The Potential and Limitation of Probability-Based Combinational Equivalence Checking [*]

Shih-Chieh Wu, Chun-Yao Wang and Jan-An Hsieh

Department of Computer Science
National Tsing Hua University, HsinChu, Taiwan R.O.C.
Email: {mr934359, wcyao, mr924372}@cs.nthu.edu.tw

Abstract

This paper presents a probability based approach to logic equivalence checking. First, a general probability assignment procedure is proposed to uniquely characterize output probability of a network. Thus, the equivalence of two networks can be asserted by the equality of output probabilities. To improve the efficiency of probability calculation, a new encoding scheme and operations are proposed. These encoding scheme and operations also solve the signal correlation issue during the output probability evaluation. As a result, an exact output probability of a network is successfully derived in one pass. Finally, the equivalence of internal gates between two networks are exploited to reduce the number of required input assignments and improve the efficiency of our approach. In the experiments, our approach is compared with a BDD based approach in terms of CPU time and memory usage. The results disclose the potential and limitation of the probabilistic approach to logic equivalence checking.

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 logic verification methods are getting popular. It is possible to guarantee the correctness of a design by using these formal methods.

Existing approaches to formally verify the equivalence of two networks can be classified into two categories [13]: (1) structural [4] [22], (2) functional [10] [16]. Structural methods identify some internal gates of two networks and use them to construct a *miter* structure [4] 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) [11]. 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 the other 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.

On the other hand, functional methods use canonic representations to represent networks. Thus, two networks are equivalent if and only if the representations are identical. Reduced Ordered Binary Decision Diagrams (ROBDDs) [6] is one of canonic representations to represent networks. One can use ROBDDs to verify the equivalence of two networks directly [1] [7]. The possible difficulty this approach encounters is ROBDDs construction. Certain circuits such as the multiplier with numerous inputs cannot be represented by ROBDDs in any variable ordering [6].

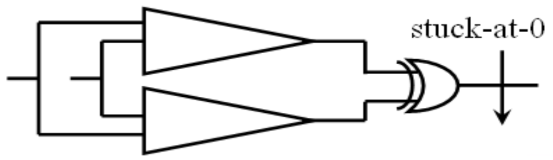


Figure 1: A miter.

The signal probability of a gate within a network has applications to power estimation [18] and testability analysis [3] [5] [8] [9]. Also, the signal probability at the output of a network was considered as a signature function [2] [12] for logic equivalence checking. Signature function is used to characterize networks' properties. If the signature values of two networks are different, the two networks are not equivalent. Otherwise, however, they are only possibly equivalent. The case that two different networks with the same signature value is called *aliasing*. When the output probabilities are not equal under the same set of input probability, the two networks are not equivalent. But the inverse is not true. That is, the aliasing could occur. Although the aliasing rate of this approach would be reduced with multiple runs of input probability assignments [12], the equality of two output probabilities still does not guarantee the equivalence of two networks. The occurrence of aliasing relies on the input probability assignments. Thus, in this paper, we propose a general probability assignment procedure to *uniquely* characterize the output probabilities of networks. Consequently, the aliasing will not occur. Thus, the equivalence of two networks is asserted by the equality of the output probabilities.

The calculation of signal probability at the gates within a network involves arithmetic operations, such as multiplication, addition, and subtraction. The more efficient operations, such as bitwise-AND (∩), bitwise-OR (∪), and shift-add operations are exploited in this paper to substitute the original arithmetic operations without sacrificing the correctness. We also propose a new encoding scheme with the bitwise operations that successfully deals with the signal correlation issue.

*This work was supported in part by the National Science Council of R.O.C. under Grant NSC95-2220-E-007-020

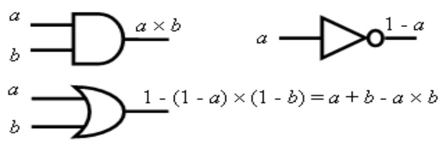


Figure 2: Probability formulae for primitive gates.

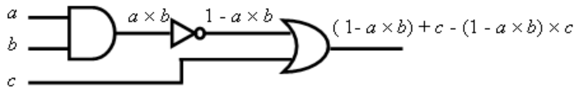


Figure 3: Probability calculation on a tree-structure network.

2. Background

This section reviews the background of signal probability in a network. Here we assume networks only consist of AND, OR, and NOT gates for simplicity. Complex gates can be decomposed into these gates. We will denote a gate in the network by an upper case letter and its probability of signal one by the corresponding lower case letter. 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* [15].

The *probability expression* of a network can be derived from the primary inputs to the primary outputs by using these probability formulae. However, this expression is correct only if the network is a tree-structure. For example, a tree-structure network is as shown in Fig. 3. Its probability expression can be obtained straightforward. If a network contains reconvergent gates, the process of deriving probability expression has to be modified due to correlated inputs. This modification is named *exponent suppression*, which replaces the term x^m with x for every gate X in the expression [14] [19]. This is because a gate X is fully correlated with itself in the reconvergent gate. After the exponent suppression, the modified probability expression is correct. For example as shown in Fig. 4, the probability expression at the output is $a \times b + b \times c - a \times b^2 \times c$ originally. After the exponent suppression modification, 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 modification is unique for a network [14] [19]. Namely, if two networks (regardless of having reconvergent gates or not) have the same probability expression after the exponent suppression modification, they are equivalent; otherwise, they are nonequivalent. Thus, the probability expression is a canonic representation.

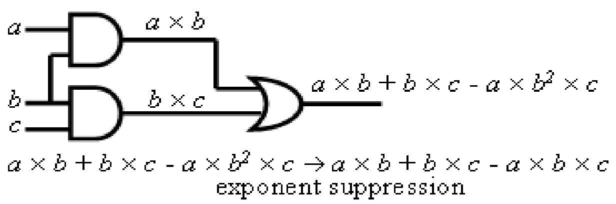

a
b
c
a × b
b × c
a × b + b × c - a × b² × c
a × b + b × c - a × b² × c → a × b + b × c - a × b × c
exponent suppression

Figure 4: Probability calculation with exponent suppression.

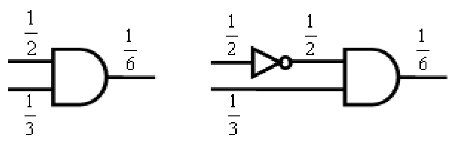


Figure 5: An example to explain the aliasing.

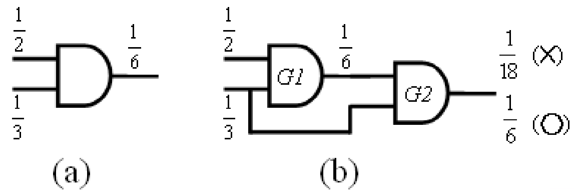


Figure 6: Probability calculation in numeral assignments (a) without signal correlation, (b) with signal correlation.

Although probability expression is a canonic representation, deriving it for a large circuit is intractable. This is because $O(n \times 2^n)$ operations [14] 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 [14].

When applying the same set of numeral assignments to the primary inputs of two networks for equivalence checking, the situation that the two different networks get the same output probability is called *aliasing*. For example in Fig. 5, the two networks are distinct but have the same output probability $\frac{1}{6}$ given input probability $\frac{1}{2}$ and $\frac{1}{3}$. Thus, $\frac{1}{2}$, $\frac{1}{3}$ is not an aliasing-free assignment.

When applying numeral assignments to the primary inputs of a network, the output probability could be erroneous if the signal correlation issue is not considered. For example, the output probability of Fig. 6(a) is $\frac{1}{2} \times \frac{1}{3} = \frac{1}{6}$ by using the AND formula. This value is correct due to the inputs are independent. However, the output probability $\frac{1}{18}$ in Fig. 6(b) is erroneous if using the same formula without considering the correlation of inputs at gate $G2$. The correct output probability in Fig. 6(b) is still $\frac{1}{6}$. Since the exponent suppression modification cannot be applied directly under numeral assignments, how to correctly calculate the signal probabilities of a network containing reconvergent gates is a challenge.

3. Probabilistic Logic Equivalence Checking

This section presents our probabilistic approach to logic equivalence checking. It consists of three parts. First of all, an aliasing-free assignment procedure is presented. Then, the probability evaluation process with aliasing-free assignment is presented. Finally, an internal tree-structure replacement method is introduced.

3.1 Aliasing-Free Probability Assignments

To ensure the equivalence of two networks after probabilities assignment, the aliasing-free probabilities assignments are crucial. Random probabilities assignments cannot guarantee uniqueness [2][11].

Given an n-input network, the number of distinct functions is 2^{2^n}. This implies that if aliasing is not permitted, the output probabilities of these functions have to be different under the same set of input probabilities.

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	$\frac{1}{17}$	$\frac{1}{5}$	$\frac{1}{3}$	$\frac{1}{255}$

Figure 7: The probability of each minterm for 3-input functions assuming $x_1 = \frac{1}{3}$, $x_2 = \frac{1}{5}$, $x_3 = \frac{1}{17}$.

To better describe the procedure of input probability assignment and explain why it results in uniqueness, we represent a logic function with its truth table. We assume 1's probability of an input variable X_i is x_i. The corresponding 0's probability of X_i is $(1 - x_i)$. Thus, for a minterm $X_n \cdot X_{n-1} \cdot \dots \cdot X_t \cdot X'_{t-1} \cdot \dots \cdot X'_1$ in an n-input function, its probability is $x_n \times x_{n-1} \times \dots \times x_t \times (1-x_{t-1}) \times \dots \times (1-x_1)$. The probability of a function is the summation of the probability of its minterms.

For an n-bit positive binary numeral system, it can represent the integers in the range of $0 \sim 2^n - 1$. Inspired by the binary numeral system, the probability value of each minterm acts as the weight of output probability of a function. If the assigned weight of each minterm is similar to that of binary numeral system, the output probability of each function is unique.

We propose a recursive function for probability assignments in an n-input network in Equation (1), which leads to aliasing-free output probabilities. The 1's probability of input variable X_i is x_i and x_i is assigned as $\frac{1}{a_i}$, where

$$\begin{aligned} a_{i+1} = (a_i - 1)^2 + 1 = {a_i}^2 - 2a_i + 2; \qquad (1) \\ i = 1 \sim n-1; \\ a_1 \geq 3 \ \&\ a_1 \in \mathrm{Z}^+; \end{aligned}$$

For example, for a 3-input function, there are $2^{2^3} = 256$ distinct functions. If we set $a_1 = 3$, $x_1 = \frac{1}{3}$; $a_2 = 5$, $x_2 = \frac{1}{5}$; and $a_3 = 17$, $x_3 = \frac{1}{17}$ according to Equation (1), the probability of each minterm is shown in Fig. 7. The probability of each minterm acts as the 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, the probability is unique for each function and they are distributed from $\frac{0}{255} \sim \frac{255}{255}$ uniformly. The uniqueness of output probability obtained by Equation (1) for $a_1 = 3$ is stated in Theorem 1.

Theorem 1: *The probability assignment in an n-input function by Equation (1) for $a_1 = 3$ results in unique output probability.*
Proof: Omitted.

Equation (1) works well for $a_1 > 3$ & $a_1 \in \mathrm{Z}^+$ as well. Fig. 8 shows the probability values for $a_1 = 4$. The numerator of probability of each minterm in Fig 8 is analogous to the ternary numeral system, hence the output probability is unique as well for each function. But this assignment causes some output probabilities not to occur, e.g., $\frac{2}{3280}$ is not a possible output probability.

X_3	X_2	X_1	x_3	x_2	x_1	prob. of minterm
0	0	0	$(1-\frac{1}{82})$	$(1-\frac{1}{10})$	$(1-\frac{1}{4})$	$\frac{2187}{3280}$
0	0	1	$(1-\frac{1}{82})$	$(1-\frac{1}{10})$	$\frac{1}{4}$	$\frac{729}{3280}$
0	1	0	$(1-\frac{1}{82})$	$\frac{1}{10}$	$(1-\frac{1}{4})$	$\frac{243}{3280}$
0	1	1	$(1-\frac{1}{82})$	$\frac{1}{10}$	$\frac{1}{4}$	$\frac{81}{3280}$
1	0	0	$\frac{1}{82}$	$(1-\frac{1}{10})$	$(1-\frac{1}{4})$	$\frac{27}{3280}$
1	0	1	$\frac{1}{82}$	$(1-\frac{1}{10})$	$\frac{1}{4}$	$\frac{9}{3280}$
1	1	0	$\frac{1}{82}$	$\frac{1}{10}$	$(1-\frac{1}{4})$	$\frac{3}{3280}$
1	1	1	$\frac{1}{82}$	$\frac{1}{10}$	$\frac{1}{4}$	$\frac{1}{3280}$

Figure 8: The probability of each minterm for 3-input functions assuming $x_1 = \frac{1}{4}$, $x_2 = \frac{1}{10}$, $x_3 = \frac{1}{82}$.

Note that $a_1 = 2$ is an infeasible assignment. This is because the 1's probability of X_1 equals its 0's probability, $x_1 = \frac{1}{2} = (1 - x_1)$. It will cause the probabilities of two minterms to be equal.

To minimize the memory usage in representing the probability of a gate, the assignment of $a_1 = 3$ is preferable. Thus, the aliasing-free assignment uses $a_1 = 3$ as the first assignment throughout the paper.

The similar idea of aliasing-free assignments is presented in [21], but its formulation is different from Equation (1). Furthermore, a_1 in our formulation is generalized to all positive integers greater than 2.

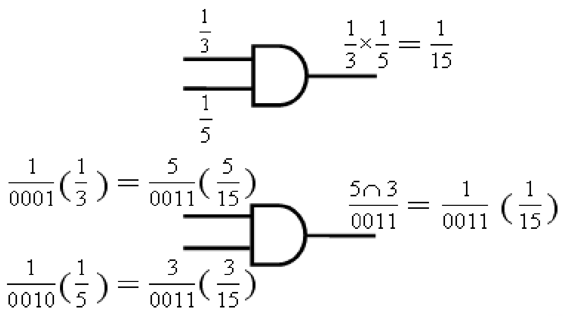
3.2 Probability Evaluation

3.2.1 Encoding Scheme and Alternative Operations

The probability formulae for AND, OR gates involve multiplication operation as shown in Fig. 2. The cost of multiplication operation is expensive in general. Thus, we propose two alternative operations, bitwise-AND ($\cap$) and bitwise-OR ($\cup$), to substitute the original probability formulae of AND and OR gate. As a result, the process of probability calculation will be more efficient.

We first introduce a new encoding scheme of the aliasing-free assignment used in our work. The assigned input probability in this work is expressed as a fractional number. Typically, both the numerator and denominator of a fractional number are encoded in binary, e.g., $\frac{2}{5}$ is expressed as $\frac{0010}{0101}$. But our encoding scheme here for denominators is different. The weight of bit b_i in the denominator is a_i (a_1=3, a_2=5, a_3=17, ..., $a_i = 2^{2^{i-1}} + 1$). For example, $\frac{1}{0001}$ represents $\frac{1}{3}$, $\frac{1}{0010}$ represents $\frac{1}{5}$, and $\frac{1}{0100}$ represents $\frac{1}{17}$. Furthermore, since denominators in all possible signal probabilities are either 3, 5, 17, ..., $2^{2^{i-1}} + 1$, or the multiplication of these numbers, we encode $\frac{1}{15}$ as $\frac{1}{0011}(3 \times 5 = 15)$, $\frac{1}{255}$ as $\frac{1}{0111}(3 \times 5 \times 17 = 255)$ and so on. This encoding scheme for denominators can significantly reduce the memory usage.

Next, we introduce how to use our encoding scheme and alternative operations in calculating the signal probability in the network. Two steps are conducted in this process. (I) Two input probabilities are transformed to its equivalent probability. The denominator of the equivalent probability is the lowest common multiple of the original denominators. (II) The two new numerators conduct bitwise-AND/bitwise-OR operation to obtain the numerator of output probability if it is an AND/OR



$\frac{1}{3}$
$\frac{1}{3}\times\frac{1}{5}=\frac{1}{15}$
$\frac{1}{5}$
$\frac{1}{0001}(\frac{1}{3})=\frac{5}{0011}(\frac{5}{15})$
$\frac{5\cap 3}{0011}=\frac{1}{0011}(\frac{1}{15})$
$\frac{1}{0010}(\frac{1}{5})=\frac{3}{0011}(\frac{3}{15})$

Figure 9: A demonstration of Example 3.1.

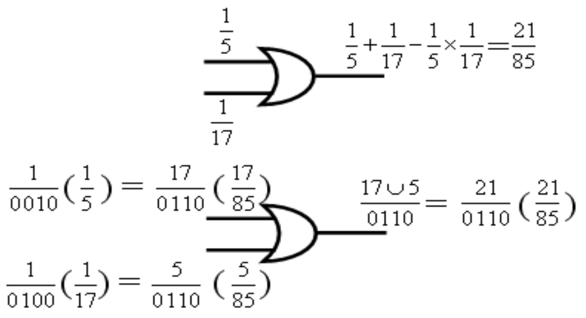
X_2	X_1	prob. of minterm
0	0	$(1-x_1)\times(1-x_2)=(1-\frac{1}{3})\times(1-\frac{1}{5})=\frac{8}{15}=\frac{8}{0011}$
0	1	$x_1\times(1-x_2)=\frac{1}{3}\times(1-\frac{1}{5})=\frac{4}{15}=\frac{4}{0011}$
1	0	$(1-x_1)\times x_2=(1-\frac{1}{3})\times\frac{1}{5}=\frac{2}{15}=\frac{2}{0011}$
1	1	$x_1\times x_2=\frac{1}{3}\times\frac{1}{5}=\frac{1}{15}=\frac{1}{0011}$

Figure 10: The probability of each minterm for 2-input functions assuming $x_1=\frac{1}{3}$, $x_2=\frac{1}{5}$.

gate. Examples 3.1 and 3.2 will demonstrate that these alternative operations work well for the probability calculation.

Example 3.1: Assume two input probabilities $\frac{1}{3}$ and $\frac{1}{5}$ are assigned to an AND gate as shown in Fig.9. The correct output probability is $\frac{1}{15}$ $(\frac{1}{0011})$ from $\frac{1}{3}\times\frac{1}{5}$. The same result can be obtained from our alternative operations. First, $\frac{1}{3}$ is encoded as $\frac{1}{0001}$, and $\frac{1}{5}$ is encoded as $\frac{1}{0010}$. We convert these two probabilities to their equivalent probabilities with the lowest common denominator, e.g., $\frac{1}{3}$ is converted to $\frac{5}{15}$ and $\frac{1}{5}$ is converted to $\frac{3}{15}$ with the lowest common denominator 15. Thus, $\frac{5}{15}$ and $\frac{3}{15}$ are encoded as $\frac{5}{0011}$ $(\frac{5}{15})$ and $\frac{3}{0011}$ $(\frac{3}{15})$, respectively. To obtain the numerator of the equivalent probability, a multiplication operation is required, e.g., $1\times5=5$ and $1\times3=3$. However, we observe that the new numerator can be obtained by shift-add operations instead of multiplication operations, i.e., numerator $5=(1\ll2)+1=4+1$, numerator $3=(1\ll1)+1=2+1$, where $\ll$ represents the shift left operator. Note that this shift-add operation is always applicable to our process due to the denominator is $2^{2^{i-1}}+1$ in our assignments. Shift left the numerator 2^{i-1} bits and add the numerator once can get the new numerator. Next, the two new numerators 5, 3 conduct bitwise-AND operation to get the numerator of the output probability, i.e., $5\cap3=1$. As a result, the correct output probability $\frac{1}{15}$ $(\frac{1}{0011})$ is obtained.

Next, we explain why the alternative operations with the new encoding scheme can also result in correct output probability. Suppose X_1 and X_2 are two independent signals in the AND function $F=X_1\cdot X_2$. The probability of $X_1=1$, $X_2=1$, and $F=1$ are x_1, x_2, and f, respectively. The function $F=X_1\cdot X_2$ can be expressed as $(X_1\cdot\overline{X_2}+X_1\cdot X_2)\cdot(\overline{X_1}\cdot X_2+X_1\cdot X2)$. Accordingly, from the probability point of view, $f=x_1\times x_2=(x_1\times(1-x_2)+x_1\times x_2)\times((1-x_1)\times x_2+x_1\times x_2)$. Since the aliasing-free assignments result in a unique probability for each function, and the unique probability of each function comes from the summation of probability of minterms as shown in Fig. 10, the multiplication operation in probability calculation can be replaced by bitwise-AND operation. In Fig. 10, $x_1=\frac{1}{3}=(x_1\times(1-x_2)+x_1\times x_2)=\frac{4}{15}+\frac{1}{15}=\frac{4}{0011}+\frac{1}{0011}=\frac{5}{0011}$, $x_2=\frac{1}{5}=((1-x_1)\times x_2+x_1\times x_2)=\frac{2}{15}+\frac{1}{15}=\frac{2}{0011}+\frac{1}{0011}=\frac{3}{0011}$. Thus, the operation $x_1\times x_2=\frac{1}{3}\times\frac{1}{5}$ can be replaced by $\frac{5\cap3}{0011}=\frac{1}{0011}=\frac{1}{15}$.



$\frac{1}{5}$
$\frac{1}{5}+\frac{1}{17}-\frac{1}{5}\times\frac{1}{17}=\frac{21}{85}$
$\frac{1}{17}$
$\frac{1}{0010}(\frac{1}{5})=\frac{17}{0110}(\frac{17}{85})$
$\frac{17\cup5}{0110}=\frac{21}{0110}(\frac{21}{85})$
$\frac{1}{0100}(\frac{1}{17})=\frac{5}{0110}(\frac{5}{85})$

Figure 11: A demonstration of Example 3.2.

Example 3.2: Assume two input probabilities $\frac{1}{5}$ and $\frac{1}{17}$ are assigned to an OR gate. The correct output probability is $\frac{21}{85}$ $(\frac{21}{0110})$ from $\frac{1}{5}+\frac{1}{17}-\frac{1}{5}\times\frac{1}{17}$. Similarly, the same result can be obtained from our alternative operations. Fig.11 shows the detail of alternative operation.

The explanation for OR gate is similar to that for AND gate, and is omitted here.

3.2.2 Dealing with Signal Correlation

For the gates whose inputs are correlated, i.e., not independent, the probability formulae shown in Fig. 2 cannot be applied directly in the calculation of output probability given numeral input probability. This subsection describes our method to deal with this problem. However, this method is only feasible when the assignments are aliasing-free described in Section 3.1. This method exploits the same encoding scheme and bitwise operations mentioned in Section 3.2.1. We use Example 3.3 to demonstrate our method when dealing with the signal correlation issue.

Example 3.3: Given a 3-input network as shown in Fig.12. Its function is $(A\cdot B)\cdot(B+C)=A\cdot B$. Given aliasing-free assignments, e.g., $a=\frac{1}{3}$, $b=\frac{1}{5}$, and $c=\frac{1}{17}$, the correct output probability is $\frac{1}{3}\times\frac{1}{5}=\frac{1}{15}=\frac{17}{255}$ from the function $A\cdot B$. On the other hand, from the network point of view, we have to derive the output probability from the primary inputs toward the primary outputs. First, the probability of gate $G1$ is $\frac{1}{15}(\frac{1}{0011})$. The probability of gate $G2$ is $\frac{21}{85}(\frac{21}{0110})$. These results are obtained from Example 3.1 and 3.2. For gate $G3$, we transform the two input probabilities $\frac{1}{0011}$ and $\frac{21}{0110}$ to their equivalent probability, i.e., $\frac{1}{0011}$ $(\frac{1}{15})$ becomes $\frac{17}{0111}$ $(\frac{17}{255})$, $\frac{21}{0110}$ $(\frac{21}{85})$ becomes $\frac{63}{0111}(\frac{63}{255})$. The new denominator 255 is the lowest common multiple of the original denominators 15 and 85, and it can be easily obtained by $0011\cup0110=0111$. Then, the numerators of these two probabilities conduct bitwise-AND operation, $17\cap63=17$, to get the numerator of probability of gate $G3$. As a result, the probability of gate $G3$ is $\frac{17}{0111}=\frac{17}{255}$. This is the same as the correct output probability.

When transforming the input probability to its equiv-

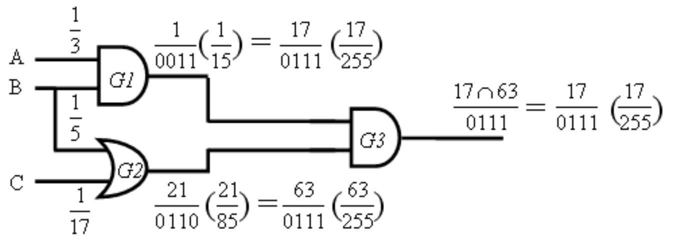

$\frac{1}{3}$
A
B
G1
$\frac{1}{5}$
C
G2
$\frac{1}{17}$
G3
$\frac{1}{0011}(\frac{1}{15}) = \frac{17}{0111}(\frac{17}{255})$
$\frac{17 \cap 63}{0111} = \frac{17}{0111}(\frac{17}{255})$
$\frac{21}{0110}(\frac{21}{85}) = \frac{63}{0111}(\frac{63}{255})$

Figure 12: A demonstration of Example 3.3.

alent probability, the lowest common multiple of original denominators is always used. This is easily achieved by the bitwise-OR of the two original denominators. *The lowest common multiple denominator suppresses the correlation of two input probabilities if the denominators of these two input probabilities have the same factor. The same factor implies the two inputs of the reconvergent gate are originated from the same input source.* Thus, our approach can deal with the signal correlation issue well when applying the aliasing-free assignments.

3.3 Internal Tree-Structure Replacement

An n-input network has 2^{2^n} distinct functions. It needs at least 2^n bits to represent each function uniquely. It is an inherent limitation for the unique representation. However, since we only consider two, instead of 2^{2^n}, networks for verification at one time, the representation complexity can be reduced by the internal tree-structure replacement. Although this idea results in the output probability of these two networks changed, it dose not affect the judgement on the equivalence of these two networks. We use a simple example to demonstrate this idea.

Example 3.4: To verify the equivalence of two networks $N1$ and $N2$ as shown in Fig. 13, we do not apply probabilities to all inputs directly. We perform depth-first traversal on $N1$ and $N2$, and then assign input probabilities sequentially. First, we assign $\frac{1}{3}$ and $\frac{1}{5}$ to input A and B in $N1$ and $N2$, the output probability of gate $G1$ in $N1$ and $N2$ can be calculated, both are $\frac{1}{15}$. Gate $G1$ in $N1$ and $N2$ are tree-structures and the output probabilities are equivalent ($\frac{1}{3}$, $\frac{1}{5}$ are aliasing-free assignments), thus, the output probability of gate $G1$ in $N1$ and $N2$ can be replaced as $\frac{1}{3}$. Then, $\frac{1}{5}$ is re-assigned to input C in $N1$ and $N2$. The probability of gate $G2$ in $N1$ and $N2$ are calculated and compared for equivalence checking. In this example, we use two input assignments, instead of three, to verify the equivalence of $N1$ and $N2$. The equivalent tree-structure replacement implicitly reduces the size of designs under verification. Thus,we can verify the equivalence of two networks more efficiently.

Although supergates [17] are also suitable for the replacement if they are equivalent, its identification in a network is a computation-intensive process. Here we do not consider the supergate replacement.

4. Experimental Results

The experiments are conducted over a set of MCNC benchmarks within SIS [20] environment on a Sun Blade 2500 workstation. These benchmarks are in BLIF format. Since the designs under verification only consist of AND,

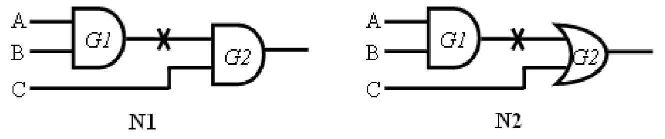


Figure 13: A demonstration of Example 3.4.

OR, and NOT gates, we decompose complex gates into these primitive gates by mapping to the SIS library (22-1.genlib). To compare two circuits that have the same functionality but different structures, we use SIS command ($map\ -m0$) to restructure one of the circuits for area optimization. Also, we separate a multiple output network into many single output subnetworks and derive the output probability of each subnetwork for comparison. To compare with BDD based approach, we use the SIS function ($ntbdd\_verify\_network$) to verify the equivalence of two networks with the arguments DFS_ORDER and ONE_AT_A_TIME.

Table 1 summarizes the experimental results of our approach against that of BDD based approach. Column 1 lists the benchmarks. Column 2 and 3 show the number of primary inputs and primary outputs of the circuit. $Max\ a_i$ in Column 4 is the maximum assignments we used after the internal tree-structure replacement. $Max\ TFI_{|PI|}$ in Column 4 is the maximum number of primary inputs in the transitive fanin (TFI) cone among all output functions. Column 5 shows if the internal tree-structure replacement can be applied in this benchmark. Y means yes, N means no. Column 6 and 7 show the CPU time and memory usage of our approach and BDD based approach on these benchmarks measured in second and mega bytes, respectively.

For example, $i3$ benchmark has 132 primary inputs and 6 primary outputs. The maximum number of primary inputs in the transitive fanin cone among 6 output functions is 32, and the internal tree-structure replacement can be used in this circuit. After the replacement, the maximum assignment is reduce from 32 to 7. Our approach spends 0.05 seconds and 3.41 mega bytes memory for calculating and comparing the output probabilities. BDD based approach spends 0.05 seconds and 4.8 mega bytes memory.

We know that the denominator of the aliasing-free input probability exponentially grows. It affects the scalability of our approach. In our experiments, the denominator of the maximal assignment is a_{24}, i.e., $2^{2^{24-1}} + 1 \approx 2^{2^{23}} \approx 10^{2525222}$, which is the largest value we can support. The benchmarks having more than 24 inputs in the transitive fanin cone of an output function may be verified by the internal tree-structure replacement. For example, $i3$ benchmark use this technique to reduce the number of assignments from 32 to 7. According to Table 1, 34 out of 48 cases have internal equivalent tree-structures. However, Table 1 excludes the benchmarks in MCNC whose $Max\ a_i$ exceeds the limit 24. These benchmarks cannot be verified currently by our approach.

Although our results are not better than that of BDD based approach for most benchmarks, they are very close in terms of time or memory usage. Since BDD based equivalence checking within SIS is a well developed ap-

Table 1: Experimental results of MCNC benchmarks.

			$Max\ a_i$	Tree	Time (s)	Mem. (MB)				$Max\ a_i$	Tree	Time (s)	Mem. (MB)
Circuits	|PI|	|PO|	$Max\ TFI_{\|PI\|}$	(Y/N)	Ours/BDD	Ours/BDD	Circuits	|PI|	|PO|	$Max\ TFI_{\|PI\|}$	(Y/N)	Ours/BDD	Ours/BDD
9symml	9	1	9 / 9	N	0.08 / 0.06	5.16 / 4.69	i1	25	16	5 / 11	Y	0.02 / 0.02	4.55 / 4.56
alu2	10	6	10 / 10	Y	0.23 / 0.13	5.59 / 5.21	lal	26	19	12 / 13	Y	0.05 / 0.05	4.77 / 4.78
x2	10	7	9 / 10	Y	0.03 / 0.03	4.38 / 4.57	pcler8	27	17	13 / 13	Y	0.05 / 0.05	4.66 / 4.64
cm152a	11	1	11 / 11	N	0.02 / 0.02	4.48 / 4.27	c8	28	18	13 / 13	Y	0.06 / 0.06	4.11 / 4.80
cm85a	11	3	9 / 10	Y	0.02 / 0.02	4.34 / 4.40	frg1	28	3	24 / 25	Y	1.67 / 0.04	15.00 / 4.42
cm151a	12	2	12 / 12	N	0.02 / 0.02	4.45 / 4.34	term1	34	10	20 / 20	Y	0.17 / 0.13	6.30 / 5.30
alu4	14	8	14 / 14	Y	0.83 / 0.30	6.67 / 6.10	count	35	16	19 / 20	Y	0.06 / 0.06	4.41 / 4.77
cm162a	14	5	10 / 11	Y	0.02 / 0.02	4.36 / 4.54	unreg	36	16	6 / 6	N	0.04 / 0.04	4.68 / 4.70
cu	14	11	13 / 13	Y	0.03 / 0.03	4.38 / 4.45	b9	41	21	14 / 14	Y	0.05 / 0.05	4.75 / 4.74
cm163a	16	5	8 / 9	Y	0.03 / 0.03	4.35 / 4.41	cht	47	36	6 / 6	N	0.07 / 0.07	4.83 / 4.86
cmb	16	4	12 / 12	Y	0.03 / 0.03	4.54 / 4.37	apex7	49	37	17 / 24	Y	0.13 / 0.13	5.20 / 5.02
go	16	13	9 / 9	Y	0.04 / 0.04	4.59 / 4.59	x1	51	35	24 / 25	Y	0.84 / 0.13	18.00 / 5.22
parity	16	1	16 / 16	N	0.03 / 0.03	4.39 / 4.43	ex2.	85	66	16 / 17	Y	0.18 / 0.18	5.26 / 5.15
pm1	16	13	8 / 9	Y	0.02 / 0.02	4.55 / 4.37	i9	88	63	13 / 13	N	0.76 / 0.66	6.66 / 6.45
t481	16	1	16 / 16	N	24.77 / 0.02	19.00 / 4.33	x4	94	71	15 / 15	Y	0.22 / 0.22	5.48 / 5.57
tcon	17	16	3 / 3	N	0.03 / 0.03	4.50 / 4.34	**i3**	**132**	**6**	**7 / 32**	**Y**	**0.05 / 0.05**	**3.41 / 4.80**
vda	17	39	17 / 17	Y	1.45 / 0.62	7.33 / 6.65	i5	133	66	19 / 19	Y	0.17 / 0.23	5.62 / 5.30
pcle	19	9	9 / 12	Y	0.03 / 0.03	4.49 / 4.59	i8	133	81	17 / 17	Y	1.71 / 1.44	11.00 / 10.00
sct	19	15	14 / 14	Y	0.05 / 0.05	4.11 / 4.70	apex6	135	99	22 / 24	Y	0.51 / 0.42	9.50 / 6.23
cc	21	20	6 / 7	Y	0.04 / 0.04	4.34 / 4.61	x3	135	99	23 / 24	Y	1.41 / 0.45	15.00 / 6.52
cm150a	21	1	21 / 21	N	0.11 / 0.02	6.52 / 4.57	i6	138	67	5 / 5	N	0.21 / 0.32	5.87 / 5.94
mux	21	1	21 / 21	N	0.06 / 0.04	5.74 / 4.61	frg2	143	139	23 / 25	Y	2.35 / 0.87	15.00 / 7.63
cordic	23	2	23 / 23	N	1.98 / 0.04	13.00 / 4.66	i7	199	67	6 / 6	N	0.25 / 0.46	6.25 / 6.33
ttt2	24	21	14 / 14	Y	0.08 / 0.08	5.01 / 4.95	des	256	245	18 / 19	Y	5.23 / 4.42	15.00 / 15.00

proach, our competitive results disclose the potential of the probabilistic approach.

5. Conclusions

This paper presents a continued work on the logic equivalence checking from the probabilistic aspect. First, an aliasing-free assignment procedure is proposed such that a unique output probability of a network is obtained. Next, more efficient operations are exploited to substitute the original probability formulae in the probability calculation process. A method that successfully deals with the signal correlation issue when applying the aliasing-free probabilities is also introduced. Finally, the internal tree-structure replacement is used to reduce the number of required input assignments. Although the work currently cannot handle those circuits which require more than 24 input assignments in the transitive fanin cone of a subnetwork, it deal with the other circuits well. This paper discloses the potential and current limitation of the probabilistic approach to logic equivalence checking. We believe the probabilistic approach would be more efficiently and effectively if further improvement is offered and the limitation is loosened.

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] V. D. Agrawal, et al., "Probabilistic Testability," in *Proc. of ICCD*, pp. 562-565, 1985.

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

[5] F. Brglez, "On Testability of Combinational Networks," in *Proc. of ISCAS*, pp. 221-225, 1984.

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

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

[8] S. Chakravarty, et al., "On Computing Signal Probability and Detection Probability of Stuck-at Faults," *IEEE Trans. on Computer*, pp. 1369-1377, Nov. 1990.

[9] S.-C. Chang, et al., "TAIR: Testability Analysis by Implication Reasoning," *IEEE Trans. on Computer-Aided Design*, pp. 152-160, Jane 2000.

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

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

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

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

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

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

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

[17] H. B. Min, et al., "Graph-theoretic Algorithm for Finding Maximal Supergates in Combinational Logic Circuits,"in *Proc. of IEE Circ. Dev. Syst.*, pp. 313-318, 1996.

[18] F. N. Najm, "Transition Density: A New Measure of Activity in Digital Circuits," *IEEE Trans. on Computer-Aided Design*, pp. 310-323, Feb. 1993.

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

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

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

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