807}	0.111	–	2.72
C1355	682	10^{-9819}	0.123	1.1	6.95
C1908	770	10^{-9795}	0.120	5.9	10.2
C2670	1076	10^{-6197}	0.378	4.93	–
C3540	1530	10^{-9370}	0.259	20.98	72.9
C5315	2447	10^{-6167}	0.565	27.45	8.67
C6288	**3540**	10^{-7644}	**0.729**	**14.52**	**2582**
C7552	3281	10^{-8495}	0.754	35.18	31.8

TABLE II
COMPARISON AMONG [6], [20], AND OUR APPROACH FOR IRREDUNDANT CIRCUITS

Circuit	Size	Original V.S. irredundant circuits			
		Ours		SAT[6]	SAT+BDD[20]
		ϵ	Time(s)		
C432	286	10^{-9136}	0.068	0.7	0.65
C499	567	10^{-9819}	0.117	1.17	1.77
C880	423	10^{-9784}	0.113	–	0.19
C1355	682	10^{-9818}	0.134	2.37	3.16
C1908	770	10^{-9794}	0.139	3.87	4.27
C2670	1076	10^{-7674}	0.418	4.46	–
C3540	1530	10^{-9388}	0.282	38.94	87.2
C5315	2447	10^{-6013}	0.635	6.96	22.4
C6288	3540	10^{-7643}	0.734	5.04	61.2
C7552	3281	10^{-8639}	0.864	23.11	16.7

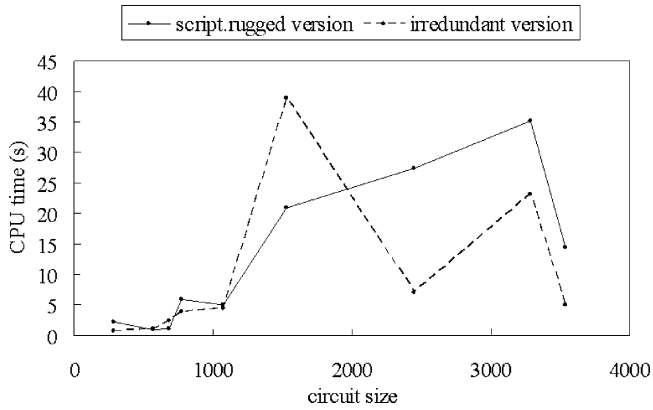

Fig. 13. CPU time (seconds) versus circuit size in SAT [6] approach.

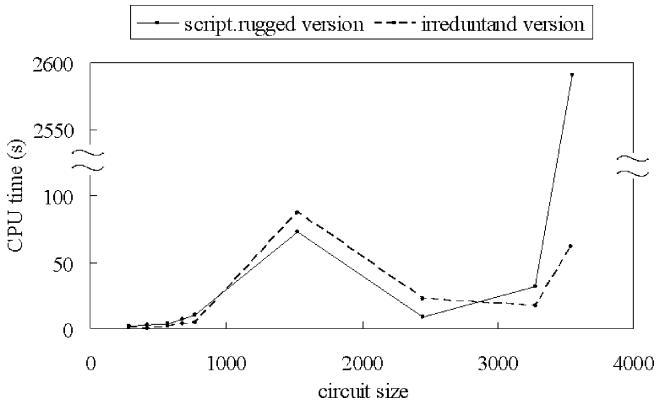

Fig. 14. CPU time (seconds) versus circuit size in SAT+BDD [20] approach.

with two state-of-the-art approaches [6], [20]. The experiment of the SAT-based approach [6] was conducted on a 167 MHz Sun Ultra Sparc-1 workstation with 256 MB memory. The experiment of SAT+BDD approach [20] was conducted on a 360 MHz Sun Ultra 10 workstation with 256 MB memory. Since we do not have the same machine nor have the codes used in [6] and [20], we conduct this experiment on a machine with a similar performance, 360 MHz Sun Ultra-60 workstation with 1 GB memory instead of Sun Blade 2500. The number of inputs in our PEACH architecture R is set to 15. The experimental results are summarized in Tables I and II. Column 1 lists the benchmarks. Column 2 shows the number of gates in a benchmark. Columns 3 and 4 show the aliasing rate (ϵ) and CPU time of our approach. The ϵ is calculated offline using (7) described in Section III-B. Columns 5 and 6 show the CPU time reported in [6] and [20]. For example, in Table I, the C6288 benchmark has 3540 gates, which our approach takes 0.729 second to verify with ϵ about 10^{-7644}. However, [6] and [20] spend 14.52 and 2582 s to verify them. Since the platforms are different, a direct exact value-to-value comparison is neither necessary nor intended. However, according to Tables I and II, it can be seen that our approach indeed efficiently reaches a virtually zero aliasing rate.

Next, we analyze CPU time among these three approaches. It can be seen that the CPU time of [6] and [20], shown in Tables I and II, strongly depend on the circuit's functionality and structure. For example, in [6], C5315 requires more CPU time than C3540 on the *script.rugged* network in Table I ($27.45 > 20.98$). But C5315 costs less CPU time than C3540 on the irredundant version in Table II ($6.96 < 38.94$). Similarly, [20] has the same phenomenon. Thus, the CPU time varies from circuit to circuit and cannot be well estimated using these state-of-the-art approaches. On the contrary, our approach is irrelevant to the circuit's functionality and structure. The CPU time of our approach is influenced by two factors. The first and the most important one is the number of gates in a benchmark. Since our probability calculation process is single-pass, more CPU time are needed for circuits with more gates. Another factor is the selection of the output probability from RPG. Since we randomly select the output probability from RPG, more CPU time is required when the numerator of the selected probability is larger. Nevertheless, our CPU time is nearly proportional to the circuit size, as shown in Tables I and II. Figs. 13–15 illustrate the CPU time against circuit size for ISCAS'85 benchmarks by [6], [20], and our approach, respectively. The solid line represents the results using the *script.rugged* version benchmark, while the dotted line represents the results using the irredundant version benchmark. It can be seen that these two lines similarly behaved in Fig. 15. On the contrary, in Figs. 13 and 14, these two lines have various traces, and the same benchmark with different structures results in very large CPU time variance. These results obviously indicate the CPU time of [6] and [20] strongly depends on the circuit's functionality and structure, and the CPU time of our approach is more predictable than that of [6] and [20].

Furthermore, the aliasing rates reported in Tables I and II are virtually zero. Refer to Fig. 16, the aliasing rate represents the

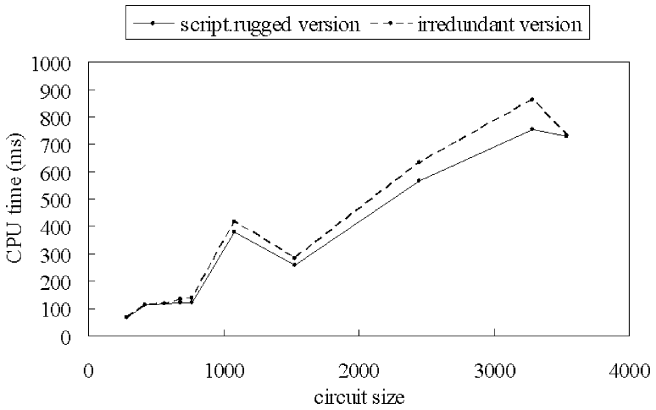

Fig. 15. CPU time (milliseconds) versus circuit size in our approach.

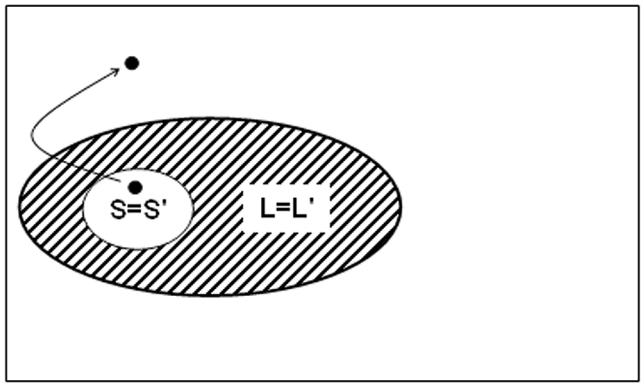

Fig. 16. Illustration that an induced error in S' very often escapes from $S = S'$ to the outside of $L = L'$.

proportion of the shadowed region to the whole rectangle. The rectangle is the sample space, and the shadow ($S \neq S' \cap L = L'$) represents the events that aliasing occurs. The virtually-zero aliasing rate in Tables I and II indicate that the shadowed region relative to the rectangle is very insignificant. As a result, if an error is induced to S' ($S \neq S'$), then it is highly likely that this event $S \neq S'$ will fall into the outside of $L = L'$ and can be easily detected. We will show this in Experiment II.

2) Experiment II: This experiment verifies two nonequivalent but *structurally similar* networks where one network contains a randomly injected error. This experiment is used to demonstrate the effectiveness of our approach on error detection. Since an erroneous network with multiple errors can be more easily detected in practice than that with single error, this experiment focuses on single error injection. We randomly inject a single error into the original network with various error types [1], [17], e.g., gate replacement, extra/missing wire, extra/missing gate, and etc. These believed to be common error types are divided into four categories in the experiment. We set $R = 15$ in our PEACH architecture and repeat the experiment 100 times for each benchmark. Table III summaries the experimental results. Columns 2–5 show the number of cases with various error categories in 100 experiments. Column 6 shows the number of cases in which the injected error is detected. Column 7 is the average CPU time measured in second for one case. Take C432 as an example, the number of cases in which randomly injected errors are WG, EW/MW, WI, and EG/MG are 27, 21, 19, and 33, respectively. These erroneous networks are all successfully detected and the average CPU time is 0.026 s. For C2670, however, one erroneous case is not detected by our approach, i.e., aliasing occurs. It is a possible outcome due to probability nature of our approach. Nevertheless, according to Table III, most of injected errors are efficiently detected.

3) Experiment III: This experiment verifies two nonequivalent networks without structural similarity, especially multipliers. In practice, CEC approaches are computationally intensive on verifying two multiplier circuits. That means the approaches are functionally sensitive. For BDD-based CEC approach, it is especially so. Furthermore, structural similarity between two networks is beneficial to performing equivalence checking and was used in many CEC approaches. Thus, the performance of known approaches strongly depend on the network's functionality and structure as being seen in Figs. 13 and 14 in Experiment I. However, if two networks are structurally dissimilar, more computation is required and hence, downgrades the performance or even fails to compare. Thus, this

TABLE III
EXPERIMENTAL RESULTS OF OUR APPROACH ON ERROR DETECTION

Circuit	# of erroneous cases with various error categories				# of detected cases	Average time (s)
	WG	EW/MW	WI	EG/MG		
C432	**27**	**21**	**19**	**33**	**100**	**0.026**
C499	30	16	20	34	100	0.046
C880	26	29	22	23	100	0.045
C1355	21	30	28	21	100	0.049
C1908	26	24	27	23	100	0.050
C2670	25	24	25	26	99	0.144
C3540	27	27	22	24	100	0.098
C5315	26	25	23	26	100	0.190
C6288	29	24	24	23	100	0.225
C7552	26	18	24	32	100	0.246

WG: wrong gate, contains gate replacement, extra inverter and missing inverter.
EW/MW: extra wire or missing wire.
WI: wrong input.
EG/MG: extra gate or missing gate.

TABLE IV
COMPARISON OF OUR APPROACH AND LEC [28] ON ERRONEOUS MULTIPLIERS

Error type	m32x32		m64x64	
	Time (s)		Time (s)	
	Ours	LEC	Ours	LEC
WG	1.588	8.7	11.114	25.86
EW	2.079	8.46	10.808	25.91
MW	1.824	8.62	11.403	25.51
WI	1.535	8.52	10.144	29.7
EG	1.367	8.69	10.191	25.92
MG	1.252	8.57	11.920	25.59
Average	1.608	8.59	10.93	26.42
Ratio	0.187	1	0.414	1

TABLE V
COMPARISON OF OUR APPROACH AND LEC [28] ON STRUCTURALLY DISSIMILAR MULTIPLIERS

Circuit	Ours		LEC
	Time (s)	ϵ	Time (s)
C6288	0.550	10^{-7638}	4.99
m16x16	0.510	10^{-7619}	456.74

TABLE VI
COMPARISON OF OUR APPROACH AND RANDOM SIMULATION ON ERRONEOUS CIRCUITS

Circuit	Size	|PI|	Ours		Random simulation	
			# of simulations	Time (s)	# of simulations	Time (s)
C432	286	36	1	0.026165	10	0.006560
C499	567	41	1	0.046062	3	0.003865
C880	423	60	1	0.141240	17	0.018334
C1355	682	41	1	0.185295	5	0.007786
C1908	770	33	1	0.196925	2261	3.765099
C2670	**1076**	**233**	**1**	**0.147994**	**661444**	**$>$ 1 hr**
C3540	1530	50	1	0.091564	1	0.003843
C5315	**2447**	**178**	**1**	**0.205292**	**453309**	**$>$ 1 hr**
C6288	3540	32	1	0.211216	1	0.010235
C7552	**3281**	**207**	**1**	**0.264952**	**323816**	**$>$ 1 hr**

TABLE VII
STATISTICS OF ALIASING IN ONE MILLION TRIALS USING DIFFERENT R

|PI| in PEACH	C2670 (138^{th} PO)		i3 (2^{th} PO)		C3540 (20^{th} PO)		C6288 (32^{th} PO)	
	aliasing	detected	aliasing	detected	aliasing	detected	aliasing	detected
R=7	425780	574220	429447	570553	80140	919860	3624	996376
R=8	177975	822025	177537	822463	6141	993859	0	1000000
R=9	32029	967971	33210	966790	3	999997	0	1000000
R=10	1182	998818	901	999099	0	1000000	0	1000000
R=11	0	1000000	0	1000000	0	1000000	0	1000000

experiment would like to show the functional independence and structural independence of our probabilistic approach. We compare our approach with a commercial tool, Cadence LEC [28], on two multiplier benchmarks, 32×32-bit and 64×64-bit multipliers. Given these two multipliers, m32 $\times$ 32 and m64 $\times$ 64, described in RT level, we create the netlist in different ways such that their structures are dissimilar. We use Synopsys DesignWare [27] to get the first network, the second network is from a cascade multiplier generator.[4] Then we inject a random error into the fan-in cone of the last output function, i.e., the 64th output in m32 $\times$ 32 and the 128th output in m64 $\times$ 64, of the second network. We use a command *compare* in LEC to verify these two networks with the argument *–noneq_stop 1*. That makes LEC terminate the program when detecting a pair of PO nonequivalent. Verification engineers often set this argument for the efficiency of verification.

Then, these two networks (one contains an error) are also translated from Verilog format to BLIF format using an internal program for our approach. The experimental results are summarized in Table IV. For each multiplier, we repeat the experiment six times with various error injection. According to Table IV, both LEC and our approach successfully determine the nonequivalence of these two networks. But our approach costs less CPU time. The ratio of average CPU time *Ours/LEC* are 18.7% and 41.4% for m32 $\times$ 32 and m64 $\times$ 64, respectively.

Next, we demonstrate that LEC is very sensitive to the structure of networks being compared while our approach is not. We use functionally equivalent 16×16-bit multipliers, C6288 and m16 $\times$ 16 as benchmarks. For C6288, the second network is restructured using *script.rugged* script in SIS. For m16 $\times$ 16, the first network is obtained by DesignWare, and the second network is obtained by a cascade multiplier generator. It can be seen that in Table V, the CPU time in the LEC column for these two cases are very different, *C*6288 only needs 4.99 s, while m16 $\times$ 16 needs 456.74 s. For our approach setting $R = 15$, however, the CPU time are very close.

[4][Online]. Available: http://www.bearcave.com/cae/cascade_mult.html/

4) Experiment IV: As mentioned in Section III-B, our probabilistic approach randomly select RPG's output probabilities as the input probabilities of DUVs. Thus, the nature of our approach includes randomness. In this experiment, we would like to demonstrate our enhancement on steadiness of general random simulation algorithms, i.e., the probabilistic approach is useful to most circuits while random simulation is only suitable for smaller circuits.

We implement a general random simulation program by modifying our probabilistic implementation. The benchmarks are ISCAS'85 with a randomly injected single error (wrong gate). The probabilistic approach ($R = 15$) and random simulation approach will be terminated if CPU time exceeds 1 hr (*abort*) or successfully detect the error. The experimental results are summarized in Table VI. Column 2 shows the number of gates in a benchmark. Column 3 is the number of PIs in a benchmark. Columns 4 and 6 show the number of simulations for detecting the error in probabilistic approach and random simulation approach, respectively. Columns 5 and 7 are the corresponding CPU time measured in seconds. According to Table VI, we can see that for the benchmarks with smaller $|\mathrm{PI}|$, the random simulation is efficient to detect the error. But for the benchmarks with larger $|\mathrm{PI}|$, e.g., C2670, C5315, and *C*7552, the CPU time exceeds 1 hr. Thus, random simulation is favorable for smaller circuits. On the contrary, the probabilistic approach detects the errors in one probability simulation for all benchmarks and the CPU time is less than 1 s. Thus, probabilistic approach is better at detecting errors than random simulation.

5) Experiment V: Equation (7) shows a theoretical analysis on aliasing rate of our approach. Although the reported aliasing rate is virtually-zero, designers may still hesitate to use this approach for CEC problem. Thus, in this experiment, we would like to show the statistics of aliasing from the aspect of experiment. The benchmarks are certain output functions from

some sample circuits. For example, the first benchmark shown in Table VII is the 138th output of C2670 circuit. We inject a random error into the function. Then, we repeat one million trials under a predefined R in the PEACH architecture, and record the number of trials that aliasing occurs. For example, in C2670 circuit, when R is set to 10, 1182 out of one million trials cause aliasing. When R is set to 11, none of trials results in aliasing. For other benchmarks in Table VII, similar results are obtained. We observe that for these four sample functions, they reach 0 aliasing in different R values. The factor influences this result is the magnitude of signal correlation in a network. If signal correlation is strong in a network, smaller R is able to reach 0 aliasing. If signal correlation is weak in a network, R could be higher for reaching 0 aliasing. In general, if R is set larger, the aliasing is infrequent. Note that the benchmark circuits in Table VII are single output functions, for multiple output functions, the error effect could be propagated to more POs and thus could reduce the number of aliasing cases. According to the statistics in Table VII, we suggest that R should be set to more than or equal to 15 for minimizing aliasing. This suggestion also explains why we set $R = 15$ in Experiment I $\sim$ III, and the injected errors are successfully detected for most cases.

In summary, our approach has the following features learned from the experiments.

1) It is a functionally and structurally independent approach.
2) It is a more efficient approach than the state-of-the-art approach, LEC, and random simulation. The CPU time is nearly proportional to the circuit size.
3) The aliasing rate is virtually-zero and hence it has a good detectability on errors.
4) The number of input in the PEACH, R, influences the occurrence of aliasing. Large R minimizes the possibility of aliasing. $R \geq 15$ is our recommendation.

As compared to exact approaches, our approach has a shortcoming that it cannot guarantee the absence of errors when the output probabilities of two networks are identical.

V. Conclusion

Non-zero aliasing rate is a major concern in probabilistic combinational equivalence checking, leading to the limited application of the method in the last decade. In this paper, we present a novel verification architecture PEACH, such that equivalence checking is efficiently performed with a virtually zero aliasing rate. The approach can be applied in two scenarios based on its features. First, since it is very efficient and has a good detectability on errors, we can use it as a preprocess for error detection. If no error is detected, we then further apply exact methods for equivalence checking. Second, for complex circuits that cannot be solved by exact methods, we can apply our approach to them. It is possible that our approach can report the inequivalence of DUVs, or give a positive answer with a very high confidence level. Thus, our probabilistic approach can efficiently complement exact methods to improve the efficiency and effectiveness of CEC algorithms.

References

[1] M. S. Abadir, J. Ferguson, and T. E. Kirkland, "Logic design verification via test generation," *IEEE Trans. Computers*, vol. 7, no. 1, pp. 138–148, Jan. 1988.

[2] V. D. Agrawal and D. Lee, "Characteristic polynomial method for verification and test of combinational circuits," in *Proc. Int. Conf. VLSI Des.*, 1996, pp. 341–342.

[3] D. Brand, "Verification of large synthesized designs," in *Proc. Int. Conf. Comput.-Aided Des.*, 1993, pp. 534–537.

[4] R. E. Bryant, "Graph-based algorithms for Boolean function manipulation," *IEEE Trans. Computers*, vol. 35, no. 8, pp. 677–691, Aug. 1986.

[5] R. E. Bryant, "Binary decision diagrams and beyond: Enabling technologies for formal verification," in *Proc. Int. Conf. Comput.-Aided Des.*, 1995, pp. 236–243.

[6] E. I. Goldberg, M. R. Prasad, and R. K. Brayton, "Using SAT for combinational equivalence checking," in *Proc. Des., Autom. Test Euro. Conf.*, 2001, pp. 114–121.

[7] A. Hett, R. Drechsler, and B. Becker, "Fast and efficient construction of BDDs," in *Proc. Des., Autom. Test Euro. Conf.*, 1997, pp. 677–691.

[8] I. Hamzaoglu and J. H. Patel, "New techniques for deterministic test pattern generation," in *Proc. VLSI Test Symp.*, 1998, pp. 446–452.

[9] A. J. Hu, "Formal hardware verification with BDDs: An introduction," in *Proc. Pacific Rim Conf. Commun., Comput., Signal Process.*, 1997, pp. 677–682.

[10] J. Jain, J. Bitner, D. S. Fussell, and J. A. Abraham, "Probabilistic design verification," in *Proc. Int. Conf. Comput.-Aided Des.*, 1991, pp. 468–471.

[11] J. Jain, A. Narayan, M. Fujita, and A. Sangiovanni-Vincentelli, "Formal verification of combinational circuits," in *Proc. Int. Conf. VLSI Des.*, 1997, pp. 218–225.

[12] S. K. Kumar and M. A. Breuer, "Probabilistic aspects of Boolean switching functions via a new transform," *J. ACM*, pp. 502–520, Jul. 1981.

[13] T. Kutzschebauch and L. Stok, "Congestion aware layout driven logic synthesis," in *Proc. Int. Conf. Comput.-Aided Des.*, 2001, pp. 216–223.

[14] T. Larrabee, "Test pattern generation using Boolean satisfiability," *IEEE Trans. Comput.-Aided Des. Integr. Circuits Syst.*, vol. 11, no. 1, pp. 4–15, Jan. 1992.

[15] S. Malik, A. Wang, R. K. Brayton, and A. Sangiovanni-Vincentelli, "Logic verification using binary decision diagrams in a logic synthesis environment," in *Proc. Int. Conf. Comput.-Aided Des.*, 1988, pp. 6–9.

[16] C. Meinel and H. Sack, "$\oplus$-OBDDs – A BDD structure for probabilistic verification," in *Proc. Workshop Probab. Methods Verif.*, 1998, pp. 141–151.

[17] D. Nayak and D. M. H. Walker, "Simulation-based design error diagnosis and correction in combinational digital circuits," in *Proc. VLSI Test Symp.*, 1999, pp. 70–78.

[18] K. P. Parker and E. J. McCluskey, "Probabilistic treatment of general combinational networks," *IEEE Trans. Comput.*, vol. C-24, no. 6, pp. 668–670, Jun. 1975.

[19] V. Paruthi and A. Kuehlmann, "Equivalence checking combining a structural SAT-solver, BDD, and simulation," in *Proc. Int. Conf. Comput. Des.*, 2000, pp. 459–464.

[20] S. Reda and A. Salem, "Combinational equivalence checking using Boolean satisfiability and binary decision diagrams," in *Proc. Des., Autom. Test Euro. Conf.*, 2001, pp. 122–126.

[21] E. M. Sentovich, K. J. Singh, L. Lavagno, C. Moon, R. Murgai, A. Saldanha, H. Savoj, P. R. Stephan, R. K. Brayton, and A. Sangiovanni-Vincentelli, "SIS: A system for sequential circuit synthesis," Electronics Research Lab, Univ. of California, Berkeley, Tech. Rep. UCB/ERL M92/41, 1992.

[22] S. C. Seth and V. D. Agrawal, "A new model for computation of probabilistic testability in combinational circuits," *Integr., VLSI J.*, vol. 7, pp. 49–75, 1989.

[23] P. Stephan, R. K. Brayton, and A. Sangiovanni-Vincentelli, "Combinational test generation using satisfiability," *IEEE Trans. Comput.-Aided Des. Integr. Circuits Syst.*, vol. 15, no. 9, pp. 1167–1176, Sep. 1996.

[24] M. Teslenko, E. Dubrova, and H. Tenhunen, "Computing a perfect input assignment for probabilistic verification," in *Proc. SPIE*, 2005, pp. 929–936.

[25] A. Veneris, A. Smith, and M. S. Abadir, "Logic verification based on diagnosis technique," in *Proc. Asia South Pacific Des. Autom. Conf.*, 2003, pp. 538–543.

[26] S.-C. Wu, C.-Y. Wang, and J.-A. Hsieh, "The potential and limitation of probability-based equivalence checking," in *Proc. Asian Test Symp.*, 2006, pp. 103–108.

[27] Synopsys, Mountain View, CA, "DesignWare," 2004 [Online]. Available: http://www.synopsys.com/

[28] Cadence, San Jose, CA, "LEC 5.1," [Online]. Available: http://www.cadence.com/

Shih-Chieh Wu received the B.S. degree in computer science and engineering from Yuan Ze University, Taiwan, R.O.C., in 2004, and the M.S. degree in computer science from National TsingHua University, Taiwan, R.O.C., in 2006.

He is currently a Military Police with the Ministry of National Defense, R.O.C. His research interests include logic synthesis, design verification, and VLSI testing.

Chun-Yao Wang (S'00-M'03) received the B.S. degree in electronics engineering from National Taipei University of Technology, Taiwan, R.O.C., in 1994, and the Ph.D. degree in electronics engineering from National Chiao Tung University, Taiwan, R.O.C., in 2002.

Since 2003, he has been an Assistant Professor with the Computer Science Department, National TsingHua University, Taiwan, R.O.C. His research interests include logic synthesis, design verification, and VLSI testing.

Yung-Chih Chen received the B.S. and M.S. degrees in computer science from National TsingHua University, Taiwan, R.O.C., in 2003 and 2005, respectively, where he is currently pursuing the Ph.D. degree in computer science.

His research interests include logic synthesis and design verification.