Mariusz RAWSKI, Mirosława NOWICKA,
Paweł TOMASZEWICZ, Tadeusz ŁUBA
Faculty of Electronics and Information Technology,
Institute of Telecommunication,
Warsaw University of Technology
Warsaw, Nowowiejska 15/19, Poland

Decomposition - based logic synthesis and its application in FPGA - oriented technology mapping

Summary: A generalized decomposition algorithm is formulated to map a Boolean function into a network of universal cells capable of implementing any function with a fixed number of inputs and outputs. The main characteristic of the method are original representation of Boolean functions as well as application of graph coloring heuristics to the serial decomposition problem. The method is applied to several standard benchmarks and the results presented. When the algorithm is targeted at a technology-specific cell structure, the cell count is reduced considerably.

1. Introduction

A renewed interest in functional decomposition in recent years is caused by the introduction of Look-up Table Field Programmable Gate Arrays (FPGAs) by Xilinx in 1986, and other companies in succeeding years (AT&T, Actel, Motorola, Altera). When it was found that the adaptation of earlier algebraic methods does not work properly for Lookup Table model, researchers switched to Boolean decomposition. Currently, in most of the systems, the decomposition is only an auxiliary process (MIS-PGA [16], FGMap[8]).

However, with the arrival of systems such as TRADE [25], TOS [21] and DEMAIN [12], that are primarily based on the decomposition, the situation is becoming to change drastically. Even the U.C. Berkeley group of Professor Brayton, who invented the algebraic factorization approach and were for long time its most stubborn adherents, are recently devoting more attention to Boolean decomposition.

Taking into consideration functional dependencies, two distinct decomposition strategies can be identified for decomposing Boolean functions [10].

The first one, called parallel decomposition, relies on partitioning the outputs into two separate groups, so that each group of outputs depends on fewer or the same number of variables as a given function F i.e. $Y = Y_{G1} \cup Y_{G2}$, where for X_{G1} and X_{G2}, the respective support sets of Y_{G1} and Y_{G2}, we have $X_{G1} \subseteq X$ and $X_{G2} \subseteq X$.

The second decomposition strategy, called serial decomposition, relies on partitioning the input variables so that to obtain a two-level functional dependency $F = H(A,G(B \cup C))$, where $A \cup B = X$ and $C \subset A$.

The general functional decomposition is an iterative process during which the original function is broken down into smaller and smaller components. At each step , either parallel or serial decomposition is performed.

In this paper, we focus on the more difficult problem of serial decomposition. Particularly we consider the application of graph coloring heuristics to the serial decomposition problem. The different coloring algorithms as well as exact or approximation procedures have been applied and tested for the efficient decomposition of truth tables. Our experiments showed that sequential strategy gives reasonable results for decomposition of large logic circuits. According to the computer experiments, strategy based on the graph coloring seems to be a source of computer run-time and memory space benefits, what makes it interesting from theoretical and practical point of view.

2. Basic notions

2.1. Partition-based representation of Boolean functions

A Boolean function F with n input and m output variables is defined as a mapping

$$F: \{0,1\}^n \rightarrow \{0,1,–\}^m.$$

The value "–" (don't care) at one of the outputs means that a 0 or a 1 will be accepted as the response of the circuit realizing this part of the function.

Every element of the domain $\{0,1\}^n$ is called a minterm. The list of minterms with the corresponding values of the function is called the truth table of F.

Let X be the set of input variables and Y the set of output variables of F. Let V_z be the domain (set of possible values) of variable $z \in X \cup Y$. Clearly, $V_z = \{0,1\}$ if $z \in X$ and $V_z = \{0,1,–\}$ if $z \in Y$. Let M be the set of minterms of F. For each $z \in X \cup Y$, we define the function a_z: $M \rightarrow V_z$ which assigns a value of variable z to each minterm in M. Function a_z represents a single column in the truth table of F, either in the input part (if $z \in X$) or in the output part (if $z \in Y$).

In general, any pair of minterms may have identical values for some number of input variables. A convenient way to reflect the similarity of two minterms is to use the so called indiscernibility relation [9,17]. This relation, denoted by IND, is defined as follows.

Let $B \subseteq X$ and $m_1, m_2 \in M$.

Then $(m_1, m_2) \in \text{IND}(B)$ iff $a_x(m_1) = a_x(m_2)$ for all $x \in B$.

In other words, $(m_1, m_2) \in \text{IND}(B)$ if the values of the arguments (input variables) from B are identical for both m_1 and m_2. Minterms m_1 and m_2 are said to be indiscernible by arguments of B. The indiscernibility relation is an equivalence relation on M and

$$\text{IND}(B) = \bigcap_{x \in B} \text{IND}(x).$$

The relation IND partitions the set of minterms M into equivalence classes M/IND(B). We denote partition M/IND(B) by P(B) and call such a partition an input partition generated by B. Thus

$$P(B) = \prod_{x \in B} P(x),$$

where Π denotes the product of partitions.

Likewise, the relation called output consistency and denoted by CON can be defined for any subset B of output variables.

Let $B \subseteq Y$ and $m_1, m_2 \in M$. Then $(m_1, m_2) \in \text{CON}(B)$ iff $a_y(m_1) \sim a_y(m_2)$ for every $y \in B$, where $a_1 \sim a_2$ if a_1 and a_2 are the same whenever both are specified (if $a_1 = 0$, then to have $a_1 \sim a_2$ we must have $a_2 = 0$ or $a_2 = –$).

Clearly, the output consistency relation is not an equivalence relation on M. Hence, it "partitions" M into non-disjoint subsets. Nevertheless, to describe the output consistency relation, we use the same notation as for the indiscernibility relation, i.e. $P_F(B)$ is an output "partition" generated by B. "Partitions" with non-disjoint blocks are referred to as rough partitions (r-partitions). The concept of an r-partition is a simple extension of that of an ordinary partition and typical operations on r-partitions are the same as used in the ordinary partition algebra [5].

2.2. The graph coloring problem

In this chapter we present a brief introduction to some problems of the graph theory which are especially applicable to decomposition algorithms [24,26].

Let graph G = (V,E) consists of a finite set V of vertices and a family E of edges (unordered pairs of distinct nodes). The k-coloring of graph vertices is a partition of the set V into k independent sets $S_1, .., S_k$. Here a set S of vertices is called independent if no two vertices S are linked by an edge. The smallest integer number k for which there exists a k-coloring of graph G is the chromatic number of G and it is denoted by $\chi(G)$. Unfortunately, the problem of finding minimal $\chi(G)$ for given graph G is very hard and complex. Generally, the answer on the question whether an arbitrary graph has a k-coloring for a given integer number k is NP-complete problem. According to this fact, the exact algorithms is useless for graphs with more than 30 vertices.

With respect to above mentioned facts, the simple sequential graph coloring algorithm was chosen as an efficient to which can help to solve decomposition problem [24,26]. Let us briefly remind, the basic idea of this algorithm. Suppose that $v_1,v_2,..,v_n$ be an ordering of vertices of a given graph G. In a sequential method for coloring graph G, vertex v_i is added to the subgraph induced by already colored vertices $v_1, v_2,..,v_{i-1}$ and a new coloring of vertices $v_1,v_2,..,v_{i-1},v_i$ is determined. This step is repeated for i = 1,2,..,n, where for i = 1 the subgraph is empty. At each step, an attempt is made to use relatively small number of colors. In a basic sequential algorithm, the vertex v_i is assigned a color with the smallest number. A pseudo-Pascal version of this algorithm is given below.

procedure SGC;
  Input   Graph $G = (V,E)$ and order $v_1,v_2,..,v_n$;
  ***Output* :**   Array color such that *color*[*i*] is a color assigned to vertex v_i ;
  begin
      for *i* := 1,2,..,*n* **do**
            ***color*[*i*]** := $\min\{k \in \mathbf{N}$: such that $color[\,j\,] \neq k$
                      for each v_j $(1 \leq j < i)$ adjacent to $v_i\}$
  end;

3. Graph coloring approach to the decomposition problem

The theorem presented below states the sufficient condition for the existence of a serial decomposition in partition algebra terms [13].

THEOREM (Decomposition Theorem):
Functions G and H represent a serial decomposition of function F = H (A,G(B)) if there exists a partition P(G) such that: $P(B) \leq P(G)$ and $P(A) \circ P(G) \leq P(Y)$, where all the partitions are over the set of minterms and the number of output variables of component G is equal to $\lceil \log_k CARD(P(G)) \rceil$; CARD(P(G)) denotes the number of blocks of partition P(G) ($\lceil x \rceil$ denotes the smallest integer equal to or larger than x).

According to above mentioned theorem, the decomposition task can be formulated as an optimization problem in the following way.

Decomposition Problem (the first version) :
Input: Given a multiple-value function F(*X*) and partition of input variables into two distinct sets: *A the free set* and *B the bond set*,
Output: The minimal decomposition of function F(*A,B*) in form H(*A*,G(*B*)) - such that partitioning *P*(*G*) is minimal in the sense of cardinality and also satisfies conditions $P(A) \circ P(G) \leq P(Y)$ and $P(B) \leq P(G)$.

Fortunately, the Decomposition Problem would be effectively (in the sense of computer working time efficiency and also computer memory space efficiency) reduced to the Graph Coloring Problem. The first of all, let us define the incompatible relation [14,25]. Let all assumptions in input to the Decomposition Problem are satisfied.

***Definition* (*Incompatible Relation*):**
$INC(F(A,B)) = \{(B_i, B_j) \in P(B) \times P(B)$: if partition P_G' obtained from partition P(B) by merging blocks B_i and B_j into a single block B_{ij} does not satisfy condition $P(A) \circ P_G' \leq P_F\}$.

The INC relation is not an equivalence relation on set P(B). The graph of incompatible relation for a given function F(A,B) will be simple called the incompatible graph and it will be denoted by symbol $G_{F(A,B)}$. Let us remark that graph $G_{F(A,B)}$ can be constructed in polynomial time with respect to the size of truth table associated with function F.

Example 1.

Consider now the problem of constructing the incompatible graph $G_{F(A,B)}$ for a certain function F described by the following partitions:

$$P(A)=\{\{1,3,8,10\},\{4,5,7\},\{2,6,9\}\},$$

$$P(B)=\{\{1,5\},\{8\},\{3,4,9\},\{7,10\},\{2\},\{6\}\},$$

$$P(Y)=\{\{1,3\},\{4,5,7,8,9\},\{2,6,10\}\}.$$

In this case: $\{\{1,5\},\{8\}\} \in INC(F(A,B))$, because partition $P(A) \circ P_1$ (where P_1 was obtained from P(B) by merging blocks {1,5} and {8} into a single block {1,5,8}) contains two minterms: 1 and 8 in the same block, whereas these minterms are located in different blocks in partition P(Y). Therefore $P(A) \circ P_1$ is not less than P(Y). On the other hand, $\{\{1,5\},\{3,4,9\}\} \notin INC(F(A,B))$ because $P(A) \circ P_2= \{\{1,3\}, \{4,5\}, \{2\},\{6\},\{7\},\{8\},\{9\},\{10\}\} \leq P(Y)$, where partition P_2 was obtained from P(B) by merging blocks {1,5} and {3,4,9}. After completion of analysis (similar to above) of relation INC for all pairs from partition P(B), the incompatible graph $G_{F(A,B)}$ (presented in Figure 2) can be constructed. For the sake of clarity, we assumed that: $v_1=\{1,5\}$, $v_2=\{8\}$, $v_3=\{3,4,9\}$, $v_4=\{7,10\}$, $v_5=\{2\}$ and $v_6=\{6\}$.

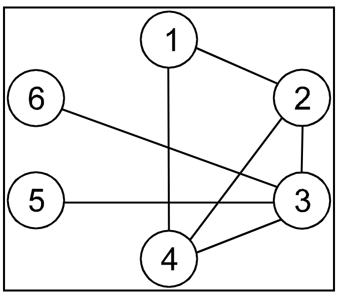

Figure 1. Incompatible Graph.

Here, the subfamily $C \subseteq P(B)$ is the class of compatible blocks for partition P(B) if it is a set of partitions which do not hold INC relation pairwise. Then, the key observation is that a subset of blocks $B' \subseteq B$ is a (maximal) class of compatible blocks for partitioning P(B) if and only if it is a (maximal) independent set in incompatible graph $G_{F(A,B)}$. Hence, taking into account this fact and all information's about graph coloring presented in chapter 2.2, the decomposition problem would be rewritten in the new form:

Decomposition Problem (the second version):

Input: Given incompatible graph $G_{F(A,B)}$ ***;***

Output: Find the minimal number of colors which are necessary to color the vertices of the graph $G_{F(A,B)}$ under restriction that adjacent vertices get different colors.

With respect to the last presented definition the following strategy for solving decomposition problem is proposed by authors.

ALGORITHM *FDH;*

Input : Given a function $F(X)$ and partitioning of F input variables set into sets A and B ($X = A \cup B$ and $\varnothing = A \cap B$);

Output : Decomposition (efficient) of function F into form $F(X) = H(A,G(B))$;

1. Construct the incompatibility graph $G_{F(A,B)}$;

2. SGC (Input: $G_{F(A,B)}$) {apply the sequential coloring procedure};

3. Reconstruct function G and H based on suboptimal coloring of graph $G_{F(A,B)}$;

The procedures 1, 2 and 3 can be effectively implemented in polynomial computer run-time with respect to the size of function F represented by input decision table. This conclusion came from simple analysis of procedures 1 and 3 time complexity and also from facts discussed in chapter 2.2. The decomposition heuristic can be applied several times to function F structure. At this moment it is very important to stress that variable partition procedure (which chooses bond and free sets) should depend on the area of decomposition heuristic application. But in general, cardinality of set B is bounded only by number of input variables of function F. Set B can be arbitrarily chosen, what would make set P(B) relatively big. It means indeed that the number of vertices of graph $G_{F(A,B)}$ would be large, unfortunately. For these reasons, the sequential graph coloring procedure (used in point 2 of decomposition strategy) seems to be quite good method of solving decomposition problem in these cases when cardinality of set P(B) is adequately large. Thus Decomposition Strategy gives reasonable good solutions in short computer run-time.

Example 2.

Consider the incompatible graph $G_{F(A,B)}$ of Figure 1. The result of sequential coloring procedure application (with order: v_1, v_2, v_3, v_4, v_5, v_6) is vertices labeling, shown in Figure 2.

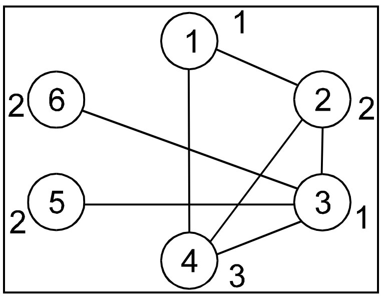

Figure 2. Graph Coloring for $G_{F(A,B)}$.

4. Experimental results

Based on the presented general decomposition algorithm, a logic synthesis system called DEMAIN has been developed for MS-DOS operating systems. DEMAIN can run in either interactive or automatic mode. The automatic mode minimizes the number of Logic Blocks. In the interactive mode, the user can steer the synthesis process, according to the requirements of the chosen type of FPGA. For example, this option can be used effectively for FPGAs with Logic Blocks capable of implementing functions with different number of variables and to optimize the number of Logic Blocks or to optimize the number of levels for FPGAs with restricted interconnections. It is also possible to balance between delay and area optimization as the situation may demand.

Table 1 shows that results of decomposition of a few benchmark circuits into five-input two-output cells, which is in fact the decomposition aimed at the Xilinx Logic Blocks. The comparison of results achieved by DEMAIN with the other published results (TRADE [25], TOS [21], MIS-PGA [16], Chortle [4]) shows that the proposed method does not suffer because of its universality, in fact it provides better solutions in many cases.

Name	DEMAIN	TRADE	TOS	MIS-PGA	Chortle
rd84	8	8	10	9	28
rd73	5	5	7	5	15
z4	4	4	4	4	4
Sao2	23	27	23	28	26
9sym	5	6	7	7	51
root	23	21	–	–	–
b9	32	29	–	32	30
alu2	17	22	47	96	94
Clip	18	29	22	23	25

Table 1.

Table 2 compares the minimum logic depth solutions generated by DEMAIN with those generated by other mapping algorithms designed for delay minimization [2,3,8,18]. It can be seen that, in spite of the universal character of DEMAIN, the results obtained are not worse than the results of the other programs aimed exclusively at delay optimization.

Name	DEMAIN	Chortle	DAG-Map	Flow-Map	MIS-PGA	ASYL
rd84	8/3	61/4	43/4	43/4	13/3	14/3
rd74	5/4	52/4	–	–	8/2	–
Sao2	23/6	58/4	–	–	45/5	30/8
9sym	5/4	63/5	61/5	61/5	7/3	8/3
alu2	17/3	227/9	169/8	162/8	122/6	60/6
Clip	18/5	83/4	–	–	54/4	33/6

Table 2.

It is worth to stress that in many experimental examples DEMAIN system achieved the same or similar results using sequential coloring as well as exact coloring. Additionally, it is obvious that the first algorithms combination is much faster than the second one.

Other experiments were performed using the MCNC benchmark circuits to compare the results produced by DEMAIN with those produced by ALTERA's MAXPlus2 software. For example, in experiment using rd84 benchmark circuit, rd84 was initially compiled directly using ALTERA's compiler. ALTERA's dedicated compiler reported that 147 Logic Cells are required to implement the benchmark. The second part of the experiment used DEMAIN to decompose rd84 into a netlist of cells, where the cells have four inputs and one output. The decomposed multilevel structure of rd84 was fed as a new project to MAXPlus2 Compiler resulting in 15 (TEN) LCs to implement the circuit. Generally, we have an average gain over MaxPlus2 of about 78 % on the MCNC benchmarks (some results are shown in Table 3).

The method was also applied in designing a Cryptographic Device, based on the algorithm specified by Data Encryption Standard (DES). The design was implemented on ALTERA's FLEX EPF8282LC84 device.

The structure decomposed using DEMAIN consisted of a total of 194 LCs (Logic Cells). Therefore, one piece of EPF8282LC84 was sufficient to realize all the DES algorithm. On the other hand, the structure proposed by ALTERA's design tool consisted of a total of 610 LCs, which means a requirement of three pieces of EPF8282LC84.

Name	DEMAIN	MAXPlus2
9sym	11	98
Clip	61	404
Rd73	10	98
root	57	113
Sao2	53	125
Z4	7	88

Table 3.

6. Conclusions

The approach to decomposition presented in this paper relies on: partition-based representation of Boolean functions [7,9,13,14], effective "balanced" decomposition strategy [12,15] and proper graph coloring for serial decomposition [20,25]. Separately the above methods were already used in decomposition algorithms, however no one applied them together in general mapper targeted at a technology specific FPGA-based cell structure. In consequence it is possible to apply the method to develop procedures that support synthesis of FPGA-based circuits with other evaluation criteria e.g. delay optimization.

References

[1] Abouzeid P., Babba B., Crastes de Paulet M., Saucier G.: Input-Driven Partitioning Methods and Application to Synthesis on Table-Lookup-Based FPGAs. *IEEE Trans on CAD*, 12, No. 7, pp. 913-925, 1993.

[2] Chen K.C., Cong J., Ding Y., Kahng A.B., and Trajmar P. DAG-Map: Graph-based FPGA technology mapping for delay optimization. *IEEE Design and Test of Computers*, pp. 7-20, Sept. 1992.

[3] Cong J., and Ding Y. FlowMap: An Optimal Technology Mapping Algorithm for Delay Optimization in Lookup-Table Based FPGA Designs. *IEEE Trans. on CAD*, 13, No. 1, pp. 1-12, 1994.

[4] R.Francis, J.Rose, Z.Vranesic,: Chortle-crf: Fast Technology Mapping Look-up Table-Based FPGAs. Proc. 28-th ACM/IEEE Design Automation Conf., pp. 227-233, 1991.

[5] Hartmanis J., Stearns R.E.: Algebraic Structure Theory of Sequential Machines. Prentice-Hall, 1966.

[6] Jóźwiak L., Volf F.: An Efficient Method for Decomposition of Multiple-Output Boolean Functions and Assigned Sequential Machines, *Proc. European Conference on Design Automation*, pp. 114-122, 1992.

[7] Jóźwiak L.: General Decomposition and its Use in Digital Circuit Synthesis. VLSI Design. Special Issue on Decompositions in VLSI Design, vol. 3, Nos. 3-4, pp. 225-248, 1995.

[8] Lai Y.T., Pan K.R., Pedram M., Vrudhula S.: FGMap: A Technology Mapping Algorithm for Look-up Table Type FPGA Synthesis. Proc. of 30-th DAC, pp. 642-647, 1993.

[9] Łuba T., Rybnik J.: Rough Sets and Some Aspects in Logic synthesis. In Intelligent Decision Support - Handbook of Application and Advances of the Rough Sets Theory, R.Słowiński (ed), Kluwer Academic Publishers, 1992.

[10] Łuba T.: Multi-level logic synthesis based on decomposition. *Microprocessors and Microsystems*, 18, No. 8, pp. 429-437, 1994.

[11] Łuba T., Lasocki R., Rybnik J.: An Implementation of Decomposition Algorithm and its Application in Information Systems Analysis and Logic Synthesis. In Rough Sets, Fuzzy Sets and Knowledge Discovery, W. Ziarko (Ed.). Workshops in Computing Series. Springer Verlag,
pp. 458-465, 1994.

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

[13] Łuba T., Selvaraj H.: 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, pp. 289-300, 1995.

[14] Łuba T.: Decomposition of Multiple-Valued Functions. 25th International Symposium on Multiple-Valued Logic. Bloomington, Indiana, pp. 256-261, 1995

[15] Łuba T., Nowicka M. and Rawski M.: Performance-oriented Synthesis for LUT-based FPGAs, Proc. Mixed Design of Integrated Circuits and Systems, pp. 96-101, Łódź, 1996.

[16] Murgai R., Shenoy N., Brayton R.K., Sangiovanni-Vincentelli A.: Improved logic Synthesis Algorithm for Table Look Up Architectures, *Proc. IEEE International Conf. on Computer Aided Design*, pp. 564-567, 1991.

[17] Pawlak Z.: Rough Sets. Theoretical Aspects of Reasoning about Data. Kluwer Academic Publishers, 1991.

[18] Shin H., Kim C.: Performance-Oriented Technology Mapping for LUT-Based FPGAs, IEEE Trans on VLSI Systems, vol. 3, No. 2, pp. 323-327, June 1995.

[19] Sasao T.: *FPGA Design by Generalized Functional Decomposition*, in *Logic Synthesis and Optimization*, Sasao T. (Ed.), Kluwer Academic Publishers, pp. 233-258, 1993

[20] Sapiecha P., Perkowski M., Łuba T.: Decomposition of the Information Systems Based on Graph Coloring Heuristics, Proc. Symposium on Modelling, Analysis and Simulation, CESA'96, vol. 2, pp. 1101-1106, Lille, 1996.

[21] Schlichtmann U.: Boolean Matching and Disjoint Decomposition for FPGA Technology Mapping. IFIP Workshop on Logic and Architecture Synthesis, pp. 83-102, Grenoble, 1993.

[22] Selvaraj H., Czerczak A., Kraśniewski A., Łuba T.: A Generalized Decomposition of Boolean Functions and its Application in FPGA-Based Synthesis. IFIP Workshop on Logic and Architecture Synthesis, pp. 147-166, Grenoble, 1993.

[23] Sicard P., Crastes M., Sakouti K., Saucier G.: Automatic Synthesis of Boolean Functions on Xilinx and Actel Programmable Devices. Proc. Euro ASIC'91, pp. 142-145, 1991.

[24] Sysło M., Deo N., Kowalik J.: Discrete Opitimization Algorithms. Prentice-Hall. Inc. Englewood Cliffs, NJ 07632, 1983.

[25] Wan W., Perkowski M.A.: A New Approach to the Decomposition of Incompletely Specified Multi-Output Function Based on Graph Coloring and Local Transformations and Its Application to FPGA Mapping. Proc. European Design Automation Conf., pp. 230-235, 1992.

[26] De Werra D.: Heuristic for Graph Coloring. Computing, Springer-Verlag Computing Suppl. 7, pp. 191-208, 1990.