J. of Mult.-Valued Logic & Soft Computing, Vol. 0, pp. 1–23
Reprints available directly from the publisher
Photocopying permitted by license only

A Linear Decomposition of Index Generation Functions: Optimization using Autocorrelation Functions

TSUTOMU SASAO

Dept. of Computer Science, Meiji University, Kawasaki, Kanagawa 214-8571, Japan
E-mail: sasao@ieee.org

Accepted in final form on April 2, 2015

This paper shows that autocorrelation functions are useful to find a decomposition of an index generation functions: $F(x_1, x_2, \ldots, x_n) = G(y_1, y_2, \ldots, y_p)$, where y_j $(j = 1, 2, \ldots, p)$ are linear functions of $x_1, x_2, \ldots x_{n-1}$, and x_n. It also shows a strategy to reduce the number of variables p to represent $F(x_1, x_2, \ldots, x_n)$ using an autocorrelation functions.

1 INTRODUCTION

Various methods exist to decompose logic functions into linear and non-linear parts. They are used to simplify AND-OR logic circuits [3, 5, 7, 9, 22], or to simplify binary decision diagrams [4, 6, 8].

The first important observation is that the linear part can be implemented with lower cost than the non-linear part [7, 9]. To find a linear part of a decomposition, Walsh spectrum [3] and autocorrelation functions [22] are used. Since autocorrelations are invariant under a linear transformation of the input variables [10], they are useful for linear decompositions.

The second important observation is that an incompletely specified function often can be represented with a fewer variables than the original number of the variables [1].

In this paper, we use linear decompositions to realize incompletely specified functions. This class of functions often appear in the practical circuits, and their cost can be reduced drastically with linear decompositions.

Given an incompletely specified function F, the problem is to find the decomposition

$$F(x_1, x_2, \ldots, x_n) = G(y_1, y_2, \ldots, y_p),$$

where y_i $(i = 1, 2, \ldots, p)$ are linear functions of $x_1, x_2, \ldots, x_n$, and the number of variables p is the minimum. When a given function is defined only for k input combinations, the number of variables p can be reduced to $2\lceil \log_2 k \rceil - 3$, in many cases. By this, the circuit cost can be reduced drastically.

In this paper, we show that the linear decomposition that reduces the number of variables can be obtained by the autocorrelation function. The rest of the paper is organized as follows: Section 2 defines index generation functions. Section 3 explains the index generation unit, that realizes an index generator function efficiently. Section 4 shows the number of variables to represent an incompletely specified index generation function. Section 5 shows a method to reduce the number of variables by a linear decomposition. Section 6 introduces autocorrelation functions. Section 7 shows properties of autocorrelation functions. Section 8 shows a method to find good linear transformations. Section 9 shows experimental results, and Section 10 summarizes the paper. A preliminary version of this paper was presented at ISMVL-2013 [19].

2 INDEX GENERATION FUNCTION

Consider a set M of k different vectors of n bits. These vectors are called **registered vectors**. A function that associates each element from M with a unique integer from $\{1, 2, \ldots, k\}$ is an **index generation function** [13, 17]. More precise definition follows:

Definition 1. *An* **incompletely specified index generation function** *of n variable is a one-to-one mapping:*

$$F : M \rightarrow \{1, 2, \ldots, k\},$$

where $|M| = k$, and $M \subset \{0, 1\}^n$. We say that k is the **weight** *of the function.*

The corresponding **completely specified index generation function** *is a mapping:*

$$F_c : \{0, 1\}^n \rightarrow \{0, 1, 2, \ldots, k\},$$

where $F_c(\vec{a}) = F(\vec{a})$ when $\vec{a} \in M$, and $F_c(\vec{a}) = 0$ when $\vec{a} \in B^n - M$.

Vector							Index
x_1	x_2	x_3	x_4	x_5	x_6	x_7	
1	0	0	0	0	0	0	1
0	1	0	0	0	0	0	2
0	0	1	0	0	0	0	3
0	0	0	1	0	0	0	4
0	0	0	0	1	0	0	5
0	0	0	0	0	1	0	6
0	0	0	0	0	0	1	7

TABLE 1
Registered Vector Table in Example 3

Example 1. *Consider the registered vectors shown in Table 1. These vectors, with associate indexes, represent a 7-variable index generation function with weight* $k = 7$.

Index generation functions are useful for address tables for the Internet, terminal access controllers of local area networks, databases, memory patch circuits, text compressions, password tables, code converters, and computer virus scanning engines [13,16,17]. In many cases, functions must be updated frequently. Thus, a reconfigurable architecture is desirable for the implementation. The next section shows an efficient method to implement an index generation function.

3 INDEX GENERATION UNIT

Figure 1 shows an **index generation unit** (IGU) [14]. The **linear circuit** has n inputs and p outputs, where $p \leq n$. It produces linear functions of

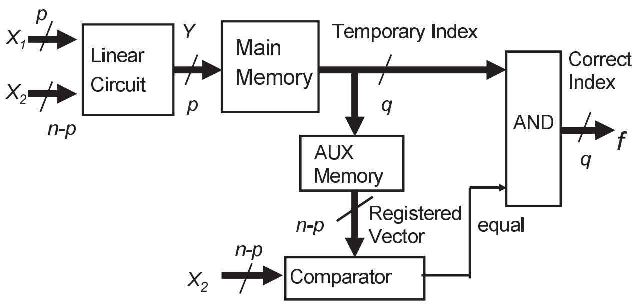


FIGURE 1
Index Generation Unit.

the input variables. It is used to reduce the size of the main memory. Let $X_1 = (x_1, x_2, \ldots, x_p)$ and $X_2 = (x_{p+1}, x_{p+2}, \ldots, x_n)$. The **main memory** has p inputs and $q = \lceil \log_2(k+1) \rceil$ outputs. The main memory produces correct indices only for registered vectors. However, it may produce incorrect indices for non-registered vectors, because the number of input variables is less than n. In a completely specified index generation function, if the input vector is non-registered, then it should produce 0 outputs. To check whether the main memory produces the correct index or not, we use the **AUX memory**. The AUX memory has $q = \lceil \log_2(k+1) \rceil$ inputs and $n - p$ outputs: It stores the X_2 part of the registered vectors for each index. The **comparator** checks if the X_2 part of the inputs is the same as the X_2 part of the registered vector. If they are the same, then the main memory produces a correct index. Otherwise, the main memory produces an incorrect index, and the input vector is non-registered. In this case, the **output AND gates** produce 0, showing that the input vector is non-registered. Note that the main memory produces the correct index only for the registered vectors. Thus, in the main memory, we can implement **an incompletely specified index generation function** instead of the completely specified function. The cost of the main memory is $q2^p$, and the cost of the AUX memory is $(n-p)2^q$. Thus, the total memory cost is

$$q2^p + (n-p)2^q.$$

When $p < n$, this cost is lower than the cost of the single-memory realization: $q2^n$. We assume that the cost of the linear part is proportional to np, which is much smaller than the cost of the memories, and can be neglected. Thus the reduction of the number of variables p is quite important. With this technique, we can reduce the cost of the IGU drastically. For the detail operations of the IGU, see [20].

4 NUMBER OF VARIABLES TO REPRESENT INCOMPLETELY SPECIFIED FUNCTIONS

In an incompletely specified function F, *don't care* values can be chosen either as 0 or 1, to minimize the number of variables to represent F. This property is useful to realize the function with a smaller look-up table (LUT).

Theorem 1. *Suppose that an incompletely specified function F is represented by a decomposition chart [11]. If each column has at most one care element, then the function can be represented by using only the column variables.*

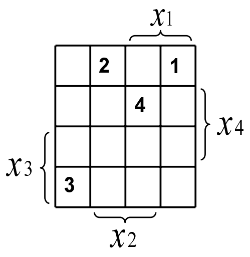


FIGURE 2
Index Generation Function in Example 2.

Proof. In each column, let the values of *don't cares* elements be set to the value of the *care* element in the column, then the function depends only on the column variables.

Example 2. *Consider the decomposition chart shown in Figure 2, where x_1 and x_2 specify the columns, and x_3 and x_4 specify the rows, and blank elements denote don't cares. Note that in Figure 2, each column has at most one care element. Thus, this function can be represented by only the column variables x_1 and x_2:*

$$F = 1 \cdot x_1\bar{x}_2 \vee 2 \cdot \bar{x}_1 x_2 \vee 3 \cdot \bar{x}_1\bar{x}_2 \vee 4 \cdot x_1 x_2,$$

where $m \cdot 1 = m,\ m \cdot 0 = 0$, and $m \vee 0 = 0 \vee m = m$ for $m \in \{1, 2, 3, 4\}$.

Algorithms to minimize the number of variables in incompletely specified functions have been developed [1,2,12,14]. They can be reduced to a minimum covering problem. (Detail will be shown in Section 8.1). As for incompletely specified index generation functions, we have the following [16]:

Conjecture 1. *When the number of the input variables is sufficiently large, most incompletely specified index generation functions with weight k (≥ 8) can be represented by $2\lceil \log_2 k \rceil - 3$ variables.*

Example 3. *Consider the registered vectors shown in Table 1. It shows an index generation function with weight $k = 7$. To represent this function by using $\{x_1, x_2, \ldots, x_7\}$, at least 6 variables are necessary.*

However, for such functions, as shown in the next section, by a linear decomposition, the number of variables to represent the function can be reduced. When we use a linear decomposition, almost all functions can be represented with the number of variables given by Conjecture 1.

As for the lower bound on the number of variables, we have the following:

Theorem 2. *[18] To represent any incompletely specified index generation function F with weight k, at least $q = \lceil \log_2 k \rceil$ variables are necessary.*

Thus, when the weight k of an n-variable index generation function is greater than 2^{n-1}, we cannot reduce the number of variables.

5 REDUCTION OF THE NUMBER OF VARIABLES BY LINEAR DECOMPOSITIONS

In the previous section, we showed a method to reduce the number of variables for a particular class of incompletely specified functions. Unfortunately, the applicability of such method is limited. In this section, we show that an index generation function can be represented with fewer variables by using a linear decomposition.

Example 4. *For the function in Table 1, each registered vector has exactly one component equal to 1. Thus, the number of variables for the function can be reduced to six: Any one variable can be removed. If we remove x_7, then we have:*

$$\begin{aligned} F = {} & 1 \cdot x_1\bar{x}_2\bar{x}_3\bar{x}_4\bar{x}_5\bar{x}_6 \vee 2 \cdot \bar{x}_1x_2\bar{x}_3\bar{x}_4\bar{x}_5\bar{x}_6 \\ & \vee 3 \cdot \bar{x}_1\bar{x}_2x_3\bar{x}_4\bar{x}_5\bar{x}_6 \vee 4 \cdot \bar{x}_1\bar{x}_2\bar{x}_3x_4\bar{x}_5\bar{x}_6 \\ & \vee 5 \cdot \bar{x}_1\bar{x}_2\bar{x}_3\bar{x}_4x_5\bar{x}_6 \vee 6 \cdot \bar{x}_1\bar{x}_2\bar{x}_3\bar{x}_4\bar{x}_5x_6 \\ & \vee 7 \cdot \bar{x}_1\bar{x}_2\bar{x}_3\bar{x}_4\bar{x}_5\bar{x}_6. \end{aligned}$$

However, we cannot remove two or more variables simultaneously.

Definition 2. *A* **linear function** *of $x_1, x_2, \ldots, x_n$ is defined as*

$$y = c_1x_1 \oplus c_2x_2 \oplus c_3x_3 \oplus \ldots \oplus c_nx_n,$$

where $c_j \in \{0, 1\}$. $\sum_{j=1}^{n} c_j$ is called a **compound degree** *of y.*

Problem 1. *Given an incompletely specified index generation function F, find a* **linear decomposition***:*

$$F(x_1, x_2, \ldots, x_n) = G(y_1, y_2, \ldots, y_p),$$

Vector			Index
y_3	y_2	y_1	
0	0	1	1
0	1	0	2
0	1	1	3
1	0	0	4
1	0	1	5
1	1	0	6
1	1	1	7

TABLE 2
Registered Vectors after Linear Transformation of Table 1

where $y_i (i = 1, 2, \ldots, p)$ *are linear functions, and* G *is an index generation function.*

A linear decomposition that minimizes the value of p is **optimum**.

Example 5. *For the function in Table 1, consider the linear decomposition:*

$$F(x_1, x_2, \ldots, x_7) = G(y_1, y_2, y_3),$$

where

$$\begin{aligned} y_1 &= x_1 \oplus x_3 \oplus x_5 \oplus x_7, \\ y_2 &= x_2 \oplus x_3 \oplus x_6 \oplus x_7, \\ y_3 &= x_4 \oplus x_5 \oplus x_6 \oplus x_7. \end{aligned}$$

The transformed registered vectors that represents the function $G(y_1, y_2, y_3)$ *are shown in Table 2. In this case, three variables* (y_3, y_2, y_1) *distinguish 7 vectors, and represents the original function* $F(x_1, x_2, \ldots, x_7)$*. Note that this is an optimum transformation.*

By Theorem 2, if the linear decomposition reduces the number of variables to $q = \lceil \log_2 k \rceil$, then it is optimum. Note that for some functions, this lower bound is not achieved. In fact, there exists a 4-variable index generation function with weight 6, for which any linear decomposition require four variables. In this case, $q = 3$ and $p = 4$.

From here, we are going to obtain a linear decomposition that reduces the number of variables to represent the given index generation function.

Definition 3. *Consider an incompletely specified index generation function of* n *variables:*

$$F : M \rightarrow \{1, 2, \ldots, k\}, M \subset \{0, 1\}^n.$$

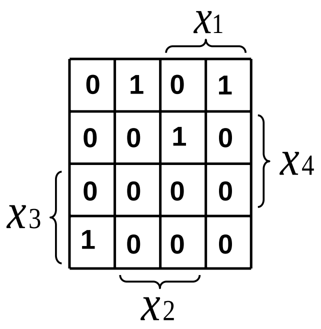


FIGURE 3
Characteristic Function for Figure 2.

The corresponding **characteristic logic function** *is*

$$f : \{0, 1\}^n \to \{0, 1\},$$

where

$$f(\vec{x}) = \begin{cases} 1 & (\vec{x} \in M) \\ 0 & \text{(Otherwise)}. \end{cases}$$

In other words, the characteristic logic function is a characteristic function of M.

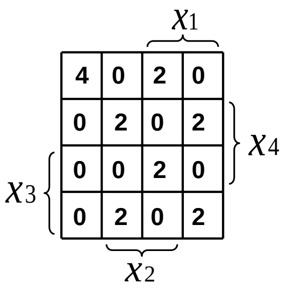
Example 6. *Figure 3 shows the characteristic logic function for the incompletely specified index generation function shown in Figure 2.*

Note that two index generation functions whose indices are permutated have the same characteristic logic function. Also, they require the same number of variables in linear decompositions. From this, we have the following:

Lemma 1. *Incompletely specified index generation functions sharing the same characteristic logic functions require the same number of variables in their linear decompositions.*

6 AUTOCORRELATION FUNCTION

In this part, we show that an autocorrelation function is useful to find the number of variables to represent an incompletely specified index generation function.



4	0	2	0
0	2	0	2
0	0	2	0
0	2	0	2

FIGURE 4
Autocorrelation Function for Figure 3.

Definition 4. *Let f be an n-variable logic function. Then, the the* **autocorrelation function** *of f is defined as*

$$B_f(\vec{\tau}) = \sum_{\vec{v}\in\{0,1\}^n} f(\vec{v}) \cdot f(\vec{v} \oplus \vec{\tau}),$$

where $\vec{\tau} \in \{0, 1\}^n$.

Example 7. *Figure 4 shows the autocorrelation function of the characteristic logic function in Figure 3.*

Definition 5. *Let $\vec{e}_i = (0, \ldots, 0, 1, 0, \ldots, 0)$ be the* **unit vector** *whose i-th component is 1.*

First, we consider the condition that a single variable can be removed.

Lemma 2. *Let f be the characteristic logic function of an incompletely specified index generation function F. Then, F can be represented without x_i iff $B_f(\vec{e}_i) = 0$.*

Proof. Consider the decomposition chart, where the row variable is x_i, and the column variables are the remaining variables. Suppose that F can be represented without using x_i. In this case, each column has at most one non-zero element. This means that $f(\vec{\tau}) \cdot f(\vec{\tau} \oplus \vec{e}_i) = 0$ for each $\vec{\tau}$. In other words, $B_f(\vec{e}_i) = 0$.

On the contrary, assume that $B_f(\vec{e}_i) = 0$. In this case, we have $f(\vec{\tau}) \cdot f(\vec{\tau} \oplus \vec{e}_i) = 0$ for each $\vec{\tau} \in B^n$. If we consider the decomposition chart, where x_i denotes the row variables, then it has hat most one non-zero element for each column. By Theorem 1, F can be represented without using x_i.

The following lemma shows the condition that two variables can be removed simultaneously:

Lemma 3. *Let f be the characteristic logic function of an incompletely specified index generation function F. Then, F can be represented without x_i and x_j iff $B_f(\vec{e}_i) = 0$, $B_f(\vec{e}_j) = 0$, and $B_f(\vec{e}_i \vee \vec{e}_j) = 0$.*

Proof. Consider the decomposition chart, where the row variables are x_i and x_j, and the column variables are remaining variables. Suppose that F can be represented without x_i and x_j. In this case, each column has at most one non-zero element. This means that

$$\begin{aligned} f(\vec{\tau}) \cdot f(\vec{\tau} \oplus \vec{e}_i) &= 0 \\ f(\vec{\tau}) \cdot f(\vec{\tau} \oplus \vec{e}_j) &= 0 \\ f(\vec{\tau}) \cdot f(\vec{\tau} \oplus \vec{e}_i \oplus \vec{e}_j) &= 0 \end{aligned}$$

for each $\tau \in B^n$. In other words, $B_f(\vec{e}_i) = 0$, $B_f(\vec{e}_j) = 0$, and $B_f(\vec{e}_i \oplus \vec{e}_j) = 0$. Note that $\vec{e}_i \oplus \vec{e}_j = \vec{e}_i \vee \vec{e}_j$.

On the contrary, assume that $B_f(\vec{e}_i) = 0$, $B_f(\vec{e}_j) = 0$, and $B_f(\vec{e}_i \oplus \vec{e}_j) = 0$, for each $\vec{\tau} \in B^n$. Consider the decomposition chart, where x_i and x_j denote the row variables. In this case, each column of the decomposition chart has at most one non-zero element. This means that F can be represented without using x_i and x_j.

Example 8. *Consider the incompletely specified function F shown in Figure 2. From the autocorrelation function in Figure 4, we can see that*

- *F can be represented without x_1, since $B_f(1, 0, 0, 0) = 0$.*
- *F can be represented without x_2, since $B_f(0, 1, 0, 0) = 0$.*
- *F can be represented without x_3 and x_4, since $B_f(0, 0, 1, 0) = B_f(0, 0, 0, 1) = B_f(0, 0, 1, 1) = 0$.*

Example 9. *First, consider the incompletely specified index generation function shown in Figure 5. The non-zero coefficients of the autocorrelation function are shown in Table 3. The last column of the table denotes the weight of $\vec{\tau}$, the number of 1's in the vector. Since $B_f(\vec{e}_i) = 0$ for $i = 1, 2, 3, 4$, any single variable can be removed. However, since $B_f(\vec{e}_i \vee \vec{e}_j) \neq 0$ for $(1 \leq i < j \leq 4)$, two variables cannot be removed simultaneously. Thus, we need at least three variables to represent F.*

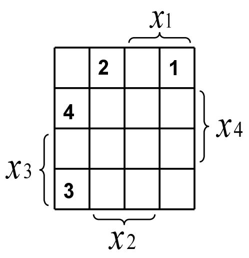


FIGURE 5
Index Generation Function in Example 9.

Second, consider the transformation:

$$\begin{aligned} y_1 &= x_1 \oplus x_4, \\ y_2 &= x_2 \oplus x_4, \\ y_3 &= x_3, \\ y_4 &= x_4. \end{aligned}$$

Since we obtain a linear transformation iteratively, from here we use the following notation:

$$\begin{aligned} x_1 &\Leftarrow x_1 \oplus x_4, \\ x_2 &\Leftarrow x_2 \oplus x_4, \end{aligned}$$

Since x_3 and x_4 are unchanged, such transformations are omitted in the above notation.

The map for the transformed function is shown in Figure 2. Note that when $x_4 = 1$, the element 4 is shifted to the right column by two. Table 4 shows the

$\vec{\tau}$				$B_f(\vec{\tau})$	$\|\vec{\tau}\|$
x_1	x_2	x_3	x_4		
0	0	0	0	4	0
0	0	1	1	2	2
0	1	0	1	2	2
0	1	1	0	2	2
1	0	0	1	2	2
1	0	1	0	2	2
1	1	0	0	2	2

TABLE 3
Autocorrelation Coefficients for Figure 5

$\vec{\tau}$				$B_f(\vec{\tau})$	$\|\vec{\tau}\|$
x_1	x_2	x_3	x_4		
0	0	0	0	4	0
1	1	1	1	2	4
1	0	0	1	2	2
0	1	1	0	2	2
0	1	0	1	2	2
1	0	1	0	2	2
1	1	0	0	2	2

TABLE 4
Autocorrelation Coefficients for Figure 2

non-zero coefficients of the autocorrelation function. In this case,

$$B_f(\vec{e}_3) = B_f(\vec{e}_4) = B_f(\vec{e}_3 \vee \vec{e}_4) = 0.$$

Thus, we can remove two variables x_3 and x_4 simultaneously, and the function can be represented with only x_1 and x_2.

Third, consider the transformation in Figure 2:

$$x_2 \quad \Leftarrow \quad x_2 \oplus x_3.$$

We have the function shown in Figure 6. Note that when $x_3 = 1$, the element 3 is moved to the right. The autocorrelation function is shown in Table 5. In this case, $B_f(\vec{e}_3) = 2$. This means that to represent F, we need x_3. However, we can remove any one of the other variables.

In general, the condition to remove s variables simultaneously is given by

Theorem 3. *Let f be the characteristic logic function of an incompletely specified index generation function F. Then, F can be represented without*

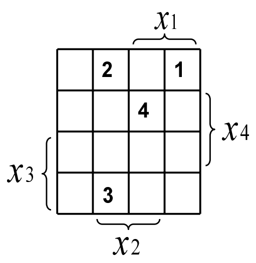


FIGURE 6
Index Generation Function in Example 9.

$\vec{\tau}$				$B_f(\vec{\tau})$	$\lvert\vec{\tau}\rvert$
x_1	x_2	x_3	x_4		
0	0	0	0	4	0
1	0	1	1	2	3
1	0	0	1	2	2
0	0	1	0	2	1
0	1	0	1	2	2
1	1	1	0	2	3
1	1	0	0	2	2

TABLE 5
Autocorrelation Coefficients for Figure 6

x_{t_1}, x_{t_2}, ..., and x_{t_s}, iff $B_f(\vec{t}) = 0$, where $\vec{t} = a_1\vec{e}_{t_1} \vee a_2\vec{e}_{t_2} \vee \ldots \vee a_s\vec{e}_{t_s}$ for $a_i \in \{0,1\}$, $\vec{t} \neq \vec{0}$.

Proof. Consider the decomposition chart of f, where the row variables are $x_{i1}, x_{i2}\ldots$, and x_{is}, and the column variables are the remaining variables. Suppose that F can be represented without $x_{i1}, x_{i2}\ldots, x_{is}$. In this case, each column has at most one non-zero element. This means that $f(\vec{\tau}) \cdot f(\vec{\tau} \oplus \vec{e}_{ij}) = 0$ for each $\tau \in B^n$, and each e_{ij},($j = 1, 2, .., s$). Similarly to the proof of Lemma 3, we have the theorem.

Example 10. *Consider the incompletely specified index generation function shown in Table 1. The non-zero coefficients of the autocorrelation function are shown in Table 6. Since $B_f(\vec{e}_i) = 0$, for $i = 1, 2, \ldots, 7$, any single variable can be removed. However, since $B_f(\vec{e}_i \vee \vec{e}_j) \neq 0$ for $(1 \leq i < j \leq 7)$, any pair of variables cannot be removed simultaneously.*

Next, consider the transformation:

$$\begin{aligned} x_1 &\Leftarrow x_1 \oplus x_3 \oplus x_5 \oplus x_7, \\ x_2 &\Leftarrow x_2 \oplus x_3 \oplus x_6 \oplus x_7, \\ x_4 &\Leftarrow x_4 \oplus x_5 \oplus x_6 \oplus x_7. \end{aligned}$$

The non-zero coefficients of the autocorrelation function is shown in Table 7. In this case, $B_f(\vec{\tau}) = 0$, where

$$\vec{\tau} = a_1\vec{e}_3 \vee a_2\vec{e}_5 \vee a_3\vec{e}_6 \vee a_4\vec{e}_7,$$

$a_i \in \{0,1\}$, and $\vec{\tau} \neq \vec{0}$.

Thus, four variables x_3, x_5, x_6, x_7 can be removed simultaneously, and the function can be represented with only x_1, x_2, and x_4.

$\vec{\tau}$							$B_f(\vec{\tau})$	$\lvert\vec{\tau}\rvert$
x_1	x_2	x_3	x_4	x_5	x_6	x_7		
0	0	0	0	0	0	0	7	0
0	0	0	0	0	1	1	2	2
0	0	0	0	1	0	1	2	2
0	0	0	0	1	1	0	2	2
0	0	0	1	0	0	1	2	2
0	0	0	1	0	1	0	2	2
0	0	0	1	1	0	0	2	2
0	0	1	0	0	0	1	2	2
0	0	1	0	0	1	0	2	2
0	0	1	0	1	0	0	2	2
0	0	1	1	0	0	0	2	2
0	1	0	0	0	0	1	2	2
0	1	0	0	0	1	0	2	2
0	1	0	0	1	0	0	2	2
0	1	0	1	0	0	0	2	2
0	1	1	0	0	0	0	2	2
1	0	0	0	0	0	1	2	2
1	0	0	0	0	1	0	2	2
1	0	0	0	1	0	0	2	2
1	0	0	1	0	0	0	2	2
1	0	1	0	0	0	0	2	2
1	1	0	0	0	0	0	2	2

TABLE 6
Autocorrelation Coefficients for the function in Table 1

7 PROPERTIES OF AUTOCORRELATION FUNCTIONS

Theorem 4. *Let M be the set of binary vectors corresponding to the minterms of f. Let D_f be the set of vectors $\vec{a} \oplus \vec{b}$, where $\vec{a}, \vec{b} \in M$, and $\vec{a} \neq \vec{b}$. Then,*

$$B_f(\vec{\tau}) = \begin{cases} |M| & \text{if } \vec{\tau} = \vec{0} \\ \geq 2 & \text{if } \vec{\tau} \in D_f \\ 0 & \text{Otherwise} \end{cases}$$

Proof. The minterm expansion of f is

$$f = \bigvee_{\vec{a} \in M} \vec{x}^{\vec{a}},$$

$\vec{\tau}$							$B_f(\vec{\tau})$	$\lvert\vec{\tau}\rvert$
x_1	x_2	x_3	x_4	x_5	x_6	x_7		
0	0	0	0	0	0	0	7	0
1	0	0	0	0	1	1	2	3
0	1	0	0	1	0	1	2	3
1	1	0	0	1	1	0	2	4
1	1	0	0	0	0	1	2	3
0	1	0	0	0	1	0	2	2
1	0	0	0	1	0	0	2	2
0	0	1	1	0	0	1	2	3
1	0	1	1	0	1	0	2	4
0	1	1	1	1	0	0	2	4
1	1	1	1	0	0	0	2	4
1	0	0	1	0	0	1	2	3
0	0	0	1	0	1	0	2	2
1	1	0	1	1	0	0	2	4
0	1	0	1	0	0	0	2	2
1	0	1	0	0	0	0	2	2
0	1	0	1	0	0	1	2	3
1	1	0	1	0	1	0	2	4
0	0	0	1	1	0	0	2	2
1	0	0	1	0	0	0	2	2
0	1	1	0	0	0	0	2	2
1	1	0	0	0	0	0	2	2

TABLE 7
Autocorrelation Coefficients for transformed function

where $\vec{x}^{\vec{a}} = 1$ iff $\vec{x} = \vec{a}$. In this case, we have

$$\begin{aligned} B_f(\vec{\tau}) &= \sum_{\vec{v}\in\{0,1\}^n} [\bigvee_{\vec{a}\in M} \vec{v}^{\vec{a}}] \cdot [\bigvee_{\vec{b}\in M} (\vec{v} \oplus \vec{\tau})^{\vec{b}}] \\ &= \bigvee_{\vec{a}\in M} \bigvee_{\vec{b}\in M} B_{(\vec{a},\vec{b})}(\vec{\tau}), \end{aligned}$$

where $B_{(\vec{a},\vec{b})}(\vec{\tau}) = 1$ iff $\vec{\tau} = \vec{a} \oplus \vec{b}$. Thus, we have $B_f(\vec{0}) = |M|$, and $B_f(\vec{a} \oplus \vec{b}) \geq 2$, where $\vec{a} \neq \vec{b}$ and $\vec{a}, \vec{b} \in M$. It is clear that $B_f(\vec{\tau}) = 0$ for other vectors.

Example 11. *Consider the function shown in Figure 6. The set of vectors corresponding to the minterms for f is $M = \{(1,0,0,0), (0,1,0,0), (0,1,1,0), (1,1,0,1)\}$. Thus, $D_f = \{(1,1,0,0), (1,1,1,0), (0,1,0,1), (0,0,1,0), (1,0,0,1), (1,0,1,1)\}$. Also, $B_f(\vec{0}) = |M| = 4$. Note that these correspond to Table 5.*

Thus, when $|M| = k$, the number of non-zero coefficients of the autocorrelation function is at most $\binom{k}{2} + 1 = \frac{k(k-1)}{2} + 1$.

It is very interesting to note that D_f corresponds to the *difference matrix* used in [21].

8 A METHOD TO FIND GOOD LINEAR TRANSFORMATIONS

8.1 Minimal Sets of Variables

From the results of the previous sections, we have the following:

Algorithm 1. *(Expression Showing Minimal Sets of Variables for an Incompletely Specified Index Generation Function)*

1. *Let f be the characteristic logic function of the index generation function F.*
2. *Let D_f be the set of non-zero vectors defined in Theorem 4.*
3. *For each vector in D_f, make a product by replacing 1 with $\bar{x}_i$, where i denotes the index of the variable, and by replacing 0 with a missing variable.*
4. *Derive the sum-of-products expression (SOP) $\bar{R}$ consisting above products.*
5. *Obtain the SOP for R.*
6. *Each product shows a minimal set of variables to represent F.*

Note that R is the **covering function** defined in [16].

Example 12.

1. *Consider the function shown in Figure 6.*
2. *Table 5 shows D_f.*
3. *The set of products to representing vectors in D_f is $\{\bar{x}_2\bar{x}_4, \bar{x}_3, \bar{x}_1\bar{x}_4, \bar{x}_1\bar{x}_2\bar{x}_3, \bar{x}_1\bar{x}_2, \bar{x}_1\bar{x}_3\bar{x}_4\}$.*
4. *The SOP for $\bar{R}$ is*

$$\bar{R} = \bar{x}_2\bar{x}_4 \vee \bar{x}_3 \vee \bar{x}_1\bar{x}_4 \vee \bar{x}_1\bar{x}_2\bar{x}_3 \vee \bar{x}_1\bar{x}_2 \vee \bar{x}_1\bar{x}_3\bar{x}_4.$$

5. *The SOP for R is*

$$R = x_2x_3x_4 \vee x_1x_3x_4 \vee x_1x_2x_3.$$

6. *Minimum sets of variables to represent F are $\{x_2, x_3, x_4\}$, $\{x_1, x_3, x_4\}$, and $\{x_1, x_2, x_3\}$.*

Example 13.

1. *Consider the non-zero coefficients of the autocorrelation function shown in Table 7.*
2. *The SOP for $\bar{R}$ is*

$$\begin{array}{rcl}\bar{R} & = & \bar{x}_1\bar{x}_6\bar{x}_7 \vee \bar{x}_2\bar{x}_5\bar{x}_7 \vee \bar{x}_1\bar{x}_2\bar{x}_5\bar{x}_6 \\ & \vee & \bar{x}_1\bar{x}_2\bar{x}_7 \vee \bar{x}_2\bar{x}_6 \vee \bar{x}_1\bar{x}_5 \vee \bar{x}_3\bar{x}_4\bar{x}_7 \\ & \vee & \bar{x}_1\bar{x}_3\bar{x}_4\bar{x}_6 \vee \bar{x}_2\bar{x}_3\bar{x}_4\bar{x}_5 \vee \bar{x}_1\bar{x}_2\bar{x}_3\bar{x}_4 \\ & \vee & \bar{x}_1\bar{x}_4\bar{x}_7 \vee \bar{x}_4\bar{x}_6 \vee \bar{x}_1\bar{x}_2\bar{x}_4\bar{x}_5 \vee \bar{x}_2\bar{x}_4 \\ & \vee & \bar{x}_1\bar{x}_3 \vee \bar{x}_2\bar{x}_4\bar{x}_7 \vee \bar{x}_1\bar{x}_2\bar{x}_4\bar{x}_6 \vee \bar{x}_4\bar{x}_5 \\ & \vee & \bar{x}_1\bar{x}_4 \vee \bar{x}_2\bar{x}_3 \vee \bar{x}_1\bar{x}_2.\end{array}$$

3. *The SOP for R is*

$$\begin{array}{rcl}R & = & x_2x_3x_4x_5x_6 \vee x_1x_3x_4x_5x_6 \vee x_1x_2x_3x_5x_6 \\ & \vee & x_2x_3x_4x_5x_7 \vee x_1x_3x_4x_6x_7 \vee x_1x_2x_5x_6x_7 \\ & \vee & x_1x_2x_4.\end{array}$$

4. *The minimum set of variables to represent F is* $\{x_1, x_2, x_4\}$.

8.2 Linear Transformations using Autocorrelation Function

Let $|M| = k$. Then, the number of non-zero coefficients in the autocorrelation function is at most $\frac{k(k-1)}{2} + 1$. When this value is not so large, we can search for a good linear transformation by the autocorrelation function instead of the original function.

For example, Table 4 can be obtained from Table 3 by the linear transformation:

$$\begin{array}{rcl}x_1 & \Leftarrow & x_1 \oplus x_4, \\ x_2 & \Leftarrow & x_2 \oplus x_4.\end{array}$$

Also, Table 5 can be obtained from Table 4 by the linear transformation:

$$x_2 \Leftarrow x_2 \oplus x_3.$$

8.3 Heuristic to Find Good Linear Transformations using Autocorrelation Functions

Since D_f contains all the necessary information to reduce the number of variables, we can work on D_f. If $B_f(\vec{e}_s) \neq 0$, then we cannot remove the variable

x_s. So, with the autocorrelation function, we try to find the linear transformation $x_i \Leftarrow x_i \oplus x_j$ that modifies such vectors to increase their weights.

To remove x_i and x_j simultaneously, we have to remove the vectors $\vec{e}_i$, $\vec{e}_j$, and $\vec{e}_i \vee \vec{e}_j$ from D_f by increasing the weight of such vectors, if any of such vectors exist in D_f. In the ISMVL-2013 paper [19], we used the following:

Heuristic 1. *Increase ξ, the total number of 1's in the vectors of D_f.*

A more effective heuristic is the following:

Heuristic 2. *Increase*

$$\prod_{i=1}^{k}\sum_{j=1}^{n} d_{ij},$$

where $d_{ij} = 1$ iff i th vector in D_f has 1 in the j the position.

This is explained as follows: $\sum_{j=1}^{n} d_{ij}$ denotes the number of literals in a product in $\bar{R}$. The smaller the number, the smaller the solution space in the covering relation. In the extreme case, if it is 1, then it shows that one variable is essential.

Example 14. *Consider the function in Table 1. To represent this function, we need 6 variables. Let ξ be the total number of 1's in the vectors in D_f. Since there are $k = \binom{7}{2} = 21$ vectors, and each vector has two 1's, we have $\xi = 21 \times 2 = 42$. By using Heuristic 2, we can reduce the number of variables to represent F. Consider the following transformations:*

$$\begin{array}{lcl}
x_1 & \Leftarrow & x_1 \oplus x_2\ (\xi = 46). \\
x_1 & \Leftarrow & x_1 \oplus x_3\ (\xi = 48). \\
x_2 & \Leftarrow & x_2 \oplus x_1\ (\xi = 52). \\
x_2 & \Leftarrow & x_2 \oplus x_4\ (\xi = 54). \\
x_2 & \Leftarrow & x_2 \oplus x_5\ (\xi = 54). \\
x_3 & \Leftarrow & x_3 \oplus x_1\ (\xi = 58). \\
x_3 & \Leftarrow & x_3 \oplus x_2\ (\xi = 60). \\
x_3 & \Leftarrow & x_3 \oplus x_4\ (\xi = 60). \\
x_4 & \Leftarrow & x_4 \oplus x_1\ (\xi = 66). \\
x_4 & \Leftarrow & x_4 \oplus x_3\ (\xi = 66). \\
x_5 & \Leftarrow & x_5 \oplus x_1\ (\xi = 72).
\end{array}$$

$$
\begin{aligned}
x_5 &\Leftarrow x_5 \oplus x_2 \ (\xi = 70). \\
x_5 &\Leftarrow x_5 \oplus x_3 \ (\xi = 72). \\
x_5 &\Leftarrow x_5 \oplus x_6 \ (\xi = 72). \\
x_6 &\Leftarrow x_6 \oplus x_1 \ (\xi = 78). \\
x_6 &\Leftarrow x_6 \oplus x_2 \ (\xi = 78). \\
x_6 &\Leftarrow x_6 \oplus x_3 \ (\xi = 78). \\
x_7 &\Leftarrow x_7 \oplus x_1 \ (\xi = 84). \\
x_7 &\Leftarrow x_7 \oplus x_2 \ (\xi = 84). \\
x_7 &\Leftarrow x_7 \oplus x_3 \ (\xi = 84). \\
x_2 &\Leftarrow x_2 \oplus x_1 \ (\xi = 84). \\
x_5 &\Leftarrow x_5 \oplus x_3 \ (\xi = 84).
\end{aligned}
$$

Note that the values of ξ after the transformations are, in most cases, non-decreasing. The results of the above transformations are:

$$
\begin{aligned}
y_1 &= x_1 \oplus x_2 \oplus x_3. \\
y_2 &= x_2 \oplus x_4 \oplus x_5. \\
y_3 &= x_2 \oplus x_3 \oplus x_5. \\
y_4 &= x_1 \oplus x_4 \oplus x_5. \\
y_5 &= x_2 \oplus x_4 \oplus x_6. \\
y_6 &= x_3 \oplus x_4 \oplus x_6. \\
y_7 &= x_3 \oplus x_4 \oplus x_7.
\end{aligned}
$$

With these transformations, F can be represented with only three variables:$\{y_3, y_4, y_7\}$.

9 EXPERIMENTAL RESULTS

We developed a program to perform the algorithm described in the previous section. As for the benchmark function, we used *m-out-of-n* code to index converters [16, 18]. In the experiment, we used INTEL Core i5-3320M CPU @2.6 GHz, and Windows 7, 64-bit operating system.

9.1 When $m = 1$

In the first experiment, m is fixed to 1, while n is changed from 6 to 24. Table 1 shows the example of $n = 7$. It is an index generation functions with weight n. The i-th variable has 1 and other variables have 0 in the input if

	# of variables		ξ		$CPU1$
n	MIN	$AUTO$	$Orig$	Aft	(ms)
6	3	3	30	54	0.3
7	3	3	42	84	0.3
8	3	3	56	127	0.3
9	4	4	72	180	0.3
10	4	4	90	250	0.3
11	4	4	110	330	0.3
12	4	4	132	431	0.7
13	4	4	156	546	1.1
14	4	4	182	685	1.5
15	4	4	210	840	1.5
16	4	4	240	1022	1.5
17	5	5	272	1224	2.3
18	5	5	306	1457	2.7
19	5	5	306	1710	3.5
20	5	5	380	1997	3.5
21	5	5	420	2310	7.4
22	5	5	462	2658	6.6
23	5	5	506	3036	9.7
24	5	5	552	3453	11.3

TABLE 8
Reduction of variables for 1-out-of-n code to index converters

and only if the value of the function is i. The minimum number of variables to represent this function is $\lceil \log_2 n \rceil$. Table 8 shows the results. The first column shows n, the number of inputs; the second column (*MIN*) shows the minimum solution; the third column (*AUTO*) shows the number of variables obtained by the presented method; the fourth column (*Orig*) shows the number of 1's in the original difference matrix; the fifth column (*Aft*) shows the number of 1's in the difference matrix after the transformation; and the last column shows the CPU time for the linear transformations. Up to $n = 24$, the program obtained **exact minimum** solutions. The presented program is faster and requires much less memory than [17–19] for this class of functions.

9.2 When $n = 20$

In the second experiment, n is set to 20, while m is changed from 1 to 4. Table 9 shows the results for m-out-of-20 code to index converters. The column headed by [18] shows the results in ASPDAC-2012, where all the transformed variables with the compound degrees with up to six were generated. This is a greedy method and requires memory proportional to

$$k \times \sum_{i=1}^{t=6} \binom{n}{i}.$$

		# of Variables and CPU time [ms]				
m	k	[18]	CPU	AUTO	CPU1	CPU2
1	20	6	2611	**5**	5	31
2	190	9	5919	9	408	753
3	1140	**11**	76939	13	15900	457
4	4845	15	1110684	15	334457	1026

TABLE 9
Number of Variables to Represent m-out-of-20 Code to Index Converter.

	Exhaustive Method ISMVL 2011	Heuristic Method ASPDAC 2012	Heuristic Method AUTO
Memoroy size	$O(k2^n)$	$O(kn^t)$	$O(nk^2)$
CPU time	Too Large	Medium	Small
Quality of Solutions	Exact Minimum	Good	Good

TABLE 10
Comparison with Existing Methods

In Table 9, CPU1 denotes the time for linear transformation, while CPU2 denotes the time for near minimal covering. The numbers of variables shown in bold face are equal to $\lceil \log_2 k \rceil$: they are optimum solutions. The presented program is faster and requires much less memory than one in [18]. Also, for $m = 4$, the method [18] obtained the solution of 15 variables when $t = 3$. However, for $t = 6$, it obtained solution of 16 variables. Table 10 compares the property of the present algorithm with the existing methods.

10 SUMMARY

Major contributions of this paper are:

- Defined the characteristic logic function f of an incompletely specified index generation function F.
- Showed that the autocorrelation function of f are useful to find the number of variables to represent F.
- Presented an algorithm to derive minimal sets of variables to represent F.
- Presented an algorithm to derive the set of vectors that produce non-zero values in the autocorrelation function of f. The number of non-zero

coefficients is at most $\frac{k(k-1)}{2}+1$, where k is the number of specified elements in the index generation function F.
- Showed a heuristic algorithm to find a good linear transformation using the autocorrelation function.
- Developed a program and presented some experimental results.
- Showed that the set of vectors that produce non-zero values in the autocorrelation function of f, is equivalent to the **covering function** defined in [16], and also to the *difference matrix* D_f used in [21].

ACKNOWLEDGMENTS

This work is partially supported by the Japan Society for the Promotion of Science (JSPS), Grant in Aid for Scientific Research. Prof. Jon T. Butler's comments were useful to improve the presentation.

REFERENCES

[1] C. Halatsis and N. Gaitanis, "Irredundant normal forms and minimal dependence sets of a Boolean functions," *IEEE Trans. on Computers*, vol. C-27, no. 11, Nov. 1978, pp. 1064–1068.

[2] Y. Kambayashi, "Logic design of programmable logic arrays," *IEEE Trans. on Computers*, vol. C-28, no. 9, Sept. l979, pp. 609–617.

[3] M. G. Karpovsky, *Finite Orthogonal Series in the Design of Digital Devices*, Wiley, 1976.

[4] M. G. Karpovsky, R. S. Stankovic, and J. T. Astola, "Reduction of sizes of decision diagrams by autocorrelation functions,"*IEEE Transactions on Computers*, vol. 52, no. 5, pp. 592–606, May, 2003.

[5] O. Keren and I. Levin, "Linearization of multi-output logic functions by ordering of the autocorrelation values,"*FACTA UNIVERSITATIS (NIS)*, vol. 20, no. 3, December 2007, pp. 479–498.

[6] O. Keren, I. Levin and R. S. Stankovic, "Determining the number of paths in decision diagrams by using autocorrelation coefficients," *IEEE Transactions on Computer-Aid Design of Integrated Circuits and Systems*, vol. 30, no. 1, Jan. 2011, pp. 31–44.

[7] R. J. Lechner, "Harmonic analysis of switching functions," in A. Mukhopadhyay (ed.), *Recent Developments in Switching Theory*, Academic Press, New York, 1971.

[8] C. Meinel, F. Somenzi, and T. Theobald, "Linear sifting of decision diagrams and its application in synthesis," *IEEE Trans. CAD*, vol. 19, no. 5, pp. 521–533, 2000.

[9] E. I. Nechiporuk, "On the synthesis of networks using linear transformations of variables," *Dokl. AN SSSR*, vol. 123, no. 4, Dec. 1958, pp. 610–612.

[10] J. Rice and J.C.Muzio, "Use of the autocorrelation function in the classification of switching functions," *Euromicro Symposium on Digital System Design*, (DSD 2002), Sept.4-6, 2002, pp. 244–251.

[11] T. Sasao, *Switching Theory for Logic Synthesis*, Kluwer Academic Publishers, 1999.

[12] T. Sasao, “On the number of dependent variables for incompletely specified multiple-valued functions,” International Symposium on Multiple-Valued Logic (ISMVL-2000), Portland, Oregon, U.S.A., May 23-25, 2000, pp. 91–97.

[13] T. Sasao, “Design methods for multiple-valued input address generators,”(invited paper) *International Symposium on Multiple-Valued Logic* (ISMVL-2006), Singapore, May 2006.

[14] T. Sasao, “On the number of variables to represent sparse logic functions,” *ICCAD-2008*, San Jose, California, USA, Nov. 10-13, 2008, pp. 45–51.

[15] T. Sasao, T. Nakamura, and M. Matsuura, “Representation of incompletely specified index generation functions using minimal number of compound variables,” *12th EUROMICRO Conference on Digital System Design, Architectures, Methods and Tools* (DSD 2009), Patras, Greece, Aug. 27-29, 2009, pp. 765–772.

[16] T. Sasao, *Memory-Based Logic Synthesis*, Springer, 2011.

[17] T. Sasao, “Index generation functions: Recent developments,” (invited paper) *International Symposium on Multiple-Valued Logic* (ISMVL-2011), Tuusula, Finland, May 23–25, 2011.

[18] T. Sasao, “Linear decomposition of index generation functions,” *17th Asia and South Pacific Design Automation Conference* (ASPDAC-2012), Jan. 30- Feb. 2, 2012, Sydney, Australia, pp. 781–788.

[19] T. Sasao, “An application of autocorrelation functions to find linear decompositions for incompletely specified index generation functions,” *International Symposium on Multiple-Valued Logic* (ISMVL-2013), May 21–24, Toyama, Japan, pp. 96–102.

[20] T. Sasao, “Index generation functions: Tutorial,”*Journal of Multiple-Valued Logic and Soft Computing*, Vol. 23, No. 3–4, pp. 235–263, 2014.

[21] D. A. Simovici, M. Zimand, and D. Pletea, “Several remarks on index generation functions,”*International Symposium on Multiple-Valued Logic* (ISMVL-2012), May 2010, pp. 179–184.

[22] D. Varma and E. Trachtenberg, “Design automation tools for efficient implementation of logic functions by decomposition,” *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, vol. 8, no. 8, pp. 901–916, Aug. 1989.