PEACH: A Novel Architecture for Probabilistic Combinational Equivalence Checking *

Shih-Chieh Wu and Chun-Yao Wang
Department of Computer Science
National Tsing Hua University, HsinChu, Taiwan R.O.C.
Email: {mr934359, wcyao}@cs.nthu.edu.tw

Abstract

This paper describes an approximate approach for combinational equivalence checking. We propose an architecture such that a virtually-zero aliasing rate is obtained in a single-pass probability calculation. Furthermore, the aliasing rate can be easily configured in various precision by designers. We conduct experiments on a set of ISCAS'85 benchmarks. Experimental results show that with virtually-zero aliasing rate, for example, 10^{-74}*, our approach is more efficient than those exact approaches.*

1.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 getting popular. It is possible to guarantee the equivalence of two networks by using these formal methods.

Existing exact approaches to formally verify the equivalence of two networks can be classified into four categories [10]: (1) ATPG based [3] [20], (2) BDD based [4] [7] [14], (3) SAT based [6] [16], (4) probability based [9] [21]. ATPG based methods identify some internal gates of two networks and use them to construct a *miter* structure [3] as shown in Fig.1. It examines if the output of the miter stuck-at-0 fault is untestable by Automatic Test Pattern Generation (ATPG) [8]. If the fault is untestable, there does not exist a pattern to distinguish the two logic cones. Hence, these internal gates are equivalent. Then, one internal gate can be replaced by its equivalent gate and the overall network is simplified. The efficiency of this approach relies on the capability of ATPG. If the fault test at the miter output is time-consuming or intractable, the approach becomes inefficient.

Reduced Ordered Binary Decision Diagrams (ROBDDs) [4] is a canonic representation to represent networks. Thus, two networks are equivalent if and only if the ROBDDs are isomorphic [1] [5]. The possible difficulty BDD based approach encounters is ROBDDs construction. Certain circuits such as the multiplier with numerous inputs cannot be represented by ROBDDs in any variable ordering [4].

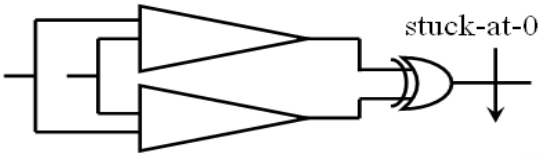

Figure 1: A miter.

Recently, Boolean Satisfiability (SAT) has been successfully used as an efficient and complete method for solving CEC problems [16] [18]. 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 [13]. However, the problem is that the SAT based approach sometimes requires large amount of time and backtracks to prove the network[16].

The signal probability at the output of a network was considered as a signature function [2] [9] for CEC problem. Signature function is used to characterize networks' properties, e.g., the number of minterms in the on-set of a network is a signature function. The probability signature, however, can be exact as well if the input probability is an aliasing-free assignment. The exact probability based approach assigns aliasing-free assignments [21] at all primary inputs (PIs) of networks, then the output probability is derived by a probability calculation process from PIs towards primary outputs (POs). Two networks are equivalent if and only if their output probabilities are equal under the aliasing-free assignments.

These exact approaches have an obvious feature in common, i.e., if two networks are too complex to be handled by these approaches, designers have no answer about the equivalence of them. However, for those complex circuits, designers would still like to know the equivalence of them, even reports an answer with a tolerance. To the best of our knowledge, approximate probability based approach, named probabilistic approach [10], is a suitable way to give such an answer for those large circuits. The ideal goal of probabilistic CEC is to effi-

*This work was supported in part by the National Science Council of R.O.C. under Grant NSC94-2220-E-007-041

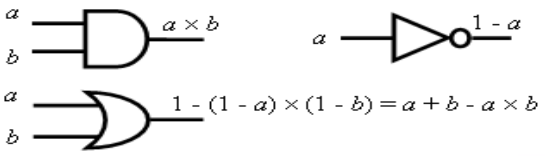

Figure 2: Probability formulae for primitive gates.

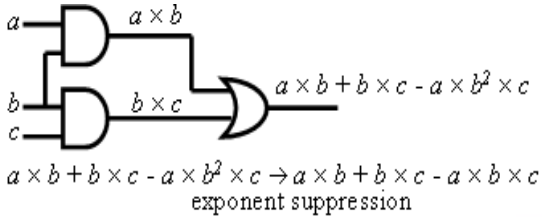

Figure 3: A demonstration of exponent suppression.

ciently obtain a virtually-zero aliasing rate.

In this paper, we propose an approximate method based on the exact probability based approach [21] for CEC problem. Our approach constructs a Probabilistic EquivAlence CHecking (PEACH) architecture such that a virtually-zero aliasing rate is obtained in a single-pass probability calculation. Furthermore, the aliasing rate can be easily configured in various precision by designers.

2. Background

This section first reviews the background of signal probability in a network. The exact probability based approach is also introduced in Section 2.2. It is the core technique of our probabilistic approach as presented in more detail in Section 3. Here we assume networks only consist of AND, OR, and NOT gates for simplicity. Complex gates can be decomposed into these gates.

2.1 Probability Expression of a Network

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

Definition 1: *Given gates s and d in the network, if there are more than one disjoint path from s to d, d is a reconvergent gate in the network* [12].

The *probability expression* of a network can be straightforwardly derived from the PIs to the POs by using these probability formulae with an exponent suppression operation. The *exponent suppression* replaces the term x^m with x for every gate X in the expression [11] [15] due to a gate X is fully correlated with itself in the reconvergent gate.

For example as shown in Fig. 3, the probability expression at the output is originally $a \times b + b \times c - a \times b^2 \times c$.

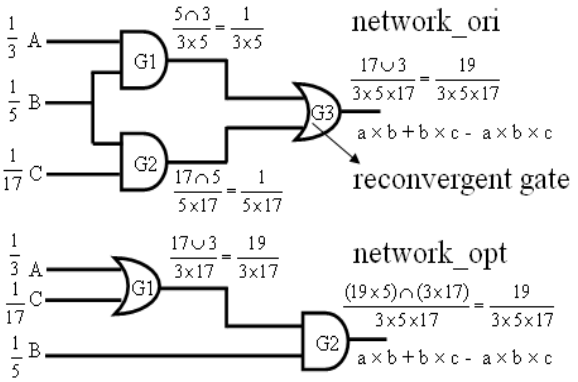

Figure 4: Output probability evaluation process.

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 [11] [15]. 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 [11] are required to get the probability expression of an n-input network. Also, the number of terms in a probability expression is 2^n in the worst case [11].

2.2 Exact Probability based Approach

The exact probability based approach assigns numeral probability to PIs and evaluates the probability at POs for comparison where the assigned probability is aliasing-free [19] [21]. Thus, it produces a unique output probability for each function.

Equation (1) is a recursive formula reported in [21] that produces aliasing-free probability assignments for an n-input network where $\frac{1}{\theta_i}$ is the 1's probability of input variable X_i, $i = 1 \sim n-1$.

$$\theta_{i+1} = (\theta_i - 1)^2 + 1 = {\theta_i}^2 - 2\theta_i + 2; \quad (1)$$
$$i = 1 \sim n-1;$$
$$\theta_1 \geq 3 \ \& \ \theta_1 \in \mathrm{Z}^+;$$

To minimize the 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 the paper. The following example demonstrates why Equation (1) results in a unique output probability for each function.

Wu et al. [21] apply these aliasing-free assignments to the DUVs, and calculate the output probability. Furthermore, the signal correlation issue at reconvergent gates is also considered. As a result, correct output probability of a network is efficiently obtained for further comparison.

Take Fig.4 as an example, we can derive the probability expression of network_ori and network_opt, both are $a \times b + b \times c - a \times b \times c$. Thus, they are equivalent networks. Following, instead of deriving proba-

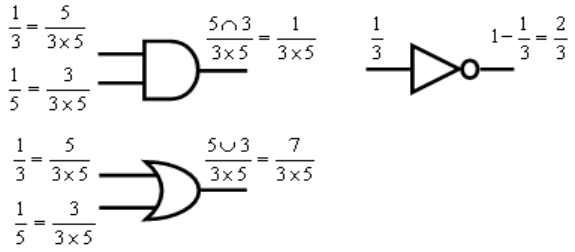

Figure 5: Alternative operations for primitive gates.

bility expression, we introduce how to verify these two networks by using aliasing-free assignments and get the output probability in a single-pass calculation. First, we assign the aliasing-free assignments, $a = \frac{1}{3}$, $b = \frac{1}{5}$, and $c = \frac{1}{17}$, to PIs A, B, and C in network_ori and network_opt, respectively. Different from original probability formulae, 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 of primitive gates are shown in Fig.5. 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 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.5, we first transform $\frac{1}{3}$ to $\frac{5}{3\times5}$ and $\frac{1}{5}$ to $\frac{3}{3\times5}$. Then, the two new numerators conduct bitwise-AND operation, $5 \cap 3 = 101_2 \cap 011_2 = 001_2 = 1$, to obtain the output probability, $\frac{1}{3\times5}$. The calculation for the other gates can also be seen in Fig.5. Applying these rules in Fig.5 to gates in the networks, we can obtain correct output probabilities even though reconvergent gates exist in the networks. For example, Fig.4 shows two networks having the same output probability, $\frac{19}{3\times5\times17}$, thus we can know they are equivalent as earlier 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. [21] report that the 24^{th} assignment, θ_{24}, $2^{2^{24-1}} + 1 \approx 2^{2^{23}} \approx 10^{2525222}$, within a single fanin cone is the maximal value it can support. Thus, this paper proposes an approximate approach for those large circuits.

3. Probabilistic CEC and Its Analysis

This section first presents a verification architecture, Probabilistic EquivAlence CHecking (PEACH), that is used for CEC and provides a configurable aliasing rate. Next, the detailed analysis on the aliasing rate with PEACH is presented.

3.1 PEACH Architecture

The PEACH architecture is shown in Fig.6. It contains three components, a random probability

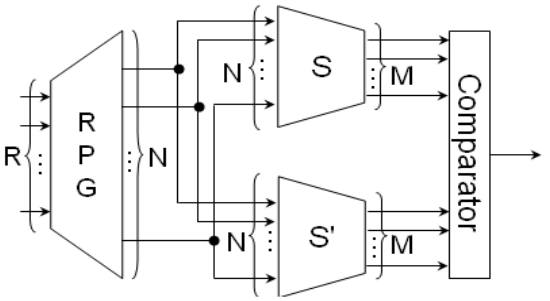

Figure 6: The PEACH architecture.

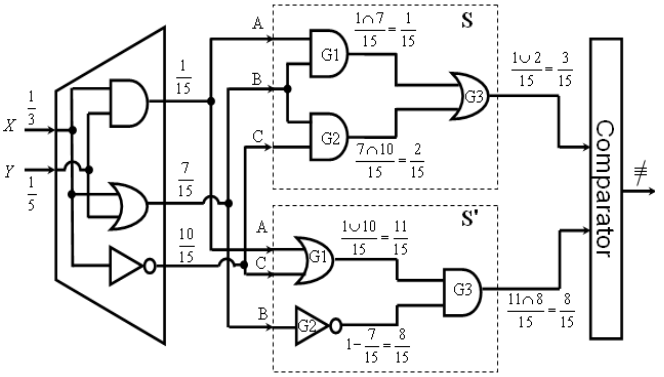

Figure 7: The illustration of Example 3.1.

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. 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 $\frac{0}{2^{2^R}-1} \sim \frac{2^{2^R}-1}{2^{2^R}-1}$ and assign them to the PIs of S and S′. Using the probability evaluation process as mentioned in Section 2.2, we can get the output probabilities of S and S′. Then, the comparator pairwisely compares 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.

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

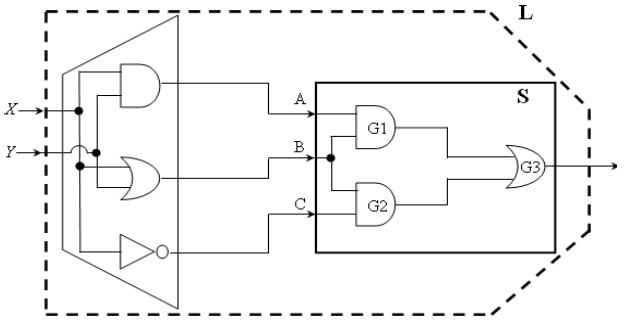

Figure 8: An example of L-transformation.

of S and S′, respectively. That is, a=$\frac{1}{15}$, b=$\frac{7}{15}$, and c=$\frac{10}{15}$. After the probability calculation, the output probability of S is $\frac{3}{15}$ and that of S′ is $\frac{8}{15}$. 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 $\frac{0}{15}$, $\frac{6}{15}$, and $\frac{0}{15}$ (repeated probability is possible), an identical output probability of S and S′, $\frac{0}{15}$, is obtained. This indicates that aliasing may occur in the PEACH architecture.

Definition 2: *Aliasing is the situation that two non-equivalent networks S and S′ having an identical output probability under the same input assignment.*

As we mentioned, a major disadvantage of aliasing-free assignments is that the assignments exponentially grow. Thus, the number of aliasing-free assignments is very limited in the exact approach in practice. Approximate approach using PEACH, however, the number of assignments can be up to 2^{2^R} at least, 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 the verification on very large circuits possible. Although PEACH architecture could cause aliasing, we observe that if R≥10, the probability that aliasing occurs is virtually-zero. Its detailed analysis will be presented in Section 3.2.

3.2 Aliasing Rate Analysis

To analyze the probability that aliasing occurs, 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. For example, network S in Fig.7 is a 3-input 1-output network. Its function is $A \cdot B + B \cdot C$. The 2-input RPG in Fig.7 performs L-transformation on S. Thus, S is transformed into a new network L as shown in Fig.8. The function of L is $(X \cdot Y) \cdot (X + Y) + (X + Y) \cdot \overline{X}$ 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

C	B	A	c	b	a	prob. of minterm
0	0	0	$(1-\frac{3}{15})$	$(1-\frac{2}{15})$	$(1-\frac{1}{15})$	$\frac{12\cap13\cap14}{15} = \frac{12}{15}$
0	0	1	$(1-\frac{3}{15})$	$(1-\frac{2}{15})$	$\frac{1}{15}$	$\frac{12\cap13\cap 1}{15} = \frac{0}{15}$
0	1	0	$(1-\frac{3}{15})$	$\frac{2}{15}$	$(1-\frac{1}{15})$	$\frac{12\cap 2\cap14}{15} = \frac{0}{15}$
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	$\frac{3}{15}$	$(1-\frac{2}{15})$	$(1-\frac{1}{15})$	$\frac{3\cap13\cap14}{15} = \frac{0}{15}$
1	0	1	$\frac{3}{15}$	$(1-\frac{2}{15})$	$\frac{1}{15}$	$\frac{3\cap13\cap 1}{15} = \frac{1}{15}$
1	1	0	$\frac{3}{15}$	$\frac{2}{15}$	$(1-\frac{1}{15})$	$\frac{3\cap 2\cap14}{15} = \frac{2}{15}$
1	1	1	$\frac{3}{15}$	$\frac{2}{15}$	$\frac{1}{15}$	$\frac{3\cap 2\cap 1}{15} = \frac{0}{15}$

Figure 9: The probability of each minterm for 3-input functions assuming $a = \frac{1}{15}$, $b = \frac{2}{15}$, $c = \frac{3}{15}$.

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 Definition 2 of aliasing, 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 rate (ϵ) is defined as the probability of S≠S′ and L=L′, and represented as pr(S≠S′ ∩ 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, we have Equation (2).

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

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 RPG's output probabilities. We use Example 3.2 to explain this point.

Example 3.2: Assume the RPG is a 2-input 3-output circuit, and aliasing-free assignments $\frac{1}{3}$ and $\frac{1}{5}$ are assigned to RPG. We randomly select three output probabilities among $\frac{0}{15} \sim \frac{15}{15}$, e.g., $\frac{1}{15}$, $\frac{2}{15}$, and $\frac{3}{15}$. Then, the probability of each minterm in S is as shown in Fig.9. According to Fig.9, we observe that only 4 probability values are appeared, $\frac{0}{15}$, $\frac{1}{15}$, $\frac{2}{15}$, and $\frac{12}{15}$ in the last column. Thus, the networks' probabilities are only these values or the summation of subset of them. Consequently, some output probabilities of this 3-input network would not happen, such as $\frac{4}{15}$, $\frac{5}{15}$, $\frac{6}{15}$, $\frac{7}{15}$, $\frac{8}{15}$, $\frac{9}{15}$, $\frac{10}{15}$, and $\frac{11}{15}$. 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 this probability selection, $\frac{1}{15}$, $\frac{2}{15}$, and $\frac{3}{15}$.

We denote the number of non-zero probability minterms as $|C|$. For example, the probability $\frac{1}{15}$, $\frac{2}{15}$, and $\frac{12}{15}$ in Fig.9 are non-zero probabilities, and $|C| = 3$. 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*

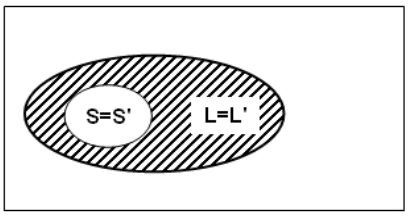

Figure 10: Aliasing rate analysis.

probability minterms, 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 Equation (3).

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

If network S is equivalent to network S′, then network L is definitely equivalent to network L′. This equivalence is asserted by observing the same output probability with 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 network L is not equivalent to network L′, then network S is not equivalent to network S′.* Thus, we obtain Equation (4).

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

Equation (4) means that the sample space of S=S′ is within that of L=L′. Their relation is illustrated in Fig.10. The inner circle represents S=S′. The outer circle represents L=L′, and L=L′ completely covers S=S′.

According to Fig.10, the shadow part represents the events that aliasing occurs. Thus, the aliasing rate (ϵ) defined in Definition 3, pr(S≠S′ ∩ L=L′), is obtained in Equation (5).

$$\begin{aligned} \epsilon &= pr(S \neq S' \cap L = L') \\ &= pr(L = L') - pr(S = S') \\ &= \frac{1}{2^{|C|}} - \frac{1}{2^{2^N}} \end{aligned} \tag{5}$$

Suppose that the value of N ≥ 25, the term $\frac{1}{2^{2^N}}$ is very close to zero and can be ignored. Thus, the ϵ is only related to $|C|$. If $|C| = 2^R$, the ϵ is the least. We call this is the best case. If $|C| = \frac{2^R}{2}$, we call this is the average case. Since R is configurable by designers, we show the ϵ of these two cases for some R in Fig.11. For example, if R=10, $|C| = 2^{10}$ in the best case. The ϵ is equal to $\frac{1}{2^{2^{10}}} = 2^{-2^{10}} \approx 10^{-308}$. Note that Fig.11 is a sample result of theoretical analysis on ϵ. The actual ϵ in the experiments will be reported in the next section.

Since networks S and S′ have M outputs, each output would have its own aliasing rate. Assume ϵ_i is the aliasing rate of the i^{th} 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 as shown in Equation (6).

$$\epsilon = 1 - (1-\epsilon_1) \times (1-\epsilon_2) \times \cdots \times (1-\epsilon_M) \tag{6}$$

When R ≥ 10, ϵ_i will approach to zero, and $\Pi(\epsilon_i)$ is even smaller. Thus, Equation (6) can be approximately rewritten as Equation (7).

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

Figure 11: The best case and average case of ϵ.

$$\epsilon = \Sigma(\epsilon_i), i = 1 \sim M \tag{7}$$

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

4. Experimental Results and Analysis

The experiments are conducted over a set of ISCAS'85 benchmarks within SIS [17] environment based on a preliminary implementation. These 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). To compare two networks with the same functionality but different structures, we restructure one network by using *script.rugged* script in SIS. Also, a free library GMP [22] is used to deal with the operation of large numbers.

The experiment compares our approach against a BDD based approach on a 1280 MHz Sun Blade 2500 workstation with 4 Gbytes memory. The number of inputs in the PEACH architecture is set to 10, i.e., R=10, and the RPG's outputs are randomly selected to connect to the inputs of S and S'. The BDD based approach is conducted by using SIS function (*ntbdd_verify_network*) with arguments DFS_ORDER and ALL_TOGETHER. Table 1 summarizes the experimental results of our approach and the BDD based approach. Column 1 lists the benchmarks, the last one (M32x32) is a 32×32 multiplier designed by us. Column 2 shows the number of gates in a benchmark. Column 3 and 4 show the CPU time and aliasing rate (ϵ) of our approach. The ϵ is off-line calculated using Equation (7) described in Section 3.2. Column 5 shows the CPU time of the BDD based approach. For example, *C*6288 benchmark has 3540 gates. Our approach spends 0.025771 seconds for verifying them with ϵ about 10^{-76}. However, the BDD based approach cannot verify them within one hour due to BDD explosion. We abort the BDD experiment when the CPU time exceeds one hour.

Next, we analyze the aliasing rate. The aliasing rate reported in the experimental results is virtually-zero. Refer to Fig.12, the aliasing rate represents the proportion of the shadow part to the whole rectangle. The rectangle is the sample space, and the shadow part is the events that aliasing occurs. The virtually-zero aliasing rate in Table 1 indicates that the shadow part

Table 1: The comparison between our approach and a BDD based approach on a 1280 MHz machine.

		Ours		BDD
Circuits	Size	Aliasing rate	Time (s)	Time(s)
C432	286	10^{-307}	0.002285	0.751328
C499	567	10^{-308}	0.004418	0.281865
C880	423	10^{-194}	0.003830	0.150609
C1355	682	10^{-308}	0.004730	0.438446
C1908	770	10^{-307}	0.005023	0.379906
C2670	1076	10^{-75}	0.013780	$>$ 1 hr
C3540	1530	10^{-75}	0.010087	25.351015
C5315	2447	10^{-134}	0.020017	0.655514
C6288	**3540**	$\mathbf{10^{-76}}$	**0.025771**	**$>$ 1 hr**
C7552	3281	10^{-75}	0.026013	$>$ 1 hr
M32x32	11648	10^{-75}	0.08765	$>$ 1 hr

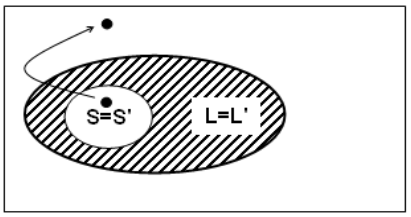

Figure 12: An illustration that an induced bug in S′ very often escapes from S=S′ to the outside of L=L′.

relative to the rectangle is very tiny. As a result, if a bug is induced to S′ (S≠S′), then it is very possible that this event S≠S′ will fall into the outside of L=L′ and can be easily detected. In addition, we can get a lower aliasing rate by increasing the number of inputs in PEACH architecture. Since this work focuses on the achievement of virtually-zero aliasing rate rather than error diagnosis, we do not report the results on verifying two circuits with different functionalities in this paper.

5. Conclusions

Non-zero aliasing rate is a major concern in probabilistic combinational equivalence checking, and causes its limited application in the last decade. In this paper, we present a novel verification architecture, PEACH, such that a virtually-zero aliasing rate is efficiently obtained. For those complex circuits which cannot be solved by formal verification methods, our approach using PEACH architecture still gives an answer with a very high confidence level. Thus, the probabilistic approach could be considered as a good alternative to combinational equivalence checking problem for larger circuits.

References

[1] J. Hu Alan, "Formal Hardware Verification with BDDs: An Introduction," in *Proc. of PACRIM*, pp. 677-682, 1997.

[2] V. D. Agrawal, et al., "Characteristic Polynomial Method for Verification and Test of Combinational Circuits," in *Proc. of Int. Conf. on VLSI Design*, pp. 341-342, 1996.

[3] D. Brand, "Verification of Large Synthesized Designs," in *Proc. of ICCAD*, pp. 534-539, 1993.

[4] R. E. Bryant, "Graph-based Algorithms for Boolean Function Manipulation," *IEEE Trans. on Computers*, pp. 677-691, Aug. 1986.

[5] R. E. Bryant, "Binary Decision Diagrams and Beyond: Enabling Technologies for Formal Verification," in *Proc. of ICCAD*, pp. 236-243, 1995.

[6] E. I. Goldberg, et al., "Using SAT for Combinational Equivalence Checking," in *Proc. of DATE*, pp. 114-121, 2001.

[7] A. Hett, et al., "Fast and Efficient Construction of BDDs," in *Proc. of the ED & TC*, pp. 677-691, 1997.

[8] I. Hamzaoglu, et al., "New Techniques for Deterministic Test Pattern Generation," in *Proc. of VTS*, pp. 446-452, 1998.

[9] J. Jain, et al., "Probabilistic Design Verification," in *Proc. of ICCAD*, pp. 468-471, 1991.

[10] J. Jain, et al., "Formal Verification of Combinational Circuits," in *Proc. of Int. Conf. on VLSI Design*, pp. 218-225, 1997.

[11] S. K. Kumar, et al., "Probabilistic Apects of Boolean Switching Functions via a New Transform," Journal of the ACM, pp. 502-520, July 1981.

[12] T. Kutzschebauch, et al., "Congestion Aware Layout Driven Logic Synthesis," in *Proc. of ICCAD*, pp. 216-223, 2001.

[13] T. Larrabee, "Test Pattern Generation Using Boolean Satisfiability," *IEEE Trans. on Computer Aided-Design*, pp. 4-15, Jan. 1992.

[14] S. Malik, et al., "Logic Verification Using Binary Decision Diagrams in a Logic Synthesis Environment," in *Proc. of ICCAD*, pp. 6-9, 1988.

[15] K. P. Parker, et al., "Probabilistic Treatment of General Combinational Networks," *IEEE Trans. on Computer*, pp. 668-670, June 1975.

[16] S. Reda, et al., "Combinational equivalence checking using Boolean satisfiability and binary decision diagrams," in *Proc. of DATE*, pp. 122-126, 2001.

[17] E. M. Sentovich, et al., "SIS: A System for Sequential Circuit Synthesis," ERL Memo. No.UCB/ERL M92/41, EECS, UC Berkeley, CA 94720.

[18] P. Stephan, et al., "Combinational Test Generation Using Boolean Satisfiability," *IEEE Trans. on Computer Aided-Design of Integrated Circuits and Systems*, Vol. 15, No. 9, Sept. 1996.

[19] M. Teslenko, et al., "Computing a Perfect Input Assignment for Probabilistic Verification," in *Proc. of SPIE*, pp. 929-936, 2005.

[20] A. Veneris, et al., "Logic Verification Based on Diagnosis Technique," in *Proc. of ASPDAC*, pp. 538-543, 2003.

[21] S.-C. Wu, et al., "Formal Combinational Equivalence Checking Using Probability," in *Proc. of SASIMI*, 2006.

[22] http://www.swox.com/gmp/