Paweł MORAWIECKI[1], **Mariusz RAWSKI**[2]
[1] WYŻSZA SZKOŁA HANDLOWA W KIELCACH, ZAKŁAD INFORMATYKI
[2] POLITECHNIKA WARSZAWSKA, ZAKŁAD PODSTAW TELEKOMUNIKACJI

Input Variable Partition Method in Functional Decomposition based on Shannon Expansion

Mgr inż. Paweł MORAWIECKI

Otrzymał stopień inżyniera w 2003 roku a rok później stopień magistra na Wydziale Elektroniki i Technik Informacyjnych Politechniki Warszawskiej. Obecnie jest asystentem w Zakładzie Informatyki w Wyższej Szkole Handlowej w Kielcach. Jego zainteresowania naukowe koncentrują się wokół syntezy logicznej układów cyfrowych, algorytmów i struktur danych, obliczeń kwantowych i logiki rewersyjnej.

e-mail: pawelm@wsh-kielce.edu.pl

Dr inż. Mariusz RAWSKI

Otrzymał stopień magistra inżyniera na Wydziale Elektroniki Politechniki Warszawskiej w 1995 roku. Stopień doktora otrzymał na tym samym wydziale w 2000 roku. Obecnie jest adiunktem na Wydziale Elektroniki i Technik Informacyjnych Politechniki Warszawskiej. Jego zainteresowania naukowe obejmują syntezę logiczną układów cyfrowych, narzędzia CAD dla syntezy i optymalizacji logicznej, projektowanie systemów cyfrowych z wykorzystaniem struktur programowalnych PLD.

e-mail: rawski@tele.pw.edu.pl

Abstract

Functional decomposition has important applications in many fields of modern engineering and science. The practical usefulness of decomposition-based methods for very complex systems is restricted by computational complexity and memory requirements of existing algorithms. Efficiency of currently used decomposition algorithms is dependent on the size of decomposed functions. One of the crucial parts of functional decomposition is the input variable partitioning. In this paper, the “divide-and-conquer” paradigm is used to propose a new input variable partitioning method. It has to be stressed that proposed method is not the input variable partition algorithm itself. It should be treated as a general scheme, method which can be combined with the algorithms generating input variable partitions (systematically, heuristically or by algorithms based on BDD).

Keywords: Functional decomposition, Shannon expansion, logic synthesis.

Metoda doboru zmiennych w dekompozycji funkcjonalnej bazująca na ekspansji Shannona

Streszczenie

Dekompozycja funkcjonalna ma zastosowania w wielu dziedzinach współczesnej nauki. W artykule zostaje zaproponowany algorytm, który pozwoli na skrócenie czasu obliczeń na etapie doboru zmiennych w dekompozycji funkcjonalnej. Opisana metoda bazuje na paradygmacie „dziel i rządź”, wykorzystuje ekspansję Shannona. Należy podkreślić, iż zaproponowana metoda nie jest algorytmem doboru zmiennych samym w sobie. Stanowi ogólny schemat, który może być wykorzystany wraz z innymi metodami doboru zmiennych (metoda systematyczna, metody heurystyczne, metody oparte na drzewach BDD).

Słowa kluczowe: Dekompozycja funkcjonalna, ekspansja Shannona, synteza logiczna.

1. Introduction

Functional decomposition consists of breaking down a complex system of discrete functions into a network of smaller and relatively independent co-operating sub-systems in such a way that the original system’s behaviour is preserved, while some constrains are satisfied and some objectives are optimised. The motivation for using functional decomposition in system analysis and design is to reduce the complexity of the problem by divide-and-conquer paradigm and to find an appropriate network of coherent sub-systems: a system is decomposed into a set of smaller subsystems, such that each of them is easier to analyse, understand or synthesise.

Although the multi-level functional decomposition gives very good results in logic synthesis of digital circuits and information systems, its practical usefulness for very complex systems is limited by lack of an efficient method for the construction of the high quality sub-systems. In the sub-system construction process the following three factors play an extremely important role: an appropriate input support selection for sub-systems, decision which (multi-valued) function will be computed by a certain subsystem and encoding of the subsystem’s function with binary output variables. Computational complexity and memory requirements of existing algorithms are strongly dependent on the size of a decomposed function. Several efficient heuristic methods have been proposed for decomposition of functions with many input variables [6, 9]. However, decomposition of complex digital systems described by large truth tables is still computationally expensive and requires a large amount of system memory.

A method was proposed in [10] that reduces the computation of functional decomposition of given Boolean function to decomposition of its cofactors. The advantage of this concept is the possibility to expand the original function with Shannon’s expansion recursively until cofactors have satisfactory size. That allows the designer to adjust the algorithm to available system resources.

This paper presents the input variable partitioning method that uses concept presented in [10]. The decomposed function is split into sets of its cofactors and the input variable partitioning for one of the cofactors is evaluated. As the satisfactory partitioning is found, it is verified with the rest of the cofactors. If it satisfies decomposition conditions for all the cofactors, the G function can be constructed. Application of “divide-and-conquer” paradigm, allows reducing the computation time, as well as system memory requirements of functional decomposition algorithms for combinational circuits described by large truth tables.

2. Basic information

Here, only some information that is necessary for an understanding of this paper is reviewed. More detailed information concerning blanket algebra and functional decomposition method can be found in the papers [2, 3, 5].

A. Functional decomposition

Functional decomposition relies on partitioning a switching function into a network of two smaller and independent co-operating sub-functions, in such a way that the original system’s behaviour is preserved.

The set X of function’s input variable is partitioned into two subsets: *free variables* U and *bound variables* V, such that $U \cup V = X$. Assume that the input variables $x_1,\ldots,x_n$ have been relabelled in such a way that:

$$U = \{x_1,...,x_r\}$$

and

$$V = \{x_{n-s+1},...,x_n\}.$$

Consequently, for an n-tuple x, the first r components are denoted by x^U, and the last s components, by x^V.

Let F be a Boolean function, with $n > 0$ inputs and $m > 0$ outputs, and let (U, V) be as above. Assume that F is specified by a set F of the function's cubes. Let G be a function with s inputs and p outputs, and let H be a function with $r + p$ inputs and m outputs. The pair (G, H) represents a serial decomposition of F with respect to (U, V), if for every minterm b relevant to F, $G(b^V)$ is defined, $G(b^V) \in \{0, 1\}^{p.}$, and $F(b) = H(b^U, G(b^V))$. G and H are called blocks of the decomposition.

Let β_V, β_U, and β_F be blankets induced on the function's F input cubes by the input sub-sets V and U, and outputs of F, respectively.

Theorem 1. Existence of the serial decomposition [2]

If there exists a blanket β_G on F such that $\beta_V \le \beta_G$, and $\beta_U \bullet \beta_G \le \beta_F$, then F has a serial decomposition with respect to (U, V).

In this approach the decomposition process consists of the following steps:

- an input variable partitioning (the most time consuming part of the process),
- calculation of partitions β_U, β_V and β_F,
- construction of partition β_G,
- creation of functions H and G.

The input variable partitioning problem is NP-hard since for optimal solution it is necessary to search through all possibilities and the search space is exponentially dependent on the size of the decomposed function. For large functions (input variables number over 20), a systematic approach is very ineffective. Several efficient heuristic methods have been proposed that produce solutions of optimal or near optimal quality [6, 9]. These methods drastically reduce the number of partitions that have to be checked to find a solution of satisfactory quality. However, since the time needed to check one input variable partitioning strongly depends on the size of the decomposed function (Fig.1), the efficiency of these methods for functions described by truth tables with large number of rows is greatly reduced.

The size of the truth table describing a decomposed function influences not only computation complexity of all decomposition steps, but has also a large impact on the size of the system memory required by an application implementing functional decomposition. This factor plays an important role in the case of large truth tables.

The importance of this problem is increased by the fact that in a multilevel decomposition, the decomposition process is applied to functions H and G repetitively. The process is repeated until each block can be directly mapped in a logic block of a specific implementation structure [4].

3. Decomposition of cofactored function

This paper presents the method based on a "divide-and-conquer" paradigm that decreases the influence of number of rows on the searching time of efficient partitioning . The method is based on the application of Shannon's expansion.

Theorem 2. Shannon's expansion

An arbitrary logic function $F(x_1,...,x_n)$ can be expanded as follows:

$$F(x_1,..., x_i,..., x_n) = \overline{x_i}\, F(x_1,..., 0,..., x_n) + x_i F(x_1,..., 1,..., x_n)$$

In the case of functions described by truth tables, Shannon's expansion results in replacing a truth table by two truth tables without variable x_i One truth table consists of rows for which variable x_i has value 1 in the original table and the second consists of rows for which variable x_i has value 0 in the original table.

The Shannon's expansion theorem is useful in the analysis and synthesis of digital systems. An n-variable function F is replaced by two $(n - 1)$-variable functions: $F(x_1,..., 0,..., x_n)$ and $F(x_1,..., 1,..., x_n)$. These functions can be recursively expanded in similar way resulting in a sum of minterms.

In [10], the concept of decomposing cofactored function is introduced. This concept is described by the theorem presented below.

Theorem 3. Decomposition of cofactored function

Let F_1 and F_2 be cofactors of function $F(x_1,...,x_n)$ with respect to variable x_i. If there exists such a function G, so that F_1 and F_2 have decomposition $F_1 = H_1(U, G(V))$ and $F_2 = H_2(U, G(V))$ respectively, then function F has also decomposition $F = H(U \cup x_i, G(V))$.

Application of this theorem reduces the process of decomposition of a function into decomposition of the function's cofactors. Instead of decomposing the original function, the function's cofactors are decomposed. Function $G(V)$ found in the decomposition of cofactors can be used in decomposing the original function, while functions $H_1(U, G(V))$ and $H_2(U, G(V))$ can be used to reconstruct function H in the decomposition of the original function. Since the cofactors' truth tables have less rows than the truth table of the decomposed function (under the best situation,, each cofactor will have half the number of rows of the original function) the decomposition process for cofactors can be performed much more efficiently as regards to computation time, as well as memory usage.

The advantage of this concept is that the original function can be expanded recursively with Shannon's expansion until cofactors have satisfactory size.

4. Input variable partitioning algorithm

In the approach proposed in [10], the decomposed function is split into sets of its cofactors. Application of Theorem 3 is possible only if such a input variable partitioning is computed that satisfies the decomposition condition from Theorem 1 for all cofactors. If the proposed variable partitioning does not satisfy the decomposition conditions for any one of the cofactors, Theorem 3 can not be applied. Thus, to find input variable partitioning that guarantees the decomposition of a function, it is sufficient to generate a set of input variable partitioning that satisfies the decomposition conditions for one cofactor and then remove from this set any variable partitioning that does not satisfy decomposition condition for any other cofactor.

In Table 1, the input variable partitioning algorithm that makes use of Theorem 3 is presented. First, the decomposed function is expanded into cofactors. Since, according to Theorem 3, variables used in expansion are fixed into free set U, significantly reducing the search space, their selection is very important. The selection of variables for expansion can be aided with r-admissibility concept [8]. After expanding the decomposed function into cofactors, the best input variable partitions are computed for one of the cofactors. For this purpose, the heuristic methods of input variable partitioning can be used [6, 9]. In case of small size of cofactors, the systematic method can be used (checking all the possible solutions and taking the best ones). Having done this, computed input variable partitions are verified to check if they satisfy the decomposition condition for the rest of the cofactors. If not, variable partition is removed from the set of possible solutions. After this the set of solutions is found that might give satisfactory decomposition. The last step of the algorithm is to check each solution, using Theorem 3 to find input variable partitioning that allows the construction of a satisfactory decomposition.

Tab. 1. Input variable partitioning algorithm
Tab. 1. Algorytm doboru zmiennych wejsciowych

```
Input:      truth table of function $F$
Output:   input variable partitioning ($U$, $V$) satisfying Theorem 1
(1) expand function $F$ into cofactors {$F_1$, …, $F_n$}
(2) for any cofactor $F_i$ find set S of the pairs ($U$, $V$) satisfying Theorem 1
(3) for each pair ($U$, $V$) from set S do
  (3.1) for each cofactor different than $F_i$ do
    (3.1.1) if pair ($U$, $V$) doesn't satisfy Theorem 1
      (3.1.1.1)   remove pair ($U$, $V$) from set S
      (3.1.1.2)   break
(4)   for each pair (U, V) from set S do
  (4.1) if pair (U, V) satisfy Theorem 3 return pair (U, V)
```

A solution quality depends on the size of S set. The more (U,V) pairs are checked, the higher probability of finding the better solution. The good solution means that is satisfies Theorem 3 and produces a small number of outputs from a G function.

The presented algorithm can be also combined with BDD (Binary Decision Diagrams) methods. In general, the calculations would be performed on smaller subtrees and consequently the time and memory usage can be reduced. More detailed description exceeds the subject of this paper.

In the Table 2 some results are presented. Since it is mainly a theoretical paper only a few examples are presented. The authors plan to make more extensive and detailed tests.

The Table 1 describes a cofactor-based method. Here the cofactors size was set to not more thane 2048 rows.

Tab. 2. Input variable partition search - comparison [s]
Tab. 2. Metody doboru zmiennych wejściowych - porównanie[s]

Example	Systematic Search	Cofactor-based method
Bin2BCD	134	63
SQR_13	649	243
Const_97	637	257
Apex3_3	24578	12783
Apex3_7	174889	104910
Σ	200887	188256
[%]	100	58,9

5. Conclusions

Efficiency of currently used decomposition methods is heavily dependent on the size of decomposed functions. Decomposition of combinational circuits described by truth tables with great number of rows is computationally expensive and requires large amount of system memory.

The most time consuming part of the decomposition process is the selection of input variable partitioning that guarantees good quality decomposition. The method described in this paper applies the "divide-and-conquer" paradigm in the form of Shannon's expansion to split the decomposed function into a set of its cofactors and computes decomposition for each cofactor separately. It is designed to help handling big functions. The method should be treated as a general scheme which can combined with other algorithms producing input variables partitions.

It has to be stressed that the very important advantage of this concept is the possibility to expand the original function with Shannon's expansion recursively until cofactors have satisfactory size. That allows the designer to adjust the algorithm to available system resources.

6. References

[1] M. Burns, M. Perkowski, L. Jóźwiak, "An Efficient Approach to Decomposition of Multi-Output Boolean Functions with Large Set of Bound Variables", Proc. Of EUROMICRO'98 Conference, Vasteras, Sweden, 1998

[2] J. A. Brzozowski, and T. Łuba, "Decomposition of Boolean Functions Specified by Cubes", Journal of Multiple-Valued Logic and Soft Computing, Vol. 9, Old City Publishing, Inc., Philadelphia, 2003, pp. 377–417.

[3] T. Łuba, H. Selvaraj, "A General Approach to Boolean Function Decomposition and its Applications in FPGA-based Synthesis", VLSI Design, Special Issue on Decompositions in VLSI Design, vol. 3, Nos. 3-4, 1995, pp. 289-300.

[4] T. Luba, H. Selvaraj, M. Nowicka, A. Kraśniewski, "Balanced multilevel decomposition and its applications in FPGA-based synthesis", In: Logic and Architecture Synthesis (G.Saucier, A.Mignotte ed.), Chapman&Hall, 1995.

[5] M. Nowicka, T. Łuba, and. M. Rawski, "FPGA-Based Decomposition of Boolean Functions. Algorithms and Implementation", Proc. of Sixth International Conference on Advanced Computer Systems, Szczecin, 1999, pp. 502–509.

[6] M. Rawski, L. Jóźwiak, and T. Łuba, „Functional Decomposition with an Efficient Input Support Selection for Sub-functions Based on Information Relationship Measures", Journal of Systems Architecture 47, 2001, 2001, pp. 137-155.

[7] M. Rawski, L. Jóźwiak, M. Nowicka, T. Luba, " Non-Disjoint Decomposition of Boolean Functions and Its Application in FPGA-oriented Technology Mapping, Proc. of the EUROMICRO'97 Conference, Budapest, Hungary, Sept. 1-4, 1997, pp.24-30, IEEE Computer Society Press.

[8] M. Rawski, H. Selvaraj, T. Łuba, "An application of functional decomposition in ROM-based FSM implementation in FPGA devices", Journal of Systems Architecture, Vol.51(2005), ELSEVIER, 2005, pp.424-434..

[9] M. Rawski, H. Selvaraj, P. Morawiecki, "Efficient Method of Input Variable Partitioning in Functional Decomposition Based on Evolutionary Algorithms", Proc. of EUROMICRO Symposium on Digital System Design'04, Rennes, France, 2004, pp. 136 - 143.

[10] M. Rawski, P. Morawiecki, H. Selvaraj, "Decomposition of Combinational Circuits Described by Large Truth Tables", Proceedings of Eighteenth International Conference on Systems Engineering, Coventry, United Kingdom, September 5-7 2006, pp. 401 - 406.

Artykuł recenzowany