Probabilistic Verification 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 the equivalence of two polynomials is checked probabilistically. 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, arising from collisions between inequivalent functions. Such a probabilistic verification can be an attractive alternative for verifying functions that are too large to be handled by deterministic verification methods.

1. Introduction

In recent years, advances in integrated circuit technology made feasible fabrication of several commercial products benefiting from multiple-valued logic, such as 256-Mbit 4-valued flash memory [1] and 4-Gbit 4-valued DRAM [2]. These products can be seen as first steps toward recognition of the increasing role of multiple-valued logic in the next generation of electronic systems. However, for further practical utilization, efficient computer-aided tools for design, testing and verification of multiple-valued logic circuits are needed. Some existing tools, such as Berkeley's tool for verification and synthesis VIS [3], provide a solution for the special case of multiple-valued input binary-valued output functions, but the general problem is still open.

This paper focuses on the problem of verification 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 [4], [5]. Similarly, verification procedures employing Reduced Ordered Binary Decision Diagrams (ROBDDs) [6] can adapted to Multiple-valued Decision Diagrams (MDD) [7] as shown in [8]. However, the MDD verification methods representing functions as single, monolithic graph might be infeasible for large functions. We believe that deterministic methods of verification will not be practical for the multiple-valued logic domain due to the increasing complexity of the problem.

In this paper we present a probabilistic method for design verification of multiple-valued logic functions, which generalizes the method introduced in [9]. We define a functional transformation A which converts a multiple-valued function $f : M^n \rightarrow M$ on a set $M \overset{df}{=} \{0, 1, \ldots, m-1\}$ into a polynomial of type $A_m[f] : Z_p^n \rightarrow Z_p$ over a finite field of integers 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 randomly chosen values of x_i from Z_p, $i \in \{1, \ldots, n\}$. 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, 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 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 A-transform

To define the transformation A, we 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 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 $m > 1$, $w : Z_p \times Z_p \to Z_p$ 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 $\overset{i}{x}$:

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

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

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 $W_n : Z_p^{2n} \to Z_p$ 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 general functions $Z_p^n \to Z_p$ over the field Z_p. Note that, since $M \subset Z_p$, the multiple-valued functions $M^n \to M$ are a subset of the field functions $Z_p^n \to Z_p$. While a field function is defined for inputs $x_i \notin M$ as well, the values which the function produces for such assignments do not participate in the definition. To distinguish between multiple-valued and field functions, 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 field functions of type $Z_p^n \to Z_p$.

Definition 3 *Given a function of type $f_z : Z_p^n \to Z_p$ and $m > 1$, the polynomial $A_m[f_z] : Z_p^n \to Z_p$ 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

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

E.g. for the 3-valued function $f(x_1, x_2) = MIN(x_1, x_2)$, the corresponding polynomial is

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

Note that the A-transform is defined only for assignments $(b_1, \ldots, b_n) \in M^n$. Therefore, if two field functions f_z and g_z have the same values for all $(b_1, \ldots, b_n) \in M^n$, then they are treated identically by the A-transform. We say that two such functions are *m-equivalent*:

Definition 4 *The functions f_z and g_z of type $Z_p^n \to Z_p$ 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 $f_z \overset{m}{=} g_z$ to denote that f_z and g_z are m-equivalent. If both f and g are multiple-valued functions of type $M^n \to M$ then $f \overset{m}{=} g$ is the same as $f = g$.

By Definition 3, $f_z \overset{m}{=} g_z$ implies $A_m[f_z] = A_m[f_z]$. However, can we conclude $A_m[f_z] \neq A_m[f_z]$ from $f_z \overset{m}{\neq} g_z$? 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 $f_z : Z_p^n \to Z_p$, $A_m[f_z] \overset{m}{=} f_z$.*

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

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

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 $A_m[f_z] \overset{m}{=} f_z$. □

It follows from the theorem that, though applying the A-transform to a multiple-valued function f increases the domain of f from M^n to Z_p^n, the polynomial $A_m[f]$ still yields the same values as f when evaluated for an 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. Computing the A-transform

Computing A-transforms using Definition 3 is feasible for very small functions. For larger functions, we developed 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. $f_{x_i=j} \overset{df}{=} f(x_1, \ldots, x_{i-1}, j, x_{i+1}, \ldots, x_n)$.

We can apply the following decomposition to a polynomial $A_m[f]$, which can be considered as a generalization of Shannon decomposition of Boolean functions:

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 $x_1, \ldots, x_n$.

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

{Definition 3}

$$= \sum_{j=0}^{m-1} \sum_{(b_2,..,b_n) \in M^{n-1}} f(j, b_2, .., b_n) \cdot W_n(j, b_2, ..., b_n, X)$$

{re-grouping}

$$= \sum_{j=0}^{m-1} \sum_{(b_2,\ldots,b_n) \in M^{n-1}} f(j, b_2, \ldots, b_n) \cdot \prod_{k \in M - \{j\}} \tfrac{k-x_1}{k-j} \cdot$$
$$\cdot W_{n-1}(b_2, \ldots, b_n, x_2, \ldots, x_n)$$

$\{w(j, x_1) = \prod_{k \in M - \{j\}} \frac{k-x_1}{k-j}$, Definition 2$\}$

$$= \sum_{j=0}^{m-1} \left(\prod_{k \in M - \{j\}} \tfrac{k-x_1}{k-j} \right) \cdot A_m[f_{x_1=j}]$$

{Definition 3} □

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 1 *For any variable x from Z_p, any fixed $j \in M$, and $m > 2$:*

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

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

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

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

where " $\oplus_m$ " denotes addition modulo m and all other operations are regular arithmetic operations in 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:

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

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

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

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

From linear algebra we know that such a system always has a solution, and this solution is unique [10]. 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 [10], 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 for the term $\prod_{k \in M - \{j\}} \frac{k-x}{k-j}$ can be simplified to:

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

□

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, $\overset{j}{x} \overset{m}{=} (m-1) \cdot \prod_{k \in M - \{j\}} \frac{k-x}{k-j}$, and thus, from Lemma 1, we can conclude that

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

We use Theorem 2 and Lemma 1 to derive another type of decomposition of $A_m[f]$, which will be used later to derive a canonical expansion for $A_m[f]$. Before giving the decomposition, we first summarize some properties of $A_m[f]$, obvious from Definition 3.

Lemma 2 *For any field functions f_z and g_z and any constant $c \in Z_p$,*
(a) $A_m[c \cdot f_z] = c \cdot A_m[f_z]$.
(b) $A_m[f_z + g_z] = A_m[f_z] + A_m[g_z]$.

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*

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

case 2: *if m is even, then*

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

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

Proof: $A_m[f]\ =$

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

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

□

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

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

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

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

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

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:

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 $C \overset{df}{=} [c_0 c_1 \ldots c_{m^n-1}]$ 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 - 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 $c \in Z_p$. 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 re-written 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 $C = (A^1 \otimes A^n) \cdot F = A^{n+1} \cdot F$. $\Box$

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

4. Computing hash code of the function

We compute the hash code of the function f by assigning randomly chosen integer values from 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 [11, 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 $\mathcal{F} = Z_p$ and $|\mathcal{F}| = p$, so we get $\frac{n}{p}$. A smaller error bound $\left(\frac{p-1}{p}\right)^n$ is derived in [9].

As 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 $A_3[f] = \frac{5}{2} x_1 x_2 - x_1 x_1^2 - x_1^2 x_2 + \frac{1}{2} x_1^2 x_2^2$, 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 $A_3[g] = -2x_1 + 2x_1 x_2 - \frac{1}{2} x_1 x_2^2 + 2x_1^2 - 2x_1^2 x_2 + \frac{1}{2} x_1^2 x_2^2$, 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 re-evaluate. 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 e, then after k runs the error probability is e^k [12]. 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 lays a theoretical foundation of a probabilistic method for verification of multiple-valued functions. We define a functional transformation, which is can be used to obtain an integer code from a multiple-valued function.

The integer codes allow us to determine probabilistically whether two functions are equivalent.

Further research remains designing an efficient procedure for computing integer hash codes. A simple way to compute a hash code would be to build an MDD or similar structure from the input function and then apply a procedure for reducing this structure to an appropriate integer. Of course, the efficiency of such a scheme would be limited by the need to create and evaluate an MDD representation of the entire function. A better approach, which we are currently pursuing, is first to symbolically decompose the function, and then hash it incrementally. This can be done by first hashing some of its parts, and then using these more compact intermediate forms to complete the hashing of the entire function.

References

[1] 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)*, (1999), 110-111.

[2] T. Okuda, T. Murotani, A four-level storage 4-Gb DRAM *IEEE Journal of Solid-State Circuits* **32**, 11, (1997), 1743 - 1747.

[3] The VIS Group, VIS: A system for verification and synthesis, *Proc. 8th Int. Conf. on Computer Aided Verification*, Springer Lecture Notes in Computer Science, **1102**, Edited by R. Alur and T. Henzinger, New Brunswick, NJ, (1996), 428-432.

[4] R. Drechsler, M. Keim, B. Becker, Fault simulation for sequential multi-valued logic networks, *Proc. 27th Int. Symp. on Multiple-Valued Logic* (1997), 145-150.

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

[6] R.E. Bryant, Graph-based algorithm for Boolean function manipulation, *IEEE Transactions on Computers* **C-35** No. 8 (1986), 677-691.

[7] D. M. Miller, Multiple-valued logic design tools, *Proc. 23rd Int. Symp. on Multiple-Valued Logic* (1993), 2-11.

[8] R. Drechsler, Verification of multiple-valued logic networks, *Proc. 26th Int. Symp. on Multiple-Valued Logic* (1996), 10-15.

[9] J. Jain, J. Bitner, D. S. Fussell, J. A. Abraham, Probabilistic verification of Boolean functions, *Formal Methods in System Design*, Kluwer Academic Publishers, **1**, (1992), 63-117.

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

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

[12] M. Blum, A. K. Chandra, M. N. Wegman, Equivalence of free Boolean graphs can be decided probabilistically in polynomial time, *Information Processing letters* **10**, No. 2, (1980), 80-82.