Probabilistic Equivalence Checking of Multiple-Valued Functions*

Elena Dubrova
Department of Electronics
Royal Institute of Technology
S-164 40 Kista
Sweden
elena@ele.kth.se

Harald Sack
FB IV - Informatik
Universität Trier
D-54286 Trier
Germany
sack@uni-trier.de

Abstract

This paper describes a probabilistic method for verifying the equivalence of two multiple-valued functions. Each function is hashed to an integer code by transforming it to a integer-valued polynomial and evaluating it for values of variables taken independently and uniformly at random from a finite field. Since the polynomial is unique for a given function, if two hash codes are different, then the functions are not equivalent. However, if two hash codes are the same, the functions may or may not be equivalent, because different polynomials may happen to hash to the same code. Thus, the method presented in this paper determines the equivalence of two functions with a known (small) probability of error, arising from collisions between inequivalent functions. Such a method seems to be an attractive alternative for verifying functions that are too large to be handled by deterministic equivalence checking methods.

Index terms: Multiple-valued function, equivalence checking, probabilistic verification

*A preliminary version of this paper has appeared in [1].

1 Introduction

Many problems in digital design, especially on high and system levels of abstraction, can be formulated as a sequence of operations on functions of symbolic variables, taking their values from a finite set of values. Symbolic variables are often more naturally associated with the problem specification than the variables obtained by a binary encoding. For example, a traffic light controller is best described by "green", "yellow" and "red" values of a symbolic variable rather than an obscure encoding "00", "01" and "10". A description using symbolic variables is usually simpler and more compact. It is also easier to understand and analyze and therefore less prone to errors compared to the one obtained by binary encoding. In addition, symbolic variables better preserve the inherent structure of the specification. This is beneficial, for example, when Decision Diagrams are used as a data structure [2], [3], since the analysis of this structure might help selecting a good ordering for the symbolic variables, minimizing the total size of the diagram.

The problem formulation involving symbolic variables can be modeled using *multiple-valued functions* of type $f : M^n \rightarrow M$ on a fixed set $M \stackrel{df}{=} \{0, 1, \ldots, m-1\}$ [4]. Each of the n variables $x_1, x_2, \ldots, x_n$ of f takes its value from M. However, the variables do not have to depend on all the values of M, i.e. their range may vary.

Apart from modeling the problems on higher levels of abstraction, multiple-valued functions are used in logic design of electronic circuits, which employ more than two discrete levels of signals. In the last few years years, advances in integrated circuit technology made feasible fabrication of several commercial multiple-valued logic products, including 256-Mbit 4-valued flash memory [5] and 4-Gbit 4-valued DRAM [6]. Multiple-valued logic circuits offer several

potential opportunities for the improvement of present VLSI circuit designs. For example, serious difficulties with limitations on the number of connections of an integrated circuit with the external world (pin-out problem) as well as on the number of connections inside the circuit encountered in some VLSI circuit synthesis could be substantially reduced if signals in the circuit are allowed to assume four or more states rather than only two. In addition, there is a clear mathematical attraction of using multiple-valued number representation in many applications. For example, residue [7], [8] and redundant number systems [9], [10] allow to reduce or eliminate the ripple-through carries which are involved in normal binary addition or subtraction, resulting in high-speed arithmetic operations.

In spite of these potential advantages, practicality of multiple-valued logic design heavily depends on the availability of efficient computer-aided tools for representation, manipulation and verification of multiple-valued logic functions. Some existing tools, such as Berkeley's tool for verification and synthesis VIS [11], provide a solution for the special case of multiple-valued input binary-valued output functions of type $M^n \rightarrow \{0, 1\}$, but the general problem is still open.

This paper focuses on the problem of equivalence checking of multiple-valued functions. Up to now, very little research is done in this area. Some techniques for verification of Boolean circuits, such as random simulation or symbolic simulation, can be directly applied to verification of multiple-valued logic case [12], [13]. Similarly, verification procedures employing Reduced Ordered Binary Decision Diagrams (ROBDDs) [2] can be adapted to Multiple-valued Decision Diagrams (MDD) [3] as shown in [14]. However, the MDD verification methods representing functions as single, monolithic graph can not guarantee to be feasible for all functions. As in the case for Boolean functions,

deterministically testing the equivalence of multiple-valued functions might cause an exponential growth in size for the used data structures, because the representation has to be unique. Otherwise, if we are employing a more compact representation, the worst case time complexity of the deterministic verification algorithms becomes exponential.

In this paper we present a probabilistic method for checking equivalence of two multiple-valued logic functions, which generalizes the method developed in [15] for the Boolean case. We define a functional transformation A which converts a multiple-valued function $f : M^n \to M$ into a polynomial of type $A_m[f] : \mathbf{Z}_p^n \to \mathbf{Z}_p$ over a finite field of integers $\mathbf{Z}_p$ modulo p, for some prime p. This polynomial is used to generate a hash code for f, by evaluating the value of $A_m[f](x_1, \ldots, x_n)$ for the assignment of variables x_i, $i \in \{1, \ldots, n\}$, taken independently and uniformly at random from $\mathbf{Z}_p$. Since a polynomial is unique for a given function, if two hash codes are different, then the functions are certainly not equivalent. On the other hand, if two hash codes are the same, the functions may or may not be equivalent, because two different polynomials may happen to hash to the same code. Thus, the method we are going to present guarantees verification of the equivalence of two functions with a known probability of error, arising from collisions between inequivalent functions.

The paper is organized as follows. In Section 1, the A-transform is defined and its properties are studied. Section 2 shows how to compute the integer-valued polynomial given by the A-transform. Section 3 defines the notion of hash code for a function. In Section 4, we give a conclusion and describe the topics for further research.

2 Definition and properties of the A-transform

In this section we generalize the functional transformation introduced in [15] for Boolean functions to the multiple-valued case. Unless specified otherwise, all the operations are over $\mathbf{Z}_p$, i.e. " $\cdot$ " is the mod p multiplication and " $+$ " is the mod p addition, with p being a prime.

To define the transformation A, we first associate a *key polynomial* with each of the m^n input assignments of a multiple-valued function $f(x_1, \ldots, x_n)$. We then sum up the key polynomials of assignments producing the non-zero output value of f, and interpret the result as a integer-valued function $A_m[f](x_1, \ldots, x_n)$ over $\mathbf{Z}_p$.

The key polynomial for a given row of the truth table is a product of terms, where each term is associated with a particular input variable x_i, $i \in \{1, \ldots, n\}$. If b_i represents the value of x_i in a given row of the truth table, then the corresponding term $w(b_i, x_i)$ in the key polynomial is defined as follows.

Definition 1 *For any $m > 1$, $w : \mathbf{Z}_p \times \mathbf{Z}_p \to \mathbf{Z}_p$ is defined by*

$$w(b, x) = \sum_{i=0}^{m-1} \left(\prod_{j \in M - \{i\}} \frac{j-b}{j-i} \cdot \prod_{j \in M - \{i\}} \frac{j-x}{j-i} \right)$$

It is easy to see that $\prod_{j \in M-\{i\}} \frac{j-b}{j-i} = 1$ for $b = i$ and $\prod_{j \in M-\{i\}} \frac{j-b}{j-i} = 0$ for $b \neq i$. Therefore, parameter b acts as a selector between the terms $\prod_{j \in M-\{i\}} \frac{j-x}{j-i}$ for different values of $i \in M$, i.e. $w(0, x) = \prod_{j \in M-\{0\}} \frac{j-x}{j}$, $w(1, x) = \prod_{j \in M-\{1\}} \frac{j-x}{j-1}$, and so on. On the other hand, each of the terms $\prod_{j \in M-\{i\}} \frac{j-x}{j-i}$, represents a polynomial which evaluates to 1 for $x = i$ and evaluates to 0 for $x \in M - \{i\}$. So, such a polynomial has a behavior similar to the behavior of the literal operator $\overset{i}{x}$, which is defined by

$$\overset{i}{x} \stackrel{df}{=} \begin{cases} m-1 & \text{if } x = i \\ 0 & \text{otherwise,} \end{cases}$$

except that for $x = i$ the literal $\overset{i}{x}$ evaluates to $m - 1$, and not to 1.

Since the definition of $w(b, x)$ involves division operation, we have to prove that the resulting value of $w(b, x)$ is always an integer from $\mathbf{Z}_p$. As shown above, the term $\prod_{j \in M - \{i\}} \frac{j - b}{j - i}$ is always either 0 or 1, depending on b. It remains to show that $\prod_{j \in M - \{i\}} \frac{j - x}{j - i} \in \mathbf{Z}_p$. This is proved in the following Lemma.

Lemma 1 *For any $i \in M$ and any $x \in \mathbf{Z}_p$, $\prod_{j \in M - \{i\}} \frac{j - x}{j - i} \in \mathbf{Z}_p$.*

Proof: For any fixed $i \in M$ and any $x \in \mathbf{Z}_p$, $\prod_{j \in M - \{i\}} \frac{j - x}{j - i}$ has the following structure:

$$\frac{(0 - x) \cdot (1 - x) \cdot \ldots \cdot ((i - 1) - x)}{(0 - i) \cdot (1 - i) \cdot \ldots \cdot ((i - 1) - i)} \cdot \frac{((i + 1) - x) \cdot \ldots \cdot ((m - 1) - x)}{((i + 1) - i) \cdot \ldots \cdot ((m - 1) - i)}$$

Note, that no term in the divisors can be 0 since we fixed i and $j \neq i$. If any of the terms on the dividends is 0, the Lemma trivially holds, so we will consider the case when none of the terms on the dividends is 0.

By multiplying all the terms of the first fraction by -1 and reversing their order, we get:

$$\frac{(x - (i - 1)) \cdot \ldots \cdot (x - 1) \cdot x}{1 \cdot 2 \cdot \ldots \cdot (i - 1) \cdot i} \cdot \frac{((i + 1) - x) \cdot \ldots \cdot ((m - 1) - x)}{1 \cdot 2 \cdot \ldots \cdot ((m - 1) - i)}$$

It is easy to see that

$$\frac{(x - (i - 1)) \cdot \ldots \cdot (x - 1) \cdot x}{1 \cdot 2 \cdot \ldots \cdot (i - 1) \cdot i} = \binom{x}{i}$$

and

$$\frac{((i + 1) - x) \cdot \ldots \cdot ((m - 1) - x)}{1 \cdot 2 \cdot \ldots \cdot ((m - 1) - i)} = \binom{(m - 1) - x}{(m - 1) - i}$$

which are in known be integers, as long as $x, i, (m - 1) - x$ and $(m - 1) - i$ are integers. Since $i \in M$ and $x \in \mathbf{Z}_p$, $x, i, (m - 1) - x$ and $(m - 1) - i$ are

always integers and therefore $\prod_{j \in M - \{i\}} \frac{j-x}{j-i} \in \mathbf{Z}_p$. □

The *key polynomial* W_n for an assignment $(b_1, \ldots, b_n) \in M^n$ of n variables is defined as the product of the $w(b_i, x_i)$ terms, $i \in \{1, \ldots, n\}$.

Definition 2 *For any $n \geq 0$ and $m > 1$, the key polynomial $W_n : \mathbf{Z}_p^{2n} \to \mathbf{Z}_p$ is defined by*

$$W_n(b_1, \ldots, b_n, x_1, \ldots, x_n) = \prod_{i=1}^{n} w(b_i, x_i).$$

For example, for $m = 3$:

$$W_2(0, 1, x_1, x_2) = \frac{1}{2}(1 - x_1)(2 - x_1)x_2(2 - x_2).$$

Similarly:

$$W_2(1, 2, x_1, x_2) = x_1(2 - x_1)(-\frac{1}{2}x_2)(1 - x_2).$$

Now, we define the transformation A as a sum of key polynomials W_n for all assignments $(b_1, \ldots, b_n) \in M^n$, each multiplied by the value of f for the corresponding assignment. We give a definition, applicable for discrete functions $\mathbf{Z}_p^n \to \mathbf{Z}_p$ over the field $\mathbf{Z}_p$. Note that, since $M \subset \mathbf{Z}_p$, the multiple-valued functions $M^n \to M$ are a subset of the discrete functions $\mathbf{Z}_p^n \to \mathbf{Z}_p$. While a discrete function over $\mathbf{Z}_p$ is defined for inputs $x_i \notin M$ as well, the values that the function produces for such assignments do not participate in the definition. To distinguish between multiple-valued functions and discrete functions over $\mathbf{Z}_p$, throughout the paper we use the unsubscribed letters f, g for multiple-valued functions of type $M^n \to M$ and the subscribed letters f_z, g_z for discrete functions of type $\mathbf{Z}_p^n \to \mathbf{Z}_p$.

Definition 3 *Given a function of type $f_z : \mathbf{Z}_p^n \to \mathbf{Z}_p$ and $m > 1$, the polynomial $A_m[f_z] : \mathbf{Z}_p^n \to \mathbf{Z}_p$ is defined by*

$$A_m[f_z](x_1, \ldots, x_n) = \sum_{(b_1, .., b_n) \in M^n} f_z(b_1, .., b_n) \cdot W_n(b_1, .., b_n, x_1, \ldots, x_n)$$

For example, for $m = 3$ and $n = 2$, the polynomial $A_3[f_z](x_1, x_2)$ is given by

$$\begin{aligned} A_3[f_z](x_1,x_2) &= \\ &= f(0,0)\cdot\tfrac{1}{2}(1-x_1)(2-x_1)(1-x_2)(2-x_2) + \\ &+ f(0,1)\cdot\tfrac{1}{2}(1-x_1)(2-x_1)x_2(2-x_2) + \\ &+ f(0,2)\cdot\tfrac{1}{4}(1-x_1)(2-x_1)(-x_2)(1-x_2) + \\ &+ f(1,0)\cdot x_1(2-x_1)(1-x_2)(2-x_2) + \\ &+ f(1,1)\cdot x_1(2-x_1)x_2(2-x_2) + \\ &+ f(1,2)\cdot x_1(2-x_1)(-x_2)(1-x_2) + \\ &+ f(2,0)\cdot\tfrac{1}{2}(-x_1)(1-x_1)(1-x_2)(2-x_2) + \\ &+ f(2,1)\cdot\tfrac{1}{2}(-x_1)(1-x_1)x_2(2-x_2) + \\ &+ f(2,2)\cdot\tfrac{1}{4}(-x_1)(1-x_1)(-x_2)(1-x_2). \end{aligned}$$

For instance, for the 3-valued function $f(x_1, x_2) = MIN(x_1, x_2)$, the corresponding polynomial is

$$A_3[f](x_1,x_2) = x_1(2-x_1)x_2(2-x_2) + x_1(2-x_1)(-x_2)(1-x_2) + (-x_1)(1-x_1)x_2(2-x_2) + 2(-x_1)(1-x_1)(-x_2)(1-x_2) = \tfrac{5}{2}x_1x_2 - x_1x_1^2 - x_1^2x_2 + \tfrac{1}{2}x_1^2x_2^2.$$

Lemma 2 summarizes some properties of the polynomial $A_m[f]$, obvious from Definition 3.

Lemma 2 *For any functions f_z and g_z over $\mathbf{Z}_p$ and any constant $c \in \mathbf{Z}_p$, the following two properties hold:*

(a) $A_m[c \cdot f_z] = c \cdot A_m[f_z]$.

(b) $A_m[f_z + g_z] = A_m[f_z] + A_m[g_z]$.

Next, we show that even though applying the A-transform to a multiple-valued function f increases the domain of the function from M^n to $\mathbf{Z}_p^n$, the value of the polynomial $A_m[f]$ equals to the value of f for any assignment $(b_1, \ldots, b_n) \in M^n$. To identify the functions treated identically by the A-transform we first introduce the notion of *m-equivalence.*

Definition 4 *The functions f_z and g_z of type $\mathbf{Z}_p^n \rightarrow \mathbf{Z}_p$ are m-equivalent if and only if $f_z(b_1, \ldots, b_n) = g_z(b_1, \ldots, b_n)$ for any assignment $(b_1, \ldots, b_n) \in M^n$.*

We write $f_z \stackrel{m}{=} g_z$ to denote that f_z and g_z are m-equivalent. If both f and g are multiple-valued functions of type $M^n \rightarrow M$ then $f \stackrel{m}{=} g$ is the same as $f = g$.

By Definition 3, $f_z \stackrel{m}{=} g_z$ implies $A_m[f_z] = A_m[f_z]$. However, can we conclude $A_m[f_z] \neq A_m[f_z]$ from $f_z \stackrel{m}{\neq} g_z$? To answer this question let us examine the behavior of polynomial $A_m[f]$, when it is evaluated for some assignment $(b_1, \ldots, b_n) \in M^n$. It is easy to see that for any $b, b' \in M$, $w(b, b') = 1$ if $b = b'$, and $w(b, b') = 0$, otherwise. Therefore, for any $(b_1, \ldots, b_n), (b'_1, \ldots, b'_n) \in M^n$, $W(b_1, \ldots, b_n, b'_1, \ldots, b'_n) = 1$ if $b_i = b'_i$ for all $i \in \{1, \ldots, n\}$, and $W(b_1, \ldots, b_n, b'_1, \ldots, b'_n) = 0$, otherwise. Using these facts, we can prove the following theorem:

Theorem 1 *For any function $f_z : \mathbf{Z}_p^n \rightarrow \mathbf{Z}_p$, $A_m[f_z] \stackrel{m}{=} f_z$.*

Proof: By Definition 3, for any $(b'_1, \ldots, b'_n) \in M^n$ we have:

$$A_m[f_z](b'_1, \ldots, b'_n) = \sum_{(b_1, \ldots, b_n) \in M^n} f_z(b_1, \ldots, b_n) \cdot W_n(b_1, \ldots, b_n, b'_1, \ldots, b'_n)$$

Since $W(b_1, \ldots, b_n, b'_1, \ldots, b'_n) = 1$ only if $b_i = b'_i$ for all $i \in \{1, \ldots, n\}$, and 0 otherwise, this gives us $A_m[f_z](b'_1, \ldots, b'_n) = f(b'_1, \ldots, b'_n) \cdot 1$. Since it holds for any $(b'_1, \ldots, b'_n) \in M^n$, we get $A_m[f_z] \stackrel{m}{=} f_z$. □

The theorem shows that the the value of $A_m[f]$ is the same as the value of f for any assignment $(b_1, \ldots, b_n) \in M^n$. Therefore, the polynomials for two different multiple-valued functions f and g differ on all assignments $(b_1, \ldots, b_n) \in M^n$ for which $f(b_1, \ldots, b_n) \neq g(b_1, \ldots, b_n)$. Consequently, $f \neq g$ implies $A_m[f] \neq A_m[g]$.

3 A method for deriving A-transform

Computing A-transforms using Definition 3 is feasible only for very small functions. For larger functions, we present an alternative method, which is described in this section.

Let $f_{x_i=j}$ denote a subfunction of the function $f(x_1, \ldots, x_n)$ with the variable x_i being fixed to the value j, i.e. $f_{x_i=j} \stackrel{df}{=} f(x_1, \ldots, x_{i-1}, j, x_{i+1}, \ldots, x_n)$.

The following Theorem presents a decomposition of $A_m[f]$ which expresses the polynomial of a function $f(x_1, \ldots, x_n)$ in terms of the polynomials of the subfunctions $f_{x_i=j}$, $i \in \{1, \ldots, n\}$, $j \in M$.

Theorem 2 *Every polynomial $A_m[f]$, $m > 1$, can be decomposed with respect to a variable x_i of f, $i \in \{1, \ldots, n\}$, in the following way*

$$A_m[f] = \sum_{j=0}^{m-1} \left(\prod_{k \in M - \{j\}} \frac{k - x_i}{k - j} \right) \cdot A_m[f_{x_i=j}].$$

Proof: In order to simplify the exposition and without loss of generality, we show the proof for the case of $x_i = x_1$. We use X as an abbreviation for $x_1, \ldots, x_n$.

$$
\begin{aligned}
A_m[f] &= \sum_{(b_1,..,b_n)\in M^n} f(b_1,\ldots,b_n)\cdot W_n(b_1,\ldots,b_n,X) \\
&\quad \{\text{Definition 3}\} \\
&= \sum_{j=0}^{m-1} \sum_{(b_2,..,b_n)\in M^{n-1}} f(j,b_2,..,b_n)\cdot W_n(j,b_2,...,b_n,X) \\
&\quad \{\text{re-grouping}\} \\
&= \sum_{j=0}^{m-1} \sum_{(b_2,...,b_n)\in M^{n-1}} f(j,b_2,\ldots,b_n)\cdot \prod_{k\in M-\{j\}} \tfrac{k-x_1}{k-j}\cdot \\
&\quad \cdot W_{n-1}(b_2,\ldots,b_n,x_2,\ldots,x_n) \\
&\quad \{w(j,x_1) = \prod_{k\in M-\{j\}} \tfrac{k-x_1}{k-j},\ \text{Definition 2}\} \\
&= \sum_{j=0}^{m-1} \left(\prod_{k\in M-\{j\}} \tfrac{k-x_1}{k-j} \right) \cdot A_m[f_{x_1=j}] \\
&\quad \{\text{Definition 3}\} \qquad \square
\end{aligned}
$$

Next, we prove a lemma showing how the term $\prod_{k\in M-\{j\}} \frac{k-x}{k-j}$, $j \in M$, can be expressed by an m-equivalent polynomial in linear form.

Lemma 3 *For any variable x from $\mathbf{Z}_p$, any fixed $j \in M$ and $m > 2$, the following m-equivalence holds:*

$$
\prod_{k\in M-\{j\}} \frac{k-x}{k-j} \underset{m}{\equiv} \begin{cases} \sum_{i=0}^{m-1} a_{i0}x^i, & \text{if } j = 0 \\ a_{jj}x^j + a_{j(m-j)}x^{m-j}, & \text{if } j \in M-\{0\} \\ & \quad \text{and } j \neq m-j \\ a_{jj}x^j, & \text{if } j \in M-\{0\} \text{ and } j = m-j, \end{cases}
$$

where $\forall i,j \in M$, $a_{ij} = \frac{D_{ij}}{D}$, with D and D_{ij} given by

$$
D = \prod_{i=1}^{m-1} i^i - \prod_{i=1}^{m-1} i^{(m-i)},
$$

$$D_{ij} = \begin{cases} \prod_{k=1}^{m-1} k^{k \oplus_m i} - \prod_{k=1}^{m-1} k^{(m-k) \oplus_m i}, & \text{if } j = 0 \\ \frac{1}{i^i} \cdot \prod_{k=1}^{m-1} k^k, & \text{if } j \neq 0 \text{ and } j = i \text{ and } j \neq m-i \\ -\frac{1}{(m-i)^i} \cdot \prod_{k=1}^{m-1} k^{m-k}, & \text{if } j \neq 0 \text{ and } j = m-i \\ & \text{and } j \neq i \\ \frac{1}{i^i} \cdot \left(\prod_{k=1}^{m-1} k^k - \prod_{k=1}^{m-1} k^{m-k} \right), & \text{if } j \neq 0 \text{ and } j = i \\ 0, & \text{otherwise,} \end{cases} \tag{1}$$

where " $\oplus_m$ " denotes addition modulo m and all other operations are regular arithmetic operations in $\mathbf{Z}_p$.

Proof: We compute the coefficients a_{ij}, $i, j \in M$, by solving the following system of m linear equations with m unknown elements:

$$\begin{cases} a_{0j} \cdot 0^0 + a_{1j} \cdot 0^1 + a_{2j} \cdot 0^2 + \ldots + a_{(m-1)j} \cdot 0^{m-1} = b_0 \\ a_{0j} \cdot 1^0 + a_{1j} \cdot 1^1 + a_{2j} \cdot 1^2 + \ldots + a_{(m-1)j} \cdot 1^{m-1} = b_1 \\ \ldots \\ a_{0j} \cdot (m-1)^0 + a_{1j} \cdot (m-1)^1 + a_{2j} \cdot (m-1)^2 + \ldots \\ \quad \ldots + a_{(m-1)j} \cdot (m-1)^{m-1} = b_{m-1}, \end{cases}$$

where $\forall i \in M$, $b_i = \prod_{k \in M - \{j\}} \frac{k-i}{k-j}$. Such a system can be described by matrices as $\mathbf{X} \cdot \mathbf{a} = \mathbf{b}$, where

$$\mathbf{X} = \begin{pmatrix} 0^0 & 0^1 & 0^2 & \ldots & 0^{m-1} \\ 1^0 & 1^1 & 1^2 & \ldots & 1^{m-1} \\ \ldots & \ldots & \ldots & \ldots & \ldots \\ (m-1)^0 & (m-1)^1 & (m-1)^2 & \ldots & (m-1)^{m-1} \end{pmatrix},$$

$$\mathbf{a} = \begin{pmatrix} a_{0j} \\ a_{1j} \\ \ldots \\ a_{(m-1)j} \end{pmatrix}, \quad \mathbf{b} = \begin{pmatrix} b_0 \\ b_1 \\ \ldots \\ b_{m-1} \end{pmatrix}.$$

From linear algebra we know that such a system always has a solution, and this solution is unique [16]. We compute the ith element of $\mathbf{a}$ by applying Kramer's

rule, which says that, for any $i \in M$, a_{ij} is given by the formula $a_{ij} = \frac{D_{ij}}{D}$, where D is the *determinant* of $\mathbf{X}$, and D_{kj} is the determinant computed after the replacement of the ith column of $\mathbf{X}$ by vector $\mathbf{b}$.

Observe that matrix $\mathbf{X}$ has a very regular structure, namely for all $i, j \in M$, $x_{ij} = i^j$. Therefore, by applying standard rules for computing determinants [16], it is easy to show that D and D_{ij} are given by the equation (1).

Examining the structure of D_{ij}, we can derive the following properties of the elements of a_{ij}:

$$a_{ij} = \begin{cases} \frac{D_{ij}}{D} & \forall i, j: \text{ such that } j = 0 \text{ or } i = j \text{ or } i = m - j \\ 0 & \text{otherwise.} \end{cases} \tag{2}$$

So, the only elements a_{ij} which can possibly have non-zero values are a_{i0} for all $i \in M$, and a_{jj} and $a_{j(m-j)}$ for all $j \in M - \{0\}$. Therefore, the expression, m-equivalent to the term $\prod_{k \in M - \{j\}} \frac{k-x}{k-j}$, can be simplified to:

$$\prod_{k \in M-\{j\}} \frac{k-x}{k-j} \stackrel{m}{=} \begin{cases} \sum_{i=0}^{m-1} a_{i0} x^i, & \text{if } j = 0 \\ a_{jj} x^j + a_{j(m-j)} x^{m-j}, & \text{if } j \in M - \{0\} \\ & \quad \text{and } j \neq m - j \\ a_{jj} x^j, & \text{if } j \in M - \{0\} \text{ and } j = m - j. \end{cases}$$

□

As we mentioned above, $\prod_{k \in M-\{j\}} \frac{k-x}{k-j}$ has a behavior similar to the behavior of the literal operator $\overset{j}{x}$, except that for $x = j$ the literal $\overset{j}{x}$ evaluates to $m - 1$, and not to 1. Therefore, $\overset{j}{x} \stackrel{m}{=} (m-1) \cdot \prod_{k \in M-\{j\}} \frac{k-x}{k-j}$, and thus, from Lemma 3, we can conclude that

$$\overset{j}{x} \stackrel{m}{=} \begin{cases} (m-1) \cdot \sum_{i=0}^{m-1} a_{0i} x^i, & \text{if } j = 0 \\ (m-1) \cdot (a_{jj} x^j + a_{j(m-j)} x^{m-j}), & \text{if } j \in M - \{0\} \\ & \quad \text{and } j \neq m - j \\ (m-1) \cdot a_{jj} x^j, & \text{if } j \in M - \{0\} \text{ and } j = m - j \end{cases}$$

We use Theorem 2 and Lemma 3 to derive another type of decomposition of $A_m[f]$, which will be used later to derive a canonical expansion for $A_m[f]$.

Theorem 3 *Every polynomial $A_m[f]$, $m > 1$, can be decomposed with respect to a variable x of f in the following way:*

case 1: *if m is odd, then*

$$A_m[f] = a_{00}A_m[f_{x=0}] + \sum_{j=1}^{m-1}((a_{j0}A_m[f_{x=0}]+ \\ +a_{jj}A_m[f_{x=j}] + a_{j(m-j)}A_m[f_{x=m-j}]) \cdot x^j\Big)$$

case 2: *if m is even, then*

$$A_m[f] = a_{00}A_m[f_{x=0}] \; + \sum_{\substack{j=1 \\ j \neq m/2}}^{m-1} ((a_{j0}A_m[f_{x=0}] + \\ + a_{jj}A_m[f_{x=j}] + a_{j(m-j)}A_m[f_{x=m-j}]) \cdot x^j + \\ + (a_{\frac{m}{2}0}A_m[f_{x=0}] + a_{\frac{m}{2}\frac{m}{2}}A_m[f_{x=m/2}]) \cdot x^{m/2}\Big)$$

where $\forall i, j \in M$, $a_{ij} = \frac{D_{ij}}{D}$, and D and D_{ij} given by (1).

Proof:

$$\begin{aligned} A_m[f] &= \sum_{j=0}^{m-1}\left(\prod_{k\in M-\{j\}}\frac{k-x_i}{k-j}\right)\cdot A_m[f_{x_i=k}] \qquad \{\text{Theorem 2}\} \\ &= \sum_{i=0}^{m-1} a_{i0}x^i \cdot A_m[f_{x=0}] + \sum_{j=1}^{m-1}(a_{jj}x^j + a_{j(m-j)}x^{m-j})\cdot \\ &\cdot A_m[f_{x=j}] \qquad \{\text{Lemma 3}\} \\ &= a_{00}A_m[f_{x=0}] + \sum_{j=1}^{m-1}(a_{j0}A_m[f_{x=0}] + a_{jj}A_m[f_{x=j}]+ \\ &+ a_{j(m-j)}A_m[f_{x=m-j}]) \cdot x^j \qquad \{\text{reordering}\} \end{aligned}$$

□

Let F be the vector of coefficients of the truth table of the function f and A^n be a transformation matrix defined as follows.

Definition 5 *The $m^n \times m^n$ matrix A^n is defined inductively by:*

1. $A^1 \stackrel{df}{=}$

$$\begin{bmatrix} a_{00} & 0 & 0 & \dots & 0 \\ a_{10} & a_{11} & 0 & \dots & a_{1(m-1)} \\ a_{20} & 0 & a_{22} & \dots & 0 \\ \dots & \dots & \dots & \dots & \\ a_{(m-2)0} & 0 & a_{(m-2)2} & \dots & 0 \\ a_{(m-1)0} & a_{(m-1)1} & 0 & \dots & a_{(m-1)(m-1)} \end{bmatrix},$$

where $\forall i, j \in M$, the coefficients a_{ij} are given by (2).

2. $A^n \stackrel{df}{=} A^1 \otimes A^{n-1}$,

where " $\otimes$ " denotes the Kronecker product of two matrices [16].

Clearly, if Theorem 3 is successively applied to the polynomials $A_m[f_{x_i=k}]$ of subfunctions $f_{x_i=k}$ about all the remaining variables, we will finally get an expression in which $A_m[f]$ is expanded in all the variables of f, allowing us to derive the coefficients of $A_m[f]$.

Theorem 4 *Every polynomial $A_m[f]$, $m > 2$, of an n-variable m-valued function f can be expressed in the following canonical form*

$$A_m[f] = \sum_{j=0}^{m^n-1} c_j \cdot x_1^{i_1} \cdot x_2^{i_2} \cdot \ldots \cdot x_n^{i_n},$$

where $(i_1 i_2 \ldots i_n)$ is the m-ary expansion of i, with i_1 being the least significant digit, and the coefficients c_i are given by the vector $C \stackrel{df}{=} [c_0 c_1 \ldots c_{m^n-1}]$ computed as $C = A^n \cdot F$.

Proof: By induction on n. We show the proof only for the case of m being odd. For m being even the proof is similar.

1) Let $n = 1$. According to Theorem 3, any polynomial $A_m[f]$ of a function

$f(x)$ of one variable x can be decomposed with respect to x as:

$$A_m[f] = a_{00} \cdot A_m[f_{x=0}] + \sum_{j=1}^{m-1} (a_{j0} \cdot A_m[f_{x=0}] + \\ + a_{jj} \cdot A_m[f_{x=j}] + a_{j(m-j)} \cdot A[f_{x=m-j}]) \cdot x^j,$$

where $f_{x=k} = f(k)$. By Lemma 2, $A_m[c] = c$ for any constant $c \in \mathbf{Z}_p$. So, we can express the above as:

$$A_m[f] = a_0 0 \cdot f(0) + \sum_{j=1}^{m-1} (a_{j0} \cdot f(0) + a_{jj} \cdot f(j) \ + \ a_{j(m-j)} \cdot f(m-j)) \cdot x^j,$$

which can be rewritten as

$$A_m[f] = \sum_{i=0}^{m-1} c_i x^i,$$

where $c_0 = a_{00} \cdot f(0)$ and $c_j = a_{j0} \cdot f(0) + a_{jj} \cdot f(j) + a_{j(m-j)} \cdot f(m-j)$, for all $j \in M - \{0\}$. Examining the structure of the matrix A^1, we can conclude that $C = A^1 \cdot F$.

2) Hypothesis: Assume the result for n. According to Theorem 3, any $A_m[f]$ of a function f of $n+1$ variables can be decomposed with respect to x_{n+1} in the following way:

$$A_m[f] = a_{00} A_m[f_{x_{n+1}=0}] + \sum_{j=1}^{m-1} (a_{j0} A_m[f_{x=0}] + \\ + a_{jj} A_m[f_{x=j}] + a_{j(m-j)} A_m[f_{x=m-j}]) \cdot x_{n+1}^j,$$

By the induction hypothesis, we can express each A_m of the subfunctions of n variables in the canonical form. We use the notation c_i^k to denote the ith coefficient of the canonical form of the subfunction $f_{x_{n+1}=k}$ and F_k for the truth table vector of $f_{x_{n+1}=k}$. To simplify the exposition, we also use the abbreviation X to stand for the term $x_1^{i_1} \cdot x_2^{i_2} \cdot \ldots \cdot x_n^{i_n}$. So, we replace each of $A_m[f_{x_{n+1}=k}]$,

$k \in M$, by $\sum_{i=0}^{m-1} c_i^k \cdot X$, with c_i^k given by $C_k = A^n \cdot F_k$. Then we get:

$$A_m[f] = a_{00} \cdot \sum_{i=0}^{m^n-1} c_i^0 \cdot X + \sum_{j=1}^{m-1} \left(a_{j0} \cdot \sum_{i=0}^{m^n-1} c_i^0 \cdot X + \right.$$
$$\left. + a_{jj} \cdot \sum_{i=0}^{n-1} c_i^j \cdot X + a_{j(m-j)} \cdot \sum_{i=0}^{m^n-1} c_i^{m-j} \cdot X \right) \cdot x_{n+1}^j$$

Since " $\cdot$ " is distributive over " $+$ ", we can re-order the above as

$$A_m[f] = \left(\sum_{i=0}^{m^n-1} a_{00} \cdot c_i^0 \cdot X \right) \cdot x_{n+1}^0 + \sum_{j=1}^{m-1} \left(\sum_{i=0}^{m^n-1} (a_{j0} \cdot \right.$$
$$\left. \cdot\, c_i^0 + a_{jj} \cdot c_i^j + a_{j(m-j)} c_i^{m-j}) \cdot X \right) \cdot x_{n+1}^j,$$

which can be re-written as

$$A_m[f] = \sum_{j=0}^{m^n-1} c_j \cdot x_1^{i_1} \cdot x_2^{i_2} \cdot \ldots \cdot x_n^{i_n},$$

where $c_i = a_{00} \cdot c_i^0$, for $0 \leq i \leq m-1$, and $c_i = a_{j0} \cdot c_i^0 + a_{jj} \cdot c_i^j + a_{j(m-j)} c_i^{m-j}$, for $j \cdot m \leq i \leq j \cdot m + m - 1$, for all $j \in M - \{0\}$. Since the coefficients c_i^j are given by $C_j = A^n \cdot F_j$, this is equivalent to $C = (A^1 \otimes A^n) \cdot F = A^{n+1} \cdot F$.

□

For example, for $m = 3$ and $n = 2$, the matrix A^2 is constructed as follows:

$$A^1 = \begin{bmatrix} 1 & 0 & 0 \\ -\frac{3}{2} & 2 & -\frac{1}{2} \\ \frac{1}{2} & -1 & \frac{1}{2} \end{bmatrix}$$

and $A^2 =$

$$\begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ -\frac{3}{2} & 2 & -\frac{1}{2} & 0 & 0 & 0 & 0 & 0 & 0 \\ \frac{1}{2} & -1 & \frac{1}{2} & 0 & 0 & 0 & 0 & 0 & 0 \\ -\frac{3}{2} & 0 & 0 & 2 & 0 & 0 & -\frac{1}{2} & 0 & 0 \\ \frac{9}{4} & -3 & \frac{3}{4} & -3 & 4 & -1 & \frac{3}{4} & -1 & \frac{1}{4} \\ -\frac{3}{4} & \frac{3}{2} & -\frac{3}{4} & 1 & -2 & 1 & -\frac{1}{4} & \frac{1}{2} & -\frac{1}{4} \\ \frac{1}{2} & 0 & 0 & -1 & 0 & 0 & \frac{1}{2} & 0 & 0 \\ -\frac{3}{4} & 1 & -\frac{1}{4} & \frac{3}{2} & -2 & \frac{1}{2} & -\frac{3}{4} & 1 & -\frac{1}{4} \\ \frac{1}{4} & -\frac{1}{2} & \frac{1}{4} & -\frac{1}{2} & 1 & -\frac{1}{2} & \frac{1}{4} & -\frac{1}{2} & \frac{1}{4} \end{bmatrix}.$$

So, we can compute $A_3[f]$ for $f = MIN(x_1, x_2)$ with $F = [0\ 0\ 0\ 0\ 1\ 1\ 0\ 1\ 2]$, as $C = A^2 \cdot F = [0\ 0\ 0\ 0\ \frac{5}{2}\ -1\ 0\ -1\ \frac{1}{2}]$, giving $A_3[f](x_1, x_2) = \frac{5}{2}x_1x_2 - x_1x_1^2 - x_1^2x_2 + \frac{1}{2}x_1^2x_2^2$.

4 Hash code computation

We compute the hash code of the function f by assigning randomly chosen integer values from $\mathbf{Z}_p$ to the input variables of f, and then evaluating $A_m[f]$ for this values. The resulting number is the *integer hash code* of f. This code requires less space than the canonical MDD representation of the same function and distinguishes any pair of multiple-valued functions with a quantifiable probability of success. The hash codes for two equivalent functions are always the same. Thus, the equivalence of two functions can be verified with a known probability of error, which arises from collisions between inequivalent functions. According to Schwartz-Zippel Theorem [17, p. 165], if the assignments of values of variables $x_1, \ldots, x_n$ are taken independently and uniformly at random from a field $\mathcal{F}$ of size $|\mathcal{F}|$, then the hash codes for two equivalent functions can be distinguished with the probability at least $\frac{n}{|\mathcal{F}|}$. In our case $\mathcal{F} = \mathbf{Z}_p$ and $|\mathcal{F}| = p$, so we get $\frac{n}{p}$. We are able to conclude this error bound in the following way:

Theorem 5 *Let f and g be the functions of type $M^n \to M$, such that $A_m[f] \neq A_m[g]$, and let $(a_1, \ldots, a_n)$ be an assignment of values of variables $x_1, \ldots, x_n$, whose elements are taken independently and uniformly at random from $\mathbf{Z}_p$. Then*

$$Prob(A_m[f](a_1, \ldots, a_n) \neq A_m[g](a_1, \ldots, a_n)) \geq \left(\frac{p-1}{p}\right)^n$$

Proof: It follows from Lemma 2 that, for any $(a_1, \ldots, a_n) \in \mathbf{Z}_p^n$, $A_m[f](a_1, \ldots, a_n) \neq A_m[g](a_1, \ldots, a_n)$ holds if and only if $A_m[f-g](a_1, \ldots, a_n) \neq 0$. To show, how

many n-tuples $(a_1, \ldots, a_n) \in \mathbf{Z}_p$ really do fulfill this property, we require the following lemma:

Lemma 4 *For any function $f_z : \mathbf{Z}_p^n \to \mathbf{Z}_p$, such that $f_z \neq 0$, and for any set $S \subseteq \mathbf{Z}_p$, $s = |S|$, there are at least $(s-1)^n$ assignments $(a_1, \ldots, a_n) \in S^n$ such that $A_m[f_z](a_1, \ldots, a_n) \neq 0$.*

Proof. For determining the number of assignments $(a_1, \ldots, a_n) \in S^n$ that fulfill the required property, we can make use of a binary encoding of the function f_z. For encoding a finite set $\mathbf{Z}_p$ with $|\mathbf{Z}_p| = p$ elements, we are using $\lceil \log p \rceil$ bits. Thus, f_z can be encoded as f_z:$\{0,1\}^{\lceil \log p \rceil \cdot n} \to \{0,1\}^{\lceil \log p \rceil}$. This means that f_z can be encoded in $\lceil \log p \rceil$ binary functions $f_{zi} : \{0,1\}^n \to \{0,1\}$, $1 \leq i \leq \lceil \log p \rceil$, and for each of these functions f_{zi} it can be shown that the required property holds. According to [15], the binary A-transform $A_2[f_{zi}]$ for f_{zi} can be written as:

$$\begin{aligned} A_2[f_{zi}] &= (1 - x_j) \cdot A_2[f_{zi_{x_j=0}}] + x_j \cdot A_2[f_{zi_{x_j=1}}] \\ &= x_j \cdot A_2[f_{zi_{x_j=1}} - f_{zi_{x_j=0}}] + A_2[f_{zi_{x_j=1}}] \end{aligned}$$

If $f_{zi_{x_j=1}} = f_{zi_{x_j=0}}$, then $A_2[f_{zi_{x_j=1}} - f_{zi_{x_j=0}}] = 0$. By applying induction on n, it is easy to show that there are at least $(s-1)^{n-1}$ assignments $(a_1, \ldots, a_{j-1}, a_{j+1}, \ldots, a_n) \in S^{n-1}$, such that $A_2[f_{zi_{x_j=0}}] \neq 0$. Since x_j can be chosen arbitrarily, we have $s(s-1)^{n-1} > (s-1)^n$ possible assignments resulting in $A_2[f_{zi}] \neq 0$.

Otherwise, there are at least $(s-1)^{n-1}$ assignments $(a_1, \ldots, a_{j-1}, a_{j+1}, \ldots, a_n) \in S^{n-1}$, such that $A_2[f_{zi_{x_j=1}} - f_{i_{x_j=0}}] \neq 0$. Since x_j can be chosen arbitrarily, with $x_j \neq -\frac{A_2[f_{zi_{x_j=0}}]}{A_2[f_{zi_{x_j=1}} - f_{zi_{x_j=0}}]}$, we have $(s-1)(s-1)^{n-1} = (s-1)^n$ possible assignments which compute $A_2[f_{zi}] \neq 0$. □

Thus, we can conclude that there are at least $(p-1)^n$ out of p^n possible assignments such that $A_m[f - g](a_1, \ldots, a_n) \neq 0$ and the claimed probability

bound can be achieved. □

Note that the error is therefore bound by $\epsilon = 1 - (1 - \frac{1}{p})^n \approx 1 - e^{-n/p}$ and if $p \gg n$, then $1 - e^{-n/p} \approx \frac{n}{p}$. Thus, the error probability can be reduced by either choosing a larger field $\mathbf{Z}_p$, or by independently performing several experiments.

For an example, consider the 3-valued 2-variable function $f = MIN(x_1, x_2)$ and let the variables be assigned the random values $x_1 = 2$, $x_2 = 4$. Since $A_3[f] = \frac{5}{2}x_1x_2 - x_1x_1^2 - x_1^2x_2 + \frac{1}{2}x_1^2x_2^2$, the hash code for f is 4.

As another example, consider the function $g = MAX(MIN(2, \overset{2}{x}_1, \overset{0}{x}_2), MIN(1, \overset{2}{x}_1, \overset{1}{x}_2))$. Since $A_3[g] = -2x_1 + 2x_1x_2 - \frac{1}{2}x_1x_2^2 + 2x_1^2 - 2x_1^2x_2 + \frac{1}{2}x_1^2x_2^2$, if we assign $x_1 = 2$ and $x_2 = 4$ we obtain the same hash code 4 as for MIN function and therefore get collision between two inequivalent functions.

However, we can substantially decrease the probability of collision by making *multiple runs*. On each run, an independent set of input variable assignment is randomly chosen, and the two function values are computed. If the values differ, we are assured that the two functions are not the same. If they are equal, we choose a new set of input assignments and reevaluate. The probability of incorrectly deciding that the functions are equal decrease exponentially with the number of runs: if the error probability of a single run is ϵ, then after k runs the error probability is ϵ^k [18]. For example, if we make a second run for the functions specified above, with the random values $x_1 = 3$ and $x_2 = 1$, then the hash code for MIN function is 0 and hash code for g function is 3. So, we can conclude that $f \neq g$.

5 Conclusion

This paper develops supporting theory for a probabilistic method for equivalence checking of two multiple-valued functions. We define a functional transformation and use it to obtain hash codes from multiple-valued functions. Comparing hash codes allow us to verify the equivalence of two functions with a known probability of error.

We are currently developing an algorithm for efficient computing of hash codes. A straightforward way to compute a hash code is to build an MDD (or similar structure) from the input function and then to apply a procedure for reducing it to an appropriate integer. However, the efficiency of such a scheme is limited by the need to create and evaluate an MDD representation of the entire function. A better approach, which we are pursuing, is first to symbolically decompose the function, and then hash it incrementally. This can be done by hashing some of the function's parts, and then using them to complete the hashing of the entire function.

Another method for probabilistic equivalence checking, based on Haar transform, has been recently presented in [19]. It would be interesting to explore how selecting a different transformation type influences the time required to compute hash codes and accuracy of the result.

References

[1] E. Dubrova, H.Sack, "Probabilistic Verification of Multiple-Valued Functions", *30th International Symposium on Multiple-Valued Logic*, pp. 461-466, May 2000.

[2] R. E. Bryant, "Graph-based algorithm for Boolean function manipulation", *IEEE Transactions on Computers*, vol. C-35, no. 8, pp. 677-691, August 1986.

[3] D. M. Miller, "Multiple-valued logic design tools", *Proc. 23rd International Symposium on Multiple-Valued Logic*, pp. 2-11, May 1993.

[4] J. C. Muzio, T. C. Wesselkamper, *Multiple-Valued Switching Theory*, Adam Hilger Ltd Bristol and Boston, 1986.

[5] A. Nozoe et al., "A 256-Mb multilevel flash memory with 2 MB/s program rate for mass storage applications", *Proc. of 1999 IEEE Int. Solid-State Circuits Conference (ISSCC'99)*, pp. 110-111, Nov. 1999.

[6] T. Okuda, T. Murotani, "A four-level storage 4-Gb DRAM" *IEEE Journal of Solid-State Circuits*, vol. 32, no. 11, pp. 1743 - 1747, Nov. 1997.

[7] K. Shimabukuro, C. Zukeran, "Reconfigurable current-mode multiple-valued residue arithmetic circuits", *Proc. 28th International Symposium Multiple-Valued Logic*, pp. 282-287, May 1998.

[8] S. Wei, K. Shimizu, "Residue arithmetic multiplier based on the radix-4 signed-digit multiple-valued arithmetic circuits", *12th International Conference on VLSI Design*, pp. 212-217, January 1999.

[9] T. Hanyu, M. Kameyama, "A 200 MHz pipelined multiplier using 1.5 V-supply multiple-valued MOS current-mode circuits with dual-rail source-coupled logic", *IEEE Journal of Solid-State Circuits*, vol. 30, no. 11, pp. 1239-1245, Nov. 1995.

[10] A. F. Gonzalez, P. Mazumder, "Multiple-valued signed digit adder using negative differential resistance devices", *IEEE Transactions on Computers*, vol. 47, no. 9, pp. 947 - 959, Sept. 1998.

[11] The VIS Group, "VIS: A system for verification and synthesis", *Proc. 8th Int. Conf. on Computer Aided Verification*, Springer Lecture Notes in Computer Science, vol. 1102, pp. 428-432, 1996.

[12] R. Drechsler, M. Keim, B. Becker, "Fault simulation for sequential multi-valued logic networks", *Proc. 27th International Symposium on Multiple-Valued Logic*, pp. 145-150, May 1997.

[13] R. E. Bryant, C.-J. H. Seger, "Digital circuit verification using partially-ordered state models", *Proc. 24th Int. Symp. on Multiple-Valued Logic*, pp. 2-7, May 1994.

[14] R. Drechsler, "Verification of multiple-valued logic networks", *Proc. 26th International Symposium on Multiple-Valued Logic*, pp. 10-15, May 1996.

[15] J. Jain, J. Bitner, D. S. Fussell, J. A. Abraham, "Probabilistic verification of Boolean functions", *Formal Methods in System Design*, Kluwer Academic Publishers, vol. 1, pp. 63-117, January 1992.

[16] G. Birkhoff, S. MacLane, *Brief Survey of Modern Algebra*, 4th ed., New York, Macmillan, 1977.

[17] R. Motwani, P. Raghavan, *Randomized Algorithms*, Cambridge University Press, 1995.

[18] M. Blum, A. K. Chandra, M. N. Wegman, "Equivalence of free Boolean graphs can be decided probabilistically in polynomial time", *Information Processing letters*, vol. 10, no. 2, pp. 80-82, Febr. 1980.

[19] M. A. Thornton, R. Drechsler, W. Günther, "Probabilistic equivalence checking using partial Haar spectral diagrams", *Proc. 4th International Workshop on Applications of Reed-Muller Expansion in Circuit Design*, pp. 123-132, August 1999.