A Heuristic Method to Find Linear Decompositions for Incompletely Specified Index Generation Functions

Tsutomu Sasao, Yuta Urano, and Yukihiro Iguchi
Dept. of Computer Science, Meiji University
Kawasaki, Kanagawa 214-8571, Japan

Abstract— This paper shows a method to find a linear transformation that reduces the number of variables to represent a given incompletely specified index generation function. It first generates the difference matrix, and then finds the minimal set of variables using a covering table. Linear transformations are used to modify the covering table to produce a smaller solution.

I. Introduction

Index generation functions [12] are useful in network applications [5] and pattern matching including computer virus scanning engines [4].

In many cases, functions must be updated frequently. Thus, a memory-based architecture is desirable for the implementation. To reduce the size of the memory to implement index generation functions, a linear decomposition shown in Fig. 1.1 is quite effective [14]. When a given function is defined for only k input combinations and $k << 2^n$, the number of variables for the general part can be often reduced. To find a good decomposition, we use a linear transformation to reduce the number of variables p for the general part. In many cases, by this, the size of the LUT for the general part is drastically reduced.

In this paper, we show a new method to find a linear transformation that reduces the number of variables to represent a given incompletely specified index generation function. The rest of the paper is organized as follows: Section 2 defines index generation functions; Section 3 shows a method to reduce the number of variables; Section 4 introduces a difference matrix to reduce the number of variables; Section 5 shows a method to reduce variables using a linear transformations; Section 6 shows a heuristic method to find a good linear transformation; Section 7 shows experimental results; and Section 8 summarizes the paper.

II. Index Generation Function

Definition 2.1 *Consider a set of k different vectors of n bits. These vectors are* **registered vectors**. *For each registered vector, assign a unique integer from 1 to k. A* **registered vector table** *shows an* **index** *for each registered vector. An* **incompletely specified index generation function** *produces a corresponding index when the input vector matches a registered vector. Otherwise, the value of the function is undefined (d, don't care). The incompletely specified index generation function represents a mapping $M \to \{1, 2, \ldots, k\}$, where $M \subset B^n$ denotes the set of registered vectors. k is the* **weight** *of the function.*

TABLE 2.1
Registered Vector Table

Vector					Index
x_1	x_2	x_3	x_4	x_5	
1	0	0	0	0	1
0	1	0	0	0	2
0	0	1	0	0	3
0	0	0	1	0	4
0	0	0	0	1	5

Example 2.1 *Consider the registered vectors shown in Table 2.1. These vectors show an index generation function with weight $k = 5$.* ■

III. 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

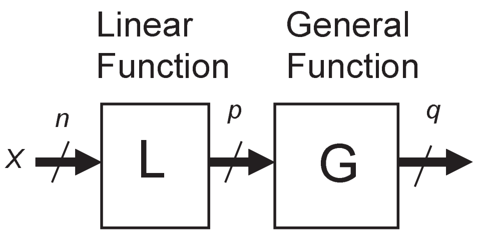

Fig. 1.1. Linear Decomposition.

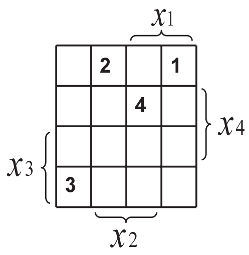

Fig. 3.1. Index Generation Function of 4 variables.

of variables to represent f. This property is useful to realize a function using a smaller look-up table (LUT).

Theorem 3.1 *Suppose that an incompletely specified function f is represented by a decomposition chart [7]. If each column has at most one care element, then the function can be represented by using only the column variables.*

(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 the column variables. □

Example 3.1 *Consider the decomposition chart shown in Fig. 3.1, 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 Fig. 3.1, each column has at most one care element. Thus, this function can be represented by only the column variables x_1 and x_2:*

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

■

Algorithms to minimize the number of variables in incompletely specified functions have been developed [2, 3, 8, 10]. As for the lower bound on the number of variables, we have the following:

Theorem 3.2 *[14] 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.

IV. Minimization of the number of Variables using Difference Matrix

In this section, we introduce difference matrix to minimize the number of variables to represent a given incompletely specified index generation function.

Definition 4.1 *[16, 15] 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 $a \neq b$. D_f is called a* **difference matrix** *of M. Note that D_f consists of $\binom{k}{2} = \frac{k(k-1)}{2}$ vectors, where $k = |M|$.*

TABLE 4.1
Registered Vectors before Linear Transformation

x_1	x_2	x_3	x_4	Index
1	0	0	0	1
0	1	0	0	2
0	1	1	0	3
1	1	0	1	4

TABLE 4.2
Difference Matrix before Linear Transformation

x_1	x_2	x_3	x_4	Tag
1	1	0	0	$(1,2)$
1	1	1	0	$(1,3)$
0	1	0	1	$(1,4)$
0	0	1	0	$(2,3)$
1	0	0	1	$(2,4)$
1	0	1	1	$(3,4)$

Example 4.1 *Consider the function shown in Fig. 4.1. Table 4.1 shows M, the set of vectors corresponding to the minterms for f. It is also called as registered vector table. Table 4.2 shows the corresponding difference matrix D_f. The last column of Table 4.2 shows tags specifying the pair of vectors in M. For example, the first vector in D_f has the tag $(1,2)$, which shows that the first and the second elements in M were used to generate the vector:*

$$(1,0,0,0) \oplus (0,1,0,0) = (1,1,0,0).$$

It shows that to distinguish the first and the second vectors in M, either x_1 or x_2 is necessary. ■

Note that the difference matrix shows the condition to distinguish all the pairs of vectors in M [8], and is essentially the same as the **covering table** [7]. Thus, we can find the minimal set of variables to represent an incompletely specified index generation function as follows:

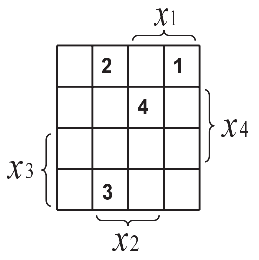

Fig. 4.1. Index Generation Function of 4 variables.

Algorithm 4.1 *(Minimal Sets of Variables to Represent an Incompletely Specified Index Generation Function)*

1. *Let M be the set of vectors showing an incompletely specified index generation function.*
2. *Generate D_f, the difference matrix, from M.*
3. *Assume that in D_f, each column corresponds to a variable x_j, and each row corresponds to a vector in D_f.*
4. *The element (i,j) of the covering table is 1 iff the j-th element of the i-th vector in D_f is 1.*
5. *Derive the minimal set of variables that covers all the rows of D_f.*

Example 4.2 *Consider the index generation function shown in Fig. 4.1. Table 4.1 shows the registered vector table. Note that the number of the columns is $n = 4$, while the number of the rows is $\binom{k}{2} = \frac{k(k-1)}{2} = \frac{4\times 3}{2} = 6$. The first row with the tag (1,2) corresponds to the first element in D_f, which show that to distinguish the 1st and 2nd vectors in M, either x_1 or x_2 is necessary. Also, note that the row with the tag (2,3) has only single one. Such a row is a* **distinguished row**, *and only the column of x_3 covers this row. Such a variable is* **essential** *and, is necessary in all solutions. Minimal sets of variables that cover all the rows are $\{x_1, x_2, x_3\}, \{x_1, x_3, x_4\}$, and $\{x_2, x_3, x_4\}$.* ∎

To find a minimal set of variables, we can use a standard method [7]. Although the method is straightforward, it takes much computation time when n and k are large.

V. Reduction of Variables By linear Transformations

In the previous section, we showed a method to reduce the number of variables for incompletely specified functions. Unfortunately, the effect of such method is limited. In this section, we show that more variables can be reduced by using linear transformations.

Example 5.1 *For the function in Table 2.1, the numbers of 1's in the registered vectors are all one. Note that, the number of variables to represent the function can be reduced to four: Any one variable can be removed. For example, if we remove x_5, then we have:*

$$F = 1 \cdot x_1\bar{x}_2\bar{x}_3\bar{x}_4 \vee\ 2 \cdot \bar{x}_1 x_2\bar{x}_3\bar{x}_4 \vee\ 3 \cdot \bar{x}_1\bar{x}_2 x_3\bar{x}_4 \vee\ 4 \cdot \bar{x}_1\bar{x}_2\bar{x}_3 x_4 \vee\ 5 \cdot \bar{x}_1\bar{x}_2\bar{x}_3\bar{x}_4.$$

However, we cannot remove two or more variables simultaneously. Thus, at least four variables are necessary to represent this function. ∎

TABLE 5.1
Registered Vectors after Linear Transformation

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

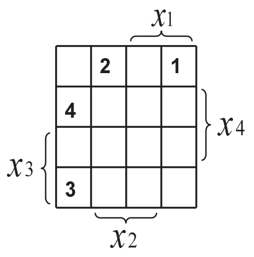

Fig. 5.1. Index Generation Function of 4 variables.

Definition 5.1 *A* **linear transformation** *is defined as*

$$\begin{aligned}
y_1 &= c_{11}x_1 \oplus c_{12}x_2 \oplus c_{13}x_3 \oplus \ldots \oplus c_{1n}x_n,\\
y_2 &= c_{21}x_1 \oplus c_{22}x_2 \oplus c_{23}x_3 \oplus \ldots \oplus c_{2n}x_n,\\
y_3 &= c_{31}x_1 \oplus c_{32}x_2 \oplus c_{33}x_3 \oplus \ldots \oplus c_{3n}x_n,\\
&\vdots\\
y_p &= c_{p1}x_1 \oplus c_{p2}x_2 \oplus c_{p3}x_3 \oplus \ldots \oplus c_{pn}x_n,
\end{aligned}$$

where $c_{ij} \in \{0,1\}$. $t_i = \sum_{j=1}^{n} c_{ij}$ is the **compound degree** *of y_i.*

Definition 5.2 *Given an incompletely specified index generation function, an* **optimum linear transformation** *is one that minimizes the number of variables p in Fig, 1.1.*

By Theorem 3.2, if the linear transformation reduces the number of variables to $q = \lceil \log_2 k \rceil$ variables, then it is optimum.

Example 5.2 *For the function in Table 2.1, consider the linear transformation:*

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

The transformed registered vectors are shown in Table 5.1. In this case, all the vectors are distinct, and three variables (y_1, y_2, y_3) distinguish five vectors. Note that this is an optimum transformation. ∎

TABLE 6.1
Registered Vectors before After Transformation

x_1	$\mathbf{y_2}$	x_3	x_4	Index
1	0	0	0	1
0	1	0	0	2
0	**0**	1	0	3
1	1	0	1	4

TABLE 6.2
Difference Matrix after Transformation

x_1	$\mathbf{y_2}$	x_3	x_4	Tag
1	1	0	0	(1, 2)
1	**0**	1	0	(1, 3)
0	1	0	1	(1, 4)
0	**1**	1	0	(2, 3)
1	0	0	1	(2, 4)
1	**1**	1	1	(3, 4)

VI. A Heuristic Method to Find Linear Transformations

A. Strategies to Find a Good Linear Transformation using Difference Matrix

Since the number of linear transformations to consider is very large [1], in this section, we present a heuristic method to find a linear transformation that reduces the number of variables.

Assume that x_i is transformed to $y_i \Leftarrow x_i \oplus x_j$ in M. Consider the effect of this linear transformation in D_f. Let $\vec{a}$ and $\vec{b}$ be different row vectors in M. Note that only the i-th part of the vectors is modified. Modified vectors in M are written as:

$$\begin{aligned}\vec{a}' &= (a_1, a_2, \ldots, a_i \oplus a_j, a_{i+1}, \ldots, a_n).\\ \vec{b}' &= (b_1, b_2, \ldots, b_i \oplus b_j, b_{i+1}, \ldots, b_n).\end{aligned}$$

Thus, we have
$\vec{a}' \oplus \vec{b}' = (a_1 \oplus b_1, a_2 \oplus b_2, \ldots, (a_i \oplus b_i) \oplus (a_j \oplus b_j), a_{i+1} \oplus b_{i+1}, \ldots, a_n \oplus b_n) = \vec{a} \oplus \vec{b} \oplus (0, 0, \ldots, 0, a_j \oplus b_j, 0, 0, \ldots, 0)$. Note that also in D_f, only the i-th part of the vectors is modified. This means that the linear transformation can be also done in D_f.

Example 6.1 *Consider the index generation function shown in Fig. 4.1, and apply the linear transformation:* $y_2 \Leftarrow x_2 \oplus x_3$. *Table 6.1 shows* M *after the transformation, while Table 6.2 shows* D_f *after the transformation. Note that in the transformed* D_f, *each row has at least two non-zero elements. Also, the total number of 1's in the transformed* D_f *is increased. In the transformed* D_f, *variable* x_3 *is not essential any more. Minimal sets of variables that cover all the rows are* $\{x_1, y_2\}, \{x_1, x_3, x_4\}$, *and* $\{y_2, x_3, x_4\}$. *Note that the linear transformation reduced the number of variables to two.* ■

TABLE 6.3
Original Difference Matrix

x_1	x_2	x_3	x_4	x_5	Tag
1	1	0	0	0	(1, 2)
1	0	1	0	0	(1, 3)
1	0	0	1	0	(1, 4)
1	0	0	0	1	(1, 5)
0	1	1	0	0	(2, 3)
0	1	0	1	0	(2, 4)
0	1	0	0	1	(2, 5)
0	0	1	1	0	(3, 4)
0	0	1	0	1	(3, 5)
0	0	0	1	1	(4, 5)

TABLE 6.4
Difference Matrix After 1st Reduction

x_1	x_2	x_3	x_4	x_5	Tag
1	1	0	0	0	(1, 2)
0	0	1	1	0	(3, 4)
0	0	1	0	1	(3, 5)
0	0	0	1	1	(4, 5)

The previous example implies that D_f with more 1's tends to produce smaller solutions. Let the **merit of a variable** be the number of rows covered by the variable. Our strategy is to **find a linear transformed variable with the maximal merit**. Then, eliminate the rows of D_f covered by the variable. Repeat this process until all the rows of D_f are eliminated.

Example 6.2 *Consider the function in Table 2.1. The difference matrix is shown in Table 6.3. First, we obtain the linear transformed variable with the maximal merit. Since* $y_1 = x_1 \oplus x_2$ *is such a variable, we select this transformation. Then, we remove the rows of* D_f *that are covered by* y_1. *Since the rows for (1,3),(1,4),(1,5),(2,3),(2,4) and (2,5) are covered by* y_1, *we remove them from* D_f, *and have the reduced difference matrix shown in Table 6.4.*

Then, we find the second linear function that maximally covers the remaining rows shown in Table 6.4. In this case, $y_2 = x_1 \oplus x_3$ *covers maximal number of rows, we select this transformation. In this case, rows for (1,2), (3,4), and (3,5) are removed, and only the row (4,5) remains. Since, the row (4,5) can be covered by* $y_3 = x_4$, *we select this as the third transformed variable. In this way, we can cover all the rows of* D_f. *The resulting linear transformed variables are exactly the same as ones introduced in Example 5.2.* ■

B. An Algorithm to Find Good Linear Transformations

From the previous observation, we have the following:

Algorithm 6.1 *(A greedy algorithm to find a set of linear transformations)*

1. *Let* M *be the set of vectors showing an incompletely specified index generation function.*

2. *Generate D_f, the difference matrix, from M.*
3. *Find a linear transformed variable y_i that covers the maximal number of rows in D_f using Algorithm 6.2.*
4. *Eliminate the rows in D_f that are covered by the linear transformed variable y_i.*
5. *Repeat this process until all the rows of D_f are eliminated.*

In Step 3, we used the following:

Algorithm 6.2 *(A greedy algorithm to find a linear transformed variable)*

1. *Find a variable x_i that has the maximal merit. Let $y_i \Leftarrow x_i$.*
2. *Find another variable x_j such that $y_i \oplus x_j$ covers the maximum number of rows in D_f. If the number of covered rows is increased, then $y_i \Leftarrow y_i \oplus x_j$.*
3. *Repeat above operation while the number of covered rows is increased.*

Algorithm 6.1 can be considered as an improvement of [16]. Our method to find linear transformed variables $y_1, y_2, \ldots, y_p$ is more efficient and effective, since we used iterative improvement method recently introduced in [15].

VII. Experimental Results

We developed a program for Algorithm 6.1. As for the benchmark function, we used *m-out-of-n* code to index converters [14]. It is an index generation functions with weight $k = \binom{n}{m}$. Table 2.1 shows the example of $n = 5$ and $m = 1$. In this case, the i-th variable has 1 and other variables have 0 in the input if and only if the value of the function is i. The minimum number of variables to represent the *1-out-of-n* code to index converter is $\lceil \log_2 n \rceil$. For up to $n = 256$, our program obtained exact minimum solutions. The CPU time for $n = 256$ is 67.7 sec. These results are much better than the previous results [16, 15]. For example, in [16], linear transformed variables were generated randomly, so experimental results for only up to $n = 12$ were reported.

Table 7.1 shows the results for m-out-of-20 code to index converters. The column headed by [14] shows the results in ASPDAC-2012. In this case, all the transformed variables with the compound degrees with up to six were considered. Note that this method requires memory proportional to

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

where t denotes the compound degree.

The presented program is faster and requires much less memory than one in [14], although the qualities of solutions are lower.

TABLE 7.1
Number of Variables to Represent m-out-of-20 Code to Index Converter.

		# of Variables and CPU time [ms]			
m	k	[14]	CPU	HEUR	CPU
1	20	6	2611	**5**	6
2	190	9	5919	10	273
3	1140	**11**	76939	13	9835
4	4845	16	1110684	16	182448

TABLE 7.2
Comparison with Existing Methods

	Exhaustive Method ISMVL 2011	Heuristic Method ASPDAC 2012	Heuristic Method SASIMI 2013
Memoroy size	$O(k2^n)$	$O(kn^t)$	$O(nk^2)$
CPU time	Too Large	Medium	Small
Quality of Solutions	Exact Minimum	Good	Good

In the experiment, we used a PC using INTEL Core i5-2450M CPU @2.5 GHz; Windows 7 64-bit operating system; and 8.00GB RAM. The figures shown in bold face denote optimum solutions.

Table 7.2 compares the present algorithm with existing ones, where t denotes the compound degree used for linear transformations. In our applications [4, 5], values of n are around 20-256, while values of k are up to 10^6. Thus, the proposed method is still too time consuming for large problems. Thus, in large problems, we have to partition the vectors into smaller groups to implement each group separately.

VIII. Summary

Major contributions of this paper are:

- Showed an algorithm to derive minimal sets of variables to represent f using a difference matrix.
- Showed a heuristic algorithm to find a good linear transformed variable to cover a difference matrix.
- Developed an efficient computer program which is much faster and requires smaller memory than previous methods.

To find a good linear transformation corresponds to modify the covering table so that the solution is reduced. We perform this by selecting a transformed variable that covers maximal number of uncovered rows, step by step.

Acknowledgments

This work is partially supported by the Japan Society for the Promotion of Science (JSPS), Grant in Aid for Scientific Research, and by the Adaptable and Seamless Technology Transfer Program through target-driven R&D, JST.

References

[1] E. Artin, *Geometric Algebra*, Interscience Publishers, New York, 1957.

[2] 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.

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

[4] H. Nakahara, T. Sasao, and M. Matsuura, "A low-cost and high-performance virus scanning engine using a binary CAM emulator and an MPU," *8th International Symposium on Applied Reconfigurable Computing*, (ARC 2012), March 19-23, 2012, Hong-Kong. Also, *Lecture Notes in Compute Science*, Vol.7199, pp. 202-214.

[5] H. Nakahara, T. Sasao and M. Matsuura, "An architecture for IPv6 lookup using parallel index generation units," *The 9th International Symposium on Applied Reconfigurable Computing* (ARC2013), March 25-27, 2013. Los Angeles. Also, in Lecture Notes in Computer Science, Vol. 7806, 2013, pp. 59-71.

[6] 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.

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

[8] 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.

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

[10] 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.

[11] 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.

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

[13] T. Sasao, "Index generation functions: Recent developments,"(invited paper) *International Symposium on Multiple-Valued Logic* (ISMVL-2011), Tuusula, Finland, May 23-25, 2011, pp.1-9.

[14] 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.

[15] T. Sasao, "An application of autocorrelation functions to find linear decompositions for incompletely specified index generation functions," *43rd International Symposium on Multiple-Valued Logic* (ISMVL-2013), May 22-24, 2013, Toyama, Japan, pp. 96-102.

[16] D. A. Simovici, M. Zimand, and D. Pletea, "Several remarks on index generation functions,"*International Symposium on Multiple-Valued Logic* (ISMVL-2012), Victoria, Canada, May 2012, pp. 179-184.