Logic Minimizers for Partially Defined Functions

Tsutomu Sasao, Kyu Matsuura, Kazuyuki Kai, Yukihiro Iguchi
Meiji University, Kanagawa 214-0034, Japan

***Abstract*—This demonstration shows a minimization system for partially defined functions. The minimizer reduces the number of the input variables to represent the function using linear transformations. Applications include implementations of random functions; code converter; IP address table; English word list; and URL list.**

I. Partially Defined Function

Given a discrete function f of n variables, a straightforward method to implement f is to use a look-up table (LUT). However, the size of the LUT to implement f is proportional to 2^n. When n is large, a realization of f by a single LUT is, in many cases, impractical. **Functional decomposition** is a technique to realize a function using smaller LUTs.

Definition 1.1: Consider a set of k different vectors of n bits. These vectors are **registered vectors**. For each registered vector, assign an index from 1 to k. A **registered vector table** shows the **index** for each registered vector. A **partially defined function** produces a corresponding index when the input vector matches a registered vector. Otherwise, the the value of the function is unspecified. In this case, k is the **weight** of the function.

A partially defined functions often appear in address tables in the internet. Such circuits must be updated frequently, thus **programmable** or **reconfigurable** realizations are desirable.

II. Reduction of Variables

In a partially defined discrete function f, *don't care* values can be chosen as either 0 or 1 to minimize the number of variables to represent f. This property is useful to realize the function using smaller LUTs. For most functions, the necessary number of variables p depends on only k, and is independent of n, the number of variables in the original functions. A **linear decomposition** [2] is useful to further reduce the number of variables. Any partially defined function with weight k can be represented with at most $n = 2\lceil \log_2(k+1) \rceil - 2$ variables as shown in Fig. 2.1. The linear part L implements the linear transformation of the input variables:

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

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

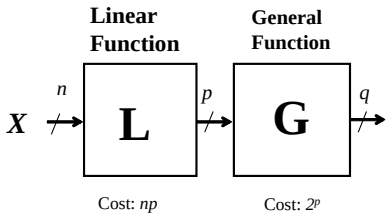

Fig. 2.1. Linear Decomposition.

III. Example

Table 3.1 shows how various functions can be compressed, where n denotes the number of original variables, k denotes the number of registered vectors, p denotes the number of variable after minimization, q denotes the number of outputs, and t denotes the maximal compound degree.

TABLE 3.1
Number of Variables to Represent Various Functions.

Function	n	k	p	q	t
Code Converter	32	4960	15	13	10
IP Address Table	32	7903	20	13	3
English word list	40	4705	18	13	5
Random	60	1000	15	10	3
URL List	245	1691	16	11	4

IV. Outline of Demonstration

In the demonstration, a PC and a poster are used to show:

- Introduction of partially defined functions.
- A method to reduce variables by linear decompositions.
- Implementation of code converters.
- Implementation of IP address tables.
- Implementation of English dictionaries.
- Implementation of random functions.
- Implementation of URL lists.

Acknowledgments

This research is partly supported by a Grant in Aid for Scientific Research of the JSPS.

References

[1] T. Sasao, *Memory-Based Logic Synthesis*, Springer, 2011.
[2] 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.
[3] T. Sasao, "Index generation functions: Minimization methods," *International Symposium on Multiple-Valued Logic* (ISMVL-2017) (invited), Novi Sad, Serbia, May 2017, pp. 197-206.